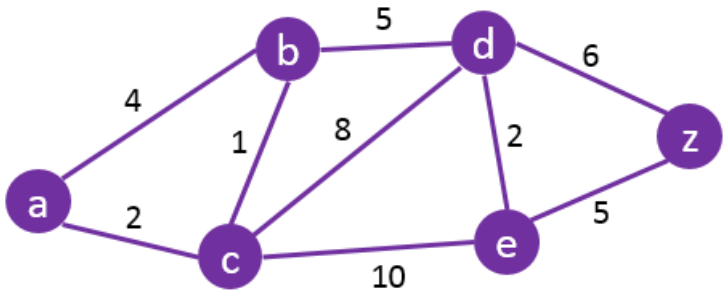
Caminos mínimos: Dijkstra

Libardo Jose Navarro Pedrozo

Descripción

El algoritmo de Dijkstra es un algoritmo clásico utilizado en teoría de grafos y en campos como la ingeniería de redes y la optimización de rutas. Su objetivo principal es encontrar el camino más corto desde un nodo de inicio hacia todos los demás nodos en un grafo con pesos.



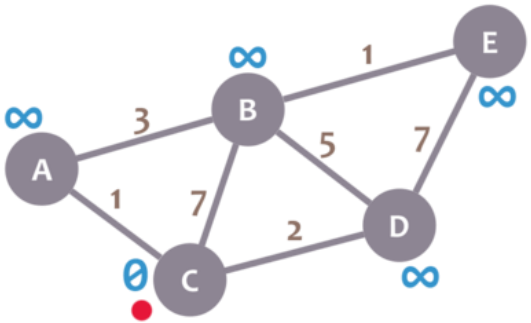
Dijkstra's Algorithm

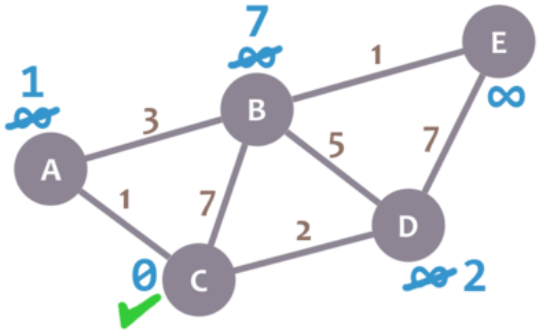
What is the shortest path to travel from A to Z?

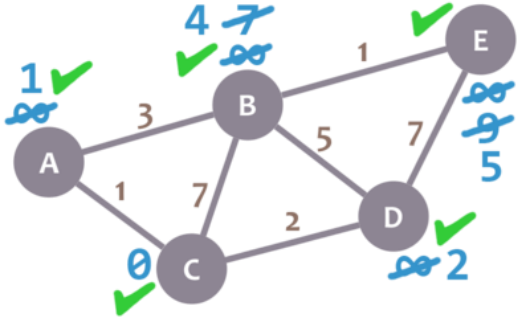
Algoritmo

Paso inicial: asignamos al vértice de inicio **“S”** una distancia de **0**, y a todos los demás vértices una distancia de infinito. Luego, en una cola de prioridad de orden ascendente de pares {**distancia, vértice**} agregamos el par {**0**, **S**}.

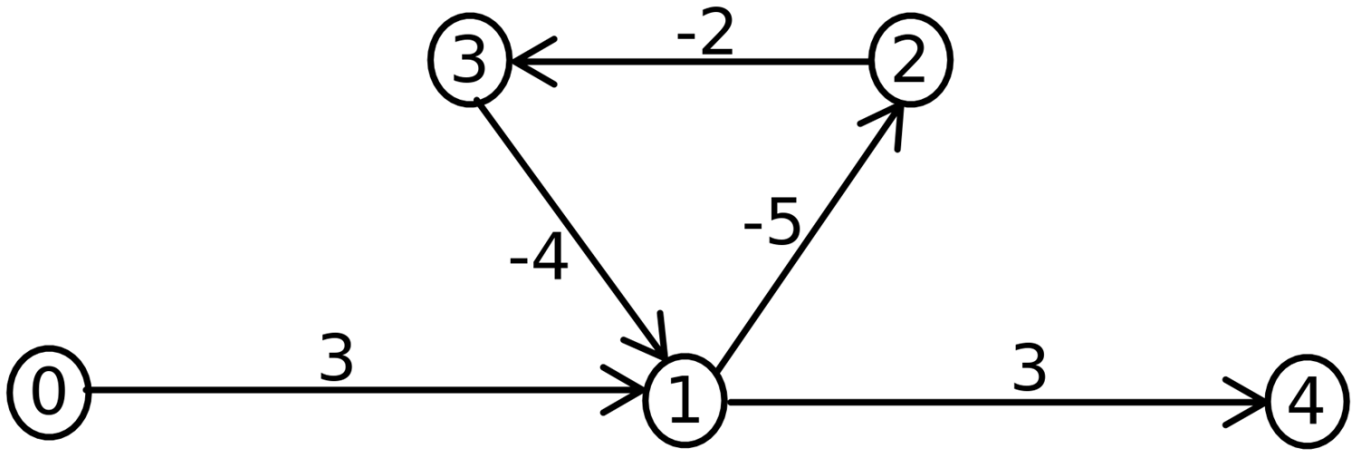
Paso de asignación de caminos: extraemos de la cola de prioridad el par mínimo {**D**, **U**}, si la distancia **D** extraída es menor que la distancia que teníamos asignada al vértice **U**, se la asignamos como nueva distancia mínima. Luego, para todos los vértices **V** que están conectados con el vértice actual **U**, agregamos a la cola de prioridad el par {**DU + W, V**}, siendo **DU** la distancia mínima a **U, y W** el peso de la arista que conecta a **U** con **V.** Este paso se repetirá mientras la cola de prioridad no esté vacía.







¿Que sucede en este grafo?



El valor del camino mínimo entre el vértice 0 y 4 no está bien definido, ya que se podría recorrer varias veces el ciclo de peso negativo 1->2->3 y disminuir infinitamente el valor del camino mínimo.

NOTA: En los grafos con ciclos de peso negativo, el algoritmo de Dijkstra se queda en un bucle infinito. Para estos casos se recomiendan otros algoritmos como por ejemplo el de **Bellman Ford**

Implementación: O((n+m)*log(m))

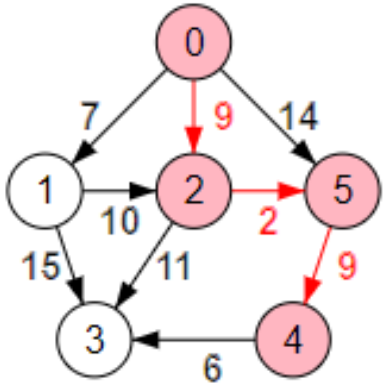
```cpp
const long long INF = 1e18;
typedef pair<long long, int> pli;

vector<long long> dijkstra(vector<vector<pair<int, int>>> &adj, int s, int n){
    vector<long long> d(n, INF);
    d[s] = 0;
    priority_queue<pli, vector<pli>, greater<pli>> pq;
    pq.push({0, s});
    while (!pq.empty()){
        pli par = pq.top(); pq.pop();
        int u = par.second;
        long long dist = par.first;
        if (dist > d[u]) continue;
        for (pli p : adj[u]){
            int v = p.first, w = p.second;
            if (d[v] > d[u] + w){
                d[v] = d[u] + w;
                pq.push({d[v], v});
            }
        }
    }
    return d;
}
```

Problema: https://redprogramacioncompetitiva.com/contests/2024/04/index.php problem H

Reconstruccion del camino

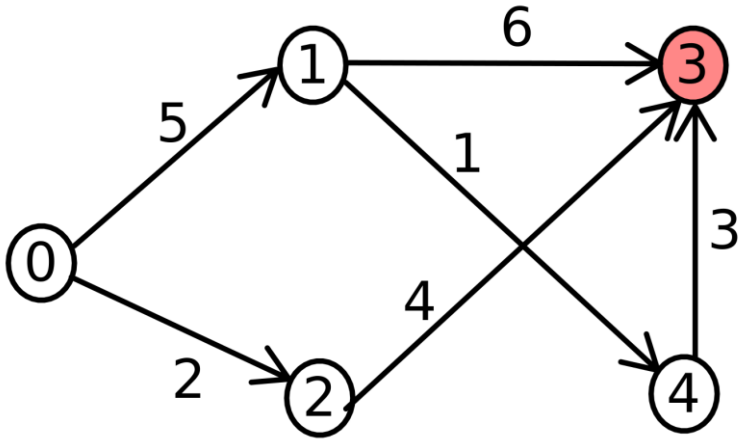
El algoritmo de Dijkstra por si solo nos dice el peso del camino mínimo, pero podemos usarlo para saber cuales son los vértices que recorremos en ese camino mínimo. Para esto creamos un vector donde almacenaremos qué vértices son los "**padres**" de cada vértice. Cada vez que agregamos un vértice **V** a la cola de prioridad, le asignamos como padre el vértice **U** por el cual llegamos a **V**.

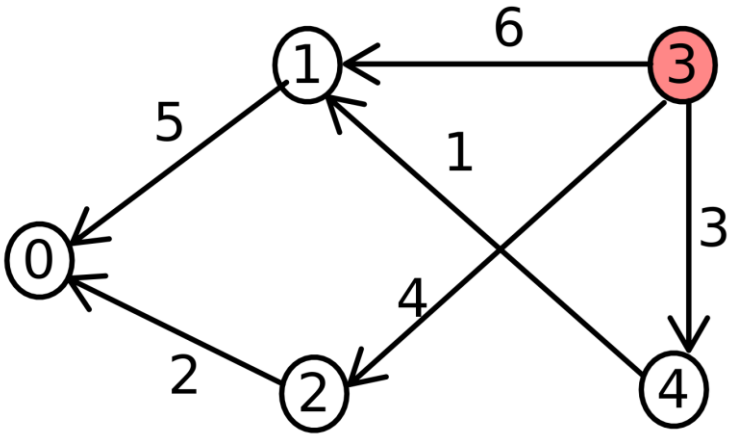


Problema: https://codeforces.com/contest/20/problem/C

Destino único

En caso de que tengamos que saber los caminos mínimos de todos los vértices a un vértice de destino **“T”** y nos dan un grafo dirigido, podemos cambiar el sentido de las aristas y usar el algoritmo de Dijkstra en este nuevo grafo, teniendo el vértice **“T”** como inicio.

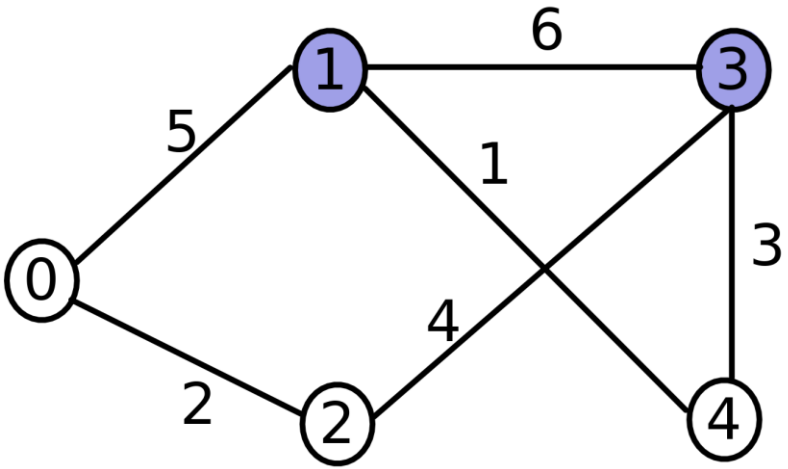




Problema:
https://redprogramacioncompetitiva.com/contests/2024/01/index.php
problem H

Orígenes múltiples

En caso de que tengamos un problema con varios vértices de origen, para resolverlo basta con agregar los pares {0, origen} a la cola de prioridad en el paso inicial del algoritmo de Dijkstra.



Problema:
https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=5038

Más problemas

https://cses.fi/problemset/task/1671

https://cses.fi/problemset/task/1195

https://codeforces.com/gym/101666 problem D

https://codeforces.com/gym/104736/problem/M

https://codeforces.com/contest/1915/problem/G