Data Structures

April 25, 2019

1 Elementary Data Structures

1.1 Stacks and Queues

Stacks and queues are *dynamic* data structures in which we are constantly modifying the elements of a partiular set. In a stack, if we want to remove an item, we ususally remove the last element in the stack, and we call it a *last-in-first-out* type of data structure, since the last element to ocme in is the furst element we remove.

1.1.1 Stacks

As mentioned, stacks are *first-in-first-out* data structures, which means that only the 'top' element is accessible at any given time. The operations we may perform are all in O(1), meaning that they are constant time operations. These include:

1. `Push(S,value)`: When we push a value onto the stack, we are adding an element to the array and moving the `S.top` attribute, increasing it by 1.

2. `Pop()`: When we pop an item, there is no need for an argument. We just reset `S.top` down by 1, and potentially overwrite the data item when we have to.

Essentially, a stack behaves like an array with n element, and we can only use the topmost one. If the top value exceeds n, we say that **stack overflow** (ayyy lmao)has occurred. The opposite, when we try to pop an empty stack, we say that **stack underflow** has ocurred.

1.1.2 Queues

For a queue, we can perform similar operations to a stack, except that they have different names. For example, an insert operation is called `enqueue`, and a delete operation is called a `dequeue`.

We can compare a queue to a line of customers, where the first one in line is the first one to proceed. The elements all come into the back of the queue, and elements are only taken out of the front, or *head*.

Another distinctive feature of queues is that the elements 'wrap around', that is, if the queue goes to the end of the array, the next element will be arr[0].