HW1

Zip

☆ Zip!

Zip a linked list from it two ends.

e.g.
Input: 1->2->3->4->5->6
Output: 1->6->2->5->3->4

Suggested time: 30 minutes

Solution: http://programming-puzzle.blogspot.com/2014/02/zip-of-linked-list.html

[Extra credit: Zip two separate lists and unzip them back into original lists. i.e. unzip(zip(L1,L2)) should return L1 and L2]

Sliding Window Maximum

☆ Sliding Window Maximum

(Description courtesy of Leetcode, otherwise a popular problem)

A long array A[] is given to you. There is a sliding window of size *w* which is moving from the very left of the array to the very right. You can only see the *w* numbers in the window. Each time the sliding window moves rightwards by one position. Following is an example: The array is [1 3 -1 -3 5 3 6 7], and *w* is 3.

```
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
```

Input: A long array A[], and a window width w
Output: An array B[], B[i] is the maximum value of from A[i] to A[i+w-1]
Requirement: Find a good optimal way to get B[i]

Hint: Modify a Queue

Solution: http://articles.leetcode.com/2011/01/sliding-window-maximum.html

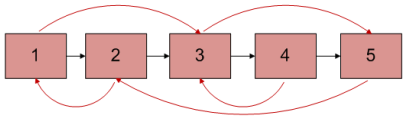
Clone a special list

☆ Clone a special list

(Courtesy, GfG)

You are given a Double Link List with one pointer of each node pointing to the next node just like in a single link list. The second pointer however CAN point to any node in the list and not just the previous node. Now write a program in **O(n) time** to duplicate this list. That is, write a program which will create a copy of this list.

Let us call the second pointer as arbit pointer as it can point to any arbitrary node in the linked list.



(Courtesy, GfG)

Like most linked list problems, please do this with constant extra memory.

Solution: http://www.geeksforgeeks.org/a-linked-list-with-next-and-arbit-pointer/

Merge a sorted linked list

☆ Merge Sort a Linked List

Sort a singly linked list using merge sort. Sort ascending.

(Trivia question: Merge Sort, in general is a good method of sorting anything, because its worst case performance is better than most other sorting methods. For Linked Lists however, Merge Sort is highly preferred, because it has a distinct advantage. What is that advantage?)

Solution: https://docs.google.com/document/d/1UBFI633FsCZLq2Vejihj2g80Vdl9BNF6-N1D4J-clsI/edit?usp=sharing

SuperStack

☆ Super Stack

You are given an empty stack. Your task is to perform following three operations:

- *push a*: Push an integer a, to the top of the stack
- *pop*: Pop the top element from the stack. It is guaranteed that stack is not empty, when performing the pop operation.
- *inc x d*: Add d to bottom x elements of the stack.

After each operation, print the top element of the stack, if after an operation, the stack becomes empty, then print *EMPTY*.

Input Format:
The first line of the input is n, total number of operations performed on the stack. Each of the next n lines is one of the three operations listed above.

Constraints:

- $0 \leq \boldsymbol{n} \leq 2 \times 10^5$
- $-10^9 \leq \boldsymbol{a}, \boldsymbol{d} \leq 10^9$
- $1 \leq \boldsymbol{x} \leq$ size of the stack at the time of the operation
- There is no pop operation when the stack is empty.

Output Format:
For each operation, output the top element of the stack. If after an operation, the stack becomes empty, print EMPTY.

Sample Input

```
12
push 4
pop
push 3
push 5
push 2
inc 3 1
pop
push 1
inc 2 2
push 4
pop
pop
```

Sample Output

```
4
EMPTY
3
5
2
3
6
1
1
4
1
8
```

Explanation
Assume S is the stack, initially *S = []*. Also, the leftmost element in the S is the bottom most element and the right most element is the top element of the stack.

push 4: We push 4 on the top of the stack, so the stack is now *S = [4]*. The top element is 4, so we print *4* after this operation.
pop: We pop the top element from the stack, so the stack is now *S = []*. The stack is empty, so we print *EMPTY* after this operation.
push 3: We push 3 on the top of the stack, so the stack is now *S = [3]*. The top element is 3, so we print *3* after this operation.
push 5: We push 5 on the top of the stack, so the stack is now *S = [3, 5]*. The top element is 5, so we print *5* after this operation.
push 2: We push 2 on the top of the stack, so the stack is now *S = [3, 5, 2]*. The top element is 2, so we print *2* after this operation.
inc 3 1: We add 1 to bottom 3 elements of the stack, so the stack is now *S = [4, 6, 3]*. The top element is 3, so we print *3* after this operation.
pop: We pop the top element from the stack, so the stack is now *S = [4, 6]*. The top element is 6, so we print *6* after this operation.
push 1: We push 1 on the top of the stack, so the stack is now *S = [4, 6, 1]*. The top element is 1, so we print *1* after this operation.
inc 2 2: We add 2 to bottom 2 elements of the stack, so the stack is now *S = [6, 8, 1]*. The top element is 1, so we print *1* after this operation.
push 4: We push 4 on the top of the stack, so the stack is now *S = [6, 8, 1, 4]*. The top element is 4, so we print *4* after this operation.
pop: We pop the top element from the stack, so the stack is now *S = [6, 8, 1]*. The top element is 1, so we print *1* after this operation.
pop: We pop the top element from the stack, so the stack is now *S = [6, 8]*. The top element is 8, so we print *8* after this operation.

Design and implement LRU cache

☆ Design and implement an LRU cache

LRU cache evicts the least recently used item from the cache. Item insertions, lookups and evictions should be very efficient.

Alternative Node Split

☆ Alternative Node Split

Given a linked list, split it into two such that every other node goes into the new list. For lists with odd number of nodes, first one should be longer. For example: an input list: {a, b, c, d, e, f, g} results in {a, c, e, g} and {b, d, f}.

Solution: https://docs.google.com/document/d/1UBFl633FsCZLq2Vejihj2g80Vdl9BNF6-N1D4J-clsl/edit?usp=sharing

Find the middle element in a singly linked list

☆ Find the middle element in a singly linked list

Find the middle element of a singly linked list. Constraint: Do it in one pass over the list. If it is even number of elements, then output the 2nd of the middle two elements.

e.g.

1. *(standard positive case) For input: 1->2->3->nil, Output should be: 2*
2. *(positive case with longer list, odd # of nodes). For input: 1->11->45->12->67->89->91->nil, Output should be 12*
3. *(positive case with longer list, even # of nodes). For input: 1->11->45->12->67->89->nil, Output should be 12 (2nd of the middle two)*
4. *For input: nil, output should be nil*

Solution : https://docs.google.com/document/d/1UBFl633FsCZLq2Vejihj2g80Vdl9BNF6-N1D4J-clsl/edit?usp=sharing

Valid matching params

☆ Valid matching parans

Write a function that checks if the given input string has matching opening and closing parentheses. Valid parantheses are: '(', ')', '{', '}', '[', ']'

Test cases:

1. Input: "((1 + 2) * 3)", Output: True
2. Input: "({ 1 + 2) * 3)", Output: False
3. Input: "(((1 + 2) * 3))", Output: True
4. Input: "{([])}", Output: True
5. Input: "} (1 * 2) + 3 * (5 - 6)": False

Solution: https://docs.google.com/document/d/1UBFl633FsCZLq2Vejihj2g80Vdl9BNF6-N1D4J-clsl/edit?usp=sharing

Longest substring with matching parans

☆ Longest substring with matching parantheses

Find the length of the longest substring that has matching opening and closing parentheses. We only need length, not the substring itself. You may assume valid input for the purpose of this exercise i.e. Input string can only have parentheses and nothing else.

e.g.
1. Input: "((((()(((()", Output: 4, for "(())"
2. Input: "((((((", Output: 0, for ""
3. Input: "()()()", Output: 6, for "()()()"
4 Input: "", Output: 0, for ""

Solution: https://docs.google.com/document/d/1UBFl633FsCZLq2Vejihj2g80Vdl9BNF6-N1D4J-clsI/edit?usp=sharing

Swap kth nodes in Linked List

☆ Swap 'k'th nodes in a Linked List

In a given singly Linked List of length N, swap kth node from the beginning, with kth node from the end. We're not swapping just the contents; we're swapping the nodes themselves. 'K' is understood as based on 1 i.e. (1<= K <=N).

e.g.:
1. Input: 1->2->3->4->7->0, K=2 Output: 1->7->3->4->2->0
2. Input: 1->2->4->7->0, K=3 Output: 1->2->4->7->0
3. Input: 1->2->3->4->7->0, K=5 Output: 1->7->3->4->2->0
4. Input: 4, K=1 Output: K=4

Solution: https://docs.google.com/document/d/1UBFl633FsCZLq2Vejihj2g80Vdl9BNF6-N1D4J-clsI/edit?usp=sharing

Reverse a linked list in groups

☆ Reverse a linked list in groups

Reverse a Linked List in groups of given size

Given a linked list, write a function to reverse every k nodes (where k is an input to the function).

```
Example:
Inputs:  1->2->3->4->5->6->7->8->NULL and k = 3
Output:  3->2->1->6->5->4->8->7->NULL.

Inputs:   1->2->3->4->5->6->7->8->NULL and k = 5
Output:  5->4->3->2->1->8->7->6->NULL.
```

Solution: http://www.geeksforgeeks.org/reverse-a-list-in-groups-of-given-size/

Duplicates in a linked list

☆ Duplicates in Linked List

Problem Statement
Complete the *optimal(Linked List)* function so that it checks a Linked List for redundant nodes, removes them, and returns the modified list without altering the order of non-redundant nodes.

Note: A *redundant node* is a node whose data matches the data of a previous node in the list (e.g.: given some node n_i containing data d_i, if $d_i == d_j$ and $i < j$, node n_j is redundant).

Constraints
Each Linked List input has 10,000 nodes.
$0 \le d_i \le 1000$.

Input Format
A Linked List passed as an argument to the *optimal(Linked List)* function.

Output Format
The *optimal(Linked List)* function should return the updated (non-redundant) Linked List.

Sample Input

```
8
3 4 3 2 6 1 2 6
```

Sample Output

```
3 4 2 6 1
```

Explanation
If we refer to each node as some n_i where $1 \le i \le list\ size$, then we can make the following statements:

- $n_3 = 3$ is redundant to $n_1 = 3$, because both nodes have matching data values and n_3 appears later in the list.
- $n_7 = 2$ is redundant to $n_4 = 2$, because both nodes have matching data values and n_7 appears later in the list.
- $n_8 = 6$ is redundant to $n_5 = 6$, because both nodes have matching data values and n_8 appears later in the list.
- In the output, the redundant nodes have been removed while preserving the initial ordering of all non-redundant elements.

Solution: http://algorithms.tutorialhorizon.com/remove-duplicates-from-an-unsorted-linked-list/ (Obviously, follow the better one here :-))

Implement Min Stack

☆ Implement a Min Stack

Your goal for this problem is to implement an additional method: "getMinimum()" for a stack. The method, as the name suggests, returns the minimum element in the entire stack. It shouldn't pop that element; it should only peek i.e. return the value. If time permits, you may implement other methods of the stack, but for now, focus on getMinimum().

Constraint: getMinimum() should return in constant time.

e.g:
1. If the stack has 1,2,3,4,5 in that order from bottom to top, then the call to getMinimum() should return 1
2. If the stack has 1,5,3,0 in that order from bottom to top, then the call to getMinimum() should return 0
3. If the stack is empty, it should return NULL

[We did this problem in the class, but here is a good solution too: http://articles.leetcode.com/stack-that-supports-push-pop-and-getmin/]

HW1 test

☆ Intersection of two linked lists

Write a function that returns the intersection of two integer singly linked lists. (Intersection of two lists starts at a common node i.e. the node with the same address).

Input: Two lists which may or may not intersect
Output: Data in the first node where intersection begins

Assume that data is all positive integers or zero.

e.g:
1. L1 of length 4, L2 of length 4, they intersect at 3rd node. Output should be data in the 3rd node.
2. L1 of length 4, L2 of length 4, they don't intersect. Output should be -1.
3. L1 of length 10, L2 of length 6, they intersect on 4th node of L2. Output should be data in the 4th node.

e.g.
List 1: 4 ==> 5 ==> 0 ==> 6 ==> 1
List 2: 7 ==> 9 ==> 0 ==> 6 ==> 1

They intersect at node with value 0. After that node, lists are the same, obviously.

Realize that when comparing two nodes, you should be comparing the addresses, and NOT values.

HackerRank doesn't let us create lists that intersect. As a result, please hard-code your own test-cases.

☆ Median

Write a program that takes a pointer to an arbitrary node in a sorted circular singly linked list, and returns the median (data value) of the linked list.

e.g.
List: 5->6->9->33->1
Median: 6

We want a linear solution.

☆ Implement a Queue using stacks

Implement a queue using exactly two stacks.

e.g:
1. Q.enqueue(1), Q.enqueue(2) Q.dequeue() Q.dequeue() should return 1, 2 in that order.
2. Q.enqueue(1), Q.dequeue(), Q.dequeue() should return 1 followed by throwing "-1" (signifying an exception)

Test case Input:
A singly linked list of enqueue/dequeue operations. Positive values are enqueues and negative values are dequeues.
e.g.
1
2
-1
-1

means
Q.enqueue(1), Q.enqueue(2), Q.dequeue and Q.dequeue

Test-case Output:
A singly linked list of dequeued values. e.g. for the above example, you should form a linked list of two values, 1 and 2.
In the case where there is nothing left to dequeue, and you still see a dequeue operation in the test input, feel free to append a node with -1 to your output linked list.

Feel free to:
1. Assume that the values are all integers
2. This is better done as a whiteboard question. So you can copy/paste an implementation of a stack class. It's not too difficult to write one quickly with push/pop, but if you want to save time, you can.

HW2

Tower of Hanoi

☆ Tower of Hanoi

A Tower of Hanoi is a game, that consists of three rods, and a number of disks of different sizes which can slide onto any rod. The game starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top. The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
3. No disk may be placed on top of a smaller disk.

Input: Number of disks in the first rod. e.g if the number is 5, you can assume that disks are arranged with 1 on top of 2, 2 on top of 3 and so on, until 4 on top of 5, with 5 at the bottom.

Output: Series of steps which leads to all disks landing on the third rod. Each step can be represented by printing the number of disks on each rod after the step is taken.

There are no test-cases set, because of the fluid nature of the output. Feel free to hard code the input and display the output on STDOUT.

Solution: http://www.cs.cmu.edu/~cburch/survey/recurse/hanoiimpl.html

(Suggested time: 45 minutes; maybe a little more if you are printing output nicely)

Wildcard

☆ Wildcard

Input: 10?
Output: 101, 100

i.e. ? behaves like a wild-card. There are two possibilities for 10?, when that ? is replaced with either 0 or 1.

Input: 1?0?
Output: 1000, 1001, 1100, 1101

Please write a program that takes given strings as input and produces the suggested output.

Suggested time: 20 minutes.

========
Solution - Pseudo-ish code
========

```
// print the output
if (input.length == i) {
System.out.println(input);
return;
}

// wild card character
if (input[i] == WILD_CHAR) {

input[i] = '0';
// recursively call
wildChar(input, i + 1);

input[i] = '1';
// recursively call
wildChar(input, i + 1);

// set the number back
// WILD_CHAR - ?
input[i] = WILD_CHAR;

} else {
wildChar(input, i + 1);
}
```

========

ExpressionEvaluator

☆ Expression Evaluator

Given a string of integers as input, put between each pair of digits, one of {"", "*", "+"} such that the expression you get will evaluate to K (a number also given as input). Putting an empty string ("") between two numbers means, that the numbers are joined to form a new number (e.g. 1 "" 2 = 12)

Order of integers given as input needs to remain the same.

Input:
1. String of positive integers
2. Target K (given constant)

Output:
All possible strings that evaluate to K

e.g.
If the input string is "222", and your target (K) is "24", two possible answers are:

1. 22+2 (Here, we put the "" operator in between first two 2s and then put a plus between the last two)
2. 2+22 (Here, we put the plus operator between first two 2s and then put the "" operator between the last two)

Realize that precedence of the operators matters. In higher to lower precedence:
1. Join ("")
2. Multiplication (*)
3. Addition (+)

Solution:
Java: http://elementsofprogramminginterviews.com/static/solutions/java/src/main/java/com/epi/AddOperatorsInString.java
or
C++: http://elementsofprogramminginterviews.com/static/solutions/Add_operators_in_string.cc

(There is no real explanation of the solution anywhere, but looking at code should be very instructive. This is the canonical solution by the author of this problem).

(Interview time: 45 minutes; practice time - several hours)

Palindromic Decomposition

☆ Palindromic Decomposition

A "Palindromic Decomposition" of string S, is a decomposition of the string into substrings, such that all those substrings are valid palindromes. A single character is considered a valid palindrome for this problem. Print out all possible palindromic decompositions of a given string.

e.g.
Input: abracadabra

Output:
a|b|r|a|c|a|d|a|b|r|a|
a|b|r|a|c|ada|b|r|a|
a|b|r|aca|d|a|b|r|a|

Solution: http://www.programcreek.com/2013/03/leetcode-palindrome-partitioning-java/

(Suggested time: 45 minutes)

N Queens

☆ N-Queens

Solve the N-queen problem using recursion. (There may be other ways of solving this problem, but for the purpose of this exercise, please use recursion only).

Problem statement: Place N queens on an NxN chessboard, such that no two queens are in line of attack of each other. In chess (and in real life too), a queen can move as far as she pleases, horizontally, vertically, or diagonally.

Input: N
Output: All possible arrangements of N queens on the board. Each arrangement can be represented by a matrix. Print the entire matrix, one per valid arrangement. For example, for N = 4, there are two possible arrangements, that can be printed as:

```
+-+-+-+-+
| | |*| |
+-+-+-+-+
|*| | | |
+-+-+-+-+
| | | |*|
+-+-+-+-+
| |*| | |
+-+-+-+-+

+-+-+-+-+
| |*| | |
+-+-+-+-+
| | | |*|
+-+-+-+-+
|*| | | |
+-+-+-+-+
| | |*| |
+-+-+-+-+
```

Tests: In your main() method, call your function for every N from 1 through 10, and print the output.

Solution: https://leetcode.com/discuss/24717/comparably-concise-java-code

(Suggested Time: 30 minutes; this is only a variation of permutations problem)

Subset of a Set

☆ Subsets of a set

Problem:
Print out all the subsets of a set.

E.g.
{x,y} ==> {{}, {x}, {y}, {x,y}} [Remember, that we are working with sets, so {y,x} is not different from {x,y}, and {x,x} is not a valid subset, unless the input also has two x's]
{1, 2, 3} ==> {{}, {1}, {2}, {3}, {1,2}, {1,3}, {2, 3}, {1,2,3}}

Input: Set, as an array
Output: Subsets in any order.

This problem does not have pre-defined test-cases, in order to give you the flexibility of doing outputs in any order, and in any print format.

Solutions:
http://stackoverflow.com/questions/728972/finding-all-the-subsets-of-a-set OR
http://stackoverflow.com/questions/4640034/calculating-all-of-the-subsets-of-a-set-of-numbers

(Suggested time: 20 minutes)

Diameter of a tree

☆ Diameter of a tree

Calculate the diameter of a tree (not necessarily a binary tree). Diameter is the longest path between two leaves of a tree. A path is the sum total of all distances (weights) attached to all edges between two nodes.

In the examples below, a tree is represented in a specific notation.

e.g. "{0,1,{5,0}}" means:
* it starts with root (0), which has one (1) child, which will follow in braces.
* Inside the braces, it says that the distance (weight) of reaching that first child is 5 and
* that there are no more children after that (0)

You can represent the tree however you like. This is just one convenient way. Other example test-cases:

```
// One node - no diameter
new TestCase("{0,0}", 0),
// One leaf
new TestCase("{0,1,{5,0}}", 5),
// Still one leaf
new TestCase("{0,1,{5,1,{4,1,{7,0}}}}", 16),
// The diameter of the first son is the diameter of the tree
new TestCase("{0,1,{5,2,{8,0},{7,0}}}", 15),
// The diameter of the last son is the diameter of the tree
new TestCase("{0,3,{1,2,{5,0},{7,0}},{1,2,{6,0},{5,0}},{1,2,{10,0},{9,0}}}", 19),
// Now the diameter is between a leaf in the first son and a leaf in the third son
new TestCase("{0,3,{5,2,{8,0},{7,0}},{5,2,{9,0},{8,0}},{5,2,{10,0},{9,0}}}", 29)
```

Test-cases: Please hard-code inputs and present output to STDOUT. Ignore the failing dummy test-case.

Solution:
http://techieme.in/tree-diameter/
OR http://www.geeksforgeeks.org/diameter-of-a-binary-tree/

(Suggested interview time, 45 minutes)

Double Power

☆ Double power

Implement a power function to raise a double to an int power, including negative powers.

e.g. pow(double d, int p) should give 'd' raised to power 'p'.

Of course, please don't use in-built methods like pow(). Idea is to implement that using recursion.

Solution: http://stackoverflow.com/questions/101439/the-most-efficient-way-to-implement-an-integer-based-power-function-powint-int

Suggested time: 10 minutes to do a brute-force and 15 with a trick that optimizes it.

Note: In HackerRank, Double is simulated with "float".

Ideas

- Check %2 to halve amount of computation

CountTrees

☆ How many trees with 'n' nodes?

Write a function that will return the number of binary trees that can be constructed, with 'n' nodes.

e.g.
for, n=1 ==> 1 possible tree (just root)
for, n=2 ==> 2 possible trees (1. root->right 2. root->left)
for, n=3 ==> 5 possible trees (1. root->right->right 2. root->right->left 3. root->left->left 4. root->left->right 5. root->left,right)

If you keep doing this, it will form a series called Catalan numbers. One can simply lookup the formula for Catalan numbers and write code for it. But that's not how we want to do this. We want to do this by understanding the underlying recursion. The recursion is based on tree-topology only, as you can see by examples above. Contents of the nodes of the tree do not matter.

Solution:
http://techieme.in/count-binary-search-trees/ OR
http://cslibrary.stanford.edu/110/BinaryTrees.html (Search for "Counttrees")

(Suggested time: 20 minutes)

Ideas

- Let f(n) be the # of binary trees for n nodes
- Therefore, f(0) = 1, f(1) = 1, f(2) = 2
- Number of binary trees is the sum of possible (left subtrees of size i, and right subtress of size (n-1-i)
- Basically a summation of n for j = 0l where (n-j-1) * j

HW2 Test

Sum of integers

☆ Sum of integers

Given Inputs:
1. An array of integers
2. Target K

Required Output:
Boolean true or false, whether there is a group of integers (may or may not be contiguous) in the input, that sums to K.

e.g.
Sum({2, 4, 8}, 6) → true
Sum({2, -4, 8}, 1) → false
Sum({2, 4, 8}, 14) → true
Sum({2, 4, 8}, 9) → false

Find all cominbations of well formed brackets

☆ Find all combinations of well-formed brackets

The task is to write a function Brackets(int n) that prints all combinations of **well-formed** brackets from 1...n. 'n' is the number of pairs.

e.g. For Brackets(3) the output would be

```
()
(())  ()()
((()))  (()())  (())()  ()(())  ()()()
```

Run your 'n' from 1 to 10, and output (to STDOUT) the bracket combinations for each value of 'n'.

Test cases: Not set for this problem, because depending on your implementation, you could be printing brackets in any order.

Recursive reverse of a string

☆ Recursive Reverse of a string - complete the code

```
Following code is supposed to reverse a string, recursively. Please read and select possible candidates for the final line.

string reverseString(string &s)
{
    if( s.length() == 0 ) // base case
        return "";

    string last(1,s[s.length()-1]);
    string reversed = reverseString(s.substr(0,s.length()-1));
    <<WHAT IS THE LAST LINE?>>;
}
```

PICK THE CORRECT CHOICES

- [ ] return last;
- [ ] return reversed;
- [ ] return last+reversed;
- [ ] return last + substr(0, reversed.length()-2);
- [ ] return substr(0, last.length()-2) + reversed;
- [ ] return reversed + last;

Clear selection

Object Modeling

```
Statement: Design an elevator system for a building having multiple elevators.

Details:
1. Need to queue elevator call requests
2. Need to queue floor requests within the elevator
        2.1 Elevator needs to keep the floor requests in ascending order if goin
        2.2 Elevator needs to keep the floor requests in descending order if goi
3. Strategy for assigning an elevator for a given call request
4. Elevators can be in maintenance or ou-of-order states
5. Elevator doors can ONLY be open if the elevator is standing and it is on a
particular
floor
6. Elevator must be able to take call requests while moving and must be able to
at the floors even if the requests came late but elevator didn't pass the floor
```

HW3

Is BST

☆ Is it a BST?

This is a very well known interview problem: Given a Binary Tree, check if it is a Binary Search Tree (BST). A valid BST **doesn't** have to be complete or balanced. Duplicate elements are not allowed in a BST.

Solution: O(N) time: http://articles.leetcode.com/2010/09/determine-if-binary-tree-is-binary.html

(Suggested Time: 30 minutes)

Merge trees

☆ Merge trees

Merge two BSTs in O(N1 + N2) time, where N1 and N2 are number of nodes in the two trees respectively. The merged tree should contain all the elements of both trees and also be a balanced BST. Finally, print the new tree level by level.

e.g.
Tree-1: 2->1,3
Tree-2: 7->6,8
Output:

6
2 7
1 3 8

The output above is a tree that's printed level by level.

(This is a very good question. It's not hard at all, but will need you to write several functions: one to parse, one to sort, one to merge, one to reconstruct and one to print. Each of these can be separate short interview questions)

Solution: http://stackoverflow.com/questions/7540546/merging-2-binary-search-trees

(This doesn't have the code to print the tree. But that's quite easy. Just do BFS with a Queue and insert a sentinel at the beginning of each level)

(Time: 45 minutes)

Post Order Traversal without Recursion

☆ PostOrder Traversal without recursion

Write a function to traverse a Binary tree PostOrder, without using recursion. As you traverse, please print contents of the nodes. (Bonus: Spend 1 minute more and also do it with recursion)

Solution: https://www.youtube.com/watch?v=hv-mJUs5mvU

(Time: 30 minutes)

Print all paths in a tree

☆ Print all paths in a tree

(Facebook question)

Given a binary tree, print out all of its root-to-leaf paths one per line.

e.g. for a tree that's

```
    1
  2   3
4 5  6 7
```

Print:

```
1 2 4
1 2 5
1 3 6
1 3 7
```

(This is a simple problem, but really tests your understanding of recursion)

Solution: http://www.geeksforgeeks.org/given-a-binary-tree-print-out-all-of-its-root-to-leaf-paths-one-per-line/

(Time: 20 minutes)

Rebuild the tree

☆ Rebuild the tree

Given the in-order and pre-order traversing results of a BST (as arrays), write a function to rebuild the tree. The function should return the pointer to the root node of the tree. Then take that pointer, and print your tree level by level.

Solutions:
http://articles.leetcode.com/2011/04/construct-binary-tree-from-inorder-and-preorder-postorder-traversal.html
http://edwardliwashu.blogspot.com/2013/01/construct-binary-tree-from-preorder-and.html
https://www.youtube.com/watch?v=PAYG5WEC1Gs

Time: 30 minutes.

LCA

☆ Least Common Ancestor (LCA)

(Another popular interview problem)

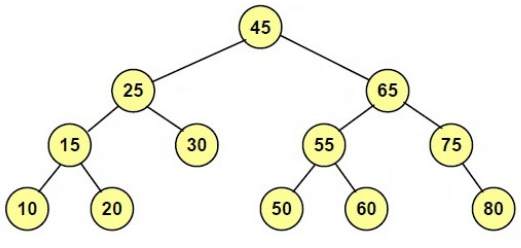
Let T be a rooted tree. The lowest common ancestor between two nodes n1 and n2 is defined as the lowest node in T that has both n1 and n2 as descendants.

The LCA of n1 and n2 in T is the shared ancestor of n1 and n2 that is located farthest from the root. Computation of lowest common ancestors may be useful, for instance, as part of a procedure for determining the distance between pairs of nodes in a tree: the distance from n1 to n2 can be computed as the distance from the root to n1, plus the distance from the root to n2, minus twice the distance from the root to their lowest common ancestor. (Source: Wikipedia)

Design an write an algorithm to find the LCA node, given two nodes in a Binary Tree.
* The tree may or may not be a BST
* Assume a Node structure that has NO parent pointer
* Assume that the two nodes are distinct and exist in the tree
* Find a solution that has runtime complexity of O(N). N is # nodes in the tree.

Test cases:

Given the following tree:



FindLCA(10,20) = 15
FindLCA(50,80) = 65
FindLCA(20,60) = 45

Desired solution: O(N) time.

Solution: http://www.geeksforgeeks.org/lowest-common-ancestor-binary-tree-set-1/

(Time: 30 minutes)

Tree Iterator

☆ Tree Iterator!

Implement an iterator over a binary search tree (BST). Your iterator will be initialized with the root node of a BST.

1. Calling next() will return the next smallest number in the BST.
2. Calling hasNext() should return whether the next element exists.

Both functions should run in average O(1) time and uses O(h) memory, where h is the height of the tree.

[Iterator is a concept in higher level languages like C++ or Java. You probably can tell what it is. If you want to know more, please feel free to Google for the concept.]

Solutions:
1. With parent pointer: http://stackoverflow.com/questions/12850889/in-order-iterator-for-binary-tree
2. Without parent pointer, but with stack: https://leetcode.com/discuss/20001/my-solutions-in-3-languages-with-stack

Choice of the solution will depend on what the interviewer asks you to do. #2 is generally preferred i.e. without assuming there is a parent pointer.

Interview Time: 30 minutes.

BST to double ll

☆ Convert a BST into a Circular Doubly Linked List

(Google question)

Write a recursive function treeToList(Node root) that takes a BST and rearranges the internal pointers to make a circular doubly linked list out of the tree nodes. The "previous" pointers should be stored in the "Left" field and the "next" pointers should be stored in the "Right" field. The list should be arranged so that the nodes are in increasing order. Return the head pointer to the new list. The operation can be done in O(n) time.

Suggested time: 45 minutes.

Solution: http://articles.leetcode.com/2010/11/convert-binary-search-tree-bst-to.html

Clone a binary tree

☆ Clone a binary tree

Given a binary tree (represented by its root node, like usual), clone it. Return the root node of the cloned tree.

Solution: http://crackprogramming.blogspot.com/2012/11/make-copy-of-binary-tree-given-pointer.html

Remember: Cloning or copying a tree is best done recursively. Notice how clean and succinct the code is. Some of you may be tempted to do it breadth-first. But that's more complicated to handle in implementation.

Populate sibling pointers

☆ Populate Sibling pointers

Given a full binary tree, populate the nextRight pointers in each node.

(Full Binary Tree = every node other than the leaves has two children)

Solution: http://articles.leetcode.com/2010/03/first-on-site-technical-interview.html

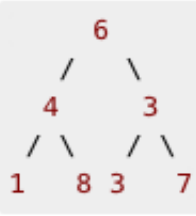
Flip a tree

☆ Flip a tree

Reverse a general binary tree, i.e. flip it from right to left.

So for example if we had the binary tree

Reversing it would create



Solution: http://stackoverflow.com/questions/9460255/reverse-a-binary-tree-left-to-right

Funny/sad story: https://twitter.com/mxcl/status/608682016205344768

Largest binary tree

☆ Largest BST

Given a binary tree, find the largest Binary Search Tree (BST), where largest means BST with largest number of nodes in it. The largest BST may or may not include all of its descendants.

(There is another problem, about finding largest Binary Search *Subtree* in a given tree. This one is different from that. When you read the solution, you'll be able to see a link to that problem too)

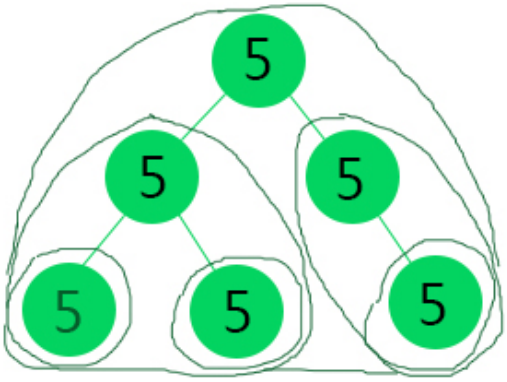
Solution: http://articles.leetcode.com/2010/11/largest-binary-search-tree-bst-in_22.html

Single value tree

☆ Single Value Tree

Given a binary tree, we need to count the number of unival subtrees (all nodes that have the same value).

e.g. Given the following tree, it has 6 unival subtrees.



Solution: https://crazycoderzz.wordpress.com/count-the-number-of-unival-subtrees-in-a-binary-tree/

Written Question

☆ Read and comment

Read solution to problem 15.9 from Elements of Programming Interviews:

The most visited pages: https://books.google.com/books?id=y6FLBQAAQBAJ&lpg=PA254&dq=for+each+page,+count+the+number+of+times+it+has+been+visited&source=bl&ots=AsLplJucvh&sig=2dUr70_eFmOKZNrd0MmEMuB9Z5Q&hl=en&sa=X&output=reader&pg=GBS.PA254

[PDF here]

Read and understand the problem, and write notes for your own self below.

I have met the author and I love that book. Many companies have now started to pick problems from that book. This one in particular is very instructive.

HW3 Test

☆ Balanced BST from a sorted array

Problem statement: Create and print a balanced BST from a sorted array

* Output tree should be a valid BST and balanced. Balanced tree can have the difference of at most ONE between left and right.
* Input array is sorted!
* Assume distinct integers in the input array.
* Create a standard Tree node structure, without the parent pointer.
* Print the resultant tree, level by level.

For example:

1:
Input: 1,2,3

Output:
2
1 3

2.
Input:
8,10,12,15,16,20,15

Output:
15
10 20
8 12 16 25

☆ Top K, in a BST

Given a BST with integer values, and a positive (or zero) integer K, print top (smallest) K elements from that BST.

* BST may or may not be balanced.
* We cannot alter the structure or data inside the given BST.

Inputs
1. A BST. You can hard-code your input, or you can use your code from a different problem.
2. A positive (or zero) integer K.

Output
K least elements.

☆ Height of a K-ary tree

Given a K-ary tree, find it's height (can also be called 'Depth', or the length of the longest path).

Input: A K-ary tree. You can hard-code your input. Construction is not a part of the test.
Output: Positive integer, that is Height.

(1<=K<=INT_MAX)

☆ Delete!

What is the average complexity of a delete operation on a BST? (N = number of elements in the tree)

PICK ONE OF THE CHOICES

- O(N)
- O(Log(N))
- O(1)
- O(NLog(N))

Clear selection

☆ BFS Queue

What is the maximum length of the queue in BFS equal to? (Or what is the longest a BFS queue can be?)

PICK ONE OF THE CHOICES

- ○ Equal to height of the tree
- ○ Equal to number of elements in the tree
- ○ Equal to minimum elements at a level
- ○ Equal to maximum elements at a level

Clear selection

☆ Recursion, again

For a tree depth-first traversal problem, if the interviewer says "Don't use recursion" or "do it iteratively", what data-structure do they most likely mean you use?

PICK ONE OF THE CHOICES

- ○ Queue
- ○ B-Tree
- ○ Stack
- ○ Heap

Clear selection

☆ True or False?

Pre order traversal of a BST produces elements in sorted order

PICK ONE OF THE CHOICES

- ○ True
- ○ False

Clear selection

HW 4

☆ Longest Common SubSequence

Given two sequences, find the longest subsequence present in both of them. (Not just the length, but the actual string)

The longest common subsequence (LCS) problem is the problem of finding the longest subsequence common to all sequences in a set of sequences (often just two sequences). It differs from problems of finding common substrings: unlike substrings, subsequences are not required to occupy consecutive positions within the original sequences. The longest common subsequence problem is a classic computer science problem, the basis of data comparison programs such as the diff utility, and has applications in bioinformatics. It is also widely used by revision control systems such as Git for reconciling multiple changes made to a revision-controlled collection of files. (Source Wikipedia).

For example, "abc", "abg", "bdf", "aeg", '"acefg", .. etc are subsequences of "abcdefg". So a string of length 'n' has 2^n different possible subsequences.

LCS for input Sequences "ABCDGH" and "AEDFHR" is "ADH" of length 3.
LCS for input Sequences "AGGTAB" and "GXTXAYB" is "GTAB" of length 4.

Return empty string if there is no such common subsequence.

Solution: http://www.geeksforgeeks.org/printing-longest-common-subsequence/

☆ Word Break

Given an input string and a dictionary of words, segment the input string into a space-separated sequence of dictionary words if possible.

For example, if the input string is "applepie" and dictionary contains a standard set of English words, then we would return the string "apple pie" as output.

If you can find multiple arrangements, then print all such combinations.

Solution: http://www.programcreek.com/2014/03/leetcode-word-break-ii-java/

☆ Coin play

Consider a row of n coins of values v1 . . . vn, where n is even. We play a game against an opponent by alternating turns. In each turn, a player selects either the first or last coin from the row, removes it from the row permanently, and receives the value of the coin. Determine the maximum possible amount of money we can definitely win if we move first.

Assume full competency by both players.

Example:
Coins given: 8, 15, 3, 7.

Player 1 can start two different ways: either picking 8 or picking 7. Depending on the choice s/he makes, the end reward will be different. We want to find the maximum reward the first player can collect.

1.
.....Player-1 chooses 8.
.....Opponent chooses 15.
.....Player-1 chooses 7.
.....Opponent chooses 3.
Total value collected by Player-1 is 15(8 + 7) (Greedy strategy i.e. pick the highest at every step)

2.
.....Player-1 chooses 7.
.....Opponent chooses 8.
.....Player-2 chooses 15.
.....Opponent chooses 3.
Total value collected by Player-1 is 22(7 + 15)

Given these two strategies, we want 22 as the answer, and not 15.

Solution: http://www.geeksforgeeks.org/dynamic-programming-set-31-optimal-strategy-for-a-game/

☆ Maximum size sub-matrix

Given a binary matrix, find out the maximum size *square* sub-matrix with all 1s.

Given this matrix:

```
0  1  1  0  1
1  1  0  1  0
0  1  1  1  0
1  1  1  1  0
1  1  1  1  1
0  0  0  0  0
```

The maximum square sub-matrix with all set bits is

```
1  1  1
1  1  1
1  1  1
```

Please hard-code your test-cases. That will give you more flexibility in input and output. For the output for example, you can either print the matrix, or simply return the top left and bottom corner index values - your choice.

(It's debatable whether the solution to this problem can strictly be categorized as a DP solution. Nevertheless, this is an important problem to solve and it's challenging!)

Solution: http://www.geeksforgeeks.org/maximum-size-sub-matrix-with-all-1s-in-a-binary-matrix/

☆ Knight's tour!

Given a phone keypad as shown below:

```
1 2 3
4 5 6
7 8 9
  0
```

How many different 10-digit numbers can be formed starting from 1? The constraint is that the movement from 1 digit to the next is similar to the movement of the Knight in a chess game.

For eg. if we are at 1 then the next digit can be either 6 or 8 if we are at 6 then the next digit can be 1, 7 or 0.

Repetition of digits are allowed - 1616161616 is a valid number. The problem requires us to just give the count of 10-digit numbers and not necessarily list the numbers.

Find a polynomial time solution, based on Dynamic Programming.

Solution: http://stackoverflow.com/questions/2893470/generate-10-digit-number-using-a-phone-keypad

☆ Matrix Chain Multiplication

(Another classical DP problem, which interviewers love)

Given a sequence of matrices, find the most efficient way to multiply these matrices together. The problem is not actually to perform the multiplications, but merely to decide in which order to perform the multiplications.

We have many options to multiply a chain of matrices because matrix multiplication is associative. In other words, no matter how we parenthesize the product, the result will be the same. For example, if we had four matrices A, B, C, and D, we would have:

```
(ABC)D = (AB)(CD) = A(BCD) = ....
```

However, the order in which we parenthesize the product affects the number of simple arithmetic operations needed to compute the product, or the efficiency. For example, suppose A is a 10 × 30 matrix, B is a 30 × 5 matrix, and C is a 5 × 60 matrix. Then,

```
(AB)C = (10×30×5) + (10×5×60) = 1500 + 3000 = 4500 operations
A(BC) = (30×5×60) + (10×30×60) = 9000 + 18000 = 27000 operations.
```

Clearly the first parenthesization requires less number of operations.

Given an array p[] which represents the chain of matrices such that the ith matrix Ai is of dimension p[i-1] x p[i]. We need to write a function MatrixChainOrder() that should return the minimum number of multiplications needed to multiply the chain.

```
Input: p[] = {40, 20, 30, 10, 30}
Output: 26000
There are 4 matrices of dimensions 40x20, 20x30, 30x10 and 10x30.
Let the input 4 matrices be A, B, C and D.  The minimum number of
multiplications are obtained by putting parenthesis in following way
(A(BC))D --> 20*30*10 + 40*20*10 + 40*10*30

Input: p[] = {10, 20, 30, 40, 30}
Output: 30000
There are 4 matrices of dimensions 10x20, 20x30, 30x40 and 40x30.
Let the input 4 matrices be A, B, C and D.  The minimum number of
multiplications are obtained by putting parenthesis in following way
((AB)C)D --> 10*20*30 + 10*30*40 + 10*40*30

Input: p[] = {10, 20, 30}
Output: 6000
There are only two matrices of dimensions 10x20 and 20x30. So there
is only one way to multiply the matrices, cost of which is 10*20*30
```

Source and Solution: http://www.geeksforgeeks.org/dynamic-programming-set-8-matrix-chain-

☆ Word Wrap

Given a sequence of words, and a limit on the number of characters that can be put in one line (line width). Put line breaks in the given sequence such that the lines are printed neatly. Assume that the length of each word is smaller than the line width.

The word processors like MS Word do task of placing line breaks. The idea is to have balanced lines. In other words, not have few lines with lots of extra spaces and some lines with small amount of extra spaces.

```
The extra spaces includes spaces put at the end of every line exce
pt the last one.
The problem is to minimize the following total cost.
 Cost of a line = (Number of extra spaces in the line)^3
 Total Cost = Sum of costs for all lines

For example, consider the following string and line width M = 15
 "Geeks for Geeks presents word wrap problem"

Following is the optimized arrangement of words in 3 lines
Geeks for Geeks
presents word
wrap problem

The total extra spaces in line 1, line 2 and line 3 are 0, 2 and 3
 respectively.
So optimal value of total cost is 0 + 2*2 + 3*3 = 13
```

Please note that the total cost function is not sum of extra spaces, but sum of cubes (or square is also used) of extra spaces. The idea behind this cost function is to balance the spaces among lines. For example, consider the following two arrangement of same set of words:

1) There are 3 lines. One line has 3 extra spaces and all other lines have 0 extra spaces. Total extra spaces = 3 + 0 + 0 = 3. Total cost = 3*3*3 + 0*0*0 + 0*0*0 = 27.

2) There are 3 lines. Each of the 3 lines has one extra space. Total extra spaces = 1 + 1 + 1 = 3. Total cost = 1*1*1 + 1*1*1 + 1*1*1 = 3.
Total extra spaces are 3 in both scenarios, but second arrangement should be preferred because extra spaces are balanced in all three lines. The cost function with cubic sum serves the purpose because the value of total cost in second scenario is less.

Solution: http://www.geeksforgeeks.org/dynamic-programming-set-18-word-wrap/

(Slightly different variation: https://leetcode.com/problems/text-justification/)

☆ Robbery

There are n houses built in a line, each of which contains some value in it. A thief is going to steal the maximal value in these houses, but he cannot steal in two adjacent houses because the owner of a stolen house will tell his two neighbors on the left and right side. What is the maximal stolen value?

For example, if there are four houses with values {6, 1, 2, 7}, the maximal stolen value is 13 when the first and fourth houses are stolen.

Solution: http://codercareer.blogspot.com/2013/02/no-44-maximal-stolen-values.html

☆ Balanced Partition

(This is an extension and generalization of SumZero problem)

Partition an array into two partitions, such that sum of two halves is the same.
* Assume that the sum of all elements in the array is even.
* Numbers can be positive or negative, viz non-zero integers
* Numbers can repeat.
* If there are multiple such partitions possible, then print any one.
* When you print, try to preserve the order of elements when you print them.

e.g.
Input: 4, 1, -5, 6, -11, 3
Output: 4, 6, -11 and 1, -5, 3 (Both sum to -1)

Solution: http://www.geeksforgeeks.org/dynamic-programming-set-18-partition-problem/

☆ Strings interleave

You're given three strings A, B and I. Write a function that checks whether I is an interleaving of A and B. String I is said to be interleaving string A and B, if it contains all characters of A and B and order of all characters in individual strings is preserved.

See test-cases below. Format {String I, String A, String B, True/False}:

Solution: http://www.geeksforgeeks.org/check-whether-a-given-string-is-an-interleaving-of-two-other-given-strings-set-2/

```
{"1234", "123", "123", 0},
{"112233", "123", "123", 1},
{"123456", "123456", "", 1},
{"123456", "", "123456", 1},
{"12345678", "1234", "5678", 1},
{"12345678", "1233", "5678", 0}
};
```

Solution:

☆ Count ways to reach the n'th stair

(This is simple. Make me proud)

There are n stairs, a person standing at the bottom wants to reach the top. The person can climb either 1 stair or 2 stairs at a time. Count the number of ways, the person can reach the top.

```
Input: n = 1
Output: 1
There is only one way to climb 1 stair

Input: n = 2
Output: 2
There are two ways: (1, 1) and (2)

Input: n = 4
Output: 5
(1, 1, 1, 1), (1, 1, 2), (2, 1, 1), (1, 2, 1), (2, 2)
```

Source and Solution: http://www.geeksforgeeks.org/count-ways-reach-nth-stair/

☆ Making Change

[We did this in class!]

You are given n types of coin denominations of values v(1) < v(2) < ... < v(n) (all integers). Give an algorithm which makes change for an amount of money C with as few coins as possible.

* Assume there are multiple coins of every denomination.
* Assume v(1) = 1, (i.e. there is always a combination that leads to C).
* There may be multiple ways of reaching C. We want a DP based solution that leads to the method using least number of coins.
* Input: C and Denominations Array
* Output: Combination using minimum number of coins (repeat coins ok) that leads to C. There may be multiple such combinations. Print one on each line, with coins separated by space

e.g.
Input:
Denominations: 1,2,3
C: 4
Output on two lines:
1,3
2,2

☆ Levenshtein Distance (also called Edit Distance)

Given two words word1 and word2, find the minimum number of steps required to convert word1 to word2. (each operation is counted as 1 step.)

You have the following 3 operations permitted on a word:

a) Insert a character

b) Delete a character

c) Replace a character

e.g. Minimum edit distance between the words 'kitten' and 'sitting', is 3

kitten → sitten (substitution of "s" for "k")
sitten → sittin (substitution of "i" for "e")
sittin → sitting (insertion of "g" at the end).

(Assume all inputs and substitutions in lower case)

More about Levenshtein Distance: https://en.wikipedia.org/wiki/Levenshtein_distance

Solution: http://www.geeksforgeeks.org/dynamic-programming-set-5-edit-distance/

HW 4 Test

☆ Cut the rope

Given a rope with length 'n', how should we cut the rope into m parts, with lengths n[0], n[1], ...,n[m-1], in order to get the maximal product of n[0]*n[1]* ... *n[m-1]?

* We have to cut once at least.
* Length of the whole length of the rope, as well as the length of each part, are in (positive) integer value. i.e. can't do partial cuts.
* Assume that the length is at least 2 units

e.g.

For n = 4, there are two cuts possible: 1 + 3 and 2 + 2. We'll pick 2 + 2, because their product (2*2 = 4) is greater than product of the first one (1*3 = 3).
For n = 10, the output is 36, for (3 + 3 + 4).

☆ Number of paths in a matrix

You are given a *m x n* matrix *a*. From any cell *(i, j)*, you can move either to the right or downwards, i.e., either to cell *(i, j + 1)* or *(i + 1, j)*. You are initially at cell *(0, 0)* and want to reach at cell *(m - 1, n - 1)*. Note that the each cell has two values, either *0* or *1*, if the value at any cell is 1, then you can move through that cell, otherwise you cannot move.

Complete the function *numberOfPaths*, that has one parameter- *m x n* matrix *a*. The function should return total number paths possible from cell *(0, 0)* to *(m - 1, n - 1)*, as the answer could be large, return its value mod $(10^9 + 7)$.

Input Format
The first line of the input is an integer *m*, number of rows in matrix. The second line of the input is an integer *n*, number of columns in matrix. Each of the next m lines contains n space separated integers.

Constraints
$1 \le n, m \le 1000$
The value of each cell of matrix a, is either *0* or *1*.

Output Format
The function should return total number of paths possible from cell (0, 0) to (m - 1, n - 1).

Sample Input 1

```
3
4
1 1 1 1
1 1 1 1
1 1 1 1
```

Sample Output 1

```
10
```

Sample Input 2

```
2
2
1 1
0 1
```

Sample Output 2

```
1
```

Explanation
Sample Case 1
Counting all possible paths from cell (0, 0) to (2, 3) gives us 10 possible paths as follows:
One path that goes all the way across the top row to *(0, 3)*.
Two paths that go right to cell *(0, 2)* then turn down.
Three paths that go right to to cell *(0, 1)* then turn down.
Four paths that start by going down to cell *(1, 0)*.

Sample Case 2
You can move only right from the top-left cell: *(0,0)* to ***(0,1)***. After that, your only move is to *(1,1)*. Hence only 1 path.

HW 5

☆ Nuts and Bolts!

A disorganized carpenter has a mixed pile of bolts and nuts and would like to find the corresponding pairs of bolts and nuts. Each nut matches exactly one bolt (and vice versa). By trying to match a bolt and a nut the carpenter can see which one is bigger, but she cannot compare two bolts or two nuts directly. Can you help the carpenter match the nuts and bolts quickly?

In other words: You are given a collection of nuts of different size and corresponding bolts. You can choose any nut & any bolt together, from which you can determine whether the nut is larger than bolt, smaller than bolt or matches the bolt exactly. However there is no way to compare two nuts together or two bolts together. (i.e. we cannot sort all nuts or sort all bolts). Write an algorithm to match each bolt to its matching nut.

You can make the following assumptions:
1. There are equal number of nuts and bolts
2. A given nut uniquely matches a bolt. i.e. There are no extra unmatched nuts or extra bolts. i.e. every nut has one and only one matching bolt and vice-versa.

e.g.
Input:
N3,N2,N1,N4
B4,B2,B3,B1

Output (in any order):
N1B1
N2B2
N3B3
N4B4

Test cases: Please hard-code test-cases in your main() method. That will be more convenient to you in this problem. Ignore the existing dummy test-case.

(Goal: Understand the application of QuickSort. Apply the concept of a Pivot and partitioning)

Solution complexity: Is the time/space complexity same as quicksort, or worse, or better?

Interview time: 30 minutes.

Solution: http://www.geeksforgeeks.org/nuts-bolts-problem-lock-key-problem/

☆ Sort all characters in a string

Sort an array of characters (ASCII only, not UTF8).

Input: A string of characters, like a full English sentence, delimited by a newline or NULL. Duplicates are okay.
Output: A string of characters, in sorted order of their ASCII values. You can overwrite the existing array.

Solution Complexity: Aim for linear time and constant additional space.

(What to understand from this problem: ASCII is great, because it's limited to 256. Remember that for any input that is bound to a range. Also known as bucket-sort)

Interview time: 15 minutes.

Solution: This is a trivial problem :-)

☆ Merge K sorted arrays, size N each

This is a popular facebook problem: Given K sorted arrays of size N each, merge them and print the sorted output. Assume N is very large compared to K. N may not even be known. i.e. the arrays could be just sorted streams, e.g. timestamp streams.

For example:

Input: K = 3, N = 4
arr[][] = { {1, 3, 5, 7},
{2, 4, 6, 8},
{0, 9, 10, 11}} ;
Output: 0 1 2 3 4 5 6 7 8 9 10 11

Test-cases: It'll be easier in this problem if you hard-code your own test cases inside the main() method. Print output to STDOUT. Ignore the given dummy test-case.

(Hint: Realize that you don't need to access all N*K elements in order to merge. Merging can start with fewer elements. When you have fewer elements to sort i.e. the input is already mostly sorted, then a specific sorting method that you learnt in the class, is going to work the best. It's the Priority Queue)

Solution runtime: Optimal known solution is NKLog(K).
Interview time: 45 minutes.

Solution: https://leetcode.com/discuss/9279/a-java-solution-based-on-priority-queue

Extra credit: Implement the Priority Queue instead of using existing library functions.

☆ Group the numbers

(If this question feels easy, that's because it is)

Given an array of numbers, positive integers only, group them in-place into evens and odds.

Input: Integer array, positive integers only, repeats possible.
Output: The same integer array, with evens on left side and odds on the right side. There is no need to preserve order within odds or within evens.

(Understand that Grouping is just a special case of sorting. It's cognitively much less complex, and can be done with easier methods, which we may not label as sorting, but are just a special case of sorting)

Solution Complexity: Aim for O(N), in-place. Ideally you'd do it in one pass of the array.

Test-cases: As long as you can ascertain that your solution is right, don't worry if the test-cases don't pass exactly as they are given here. They expect evens to the left and odds to the right. Plus you may have different order.

Interview time: 15 minutes.

Solution: This is a trivial problem :-)

☆ Top K

[Popular Facebook problem and a Computer Science classic, which you may have touched in the class also]

Problem statement: Find K largest elements from a given stream of numbers. By definition, we don't know the size of the input stream. Hence produce K largest elements seen so far, at any given time.

* Input may or may not be sorted and could have duplicates
* Represent input stream as an array. Don't rely on its size.
* Feel free to use built-in functions if you need a specific data-structure.
* If your output is correct, but a test-case is failing because order of output elements is different, then don't worry about it. Move on.

=====

Sorting Homework

Interview time: 45 minutes

Solution(s): http://www.geeksforgeeks.org/k-largestor-smallest-elements-in-an-array/ (Solution #6 is the only one that'll work when size of the input stream is not known)

[Trivia: When size of input array (stream) is known to be N, then either #5 or #6 will work. Both solutions have same complexity, but #6 will work faster because it doesn't muck with the entire input, and only deals with K elements separately]

☆ Duplicate in a loose permutation

Find a duplicated number in a loose permutation of numbers. A permutation is an array that is size N, and also has positive numbers from 1 thru N. A loose permutation is a permutation where some numbers are missing and some are duplicated, but the total number is still N.

* We want to find *any one* duplicated number; not necessarily the first or the least.
* It can occur anywhere in the input array, and we don't care how many times it's duplicated.
* Input array may nor may not be sorted.
* You can only use constant extra memory.
* There is no limit/constraint on N i.e it is a normal 4-byte integer.

e.g.
Input: 1,7,4,3,2,7,4: This array has 7 numbers from 1 thru 7, with some missing (5 and 6) and some duplicated (4 and 7). Albeit unsorted, but sorting is irrelevant to a permutation.
Output: 4 or 7

Input: 3,1,2: This array has nothing missing.
Output: -1

Interview time: 30 minutes (This is a surprisingly aha! problem)

Solution: Aim for constant memory and linear run time. Think bucket sort.

About Test-cases: Test-cases only display one duplicated number. If that's not what your answer is, but your answer is still a duplicate number, then that's acceptable.

Solution: https://soham.box.com/s/wte4tqbjbfocnhy76teobzki7sxi5lq8

☆ Nearest Neighbor

Given a point P, and other N points in two dimensional space, find K points out of the N points which are nearest to P.

* Distance between two points is measured by standard euclidean method.

(Hint: This problem can either be done with QuickSort partitioning, or can be done with Heaps. Which one would you use? Why? Why not try both in your IDE and see the runtime for large inputs?)

Interview time: 45 minutes

Solution: http://stackoverflow.com/questions/20398799/find-k-nearest-points-to-point-p-in-2-dimensional-plane

☆ Implement Merge Sort

Please convert the standard merge-sort algorithm into code.

Input: Integers in an array, duplicates possible
Output: Same integers in ascending order, in a new array. Preserve the input array.

(Goal of this homework problem: The concept of partitioning, concept of merging partitions, the clarity that merge-sort needs extra space and why worst-case is better than QuickSort viz. O(NLogN))

Suggested time: 30 minutes

Solution: http://www.codenlearn.com/2011/10/simple-merge-sort.html
Trivia: In the solution above, what is the Complexity of memory requirement? Can we do better than that?

☆ 3Sum problem

(Here is an example of a problem, where sorting is inevitable, but it doesn't matter which sorting technique you use)

Given an array of N integers, find all triplets that sum to a given integer 0 (zero).

* Triplets may or may not be consecutive numbers.
* The array can include duplicate elements.
* Array is not necessarily sorted.

* Print output as shown. i.e. as strings, one per line, comma separated elements.
* Order of elements inside the answer triplets does not matter. i.e. if your output elements are the same, but only in different order, then it's an acceptable solution.
* Do not print duplicate triplets.
* If no such triplets are found, then print nothing.

Interview time: 30 minutes

Solution complexity: Cant' do better than N^2. This is one of those nasty problems, where even N^2 is not intuitive.
Solution: http://www.programcreek.com/2012/12/leetcode-3sum/

Trivia: There are some similar problems to this e.g. http://www.geeksforgeeks.org/find-number-of-triangles-possible/

☆ Find an integer not among four billion given ones

(This is a popular interview problem, from Programming Pearls)

Given an input file with four billion integers, provide an algorithm to generate an integer which is not contained in the file. Assume you have 1 GiB memory. Follow up with what you would do if you have only 10 MiB of memory.

It's difficult to test the code for this problem in HackerRank, because of memory constraints. But write your solution as if the constraints are present.

Interview time: 30 minutes

Solution: http://stackoverflow.com/questions/7153659/find-an-integer-not-among-four-billion-given-ones?rq=1

HW 5 Test

☆ Sort the integers

You are given approximately 10 million pairs. Each pair consists of a 10-bit integer and an object. Implement an efficient sorting algorithm to sort the objects.

Sort order of the objects is determined by its paired integer.
* Each integer corresponds to the object at the same index.
* If two integers are same, the order of objects doesn't matter.

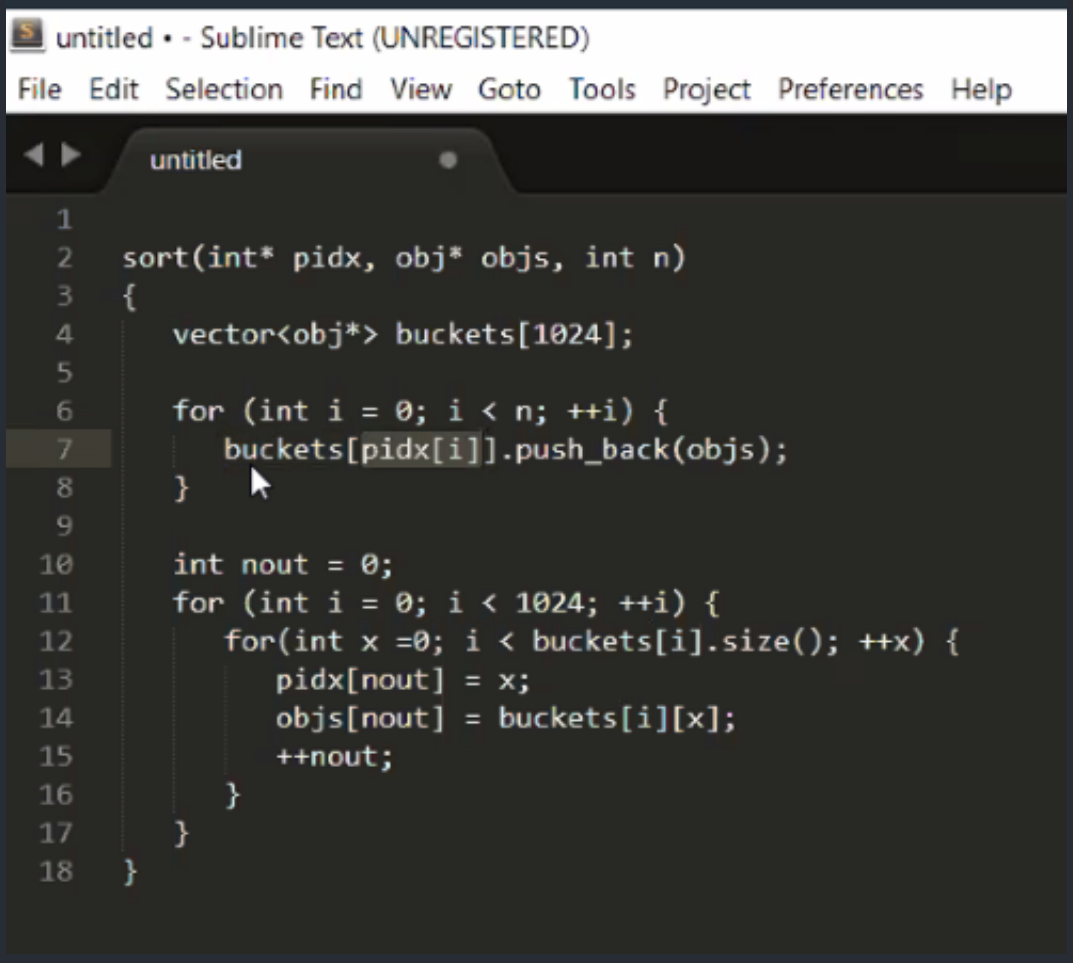
Test cases: HackerRank doesn't provide a way for us to create test-cases involving generic objects. Please hence hard-code your own test-cases.

Inputs:
1. First array, of numbers, size N
2. Second array, of Objects, size N (assume 1-1 correlation)

Output:
One array of sorted objects

Expected interview time: 20-25 minutes.

ANS



☆ Merge first sorted array into another one

You're given two sorted arrays:

1. Array1, size 'N', which has N sorted positive integers
2. Array2, size '2N' (twice the size of first), which also has only N sorted positive integers in its first half. Second half of this array is empty. [Empty elements are marked by 0].

Write a function that takes these two arrays, and merges the first one into second one, resulting in one fully sorted array of 2N elements.

Constraints: You can use only constant extra space. Need a linear solution. Repeats are allowed. Only positive non-zero integers are found in input. 0 denotes an empty space.

(Expected interview time: 20 minutes)

☆ Partially sorted

In an interview problem, if

- the input is almost sorted
- the input is read-only
- and the goal is to fully sort the input

then which sorting algorithm should come to mind first?

PICK ONE OF THE CHOICES

- ○ **Merge Sort, because merge sort can merge partially sorted arrays**
- ○ **Heap Sort, because then we can create a separate heap of only those elements that are not sorted**
- ○ **Quick Sort, because Quick Sort is fast regardless**

Clear selection

☆ Complexity

The average complexity of Heap Sort (in broad terms) is:

PICK ONE OF THE CHOICES

- ○ **O(NLog(N))**
- ○ **O(N)**
- ○ **O(N) + O(LogN)**
- ○ **O(2N)**

Clear selection

☆ Parallel!

What will you use for large scale parallel sorting?

PICK ONE OF THE CHOICES

- ○ Bucket Sort
- ○ Merge Sort
- ○ Radix Sort

Clear selection

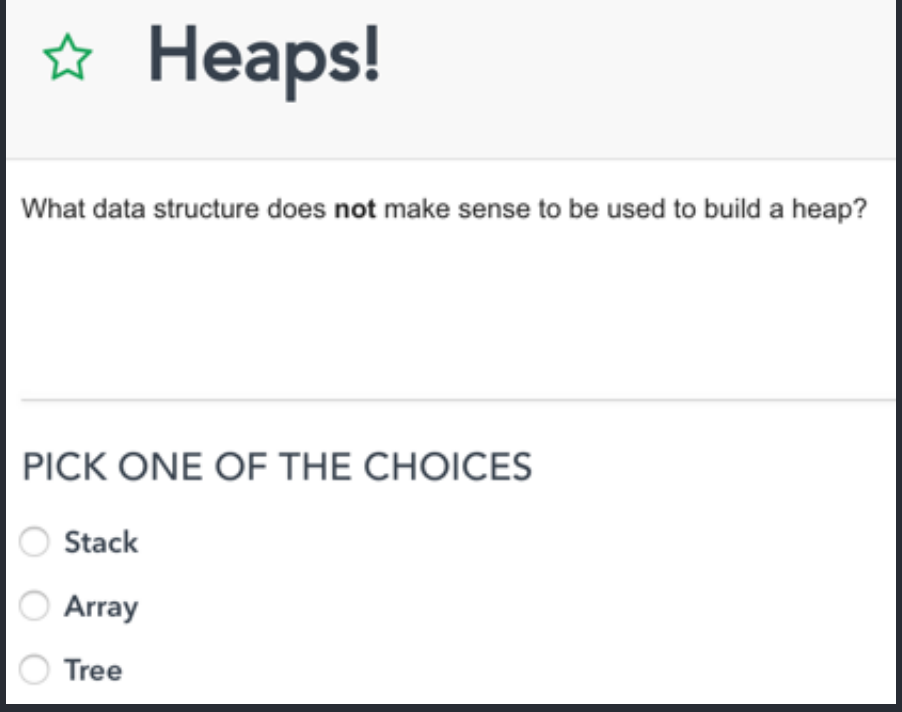
☆ Unstable!

Which of the following sorts is (are) unstable?

PICK THE CORRECT CHOICES

- ☐ Heap
- ☐ Merge
- ☐ Quick

Clear selection


Heaps!
What data structure does not make sense to be used to build a heap?
PICK ONE OF THE CHOICES
Stack
Array
Tree

HW 6

☆ Given a number, find the next palindromic number

(This question is quite rampant. Not sure why)

Given a number, find the next smallest palindromic number, larger than this number.
e.g
Input: 23545
Output: 23632 (This is a palindromic number, and bigger than the input. There is no palindromic number less than this and bigger than the input)

Input: 99
Output: 101

Input: 6789876
Output: 6790976 (Note that the input may or may not be a palindrome itself)

Input: 8998 (Note that input can have even number of digits)
Output: 9009

Solution: http://www.geeksforgeeks.org/given-a-number-find-next-smallest-palindrome-larger-than-this-number/

☆ Array product

[Again, this is also somehow popular]

Given an array of numbers, `nums`, return an array of numbers `products`, where `products[i]` is the product of all `nums[j], j != i`.

```
Input : [1, 2, 3, 4, 5]
Output: [(2*3*4*5), (1*3*4*5), (1*2*4*5), (1*2*3*5), (1*2*3*4)]
      = [120, 60, 40, 30, 24]
```

You must do this in `O(N)` time, and constant space, **without using division**. Usage of `products` array is not considered extra space.

Without using division is the key constraint to remember.

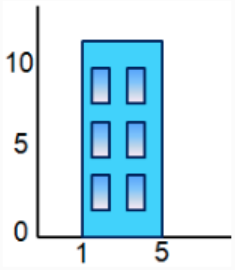
Solution: http://stackoverflow.com/questions/2680548/given-an-array-of-numbers-return-array-of-products-of-all-other-numbers-no-div

☆ Skyline!

This is a very popular problem. Make sure you nail it. Here is the problem definition:

You are given a set of rectangles in no particular order. They have varying widths and heights, but their bottom edges are collinear, so that they look like buildings on a skyline. For each rectangle, you're given the x position of the left edge, the x position of the right edge, and the height. Your task is to draw an outline around the set of rectangles so that you can see what the skyline would look like when silhouetted at night.

Each **building** is represented by triplet (left, ht, right)
'left': is x coordinated of left side (or wall).
'right': is x coordinate of right side
'ht': is height of building.



For example, the building above, is represented as (1, 11, 5).

A **skyline** is a collection of rectangular strips. A rectangular **strip** is represented as a pair (left, ht) where left is x coordinate of left side of strip and ht is height of strip.

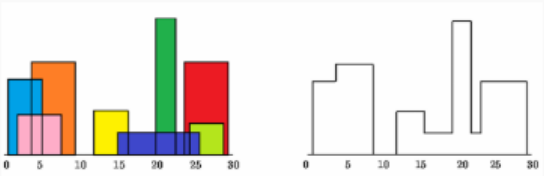
Examples:

```
Input: Array of buildings
       { (1,11,5), (2,6,7), (3,13,9), (12,7,16), (14,3,25),
         (19,18,22), (23,13,29), (24,4,28) }

Output: Skyline (an array of rectangular strips)
        A strip has x coordinate of left side and height
        (1, 11), (3, 13), (9, 0), (12, 7), (16, 3), (19, 18),
        (22, 3), (25, 0)

The below figure demonstrates input and output.

The left side shows buildings and right side shows skyline.
```




```
Consider following as another example when there is only one
building
Input:  {(1, 11, 5)}
Output: (1, 11), (5, 0)
```

Solution:
Good explanation with visuals: https://briangordon.github.io/2014/08/the-skyline-problem.html
Succinct explanation: http://www.geeksforgeeks.org/divide-and-conquer-set-7-the-skyline-problem/
More granular problem definition: https://leetcode.com/problems/the-skyline-problem/

☆ 2D array search

Find a number in a sorted 2D array.

You're given a 2d array (N x M) where all the numbers (integers) in the array are in increasing order from left to right and top to bottom. You're also given a target number, to be searched inside the array. What is the best way to search and determine if a target number is in the array?

Solution: take inspiration from http://www.geeksforgeeks.org/search-in-row-wise-and-column-wise-sorted-matrix/

(Notice that this can be a very deceptive problem in a good way. Solution seems difficult, but it's actually quite simple.)

☆ Merge Overlapping Intervals

Given a set of time intervals in any order, merge all overlapping intervals into one and output the result which should have only mutually exclusive intervals.

e.g. for this input: {{1,3}, {2,4}, {5,7}, {6,8} }. The intervals {1,3} and {2,4} overlap with each other, so they should be merged and become {1, 4}. Similarly {5, 7} and {6, 8} should be merged and become {5, 8}.

Write a function which produces the set of merged intervals for the given set of intervals.

Solution: http://www.geeksforgeeks.org/merging-intervals/

☆ Alternating positive and negative

Given an array containing both +ve and -ve integers, return an array of alternating positive integers and negative integers such that each set of integers are in the same order as in the input array (stable).
e.g.
input {2, 3, -4, -9, -1, -7, 1, -5, -6}
output {2, -4, 3, -9, 1, -1, -7, -5, -6}.

Can you implement it without using any additional space?

Solution:

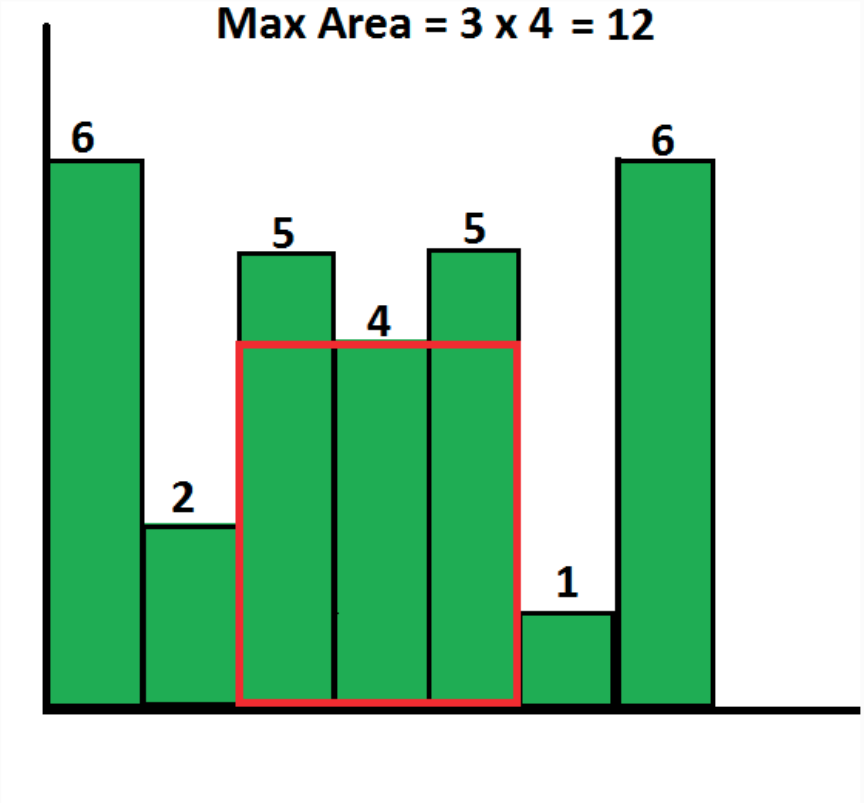
```
>>> A=[2, 3, -4, -9, -1, -7, 1, -5, -6]
>>> for i in range(len(A)):
...     j=0
...     while j+2 < len(A):
...             S=[A[j], A[j+1], A[j+2]]
...             S=[1 if x >=0 else -1 for x in S]
...             if S[0]==S[1] and S[1]!=S[2]:
...                     A[j+1],A[j+2]=A[j+2],A[j+1]
...             j+=1
...
>>> A
[2, -4, 3, -9, 1, -1, -7, -5, -6]
```

[With no additional space, you can't do better than O(N^2)]

☆ Area under histogram

(This is different from the Skyline problem, but some thinking is common)

Find the largest rectangular area possible in a given histogram where the largest rectangle can be made of a number of contiguous bars. For simplicity, assume that all bars have same width and the width is 1 unit.
For example, consider the following histogram with 7 bars of heights {6, 2, 5, 4, 5, 2, 6}. The largest possible rectangle possible is 12 (see the below figure, the max area rectangle is highlighted in red).



Solution: http://www.geeksforgeeks.org/largest-rectangle-under-histogram/

(Ideally, this would be with Linked Lists, Stacks and Queues homework, but that homework already has too many problems :-))

☆ Print Pascal's triangle

Pascal's triangle is a triangular array of the binomial coefficients. Write a function that takes an integer value n as input and prints first n lines of the Pascal's triangle. Following are the first 6 rows of Pascal's Triangle.

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
```

Solution: http://www.geeksforgeeks.org/pascal-triangle/

☆ Find minimum in a rotated, sorted array

(Relatively easy problem, but still popular)

We're given an array with sorted numbers. The array has been rotated an unknown number of times. We need to figure out the minimum number in such an array. What would be a fast method that uses only constant space?

Solution: http://www.geeksforgeeks.org/find-minimum-element-in-a-sorted-and-rotated-array/

☆ Neuronyms

What's a Neuronym?

e.g. L10n, is called a Numeronym of the word Localization, where 10 stands for the number of letters between the first 'L' and the last 'n' in the word.

Generate all such possible Numeronyms for any given string (character array). for eg. for "nailed" :

"n4d"
"na3d", "n3ed"
"n2led", "na2ed", "nai2d"

e.g. for the word "batch"

"b3h"
"ba2h" "b2ch"

etc.

Print progressively longer strings, until there is a '2' in the permutations. There is no point of going below 2, because that won't compress the string.

Solution: Take inspiration from http://www.careercup.com/question?id=5733696185303040

☆ Hamming Weight

We're given a large array of 4-byte integers. We need to write a method to find out how many total bits are turned on (i.e. 1s are set) inside such an array. [Such a digital sum of binary representation of a number, is also called its Hamming Weight].

e.g.
1. if input array has two numbers: 31 and 51, the answer is 9, because 31 has 5 bits turned on (out of 32) and 51 has 4.
2. if the input is 2147483647 and 3, the answer is 31 + 2 = 33

We're looking for a fast solution, even if it uses extra memory. While it is possible to optimize solutions based on the machine architecture, we're not looking for intense bit-hackery. Assume input in base-10. No floating points.

Hint: Think hash tables.

Solution: See the top solution here: http://stackoverflow.com/questions/8871204/count-number-of-1s-in-binary-representation

Interview time: 25 minutes.

☆ SumZero

Given a set of integers, find a contiguous subset whose sum is zero. There can be duplicate numbers in the input.

Input: Integer array e.g. 5,1,2,-3,7,-4
output: A subset that sums to zero.
e.g. 1,2,-3 OR -3,7,-4

* If there are no such subsets, then print nothing
* If there are multiple such subsets, then print any one
* If a matching subset is a subset of a larger matching subset, then print either one
* If there is a number '0' in the array, then it counts as a valid answer subarray.

What would be the complexity of the solution, if we were to print all subsets that sum to zero (instead of just one)?

Solution: http://www.geeksforgeeks.org/find-if-there-is-a-subarray-with-0-sum/ (This is a variation of the Maximum Subarray problem)

Interview time: 25 minutes.

☆ Run Length Encoding

Simple version of the problem: Compress a string (only has alphabet characters), with basic encoding, where you simply count the number of repeated characters. Then also write a routine to decompress it.

e.g.
Input: "AAAAA"
Output: "5A"

Input: "BAAAB"
Output: "B3AB"

Input: "ABAB"
Output: "ABAB" [We are not concerned about characters repeating in groups]

Solution: Compression solution to this is very simple. It pretty much needs one loop. Decompression is equally simple. Let us know if that is not clear.

Important twists to the problem:
* String can have any character from the basic ASCII set (ASCII values 0 to 127). i.e. it can now include numbers.
* Compressed length must not exceed original length. It can be same or less.

Solution hint: Given that you cannot have numbers in your solution, can you use something else? Is there a one-character solution? Can you make use of higher order ASCII values?

Test cases are given for compression of 2nd (twisted) case.

HW 6 Test

☆ Rotation of a Palindrome

Given a string, check if it is a rotation of a palindrome. For example your function should return true for “aab” as it is a rotation of “aba”.

(Expect an O(N^2) solution!! It's the only solution to this problem, that's trivial to write in an interview)

☆ Check if a number is prime

There is a stream of numbers coming to you. You have to check if each of them is a prime number or not.

Write a function that takes a number and returns a boolean saying whether that number is a prime number or not.

☆ Read(n), with Read(k)

Given

You're given a function "int readK(char[] output)", which returns K bytes (characters) from an underlying file, puts into the 'output' array, and moves K bytes forward. The function always reads exactly K bytes, except if there aren't that many bytes available in the file (in which case it only moves that may bytes forward). Return value is the actual number of bytes read. K is a constant.

Implement

Using the function above, you're asked to implement "int readN(char [] output, int N)", which does exactly the same thing for N elements.

Constraints/allowances

1. N > K. They are both positive integers. K is a constant.

2. readN can also be called multiple times by the same caller.

3. Assume that there is nobody else touching that underlying file. Yours is the only interface.

4. You can use the function readK multiple times in your solution.

HW 7

☆ Print a string Sinusoidally

[This is just a stupid problem, that has no relation to anything else. It's there primarily because we see it on and off. It's a string puzzle disguised as a programming problem]

Also called "SnakeString". For example, the phrase "Google Worked" should be printed as follows (where ~ is the word separator):

```
   o     ~     k
 o g e W r   e
G   l   o     d
```

☆ Neuronyms

What's a Neuronym?

e.g. L10n, is called a Numeronym of the word Localization, where 10 stands for the number of letters between the first 'L' and the last 'n' in the word.

Generate all such possible Numeronyms for any given string (character array). for eg. for "nailed" :

"n4d"
"na3d", "n3ed"
"n2led", "na2ed", "nai2d"

e.g. for the word "batch"

"b3h"
"ba2h" "b2ch"

etc.

Print progressively longer strings, until there is a '2' in the permutations. There is no point of going below 2, because that won't compress the string.

Solution: Take inspiration from http://www.careercup.com/question?id=5733696185303040

☆ KMP

There are four well-known string matching algorithms, with their own advantages:

* Rabin Karp (short, understandable, we did this in the class)
* Knuth Morris Pratt (KMP, short, understandable)
* Aho-Corasick
* Boyer-Moore (Generally accepted benchmark)

Usually these algorithms are not asked in interviews directly. But there are two advantages of studying some of these:

1. They do ask generic question of finding text within text. They don't expect that you'd come up with one of these, but exposure to these can give you a lot of clarity and some quality talking points. Rabin Karp and KMP are the two easier ones that fit that bill.

2. The ideas in these algorithms make very good interview practice. Generally speaking, your interview-practice is not complete without going through them.

In this problem, we'd like you to implement the basic KMP algorithm: Given a (short) pattern and a (long) text, both strings, determine whether the pattern appears somewhere in the text.

Good explanation:
1. http://jakeboxer.com/blog/2009/12/13/the-knuth-morris-pratt-algorithm-in-my-own-words/
2. https://www.topcoder.com/community/data-science/data-science-tutorials/introduction-to-string-searching-algorithms/

Python: https://www.ics.uci.edu/~eppstein/161/python/kmp.py
C++: https://www.ics.uci.edu/~eppstein/161/kmp/
Java: http://algs4.cs.princeton.edu/53substring/KMP.java.html

☆ Minimum Window Substring

[We did this verbatim in the class]

Given a string S and a string T, find the minimum window in S which will contain all the characters in T.

e.g.
S = "AYZABOBECODXBANC"
T = "ABC"

Minimum window is "BANC",which contains all letters - A B and C.

* If no such window exists, then return an empty string
* If there are multiple minimum windows of the same length, then return any one
* Characters may be repeated

☆ Longest Substring with At Most Two Distinct Characters

Given a string, find the length of the longest substring T that contains at most 2 distinct characters.

For example, Given s = “eceba”,

T is "ece" which its length is 3.

* IF there are no such substrings (all same characters), then print nothing
* If there multiple such strings, then print any one

Solution: Take inspiration from: http://www.geeksforgeeks.org/find-the-longest-substring-with-k-unique-characters-in-a-given-string/

☆ Join words to make a palindrome

[This uses technique similar to the longest palindrome problem we did in the class]

Given a list of words, is there any pair of words, that can be joined (in any order) to form a palindrome?

Example 1:-
Consider a list {bat, tab, cat}. Then bat and tab can be joined together to form a palindrome.

Example 2 :-
{ab, deedba} can be joined to form a palindrome.

Example 3 :-
{'ant', 'cat', 'dog'}: No two words here can be joined to form a palindrome.

There can be multiple pairs, just return the first one if found.

About expected solutions:

Given n = number of words and k is length of the longest word, O(N.K^2) solution for this problem is relatively straightforward to come by, using Maps or Tries. That may suffice for most interviews.
e.g.
https://discuss.leetcode.com/topic/40657/150-ms-45-lines-java-solution
or
https://discuss.leetcode.com/topic/39585/o-n-k-2-java-solution-with-trie-structure-n-total-number-of-words-k-average-length-of-each-word

In order to increase the average case efficiency, see if you can apply the Mancher technique we learnt in the class. That can bring it down to O(N.K): https://www.quora.com/Given-a-list-of-words-can-two-words-be-joined-together-to-form-a-palindrome

☆ Regex Matcher

[We did a similar in the class]

Implement a regular expression matcher, supporting the following characters

- a dot ('.') '.' Matches any single character.
- a star ('*') '*' Matches zero or more of the preceding element.

☆ Longest repeated substring

Given a string, find the longest repeated substring in it.

* Repeated is occurring more than once. It doesn't matter how many times it occurs beyond 2 times.
* If there are multiple such strings of the same size, then get any one of them
* If there are no repeated substrings, then don't print anything (empty output)

Solution: http://www.geeksforgeeks.org/suffix-tree-application-3-longest-repeated-substring/

[Context: This is purely an exercise in building a Suffix Tree

Suffix trees are difficult. You'd probably wonder if they really ask those in an interview.

They are in-fact, are rarely asked, which is why we don't cover it in the class. But we've seen them at FB and Uber. In all occasions, it's been asked as a follow up question. Once you code up an N^2 algorithm for the problem on hand, there are a few minutes left, in which time, the interviewer would wonder if you know of Suffix trees. It is NEVER asked to implement one in an interview. That's stupid.

If at that time, you do know of suffix trees, then you have a chance to convert that interviewer from a 3 (good) to a 4 (advocate). It suggests you have taken a keen interest in your prep work and by extension, in general CS.

Another reason we include it in the course: It's possibly one of the hardest data structures. Once you have a handle on it, a lot of other things will look easy ;-)

Doing difficult problems like these also has a strong ancillary benefit: it helps you indirectly interview your interviewer/company.

You want to work for a team that challenges you; not the team that gives you a free pass.

i.e. Don't skimp on it. Take it head on - there are clear benefits.]

☆ Boggle Solver

[This needs knowledge of basic backtracking. Problem definition courtesy Geeks For Geeks. Otherwise it's a popular and age-old problem as well as a popular game]

Given a dictionary, a method to do lookup in dictionary and a M x N board where every cell has one character. Find all possible words that can be formed by a sequence of adjacent characters. Note that we can move to any of 8 adjacent characters, but a word should not have multiple instances of same cell.

Example:

```
Input: dictionary[] = {"GEEKS", "FOR", "QUIZ", "GO"};
       boggle[][]   = {{'G','I','Z'},
                       {'U','E','K'},
                       {'Q','S','E'}};
      isWord(str): returns true if str is present in dictionary
                   else false.

Output:  Following words of dictionary are present
         GEEKS
         QUIZ
```

This is easy to understand, but is inefficient compared to using a Trie: http://www.geeksforgeeks.org/boggle-find-possible-words-board-characters/

This uses a Trie: http://stackoverflow.com/a/746102/327310 (This is usually enough for an interview at everywhere but the top places)

This uses a Trie and does some DP: https://github.com/bilash/boggle-solver (At top places, this is what is expected)

HW 7 Test

☆ Print a Matrix in Spiral order

Print a matrix in spiral order.
e.g.

X Y A
M B C
P Q R

Output: XYACRQPMB

* This problem is less about logic, but more about careful index manipulation.
* Assume a full and valid matrix. No need to spend time validating the input.
* Matrix need not be square. It can be rectangular.
* Avoid recursion.

* Hint - It may be faster to write this, if you name your variables clearly. Instead of i,j,k,l etc, try naming them like row, col, start, end etc. That will also help your interviewer follow along more easily.

☆ Reverse words in a String

Given a string containing a set of words, transform it such that the words appear in the reverse order. Words are separated only by one or more whitespace characters.

Examples:
1. “I will do it.” should change to “it. do will I”.
2. "[ ][ ]word1[ ]word2[ ]" should transform into "[ ]word2[ ]word1[ ][ ]" (where [ ] denotes a single space)
3. "word1;[ ]word2." should transform into "word2.[ ]word1;".

* Notice that the punctuation is part of the word.
* We need to preserve case, but we don't need to preserve the original string.
* Usage of inbuilt string functions isn't allowed.

* An in-place linear solution is expected
* For languages that have immutable strings, convert the input string into a Character Array and work in-place on that array. Convert it back to the string before returning. (For the purpose of this problem, ignore the extra linear space used in that conversion, as long as you're only using constant space after conversion to character array)

[Trivia: This is a very old interview question, which Google used last year as one of their qualifier questions in Google CodeJam)

☆ Move all letters to left side

You're given a character array, which may contain alphabet letters (a to z or A to Z) as well as numbers (0 to 9, represented as characters), in random order. You have to make alphabet letters appear on left side, inside the same array.

e.g. If your input is [0,a,1,9,3,z,b,r,6], then in your output, letters a, z, b, and r, should be seen on left side in the array.

* Original order of letters needs to be preserved.
* Order of numbers doesn't need to be preserved. If a test-case fails, then check if it's only because the order of numbers is different. If so, then the failure is irrelevant and can be ignored.
* Repeats are allowed.

* An in-place linear solution is expected
* For languages that have immutable strings, convert the input string into a Character Array and work in-place on that array. Convert it back to the string before returning. (For the purpose of this problem, ignore the extra linear space used in that conversion, as long as you're only using constant space after conversion to character array)

* Extension: Minimize the number of array-writes in your solution. i.e. Re-read the problem-statement, and think whether you really need to write/over-write that letter/number.

Test cases given, are for the extension problem.

HW 8

☆ Snakes and Ladders Matrix

Given a snake and ladder rectangular MxN board-game, find the minimum number of dice throws required to reach the final cell from the 1st cell.

Rules are as usual: If after a dice-throw, the player reaches a cell where the ladder starts, the player has to climb up that ladder and if the player reaches a cell that has the mouth of the snake, s/he has to go down to the tail of snake.

For example, in the board given below, it will take minimum 4 throws to reach from 1 to 36. That can be done with the following sequence of throws: {1,6,4,1}. There may be more such sequences of the same length viz. {4,2,6,4} etc.

Please hard-code input game boards as your test-cases. There are different ways of doing so. e.g. one simple way, is to represent it using a one-dimensional array of length MxN, with each element representing a cell. Values in the array, are the destination cell id for snakes (lower numbers) and ladders (higher numbers).

Solution: http://www.geeksforgeeks.org/snake-ladder-problem-2/

(Suggested time: 45 minutes)

☆ Knight's tour on a Chess board (graph)

(This is an interview twist on a classical CS problem)

Assume you're given a normal chessboard and a knight that moves like in normal chess. You are then given two inputs: starting location and ending location in the form of x,y co-ordinates. The goal is to then calculate and print the shortest path that the knight can take to get to the target location.

Solution: http://stackoverflow.com/questions/2339101/knights-shortest-path-chess-question

☆ Detecting Cycle in a graph

Given a directed graph, check whether there is a cycle in it.

* There can be multiple cycles
* We don't need to print all the cycles. Just return a boolean true/false if there is/is-not at least one cycle respectively

Solution: http://www.geeksforgeeks.org/detect-cycle-in-a-graph/

Suggested time: 45 minutes max.

☆ Topological Sort

Given a sorted dictionary of an alien language, find order of characters.

Example-1:
Input: words[] = {"baa", "abcd", "abca", "cab", "cad"}
Output: Order of characters is 'b', 'd', 'a', 'c'

Note that words are sorted and in the given language. "baa" comes before "abcd", therefore 'b' is before 'a' in output.

Example-2:
Input: words[] = {"caa", "aaa", "aab"}
Output: Order of characters is 'c', 'a', 'b'

Solution:
http://www.geeksforgeeks.org/topological-sorting/
http://www.geeksforgeeks.org/given-sorted-dictionary-find-precedence-characters/

☆ Skip Lists!

Implement a simple Skip List. (Certain startups are enamored with Skip Lists in their interviews)

Solution: http://igoro.com/archive/skip-lists-are-fascinating/

Bonus points: Read this discussion on StackOverflow, that compares and contrasts skip lists with BSTs: http://stackoverflow.com/questions/256511/skip-list-vs-binary-tree

☆ Rainfall Challenge

(This is a popular Palantir problem)

Problem Statement

A group of farmers has some elevation data, and we're going to help them understand how rainfall flows over their farmland.

We'll represent the land as a two-dimensional array of altitudes and use the following model, based on the idea that water flows downhill:

If a cell's four neighboring cells all have higher altitudes, we call this cell a sink; water collects in sinks.

Otherwise, water will flow to the neighboring cell with the lowest altitude. If a cell is not a sink, you may assume it has a unique lowest neighbor and that this neighbor will be lower than the cell.

Cells that drain into the same sink – directly or indirectly – are said to be part of the same basin.

Your challenge is to partition the map into basins. In particular, given a map of elevations, your code should partition the map into basins and output the sizes of the basins, in descending order.

Assume the elevation maps are square. Some farmers have small land plots such as the examples below, while some have larger plots. However, in no case will a farmer have a plot of land larger than S = 5000.

Your code should output a space-separated list of the basin sizes, in descending order.

Suggested time: 55 minutes. This is difficult to understand, code and test in under an hour. But that's the expectation.

Solution

The problem and the solution both, are from here: http://codereview.stackexchange.com/questions/38500/rainfall-challenge Please be sure to read author's solution and criticism of the answer. It's very instructive.

A few examples are below:

```
----------------------------------------
Input:                 Output:
 3                      7 2
 1 5 2
 2 4 7
 3 6 9

The basins, labeled with A's and B's, are:
 A A B
 A A B
 A A A
----------------------------------------
Input:                  Output:
 1                       1
 10

There is only one basin in this case.
The basin, labeled with A's is:
 A
----------------------------------------
Input:                  Output:
 5                       11 7 7
 1 0 2 5 8
 2 3 4 7 9
 3 5 7 8 9
 1 2 5 4 3
 3 3 5 2 1

The basins, labeled with A's, B's, and C's, are:
 A A A A A
 A A A A A
 B B A C C
```

☆ Reverse Index

Given a text file and a word, find the positions that the word occurs in the file. We'll be asked to find the positions of many words in the same file. i.e. think precomputing with a reverse index.

Implement the solution using a HashTable and a Trie. Write full implementation of Trie. You can use existing implementation of a HashTable.

[This is almost a mini-project. Best done outside HackerRank. If you want to do inside, you'll need to write to a file first before reading from it: https://www.hackerrank.com/environment/writing-to-file]

Solution: http://www.ardendertat.com/2011/12/20/programming-interview-questions-23-find-word-positions-in-text/

☆ Graph representations

How many ways can a graph be represented?

PICK THE CORRECT CHOICES

- [ ] Adjacency List
- [ ] Matrix
- [ ] One variable per node
- [ ] Neighbor list
- [ ] All of the above
- [ ] None of the above

Clear selection

Submit answer & continue

You can change your submission later.

☆ Bloom Filter

Code a simple Bloom Filter.

1. Read words of an English dictionary from a file. A large number (>100K) is required. Create one, or use one on your system (e.g. /usr/share/dict/words)
2. Add the words to a bloom filter (not a hash table, but a bloom filter).
3. Look up random words from the bloom filter
4. Do a comparison with linear-time searching the dictionary. You can also compare with Hash-table, but depending on how fast our machine is and how large your data-set, you may or may not see much speed up.

Solution: http://www.maxburstein.com/blog/creating-a-simple-bloom-filter/

☆ Convert string from a to b, using a dictionary of words

This is an example of a problem, that can be modeled as a Graph problem. It's a simplified version of the famous Edit Distance problem (which is DP).

You have a dictionary of words and two strings `a` and `b`.

How can one convert a to b by changing only one character at a time and making sure that all the intermediate words are in the dictionary?

Example:

```
dictionary: {"cat", "bat", "hat", "bad", "had"}
a = "bat"
b = "had"
```

solution:

`"bat" -> "bad" -> "had"`

Solution: http://stackoverflow.com/questions/17514999/convert-string-a-to-b-using-a-dictionary-of-words

HW 8 Test

☆ Find the number of islands

Given a boolean 2D matrix, find how many islands it has.

An island is a group of connected '1's. For example, the matrix below has 5 islands:

{1, 1, 0, 0, 0},
{0, 1, 0, 0, 1},
{1, 0, 0, 1, 1},
{0, 0, 0, 0, 0},
{1, 0, 1, 0, 1}

* Input matrix may or may not be square
* Group of connected '1's can be in any direction i.e. up, down, sideways or diagonally. There are 8 of these possible directions
* Individual '1s' are considered an island by themselves. There are 3 of those above, in the last line
* You don't have to provide co-ordinates of islands; just have to tell the total count of islands

About expected solution
* You are allowed to modify the input matrix
* Use as little extra memory as you can

☆ Clone a graph and verify the clone

Given a reference to a connected, undirected graph, clone it, verify it and return the reference to the new graph.

* Please hard code a graph input in your solution, clone it like the problem asks, compare the clone to the original graph, and return a reference to the clone.
* Structure of each node of the graph has: a value and a list of neighbors.
* Edges are not directional, and not weighted. We don't care about what value each node has.
* This is an exercise in cloning, as well as traversing two graphs together (input and cloned), for comparison.

* Sample inputs to test with:

1. Single node graph
2. A binary tree (which is also a graph)
3. A linked list (which is also a graph)
4. A graph with a loop

Pramp

Interview 1 Instructor

Smallest Substring of All Characters

Given an array with unique characters arr and a string str, find the smallest substring of str containing all characters of arr.

Example:
arr: [x,y,z], str: xyyzyzyx
result: zyx

Implement your solution and analyze the runtime complexity

Hints & Tips
If your peer is stuck, ask how can we determine if a given substring is valid (all chars from set are in it) and then ask how to apply that to a solution

If your peer is using a naive solution of checking all possible substrings, try to ask how can you avoid duplicate work

Make sure proper initializations are made

Watch for unnecessary variables and steps

For other solutions, make sure that any permutation of the characters in set can be found by the algorithm

make sure your peer understand why we should increase tail only after head is increased

Solution
We iterate the string from left to right, while using two indices - tailIndex and h.
At each iteration step, we examine the temp substring [str.charAt(tailIndex), str.charAt(tailIndex+1) …, str.charAt(h)] and keep a copy of the shortest vaild substring we've seen so far.

To examine substrings we use 2 counters:
uniqueCounter (integer) - number of unique characters of arr in our temp substring
countMap (map/object/associative array - depends of your language of choice) - number of occurrences of each char from arr in our substring

function getShortestUniqueSubstring(arr, str):
t = 0
result = null
uniqueCounter = 0
countMap = new Map()

initialize countMap:

for i from 0 to length(arr)-1:
countMap.setValueOf(arr[i], 0)

scan str

for h from 0 to length(str)-1:
# handle the new head
head = str.charAt(h)
if countMap.keyExists(head) == false:
continue
headCount = countMap.getValueOf(head)
if headCount == 0:
uniqueCounter = uniqueCounter + 1
countMap.setValueOf(head, headCount + 1)
# push tail forward
while uniqueCounter == length(arr):

tempLength = h - t + 1
if tempLength == arr.length:
return str.substring(t, h)
if (!result or tempLength < length(result)):
result = str.substring(t, h)
tail = str.charAt(t)
if countMap.keyExists(tail):
tailCount = countMap.getValueOf(tail) - 1
if tailCount == 0:
uniqueCounter = uniqueCounter - 1
countMap.setValueFor(tail, tailCount)
t = t + 1
return result
Runtime Complexity: we're doing a linear iteration of both str and arr of lengths n and m respectively, so the runtime complexity is a linear O(n+m).

Space Complexity: depends of your implementation for the mapping, but generally: we're using countMap with m keys (the length of arr) plus few constant size counters - O(m) space complexity.

Interview 1 Interviewee

The "Award Budget Cuts" Problem

The awards committee had planned to give n research grants this year, out of a its total yearly budget.
However, the budget was reduced to b dollars. The committee members has decided to affect the minimal number of highest grants, by applying a maximum cap c on all grants: every grant that was planned to be higher than c will now be c dollars.
Help the committee to choose the right value of c that would make the total sum of grants equal to the new budget.

Given an array of grants g and a new budget b, explain and code an efficient method to find the cap c. Assume that each grant is unique.
Analyze the time and space complexity of your solution.

??? SOLUTION?

Interview 2 Instructor

K-Messed Array Sort

Given an array arr of length n where each element is at most k places away from its sorted position,
Plan and code an efficient algorithm to sort arr.
Analyze the runtime and space complexity of your solution.

Example: n=10, k=2. The element belonging to index 6 in the sorted array, may be at indices 4, 5, 6, 7 or 8 on the given array.

Hints & Tips
Try to help your peer think about the advantages of this nearly k-sorted array. Then ask how can it be useful.

As a hint ask your peer, for the 0-index place on the array, how far can the item that belongs there be, and how can that be generalized (building the iterative step).

This question is a good opprtunity to check if your peer remebers what are insertion sort and heap sort, and for you both to brush up on that. A good source to get that covered is Sorting Algorithm article on Wikipedia.

If your peer does not know about insertion sort or heap sort, make sure his knowledge section on the interview feedback reflect that

To get the best feedback rating on the problem solving section your peer should plan, explain and execute an O(n • log k) solution

If min-heap doesn't exist on the interview's coding language let your use a min-heap object as if it exists:
Valid operations are new MinHeap(), extractMin() and insert()

Watch for correct calculations and usage of array indices

If your peer is completely stuck, help the thought process by asking what can you do with a sliding window of size k+1

If relevant data structures (heap or others used by your peer) don't exist in your language of choice, you can assume their existence once your peer explain their concepts and operations and you both agree on its API.

Solution

The suboptimal approach to solve is by using Insertion Sort:
We iterate the arr from left to right:
On each iteration we bring arr[i] to its place in the subarray arr[0,1,2 … i], by shifting any subarray elements that are bigger than A[i] one place right.

```
function insertionSort(arr):
for i from 1 to arr.length-1:
x = arr[i]
j = i-1
while (j >= 0 and arr[j] > x):
arr[j+1] = arr[j]
j–
arr[j+1] = x
return arr
```

Runtime complexity: iteration over array of length n and switching at most k pairs for each iteration (by definition of the given array), takes O(n • k).
If k is constant and relatively small we can argue that it's actually close to a linear O(n) case.

Space complexity: constant O(1), all we need is 2 indices.

However, we can do better:

If we use a modified Heap Sort we can get better runtime complexity:
We define a virtual 'sliding window' of from the first k+1 elements of arr.
First we build a min heap from the elements in the window.
Then, we start sliding: on each step we extract the minimum from the heap, move the window one place right, place the min we've extracted into index that is now left to the window and insert the new element at the end of the window to the heap. We repeat that until the window reaches the end of the array, then we extract the minimum from the heap and place it on the next index of arr, until the heap is empty.

```
function kHeapSort(arr, k)
h = new MinHeap()
n = length(arr)
for i from 0 to k:
h.insert(arr[i])
for i from k+1 to n-1:
arr[i-(k+1)] = h.extractMin()
h.insert(arr[i])
for i from 0 to k:
arr[n-k-1 + i] = h.extractMin()
```

return arr
Runtime complexity:
Building a heap takes linear O(k) for k+1 elements.
Operating the heap later involves extracting min on a min-heap / inserting to the heap. These actions take O(log k) each. We do at least one of these actions n times, so the cost here is O(n • log k).
the overall runtime complexity of the heap solution is O(n • log k). again, if k is constant, we may argue the complexity is close to linear.

Space complexity:
we need to hold a min heap of k+1 elements.
Since a heap is usually implemented with an array the space complexity is O(k+1).
However, we can implement and maintain the heap manually on our conceptual sliding window subarray. If we handled right, it can lead us to a constant O(1) space complexity.

Interview 2 Interviewee

The “Quad Combination” Problem

Given an array of numbers arr and a number S, find 4 different numbers in arr that sum up to S.

Write a function that gets arr and S and returns an array with 4 indices of such numbers in arr, or null if no such combination exists.
Explain and code the most efficient solution possible, and analyze its runtime and space complexity.

Solution

The naive solution is to iterate on every possible combination of 4 numbers from arr until the required combination if found. Using 4 nested loop involves O(n4) time complexity and O(1) space complexity. This is quite inefficient.

We can do better, if we look at all the pairs in arr, and then try to build the sum S from 2 different pairs.
First, we iterate over all the possible pairs in arr with 2 nested loops and hash each pair by its sum. Then, for each pairSum in the pairs hash table, we look for its complement S - pairSum. When we find two pairs that sum up to S, we need to check that these pairs are drawn from 4 different indices in arr (in other words: that no number is used twice to reach the desired sum).

```
function findArrayQuadCombination(arr, S):
if (arr == null OR S == null):
return null
n = length(arr)
if (n < 4):
return null
```

hashing implementation language dependent:

```
pairHash = new HashTable()
for i from 0 to n-1
for j from i+1 to n-1
if !pairHash.isMapped(arr[i]+arr[j]):
pairHash.map(arr[i]+arr[j], [])
pairHash.get(arr[i]+arr[j]).push([i, j])

for pairSum in pairHash.getKeys()
if pairHash.isMapped(S - pairSum):
pairsA = pairHash.get(pairSum)
pairsB = pairHash.get(S - pairsSum)
combination = find4Uniques(pairsA, pairsB)
if (combination != null):
return combination
return null
```

Helper function.

Gets 2 arrays of sub-arrays of 2 numbers

Gets 4 unique numbers, from 2 sub-arrays of different arrays

```
function find4Uniques(A, B):
lenA = length(A)
lenB = length(B)
for i from 0 to lenA-1:
for j from 0 to lenB-1:
if ( A[i][0] == B[j][0] OR A[i][1] == B[j][1] OR
A[i][0] == B[j][1] OR A[i][1] == B[j][0] ):
continue
else:
return [A[i][0], A[i][1], B[j][0], B[j][1]]
return null
```

Time Complexity: Let n be the length or arr. Hashing all pairs in arr by their sum and iterating over all sums and their complements takes O(n2) time (n2 pairs and constant number of actions for each). Uniqueness check for all indices of the pairs of sums that adds up to S until a valid combination is found, is also O(n2) (checking at most n2 pairs with 4 comparisons for each). Overall: quadratic O(n2) time complexity.

Space Complexity: n2 pairs have up to n2 different sums. Hashing them takes O(n2) space complexity.

Interview 3 Interviewee

```
function findComplementingWeights(arr, limit):
h = new hashTable()
for (index, w) in arr:
complementIndex = h.findKey(limit - w)
if (complementIndex != null):
return [index, complementIndex]
else:
h.insert(w, index)
return -1
```

Interview 4 Interviewee

```
class Pramp {

static void print(int[][] M) {
printHelper(M, M[0].length, M.length, 0, 0);
}

static void printHelper(int[][] M, int n, int m, int row, int col) {
```

```
    if (row > m / 2 || col > n / 2) {
        return;
    }

    /*
    int left = 0, right = M[0].length-1, top = 0, bottom = M.length-1;

    for (int j=left; j<=right; j++) {
        print(M[0][j]);
    }

    top ++;
    for (int i=top; i<=bottom; i++)
        print(M[][]);

    right --;


    */

    // Print top
    for (int j = row; j < n - col; j++) {
        System.out.println(M[row][j]);
    }

    // Print right column
    for (int i = row + 1; i < m - (row + 1); i++) {
        System.out.println(M[i][col]);
    }

    // Print bottom
    for (int j = n - col - 1; j >= col; j--) {
        System.out.println(M[m-row-1][j]);
    }

    // Print left column
    for (int i = m - row; i > row; i--) {
        System.out.println(M[i][col]);
    }

    printHelper(M, n, m, row + 1, col + 1);
```

}

public static void main(String[] args) {
String pramp = "Practice Makes Perfect";
System.out.println(pramp);

}
}*/

Interview 5 Interviewee

islands question

Interview 6 Interviewee

arrays question

Interview 7 Interviewee

The “Word Count Engine” Problem

Implement a document scanning engine that receives a text document doc and returns a list of all unique words in it and their number of occurrences, sorted by the number of occurrences in descending order.

Example:
for doc: “practice makes perfect. get perfect by practice. just practice!”
the engine returns the list: { practice: 3, perfect: 2, makes: 1, get: 1, by: 1, just: 1 }.

The engine should ignore punctuation and white-spaces.
Find the minimal runtime complexity and analyze it.

static class Result {
String word;
int count;
public Result(String word, int count) {
this.word = word;
this.count = count
}
}

static List wordCount(List doc) {

List result = new ArrayList<>();

Map<String, Integer> textToCount = new HashMap<>();
TreeMap<Integer, List> reverseMap = new TreeMap<>();
for (String word : doc) {
if (!textToCount.containsKey(word)) {
textToCount.put(word, 0);
}

```
   int currCount = textToCount.get(word);
   int newCount = currCount + 1;

   if (!reverseMap.containsKey(newCount)) {
      reverseMap.put(newCount, new ArrayList<>());
   }

   reverseMap.get(newCount).add(word);
   textToCount.put(word, textToCount.get(word) + 1);
```

}

// Output
Set printed = new HashSet<>();
for (Integer count : reverseMap.descendingKeySet()) {
List words = reverseMap.get(count);
for (String word : words) {
if (!printed.contains(word)) {
result.add(new Result(word, count));
printed.add(word);
}
}

}
return result;
}

Interviewee 8

The “Busiest Time in The Mall” Problem

The mall management is trying to figure out what was the busiest moment in the mall in the last year.
You are given data from the door detectors: each data entry includes a timestamp (seconds in Unix Epoch format), an amount of people and whether they entered or exited.

Example of a data entry:
{ time: 1440084737, count: 4, type: “enter” }

Find what was the busiest period in the mall on the last year. Return an array with two Epoch timestamps representing the beginning and end of that period. You may assume that the data your are given is accurate and that each second with entries or exits is recorded. Implement the most efficient solution possible and analyze its time and space complexity.

Interviewee 9

The “BST Successor Search” Problem

Given a node n in a binary search tree, explain and code the most efficient way to find the successor of n.
Analyze the runtime complexity of your solution.

Interviewee 10

Dictionary print thing

Given a node n in a binary search tree, explain and code the most efficient way to find the successor of n.
Analyze the runtime complexity of your solution.

Interview.io

Steps question, recursively

generate prime factorization

Divya 1

pacific, atlantic, dfs
Given BST and a target value. Count the number of pairs of values which sum to the target value.

Given BST and a target value. Count the number of pairs of values which sum to the target value.

Ex: 0
-2 4
-2 4 5
-3 6
11
target = 4
output: 2 (0,4) (6,-2)

// n is # nodes
// For each node in tree: O(n)
// val = node.val
// Find if target - val exists O(logn)
// Final dedup step, O(n)

// Runtime : O(nlogn)
// Space: O(n^2) – due to no dedup

// Traverse tree once, both ends
// (-3, 11), t = 4,
// (-3, 11)

// Case 1: assuming no duplicates, move both pointers to sucessor and predecessor
// Case 2: val > targer, where val is sum of pointers
// Case 3:

// Recursive function that returns leftmost node, rightmost node in an LL
// Base case: single BST leaf node: {node, node}
//

static class Spine {
Node precursor;
Node successor;
}

Ex: 0
-2 4
-2 4 5
-3 6
11

static Spine convertBST(Node n) {
if (n == null) {
return null;
}

```
if (n.left == null && n.right == null) {
    return new Spine(n, n);
}

Spine right = convertBST(n.right);
Spine left = convertBST(n.left);

// Link the spines up
n.left = left.sucessor;
if (left != null && left.sucessor != null) {
    left.sucessor.right = n;
}
n.right = right.precursor;
if (right != null && right.precursor != null) {
    right.precursor.left = n;
}

// Return the linked up spine
return new Spine(left.precursor, right.sucessor);
```

}

// Returns the head of the new LL
static Spine convertToLL(Node n) {

return convertBST(n); // head and tail of LL
}

l r
-3 -2 -2 0 4 5 6 11

-4

target = 8
-3, 11, val = 8, count = 1
-2, 6, val = 4, count = 1,
-2, 6, val = 4, count = 1,
0, 6, val = 6, count = 1,
4, 6, val = 10, count = 1,
4, 5, val = 9, count = 1,
4, 4, val = 8, count = 2,

static int countPairs(Node root, int target) {
if (root == null) {
throw new IllegalArgumentException();
}

```
int count = 0;
Spine spine = convertToLL(root);
Node leftN = spine.precursor;
Node rightN = spine.sucessor;

while (leftN != rightN && rightN.right != leftN) {
    int val = leftN.val + rightN.val;

    // Move both pointers inward if val == target
    if (val == target) {
        count++;
        leftN = leftN.right;
        rightN = rightN.left;
    } else if (val < target) {
        // Increase val by moving left pointer inward
        leftN = leftN.right;
    } else {
        // Decrease val by moving right pointer inward
        rightN = rightN.left;
    }
}
return count;
```

}

$$R_{\mu}$$

```
static String[] palindromicDecomposition(String str) {

    Deque<Interval> curr = new LinkedList<>();
    List<List<String>> results = new LinkedList<>();

    generatePalindromes(results, curr, 0, str);

    List<String> result = new LinkedList<>();
    for (List<String> l : results) {
        Iterator<String> it = l.iterator();
        StringBuilder builder = new StringBuilder();
        while(it.hasNext()) {
            builder.append(it.next());
            if (it.hasNext()) {
                builder.append('|');
            }
        }
        result.add(builder.toString());
    }

    return result.toArray(new String[result.size()]);
}
```