Shortest path

Friday, October 27, 2023 5:39 AM

Djikstra's shortest path algorithm

- Single source shortest path algo.
- Only works for graphs with +ve edges.

Time complexity : O(E+logV)

- Djikstra's algorithm is a greedy algo.

infinitely finds shorter path in case of -ve cycles

Code :

```
'''
* check if curr node is in the distance hashset or not
* if not, create an entry and add the current cost
* if present, find minimum and add
* in this approach, we push the new cost in the heap regardless of it being best or worst value
* notice that we do not need the visited boolean array in this approach
'''

def djikstra(edges, N, K):
    q, d, adj = [(0, K)], {}, collections.defaultdict(list)
    for u, v, w in edges:
        adj[u].append((v, w))
    while q:
        cost, node = heapq.heappop(q)
        if node not in d:
            d[node] = cost
            for v, w in adj[node]:
                heapq.heappush(q, (cost + w, v))
    return d

## normal implementation (more intuitive)
def djikstra(edges, N, K):
    q, d, adj, visited = [(0, K)], [10**8 for i in range(N+1)], collections.defaultdict(list), set()
    for u, v, w in edges:
        adj[u].append((v, w))
    d[K] = 0
    while q:
        cost, node = heapq.heappop(q)
        visited.add(node)

        for v, w in adj[node]:
            if v in visited: continue
            if w + cost < d[v]:
                d[v] = w + cost
                heapq.heappush(q, (d[v], v))
    return d
```

By property of heap.

This will always be the best cost

We directly push

Bellman ford SSP algo

- Better to use Djikstra O((E+V)logV)
- Used with graphs have -ve edges.
  - ↳ eg. performing arbitrage
    - ↳ convert currency.
- Essence is in performing relaxan for n-1 times.
  - No. of nodes in the graph.
- A graph without -ve cycles, the node distances will not improve beyond (n-1) iterations.
  - ↳ If that happens, the graph has a negative cycle.

```
Algorithm:
1. Mark all initial distances for each node as infinity..
2. for n - 1 times, explore all edges in the graph
3. record all shorter paths to each node for every edge

In case of a complete graph(edge between every pair of nodes), there can be n*(n - 1)/2 edges. The time complexity of Bellman Ford algorithm can be O(n^3) or O(E^2 V)
*/

void add_edge(int u, int v, int w){
    edges[cnt][0] = u;
    edges[cnt][1] = v;
    edges[cnt][2] = w;
    cnt++;
}

void bellmanFord(int src){
    fill(dist, dist + N, INF);
    dist[src] = 0;

    for(int i = 0; i < n-1; i++){
        for(int j = 0; j < cnt; j++){
            if(dist[edges[j][0]] + edges[j][2] < dist[edges[j][1]])
                dist[edges[j][1]] = dist[edges[j][0]] + edges[j][2];
        }
    }

    // check for negative cycles
    for(int i = 0; i < cnt; i++){
        if(dist[edges[i][0]] + edges[i][2] < dist[edges[i][1]]){
            cout << "Graph Contains Negative edge cycle !!\n";
            return;
        }
    }

    cout << "Distance of source \n";
    for(int i = 0; i < n; i++)
        cout << i << " " << dist[i] << "\n";
}
```

longest path without a cycle is of n-1 edges. ∴ we run n-1 relaxan's

relaxan

nth iteran determines if there is -ve cycle.

- In Bellman Ford it is better to travel edges.

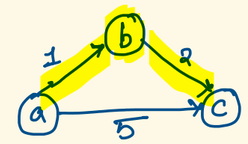
Relax edges `n-1` times.

- for edge (u,v) → w

check dist_till_u + edge_cost < dist_till_v

Floyd Warshall algorithm

- Good for find All pair shortest path
- Main idea is to find if there is an intermediate path between two nodes with less cost.



- Consider all intermediate paths O(V³).

```
void init(){
    for(int i = 0; i < n; i++)
    for(int j = 0; j < n; j++)
        i == j ? adj_mat[i][j] = 0 : adj_mat[i][j] = INF;
}

void add_edge(int u, int v, int w){ adj_mat[u][v] = w; }

void floyd_warshall(){
    for(int k = 0; k < n; k++)
    for(int i = 0; i < n; i++)
    for(int j = 0; j < n; j++)
        adj_mat[i][j] = min(adj_mat[i][j], adj_mat[i][k] + adj_mat[k][j]);
}
```

worse val if no direct edge

binding inter node

direct edge cost

- To find out if there is a -ve cycle.
  - ↳ Rerun the Floyd Warshall's algo. If there are still better paths are found, it means there is -ve cycle.

- While adding a new edge, we need to relax all paths.

```
new-edge → u,v,w
for i → 0 to n-1
    for j → 0 to n-1
        cost(i,j) = min(
                cost(i,j),
                cost(i,u) + w + cost(v,j)
        );
```