Grafs

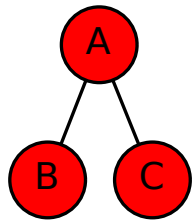
ETSEIB/GIE

28 de novembre de 2020

Creació i consulta d'un graf


```
>>> import networkx as nx
>>> g = nx.Graph()
>>> g
<networkx.classes.graph.Graph object at ...>
>>> g.add_nodes_from(('a','b','c'))
>>> g.add_edges_from((('a','b'),('a','c'),('b','a')
),('c','a')))
```

En aquest graf, al tenir tots els nodes connectats, no cal cridar `add_nodes_from`. Entrant només les arestes donem d'alta tots els nodes del graf.

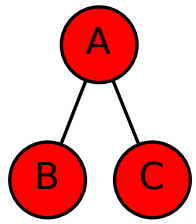


graf g

Creació i consulta d'un graf


```
>>> import networkx as nx
>>> g = nx.Graph()
>>> g
<networkx.classes.graph.Graph object at ...>
>>> g.add_nodes_from(('a','b','c'))
>>> g.add_edges_from((('a','b'),('a','c'),('b','a')
),('c','a')))
```

En aquest graf, al tenir tots els nodes connectats, no cal cridar `add_nodes_from`. Entrant només les arestes donem d'alta tots els nodes del graf.

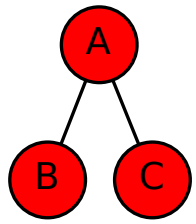


graf g

Creació i consulta d'un graf

```
>>> import networkx as nx
>>> g = nx.Graph()
>>> g
<networkx.classes.graph.Graph object at ...>
>>> g.add_nodes_from(('a','b','c'))
>>> g.add_edges_from((('a','b'),('a','c'),('b','a'
),('c','a')))
```

En aquest graf, al tenir tots els nodes connectats, no cal cridar `add_nodes_from`. Entrant només les arestes donem d'alta tots els nodes del graf.

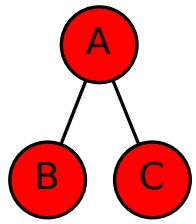


graf g

Creació i consulta d'un graf

```
>>> import networkx as nx
>>> g = nx.Graph()
>>> g
<networkx.classes.graph.Graph object at ...>
>>> g.add_nodes_from(('a','b','c'))
>>> g.add_edges_from((('a','b'),('a','c'),('b','a')
),('c','a')))
```

En aquest graf, al tenir tots els nodes connectats, no cal cridar `add_nodes_from`. Entrant només les arestes donem d'alta tots els nodes del graf.

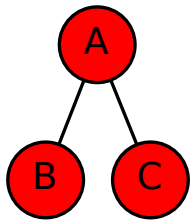


graf g

Creació i consulta d'un graf

```
>>> import networkx as nx
>>> g = nx.Graph()
>>> g
<networkx.classes.graph.Graph object at ...>
>>> g.add_nodes_from(('a','b','c'))
>>> g.add_edges_from((('a','b'),('a','c'),('b','a'),('c','a')))
```

En aquest graf, al tenir tots els nodes connectats, no cal cridar `add_nodes_from`. Entrant només les arestes donem d'alta tots els nodes del graf.

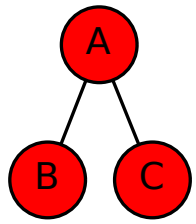


graf g

Creació i consulta d'un graf

```
>>> import networkx as nx
>>> g = nx.Graph()
>>> g
<networkx.classes.graph.Graph object at ...>
>>> g.add_nodes_from(('a','b','c'))
>>> g.add_edges_from((('a','b'),('a','c'),('b','a'),('c','a')))
```

En aquest graf, al tenir tots els nodes connectats, no cal cridar `add_nodes_from`. Entrant només les arestes donem d'alta tots els nodes del graf.

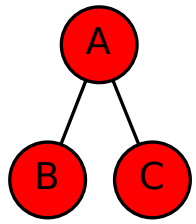


graf g

Creació i consulta d'un graf

```
>>> import networkx as nx
>>> g = nx.Graph()
>>> g
<networkx.classes.graph.Graph object at ...>
>>> g.add_nodes_from(('a','b','c'))
>>> g.add_edges_from((('a','b'),('a','c'),('b','a'
),('c','a')))
```

En aquest graf, al tenir tots els nodes connectats, no cal cridar `add_nodes_from`. Entrant només les arestes donem d'alta tots els nodes del graf.

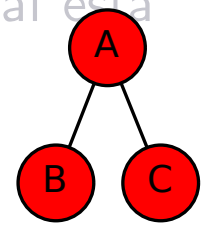


graf g

Consulta d'un graf


```
>>> g.add_edges_from((('a','b'),('a','c'),('b','a')
),('c','a')))
>>> g.add_node('b')
>>> g.add_node('b')
>>> sorted(g.nodes())
['a', 'b', 'c']
>>> g.edges()
EdgeView([('a', 'b'), ('a', 'c')])
```

Noteu que hem intentat posar dos cops quatre parelles de nodes i finalment tenim dues arestes. El mateix passa si insistim en posar un mateix node. L'ordre de les parelles depèn de l'execució. El graf està basat en diccionaris.

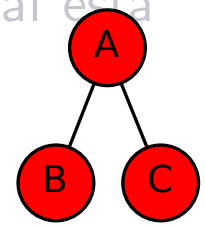


graf g

Consulta d'un graf


```
>>> g.add_edges_from((('a','b'),('a','c'),('b','a')
),('c','a')))
>>> g.add_node('b')
>>> g.add_node('b')
>>> sorted(g.nodes())
['a', 'b', 'c']
>>> g.edges()
EdgeView([('a', 'b'), ('a', 'c')])
```

Noteu que hem intentat posar dos cops quatre parelles de nodes i finalment tenim dues arestes. El mateix passa si insistim en posar un mateix node. L'ordre de les parelles depèn de l'execució. El graf està basat en diccionaris.

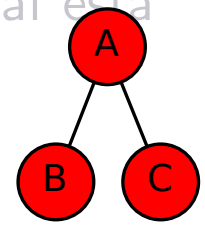


graf g

Consulta d'un graf

```
>>> g.add_edges_from((('a','b'),('a','c'),('b','a'
),('c','a')))
>>> g.add_node('b')
>>> g.add_node('b')
>>> sorted(g.nodes())
['a', 'b', 'c']
>>> g.edges()
EdgeView([('a', 'b'), ('a', 'c')])
```

Noteu que hem intentat posar dos cops quatre parelles de nodes i finalment tenim dues arestes. El mateix passa si insistim en posar un mateix node. L'ordre de les parelles depèn de l'execució. El graf està basat en diccionaris.

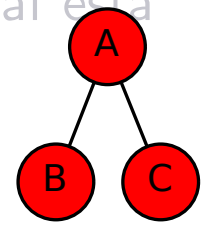


graf g

Consulta d'un graf

```
>>> g.add_edges_from((('a','b'),('a','c'),('b','a'
),('c','a')))
>>> g.add_node('b')
>>> g.add_node('b')
>>> sorted(g.nodes())
['a', 'b', 'c']
>>> g.edges()
EdgeView([('a', 'b'), ('a', 'c')])
```

Noteu que hem intentat posar dos cops quatre parelles de nodes i finalment tenim dues arestes. El mateix passa si insistim en posar un mateix node. L'ordre de les parelles depèn de l'execució. El graf està basat en diccionaris.

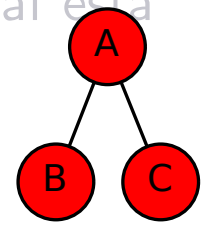


graf g

Consulta d'un graf

```
>>> g.add_edges_from((('a','b'),('a','c'),('b','a'
),('c','a')))
>>> g.add_node('b')
>>> g.add_node('b')
>>> sorted(g.nodes())
['a', 'b', 'c']
>>> g.edges()
EdgeView([('a', 'b'), ('a', 'c')])
```

Noteu que hem intentat posar dos cops quatre parelles de nodes i finalment tenim dues arestes. El mateix passa si insistim en posar un mateix node. L'ordre de les parelles depèn de l'execució. El graf està basat en diccionaris.

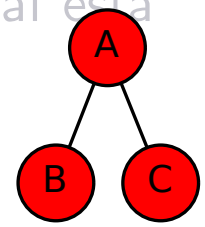


graf g

Consulta d'un graf

```
>>> g.add_edges_from((('a','b'),('a','c'),('b','a'
),('c','a')))
>>> g.add_node('b')
>>> g.add_node('b')
>>> sorted(g.nodes())
['a', 'b', 'c']
>>> g.edges()
EdgeView([('a', 'b'), ('a', 'c')])
```

Noteu que hem intentat posar dos cops quatre parelles de nodes i finalment tenim dues arestes. El mateix passa si insistim en posar un mateix node. L'ordre de les parelles depèn de l'execució. El graf està basat en diccionaris.

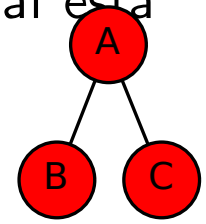


graf g

Consulta d'un graf

```
>>> g.add_edges_from((('a','b'),('a','c'),('b','a'
),('c','a')))
>>> g.add_node('b')
>>> g.add_node('b')
>>> sorted(g.nodes())
['a', 'b', 'c']
>>> g.edges()
EdgeView([('a', 'b'), ('a', 'c')])
```

Noteu que hem intentat posar dos cops quatre parelles de nodes i finalment tenim dues arestes. El mateix passa si insistim en posar un mateix node. L'ordre de les parelles depèn de l'execució. El graf està basat en diccionaris.



graf g

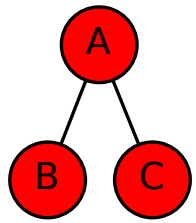
Més consultes d'un graf

Consulta del node a

```
>>> g['a']
AtlasView({'b': {}, 'c': {}})
```

Consulta de l'aresta ('a','b')

```
>>> g['a']['b']
{}
```



graf g

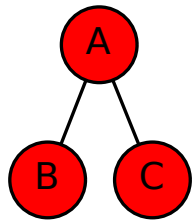
Més consultes d'un graf

Consulta del node a

```
>>> g['a']
AtlasView({'b': {}, 'c': {}})
```

Consulta de l'aresta ('a','b')

```
>>> g['a']['b']
{}
```



graf g

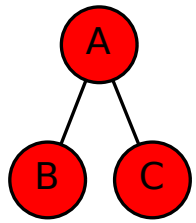
Més consultes d'un graf

Consulta del node a

```
>>> g['a']
AtlasView({'b': {}, 'c': {}})
```

Consulta de l'aresta ('a','b')

```
>>> g['a']['b']
{}
```



graf g

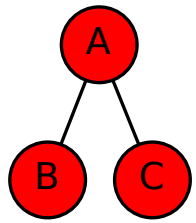
Més consultes d'un graf

Consulta del node a

```
>>> g['a']
AtlasView({'b': {}, 'c': {}})
```

Consulta de l'aresta ('a','b')

```
>>> g['a']['b']
{}
```



graf g

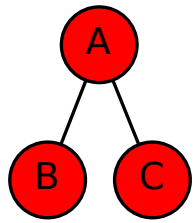
Més consultes d'un graf

Consulta del node a

```
>>> g['a']
AtlasView({'b': {}, 'c': {}})
```

Consulta de l'aresta ('a','b')

```
>>> g['a']['b']
{}
```

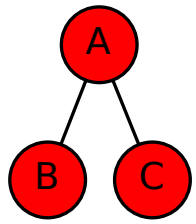


graf g

Recorregut de nodes del graf

`iter(g)` retorna un iterador de tots els nodes del graf

```
>>> iter(g)
<dict_keyiterator object at ...>
>>> for node in g:
...     print(node)
a
b
c
```

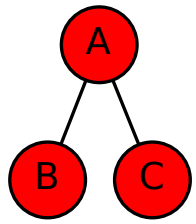


graf g

Recorregut de nodes del graf

`iter(g)` retorna un iterador de tots els nodes del graf

```
>>> iter(g)
<dict_keyiterator object at ...>
>>> for node in g:
...     print(node)
a
b
c
```

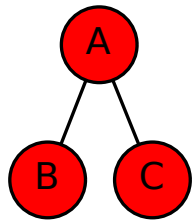


graf g

Recorregut de nodes del graf

`iter(g)` retorna un iterador de tots els nodes del graf

```
>>> iter(g)
<dict_keyiterator object at ...>
>>> for node in g:
...     print(node)
a
b
c
```

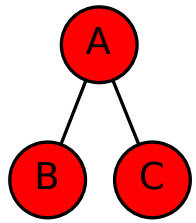


graf g

Recorregut de nodes del graf

`iter(g)` retorna un iterador de tots els nodes del graf

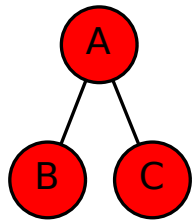
```
>>> iter(g)
<dict_keyiterator object at ...>
>>> for node in g:
...     print(node)
a
b
c
```



graf g

Veïns d'un node del graf

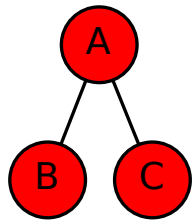
```
>>> for node in g['b']:
...     print(node)
a
>>> for node in g.neighbors('b'):
...     print(node)
a
```



graf g

Veïns d'un node del graf

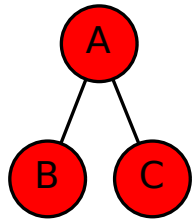
```
>>> for node in g['b']:
...     print(node)
a
>>> for node in g.neighbors('b'):
...     print(node)
a
```



graf g

Veïns d'un node del graf

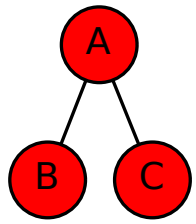

```
>>> for node in g['b']:
...     print(node)
a
>>> for node in g.neighbors('b'):
...     print(node)
a
```



graf g

Veïns d'un node del graf

```
>>> for node in g['b']:
...     print(node)
a
>>> for node in g.neighbors('b'):
...     print(node)
a
```



graf g

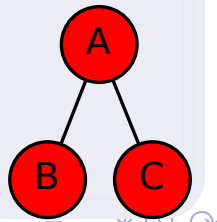
Etiquetes o atributs d'una aresta

afegim una etiqueta (atribut) *nom* a l'aresta ('a','b')

```
>>> g.add_edge('a','b', nom='ab')
```

El conjunt d'arestes queda igual

```
>>> g.edges()
EdgeView([('a', 'b'), ('a', 'c')])
>>> g['a']['b']
{'nom': 'ab'}
>>> g['a']['b']['nom']
'ab'
>>> g['b']['a']
{'nom': 'ab'}
>>> g['b']['a']['nom']
'ab'
```



graf g

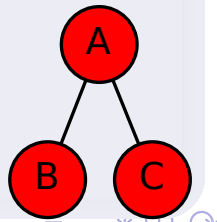
Etiquetes o atributs d'una aresta

afegim una etiqueta (atribut) *nom* a l'aresta ('a','b')

```
>>> g.add_edge('a','b', nom='ab')
```

El conjunt d'arestes queda igual

```
>>> g.edges()
EdgeView([('a', 'b'), ('a', 'c')])
>>> g['a']['b']
{'nom': 'ab'}
>>> g['a']['b']['nom']
'ab'
>>> g['b']['a']
{'nom': 'ab'}
>>> g['b']['a']['nom']
'ab'
```



graf g

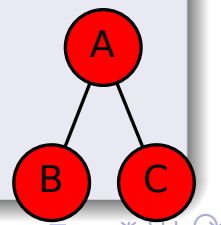
Etiquetes o atributs d'una aresta

afegim una etiqueta (atribut) *nom* a l'aresta ('a','b')

```
>>> g.add_edge('a','b', nom='ab')
```

El conjunt d'arestes queda igual

```
>>> g.edges()
EdgeView([('a', 'b'), ('a', 'c')])
>>> g['a']['b']
{'nom': 'ab'}
>>> g['a']['b']['nom']
'ab'
>>> g['b']['a']
{'nom': 'ab'}
>>> g['b']['a']['nom']
'ab'
```



graf g

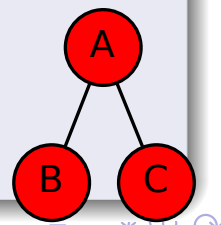
Etiquetes o atributs d'una aresta

afegim una etiqueta (atribut) *nom* a l'aresta ('a','b')

```
>>> g.add_edge('a','b', nom='ab')
```

El conjunt d'arestes queda igual

```
>>> g.edges()
EdgeView([('a', 'b'), ('a', 'c')])
>>> g['a']['b']
{'nom': 'ab'}
>>> g['a']['b']['nom']
'ab'
>>> g['b']['a']
{'nom': 'ab'}
>>> g['b']['a']['nom']
'ab'
```



graf g

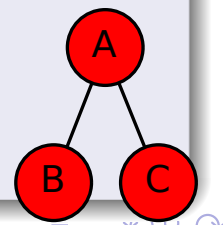
Etiquetes o atributs d'una aresta

afegim una etiqueta (atribut) *nom* a l'aresta ('a','b')

```
>>> g.add_edge('a','b', nom='ab')
```

El conjunt d'arestes queda igual

```
>>> g.edges()
EdgeView([('a', 'b'), ('a', 'c')])
>>> g['a']['b']
{'nom': 'ab'}
>>> g['a']['b']['nom']
'ab'
>>> g['b']['a']
{'nom': 'ab'}
>>> g['b']['a']['nom']
'ab'
```



graf g

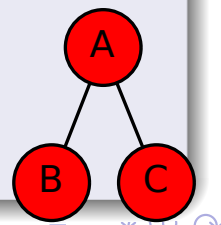
Etiquetes o atributs d'una aresta

afegim una etiqueta (atribut) *nom* a l'aresta ('a','b')

```
>>> g.add_edge('a','b', nom='ab')
```

El conjunt d'arestes queda igual

```
>>> g.edges()
EdgeView([('a', 'b'), ('a', 'c')])
>>> g['a']['b']
{'nom': 'ab'}
>>> g['a']['b']['nom']
'ab'
>>> g['b']['a']
{'nom': 'ab'}
>>> g['b']['a']['nom']
'ab'
```



graf g

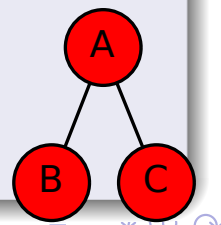
Etiquetes o atributs d'una aresta

afegim una etiqueta (atribut) *nom* a l'aresta ('a','b')

```
>>> g.add_edge('a','b', nom='ab')
```

El conjunt d'arestes queda igual

```
>>> g.edges()
EdgeView([('a', 'b'), ('a', 'c')])
>>> g['a']['b']
{'nom': 'ab'}
>>> g['a']['b']['nom']
'ab'
>>> g['b']['a']
{'nom': 'ab'}
>>> g['b']['a']['nom']
'ab'
```



graf g

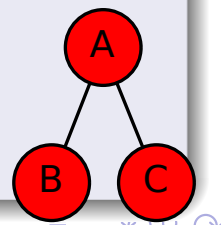
Etiquetes o atributs d'una aresta

afegim una etiqueta (atribut) *nom* a l'aresta ('a','b')

```
>>> g.add_edge('a','b', nom='ab')
```

El conjunt d'arestes queda igual

```
>>> g.edges()
EdgeView([('a', 'b'), ('a', 'c')])
>>> g['a']['b']
{'nom': 'ab'}
>>> g['a']['b']['nom']
'ab'
>>> g['b']['a']
{'nom': 'ab'}
>>> g['b']['a']['nom']
'ab'
```



graf g

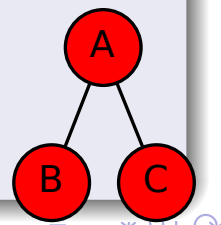
Etiquetes o atributs d'una aresta

afegim una etiqueta (atribut) *nom* a l'aresta ('a','b')

```
>>> g.add_edge('a','b', nom='ab')
```

El conjunt d'arestes queda igual

```
>>> g.edges()
EdgeView([('a', 'b'), ('a', 'c')])
>>> g['a']['b']
{'nom': 'ab'}
>>> g['a']['b']['nom']
'ab'
>>> g['b']['a']
{'nom': 'ab'}
>>> g['b']['a']['nom']
'ab'
```



graf g

Més consultes d'un graf

Nombre d'arestes del graf (mida d'un graf)

```
>>> g.size()
2
```

Nombre de nodes del graf (ordre d'un graf)

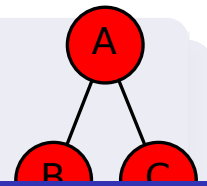
```
>>> len(g)
3
```

Grau d'un node

```
>>> g.degree('a')
2
```

Graus de tots els nodes del graf

```
>>> g.degree()
DegreeView({'a': 2, 'b': 1, 'c': 1})
```



Més consultes d'un graf

Nombre d'arestes del graf (mida d'un graf)

```
>>> g.size()
2
```

Nombre de nodes del graf (ordre d'un graf)

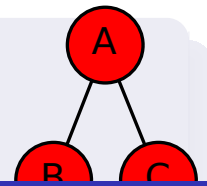
```
>>> len(g)
3
```

Grau d'un node

```
>>> g.degree('a')
2
```

Graus de tots els nodes del graf

```
>>> g.degree()
DegreeView({'a': 2, 'b': 1, 'c': 1})
```



Més consultes d'un graf

Nombre d'arestes del graf (mida d'un graf)

```
>>> g.size()
2
```

Nombre de nodes del graf (ordre d'un graf)

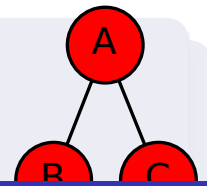
```
>>> len(g)
3
```

Grau d'un node

```
>>> g.degree('a')
2
```

Graus de tots els nodes del graf

```
>>> g.degree()
DegreeView({'a': 2, 'b': 1, 'c': 1})
```



Més consultes d'un graf

Nombre d'arestes del graf (mida d'un graf)

```
>>> g.size()
2
```

Nombre de nodes del graf (ordre d'un graf)

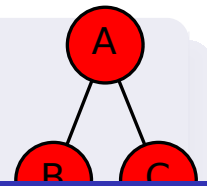
```
>>> len(g)
3
```

Grau d'un node

```
>>> g.degree('a')
2
```

Graus de tots els nodes del graf

```
>>> g.degree()
DegreeView({'a': 2, 'b': 1, 'c': 1})
```



Més consultes d'un graf

Nombre d'arestes del graf (mida d'un graf)

```
>>> g.size()
2
```

Nombre de nodes del graf (ordre d'un graf)

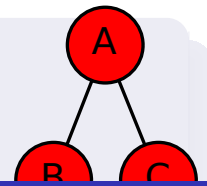
```
>>> len(g)
3
```

Grau d'un node

```
>>> g.degree('a')
2
```

Graus de tots els nodes del graf

```
>>> g.degree()
DegreeView({'a': 2, 'b': 1, 'c': 1})
```



Més consultes d'un graf

Nombre d'arestes del graf (mida d'un graf)

```
>>> g.size()
2
```

Nombre de nodes del graf (ordre d'un graf)

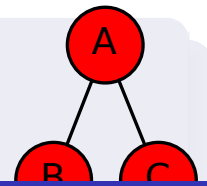
```
>>> len(g)
3
```

Grau d'un node

```
>>> g.degree('a')
2
```

Graus de tots els nodes del graf

```
>>> g.degree()
DegreeView({'a': 2, 'b': 1, 'c': 1})
```



Més consultes d'un graf

Nombre d'arestes del graf (mida d'un graf)

```
>>> g.size()
2
```

Nombre de nodes del graf (ordre d'un graf)

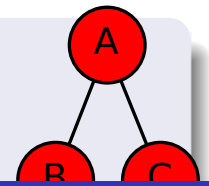
```
>>> len(g)
3
```

Grau d'un node

```
>>> g.degree('a')
2
```

Graus de tots els nodes del graf

```
>>> g.degree()
DegreeView({'a': 2, 'b': 1, 'c': 1})
```



Més consultes d'un graf

Nombre d'arestes del graf (mida d'un graf)

```
>>> g.size()
2
```

Nombre de nodes del graf (ordre d'un graf)

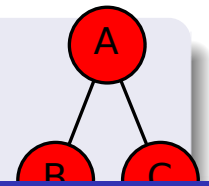
```
>>> len(g)
3
```

Grau d'un node

```
>>> g.degree('a')
2
```

Graus de tots els nodes del graf

```
>>> g.degree()
DegreeView({'a': 2, 'b': 1, 'c': 1})
```



Més consultes d'un graf

Nombre d'arestes del graf (mida d'un graf)

```
>>> g.size()
2
```

Nombre de nodes del graf (ordre d'un graf)

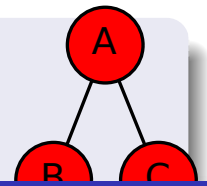
```
>>> len(g)
3
```

Grau d'un node

```
>>> g.degree('a')
2
```

Graus de tots els nodes del graf

```
>>> g.degree()
DegreeView({'a': 2, 'b': 1, 'c': 1})
```



Camins: Creació d'un graf exemple

```
>>> import networkx as nx
>>> g = nx.Graph()
>>> nx.add_path(g, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(g, ['a', 'e', 'c'])
```

Camins: Creació d'un graf exemple

```
>>> import networkx as nx
>>> g = nx.Graph()
>>> nx.add_path(g, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(g, ['a', 'e', 'c'])
```

Camins: Creació d'un graf exemple

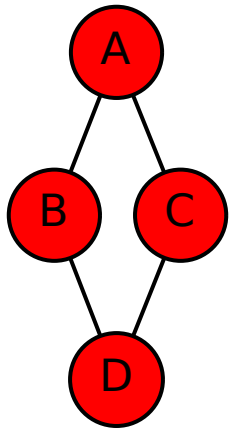

```
>>> import networkx as nx
>>> g = nx.Graph()
>>> nx.add_path(g, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(g, ['a', 'e', 'c'])
```

Camins: Creació d'un graf exemple


```
>>> import networkx as nx
>>> g = nx.Graph()
>>> nx.add_path(g, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(g, ['a', 'e', 'c'])
```

Camins: Creació d'un graf exemple

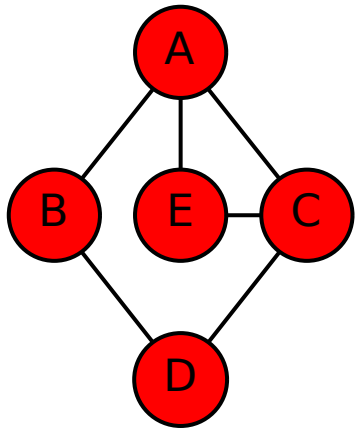

```
>>> import networkx as nx
>>> g = nx.Graph()
>>> nx.add_path(g, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(g, ['a', 'e', 'c'])
```



g

Camins: Creació d'un graf exemple

```
>>> import networkx as nx
>>> g = nx.Graph()
>>> nx.add_path(g, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(g, ['a', 'e', 'c'])
```



g

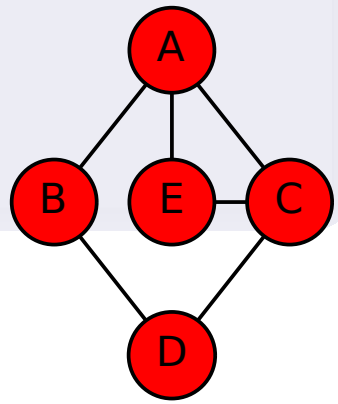
Camins: consultes

Consultem si hi ha camí entre 'a' i 'd':

```
>>> nx.has_path(g, 'a', 'd')
True
```

Consultem tots els camins simples entre 'a' i 'd'

```
>>> for c in sorted(nx.all_simple_paths(g, 'a', 'd
')):
...     print(c)
['a', 'b', 'd']
['a', 'c', 'd']
['a', 'e', 'c', 'd']
```



g

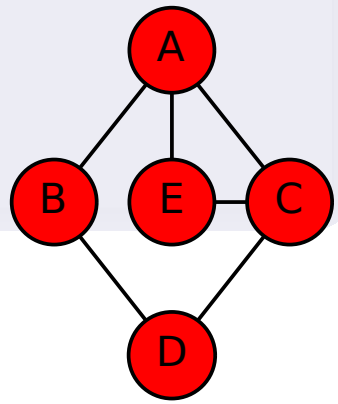
Camins: consultes

Consultem si hi ha camí entre 'a' i 'd':

```
>>> nx.has_path(g, 'a', 'd')
True
```

Consultem tots els camins simples entre 'a' i 'd'

```
>>> for c in sorted(nx.all_simple_paths(g, 'a', 'd
')):
...     print(c)
['a', 'b', 'd']
['a', 'c', 'd']
['a', 'e', 'c', 'd']
```



g

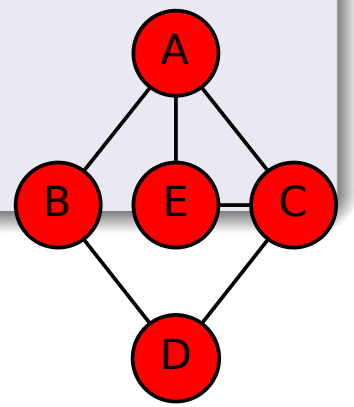
Camins: consultes

Consultem si hi ha camí entre 'a' i 'd':

```
>>> nx.has_path(g, 'a', 'd')
True
```

Consultem tots els camins simples entre 'a' i 'd'

```
>>> for c in sorted(nx.all_simple_paths(g, 'a', 'd
')):
...     print(c)
['a', 'b', 'd']
['a', 'c', 'd']
['a', 'e', 'c', 'd']
```



g

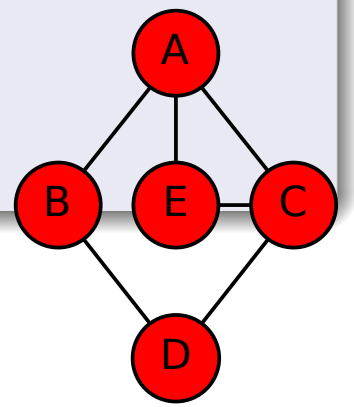
Camins: consultes

Consultem si hi ha camí entre 'a' i 'd':

```
>>> nx.has_path(g, 'a', 'd')
True
```

Consultem tots els camins simples entre 'a' i 'd'

```
>>> for c in sorted(nx.all_simple_paths(g, 'a', 'd
')):
...     print(c)
['a', 'b', 'd']
['a', 'c', 'd']
['a', 'e', 'c', 'd']
```



g

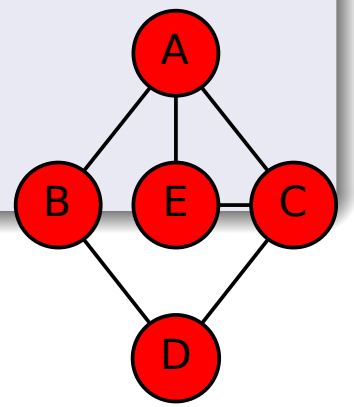
Camins: consultes

Consultem si hi ha camí entre 'a' i 'd':

```
>>> nx.has_path(g, 'a', 'd')
True
```

Consultem tots els camins simples entre 'a' i 'd'

```
>>> for c in sorted(nx.all_simple_paths(g, 'a', 'd
')):
...     print(c)
['a', 'b', 'd']
['a', 'c', 'd']
['a', 'e', 'c', 'd']
```



g

Camins mínims

Consultem la longitud del camí mínim

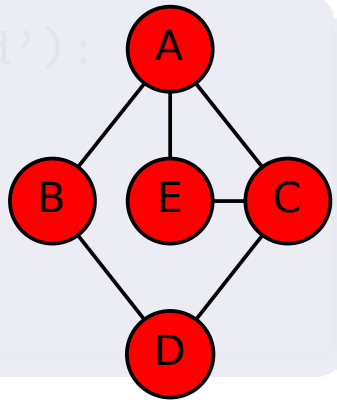
```
>>> nx.shortest_path_length(g, 'a', 'd')
2
```

Consultem un camí mínim

```
>>> nx.shortest_path(g, 'a', 'd')
['a', 'b', 'd']
```

Consultem tots els camins mínims

```
>>> for c in nx.all_shortest_paths(g, 'a', 'd'):
...     print(c)
...
['a', 'b', 'd']
['a', 'c', 'd']
```



g

Camins mínims

Consultem la longitud del camí mínim

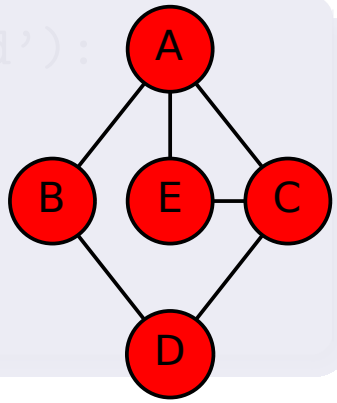
```
>>> nx.shortest_path_length(g, 'a', 'd')
2
```

Consultem un camí mínim

```
>>> nx.shortest_path(g, 'a', 'd')
['a', 'b', 'd']
```

Consultem tots els camins mínims

```
>>> for c in nx.all_shortest_paths(g, 'a', 'd'):
...     print(c)
...
['a', 'b', 'd']
['a', 'c', 'd']
```



g

Camins mínims

Consultem la longitud del camí mínim

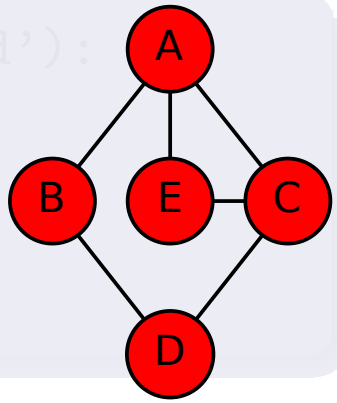
```
>>> nx.shortest_path_length(g, 'a', 'd')
2
```

Consultem un camí mínim

```
>>> nx.shortest_path(g, 'a', 'd')
['a', 'b', 'd']
```

Consultem tots els camins mínims

```
>>> for c in nx.all_shortest_paths(g, 'a', 'd'):
...     print(c)
...
['a', 'b', 'd']
['a', 'c', 'd']
```



g

Camins mínims

Consultem la longitud del camí mínim

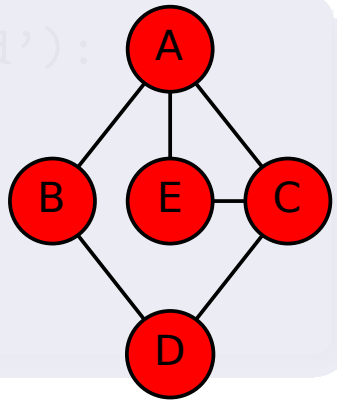
```
>>> nx.shortest_path_length(g, 'a', 'd')
2
```

Consultem un camí mínim

```
>>> nx.shortest_path(g, 'a', 'd')
['a', 'b', 'd']
```

Consultem tots els camins mínims

```
>>> for c in nx.all_shortest_paths(g, 'a', 'd'):
...     print(c)
...
['a', 'b', 'd']
['a', 'c', 'd']
```



g

Camins mínims

Consultem la longitud del camí mínim

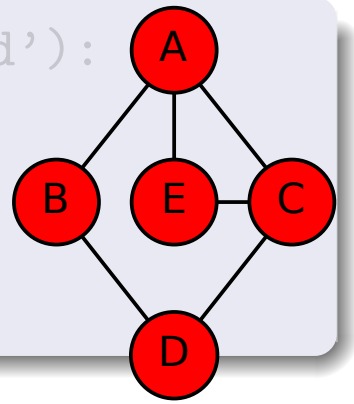
```
>>> nx.shortest_path_length(g, 'a', 'd')
2
```

Consultem un camí mínim

```
>>> nx.shortest_path(g, 'a', 'd')
['a', 'b', 'd']
```

Consultem tots els camins mínims

```
>>> for c in nx.all_shortest_paths(g, 'a', 'd'):
...     print(c)
...
['a', 'b', 'd']
['a', 'c', 'd']
```



g

Camins mínims

Consultem la longitud del camí mínim

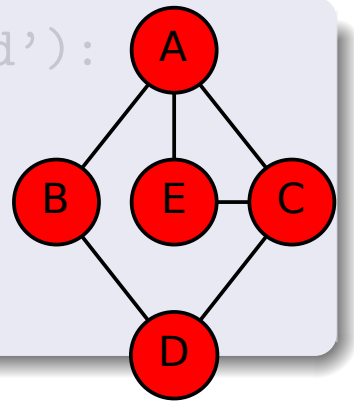
```
>>> nx.shortest_path_length(g, 'a', 'd')
2
```

Consultem un camí mínim

```
>>> nx.shortest_path(g, 'a', 'd')
['a', 'b', 'd']
```

Consultem tots els camins mínims

```
>>> for c in nx.all_shortest_paths(g, 'a', 'd'):
...     print(c)
...
['a', 'b', 'd']
['a', 'c', 'd']
```



g

Camins mínims

Consultem la longitud del camí mínim

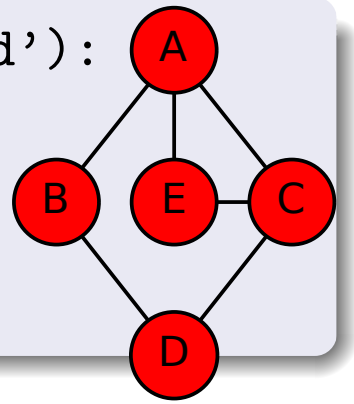
```
>>> nx.shortest_path_length(g, 'a', 'd')
2
```

Consultem un camí mínim

```
>>> nx.shortest_path(g, 'a', 'd')
['a', 'b', 'd']
```

Consultem tots els camins mínims

```
>>> for c in nx.all_shortest_paths(g, 'a', 'd'):
...     print(c)
...
['a', 'b', 'd']
['a', 'c', 'd']
```

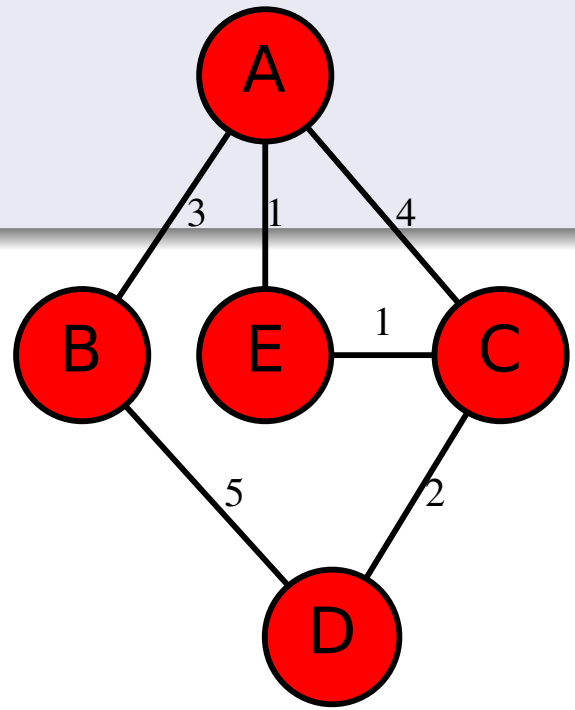


g

Camins mínims amb atributs a valorar

Afegim un atribut distància a les arestes

```
>>> g['a']['b']['dist'] = 3
>>> g['a']['c']['dist'] = 4
>>> g['a']['e']['dist'] = 1
>>> g['b']['d']['dist'] = 5
>>> g['c']['d']['dist'] = 2
>>> g['c']['e']['dist'] = 1
```

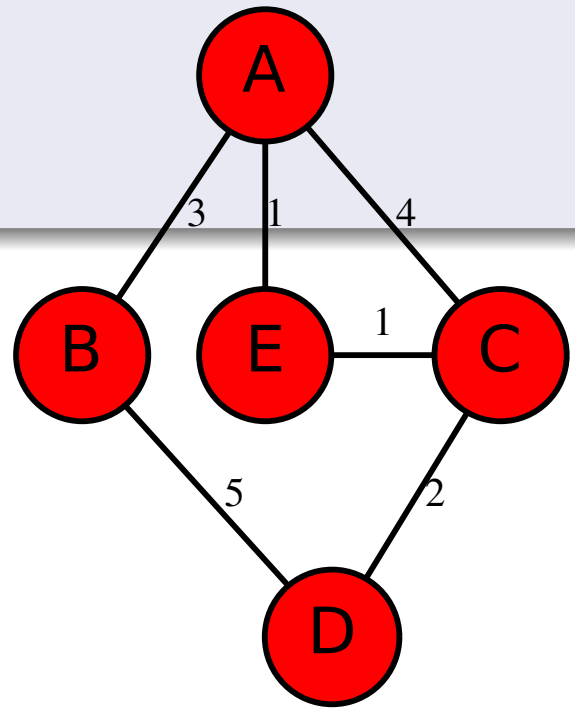


graf g

Camins mínims amb atributs a valorar

Afegim un atribut distància a les arestes

```
>>> g['a']['b']['dist'] = 3
>>> g['a']['c']['dist'] = 4
>>> g['a']['e']['dist'] = 1
>>> g['b']['d']['dist'] = 5
>>> g['c']['d']['dist'] = 2
>>> g['c']['e']['dist'] = 1
```

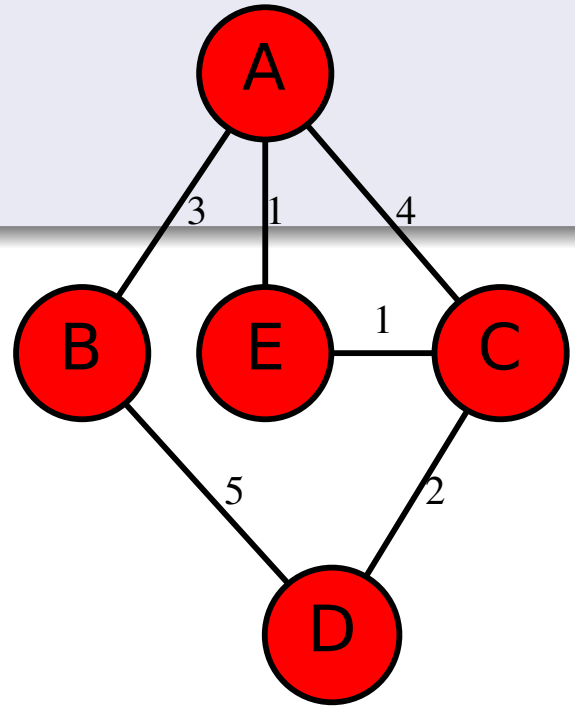


graf g

Camins mínims amb atributs a valorar

Afegim un atribut distància a les arestes

```
>>> g['a']['b']['dist'] = 3
>>> g['a']['c']['dist'] = 4
>>> g['a']['e']['dist'] = 1
>>> g['b']['d']['dist'] = 5
>>> g['c']['d']['dist'] = 2
>>> g['c']['e']['dist'] = 1
```

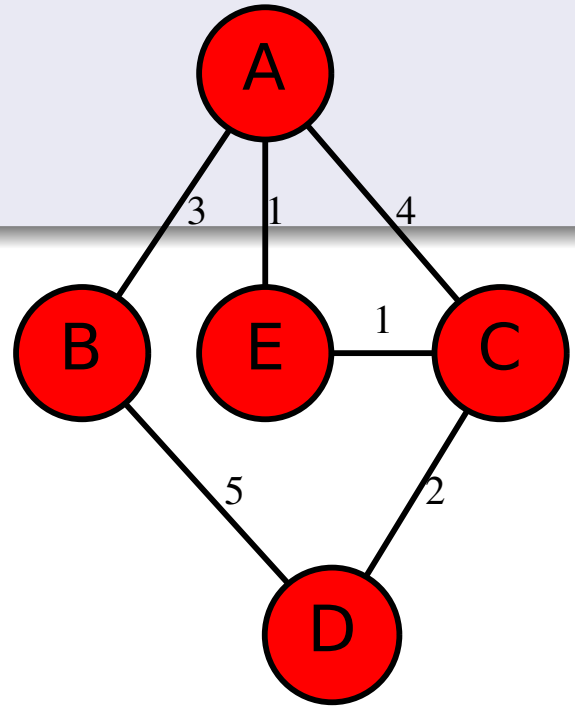


graf g

Camins mínims amb atributs a valorar

Afegim un atribut distància a les arestes

```
>>> g['a']['b']['dist'] = 3
>>> g['a']['c']['dist'] = 4
>>> g['a']['e']['dist'] = 1
>>> g['b']['d']['dist'] = 5
>>> g['c']['d']['dist'] = 2
>>> g['c']['e']['dist'] = 1
```

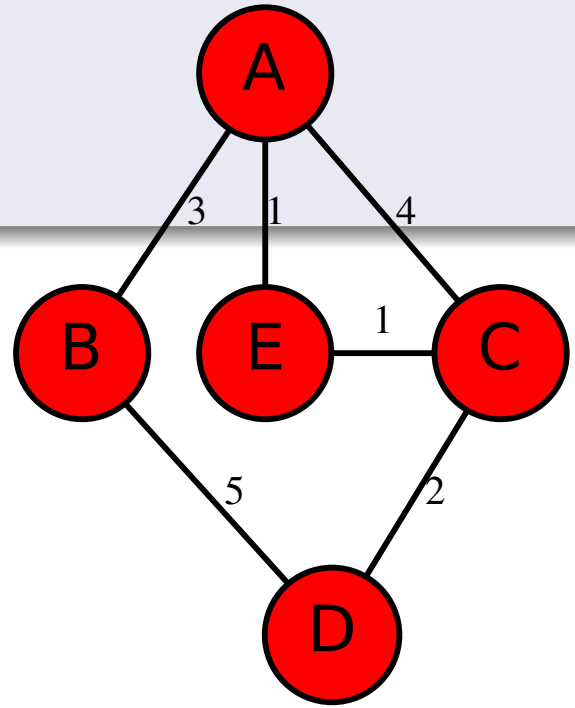


graf g

Camins mínims amb atributs a valorar

Afegim un atribut distància a les arestes

```
>>> g['a']['b']['dist'] = 3
>>> g['a']['c']['dist'] = 4
>>> g['a']['e']['dist'] = 1
>>> g['b']['d']['dist'] = 5
>>> g['c']['d']['dist'] = 2
>>> g['c']['e']['dist'] = 1
```

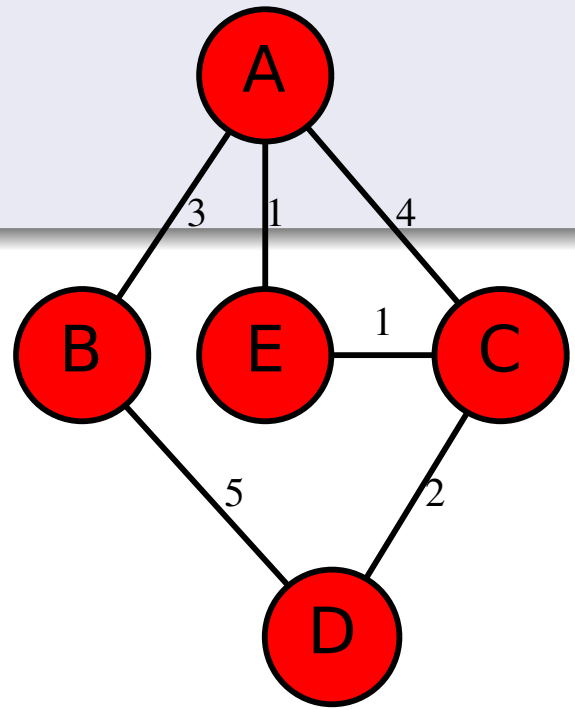


graf g

Camins mínims amb atributs a valorar

Afegim un atribut distància a les arestes

```
>>> g['a']['b']['dist'] = 3
>>> g['a']['c']['dist'] = 4
>>> g['a']['e']['dist'] = 1
>>> g['b']['d']['dist'] = 5
>>> g['c']['d']['dist'] = 2
>>> g['c']['e']['dist'] = 1
```

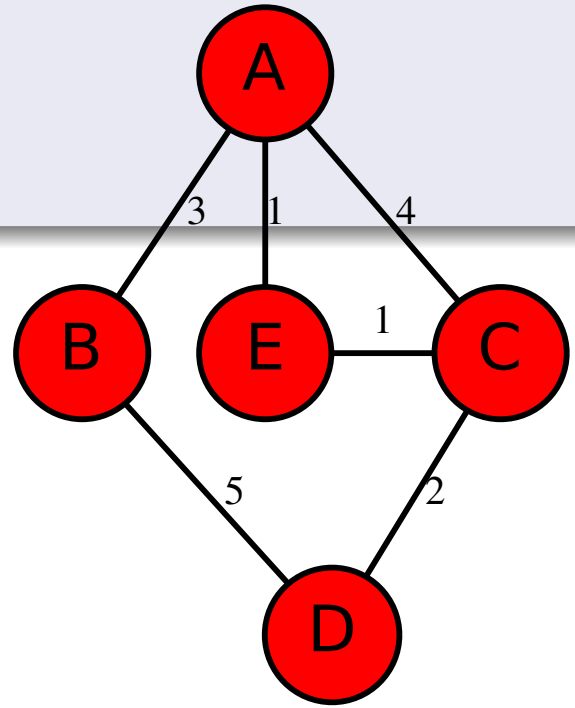


graf g

Camins mínims amb atributs a valorar

Afegim un atribut distància a les arestes

```
>>> g['a']['b']['dist'] = 3
>>> g['a']['c']['dist'] = 4
>>> g['a']['e']['dist'] = 1
>>> g['b']['d']['dist'] = 5
>>> g['c']['d']['dist'] = 2
>>> g['c']['e']['dist'] = 1
```

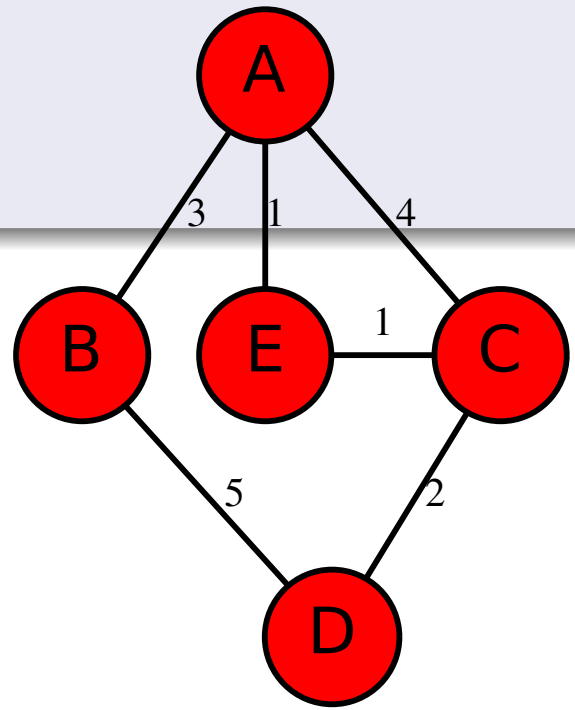


graf g

Camins mínims amb atributs a valorar

Afegim un atribut distància a les arestes

```
>>> g['a']['b']['dist'] = 3
>>> g['a']['c']['dist'] = 4
>>> g['a']['e']['dist'] = 1
>>> g['b']['d']['dist'] = 5
>>> g['c']['d']['dist'] = 2
>>> g['c']['e']['dist'] = 1
```



graf g

Camins mínims amb atributs a valorar

Consultem la longitud del camí mínim segons l'etiqueta 'dist'

```
>>> nx.shortest_path_length(g, 'a', 'd', weight='dist')
4
```

Consultem un camí mínim

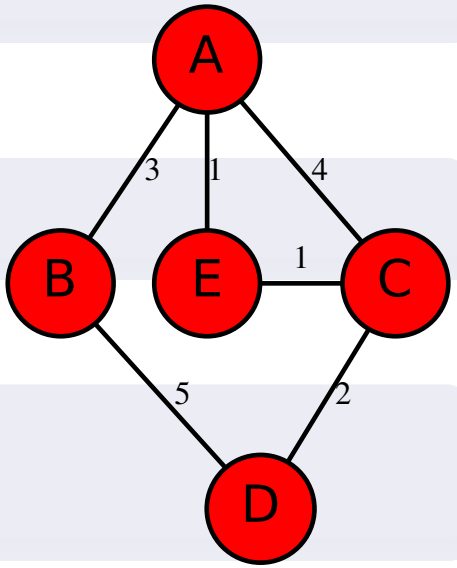
```
>>> nx.shortest_path(g, 'a', 'd', weight='dist')
['a', 'e', 'c', 'd']
```

Afegim un altre camí

```
>>> nx.add_path(g, ['f', 'g'])
```

Hi ha camí entre 'a' i 'f'?

```
>>> nx.has_path(g, 'a', 'f')
False
```



graf g

Camins mínims amb atributs a valorar

Consultem la longitud del camí mínim segons l'etiqueta 'dist'

```
>>> nx.shortest_path_length(g, 'a', 'd', weight='dist')
4
```

Consultem un camí mínim

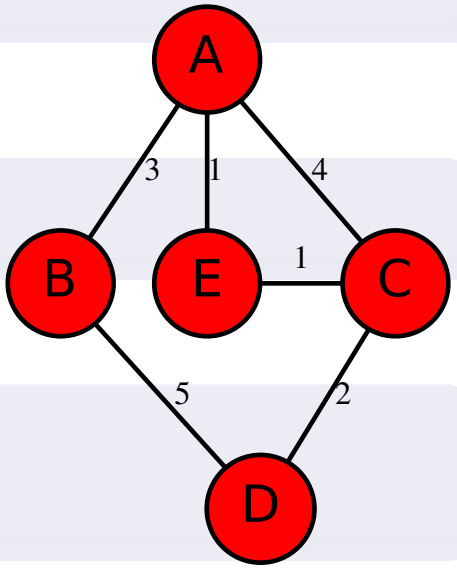
```
>>> nx.shortest_path(g, 'a', 'd', weight='dist')
['a', 'e', 'c', 'd']
```

Afegim un altre camí

```
>>> nx.add_path(g, ['f', 'g'])
```

Hi ha camí entre 'a' i 'f'?

```
>>> nx.has_path(g, 'a', 'f')
False
```



graf g

Camins mínims amb atributs a valorar

Consultem la longitud del camí mínim segons l'etiqueta 'dist'

```
>>> nx.shortest_path_length(g, 'a', 'd', weight='dist')
4
```

Consultem un camí mínim

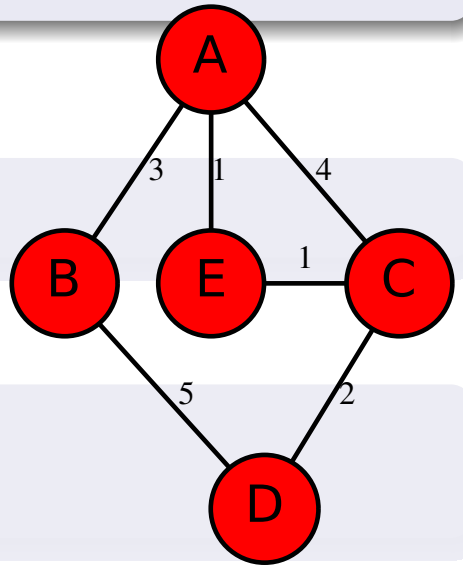
```
>>> nx.shortest_path(g, 'a', 'd', weight='dist')
['a', 'e', 'c', 'd']
```

Afegim un altre camí

```
>>> nx.add_path(g, ['f', 'g'])
```

Hi ha camí entre 'a' i 'f'?

```
>>> nx.has_path(g, 'a', 'f')
False
```



graf g

Camins mínims amb atributs a valorar

Consultem la longitud del camí mínim segons l'etiqueta 'dist'

```
>>> nx.shortest_path_length(g, 'a', 'd', weight='dist')
4
```

Consultem un camí mínim

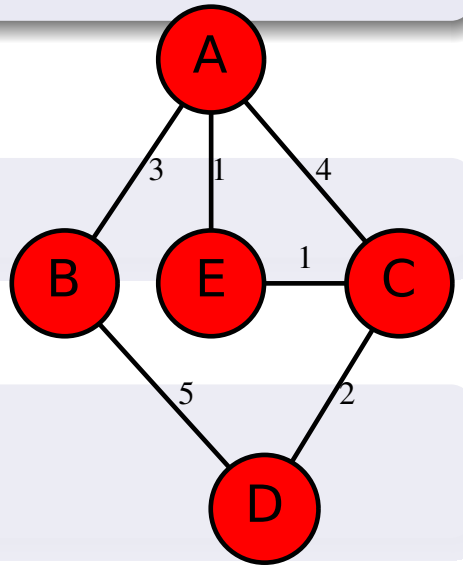
```
>>> nx.shortest_path(g, 'a', 'd', weight='dist')
['a', 'e', 'c', 'd']
```

Afegim un altre camí

```
>>> nx.add_path(g, ['f', 'g'])
```

Hi ha camí entre 'a' i 'f'?

```
>>> nx.has_path(g, 'a', 'f')
False
```



graf g

Camins mínims amb atributs a valorar

Consultem la longitud del camí mínim segons l'etiqueta 'dist'

```
>>> nx.shortest_path_length(g, 'a', 'd', weight='dist')
4
```

Consultem un camí mínim

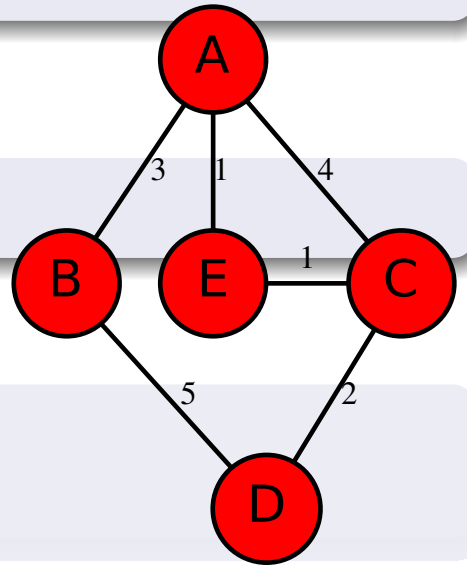
```
>>> nx.shortest_path(g, 'a', 'd', weight='dist')
['a', 'e', 'c', 'd']
```

Afegim un altre camí

```
>>> nx.add_path(g, ['f', 'g'])
```

Hi ha camí entre 'a' i 'f'?

```
>>> nx.has_path(g, 'a', 'f')
False
```



graf g

Camins mínims amb atributs a valorar

Consultem la longitud del camí mínim segons l'etiqueta 'dist'

```
>>> nx.shortest_path_length(g, 'a', 'd', weight='dist')
4
```

Consultem un camí mínim

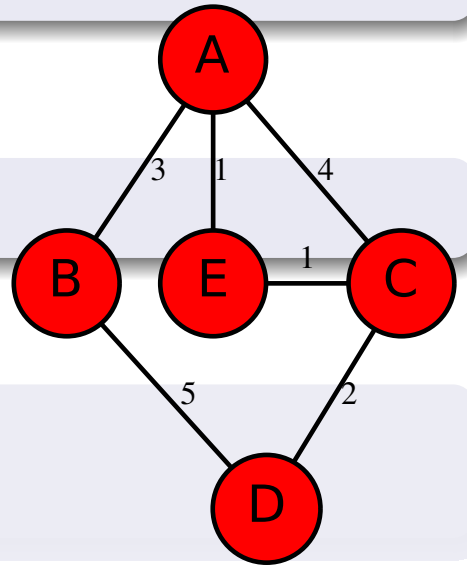
```
>>> nx.shortest_path(g, 'a', 'd', weight='dist')
['a', 'e', 'c', 'd']
```

Afegim un altre camí

```
>>> nx.add_path(g, ['f', 'g'])
```

Hi ha camí entre 'a' i 'f'?

```
>>> nx.has_path(g, 'a', 'f')
False
```



graf g

Camins mínims amb atributs a valorar

Consultem la longitud del camí mínim segons l'etiqueta 'dist'

```
>>> nx.shortest_path_length(g, 'a', 'd', weight='dist')
4
```

Consultem un camí mínim

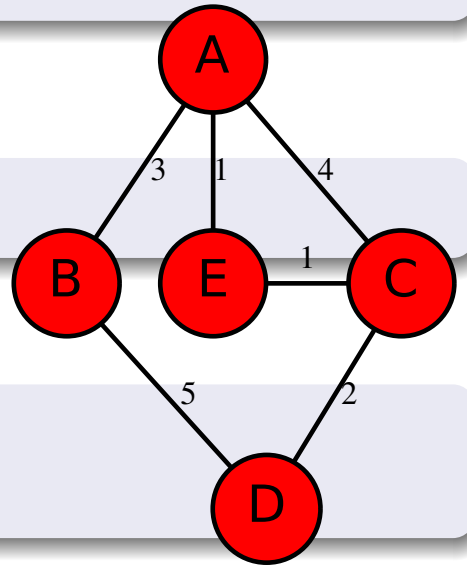
```
>>> nx.shortest_path(g, 'a', 'd', weight='dist')
['a', 'e', 'c', 'd']
```

Afegim un altre camí

```
>>> nx.add_path(g, ['f', 'g'])
```

Hi ha camí entre 'a' i 'f'?

```
>>> nx.has_path(g, 'a', 'f')
False
```



graf g

Camins mínims amb atributs a valorar

Consultem la longitud del camí mínim segons l'etiqueta 'dist'

```
>>> nx.shortest_path_length(g, 'a', 'd', weight='dist')
4
```

Consultem un camí mínim

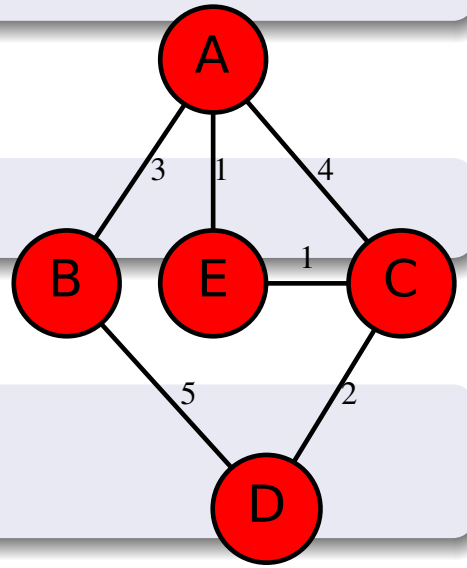
```
>>> nx.shortest_path(g, 'a', 'd', weight='dist')
['a', 'e', 'c', 'd']
```

Afegim un altre camí

```
>>> nx.add_path(g, ['f', 'g'])
```

Hi ha camí entre 'a' i 'f'?

```
>>> nx.has_path(g, 'a', 'f')
False
```



graf g

Camins mínims amb atributs a valorar

Consultem la longitud del camí mínim segons l'etiqueta 'dist'

```
>>> nx.shortest_path_length(g, 'a', 'd', weight='dist')
4
```

Consultem un camí mínim

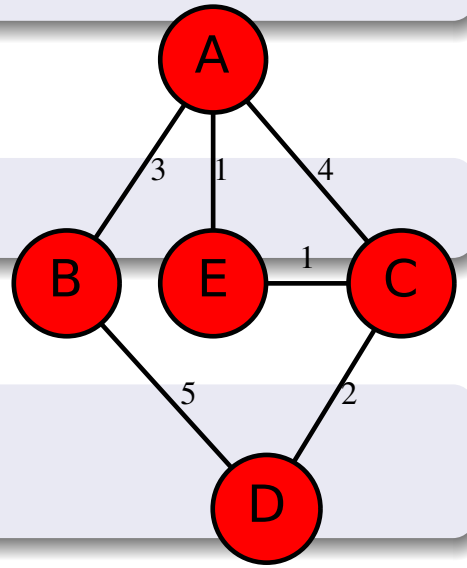
```
>>> nx.shortest_path(g, 'a', 'd', weight='dist')
['a', 'e', 'c', 'd']
```

Afegim un altre camí

```
>>> nx.add_path(g, ['f', 'g'])
```

Hi ha camí entre 'a' i 'f'?

```
>>> nx.has_path(g, 'a', 'f')
False
```



graf g

Components connectats. Graf exemple

```
>>> import networkx as nx
>>> g = nx.Graph()
>>> nx.add_path(g, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(g, ['a', 'e', 'c'])
```

Components connectats. Graf exemple

```
>>> import networkx as nx
>>> g = nx.Graph()
>>> nx.add_path(g, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(g, ['a', 'e', 'c'])
```

Components connectats. Graf exemple

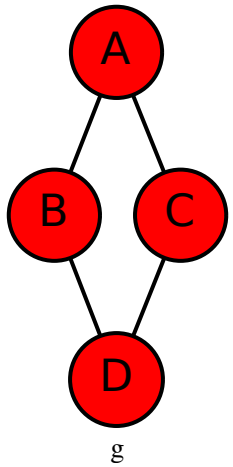

```
>>> import networkx as nx
>>> g = nx.Graph()
>>> nx.add_path(g, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(g, ['a', 'e', 'c'])
```

Components connectats. Graf exemple

```
>>> import networkx as nx
>>> g = nx.Graph()
>>> nx.add_path(g, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(g, ['a', 'e', 'c'])
```

Components connectats. Graf exemple

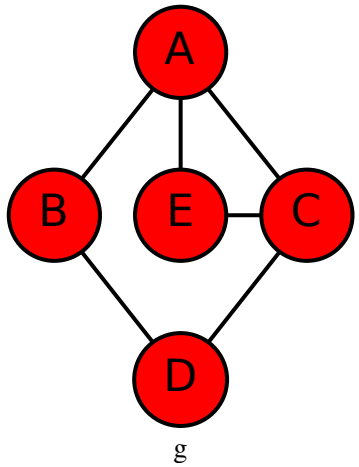

```
>>> import networkx as nx
>>> g = nx.Graph()
>>> nx.add_path(g, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(g, ['a', 'e', 'c'])
```



g

Components connectats. Graf exemple

```
>>> import networkx as nx
>>> g = nx.Graph()
>>> nx.add_path(g, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(g, ['a', 'e', 'c'])
```



g

Components connectats. Consultes

És connex?

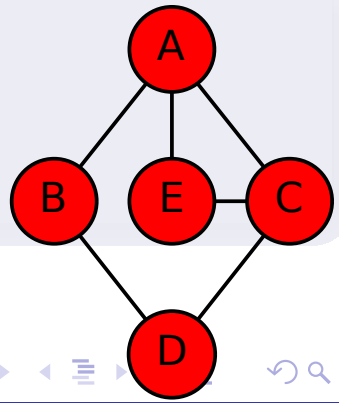
```
>>> nx.is_connected(g)
True
```

Quants components té?

```
>>> nx.number_connected_components(g)
1
```

Quins components té?

```
>>> for c in nx.connected_components(g):
...     print(sorted(c))
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

És connex?

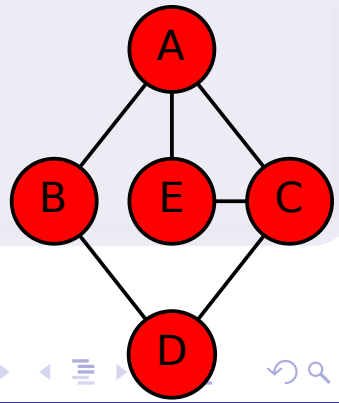
```
>>> nx.is_connected(g)
True
```

Quants components té?

```
>>> nx.number_connected_components(g)
1
```

Quins components té?

```
>>> for c in nx.connected_components(g):
...     print(sorted(c))
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

És connex?

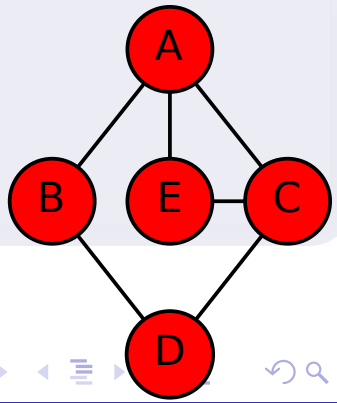
```
>>> nx.is_connected(g)
True
```

Quants components té?

```
>>> nx.number_connected_components(g)
1
```

Quins components té?

```
>>> for c in nx.connected_components(g):
...     print(sorted(c))
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

És connex?

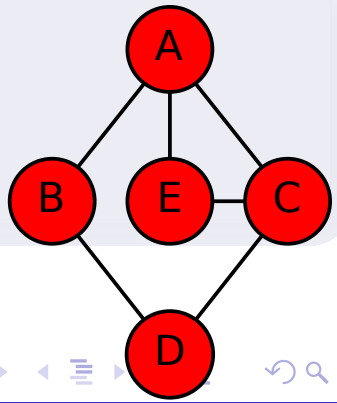
```
>>> nx.is_connected(g)
True
```

Quants components té?

```
>>> nx.number_connected_components(g)
1
```

Quins components té?

```
>>> for c in nx.connected_components(g):
...     print(sorted(c))
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

És connex?

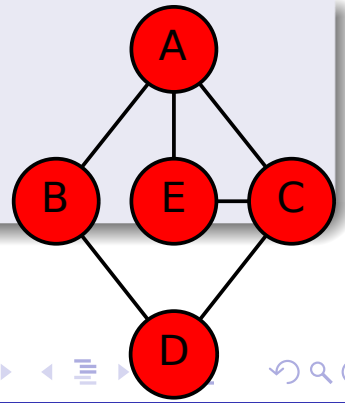
```
>>> nx.is_connected(g)
True
```

Quants components té?

```
>>> nx.number_connected_components(g)
1
```

Quins components té?

```
>>> for c in nx.connected_components(g):
...     print(sorted(c))
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

És connex?

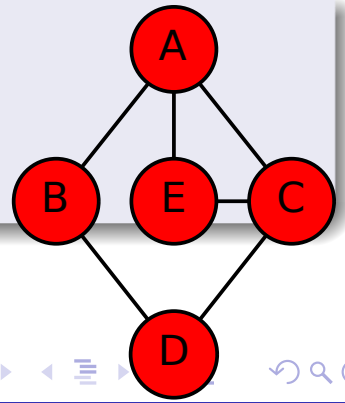
```
>>> nx.is_connected(g)
True
```

Quants components té?

```
>>> nx.number_connected_components(g)
1
```

Quins components té?

```
>>> for c in nx.connected_components(g):
...     print(sorted(c))
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

És connex?

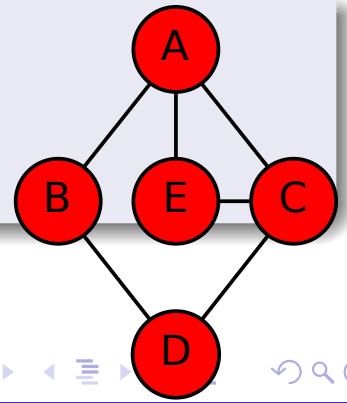
```
>>> nx.is_connected(g)
True
```

Quants components té?

```
>>> nx.number_connected_components(g)
1
```

Quins components té?

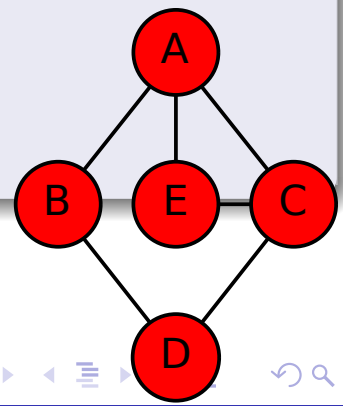
```
>>> for c in nx.connected_components(g):
...     print(sorted(c))
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

Hi afegim un altre camí

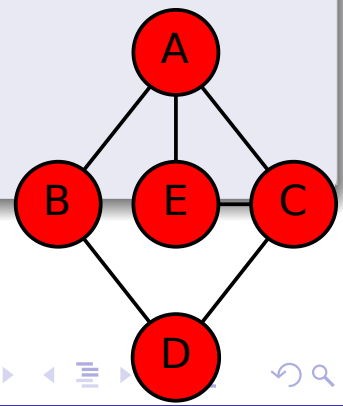
```
>>> nx.add_path(g, ['f', 'g'])
>>> nx.is_connected(g)
False
>>> nx.number_connected_components(g)
2
>>> for c in sorted(nx.connected_components(g), ke
y=len):
...     print(sorted(c))
...
['f', 'g']
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

Hi afegim un altre camí

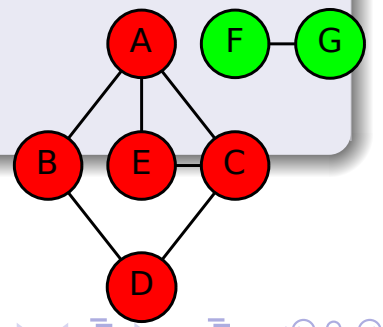
```
>>> nx.add_path(g, ['f', 'g'])
>>> nx.is_connected(g)
False
>>> nx.number_connected_components(g)
2
>>> for c in sorted(nx.connected_components(g), ke
y=len):
...     print(sorted(c))
...
['f', 'g']
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

Hi afegim un altre camí

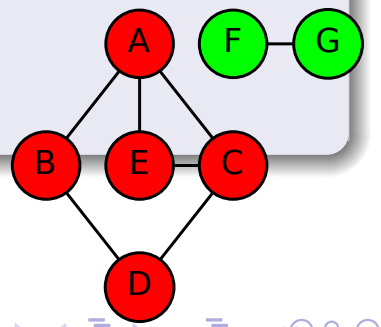
```
>>> nx.add_path(g, ['f', 'g'])
>>> nx.is_connected(g)
False
>>> nx.number_connected_components(g)
2
>>> for c in sorted(nx.connected_components(g), ke
y=len):
...     print(sorted(c))
...
['f', 'g']
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

Hi afegim un altre camí

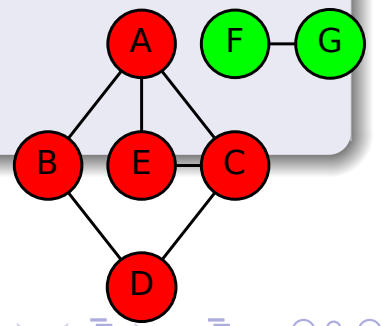
```
>>> nx.add_path(g, ['f', 'g'])
>>> nx.is_connected(g)
False
>>> nx.number_connected_components(g)
2
>>> for c in sorted(nx.connected_components(g), ke
y=len):
...     print(sorted(c))
...
['f', 'g']
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

Hi afegim un altre camí

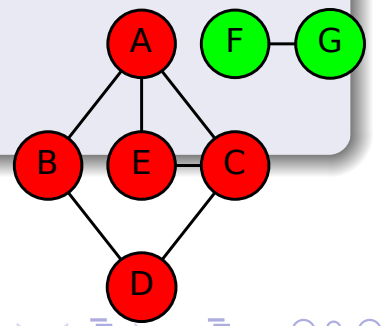
```
>>> nx.add_path(g, ['f', 'g'])
>>> nx.is_connected(g)
False
>>> nx.number_connected_components(g)
2
>>> for c in sorted(nx.connected_components(g), ke
y=len):
...     print(sorted(c))
...
['f', 'g']
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

Hi afegim un altre camí

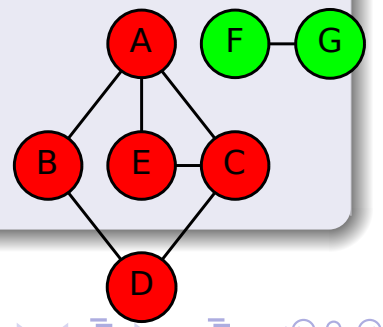
```
>>> nx.add_path(g, ['f', 'g'])
>>> nx.is_connected(g)
False
>>> nx.number_connected_components(g)
2
>>> for c in sorted(nx.connected_components(g), ke
y=len):
...     print(sorted(c))
...
['f', 'g']
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

Hi afegim un vèrtex

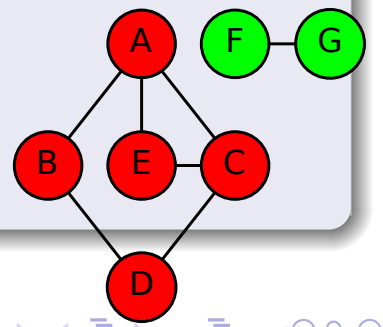
```
>>> g.add_node('h')
>>> nx.is_connected(g)
False
>>> nx.number_connected_components(g)
3
>>> for c in sorted(nx.connected_components(g), ke
y=len):
...     print(sorted(c))
...
['h']
['f', 'g']
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

Hi afegim un vèrtex

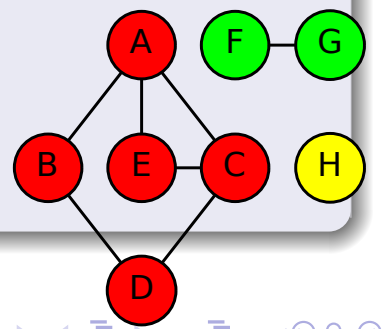
```
>>> g.add_node('h')
>>> nx.is_connected(g)
False
>>> nx.number_connected_components(g)
3
>>> for c in sorted(nx.connected_components(g), ke
y=len):
...     print(sorted(c))
...
['h']
['f', 'g']
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

Hi afegim un vèrtex

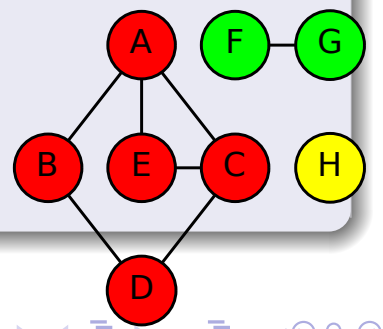
```
>>> g.add_node('h')
>>> nx.is_connected(g)
False
>>> nx.number_connected_components(g)
3
>>> for c in sorted(nx.connected_components(g), ke
y=len):
...     print(sorted(c))
...
['h']
['f', 'g']
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

Hi afegim un vèrtex

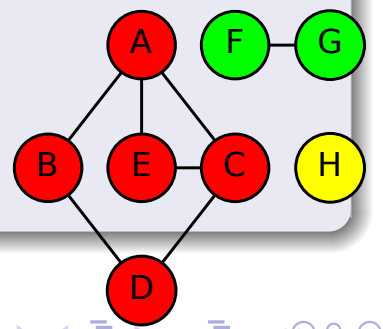
```
>>> g.add_node('h')
>>> nx.is_connected(g)
False
>>> nx.number_connected_components(g)
3
>>> for c in sorted(nx.connected_components(g), ke
y=len):
...     print(sorted(c))
...
['h']
['f', 'g']
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

Hi afegim un vèrtex

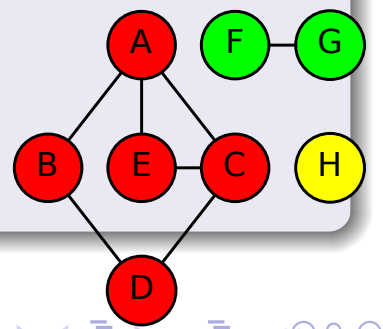
```
>>> g.add_node('h')
>>> nx.is_connected(g)
False
>>> nx.number_connected_components(g)
3
>>> for c in sorted(nx.connected_components(g), ke
y=len):
...     print(sorted(c))
...
['h']
['f', 'g']
['a', 'b', 'c', 'd', 'e']
```



Components connectats. Consultes

Hi afegim un vèrtex

```
>>> g.add_node('h')
>>> nx.is_connected(g)
False
>>> nx.number_connected_components(g)
3
>>> for c in sorted(nx.connected_components(g), ke
y=len):
...     print(sorted(c))
...
['h']
['f', 'g']
['a', 'b', 'c', 'd', 'e']
```



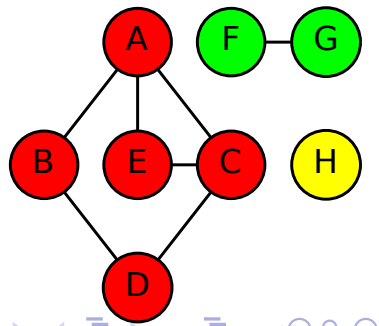
Components connectats. Consultes

Quin és el component de 'b'?

```
>>> nx.node_connected_component(g, 'b')
{'b', 'a', 'd', 'c', 'e'}
```

I el de 'f'?

```
>>> nx.node_connected_component(g, 'f')
{'g', 'f'}
```



components

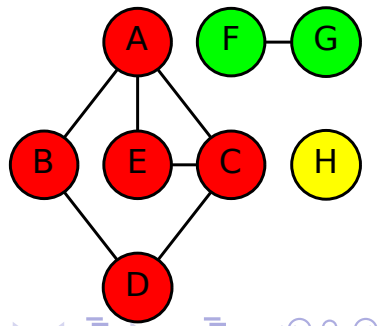
Components connectats. Consultes

Quin és el component de 'b'?

```
>>> nx.node_connected_component(g, 'b')
{'b', 'a', 'd', 'c', 'e'}
```

I el de 'f'?

```
>>> nx.node_connected_component(g, 'f')
{'g', 'f'}
```



components

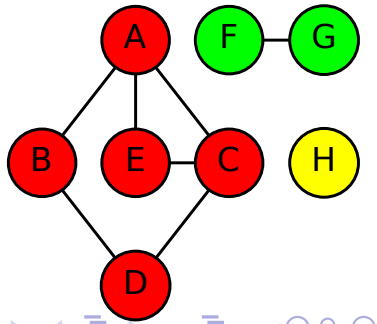
Components connectats. Consultes

Quin és el component de 'b'?

```
>>> nx.node_connected_component(g, 'b')
{'b', 'a', 'd', 'c', 'e'}
```

I el de 'f'?

```
>>> nx.node_connected_component(g, 'f')
{'g', 'f'}
```



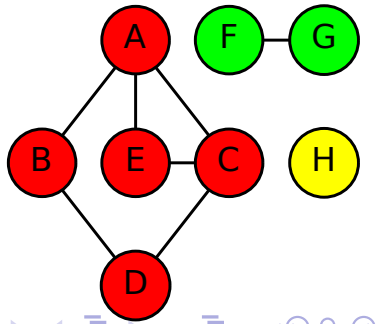
Components connectats. Consultes

Quin és el component de 'b'?

```
>>> nx.node_connected_component(g, 'b')
{'b', 'a', 'd', 'c', 'e'}
```

I el de 'f'?

```
>>> nx.node_connected_component(g, 'f')
{'g', 'f'}
```



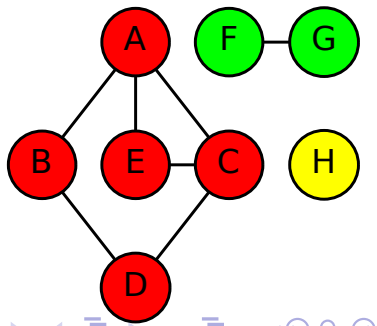
Components connectats. Consultes

Quin és el component de 'b'?

```
>>> nx.node_connected_component(g, 'b')
{'b', 'a', 'd', 'c', 'e'}
```

I el de 'f'?

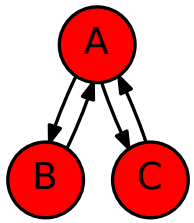
```
>>> nx.node_connected_component(g, 'f')
{'g', 'f'}
```



Creació i consulta d'un graf d'un graf dirigit


```
>>> import networkx as nx
>>> gd = nx.DiGraph()
>>> gd
<networkx.classes.digraph.DiGraph object at ...>
>>> gd.add_nodes_from(('a','b','c'))
>>> gd.add_edges_from((('a','b'),('a','c'),('b','a
'),('c','a')))
```

Ara tindrem quatre arestes, doncs les arestes ara tenen sentit.

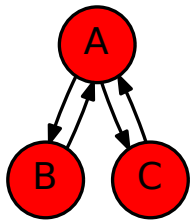


Graf dirigit dg

Creació i consulta d'un graf d'un graf dirigit


```
>>> import networkx as nx
>>> gd = nx.DiGraph()
>>> gd
<networkx.classes.digraph.DiGraph object at ...>
>>> gd.add_nodes_from(('a','b','c'))
>>> gd.add_edges_from((('a','b'),('a','c'),('b','a
'),('c','a')))
```

Ara tindrem quatre arestes, doncs les arestes ara tenen sentit.

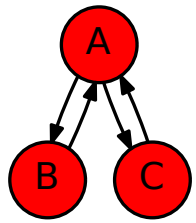


Graf dirigit dg

Creació i consulta d'un graf d'un graf dirigit


```
>>> import networkx as nx
>>> gd = nx.DiGraph()
>>> gd
<networkx.classes.digraph.DiGraph object at ...>
>>> gd.add_nodes_from(('a','b','c'))
>>> gd.add_edges_from((('a','b'),('a','c'),('b','a
'),('c','a')))
```

Ara tindrem quatre arestes, doncs les arestes ara tenen sentit.

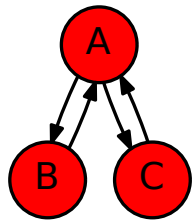


Graf dirigit dg

Creació i consulta d'un graf d'un graf dirigit


```
>>> import networkx as nx
>>> gd = nx.DiGraph()
>>> gd
<networkx.classes.digraph.DiGraph object at ...>
>>> gd.add_nodes_from(('a','b','c'))
>>> gd.add_edges_from((('a','b'),('a','c'),('b','a
'),('c','a')))
```

Ara tindrem quatre arestes, doncs les arestes ara tenen sentit.

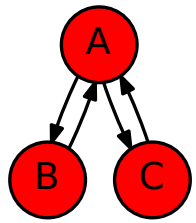


Graf dirigit dg

Creació i consulta d'un graf d'un graf dirigit


```
>>> import networkx as nx
>>> gd = nx.DiGraph()
>>> gd
<networkx.classes.digraph.DiGraph object at ...>
>>> gd.add_nodes_from(('a','b','c'))
>>> gd.add_edges_from((('a','b'),('a','c'),('b','a
'),('c','a')))
```

Ara tindrem quatre arestes, doncs les arestes ara tenen sentit.

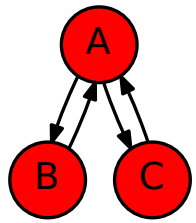


Graf dirigit dg

Creació i consulta d'un graf d'un graf dirigit


```
>>> import networkx as nx
>>> gd = nx.DiGraph()
>>> gd
<networkx.classes.digraph.DiGraph object at ...>
>>> gd.add_nodes_from(('a','b','c'))
>>> gd.add_edges_from((('a','b'),('a','c'),('b','a'),('c','a')))
```

Ara tindrem quatre arestes, doncs les arestes ara tenen sentit.

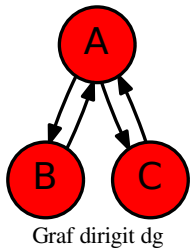


Graf dirigit dg

Creació i consulta d'un graf d'un graf dirigit


```
>>> import networkx as nx
>>> gd = nx.DiGraph()
>>> gd
<networkx.classes.digraph.DiGraph object at ...>
>>> gd.add_nodes_from(('a','b','c'))
>>> gd.add_edges_from((('a','b'),('a','c'),('b','a
'),('c','a')))
```

Ara tindrem quatre arestes, doncs les arestes ara tenen sentit.



Graf dirigit dg

Consulta d'un graf dirigit

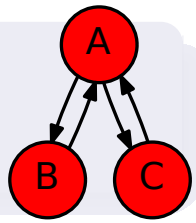

```
>>> gd.nodes()
NodeView(('a', 'b', 'c'))
>>> gd.edges()
OutEdgeView([('a', 'b'), ('a', 'c'), ('b', 'a'), (
'c', 'a')])
```

Consulta del node a

```
>>> gd['a']
AtlasView({'b': {}, 'c': {}})
```

Consulta de l'aresta ('a','b')

```
>>> gd['a']['b']
{}
```



Graf dirigit dg

Consulta d'un graf dirigit

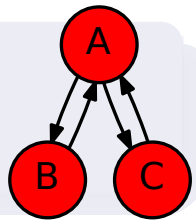

```
>>> gd.nodes()
NodeView(('a', 'b', 'c'))
>>> gd.edges()
OutEdgeView([('a', 'b'), ('a', 'c'), ('b', 'a'), (
'c', 'a')])
```

Consulta del node a

```
>>> gd['a']
AtlasView({'b': {}, 'c': {}})
```

Consulta de l'aresta ('a','b')

```
>>> gd['a']['b']
{}
```



Graf dirigit dg

Consulta d'un graf dirigit

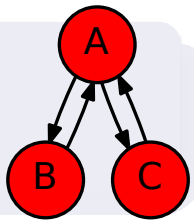
```
>>> gd.nodes()
NodeView(('a', 'b', 'c'))
>>> gd.edges()
OutEdgeView([('a', 'b'), ('a', 'c'), ('b', 'a'), (
'c', 'a')])
```

Consulta del node a

```
>>> gd['a']
AtlasView({'b': {}, 'c': {}})
```

Consulta de l'aresta ('a','b')

```
>>> gd['a']['b']
{}
```



Graf dirigit dg

Consulta d'un graf dirigit

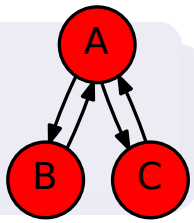
```
>>> gd.nodes()
NodeView(('a', 'b', 'c'))
>>> gd.edges()
OutEdgeView([('a', 'b'), ('a', 'c'), ('b', 'a'), (
'c', 'a')])
```

Consulta del node a

```
>>> gd['a']
AtlasView({'b': {}, 'c': {}})
```

Consulta de l'aresta ('a','b')

```
>>> gd['a']['b']
{}
```



Graf dirigit dg

Consulta d'un graf dirigit

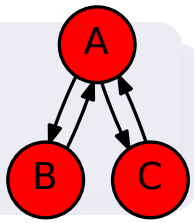
```
>>> gd.nodes()
NodeView(('a', 'b', 'c'))
>>> gd.edges()
OutEdgeView([('a', 'b'), ('a', 'c'), ('b', 'a'), (
'c', 'a')])
```

Consulta del node a

```
>>> gd['a']
AtlasView({'b': {}, 'c': {}})
```

Consulta de l'aresta ('a','b')

```
>>> gd['a']['b']
{}
```



Graf dirigit dg

Consulta d'un graf dirigit

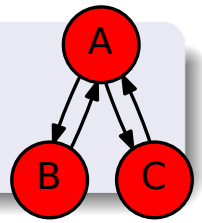

```
>>> gd.nodes()
NodeView(('a', 'b', 'c'))
>>> gd.edges()
OutEdgeView([('a', 'b'), ('a', 'c'), ('b', 'a'), (
'c', 'a')])
```

Consulta del node a

```
>>> gd['a']
AtlasView({'b': {}, 'c': {}})
```

Consulta de l'aresta ('a','b')

```
>>> gd['a']['b']
{}
```



Graf dirigit dg

Consulta d'un graf dirigit

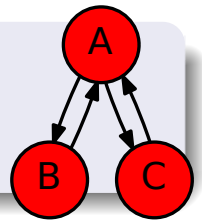

```
>>> gd.nodes()
NodeView(('a', 'b', 'c'))
>>> gd.edges()
OutEdgeView([('a', 'b'), ('a', 'c'), ('b', 'a'), (
'c', 'a')])
```

Consulta del node a

```
>>> gd['a']
AtlasView({'b': {}, 'c': {}})
```

Consulta de l'aresta ('a','b')

```
>>> gd['a']['b']
{}
```



Graf dirigit dg

Consulta d'un graf dirigit

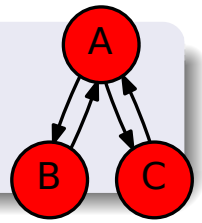
```
>>> gd.nodes()
NodeView(('a', 'b', 'c'))
>>> gd.edges()
OutEdgeView([('a', 'b'), ('a', 'c'), ('b', 'a'), (
'c', 'a')])
```

Consulta del node a

```
>>> gd['a']
AtlasView({'b': {}, 'c': {}})
```

Consulta de l'aresta ('a','b')

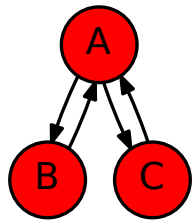
```
>>> gd['a']['b']
{}
```



Graf dirigit dg

Etiquetes o atributs d'una aresta

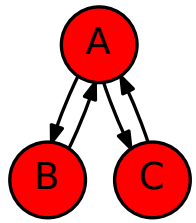

```
>>> gd.add_edge('a','b', nom='ab')
>>> gd.edges()
OutEdgeView([('a', 'b'), ('a', 'c'), ('b', 'a'), (
'c', 'a')])
>>> gd['a']['b']
{'nom': 'ab'}
>>> gd['a']['b']['nom']
'ab'
>>> gd['b']['a']
{}
```



Graf dirigit dg

Etiquetes o atributs d'una aresta

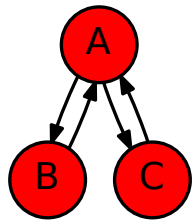

```
>>> gd.add_edge('a','b', nom='ab')
>>> gd.edges()
OutEdgeView([('a', 'b'), ('a', 'c'), ('b', 'a'), (
'c', 'a')])
>>> gd['a']['b']
{'nom': 'ab'}
>>> gd['a']['b']['nom']
'ab'
>>> gd['b']['a']
{}
```



Graf dirigit dg

Etiquetes o atributs d'una aresta

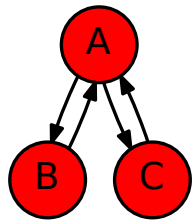

```
>>> gd.add_edge('a','b', nom='ab')
>>> gd.edges()
OutEdgeView([('a', 'b'), ('a', 'c'), ('b', 'a'), (
'c', 'a')])
>>> gd['a']['b']
{'nom': 'ab'}
>>> gd['a']['b']['nom']
'ab'
>>> gd['b']['a']
{}
```



Graf dirigit dg

Etiquetes o atributs d'una aresta

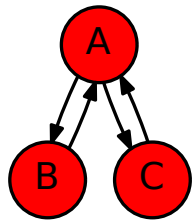

```
>>> gd.add_edge('a','b', nom='ab')
>>> gd.edges()
OutEdgeView([('a', 'b'), ('a', 'c'), ('b', 'a'), (
'c', 'a')])
>>> gd['a']['b']
{'nom': 'ab'}
>>> gd['a']['b']['nom']
'ab'
>>> gd['b']['a']
{}
```



Graf dirigit dg

Etiquetes o atributs d'una aresta

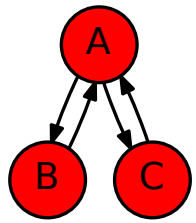

```
>>> gd.add_edge('a','b', nom='ab')
>>> gd.edges()
OutEdgeView([('a', 'b'), ('a', 'c'), ('b', 'a'), (
'c', 'a')])
>>> gd['a']['b']
{'nom': 'ab'}
>>> gd['a']['b']['nom']
'ab'
>>> gd['b']['a']
{}
```



Graf dirigit dg

Etiquetes o atributs d'una aresta

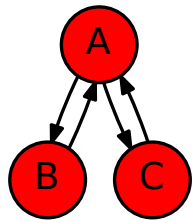

```
>>> gd.add_edge('a','b', nom='ab')
>>> gd.edges()
OutEdgeView([('a', 'b'), ('a', 'c'), ('b', 'a'), (
'c', 'a')])
>>> gd['a']['b']
{'nom': 'ab'}
>>> gd['a']['b']['nom']
'ab'
>>> gd['b']['a']
{}
```



Graf dirigit dg

Etiquetes o atributs d'una aresta


```
>>> gd.add_edge('a','b', nom='ab')
>>> gd.edges()
OutEdgeView([('a', 'b'), ('a', 'c'), ('b', 'a'), (
'c', 'a')])
>>> gd['a']['b']
{'nom': 'ab'}
>>> gd['a']['b']['nom']
'ab'
>>> gd['b']['a']
{}
```



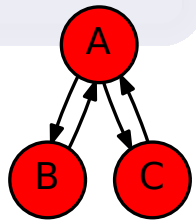
Graf dirigit dg

Més consultes d'un graf dirigit


```
>>> gd.size()
4
>>> len(gd)
3
```

recorregut nodes

```
>>> for node in gd:
...     print(node)
a
b
c
```



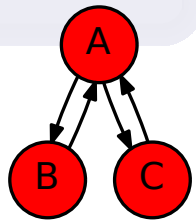
Graf dirigit dg

Més consultes d'un graf dirigit


```
>>> gd.size()
4
>>> len(gd)
3
```

recorregut nodes

```
>>> for node in gd:
...     print(node)
a
b
c
```



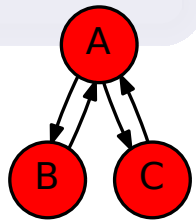
Graf dirigit dg

Més consultes d'un graf dirigit

```
>>> gd.size()
4
>>> len(gd)
3
```

recorregut nodes

```
>>> for node in gd:
...     print(node)
a
b
c
```



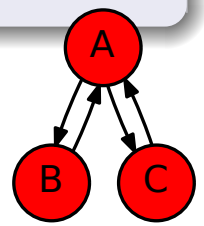
Graf dirigit dg

Més consultes d'un graf dirigit

```
>>> gd.size()
4
>>> len(gd)
3
```

recorregut nodes

```
>>> for node in gd:
...     print(node)
a
b
c
```



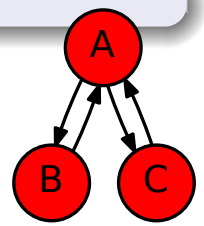
Graf dirigit dg

Més consultes d'un graf dirigit

```
>>> gd.size()
4
>>> len(gd)
3
```

recorregut nodes

```
>>> for node in gd:
...     print(node)
a
b
c
```



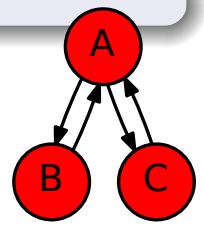
Graf dirigit dg

Més consultes d'un graf dirigit

```
>>> gd.size()
4
>>> len(gd)
3
```

recorregut nodes

```
>>> for node in gd:
...     print(node)
a
b
c
```



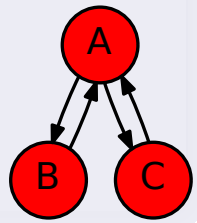
Graf dirigit dg

Veïns, sucessors, predecessors.


```
>>> for node in gd['b']:
...     print(node)
a
>>> for node in gd.neighbors('b'):
...     print(node)
a
```

neighbors aquí és el mateix que successors

```
>>> for node in gd.successors('b'):
...     print(node)
a
>>> gd.succ['b']
AtlasView({'a': {}})
>>> for node in gd.predecessors('b'):
...     print(node)
a
>>> gd.pred['b']
AtlasView({'a': {'nom': 'ab'}})
```

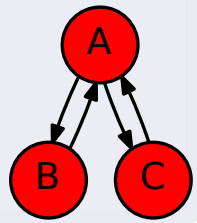


Veïns, sucessors, predecessors.


```
>>> for node in gd['b']:
...     print(node)
a
>>> for node in gd.neighbors('b'):
...     print(node)
a
```

neighbors aquí és el mateix que successors

```
>>> for node in gd.successors('b'):
...     print(node)
a
>>> gd.succ['b']
AtlasView({'a': {}})
>>> for node in gd.predecessors('b'):
...     print(node)
a
>>> gd.pred['b']
AtlasView({'a': {'nom': 'ab'}})
```

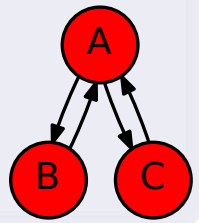


Veïns, sucessors, predecessors.


```
>>> for node in gd['b']:
...     print(node)
a
>>> for node in gd.neighbors('b'):
...     print(node)
a
```

neighbors aquí és el mateix que successors

```
>>> for node in gd.successors('b'):
...     print(node)
a
>>> gd.succ['b']
AtlasView({'a': {}})
>>> for node in gd.predecessors('b'):
...     print(node)
a
>>> gd.pred['b']
AtlasView({'a': {'nom': 'ab'}})
```

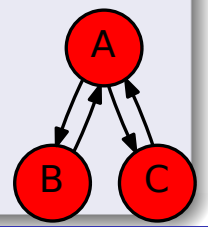


Veïns, sucessors, predecessors.

```
>>> for node in gd['b']:
...     print(node)
a
>>> for node in gd.neighbors('b'):
...     print(node)
a
```

neighbors aquí és el mateix que successors

```
>>> for node in gd.successors('b'):
...     print(node)
a
>>> gd.succ['b']
AtlasView({'a': {}})
>>> for node in gd.predecessors('b'):
...     print(node)
a
>>> gd.pred['b']
AtlasView({'a': {'nom': 'ab'}})
```

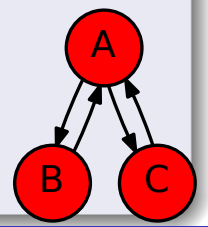


Veïns, sucessors, predecessors.

```
>>> for node in gd['b']:
...     print(node)
a
>>> for node in gd.neighbors('b'):
...     print(node)
a
```

neighbors aquí és el mateix que successors

```
>>> for node in gd.successors('b'):
...     print(node)
a
>>> gd.succ['b']
AtlasView({'a': {}})
>>> for node in gd.predecessors('b'):
...     print(node)
a
>>> gd.pred['b']
AtlasView({'a': {'nom': 'ab'}})
```

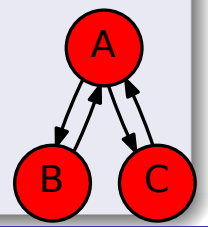


Veïns, sucessors, predecessors.

```
>>> for node in gd['b']:
...     print(node)
a
>>> for node in gd.neighbors('b'):
...     print(node)
a
```

neighbors aquí és el mateix que successors

```
>>> for node in gd.successors('b'):
...     print(node)
a
>>> gd.succ['b']
AtlasView({'a': {}})
>>> for node in gd.predecessors('b'):
...     print(node)
a
>>> gd.pred['b']
AtlasView({'a': {'nom': 'ab'}})
```

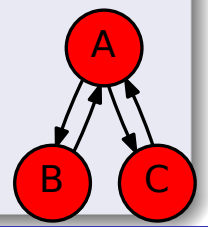


Veïns, sucessors, predecessors.

```
>>> for node in gd['b']:
...     print(node)
a
>>> for node in gd.neighbors('b'):
...     print(node)
a
```

neighbors aquí és el mateix que successors

```
>>> for node in gd.successors('b'):
...     print(node)
a
>>> gd.succ['b']
AtlasView({'a': {}})
>>> for node in gd.predecessors('b'):
...     print(node)
a
>>> gd.pred['b']
AtlasView({'a': {'nom': 'ab'}})
```

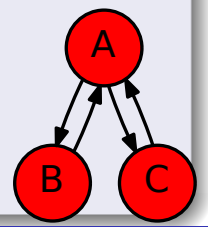


Veïns, sucessors, predecessors.

```
>>> for node in gd['b']:
...     print(node)
a
>>> for node in gd.neighbors('b'):
...     print(node)
a
```

neighbors aquí és el mateix que successors

```
>>> for node in gd.successors('b'):
...     print(node)
a
>>> gd.succ['b']
AtlasView({'a': {}})
>>> for node in gd.predecessors('b'):
...     print(node)
a
>>> gd.pred['b']
AtlasView({'a': {'nom': 'ab'}})
```

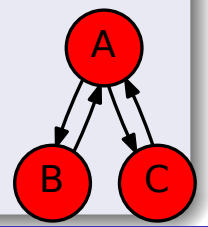


Veïns, sucessors, predecessors.

```
>>> for node in gd['b']:
...     print(node)
a
>>> for node in gd.neighbors('b'):
...     print(node)
a
```

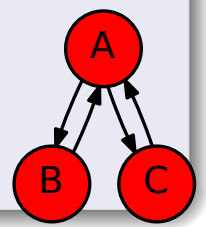
neighbors aquí és el mateix que successors

```
>>> for node in gd.successors('b'):
...     print(node)
a
>>> gd.succ['b']
AtlasView({'a': {}})
>>> for node in gd.predecessors('b'):
...     print(node)
a
>>> gd.pred['b']
AtlasView({'a': {'nom': 'ab'}})
```



Veïns, sucessors, predecessors.

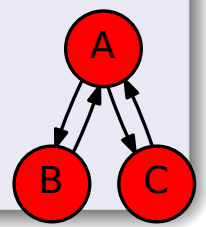

```
>>> gd.remove_edge('a','b')
>>> for node in gd.predecessors('b'):
...     print(node)
...
>>> for node in gd.predecessors('a'):
...     print(node, end=', ')
...
b, c,
>>> for node in gd.successors('a'):
...     print(node, end=', ')
...
c,
>>> for node in gd.neighbors('a'):
...     print(node, end=', ')
...
c,
```



Graf dirigit dg

Veïns, sucessors, predecessors.

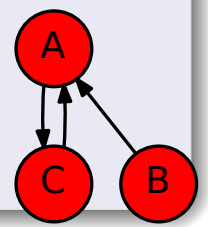

```
>>> gd.remove_edge('a','b')
>>> for node in gd.predecessors('b'):
...     print(node)
...
>>> for node in gd.predecessors('a'):
...     print(node, end=', ')
...
b, c,
>>> for node in gd.successors('a'):
...     print(node, end=', ')
...
c,
>>> for node in gd.neighbors('a'):
...     print(node, end=', ')
...
c,
```



Graf dirigit dg

Veïns, sucessors, predecessors.

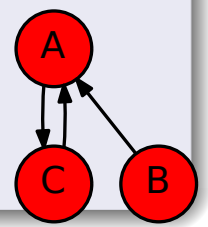

```
>>> gd.remove_edge('a','b')
>>> for node in gd.predecessors('b'):
...     print(node)
...
>>> for node in gd.predecessors('a'):
...     print(node, end=', ')
...
b, c,
>>> for node in gd.successors('a'):
...     print(node, end=', ')
...
c,
>>> for node in gd.neighbors('a'):
...     print(node, end=', ')
...
c,
```



Graf dirigit dg

Veïns, sucessors, predecessors.

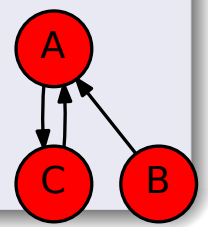

```
>>> gd.remove_edge('a','b')
>>> for node in gd.predecessors('b'):
...     print(node)
...
>>> for node in gd.predecessors('a'):
...     print(node, end=', ')
...
b, c,
>>> for node in gd.successors('a'):
...     print(node, end=', ')
...
c,
>>> for node in gd.neighbors('a'):
...     print(node, end=', ')
...
c,
```



Graf dirigit dg

Veïns, sucessors, predecessors.


```
>>> gd.remove_edge('a','b')
>>> for node in gd.predecessors('b'):
...     print(node)
...
>>> for node in gd.predecessors('a'):
...     print(node, end=', ')
...
b, c,
>>> for node in gd.successors('a'):
...     print(node, end=', ')
...
c,
>>> for node in gd.neighbors('a'):
...     print(node, end=', ')
...
c,
```



Graf dirigit dg

Veïns, sucessors, predecessors.

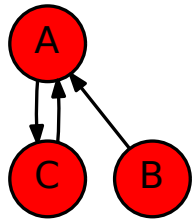
```
>>> gd.remove_edge('a','b')
>>> for node in gd.predecessors('b'):
...     print(node)
...
>>> for node in gd.predecessors('a'):
...     print(node, end=', ')
...
b, c,
>>> for node in gd.successors('a'):
...     print(node, end=', ')
...
c,
>>> for node in gd.neighbors('a'):
...     print(node, end=', ')
...
c,
```

Veïns, sucessors, predecessors.


```
>>> gd.remove_edge('a','b')
>>> for node in gd.predecessors('b'):
...     print(node)
...
>>> for node in gd.predecessors('a'):
...     print(node, end=', ')
...
b, c,
>>> for node in gd.successors('a'):
...     print(node, end=', ')
...
c,
>>> for node in gd.neighbors('a'):
...     print(node, end=', ')
...
c,
```

Graus d'un node

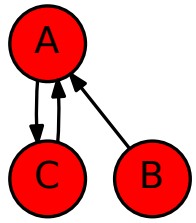

```
>>> gd.in_degree('a')
2
>>> gd.out_degree('a')
1
>>> gd.degree('a')
3
>>> gd.in_degree('b')
0
>>> gd.degree()
DiDegreeView({'a': 3, 'b': 1, 'c': 2})
```



Graf dirigit dg

Graus d'un node

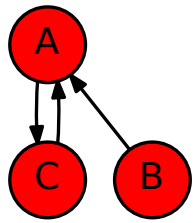

```
>>> gd.in_degree('a')
2
>>> gd.out_degree('a')
1
>>> gd.degree('a')
3
>>> gd.in_degree('b')
0
>>> gd.degree()
DiDegreeView({'a': 3, 'b': 1, 'c': 2})
```



Graf dirigit dg

Graus d'un node

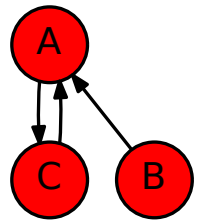

```
>>> gd.in_degree('a')
2
>>> gd.out_degree('a')
1
>>> gd.degree('a')
3
>>> gd.in_degree('b')
0
>>> gd.degree()
DiDegreeView({'a': 3, 'b': 1, 'c': 2})
```



Graf dirigit dg

Graus d'un node

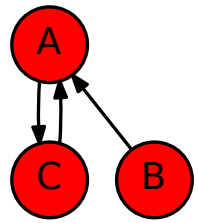

```
>>> gd.in_degree('a')
2
>>> gd.out_degree('a')
1
>>> gd.degree('a')
3
>>> gd.in_degree('b')
0
>>> gd.degree()
DiDegreeView({'a': 3, 'b': 1, 'c': 2})
```



Graf dirigit dg

Graus d'un node

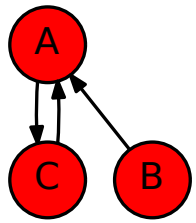

```
>>> gd.in_degree('a')
2
>>> gd.out_degree('a')
1
>>> gd.degree('a')
3
>>> gd.in_degree('b')
0
>>> gd.degree()
DiDegreeView({'a': 3, 'b': 1, 'c': 2})
```



Graf dirigit dg

Graus d'un node

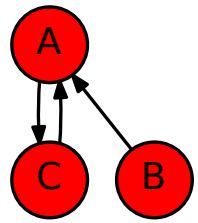

```
>>> gd.in_degree('a')
2
>>> gd.out_degree('a')
1
>>> gd.degree('a')
3
>>> gd.in_degree('b')
0
>>> gd.degree()
DiDegreeView({'a': 3, 'b': 1, 'c': 2})
```



Graf dirigit dg

Graus d'un node


```
>>> gd.in_degree('a')
2
>>> gd.out_degree('a')
1
>>> gd.degree('a')
3
>>> gd.in_degree('b')
0
>>> gd.degree()
DiDegreeView({'a': 3, 'b': 1, 'c': 2})
```

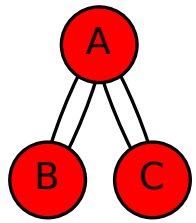


Graf dirigit dg

Creació d'un multigraf


```
>>> import networkx as nx
>>> mg = nx.MultiGraph()
>>> mg.add_edges_from((('a','b'),('a','c'),('b','a
'),('c','a')))
[0, 0, 1, 1]
```

Ara tindrem arestes repetides.

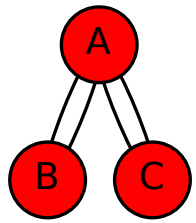


multigraf mg

Creació d'un multigraf


```
>>> import networkx as nx
>>> mg = nx.MultiGraph()
>>> mg.add_edges_from((('a','b'),('a','c'),('b','a
'),('c','a')))
[0, 0, 1, 1]
```

Ara tindrem arestes repetides.

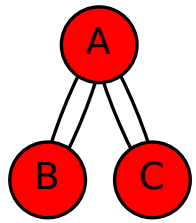


multigraf mg

Creació d'un multigraf


```
>>> import networkx as nx
>>> mg = nx.MultiGraph()
>>> mg.add_edges_from((('a','b'),('a','c'),('b','a
'),('c','a')))
[0, 0, 1, 1]
```

Ara tindrem arestes repetides.

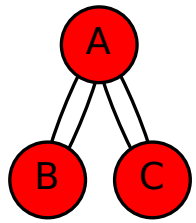


multigraf mg

Creació d'un multigraf


```
>>> import networkx as nx
>>> mg = nx.MultiGraph()
>>> mg.add_edges_from((('a','b'),('a','c'),('b','a
'),('c','a')))
[0, 0, 1, 1]
```

Ara tindrem arestes repetides.

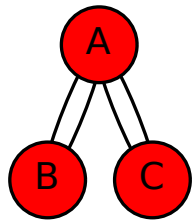


multigraf mg

Creació d'un multigraf


```
>>> import networkx as nx
>>> mg = nx.MultiGraph()
>>> mg.add_edges_from((('a','b'),('a','c'),('b','a
'),('c','a')))
[0, 0, 1, 1]
```

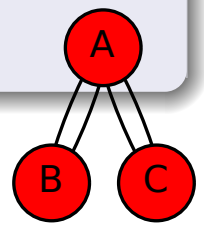
Ara tindrem arestes repetides.



multigraf mg

Consultes

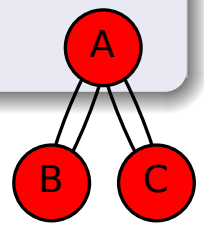
```
>>> mg.edges()
MultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', '
c'), ('a', 'c')])
>>> mg.nodes()
NodeView(('a', 'b', 'c'))
>>> mg['a']
AdjacencyView({'b': {0: {}, 1: {}}, 'c': {0: {}, 1
: {}}})
>>> mg['a']['b']
AtlasView({0: {}, 1: {}})
>>> mg['a']['b'][0]
{}
```



multigraf mg

Consultes

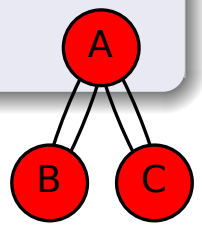
```
>>> mg.edges()
MultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', '
c'), ('a', 'c')])
>>> mg.nodes()
NodeView(('a', 'b', 'c'))
>>> mg['a']
AdjacencyView({'b': {0: {}, 1: {}}, 'c': {0: {}, 1
: {}}})
>>> mg['a']['b']
AtlasView({0: {}, 1: {}})
>>> mg['a']['b'][0]
{}
```



multigraf mg

Consultes

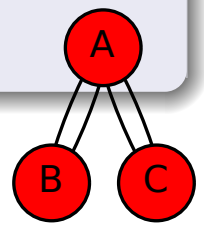

```
>>> mg.edges()
MultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', '
c'), ('a', 'c')])
>>> mg.nodes()
NodeView(('a', 'b', 'c'))
>>> mg['a']
AdjacencyView({'b': {0: {}, 1: {}}, 'c': {0: {}, 1
: {}}})
>>> mg['a']['b']
AtlasView({0: {}, 1: {}})
>>> mg['a']['b'][0]
{}
```



multigraf mg

Consultes

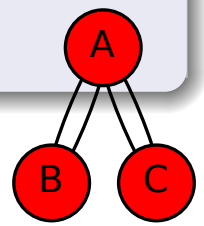
```
>>> mg.edges()
MultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', '
c'), ('a', 'c')])
>>> mg.nodes()
NodeView(('a', 'b', 'c'))
>>> mg['a']
AdjacencyView({'b': {0: {}, 1: {}}, 'c': {0: {}, 1
: {}}})
>>> mg['a']['b']
AtlasView({0: {}, 1: {}})
>>> mg['a']['b'][0]
{}
```



multigraf mg

Consultes

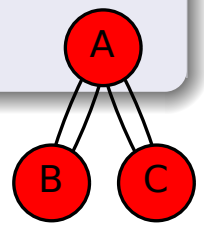
```
>>> mg.edges()
MultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', '
c'), ('a', 'c')])
>>> mg.nodes()
NodeView(('a', 'b', 'c'))
>>> mg['a']
AdjacencyView({'b': {0: {}, 1: {}}, 'c': {0: {}, 1
: {}}})
>>> mg['a']['b']
AtlasView({0: {}, 1: {}})
>>> mg['a']['b'][0]
{}
```



multigraf mg

Consultes

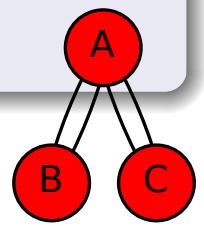
```
>>> mg.edges()
MultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', '
c'), ('a', 'c')])
>>> mg.nodes()
NodeView(('a', 'b', 'c'))
>>> mg['a']
AdjacencyView({'b': {0: {}, 1: {}}, 'c': {0: {}, 1
: {}}})
>>> mg['a']['b']
AtlasView({0: {}, 1: {}})
>>> mg['a']['b'][0]
{}
```



multigraf mg

Consultes

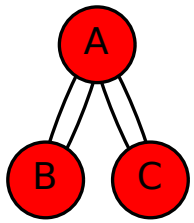
```
>>> mg.edges()
MultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', '
c'), ('a', 'c')])
>>> mg.nodes()
NodeView(('a', 'b', 'c'))
>>> mg['a']
AdjacencyView({'b': {0: {}, 1: {}}, 'c': {0: {}, 1
: {}}})
>>> mg['a']['b']
AtlasView({0: {}, 1: {}})
>>> mg['a']['b'][0]
{}
```



multigraf mg

Etiquetes

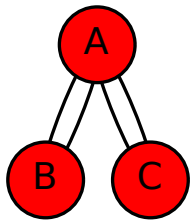

```
>>> mg['a']['b'][0]['nom']='ab'
>>> mg['a']['b'][0]
{'nom': 'ab'}
```



multigraf mg

Etiquetes

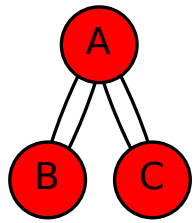

```
>>> mg['a']['b'][0]['nom']='ab'
>>> mg['a']['b'][0]
{'nom': 'ab'}
```



multigraf mg

Etiquetes

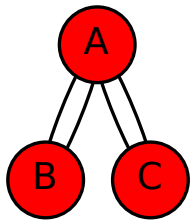

```
>>> mg['a']['b'][0]['nom']='ab'
>>> mg['a']['b'][0]
{'nom': 'ab'}
```



multigraf mg

Etiquetes

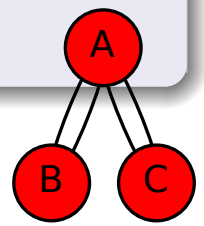

```
>>> mg['a']['b'][0]['nom']='ab'
>>> mg['a']['b'][0]
{'nom': 'ab'}
```



multigraf mg

altres consultes, . . . veïns

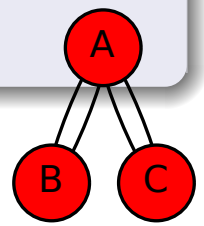

```
>>> mg.size()
4
>>> len(mg)
3
>>> for node in mg.neighbors('a'):
...     print(node)
b
c
>>> for node in mg['a']:
...     print(node)
b
c
```



multigraf mg

altres consultes, . . . veïns

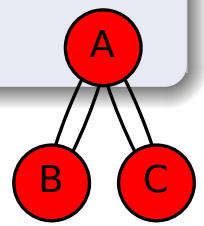

```
>>> mg.size()
4
>>> len(mg)
3
>>> for node in mg.neighbors('a'):
...     print(node)
b
c
>>> for node in mg['a']:
...     print(node)
b
c
```



multigraf mg

altres consultes, . . . veïns

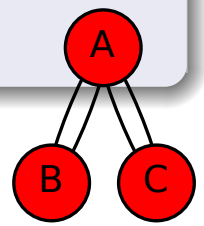

```
>>> mg.size()
4
>>> len(mg)
3
>>> for node in mg.neighbors('a'):
...     print(node)
b
c
>>> for node in mg['a']:
...     print(node)
b
c
```



multigraf mg

altres consultes, . . . veïns

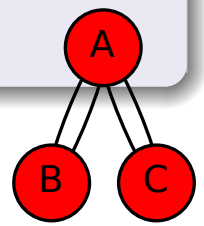

```
>>> mg.size()
4
>>> len(mg)
3
>>> for node in mg.neighbors('a'):
...     print(node)
b
c
>>> for node in mg['a']:
...     print(node)
b
c
```



multigraf mg

altres consultes, . . . veïns

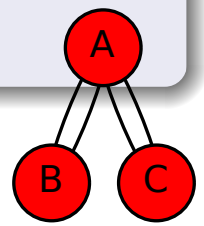
```
>>> mg.size()
4
>>> len(mg)
3
>>> for node in mg.neighbors('a'):
...     print(node)
b
c
>>> for node in mg['a']:
...     print(node)
b
c
```



multigraf mg

altres consultes, . . . veïns


```
>>> mg.size()
4
>>> len(mg)
3
>>> for node in mg.neighbors('a'):
...     print(node)
b
c
>>> for node in mg['a']:
...     print(node)
b
c
```



multigraf mg

Un altre exemple de multigraf


```
>>> mg = nx.MultiGraph()
>>> nx.add_path(mg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mg, ['a', 'e', 'c'])
>>> mg.add_edge('a','b')
1
>>> mg.add_edge('b','a')
2
>>> mg.add_edge('b','a')
3
>>> mg.add_edge('b','d')
1
>>> mg.add_edge('b','d')
2
```

Un altre exemple de multigraf


```
>>> mg = nx.MultiGraph()
>>> nx.add_path(mg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mg, ['a', 'e', 'c'])
>>> mg.add_edge('a','b')
1
>>> mg.add_edge('b','a')
2
>>> mg.add_edge('b','a')
3
>>> mg.add_edge('b','d')
1
>>> mg.add_edge('b','d')
2
```

Un altre exemple de multigraf

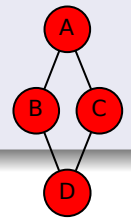

```
>>> mg = nx.MultiGraph()
>>> nx.add_path(mg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mg, ['a', 'e', 'c'])
>>> mg.add_edge('a','b')
1
>>> mg.add_edge('b','a')
2
>>> mg.add_edge('b','a')
3
>>> mg.add_edge('b','d')
1
>>> mg.add_edge('b','d')
2
```

Un altre exemple de multigraf


```
>>> mg = nx.MultiGraph()
>>> nx.add_path(mg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mg, ['a', 'e', 'c'])
>>> mg.add_edge('a','b')
1
>>> mg.add_edge('b','a')
2
>>> mg.add_edge('b','a')
3
>>> mg.add_edge('b','d')
1
>>> mg.add_edge('b','d')
2
```

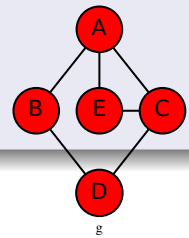
Un altre exemple de multigraf


```
>>> mg = nx.MultiGraph()
>>> nx.add_path(mg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mg, ['a', 'e', 'c'])
>>> mg.add_edge('a','b')
1
>>> mg.add_edge('b','a')
2
>>> mg.add_edge('b','a')
3
>>> mg.add_edge('b','d')
1
>>> mg.add_edge('b','d')
2
```



Un altre exemple de multigraf

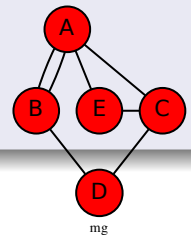

```
>>> mg = nx.MultiGraph()
>>> nx.add_path(mg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mg, ['a', 'e', 'c'])
>>> mg.add_edge('a','b')
1
>>> mg.add_edge('b','a')
2
>>> mg.add_edge('b','a')
3
>>> mg.add_edge('b','d')
1
>>> mg.add_edge('b','d')
2
```



g

Un altre exemple de multigraf

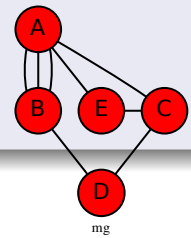

```
>>> mg = nx.MultiGraph()
>>> nx.add_path(mg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mg, ['a', 'e', 'c'])
>>> mg.add_edge('a','b')
1
>>> mg.add_edge('b','a')
2
>>> mg.add_edge('b','a')
3
>>> mg.add_edge('b','d')
1
>>> mg.add_edge('b','d')
2
```



mg

Un altre exemple de multigraf

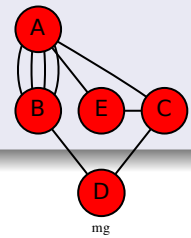

```
>>> mg = nx.MultiGraph()
>>> nx.add_path(mg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mg, ['a', 'e', 'c'])
>>> mg.add_edge('a','b')
1
>>> mg.add_edge('b','a')
2
>>> mg.add_edge('b','a')
3
>>> mg.add_edge('b','d')
1
>>> mg.add_edge('b','d')
2
```



mg

Un altre exemple de multigraf

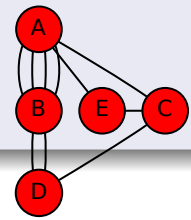

```
>>> mg = nx.MultiGraph()
>>> nx.add_path(mg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mg, ['a', 'e', 'c'])
>>> mg.add_edge('a','b')
1
>>> mg.add_edge('b','a')
2
>>> mg.add_edge('b','a')
3
>>> mg.add_edge('b','d')
1
>>> mg.add_edge('b','d')
2
```



mg

Un altre exemple de multigraf

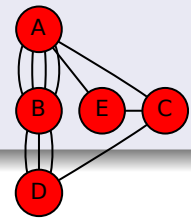

```
>>> mg = nx.MultiGraph()
>>> nx.add_path(mg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mg, ['a', 'e', 'c'])
>>> mg.add_edge('a','b')
1
>>> mg.add_edge('b','a')
2
>>> mg.add_edge('b','a')
3
>>> mg.add_edge('b','d')
1
>>> mg.add_edge('b','d')
2
```



mg

Un altre exemple de multigraf


```
>>> mg = nx.MultiGraph()
>>> nx.add_path(mg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mg, ['a', 'e', 'c'])
>>> mg.add_edge('a','b')
1
>>> mg.add_edge('b','a')
2
>>> mg.add_edge('b','a')
3
>>> mg.add_edge('b','d')
1
>>> mg.add_edge('b','d')
2
```

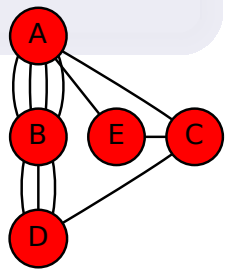


mg

Consultes, camins, grau


```
>>> mg.edges()
MultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', 'b'),
('a', 'b'), ('a', 'c'), ('a', 'e'), ('b', 'd'), ('b', '
d'), ('b', 'd'), ('d', 'c'), ('c', 'e')])
```

```
>>> for cami in nx.all_shortest_paths(mg, 'a','d')
:
...     print(cami)
['a', 'b', 'd']
['a', 'c', 'd']
>>> mg.degree('a')
6
```

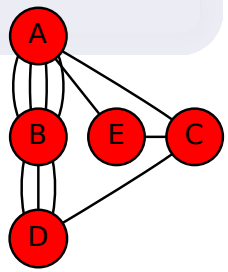


mg

Consultes, camins, grau


```
>>> mg.edges()
MultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', 'b'),
('a', 'b'), ('a', 'c'), ('a', 'e'), ('b', 'd'), ('b', '
d'), ('b', 'd'), ('d', 'c'), ('c', 'e')])
```

```
>>> for cami in nx.all_shortest_paths(mg, 'a','d')
:
...     print(cami)
['a', 'b', 'd']
['a', 'c', 'd']
>>> mg.degree('a')
6
```

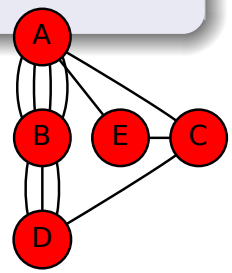


mg

Consultes, camins, grau


```
>>> mg.edges()
MultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', 'b'),
('a', 'b'), ('a', 'c'), ('a', 'e'), ('b', 'd'), ('b', '
d'), ('b', 'd'), ('d', 'c'), ('c', 'e')])
```

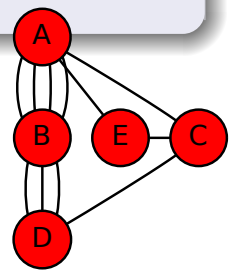
```
>>> for cami in nx.all_shortest_paths(mg, 'a','d')
:
...     print(cami)
['a', 'b', 'd']
['a', 'c', 'd']
>>> mg.degree('a')
6
```



Consultes, camins, grau


```
>>> mg.edges()
MultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', 'b'),
('a', 'b'), ('a', 'c'), ('a', 'e'), ('b', 'd'), ('b', '
d'), ('b', 'd'), ('d', 'c'), ('c', 'e')])
```

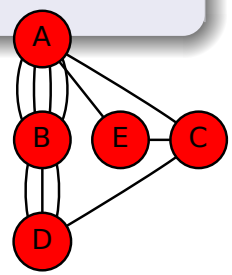
```
>>> for cami in nx.all_shortest_paths(mg, 'a','d')
:
...     print(cami)
['a', 'b', 'd']
['a', 'c', 'd']
>>> mg.degree('a')
6
```



Consultes, camins, grau


```
>>> mg.edges()
MultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', 'b'),
('a', 'b'), ('a', 'c'), ('a', 'e'), ('b', 'd'), ('b', '
d'), ('b', 'd'), ('d', 'c'), ('c', 'e')])
```

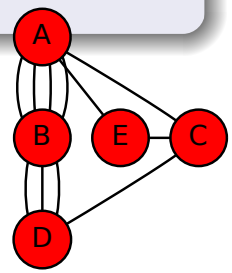
```
>>> for cami in nx.all_shortest_paths(mg, 'a','d')
:
...     print(cami)
['a', 'b', 'd']
['a', 'c', 'd']
>>> mg.degree('a')
6
```



Consultes, camins, grau

```
>>> mg.edges()
MultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', 'b'),
('a', 'b'), ('a', 'c'), ('a', 'e'), ('b', 'd'), ('b', '
d'), ('b', 'd'), ('d', 'c'), ('c', 'e')])
```

```
>>> for cami in nx.all_shortest_paths(mg, 'a','d')
:
...     print(cami)
['a', 'b', 'd']
['a', 'c', 'd']
>>> mg.degree('a')
6
```



creació multigrafs dirigits.

```
>>> import networkx as nx
>>> mdg = nx.MultiDiGraph()
>>> nx.add_path(mdg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mdg, ['a', 'e', 'c'])
>>> mdg.add_edge('a','b')
1
>>> mdg.add_edge('b','a')
0
>>> mdg.add_edge('b','a')
1
>>> mdg.add_edge('b','d')
1
>>> mdg.add_edge('b','d')
2
```

creació multigrafs dirigits.

```
>>> import networkx as nx
>>> mdg = nx.MultiDiGraph()
>>> nx.add_path(mdg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mdg, ['a', 'e', 'c'])
>>> mdg.add_edge('a','b')
1
>>> mdg.add_edge('b','a')
0
>>> mdg.add_edge('b','a')
1
>>> mdg.add_edge('b','d')
1
>>> mdg.add_edge('b','d')
2
```

creació multigrafs dirigits.

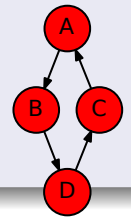
```
>>> import networkx as nx
>>> mdg = nx.MultiDiGraph()
>>> nx.add_path(mdg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mdg, ['a', 'e', 'c'])
>>> mdg.add_edge('a','b')
1
>>> mdg.add_edge('b','a')
0
>>> mdg.add_edge('b','a')
1
>>> mdg.add_edge('b','d')
1
>>> mdg.add_edge('b','d')
2
```

creació multigrafs dirigits.

```
>>> import networkx as nx
>>> mdg = nx.MultiDiGraph()
>>> nx.add_path(mdg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mdg, ['a', 'e', 'c'])
>>> mdg.add_edge('a','b')
1
>>> mdg.add_edge('b','a')
0
>>> mdg.add_edge('b','a')
1
>>> mdg.add_edge('b','d')
1
>>> mdg.add_edge('b','d')
2
```

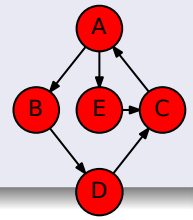
creació multigrafs dirigits.


```
>>> import networkx as nx
>>> mdg = nx.MultiDiGraph()
>>> nx.add_path(mdg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mdg, ['a', 'e', 'c'])
>>> mdg.add_edge('a','b')
1
>>> mdg.add_edge('b','a')
0
>>> mdg.add_edge('b','a')
1
>>> mdg.add_edge('b','d')
1
>>> mdg.add_edge('b','d')
2
```



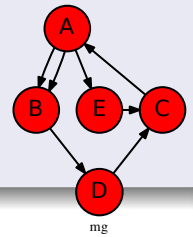
creació multigrafs dirigits.


```
>>> import networkx as nx
>>> mdg = nx.MultiDiGraph()
>>> nx.add_path(mdg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mdg, ['a', 'e', 'c'])
>>> mdg.add_edge('a','b')
1
>>> mdg.add_edge('b','a')
0
>>> mdg.add_edge('b','a')
1
>>> mdg.add_edge('b','d')
1
>>> mdg.add_edge('b','d')
2
```



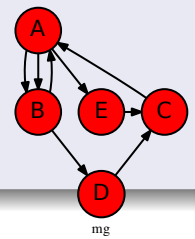
creació multigrafs dirigits.


```
>>> import networkx as nx
>>> mdg = nx.MultiDiGraph()
>>> nx.add_path(mdg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mdg, ['a', 'e', 'c'])
>>> mdg.add_edge('a','b')
1
>>> mdg.add_edge('b','a')
0
>>> mdg.add_edge('b','a')
1
>>> mdg.add_edge('b','d')
1
>>> mdg.add_edge('b','d')
2
```



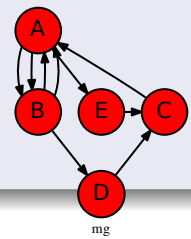
creació multigrafs dirigits.


```
>>> import networkx as nx
>>> mdg = nx.MultiDiGraph()
>>> nx.add_path(mdg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mdg, ['a', 'e', 'c'])
>>> mdg.add_edge('a','b')
1
>>> mdg.add_edge('b','a')
0
>>> mdg.add_edge('b','a')
1
>>> mdg.add_edge('b','d')
1
>>> mdg.add_edge('b','d')
2
```



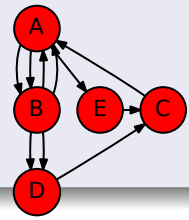
creació multigrafs dirigits.


```
>>> import networkx as nx
>>> mdg = nx.MultiDiGraph()
>>> nx.add_path(mdg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mdg, ['a', 'e', 'c'])
>>> mdg.add_edge('a','b')
1
>>> mdg.add_edge('b','a')
0
>>> mdg.add_edge('b','a')
1
>>> mdg.add_edge('b','d')
1
>>> mdg.add_edge('b','d')
2
```



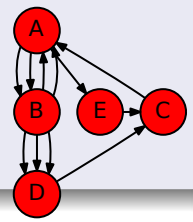
creació multigrafs dirigits.


```
>>> import networkx as nx
>>> mdg = nx.MultiDiGraph()
>>> nx.add_path(mdg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mdg, ['a', 'e', 'c'])
>>> mdg.add_edge('a','b')
1
>>> mdg.add_edge('b','a')
0
>>> mdg.add_edge('b','a')
1
>>> mdg.add_edge('b','d')
1
>>> mdg.add_edge('b','d')
2
```



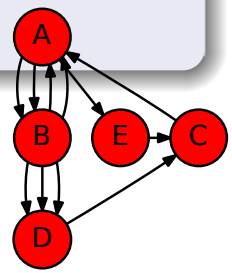
creació multigrafs dirigits.


```
>>> import networkx as nx
>>> mdg = nx.MultiDiGraph()
>>> nx.add_path(mdg, ['a', 'b', 'd', 'c', 'a'])
>>> nx.add_path(mdg, ['a', 'e', 'c'])
>>> mdg.add_edge('a','b')
1
>>> mdg.add_edge('b','a')
0
>>> mdg.add_edge('b','a')
1
>>> mdg.add_edge('b','d')
1
>>> mdg.add_edge('b','d')
2
```



consultes, camins,

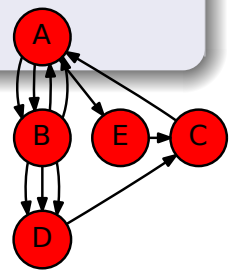

```
>>> mdg.edges()
OutMultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', 'e'
), ('b', 'd'), ('b', 'd'), ('b', 'd'), ('b', 'a'), ('b'
, 'a'), ('d', 'c'), ('c', 'a'), ('e', 'c')])
>>> for cami in nx.all_shortest_paths(mdg, 'a','d'):
...     print(cami)
['a', 'b', 'd']
>>> mdg.degree('a')
6
>>> mdg.in_degree('a')
3
>>> mdg.out_degree('a')
3
```



mdg

consultes, camins,

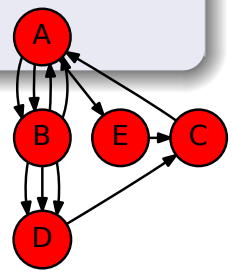

```
>>> mdg.edges()
OutMultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', 'e'
), ('b', 'd'), ('b', 'd'), ('b', 'd'), ('b', 'a'), ('b'
, 'a'), ('d', 'c'), ('c', 'a'), ('e', 'c')])
>>> for cami in nx.all_shortest_paths(mdg, 'a','d'):
...     print(cami)
['a', 'b', 'd']
>>> mdg.degree('a')
6
>>> mdg.in_degree('a')
3
>>> mdg.out_degree('a')
3
```



mdg

consultes, camins,

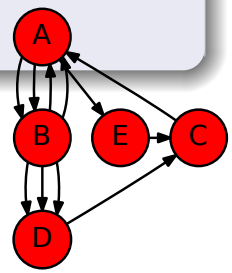
```
>>> mdg.edges()
OutMultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', 'e'
), ('b', 'd'), ('b', 'd'), ('b', 'd'), ('b', 'a'), ('b'
, 'a'), ('d', 'c'), ('c', 'a'), ('e', 'c')])
>>> for cami in nx.all_shortest_paths(mdg, 'a','d'):
...     print(cami)
['a', 'b', 'd']
>>> mdg.degree('a')
6
>>> mdg.in_degree('a')
3
>>> mdg.out_degree('a')
3
```



mdg

consultes, camins,

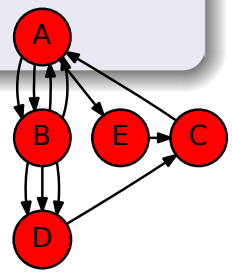

```
>>> mdg.edges()
OutMultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', 'e'
), ('b', 'd'), ('b', 'd'), ('b', 'd'), ('b', 'a'), ('b'
, 'a'), ('d', 'c'), ('c', 'a'), ('e', 'c')])
>>> for cami in nx.all_shortest_paths(mdg, 'a','d'):
...     print(cami)
['a', 'b', 'd']
>>> mdg.degree('a')
6
>>> mdg.in_degree('a')
3
>>> mdg.out_degree('a')
3
```



mdg

consultes, camins,

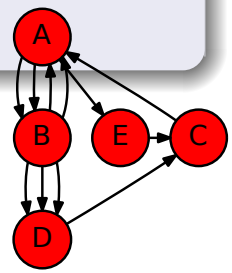

```
>>> mdg.edges()
OutMultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', 'e'
), ('b', 'd'), ('b', 'd'), ('b', 'd'), ('b', 'a'), ('b'
, 'a'), ('d', 'c'), ('c', 'a'), ('e', 'c')])
>>> for cami in nx.all_shortest_paths(mdg, 'a','d'):
...     print(cami)
['a', 'b', 'd']
>>> mdg.degree('a')
6
>>> mdg.in_degree('a')
3
>>> mdg.out_degree('a')
3
```



mdg

consultes, camins,

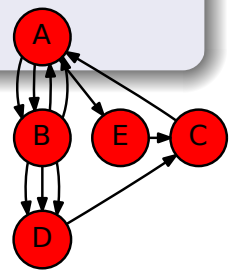
```
>>> mdg.edges()
OutMultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', 'e'
), ('b', 'd'), ('b', 'd'), ('b', 'd'), ('b', 'a'), ('b'
, 'a'), ('d', 'c'), ('c', 'a'), ('e', 'c')])
>>> for cami in nx.all_shortest_paths(mdg, 'a','d'):
...     print(cami)
['a', 'b', 'd']
>>> mdg.degree('a')
6
>>> mdg.in_degree('a')
3
>>> mdg.out_degree('a')
3
```



mdg

consultes, camins,

```
>>> mdg.edges()
OutMultiEdgeDataView([('a', 'b'), ('a', 'b'), ('a', 'e'
), ('b', 'd'), ('b', 'd'), ('b', 'd'), ('b', 'a'), ('b'
, 'a'), ('d', 'c'), ('c', 'a'), ('e', 'c')])
>>> for cami in nx.all_shortest_paths(mdg, 'a','d'):
...     print(cami)
['a', 'b', 'd']
>>> mdg.degree('a')
6
>>> mdg.in_degree('a')
3
>>> mdg.out_degree('a')
3
```



mdg