# 3045. Count Prefix and Suffix Pairs II

Solved

Hard | Topics | Companies | Hint

You are given a **0-indexed** string array `words`.

Let's define a **boolean** function `isPrefixAndSuffix` that takes two strings, `str1` and `str2`:

- `isPrefixAndSuffix(str1, str2)` returns `true` if `str1` is **both** a prefix and a suffix of `str2`, and `false` otherwise.

For example, `isPrefixAndSuffix("aba", "ababa")` is `true` because `"aba"` is a prefix of `"ababa"` and also a suffix, but `isPrefixAndSuffix("abc", "abcd")` is `false`.

Return *an integer denoting the **number** of index pairs* `(i, j)` *such that* `i < j`, *and* `isPrefixAndSuffix(words[i], words[j])` *is* `true`.

#⭐, #algorithm/trie, #LeetCode/Medium

**Example 1:**

```
Input: words = ["a","aba","ababa","aa"]
Output: 4
Explanation: In this example, the counted index pairs are:
i = 0 and j = 1 because isPrefixAndSuffix("a", "aba") is true.
i = 0 and j = 2 because isPrefixAndSuffix("a", "ababa") is true.
i = 0 and j = 3 because isPrefixAndSuffix("a", "aa") is true.
i = 1 and j = 2 because isPrefixAndSuffix("aba", "ababa") is true.
Therefore, the answer is 4.
```

**Example 2:**

```
Input: words = ["pa","papa","ma","mama"]
Output: 2
Explanation: In this example, the counted index pairs are:
i = 0 and j = 1 because isPrefixAndSuffix("pa", "papa") is true.
i = 2 and j = 3 because isPrefixAndSuffix("ma", "mama") is true.
Therefore, the answer is 2.
```

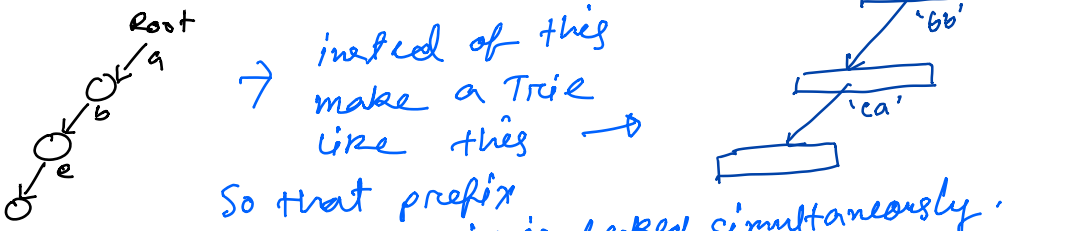
**Example 3:**

```
Input: words = ["abab","ab"]
Output: 0
Explanation: In this example, the only valid index pair is i = 0 and j = 1, and
isPrefixAndSuffix("abab", "ab") is false.
Therefore, the answer is 0.
```

**Constraints:**

# Trie is benchmark for prefix search or suffix search alone with little modification it can do both simultaneously.

# catch 1 :- "abc", "abcxyabc"




→ instead of this make a Trie like this →

So that prefix as well as suffix is checked simultaneously.

# catch 2 :- struct Trienode { bool end of word, Trienode* children[26][26] }

↳ Problem with this is its extremely memory inefficient.

struct Trienode { bool end of word; unordered map<int, Trienode*> children } → use unordered map.

this is a hash value for "ac" hash = a×128 + c → direct or hash = (a-'a')×26 + (c-'a') → char to int.

# 995. Minimum Number of K Consecutive Bit Flips

Solved

Hard | Topics | Companies

You are given a binary array `nums` and an integer `k`.

A **k-bit flip** is choosing a **subarray** of length `k` from `nums` and simultaneously changing every `0` in the subarray to `1`, and every `1` in the subarray to `0`.

Return *the minimum number of **k-bit flips** required so that there is no `0` in the array*. If it is not possible, return `-1`.

A **subarray** is a **contiguous** part of an array.

#⭐, #LeetCode/Hard

#algorithm/array

#algorithm/2 Pointers

#algorithm/Queue

#algorithm/Sliding Window

**Example 1:**

```
Input: nums = [0,1,0], k = 1
Output: 2
Explanation: Flip nums[0], then flip nums[2].
```

**Example 2:**

```
Input: nums = [1,1,0], k = 2
Output: -1
Explanation: No matter how we flip subarrays of size 2, we cannot make the array
become [1,1,1].
```

**Example 3:**

```
Input: nums = [0,0,0,1,0,1,1,0], k = 3
Output: 3
Explanation:
Flip nums[0],nums[1],nums[2]: nums becomes [1,1,1,1,0,1,1,0]
Flip nums[4],nums[5],nums[6]: nums becomes [1,1,1,1,1,0,0,0]
Flip nums[5],nums[6],nums[7]: nums becomes [1,1,1,1,1,1,1,1]
```

**Constraints:**

- `1 <= nums.length <= 10^5`
- `1 <= k <= nums.length`

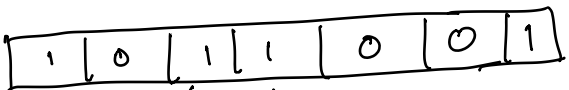
(Biweekly 22 June 82)

→ The previous question, where $k = 3$ constant

it was easy just simulating ;

For variable $k$ $1 < k < n$ that will give TLE.

catch




$k = 4$

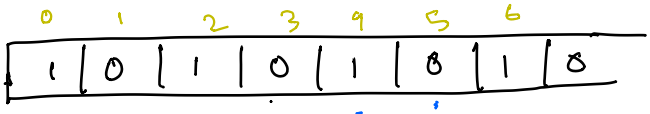
Flip all there

Flip all there

Flip all there

} possibly flip $n \times n$ time

Instead




$k = 4$.

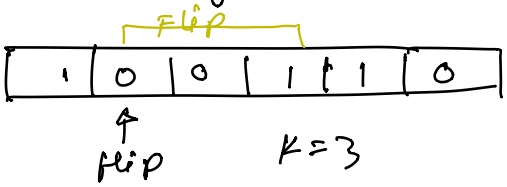
when i reaches here it shall now we need to flip.

Boundary 2 = 5

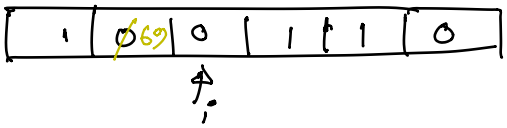
bool toggle ;

queue<int> boundaryQueue ; * To store the boundary value when i surpasses boundary toggle need to change.

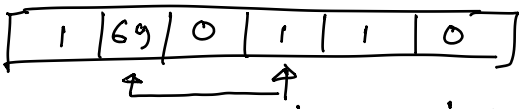
without using any extra space ; -




update here with a value that won't appear in list




# use the array to note the starting position of the flipping boundary.




nums[i-k-1] == 69
time to decrease flip count.

# ⇨ Maximum Profit Assigning Work.

#algorithm/greedy

worker : can do upto a difficulty limit
[40,40,50,80]

jobs have profit & Difficulty

profit : [10, 100, 80, 30, 80]
difficulty : [20, 50, 30, 60, 80]

assign any Job to each worker.

→ arrange difficulty v/s profit like this :

| 20 | 30 | 50 | 60 | 80 |
|---|---|---|---|---|
| 10 | 80 | 100 | 100 | 100 |

→ max profit till this difficulty.

Binary search on this to find max profit for each worker[i].

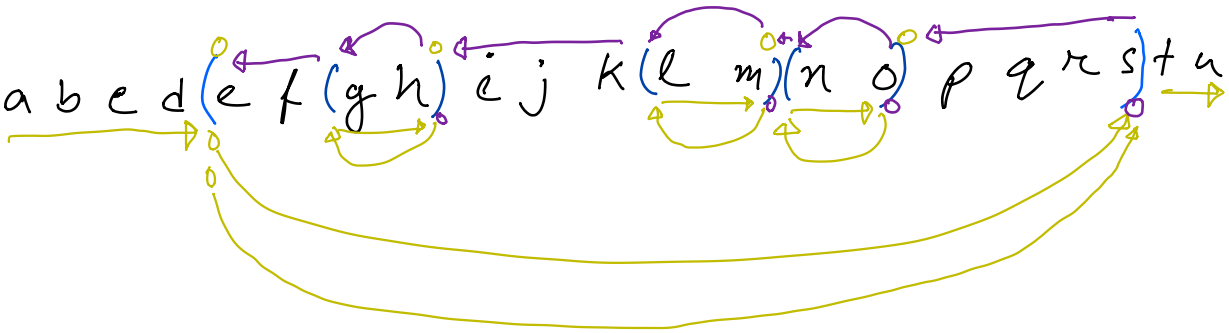
#⭐....

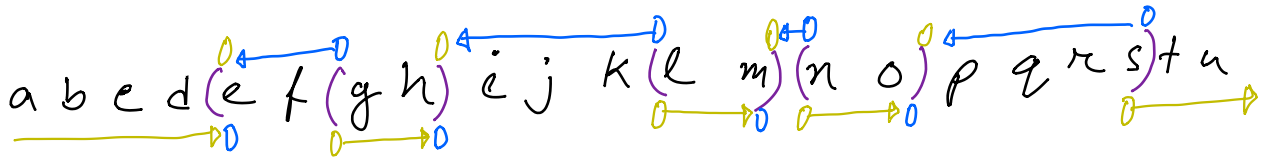
# Reverse String :-

"abcd(ef(gh)ijk(lm)(no)pqrs)tu"

→ start from internal () & flip that.

o/p: abcd tu




Result: abcd s r q p n o l m k j i g h f e t u

→ forward march
→ Backward march
} Whenever any ( or ) hit march reverses & i = door[i]

---

# door = unordered map <int, int>

door[ open br index ] = close br index & vice versa.

#algorithm/stack

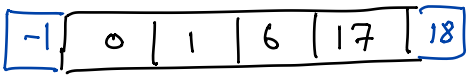
```
for(int i=0, i<n; i += march){
    if(s[i] == '(' || s[i] == ')'){
        march = -march;
        i = door[i]; }
    else { result.push_back(s[i]); } }
```

→ i will eventually become n.

# Longest Valid Parenthesis

after iteration stack =


arr:

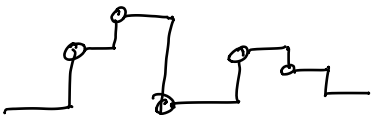



ans = max ( arr[i] - arr[i-1] - 1 )

#LeetCode/Hard, #Revision, #Trick

# To solve this in one pass ;
( incoming index - stacktop index ) max value is the ans.

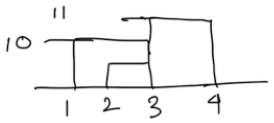
# The Skyline Problem



#LeetCode/Hard

buildings [2, 9, 10]


Sorting , Priority Queue / Set.. (use (2,-10) & (9,10) instead opposite)




[3,-11], [3,10], [3,10] ✓

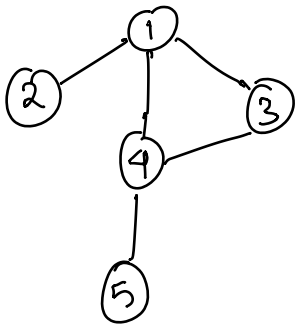
[1,-10] [2,-10] [3,10] [3,10] [3,-11], [4,11]

0 10 10 10 0 11


# Second Minimum Time to reach Destination :-

#LeetCode/Hard, #Revision

#algorithm/Graph



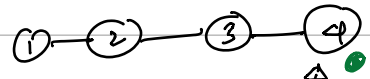

Start from 1 go to 5 ;

✓ after T seconds all node switch color from 🔴 to 🟢

✓ all edges take x second to walk off.

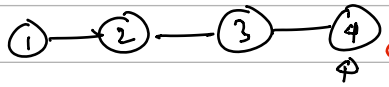
3, 5, 6, 10, 12, 15, 17, 20
0 1 1 2 2 3 3

few miscellaneous things :

→ 

● → can pass

if time passed = 12



● X can't pass wait till t = 10

if time passed = 6

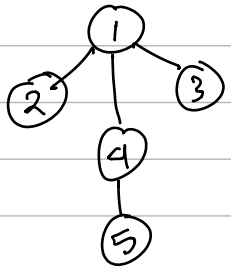
if t = 5

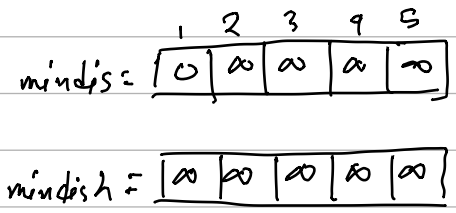

Total Time passed / t = gives which color the signal currently is.

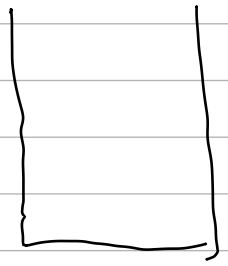
odd = 🟢
even = 🔴

→ Dijkstra will work ; but need a second array to store the second minimum Distance.

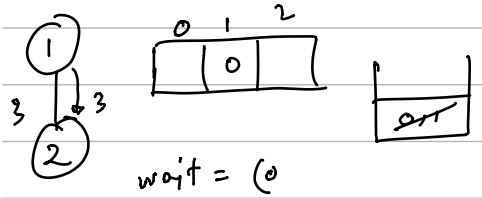
Dry Run



min Heap



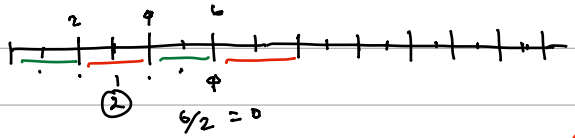




wait = (0

=

=




6/2 = 0

update :

```
pq.push({0, 1});

while(!pq.empty()){
    int node = pq.top().second;
    int timePassed = pq.top().first;
    pq.pop();

    for(int next:adj[node]){
        int wait = (((timePassed + time)/change) + 1)*change;
        int timeToReachNext = ((timePassed + time)/change) & 1? wait : timePassed + time;
        if(timeToReachNext < minDist[next]){
            secondMinDist[next] = minDist[next];
            minDist[next] = timeToReachNext;
            pq.push({timeToReachNext, next});
        }
        else if(timeToReachNext < secondMinDist[next]){
            secondMinDist[next] = timeToReachNext;
            pq.push({timeToReachNext, next});
        }
    }
}

return secondMinDist[n];
}
```

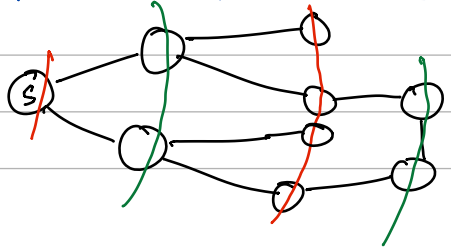
Bug fix

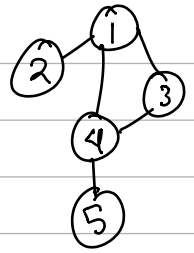
waitTill = (Timepassed/change) & 1 ? ((timepassed/change)+1) * change : Timepassed.

nextnodeReachat = waitTill + time to walk off edge.

Alternate Sol$^{n}$ : can be done using BFS.

idea :






d1[ 0, ∞3, ∞3, ∞3, ∞6 ]

d2[ ∞6, ∞13, ∞6, ∞6, ∞13 ]

q: ① | ③ ④ ② | ① ⑤ ⑥ ② | ② ③ ④ ①

T.P = 3 T = 6 T = 13

## 0 Count no. of Substrings with Dominant Ones :-

1 0 0 1 1 1 → Dominant 1 #1 >= (#0)^2

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

l = 0

r = 1-1 — No zero — ans++

r = 2-1 — 1 zero — ans++

r = 4-1 — X

r = 8-1

[1, 2, 4, 8]

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

l = 1

r = 1 — X

r = 4-1 — X

r = 7 — X

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

l = 1

r = 4-1 — ans++

r = 7 — ans++

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

l = 1, r = 1 — ans++

cnt1 = 1, cnt0 = 1 — idx Okay

r = 7 — ans += 3

# Vlad's sol'n : window condition cnt[0] = z where z = 0 .... z + z*z <= n.

1 0 1 0 1 1 . . . . .

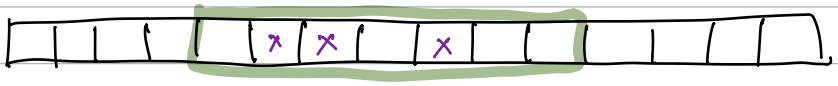
in any window where cnt[0] == z & cnt[1] > 0 & cnt[1] >= z*z its a valid window.

count of possible substrings are min(p - i, cnt[1] - z*z)

[ 1 1 1 1 1 0 0 1 1 1 ..... ]

i p

## To find min K element's sum in a window of L.



set 1 = minimum K

set 2 = Rest

# when an element in set1 gets out of window the next candidate for minimum K value is set2.begin()...

## Largest Number :-

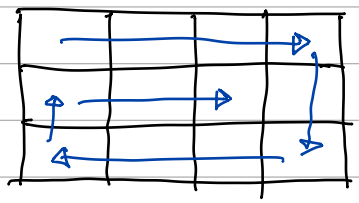
num : [3, 30, 34] ans: "34330"

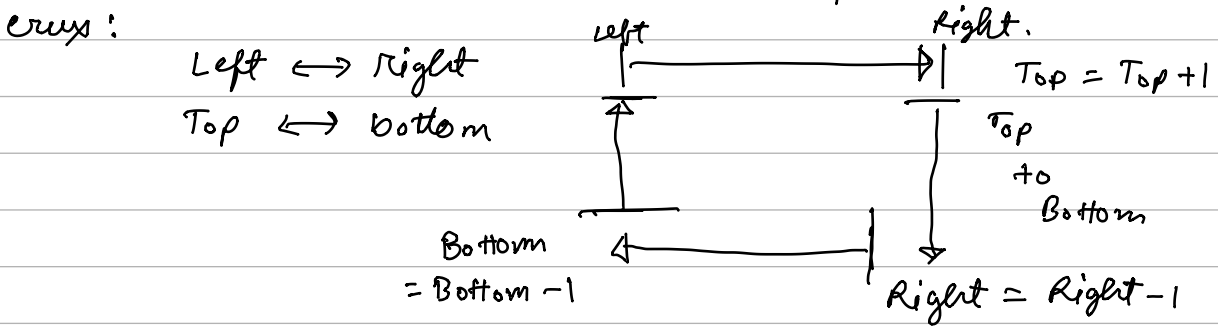
#Trick

```
{
    sort(num.begin(), num.end(), [](string a, string b){
        return a+b > b+a; });
```

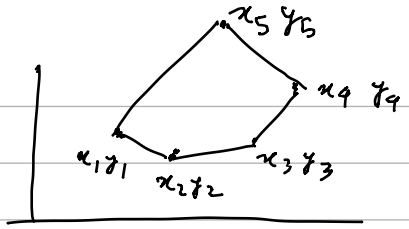
it'll sort [34, 3, 30]

## Spiral Matrix :-

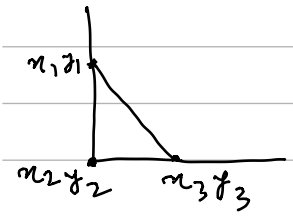




## Area of Polygon :




$$\text{Area} = \frac{1}{2}\left[(x_1 y_2 - y_1 x_2) + (x_2 y_3 - y_2 x_3) + \cdots (x_n y_1 - y_n x_1)\right]$$




$$\text{Area of Triangle} = \frac{1}{2}\left[(x_1 y_2 - y_2 x_1) + (x_2 y_3 - y_2 x_3) + (x_3 y_1 - y_3 x_1)\right]$$

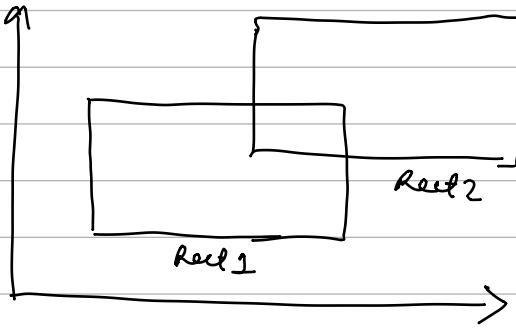
| x | y |
|---|---|
| 0 | 2 |
| 0 | 0 |
| 1 | 0 |

$$\Delta = \frac{1}{2}\left[(0-0) + (0-0) + (2-0)\right]$$
$$= \frac{1}{2} \times 2 = 1 \text{ u}$$

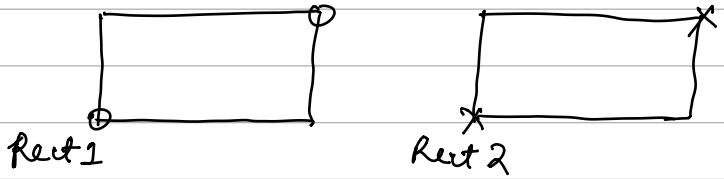
## Rectangular overlap :-

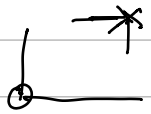
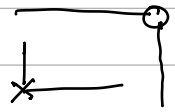
rect 1 : $[x_1, y_1, x_2, y_2]$
rect 2 : $[x_1, y_1, x_2, y_2]$

return if they overlap.




⇒ most simple & elegant conditional statement.



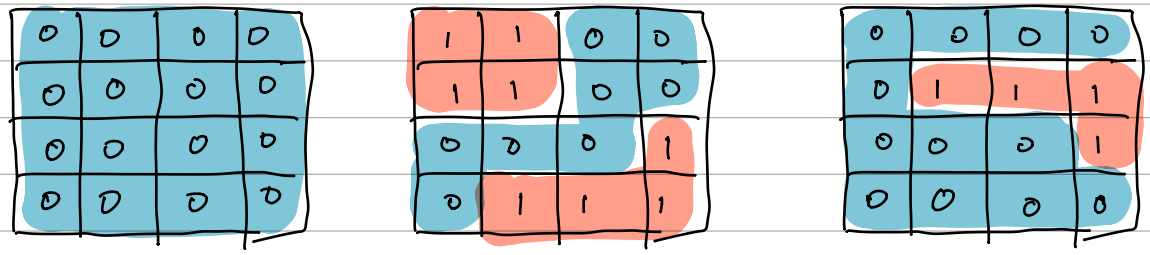

They will overlap if  && 

Top corner of Rect2 is higher & Right of Rect1's bottom corner && bottom corner of Rect2 is below and left of rect 1's Top Corner.

#Trick, #Tricky business

#Revision

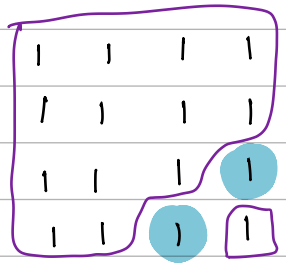
## Minimum no. of days to make Island Disconnected :



0 Island | 2 Island | 2 Island

ans = 0 (already disconnected) | 0 | 1

fill (1, 2) th box 0 then it'll be disconnected.



→ Now disConnected in 2 Days

possible answers 0, 1, 2. no more extra days needed to make any arrangement of Island to be disconnected.

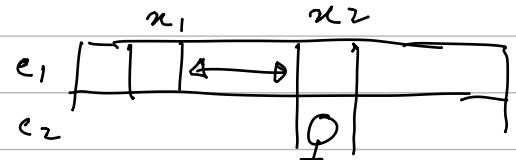
## Maximum Distance in array:

[[1, 2, 1, 5], [100, 2, 3], [9, 8], [7, 6]]

maximum Diff of max of one array vs min of another array.

## Maximum no. of points with cost.

2D-DP




add point = arr[$c_1$][$x_1$] − (abs($x_1$ − $x_2$))

| 5 | 1 | 4 | 2 | 1 |
|---|---|---|---|---|
| 3 | 2 | 3 | 4 | 3 |

Trick

| $a_1$ | $a_2$ | $a_3$ | $a_4$ |
|---|---|---|---|
| $b_1$ | $b_2$ | $b_3$ | $b_4$ |

then Reverse!

dp:

| $a_1$ | $a_2$ | $a_3$ | $a_4$ |
|---|---|---|---|
| $a_1+b_1$ | $b_2$ + | | |

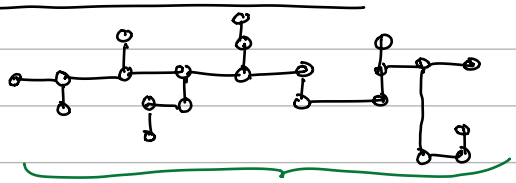
$\max(dp[i][j-1]-1, a_2) = b_2 + a_1 - 1$

$b_3 + a_1 - 2$

$b_4 + a_4$

| $a_1$ | $a_2$ | $a_3$ | $a_4$ |
|---|---|---|---|
| | $b_2$ + | $b_3$ + | $b_4 + a_4$ |
| $b_2$ + $a_1$ | $a_4 - 2$ | $\max(dp[i][j+1]-1, a_3) = b_3 + a_4 - 1$ | |

## Most stones Removed with Same Row/column



Main Logic

this whole cluster is Row/column Connected

* It's always possible to Remove $n-1$ element and only 1 stone in whole cluster remaining

To count no. of Islands.

Approach 1) DFS

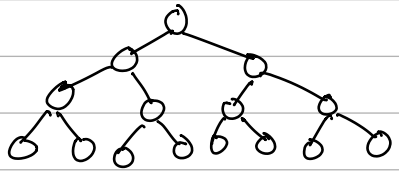
Approach 2) Union Find

Don't make Actual grid make use of an array for DSU

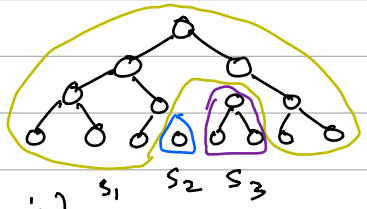
For DFS mark Adj X and Adj Y x-axis & y-axis neighbor nodes.

Then run DFS & count # connected Islands.

## Minimum Score after Removal on a Tree is- Hard



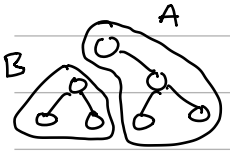
remove any 2 edge



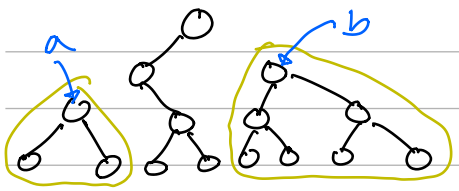

ans = min ( max − min)

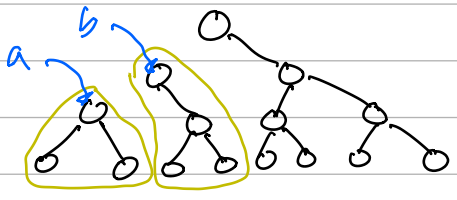
max = maximum XOR among $S_1$ $S_2$ $S_3$

min = minimum " " $S_1$ $S_2$ $S_3$




XOR(A) = XOR(Total) − XOR(B)




① a & b are not ancestor-Predecessor

3 XOR are Total^a^b, a, b

② a is anc of b or b is anc of a.

Total^a, a^b, b

or Total^b, b^a, a.

* To check ancestory unordered set can be used,

or

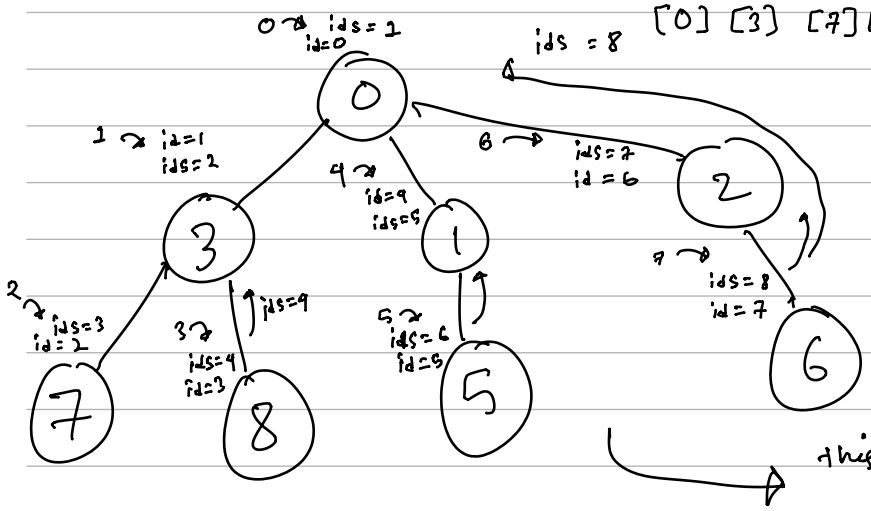
more optimally, unique Id and Last node's value can help.

(Vlad's soln)

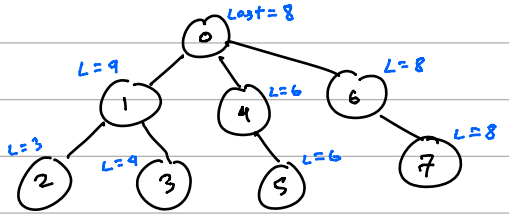
Last: 0 1 2 3 4 5 6 7 8 9 10

8 4 3 4 6 6 8 8

[0] [3] [7] [8] [1] [5] [2] [6]




this tree in term of Id




i.e. id = 1 has seen 8th

id = 5 has seen 6th

id: 0 1 2 3 4 5 6 7 8

Hence, seen: | 8 | 4 | 3 | 4 | 6 | 6 | 8 | 8 | |

if id1 = i , id2 = j

if j < seen[i] means id2 = j is subtree of

#miscellaneous-STL, #Revision, #Trick, #Tricky business