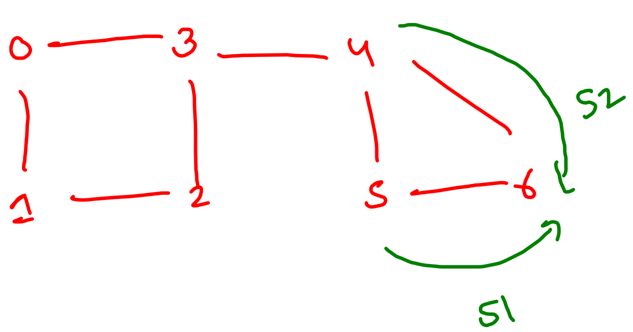
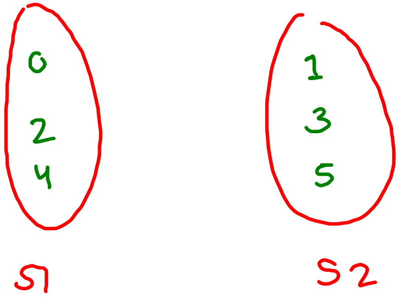


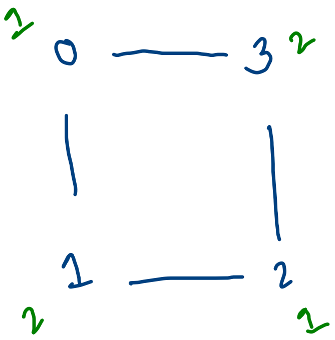


non-bipartite graph

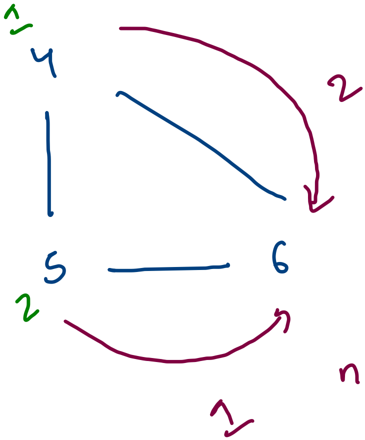
$S1 \cup S2 =$ all vertices

$S1 \cap S2 = \phi$

all edges Across the set



bipartite



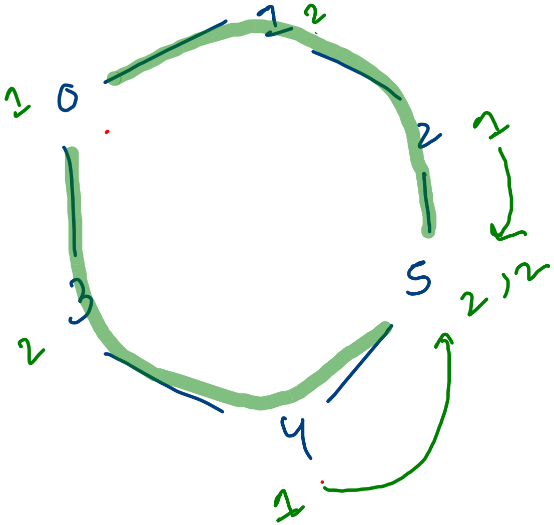
non-bipartite

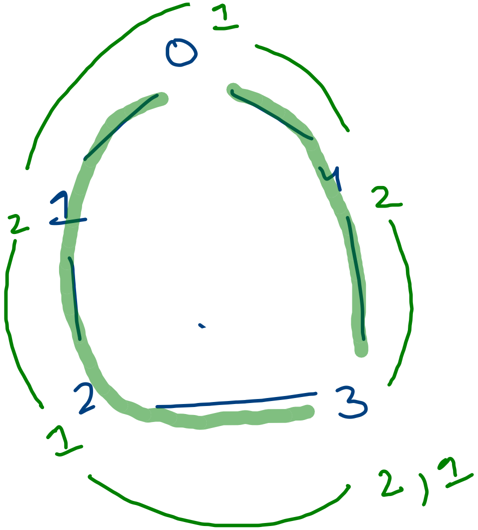
Acyclic graph is always bipartite

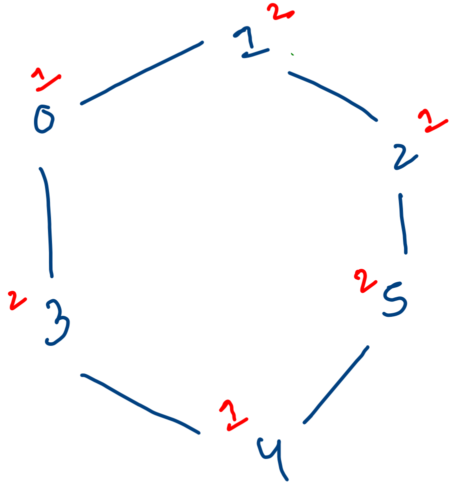
→ Acyclic graph → bipartite

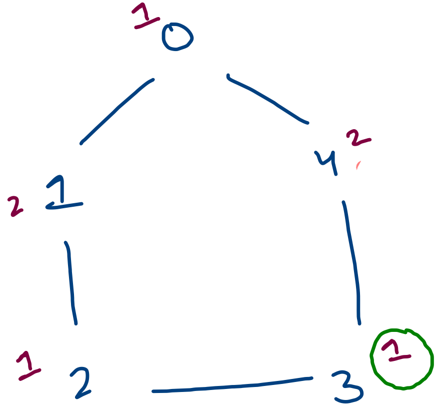
→ cyclic graph → cycle — odd (non - bipartite)
why? — even (bipartite)

→ A graph is bipartite, when all comps are bipartite









~~0-1~~	~~1-2~~	~~3-2~~	~~2-1~~	~~4-1~~	~~5-2~~	~~5-2~~

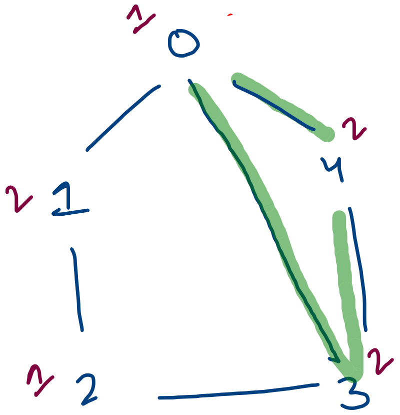
~~0-1~~	~~1-2~~	~~4-2~~	~~2-1~~	~~3-1~~	~~3-2~~

remove, mark*, work, add nbr*

Pair ;

int v

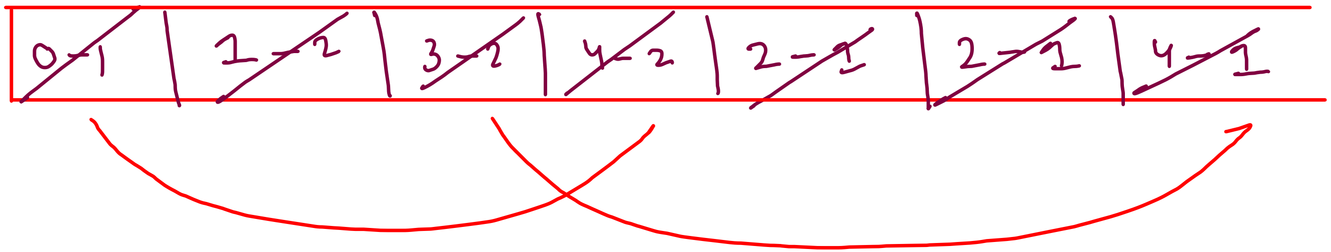
int s



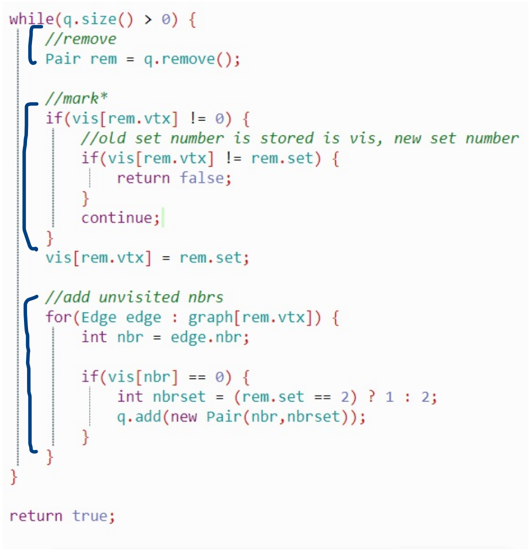
remove

mark*

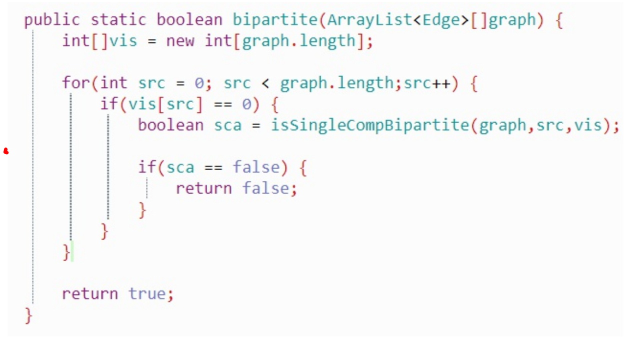
add nbr



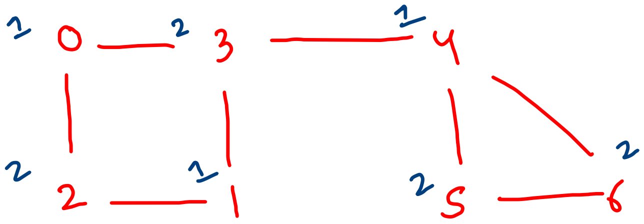
~~0-1~~	~~1-2~~	~~3-2~~	~~4-2~~	~~2-1~~	~~2-1~~	~~4-1~~

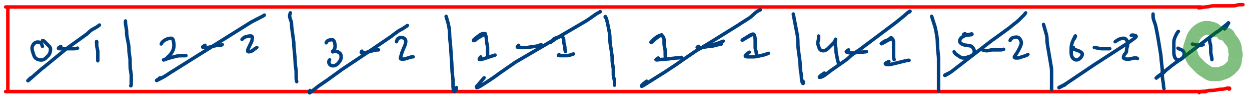


```
while(q.size() > 0) {
    //remove
    Pair rem = q.remove();

    //mark*
    if(vis[rem.vtx] != 0) {
        //old set number is stored is vis, new set number
        if(vis[rem.vtx] != rem.set) {
            return false;
        }
        continue;
    }
    vis[rem.vtx] = rem.set;

    //add unvisited nbrs
    for(Edge edge : graph[rem.vtx]) {
        int nbr = edge.nbr;

        if(vis[nbr] == 0) {
            int nbrset = (rem.set == 2) ? 1 : 2;
            q.add(new Pair(nbr,nbrset));
        }
    }
}

return true;
```



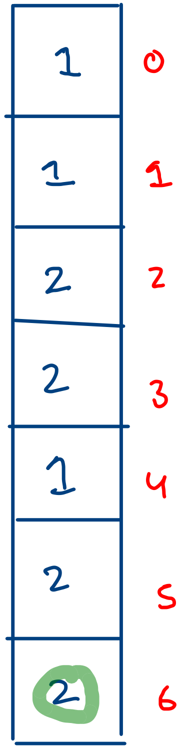
```
public static boolean bipartite(ArrayList<Edge>[]graph) {
    int[]vis = new int[graph.length];

    for(int src = 0; src < graph.length;src++) {
        if(vis[src] == 0) {
            boolean sca = isSingleCompBipartite(graph,src,vis);

            if(sca == false) {
                return false;
            }
        }
    }

    return true;
}
```



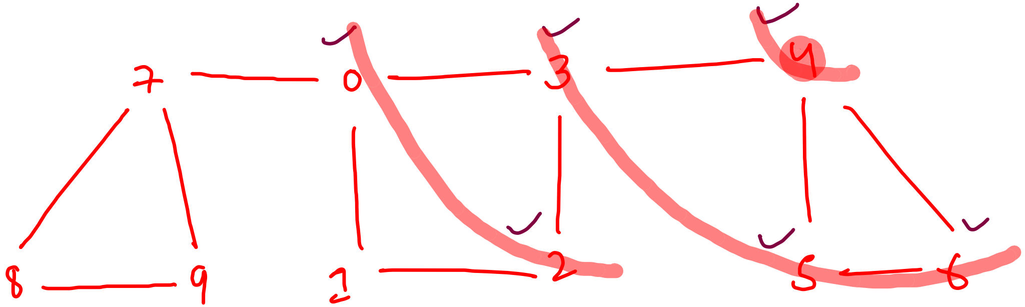


~~0-1~~ | ~~2-2~~ | ~~3-2~~ | ~~1-1~~ | ~~1-1~~ | ~~4-1~~ | ~~5-2~~ | ~~6-2~~ | ~~6-1~~

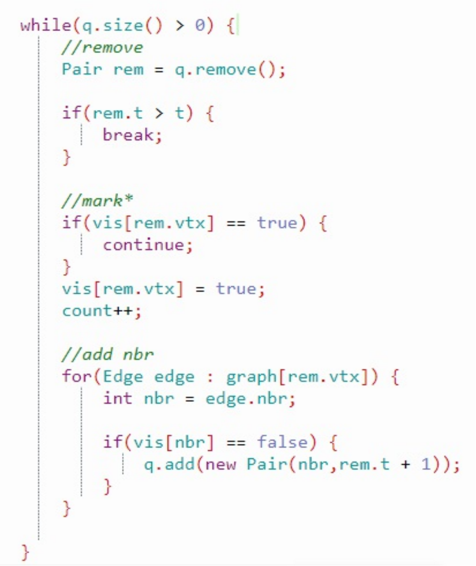
non-bipartite



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



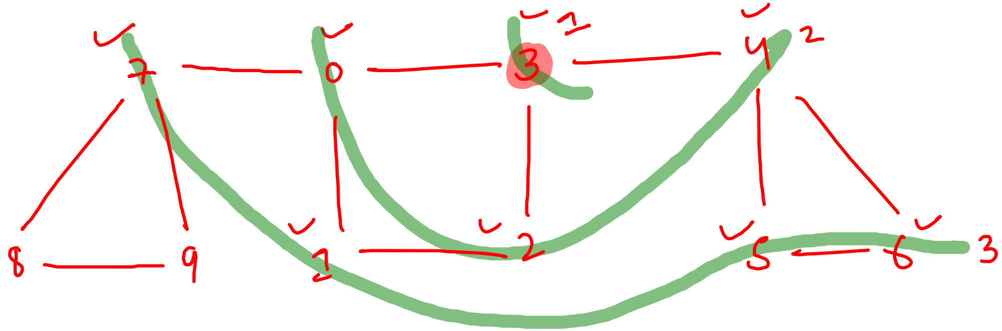
t = 3



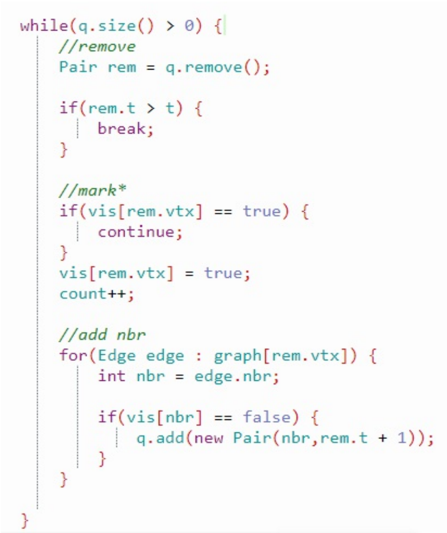
```
while(q.size() > 0) {
    //remove
    Pair rem = q.remove();

    if(rem.t > t) {
        break;
    }

    //mark*
    if(vis[rem.vtx] == true) {
        continue;
    }
    vis[rem.vtx] = true;
    count++;

    //add nbr
    for(Edge edge : graph[rem.vtx]) {
        int nbr = edge.nbr;

        if(vis[nbr] == false) {
            q.add(new Pair(nbr,rem.t + 1));
        }
    }

}
```

~~4-1~~	~~3-2~~	~~5-2~~	~~6-2~~	~~0-3~~	~~2-3~~	~~6-3~~	~~1-4~~	7-4	1-4

count = ~~0~~ ~~1~~ ~~2~~
~~3~~
~~4~~
~~5~~
6



t = 3



```
while(q.size() > 0) {
    //remove
    Pair rem = q.remove();

    if(rem.t > t) {
        break;
    }

    //mark*
    if(vis[rem.vtx] == true) {
        continue;
    }
    vis[rem.vtx] = true;
    count++;

    //add nbr
    for(Edge edge : graph[rem.vtx]) {
        int nbr = edge.nbr;

        if(vis[nbr] == false) {
            q.add(new Pair(nbr,rem.t + 1));
        }
    }
}
```

~~3-1~~	~~0-2~~	~~2-2~~	~~4-2~~	~~1-3~~	~~7-3~~	~~1-3~~	~~5-3~~	~~6-3~~	~~8-4~~	9-4	6-4

Count = ~~0~~ ~~1~~
~~2~~ ~~3~~ ~~4~~
~~5~~
~~6~~ ~~7~~
8

remove
check
mark*
count++
add unvisitd nbr

Priority queue

pq.add(10);

pq.add(5);

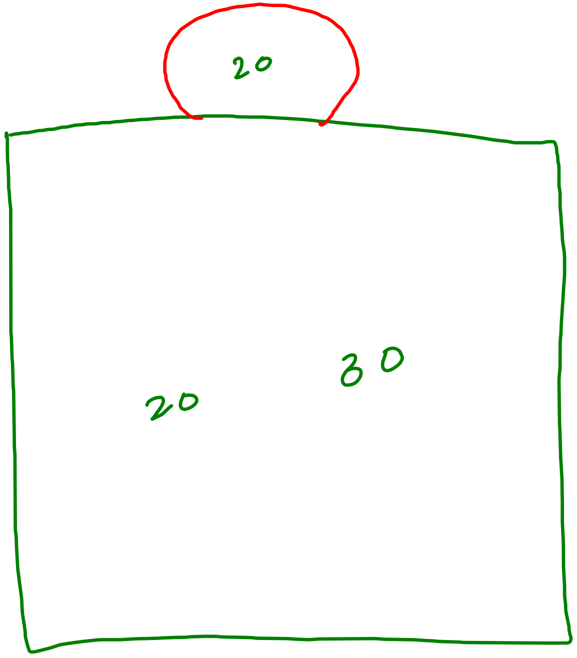
pq.add(20);

pq.add(30);

pq.remove() -> 5

pq.remove() -> 10

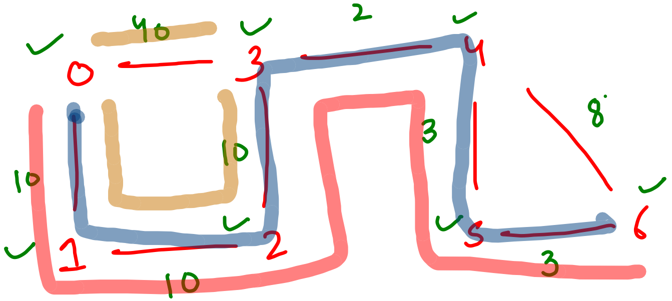
pq.peek() -> 20



min heap

smallest value

highest priority



src = 0

priority queue
(wsf)

(dest - psf - wsf)

~~0 - 0 - 0~~

~~1 - 01 - 10~~

~~3 - 03 - 40~~

~~2 - 012 - 20~~

~~3 - 0123 - 30~~

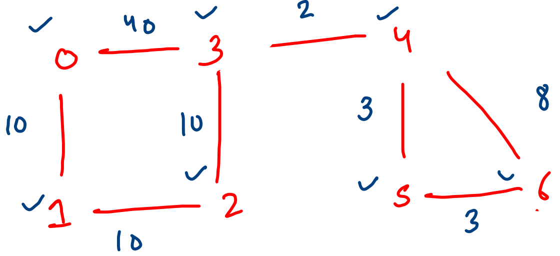
~~4 - 01234 - 32~~

~~5 - 012345 - 35~~

~~6 - 012346 - 40~~

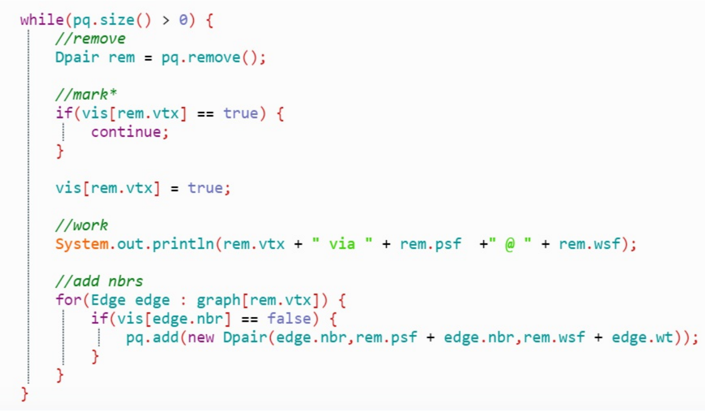
~~6 - 0123456 - 38~~

vtx	path	wsf
✓ 0	via 0	@ 0
✓ 1	via 01	@ 10
✓ 2	via 012	@ 20
✓ 3	via 0123	@ 30
✓ 4	via 01234	@ 32
✓ 5	via 012345	@ 35
✓ 6	via 0123456	@ 38

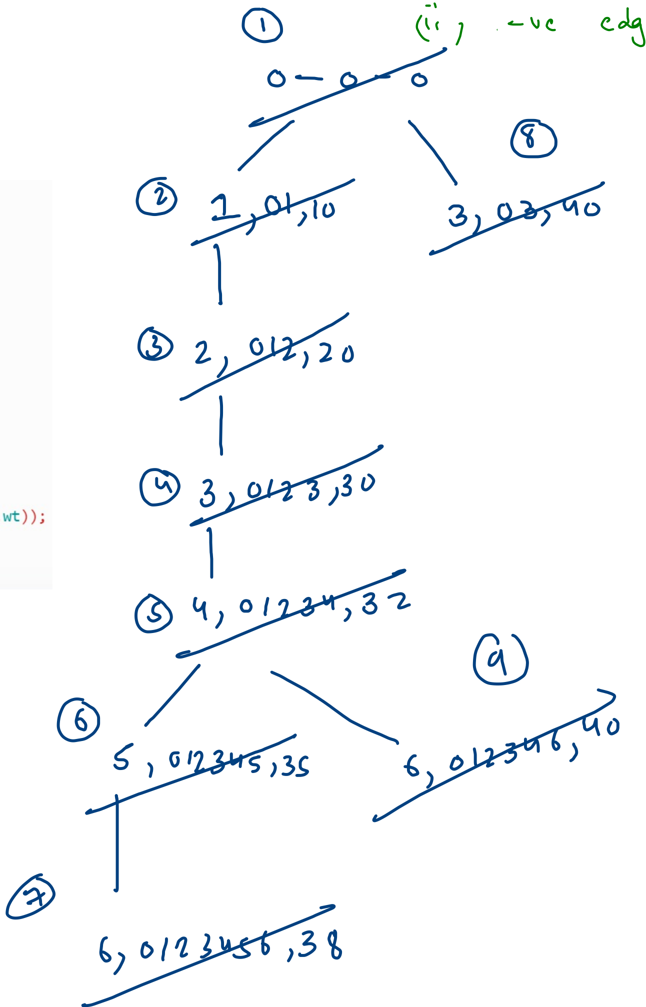


(i) Single src all dest shortest path (wt).

(ii) -ve edge wt X



```
while(pq.size() > 0) {
    //remove
    Dpair rem = pq.remove();

    //mark*
    if(vis[rem.vtx] == true) {
        continue;
    }

    vis[rem.vtx] = true;

    //work
    System.out.println(rem.vtx + " via " + rem.psf  +" @ " + rem.wsf);

    //add nbrs
    for(Edge edge : graph[rem.vtx]) {
        if(vis[edge.nbr] == false) {
            pq.add(new Dpair(edge.nbr,rem.psf + edge.nbr,rem.wsf + edge.wt));
        }
    }
}
```



0 via 0 @ 0

1 via 01 @ 10

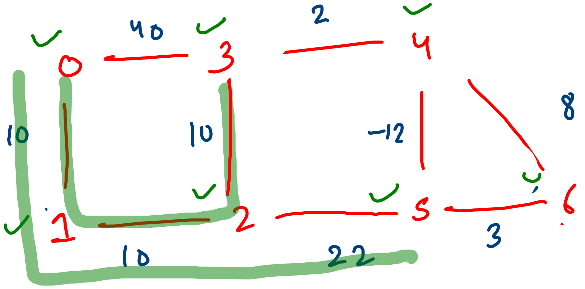
2 via 012 @ 20

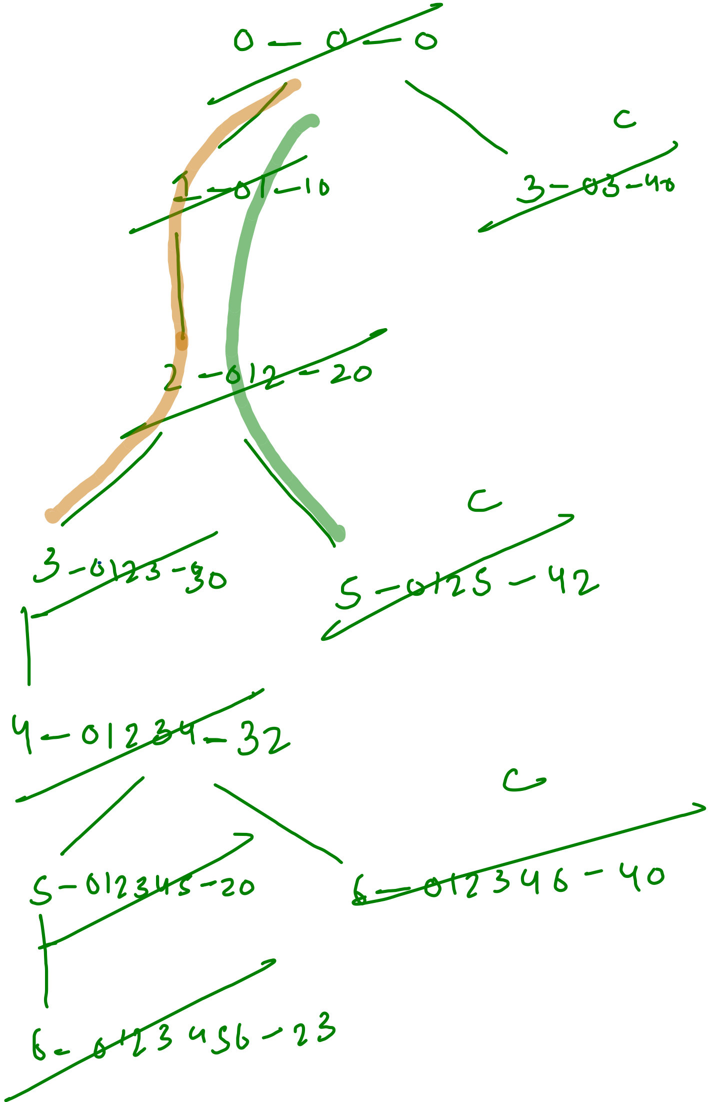
3 via 0123 @ 30

4 via 01234 @ 32

5 via 012345 @ 35

6 via 0123456 @ 38





0 via 0 @ 0

1 via 01 @ 10

2 via 012 @ 20

3 via 0123 @ 30

4 via 01234 @ 32

5 via 012345 @ 20

6 via 0123456 @ 23