DSA Mids Lab exam

Section B

May 18, 2022

Instructions for the exam

1. This is a 2 hour exam.
2. You will be submitting your problems on the OJ
3. All the data structures will be provided in a .c file. You can copy them into your code and use them as required. You can modify them as well. Please go through the template code before using it (at-least the function definitions).
4. Problems A and B are worth **100 points** each, and Problem C is worth **200** points.
5. Refer to the time limit and memory limit of each problem on OJ.

Problem A

Definition. Two words are said to be anagrams of each other if they can be obtained from each other by rearranging the letters of the words. For example *"see"*, *"ese"* and *"ees"* are anagrams of each other, but *"show"* and *"wash"* are not anagrams of each other.

Given two linked lists $ll1$, $ll2$ with character values and n elements each, you have to implement the function *int ans(LL* ll1,LL* ll2)*, which **returns 1 if the strings formed by these linked lists are anagrams of each other, and 0 otherwise.**

For this question, you will be provided the template code in file *a.c* which will contain the ADT for linked list. You can write additional methods to solve this problem, but ONLY inside the designated area specified by comments. **Any change in code outside designated area will lead to a straight 0 in this problem**.

Constraints

Number of test cases $1 \leq T \leq 10^3$
$1 \leq n \leq 10^3$

Input format

- First line contains and integer T, the number of test cases.
- For every test case the first line contains an integer n, the number elements in the linked lists.
- Then the next two lines contain n character strings each denoting the strings formed by the two linked lists. The strings contain lowercase Latin alphabets (a-z) ONLY.

Examples

Sample Testcase 1

Input

2
8
triangle
integral
5
house
mouse

Output

Yes
No

Explanation

The two words triangle and integral are anagrams of each other, so the function *int ans(LL* ll1,LL* ll2)* should return 1
The two words house and mouse are not anagrams of each other, so the function *int ans(LL* ll1,LL* ll2)* should return 0

Problem B

Definition. A balanced binary tree is a binary search tree where for each of its node, the height of its left child and right child differs by at most 1.
Or in simpler words, a balanced binary search tree of N nodes has height of $\log N$

Definition. The lowest common ancestor of two nodes u, v in a tree is the lowest node in the tree r such that u and v are in the subtree rooted at r.

Given a balanced binary search tree (BST) with N nodes, print the lowest common ancestor (LCA) of two given nodes in the BST. *(Note that the LCA of two nodes in a tree always exists)*

Constraints

$1 \leq N \leq 10^6$
$1 \leq Q \leq 10^6$
$-10^8 \leq$ node values $\leq 10^8$
All node values are distinct

Input format

- First line contains N and Q, the number of nodes in the balanced binary search tree and the number of queries.
- Then N lines follow that contain three integers; p,wc and val. p ($-1 \leq p < n$) denotes the parent of the current node (-1 if it is the root). If wc ($-1 \leq wc \leq 1$) is 0, then the current node is the left child of the parent, and if it is 1, then the current node is the right child of the parent (It is -1 when the node is the root). val ($-10^8 \leq val \leq 10^8$) is the value of the node. *(Note that p is zero-indexed)*
- No two nodes will have the same value
- The input is a valid balanced binary search tree.
- Next Q lines contain two integers, a ($-10^8 \leq a \leq 10^8$) and b ($-10^8 \leq b \leq 10^8$), the values of the nodes whose LCA you have to find. *(There will always exist a node with node value a, and same for b)*

In the binary tree template file, a function *BT create_BT(int **arr, int n)* is included. The template has the code for taking input of the specified format and creating a binary tree from the input. Use it to construct the binary tree.

Output format

Output Q lines, i^{th} of which has the answer for the i^{th} query, which is the value stored in the LCA of the queried nodes.

Example

Sample Test case 1

Input

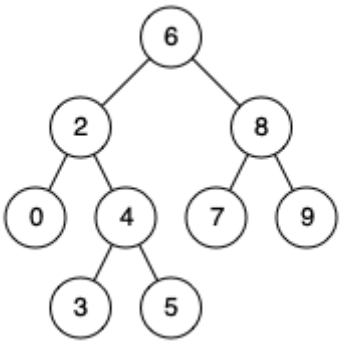
9 4
-1 -1 6
0 0 2
1 0 0
1 1 4
3 0 3
3 1 5
0 1 8
6 0 7
6 1 9
0 4
7 5
2 5
7 9

Output

2
6
2
8

Explanation

The tree in this test case is depicted as



Note the third query in particular, LCA of two nodes can be one of the nodes itself

Problem C

Definition. A binary string is a string whose characters can only be one of *'0'* or *'1'*. For example *"01001"* is a binary string, but *"0120"*, *"010a"* are not binary strings.

Definition. A string a is a prefix of string b if there exists some string c such that a concatenated with c forms b. For example *"abc"* is a prefix of *"abcde"*, but *"abd"* is not a prefix of *"abcd"*.

For a binary string s, you can split it into 2 parts s_1 and s_2 only if the ratio $\frac{\text{number of 1s}}{\text{number of 0s}}$ for both s_1 and s_2 is equal. *(If there are no 0s, then we consider the ratios $\frac{1}{0}$ and $\frac{2}{0}$ to be equal)*

Now for every prefix of a given binary string s you need to print the maximum number of parts the prefix string can be split into following previously mentioned constraint.

Constraints

Subtask 1 (20 points)

$1 \leq n \leq 10^3$

Subtask 2 (80 points)

$1 \leq n \leq 2 \times 10^5$

Input format

- First line contains an integer n which is the length of the string s.
- Second line contains an n character binary string s.

Output format

One line with n space separated numbers, where the i^{th} number denotes the maximum number of parts the prefix string consisting of the first i characters can be split into following the constraint mentioned in the problem statement.

Example

Sample Testcase 1

Input

3
001

Output

1 2 1

Explanation

There is no way to partition *"0"* or *"001"* into more than one block with equal ratios of numbers of 0s and 1s, while you can split *"00"* into *"0"* and *"0"*.

Sample Testcase 2

Input

6
000000

Output

1 2 3 4 5 6

Explanation

you can split each prefix of length i into i blocks *"0"*.