Lecture :- Graphs 4

Agenda

- Rotten oranges
- Graph coloring
- Bipartite graph
- Construct graph.

Qu Given a matrix of integers.

A[i][j]
- 0 → empty
- 1 → fresh
- 2 → rotten

Every minute a fresh orange adjacent to a rotten orange becomes rotten. find the time when all the oranges become rotten. If not possible, return -1.

Eg:

0	1	0	0
1	1	0	1
2	0	0	2

$t = \emptyset \times \not{2}\ 3$

1	2	1	0	1
1	0	0	0	2
0	1	0	0	1
0	1	2	1	1

$t = \emptyset \times 2$ Ans

Approach

	0	1	2	3	4
0	+2	2	+2	0	+2
1	+2	0	0	0	2
2	0	+2	0	0	+2
3	0	+2	2	+2	+2

t = 0 / 1 / 2 Ans

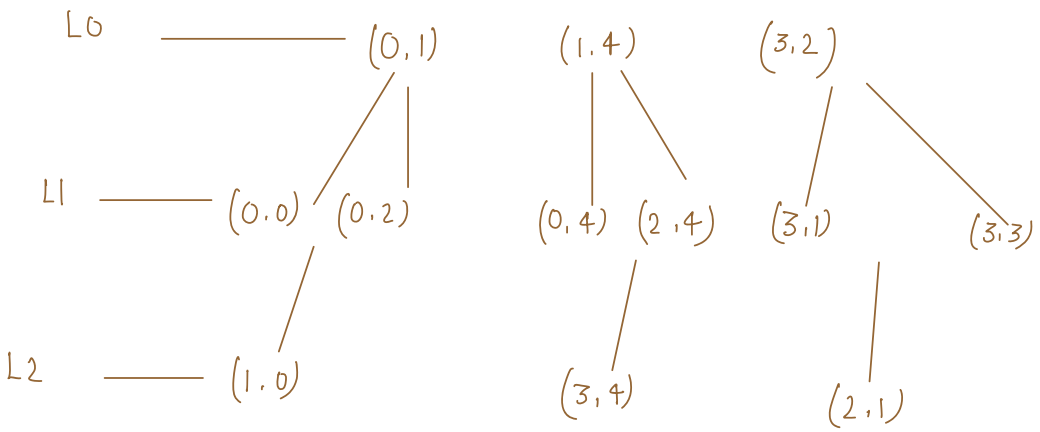
Queue:

~~(0,1)~~ ~~(1,4)~~ ~~(3,2)~~ ~~(0,0)~~ ~~(0,2)~~ ~~(0,4)~~ ~~(2,4)~~

~~(3,1)~~ ~~(3,3)~~ ~~(1,0)~~ ~~(3,4)~~ ~~(2,1)~~

↓

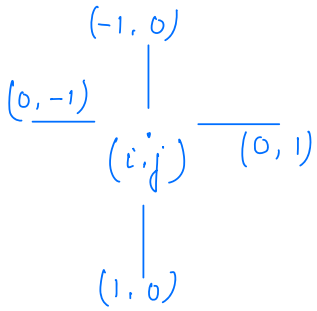
Multisource bfs



Code

```
int minNoofDays( mat[][] ) {
    r = mat.length;
    c = mat[0].length;
    int days = 0;
    int ones = 0;
    Queue<Pair> q;
    for(i=0; i<r; i++) {
        for(j=0; j<c; j++) {
            if (mat[i][j]==2) {
                q.add( new Pair(i,j));
            } else if( mat[i][j]==1) {
                ones ++;
            }
        }
    }
    if( ones==0) {
        return 0;
    }
    while( !q.isEmpty()) {
        n = q.size();
        while( n > 0) {
            Pair p = q.poll();
            i = p.x;
            j = p.y;
            for(k=0; k<4; k++) {
                ni = i + row[k];
                nj = j + col[k];
                if(ni<0 || nj<0 || ni>=r || nj>=c) {
                    continue;
                }
                if( mat[ni][nj]==1) {
                    q.add( new Pair(ni,nj));
                    mat[ni][nj] = 2;
                    ones--;
                }
            }
            n--;
        }
        if( q.size() > 0) {
            days ++;
        }
    }
    return ones==0 ? days : -1;
```

```
class Pair {
    int x;
    int y;
}
```



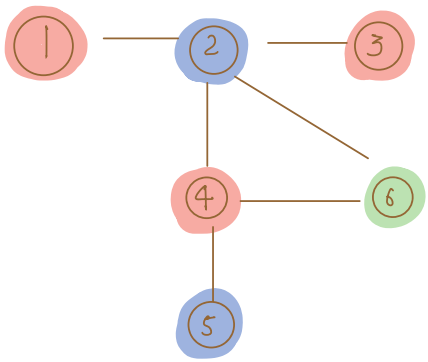
row[] = { 0, -1, 0, 1 }

col[] = { -1, 0, 1, 0 }

Graph coloring

Colour all nodes of a graph such that no two adjacent nodes are of same colour and no. of distinct colours is minimum.

↓ Chromatic number.

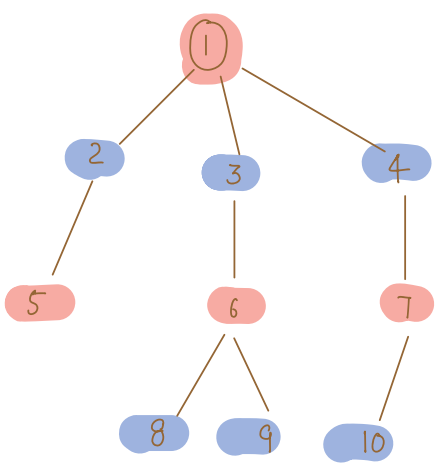


ans = 3

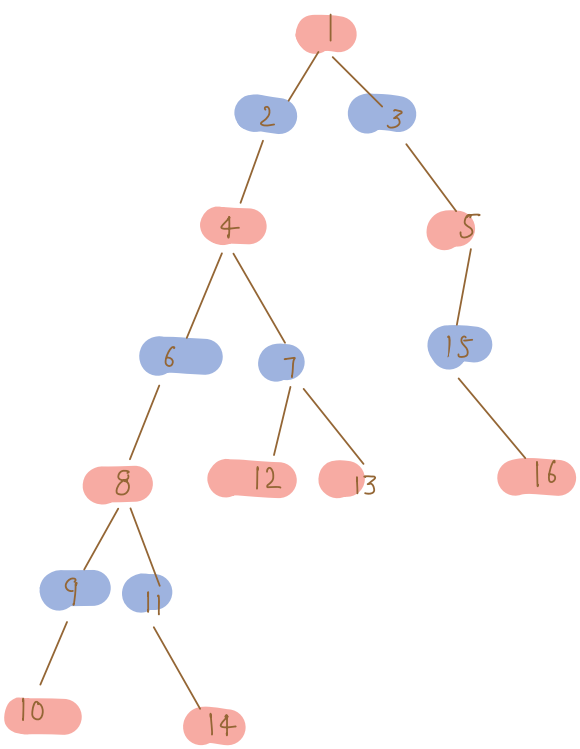
CN = 3

Special cases

1.) Trees — Chromatic number = 2.



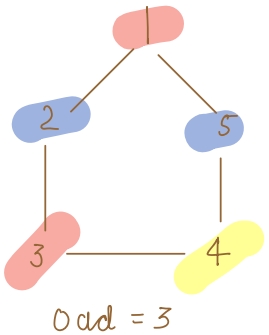
CN = 2.



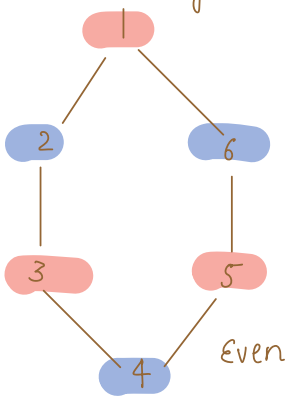
CN = 2

2.) **Cyclic graph.** $CN = 2 + (n \,\%\, 2)$

↑ no. of nodes in a cycle

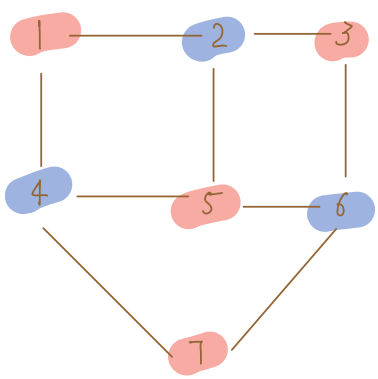


Odd = 3



Even = 2

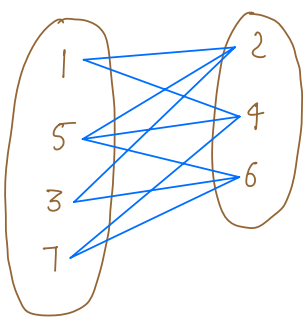
Bipartite graph Chromatic no = 2.



CN = 2

→ Can we colour our graph using two different colours ?

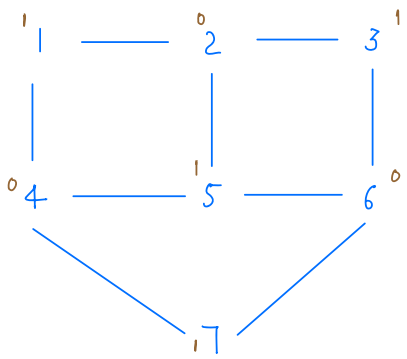
→ Nodes can be divided into two sets, such that there is no edge b/w nodes of same set.



Qu Check if given graph is bipartite ?

red = 0

blue = 1



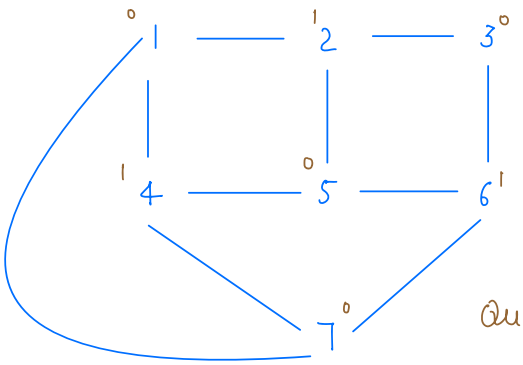
color[] =

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

Queue:

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

true



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

Queue:

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

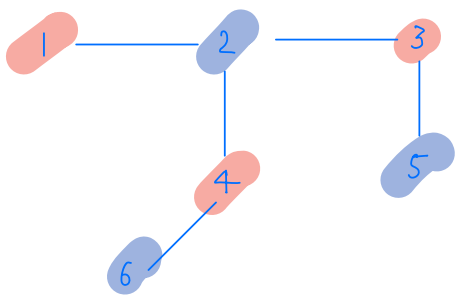
false

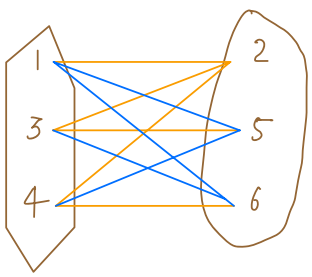
Code

```
bool isBipartiteGraph(src, List<Integer> graph[]) {

    Queue<Integer> q;
    q.add(src);

    color[src] = 0;

    while(!q.isEmpty()) {

        int curr = q.poll();

        for(i=0; i< graph[curr].size(); i++) {

            u = graph[curr].get(i);

            if(colour[u] == -1) {

                colour[u] = 1- colour[curr];

                q.add(u);

            } else if(colour[u] == colour[curr]) {

                return false;

            }

        }

    }

    return true;

}
```

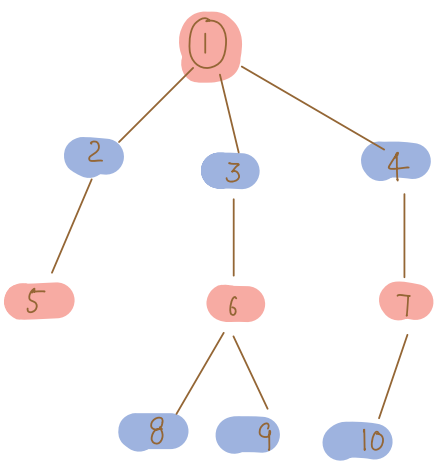
Qu: A country consists of n cities, all are connected by n-1 roads. king of that country want to construct maximum roads such that cities can be divided into 2 sets and there is no road b|w cities in the same set. find max roads the king can construct. | n > 1 |

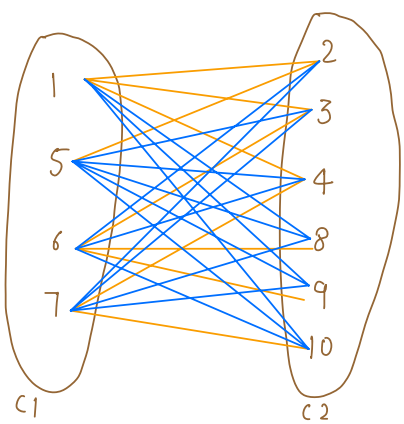
Eg:





ans = 4





ans = (c1 * c2) − [king has already constructed]

ans = (4 * 6) − 9 = 15 Ans

Code

Use previous code

↓

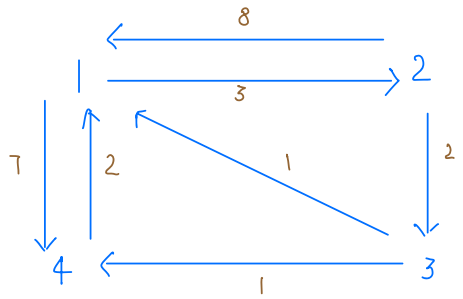
col[ ]

```
int maxRoads( col[ ], edges) {

    c1 = 0;
    c2 = 0;
    for( i=0; i< col.length; i++) {
        if( col[i] == 0) {
            c1 ++;
        } else {
            c2++;
        }
    }
    return (c1 * c2) - (n-1);
}
```

Break: 8:43 - 8:53

flyod warshall Algorithm

All pair shortest path.

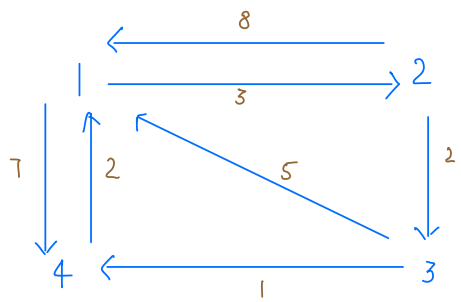


⇒ for all nodes —

Apply dijkastra's

TC: $O(n^3)$

Approach



$A^0 =$

	1	2	3	4
1	0	3	∞	7
2	8	0	2	∞
3	5	∞	0	1
4	2	∞	∞	0

$A^0 =$

	1	2	3	4
1	0	3	∞	7
2	8	0	2	∞
3	5	∞	0	1
4	2	∞	∞	0

$A^1 =$

	1	2	3	4
1	0	3	∞	7
2	8	0	2	15
3	5	8	0	1
4	2	5	∞	0

intermediate node is 1

$A^1[2][3] = 2$

$A[2][1] + A[1][3]$

$8 + \infty \Rightarrow \infty$

$A^1[2][4] = \infty$

$A[2][1] + A[1][4]$

$8 + 7 \Rightarrow 15$

$A^1 =$

	1	2	3	4
1	0	3	∞	7
2	8	0	2	15
3	5	8	0	1
4	2	5	∞	0

intermediate node is 1

$A^2 =$

	1	2	3	4
1	0	3	5	7
2	8	0	2	15
3	5	8	0	1
4	2	5	7	0

intermediate node is 2

$A^2[1][3] = \infty$

$A[1][2] + A[2][3]$

$3 + 2 = 5$

$A^2[1][4] = 7$

$A[1][2] + A[2][4]$

$3 + 15 = 18$

$A^2[3][4] = 1$

$A[3][2] + A[2][4]$

$8 + 15 \Rightarrow 23$

$A^4 =$

$=$

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

intermediate node is 4

H|w

Ans

Thankyou ☺