Question 6 (seed = 85226)

Subscribe for email updates. UNRESOLVED

No tags yet. + Add Tag

Sort replies by: Oldest first Newest first Most popular

Vitaliy Belyy · 10 days ago

Could, please, anyone explain where is my mistake?

Question:

```
Suppose that you have a data type for a sequence of N items and that it is repr
esented internally using a resizing array (where the ith item in the sequence i
s stored a[N-i-1]). Assume that the data type is implemented in an efficient an
d natural manner given the specified representation.
```

Operation:

```
 remove and return the last item in the sequence
```

We need to match operation with its amortized time.

I've answered that it takes constant amortized time. As I unsderstand, it should looks like:

```
int lastItemIndex = a.length - SeqSize
Item result = a[lastItemIndex]
a[lastItemIndex] = null
SeqSize--
if (SeqSize < a.length/4) resize(a.length / 2)
return result
```

But the right answer was that operation takes linear amortized time.

Where I was wrong?

0 · flag

Bruno Lehouque COMMUNITY TA · 10 days ago

There are two flaws in your implementation.

1. The index is 0.
2. I let you try to tind the 2nd one.

-1 · flag

+ Comment

Vitaliy Belyy · 10 days ago

Quote from question:

```
where the ith item in the sequence is stored a[N-i-1]
```

As I understand we just store elements starting from the end of array.
So, sequence containing one element will have SeqSize = 1, and single (0-th) element will have index N-0-1=N-1 (N - SeqSize). Where is flaw?

And I say nothing about such kind of indexing. Especially in context:

```
data type is implemented in an efficient and natural manner
```

1 · flag

+ Comment

Bruno Lehouque COMMUNITY TA · 10 days ago

> As I understand we just store elements starting from the end of array.

N is the number of elements, not the array size.

> And I say nothing about such kind of indexing. Especially in context:

What's wrong with it? This could be used to represent a stack, for example.

0 · flag

+ Comment

Vitaliy Belyy · 10 days ago

We can implement stack with first element in 0th position and last element in (N-1)th position. And it will looks like Ok.
But if we start from the end it looks quite unnatural.

If N is number of elements, as you say. It will be "slightly" inefficient. We will have to move N element after each insertion, and after each deletion. I can't imagine reason to do so. And how we can say in this case, that data type implemented in efficient manner? (I don't say about naturality)

0 · flag

+ Comment

Bruno Lehouque COMMUNITY TA · 10 days ago

> We can implement stack with first element in 0th position and last element in (N-1)th position. And it will looks like Ok. But if we start from the end it looks quite unnatural.

You're thinking backwards. That's why you got the wrong answer (and the wrong implementation). In a stack, you push/pop the first element, not the last one.

> If N is number of elements, as you say

That's how N is defined in the question.

0 · flag

+ Comment

Vitaliy Belyy · 10 days ago

Actually, no matter how to think about Stack. It will be just Stack. You can push and pop elements, and it's all, Way to store items inside is just question of efficiency.

And there is nothing wrong with my implementation. I've seen something very similar in one book about Algorithms:

```
import java.util.Iterator;
public class ResizingArrayStack<Item> implements Iterable<Item>
{
  private Item[] a = (Item[]) new Object[1]; // stack items
  private int N = 0; // number of items
  public boolean isEmpty() { return N == 0; }
  public int size() { return N; }

  private void resize(int max)
```

```
  { // Move stack to a new array of size max.
    Item[] temp = (Item[]) new Object[max];
    for (int i = 0; i < N; i++)
      temp[i] = a[i];
    a = temp;
  }

  public void push(Item item)
  { // Add item to top of stack.
    if (N == a.length) resize(2*a.length);
    a[N++] = item;
  }

  public Item pop()
  { // Remove item from top of stack.
    Item item = a[--N];
    a[N] = null; // Avoid loitering (see text).
    if (N > 0 && N == a.length/4) resize(a.length/2);
    return item;
  }

  public Iterator<Item> iterator()
  { return new ReverseArrayIterator(); }

  private class ReverseArrayIterator implements Iterator<Item>
  { // Support LIFO iteration.
    private int i = N;
    public boolean hasNext() { return i > 0; }
    public Item next() { return a[--i]; }
    public void remove() { }
  }
}
```

N is a bit overloaded term. It can have different meanings in different contexts. And there in question is said about efficiency too.
IMHO, when someone see words "sequence represented internally using a resizing array" and "efficient and natural manner" he will think about implementation similar to represented above.

0 · flag

+ Comment

Vitaliy Belyy · 10 days ago

Maybe I'm grumbling a lot. Thank you for response, Bruno. I'll clear possible misunderstanding before submitting next attempt.

0 · flag

Bruno Lehouque COMMUNITY TA · 10 days ago

This implementation is exactly the structure described in the question. The first item is in last position and the last item is at position 0 (look at push/pop and the iterator order). Items are not stored from 0 to N-1 but from N-1 to 0 (what you considered "unnatural").

Removing the last item, which is at position 0, is not implemented. You can only remove the first one. Now, if you had to implement it (what you did incorrectly), to maintain the invariant "*the ith item in the sequence is stored a[N-i-1]*", you'd need to shift all the elements to the left.

> N is a bit overloaded term. It can have different meanings in different contexts.

Throughout the whole course, N has always been the number of elements in a data structure (when used). And here, it's explicitnely stated in the first sentence.

1 · flag

+ Comment

New post

To ensure a positive and productive discussion, please read our forum posting policies before posting.

B | I | Link | <code> | Pic | Math | Edit: Rich | Preview

Make this post anonymous to other students

Subscribe to this thread at the same time

Add post