Trees & Graphs Lecture 8

Saturday, 24 August 2024 6:10 AM

Single Source Shortest Paths

- Trees — DFS/BFS
- Graphs
  - UnWeighted — BFS / DFS
  - weighted
    - Dijkstra
    - Bellman Ford

Unweighted Graphs

https://www.geeksforgeeks.org/problems/shortest-path-in-undirected-graph-having-unit-distance/1

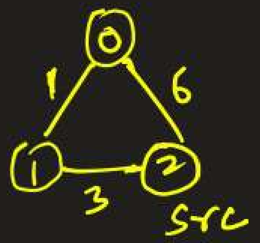
```
// User function Template for C++
class Solution {
  public:
    vector<int> shortestPath(vector<vector<int>>& edges, int N,int M, int src){
        vector<int> adj[N];
        for(auto e: edges) {
            adj[e[0]].push_back(e[1]);
            adj[e[1]].push_back(e[0]);
        }
        vector<int> ans(N, -1);
        queue<int> q;
        q.push(src);
        ans[src] = 0;
        while(!q.empty()) {
            for(int v: adj[q.front()]) {
                if(ans[v] == -1) {
                    ans[v] = ans[q.front()]+1;
                    q.push(v);
                }
            }
            q.pop();
        }
        return ans;
    }
};
```

$T = O(V+E)$

$S = O(V)$

Weighted Graphs

https://www.geeksforgeeks.org/problems/implementing-dijkstra-set-1-adjacency-matrix/1



dis: 4 ~~6~~ 3

~~∞~~	~~∞~~	0
0	1	2

$d = \cancel{0}\ \cancel{3}\ \cancel{4}\ \cancel{6}$

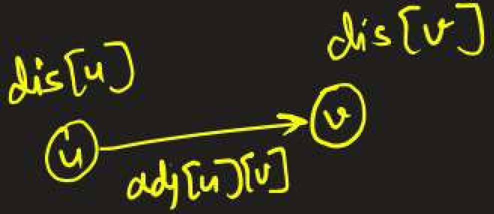
$u = \cancel{2}\ \cancel{1}\ \cancel{0}\ \cancel{0}$

pq:

~~{4, 0}~~
~~{3, 1}~~
~~{6, 0}~~
~~{0, 2}~~

pq {w, v}

```
for e in adj[u]:
    if dis[u] + e.w[v] < dis[v]:
relax   dis[v] = dis[u] + e.w[v]
```



if dis[v] > dis[u] + adj[u][v]

=

```cpp
class Solution
{
    public:
    //Function to find the shortest distance of all the vertices
    //from the source vertex S.
    vector <int> dijkstra(int V, vector<vector<int>> adj[], int S) {
        priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
        vector<int> dis(V, INT_MAX);
        dis[S] = 0;
        pq.push({0, S});
        while(!pq.empty()) {
            for(auto v: adj[pq.top().second]) {
                if(v[1] + pq.top().first < dis[v[0]]) {
                    dis[v[0]] = v[1] + pq.top().first;
                    pq.push({dis[v[0]], v[0]});
                }
            }
            pq.pop();
        }
        return dis;
    }
};
```

if dis[v] > d + e

dis[v] = d + e

$\infty > 0 + 6$

dis

6 ~~∞~~	∞	0
0	1	2

{6, 0}

~~{0, 2}~~

pq.top = {0, 2}

v = [0, 6]

v = [1, 3]

if pq.t.f + v[1] < dis [v[0]]

{ }

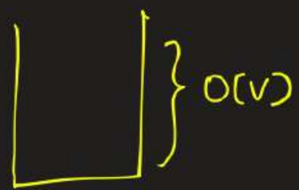
```cpp
class Solution
{
    public:
    //Function to find the shortest distance of all the vertices
    //from the source vertex S.
    vector <int> dijkstra(int V, vector<vector<int>> adj[], int S) {
        priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
        vector<int> dis(V, INT_MAX);
        dis[S] = 0;
        pq.push({0, S});
        while(!pq.empty()) {
            pair<int, int> top = pq.top();
            int du = top.first, u = top.second;
            for(auto e: adj[u]) {
                int v = e[0], ew = e[1];
                if(ew + du < dis[v]) {
                    dis[v] = ew + du;
                    pq.push({dis[v], v});
                }
            }
            pq.pop();
        }
        return dis;
    }
};
```

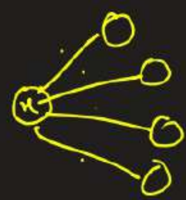
$O(V)$

$O(\log V)$

$O(V-1)$

$O(\log V)$





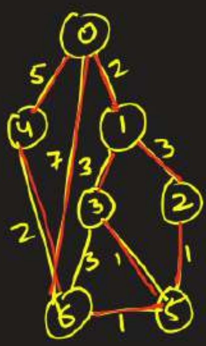
$$\boxed{\begin{array}{l} T = O(E \log V) \\ S = O(V) \end{array}}$$

$$O\left(V\left[\log V + (V-1)\log V\right]\right)$$

$$= O(V \log V (V)) \quad = O(V^2 \cdot \log V)$$

$$= \boxed{O(E \log V)}$$

https://leetcode.com/problems/number-of-ways-to-arrive-at-destination/



n = 7

0 → 6

len = 7:
- 0 → 4 → 6
- 0 → 1 → 2 → 5 → 6
- 0 → 1 → 3 → 5 → 6

pq {w, u}:

~~{6, 5}~~
~~{5, 3}~~
~~{5, 2}~~
~~{7, 6}~~
~~{5, 4}~~
~~{2, 1}~~
~~{0, 0}~~

dis

	0	1	2	3	4	5	6
dis	0	~~∞~~ 2	~~∞~~ 5	~~∞~~ 5	~~∞~~ 5	~~∞~~ 6	~~∞~~ 7
# ways	1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ ~~1~~ 2	~~0~~ ~~1~~ ~~2~~ 4

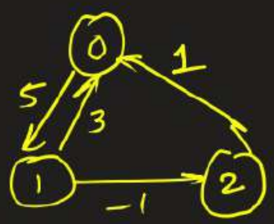
```
du = 0̸ 2
u = 0̸ 1
for {ew, v} in adj[u]:
    if d[v] > ew + du:
        d[v] = ew + du
        pq.push({d[v], v})
        - w[v] = w[u]
    if d[v] == ew + du:
        - w[v] = w[v] + w[u]
```

```python
import heapq

class Solution:
    def countPaths(self, n: int, roads: List[List[int]]) -> int:
        MOD = int(1e9+7)
        adj = [{} for _ in range(n)]
        for [u, v, w] in roads:
            adj[u][v] = w
            adj[v][u] = w
        pq = [(0, 0)]
        w = [0]*n
        w[0] = 1
        dis = [int(1e18)]*n
        dis[0] = 0

        while pq:
            (du, u) = heapq.heappop(pq)
            for v in adj[u]:
                if du + adj[u][v] < dis[v]:
                    dis[v] = du + adj[u][v]
                    heapq.heappush(pq, (dis[v], v))
                    w[v] = w[u]
                elif du + adj[u][v] == dis[v]:
                    w[v] = (w[u] + w[v])%MOD

        return w[n-1]
```

$T = O(E \log V)$

$S = O(V)$

Bellman - Ford Algorithm



Edges

{0, 1, 5}

{1, 0, 3}

{1, 2, -1}

{2, 1, 0}

Relax all edges in same order (n-1) times

↳ 1 more time if the dis of some node still reduces

↳ Graph has (-)ve weight cycles

```python
class Solution:
    # Function to construct and return cost of MST for a graph
    # represented using adjacency matrix representation
    '''
    V: nodes in graph
    edges: adjacency list for the graph
    S: Source
    '''
    def bellman_ford(self, V, edges, S):
        dis = [int(1e8)]*V
        dis[S] = 0
        for _ in range(V-1):
            for [u, v, w] in edges:
                if dis[u] != int(1e8) and dis[u] + w < dis[v]:
                    dis[v] = dis[u] + w
        for [u, v, w] in edges:
            if dis[u] != int(1e8) and dis[u] + w < dis[v]:
                return [-1]
        return dis
```

$\rightarrow O(V-1)$

$\rightarrow O(E)$

$\rightarrow O(E)$

$$O((V-1)E + E)$$

$$= \boxed{O(V \cdot E)}$$

$$\simeq \underline{O(V^3)}$$