CS218 Design and analysis of algorithms **Jan-Apr 2023**

Midsem exam

Total Marks: 60 *Time: 120 minutes*

Instructions.

- Please write your answers concisely.
- Describe the steps in your algorithms using English text. Use pseudocode only when necessary.
- Explain the reasoning behind the steps in your algorithm, whenever it's not obvious.

Que 1. Suppose we have a directed graph $G(V, E)$ with a designated source vertex s and destination vertex t. We want to find out if there is a path from s to t which has exactly k edges. By definition, a path cannot have repeated vertices.

(a) [1 marks]. Let us first find a shortest path from s to t. Suppose it has ℓ edges. Can we say that if $\ell \leq k$, then the answer to the above question is yes?

(b) [3 marks]. For any vertex v and for any number i, let us define a Boolean variable path(v, i), which is supposed to be True if and only if there is a path from v to t with exactly i edges. As the base case we take path$(t, 0)$ to be True. We write the following procedure to compute path(v, i):

For each vertex u such that there is an edge from v to u, we check path$(u, i-1)$. If any of them is True then we set path(v, i) as True. Using this logic, we compute path(v, i) for all pairs (v, i). We finally output path(s, k). This algorithm is incorrect. Explain why.

(c) [1 marks]. Do you think there can be a polynomial time algorithm for this problem?

Que 2. Recall the definition of Fourier transform. Let ω be a primitive d-th root of unity. For a given length-d tuple $(a_0, a_1, \ldots, a_{d-1})$, we want to compute d values, i.e.,

$$a_0 + a_1\omega^i + a_2\omega^{2i} + \cdots + a_{d-1}\omega^{(d-1)i},$$

for each $0 \leq i \leq d-1$. We had discussed a divide and conquer approach for fast Fourier transform. It is possible to do an iterative implementation of that approach, which uses only $O(d)$ size memory. For simplicity assume $d = 2^r$. Consider the following pseudocode:

```
1. Let A be a length d array initialized as [a_0, a_1, ..., a_{d-1}].
2. for k = 1 to r {
3.     for i = 0 to (2^{r-k} - 1) {
4.         for j = 0 to (2^{k-1} - 1) {
5.             define index_1 = i × 2^k + j and index_2 = index_1 + 2^{k-1}.
6.             Initialize two temporary variables temp_1 and temp_2.
7.             Compute temp_1 and temp_2 using A[index_1] and A[index_2].
8.             Copy temp_1 into A[index_1] and temp_2 into A[index_2].
9.         }
10.    }
11. }
```

(a) [3+3 marks]. Describe how will you compute $temp_1$ and $temp_2$ in line 7.

(b) [3 marks]. What is the time complexity of this algorithm, in terms of parameter d? Explain briefly. Assume basic arithmetic operations are unit cost.

Que 3. You are going on a car trip from city A to city B that will take multiple days. On the way, you will encounter many intermediate cities. You plan to drive only during the day time and on each night you will stay in one of the intermediate cities. Suppose you can drive at most d kilometers in a day. Imagine the cities are located on the x-axis. City A is located at $x = 0$. The intermediate cities are located at $x_1, x_2, \ldots, x_{n-1}$ and city B is located at x_n. Here $0 < x_1 < x_2 < \cdots < x_n$ are given in kilometers. You are also given the costs of staying in the intermediate cities for one night, say, these are $c_1, c_2, \ldots, c_{n-1}$, respectively.

(a) [2 marks]. If you want to minimize the total number of stays in the intermediate cities, what will be your strategy?

Suppose you want to find a travel schedule, that is, in which all cities you should do a night stay, such that your total cost of staying is minimized.

(b) [2 marks]. Show an example, where the following greedy strategy will **not** give an optimal solution: consider all the upcoming cities which are within distance d from the current city. Among these choose the one with cheapest cost. In the example, you should specify the values of n, d, c_i's, x_i's.

(c). Design a dynamic programming algorithm for finding the optimal cost.

- **[3 marks]** What will be a good way to categorize all possible solutions into multiple categories. Describe how the optimal solution from a particular category can be found by solving a subproblem.
- **[2 marks]** Relate the optimal cost for the given problem to the optimal costs of these subproblems.
- **[3 marks].** In the iterative implementation, you might want to store the optimal costs for different subproblems into an array. What the i-th entry of this array should be and what is the length of the array. Describe how the i-th entry will be computed using other entries of the array and in which order you will compute the entries of this array. How will you get the optimal cost for the given problem from this array?
- **[4 marks].** Give a pseudocode that outputs the optimal solution, i.e., the list of cities where you should stay.

Que 4. Suppose you have n empty cylindrical containers in your kitchen. Let (H_i, A_i) be the height and the base area of the ith container, respectively. A container i fits inside container j if and only if $H_i < H_j$ and $A_i < A_j$. You are shifting to a new house. To save space, you want to pack these containers inside one another as much as possible. To be precise, you want to minimize the number of piles you make. For example, suppose we have five containers with heights and areas $(2, 10), (8, 30), (5, 20), (6, 15), (10, 25)$. One way to arrange these containers into two piles is shown in Figure 1.

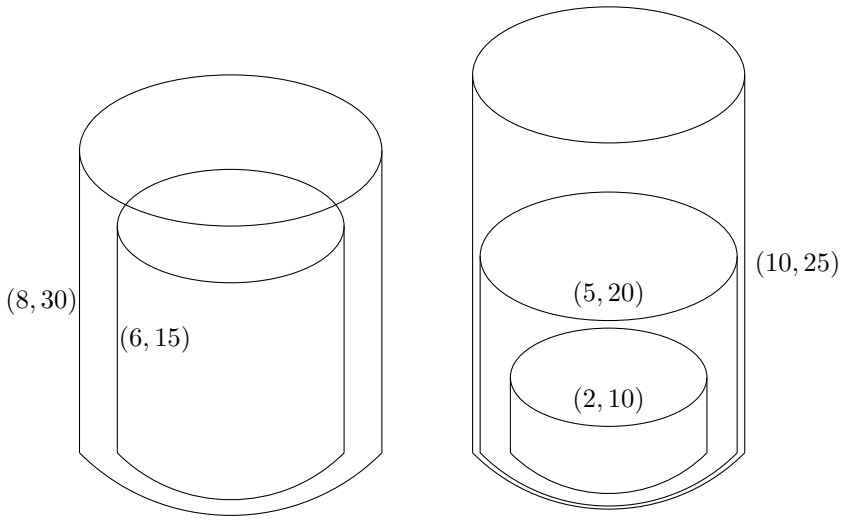

Figure 1: Containers arranged in two piles

We propose the following greedy approach. For simplicity, assume that all areas and heights are distinct. Sort the containers in a decreasing order of heights. Let the containers be $C_1, C_2, \ldots, C_n$, in this order. Now, we will put the containers one by one into the piles. Suppose for the first $i-1$ containers, we have arranged them into k piles. Let $A_{j_1}, A_{j_2}, \ldots, A_{j_k}$ be the areas of the inner most containers in the k piles. In Figure 1, these numbers are 15, 10. Now, clearly the incoming container C_i has height smaller than any of first $i-1$ containers (because of initial sorting). But, its area can be arbitrary. If the ith container does not fit into any of the current piles (i.e., $A_i > \max\{A_{j_1}, A_{j_2}, \ldots, A_{j_k}\}$) then we create a new pile and put the ith container there.

(a) [4 marks]. If C_i fits into many of the current piles, then in which pile will you choose to put C_i?

- Among the piles where C_i fits, choose the one which has the least number of containers (in Figure 1, if the next container fits into both the piles then, this strategy will choose pile 1).
- Among the piles where C_i fits, choose the one which has the most number of containers (in Figure 1, if the next container fits into both the piles then, this strategy will choose pile 2).
- Among the piles where C_i fits, choose the one whose inner most container has the largest area (in Figure 1, if the next container fits into both the piles then, this strategy will choose pile 1).
- Among the piles where C_i fits, choose the one whose inner most container has the smallest area (in Figure 1, if the next container fits into both the piles then, this strategy will choose pile 2).

(b) [4 marks]. Argue the correctness of your greedy choice. You can argue like this. Suppose the optimal arrangement of the n containers agrees with your strategy on the arrangement of containers $C_1, C_2, \ldots, C_{i-1}$, but does not agree on the arrangement of container C_i. Then you can construct another optimal arrangement of n containers which will agree with your strategy on the arrangement of $C_1, C_2, \ldots, C_i$.

(c) [3 marks]. Argue that with the use of an appropriate data structure, the algorithm takes only $O(n \log n)$ time.

(d) [6 marks]. This part is challenging. Suppose your chosen greedy algorithm outputs ℓ piles at the end. Show that you can find a set S of ℓ containers, one picked from each pile, such that no container in S fits into another container in S. For example, if the piles were

- Pile 1: (5,20), (2,10)
- Pile 2: (4,50), (1,40)
- Pile 3: (3,30),

then we can pick containers (5,20), (1,40), and (3,30) in our set S. None of the three containers can fit into any other of the three.

(e) [3 marks]. Argue that among the given n containers, the largest possible subset where no container fits into another has size ℓ.

Que 5. Suppose there are n objects with their weights being $w_1, w_2, \ldots, w_n$ and their costs being $v_1, v_2, \ldots, v_n$. You want to select a subset of the objects such that the total weight is at most W, while the total cost is maximized. We had discussed an algorithm which takes time $O(nW)$. However, in our setting W is very large and this is undesirable. Design an algorithm to find the optimal cost, which takes time $O(nV)$, where $V = \sum_i v_i$. Assume that addition, subtraction, comparison of weights and costs are unit cost operations.

- **[4 marks].** Your dynamic programming solution might create an 2 dimensional array of size $(n \times V)$. What the (i, j) entry is supposed to compute?
- **[3 marks].** Describe how the (i, j) entry will be computed using other entries of the array.
- **[3 marks].** How will you find the optimal cost from this array.