Practice Problems 7

Problem 1. A pharmacist has W pills and n empty bottles. Let p_i denote the number of pills that can fit in bottle i, and let c_i denote the cost of purchasing bottle i. Given W, p_i's and c_i's, we wish to compute the cheapest subset of bottles into which to place all W pills. (You may assume that $\sum_i p_i \geq W$, so there is a feasible solution.)

Whenever a bottle contains even a single pill, the pharmacist must pay the *full price* of the bottle (see Fig. 1).

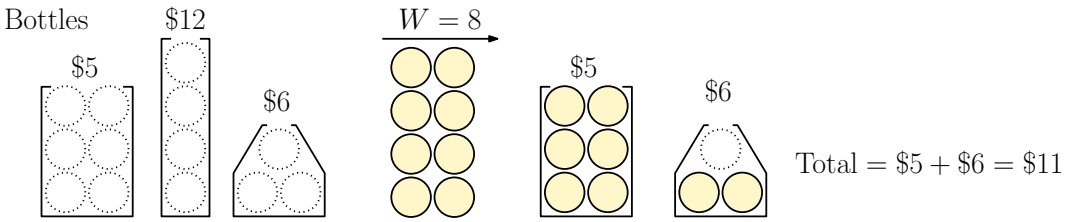

Figure 1: Filling bottles.

(a) There are two natural greedy approaches. One is to fill the bottles in increasing order of cost c_i, and the other is to fill them in increasing order of the ratio c_i/p_i, which intuitively represents the per-pill cost of each bottle. Present a *single* counterexample that shows that neither of these greedy solutions is optimal. (Try to make your counterexample as simple as possible.)

(b) Present an algorithm for this problem. Justify your algorithm's correctness and derive its running time. (Hint: Use DP. It suffices to give the recursive formulation. Aim for a running time of $O(nW)$.)

(c) Suppose that there is an infinite supply of each bottle. Explain how to modify your answer from (b) for this variant.

Problem 2. Work through the chain-matrix multiplication algorithm on a sequence of matrices $A_1 \cdot A_2 \cdot A_3 \cdot A_4$, where matrices are of dimensions 2×2, 2×5, 5×3, and 3×1, respectively (see Fig. 2). Following the algorithm from Lecture 10, present both the M and H matrices and show the final multiplication order.

Show your matrices in the rotated form that is used in Figure 5 (page 5) of Lecture 10. You may present the final multiplication order as a tree or by adding parentheses to "$A_1 \cdot A_2 \cdot A_3 \cdot A_4$".

Problem 3. You are given a set of n points $P = \{p_1, \ldots, p_n\}$ in 2-dimensional space sorted by x-coordinates. Let $p_i = (x_i, y_i)$. An *increasing subsequence* is any subsequence of P such that the y-coordinates are strictly increasing. The *longest increasing subsequence* (LIS) is the increasing subsequence having the largest number of points. For example, the LIS for the set of points in Fig. 3 has length 5, consisting of the subsequence $\langle p_2, p_4, p_6, p_7, p_8 \rangle$.

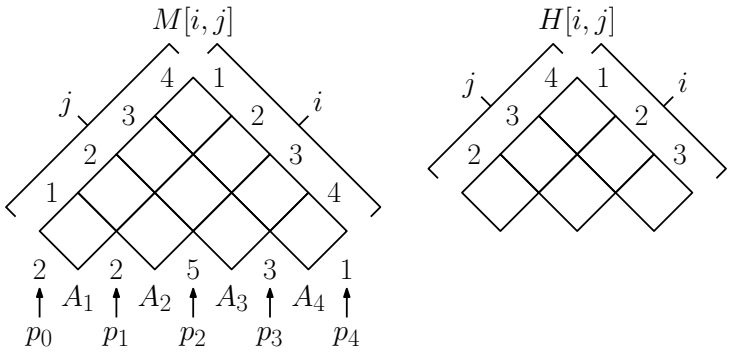

Figure 2: Chain-matrix multiplication.

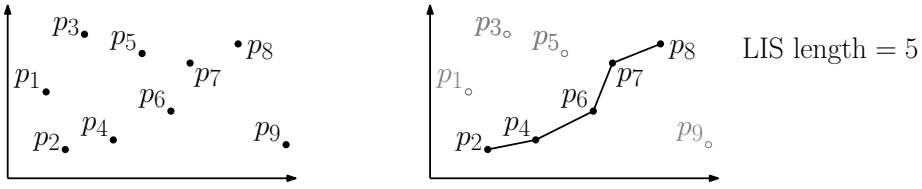

Figure 3: Computing the length of the maximum LIS.

(a) Give a recursive DP formulation for solving this problem. (**Hint:** For $0 \leq i \leq n$, let $\text{lis}(i)$ denote the length of the LIS that ends with point p_i. You may assume that the points are given to you in increasing order of x-coordinates, and there are no duplicate x- or y-coordinates. Be sure to include the basis case, and how to obtain the final answer.)

(b) Express your recursive formulation in pseudocode (either recursively or iteratively). For full credit, it should run in $O(n^2)$ time.

Problem 4. A popular game show has come to campus, and you have been selected to be one of the participants. (Lucky you!) The host of the show explains how the game works. There is a designated starting point in the end-zone of the football field, and a number of \$100 bills have been placed throughout the field. You are to run and pick up all the bills as fast as you can and then return to the starting point. You get to keep the money if you are the fastest contestant. Therefore, you want to compute a path that hits all the bills and is as short as possible.

There is one additional constraint. Your path must consist of two parts, an *outward path* where you run only to the right (x-coordinates increasing) from the starting end-zone, and an *return path* where you run only to the left (x-coordinates decreasing) (see Fig. 4).

The input consists of a set of points $P = \{p_1, \ldots, p_n\}$ where the bills are and a start point p_0. Each point is given by its (x, y)-coordinates, $p_i = (x_i, y_i)$. You may assume that the points are sorted from left to right (that is, by increasing x-coordinates). You may assume you have access to a function $\text{dist}(p_i, p_j)$, which computes the distance between any two points.

Give an efficient algorithm that computes the length of the shortest path that hits all the bills and has the desired outward-return structure. (Hint: Use DP, working from left to right. Build both paths, outward and return, simultaneously. $O(n^2)$ time is possible.)

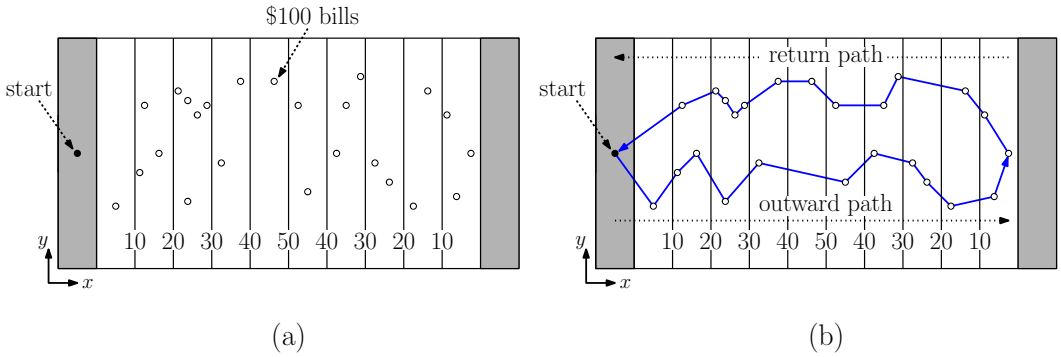

Figure 4: Problem 2.

Problem 5. A shipping company wants to ship n objects of weights $\{w_1, \ldots, w_n\}$. Each weight is a positive integer. The company wants to partition these objects between two different cargo ships, so that the total weight of the two ships is as similar as possible. In particular, if W_1 is the total weight of objects on Ship 1, and W_2 is the total weight on Ship 2, then the objective is to minimize the *weight ratio*,

$$\frac{\max(W_1, W_2)}{\min(W_1, W_2)}.$$

Observe that this ratio is never smaller than 1, and it equals 1 if and only if the two ships are carrying identical total weights.

For example, suppose the inputs are $w_1 = 40$, $w_2 = 70$, $w_3 = 20$, $w_4 = 30$, $w_5 = 60$, and $w_6 = 50$. If we partition the elements as Ship-1 $= \{2, 5\}$ and Ship-2 $= \{1, 3, 4, 6\}$, then the total weights are $70 + 60 = 130$ and $40 + 20 + 30 + 50 = 140$. The final weight ratio is $140/130 \approx 1.077$.

This is called the *Partition Problem*. Present an efficient algorithm, which given the weights $\{w_1, \ldots, w_n\}$, computes the optimum weight ratio. Justify your algorithm's correctness and derive its running time. Express the running time as a function of both n and the total weight $W = \sum_{i=1}^{n} w_i$.

(**Hint:** Use Dynamic Programming. It suffices just to give the DP formulation. You need only compute the optimum weight ratio, not the actual partition. Justify your algorithm's correctness and derive its running time. Note that $O(n \cdot W)$ time is possible. Additional hint: It suffices to focus on computing the total weight carried by just one of the ships, since the other must carry all the remaining weight.)