Sunday, August 29, 2021 5:57 AM

COL 351 : Analysis and Design of Algorithms

Tutorial Sheet - 5

Question 1 You are given a server and a collection of n jobs numbered 1 to n, wherein the i^{th} job has a start time s_i and a termination t_i. Further, for $i \in [1, n]$, scheduling job i on the server generates a revenue R_i. Your task is to find a subset $\mathcal{I}$ of non-overlapping jobs for which $\sum_{i \in \mathcal{I}} R_i$ is maximized.

Solution: Sort the jobs in non-decreasing order of their termination time. We will maintain two arrays of size n, namely OPT and P satisfying:

$$OPT(j) = \text{the optimal solution considering only jobs } 1, \ldots, j.$$
$$P(j) = \text{the largest } i < j \text{ such that job } i \text{ doesn't overlap with job } j.$$

Now if you have computed the first $j-1$ entries of OPT and P, then the j^{th} entry can be computed as follows:

- To compute $P(j)$ we just scan the jobs in increasing order of their finish time and check whether or not they overlap with $[s_j, t_j]$.
- The value of $OPT(j)$ will then be $\max\{OPT[j-1],\ R[j] + OPT(\ P[j]\ )\}$.

To find the set $\mathcal{I}$, you can do backtracking.

1

Question 2 You are given a checker-board that has 4 rows and n columns, and with an integer written in each square. You are also given a set of $2n$ pebbles, and your task is to place some or all of these pebbles on the checker-board (each pebble can be placed on exactly one square) so as to maximize the sum of the integers in the squares that are covered by pebbles. There is one constraint: for a placement of pebbles to be legal, no two of them can be on horizontally or vertically adjacent squares (diagonal adjacency is fine). Give an $O(n)$ time algorithm to find an optimal placement of the pebbles.

Solution: There are 8 ways to put pebbles in a column as below:

$$\mathcal{C} = \{1010,\ 1001,\ 1000,\ 0101,\ 0100,\ 0010,\ 0001,\ 0000\}$$

For a configuration $c \in \mathcal{C}$, let $\pi(c)$ be a subset of all those configurations $c' \in \mathcal{C}$ for which, placing pebbles in accordance to configurations c, c' in two adjacent columns (one on left, other on right) does not violate any rules.

Make a 2D table T of size $n \times 8$, where for $i \in [1, n]$ and $c \in C$, the value $T[i, c]$ is the optimal cost for placing for placing pebbles in first i columns under the constraint i^{th} column is arranged according to configuration c. Then,

$$T[i, c] = value(i^{th} \text{ column according to configuration } c) + \max_{c' \in \pi(c)} T[i-1, c'].$$

The optimal arrangement of pebbles can be obtained by backtracking on T.

Question 3 For a pair of strings $X = (x_1, \dots, x_n)$, $Y = (y_1, \dots, y_m)$, a string $Z = (z_1, \dots, z_{m+n})$ is said to to be an interleaving of X and Y if the indices of Z can be partitioned into two sets I, J respectively of size n, m, such that restriction of Z to I gives X and restriction of Z to J gives Y. For example, if $X = (1010)$ and $Y = (0011)$, then $Z = (10001101)$ is an interleaving because the restriction of Z to odd indices gives X, and restriction of Z to even indices gives Y. Provide an $O(mn)$ time algorithm to verify whether Z is an interleaving of X and Y.

Solution: Make a 2D table T of size $n \times m$, where for $i \in [1, n]$ and $j \in [1, m]$, the value $T[i, j]$ tells whether or not $(z_1, \dots, z_{i+j})$ is an interleaving of $(x_1, \dots, x_i)$ and $(y_1, \dots, y_j)$. Then, for $i, j \geq 2$,

$$T[i, j] = \begin{cases} False & x_i \neq z_{i+j} \ and \ y_j \neq z_{i+j} \\ T[i-1, j] & x_i = z_{i+j} \ and \ y_j \neq z_{i+j} \\ T[i, j-1] & x_i \neq z_{i+j} \ and \ y_j = z_{i+j} \\ T[i-1, j] \vee T[i, j-1] & otherwise \end{cases}$$

The solution for the scenario $i = 1$ or $j = 1$ is left as an exercise. The final output will be $T[n, m]$.

Question 4 A town has n residents labeled $1, \ldots, n$ living on a straight road. It has decided to open k covid test centers along this road. The goal is to minimize the sum total of distance that all the residents need to travel to get to their nearest testing center. As input, you are given $n,\ k$, and an array A of size n, where $A[i]$ is the location of resident i. You can assume A has integral entries and is sorted in non-decreasing order. Your task is to compute a k-sized integer array C of locations such that the following sum is minimized:

$$\sum_{i=1}^{n} d_i, \text{ where, } d_i = \min_{j \in \{1,\ldots,k\}} |A[i] - C[j]|.$$

Solution: We first define the concept of *median*.

Definition: Given K real numbers $x_1, \ldots, x_L$, the ***median*** of $x_1, \ldots, x_L$ is a real number y satisfying $\sum_{i=1}^{L} |x_i - y|$ is minimized.

Fact (exercise, try on simple examples): If real numbers $x_1, \ldots, x_L$ are sorted in non-decreasing order then $y = x_{\lfloor L/2 \rfloor}$ and $y = x_{\lceil L/2 \rceil}$ are *medians* of $x_1, \ldots, x_L$.

Make a 2D table T of size $n \times k$, where for $i \in [1, n]$ and $j \in [1, k]$, the value $T[i, j]$ denotes the minimum sum total distance traveled for the sub-problem with first i residents if j testing centers are to be opened.

Note that is the j center positions were already decided, then the residents can be partition into j intervals such that for the first interval first covid-test center is closest, for the second interval second covid-test center is closest, and so on. Based upon this observation, the recursive approach to build table T is as follows:

$$T(i,j) = \min_{\alpha \in [1,i]} \Big(T(\alpha - 1, j - 1) \ + \ \sum_{r=\alpha}^{i} \Big| A[r] - A\big[\lceil \tfrac{\alpha + i}{2} \rceil\big] \Big| \Big)$$

In the above formulation for having j centers for first i resident, we are trying all possibilities for the last interval, that is, corresponding to j^{th} test center. For the optimal choice of α, the median of resident locations $A[\alpha], \ldots, A[i]$, that is, $A\big[\lceil \frac{\alpha+i}{2} \rceil\big]$ is the optimal position for j^{th} test center.