Graphs

Friday, 24 September 2021 11:47 PM

Total No. of questions + Algo = 41 → Total

No. of questions per day = 4 questions.

Total no. of days = 10 days.

2 weeks + 1 class

complexity →
1. DFS
2. BFS
3. All path] pure Recursion

Discussion

Day 1 (Morning)

1. Count islands
2. count distinct islands
3. No. of Enclaves
4. Rotting oranges

Day 2 (Evening)

1. Bus Routes
2. Prims → Min cost to connect cities
3. Zero one matrix
4. As far from land as possible

Complexity Analysis

Saturday, 25 September 2021 10:25 AM

```
int n = 7;
ArrayList<Edge>[] graph = new ArrayList[n];
```

Initial state -

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

0 → 0-3 @ 40, 0-1 @ 10

1 → 1-0 @ 10, 1-2 @ 10

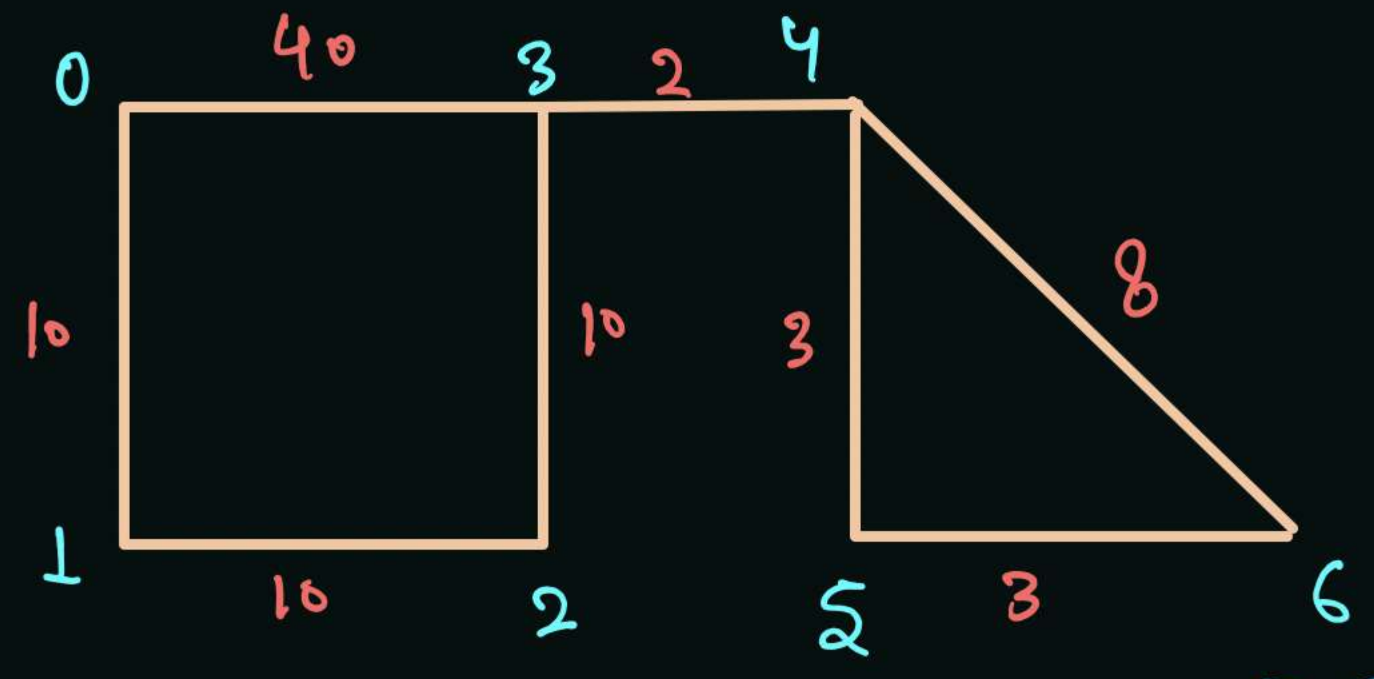
2 → 2-1 @ 10, 2-3 @ 10

3 → 3-0 @ 40, 3-2 @ 10, 3-4 @ 2

4 → 4-3 @ 2, 4-5 @ 3, 4-6 @ 8

5 → 5-4 @ 3, 5-6 @ 3

6 → 6-4 @ 8, 6-5 @ 3



BFS → queue.

1. get + remove
2. mark*
3. work
4. Add neighbour

DFS → stack

v = 0 1 2 3 4 skip

res = res || dfs( );

⇒ logical OR

⇒ to aware from these type of codes

res = False.

v0

v1, v2, v3, v4 — true neighbour

res = False || dfs(from 0) → False

res = False || dfs(from 1) → False

res = False || dfs(from 2) → True.

res = True || dfs(from 3)

res = True || dfs(from 4)

```java
public static boolean dfs(ArrayList<Edge>[] graph, int src, int dst,
    boolean[] vis, String psf) {
    if(src == dst) {
        psf += dst;
        System.out.println(psf);
        return true;
    }

    vis[src] = true;
    boolean res = false;
    for(Edge e : graph[src]) {
        if(vis[e.nbr] == false) {
            res = res || dfs(graph, e.nbr, dst, vis, psf + src);
        }
    }
    return res;
}
```

constant 'C'

Neighbour Ni

Time complexity →

Time = $\underbrace{C + N_i}_{\text{for single src.}} + C + N_i + C + N_i + \cdots$

$\underbrace{\qquad\qquad\qquad\qquad}_{V \text{ time.}}$

Time = $V \cdot C + \underbrace{N_0 + N_1 + N_2 + \cdots N_v}_{N_0 + N_1 + N_2 + \cdots N_v}$

Total No. of Edges.

Time = $V \cdot C + E$

in order $\boxed{T = O(V + E)}$

If E is order of V^2

$= V + V^2$

$= V^2$

```java
public static boolean bfs(ArrayList<Edge>[] graph, int src,
    int dst, boolean[] vis) {
    Queue<BPair> qu = new LinkedList<>();
    qu.add(new BPair(src, 0, "" + src));
    boolean res = false;
    while(qu.size() > 0) {
        // 1. get + remove
        BPair rem = qu.remove();
        // 2. mark *
        if(vis[rem.vtx] == true)
            continue;
        vis[rem.vtx] = true;
        // 3. work
        System.out.println(rem.vtx + " " + rem.psf + " @ " + rem.wsf);
        if(src == dst) {
            res = true;
            break;
        }
        // 4. add neighbours
        for(Edge e : graph[rem.vtx]) {
            if(vis[e.nbr] == false) {
                qu.add(new BPair(e.nbr, rem.wsf + e.wt, rem.psf + e.nbr));
            }
        }
    }
    return res;
```

Constant C

const + C

Ni

For single vertex = $C + N_i$

For V vertex = $C+N_0 + C+N_1 + C+N_2 + \cdots + C+N_{V-1}$

$= V \cdot C + N_0 + N_1 + N_2 + \cdots N_{V-1}$

E

$E = \sum_{i=0}^{V-1} N_i = N_0 + N_1 + \cdots + N_{V-1}$

$\Rightarrow$ Time $= V \cdot C + E$

in order

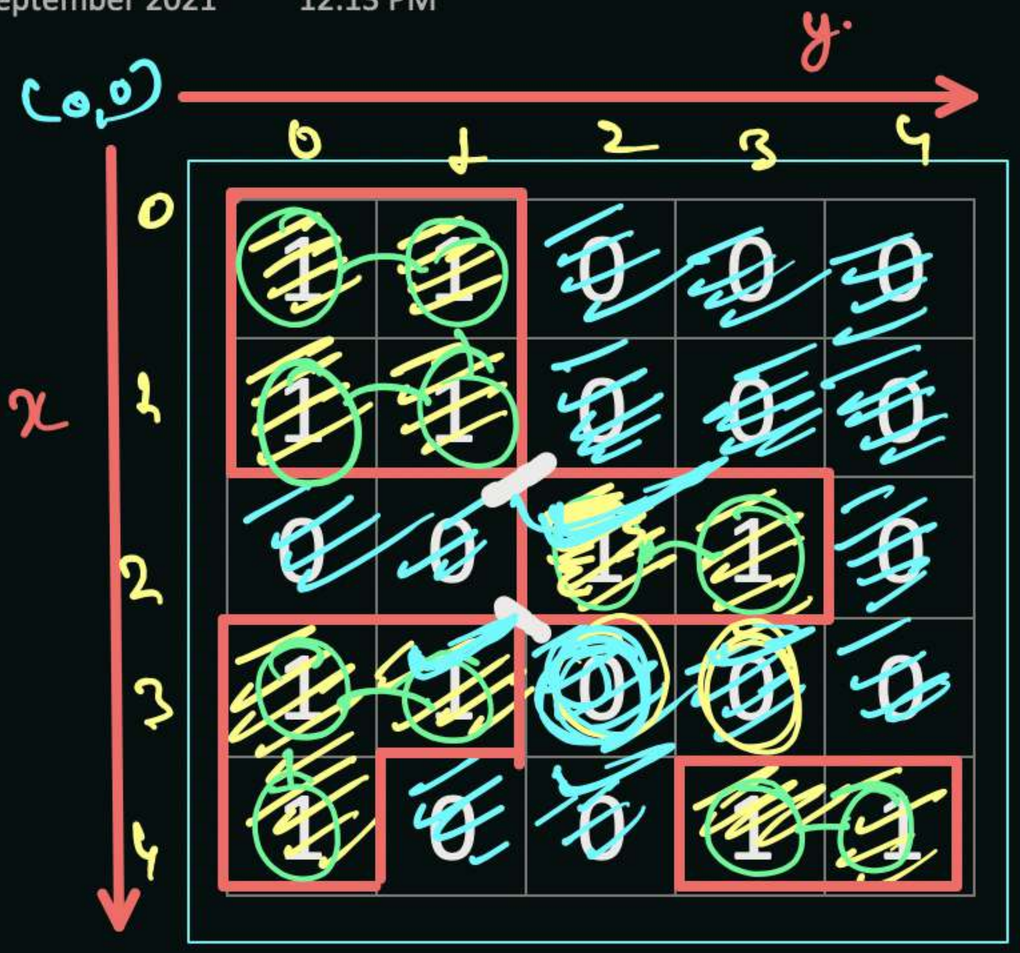
Time $= O(V+E)$

All path = Recursion (DFS)

(No. of call) height = (V) particular case : (Max. no. of neighbour) height of recursive stack (max) conditions

Number of Islands

Saturday, 25 September 2021 12:13 PM



1 → land
0 → water

island ⇒ land surrounded with water.

No. of calls DFS ⇒ 4

```
for(int r=0; r< row; r++){
    for(int c=0; c< col; c++){
        if(grid[i][j]==1){
            DFS( )
            island++;
        }
    }
}
```

(3,2) · 2 2 · 3 1 · 4 2 · 3 3

top → left → Down Right

Coordinate System in terms of matrix →

Top (x-1, y)

left (x, y-1)

(x, y) — radius 1

Right (x, y+1)

Down (x+1, y)

increment / Decrement

xdir = [-1, 0, 1, 0]

ydir = [0, -1, 0, 1]

(T, L, D, R)

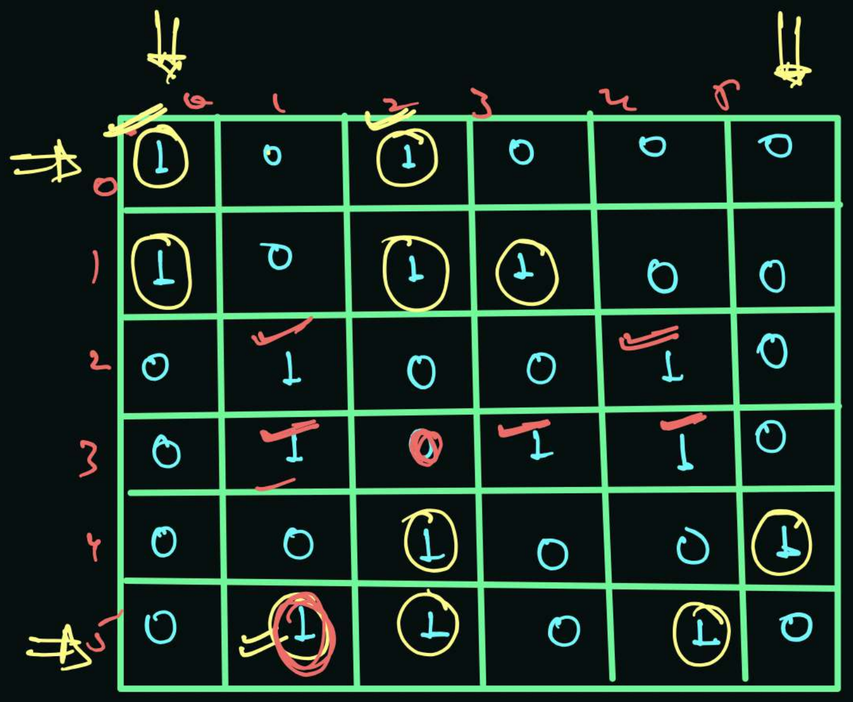
(3,2)

3 + xdir[0] = 2 2 + ydir[0] = 2
3 + xdir[1] = 3 2 + ydir[1] = 1
3 + xdir[2] = 4 2 + ydir[2] = 2
3 + xdir[3] = 3 2 + ydir[3] = 3

Number of Enclaves

Saturday, 25 September 2021 12:43 PM

No. of '1'. such that there is no way to reach boundary from that '1'. We can travel in top, left, Down, Right direction if it have '1'.



count = 5 Result

3,2 to 4,5 Shortest path

(1) Travel on boundary and marked connected 1 from boundary
↳ DFS.

(2) Count Remaining 1.

↳ Result.

only searching is task → BFS or DFS

Shortest path in term of Edge

↳ BFS.

↳ BFS.

Rotting Oranges

Saturday, 25 September 2021 1:05 PM

Rate of rotting : 1 orange per minute in each 4 direction

TLDR

0 → Empty
1 → Fresh
2 → Rotted

Min time to rott all oranges.

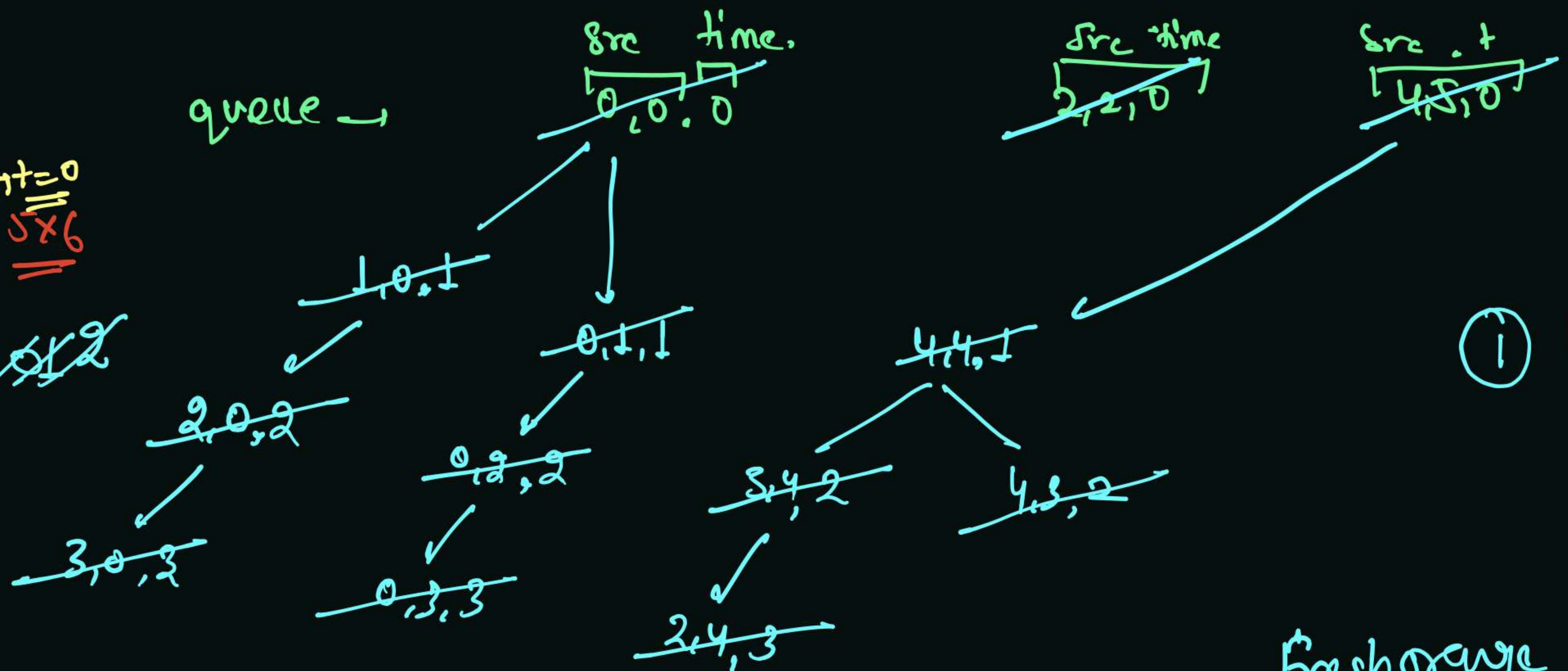
	0	1	2	3	4	5
0	2	1	1	1	0	1
1	1	0	0	0	0	0
2	1	0	2	0	1	0
3	1	0	0	0	1	0
4	0	0	0	1	1	2

→ Remaining

→ t = 0

5×6

queue →



BFS
1) get
2) remove
3) Mark
5) work
6) add nbs

→ max Time ~~5,1,2~~
3

Marking with = -2

1) Manage count of Rotted orange.

Freshorange != 0 ? -1 : maxTime

Number of Distinct Islands

Saturday, 25 September 2021 1:05 PM

DFS

Hint → Hashing

$\Rightarrow$ xrd

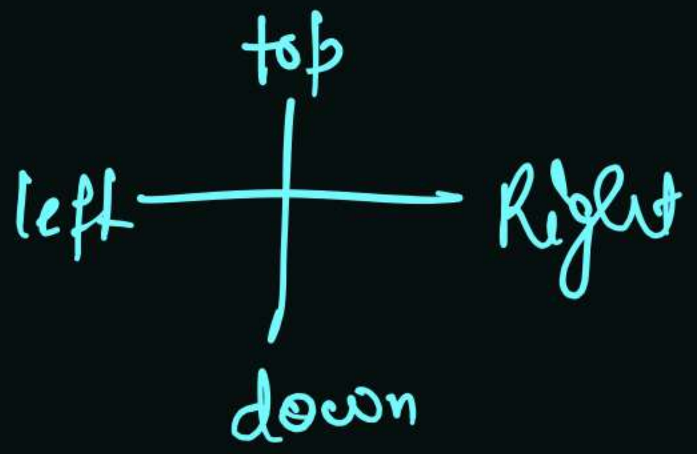
1	1	0	0	0
1	0	0	1	1
0	0	0	1	0
1	1	0	0	0

No. of distinct island = 2

x→r→d = xrd

x → r

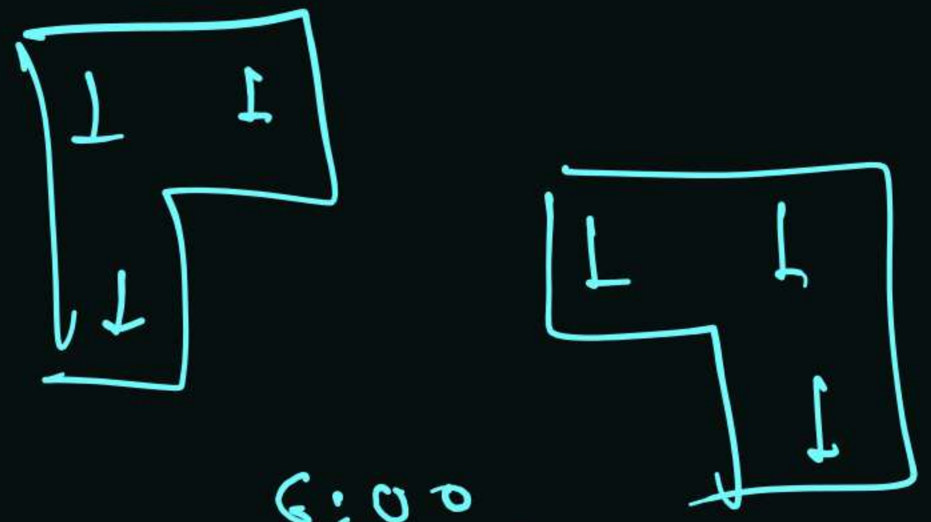
xr



$\Rightarrow$ xrd

$\Rightarrow$ xrd

$\Rightarrow$ xr



6:00

30 min

7:00 - 11:00