Analysis of Algorithms
Lab 6

Andrés Calderón, Ph.D.

March 19, 2025

1 Introduction

Dynamic Programming (DP) is a powerful technique used to solve complex optimization problems efficiently. It is widely applied in various domains, including operations research, artificial intelligence, and competitive programming.

In this lab, we will deepen our understanding of DP by exploring real-world examples and algorithmic challenges. We will start by reviewing a classic problem –the Minimum Coins problem– before analyzing a more advanced scenario from MIT OpenCourseWare's Analysis of Algorithms course. Additionally, we will tackle a practical problem, "Everything on Sale!", which models decision-making under budget constraints using DP.

Through a series of exercises and algorithmic challenges, you will learn how to structure DP solutions, analyze their computational complexity, and implement efficient strategies to optimize outcomes. By the end of this lab, you should have a strong grasp of DP techniques and their practical applications.

Let's get started!

2 A YouTube Video

The first thing we will do in this lab is reinforce one of the examples where DP is quite handy. This video will help us gain a better understanding of the Minimum Coins problem we saw in the previous lab. It is essential to understand how the DP table is computed and the general approach followed here, as it will be useful in the subsequent sections. You do not need to submit any particular item for this section—just pay attention and take notes on the procedure.

3 Let's Practice With an Example

A further class on the Analysis of Algorithms from MIT OpenCourseWare was also taught in Spring 2015. It was *6.046J Design and Analysis of Algorithms*, and it is available at this link. Lecture 10 discusses an interesting example that illustrates the use of DP to solve an extremely simple game aimed at maximizing the amount (value in $) of coins a player can collect from an array to defeat an opponent.

Let's watch this video from minute 54:15 until the end. Remember that you can enable English (auto-generated) subtitles and then use the auto-translate feature to select the Spanish option in the video's settings if you feel more comfortable with that.

From the video, please address the following requests:

Exercises

1. Explain how the number of subproblems is computed and analyze the complexity of the proposed DP solution.
2. Draw a DP table that solves the initial example shown in the video by applying the DP solution.

4 Individual Work

You will follow the next statement and then answer a couple of questions.

4.1 Everything on Sale!

Nancy is going shopping and discovers that the shopping mall is having a big sale. There are n items on sale. For each item, Nancy knows the original price p_i and the current price t_i. However, there are some limitations to the sale: some items are categorized into the same group, and the discount can apply to **at most** one item in each group (i.e., she must pay the original price p_i if she wants to buy a second item from the same group). Each item belongs to exactly one group. You can assume that all element *ids* in group k are stored in an array $L_k[]$. Nancy plans to buy **at most one item** from each group.

All p_i and t_i are positive integers, and it is always true that $p_i > t_i$ for all i. Nancy considers the benefit of buying an item to be the difference between p_i and t_i (i.e., $p_i - t_i$). She has a budget of W dollars and wants to determine the maximum total benefit she can achieve within this budget.

Questions

1. Describe an algorithm that helps Nancy select items within W dollars while maximizing her total benefit. Your algorithm should run in $O(nW)$ time. You only need to output the highest benefit in this problem (i.e., there is no need to specify the exact items to buy).
2. Nancy realized that there can be multiple ways to achieve the highest benefit. In this case, she prefers the one with the fewest items, as it makes it easier for her to carry them back 😜. Please modify your algorithm from Question 1 to also output the minimum number of items needed to achieve the highest benefit. In this question, you only need to output the number of items, rather than the exact selection. Your algorithm should run in $O(nW)$ time.
3. Finally, modify your algorithm from Question 2 to also output the list of items that should be purchased to achieve the highest benefit (i.e., maximizing the benefit while using the fewest number of items). Your algorithm should run in $O(nW)$ time.

5 What Do We Expect?

You will compile all your answers into a well-structured report and submit it before the lab on **April 2, 2025**. Please submit your report in **PDF format** and email it to me with the subject line formatted as `[ADA]` **tag + lab number**.

Happy Hacking 😎!