CS6013: Advanced Data Structures and Algorithms

Programming Assignment II (out of 10 marks)
(Start Date: 12 October 2021)
(Submission Deadline: 11:59 pm, Sunday, 24 October 2021)

0.1 Problem Description

Vijay is a contestant in the 'Kaun Banenga Master Chef' (KBMC) program. KBMC organizers have published a list of n dishes, namely $D_1, D_2, \ldots, D_n$. They have also published the amount of points per dish. Preparing a dish D_i would fetch a participant p_i points, where $1 \leq i \leq n$. Vijay knows the amount of time he needs to prepare each dish. He needs m_i minutes to prepare Dish D_i, where $1 \leq i \leq n$. The KBMC contest is conducted in the following manner. Each participant is given M minutes. He/she who wins the maximum number of points in the given M minutes is the winner. There is no part marking. That is, a participant gets p_i points for a dish D_i only if he/she prepares it fully. Otherwise, he/she gets zero points for that dish.

Vijay's primary goal is to make those dishes so that the total number of points gained is maximized. He also wants to do this with minimal effort, which is his secondary goal (as this is only the first round, which is not an elimination round, of the KBMC competition and he needs to reserve his energy for subsequent rounds). Please help Vijay achieve his goals.

Part 1: In this part, assume that all points are equal, that is, $p_1 = p_2 = \cdots = p_n$. Therefore, the primary goal of Vijay reduces to making as many dishes as possible. The obvious greedy strategy of preparing the easiest dish first works in this case, and addresses his secondary goal too.

Write an $O(n \log n)$-time HEAPSORT sorting algorithm (do not use any built-in library function) to sort the array $m_1, m_2, \ldots, m_n$ in a non-decreasing order. Keep on cooking dishes from the easy end until M minutes are over. Print this optimal solution. Write a function *solveGreedy*() for solving this part.

Part 2: Now, assume that the points array $p_1, p_2, \ldots, p_n$ contains arbitrary integer values. There need not be any correlation between the efforts and the points. KBMC evaluators may have rated the dishes using metrics other than raw effort (like novelty, taste, nutrition value, and so on). Greedy strategies do not work now. Dynamic programming helps you instead.

Let the total points for all the n dishes be

$$P = p_1 + p_2 + \cdots + p_n.$$

Build a two-dimensional table T of size $(n+1) \times (P+1)$ such that $T[i][p]$ is intended to store the minimum possible effort to attempt exactly p points from the dishes $D_1, D_2, \ldots, D_i$. If exactly p points cannot be achieved from $p_1, p_2, \ldots, p_i$, you should store $T[i][p] = \infty$.

First, notice that $T[i][0] = 0$ for all i (irrespective of the number of dishes, if Vijay does not prepare any dish, his attempted point is 0). Another boundary condition pertains to $i = 0$, which means that no dish is considered. Therefore, $T[0][0] = 0$, whereas $T[0][p] = \infty$ whenever $p > 0$.

Now, take $1 \leq i \leq n$ and $1 \leq p \leq P$. If $p < p_i$, Vijay cannot attempt the i-th dish, so $T[i][p] = T[i-1][p]$. If $p \geq p_i$, Vijay has two possibilities: if he does not prepare the i-th dish, then take $M_0 = T[i-1][p]$, whereas if he prepare the i-th dish, then take $M_1 = m_i + T[i-1][p-p_i]$. If $M_1 > M$, the second option is not valid, so set $T[i][p] = M_0$, otherwise set $T[i][p] = \min(M_0, M_1)$.

Write a function solveDP() to build the table T using the approach just mentioned. The maximum achievable point is the largest p such that $T[n][p] \neq \infty$. To achieve this p, the minimum effort needed is $T[n][p]$.

Part 2 (Extended): Augment the function solveDP() of Part 2 to compute a choice of the dishes achieving the optimal solution. Print the solution (both effort-wise and point-wise breakups). Do not write a new function. Just update the function solveDP().

The main() function

- Read n (the number of dishes) and M (the total time Vijay has) from the user.
- Read only the efforts $m_1, m_2, \ldots, m_n$ from the user. Assume that $p_1 = p_2 = \cdots = p_n$ (the exact value need not be specified). Call *solveGreedy*() to find and print the greedy solution of Part 1.
- In Part 1, you have already sorted the effort array. In Part 2, you should work with an arbitrary array. So re-read a new set of efforts $m_1, m_2, \ldots, m_n$ from the user. Also re-read the points $p_1, p_2, \ldots, p_n$ from the user. Call solveDP() to compute the optimal solution in this case. Print the solution (maximum point and minimum effort). After you augment the function as specified in Part 2 (Extended), the point-wise and effort-wise breakups for the optimal solution are also printed.

Sample Output:
$n = 10$
$M = 100$
+++ Efforts : 23 23 28 28 13 5 21 25 31 5
+++ Part 1 (Greedy)
Minimum effort $= 90 = 5 + 5 + 13 + 21 + 23 + 23$
+++ Points : 8 2 7 4 5 3 8 2 2 10
+++ Efforts : 15 9 22 25 25 30 26 4 29 39
+++ Part 2 (DP)
Maximum points $= 30 = 8 + 7 + 5 + 8 + 2$
Minimum effort $= 92 = 15 + 22 + 25 + 26 + 4$

0.2 Program Related Instructions

1. You can write your program in one of C, C++, Java, or Python.

0.3 Submission Guidelines

1. Your submission will be one zip file named <roll-number>.zip , where you replace roll-number by your roll number (e.g. cs20mtech11003.zip), all in small letters. The compressed file should contain the below mentioned files:
   (a) Programming files (please do not submit python notebooks or IDE files). **The entire source code has to be in one file named main_prog.c (or main_prog.cpp, or ...).**
   (b) **No need to submit a report**. However, if you wish you may submit a text/doc file giving a detailed description of your program. No marks for this.
   (c) Upload your zip file in Google Classroom at Classwork→Week 9→Assignment 2. No delays permitted.
2. Failure to comply with instructions (file-naming, upload, input/output specifications) will result in your submission not being evaluated (and you being awarded 0 for the assignment).
3. **Plagiarism policy**: If we find a case of plagiarism in your assignment (i.e. copying of code, either from the internet, or from each other, in part or whole), you will be awarded a zero and will lead to a FR grade for the course in line with the department Plagiarism Policy (https://cse.iith.ac.in/academics/plagiarism-policy.html). Note that we will not distinguish between a person who has copied, or has allowed his/her code to be copied; both will be equally awarded a zero for the submission.

0.4 Evaluation Scheme

Your assignment will be awarded marks based on the following aspects:

- Code clarity (includes comments, indentation, naming of variables and functions, etc.): 1 mark
- Perfect output (includes Part 1, Part 2 and its extension): 2 mark
- Logic in the code in Part 1: 2 (HEAPSORT) + 1 = 3 marks.
- Logic in the code in Part 2: 2 marks.
- Logic in the code in Part 2 (Extended): 2 marks.