CS 170 Homework 5 (Optional)

No due date. This homework is for your practice and will not be graded. The solutions are released along with the homework.

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write "none".

As this homework is optional, it will be worth no points.

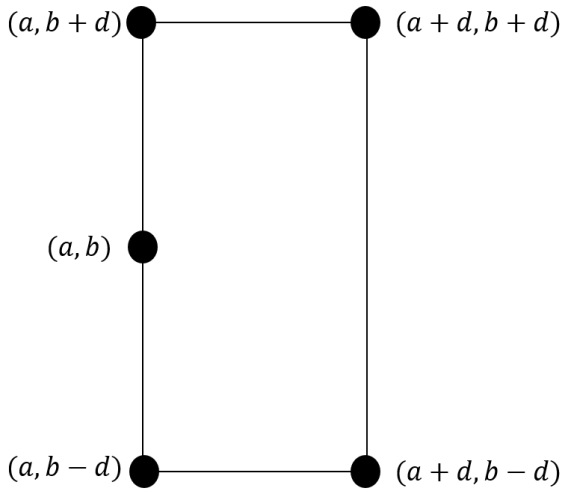
2 Agent Meetup

Manhattan has a road system where streets form a checkerboard pattern, and all roads are either straight North-South or East-West. We simplify Manhattan's roadmap by assuming that each pair x, y, where x and y are integers, corresponds to an intersection. As a result, the distance between any two intersections can be measured as the *Manhattan distance* between them, i.e. $|x_i - x_j| + |y_i - y_j|$. You, working as a mission coordinator at the CS 170 Secret Service Agency, have to arrange a meeting between two of n secret agents located at intersections across Manhattan. Hence, your goal is to find the two agents that are the closest to each other, as measured by their Manhattan distance.

In this problem, you will devise an efficient algorithm for this special purpose. As mentioned before, you can assume that the coordinates of the agents are integer values, i.e. the ith agent is at location (x_i, y_i) where x_i, y_i are integers.

Note: This problem is very geometric, we suggest you draw examples when working on it!

(a) Let (a, b) be an arbitary intersection. Suppose all agents i for which $x_i > a$ are Manhattan distance strictly greater than d apart from each other, where $d > 0$. Prove that we can have up to 8 agents that satisfy $a \leq x_i \leq a + d$ and $b - d \leq y_i \leq b + d$ for each agent i. In other words, show that we can fit up to 8 agents within this rectangle (visualized below) without two of these agents being Manhattan distance d or less apart?



(b) Design a divide and conquer algorithm to find the minimum Manhattan distance between any two agents in $O(n \log^2 n)$ time. **Give a three-part solution.** Note that the optimal runtime is $\Theta(n \log n)$, but it is not required for full credit.

Hint: Try sorting the list of agents by x-coordinate, splitting the agents, and then sorting some agents by y-coordinate. Note that since all distances are integer values, we only need to consider pairs that are $d-1$ values away from each other, where d is the smallest distance that we have computed so far

3 2-SAT

Please provide solutions to parts (d), (e) and (f) of Question 3.28 from the textbook.

4 Counting Shortest Paths

Given an undirected unweighted graph G and a vertex s, let $p(v)$ be the number of distinct shortest paths from s to v. We will use the convention that $p(s) = 1$ in this problem. Give an $O(|V| + |E|)$-time algorithm to compute $p(v) \bmod 1337$ for all vertices. Only the main idea and runtime analysis are needed.

(*Hint: For any v, how can we express $p(v)$ as a function of other $p(u)$?*)

Note: As a secondary question, you should ask yourself whether the runtime would remain the same if we were computing $p(v)$ rather than $p(v) \bmod 1337$.

5 Sum of Products

This question guides you through writing a proof of correctness for a greedy algorithm. You have n computing jobs to perform, with job i requiring t_i units of CPU time to complete. You also have access to n machines that you can assign these jobs to. Since the machines are in high demand, you can only assign one job to any machine. The jth machine costs c_j dollars for each unit of CPU time it spends running a job, so assigning job i to machine j will cost you $t_i \cdot c_j$ dollars (each job takes the same amount of CPU time to complete, regardless of which machine is used). Your goal is to find an assignment of jobs to machines that minimizes the total cost.

Assume the jobs and machines are sorted and have distinct runtimes/costs, i.e. $t_1 > t_2 > \cdots > t_n$, and $c_1 > c_2 > \cdots > c_n$

(a) What assignment of jobs to machines minimizes the total cost? (*Hint: What machine should we assign the longest job to? What machine should we assign the second longest job to? It might help to solve a small example by hand first.*)

(b) Given an assignment of jobs to machines, consider the following modification: If there is a pair of jobs i, j such that job i is assigned to machine i', job j is assigned to machine j', and $t_i > t_j$ and $c_{i'} > c_{j'}$, instead assign job i to machine j' and job j to machine i'. Show that this modification decreases the total cost of an assignment.

(c) Use part b to show that the assignment you chose in part a has the minimum total cost (*Hint: Show that for any assignment other than the one you chose in part a, you can apply the modification in part b. Conclude that the assignment you chose in part a is the optimal assignment.*)