203. Remove Linked List Elements

Remove all elements from a linked list of integers that have value *val*.

Example:

```
Input:  1->2->6->3->4->5->6, val = 6

Output: 1->2->3->4->5
```

Got this on the first try. Anytime with a linked list, make sure to check for nulls. If we have val at the beginning 6->6->6->6->->7, first do a separate while loop to get rid of starting 6s. Then get to the meat of the problem.

206. Reverse Linked List

Reverse a singly linked list.

Example:

```
Input: 1->2->3->4->5->NULL
Output: 5->4->3->2->1->NULL
```

Follow up:

A linked list can be reversed either iteratively or recursively. Could you implement both?

Do the trick with 3 nodes. Prev, while(curr != null), and Node next = curr.next. When loop ends, return prev (which will be the new head).

207. Course Schedule

There are a total of *n* courses you have to take, labeled from 0 to n-1.

Some courses may have prerequisites, for example to take course 0 you have to first take course 1, which is expressed as a pair: [0,1]

Given the total number of courses and a list of prerequisite pairs, is it possible for you to finish all courses?

Example 1:

```
Input: 2, [[1,0]]

Output: true

Explanation: There are a total of 2 courses to take.

             To take course 1 you should have finished course 0. So it is possible.
```

Example 2:

```
Input: 2, [[1,0],[0,1]]

Output: false

Explanation: There are a total of 2 courses to take.

             To take course 1 you should have finished course 0, and to take course 0
you should

             also have finished course 1. So it is impossible.
```

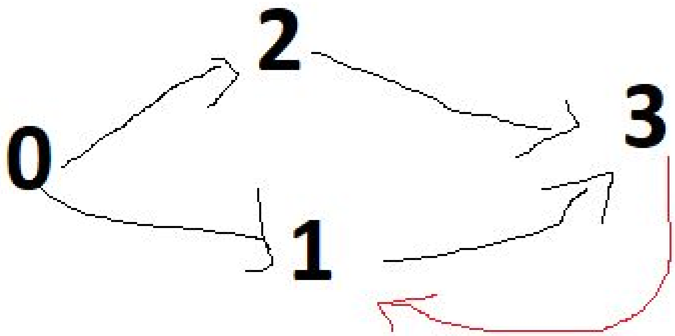
Note:

1. The input prerequisites is a graph represented by a list of edges, not adjacency matrices. Read more about how a graph is represented.

2. You may assume that there are no duplicate edges in the input prerequisites.

```
Courses: 4
```

```
[[2,0],[1,0],[3,1],[3,2],[1,3]]
```



Input is the test case that failed. I have a graph pic of this

int course = 4;

int[][] prerequisites = new int[5][];

prerequisites[0] = new int[] { 2, 0 };

prerequisites[1] = new int[] { 1, 0 };

prerequisites[2] = new int[] { 3, 1 };

prerequisites[3] = new int[] { 3, 2 };

prerequisites[4] = new int[] { 1, 3 };

List<int> array

[0] = { 2, 1 }

[1] = { 3 }

[2] = { 3 }

[3] = { 1 }

Do DFS on every one, see if we are successful

```
Process = 0
visited[0] = true
visited[1]
visited[2]
visited[3]
        Now Process { 2, 1 } because [0] = { 2, 1 }
        visited[0] = true
        visited[1]
        visited[2] = true
        visited[3]
                Now Process { 3 } because [3] = { 1 }
                visited[0] = true
                visited[1]
                visited[2] = true
                visited[3] = true
                        Now Process { 1 } because [3] = { 1 }
                        visited[0] = true
                        visited[1] = true
                        visited[2] = true
                        visited[3] = true
                                Now Process { 3 } because [3] = { 1 }, however, visited[0] = true
                                We have a cycle so can't take the course.
```

208. Implement Trie (Prefix Tree)

Implement a trie with insert, search, and startsWith methods.

Example:

```
Trie trie = new Trie();


trie.insert("apple");
trie.search("apple");   // returns true
trie.search("app");     // returns false
trie.startsWith("app"); // returns true
trie.insert("app");
trie.search("app");     // returns true
```

Note:

- You may assume that all inputs are consist of lowercase letters a-z.
- All inputs are guaranteed to be non-empty strings.

Super easy,
1 Create Node Object

```
    public class Node
    {
        public bool IsEndOfWord { get; set; }
        public Dictionary<char, Node> Children { get; set; }
        public char CurrentCharacter { get; set; }

        public Node()
        {
            Children = new Dictionary<char, Node>();
        }
    }
```

2) Trie class will contains 26 Node Objects. NOTE: Make a head object, chich contains letters a-z. Array doesn't make sense. Second implementation i did this and inserts are much easier.
3) When we get a letter, keep setting children. On the last letter of the word, set IsEndOfWork to true. This way we know it is a end of word.

Another idea:
Note: saw someone's implementation, instead of using a dictionary, he used an array. This way we check for null instead of contains key.

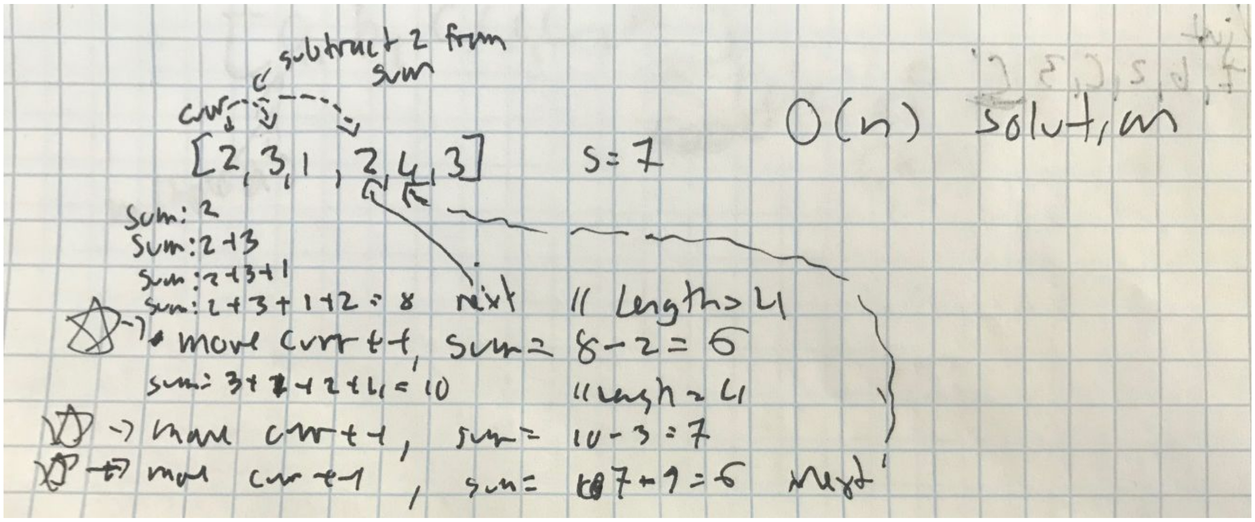
209. Minimum Size Subarray Sum

Given an array of n positive integers and a positive integer s, find the minimal length of a contiguous subarray of which the sum ≥ s. If there isn't one, return 0 instead.

Example:

```
Input: s = 7, nums = [2,3,1,2,4,3]
Output: 2
Explanation: the subarray [4,3] has the minimal length under the problem constraint.
```

Follow up:
If you have figured out the *O*(*n*) solution, try coding another solution of which the time complexity is *O*(*n* log *n*).



210. Course Schedule II

There are a total of *n* courses you have to take, labeled from `0` to `n-1`.

Some courses may have prerequisites, for example to take course 0 you have to first take course 1, which is expressed as a pair: `[0,1]`

Given the total number of courses and a list of prerequisite pairs, return the ordering of courses you should take to finish all courses.

There may be multiple correct orders, you just need to return one of them. If it is impossible to finish all courses, return an empty array.

Example 1:

```
Input: 2, [[1,0]]
Output: [0,1]
Explanation: There are a total of 2 courses to take. To take course 1 you should have
finished
             course 0. So the correct course order is [0,1] .
```

Example 2:

```
Input: 4, [[1,0],[2,0],[3,1],[3,2]]
Output: [0,1,2,3] or [0,2,1,3]
Explanation: There are a total of 4 courses to take. To take course 3 you should have
finished both
             courses 1 and 2. Both courses 1 and 2 should be taken after you finished
course 0.
             So one correct course order is [0,1,2,3]. Another correct ordering is

[0,2,1,3] .
```

Note:

1. The input prerequisites is a graph represented by a list of edges, not adjacency matrices. Read more about how a graph is represented.
2. You may assume that there are no duplicate edges in the input prerequisites.

Follow the standard Topological sort templates:

For: `4, [[1,0],[2,0],[3,1],[3,2]]`

1 Create a map and indegree. They are almost the opposite of each other.

InDegree: tells number of nodes pointing to it

[1] = 1

[0] = 0

[2] = 1

[3] = 2

Map: shows the list of nodes depending on its completion
[1] = {3}
[0] = {1,2}
[2] = {3}
[3] = {}

2) Perform topological sort recursively

- If we don't have any indegree of zeros, return
    - if(zeros.Count == 0)
    - return;
- If we don't have a value in indegree, there is a cycle
    - if(!inDegree.ContainsKey(m))
    - return;

211. Add and Search Word - Data structure design

Design a data structure that supports the following two operations:

```
void addWord(word)

bool search(word)
```

search(word) can search a literal word or a regular expression string containing only letters `a-z` or `.`. A `.` means it can represent any one letter.

Example:

```
addWord("bad")

addWord("dad")

addWord("mad")

search("pad") -> false

search("bad") -> true

search(".ad") -> true

search("b..") -> true
```

Note:

You may assume that all words are consist of lowercase letters a-z.

Very similar to 208, create a Trie, but when Searching, we have wildcards. So the substring trick learned for wildcard matching. Basically do world.Substring(1) and pass in list of valid nodes. When we have a “.” list will contain all values of the dictionary. If not list will only contain one node.

212. Word Search II

iven a 2D board and a list of words from the dictionary, find all words in the board.

Each word must be constructed from letters of sequentially adjacent cell, where "adjacent" cells are those horizontally or vertically neighboring. The same letter cell may not be used more than once in a word.

Example:

```
Input:
words = ["oath","pea","eat","rain"] and board =
[
  ['o','a','a','n'],
  ['e','t','a','e'],
  ['i','h','k','r'],
  ['i','f','l','v']
]

Output: ["eat","oath"]
```

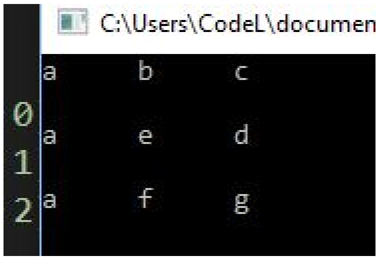
Note:

You may assume that all inputs are consist of lowercase letters a-z.

I used the Trie from 208, For this problem, do not forget to backtrack. If we set one point as visited, have to undo it. I got hit by the edge case.

string[] words = { "abcdefg", "gfedcbaaa", "eaabcdgfa", "befa", "dgc", "ade" };

Matrix:



```
a       b       c
a       e       d
a       f       g
```

If you do not backtrack, you might not get eaabcdgfa because the a in [3,0] gets marked as visited. However, we can find eaabcdgfa in matrix.

Alos, we do not need a full Trie data structure, We can have a TrieNode and do all the work from the Node. I think it is better to have it.
Also, this guy is slick, keeps the word in TrieNode. If we do this, we do not have to pass in strings and each level add a char to string.

```
class TrieNode {
    TrieNode[] next = new TrieNode[26];
    String word;
}
```

215. Kth Largest Element in an Array

Find the kth largest element in an unsorted array. Note that it is the kth largest element in the sorted order, not the kth distinct element.

Example 1:

```
Input: [3,2,1,5,6,4] and k = 2
```

```
Output: 5
```

Example 2:

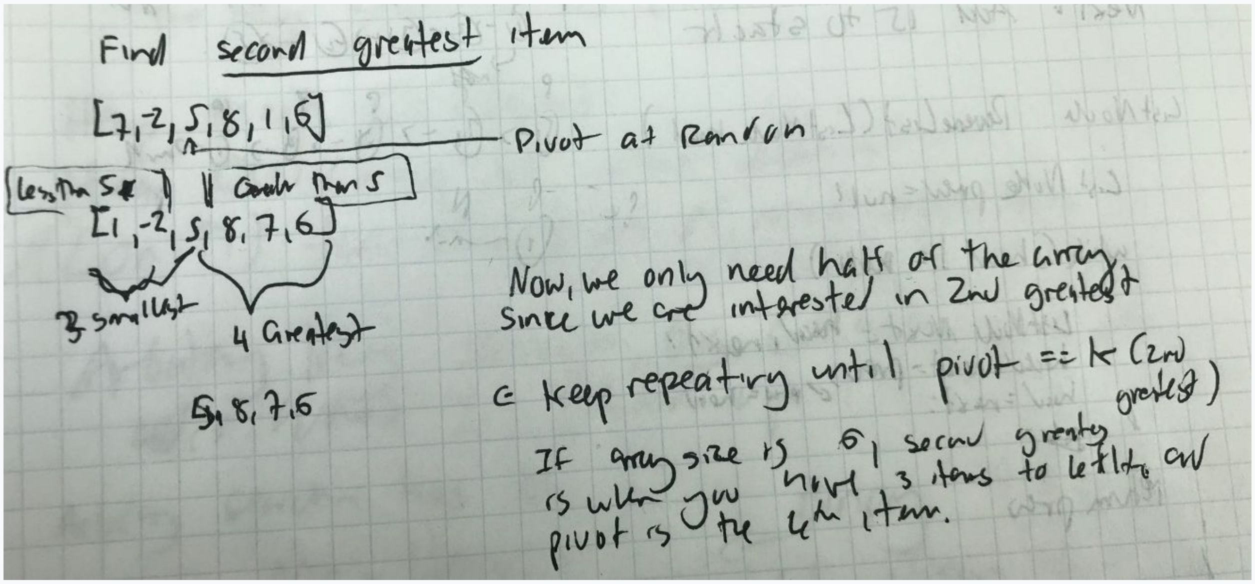
```
Input: [3,2,3,1,2,4,5,5,6] and k = 4
```

```
Output: 4
```

Note:

You may assume k is always valid, 1 ≤ k ≤ array's length.

1) The solution to the question is in the question. Notice how k is 1 <= k <= array.Length. So simply, sort the array, return nums[nums.Length - k];
2) Another option is to use a priority queue.
3) The smart approach for this problem is to use the selection algorithm (based on the partion method - the same one as used in quicksort). See image below



Now, we only need half of the array since we are interested in 2nd greatest

⊂ keep repeating until pivot == k (2nd greatest)

If array size is 6, second greatest is when you have 3 items to left and pivot is the 4th item.

216. Combination Sum III

Find all possible combinations of *k* numbers that add up to a number *n*, given that only numbers from 1 to 9 can be used and each combination should be a unique set of numbers.

Note:

- All numbers will be positive integers.
- The solution set must not contain duplicate combinations.

Example 1:

`Input: k = 3, n = 7`

`Output: [[1,2,4]]`

Example 2:

`Input: k = 3, n = 9`

`Output: [[1,2,6], [1,3,5], [2,3,4]]`

Backstrack motherfucker, but be smart. Sink K numbers should add up to n, we don't have to go into next backtrack recursive call if we go past n. We are only doing 1 to k in that order 1,2,3,4...k.

217. Contains Duplicate

Given an array of integers, find if the array contains any duplicates.

Your function should return true if any value appears at least twice in the array, and it should return false if every element is distinct.

Example 1:

```
Input: [1,2,3,1]
Output: true
```

Example 2:

```
Input: [1,2,3,4]
Output: false
```

Example 3:

```
Input: [1,1,1,3,3,4,3,2,4,2]
Output: true
```

Hashset, if Hashset.Add() == false, we have a dupe.

219. Contains Duplicate II

Given an array of integers and an integer *k*, find out whether there are two distinct indices *i* and *j* in the array such that nums[i] = nums[j] and the absolute difference between *i* and *j* is at most *k*.

Example 1:

```
Input: nums = [1,2,3,1], k = 3
Output: true
```

Example 2:

```
Input: nums = [1,0,1,1], k = 1
Output: true
```

Example 3:

```
Input: nums = [1,2,3,1,2,3], k = 2
Output: false
```

Slick solution with a Hashset. In a loop, keep adding items to the hashset. The minute i >k, remove the first item that was added to hashset. Keep repeating.

Removing values from set: set.Remove(nums[i - k - 1]);

This way if there is a dupe, the difference between the dupts is at most k.

222. Count Complete Tree Nodes

Given a complete binary tree, count the number of nodes.

Note:

Definition of a complete binary tree from Wikipedia:

In a complete binary tree every level, except possibly the last, is completely filled, and all nodes in the last level are as far left as possible. It can have between 1 and 2h nodes inclusive at the last level h.

Example:

```
Input:
    1
```

```
   / \
  2   3
 / \  /
4  5 6
```

```
Output: 6
```

Not sure why this is a medium we just have to count the number of nodes. Only trick is to pass in the integer count via reference. If you pass by value, you might be fucked. Remember with recursion and ints, it is better to pass by reference.

224. Basic Calculator

Implement a basic calculator to evaluate a simple expression string.

The expression string may contain open `(` and closing parentheses `)`, the plus `+` or minus sign `-`, non-negative integers and empty spaces .

Example 1:

```
Input: "1 + 1"
Output: 2
```

Example 2:

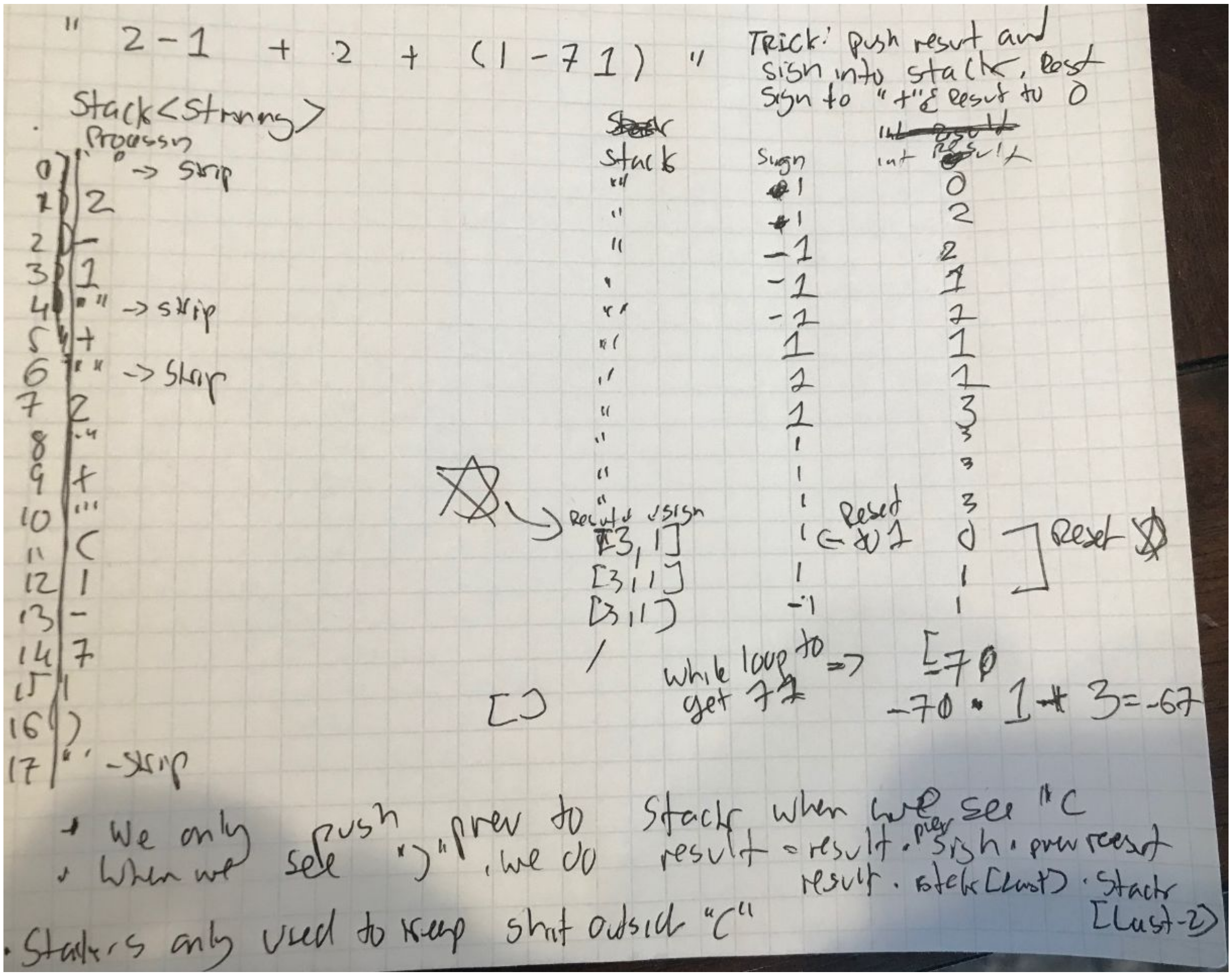
```
Input: " 2-1 + 2 "
Output: 3
```

Example 3:

```
Input: "(1+(4+5+2)-3)+(6+8)"
Output: 23
```

Note:

- You may assume that the given expression is always valid.
- Do not use the `eval` built-in library function.



" 2 - 1 + 2 + (1 - 7 1) "

TRick: push resut and sign into stack, Rest sign to "+" & Resut to 0

Stack<Strnng>

	Processn	~~Stack~~ Stack	Sign	~~int result~~ int Result	
0	" " -> Strip	""	1	0	
1	2	""	1	2	
2	-	""	-1	2	
3	1	""	-1	1	
4	" " -> strip	""	-1	1	
5	+	""	1	1	
6	" " -> Strip	""	1	1	
7	2	""	1	3	
8	" "	""	1	3	
9	+	""	1	3	
10	" "	""	1	3	
11	(	Result & sign [3, 1]	1 ← Reset to 1	0	Reset 0
12	1	[3, 1]	1	1	
13	-	[3, 1]	-1	1	
14	7	/	while loop to get 71 =>	[-70	
15	1				
16	)	[ ]		-70 * 1 + 3 = -67	
17	" " -Strip				

- We only push prev to Stack when we see "(
- When we see ")", we do result = result * prev sign * prev resut
  result * stack[Last] * Stack[Last-2]

• Stack is only used to keep shit outside "("

226. Invert Binary Tree

Invert a binary tree.

Example:

Input:

```
     4
   /   \
  2     7
 / \   / \
1   3 6   9
```

Output:

```
    4
   /   \
  7     2
 / \   / \
9   6 3   1
```

Trivia:

This problem was inspired by this original tweet by Max Howell:

Google: 90% of our engineers use the software you wrote (Homebrew), but you can’t invert a binary tree on a whiteboard so f*** off.

Just swap the left and right sides. Then call recursive on left side and right side.
How to swap sides
TreeNode left = root.left;
root.left = root.right;
root.right = left;

228. Summary Ranges

Given a sorted integer array without duplicates, return the summary of its ranges.

Example 1:

```
Input:  [0,1,2,4,5,7]
Output: ["0->2","4->5","7"]
Explanation: 0,1,2 form a continuous range; 4,5 form a continuous range.
```

Example 2:

```
Input:  [0,2,3,4,6,8,9]
Output: ["0","2->4","6","8->9"]
Explanation: 2,3,4 form a continuous range; 8,9 form a continuous range.
```

The loop I had was bad. Easy way to do loop here.

1) Let int a = nums[i];
2) Increment i until (nums[i + 1] - nums[i]) == 1
3) If a != nums[i]
    a) list.Add(a + "->" + nums[i]);

b) Else it’s a single number, like 6, in the example above, so just add to list.

230. Kth Smallest Element in a BST

Given a binary search tree, write a function kthSmallest to find the kth smallest element in it.

Note:

You may assume k is always valid, 1 ≤ k ≤ BST's total elements.

Example 1:

```
Input: root = [3,1,4,null,2], k = 1
   3
  / \
 1   4
  \
   2
Output: 1
```

Example 2:

```
Input: root = [5,3,6,2,4,null,null,1], k = 3
       5
      / \
     3   6
    / \
   2   4
  /
 1
Output: 3
```

Follow up:

What if the BST is modified (insert/delete operations) often and you need to find the kth smallest frequently? How would you optimize the kthSmallest routine?

Again my trick is to pass k by reference. Then do a PreOrder traversal and decrease the K after we are done exploring the left hand side. When k==0, we know we have the kth smallest value.

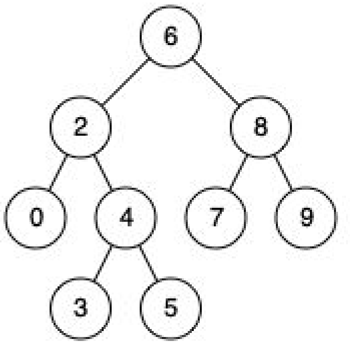
```
Helper(root.left)

K-- or set kth

Helper(root.right)
```

235. Lowest Common Ancestor of a Binary Search Tree

Given a binary search tree (BST), find the lowest common ancestor (LCA) of two given nodes in the BST.

According to the definition of LCA on Wikipedia: “The lowest common ancestor is defined between two nodes p and q as the lowest node in T that has both p and q as descendants (where we allow a node to be a descendant of itself).”

Given binary search tree: root = [6,2,8,0,4,7,9,null,null,3,5]



Example 1:

```
Input: root = [6,2,8,0,4,7,9,null,null,3,5], p = 2, q = 8
Output: 6
Explanation: The LCA of nodes 2 and 8 is 6.
```

Example 2:

```
Input: root = [6,2,8,0,4,7,9,null,null,3,5], p = 2, q = 4
Output: 2
Explanation: The LCA of nodes 2 and 4 is 2, since a node can be a descendant of
itself according to the LCA definition.
```

Note:

- All of the nodes' values will be unique.
- p and q are different and both values will exist in the BST.

Note: This is a binary search tree.

1) If root is between p and q or q and p, root is the lowest common ancestor. Remember, the problem does not say if p is > 1 or q is > p.

If 1 is not the case, we know the both p and q should be on the left side of root or right side of root. So if

- root > p or q, we have to recursively check root.left.
- root < p or q, we have to recursively check root.right

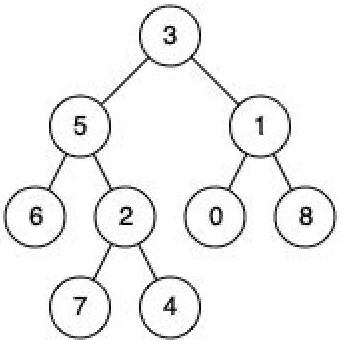
When checking recursively, if root == p or root == q, we know that (p or q) is the lowest common ancestor. We might have a case where p or q is within the subtree of p or q.

236. Lowest Common Ancestor of a Binary Tree

Given a binary tree, find the lowest common ancestor (LCA) of two given nodes in the tree.

According to the definition of LCA on Wikipedia: "The lowest common ancestor is defined between two nodes p and q as the lowest node in T that has both p and q as descendants (where we allow a node to be a descendant of itself)."

Given the following binary tree: root = [3,5,1,6,2,0,8,null,null,7,4]



Example 1:

```
Input: root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 1
Output: 3
```

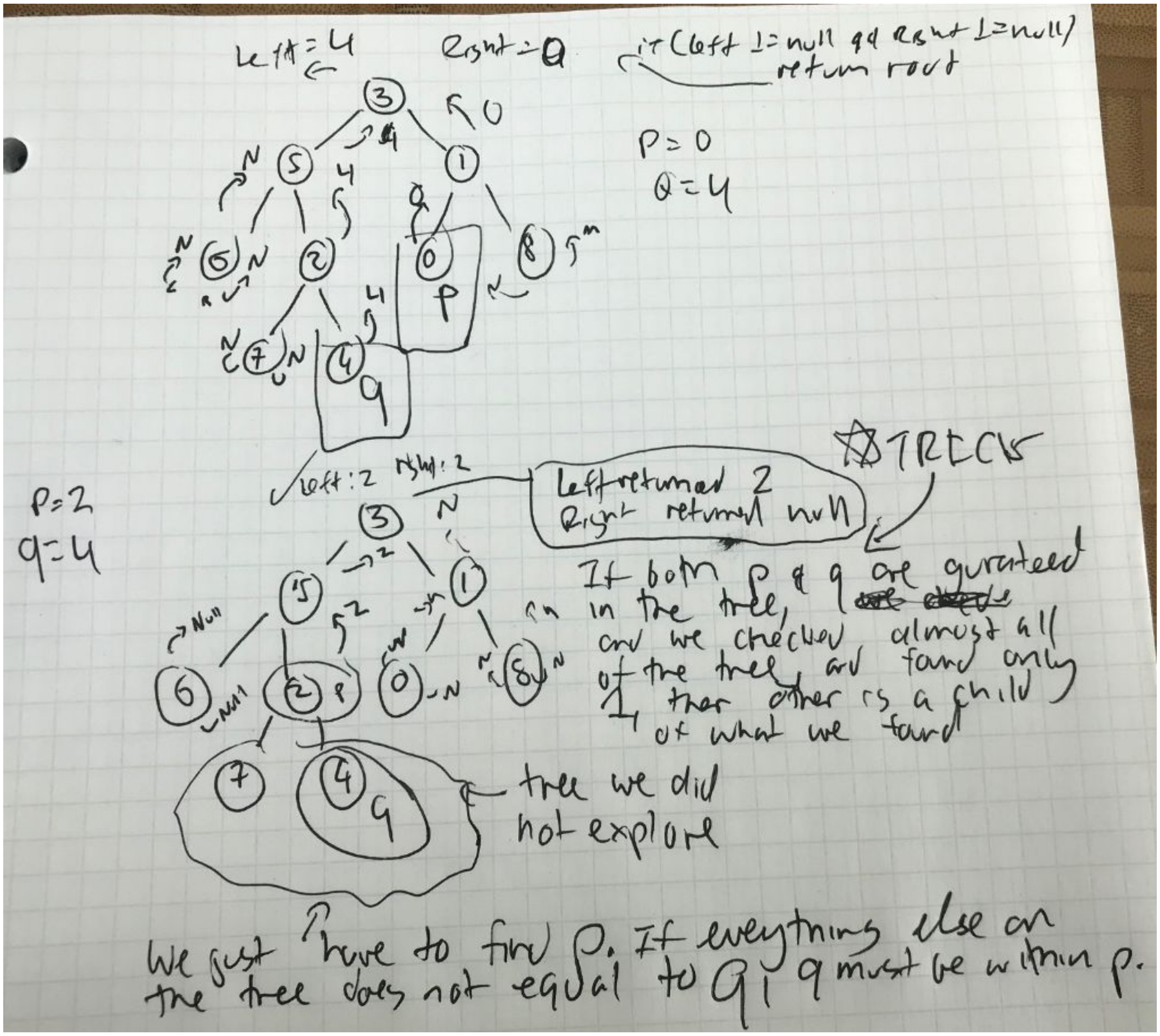
```
Explanation: The LCA of nodes 5 and 1 is 3.
```

Example 2:

```
Input: root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 4
Output: 5
Explanation: The LCA of nodes 5 and 4 is 5, since a node can be a descendant of
itself according to the LCA definition.
```

Note:

- All of the nodes' values will be unique.
- p and q are different and both values will exist in the binary tree.



Left returned 2
Right returned null

If both p & q are guranteed in the tree, ~~we check~~ and we checked almost all of the tree, and found only 1, than other is a child of what we found

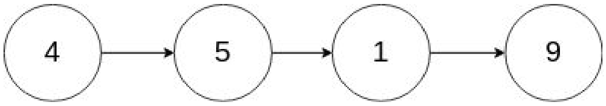
tree we did not explore

We just have to find p. If everything else on the tree does not equal to q, q must be within p.

237. Delete Node in a Linked List

Write a function to delete a node (except the tail) in a singly linked list, given only access to that node.

Given linked list -- head = [4,5,1,9], which looks like following:



Example 1:

```
Input: head = [4,5,1,9], node = 5
Output: [4,1,9]
Explanation: You are given the second node with value 5, the linked list should
become 4 -> 1 -> 9 after calling your function.
```

Example 2:

```
Input: head = [4,5,1,9], node = 1
Output: [4,5,9]
Explanation: You are given the third node with value 1, the linked list should become
4 -> 5 -> 9 after calling your function.
```

Note:

- The linked list will have at least two elements.
- All of the nodes' values will be unique.
- The given node will not be the tail and it will always be a valid node of the linked list.
- Do not return anything from your function.

We have 2 points, next and curr.
Copy next value into current value. Then increment current to curr.next.
Trick is not to increment when curr.next is tail.

238. Product of Array Except Self

Given an array nums of *n* integers where *n* > 1, return an array output such that output[i] is equal to the product of all the elements of nums except nums[i].

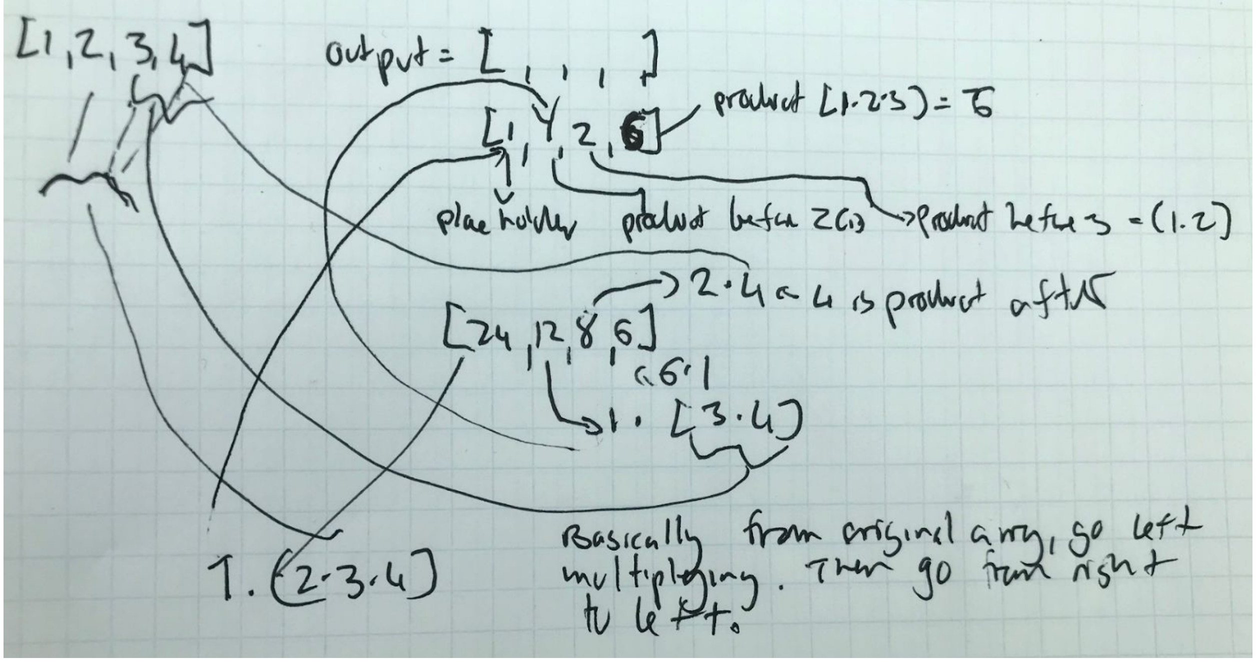
Example:

```
Input:  [1,2,3,4]
Output: [24,12,8,6]
```

Note: Please solve it without division and in O(*n*).

Follow up:

Could you solve it with constant space complexity? (The output array does not count as extra space for the purpose of space complexity analysis.)



239. Sliding Window Maximum

Given an array *nums*, there is a sliding window of size *k* which is moving from the very left of the array to the very right. You can only see the *k* numbers in the window. Each time the sliding window moves right by one position. Return the max sliding window.

Example:

```
Input: nums = [1,3,-1,-3,5,3,6,7], and k = 3
Output: [3,3,5,5,6,7]
Explanation:

Window position                Max
---------------               -----
[1  3  -1] -3  5  3  6  7       3
 1 [3  -1  -3] 5  3  6  7       3
 1  3 [-1  -3  5] 3  6  7      5
 1  3  -1 [-3  5  3] 6  7       5
 1  3  -1  -3 [5  3  6] 7       6
 1  3  -1  -3  5 [3  6  7]      7
```

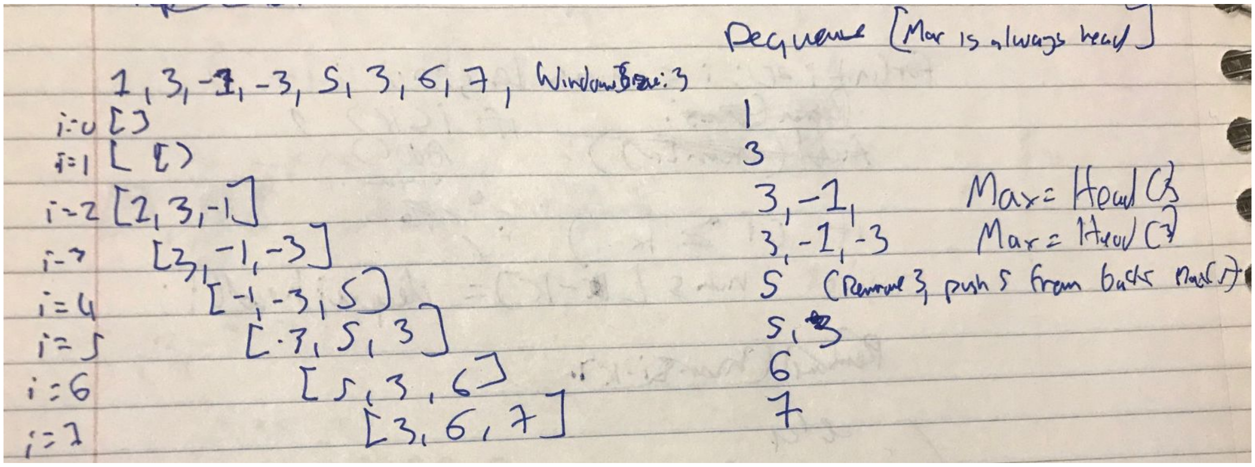
Note:

You may assume *k* is always valid, 1 ≤ k ≤ input array's size for non-empty array.

Follow up:

Could you solve it in linear time?

Use a double linked list as a Dequeue. The Dequeue contains numbers in increasingly order,



Dequeue [Max is always head]

1, 3, -1, -3, 5, 3, 6, 7, WindowSize: 3

i=0	[ ]	1	
i=1	[[)	3	
i=2	[1, 3, -1]	3, -1,	Max= Head (3)
i=3	[3, -1, -3]	3, -1, -3	Max= Head (3)
i=4	[-1, -3, 5]	5	(Remove 3, push 5 from back Max(5))
i=5	[-3, 5, 3]	5, 3	
i=6	[5, 3, 6]	6	
i=7	[3, 6, 7]	7	

242. Valid Anagram

Given two strings *s* and *t* , write a function to determine if *t* is an anagram of *s*.

Example 1:

```
Input: s = "anagram", t = "nagaram"
Output: true
```

Example 2:

```
Input: s = "rat", t = "car"
Output: false
```

Dictionary sucka

250. Count Univalue Subtrees

Given a binary tree, count the number of uni-value subtrees.

A Uni-value subtree means all nodes of the subtree have the same value.

Example :

```
Input:  root = [5,1,5,5,5,null,5]

              5
             / \
            1   5
           / \   \
          5   5   5

Output: 4
```

1) By default, all leaves are univalue subtrees. So we count leaves.
2) Then we propagate the state of the child to parent. I used a tuple and at the child level, Val1 = 5 and Val2 = true. Then on the next level, if we don't have a 5, we propagate back Val = 5, Val2 = false.

252. Meeting Rooms

Given an array of meeting time intervals consisting of start and end times [[s1,e1],[s2,e2],...] ($s_i < e_i$), determine if a person could attend all meetings.

Example 1:

```
Input: [[0,30],[5,10],[15,20]]
Output: false
```

Example 2:

```
Input: [[7,10],[2,4]]
Output: true
```

Order intervals by start time. Then, in a loop, check if prev.end > i.start.

253. Meeting Rooms II

Given an array of meeting time intervals consisting of start and end times [[s1,e1],[s2,e2],...] ($s_i < e_i$), find the minimum number of conference rooms required.

Example 1:

```
Input: [[0, 30],[5, 10],[15, 20]]

Output: 2
```

Example 2:

```
Input: [[7,10],[2,4]]

Output: 1
```

Create an array for startTime [0][5][15]
Create an array for endTime [10][20][30]
Loop through intervals and populate that array. Then sort it. It should look like the arrays above.

Loop through both arrays, each time start < end, increment output. Else decrement output. Keep track when output was max.

257. Binary Tree Paths

Given a binary tree, return all root-to-leaf paths.

Note: A leaf is a node with no children.

Example:

```
Input:


   1

 /   \

2     3

 \

  5


Output: ["1->2->5", "1->3"]
```

```
Explanation: All root-to-leaf paths are: 1->2->5, 1->3
```

PreOrder, add to List<string> when left and right of the node is null;

266. Palindrome Permutation

Given a string, determine if a permutation of the string could form a palindrome.

Example 1:

```
Input: "code"
Output: false
```

Example 2:

```
Input: "aab"
Output: true
```

Example 3:

```
Input: "carerac"
Output: true
```

Add to hashset, remove from hashset. If hashset.count == 0 or 1, it’s a Palindrome permutation.

268. Missing Number

Given an array containing *n* distinct numbers taken from `0, 1, 2, ..., n`, find the one that is missing from the array.

Example 1:

```
Input: [3,0,1]
Output: 2
```

Example 2:

```
Input: [9,6,4,2,3,5,7,0,1]
Output: 8
```

Note:

Your algorithm should run in linear runtime complexity. Could you implement it using only constant extra space complexity?

Obviously, this can be easily done using a HashSet.
Add all numbers between 0 < nums.Length to Hashset. Then loop through array, if number is not in hashset, we have a winner.
If Hashset is empty, then we know nums.Length is missing.

If we need to do this without any space complexity, we need to use Gauss formula:
int expectedSum = nums.Length * (nums.Length + 1) / 2;

Then loop through array and see what the actual sum it. The difference is the missing number. If they are equal, the difference is zero, which mean array is missing 0. Again this is the difference.

If you don’t want to use Gauss, we can add all the numbers in a loop, then add actual numbers. The difference is the missing number.

269. Alien Dictionary

There is a new alien language which uses the latin alphabet. However, the order among letters are unknown to you. You receive a list of non-empty words from the dictionary, where words are sorted lexicographically by the rules of this new language. Derive the order of letters in this language.

Example 1:

```
Input:
[
  "wrt",
  "wrf",
  "er",
  "ett",
  "rftt"
]
```

```
Output: "wertf"
```

Example 2:

```
Input:
[
  "z",
  "x"
]

Output: "zx"
```

Example 3:

```
Input:
[
  "z",
  "x",
  "z"
]

Output: ""

Explanation: The order is invalid, so return "".
```

Note:

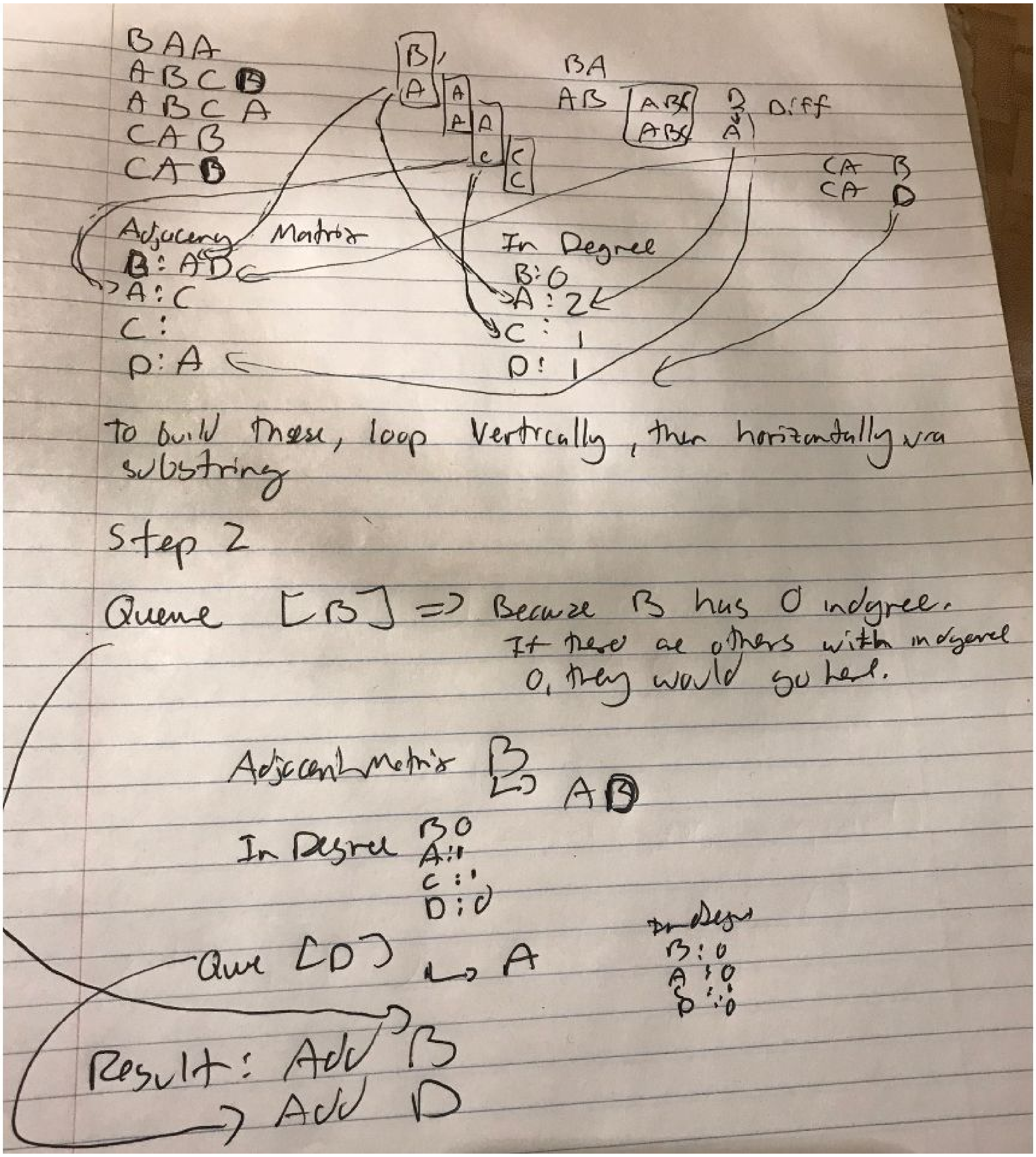
1. You may assume all letters are in lowercase.
2. You may assume that if a is a prefix of b, then a must appear before b in the given dictionary.
3. If the order is invalid, return an empty string.
4. There may be multiple valid order of letters, return any one of them is fine.

Build an adjacency matrix and in-degree list. Afterwards do a BFS on queue. Items with 0 indegree goes to the queue. Then reduce the count of indegree. If indegree count is zero, push that item to queue.

For "wrt", "wrf", "er", "ett", "rftt": : here is the adjacency matrix and indegree shit.

Adjacency
w:e
r:t
t:f
f:
e:r

InDegree
w:0
r:1
t:1
f:1
e:f



BAA
ABCD
ABCA
CAB
CAD

B
A

A
A
A
C
C
C

BA
AB
ABC
ABC
B
A
Diff

CA B
CA D

Adjacency Matrix
B: ADC
A: C
C:
D: A

In Degree
B: 0
A: 2
C: 1
D: 1

To build these, loop vertically, then horizontally via substring

Step 2

Queue [B] => Because B has 0 indegree. If there are others with indegree 0, they would go here.

Adjacent Matrix B → AD

In Degree B 0
A: 1
C: 1
D: 0

Que [D] → A

In Degree
B: 0
A: 0
D: 0

Result: Add B
Add D

270. Closest Binary Search Tree Value

Given a non-empty binary search tree and a target value, find the value in the BST that is closest to the target.

Note:

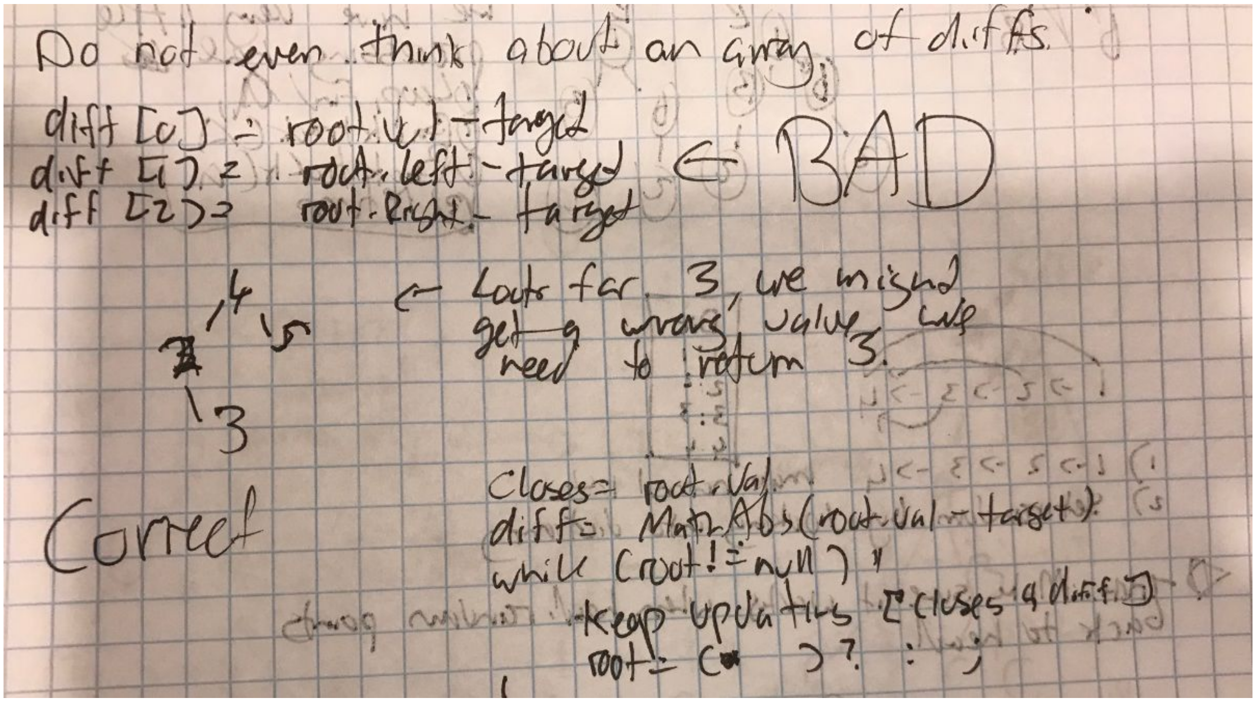
- Given target value is a floating point.
- You are guaranteed to have only one unique value in the BST that is closest to the target.

Example:

```
Input: root = [4,2,5,1,3], target = 3.714286
```

```
    4
   / \
  2   5
 / \
1   3
```

```
Output: 4
```



Do not even think about an array of diffs.

diff [0] = root.val - target
diff [1] = root.left - target
diff [2] = root.Right - target ← BAD

← Look for 3, we might get a wrong value. we need to return 3.

Correct

```
Closes= root.val
diff= Math.Abs(root.val - target)
while (root!=null) {
    keep updating [closes & diff]
    root= (      ) ?   :   ;
```

273. Integer to English Words

Convert a non-negative integer to its english words representation. Given input is guaranteed to be less than 2^{31} - 1.

Example 1:

```
Input: 123
Output: "One Hundred Twenty Three"
```

Example 2:

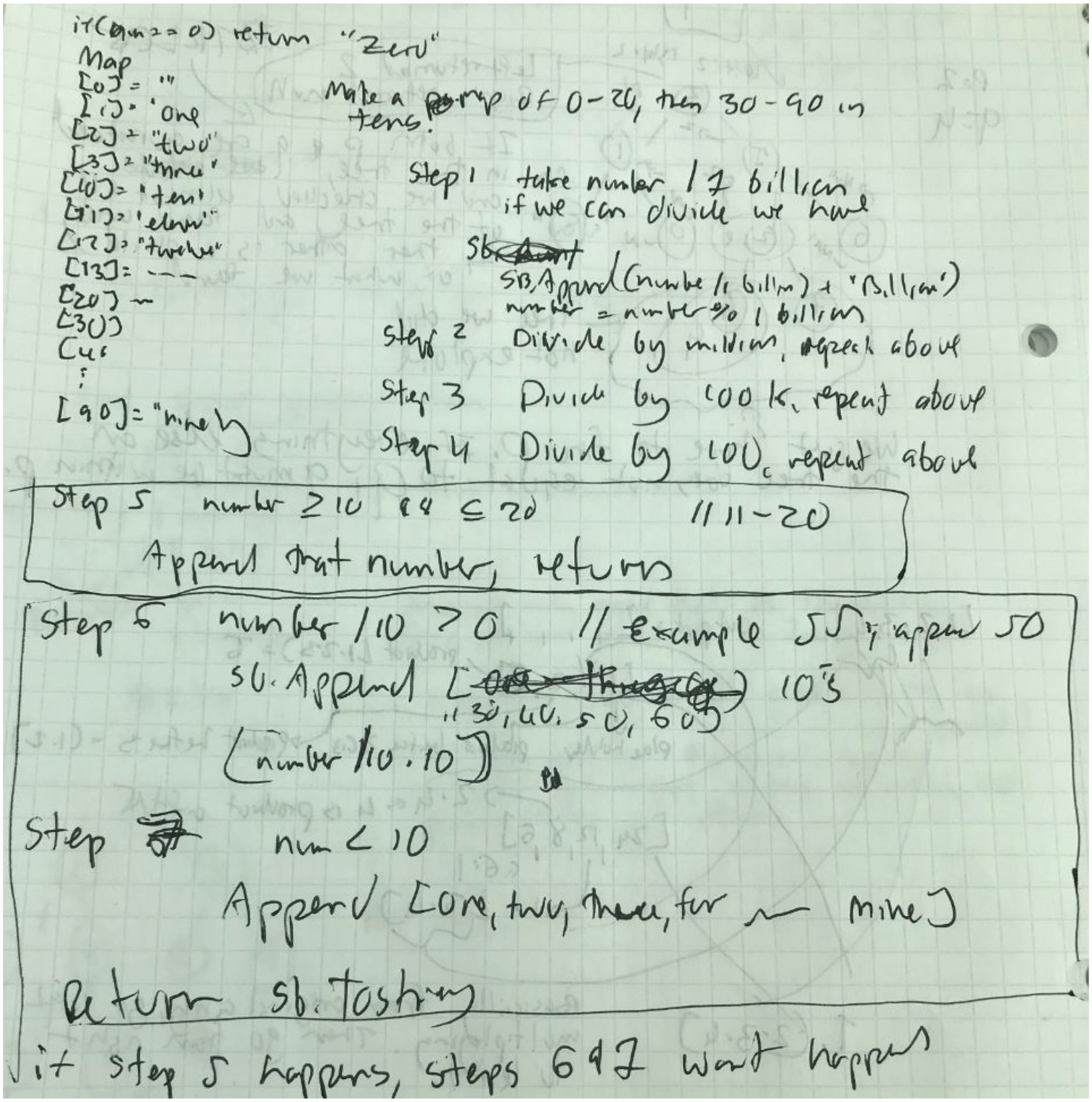
```
Input: 12345
Output: "Twelve Thousand Three Hundred Forty Five"
```

Example 3:

```
Input: 1234567
Output: "One Million Two Hundred Thirty Four Thousand Five Hundred Sixty Seven"
```

Example 4:

```
Input: 1234567891
Output: "One Billion Two Hundred Thirty Four Million Five Hundred Sixty Seven
Thousand Eight Hundred Ninety One"
```



```
if(num == 0) return "Zero"
Map
[0] = ""
[1] = 'one
[2] = "two"
[3] = "three'
[10] = 'ten'
[11] = 'eleven"
[12] = "twelve"
[13] = ---
[20] ~
[30]
[40
  :
[90] = "ninety

Make a ~~p~~ map of 0-20, then 30-90 in
  tens!

Step 1   take number / 1 billion
         if we can divide we have
         sb
         SB.Append(number / 1 billion) + "Billion")
         number = number % 1 billion
Step 2   Divide by million, repeat above
Step 3   Divide by 100 k, repeat above
Step 4   Divide by 100, repeat above

Step 5   number ≥ 10 && ≤ 20          // 11-20
      Append that number, return

Step 6   number / 10 > 0    // example 55, append 50
         sb.Append [~~one three~~] 10's
                   "30, 40, 50, 60]
         [number / 10 . 10]]

Step 7      num < 10
         Append [one, two, three, four ~ nine]

Return sb.toString
if step 5 happens, steps 6 & 7 won't happen
```

276. Paint Fence

There is a fence with n posts, each post can be painted with one of the k colors.

You have to paint all the posts such that no more than two adjacent fence posts have the same color.

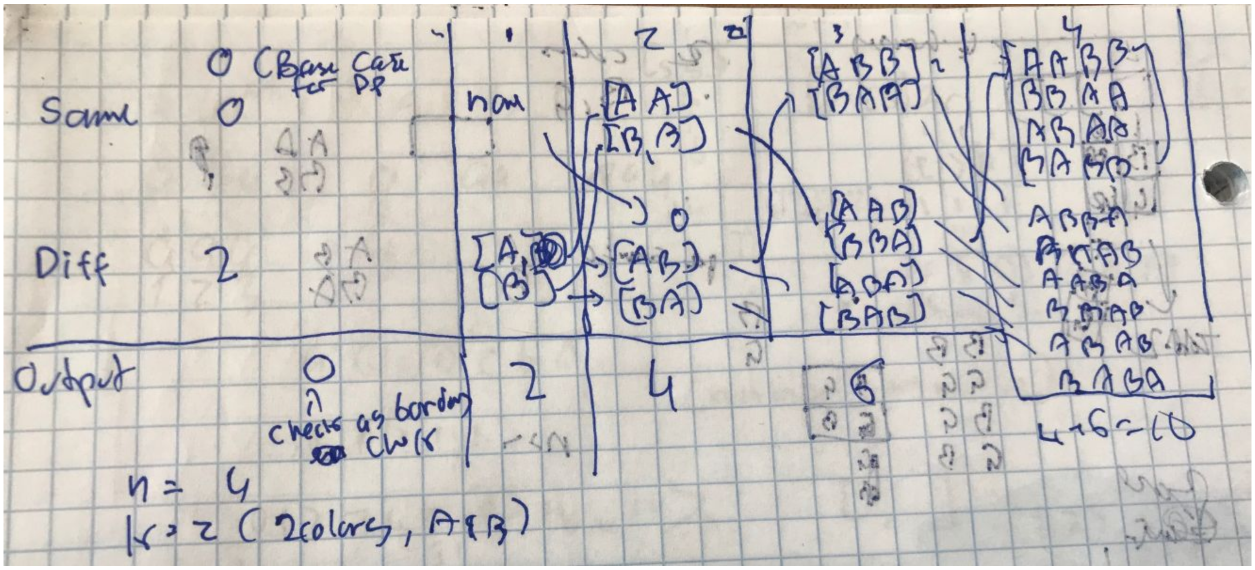
Return the total number of ways you can paint the fence.

Note:

n and k are non-negative integers.

Example:

```
Input: n = 3, k = 2
Output: 6
Explanation: Take c1 as color 1, c2 as color 2. All possible ways are:

            post1  post2  post3
 -----      -----  -----  -----
   1         c1     c1     c2
   2         c1     c2     c1
   3         c1     c2     c2
   4         c2     c1     c1
   5         c2     c1     c2
   6         c2     c2     c1
```



Same = diff[i-1]
Diff = (k-1) * Same[i-1] + (k - 1) * diff[i - 1]

Different is basically

- ones less than times previous same
- One less than previous diff

Better illustration might have been if we have more than 2 colors. One less makes sense. If previous is AA, then we can not have AAA, so A is out for the next round.

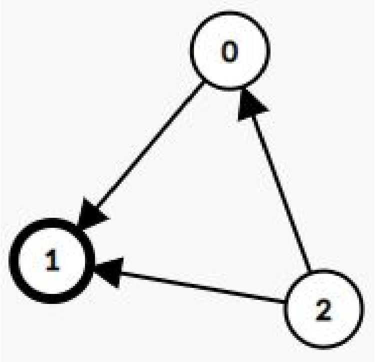
277. Find the Celebrity

Suppose you are at a party with `n` people (labeled from `0` to `n - 1`) and among them, there may exist one celebrity. The definition of a celebrity is that all the other `n - 1` people know him/her but he/she does not know any of them.

Now you want to find out who the celebrity is or verify that there is not one. The only thing you are allowed to do is to ask questions like: "Hi, A. Do you know B?" to get information of whether A knows B. You need to find out the celebrity (or verify there is not one) by asking as few questions as possible (in the asymptotic sense).

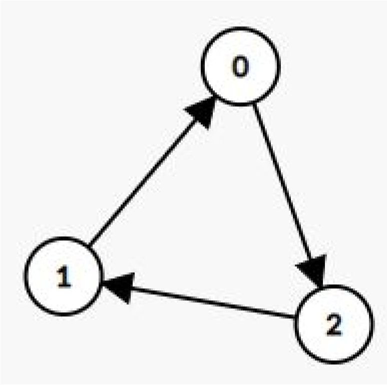
You are given a helper function `bool knows(a, b)` which tells you whether A knows B. Implement a function `int findCelebrity(n)`. There will be exactly one celebrity if he/she is in the party. Return the celebrity's label if there is a celebrity in the party. If there is no celebrity, return `-1`.

Example 1:



```
Input: graph = [
  [1,1,0],
  [0,1,0],
  [1,1,1]
]
Output: 1
Explanation: There are three persons labeled with 0, 1 and 2. graph[i][j] = 1 means
person i knows person j, otherwise graph[i][j] = 0 means person i does not know
person j. The celebrity is the person labeled as 1 because both 0 and 2 know him but
1 does not know anybody.
```

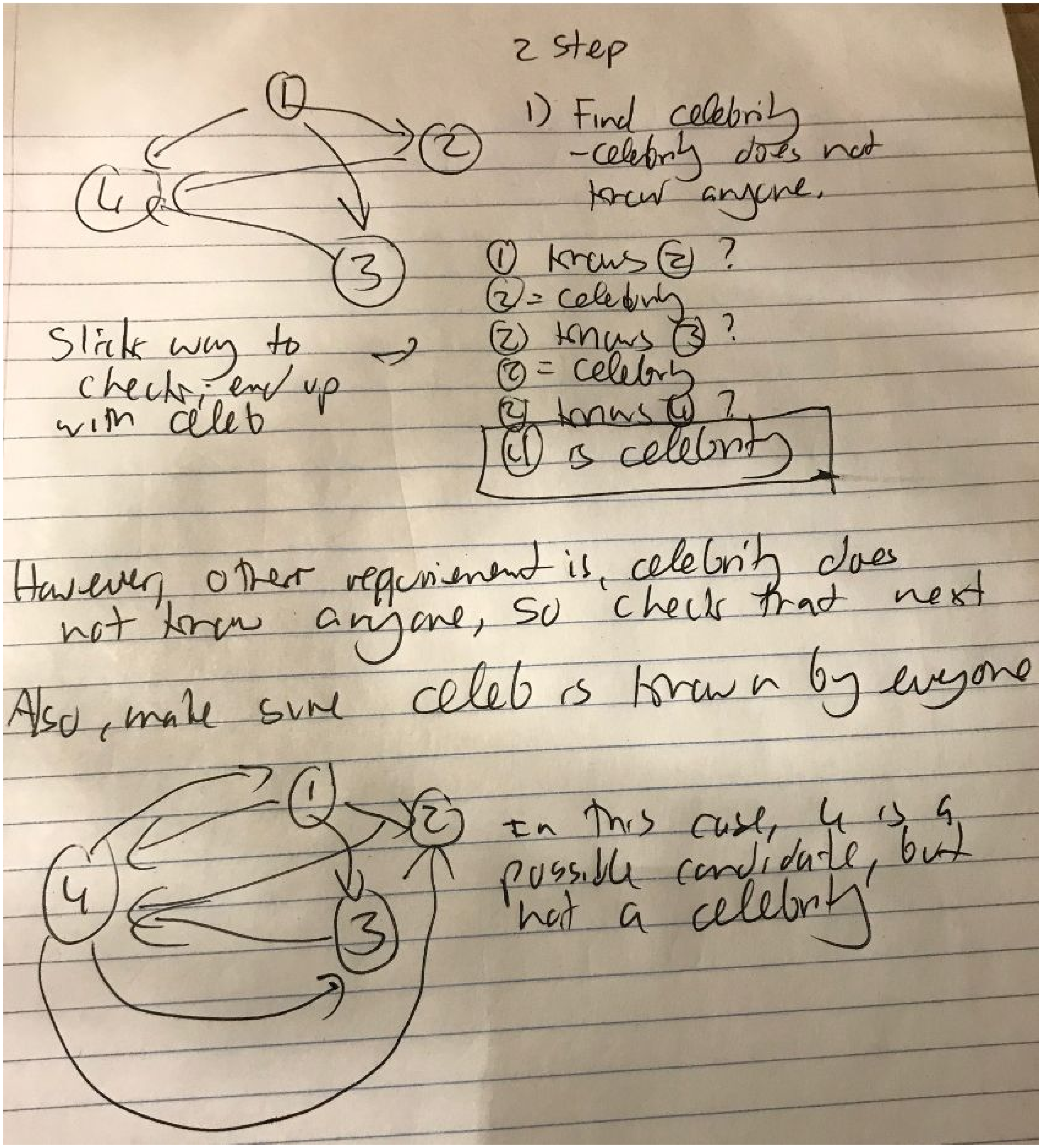
Example 2:



```
Input: graph = [
  [1,0,1],
  [1,1,0],
  [0,1,1]
]
Output: -1
Explanation: There is no celebrity.
```

Note:

1. **The directed graph is represented as an adjacency matrix, which is an `n x n` matrix where `a[i][j] = 1` means person `i` knows person `j` while `a[i][j] = 0`means the contrary.**
2. **Remember that you won't have direct access to the adjacency matrix.**



2 step

1) Find celebrity
-celebrity does not know anyone.

① knows ② ?
② = celebrity
② knows ③ ?
② = celebrity
② knows ④ ?
④ is celebrity

Slick way to check, end up with celeb

However, other requirement is, celebrity does not know anyone, so checks that next

Also, make sure celeb is known by everyone

In this case, 4 is a possible candidate, but not a celebrity

278. First Bad Version

You are a product manager and currently leading a team to develop a new product. Unfortunately, the latest version of your product fails the quality check. Since each version is developed based on the previous version, all the versions after a bad version are also bad.

Suppose you have n versions [1, 2, ..., n] and you want to find out the first bad one, which causes all the following ones to be bad.

You are given an API bool isBadVersion(version) which will return whether versionis bad. Implement a function to find the first bad version. You should minimize the number of calls to the API.

Example:

```
Given n = 5, and version = 4 is the first bad version.


call isBadVersion(3) -> false

call isBadVersion(5) -> true

call isBadVersion(4) -> true


Then 4 is the first bad version.
```

Binary Search, while (start < end) and return start when loop breaks. No need to check mid - 1 crap.

279. Perfect Squares

Given a positive integer *n*, find the least number of perfect square numbers (for example, 1, 4, 9, 16, ...) which sum to *n*.

Example 1:

```
Input: n = 12

Output: 3

Explanation: 12 = 4 + 4 + 4.
```

Example 2:

```
Input: n = 13

Output: 2

Explanation: 13 = 4 + 9.
```

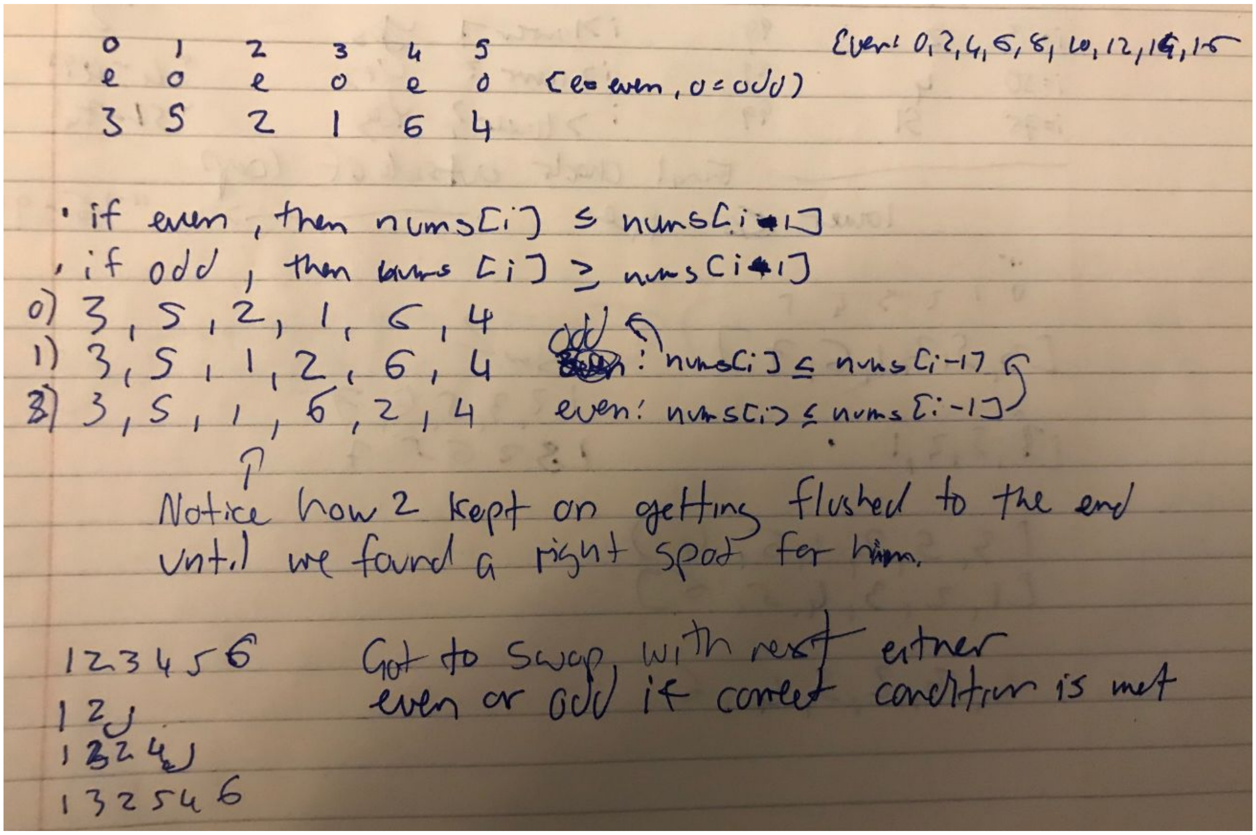
Can not understand the DP Solution. Look it up again when I am in a better state.

280. Wiggle Sort

Given an unsorted array nums, reorder it in-place such that nums[0] <= nums[1] >= nums[2] <= nums[3]....

Example:

```
Input: nums = [3,5,2,1,6,4]
Output: One possible answer is [3,5,1,6,2,4]
```



0	1	2	3	4	5
e	o	e	o	e	o
3	5	2	1	6	4

(e= even, o=odd)

Even: 0, 2, 4, 6, 8, 10, 12, 14, 16

- if even, then nums[i] ≤ nums[i+1]
- if odd, then nums [i] ≥ nums[i+1]

0) 3, 5, 2, 1, 6, 4
1) 3, 5, 1, 2, 6, 4   odd: nums[i] ≤ nums[i-1]
3) 3, 5, 1, 6, 2, 4   even: nums[i] ≤ nums[i-1]

Notice how 2 kept on getting flushed to the end until we found a right spot for him.

123456
12
1324
132546

Got to swap with next either even or odd if correct condition is met

283. Move Zeroes

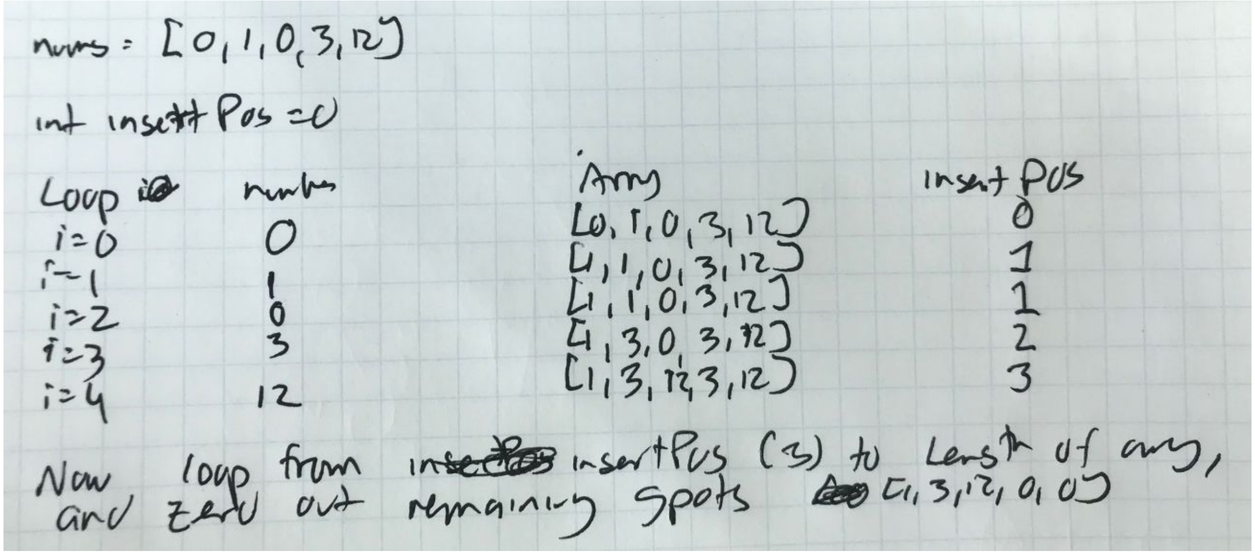
Given an array nums, write a function to move all 0's to the end of it while maintaining the relative order of the non-zero elements.

Example:

```
Input: [0,1,0,3,12]

Output: [1,3,12,0,0]
```

Note:

1. You must do this in-place without making a copy of the array.
2. Minimize the total number of operations.



nums = [0,1,0,3,12]

int insertPos =0

Loop	number	Array	insert Pos
i=0	0	[0,1,0,3,12]	0
i=1	1	[1,1,0,3,12]	1
i=2	0	[1,1,0,3,12]	1
i=3	3	[1,3,0,3,12]	2
i=4	12	[1,3,12,3,12]	3

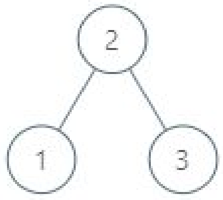
Now loop from insertPos (3) to Length of arry, and zero out remaining spots [1,3,12,0,0]

285. Inorder Successor in BST

Given a binary search tree and a node in it, find the in-order successor of that node in the BST.

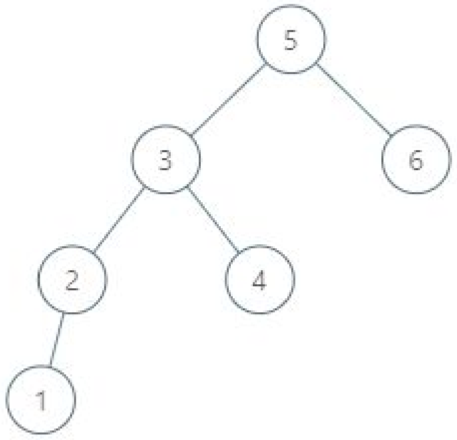
The successor of a node `p` is the node with the smallest key greater than `p.val`.

Example 1:



```
Input: root = [2,1,3], p = 1
Output: 2
Explanation: 1's in-order successor node is 2. Note that both p and the return value
is of TreeNode type.
```

Example 2:



```
Input: root = [5,3,6,2,4,null,null,1], p = 6
Output: null
Explanation: There is no in-order successor of the current node, so the answer is
null.
```

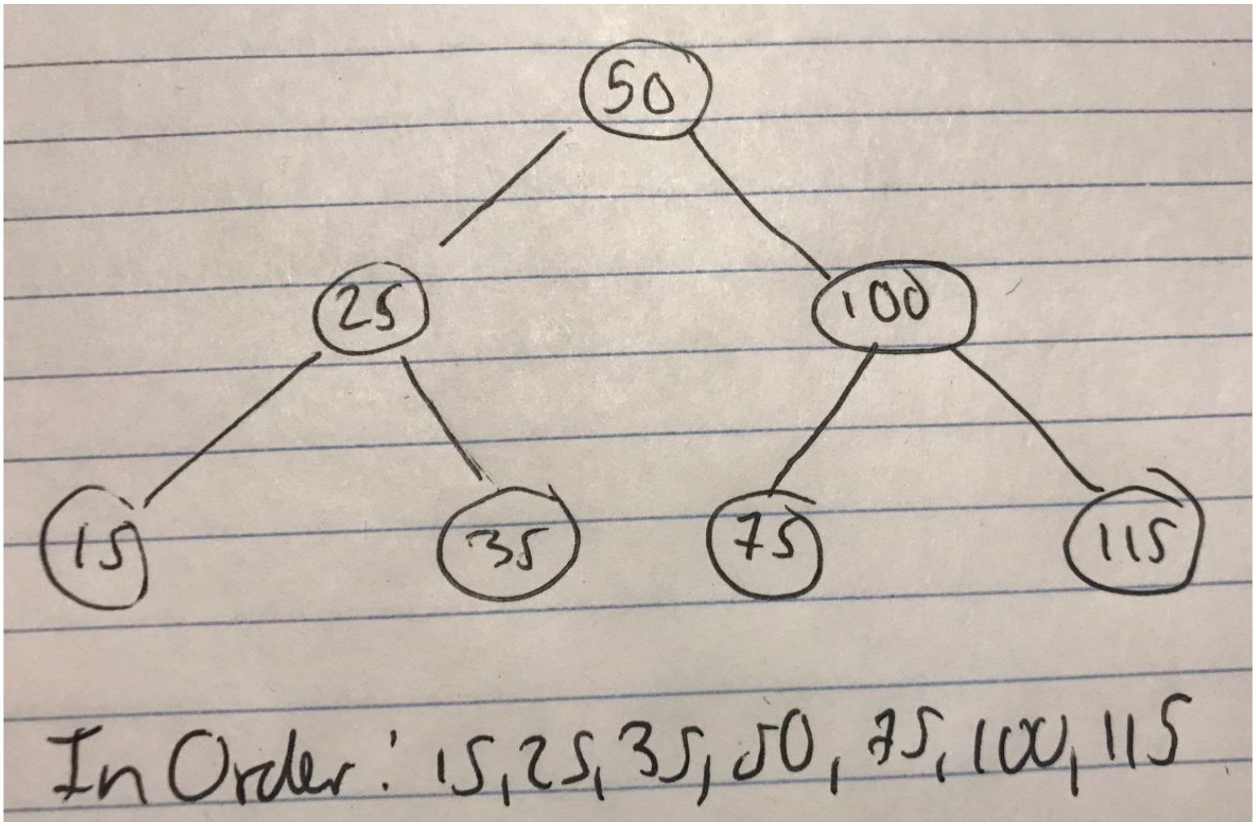
Note:

1. **If the given node has no in-order successor in the tree, return `null`.**

2. **It's guaranteed that the values of the tree are unique.**

Super Easy, BST, so In Order will give you the sorted data. Just do an In Order search and return the **first** value where root.val > p.val;

Example of how In Order looks.



286. Walls and Gates

You are given a *m x n* 2D grid initialized with these three possible values.

1. -1 - A wall or an obstacle.
2. 0 - A gate.
3. INF - Infinity means an empty room. We use the value $2^{31} - 1 = 2147483647$ to represent INF as you may assume that the distance to a gate is less than 2147483647.

Fill each empty room with the distance to its *nearest* gate. If it is impossible to reach a gate, it should be filled with INF.

Example:

Given the 2D grid:

```
INF  -1  0  INF
INF INF INF  -1
INF  -1 INF  -1
  0  -1 INF INF
```

After running your function, the 2D grid should be:

```
 3  -1   0   1
  2   2   1  -1
  1  -1   2  -1
  0  -1   3   4
```

Push all gates into queue first. Then for each gate update its neighbor cells and push them to the queue.

Repeating above steps until there is nothing left in the queue.

287. Find the Duplicate Number

Given an array *nums* containing *n* + 1 integers where each integer is between 1 and *n* (inclusive), prove that at least one duplicate number must exist. Assume that there is only one duplicate number, find the duplicate one.

Example 1:

```
Input: [1,3,4,2,2]
Output: 2
```

Example 2:

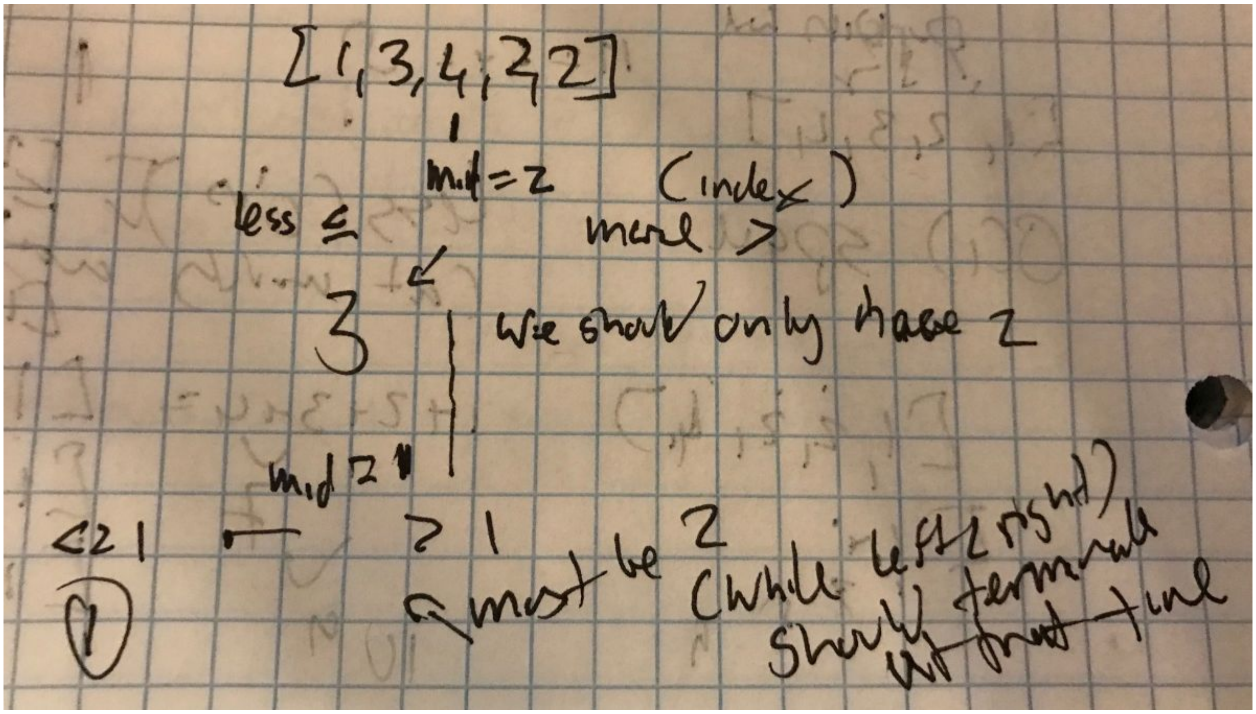
```
Input: [3,1,3,4,2]
Output: 3
```

Note:

1. You must not modify the array (assume the array is read only).
2. You must use only constant, *O*(1) extra space.
3. Your runtime complexity should be less than *O*(*n*2).
4. There is only one duplicate number in the array, but it could be repeated more than once.

At first the search space is numbers between 1 to n. Each time I select a number mid (which is the one in the middle) and count all the numbers equal to or less than mid. Then if the count is more than mid, the search space will be [1 mid] otherwise [mid+1 n]. I do this until search space is only one number.

Let's say n=10 and I select mid=5. Then I count all the numbers in the array which are less than equal mid. If the there are more than 5 numbers that are less than 5, then by Pigeonhole Principle (https://en.wikipedia.org/wiki/Pigeonhole_principle) one of them has occurred more than once. So I shrink the search space from [1 10] to [1 5]. Otherwise the duplicate number is in the second half so for the next step the search space would be [6 10].



289. Game of Life

According to the Wikipedia's article: "The Game of Life, also known simply as Life, is a cellular automaton devised by the British mathematician John Horton Conway in 1970."

Given a *board* with *m* by *n* cells, each cell has an initial state *live* (1) or *dead* (0). Each cell interacts with its eight neighbors (horizontal, vertical, diagonal) using the following four rules (taken from the above Wikipedia article):

1. Any live cell with fewer than two live neighbors dies, as if caused by under-population.
2. Any live cell with two or three live neighbors lives on to the next generation.
3. Any live cell with more than three live neighbors dies, as if by over-population..
4. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

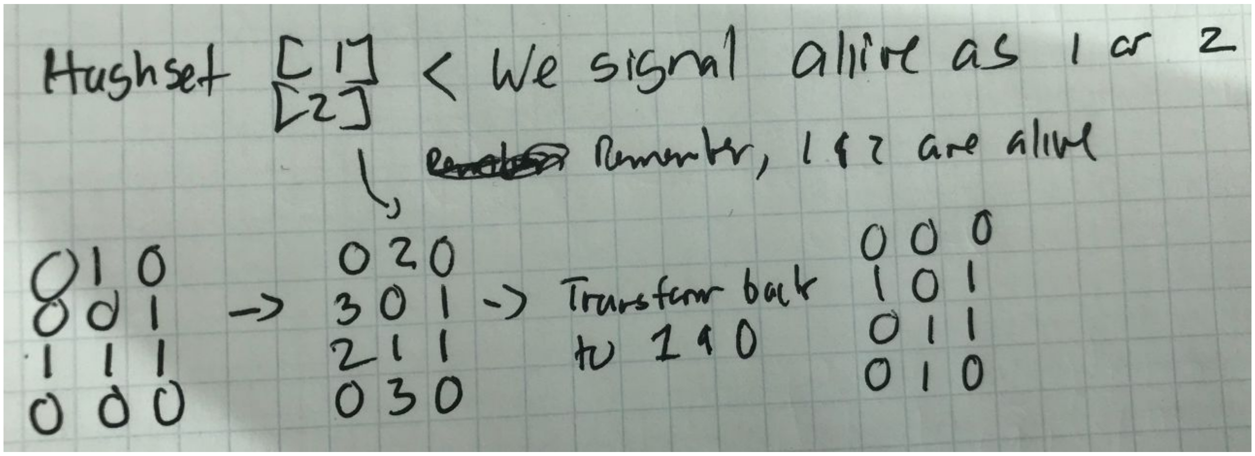
Write a function to compute the next state (after one update) of the board given its current state. The next state is created by applying the above rules simultaneously to every cell in the current state, where births and deaths occur simultaneously.

Example:

```
Input:
[
  [0,1,0],
  [0,0,1],
  [1,1,1],
  [0,0,0]
]
Output:
[
  [0,0,0],
  [1,0,1],
  [0,1,1],
  [0,1,0]
]
```

Follow up:

1. Could you solve it in-place? Remember that the board needs to be updated at the same time: You cannot update some cells first and then use their updated values to update other cells.
2. In this question, we represent the board using a 2D array. In principle, the board is infinite, which would cause problems when the active area encroaches the border of the array. How would you address these problems?



Hashset [1] < We signal alive as 1 or 2
[2]

Remember, 1 & 2 are alive

```
                      0 0 0
0 1 0       0 2 0
0 0 1  ->   3 0 1  -> Transform back   1 0 1
1 1 1       2 1 1      to 1 & 0        0 1 1
0 0 0       0 3 0                      0 1 0
```

295. Find Median from Data Stream

Median is the middle value in an ordered integer list. If the size of the list is even, there is no middle value. So the median is the mean of the two middle value.

For example,
[2,3,4], the median is 3

[2,3], the median is (2 + 3) / 2 = 2.5

Design a data structure that supports the following two operations:

- void addNum(int num) - Add a integer number from the data stream to the data structure.
- double findMedian() - Return the median of all elements so far.

Example:

```
addNum(1)

addNum(2)

findMedian() -> 1.5

addNum(3)

findMedian() -> 2
```

Follow up:

1. If all integer numbers from the stream are between 0 and 100, how would you optimize it?

2. If 99% of all integer numbers from the stream are between 0 and 100, how would you optimize it?

Use a List<int>. For add, I did a binary search on the list and added. For FindMedian, it was getting the middle element of list (list.cout / 2) or getting 2 elements and taking their average if the list is even.

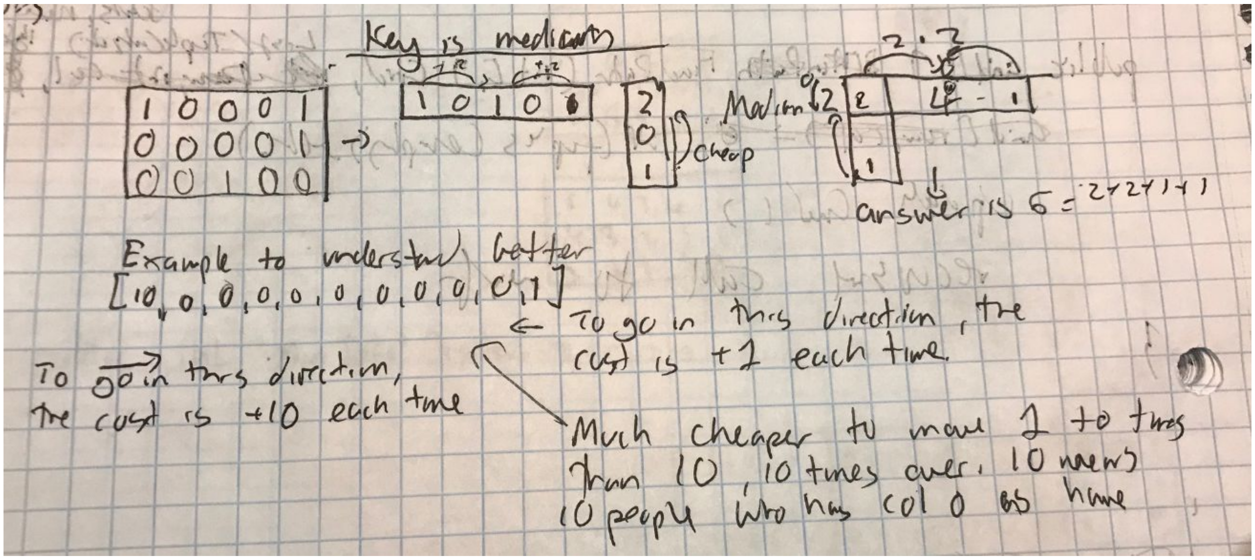
Other solutions used a priority queue to add (before finding median).

296. Best Meeting Point

A group of two or more people wants to meet and minimize the total travel distance. You are given a 2D grid of values 0 or 1, where each 1 marks the home of someone in the group. The distance is calculated using Manhattan Distance, where distance(p1, p2) = |p2.x - p1.x| + |p2.y - p1.y|.

Example:

```
Input:

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

Output: 6

Explanation: Given three people living at (0,0), (0,4), and (2,2):
             The point (0,2) is an ideal meeting point, as the total travel distance
             of 2+2+2=6 is minimal. So return 6.
```



Example to understand better
[10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

To go in this direction, the cost is +10 each time

To go in this direction, the cost is +1 each time.

Much cheaper to move 1 to this than 10, 10 times over. 10 means 10 people who has col 0 as home

The answer I came up with worked but it was not the fastest. Much better to find the total people starting in row, total people starting in column. Then get the distance to median.

297. Serialize and Deserialize Binary Tree

Serialization is the process of converting a data structure or object into a sequence of bits so that it can be stored in a file or memory buffer, or transmitted across a network connection link to be reconstructed later in the same or another computer environment.

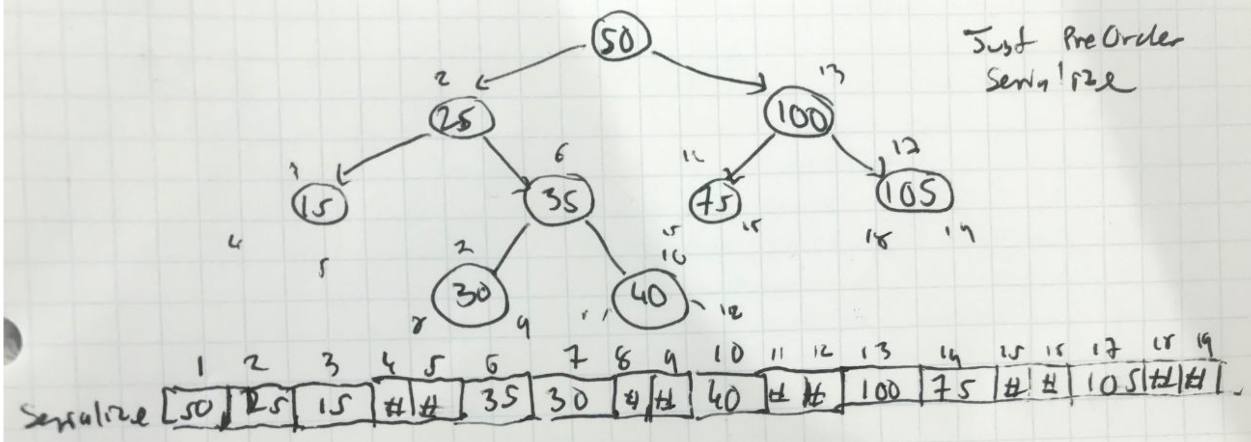
Design an algorithm to serialize and deserialize a binary tree. There is no restriction on how your serialization/deserialization algorithm should work. You just need to ensure that a binary tree can be serialized to a string and this string can be deserialized to the original tree structure.

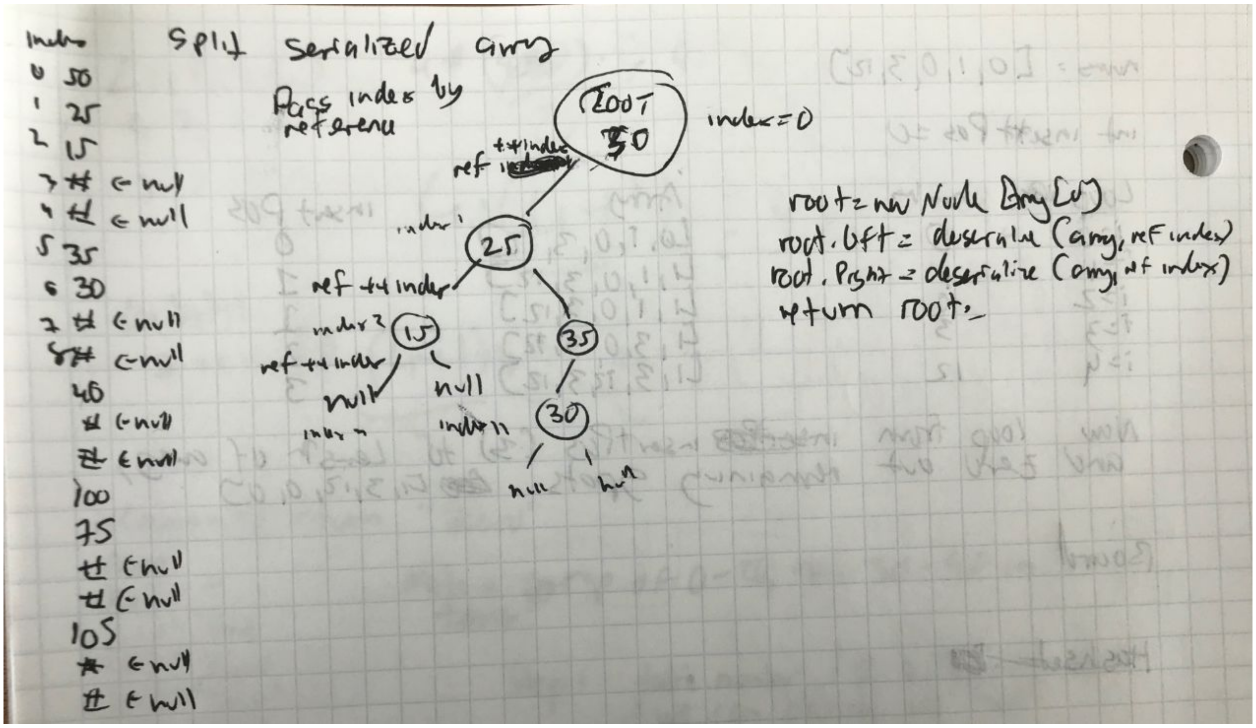
Example:

```
You may serialize the following tree:

    1
   / \
  2   3
     / \
    4   5

as "[1,2,3,null,null,4,5]"
```

Clarification: The above format is the same as how LeetCode serializes a binary tree. You do not necessarily need to follow this format, so please be creative and come up with different approaches yourself.

Note: Do not use class member/global/static variables to store states. Your serialize and deserialize algorithms should be stateless.





```
root = new Node(Array[0])
root.left = deserialize(array, ref index)
root.Right = deserialize(array, ref index)
return root;
```