HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

APPLIED ALGORITHMS

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

APPLIED ALGORITHMS

Dijkstra algorithm with priority queues

ONE LOVE. ONE FUTURE.

Shortest Path from a source node in a directed graph non-negative weights

- Given a directed graph $G = (V, E, A, w)$ in which
  - V: set of nodes
  - E: set of arcs
  - $A[v]$: list of adjacent nodes of v
  - $w(u, v)$: non-negative weight of the arc (u, v)
- Given a source node s and a destination nodes t, find the shortest path from s to t in G
- Dijkstra algorithm
  - $d[v]$: upper bound of the length of the shortest path from s to v
  - Upper bound improvement
    - If $d[v] > d[u] + w(u, v)$ then
      - update $d[v] = d[u] + w(u, v)$

```
Dijkstra(G = (V,E,w)) {
  for(v ∈ V) {
     d(v) = w(s,v);
  }
  S = V \ {s};// tap cac dinh chua tim duoc duong
              // di ngan nhat
  while(S ≠ {}) {
    u = select a node ∈ S having minimum d(u);
    if (u = t) break; // found the shortest path
    S = S \ {u};
    for(v ∈ S) {
      if(d(v) > d(u) + w(u,v) {
        d(v) = d(u) + w(u,v);
      }
    }
  }
}
```

Shortest Path from a source node in a directed graph non-negative weights

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e5+1;
const int INF = 1e9;

struct Arc{
    int node;
    int w;
    Arc(int _node, int _w){
        node = _node; w = _w;
    }
};
int n,m;
vector<Arc> A[N];
int s,t;
int d[N]; // d[v] is the distance from s to v
```

```
void solve(){
    for(int v = 1; v <= n; v++) d[v] = INF;
    priority_queue<pair<int,int>, vector<pair<int,int> >,
                                        greater<pair<int,int> > > pq;
    d[s] = 0;    pq.push(make_pair(d[s],s));
    while(!pq.empty()){
        pair<int,int> p = pq.top(); pq.pop();
        int u = p.second;
        if(u == t)    break;
        for(int i = 0; i < A[u].size(); i++){
            Arc a = A[u][i];     int v = a.node;      int w = a.w;
            if(w + d[u] < d[v]){
                d[v] = d[u] + w;      pq.push(make_pair(d[v],v));
            }
        }
    }
    cout << "Length of the shortest Path from s to t is " << d[t] << endl;
}
```

HUST

THANK YOU !