Weekly Assignment 6
Total: 100 (+70 bonus)

CS 2500: Algorithms

Due Date: December 2, 2024 at 11.59 PM

Instructions

- Submit your solutions by the deadline specified above.
- Ensure that your work is your own.
- Write your answers clearly and show all your work.
- If you have any questions, ask during recitations or office hours.

Problems [100 points]

1. [**Divide Workloads (20 points)**] A company wants to divide a group of employees into two teams for a project. Each employee is assigned a workload score based on their skill level, represented by an array *workloads*[]. The goal is to divide the employees into two teams such that the difference in the total workload between the two teams is minimized.

   (a) Given the workload scores $workloads = [3, 1, 4, 2, 2]$, devise an approach to determine how the teams should be formed to minimize the difference in their total workload. Explain the process, starting from defining the problem mathematically to computing the result using dynamic programming. [**5**]

   (b) Implement the solution for the above array and report:
   - The minimum difference in workload between the two teams. [**5**]
   - The workload distribution across the two teams. [**5**]

   (c) Discuss how this solution could be extended to handle cases where workloads can be negative (e.g., representing debts or negative contributions). [**5**]

2. **Charity Vouchers (15 Points)** A charity organization is distributing food vouchers to families in need. Each voucher has a denomination value, and the organization wants to minimize the number of vouchers given to each family while ensuring the total voucher amount exactly matches the required amount of assistance.

   The available voucher denominations are $[5, 10, 20, 50, 100]$ (in dollars).

   (a) A family qualifies for \$75 in assistance.
   - Determine the minimum number of vouchers needed to meet this requirement. [**3**]
   - List the denominations of the vouchers provided. [**2**]

   (b) Due to a shortage of \$10 vouchers, they are temporarily unavailable. Solve the problem for a family needing \$85 in assistance and describe how this impacts the distribution. [**5**]

(c) Propose an approach the organization could take to handle dynamic changes in available voucher denominations efficiently, such as during supply shortages. **[5]**

3. **Delivery Optimization (15 Points)** An e-commerce company is preparing orders for delivery. Each order is packed into a box, and the delivery van has a weight capacity of 50 kg. Each box has an associated weight and an urgency score (indicating the priority of delivering the box). The company wants to load the van to maximize the total urgency score of the boxes while staying within the weight limit.

   The available boxes are:

Box	Weight (kg)	Urgency Score
1	10	60
2	20	100
3	30	120

   (a) Determine which boxes should be loaded to maximize the total urgency score, and compute the maximum score. **[5]**

   (b) Due to an unexpected delay, Box 2 must be delivered immediately and cannot be excluded. Recompute the maximum score achievable under this constraint and list the boxes to be included. **[5]**

   (c) How would you extend this approach to handle cases where the urgency scores vary dynamically (e.g., based on customer demand or time sensitivity)? **[5]**

4. **Spell Checker (15 Points)** You are designing a spell-checker for a word processor that suggests corrections for misspelled words. To determine the best suggestions, you need to compute how many edits it takes to transform a misspelled word into a valid dictionary word.

   The allowed edit operations are:

   - **Delete** a character (e.g., transform ‘hte‘ into ‘he‘ by deleting ‘t‘).
   - **Add** a character (e.g., transform ‘hte‘ into ‘hate‘ by adding ‘a‘).

   For example, to correct the misspelled word hte into the:

$$\begin{aligned} \text{hte} &\to \text{he} && \text{(delete t)} \\ \text{he} &\to \text{the} && \text{(add t).} \end{aligned}$$

   This requires 2 edits.

   (a) Write an algorithm to compute the minimum number of edit operations (additions and deletions only) needed to transform a misspelled word u into a valid word v. Your algorithm should also return the sequence of operations (e.g., add or delete). **[5]**

   (b) Given u = “hte” and v = “the”, use your algorithm to:
   - Compute the minimum number of edits needed. **[4]**
   - List the sequence of operations required. **[3]**

   (c) Analyze the time complexity of your algorithm as a function of the lengths of u and v. **[3]**

5. **Longest Repeating Subsequence (10 Points)** A software company is developing a plagiarism detection tool that analyzes text to find repeated patterns within a single document. To help automate this process, you are tasked with finding the **longest repeating subsequence** within a given string.

   A subsequence is a sequence derived from the original string by deleting some or no characters without changing the order of the remaining characters. The repeated subsequence must occur at least twice but cannot reuse the same index of the string.

   Scenario: You are given a document represented as a string S = “AABEBCDD”. Your task is to determine the longest repeating subsequence within this string and explain how you arrived at your answer.

   (a) i. Explain what it means to find a “longest repeating subsequence.” **[1]**

ii. How does this differ from finding the longest common subsequence (LCS) between two different strings? **[2]**

(b) i. Write a dynamic programming recurrence relation for finding the longest repeating subsequence in a string S. Your recurrence should address the following: **[3]**
- When two characters at different indices in the string match.
- When they do not match.

(c) i. Apply your algorithm to the given string $S =$ "AABEBCDD". Construct the DP table and find the length of the longest repeating subsequence. **[2]**

ii. Trace back through the table to identify the subsequence. **[2]**

6. [**Resource Allocation in a Factory(25 points)**] A factory produces two types of products: P_1 and P_2. Each product requires a certain amount of raw materials, and the factory has an unlimited supply of raw materials available. The profit earned for each unit of a product is different, and the factory wants to maximize its profit.

   The production constraints are as follows:

   - P_1 requires 3 units of raw material and yields a profit of 6 per unit.
   - P_2 requires 2 units of raw material and yields a profit of 5 per unit.
   - The factory has 8 units of raw material available in total.

   (a) **Problem Modelling**

   i. Explain how this problem can be modeled as an optimization problem in computer science. **[3]**

   ii. Identify what the "weights," "values," and "capacity" represent in this context. **[3]**

   (b) **Algorithm Design**

   i. Write a recurrence relation for solving the problem. The recurrence should consider the profit of including one or more units of P_1 or P_2 versus excluding them. **[4]**

   ii. Explain why the algorithm needs to account for the fact that multiple units of the same product can be produced. **[5]**

   (c) Use your recurrence relation to compute the maximum profit the factory can achieve with the given constraints. Explicitly show the computation using a table. **[5]**

   (d) **Production Plan Analysis**

   i. Based on your solution, determine how many units of P_1 and P_2 should be produced to achieve the maximum profit. **[3]**

   ii. Explain whether there is more than one way to achieve the maximum profit. **[2]**

Bonus Problems for Extra Credit [70 points]

1. [**Multiple Optimal Solutions (10 points)**] An instance of the knapsack problem may have several different optimal solutions.

   (a) How would you discover multiple optimal solutions for a given instance? **[5]**

   (b) Does the dynamic programming table allow you to recover more than one solution in this case? Explain your approach. **[5]**

2. [**Relaxed Knapsack Problem (10 points)**] In the standard knapsack problem, we assume n items, each of which can either be included ($x_i = 1$) or excluded ($x_i = 0$). Suppose instead that we have n types of items available, with an unlimited supply of each type. Formally, this replaces the binary constraint $x_i \in \{0, 1\}$ with the looser constraint that x_i must be a nonnegative integer.

   (a) Adapt the dynamic programming algorithm to handle this relaxed version of the problem. **[4]**

   (b) Illustrate your approach with a specific example of $n = 3$ types of items and a knapsack capacity $W = 10$. **[6]**

3. **[Enhanced Spell Checker (20 points)]** In the Problem #5 above, you implemented a spell checker that transforms a misspelled word into a valid dictionary word using only *add* and *delete* operations. Now, you want to improve the algorithm by adding a third operation:

   - **Change (or substitute)** a character: Replace one character in the misspelled word u with a different character to match the valid word v.

   This new operation allows transformations to be more efficient in certain cases. For example:

   $$\text{“hte”} \rightarrow \text{“the”}$$

   - Without *change*, this requires two operations: “hte” $\rightarrow$ “he” (delete t) and “he” $\rightarrow$ “the” (add t).
   - With *change*, this requires only one operation: “hte” $\rightarrow$ “the” (substitute t for h).

   (a) The goal is to compute the minimum number of operations needed to transform the first i characters of string u into the first j characters of string v.

   i. Define $dp[i][j]$ as the minimum number of operations required to transform the first i characters of u into the first j characters of v. Using this definition, consider what happens when you:

   A. Delete the i-th character from u: What subproblem does this reduce to? How does it affect $dp[i][j]$? **[2]**

   B. Add a new character to u: What subproblem does this reduce to? How does it affect $dp[i][j]$? **[2]**

   C. Substitute the i-th character of u with the j-th character of v: When do you incur a cost of 1? When is the cost 0? **[2]**

   ii. Combine the above cases into a single recurrence relation. How do you ensure that $dp[i][j]$ represents the minimum number of operations? Write your derived formula. **[4]**

   (b) What values should $dp[0][j]$ and $dp[i][0]$ take? Fill in these values for dp when u = “kitten” and v = “sitting”. **[3]**

   (c) Explicitly compute $dp[i][j]$ for $i = 1, 2, \ldots, 6$ and $j = 1, 2, \ldots, 7$. Step through the process of filling the DP table row by row for u = “kitten” and v = “sitting”. **[4]**

   (d) Once the table is complete, trace back from $dp[6][7]$ to identify the sequence of operations required to transform u into v. For each step, state the operation (add, delete, or change) and the characters involved. **[3]**

4. **[Maximum Profit in Transportation (30 points)]** A logistics company is analyzing a transportation network where nodes represent distribution hubs, and edges represent roads between hubs. Each road has a weight representing the profit earned by transporting goods along that road.

   The company wants to determine the **maximum profit path** between any two hubs, considering all possible routes. You are tasked with designing an algorithm to solve this problem.

   (a) i. Explain what the nodes and edges in the network represent. **[1]**

   ii. What does the weight of an edge represent in terms of profit? **[1]**

   iii. What does finding the “maximum profit path” correspond to in graph terms? **[1]**

   (b) Assume the transportation network is represented as a directed graph $G = (V, E)$, where:

   - V is the set of distribution hubs (nodes).
   - E is the set of roads (edges) with weights $w(u, v)$ denoting profits.

   How would you represent this graph in your algorithm? Discuss the data structures you would use. **[2]**

   (c) The company requires the profit maximization path between **any two hubs**. Modify the Floyd-Warshall algorithm to compute the maximum profit path:

   i. Write the recurrence relation for updating the maximum profit between any two hubs. Consider:

A. If there is a direct road between two hubs i and j, how would the profit be calculated?

B. If a more profitable path exists through an intermediate hub k, how would you update the profit between i and j?

Using these hints, derive a recurrence relation to compute the maximum profit between any two hubs. [**5**]

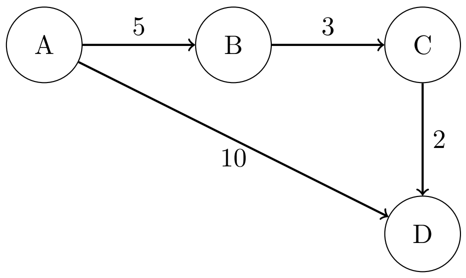
ii. What are the **base cases** for initializing the *dp* table? (Hint: How would you initialize $dp[i][j]$ for direct roads or when $i = j$?) [**3**]

iii. Explain how this recurrence handles:

A. Direct roads between hubs. [**3**]

B. Indirect paths involving intermediate hubs. [**3**]

(d) Use your modified Floyd-Warshall algorithm to solve the following example:



i. Construct the initial *dp* table based on the graph. [**2**]

ii. Fill the table using the recurrence relation. [**5**]

iii. Find the maximum profit path between A and D and B and D. [**4**]