CS2710 - Programming and Data Structures Lab

Lab Session - 1

Aug 4, 2021

Instructions

- This lab is ungraded. You are expected to solve ALL the three problems on repl.it using C++.
- The questions are based on your training in programming in CS1111. So no additional inputs are needed, except the changes needed to switch from C to C++.
- You are expected to solve each problem on your own. If you need assistance, ask your TA, not your classmate.

Problems to be solved in lab

1. Given an array of integers, print the **number of *magic*** elements in the array. A *magic* element is an element in the array which is less than all the elements to its left in the array. The first element of the array, that is element indexed at 0 is not magic by definition. The array may also contain duplicates.

   Input format : First line of the input contains a single integer N (>= 2), denoting the size of the array. Next line contains N space separated integers like 10 21 34 23 53.

   Example :
   Input :
   5
   6 9 12 4 1
   Output :
   2
   Explanation :
   For the array {6,9,12,4,1}, the magic elements are 4 and 1, because they are less than every element to their left. Hence there are two numbers which are magic numbers and the output is 2.
   Constraints :
   $1 \leq N \leq 200$

Input	Output
10 30 100 120 -40 87 -30 5 20 -10 -50	2
15 450 348 301 247 232 211 172 95 0 -7 -23 -121 -234 -319 -415	14

2. In this question the input, output and goal are exactly the same as Question 1. The test cases are also exactly the same. However, you are expected to write code which is "efficient". While solving Question 1, you are free to use nested loops. However to earn the two marks in this question, you have to solve Question 1 **without** using nested loops. One idea for doing so is to maintain the **minimum** value for elements ranging from 0 to i and use the minimum value while processing element i+1. If you have already coded Question 1 without using nested loops then there is nothing to be done for you in Question 2.

3. Given an array of integers, print the number of sub arrays whose product is divisible by 5. A sub array is a contiguous part of the original array.

   Input format : First line of the input contains a single integer N (>= 2), denoting the size of the array. Next line contains N space separated integers like 10 21 34 23 53.

   Example :
   Input :
   3
   1 2 5
   Output :
   3
   Explanation : For the array {1,2,5}, the sub arrays are {1}, {1,2}, {1,2,5}, {2}, {2,5}, {5}. Of these sub arrays, the ones which have product divisible by 5 are {1,2,5}, {2,5} and {5}. Hence the output is 3.
   Constraints :
   $2 \leq N \leq 15$
   $0 \leq i < N$
   $-15 \leq arr[i] \leq 15$

Input	Output
10 -2 -1 7 6 3 -3 9 4 -4 5	10
12 13 -9 11 -7 4 -3 -1 7 1 5 -5 -13	32
15 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10	120

CS2710 - Programming and Data Structures Lab

Lab Session - 2

Aug 11, 2021

Instructions

- This lab is **graded**. You are expected to solve ALL the three problems on repl.it using C++.
- The questions are based on your training in programming in CS1111. So no additional inputs are needed, except the changes needed to switch from C to C++.
- You are expected to solve each problem on your own. If you need assistance, ask your TA, not your classmate.
- For all the problems the maximum size of the test case is 10000.
- For Q1, there are 6 public test cases. You get 0.5 mark for each test case. For Q2, TAs will evaluate 2 private test cases for 0.5 marks each and also verify that your code uses the ”efficient idea”. The 1 mark for Q2 is reserved for the efficient idea. Finally Q3 will have 6 public test cases worth 3 marks and 4 private test cases worth 2 marks. Note that any correct program is fine for Q3.

Problems to be solved in lab

1. [3] In a parade ground, army soldiers of Regiment-A are standing in a line in **increasing** order of their height (from left to right). The soldiers of Regiment-B are standing in **decreasing** order of their height again from left to right. For some reason these two regiments merge without disturbing the relative orders amongst themselves. That is, the soldiers in Regiment-B stand to the right of the last soldier in Regiment-A. The Lieutenant arrives and wants the tallest soldier to lead the parade. You need to write program to help the Lieutenant in finding the position (or index) of a soldier with maximum height.

   Input format: First line of the input contains a single integer N, denoting the number of the soldier. Next line contains N space separated integers like 10 21 34 23 53. The sequence of integers $a_0, a_1, \ldots, a_{N-1}$ satisfies the property that there exists an index $0 \leq p \leq N-1$ such that $a_0 < a_1 < a_2 < \ldots < a_p$ and $a_p > a_{p+1} > \ldots a_{n-1}$.

   Example :

   Input :
   7
   100 200 340 780 600 550 250
   Output :
   3
   Explanation :
   Soldier with maximum height (780) is standing at position 3.

   Note:
   Indexing starts with index zero i.e. 0.
   Your code should consider the possibility of Regiment-A or Regiment-B having no soldiers at all.

2. (2 points) In this question the input, output and goal are exactly the same as Question 1. The test cases are also exactly the same. However, you are expected to write code which is "better than" the one in the previous program. **Hint:** Think of binary search that you have studied. Can you use a similar idea?

3. (5 points) **Tug of war** is a sport that pits two teams against each other in a test of strength: teams pull on opposite ends of a rope, with the goal being to bring the rope a certain distance in one direction against the force of the opposing team's pull.
   Mr. Krishnamurti, a sports faculty, asked students of Class 10 to stand in a straight line. In order to conduct a fair game Mr. Krishnamurti has to divide students in such a way that the sum of weight of students on the left side is equal to sum of weight of students on the right side. Since the game requires a referee Mr. Krishnamurti has decided to select a student as Referee who is standing at a point of equilibrium i.e. sum of weight of student in left side is equal to sum of weight of student in right side. You need to help Mr. Krishnamurti in selecting a student as Referee.
   Write a program to return the position (or index) of the Referee if one exists. If no such split is possible, return -1.

   Note:
   The students cannot be reordered.
   Indexing starts with index zero i.e. 0.
   Weight of students can be negative.

   Input format : First line of the input contains a single integer N, denoting the number of students. Next line contains N space separated integers like 10 21 34 23 53.

Example 1:
Input :
7
6 2 5 2 4 3 6
Output :
3
Explanation :
Weight of students are given as : Wt []= 6, 2, 5, 2, 4, 3, 6
As, Wt[0] + Wt[1] + Wt[2] = Wt[4] + Wt[4] + Wt[6]
Referee position is : 3

Example 2:
Input :
7
-7 12 4 6 -4 3 0
Output :
2
Explanation :
Weight of students are given as : Wt []= -7, 12, 4, 6, -4, 3, 0
As, Wt[0] + Wt[1] = Wt[3] + Wt[4] + Wt[5] + Wt[6]
Referee position is : 2

Example 3:
Input :
3
1 2 3
Output :
-1
Explanation : No split is possible.

This part is for students to try and play around with their codes. This is not needed for the lab. You can use the following code for timing parts of your code. Note that you are not expected to output the time as a part of the output.

```
#include<iostream>
#include <chrono>
using namespace std::chrono;
using namespace std;

int main() {
    auto start = high_resolution_clock::now();

    for (int i=0; i<100000; i++) {
        // do nothing.
    }
    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(stop - start);
    cout << "Time taken by above code: "
         << duration.count() << " microseconds" << endl;
}
```

CS2710 - Programming and Data Structures Lab

Lab Session - 3

Aug 18, 2021

Instructions

- This lab is **graded**. You are expected to solve ALL the three problems on repl.it using C++.
- You are expected to solve each problem on your own. If you need assistance, ask your TA, not your classmate.
- For all the problems the maximum size of the test case is 10000.
- For each question there will be public and private test cases.

Problems to be solved in lab

1. (4 points) A credit card company has maintained a record of all it's customers credit status in the form of an N x N matrix P, where each of N rows denotes the remaining credit amount of a customer over past N months. A customer who has exceeded his credit amount in a particular month is denoted by a negative number. Moreover the manager has **sorted** the dataset in **non- decreasing** order **both row wise and column wise**. Help the manager to find the total number of times the credit limit is exceeded in the whole dataset.

   Input format:

   - First line of the input contains a single integer N, denoting the number of rows and columns.
   - Each of the next N lines contain N space separated integers denoting each element of the matrix.

   Input Constraints :

   - Maximum size of the matrix P will be 375 x 375
   - $-250000 <= P[i] <= 250000$

   Example :

   Input :

   ```
    5
   -215  -133  -65   39   150
   -129   -76  -29   67   192
   -121   -30  214  242   311
    -21   391  413  422   507
    215   567  714  842   911
   ```

   Output :
   9

   Explanation :
   In the matrix there are 9 persons who have exceeded their credit amount (negative valued entries).

   Note:
   There is a trivial $\Theta(N^2)$ algorithm for the problem. You are supposed to implement a "better algorithm" that is, an algorithm that runs in little $o(N^2)$ time. The trivial approach will not fetch any credit even if all test cases pass.

   Since there are multiple ways, we expect you to select an efficient approach that you can think of and briefly write down the idea of your algorithm as a comment in the code. This will be evaluated by the TA.

2. (4 points) Given N x N Matrix P and an of N positive integers as *array*, following operations are done on the matrix:

   Rotate 0-th row right side array[0] times.
   Rotate 1-st row right side array[1] times.
   Rotate 3-rd row right side array[2] times.
   .
   .
   Rotate $(N-1)$-th row right side array[$N-1$] times.

   Finally print the sum of diagonal elements.

   Important Note: In this problem, you are not allowed to modify the input matrix, neither are you allowed to store an auxilary matrix. The space complexity of your algorithm should be $O(1)$.

   Input format:

   - First line of input contains N.
   - Each of N lines contains$N$ spaced integers.
   - Last line of input contains N integers denoting the elements of the *array*.

   Input Constraints :

   - Maximum size of the matrix P will be 800 x 800
   - $-1000 <= P[i] <= 1000$
   - $0 <= arr[i] <= 1000$

   Example :
   Input :
   5
   1 2 3 4 5
   6 7 8 9 10
   11 12 13 14 15
   16 17 18 19 20
   21 22 23 24 25
   3 4 5 7 11

   Output :
   65

   Explanation :
   After rotating according to the given array, the matrix will look like the following:

   3 4 5 1 2
   7 8 9 10 6
   11 12 13 14 15
   19 20 16 17 18
   25 21 22 23 24

   Now summing the diagonals will result in : $3 + 8 + 13 + 17 + 24 = 65$

3. (2 points) Vivek and Abdul are friends. Abdul has a habit of collecting chocolate wrappers . In the backside of each wrapper an **integer number** is printed. One day Abdul is playing with these wrappers and arranged them in a square matrix P. Abdul feels bored and asks Vivek to rotate some portion of wrappers to **180** degrees by **excluding a few rows and columns**. Vivek is confused. So, you have to help Vivek to complete the task.

   Abdul will give integer M , Vivek has to **exclude first M rows, last M rows , first M columns, last M columns** and remaining rows and columns should be rotated 180 degrees.

   Input format :

   - First line of the input contains two space separated integers N and M.
   - Next N lines contains N space separated integers each, denoting the entries of the matrix P.

   Input Constraints :

   - Maximum size of the matrix P will be 350 x 350
   - $-1000 <= P[i] <= 1000$

   Example :
   Input :

   6 1

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

   Output :

1	2	3	4	5	6
7	29	28	27	26	12
13	23	22	21	20	18
19	17	16	15	14	24
25	11	10	9	8	30
31	32	33	34	35	36

   Explanation :
   Since, $M = 1$, the 1^{st} row, 1^{st} column, last row and last column has been excluded and rest of the matrix is rotated.

   Note:
   The grid lines are for explanation purpose, you have to just output the matrix normally.

   An example of rotation for your convenience :

   A = $\begin{matrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{matrix}$

After rotating A 90 degrees clock wise:

A1 =
$$\begin{matrix} 13 & 9 & 5 & 1 \\ 14 & 10 & 6 & 2 \\ 15 & 11 & 7 & 3 \\ 16 & 12 & 8 & 4 \end{matrix}$$

Again rotating A1 90 degrees clockwise:

$$\begin{matrix} 16 & 15 & 14 & 13 \\ 12 & 11 & 10 & 9 \\ 8 & 7 & 6 & 5 \\ 4 & 3 & 2 & 1 \end{matrix}$$

(This is the final 180 degree rotation of matrix A)

CS2710 - Programming and Data Structures Lab

Lab Session - 4

Aug 25, 2021

Instructions

- This lab is **graded**. You are expected to solve the two problems on repl.it using C++.
- You are expected to solve each problem on your own. If you need assistance, ask your TA, not your classmate.
- For all the problems the maximum size of the test case is 10000.
- The first problem must be submitted in lab latest by 4.45pm. The second problem can be submitted by 11.00pm today (Aug 25). Late submissions will not be evaluated.
- For each Q1 there are 4 public test cases each worth 0.5 marks. There are 4 private test cases each worth 0.5 marks. For Q2 there are 4 public test cases each worth 0.5 marks. You will fetch 1 mark for the correct insertToFront function. There are 3 private test cases each worth 1 mark.

Problems to be solved in lab

1. (4 points) This question is an extension of the linked list implementation of list ADT that you have studied in class. The only modification is that the list contains integer elements (as opposed to characters that we studied in class). For the first part (PartA) you are the implementer of some functions for the list class. For the second part (PartB) you are the *client* of the list class. Hence you cannot implement any member functions in PartB, but you are allowed to make use of the member functions that are provided or you the ones that you implemented in PartA.

   You are given the a template code with some functions implemented. You must use the same class definitions.

   PartA
   Assume that the elements in the list are numbered as $A_1, A_2, \ldots, A_n$. Here you are expected to implement two functions as follows.

   - `printk (int k)`: The function takes as input a positive integer `k` and prints the k-th element from the start in the list. You may assume that $1 \leq k \leq size$ where size denotes the number of elements in the list when `printk` is invoked.
   - `reorder (int k)`: The function takes as input a positive integer `k` and modifies the list such that the k-element from the start of the list is moved to the head of the list. The remaining elements continue to be in their relative positions. You may assume that $1 \leq k \leq size$ where size denotes the number of elements in the list when `reorder` is invoked.
     For example if the list contains elements `10, 9, 1, 20, 4, 13` and `reorder(4)` is invoked, then the list now contains elements `20, 10, 9, 1, 4, 13`.
     You must not make use of **"new"** and **"delete"** in the `reorder` function. That is, the reorder must be achieved using only pointer manipulations.

   PartB
   Recall that in this part you are the *client* of the list class. Hence you are not allowed to define any member functions or access any private members of the list class. You need to implement the following functions using the functions provided / implemented by you in PartA. Note that these functions are NOT supposed to be member functions of the list class.

   - `printList(List L1)` : This is the standard print function which prints the elements of the list L1 in the order $A_1, A_2, \ldots, A_n$.
   - `reverseList(List L1)` : This function reverses the elements of the list L1. If L1 contains `10, 9, 1, 20, 4, 13`, after a call to reverseList(·), the list contains `13, 4, 20, 1, 9, 10`.

   Input format:

   First line of input will be number of entries in the linked list denoted by N.

   Next line contains N space separated integers which are the elements of the linked list.

   Next line of input will be number of queries denoted by Q.

   There will be four types of queries:

   - P K, where K is an integer denoting the position
   - R K, where K is an Integer denoting the position
   - V, reverse the entire linked list
   - A, print all the elements in the linked list

Input Constraints :

- $1 <= N <= 1000$
- $0 <=$value of elements$<= 10^9$
- $4 <= Q <= 100$
- $1 <= k <= N$

Example :

Input :

5
6 7 8 9 10
4
R 4
P 3
V
A

Output :
7
10 8 7 6 9

Explanation :
Given linked list input : 6 7 8 9 10.
After performing R 4 that is, `reorder 4`, we have linked list as : 9 6 7 8 10
For P 3, that is, `print 3` we print the third element in the current list : 7
For V, that is reverse, we have linked list as : 10 8 7 6 9
Finally for A, that is, printall, we print : 10 8 7 6 9

Note:

- Expected space complexity : $O(1)$ (other than storing the linked list while reading input).
- Solutions that do not adhere to the constraints will not fetch credit.

You can submit the second problem by 11.00pm latest

2. (6 points) In this problem we are given two sequences S1 and S2 of integers sorted in decreasing order. The individual sequences contain distinct integers. In addition, we are given an integer value X. We are interested in finding the number of pairs of (a, b) such that $a \in$ S1 and $b \in$ S2 and $a + b = X$. For example if S1 = `3, 2, 1` and S2 = `5, 4, 3`, then for X=5, we have two pairs (1, 4) and (2, 3) which satisfy the criterion. Hence our answer is 2.

   There are several ways to solve the problem. However, you are expected to use lists to solve the problem. In particular, you are expected to maintain S1 and S2 as lists. However, note that in addition to the standard functions, the list interface provides an `insertToFront`. This function inserts the given element at the head of the current list. For instance if the list is containing `7, 10, 2`, then after insertToFront(15) we have the list `15, 7, 10, 2`.

   You need to implement the function insertToFront. Spend some time thinking how the function can help to solve the problem given to you.

   Note: It is expected that the two sequences will be maintained as lists. You are allowed to use O(1) auxilary storage apart from the lists stored. Solutions which store the sequences as arrays will not fetch any credit.

   Additional Note: It is expected that the running time of your program is O(N1+N2). However, you will fetch credit even if you have an inefficient implementation as far as the public test cases are concerned. The private test cases will be awarded credit only if the solution is O(N1+N2).

   Input format:

   - First line of input contains an integer $N1$ denoting the size of the first linked list.
   - Second line will contain $N1$ space separated integers denoting the elements of the first linked list.
   - Third line will contain an integer $N2$ denoting the size of the second linked list.
   - Fourth line will contain $N2$ space separated numerical values denoting the elements of the second linked list.
   - Last line will contain an integer X.

   Input Constraints :

   - $1 <= \text{N1, N2} <= 10^5$
   - $0 <=$ Elements of linked list $<= 10^9$
   - $0 <= \text{X} <= 2 * 10^9$

   Example :
   Input :
   3
   3 2 1
   3
   5 4 3
   5
   Output :
   2

   Explanation :
   Given S1 = `3, 2, 1` and S2 = `5, 4, 3` and X = 5. The number of all pairs whose sum is equal to 5 : (1,4), (2, 3) hence 2 pairs.

CS2710 - Programming and Data Structures Lab

Lab 6 - Practice

Sep 8, 2021

Instructions

- This lab is **not graded**. You are expected to solve the problem(s) on repl.it using C++.
- You are expected to solve each problem on your own. You can ask TAs for assistance.
- All problems must be submitted in lab latest by 4.30pm.
- There is only one question in this lab. All testcases are already made public.

---

1. Today's task is to create a doubly linked list and perform the following operations:
   i) Check for any loop in the list
   ii) Delete the loop node (The node to which the cycle pointer of some other node points)
   iii) After deletion, check whether the sequence of elements in the list form a palindrome.

   Each node in the list has 3 pointers namely previous, next and cycle. The previous pointer contains the address of the previous node and the next pointer contains the address of the next node in the sequence. The cycle pointer will point to any one of the nodes in the list thereby forming a cycle.

   The doubly linked list should be created using the following commands:
   1) add
   2) save
   3) loop

   Add command adds a node with given data to the end of the list.
   Save command saves the address of the node at tail.
   Loop command adds a pointer from tail node to the node whose address was saved previously.

   Note : The save command works if and only if there is atleast one node in the list. There should be some saved node for the loop command to work.

   Input format:

   - The first line is an integer n representing the number of commands.
   - The subsequent n lines represent n commands. The commands can take 3 values 1, 2 or 3. 1 corresponds to add, 2 to save and 3 to loop.
   - If the command is 1, it should be followed by an integer m denoting the number of nodes to be added and m integers representing the data d. If the command is 2 or 3, no additional data follows.

Output Format

The output consists of 3 lines. The first line consists of the node's data in the list after its creation and is separated by space.
The second line consists of the elements in the list after the deletion of loop nodes (if any) and is separated by space. NULL should be printed if the list is empty.
The third line is a string S that takes two values. YES if the sequence is a palindrome and NO otherwise. If the list is empty, then it should be considered as no palindromic sequence.

Input Constraints :

- $1 <= n <= 10^6$
- $0 <= m <= 10^6$
- $0 <= d <= 10^3$

Example :

Input 1:
5
1 3 88 7 6
2
1 1 7
3
1 1 88

Output 1:
88 7 6 7 88
88 7 7 88
YES

Explanation 1:
First operation adds the three nodes in the list. List = [88, 7, 6].
Second operation saves the address of tail (that is 6).
Third operation adds one more node at the end. List = [88, 7, 6, 7].
Fourth operation makes the loop variable of tail (that is 7) to point to saved address (having value 6).
Fifth operation adds 88 to the end of list. List = [88, 7, 6, 7, 88]. Moreover, remember that the fourth node has the loop with third node as well.
Now after the operations, we give output in desired format. First, the list:
88 7 6 7 88
After this, we delete the node pointed by loop variable:
88 7 7 88
Since it is a palindrome, we print the third line:
YES

CS2710 - Programming and Data Structures Lab

Lab Session - 7

Sept 15, 2021

Instructions

- This lab is **graded**. You are expected to solve ALL the three problems on repl.it using C++.
- You are expected to solve each problem on your own. If you need assistance, ask your TA, not your classmate.
- For each question there will be public and private test cases.
- For this lab you are allowed to use STL stacks and queues if required. To see the uses of STL stack refer to the class code given for infix to postfix. You are not allowed to use vectors.

Problems to be solved in lab

1. (4 points) You are given the heights of n towers arranged from left to right in a straight line. The input is given to you as n integers and you are expected to store them in an array called `T`. You are expected to answer two kind of queries.

    - Query 1: Which will contain a character L and a non negative integer k, for this query you need to print the largest index j of a tower in T such that j $<$ k and height of tower j is less than height of tower k. In case of no such tower exist print -1. You may assume that $0 \leq k \leq n-1$ where n denotes the number of towers.
    - Query 2: Which will contain a character R and a non negative integer k, for this query you need to print the smallest index j of a tower in T such that j $>$ k and height of tower j is less than height of tower k. In case of no such tower exist print n. You may assume that $0 \leq k \leq n-1$ where n denotes the number of towers.

    Input format:
    First line of input will be the number of towers denoted by n.
    Next line contains n space separated integers which are the heights of the towers.
    Next line of input will be the number of queries denoted by Q.
    There will be two types of queries as mentioned above:

    - L k,where k is an integer denoting the position of the tower, and you need to print whatever is asked in Query 1.
    - R k, where k is an integer denoting the position of the tower, and you need to print whatever is asked in Query 2.

Note:

- 0 based indexing is used for the array T and all the queries as well.
- You need to answer each query in $O(1)$ time, while for prepossessing you can use extra space of $O(n)$ and have time complexity of $O(n)$.

Input Constraints :

- $1 <= n <= 100000$
- $0 <=$height of tower $<= 10000$
- $1 <= Q <= 10000$
- $1 <= k <= n - 1$

Example :

Input :

5
13, 9, 11, 7, 8
5
L 3
R 3
L 4
R 1
R 4

Output :
-1
5
3
3
5

Explanation :
L 3 means we need to find first tower on left of tower 3 such that its height is less than that of the tower 3. Height of the tower at position 3 is 7. There are no towers on the left of this tower having height less than 7 so print -1.

R 3 means we need to find first tower on right of tower 3 such that its height is less than that of the tower 3.Height of the tower at position 3 is 7. There are no towers on the right of this tower having height less than 7 so print 5(number of towers).

L 4 means we need to find first tower on left of tower 4 such that its height is less than that of the tower 4. Height of the tower at position 4 is 8. Tower at position 3 is having height 7 which is less than 8 so print 3.

R 1 means we need to find first tower on right of tower 1 such that its height is less than that of the tower 1. Height of the tower at position 1 is 9. Tower at position 2 is having height 7 which is less than

9 so print 2.

R 4 means we need to find first tower on right of tower 4 such that its height is less than that of the tower 4. Tower 4 is the last tower, we cannot go further right. So print 5(number of towers).

2. (2 points) This is an extension of the previous question. Here also you are given the heights of n towers arranged from left to right in a straight line. The input is given to you as n integers and you are expected to store them in an array called T. Apart from that you will also be provided with width of the tower denoted by W. All towers have same width. You will be given an integer i between 0 to n-1 as query and you need to print the maximum area of rectangle that can be formed by taking completely the tower at index i in the rectangle and few consecutive towers on the left of it and few consecutive towers on the right of it. Read example for better explanation.

Input format:

First line of input will be two space separated integers n and W denoting the number of towers and their width respectively.
Next line contains n space separated integers which are the heights of the towers.
Next line of input will be the number of queries denoted by Q.
For each query you will be given an integer i where $0 \leq i \leq n-1$.
For each query you need to print the area of the rectangle that can be formed as mentioned above.
Note:

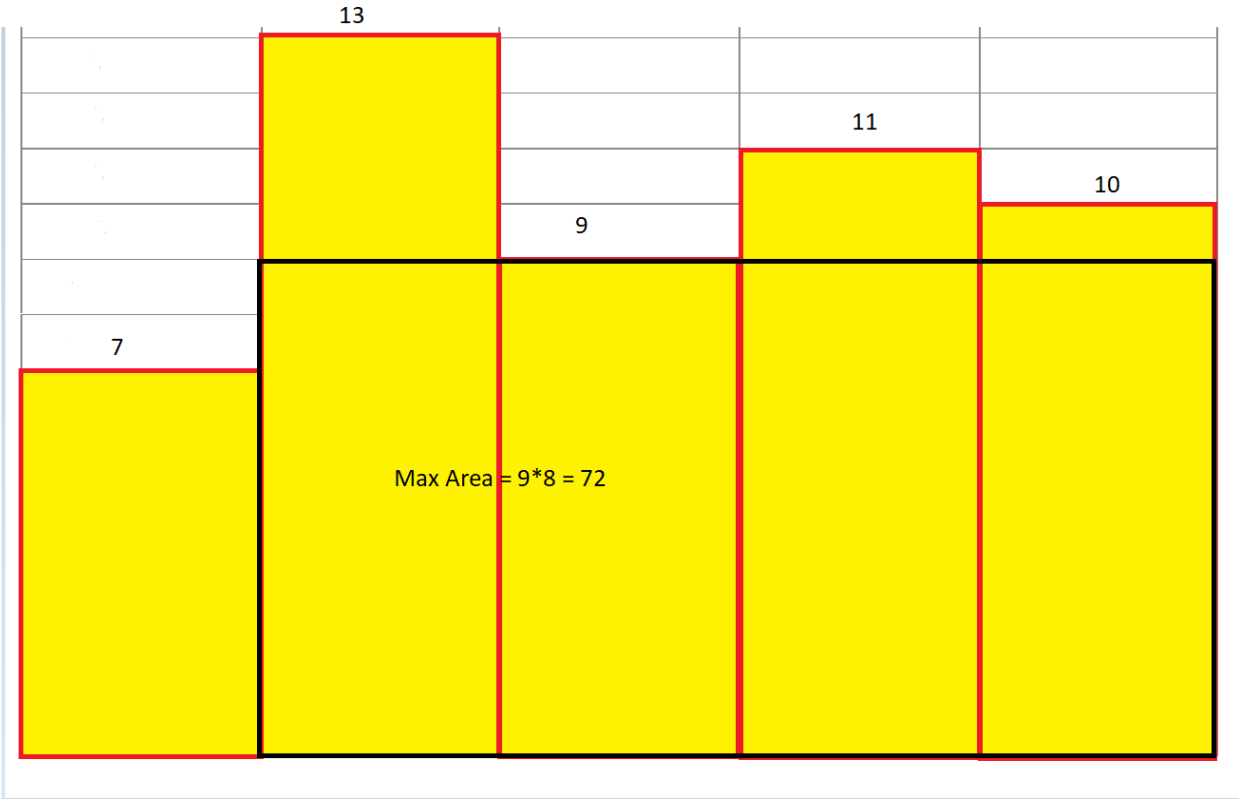
- 0 based indexing is used for the array T and all the queries as well.
- Each Query need to be answered in $O(1)$, while you may use a space of $O(n)$ and time complexity of $O(n)$ for preprocessing.
- Hint: Some computation from the first question can be used as it is in the second question.

Input Constraints :

- 1 <= n <= 100000
- 0 <= height of tower <= 10000

Example :
Input :
5, 2
7, 13, 9, 11, 10
5
2
0
4
1
3
Output :
72
70
40
26
22

Explanation :



First query is 2, so we look for the towers on the left of tower 2 and on the right of tower 2 such that they form a set of consecutive towers and we can form a rectangle of height equal to height of tower 2 and its area is maximum.
It is clear from the rectangle drawn in black color that we can include the tower from index 1 to index 4, and they will form a rectangle with height equal to 9 and width equal to 8 and hence the area will be $9 * 8 = 72$.
Second query is 0, so we look for the towers on the left of tower 0 and on the right of tower zero such that they form a set of consecutive towers and we can form a rectangle of height equal to height of the tower 0 and its area is maximum.
Clearly there is no tower on the left of tower 0, but we can include all the towers on the right of tower 0, hence maximum area possible by including tower 0 is $7 * 10 = 70$.
Similarly when the query is 4, only towers at index 3 and index 4 can be considered and width of the rectangle will be $2 * 2 = 4$ and height will be 10 and hence the area will be 40.
Similarly when the query is 1, only tower at index 1 can be included and the area will be $13 * 2 = 26$.
Similarly when the query is 3, area of the rectangle will be $11 * 2 = 22$.

3. (4 points) In this problem you are given a singly linked list containing the details of each student. Each student has a name and a roll number. Insertion and deletion operations occur at the front of the list. You need to write a method to print all the student details stored in the linked list in **backward order**. You are **not allowed to use recursion** and any use of recursion would fetch zero marks. Also use of any array is prohibited.

   You are given a template code with some methods implemented. You only need to complete the method named **printBackward** in LinkedList. You must use the same class definitions.

   Input format:

   First line of input contains an integer N denoting the number of queries.

   Next N lines denotes a query Q.

   There will be four types of queries:

   - *I Name Roll* : *I* indicates the insertion operation, *Name* is a string denoting name of the student and *Roll* is an integer denoting roll of the student.
   - *D* : Deletes first element of the linked list.
   - *F* : Prints the linked list forward.
   - *B* : Prints the linked list backward.

   Input Constraints :

   - $1 <= N <= 1000$

   Expected Complexity :

   - Time : $O(N)$
   - Space : $O(N)$

   Example :

   Input :

   7
   I Mohan 13
   I Rajdip 4
   I Aman 29
   B
   D
   B
   F

   Output :

   | Mohon 13 | Rajdip 4 | Aman 29
   | Mohan 13 | Rajdip 4
   | Rajdip 4 | Mohan 13

CS2710 - Programming and Data Structures Lab

Lab Session - 8

Sep 22, 2021

Instructions

- This lab is **graded**. You are expected to solve the problem on repl.it using C++.
- You are expected to solve the problem on your own. If you need assistance, ask your TA, not your classmate.
- You are not allowed to use arrays to store nodes, but you can use queues in your program for level order traversal only.
- **Deadline for submission: 11:00pm, 23-Sept-21**

---

1. (10 points) This problem is a problem on Binary trees. Note that it is NOT a binary search tree. You are expected to perform different type of operations on a binary tree as shown below:-

   - **i num-to-insert:** With this command *i* you have to insert the number in the tree if it is not present already. Insertion has to be done in level-order from left to right. See example given below.
   - **a num1 num2:** With this command you have to find the immediate common ancestor of given num1 and num2. Additionally, the ancestor has to be replaced with the last node in the tree followed by the deletion of the last node in the tree. Note that the last node is the last node in the level-order traversal of the tree.
   - **p:** With this command you have to print the tree in PRE-ORDER TRAVERSAL

   Note : Use the defined structure Node to build the tree.

```
struct Node {
    long long int data; // Will store the value of node
    int level; // level at which this node is present. root node is a level 0.
    struct Node *left; // To store the link to left child
    struct Node *right; // To store the link to right child
    struct Node *parent; // To store the link to parent
};
```

   Notes: You may find these useful.

   - The Node data structure contains a member called `level`. Make appropriate use of the member for various operations that you need to implement.
   - It is helpful to write a function `isPresent` which takes as input the integer data item and any other arguments as needed and returns the pointer to the node in the tree if the data is already present, else returns NULL.
   - For inserting a new item, one needs to find the correct place where insertion should happen. Use level-order traversal for the same. You are free to refer to the level-order travsersal code for the BST (uploaded on the class homepage).

   Input format:

   - The first line of input contains an integer N representing the number of commands to perform. The next N lines of the input consist of commands in form of (i num-to-insert) or (a num1 num2) or (p).

   Input Constraints :

   - $1 <= N <= 10^5$
   - $1 <= num1, num2, num - to - insert <= 10^5$

   Output format:

   - **Insert(i num)** -
     If element gets inserted then print **INSERTED** else **if number is already present then print NOT INSERTED**.
   - **Ancestor(a m n)** -
     If the command is successful then print **DELETED ANCESTOR : ancestor-data** else print **NODES NOT FOUND**. For this query, m can be equal to n and further if number($m == n$) is present in the tree then ancestor will be m or n itself and you have to proceed accordingly. Another case that one may encounter is if m is the ancestor of n then m will be considered as the ancestor.

- **Print(p) -**
  If tree has data then print the values of the tree in the **PRE-ORDER TRAVERSAL** else print **TREE IS EMPTY**.

Example :

Input 1:
8
p
i 2
i 5
i 4
a 2 3
i 2
a 5 4
p

Output 1:
TREE IS EMPTY
INSERTED
INSERTED
INSERTED
NODES NOT FOUND
NOT INSERTED
DELETED ANCESTOR : 2
4 5

Explanation 1:
The first command is p, but the tree is empty. The second command inserts the value 2 into the tree. The third command inserts 5 into the tree. The fourth command inserts 4 into the tree. In the fifth command we are supposed to find the anscestor of 2 and 3 but since 3 is not present, the command is not successful. In the sixth command, we are supposed to insert 2, however it is not inserted since it is already present in the tree. In the seventh command, we are supposed to find the anscestor of the nodes 5 and 4. The anscestor is 2. The data in this node is to be replaced with the data in the last node (in level order traversal) which is 4 and then the node containing 4 is to be deleted. In the eighth command the entries of the tree are to be printed in pre-order traversal.

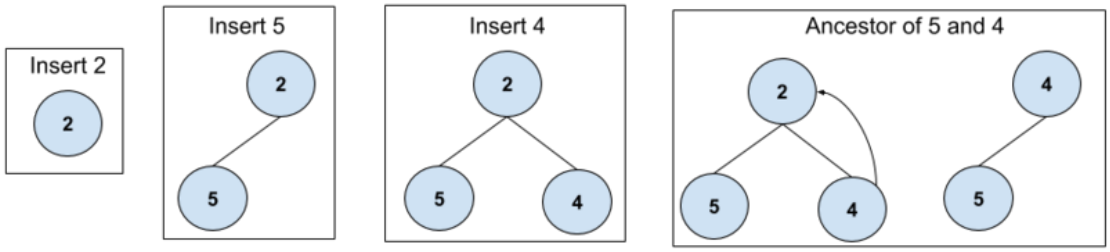


Figure 1:

Input 2:
10
p

i 2
i 10
i 8
a 2 8
p
a 10 8
a 10 10
p
a 8 10

Output 2:
TREE IS EMPTY
INSERTED
INSERTED
INSERTED
DELETED ANCESTOR : 2
8 10
DELETED ANCESTOR : 8
DELETED ANCESTOR : 10
TREE IS EMPTY
NODES NOT FOUND

CS2710 - Programming and Data Structures Lab

Lab Session - 9

Sep 29, 2021

Instructions

- This lab is **graded**. You are expected to solve BOTH the problems on repl.it using C++.
- You are expected to solve each problem on your own. If you need assistance, ask your TA, not your classmate.
- For all the problems the maximum size of the test case is 10000.
- For each question there will be public and private test cases.

---

1. (6 points) This question is an extension of the BST implementation that you have studied in class. You are supposed to support two queries `i` : insert and `s` : search.

   - To support the `i` query you are expected to implement the following **iterative** function :
     `void insertIterative (Datatype val)`: The function takes as input an element `val` of the type Datatype and inserts it into the BST. See template for more information about Datatype. Throughout the sequence of queries the input will contain unique keys for insert.

   - To support the `s` query you are expected to implement the following function :
     `Datatype searchLessthanOrEqualTo (Datatype k)`: The function takes as input an element `k` of the type Datatype and returns the largest element in the BST such that it is less than or equal to `k`. If not found, return -1.

   Input format:

   First line of input will contain an integer Q denoting the number of queries.

   Following this there will be Q lines each containing one of the following :

   `i` val : This query inserts val into the current BST.

   `s` k : For this query invoke the function `searchLessthanOrEqualTo (k)`.

   Output format:

   For the `i` query there is no output expected.

   For the `s` query the value returned by the query needs to be printed on the screen.

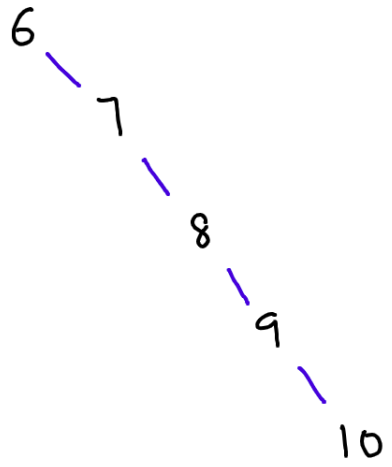
   Input Constraints :

   - $1 <= Q <= 1000$
   - $0 <=$val$<= 10^4$
   - $1 <= k <= 10^4$

   Example :
   Input :
   8
   i 6
   i 7
   i 8
   i 9
   i 10
   s 5
   i 4
   s 5
   Output :

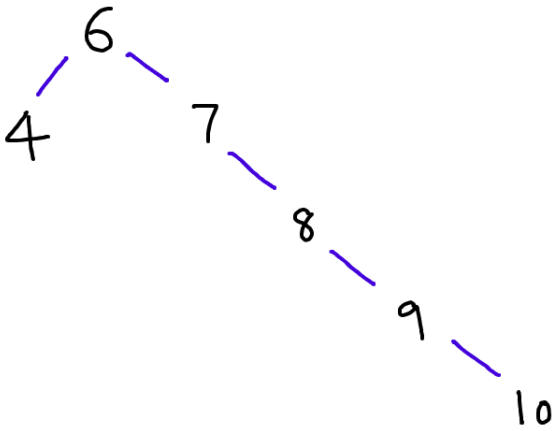
-1
4

Explanation :
After the first five insert queries, the BST will look like the following :



Hence, the result of ”s 5” is -1.

After ”i 4” the BST will look like the following :



Hence, the result of ”s 5” is 4.

2. (4 points) Given the preorder traversal of a BST as an input array, implement a recursive function to compute the height of the BST without explicitly constructing the BST.

Input format:

First line of input will contain an integer N denoting the number of keys in the tree.

Second line will contain N space separated values denoting the preorder traversal of a BST.

Output format:

Print the height of the BST.

Input Constraints :

- $1 <= N <= 1000$
- $0 <=$value of elements$<= 10^4$

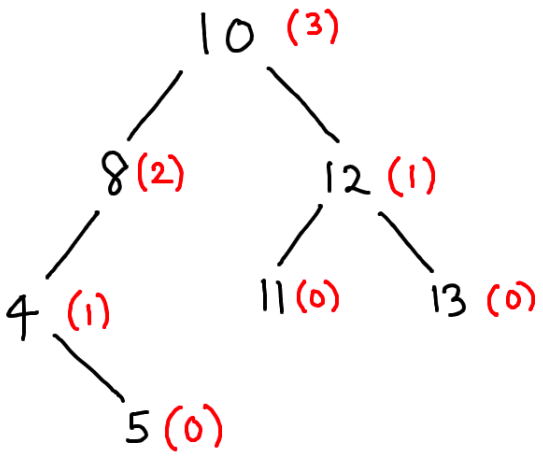
NOTE :

- You are not allowed to store the inorder traversal of the given BST.
- Expected time and space complexity : O(N), O(1) respectively.

Example :
Input :
7
10 8 4 5 12 11 13
Output :
3

Explanation :
Given the preorder traversal 10 8 4 5 12 11 13. Hence, the height is 3.

In the below figure, the numbers in red denote the height of each node.


10 (3)
8 (2)
12 (1)
4 (1)
11 (0)
13 (0)
5 (0)

CS2710 - Programming and Data Structures Lab

Lab Session - 12

Oct 22, 2021

Instructions

- This lab is **graded**. You are expected to solve the problems on repl.it using C++.
- You are expected to solve each problem on your own. If you need assistance, ask your TA, not your classmate.
- There will be public and private test cases.
- **Deadline for submission: 11:00 pm, 26-Oct-2021**

Problem

1. (8 points) In this problem you are required to implement a binary search tree with the following properties:

   - Every node of the tree will have **at most 2 children**.
   - The left subtree of a node contains nodes with keys **greater** than node's key.
   - The right subtree of a node contains nodes with keys **less** than node's key.
   - For every node, the difference between the height of the left and right subtrees **is at most one**.

   You are expected to answer two types of queries :

   - *I key*: Insert the key into the tree.
   - *S key*: Search for the key into the tree. If found print FOUND else print NOT FOUND
   - *P* : Print the preorder traversal of the tree.

   Input format:

   First line of input will be the number of queries denoted by Q.
   Next Q lines denotes any of the query mentioned above.

   Input Constraints :

   - $1 <= Q <= 1000$
   - $1 <= key <= 1000000$
     All the keys are unique

Example :

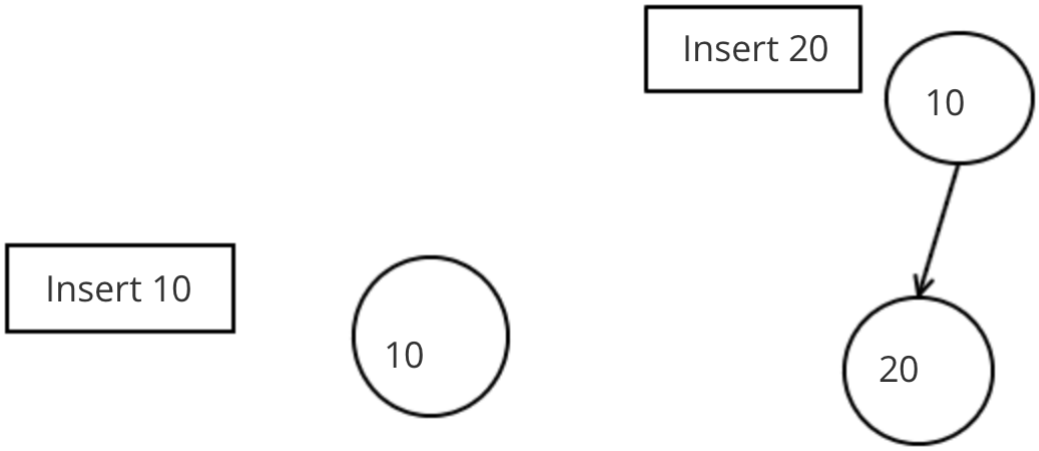
Input :

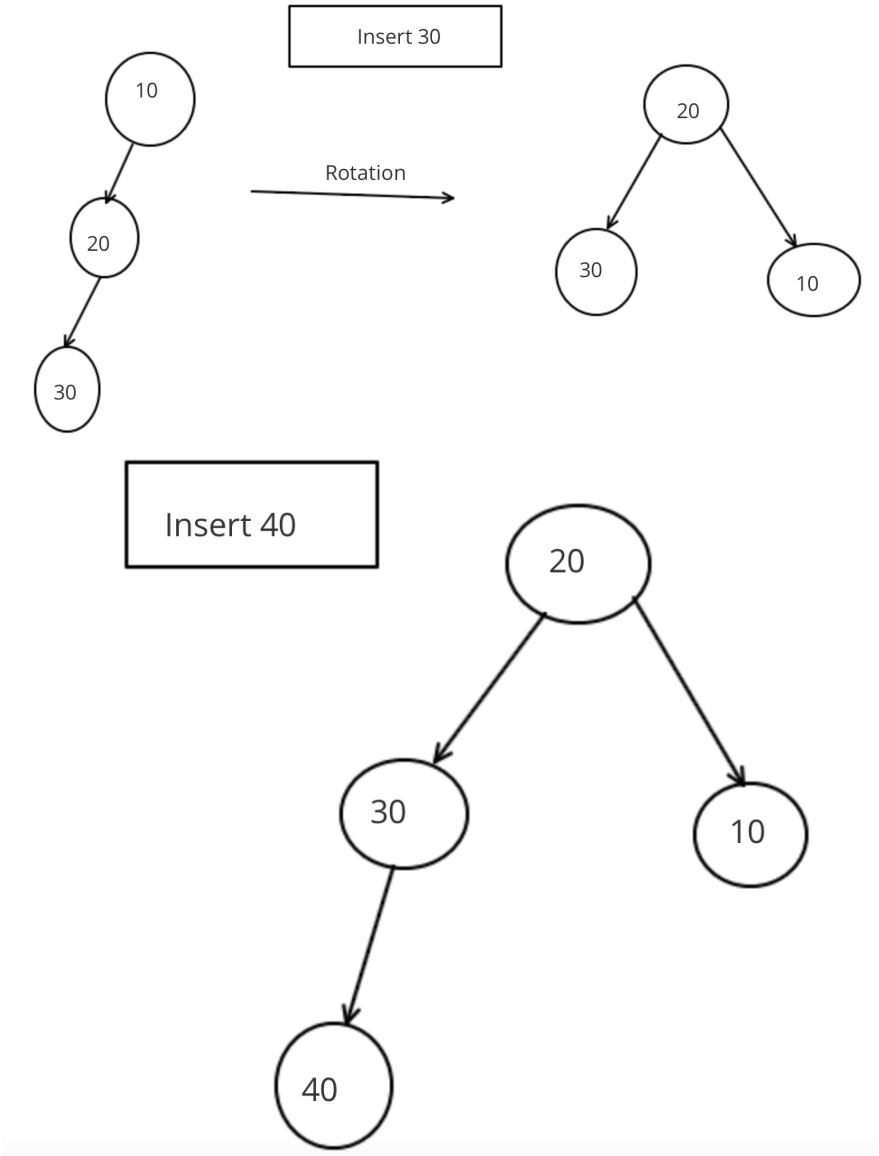
9
I 10
I 20
I 30
I 40
I 50
I 25
P
S 10
S 23

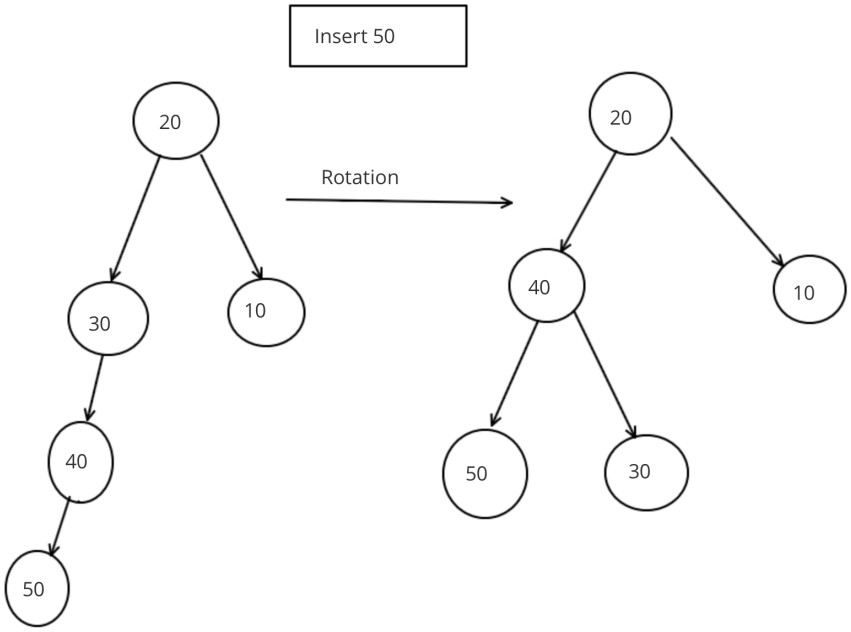
Output :

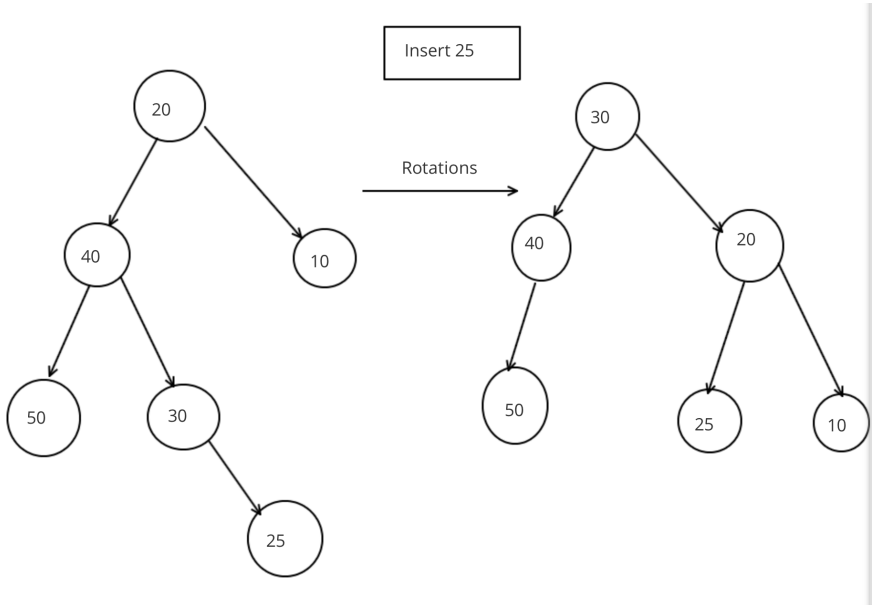
30 40 50 20 25 10
FOUND
NOT FOUND

Explanation :




Insert 30
10
20
30
Rotation
20
30
10
Insert 40
20
30
10
40


Insert 50
20
30
10
40
50
Rotation
20
40
10
50
30


Insert 25
20
40
10
50
30
25
Rotations
30
40
20
50
25
10

CS2710 - Programming and Data Structures Lab

Lab 13

Aug 27, 2021

Instructions

- This lab is **graded**. You are expected to solve ALL the two problems on repl.it using C++.
- You are expected to solve each problem on your own. If you need assistance, ask your TA, not your classmate.
- For each question there will be 2 public and 3 private test cases. Each test case carries 1 mark.

Problems to be solved in lab

1. (5 points) A magic number is defined as a positive integer which can be written in the form of $3^i \times 5^j \times 7^k$ where $i \geq 0$, $j \geq 0$, $k \geq 0$. Given an integer n, you need to print first n magic numbers.
   Example :

   Input :
   6
   Output :
   1 3 5 7 9 15

   Explanation :
   First 6 magic numbers are as follows:

   - $1 = 3^0 \times 5^0 \times 7^0$
   - $3 = 3^1 \times 5^0 \times 7^0$
   - $5 = 3^0 \times 5^1 \times 7^0$
   - $7 = 3^0 \times 5^0 \times 7^1$
   - $9 = 3^2 \times 5^0 \times 7^0$
   - $15 = 3^1 \times 5^1 \times 7^0$

   Note:

   - The smallest magic number is 1.
   - Any magic number can be created using a smaller magic number by an appropriate operation.
   - Your code must maintain a min heap of distinct magic numbers and process them till you compute the n^{th} magic number.

- You can use the the `priority_queue` from stl to get the functionality of min heap in c++.
- A min priority_queue for data type "t" can be declared as `priority_queue<t, vector<t>, greater<t>>minH`.
- For example if we want to store data of type `int` then we will declare the priority_queue in the following way.
- `priority_queue<int, vector<int>, greater<int>>minH`
- `minH.push(x)` will push an element x in the priority_queue (minH).
- `minH.top()` will return the smallest element present in minH.
- `minH.pop()` will delete the smallest element from minH.

Expected Time Complexity: $O(n \log n)$
Expected Space Complexity: $O(n)$
Input format:

- First line of input contains an integer n denoting the number of magic numbers to be printed.

Input Constraints :

- $n \leq 10^3$

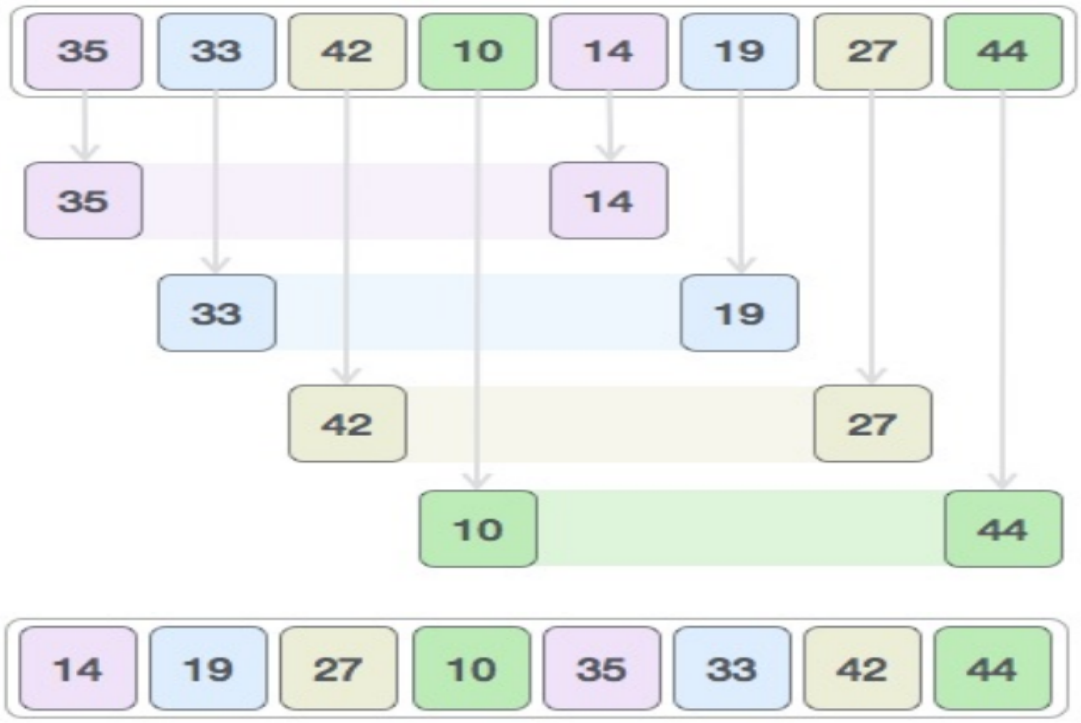
Example :
Input :
15
Output :
1 3 5 7 9 15 21 25 27 35 45 49 63 75 81

Explanation :
You can check that each number in the output has only 3, 5, 7 as their prime factors and hence can be written in the form of $3^i \times 5^j \times 7^k$ and these are the smallest 15 such number.

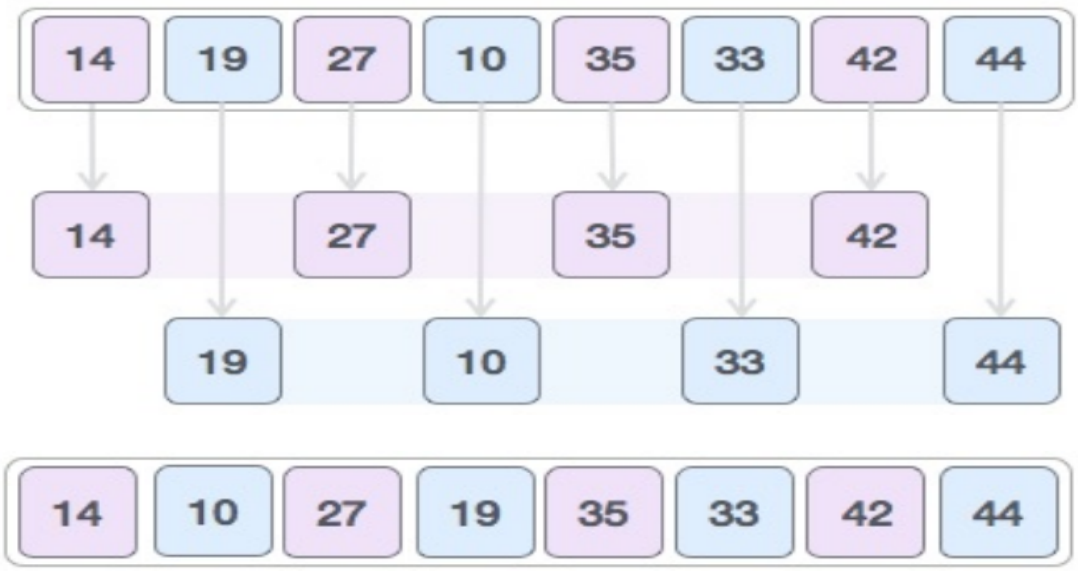
2. (5 points) In this problem you are expected to implement a new sorting algorithm and insertion sort algorithm which sorts an array of integers in non decreasing order that may contain duplicates using the template provided to you. You must not delete or comment any line of the template but you are free to add your code at the appropriate places.The new sorting algorithm works as follows.
The approach is similar to insertion sort except that only the elements separated by the interval k are compared to obtain a k-sorted array. An array is said to be k-sorted if starting anywhere, taking every k^{th} element produces a sorted list.
Example of 3-sorted array:
3 2 1 6 5 4 9 8 7
If we pick every 3rd element starting from
0^{th} index : 3 6 9
1^{st} index : 2 5 8
2^{nd} index : 1 4 7
3^{rd} index : 6 9
4^{th} index : 5 8
5^{th} index : 4 7
Notice that array is not sorted but each of the subarray is sorted.
Start with interval $k = \lfloor \frac{n}{2} \rfloor$ and keep reducing interval by half till interval becomes 1.

Example of how the new sorting algo works:

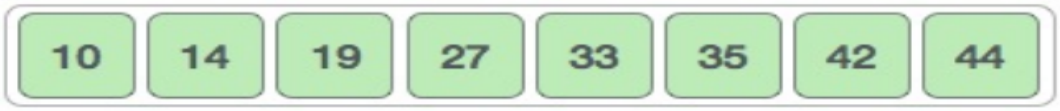
Interval = 4



Interval = 2



Interval = 1



Expected Time Complexity: $O(n^2)$
Expected Space Complexity: $O(1)$

Input format:

- First line of input contains an integer n denoting the number of integers in array.
- Second line contains n space separated integers.

Output format:
Print n space separated integers in sorted order corresponding to intervals starting from $\lfloor \frac{n}{2} \rfloor$ to 1 and reducing the interval by half every time.
Also at last print n space separated integers after running insertion sort on the input.
Input Constraints :

- $n \leq 10^3$
- $a[i] \leq 3 \times 10^3$

Example :
Input :
15
45 81 75 5 1 9 63 25 49 35 15 21 27 7 3
Output :
3 49 35 5 1 9 7 25 81 75 15 21 27 63 45
3 1 9 5 15 21 7 25 35 27 49 45 75 63 81
1 3 5 7 9 15 21 25 27 35 45 49 63 75 81
1 3 5 7 9 15 21 25 27 35 45 49 63 75 81

CS2710 - Programming and Data Structures Lab

Lab 14

Nov 3, 2021

Instructions

- This lab is **a graded lab %**. You are expected to solve the problem on repl.it using C++.
- You are expected to solve each problem on your own. Ask TA for assistance not your friends.
- All problems must be submitted in lab latest by 11 PM. Late submissions will not be evaluated.
- There is a single questions in this lab. Total of 10 test cases - 7 private and 3 public. Each test case will fetch 1 marks.

---

1. (10 points) In this lab we will look into different techniques of hashing. You are expected to implement a few different variations of a hash table for **strings** using different hash functions and different collision resolving scheme.

   Hash functions to use:
   You are expected to implement these two hash functions.

   - Hash 1 : Sum of ASCII values of all characters in string.
   - Hash 2 : Here we left shift the running hash by 5 bits before adding the ASCII values (done before adding each character, iterating from left to right in string).

   Working of the above 2 hash functions:
   Let the string be "*abc*", and the table size (m) = 101 for both tables. Observe how the two hash functions are working where the second one is only a slight modification to the first hash function.

   - Hash 1 : sum of ascii values of 'a', 'b', 'c', is (97+98+99) % (101) = 294 % 101 = 92
   - Hash 2 : Here we iterate from left to right in the string, the hashed index will be (starting with hash = 0). For safety, We also take a mod with m at each iteration-
     - left shift by 5 bits ($hash = (00000000000000000000000000000000)_2 = (0)_{10}$) and add ascii value of 'a', hash = (0+97) % 101 = 97.
     - left shift by 5 bits ($hash = (00000000000000000000110000100000)_2 = (3104)_{10}$) and add ascii value of 'b', hash = (3104+98) % 101 = 71.
     - left shift by 5 bits ($hash = (00000000000000000100011100000)_2 = (2272)_{10}$) and add ascii value of 'c', hash = (2272+99) %101 = 48.

     Hence the hashed index of the string using H2 is 48.

   Note: If we do the shift after addition, the hash value will always have its last 5 bits as 0, which will result in more chances of collision.

   Collision resolution techniques :

   - Chaining - Here each position in table points to a list of strings.
   - Linear probing - The return value of hash function determines the first possible starting point suitable for the string.

   In this lab, a total 4 hash-tables have to be designed

   - Chaining with Hash 1.
   - Chaining with Hash 2.
   - Linear probing with Hash 1.
   - Linear probing with Hash 2.

   We want to do only two types of operations in our hash tables :

   - **insertion** - when a string is entered, print the number of collisions for each of the 4 cases in the given order (opcode i).
   - **search** - print YES if string is present or NO if absent, followed by the number of probes done (opcode s).

Note: In chaining assume it is a vector of strings instead of linked list, so that you can use the $vector.push\_back()$ operation for efficiency. The number of collisions during insertion is defined as the size of the vector before inserting, as we are inserting to the end of vector.

Our goal is to compare the performance of the four hash tables, but since chaining uses additional memory whereas linear probing uses a single table to fit all entries, having them of the same size is not fair comparison. Hence, we use different size of tables using chaining and tables using linear probing.

The approximate value we are using is $m1 \approx n/2$ for chaining and $m2 \approx 2 * n$ for linear probing, where n is the number of entries we want to store.

Input format:

- Size of chaining hash table (m1), size of linear probing hash table(m2). You do not need to worry about choice of these values, tables won't run out of space.
- Integer Q, denoting the number of queries
- Next Q lines each have a character followed by a string denoting the opcode and the query string.

All input strings are distinct.

Output format: For each query, print 4 lines according to the operation (in the given order of hash tables), followed by a new-line. See example below for reference.

Input Constraints :

- $1 <= Q <= 10000$
- $2 <= m1 <= m2 < 10000$
- $string.length < 25$

Example:

Input 1:

3 7
5
i abcd
i dcba
i fff
s bce
s dcba

Output 1:

0
0
0
0

1
0
1
0

0
0
0
0

NO 2
NO 1
NO 0
NO 1

YES 1
YES 0
YES 1
YES 0

Explanation for example:

- The first insertion (abcd) is done in an initially empty table, all four tables have 0 collisions.
- The second insertion (dcba) has same ascii sum as the previous string (abcd), hence the tables using hash function 1 face collision.
- The third insertion (fff) does not have same hash value as the previous strings. Also suitable table places are empty, hence 0 collisions.
- Fourth query is a search query (bce), the string is not present, so a NO is printed followed by the number of probes -1, (i.e. the extra comparisions made) each table.
- Fifth query is a search query (dcba), string is present, so a YES is printed followed by the number of positions searched for each table.

Note : For search queries, all 4 tables must print ”YES” or all must say ”NO”. For search queries, we are printing the number of additional positions we had to look at before we can definitely say whether the required string is present or not. For example - if the entry at hash index obtained for a string is empty in the table, we output ”NO 0”, since 0 additional slots were looked at.

CS2710 - Programming and Data Structures Lab

Lab Session - 15

Nov 10, 2021

Instructions

- This lab is **graded**. You are expected to solve the problems on repl.it using C++.
- You are expected to solve each problem on your own. If you need assistance, ask your TA, not your classmate.
- There will be public and private test cases.
- **Deadline for submission: 11:00 pm, 11-Nov-2021**

Problem

In this problem we are given N cities. A pair of cities may be connected by a road which is two way (an undirected edge) with a cost associated with the road. The road costs can take only one of two values w_1 or w_2. The road costs will be used only later.

In addition, a pair of cities may have a bad road between them. This is indicated by a red edge in the examples. If a road is not bad, we call it a good road.

Our goal is to select a subset of cities from the input such that the following conditions hold:

- If a city is selected then every city to which it has a direct connection via a good road must be selected.
- If k cities are selected, it must be possible to reach from any one of the selected city to another selected city. It is acceptable to reach from one city to another via an intermediate city using good roads.
- Amongst the selected cities there should not be any pair which have a bad road between them (even if there is an alternate good path via another city).
- The collection of cities selected should be as large as possible.

Note that there may not exist any subset of cities that satisfy the given conditions. Hence the solution may be an empty set. If there exists multiple non-empty subsets (all of the same size), then our goal is to select the one where we need to pay the minimum cost. The cost associated with a set of cities is the minimum total road cost needed to connect these cities. (This cannot contain a cycle. Why?)

1. **Input format:**

   The first line of input contains an integer N which denotes the number of cities.

   The second line contains an integer m, the number of good roads amongst these cities. Each of the next m lines contains three space separated integers u, v, w. This denotes that there is a good road between

u and v of cost w. Recall that w can take only two values which is one of w_1 or w_2.

The next line contains an integer p denoting the number of bad roads. Each of the next p lines contains two space separated integers u, v. This denotes that there is a bad road between city u and v.

Output format:

Output two space-separated integers - the size of largest subset that can be selected and the minimum cost of the selected subset.
If it is not possible to select any subset , then print -1.

Input Constraints :

- $1 <= N <= 10^5$
- $1 <= u, v <= N$
- $1 <= w <= 10^5$
- $0 <= m <= 10^6$
- $0 <= p <= 10^6$

Example :

Input 1 :

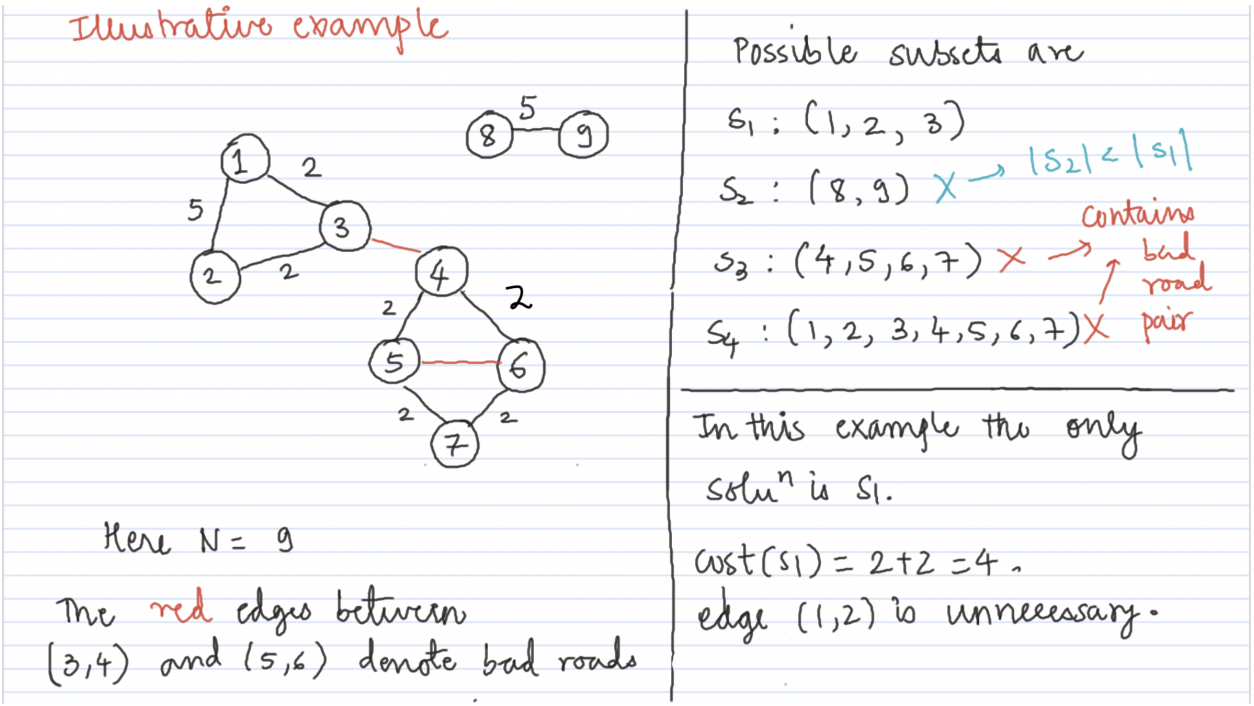
2
0
1
1 2

Output 1:
-1

Explanation :
Cities 1 and 2 have a bad road between them. Hence, none of the nodes can be selected.

Input 2 :

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


Illustrative example
8
5
9
1
2
5
3
2
2
4
2
2
5
6
2
2
7
Here N = 9
The red edges between (3,4) and (5,6) denote bad roads
Possible subsets are
S1 : (1, 2, 3)
S2 : (8, 9) X
|S2| < |S1|
S3 : (4,5,6,7) X
contains bad road pair
S4 : (1, 2, 3, 4, 5, 6, 7) X
In this example the only solu^n is S1.
cost(S1) = 2+2 = 4.
edge (1,2) is unnecessary.

Output 2:
3 4

CS2710 - Programming and Data Structures Lab

Lab Exam - 1

Sep 1, 2021

Instructions

- This lab is **graded exam worth 15%**. You are expected to solve the two problems on repl.it using C++.
- You are expected to solve each problem on your own. Since this is an exam, TAs wont be assisting you.
- You are not expected to use the internet for browsing, discuss anything with your friends or refer to any source. Any violations, if found, will be dealt with very strictly.
- You are not expected to use any inbuilt functions apart from cin and cout.
- All problems must be submitted in lab latest by 4.30pm. Late submissions will not be evaluated.
- There are two questions in this exam. The first question is based on arrays and the second on doubly linked lists. Q1 is worth 7 marks and has 4 public and 3 private test cases. Q2 is worth 8 marks and has 5 public and 3 private. Each test case will fetch 1 mark. The code will also be manually checked for satisfying the constraints specified in the problems.

---

1. (7 points) Given an array of N integers in non-decreasing order, find the number of occurrences of a given query integer X in the array.
A simple linear scan of the array will solve the problem. However, such an implementation will not fetch any credit. You are expected to write a function
`int get_count(int A[], int n, int X)`
which takes as input the array A, the number of elements in A, and the query integer X. It returns the number of occurrences of X in A.
This function should run in worst case time which is $O(k + \log N)$. Here k denotes the number of occurrences of X in the input array. Also the function is expected to use constant extra space (O(1)).

Input format:

- First line of input will be number of entries in the array denoted by N.
- Next line contains N space separated integers which are the elements of the array.
- Next line will be the query integer X, as described in problem.

Input Constraints :

- $1 <= N <= 10000$
- $-10^5 <= A[i] <= 10^5$
- $-10^5 <= X <= 10^5$

Example :

Input 1:
6
1 2 3 3 4 5
3
Output 1:
2
Explanation 1:
The number 3 occurs twice in the given array.

Input 2:
7
1 2 2 4 4 4 5
4
Output 1:
3
Explanation 1:
The number 4 is present thrice in the array.

2. (8 points) In this problem we store character strings as a doubly linked list (of characters). All our strings have only lowercase English alphabets { a, . . . , z}.
We wish to support basic string operations using doubly linked list implementation. You are supposed to implement these operations for the doubly linked list. We assume that the strings are indexed from 1 to n. That is, for a string "programming", index 1 contains 'p', index 2 contains 'r' and so on.

   - setK (int k, char c) : Set the k-th character in the string to the given character c. This operation will be invoked with a query code S. For instance a setK(9, 'a') on the string "programming" will change it to "programmang".
   - increment(): This operation increments the character at the j-th index by j characters, for all $1 <= j <= n$. This operation will be invoked with a query code I. For instance increment() on the string "program" will change it to "qtrkwgt". Similarly increment on "day" will change it to "ecb". Note that, we consider that the alphabets a to z are circularly arranged, hence the increment of 'z' by 1 leads to 'a'. Therefore the increment of 'y' by 3 in "day" leads to 'b'.
   - moveToFront(k): This query moves the k-th element from the **tail of the list** and moves it to the front of the list. This operation will be invoked with the query code R. Note that for this operation, you are not expected to copy characters, but achieve this via pointer manipulations only. For instance, moveToFront(3) on string "program" will result in "rprogam".

   Note: You are provided with the code to insert a character to the end of the list and also to print the list forward and backwards. Additionally main function is implemented for you. You are not supposed to edit these functions. Go through the template2.cpp provided to understand the implementation of the DoublyLinkedList.

   Note:

   - For all cases you are allowed to use O(1) auxiliary storage apart from the lists stored. Creating new nodes is not allowed.
   - It is expected that the strings will be maintained as doubly linked lists, maintaining them as arrays or sets will fetch no credit. Also as mentioned earlier, the moveToFront must involve only pointer manipulation and not copying of data.

   Input format:

   - First line of input contains an integer N denoting the size of the first string.
   - Second line will contain N space separated characters denoting the elements of the first string.
   - Third line will contain an integer Q denoting the number of queries to be performed.
   - Following Q lines will contain the query character, followed by parameters (if required).
     integer k followed by a character c is provided for query code S; integer k will be provided for query code R; whereas no addition parameter is needed for query code I.

   Output format: Print the string in forward and backward form after each query. This is to check that both pointers of linked list is maintained properly. This has been taken care of, check the template code for details.

   Input Constraints :

   - $1 <= N <= 10^4$
   - $a <=$ characters $<= z$
   - $1 <= Q <= 10^2$
   - $1 <= k <= N$

   Note : Assume all parameters provided will be within bounds.

Example :
Input :
5
a b c d e
3
S 4 f
I
R 3
Output :
abcfe
efcba
bdfjj
jjfdb
fbdjj
jjdbf
Explanation :
For the query S 4 f: we change the 4th character to ”f”, list becomes ”abcfe”.
For the query I: we parse the list, a+1 gives b; b+2 gives d; c+3 gives f; f+4 gives j; e+5 gives j and the string is modified to ”bdfjj”.
Finally, for the query R 3: we take the third element from the tail (which is f), and move it to the first position. The resulting string is ”fbdjj”.

Example :
Input :
7
a b x y z p q
3
S 2 d
I
R 3
Output :
adxyzpq
qpzyxda
bfacevx
xvecafb
ebfacvx
xvcafbe
Explanation :
For the query S 2 d: we change the 2nd character to ”d”, string becomes ”adxyzpq”.
For the query I : we perform the increment operations, and the string becomes ”bfacevx”.
Note the roll over for z by 3 and y by 4 due to the increment operation. For instance, ’y’ is to be incremented by 4, it will change to $'y' + 4 =' c'$.
Finally for the query R 3: we take the 3rd element from the tail (that is e), and move it to the first position. The string becomes ”ebfacvx”.

CS2710 - Programming and Data Structures Lab

Lab Exam - 2

Oct 6, 2021

Instructions

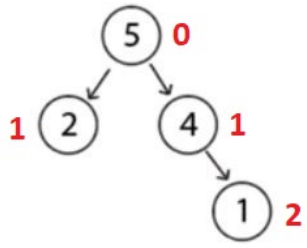
- This lab is **graded exam worth 15%**. You are expected to solve the two problems on repl.it using C++.
- You are expected to solve each problem on your own. Since this is an exam, TAs wont be assisting you.
- You are NOT expected to use the internet for browsing, discuss anything with your friends or refer to any source. Any violations, if found, will be dealt with very strictly.
- All problems must be submitted in lab latest by 5:00pm. Late submissions will not be evaluated.
- There are two questions in this exam. The first question is based on post order traversal of binary trees and the second question is based on level order traversal of binary search tree. Q1 is worth 8 marks and has 4 public and 4 private test cases. Q2 is worth 7 marks and has 4 public and 3 private test cases. Each test case will fetch 1 mark. The code will also be manually checked for satisfying the constraints specified in the problems.

---

1. (8 points) N students appear for CS2710 Quiz 2 exam, and each one of them score different marks in the exam. You are given an array where $i-th$ element of the array represents the marks obtained by the $i-th$ student. The rank of each student is defined in the following way.
The array can be visualized as a Binary Tree (NOT BST) in the following manner:

   - The maximum element of the array becomes the root of the tree.
   - All the elements in the array to it's left(if present) form the left subtree.
   - All the elements in the array to it's right(if present) form the right subtree.
   - The first rule is recursively applied to the left and right sub trees(if they exist).

   The rank of each student is the depth of the node (from the root) in this tree. You will be given K number of queries. Each query consists of an integer i where $0 \leq i \leq N-1$. You need to print the rank of the $i-th$ student. For example if the array contains 2, 5, 4, 1 then it can be visualized as the tree shown below, where red color digit represents the corresponding rank.



   The final tree looks like as shown above. Clearly rank of 0-th student i.e. who obtained 2 marks is 1, and that of 1st student i.e. student who obtained 5 marks is 0 and that of 2nd student who obtained 4 marks is 1 and that of 3rd student i.e. the one who scored 1 mark is 2.

   Note:

   - Rank of student and student numbering is starting from 0.
   - You should not construct the tree explicitly for answering the queries.

   Input format:

   - First line of input will be the number of students denoted by N.
   - Next line contains N space separated integers which denotes the marks of students.
   - Next line will be the number of queries denoted by integer K.
   - Next line will contain K integers separated by space where each integer is between 0 to $N-1$.

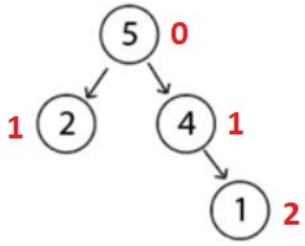
   Input Constraints :

   - $1 <= N <= 2 * 10^3$
   - $0 <= arr[i] <= 10^8$
   - $0 <= K <= 10^4$

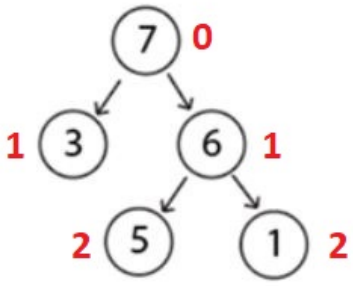
   Expected Time Complexity: $O(N^2)$ for preproceesing work.
   Each query need to be answered in $O(1)$.
   Expected Space Complexity: $O(N)$.

Example :

Input 1:
4
2 5 4 1
3
1 3 1
Output 1:
0 2 0
Explanation 1:



The final tree looks like as shown above. Clearly rank of 1st student i.e. student who obtained 5 marks is 0 and that of 3rd student i.e. the one who scored 1 mark is 2.
Input 2:
5
3 7 5 6 1
5
0 1 2 3 4
Output 2:
1 0 2 1 2
Explanation 2:



The final tree looks like as shown above. Clearly rank of the students are 1 0 2 1 2.

2. (7 points) Given a **Binary Search Tree** with N nodes, and you need to answer some queries. You will be given a number *K* denoting the number of queries. Each query will be either **L** or **I** followed by two space separated integers, or **E** followed by two space separated integers.

Input Format:

- **N :** number of nodes in BST.
- **N** space separated integers.
- **K :** K represents number of queries.
- **L :** Print level order traversal from **right to left**.
- **I level1 level2:** Print level order traversal from **left to right** and from level1 to level2 **including** the nodes on level1 and level2 .
- **E level1 level2:** Print level order traversal from **left to right** and from level1 to level2 **excluding** the nodes on level1 and level2.

Input Constraints :

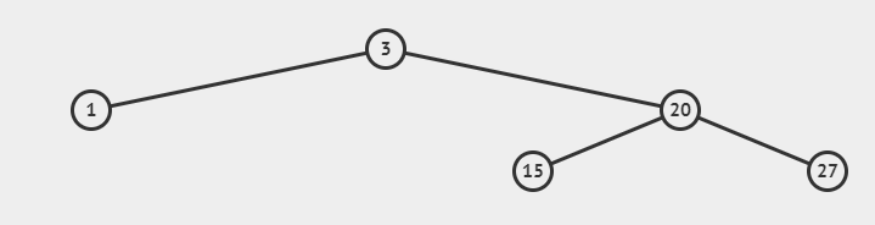
- $1 <= N <= 10^5$
- $0 <= level1, level2 <= 10^5$
- $level1 <= level2$
- $1 <= K <= 100$

Note:

- A template is provided to you in which you are supposed to implement the `queries(BinaryTree* root)` function and take the corresponding inputs accordingly.
- Root is at level 0.
- If no nodes are present between the levels, print "NO NODES"

Expected Time Complexity: $O(N)$ per query.
Expected Space Complexity: $O(N)$.

Example 1:



Input:
5
3 20 1 27 15
4
L
I 1 2
E 0 2
E 0 1

Output:
3 20 1 27 15

1 20 15 27
1 20
NO NODES