National University of Computer & Emerging Sciences, Lahore
Department of Computer Science (Spring 2025)

CS2009 – Design and analysis of Alogrithms (BCS-6A)

Assignment #3

Course Instructor: Usama Hassan **Due Date: April 22 , 2025**

Course TA: Talha Azhar **Total Marks:150**

Instructions:

1. Submit with your Roll Number (format: 22L-1234_A3.pdf) and hard form in class.
2. Do not submit multiple copies or empty files.
3. It should be clear that your assignment will not get any credit if:
   - The assignment is submitted after the due date.
   - Assignment is plagiarized from any source. (Online/other students/etc.)

Do not panic and remember to take breaks from your device screen while attempting this assignment!

Assignment Overview: efficient algorithms

This assignment focuses on key algorithmic techniques, including Dynamic Programming, Greedy Algorithms, Linear Algorithms, and Huffman Encoding. It aims to develop efficient problem-solving skills by exploring optimization strategies, time-efficient solutions, and data compression methods.

General Guidelines:

- For Task 1, attempt any 6 questions.
- For Task 2, attempt all questions.
- For Task 3, attempt any 6 questions.
- For Task 4, attempt 2 questions.
- Please complete one task questions before moving to next.
- Your Algorithms must be generic.
- Your assignment should be neat in such a way that it is readable.

Task 1: Greedy Algorithms — [60 Marks]

Question 1:
Suppose in Activity Selection Problem instead of always selecting the first activity to finish, we instead select the last activity to start that is compatible with all previously selected activities.

a. Describe how this approach is a greedy algorithm
b. Prove that it yields an optimal solution
c. Write a Greedy Algorithm for it

Question 2:
Suppose you are given a set S = {a1, a2, …… an} of tasks, where task ai requires pi units of processing time to complete, once it has started. You have one computer on which to run these tasks, and the computer can run only one task at a time. Let ci be the completion time of task ai , that is, the time at which task ai completes processing. Your goal is to minimize the average completion time.

Question 3:
Consider the weighted interval scheduling problem. Here the input is a set of n jobs, each with a start time finish time, and reward. Our task is to schedule some of these jobs on a single machine. The output is a non-overlapping subset of the jobs, and the goal is to maximize the total reward from the jobs in the set. Consider following greedy strategy: Sort the jobs by reward and schedule them one by one starting with the highest reward, rejecting any that overlap with jobs already scheduled. Give counter example to prove that it does not guarantee optimal solution.

Question 4:
Given two arrays A and B of equal size n, you have to design an efficient algorithm that minimizes the sum A[1] x B[1] + A[2] x B[2] + … + A[n] x B[n]. You are allowed to shuffle the elements of each array, A and B.
Example: N = 3 A = {3, 1, 1}, B = (6, 5, 4)
Minimum sum = 1 x 6 + 3 x 4 + 1 x 5 = 23
There are other possible ways of taking the sum like 3 x 6+ 1 x4+1 x5 = 27, but the minimum sum is 23.

Question 5:
You have k lists of sorted integers in non-decreasing order. Find the smallest range that includes at least one number from each of the k lists.
We define the range [a, b] is smaller than range [c, d] if b - a < d - c or a < c if b - a == d - c.
Example 1:
Input: nums = [[4,10,15,24,26],[0,9,12,20],[5,18,22,30]]
Output: [20,24]
Explanation:

List 1: [4, 10, 15, 24,26], 24 is in range [20,24].
List 2: [0, 9, 12, 20], 20 is in range [20,24].
List 3: [5, 18, 22, 30], 22 is in range [20,24].
Example 2:
Input: nums = [[1,2,3],[1,2,3],[1,2,3]]
Output: [1,1]

Given N algorithm design problems in an exam with a total exam time of T minutes, each problem has a certain number of checkpoints and marks of a problem are equally distributed for each checkpoint. You are not sure whether you will be able to attempt all the questions or even all the checkpoints of a certain problem. The goal is to maximize the score within the given time constraint. You are given the information of total exam time (T), total number of questions (N), marks and expected time for each problem in arrays M[1.N] and E_Time(1..N] respectively. A student devised an algorithm to maximize the score by prioritizing the problems with maximum marks using a greedy approach. Do you think this greedy approach will always provide an optimal solution. If not, then provide a counter example where this approach will fail. In case of a counter example, you are supposed to follow the same pattern of example input i.e., clearly mention all the information and a proper justification is required which shows that why this greedy approach will fail to provide an optimal solution.
SampleInput:

T=60 minutes N=5 Marks: {30,25,20,15,10} Expected Time: {25, 20, 15, 30, 20} Max Achievable Score = 75

Question 7:

You are tasked with managing a team of superheroes who must complete several missions. Each superhero has a unique skill and a strength rating. Each mission requires a superhero with a specific skill and a strength threshold. Your job is to assign superheroes to missions in a way that maximizes the number of missions completed.

Input:

- n superheroes, where each superhero is represented by:
  - A skill (an integer).
  - A strength rating (an integer).
- m missions, where each mission is represented by:
  - A required skill (an integer).

- A strength threshold (an integer), which is the minimum strength a superhero needs to be assigned to that mission.

Output:

- Your task is to assign superheroes to missions such that:
  1. Each superhero can be assigned to only one mission.
  2. Each mission can have only one superhero.
  3. A superhero can only be assigned to a mission if their skill matches the required skill of the mission and their strength is greater than or equal to the mission's strength threshold.
- Maximize the number of missions that can be completed (i.e., the number of superheroes successfully assigned to missions).

Example:

You are given the following superheroes and missions:

Superheroes:

- (Skill=3,Strength=5)(Skill = 3, Strength = 5)(Skill=3,Strength=5)
- (Skill=1,Strength=10)(Skill = 1, Strength = 10)(Skill=1,Strength=10)
- (Skill=2,Strength=6)(Skill = 2, Strength = 6)(Skill=2,Strength=6)
- (Skill=3,Strength=7)(Skill = 3, Strength = 7)(Skill=3,Strength=7)

Missions:

- (RequiredSkill=3,StrengthThreshold=6)(Required Skill = 3, Strength Threshold = 6)(RequiredSkill=3,StrengthThreshold=6)
- (RequiredSkill=2,StrengthThreshold=5)(Required Skill = 2, Strength Threshold = 5)(RequiredSkill=2,StrengthThreshold=5)
- (RequiredSkill=1,StrengthThreshold=9)(Required Skill = 1, Strength Threshold = 9)(RequiredSkill=1,StrengthThreshold=9)

Expected Output:

- Number of missions completed: 3
- Assignments:
  - Mission (2, 5) → Superhero (2, 6)
  - Mission (3, 6) → Superhero (3, 7)
  - Mission (1, 9) → Superhero (1, 10)

Constraints:

- 1≤n,m≤10^4 (number of superheroes and missions).
- The strength and skill of each superhero, as well as the required strength and skill of each mission, are positive integers and can be at most 10^5

Question 8:

You are given an integer num. You can swap two digits at most once to get the maximum valued number.

Return the maximum valued number you can get.

Example 1:

Input: num = 2736

Output: 7236

Explanation: Swap the number 2 and the number 7.

Example 2:

Input: num = 9973

Output: 9973

Explanation: No swap.

Constraints:

- 0 <= num <= 10^8

Question 9:

Suppose we are given a set of activities/tasks specified by pairs of the start times and finish times as T = {(1, 2), (1, 3), (1, 4), (2, 5), (3, 7), (4, 9), (5, 6), (6, 8), (7, 9)}. Solve the activity selection problem for this set of tasks. Select maximum set of activities that can be scheduled without overlap.

a) How many activities are selected?

b) Which activities are selected?

Task 2: huffman encoding — [10 Marks]

Question 1:

Take a random message with at least 12 characters and of length more than 100 characters. Generate huffman codes.

a) Show the coded message clearly

b) What % benefit it achieved over using ASCII codes

Task 3: Dymanic Programming — [60 Marks]

Question 1: Minimum Sum Partition Problem

Given a set of positive integers, the task is to divide this set into two sets S1 and S2 such that the absolute difference between their sums is minimum.
If there are "N" elements in the original input set "S", then if we assume Subset1 has "M" elements, Subset2 must have N-M elements and the value of abs(sum(Subset1) – sum(Subset2)) should be minimum. There may exists multiple subsets providing the optimal solution.

Example: S[] = {10,20,15,5,25} Consider the following partition for an optimal solution. S1[] = {10,20,5} => s1_sum = 35 S2[] = {15,25} => s2_sum = 40 Abs(s1_sum – s2_sum) = 5	Let's consider another partition for an optimal solution. S1[] = {10,25} => s1_sum = 35 S2[] = {20,15,5} => s2_sum = 40 Abs(s1_sum – s2_sum) = 5 //Both the partitions are providing the same answer.

Question 2:

(a): Longest Increasing Sub sequence

You are given an unsorted array of integers of size "N". Your task is to design an algorithm using dynamic programming to determine the length of longest increasing subsequence of the array. You are not allowed to sort the input array.

Subsequence, Increasing subsequence and longest increasing subsequence?

A subsequence of an array is a list of elements of the array where some elements are deleted (or not deleted at all) and they should be in the same order in the subsequence as in the original array. For example, for the array: [4,2,7], the subsequences will be [{4},{2},{7},{4,2},{4,7},{2,7},{4,2,7}} but {7,4} is not a subsequence because its elements are not in the same order as the original array. The subsequences {4,7} and {2,7} are increasing subsequences so the length of longest increasing subsequence will be 2.

Sample Input Array: [3, 10, 2, 1, 20]

Output: 3 => (3,10,20)

b): Longest Palindromic Subsequence

You are given an input string "str" of length "N". Your task is to determine the length of longest subsequence in a string that is a palindrome. **Remember, we are talking about subsequence not sub-array**.

Example:

Sample Input Array: ABBDCACB
Output: length = 5, longest palindromic subsequence (BCACB)
Sample Input Array: BBABCBCAB
Output: length = 7, longest palindromic subsequence (BABCBAB)

Question 3:
Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.

Example 1:

Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]

Output: 6

Explanation: The above elevation map (black section) is represented by array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are being trapped.

Example 2:
Input: height = [4,2,0,3,2,5]
Output: 9

Constraints:
n == height.length
1 <= n <= 2 * 10^4
0 <= height[i] <= 10^5

Question 4:
Given an input string s and a pattern p, implement regular expression matching with support for '.' and '*' where:

- '.' Matches any single character.
- '*' Matches zero or more of the preceding element.

The matching should cover the **entire** input string (not partial).

Example 1:

Input: s = "aa", p = "a"

Output: false

Explanation: "a" does not match the entire string "aa".

Example 2:

Input: s = "aa", p = "a*"

Output: true

Explanation: '*' means zero or more of the preceding element, 'a'. Therefore, by repeating 'a' once, it becomes "aa".

Example 3:

Input: s = "ab", p = ".*"

Output: true

Explanation: ".*" means "zero or more (*) of any character (.)".

Constraints:

- 1 <= s.length <= 20
- 1 <= p.length <= 20
- s contains only lowercase English letters.
- p contains only lowercase English letters, '.', and '*'.
- It is guaranteed for each appearance of the character '*', there will be a previous valid character to match.

Question 5: Palindrome partitioning:

Given a string **str,** a partitioning of the string is palindrome partitioning if every sub-string of partition is a palindrome, the task is to **find minimum number of cuts needed for palindrome partitioning of the given string**.

Example: str[] = "madamadaseestopspot"
palindromic substrings of given string are => madam| ada | sees | topspot
So minimum 3 cuts are required for palindrome partitioning of the given string.

Question#6: Target Sum Problem

You are given an integer array **nums[1...N]** and an integer target "**K**". The task is to build an expression from the given array where we can place a '+' or '-' sign before of an integer. We want to place a sign before every integer of the array and get our required target. We need to **count the number of ways in which we can achieve our required target**.

Example: nums[] = {1,2,3,1}, K = 3
+1-2+3 +1 = 3 = K
-1+2+3-1 = 3 = K
So there are two possible ways to achieve our target.
// You can think of 0/1 knapsack variant where we need to return all possible solutions to the problem in such a way that there are no remaining items left whose weight is less than the remaining capacity of the knapsack. This problem is pretty much similar to this variant with one key difference. Instead of deciding whether to include an item or not, we now must decide whether to add or subtract an item.

Question 7:

You are given an array prices where prices[i] is the price of a given stock on the i^{th} day.
Find the maximum profit you can achieve. You may complete as many transactions as you like (i.e., buy one and sell one share of the stock multiple times) with the following restrictions:

- After you sell your stock, you cannot buy stock on the next day (i.e., cooldown one day).

Note: You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy again).

- **Example 1:**
  Input: prices = [1,2,3,0,2]
  Output: 3
  Explanation: transactions = [buy, sell, cooldown, buy, sell]
- **Example 2:**
  Input: prices = [1]
  Output: 0

Constraints:

- 1 <= prices.length <= 5000
- 0 <= prices[i] <= 1000

Question 8:

We have studied various dynamic programming algorithms to find optimal solutions like (rod cutting, longest common subsequence, coin change etc.) The optimal solution may not be unique. For example, there can be multiple longest common subsequences.

a) Write an efficient dynamic programming algorithm that **counts the number of different optimal solutions for the rod cutting problem** given the rod length "N" and an array of prices "Pr[1…N]" having prices of various length of rod.

b) Consider a modification of the rod-cutting problem in which, in addition to a price p[i] for each rod, each cut incurs a fixed cost of c. The revenue associated with a solution is now the **sum of the prices of the pieces minus the costs of making the cuts**. Give a dynamic-programming algorithm to solve this modified problem.

Task 4: Linear Problems — [20 Marks]

Question 1:
Given an array A of n numbers in the range of(-n, n), devise an algorithm that finds the number of distinct pairs (i,j) such that j>I and A[i]=A[j]. your Algorithm must work in O(n) time.

Sample input: [1,2,1,2,3,1,4]
Sample output: 4
Sample input: [-1,-1,-1,-1,4]
Sample output: 6
Note: you are not allowed to use hashMap.

Question 2:
Given an integer array nums, return the maximum difference between two successive elements in its sorted form. If the array contains less than two elements, return 0.
You must write an algorithm that runs in linear time and uses linear extra space.

- **Example 1:**
  Input: nums = [3,6,9,1]
  Output: 3
  Explanation: The sorted form of the array is [1,3,6,9], either (3,6) or (6,9) has the maximum difference 3.

- **Example 2:**
  Input: nums = [10]
  Output: 0
  Explanation: The array contains less than 2 elements, therefore return 0.

Constraints:
- $1 <= nums.length <= 10^5$
- $0 <= nums[i] <= 10^9$

Hint : think of pigeonhole principle

Question 3:

Given an array of strings words and an integer k, return *the* k *most frequent strings*.
Return the answer sorted by the frequency from highest to lowest. Sort the words with the same frequency by their lexicographical order.

Example 1:

Input: words = ["i","love","leetcode","i","love","coding"], k = 2
Output: ["i","love"]
Explanation: "i" and "love" are the two most frequent words.

Note that "i" comes before "love" due to a lower alphabetical order.

Example 2:

Input: words = ["the","day","is","sunny","the","the","the","sunny","is","is"], k = 4
Output: ["the","is","sunny","day"]
Explanation: "the", "is", "sunny" and "day" are the four most frequent words, with the number of occurrence being 4, 3, 2 and 1 respectively.

BONUS: Attempting all questions will land you bonus points!!!

Good Luck and Happy Coding :)