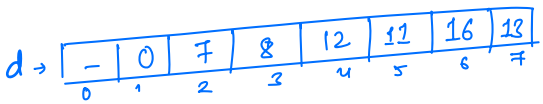
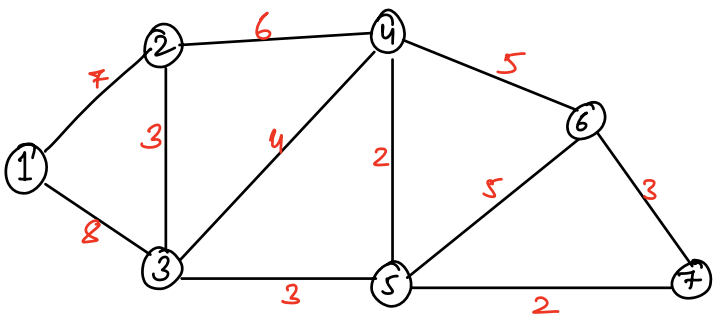
Dijkstra's Algorithm → single source shortest path with +ve edge wts.

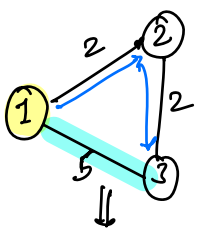
Q.1) There are N cities in a country, you are living in city-1. Find minimum distance to reach every other city from city-1.





d[i] - minimum distance to reach i^{th} city starting from source.

d[src] = 0.

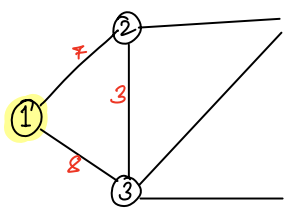


d[1] = 0
d[2] = 2
d[3] = 4

Relaxing an edge.

x – y edge is never used

∴ x – z – y is better option.

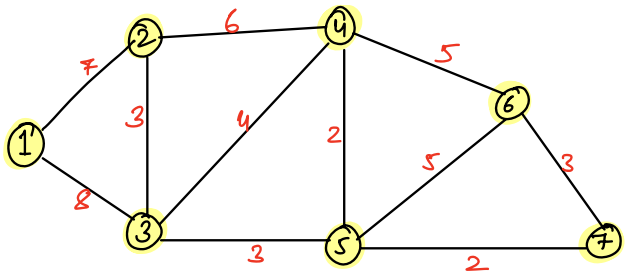


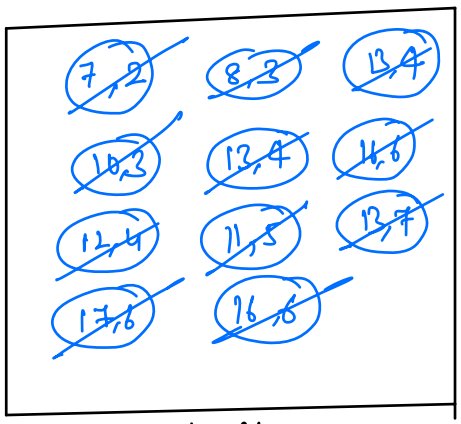
d → | – | 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
0 1 2 3 4 5 6 7

minimum edge wt. starting from source.

dist-

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





min. Heap.

[d[u] + edge wt, v]

⇓

distance so far

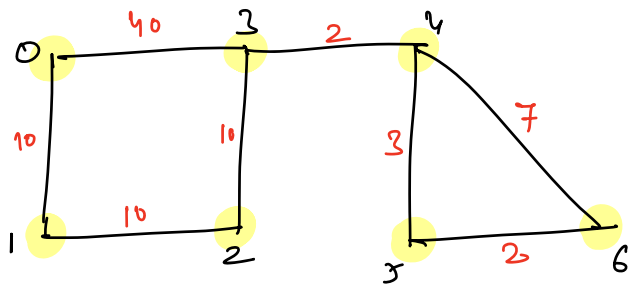
Pair { wt, v }

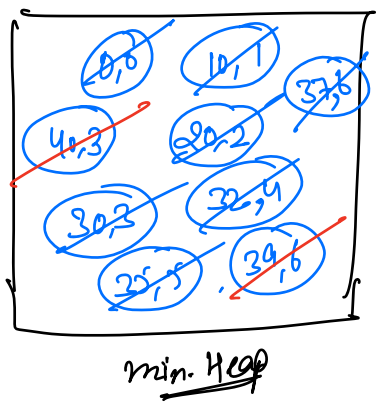
pseudo-code

```
∀i, dist[i] = Integer.MAX_VALUE;

min.Heap < pair > heap;

heap.insert({0, src});

while( heap.size() > 0 ){

    Pair rp = heap.extractMin();

    if( dist[rp.v] == Integer.MAX_VALUE){

        dist[rp.v] = rp.wt;

        for( nbr : Adj[rp.v]){

            if( dist[nbr] == Integer.MAX_VALUE){

                heap.insert(d[rp.v] + wt, nbr);

            }

        }

    }

}
```

T.C → O(E log E + N)

S.C → O(E)

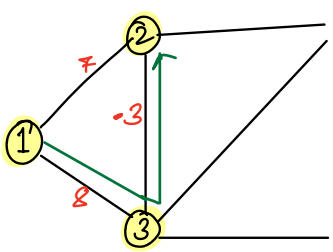




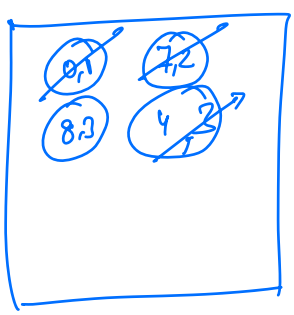
dist →

0	10	20	30	32	35	37
0	1	2	3	4	5	6

Q: Can it work if any wt < 0 ? No.

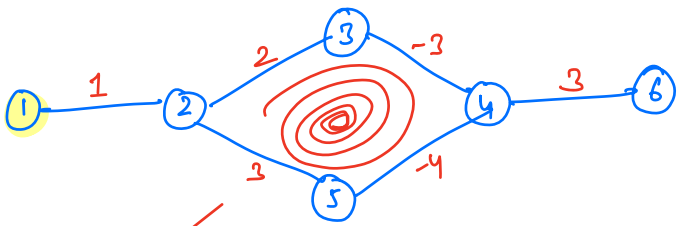


	7	4			
0	1	2	3	4	5
0	∞	∞	∞	∞	∞



Q How to find shortest path to all nodes from a single source if -ve weights are present.

Is it always possible to find shortest path? No



d[6] = ?

1 → 2 → 3 → 4 → 6 = 3

1 → 2 → 3 → 4 → 5 → 2 → 3 → 4 → 6 = 1

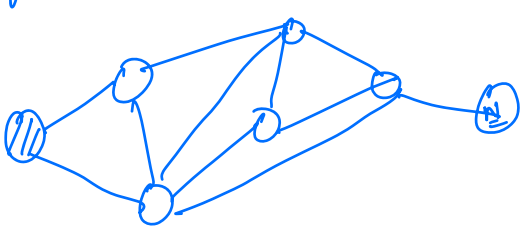
[Negative weight cycle is present]

Σ wt of edges in a cycle < 0 (Ans is not possible)

Break → [10:29 → 10:36]

Bellman Ford

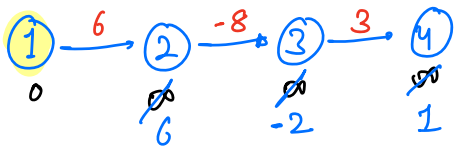
Minimum distance can be found by updating/relaxing all the edges (N-1) times, irrespective of the order in which the edges are selected.



N nodes.

⇓

(N-1) edges in any path. starting from src.

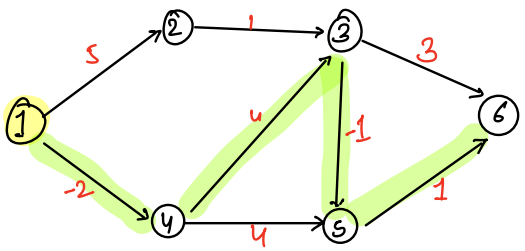



```
if( d[u] + w[u,v] < d[v] ){
    d[v] = d[u] + w[u,v]
}
```

1—2 ✓
2—3 ✓
3—4 ✓

	①	②	③
3—4	✓	✓	✓
2—3	✓	✓	✓
1—2	✓	✓	✓

order matters.



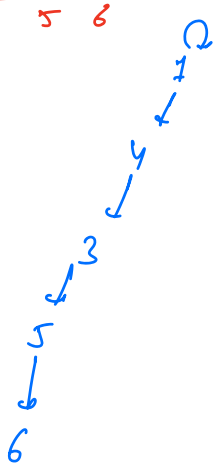
d→

0	1	2	3	4	5	6
∞	0	5	2	-2	1	2

p→

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

	itr=1.	itr=2	itr=3.
2 —1→ 3	×	×	×
1 —5→ 2	✓	×	×
2 —-1→ 5	×	✓	×
3 —3→ 6	×	✓	×
5 —1→ 6	×	✓	×
4 —4→ 5	×	×	×
1 —-2→ 4	✓	×	×
4 —4→ 3	✓	×	×



pseudo-code.

```
∀ i,   d[i] = ∞ ,   p[i] = -1

   d[src] = 0 ,   p[src] = src

for( i = 1 ; i <= N-1 ; i++){
    boolean stop = true
    for( (u,v) in edges){
        if( d[u] + w[u,v] < d[v]){
            d[v] = d[u] + w[u,v]
            stop = false
            p[v] = u;
        }
    }
    if(stop == true) { break }
}
```

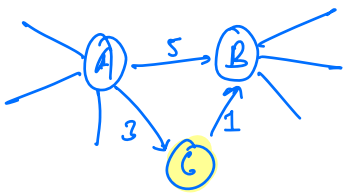
T.C → O(N * E)

S.C → O(N)

Q) Find shortest distance from every node to every other node.

Floyd Warshall's Algorithm

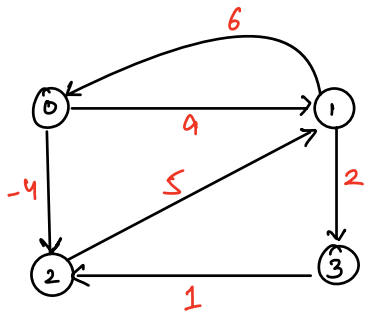
↳ All pair shortest path.



is shortest distance b/w A & B is due to direct edge? No.

intermediate node.

idea → consider every node as intermediate node and try to relax the edges with large weight.



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

Adjacency matrix

intermediate node → 0

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

$d[1][0] + d[0][2] < d[1][2]$

$6 + (-4) < \infty$

$d[1][0] + d[0][3] < d[1][3]$

$6 + \infty < 2$

intermediate node → 1

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

intermediate node → 2

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

intermediate node → 3.

H.W.

pseudo-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(N³) S.C → O(1)]

Revision.

- L.L
- Trees
- Searching
- Sorting
- Hashing
- Stack n Queue

Do it topicwise.

{ Revise the class notes. }
↓
{ try out some already done questions. }
↓
{ Always try new questions }

- Heaps
- Strings.
- Recursion
- D.P ← [Google, Microsoft, Amazon]
- Graph ←

- B.M
- Tries
- Z-Algo