[CSC-531] Tutorial 1
Amortized Analysis

1 Tall Buildings

Chicago has many tall buildings, but only some of them have a clear view of Lake Michigan. Suppose we are given an array $A[1 \cdots n]$ that stores the height of n buildings on a city block, indexed from west to east. Building i has a good view of Lake Michigan if and only if every building to the east of i is shorter than i. Use a stack to design an algorithm that computes which buildings have a good view of Lake Michigan in $O(n)$ total time.

[**Hint**] Use a stack.

2 Queue using two stacks

Queues and Stacks are two commonly used data structures, queues implement First In First Out policy and stacks implement Last In First Out Policy. Describe an implementation of the *push* and *pop* operations of a queue, using two stacks where the operations of stacks are used as a black box. Show that the queue takes $O(1)$ amortized time per operation.

3 Dynamic Arrays

A dynamic array is a data structure that stores a sequence of items and supports the following operations.

- **InsertEnd(x)** inserts x to the end of the array.
- **DeleteEnd()** deletes element from the end of the array.

We maintain an array of capacity k, while the current number of elements is n, as follows.

Algorithm 1: Dynamic Arrays with size n and capacity k

```
InsertEnd(x):
if $n == k$ then
    Allocate new array of size $2k$;
    Copy elements to new array;
    $k \leftarrow 2k$;
$D[++n] \leftarrow x$;

DeleteEnd(x):
if $n == k/4$ then
    Allocate new array of size $k/2$;
    Copy elements to new array;
    $k \leftarrow k/2$;
$n \leftarrow n - 1$;
```

Show that dynamic arrays use $O(n)$ space, and amortized $O(1)$ time per operation.

PRACTICE PROBLEMS

Solve the problems from Chapter 17 Amortized analysis from Textbook CLRS for practice.