CS-GY 6033: Design and Analysis of Algorithms I

Fall 2024

Instructor: Nairen Cao **NYU Tandon School of Engineering**

Problems	1	2	3	4	5 (ungraded)	6 (Bonus)	Total
Max. Score	25	25	20	30	0	10	100

Table 1: Score Summary

Homework #4

Assigned: 10/23/2024 **Due: 11/7/2024**

General Instructions Write all solutions clearly, concisely, and legibly. It is okay to provide handwritten and scanned/photographed solutions as long as they are clearly legible. If typing, using LaTeX is recommended.

You should generally describe algorithms in words and high-level pseudocode. Specifically, emphasize the goal of each step in your algorithm rather than the coding details. After giving a high-level description, provide more detail on the implementation of each step where appropriate.

The purpose of having these problem sets is to enhance your mathematics and programming skills in the scope of designing algorithms. These problems may be quite challenging, so discussions with classmates are encouraged. However, you should spend at least half an hour thinking about the problems individually before discussing them. Although discussions (and consulting ChatGPT and WolframAlpha) are allowed, you need to write the solution in your own words and by yourself. Please acknowledge any person or reference you discuss with or consult from.

Special Instructions

- Please note that the starting index for this homework is 1. This is because I want you to define the base case as $dp[0]$ in most of the problems.
- **Homework Theme:** Dynamic Programming.

Problem 4-1. Billboard Placement

Suppose you are managing the construction of billboards on an east-west highway that extends in a straight line. The possible sites for billboards are given by numbers $x_1, x_2, \ldots, x_n$ with $0 \leq x_1 < x_2 < \cdots < x_n$, specifying their distance in miles from the west end of the highway. If you place a billboard at location x_i, you receive payment $p_i > 0$.

Regulations imposed by the county's Highway Department require that any pair of billboards be more than 5 miles apart. You'd like to place billboards at a subset of the sites so as to maximize your total revenue, subject to that placement restriction.

Example. Suppose $n = 4$, with

$$\langle x_1, x_2, x_3, x_4 \rangle = \langle 6, 7, 12, 14 \rangle ,$$
$$\text{and} \qquad \langle p_1, p_2, p_3, p_4 \rangle = \langle 5, 6, 5, 1 \rangle .$$

The optimal solution is to place billboards at x_1 and x_3, for a total revenue of $p_1 + p_3 = \$10$.

(a) Professor Manfine proposes the following "greedy" algorithm, which operates by placing billboards as early as possible to satisfy the 5-mile restriction: Give an example (i.e., choose

Algorithm 1 Billboard Placement Algorithm

```
1: Place a billboard at x_1
2: lastboard ← x_1
3: P ← p_1                                              ▷ total revenue
4: for i ← 2 to n do
5:     if x_i > lastboard + 5 then
6:         Place a billboard at x_i
7:         lastboard ← x_i
8:         P ← P + p_i
9:     end if
10: end for
11: return P
```

input $\{x_i\}, \{p_i\}$) for which Professor Manfine's algorithm does not return the maximum revenue.

The remainder of the problem asks you to provide a dynamic-programming algorithm that takes as input an instance (i.e., locations $\{x_i\}$ given in sorted order and their prices $\{p_i\}$) and returns the maximum revenue obtainable from a valid subset of sites.

Your running time should only depend on n, not on the value of x_n or any of the p_i.

(b) For this problem part, suppose you have access to a function *prev*(j) that returns the latest billboard i that precedes billboard $j > i$ by more than 5 miles, i.e., *prev*$(j) = \max\{i < j | x_i < x_j - 5\}$. Define a recursive formula for the value of the solution. You will want to use *prev*(j).

(c) Write pseudocode for a dynamic program that returns the maximum possible revenue, again using *prev* as a blackbox.

(d) Give an algorithm to implement *prev*(j). Analyze the running time of your dynamic program given this algorithm for *prev*.

It's actually possible to achieve a $\Theta(n)$-time algorithm, but it's likely that your first crack at part d led to something much slower. That's okay — **a slower (polynomial time) solution is still worth full credit.** However, it's worth thinking about how to improve your solution. Can you figure out how to reduce the running time to $O(n \log n)$ or, even better, $\Theta(n)$?

This problem highlights a general algorithm design principle — your high-level algorithm should specify what you want, not how to do it. I've already pushed you in that direction by defining the prev function. Only after you've given the main ideas of the algorithm should you resolve implementation details. Doing it this way yields a more flexible algorithm with better modularity: if you can improve the solution for any particular step, you can swap it in without breaking your main algorithm.

Problem 4-2. Chain Splitting

Suppose Scrooge McDuck wants to split a gold chain into $n + 1$ smaller chains to distribute to his numerous nephews and nieces as their inheritance. Scrooge has carefully marked the n locations where he wants the chain to be cut.

McDuck's jeweler, Fenton Crackshell, agrees to cut any chain of length ℓ into two smaller pieces, at any specified location, for a fee of $\$\ell$. To cut a chain into more pieces, Scrooge must pay Crackshell separately for each individual cut. As a result, the cost of breaking the chain into multiple pieces depends on the order that Crackshell makes his cuts. Obviously, Scrooge wants to pay Crackshell as little as possible.

Example. Suppose the chain is 12 inches long, and the cut locations are at 3 inches, 7 inches, and 9 inches one end of the chain. All of these positions must be cut—the only question is what order should the cuts be made in. If Crackshell cuts first at the 3-inch mark, then at the 7-inch, then at the 9-inch mark, Scrooge must pay $\$12 + \$9 + \$5 = \$26$. If Crackshell cuts the chain first at the 9-inch mark, then at the 7-inch mark, and then at the 3-inch mark, Scrooge must pay $\$12 + \$9 + \$7 = \$28$. Finally, if Crackshell makes his first cut at the 7-inch mark, Scrooge only owes him $\$12 + \$7 + \$5 = \$24$. This last solution is the best answer for this input.

Input format. The input to the algorithm is a set of integers $p_0, p_1, p_2, \ldots, p_{n+1}$, with $0 = p_0 < p_1 < p_2 < \cdots < p_{n+1}$. The number $p_0 = 0$ is the beginning of the chain, and the number p_{n+1} is the end of chain (or equivalently its total length). The numbers $p_1, \ldots, p_n$ specify the positions at which cuts need to be made. In the above example, the input would be $0, 3, 7, 9, 12$.

(a) Scrooge's great-nephew Huey suggests the following greedy strategy: cut the median position first, then recurse. This algorithm is given by the following pseudocode, starting with a call to MAKECUTS$(0, n+1)$.

Algorithm 2 MAKECUTS(i, j) ▷ Cut the subchain $p_i, p_{i+1}, \ldots, p_j$ at $p_{i+1}, \ldots, p_{j-1}$

```
1: if j ≤ i + 1 then
2:     No cut to be made. return
3: end if
4: mid ← ⌊(i+j)/2⌋
5: Cut at position p_mid
6: MAKECUTS(i, mid)
7: MAKECUTS(mid, j)
```

Give an input for which Huey's algorithm does not produce the cheapest ordering or cuts. *You don't need an input larger than $n = 3$, but the example above doesn't break Huey's algorithm.*

The remainder of the problem is about designing a good dynamic-programming solution.

(b) Define a recursive formula for the optimal value of cutting the chain.

(c) Write pseudocode for a dynamic program that determines the value of the optimal solution. (You do not have to output the order of cuts.) *To simplify the code, you can feel free to use mathematical expressions like taking the min of multiple choices.*

(d) Analyze the running time of your algorithm.

Problem 4-3. Biking

You've decided to go on a long-distance bicycle trip across the country. You've already designed your route and identified reasonable campsites along the way. The question is where you should set up camp.

Your entire trip can be modeled as a straight line, starting at mile marker 0. The problem input specifies the campsites along the way, at mile markers $0 < x_1 < x_2 < \cdots < x_n$, where x_i is the distance from the starting point of the i-th possible campsite. You must stop at the final campsite (at distance x_n), which is your destination.

The most pleasant ride for you is 50 miles a day, but this may not be possible (depending on the spacing of campsites). If you travel x miles during a day, the penalty for that day is $(50 - x)^2$. You'd like to plan your campsites to minimize the total penalty—that is, the sum, over all days, of the daily penalties.

(a) Clearly define a recursive formula for the minimum total penalty possible.

(b) Write pseudocode for a dynamic program that computes the minimum total penalty for your trip.

(c) Analyze the running time of your algorithm.

(d) Augment your code to also output which campsites to stop at.

Problem 4-4. Minimizing Chapter Unbalancedness

Mr. Hugo has written a long article consisting of n paragraphs. The i-th paragraph contains a_i words. As an editor, your task is to help Mr. Hugo assemble his article into several chapters by dividing the paragraphs under the following conditions:

1. Paragraphs must remain in their original order.
2. Each paragraph must belong entirely to one chapter. It cannot be split between chapters.

Your goal is to split the article into at least two chapters such that the **unbalancedness** of the book is minimized. The unbalancedness is defined as the largest difference between the number of words in the longest and the shortest chapters.

Example
Given the following input:

- Paragraph word counts: $A = [4, 3, 8, 5, 2, 6]$
- Explanation: We will divide all paragraphs into 4 chapters. The first chapter contains paragraphs $A[1], A[2]$, the second chapter contains paragraph $A[3]$, the third chapter contains paragraphs $A[4], A[5]$, and the fourth chapter contains paragraph $A[6]$. The unbalancedness is $8 - 6 = 2$, where 8 is the word count of the longest chapter (Chapter 2), and 6 is the word count of the shortest chapter (Chapter 4).

The algorithm should find the optimal way to split these 6 paragraphs into 4 chapters, minimizing the unbalancedness.

(a) Define a recursive formula for the optimal value of cutting the chain.

(b) Write pseudocode describing your algorithm.

(c) Analyze the running time of your algorithm.

Problem 4-5. Lawrence of Arabia

T. E. Lawrence was a controversial figure during World War I. He was a British officer who served in the Arabian theater and led a group of Arab nationals in guerrilla strikes against the Ottoman Empire. His primary targets were the railroads.

You are to design an algorithm to help Lawrence figure out how best to use his limited resources. You have some information from British Intelligence. First, the rail line is completely linear. There are n depots, numbered $1, 2, \ldots, n$, along this linear stretch of track. Second, British Intelligence has assigned a strategic value $v(i, j)$ to every connected stretch of depots $i, i + 1, \ldots, j$. There is no strategic value for an unconnected depot.

Lawrence is not able to attack the depots directly, but he can destroy sections of track between depots. Lawrence only has enough resources to perform m attacks. Your goal is to help Lawrence determine where to perform his attacks. The strategic value of the full railroad is the sum of the strategic values of each maximal undestroyed interval of track. For example, if Lawrence performs two attacks, destroying the track segments connecting depots 2-3 and the segment connecting depots 6-7, then the remaining strategic value of the railroad is $v(1, 2) + v(3, 6) + v(7, n)$. Lawrence's goal is to perform attacks to minimize the remaining strategic value of the railroad.

To clarify, the input is a number n of depots, a number $m < n$ of attacks to perform, and a table of strategic values $v(i, j) > 0$ for $i < j$. As a convenient shorthand, you may refer to $v(j, j)$ as well (which is equal to 0). You can assume that performing an attack always reduces the strategic value, so Lawrence may as well perform all m attacks.

(a) Give a polynomial-time dynamic-programming algorithm for determining the best (i.e., minimum) possible strategic value that Lawrence can obtain by destroying m segments of track. You must include a clearly defined recursive formula.

(b) Write pseudocode describing your algorithm.

(c) Analyze the running time of your algorithm.

Problem 4-6. Inversions Revisited 2

Given an array A of n numbers, we say that a 5-tuple $(i_1, i_2, \cdots, i_5)$ of integers is inverted if $1 \leq i_1 < i_2 < i_3 < \cdots < i_5 \leq n$ and $A[i_1] > A[i_2] > A[i_3] > \cdots > A[i_5]$.

(a) Give an $O(n^2)$-time algorithm to count the number of such inverted tuples w.r.t A.

(b) Give an $O(n \lg n)$-time algorithm to count the number of such inverted tuples w.r.t A.