Recorrido sobre grafos

Edelmira Pasarella

Marzo, 2008

1 Recorrido en profundidad (Depth first search, DFS)

```
void dfs(Grafo G){
{pre: G = ⟨V,E⟩ no dirigido }
{post: Todos los vertices de G han sido visitados_DFS }
        Conjunto[Vertice] visitados = ∅;
        while (V\visitados ≠ ∅) {
                v = escoger(V\visitados);
                dfs(G,v, visitados);
        }
}
```

```
void dfs(Grafo G, Vertice v,Conjunto[Vertice]& visitados){
{pre: G = ⟨V,E⟩ no dirigido ∧ v ∈ V ∧ v ∉ visitados_DFS}
{post: v ∈ visitados_DFS ∧ ∀ w ∈ Ady(G,v), w ∈ visitados_DFS}
        visitados = visitados + v;
        for each w∈ Ady(G,v) {
            if (w ∉ visitados)
                    dfs(G,w, visitados);
        }
}
```

Coste Sea G = ⟨V,E⟩ con |V|=n y |E|=e:
Matriz de adyacencia: $\Theta(n^2)$
Listas de adyacencia: $\Theta(n + e)$

El algoritmo de recorrido DFS es útil para muchas aplicaciones. Si consideramos la siguiente plantilla, los rectángulos indican los puntos del algoritmo donde suelen colocarse instrucciones que lo particularizan.

```
void dfs(Grafo G){
{pre: G = ⟨V,E⟩ no dirigido }
{post: Todos los vertices de G han sido visitados_DFS }
        Conjunto[Vertice] visitados = ∅;
        [          ]
        while (V\visitados ≠ ∅}) {
                v = escoger(V\visitados);
                [          ]
                dfs(G,v, visitados, [      ]);
        }
}
```

```
void dfs(Grafo G, Vertice v,Conjunto[Vertice]& visitados, [      ]){
{pre: G = ⟨V,E⟩ no dirigido ∧ v ∈ V ∧ v ∉ visitados_DFS}
{post: v ∈ visitados_DFS ∧ ∀ w ∈ Ady(G,v), w ∈ visitados_DFS}
        visitados = visitados + v;
        [          ]
        for each w∈ Ady(G,v) {
            if (w ∉ visitados)
                    dfs(G,w, visitados, [      ]);
        }
        [          ]
}
```

1.1 Ejemplo: Número de componentes conexas

```
int dfs_ncc(Grafo G){
{pre: G = ⟨V,E⟩ no dirigido }
{post: Todos los vertices de G han sido visitados_DFS y dfs_ncc devuelve
       el numero de componentes conexas de G}
       Conjunto[Vertice] visitados = ∅;
       int ncc = 0;
       while (V\visitados ≠ ∅}) {
             v = escoger(V\visitados);
             ncc++;
             dfs_ncc(G,w, visitados, _____);
       }
       return ncc; }

void dfs_ncc(Grafo G, Vertice v,Conjunto[Vertice]& visitados, _____){
{pre: G = ⟨V,E⟩ no dirigido ∧ v ∈ V ∧ v ∉ visitados_DFS}
{post: v ∈ visitados_DFS ∧ ∀ w ∈ Ady(G,v), w ∈ visitados_DFS}
       visitados = visitados + v;

       _____
       for each w∈ Ady(G,v) {
           if (w ∉ visitados)
                 dfs(G,w, visitados, _____);
       }

       _____
}
```

1.2 Ejemplo: Numerar vértices

```
void dfs_num(Grafo G){
{pre: G = ⟨V,E⟩ no dirigido }
{post: Todos los vertices de G han sido visitados_DFS }
        Conjunto[Vertice] visitados = ∅;
        for each v∈ V { v.dfs_num = 0; v.inv_ dfs_num = 0;}
        int num_dfs = 0; int inv_dfs = 0;
        while (V\visitados ≠ ∅}) {
                v = escoger(V\visitados);
                dfs_num(G,v, visitados, num_dfs, inv_dfs );
        }
}

void dfs_num(Grafo G, Vertice v,Conjunto[Vertice]& visitados,
                int& num, int& i_num ){
{pre: G = ⟨V,E⟩ no dirigido ∧ v ∈ V ∧ v ∉ visitados_DFS}
{post: v ∈ visitados_DFS ∧ ∀ w ∈ Ady(G,v), w ∈ visitados_DFS}
        visitados = visitados + v;
        num++; v.dfs_num = num;
        for each w∈ Ady(G,v) {
            if (w ∉ visitados)
                    dfs_num(G,w, visitados, num, i_num);
        }
        i_num++; v.inv_ dfs_num = i_num;
}
```

1.3 Ejemplo: Ordenamiento topológico inverso

```
Lista[Vertice] dfs(Grafo G){
{pre: G = ⟨V,E⟩ es un DAG }
{post: Todos los vertices de G han sido visitados_DFS y en L se listan
       los vertices en orden topologico inverso }
       Conjunto[Vertice] visitados = ∅;
       L = ⟨⟩
       while (V\visitados ≠ ∅}) {
             v = escoger(V\visitados);
             ord_top_inv(G,v, visitados,L);
       }
       return L;
}

void ord_top_inv(Grafo G, Vertice v,Conjunto[Vertice]& visitados,
                 Lista[Vertice]&L ){
{pre: G = ⟨V,E⟩ no dirigido ∧ v ∈ V ∧ v ∉ visitados_DFS}
{post: v ∈ visitados_DFS ∧ ∀ w ∈ Ady(G,v), w ∈ visitados_DFS
       y en L se listan los vertices que han sido visitados
       en orden topologico inverso }
       visitados = visitados + v;
       for each w∈ Suc(G,v) {
           if (w ∉ visitados)
                 ord_top_inv(G,w, visitados,L);
       }
       append(L,v)
}
```

2 Recorrido en anchura (Breadth first search, BFS)

```
void bfs(Grafo G){
{pre: G = ⟨V,E⟩ no dirigido }
{post: Todos los vertices de G han sido visitados }
        Conjunto[Vertice] visitados = ∅;
        while (V\visitados ≠ ∅) {
                v = escoger(V\visitados);
                bfs(G,v, visitados);
        }
}
```

```
void bfs(Grafo G, Vertice v,Conjunto[Vertice]& visitados){
{pre: G = ⟨V,E⟩ no dirigido ∧ v ∈ V ∧ v ∉ visitados_BFS }
{post: v ∈ visitados_BFS ∧ ∀ w ∈ Ady(G,v), w ∈ visitados_BFS }
        Cola[Vertice] c = ⟨⟩;
        encolar(c,v);
        while not(vacia(c)) {
                w = primero(c);
                desencolar(c);
                if (w ∉ visitados) {
                        visitados = visitados + w;
                        for each u ∈ Ady(G,w) {
                            if (u ∉ visitados)
                                encolar(c,u);
                        }
                }
        }
}
```

Coste Sea G = ⟨V,E⟩ con |V|=n y |E|=e:
Matriz de adyacencia: $\Theta(n^2)$
Listas de adyacencia: $\Theta(n + e)$