CSCI 5454 – Fall 2024
Design and Analysis of Algorithms

Homework 2

Instructions:

- **Topics:** Analysis of algorithms, the Euclidean algorithm and its analysis.
- **Due date:** This homework is due by 11:59PM Mountain Time on **Friday, September 13th**. Late assignments will not be accepted. Your lowest two homework scores for the semester will be dropped.
- You are welcome and encouraged to discuss the problems with classmates, but **you must write up and submit your own solutions and code**. You must also write the names of everyone in your group on the top of your submission.
- The primary resources for this class are the lectures, lecture slides/notes, the CLRS and Erickson algorithms textbooks, the teaching staff, your collaborators, and the class Piazza forum. We strongly encourage you only to use these resources. If you do use another resource, make sure to cite it and explain why you needed it. **Using generative AI tools (e.g., ChatGPT), Q&A forums (e.g., Stack Exchange, Quora), or cheating repositories (e.g., Chegg, CourseHero) is not allowed.** See the syllabus for a more detailed explanation.
- There are three questions, worth 50 points in total.
- You must justify all of your answers unless specifically stated otherwise.
- We strongly encourage (but do not require) you to write your solutions in LaTeX, and have provided a skeleton file.[1] However, if your homework is illegible then we reserve the right not to grade it.

[1] If you have not used Latex before, it's easy to get started using Overleaf. See their 30-minute tutorial: https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes.

Questions

Question 1. (Linear-Time Divide and Conquer for Max Subarray Sum Problem.) We would like to develop a linear time algorithm for the maximum subarray sum problem that we studied in class. Let A be an array of n elements indexed from 0 to $n-1$. Recall the following definitions:

Max Subarray Sum: $\max_{l\in\{0,\ldots,n-1\}} \max_{u\in\{l,\ldots,n-1\}} \sum_{j=l}^{u} A[j]$.

Max Prefix Sum: $\max_{u\in\{0,\ldots,n-1\}} \sum_{j=0}^{u} A[j]$.

Max Suffix Sum: $\max_{l\in\{0,\ldots,n-1\}} \sum_{j=l}^{n-1} A[j]$.

In the original divide and conquer algorithm for computing max subarray sum, we proceeded as follows (for arrays of length 2 or more):

- Divide the array into two sub-arrays each of which is half the size of the original array.
- Recursively solve for each subarray and compute the max-subarray sums m_l, m_r for the left and right subarrays respectively.
- Compute the *max suffix sum* suff_l for left subarray and *max prefix sum* pref_r for the right sub-array. Add them up to obtain $m = \mathsf{suff}_l + \mathsf{pref}_r$.
- The overall result is $\max(m_l, m_r, m)$.

This yielded a $O(n\log(n))$ time algorithm. In order to develop an $O(n)$ divide and conquer algorithm, we need to compute the combine step in $O(1)$ time. We will do so by tracking additional information to help us achieve this goal.

1. (5 points) Let s_l, s_r be the total sum of all numbers of the left and right subarrays. Also, let $\mathsf{pref}_l, \mathsf{pref}_r$ be the maximum prefix sums of the left and right subarrays, respectively while $\mathsf{suff}_l, \mathsf{suff}_r$ be the maximum suffix sums of the left and right subarrays, respectively. Write an expression for the maximum prefix sum of the original array (pref) and maximum suffix sum of the original array (suff) in terms of $s_l, s_r, \mathsf{pref}_l, \mathsf{pref}_r, \mathsf{suff}_l, \mathsf{suff}_r$.
2. (15 points) Using the information above, devise a divide and conquer scheme for the maximum subarray sum problem that simultaneously returns the following information: (a) sum of all elements, (b) maximum prefix sum, (c) maximum suffix sum, (d) maximum subarray sum.

Question 2. (Maximum Submatrix Sum) In class we looked at algorithms for computing the maximum subarray sum. Consider the 2D version of the problem that we will call *maximum submatrix sum*. We are given as input a square $n \times n$ matrix A of numbers. Our goal is to find a "rectangular" sub-matrix $A[l_1, \ldots, l_2; r_1, \ldots, r_2]$, wherein $0 \le l_1 \le l_2 \le n-1$ and $0 \le r_1 \le r_2 \le n-1$ with the largest sum. In other words, consider all entries in the "rectangle" with corners (l_1, r_1) (top-left) and (l_2, r_2) bottom right.

Formally, for $l_1 \in \{0, \ldots, n-1\},\ l_2 \in \{l_1, \ldots, n-1\},\ r_1 \in \{0, \ldots, n-1\},\ r_2 \in \{r_1, \ldots, n-1\}$:

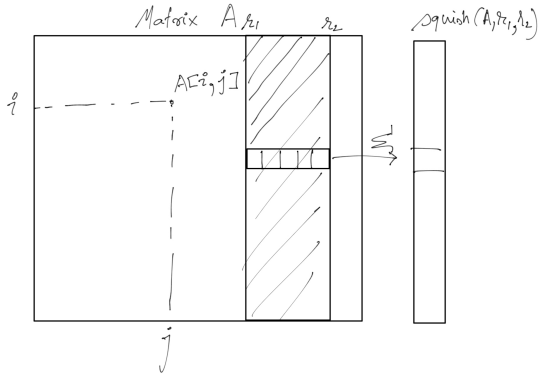
$$\max_{(l_1,l_2,r_1,r_2)} \sum_{i=l_1}^{l_2} \sum_{j=r_1}^{r_2} A[i,j]\,.$$

We will work through an algorithm that solves for the maximum submatrix sum using the maximum subarray sum.

(A, 5 points) For each pair $0 \leq r_1 \leq r_2 \leq n-1$ define a (one-dimensional) array

$$\mathsf{squish}(A, r_1, r_2)[i] = \sum_{j=r_1}^{r_2} A[i,j]$$

Here is a pictorial representation of the squishing process.



Write down an expression for each entry of $\mathsf{squish}(A, r_1, r_2+1)$ in terms of $\mathsf{squish}(A, r_1, r_2)$ and some entries of the matrix A.

(B, 15 points) Derive an $O(n^3)$ algorithm for maximum sub-matrix sum that uses Kadane's algorithm for maximum subarray sum.

(**Hint**: What does running Kadane's algorithm on $\mathsf{squish}(A, r_1, r_2)$ yield? Draw a picture and visualize.)

Question 3. (Solve recurrences by expansion) For each of the recurrences $T(n)$ below, use the recursion tree (tree expansion) method to find a closed-form solution of the form $T(n) = \Theta(g(n))$ for some function $g(n)$. You must use the recursion tree method and you must show your work. However, we encourage you to use the Master Theorem to check your final answer.

a. (5 points) $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n/4) + \sqrt{n} & \text{if } n > 1 \end{cases}$

b. (5 points) $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(n/3) + n^3 & \text{if } n > 1 \end{cases}$

c. (5 points) $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 7T(n/2) + n^2 & \text{if } n > 1 \end{cases}$