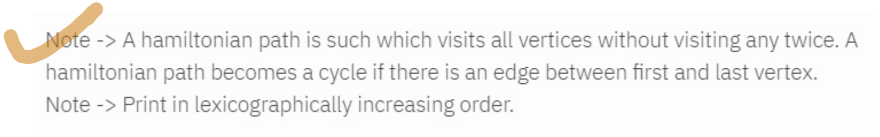
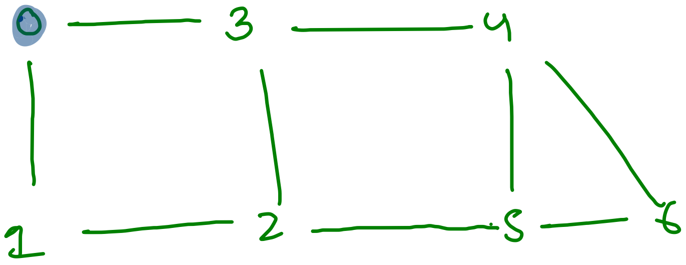

Note -> A hamiltonian path is such which visits all vertices without visiting any twice. A hamiltonian path becomes a cycle if there is an edge between first and last vertex.
Note -> Print in lexicographically increasing order.

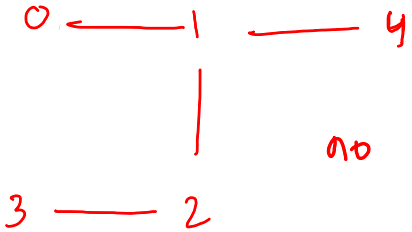


0 3 4 6 5 2 1 *

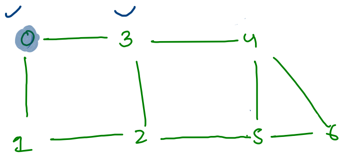
0 1 2 3 4 5 6 .

0 1 2 5 6 4 3 *

0 1 2 3 4 6 5 .

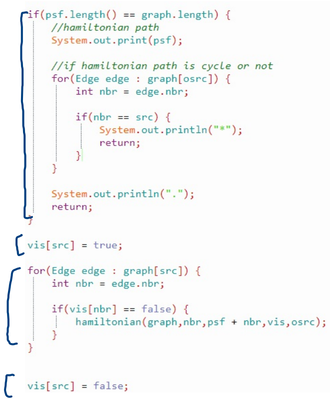


no hamiltonian path



osrc = 0

graph.len = 7



S

```
if(psf.length() == graph.length) {
    //hamiltonian path
    System.out.print(psf);

    //if hamiltonian path is cycle or not
    for(Edge edge : graph[osrc]) {
        int nbr = edge.nbr;

        if(nbr == src) {
            System.out.println("*");
            return;
        }
    }

    System.out.println(".");
    return;
}
```

M

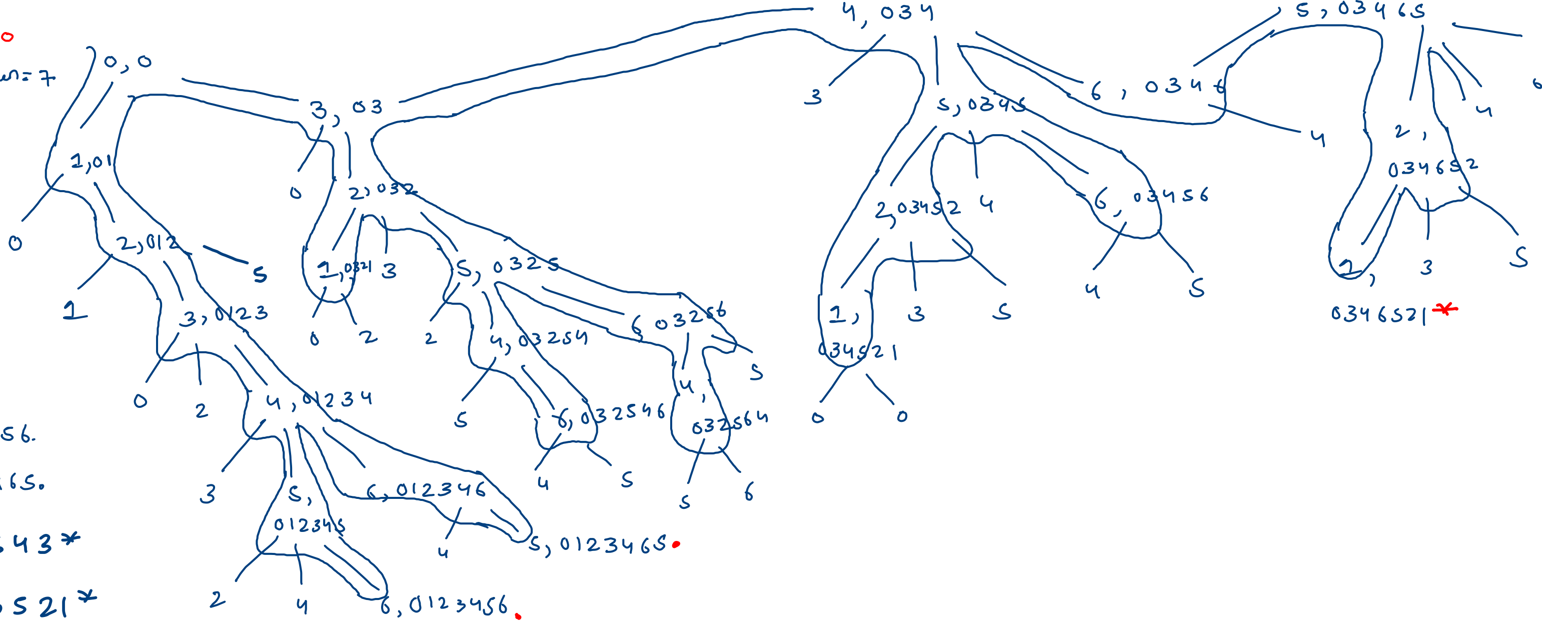
```
vis[src] = true;

for(Edge edge : graph[src]) {
    int nbr = edge.nbr;

    if(vis[nbr] == false) {
        hamiltonian(graph,nbr,psf + nbr,vis,osrc);
    }
}
```

O

```
vis[src] = false;
```

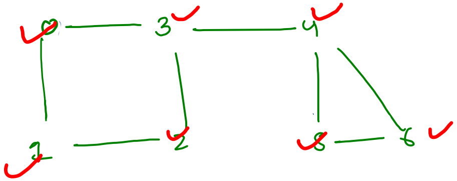


0123456.

0123465.

0125643*

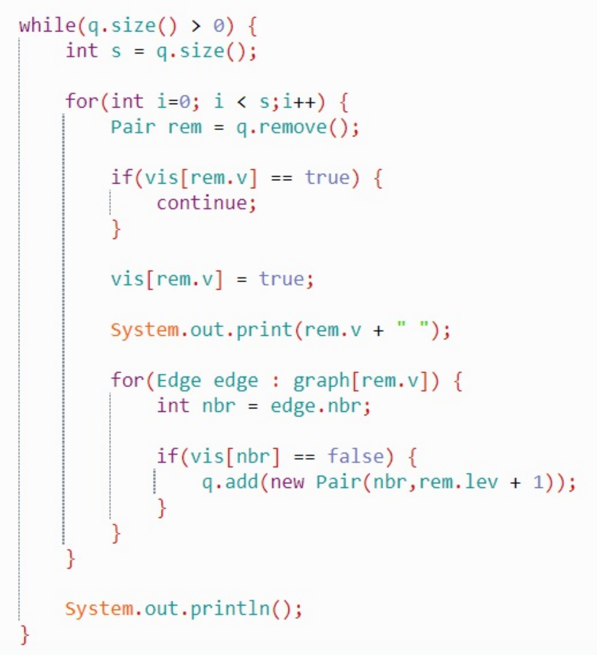
0346521*



src, dest

s = 1

~~0,0~~	~~1,1~~	~~3,1~~	~~2,2~~	~~2,2~~	~~4,2~~	~~5,3~~	~~6,3~~	~~6,4~~



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

    for(int i=0; i < s;i++) {
        Pair rem = q.remove();

        if(vis[rem.v] == true) {
            continue;
        }

        vis[rem.v] = true;

        System.out.print(rem.v + " ");

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

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

    System.out.println();
}
```

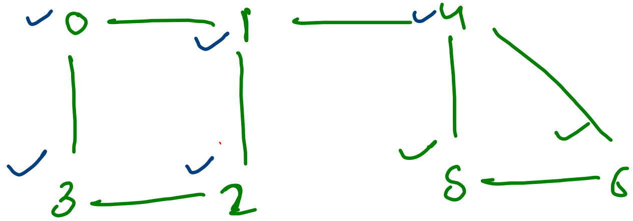
rem, mark*, work, add* nbr

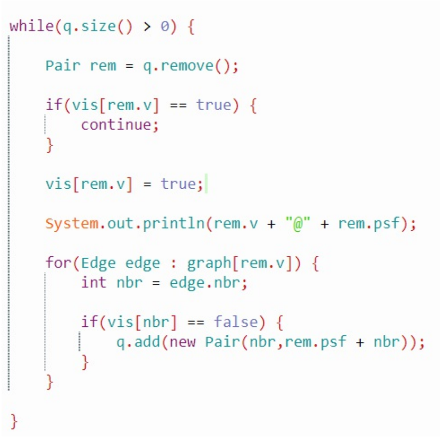
→ 0

→ 1 3

→ 2 4

→ 5 6





```
while(q.size() > 0) {

    Pair rem = q.remove();

    if(vis[rem.v] == true) {
        continue;
    }

    vis[rem.v] = true;

    System.out.println(rem.v + "@" + rem.psf);

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

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

}
```

remove, mark*, work,

add nbr*.

2 @ 2

1 @ 21

3 @ 23

0 @ 210

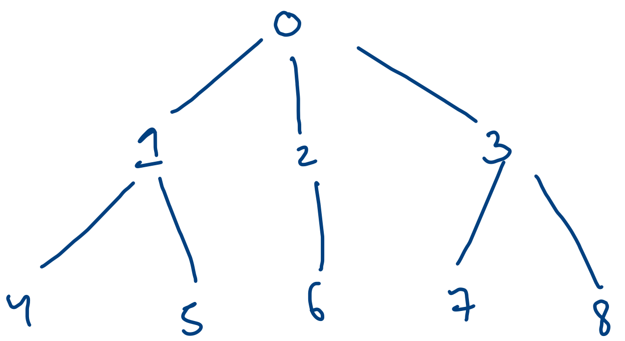
4 @ 214

5 @ 2145

6 @ 2146

~~2, 2~~	~~1, 21~~	~~3, 23~~	~~0, 210~~	~~4, 214~~	~~0, 230~~	~~5, 2145~~	~~6, 2146~~	~~6 @ 21456~~

Trees



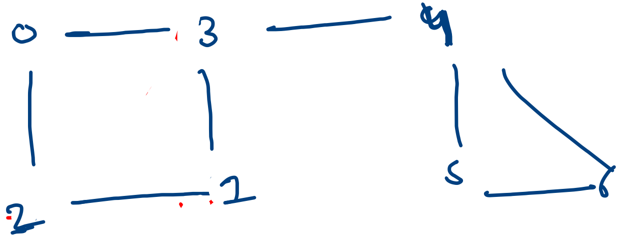
lo : 0 1 2 3 4 5 6 7 8

lo lw : 0

1 2 3

4 5 6 7 8

graphs



src = 2

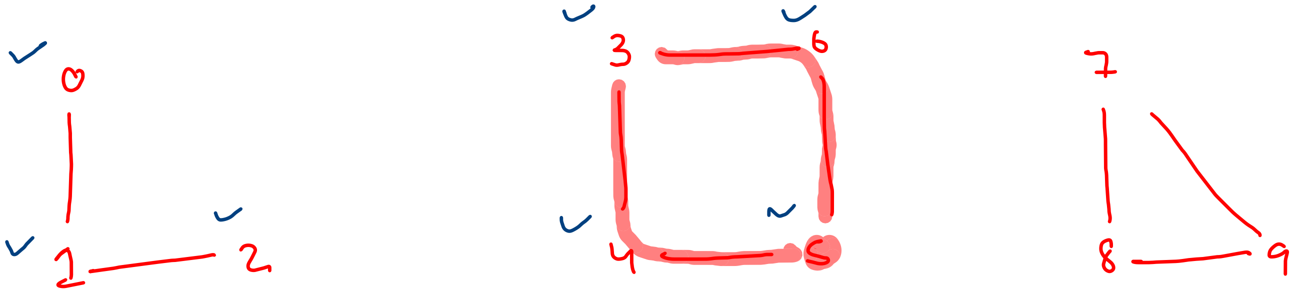
lo : 2 1 3 0 4 5 6

lo lw : 2

1 3

0 4

5 6



~~0~~	~~1~~	~~2~~	

~~3~~	~~4~~	~~6~~	~~5~~	~~5~~

False

true