Real Programming Day 1

Total Questions : 5

Marks: 1 and 2 are of 5 marks, 2 and 3 are of 10 marks and 5 is of 15 marks.

1. **Problem Definition**

   As some of you may know, there is no name better than JOHN. Let's define the rules for comparing names. Each letter has a weight ('A' - 1, 'B' - 2, ..., 'Z' - 26). The weight of a name is the sum of the weights of all its letters. For example, the name MARK has weight 13 + 1 + 18 + 11 = 43.

   When comparing two names, the one with the larger weight is considered better. In case of a tie, the one that comes earlier lexicographically is better. But there is one exception - the name JOHN is the best name of all.

   You are given a String[] names, each element of which contains a single name. Sort the names from best to worst and return the sorted String[].

   Definition

   Class: LuckyName
   Method: sort
   Parameters: String[]
   Returns: String[]
   Method signature: String[] sort(String[] names)
   (be sure your method is public)

   Constraints

   - names will contain between 1 and 50 elements, inclusive.
   - Each element of names will contain between 1 and 50 characters, inclusive.
   - Each element of names will contain only uppercase letters ('A'-'Z').

   Examples

   0)

   {"JOHN", "PETR", "ACRUSH"}
   Returns: {"JOHN", "ACRUSH", "PETR" }
   PETR has weight 59, ACRUSH has weight 70 and JOHN has a weight of only 47. But nevertheless JOHN is the lucky name, ACRUSH takes second place and PETR is the last.

   1)

{"GLUK", "MARGARITKA"}
Returns: {"MARGARITKA", "GLUK" }
MARGARITKA is definitely better than GLUK.
2)

{"JOHN", "A", "AA", "AAA", "JOHN", "B", "BB", "BBB", "JOHN", "C", "CC", "CCC", "JOHN"}
Returns:
{"JOHN",
"JOHN",
"JOHN",
"JOHN",
"CCC",
"BBB",
"CC",
"BB",
"AAA",
"C",
"AA",
"B",
"A" }
AA and B both have the same weight, but AA is better as it comes earlier lexicographically. For the same reason, AAA is better than C and BBB is better than CC.
3)

{"BATMAN", "SUPERMAN", "SPIDERMAN", "TERMINATOR"}
Returns: {"TERMINATOR", "SUPERMAN", "SPIDERMAN", "BATMAN" }

2.
Problem Statement

The candy industry is going through a hard time in Byteland. Some of the biggest companies in the business have decided to perform a series of mergers so as to become one company in the end. Surprisingly, empirical studies conducted by the economists of Byteland have shown that for any m >= 2 the revenue of a company that is created by simultainously merging m companies with revenues equal to r0, r1, ..., rm - 1 is equal to the average of these revenues, that is (r0 + r1 + ... + rm - 1) / m.

You are given a int[] revenues. The revenue of the i-th of the companies that want to merge is equal to revenues[i]. Return the maximum possible revenue of the final company that can be created in any series of mergers that joins all the companies. In each of the mergers, we may merge as many of the currently existing companies as we wish.

Definition

Class: BestMerging
Method: findMaximum
Parameters: int[]
Returns: double
Method signature: double findMaximum(int[] revenues)
(be sure your method is public)

Notes

- The returned value must have an absolute or relative error less than 10-9.
- Please note that the revenue of a company may be negative; this means that the company is actually losing money.
- It is always possible to merge all companies into a single one: for example, by merging all of them in a single step.

Constraints

- revenues will contain between 2 and 50 elements, inclusive.
- Each element of revenues will be between -1,000 and 1,000, inclusive.

Examples

0)

{5, -7, 3}
Returns: 1.5
The optimal way is to first merge companies 1 and 2, obtaining a company with total revenue -2, and then merge that company with company 0.

1)

{10, -17}
Returns: -3.5

2)

{12, 12, 12, 12, 12}

Returns: 12.0
We can just merge all the companies at once.
3)

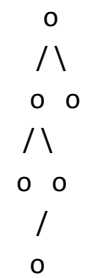
{0, 0, 0, 0, 0, 100}
Returns: 50.0
We may first merge companies 0 through 4 and then merge the resulting company with company 5.
4)

{10, -10, 100, -100, 1000, -1000}
Returns: 491.25

3.
Problem Statement

Define the height of a binary tree to be the number of nodes in the longest path from the root to a leaf. The empty tree is considered to have height 0. A node is k-balanced if its left and right subtrees differ in height by at most k. A tree is k-balanced if all of its nodes are k-balanced. The empty tree is considered to be k-balanced.

For example, the tree below has height 4.



This tree is 2-balanced but not 1-balanced, because the left subtree of the root has height 3 and the right subtree of the root has height 1.
Your task is to write a method that takes a balance factor k and a number of nodes n and returns the maximum height of a k-balanced tree with n nodes.

Definition

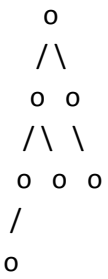
Class: BalancedTrees
Method: maxHeight
Parameters: int, int

Returns: int
Method signature: int maxHeight(int k, int n)
(be sure your method is public)

Constraints
- k is between 1 and 100, inclusive.
- n is between 1 and 1000000, inclusive.

Examples
0)

1
7
Returns: 4
A tree that achieves the maximum height for 7 nodes and balance factor 1 is



1)

2
40
Returns: 9
2)

10
5
Returns: 5
With k=10, a tree of size 5 can be completely linear (eg, every right subtree is empty) without violating the balance factor.

4.

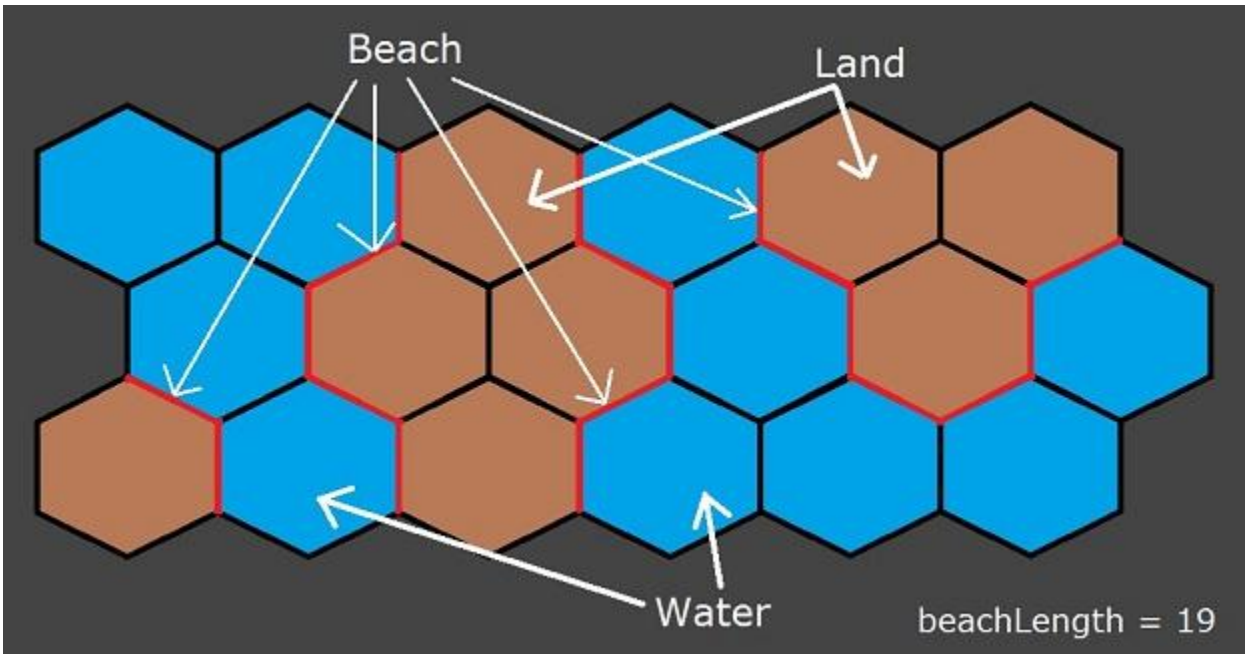
Problem Statement

The king is trying to find new ways to generate revenue, and he is currently exploring tourism as one potential avenue. The kingdom is a group of islands, and the amount of revenue that can be generated depends on the combined total length of beaches on all the islands.

You are given a String[] kingdom consisting of '.' or '#' characters. '#' represents a land mass, whereas '.' represents water. kingdom[i][j] represents a regular-hexagon shaped area with each side of unit length. Since the cells are hexagonal in shape, the odd-numbered rows (0-based) are 'shifted' towards the right. A beach is a segment which has water on one side, and land on the other.

An example String[] and the corresponding image are given below to illustrate. The beaches are marked in red.

{"..#.##",
 ".##.#.",
 "#.#..."}



Return the combined total length of beaches on all the islands.

Definition

Class: Islands
Method: LengthOfCoast
Parameters: String[]

Returns: int
Method signature: int beachLength(String[] kingdom)
(be sure your method is public)

Constraints
- kingdom will contain between 1 and 50 elements, inclusive.
- Each element of kingdom will contain between 1 and 50 characters, inclusive.
- Each element of kingdom will contain the same number of characters.
- Each character in kingdom will be either '.' or '#'.

Examples
0)

{".#...#.."}
Returns: 4
There are two small islands with water on two sides of each island.
1)

{"..#.##",
 ".##.#.",
 "#.#..."}
Returns: 19
The example in the problem statement.
2)

{"#...#.....",
 "##..#...#."}
Returns: 15
3)

{"....#.",
 ".#....",
 "..#..#",
 "####.."}
Returns: 24

5.
Problem Statement

A sequence of numbers is called a zig-zag sequence if the differences between successive numbers strictly alternate between positive and negative. The first difference (if one exists) may be either positive or negative. A sequence with fewer than two elements is trivially a zig-zag sequence.

For example, 1,7,4,9,2,5 is a zig-zag sequence because the differences (6,-3,5,-7,3) are alternately positive and negative. In contrast, 1,4,7,2,5 and 1,7,4,5,5 are not zig-zag sequences, the first because its first two differences are positive and the second because its last difference is zero.

Given a sequence of integers, sequence, return the length of the longest subsequence of sequence that is a zig-zag sequence. A subsequence is obtained by deleting some number of elements (possibly zero) from the original sequence, leaving the remaining elements in their original order.

Definition

Class: AlternateLongest
Method: longestZigZag
Parameters: int[]
Returns: int
Method signature: int longestZigZag(int[] sequence)
(be sure your method is public)

Constraints

- sequence contains between 1 and 50 elements, inclusive.
- Each element of sequence is between 1 and 1000, inclusive.

Examples

0)

{ 1, 7, 4, 9, 2, 5 }
Returns: 6
The entire sequence is a zig-zag sequence.

1)

{ 1, 17, 5, 10, 13, 15, 10, 5, 16, 8 }
Returns: 7
There are several subsequences that achieve this length. One is 1,17,10,13,10,16,8.

2)

{ 44 }
Returns: 1
3)

{ 1, 2, 3, 4, 5, 6, 7, 8, 9 }
Returns: 2
4)

{ 70, 55, 13, 2, 99, 2, 80, 80, 80, 80, 100, 19, 7, 5, 5, 5, 1000, 32, 32 }
Returns: 8
5)

{ 374, 40, 854, 203, 203, 156, 362, 279, 812, 955,
600, 947, 978, 46, 100, 953, 670, 862, 568, 188,
67, 669, 810, 704, 52, 861, 49, 640, 370, 908,
477, 245, 413, 109, 659, 401, 483, 308, 609, 120,
249, 22, 176, 279, 23, 22, 617, 462, 459, 244 }
Returns: 36