Esteban Foronda Sierra
Santiago Vanegas Gil

Introducción a grafos

¿Qué es un Grafo?

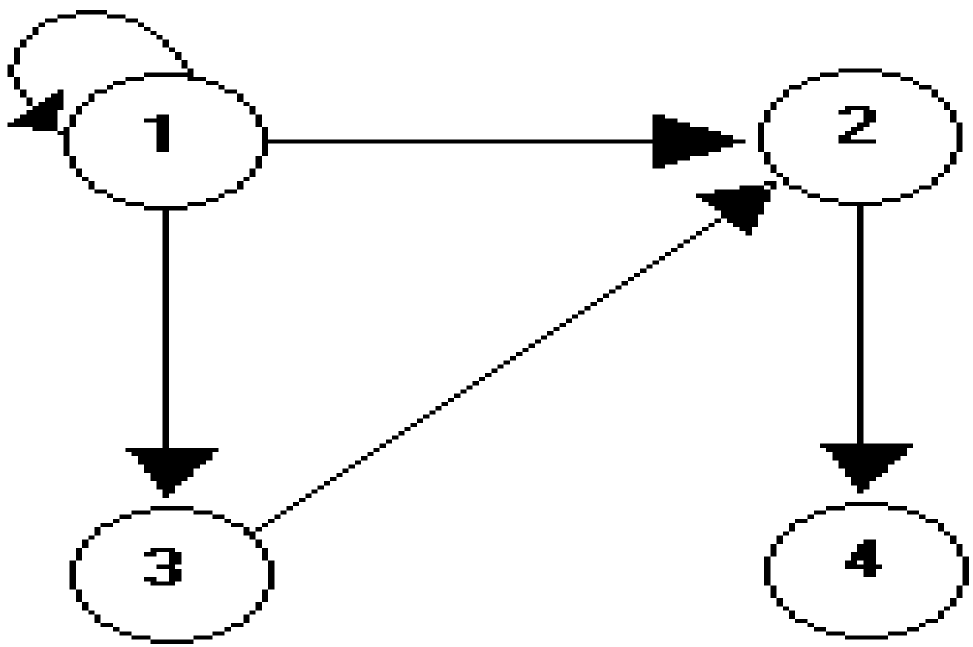
- Un grafo G es un par (V; E) donde V es un conjunto finito de nodos (vértices) y E es un conjunto de parejas ordenadas donde cada elemento es un elemento de V y es llamado conjunto de aristas (edges).

Clases de grafos

- Grafo dirigido
- Grafo no dirigido

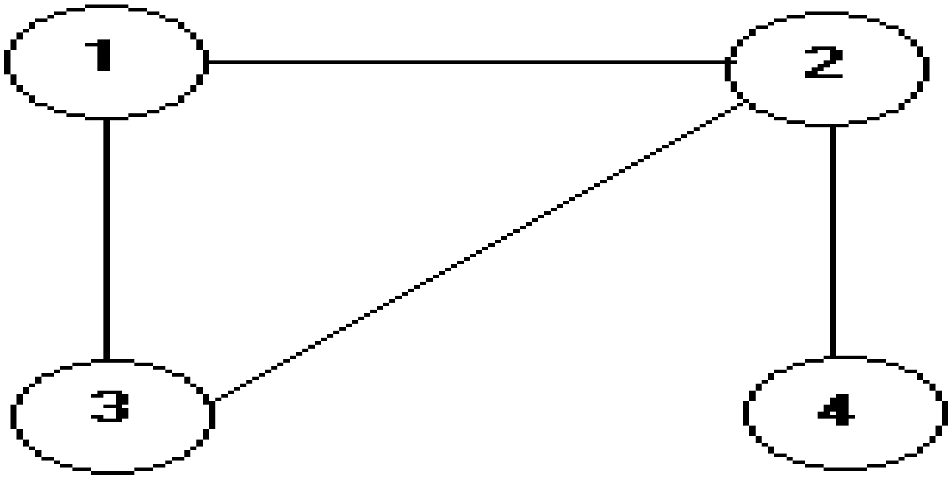
Grafo dirigido

- V = (1; 2; 3; 4;)
- E = (1; 1);(1; 2);(1; 3);(3; 2);(2;4)
- Tiene dirección solo se puede ir de 1 a 2.



Grafo no dirigido

- V = (1; 2; 3; 4;)
- E = (1; 1);(1; 2);(1; 3);(3; 2);(2;4)
- No tiene dirección se puede ir de 1 a 2 o de
- 2 a 1 por el mismo camino

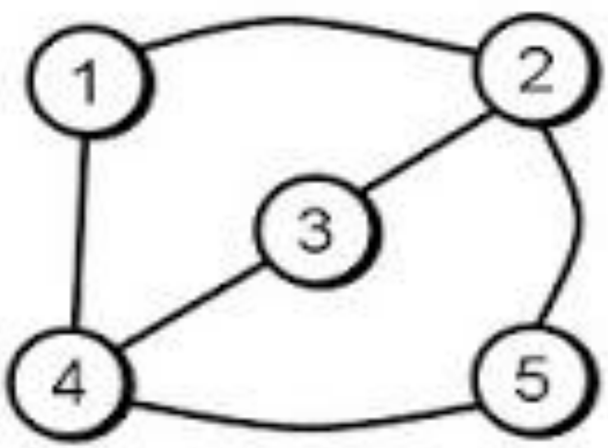


¿Cómo se representa un grafo?

- Matriz de adyacencia
- Lista de adyacencia

Matriz de adyacencia

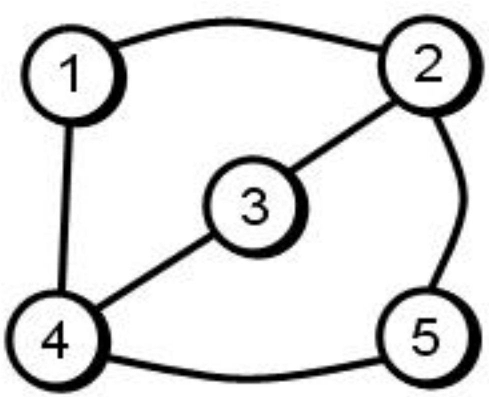
- La representación de un grafo G = (V; E) como matriz de adyacencia consiste en una matriz M de tamaño |V|.|V|
- La matriz contiene 1 si ***u*** es adyacente a ***v*** y 0 si no lo es

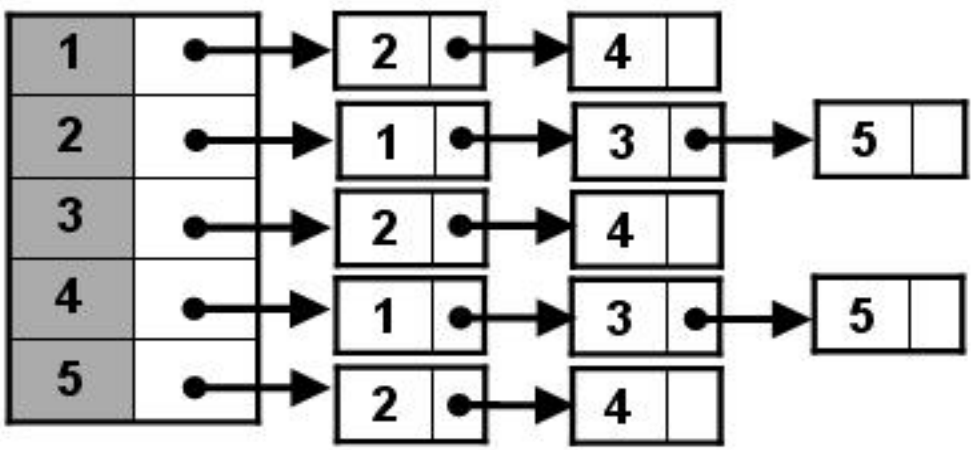


M	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	0	1
3	0	1	0	1	0
4	1	0	1	0	1
5	0	1	0	1	0

Lista de adyacencia

- La representación de un grafo G = (V; E) como lista adyacencia consiste en un arreglo Adj de |V| vectores.
- Adj[u] contiene una lista (vector) con todos
- En los grafos no dirigidos, dada la arista (u; v) se agregaría v a Adj[u] y u a Adj[v].





Implementación Matriz de adyacencia

- http://pastie.org/8955455

Implementación Lista de adyacencia

- http://pastie.org/8955463

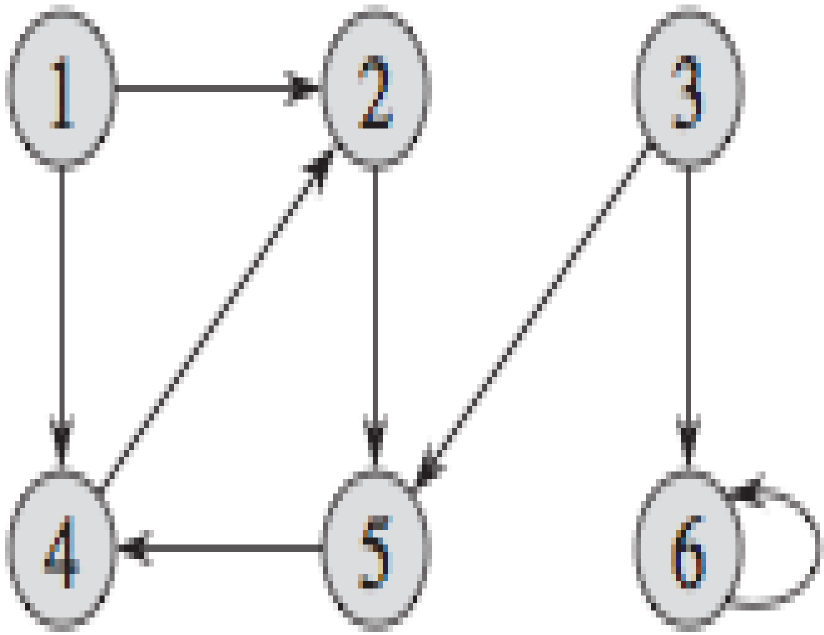
Lista vs Matriz

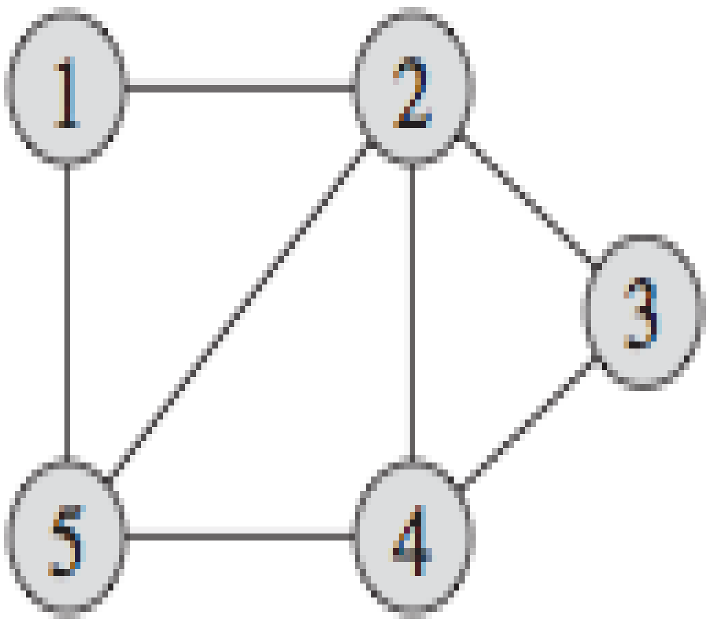
- La representación como lista de adyacencia es mas común en grafos dispersos, cuando el numero de aristas es pequeño (|E| es mucho menor que |V|.|V|)
- La representación como matriz de adyacencia es común en grafos densos, cuando el numero de aristas es grande (|E| es cercano a |V|)
- Verificar si dos nodos están conectados en un la matriz adyacencia es mas fácil que hacerlo en la adyacencia.
- La lista consume menos memoria que una matriz en grafos dispersos

BFS: Breadth-First Search

- Algoritmo para recorrer o buscar elementos en un grafo.
- Se comienza desde un nodo y se exploran todos los vecinos de este nodo.
- Luego, para cada uno de los vecinos, se exploran sus respectivos vecinos(que no se hayan visto antes).
- Se continúa de esta manera hasta que se haya recorrido todo el grafo

Ejemplos





Algoritmo

- http://pastie.org/9513307

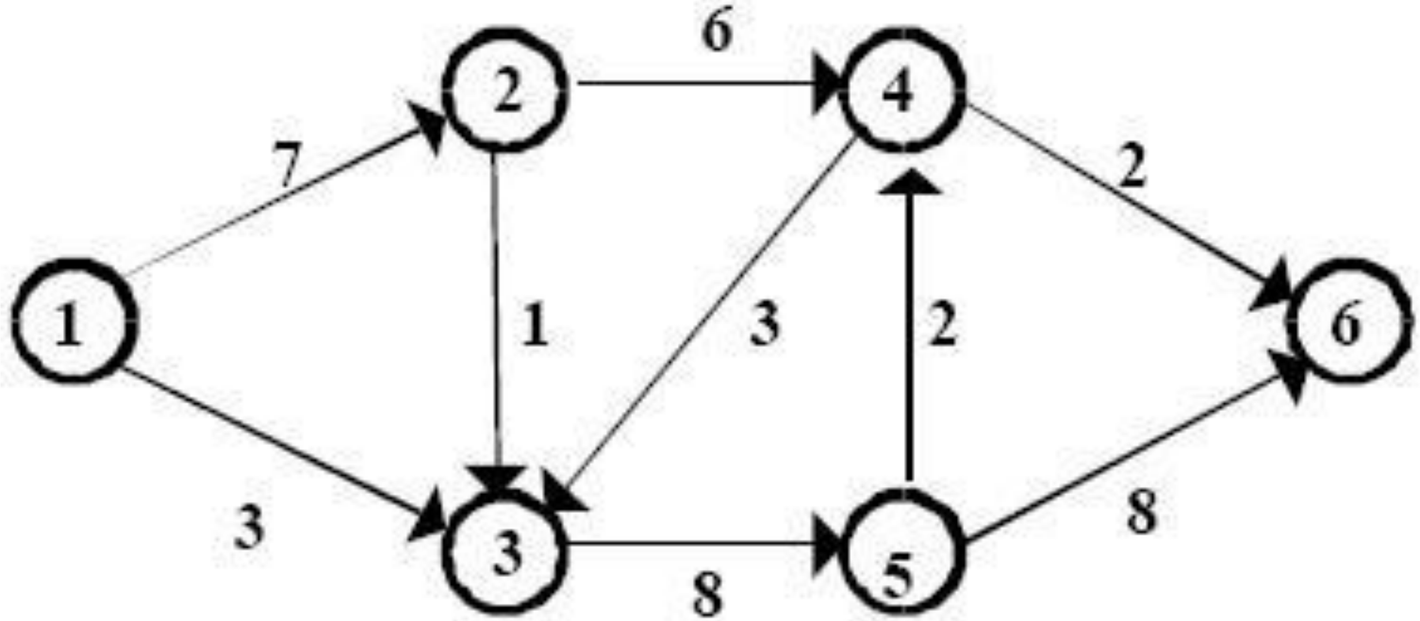
Aplicaciones

- Buscar o recorrer elementos en un grafo
- Hallar mínimo numero de aristas para llegar de la fuente a cualquier nodo.
- Hallar los nodos alcanzables desde la fuente (Ver si existe un camino de la fuente a cualquier nodo).

Grafos con peso

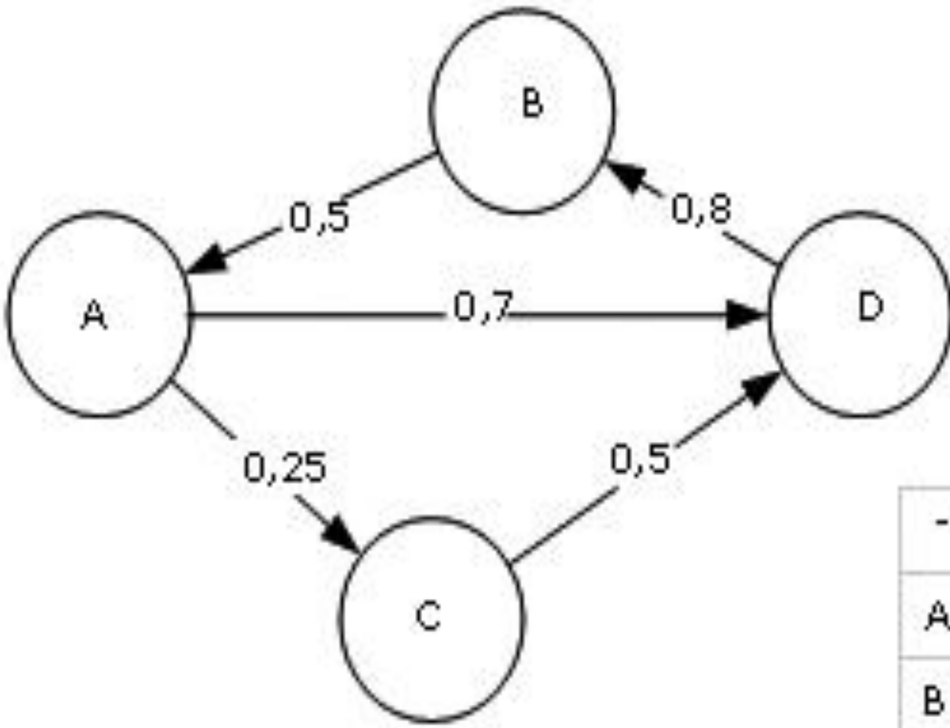
- Un grafo con pesos (weighted graph) es un grafo cuyas aristas tienen asociado un peso. Si el peso de la arista (u; v) es ***w*** entonces ir de ***u*** a ***v*** tiene un costo ***w***.

Grafos con peso



Representación Matriz

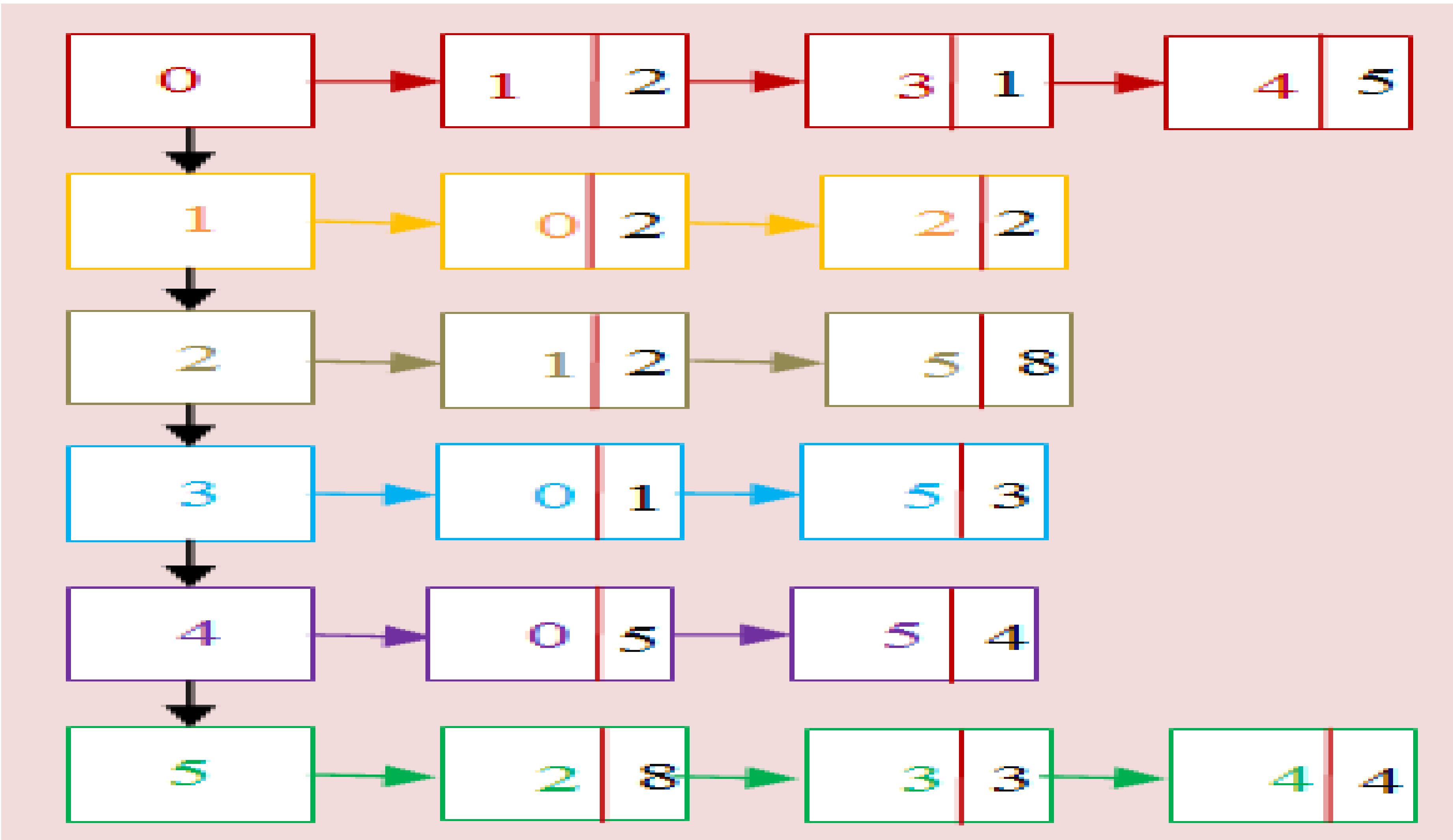
- En la representación como matriz, en lugar de almacenar un 1 en la posición u, v si los nodos u y v están conectados, almacenar el peso con el que están conectados el nodo u y el nodo v. Si los dos nodos no están conectados almacena -INFINITO



-	A	B	C	D
A	0	-	0,25	0,7
B	0,5	0	0	0
C	0	0	0	0,5
D	0	0,8	0	0

Representación Lista

- En la representación como lista de adyacencia, si el nodo ***u*** esta conectado con el ***v*** con un peso ***w***, en g[u] se almacena la pareja (v; w).
- Para esto es útil el tipo de dato pair <int, int> que es una pareja de enteros.
- El grafo podra ser vector < pair <int, int> > g[MAXN].



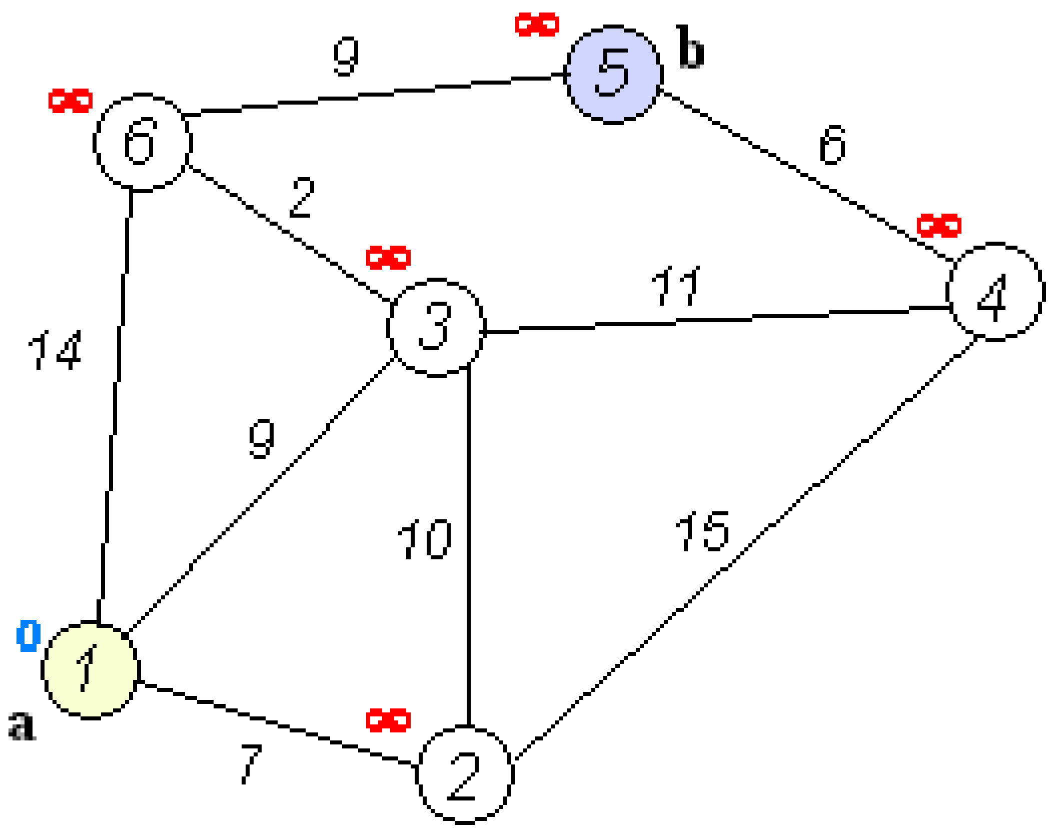
0
1 | 2
3 | 1
4 | 5
1
0 | 2
2 | 2
2
1 | 2
5 | 8
3
0 | 1
5 | 3
4
0 | 5
5 | 4
5
2 | 8
3 | 3
4 | 4

DIJKSTRA

- Dado un grafo G = (V; E), el algoritmo de Dijkstra halla el camino mas corto desde una fuente ***s*** a todos los ***v*** cuando los pesos de las aristas son **no negativos**.

- Este es el algoritmo **mas rápido** conocido para problema de SSP para grafos arbitrarios con pesos **no negativos**.

Funcionamiento



Pseudo- Codigo

- Sean ***s*** el nodo fuente d[v] la distancia del nodo s al nodo ***v***
- p[v] el nodo predecesor a ***v*** en el camino mas corto de ***s*** a ***v***.
- --
- 1 Hacer d[s] = 0 y d[v] = INF para todos los demás nodos
- 2 Hacer p[v] = -1 para todos los nodos
- 3 Agregar a la lista de nodos pendientes el nodo ***s*** con distancia 0
- 4 Mientras que haya nodos en la lista de pendientes
- 5 Extraer el nodo con la menor distancia (llamémoslo cur)
- 6 Recorrer cada vecino de cur (llamémoslo next)
- 7 Sea w_extra el peso de la arista de cur a next
- 8 Si d[next] > d[cur] + w_extra
- 9 d[next] = d[cur] + w_extra
- 10 p[next] = cur
- 11 Agregar next con d[next] a la lista de nodos pendientes

Implementación

- http://pastie.org/8955588

Problemas

- http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=870

- http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=1927