Instructions.

1. You could submit a pdf of the homework, either printed or hand-written and scanned, as long as it is easily readable.

2. If your solution is not clearly written, it might not be graded.

3. Prove the correctness of your answer. A correct answer without a proof might not be accepted.

4. If you have discussed the solution with other students, mention their names clearly on the homework. These discussions are not forbidden and are actually encouraged. However, you must write your whole solution yourself.

5. All questions have same weight.

6. You could refer to a data structure studied in class, and just mention briefly their guaranties. For example “It is known that a Red-Black tree could support the insert/delete/find operations on a set of n elements in time $O(\log n)$.”

7. If your answer uses one of the data structures or algorithms that were studied in class, you could refer to it without having to repeat details studied in class. If your answer requires only modifications of one of the algorithm, it is enough to mention the required modifications, and what’s the effect (if any) on the running time and on other operations that the algorithm performs.

8. In general, a complete answer should contain the following parts:

   (a) High level description of the data structures (if needed). E.g. We use a binary balanced search tree. Each node contains, a key and pointers to its children. We augment the tree so each node also contains a field...

   (b) High level description of the algorithms

   (c) Proof of correctness (why your algorithm provides what is required).

   (d) A claim about the running time, and a proof showing this claim.
1. Let $M$ be a given arbitrary large integer. Let $\{h_1 \ldots h_n\}$ be a set of halfplanes in 2D, and let $c(x, y) = x$ be a cost function we wish to maximize. Let $v$ be the optimal solution to the LP problem, and let $u$ be the solution to the ILP problem.

Give an example of 3 halfplanes $\{h_1, h_2, h_3\}$ such that $u = (0, 0)$ while the $x$-coordinate of $v$ is $\geq M$. That is, the solutions to the two problems are at distance $\geq M$ from each other.

The purpose of this question is to demonstrate the LP and ILP, both with the same set of constraints, might have optimal solutions that are arbitrary far from each other.

2. Let $G(V, E)$ be a graph. Each edge $e_i \in E$ is given with a positive weight $w_i$. For any subset $E' \subseteq E$, we define the \textit{weight of $E'$} (denoted by $w(E')$) to be the sum of weights of all edges of $E'$. That is, $w(E') = \sum_{e \in E'} w(e)$

   (a) A set $F \subseteq E$ of edges is an \textit{vertex cover} of $V$ if every vertex of $V$ is incident to at least one edge of $F$.

   Given $G(V, E)$, you would like to find a min-weight vertex cover for $G(V, E)$. Show that this problem could be expressed as an ILP problem.

   \textbf{Hint:} start by assigning to each edge $e_j \in E$ a boolean variable $x_j$, when $x_j = 1$ if $e_j \in F$, and $x_j = 0$ if $e_j \notin F$.

   (b) A set $M \subseteq E$ of edges is a \textit{matching} if every vertex $v \in V$ is incident to at most one edge of $M$. Show that this problem could be expressed as an ILP problem.

   \textbf{Answer:} As usual, denote $n = |V|$ and $m = |E|$. We are using ILP. For each edge $e_j \in E$ we assign a \textit{boolean} variable $x_j$. The edge $e_j$ is in the matching $M$ iff $x_j = 1$. The summation line (1) indicates that the ILP will aim at maximizing the sum of weights of edges in $M$. (that is, the ones for which $x_j = 1$). Note that if $e_j \notin M$ then $x_j = 0$, and of course $x(e_j)w(e_j) = 0$, implying that this edge has no contribution in the summation (1).

\[
\text{Maximize } \sum_{e_j \in E} x_j w(e_j)
\]

\textbf{Subject to}

\[
\begin{align*}
0 \leq x_j & \leq 1 & \forall e_j \in E \\
x_j \text{ is an integer} & \forall e_j \in E \\
\sum_{e \in E} x_e & \leq 1 & \forall v \in V \\
\{\forall e \in E \text{ s.t. } e \text{ is incident to } v\}
\end{align*}
\]

Line (2) and (3) together ensure that each $x_j$ is a boolean variable, and the only values it could accept are 0, 1. Line (4) ensures that for any vertex $v \in V$, only one of all the edges that are incident to $v$ could be in $M$.

3. You are given a set $S = \{p_1 \ldots p_n\}$ of $n$ points in the plane. Each point $p_i$ is given by its coordinates $p_i = (x_i, y_i)$. Let $\ell$ be a line $\{y = ax + b\}$. We define the vertical distance $d(p_i, \ell) = |y_i - (ax_i + b)|$. We define the distance

\[
D(S, \ell) = \max_{i=1}^{n} d(p_i, \ell)
\]

Suggest an expected $O(n)$-time algorithm that finds the line $\ell$ that minimizes $D(S, \ell)$. That is, this is the line that brings to minimum the distance from $\ell$ to any point in $S$. See Figure ??
Hint: Assume first that we just want to decide if there is a \( \ell \) for which \( D(S, \ell) \leq 1 \). Use HW5 for this problem. Next, suggest a linear programming problem with 3 variables \( a, b, \rho \), when the goal is to minimize \( \rho \), then then constraints impose that
\[
\forall p_i \in S, \ d(p_i, \ell) \leq \rho
\]

Note also the the randomized incremental algorithm could be generalized to solve LP problems in 3 dimensions, (as discussed in class) and its expected running time is still \( O(n) \). You could refer to this fact even if you don’t know how to prove it.

**Answer:** The LP has 3 variables: \( a, b \) and \( \rho \). We need to write something like this:

(5) Minimize \( \rho \)

subject to

(6) \( |ax_i + b - y_i| \leq \rho \) \( \forall p_i \in S \)

But since we could not express absolute value as a linear function, we need to express each constraint by a pair of linear constraints:

(1) Minimize \( \rho \)

subject to

(2) \( (ax_i + b) - y_i \leq \rho \) \( \forall p_i \in S \) // The case that \( \ell \) above \( p_i \)

(3) \( y_i - (ax_i + b) \leq \rho \) \( \forall p_i \in S \) // The case that \( \ell \) is below \( p_i \)

Since it is an LP in 3 dimensions, (3 variables) we could use the randomize incremental algorithm, which solves an LP in expected time \( O(n) \), when \( n \) is the number of constraints.

Note: Another way to understand the solution is by noting that in general, instead of writing \( |x| \leq 1 \), (which is not a linear constraint) we could write a couple of linear constrains

(1) \( x \leq 1 \)

(2) \( x \geq -1 \)

**BTW:** A good motivation to this problem is the following: Assume \( S \) are the results of a set of experiments on a phenomena that depends on two variables, and suspect that we could find

![Figure 1](attachment:image_url)
a linear correlation between them, but taking into account measurements errors. Then \( \ell \) is picked such that all errors measures are below a threshold. The smaller this threshold is, the better the fit.

4. The question refers to a quad tree \( T \) built on a green-red image, as shown in the slide 2 and 3. All pixels of the image are taken from a \( 2^h \times 2^h \) integers grid, where \( h \) is an arbitrary integer. Give an example of image, for which \( T \) has \( 4^h \) leaves.

5. Consider a quadtree \( T \), built for storing a set \( S \) of \( n \) points in the plane, all taken from the \( 2^h \times 2^h \) integers grid, where \( h \) is an arbitrary integer. Assume that each leaf of \( T \) contains \( \leq 1 \) points.

Describe the pseudo-code of an algorithm that receives a node \( v \) of \( T \) (initially root(\( T \))) and a new point \( p \), and insert \( p \) into \( T \). The algorithm should be as efficient as possible. What is its worst running time (as a function of \( h \))? What is its best running time?

The algorithm could be recursive, but this is not a requirement.

6. The question refers to a Quadtree constructed for for storing a set of points (as discussed in the slides). Let \( S \) be a set of points, all from the \( 2^h \times 2^h \) grid. Let \( T \) be a quadtree built on \( S \), as shown in the slides.

Let \( Q \) be a query disk (given to you by its center and radius). The operation \( \text{count}(T, Q) \) reports \( |Q \cap S| \) (the number of points of \( S \) which are also inside \( Q \)).

Suggest a slight modification of \( T \), such that when a new query disk \( Q \) is given, you could compute \( \text{count}(T, Q) \) efficiently.

Write a recursive pseudocode for the operation \( \text{count}(v, Q) \), where \( v \) is a node of \( T \). The first call to this function is with \( \text{count}(\text{root}(v), Q) \)

Note that we are interested in which points of \( S \) lie inside \( Q \). Only in their number. Your code should be as efficient as possible (and would not be accepted otherwise).

**Answer:** Essentially, each node \( v \) of the \( T \) is augmented with a field \( \text{size}[v] \) containing the number of points of stored at \( R(v) \) (the region the \( v \) is associated with). In other words, these are the points of \( S \) stored at the subtree rooted at \( v \).

The pseudo code is similar to the \( \text{report} \) function mentioned in the slides, where the main difference is that when \( R(v) \subseteq Q \), we just return \( \text{size}(v) \), and return, rather than visit each node in \( v \)'s subtree.

**Algorithm** \( \text{Count}(Q, v) \)
*recursive function*

**Input:** \( v \) is a node of \( T \), and \( Q \) is the query disk

**Output:** The number of points of \( S \) in the subtree rooted at \( v \)

1. if \( v = \text{NULL} \) return 0
2. if \( R(v) \cap Q = \emptyset \) return 0
3. if \( R(v) \subseteq Q \) return \( \text{size}(v) \)
4. if \( v \) is a leaf
5. then for each point \( p \) stored at \( v \) check if \( p \) is inside \( Q \)
6. return the number of points stored at \( v \) which are inside \( Q \)
7. return \( \text{Count}(Q, NW(v)) + \text{Count}(Q, NE(v)) + \text{Count}(Q, SW(v)) + \text{Count}(Q, SE(v)) \)