Analysis of Algorithms
Lab 7

Andrés Calderón, Ph.D.

March 27, 2025

1 Introduction

Lab 7 focuses on solving algorithmic problems through strategic thinking and efficient design. The first challenge, Burning the Candle at Both Ends, involves a two-player game where the goal is to maximize one's score using dynamic programming, even against an optimal opponent. A variation of the game adds complexity by changing how players pick numbers.

The second problem, Roller Coaster, requires finding the longest subarray where height changes stay within a given limit. We approach this using two algorithms with different time complexities, depending on the size of the constraint.

This lab highlights key algorithmic techniques like dynamic programming and sliding windows, with an emphasis on performance and problem-solving strategies.

2 Burning the Candle at Both Ends

When Yihan and Yan are free, they play the following game. Yan will write a list of 2n positive numbers (the length is always an even number) like the following:

$$5, 1, 3, 9, 6, 2, 4, 8$$

Then Yihan and Yan will in turn take the numbers from the two ends. Yihan makes the first move, and she can pick either 5 or 8. Then Yan will make the next move, and he can pick either of the two ends at that this point (1 / 8 if Yihan picked 5 first, or 5 / 4, if Yihan picked 8 first). They will repeat until no number is left. Finally, each of them will compute the sum of all numbers s/he picked as their score. Whoever gets the higher score wins (if the sums are the same, then that's a tie).

Yihan is very greedy. Not only does she wants to win, she also wants to win as much as possible (i.e., maximize the number of points she gets at the end!), even if Yan is very smart and will always play optimally. In all the following questions, you can assume that Yihan always plays first.

Exercises

1. Design an algorithm to help Yihan compute the highest score she can guarantee to get, however Yan plays the game. Simulate your strategy on the example input given above: what is the highest score Yihan can get in that example? Your algorithm must have time complexity $O(n^2)$.

2. Since Yihan found a winning strategy, Yan was very unhappy and decided to change the rule. Assume now each player can take two numbers in a round. They can be both from one end, or one at both ends. Namely, Yihan will first pick two, then Yan will picks two, and repeat until no numbers are in the list. Can you still design an algorithm to help Yihan know what is the highest score she can get? Your algorithm must have time complexity $O(n^2)$.

3 Roller Coaster

Yihan went to the Universal Studio. She really wanted to try the roller coaster, but she is timid –it's so horrible to change height rapidly! In fact, the maximum height changing during the whole roller coaster riding she can accept is some integer k. The roller coaster is divided into n segments. The height of the i^{th} segment is $h[i]$. All elements in array h are integers.

To help Yihan enjoy the roller coaster, the staff at Universal Studio proposed a solution: Yihan can select a contiguous interval in the whole roller coaster, with at most k difference in height. That is to say, in the array $h[i]$, Yihan needs to select a contiguous interval $h[s \dots t]$, where $max_{s \le i \le t} h[i] - min_{s \le j \le t} h[j] \le k$. Yihan still want to maximize the experience of riding roller coaster: she wants to maximize the length of this selected interval.

For example, if the sequence of roller coaster height is {3, 5, 2, 6, 8, 7, 9, 5}, and $k = 3$, Yihan can select {6, 8, 7, 9}. The maximum length is 4. Actually, this is the longest possible interval she can find. The height of each segment can be general integers (i.e., it can be more than n).

Although the roller coaster can be super long, the "safe" range that Yihan can accept is small 😜. We can assume k is a small value, even $o(\log n)$. We will take advantage of this condition to find a solution.

Exercises

1. Design an algorithm that runs in $O(n \log n)$ time.

2. Design an algorithm that runs in $O(nk)$ time. This is more efficient than the $O(n \log n)$ algorithm when k is small.

For both of the problems, you need to justify the time complexity.

4 What Do We Expect?

You will compile all your answers into a well-structured report and submit it before the lab on **April 9, 2025**. Please submit your report in **PDF format** and email it to me with the subject line formatted as `[ADA] Lab 7` and your names in the body of the email.

Happy Hacking 😎!