

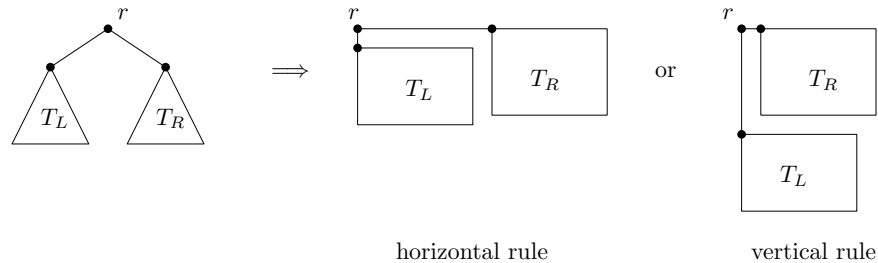
# CS 473 (Fall 2026)

## Homework 3 (due Sep 24 Thu 10am)

**Instructions:** Carefully read <http://courses.grainger.illinois.edu/cs473/fa2026/policies.html#hw> and <http://courses.grainger.illinois.edu/cs473/fa2026/integrity.html> (in particular, remember to include the 3 disclosure statements, on collaborators, outside sources, and AI use, for each problem).

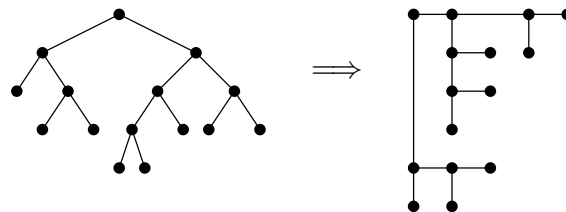
**Problem 3.1:** This problem is about how to draw a binary tree, where vertices are to be placed at integer points in the plane and edges are to be drawn horizontally or vertically without crossings. We define a class of *neat drawings* (and their *widths* and *heights*) recursively as follows. Let  $T$  be a binary tree with root  $r$  and left and right subtrees  $T_L$  and  $T_R$ . We can combine a neat drawing of  $T_L$  with width  $W_L$  and height  $H_L$  and a neat drawing of  $T_R$  with width  $W_R$  and height  $H_R$ , to obtain a neat drawing of  $T$ , in 2 possible ways:

- *horizontal rule:* place the root of  $T_R$  at  $W_L + 1$  units to the right of  $r$ , and place the root of  $T_L$  at 1 unit below  $r$ ; the resulting drawing of  $T$  has width  $W_L + 1 + W_R$  and height  $\max\{H_L + 1, H_R\}$ ; or
- *vertical rule:* place the root of  $T_L$  at  $H_R + 1$  units below  $r$ , and place the root of  $T_R$  at 1 unit to the right of  $r$ ; the resulting drawing of  $T$  has width  $\max\{W_L, W_R + 1\}$  and height  $H_L + 1 + H_R$ .



For a tree with just one node, there is a neat drawing of width 0 and height 0. The *area* of a drawing is defined as its width times height.

The example below shows one neat drawing of a binary tree, with width 4 and height 5 (and thus area 20):



Given a binary tree with  $n$  nodes, we want to find a neat drawing that minimizes the area. Present an efficient (polynomial time) algorithm for this problem using dynamic programming.

Include the following steps: (a) first define your subproblems precisely, (b) derive the recursive formula (including base cases) with brief justifications, (c) specify a valid evaluation order, and (d) analyze the running time and space (as a function of  $n$ ). For this problem, you do not need to write pseudocode if your recursive formula and evaluation order are described clearly. And your algorithm only needs to output the minimum area, and not the drawing itself.

[Hint: solve a more general problem, of finding the minimum-height drawing for any given width.]

**Problem 3.2:** We are given a set  $S$  of  $n$  real numbers between 0 and 1. We are also given an integer  $L \leq n$ . We want to select a subset  $Q \subset S$  of  $L$  numbers to minimize the cost function

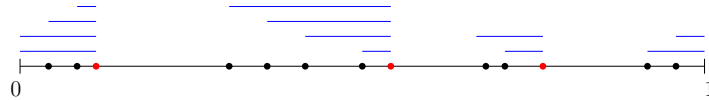
$$c(S, Q) = \sum_{x \in S} \min_{q \in Q} \delta(x, q),$$

where

$$\delta(x, q) = \begin{cases} q - x & \text{if } q \geq x \\ q - x + 1 & \text{if } q < x \end{cases}$$

(We can think of  $\delta(x, q)$  as the distance from  $x$  to  $q$  but with “wrap-around”. As an application, imagine we want to build  $L$  hospitals on a one-way loop, minimizing average response time...)

In the following example,  $L = 3$ , and a feasible solution  $Q$  is drawn in red, where the cost is the sum of the lengths of the blue line segments.



- (a) (70 pts) First describe an  $O(n^3 \log L)$ -time dynamic programming algorithm for this problem. Include the following steps: (a) first define your subproblems precisely, (b) then derive the recursive formula (including base cases) with brief justifications, (c) specify a valid evaluation order, and (d) analyze the running time. For this problem, you do not need to write pseudocode if your recursive formula and evaluation order are described clearly. And you do not need to write pseudocode to output the optimal subset itself.
- (b) (20 pts) Let  $x_1, \dots, x_n$  be the numbers of  $S$  in increasing order. For  $i \leq j$ , define

$$d(i, j) = \sum_{k=i}^{j-1} (x_j - x_k).$$

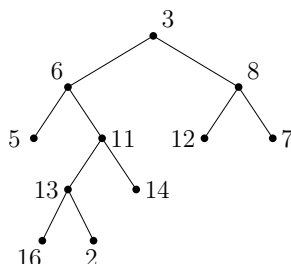
Prove that  $d$  satisfies the concave Monge property, i.e., for any  $i \leq i' \leq j \leq j'$ , we have  $d(i, j) + d(i', j') \leq d(i, j') + d(i', j)$ .

- (c) (10 pts) Using results stated in class, argue that the running time of your algorithm from (a) can be improved to  $O(n^2 \log L)$ .

---

**Optional Problem 3.3 (not graded):** (*Turning a sequence into a tree*) We are given a sequence of  $n$  numbers  $\langle a_1, \dots, a_n \rangle$ . We want to compute a proper binary tree  $T$  with nodes  $a_1, \dots, a_n$  such that the *inorder traversal* is precisely  $\langle a_1, \dots, a_n \rangle$ , while minimizing the cost function  $c(T) = \sum_{(a_i, a_j) \text{ is an edge of } T} |a_i - a_j|$ . (A proper binary tree refers to a rooted tree where each internal node of  $T$  has exactly 2 children.)

For example, for the input sequence  $\langle 5, 6, 16, 13, 2, 11, 14, 3, 12, 8, 7 \rangle$ , one feasible solution is the following tree, with cost  $|3 - 6| + |3 - 8| + |6 - 5| + |6 - 11| + |8 - 12| + |8 - 7| + |11 - 13| + |11 - 14| + |13 - 16| + |13 - 2| = 38$  (we do not claim that this is optimal).



Describe a dynamic programming algorithm to solve this problem. For full credit, the running time should be at most  $O(n^4)$ . Include the following steps: (i) first define your subproblems precisely, (ii) then derive the recursive formula (including base cases) with brief justifications, (iii) specify the evaluation order, and (vi) analyze the running time and space. For this problem, you do not need to write pseudocode if your recursive formula and evaluation order are described clearly. And you do not need to write pseudocode to output an optimal solution; just the optimal cost suffices.

[Hint: define  $C(i, j, m)$  to be the minimum cost over all binary trees  $T$  for the subsequence  $\langle a_i, a_{i+1}, \dots, a_j \rangle$  with the extra constraint that the root is  $a_m$ .]

**Optional Problem 3.4 (not graded):** We are given an edge-weighted directed graph  $G = (V, E)$  with  $n$  vertices and  $m$  edges (with  $m \geq n$ ). In addition, we are given a *source* vertex  $s \in V$ , and a number  $K \leq n$ .

We want to find a path  $\pi$  from  $s$  to some vertex  $v$  using at most  $K$  edges, minimizing the total edge weight. Note that in this problem, we want to output not just the optimal cost but also an optimal path.

- (a) (50 pts) Describe a dynamic programming algorithm to solve this problem in  $O(mK)$  time and  $O(nK)$  space.

Include *all* of the following steps: (i) first define your subproblems precisely, (ii) then derive the recursive formula (including base cases) with brief justifications, (iii) specify the evaluation order, (iv) write pseudocode to output the optimal cost, (v) write pseudocode to output an optimal path, and (vi) analyze the running time and space.

[Hint: define  $C(u, k)$  to be the minimum cost of a path from  $u$  using at most  $k$  edges. . .]

- (b) (50 pts) Design and analyze a slightly slower but more space-efficient algorithm to solve this problem in  $O(mK \log K)$  time and  $O(n \log K)$  space.

[Hint: use divide-and-conquer in combination with your algorithm from (a) (somewhat similar to the space-saving trick from class but with a different recurrence).]