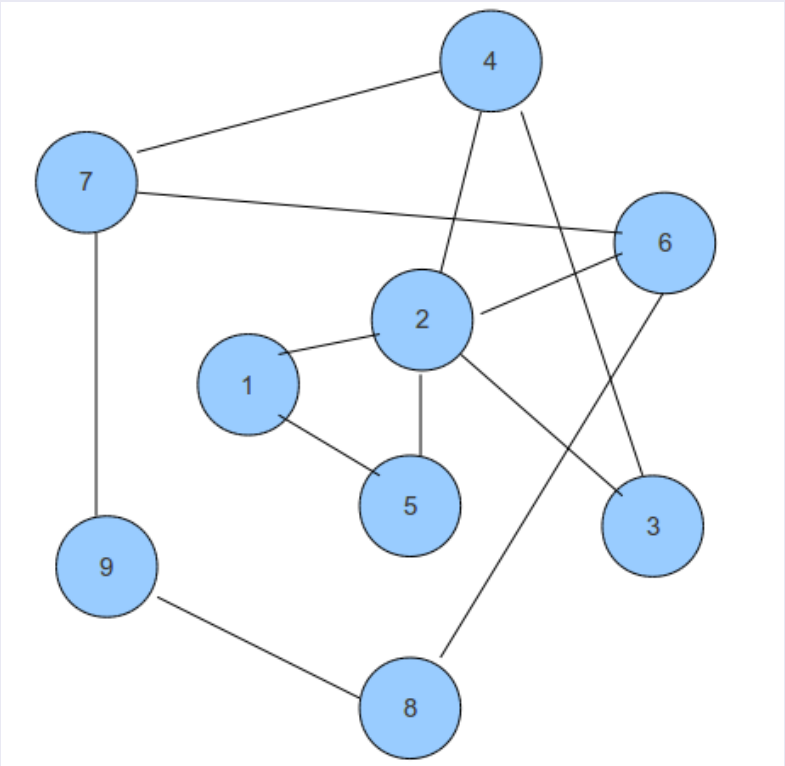
Grafos

¿Qué es un grafo?



Definiciones posibles

Wikipedia

"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."

Real Academia Española

"Diagrama que representa mediante puntos y líneas las relaciones entre pares de elementos y que se usa para resolver problemas lógicos, topológicos y de cálculo combinatorio."

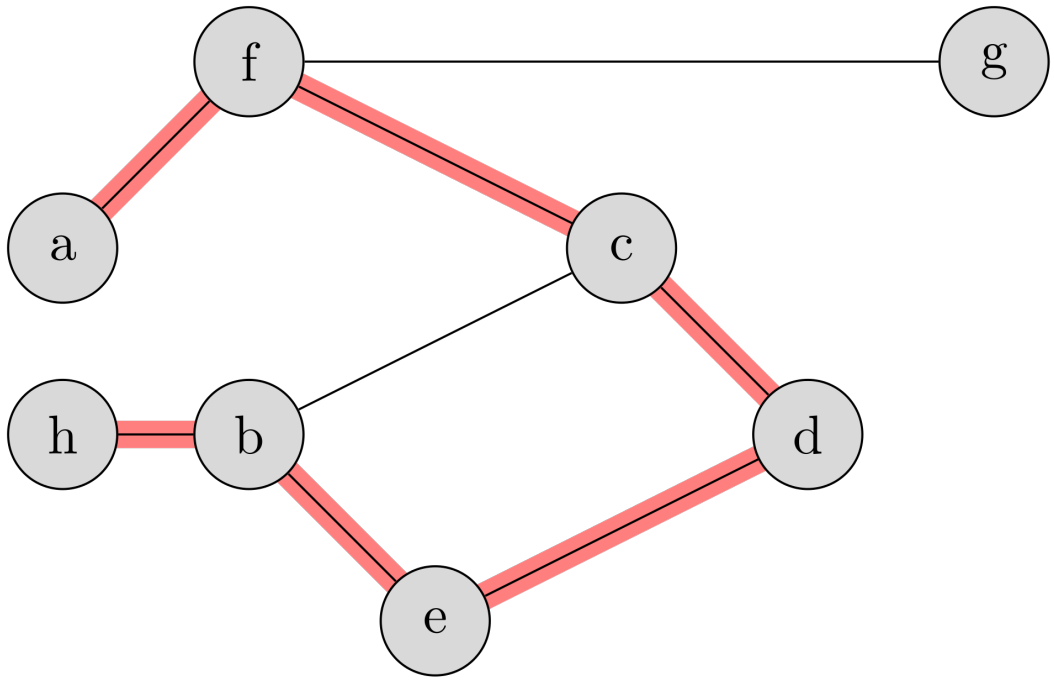
La Posta

"Un grafo es un conjunto de puntos y líneas que unen pares de esos puntitos"

Caminos

Definicion

Un camino entre dos nodos v y w es una lista de $n+1$ vértices $v = v_0, v_1, ..., v_n = w$ tales que para $0 \leq i < n$ los vértices v_i y v_{i+1} estan conectados por una arista.



longitud de un camino

Camino simple

Un camino entre v y w, se dice un camino simple, si se cumple que ninguno de los vértices que visita, es visitado más de una vez.

Longitud de un camino

Diremos que un camino entre v y w es de longitud n, si ese camino es una lista de vértices de longitud n.

Distancia

Distancia entre dos vértices

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 ∞.

Componentes conexas

Conexo

Un grafo se dice conexo si para todo par de vertices hay un camino que los conecta

Componentes conexas

Son los subgrafos conexos maximales. Es decir son aquellos vertices que si me quedo solo con ellos y las aristas entre ellos forman un grafo conexo y además no puedo añadir ningun vertice de afuera para hacerlo conexo y mas grande.

Arboles

Definición 1

Un grafo es un árbol si es conexo y no tiene ciclos

Definición 2

Un grafo es un árbol si para todo par de nodos existe un ÚNICO camino que los conecta.

Definición 3

Un grafo de n vertices es un arbol si es conexo y tiene $n - 1$ aristas.

Formas de representar un Grafo

Matriz de adyacencia

La matriz de adyacencia es una matriz de $n \times n$ donde n es la cantidad de nodos del grafo, que en la posición (i, j) tiene un 1 (o true) si hay una arista entre los nodos i y j, o 0 (o false) en caso contrario.

Formas de representar un Grafo

Matriz de adyacencia

La matriz de adyacencia es una matriz de $n \times n$ donde n es la cantidad de nodos del grafo, que en la posición (i, j) tiene un 1 (o true) si hay una arista entre los nodos i y j, o 0 (o false) en caso contrario.

Lista de adyacencia

La lista de adyacencia es un vector de vectores de enteros, que en el i-ésimo vector tiene el número j si hay una arista entre los nodos i y j. Esta representación es la que usaremos para los algoritmos que recorren un grafo (DFS Y BFS por ejemplo).

A partir de ahora en nuestras implementaciones n será la cantidad de nodos y m la cantidad de ejes.

Implementaciones de lista de adyacencia

```
#include <iostream>
#include <vector>
using namespace std;

vector<int> graph[10000];
\\ voy a tener n<=10000
int n,m;

int main(){
  cin>>n>>m;
  for(int i=0;i<m;i++){
    int a,b;
    cin>>a>>b;
    graph[a].push_back(b);
    graph[b].push_back(a);
  }
}
```

Implementaciones de lista de adyacencia

```cpp
#include <iostream>
#include <vector>
using namespace std;

vector<int> graph[10000];
\\ voy a tener n<=10000
int n,m;

int main(){
  cin>>n>>m;
  for(int i=0;i<m;i++){
    int a,b;
    cin>>a>>b;
    graph[a].push_back(b);
    graph[b].push_back(a);
  }
}
```

```cpp
#include <iostream>
#include <vector>
using namespace std;

vector<vector<int > > graph;
int n,m;

int main(){
  cin>>n>>m;
  graph.resize(n);
  for(int i=0;i<m;i++){
    int a,b;
    cin>>a>>b;
    graph[a].push_back(b);
    graph[b].push_back(a);
  }
}
```

Implementaciones de matriz de adyacencia

```
1  #include <iostream>
   using namespace std;
3
   int main(){
5     int n,m;
      cin>>n>>m;
7     int graph[n][n] = { 0 };
      for(int i=0;i<m;i++){
9        int a,b;
         cin>>a>>b;
11       graph[a][b] = 1;
         graph[b][a] = 1;
13    }
   }
15
```

Formas de recorrer un Grafo (BFS)

Formas de recorrer un Grafo (BFS)

```
1  void BFS(int node){
     queue<int> q;
3    q.push(node);
     encolado[node] = true;
5    while(q.size()){
       int current = q.front();
7      q.pop();
       cout<<"Estoy viendo el nodo: "<<current<<endl;
9      for(int i=0;i<graph[current].size();i++){
         int t = graph[current][i];
11       if(!encolado[t]){
           q.push(t);
13         encolado[t] = true;
         }
15     }
     }
17 }
```

Formas de recorrer un Grafo (DFS)

```
1  void DFS(int node){
     stack<int> s;
3    s.push(node);
     encolado[node] = true;
5    while(s.size()){
       int current = s.top();
7      s.pop();
       cout<<"Estoy viendo el nodo: "<<current<<endl;
9      for(int i=0;i<graph[current].size();i++){
         int t = graph[current][i];
11       if(!encolado[t]){
           s.push(t);
13         encolado[t] = true;
         }
15     }
     }
17 }
```

DFS recursivo

```
1  #include <iostream>
   #include <vector>
3  using namespace std;
   vector<vector<int > > graph;
5  vector<bool> visit;
   int n,m;
7
   void DFS(int node){
9    visit[node] = true;
     cout<<"estoy viendo el nodo: "<<node<<endl;
11   for(int i=0;i<graph[node].size();i++){
       int current = graph[node][i];
13     if(!visit[current]) DFS(current);
     }
15 }

17
```

Usos comúnes de DFS y BFS

- Ambos permiten calcular las componentes conexas de un grafo. (Voy marcando cuales son los nodos que visite, y los que quedan marcados son los que están en las misma componente que el nodo inicial)
- Se puede usar para calcular caminos entre dos nodos.
- El BFS se puede usar para calcular las distancias mínimas y caminos mínimos en un grafo sin pesos.
- El DFS se puede usar para encontrar ciclos.
- Muchos usos más, algunos los veremos en proximas clases.

Encontrar ciclos en grafos no dirigidos

- Corro un DFS guardandome para cada nodo su "padre" en el dfs. (guardo quien me llama en la recursión en un arreglo).
- Si en algún momento encuentro un nodo ya visitado, si este no es mi padre, es porque hay un ciclo.
- Veo la lista de padres de cada uno de los nodos (el que estoy actualmente y el que estoy viendo que ya esta visitado), y encuentro el primer nodo en común. Todos los nodos desde cada lado hasta el nodo en común son parte del ciclo.

¿Qué pasaría en grafos no dirigidos?

Diámetro en un árbol

Ver problema acá: `http://www.spoj.com/problems/PT07Z/`

Definición

Es la máxima distancia entre un par de nodos en un árbol. (Existe tal distancia porque no hay ciclos)

¿Cómo la calculo usando DFS o BFS?

Solución

Solución

- Hago un BFS desde un nodo arbitrario v para encontrar un nodo u que esté a una distancia máxima desde donde empecé.

Solución

- Hago un BFS desde un nodo arbitrario v para encontrar un nodo u que esté a una distancia máxima desde donde empecé.
- Hago un nuevo BFS desde el nodo u para encontrar un nodo w a distancia máxima del nodo u.

Solución

- Hago un BFS desde un nodo arbitrario v para encontrar un nodo u que esté a una distancia máxima desde donde empecé.
- Hago un nuevo BFS desde el nodo u para encontrar un nodo w a distancia máxima del nodo u.
- La distancia $d(u, w)$ es el diámetro del árbol.

Solución

- Hago un BFS desde un nodo arbitrario v para encontrar un nodo u que esté a una distancia máxima desde donde empecé.
- Hago un nuevo BFS desde el nodo u para encontrar un nodo w a distancia máxima del nodo u.
- La distancia $d(u, w)$ es el diámetro del árbol.

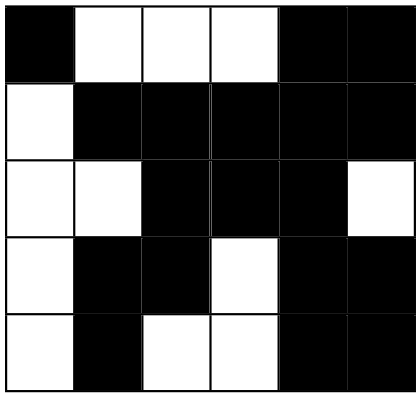
NOTA: Como entre cada par de nodos hay un único camino en un árbol, se puede usar DFS en vez de BFS para calcular distancias!! el código así es más fácil.

Código de la solución (con DFS)

Ver aquí: `https://pastebin.com/Xu0VX8SF`

Distancias mínimas con BFS (en un tablero)

Problema: `http://www.spoj.com/problems/BITMAP/`
Tengo un tablero $n \times m$ con casillas blancas y negras.



Para cada casilla, quiero encontrar la mínima distancia a una casilla blanca. Donde la distancia es en el grafo donde los nodos son las casillas, y los vecinos son los casilleros arriba, abajo, izquierda y derecha de la casilla dada.

Solución

- (Intento de solución) Puedo correr un BFS desde cada nodo negro y ver cual es el nodo blanco con distancia mínima. PROBLEMA: Debería hacer n^2 BFS, y se me va en tiempo.

Solución

- (Intento de solución) Puedo correr un BFS desde cada nodo negro y ver cual es el nodo blanco con distancia mínima. PROBLEMA: Debería hacer n^2 BFS, y se me va en tiempo.
- Debo correr un BFS simultaneo desde todos los nodos blancos. Eso lo puedo hacer agregando un nodo imaginario que se conecta con todos los nodos blancos, y calculo distancias desde ahí.

Código de la solución

Ver aquí:`https://pastebin.com/LkWjyixi`

Problemas para practicar

- http://www.spoj.com/problems/TDBFS/
- http://www.spoj.com/problems/MAKEMAZE/
- http://codeforces.com/problemset/problem/377/A
- http://codeforces.com/problemset/problem/103/B
- http://codeforces.com/problemset/problem/115/A
- http://codeforces.com/contest/791/problem/B