Today's agenda

↳ Shortest Path Algo

↳ Dijkstra's

↳ bellman ford

AlgoPrep

↳ Shortest Path Algo

↳ BFS: Shortest Path in terms of edge count.

In terms of edge weights

$O(E*\log V)$ ← ↳ Dijkstra's: Single src Shortest Path algo, only for non-negative edge weights.

$O(E*V)$ ← ↳ Bellman ford: Single src Shortest Path algo, for -ve edge wts as well.

↳ $O(E*V)*V \Rightarrow O(E.V^2)$

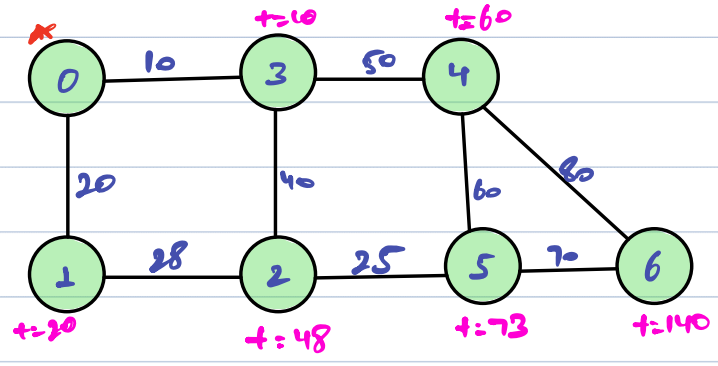
↳ $O(V^2 V^2) \Rightarrow O(V^4)$

$O(V^3)$ ← ↳ floyd warshall: Any to Any Shortest Path algo, +ve, 0, -ve for all edge wts.

AlgoPrep

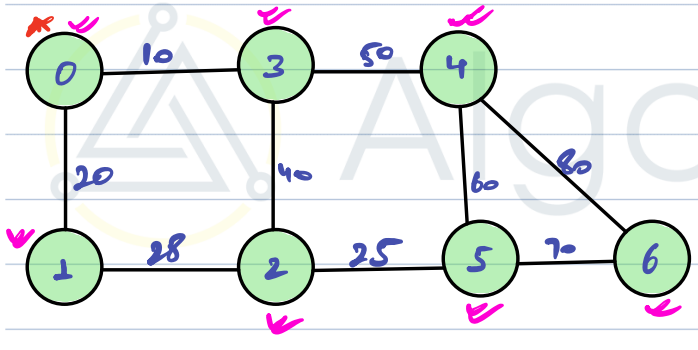
Dijkstra's algo

→ SOC = O



t=10 → 3

t=20 → 1



{v, w*}

{min PQ}

~~{5, 73}~~ ~~{4, 140}~~

~~{0, 0}~~ ~~{1, 20}~~

~~{1, 20}~~ ~~{2, 68}~~

~~{3, 10}~~ ~~{2, 48}~~

~~{6, 140}~~ ~~{6, 143}~~

0	1	2	3	4	5	6
0 ~~+∞~~	20 ~~+∞~~	48 ~~+∞~~	10 ~~+∞~~	60 ~~+∞~~	73 ~~+∞~~	140 ~~+∞~~

rem: {5, 73}

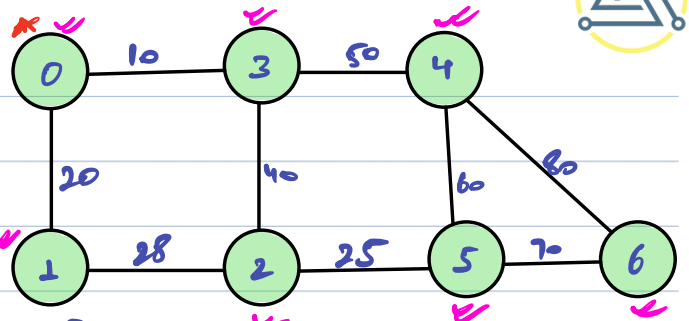
AlgoPrep

// Pseudo code

```
Class Pair1 {  → Graph
    int v;
    int wt;
}

Class Pair2 {  → PQ
    int vtx;
    int wsf;
}

int[] Dijkstra ( List<List<Pair1>> graph, int src){

    PriorityQueue <Pair2> PQ;
        int ans[n] = {+∞};

    PQ.add( new Pair2(src, 0));

T.C: O(E log v)
S.C: O(E)

    while ( PQ.size() >0 ){
        Pair2 rem = PQ.remove(); → rem.vtx
                                 → rem.wsf

        if (ans[rem.vtx] != +∞){ Continue; }
        ans[rem.vtx] = rem.wsf;

        List<Pair1> nbrs = graph.get(rem.vtx);
        for(Pair1 n: nbrs){
            if (ans[n.v] != +∞){ Continue; }
            PQ.add(new Pair2(n.v, rem.wsf + n.wt));
        }
    }
    return ans;
}
```

Tracing

```
class Pair1 {  → Graph
    int v;
    int wt;
}

class Pair2 {  → PQ
    int vtx;
    int wt;
}

int[] Dijkstra (List<List<Pair1>> graph, int src){

    PriorityQueue<Pair2> PQ;
    int ans[n] = {+∞};

    PQ.add(new Pair2(src, 0));

    while (PQ.size() > 0 ){
        Pair2 rem = PQ.remove();   → rem.vtx, → rem.wt
        if (ans[rem.vtx] != +∞){ continue; }
        ans[rem.vtx] = rem.wt;

        List<Pair1> nbrs = graph.get(rem.vtx);
        for (Pair1 n : nbrs){
            if (ans[n.v] != +∞){ continue; }
            PQ.add(new Pair2(n.v, rem.wt + n.wt))
        }
    }
    return ans;
}
```



0	1	2	3	4	5	6
{1,20}	{0,10}	{1,28}	{2,40}			
{3,10}	{2,28}	{3,40}	{0,10}			
		{5,25}	{4,50}			

Pair1

rem.vtx, rem.wt

rem = {3, 10}

min PQ (Pair2)

~~{0,10}~~
~~{0,0}~~
{2,50}
{1,20}
{4,60}

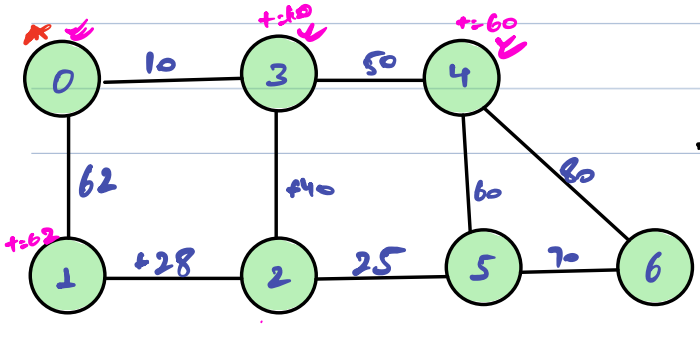
0	1	2	3	4	5	6
~~0~~	∞	∞	~~∞~~ 10	∞	∞	∞

nbrs → {2,40} {0,10} {4,50}

n.v n.wt

TC: O(E log E) → E = O(V²)

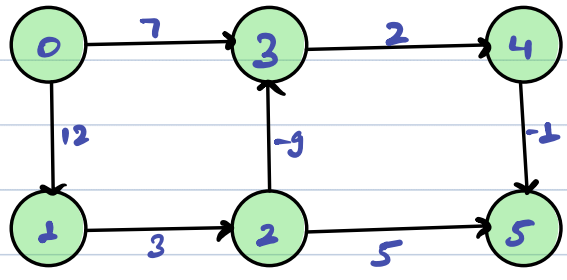
↳ E log V² = 2E log V ≃ O(E log V)



→ works only for +ve edge wt.

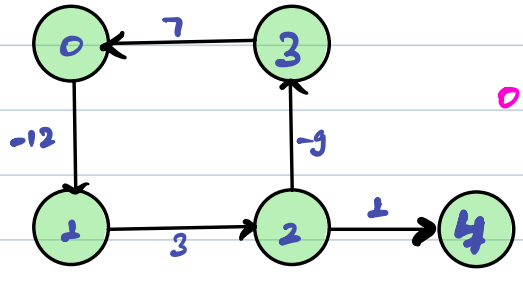
Q) Bellman ford

↳ Shortest Path from a given src to all the remaining nodes in terms of edge weight.



Obs 1:

Cycle in graph with overall -ve weight, Can we have Shortest Path?



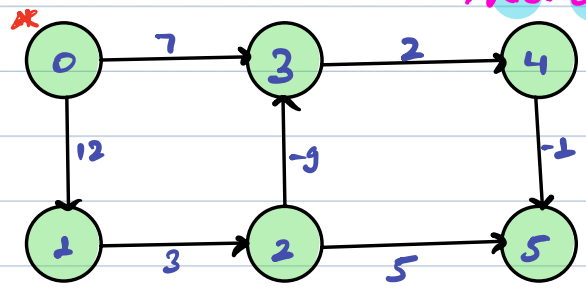
0→2 → -9

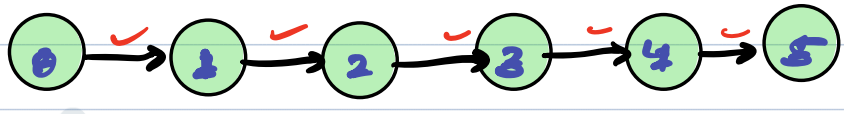
0→4 → -8, -20, -32 . .

→ Shortest Paths are not Possible for Such Case.

Obs 2: What can be the Path length in a graph while finding Shortest Path.

↳ No. of nodes − 1.



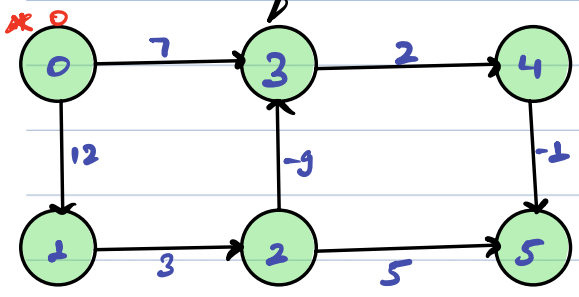


↳ No. of nodes − 1.

There is no cycle Possible.

AlgoPrep

→ Bellman ford



u	v	wt
4	5	-1
2	5	5
3	4	2
2	3	-9
1	2	3
0	1	12
0	3	7

→

ans:

0	1	2	3	4	5
0	~~∞~~ 12	~~∞~~ 15	~~∞~~ ~~7~~ 6	~~∞~~ 9	~~∞~~ 8

```
if(ans[u] + wt < ans[v])
    ans[v] = ans[u] + wt;
```

iter 1: 1 and 3

iter 2: 2 and 4

iter 3: 3 and 5

⋮

iter v-1:

AlgoPrep

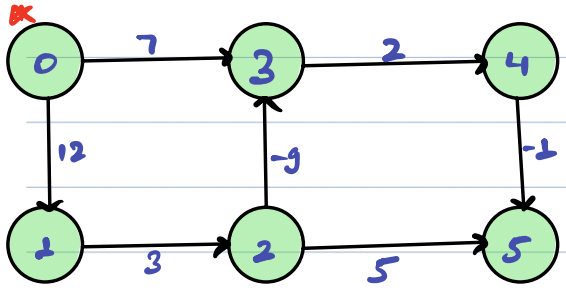
//Pseudo Code

```
int[] Bellmanford (int[E][3] edges, int src, int v ){

        int ans[v] = {∞};
          ans[src] = 0;

        for (int i=0; i<v-1; i++ ){ → iterations
            for (int j=0; j<E; j++ ){
                int u= edges[j][0];
                int v= edges[j][1];
                int wt= edges[j][2];
                if (ans[u] != ∞ && ans[u]+wt <ans[v]){
                    ans[v] = ans[u]+wt;
                }
            }
        }
        return ans;

}
```

T.C: O(V*E)

S.C: ~~O(V)~~ O(1)

AlgoPrep



u	v	wt
4	5	-1
2	5	5
3	4	2
2	3	-9
1	2	3
0	1	12
0	3	7

ans

0	1	2	3	4	5
0	12 ~~∞~~	15 ~~∞~~	~~∞~~ 6	8 ~~∞~~	~~∞~~ ~~20~~ 7

→ It is Possible that you will get all your answer in 1st iteration, we do v-1 iterations considering the worst case scenario.

AlgoPrep

Q) Negative wt. cycle detection.

```
void negwtcycledetect (int[E][3] edges, int src, int v ){

        int ans[v] = {∞};
          ans[src] = 0;

        for (int i=0; i<v-1; i++){ → iterations
            for (int j=0; j<E; j++){
                int u= edges[j][0];
                int v= edges[j][1];
                int wt= edges[j][2];
                if(ans[u] != ∞ && ans[u]+wt <ans[v]){
                    ans[v] = ans[u]+wt;
                }
            }
        }
        boolean flag = false;
        for (int j=0; j<E; j++){
            int u= edges[j][0];
            int v= edges[j][1];
            int wt= edges[j][2];
            if(ans[u] != ∞ && ans[u]+wt <ans[v]){
                flag = true; break;
            }
        }
        if (flag == true){ S.O.P("Negative wt. cycle");}
}
```

T.C: O(V*E)

S.C: O(V) O(1)

AlgoPrep