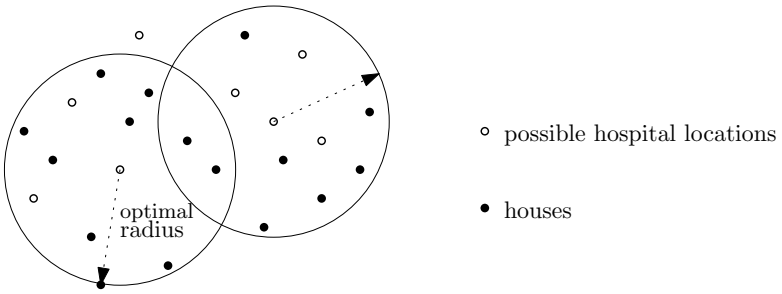
CS 473, Spring 2023
Homework 2 (due Feb 8 Wed 9pm)

Instructions: As in Homework 1.

Problem 2.1: We are given a list L of n locations, and a list H of n houses. For each location $p \in L$ and each house $h \in H$, we are given a distance value $d(p, h)$. We want to find two locations $p, q \in L$ to build two hospitals so that $\max_{h \in H} \min\{d(p, h), d(q, h)\}$ is as small as possible. (The quantity $\min\{d(p, h), d(q, h)\}$ represents the distance of the house h to the closer of the two hospital locations p, q. The motivation is to make the worst such distance as small as possible.)



(a) (70 pts) First give a subcubic algorithm for the following decision problem: given a value r, do there exist $p, q \in L$ such that $\max_{h \in H} \min\{d(p, h), d(q, h)\} \leq r$?
[Hint: use an algorithm from class.]

(b) (30 pts) Now give a subcubic algorithm for the original problem, using (a).

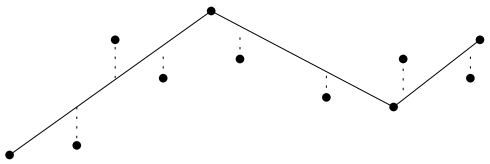
Problem 2.2: We are given a sequence of points $p_1 = (x_1, y_1), \ldots, p_n = (x_n, y_n)$ sorted from left to right (i.e., $x_1 < x_2 < \cdots < x_n$) and a number k between 1 and n. We want to find a minimum-error polygonal path from p_1 to p_n with k edges that goes from left to right, where the *error* of a path is the sum of the vertical distances of the points $p_1, \ldots, p_n$ to the polygonal path. (The motivation is in finding a piecewise linear function that best "fits" the data points.)

More precisely, we want a subsequence $p_{i_0}, p_{i_1}, p_{i_2}, \ldots, p_{i_k}$ where $1 = i_0 < i_1 < i_2 < \cdots < i_{k-1} < i_k = n$, minimizing the error function $f(i_0, i_1) + f(i_1, i_2) + \cdots + f(i_{k-1}, i_k)$, where

$$f(a, b) = \sum_{i=a+1}^{b-1} \left| (y_i - y_a) - \left(\frac{y_b - y_a}{x_b - x_a} \right) (x_i - x_a) \right|$$

represents the sum of the vertical distances of the points $p_{a+1}, \ldots, p_{b-1}$ to the line through $p_a p_b$.

Describe an efficient (polynomial-time) algorithm to solve this problem using dynamic programming. Include the following steps: (a) first define your subproblems precisely, (b) then



derive the recursive formula (including base cases) with brief justifications (no need for long correctness proofs), (c) write pseudocode to output the optimal value (i.e., the minimum error), (d) write pseudocode to output the optimal subsequence, and (e) analyze the running time and space (as a function of n and k).

Problem 2.3: We have n jobs, where job i must start at time t_i (these start times are all given in advance; you may assume they are sorted $t_1 < t_2 < \cdots < t_n$). We have 3 servers, each working at different speeds: for each $j \in \{1, 2, 3\}$, server j requires L_j units of time to handle each job. The goal is to choose a largest subset of jobs that can be handled by the 3 servers. Each job can only be assigned to one server. Thus, the main constraint is that whenever two different jobs i and i' are assigned to the same server j, we must have $|t_{i'} - t_i| > L_j$.

Describe an efficient (polynomial-time) algorithm to compute the optimal value (i.e., size of the largest feasible subset) using dynamic programming. Include the following steps: (a) first define your subproblems precisely, (b) then derive the recursive formula (including base cases) with brief justifications (no need for long correctness proofs), (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 you do not need to write pseudocode to output the optimal subset or assignment of jobs to servers.

Example: Suppose we have 6 jobs with start times $t_1 = 3$, $t_2 = 8$, $t_3 = 11$, $t_4 = 12$, $t_5 = 16$, $t_6 = 18$, and suppose we have 3 servers with $L_1 = 6$, $L_2 = 7$, $L_3 = 9$. A feasible solution is to let server 1 handle jobs 1, 3, and 6; and server 2 handle jobs 2 and 5; and server 3 handle job 4. The optimal value is 6 (all 6 jobs can be handled).