Graphs (Review).

Curs 2016
Graphs

Graph:  $G = (V, E)$ where $V$ set of vertices, $|V| = n$. $E \subset V \times V$ set of edges. $|E| = m$,

- Graphs: *directed* (digraphs) or *undirected*.
- Undirected graphs could be connected. If $G$ is connected, then $\frac{n(n-1)}{2} \geq m \geq n - 1$.
- The degree of a vertex $v$, $d(v)$ is the number of edges connected to $v$.
- A clique $K_n$ of size $n$ is a complete graph on $n$ vertices.

![Undirected Connected](image1)

![Directed Strongly connected](image2)

![Directed Not SC](image3)
Graphs

- Digraphs could be strongly connected.
- A digraph has at most \(n(n - 1)\) edges.
Density

Not clear-cut concept

A graph $G$ with $|V| = n$ is dense if $|E| = \Theta(n^2)$ and it is sparse if $|E| = o(n^2)$. 
Data structures for graphs.

Assume $G$ with $V = \{1, 2, \ldots, n\}$.

- Graphs are one of the best tool to simulate different kinds of real life situations.
- To solve graphs problem we have to examine the graph.
- We need a friendly way to store and manipulate efficiently graphs

The two main representation of graphs:

- Adjacency list
- Adjacency matrix
Adjacency lists

\[\begin{align*}
\text{a} & \to \text{b} \to \text{d} \\
\text{b} & \to \text{a} \to \text{d} \\
\text{c} & \to \text{d} \\
\text{d} & \to \text{a} \to \text{b}
\end{align*}\]
Define the $n \times n$ adjacency matrix of $G$ by

$$A[i, j] = \begin{cases} 1 & \text{if } (i, j) \in E, \\ 0 & \text{if } (i, j) \notin E. \end{cases}$$
The matrix of an undirected graphs is symmetric.
If $A$ is the matrix of $G$, then $A^2$ yield all length-2 paths between vertices.
For weighted graphs $(w_{i,j}$ on $(i,j) \in E)$, the adjacency matrix has $w_{ij}$ in position $i, j$.
The use of the adjacency matrix make possible to use matrix algebra a powerful tool in algorithmic and combinatorics.
Comparison between lists and matrices

**Space:**
- **Lists** use one list node per edge and two $2 \times 64$ bits per slot in list per edge: $128m$ bits = $\Theta(m)$.
- **Matrix** uses $n \times n$ entries, each entry 1 bit per entry: $\Theta(n^2)$ bits.
- Matrix better for dense graphs

**Time:**
- **Add an edge:** both DS are $\Theta(1)$.
- **Query if there is an edge $u$ to $v$:**
  - Matrix: $\Theta(1)$.
  - List could be $O(n)$
- **Visit all neighbors of $v$:**
  - Matrix: $\Theta(n)$
  - List: $O(|d(v)|)$
- **Remove edge:** both DS like querying an edge.
Searching a graph: Breadth First Search

1. start with vertex \( v \), visit and list all their neighbors at distance\(=1 \)
2. then all their neighbors at distance 2 from \( v \).
3. Repeat until all vertices visited

BFS use a QUEUE, (FIFO)

Problem: If unknowingly the BFS revisits a vertex, the algorithm could yield the wrong notion of distance. The solution to avoid that loop is to label each vertex we visit for first time, and ignore it when we revisit it.
Searching a graph: Depth First Search

1. From current vertex, move to another
2. Until you get stuck
3. Then backtrack till new place to explore.

DFS use a STACK, (LIFO)
Time Complexity of DFS and BFS

• DFS:
For undirected and directed graphs: $O(|V| + |E|)$
In the case of sparse graphs $T(n) = O(n)$
For the case of a dense graph $T(n) = O(n^2)$

• BFS:
For undirected and directed graphs: $O(|V| + |E|)$

Therefore, the complexity of both procedures is linear in the size of the graph.
Connected components un undirected graphs.

Undirected Connected Components
INPUT: undirected graph $G$
QUESTION: Find if $G$ is connected (if there is a path between any pair of vertices in $V(G)$.

To find connected components in $G$ apply DFS and count how many times explore is called. each time DFS calls explore on a vertex, it yields exactly the connected component to which the vertex belongs.

The problem can be solved in $O(|V| + |E|)$. 
Strongly connected components in a digraph

Every digraph is a *directed acyclic graph (dag)* of its strongly connected components.

Complexity strongly connected components: $T(n) = O(|V| + |E|)$
Graphs in life: Complex Networks

Complex network: graphs with non-trivial topological features that model ”real life systems”.

- **Small-world property** Milgram experiment (1969): Six degrees of separation
  Any pair of vertices in the network are very close distance
  People can find those short paths with local information

- **Power law degree distribution**
  20% of the nodes have 80% of the connexions Those vertices are denoted hubs
  Power Law $G$ has power law distribution with given exponent $k$ if the number of vertices with degree $d$ is $1/d^{-c}$. 
• Clustering coefficient The probability that two random nodes connected to \( v \) are connected among them. Complex networks have high clustering coefficient.
Examples of Complex Networks

- **Internet**
  Undirected graph with $12 \times 10^9$ vertices (2013)

- **The WWW**
  Directed graph with $15 \times 10^9$ vertices (2015)
Examples of Complex Networks

- Biological network

- Social nets: Facebook
For more information on the topic