EECS4020 Design and Analysis of Algorithms

Homework 3

(On a voluntary basis: No need to submit)

You are encouraged to work in groups. Please submit your solution (of any question) to our tutors on or before April 28, 2022, so that they may have a chance to give feedback of your work.

I. Dynamic Programming

1. Consider a chessboard of size $k \times n$. How many ways are there to cover the chessboard completely with n rectangular bars, each of size $k \times 1$ or $1 \times k$?

   For instance, when $k = 2$, $n = 3$, there are three different ways:

   (a) cover the leftmost column by a vertical bar, and the remaining region by two horizontal bars;

   (b) cover the rightmost column by a vertical bar, and the remaining region by two horizontal bars; or

   (c) cover each column with a vertical bar.

   Design an $O(n)$-time algorithm to compute the desired answer for any input k and n.

2. Let $S = \langle s_1, s_2, \ldots, s_n \rangle$ be a sequence of n distinct integers. We hope to find a longest subsequence of S such that the values in the subsequence are increasing.

   For instance, if $S = \langle 3, 2, 4, 1, 5 \rangle$, then $\langle 2, 4, 5 \rangle$ is one of the longest subsequence whose values are increasing.

   Design an $O(n^2)$-time algorithm to compute one of the longest increasing subsequences.

   Remark 1: This problem can be solved by using the same algorithm as finding LCS.

   Remark 2: However, it can be solved in $O(n \log n)$ time with a different DP, together with help from some suitable data structure.

3. Peter is an owner of a Japanese sushi restaurant and today he invites you for dinner at his restaurant. In front of you are n sushi dishes that are arranged in a line. All dishes are different and they have different costs.

   Peter hopes that you can select the dishes you want, starting from the left to the right. However, there is a further restriction: when you select a dish, say A, the next dish you can select must cost higher than A.

   Design an $O(n^2)$-time algorithm to select the dishes so as to maximize the total costs.

4. There is a city called `Trellisland`. The shape of the city looks like a tree, such that each node in the tree has a building, and each edge in the tree is a road connecting two buildings. In total, there are n buildings.

   As Universiade, the international university sports competition, is to be held in Trellisland next year, the police chief wants to make sure that the city is safe. His plan is to place officers around, so that for each edge, one or both of its endpoints will be guarded by an officer.

Design an $O(n)$-time algorithm to place the minimum number of officers so that each edge is guarded.

Remark: This problem can be solved by DP or a greedy algorithm.

5. Tony wants to open stinky tofu stores in Trellisland. According to the city rules, each building can have at most one store, and each store cannot be adjacent to each other on the same road (otherwise, it becomes *too* stinky).

   Tony has done some preliminary survey, and can now tell the number of customers who will visit each building if a store is open there. Now, your target is to help Tony decide where to open the stores so as to maximize the total number of customers.

   Design an $O(n)$-time algorithm to achieve the above target.

6. After a lot of hard work, you have successfully helped Tony to open the stinky tofu stores. Tonight, you are going to have a big dinner at a buffet restaurant.

   The restaurant serves n distinct types of dishes, each has a different volume (which is an integer). It is obvious that your stomach capacity, V (which is also an integer), is not large enough to hold all the dishes. Also, some dish looks expensive and some looks so-so, so that each dish has a different value to you.

   You decide the following strategy: (i) Never take the same dish twice, and (ii) do not overeat (so that total volume is at most V).

   Design an $O(nV)$-time algorithm to select the dishes so as to maximize the total values.

II. Greedy Algorithm

1. John wants to drive from Hsinchu to Hualien using his car. His car's gas tank, when full, holds enough gas to travel exactly n km. John has already made up his route, and his only concern now is to plan where to refill his gas tank along the way.

   Suppose that there are k gas stations $x_1, x_2, \ldots, x_k$ along the route, and the distance between any two of them are known. Also, for any two consecutive gas stations, the distance is at most n km. To save time, John wants to stop for as few gas stations as possible.

   Give an efficient method to find out the gas stations for John to stop, and show that your method yields an optimal solution.

2. Consider n points located on the y-axis, with x-coordinates $x_1 < x_2 < \cdots < x_n$. We want to cover these points using unit-length line segments along the line $y = 0$. For instance, if we have points $1.1, 1.5, 7.5$, then we can cover these points using two unit-length line segments (say, $[1, 2]$, and $[6.5, 7.5]$).

   Give an $O(n)$-time algorithm to find out the minimum number of unit-length line segments we need to cover all the points.

3. Consider n items $s_1, s_2, \ldots, s_n$. You want to pack the items using bags, but each of your bags can hold a total weight of W. (Here, we assume that the weight of each item is at most W.) To minimise the number of bags, you try the following strategy:

> Start with an empty bag B_1. Set $i = 1$.
>
> **while** (there is an item s not in the bags)
>
> **if** (any of the bags $B_1, B_2, \ldots, B_i$ can hold s)
>
> Put s in that bag;
>
> **else**
>
> Put s in a new empty bag B_{i+1}, and then update i as $i + 1$.

   (a) Show that the above strategy does not necessarily give an optimal result (i.e., it may use more the the minimum number of bags).

   (b) Show that in the above strategy, if we use m bags, then at least $m - 1$ bags will be more than half full.

   (c) Let OPT denote the minimum number of bags to pack the items. Using the result of (b), show that $OPT > (m - 1)/2$, and thereby conclude that $m \leq 2 \cdot OPT$.

   Remark: The above discussion indicates that although the greedy heuristic does not give an optimal result, it still gives a result within a factor of 2 from the optimal.

4. There are n children in a nursery school. There are n toys for them to share and play. Today, there is a charity event, and the principal decides to donate some of these toys. In order not to break the heart of the children, the principal asks each of the children what his or her top three favorite toys are. The principal promises to the children that for each child, at least one of the top three favorite will not be donated.

   Design an $O(n)$-time greedy algorithm to select which toys to donate, so that the number of donated toys is at least $n/3$.

   Remark: The algorithm does not need to find the maximum number of toys that can be donated. It just indicates that the maximum must be at least $n/3$.