→ Dijkstra's Algo (Revision)

→ Bellman Ford

→ Floyd Warshall

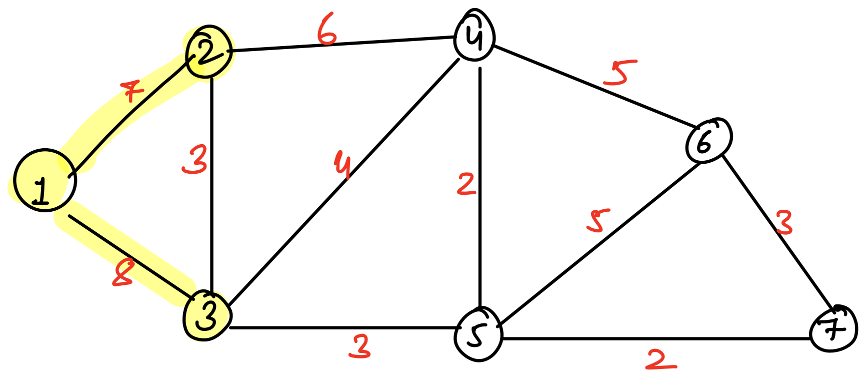
Dijkstra's Algorithm [Single sourced shortest path]

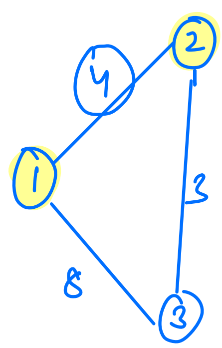
Q) There are N cities in a country, you are living in city-1. Find minimum distance to reach every other city from city-1.

[edge wt. > 0]

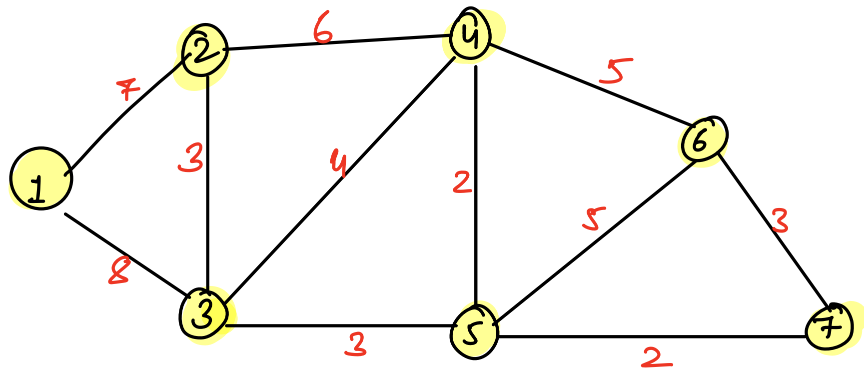
Expected output

dist	0	7	8	12	11	16	13
	1	2	3	4	5	6	7



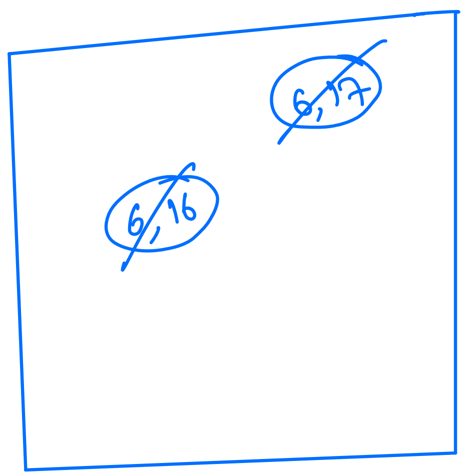


→ minimum wt. so far starting from the source.



dist →

0	1	2	3	4	5	6	7
X	~~∞~~ 0	~~∞~~ 7	~~∞~~ 8	~~∞~~ 12	~~∞~~ 11	~~∞~~ 16	∞ 13



min-Heap.

6, 16

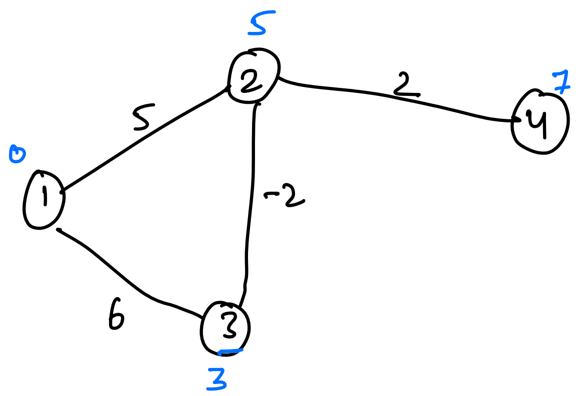
```
pair {
    int v;   (vertex)
    int wsf; (wt. so far)
}
```

code :-

```
dist [N+1] .  // ∀i, dist[i] = INT_MAX ;
minHeap <Pair> heap ;
heap.insert( new Pair (src, 0)) ;
while( heap.size () != 0){
        Pair  rp  =  heap.removeMin();
        if ( dist [rp.v]  != INT_MAX) { continue }
        else {
                dist [rp.v] = rp.wsf ;
                for( int nbr : graph[rp.v]){
                        if( dist[nbr] == INT_MAX){
                                heap.insert(new Pair( nbr, rp.wsf + wt of ));
                                                                 current
                                                                 edge
                        }
                }
        }
}
return dist[] ;
```

$$\boxed{\begin{array}{l} T.C \rightarrow O(E \log E) \\ S.C \rightarrow O(E) \end{array}}$$

→ If we have -ve weights.

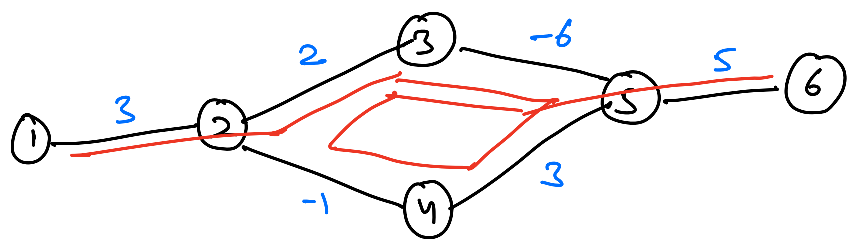


shortest distance from 1 to 4 → 6

But, if we apply Dijkstra's algo → 7. which is wrong.

Greedy algorithm ⇒ it fails with -ve weights.

Is it always possible to find the shortest distance from one node to another node? ⇒ No



3 + 2 + (-6) + 3 + (-1) + 2 + (-6) + 5 = 2.

If a graph doesn't have -ve edge wt cycle, then how to find min distance to all nodes?

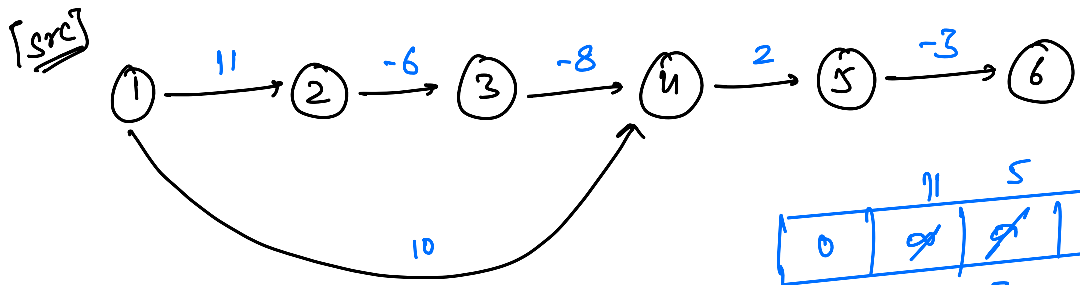
⇓

[**Bellman Ford Algorithm**]

Key idea. → Minimum distance can be found by relaxing all the edges at max N-1 times, irrespective of the order in which edges are selected.

Why N-1 times?

We can have a maximum of N-1 edges b/w any pair of nodes.



1	2	3	4	5	6
0	~~∞~~ 11	~~∞~~ 5	~~∞~~ ~~18~~ -3	~~∞~~ ~~12~~ -1	~~∞~~ ~~9~~ -4

$N = 6$, $E = 6$

edges:

edges	itr→1	itr→2	itr→3	itr→4	itr→5
5, 6, -3	✗	✗	✓	✗	✓
4, 5, 2	✗	✓	✗	✓	✗
1, 4, 10	✓	✗	✗	✗	✗
3, 4, -8	✗	✗	✓	✗	✗
2, 3, -6	✗	✓	✗	✗	✗
1, 2, 11	✓	✗	✗	✗	✗

$$\left[\begin{array}{l} d[u] + ed.wt(u,v) < d[v] \\ \quad d[v] = d[u] + ed.wt(u,v) \end{array}\right]$$

→ if there is an updation in N^{th} iteration, that means there exists a path where no. of edges are greater than $N-1$.

⇒ At least one of the edge is visited multiple times.

∴ Detection of -ve edge weight cycle.

code →

```
int dist[N+1],    ∀i, dist[i] = ∞
dist[src] = 0;
for( i = 1; i < N ; i++){
    for( j = 1; j <= E; j++){
        if( dist[graph[j][0]] + graph[j][2] < dist[graph[j][1]]){
            dist[graph[j][1]] = dist[graph[j][0]] + graph[j][2];
        }
    }
}
return dist[];
```

T.C → $O(N \times E)$
S.C → $O(1)$

Q) Shortest Path from Every Node to Every Other Node

(A1) - if all edge wts. are +ve

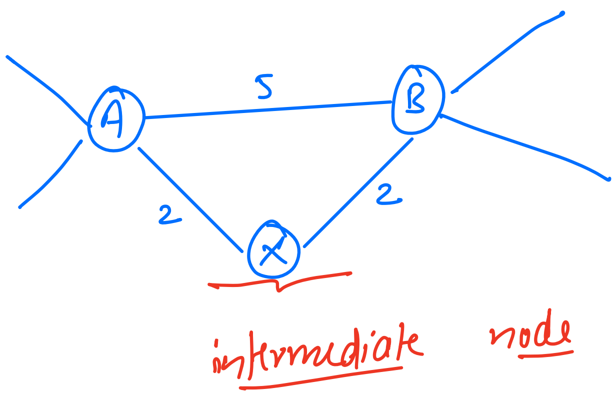
Apply Dijkstra's algo by considering every node as the source node.

T.C → $O(N \cdot E \cdot \log E)$

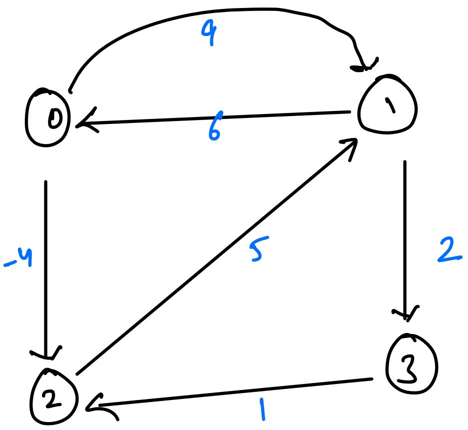
(A2) → Apply Bellman Ford's Algo considering every node as source node.

T.C → $O(N^2 \cdot E)$

(A3) → Floyd Warshall's Algo



∀ nodes, consider them as intermediate nodes & try to relax the connected edges.



0

$$\underbrace{d[i][1] + d[1][j] < d[i][j]}$$

adj matrix

	0	1	2	3
0	0	9	-4	∞
1	6	0	∞	2
2	∞	5	0	∞
3	∞	∞	1	0

node→0 as intermediate node

	0	1	2	3
0	0	9	-4	∞
1	6	0	2	2
2	∞	5	0	∞
3	∞	∞	1	0

1 → int node

	0	1	2	3
0	0	9	-4	11
1	6	0	2	2
2	11	5	0	7
3	∞	∞	1	0

2 → int. node

	0	1	2	3
0	0	1	-4	3
1	6	0	2	2
2	11	5	0	7
3	12	6	1	0

3 → int. node.

	0	1	2	3
0	0	1	-4	3
1	6	0	2	2
2	11	5	0	7
3	12	6	1	0

code →

```
for( k = 0; k < N; k++){
    for( i = 0; i < N; i++){
        for( j = 0; j < N; j++){
            if( d[i][k] + d[k][j] < d[i][j] ){
                d[i][j] = d[i][k] + d[k][j];
            }
        }
    }
}
```

T.C → $O(N^3)$
S.C → $O(1)$

→ 30th May
→ 1st June → Contest discussion

→ LLD (next date)
→ SQL (27th may)