# A2

ddl: 2023/10/1 23:59

## 1 Array-based Data Structures

### 1.1 Implement two Stacks in one Array

*From 10.1-3 of CLRS*

Explain how to implement two stacks in one array $A[1:n]$ in such a way that neither stack overflows unless the total number of elements in both stacks together is $n$. The `PUSH` and `POP` operations should run in $O(1)$ time.

## 2 Linked List

### 2.1 Reverse a Linked List

*From 10.2-5 of CLRS*

Give a $\Theta(n)$-time non-recursive procedure that reverses a singly linked list of $n$ elements. The procedure should use no more than constant storage beyond that needed for the list itself.

## 3 Divide-and-Conquer & Recursion

### 3.1 Multiply Matrices with Squaring Algorithm

*From 4.2-6 of CLRS*

Suppose that you have a $\Theta(n^\alpha)$-time algorithm for squaring $n \times n$ matrices, where $\alpha \geq 2$ is a constant. Show how to use that algorithm to multiply two different $n \times n$ matrices in $\Theta(n^\alpha)$ time.

### 3.2 Sort the Pancakes

*From 1.9 of Algorithms by Jeff Erickson*

Suppose you are given a stack of $n$ pancakes of different sizes. You want to sort the pancakes so that smaller pancakes are on top of larger pancakes. The only operation you can perform is a *flip*—insert a spatula under the top $k$ pancakes, for some integer $k$ between 1 and $n$, and flip them all over.

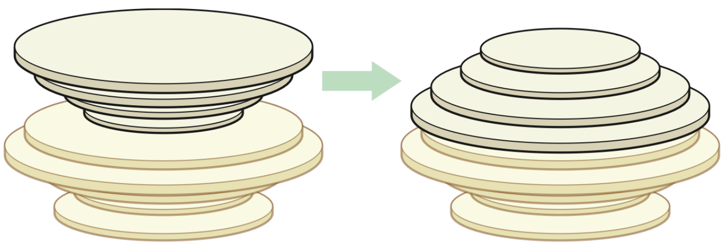

**Figure 1.19.** Flipping the top four pancakes.

1. Describe an algorithm to sort an arbitrary stack of $n$ pancakes using $O(n)$ flips. *Exactly* how many flips does your algorithm perform in the worst case?[1]

2. Now suppose one side of each pancake is burned. Describe an algorithm to sort an arbitrary stack of $n$ pancakes, so that the burned side of every pancake is facing down, using $O(n)$ flips. *Exactly* how many flips does your algorithm perform in the worst case?

## 3.3 Calculate the Inversions

*From 1.13 of Algorithms by Jeff Erickson*

An **inversion** in an array $A[1 \ldots n]$ is a pair of indices $(i, j)$ such that $i < j$ and $A[i] > A[j]$. The number of inversions in an $n$-element array is between $0$ (if the array is sorted) and $\binom{n}{2}$ (if the array is sorted backward). Describe and analyze an algorithm to count the number of inversions in an $n$-element array in $O(n \log n)$ time. [Hint: Modify mergesort.]

1. The *exact* worst-case optimal number of flips required to sort $n$ pancakes (either burned or unburned) is an long-standing open problem; just do the best you can. ↩