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 : $O(E + \log V)$

- Djikstra's algorithm is a greedy algo.

  infinitely finds shorter path in case of -ve cycles

Code :

```
1  '''
2  * check if curr node is in the distance hashset or not¬
3  * if not, create an entry and add the current cost¬
4  * if present, find minimum and add¬
5  * in this approach, we push the new cost in the heap
   regardless of it being best or worst value
6  * notice that we do not need the visited boolean array
   in this approach¬
7  '''¬
8  ¬
9  def djikstra(edges, N, K):¬
10 ····q, d, adj = [(0, K)], {},
   collections.defaultdict(list)¬
11 ····for u, v, w in edges:¬
12 ········adj[u].append((v, w))¬
13 ····while q:¬
14 ········cost, node = heapq.heappop(q)¬
15 ········if node not in d:¬
16 ············d[node] = cost¬
17 ············for v, w in adj[node]:¬
18 ················heapq.heappush(q, (cost + w, v))¬
19 ····return d¬
20 ¬
21 ## normal implementation (more intuitive)¬
22 def djikstra(edges, N, K):¬
23 ····q, d, adj, visited = [(0, K)], [10**8 for i in
   range(N+1)], collections.defaultdict(list), set()¬
24 ····for u, v, w in edges:¬
25 ········adj[u].append((v, w))¬
26 ····d[K] = 0¬
27 ····while q:¬
28 ········cost, node = heapq.heappop(q)¬
29 ········visited.add(node)¬
30 ¬
31 ········for v, w in adj[node]:¬
32 ············if v in visited: continue¬
33 ············if w + cost < d[v]:¬
34 ················d[v] = w + cost¬
35 ················heapq.heappush(q, (d[v], v))¬
36 ····return d¬
```

By property of heap.

This will always be the best cost

we directly push!

Bellman ford SSP algo

- Better to use Djikstra $O((E+V)\log V)$
- Used with graphs have -ve edges.
  ↳ eg. performing arbitrage
       ↳ convert currency.

- Essence is in performing relaxan for $n-1$ times.
  $\overline{\text{No. of nodes in the graph.}}$
- A graph without -ve cycles, the node distances will not improve beyond $(n-1)$ iterations.
  ↳ If that happens, the graph has a negative cycle.

```
20 Algorithm:
21 1. Mark all initial distances for each node as infinity..
22 2. for n - 1 times, explore all edges in the graph
23 3. record all shorter paths to each node for every edge
24
25 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)
26 */
27
28 void add_edge(int u, int v, int w){
29     edges[cnt][0] = u;
30     edges[cnt][1] = v;
31     edges[cnt][2] = w;
32     cnt++;
33 }
34
35 void bellmanFord(int src){
36     fill(dist, dist + N, INF);
37     dist[src] = 0;
38
39     for(int i = 0; i < n-1; i++){
40         for(int j = 0; j < cnt; j++){
41             if(dist[edges[j][0]] + edges[j][2] < dist[edges[j][1]]){
42                 dist[edges[j][1]] = dist[edges[j][0]] + edges[j][2];
43             }
44         }
45     }
46     // check for negative cycles
47     for(int i = 0; i < cnt; i++){
48         if(dist[edges[i][0]] + edges[i][2] < dist[edges[i][1]]){
49             cout << "Graph Contains Negative edge cycle !!\n";
50             return;
51         }
52     }
53
54     cout << "Distance of source \n";
55     for(int i = 0; i < n; i++)
56         cout << i << " " << dist[i] << "\n";
57 }
```

*longest path without a cycle is of $n-1$ edges.
∴ we run n-1 relaxaⁿs*

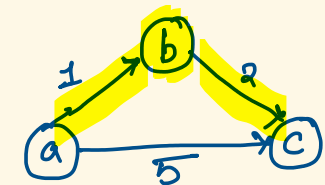*relaxaⁿ*

nᵗʰ iteraⁿ determines if there is -ve cycle.

- In Bellman Ford it is better to travel edges.

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 $O(V^3)$.

```
9  void init(){
10     for(int i = 0; i < n; i++)
11         for(int j = 0; j < n; j++)
12             i == j ? adj_mat[i][j] = 0 : adj_mat[i][j] = INF;
13 }
14 void add_edge(int u, int v, int w){ adj_mat[u][v] = w; }
15
   void floyd_warshall(){
16     for(int k = 0; k < n; k++)
17         for(int i = 0; i < n; i++)
18             for(int j = 0; j < n; j++)
19                 adj_mat[i][j] = min(adj_mat[i][j], adj_mat[i][k] + adj_mat[k][j]);
20 }
```

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.