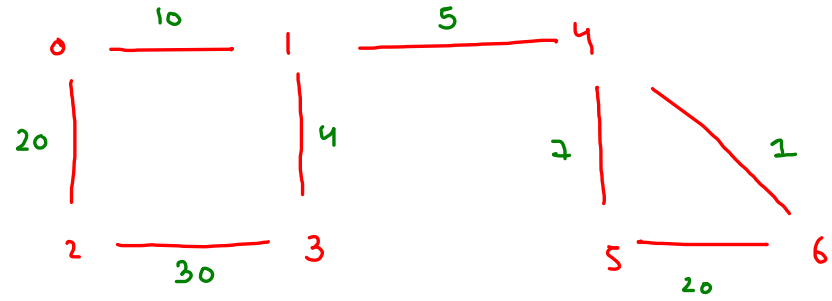
Intro:



Vertices

edges

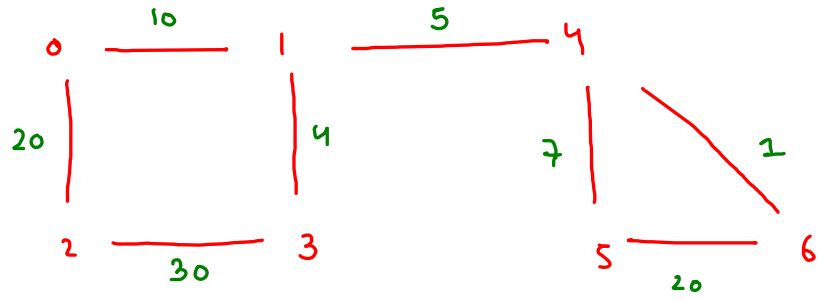
(i) connected comp

(ii) dfs, bfs

(iii) dijkstra (single src all dest)

(iv) MST

(v) undirected vs directed graph



Adjacency matrix

	0	1	2	3	4	5	6
0	∞	10	20	∞	∞	∞	∞
1	10	∞	∞	4	5	∞	∞
2	20	∞	∞	30	∞	∞	∞
3	4	∞	30	∞	∞	∞	∞
4	5	∞	∞	∞	∞	7	1
5	∞	∞	∞	∞	7	∞	20
6	∞	∞	∞	1	∞	20	∞

AL<edge> [ ]

Adjacency list:

0 -> (0,1,10) , (0,2,20)

1 -> (1,0,10) , (1,3,4) , (1,4,5)

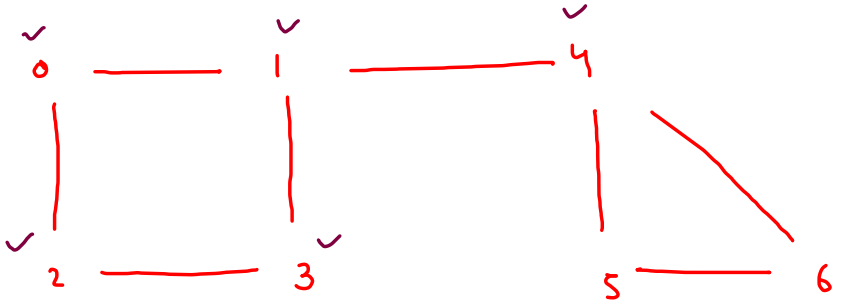
2 -> (2,0,20) , (2,3,30)

3 -> (3,2,30) , (3,1,4)

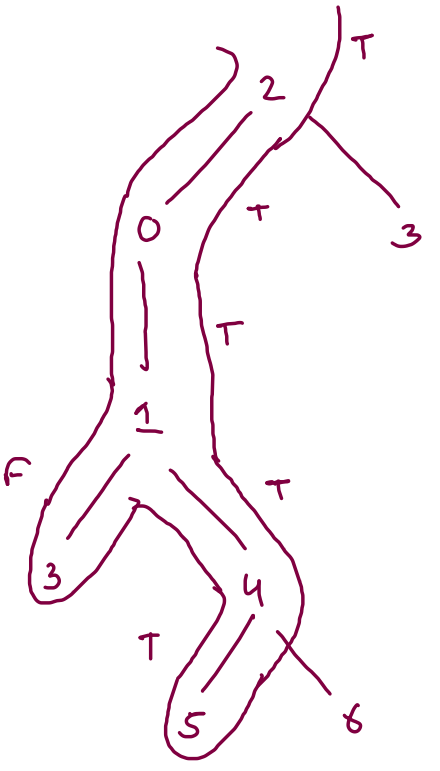
4 -> (4,1,5) , (4,5,7) , (4,6,1)

5 -> (5,4,7) , (5,6,20)

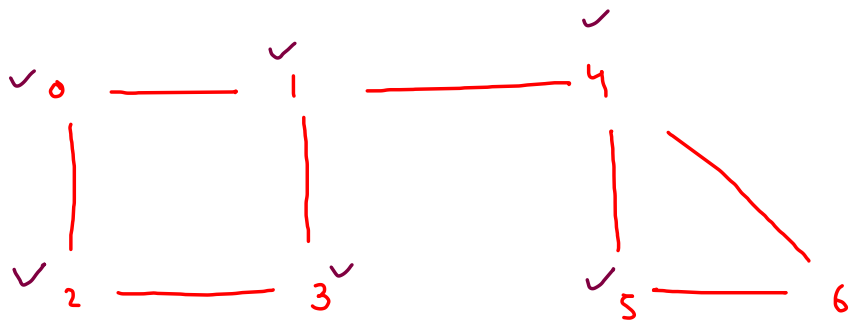
6 -> (6,5,20) , (6,4,1)



hasPath (2, 5)



DFT / dfs



hasPath BFS

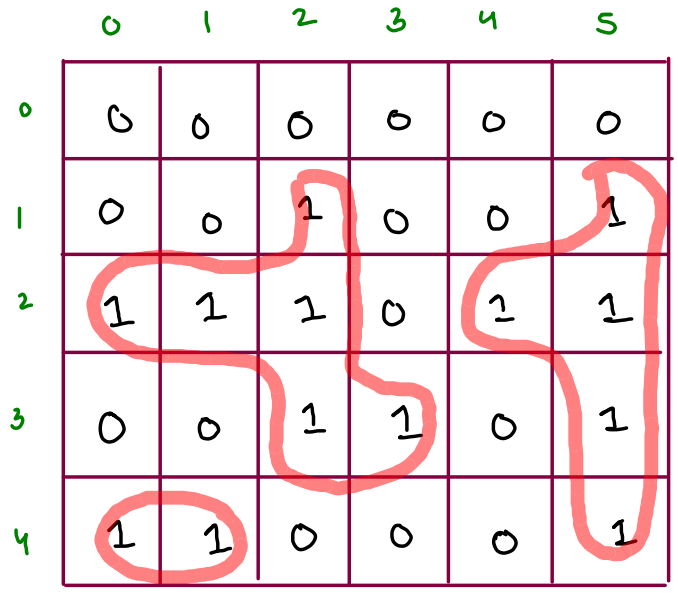
src : 2

dest : 5

x	0	3	x	x	4	5	6

0 m* w a*

200. Number of Islands

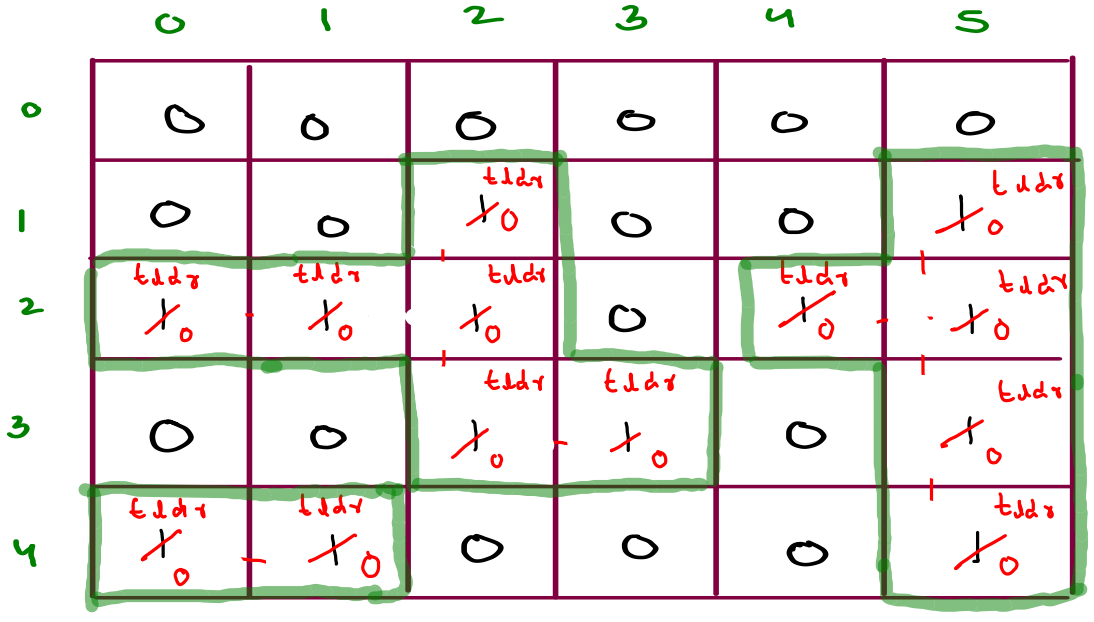


conn. comps

0 -> water

1 -> land

count = 0 1 2 3




```
int count = 0;

for(int i=0; i < grid.length;i++) {
    for(int j=0; j < grid[0].length;j++) {
        if(grid[i][j] == '1') {
            count++;
            dfs(i,j,grid);
        }
    }
}

return count;
```

1, 2

1, 5

4, 0

```
public static void dfs(int i,int j,char[][]grid) {
    if(i < 0 || j < 0 || i >= grid.length || j >= grid[0].length || grid[i][j] == '0') {
        return;
    }

    grid[i][j] = '0';

    dfs(i-1,j,grid); //top
    dfs(i,j-1,grid); //left
    dfs(i+1,j,grid); //down
    dfs(i,j+1,grid); //right
}
```

860 · Number of Distinct Islands

tldr -> order

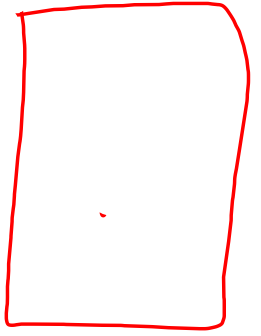
	0	1	2	3	4	5
0	1	0	1	1	1	0
1	1	1	0	1	0	0
2	0	0	0	0	0	0
3	1	0	0	1	1	0
4	1	1	0	0	1	1

add a character for backtracking as well.

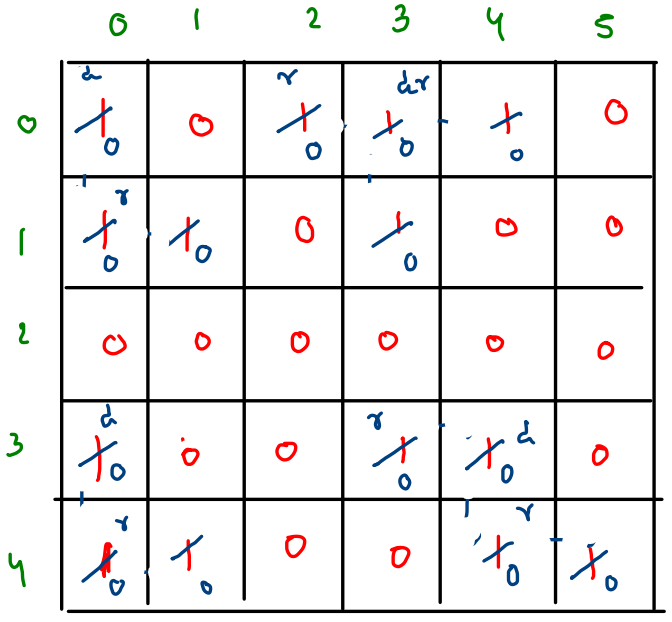
'z'

rdzrzzz

rdrzzzz



comp = rdrzzzz tddr




```java
static String comp;
public int numberofDistinctIslands(int[][] grid) {
    HashSet<String>set = new HashSet<>();

    for(int i=0; i < grid.length;i++) {
        for(int j=0; j < grid[0].length;j++) {
            if(grid[i][j] == 1) {
                comp = "";
                dfs(i,j,grid);
                set.add(comp);
            }
        }
    }

    return set.size();
}
```

drzzz

rdzrzzz

rdrzzzz

```java
public static void dfs(int i,int j,int[][]grid) {

    grid[i][j] = 0;

    //top
    if(i-1 >= 0 && grid[i-1][j] == 1) {
        comp += 't';
        dfs(i-1,j,grid);
    }

    //left
    if(j-1 >= 0 && grid[i][j-1] == 1) {
        comp += 'l';
        dfs(i,j-1,grid);
    }

    //down
    if(i+1 < grid.length && grid[i+1][j] == 1) {
        comp += 'd';
        dfs(i+1,j,grid);
    }

    //right
    if(j+1 < grid[0].length && grid[i][j+1] == 1) {
        comp += 'r';
        dfs(i,j+1,grid);
    }

    comp += 'z';

}
```

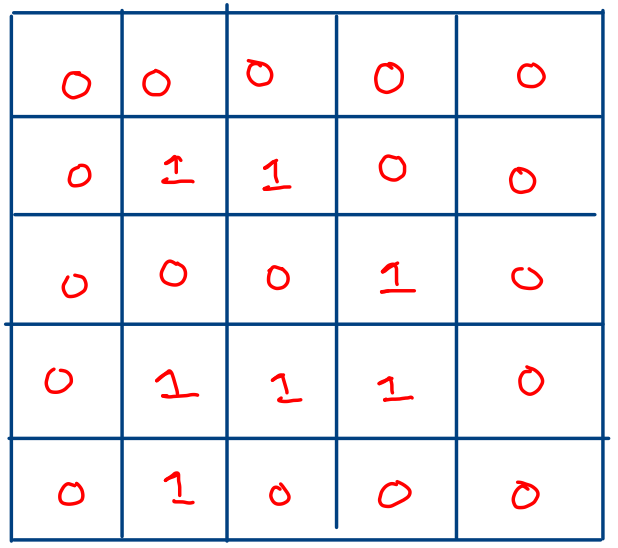
1020. Number of Enclaves

Medium 783 29 Add to List Share

You are given an `m x n` binary matrix `grid`, where `0` represents a sea cell and `1` represents a land cell.

A **move** consists of walking from one land cell to another adjacent (**4-directionally**) land cell or walking off the boundary of the `grid`.

Return *the number of land cells in* `grid` *for which we cannot walk off the boundary of the grid in any number of* ***moves***.



ans = 2

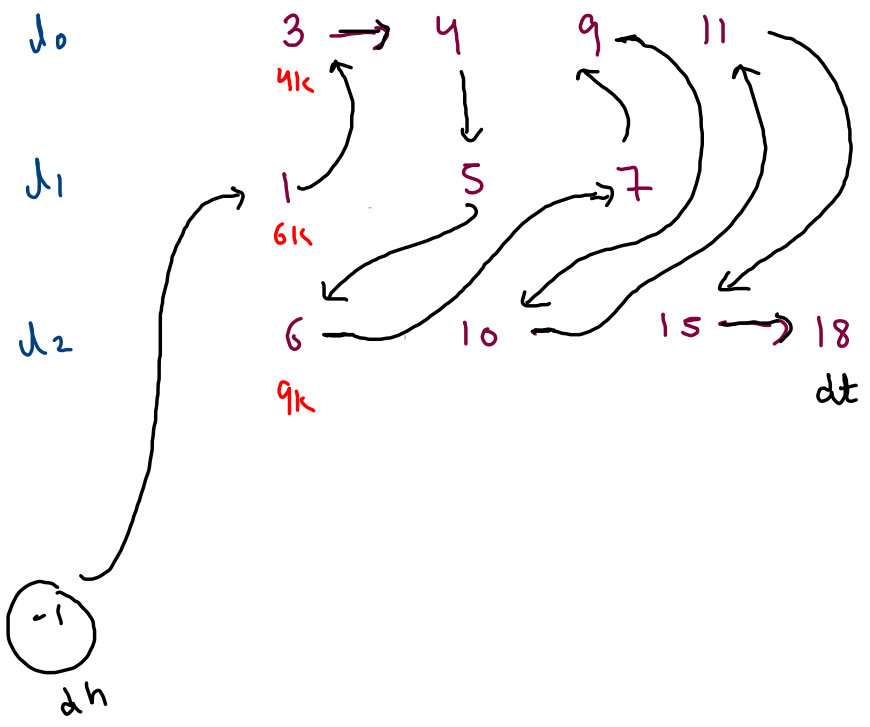
drzzzp

zzzrzpr

rdrzzzzp

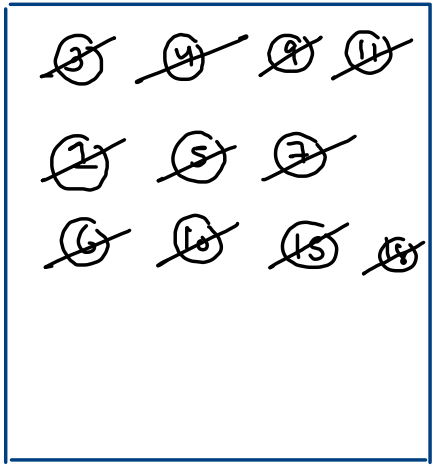
23. Merge k Sorted Lists

(i) using pq



lists

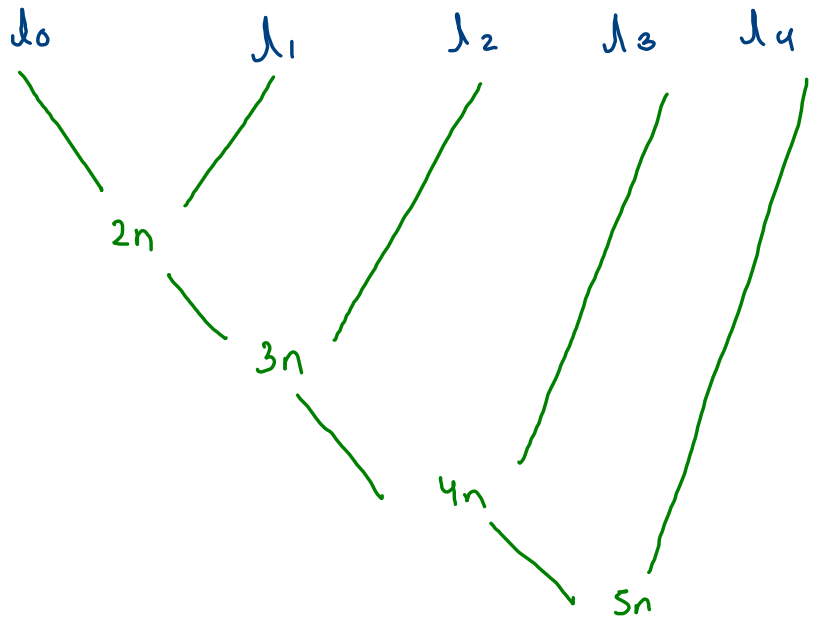
pq : ListNode



pq

(ii) using merging (simple)

avg. size of list : n



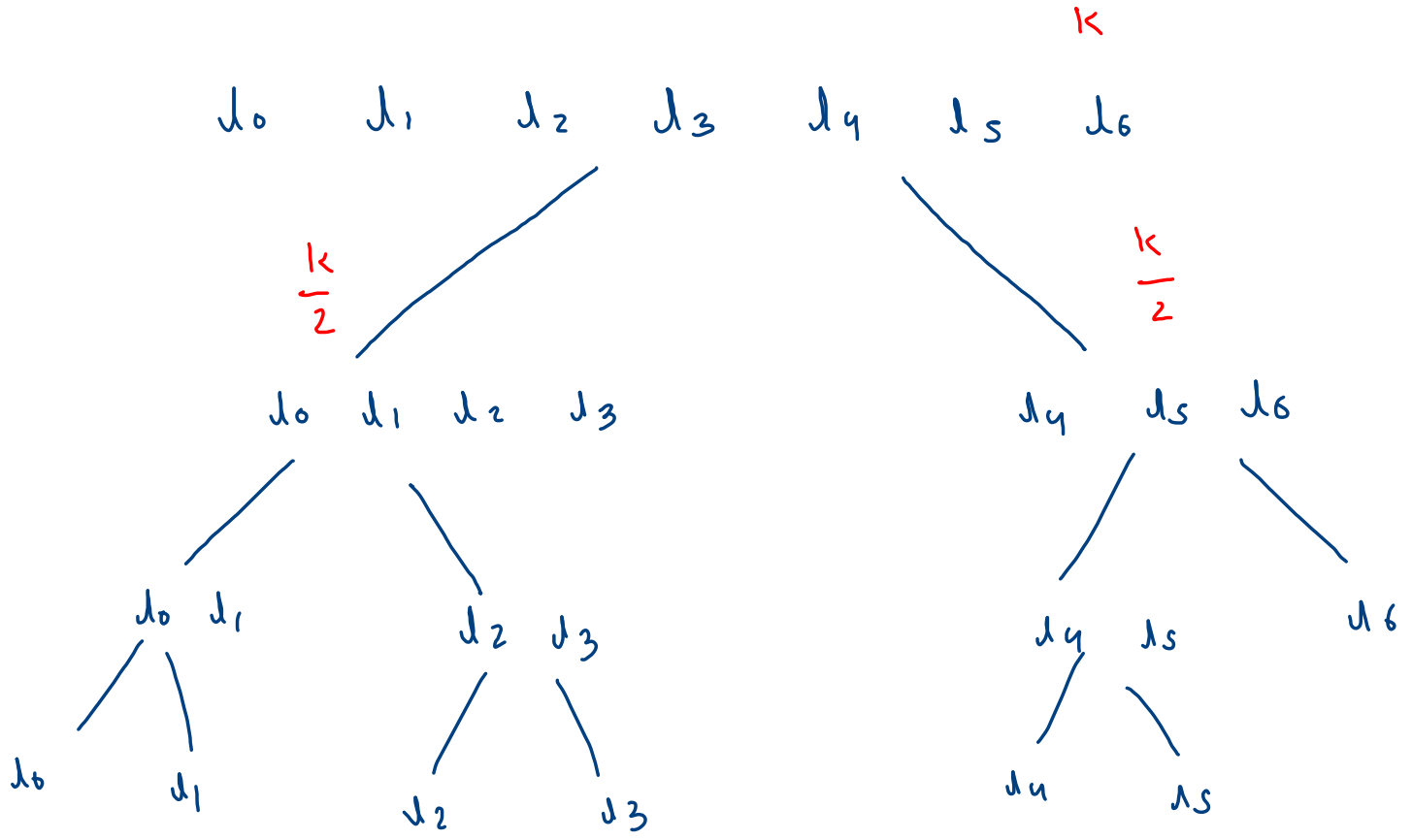
$2n + 3n + 4n + 5n + \cdots + kn$

$\rightarrow n(2 + 3 + 5 + \cdots k)$

$n\left(\frac{k(k+1)}{2}\right)$

$T \propto nk^2$

(iii) using merging (divide and conquer)



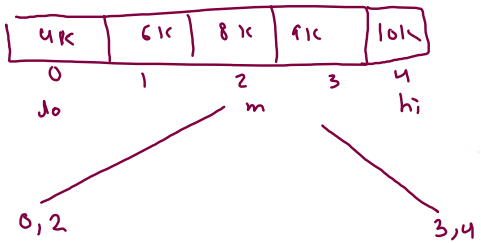
$Nk \log k$

```java
public static ListNode merging(ListNode[]lists,int lo,int hi) {
    if(lo == hi) {
        return lists[lo];
    }

    int mid = (lo + hi) / 2;

    ListNode left = merging(lists,lo,mid);
    ListNode right = merging(lists,mid+1,hi);

    ListNode ans = mergeTwoSortedLL(left,right);
    return ans;
}
```



T

$T(k) = 2T\left(\frac{k}{2}\right) + nk$

$k \to \frac{k}{2}$

$2T\left(\frac{k}{2}\right) = 2T\left(\frac{k}{4}\right) + \frac{nk}{2}$ X 2

$k \to \frac{k}{2}$

$4T\left(\frac{k}{4}\right) = 8T\left(\frac{k}{8}\right) + \frac{nk}{4}$ X 4

$T(1) = c$

$T(k) = nk \times T$

$T = nk \log_2 k$

$S = \log_2 k$ (recursion space)

$k, \frac{k}{2}, \frac{k}{4} \ldots\ldots 2$

$a = k$

$r = \frac{1}{2}$

$dt = 1$

$ar^{T-1} = 1$

$k\left(\frac{1}{2}\right)^{T-1} = 1$

$\frac{k}{2^{T-1}} = 1$

$k = 2^{T-1}$

$\log_2 k = T - 1$

$T = \log_2 k$