2020CS10348 Harshit Mawandia

Problem 1 : Algorithm

Let the two stacks be S_1 and S_2 and the n numbers in the queue be $a_1, a_2 \ldots a_n$.

What we do is that first we insert a_1 into stack S_1.

1) Then for each a_i where $i > 1$ first we check that if the topmost element in S_2 (let it be a_k, $k < i$) is larger than the a_i if $a_k \geq a_i$ we add a_i to S_1, else if $a_k < a_i$, we pop a_k from S_2 and insert it into S_1. Then again we check if the same for next element of S_2 with a_i.

2) Now if a_i is smaller that topmost element of S_2, we check if it is larger than topmost element of S_1. If it is, Then we add a_1 to S_1, else we pop elements from S_1 and keep adding them to S_2 until we find an element which is smaller than a_i.

3) finally, when we have done this for all a_i $1 \leq i \leq n$, We pop elements from S_2 and add them to S_1 till S_2 is empty and we get a sorted stack S_1 with minimin element at bottom.

The time complexity for this algorithm would be n^2 since for each step we might have to shift the entire i sized stack from 1 stack to another and we have to do this n times.

Problem

Harshit Mavandia

Marshit Menaria

Problem 2 :

For function Int mod (p) - overflow (x, y) we first use the function Int sum (x, y) and then use the function Int - less than equal to

Int less than equal (x, y)

Let the result of Int sum be (z)

We now try Int less than equal to (z, p)

if this returns false (0) we return 1

if this returns true (1), we try

Int is equal (z, p). If this returns 1 we return 1, else we return 0.

Navshit

Problem 2.2. Navshit Nawandar

For Int Base(p) - Create(a).

1) First we use the function from Problem 2.1 Int-mod(p) - overflow(a, 0).

Int-mod(p) - overflow(a, 0).

If this returns 1, we use Int-mod(a, p) and store the result in and Int-list, then we use Int-div(a, p) and then restore the step 1.

now we use Int list - add front to add this result to the front of the list.

2) Now when overflow returns 0, we just add the remaining value to the front of the list.

Harshit Marwaha

Date
Page No.

Problem 2.3:

We first ~~use~~ ~~Int.Base(p)~~ make two lists to be n

take length of the lists to be n

now for $i = 0$, $i \leq n$,

we iterate.

For any i^{th} element we have a carry (say k) if $i \geq 1$ ($k \geq 0$) and element at position (i)

we use Int-sum (k, a_i, b_i) (a_i, b_i be the i^{th} elements of 2 lists)

Then we use Int-mod (Int-sum(k, a_i, b_i), p)

which will be our i^{th} element of answer.

and Int-div (Int-sum(k, a_i, b_i), p) which will be our k_{i+1}.

Now when $i > n$ we just use k_{n+1} with a_i (the element of larger list)

and take rest b_i's to be 0

to get our final Int-mod(p).list.