Grafos

19 de diciembre de 2013

Grafo

Un grafo es un conjunto, no vacío, de objetos llamados vértices (o nodos) y una selección de pares de vértices, llamados aristas (edges en inglés) que pueden ser orientados (dirigidos) o no. (wikipedia).
Un grafo G se define como una tupla (V, E), con s V un conjunto de vértices o nodos, y un conjunto $E \subseteq V \times V$. . Un grafo puede ser dirigido, es decir, $(a, b) \in E$ no es lo mismo que $(b, a) \in E$ o no dirigido, cuando $(a, b) = (b, a)$.

Grafos (cont.)

Los grafos pueden ser:

- Dirigidos versus no dirigidos
- Con pesos versus sin pesos
- Simples versus no simples
- Topológicos versus autocontenidos
- Cíclicos versus no cíclicos
- Densos versus no densos
- Etiquetado versus no etiquetados

Grafos: representaciones

Los grafos pueden ser:

- Matriz de adyacencia
- Lista de adyacencia

Representación (cont.)

Comparación	Sugerencia
Rápida para ver si (x, y) está en el grafo	Matriz de adyacencia
Rápida para encontrar sel grado de un vértice	Lista de adyacencia
Usa menos memoria en grafos pequeños	Lista de adyacencia
Usa menos memoria en grafos grandes	Matriz de adyacencia (leve)
Rápida para insertar o eliminar arcos	Matriz de adyacencia
Rápida para recorrer un grafo	Lista de adyacencia
Mejor para la mayoría de problemas	Lista de adyacencia

Representación (cont.)

- Decimos que dos vértices v_1 y v_2 son adyacentes si $(v_1, v_2) \in E$. En este caso decimos que hay una arista entre v_1 y v_2.
- Un camino de largo n entre v y w es una lista de $n+1$ vértices $v = v_0, v_1, \ldots, v_n = w$ tales que para $0 \leq i < n$ los vértices v_i y $v_i + 1$ son adyacentes. La distancia entre dos vértices v y w se define como el menor número n tal que existe un camino entre v y w de largo n. Si no existe ningún camino entre v y w decimos que la distancia entre v y w es ∞.
- Si la distancia entre v y w es n, entonces un camino entre v y w de largo n se llama camino mínimo.
- Un problema muy frecuente es tener que encontrar la distancia entre dos vértices, este problema se resuelve encontrando un camino mínimo entre los vértices.
- Este problema es uno de los problemas más comunes de la teoría de grafos y se puede resolver de varias maneras, una de ellas es el algoritmo llamado BFS (Breadth First Search).

Recorridos

- Búsqueda en profundidad (Depth First Search)
- Búsqueda en anchura (Breath First Search)

Los recorridos DFS o BFS son útiles para varias cosas, entre ellas, encontrar los componentes conectados de un grafo, colorear o etiquetar componentes, spanning tree, grafos bipartitos, encontrar puntos de articulación o quiebres, componentes fuertemente conectados

BFS

```
1  vector<int> BFS(vector<vector<int> > &lista, int nodoInicial){
2      int n = lista.size(),t;
3      queue<int> cola;
4      vector<int> distancias(n,n);
5      cola.push(nodoInicial);
6      distancias[nodoInicial] = 0;
7      while(!cola.empty()){
8          t = cola.front();
9          cola.pop();
10         for(int i=0;i<lista[t].size();i++){
11             if(distancias[lista[t][i]]!=n){
12                 distancias[lista[t][i]] = distancias[t]+1;
13                 cola.push(lista[t][i]);
14             }
15         }
16     }
17     return distancias;
18 }
```

Extraído del material del Training Camp ICPC 2012

BFS (cont.)

- *lista* es una lista de nodos, donde cada elemento de la lista es una lista de nodos adyacentes.
- Se usa una cola para ir almacenando los nodos adyacentes en el orden que deben ser recorridos una vez se revisen los nodos adyacentes del nodo actual, y así sucesivamente.

DFS: Usado para determinar si un grafo es un árbol

```
1  bool esArbol(vector<vector<int> > &lista, int t, vector<bool> &toc, int
       padre)
2  {
3      toc[t] = true;
4      for(int i=0;i<lista[t].size();i++)
5      {
6          if((toc[lista[t][i]]==true&&lista[t][i]!=padre))
7              return false;
8          if(toc[lista[t][i]]==false)
9          if(esArbol(lista,lista[t][i],toc,t)==false))
10             return false;
11     }
12     return true;
13 }
```

Extraído del material del Training Camp ICPC 2012

DFS (cont.)

- *t* es el nodo en el que estamos parados, *toc* guarda los nodos que ya tocamos, y *padre* es el nodo desde el que venimos.
- *toc* comienza en *false* y el tamaño de *toc* es el mismo que el de lista.
- Estamos asumiendo que el grafo es conexo, ya que si no lo es, la función *esArbol* puede devolver *true* sin ser un árbol.
- La forma de chequear si el grafo es conexo es chequear que hayamos tocado todos los nodos, es decir, que todas las posiciones de *toc* terminen en true.

Problemas

- IOI 2006: flood
- IOI 2006: robot
- UVa 11902 - Dominator