CS 170 HW 5 (Optional)

Due **2021-10-04, at 10:00 pm**

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.

In addition, we would like to share correct student solutions that are well-written with the class after each homework. Are you okay with your correct solutions being used for this purpose? Answer "Yes", "Yes but anonymously", or "No"

You may submit your solutions if you wish them to be graded, but they will be worth no points

2 True and False Practice

For the following problems, justify your answer or provide a counterexample.

(a) True or False: Kruskal's works with negative edges.

(b) We modify a graph G with negative edges by adding a large positive constant to each edge, making all edges positive. Let's call this modified graph G'.

True or False: If we run Dijsktra's on G', the resulting shortest paths on G' are also the shortest paths on G.

(c) Let $G = (V, E)$ be a DAG with positive edge weights. We first run Dijkstras algorithm to compute the distance from the source s to every other vertex v. Afterwards, we store the vertices in increasing order of their distance from s.
True or False: this sequence of vertices be a valid topological sort of G.

(d) Let $G = (V, E)$ be an undirected graph. Let $G' = (V', E')$ where $V' = V \bigcup \{u\}$ and $E' = E \bigcup E_u$, where E_u is some set of edges that include u.
True or False: any MST of G is the subset of some MST of G'.

3 Bounding Sums

Let $f(\cdot)$ be a function. Consider the equality

$$\sum_{i=1}^{n} f(i) \in \Theta(f(n)),$$

Give a function f_1 such that the equality holds, and a function f_2 such that the equality does not hold.

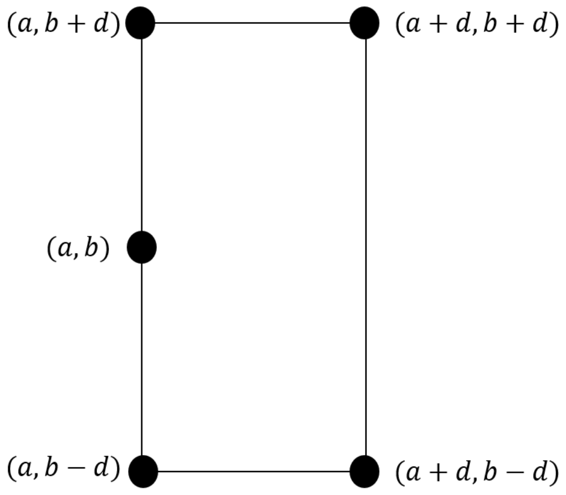
4 Agent Meetup

Manhattan has an “amazing” 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 $d > 0$. Give an upper bound on the number of agents i that satisfy $a \le x_i \le a + d$ and $b - d \le y_i \le b + d$. In other words, how many agents can fit in this rectangle (visualized below) without two of these agents being Manhattan distance d or less apart?



Briefly justify your answer. A reasonable bound suffices, you are not expected to provide the tightest possible bound.

(b) Design a divide and conquer algorithm to find the minimum Manhattan distance between any two agents. **Give a three-part solution.** A solution that has runtime within logarithmic factors of the optimal runtime will still get full credit.

Hint: Try sorting the list of agents by y-coordinate, and then sorting the list of agents by x 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

5 Box Union

There are n boxes labeled $1, \dots, n$, and initially they are each in their own stack. You want to support two operations:

- put(a, b): this puts the stack that a is in on top of the stack that b is in.
- under(a): this returns the number of boxes under a in its stack.

The amortized time per operation should be the same as the amortized time for find$(\cdot)$ and union$(\cdot, \cdot)$ operations in the union find data structure.
Hint: use "disjoint forest" and augment nodes to have an extra field z stored. Make sure this field is something easily updateable during "union by rank" and "path compression", yet useful enough to help you answer under$(\cdot)$ *queries quickly. It may be useful to note that your algorithm for answering under queries gets to see the z values of all nodes from the query node to its trees root if you do a find.*