Sliding puzzle

BFS

	0	1	2
0	1 (0)	3 (1)	2 (2)
1	4 (3)	5 (4)	0 (5)

→

1 2 3

4 5 0

nbr : [[1,3] , [0,2,4], [1,5], [0,4], [1,3,5], [2,4]]

0 1 2 3 4 5

132450	130452	132540	103452	013452	153402	401352

0 1 2

153042	153420

dest -> 123450

132450

130452

132540 401352

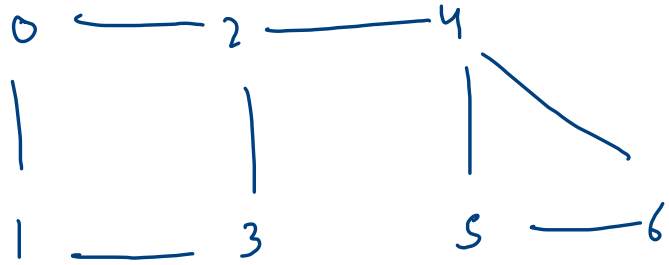
103452

013452 153402

153042

153420

dw = 3



travelling

mark when remove

remove

mark *

work

add unvis nbr

pass

said: cycle,

bipartite,

dijkstra,

prim

mark while adding

remove

work

add unvis nbr as well as

mark them.

```
int[][]nbr = {{1,3},{0,2,4},{1,5},{0,4},{1,3,5},{2,4}};
```

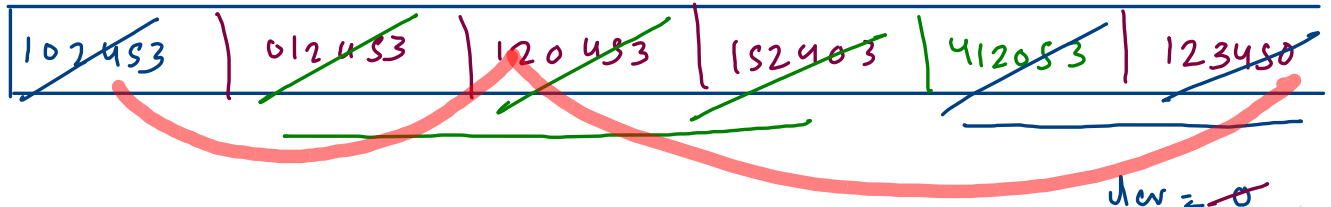
```
while(q.size() > 0) {
    int count = q.size();

    while(count-- > 0) {
        //remove
        String rem = q.remove();

        //work
        if(rem.equals(dest) == true) {
            return lev;
        }

        //add unvisited nbr

        //1. find 0's index
        int idx = -1;
        for(int i=0; i < rem.length();i++) {
            if(rem.charAt(i) == '0') {
                idx = i;
                break;
            }
        }

        //2. go to nbr
        for(int i=0; i < nbr[idx].length;i++) {
            int nbri = nbr[idx][i];

            //swap
            String nbrs = findNbrAfterSwapping(rem,idx,nbri);

            if(vis.contains(nbrs) == false) {
                q.add(nbrs);
                vis.add(nbrs);
            }
        }

    }

    lev++;
}
```

1 0 2

4 5 3

src = 102453

dest : 123450

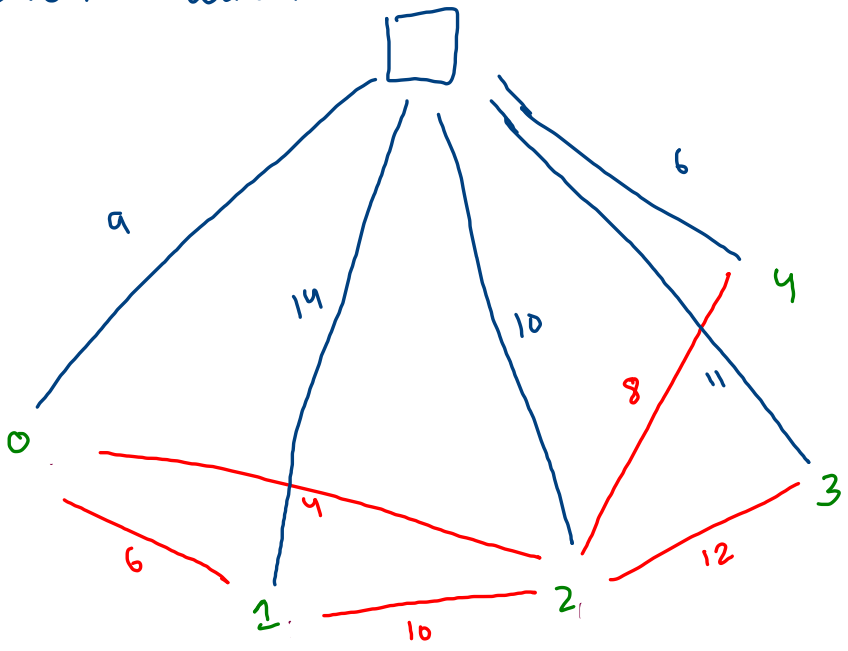


lev = ~~0~~ ~~1~~ 2

hs

102453	152403
012453	412053
120453	123450

optimize water distribution :



cost : 9 14 10 11 6
(well)

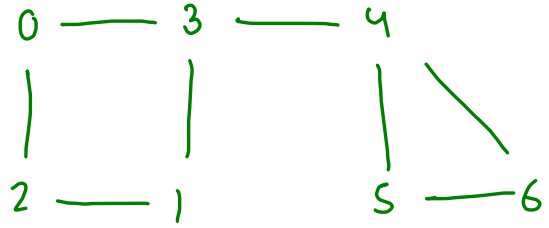
pipes :

0	1	6
0	2	4
1	2	10
2	3	12
2	4	8

atleast 1 well dig

Eulerian path and circuit; (based edges)

You don't need to read or print anything. Your task is to complete the function **isEulerCircuilt()** which takes number of vertices in the graph denoting as V and adjacency list of graph denoting as adj and returns 1 if graph contains Eulerian Path, 2 if graph contains Eulerian Circuit 0 otherwise.



1 -> eulerian path

2 -> eulerian circuit

0 -> otherwise

undirected graph : (i) eulerian circuit

each vertex degree -> even

(ii) eulerian path (n-2) vertices even and

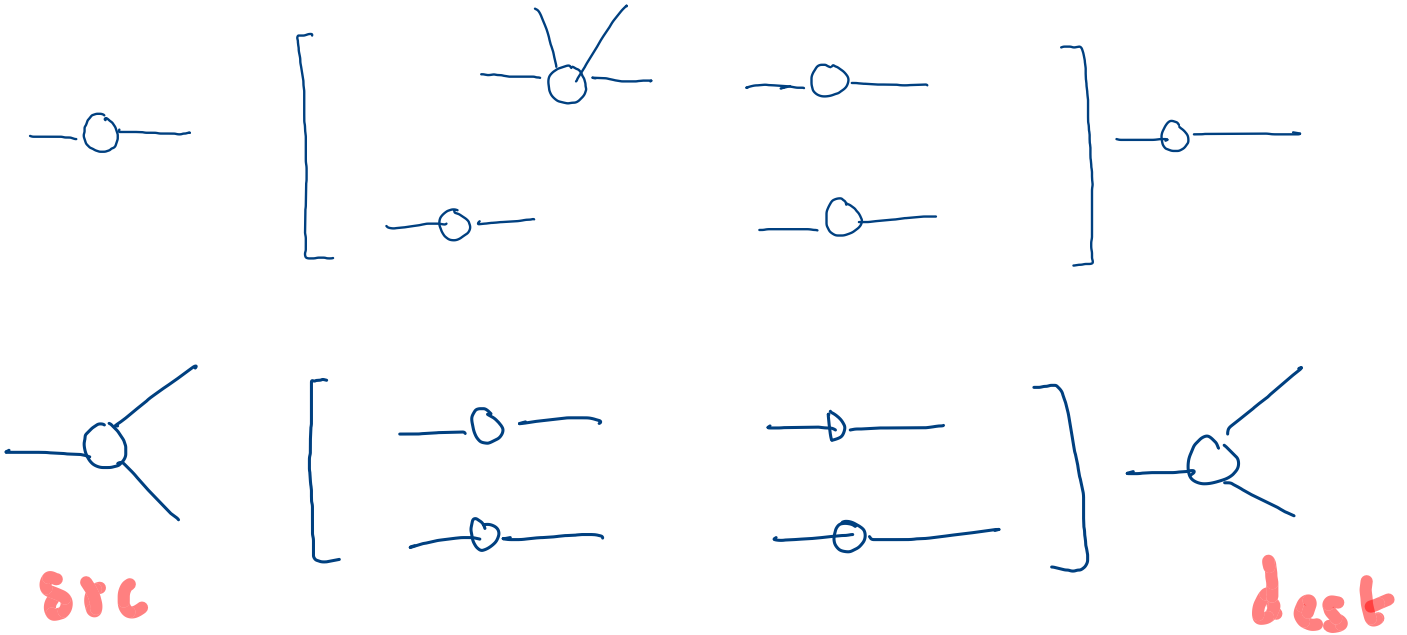
2 vertices have odd degree.

undirected graph : (i) eulerian circuit

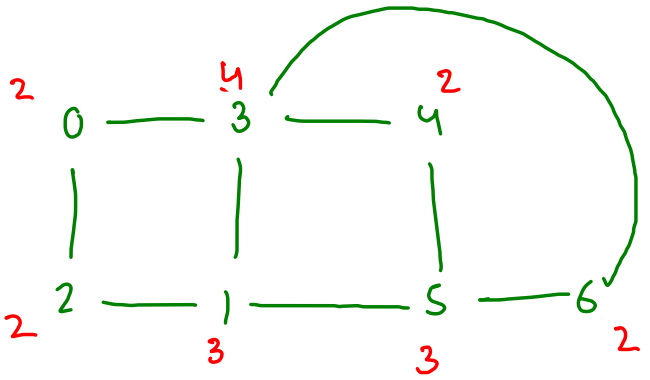
each vertex degree -> even

(ii) eulerian path (n-2) vertices even and

2 vertices have odd degree.

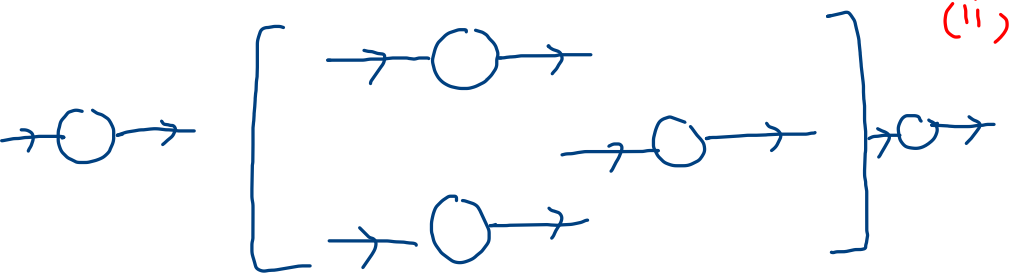


eulerian path


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

direction graph (i) eulerian circuit

all vertices have equal indegree outdegree



(ii) eulerian path

(n-2) vertices should have indeg == outdeg

src v -> outdegree = indeg + 1

dest v -> indeg = outdeg + 1