Instructions.

1. Solution may \textbf{not} be submitted by students in pairs.

2. You may submit a pdf of the homework, either printed or hand-written and scanned, as long as it is \textit{easily} readable.

3. If your solution is illegible and not clearly written, it might not be graded.

4. Unless otherwise stated, you should prove the correctness of your answer. A correct answer without justification may be worth less.

5. If you have discussed any problems with other students, mention their names clearly on the homework. These discussions are not forbidden and are actually \textbf{encouraged}. However, you must write your whole solution yourself.

6. Unless otherwise specified, all questions have the same weight.

7. You may refer to data structures or their properties studied in class without having to repeat details, and may reference theorems we have studied without proof. If your answer requires only modifications to one of the algorithms, it is enough to mention the required modifications, and the effect (if any) on the running time and on other operations that the algorithm performs.

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

   (a) A high level description of the data structures (if needed). \textit{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) A clear description of the main ideas of the algorithm. You may include pseudocode in your solution, but this may not be necessary if your description is clear.

   (c) Proof of correctness (e.g. show that your algorithm always terminates with the desired output).

   (d) A claim about the running time, and a proof showing this claim.
Questions.

1. You are given a list \( V \) of tasks that your CPU needs to execute. Moreover, you are given a list \( L \) of directed pairs \((i,j)\) such that if a pair \((i,j)\) is in \( L \), then task \( j \) needs an input from task \( i \). How would you find if there is any order of executing these tasks, so no task need to wait for another task.

2. In Kuhn algorithm we maintain for each node \( v \) the number \( \text{InDeg}(v) \) of edges entering \( v \). That is

\[
|\{u \in V \mid (u,v) \in E\}|
\]

What is the purpose of this counter?

3. Let \( P = \{p_1 \ldots p_n\} \) be a set of points in the plane. Show that you could find in \( O(n) \) time the axis-parallel square with the smallest perimeter that encloses \( P \).

4. Let \( G(V,E) \) be a graph with positive weights assigned to its edges, and let \( v_1, v_n \in V \) be a source and destination. Show a linear programming problem whose solution is the distance \( \delta(v_1, v_n) \). Hint: There an \( n \) variables \( \{d_1 \ldots d_n\} \). The variable \( d_i \) will contain, when the algorithm terminates, \( \delta(v_1, v_n) \) (for every \( i \)). The cost function is (somehow counterintuitively) is

\[
\max d_n
\]

and \( d_1 = 0 \). Now what are the constrains of the other variables? Note - you only need to write the linear programming. You do not need to run Dijkstra.

5. Give an example of an input for the Incremental Linear Programming Algorithm in two dimension requires \( \Omega(n^2) \) time. Specify the including an order of insertions of the halfplane, and prove your answer.

6. Repeat HW3 question 10 but this time aim for \( O(n^2) \) running time (hint - use the fact that all segments are vertical). Note that the running time specified in HW3 Question 10 was \( O(n^2 \log n) \).

7. Show an example to the diet problem in two dimensions, where the solution is not unique.

8. Let \( R \) be a set on \( n \) points and let \( B \) be another set of \( n \) points. It will be convenient to refer to them as red points and blue points.

   (a) Consider the line \( \ell \) defined by the equation \( y = \alpha x + \beta \) where \( \alpha, \beta \) are given constants. Assume that all point of \( B \) are below \( \ell \), and all the points of \( R \) are above \( \ell \). Assume \((x_0, y_0)\) is one of these points. What could you say about the number \( y_i - (\alpha x_i + \beta) \) ?

   (b) Next assume that you are only given the coordinates of the points in \( R \) and \( B \). Suggest an algorithm with expected running time \( O(n) \) that finds a line \( \ell \) that separates \( R \) from \( B \).

   (c) Next assume that \( R \) and \( B \) are points in the 3-dimensional space. Suggest a linear programming problem that finds if there is a plane separating \( R \) from \( B \). Can you generalize this algorithm to the \( d \)-dimensional space?

9. Let \( \{\alpha_1 \ldots \alpha_n\} \) be given values.

   (a) Show a linear programming problem in \( n + 1 \) dimensions that finds a value \( \beta \) that minimizes

\[
\sum_{i=1}^{n} |\beta - \alpha_i|.
\]

Hint: start by finding a way to express the problem \(|x| \leq 1\) as a linear programming with 2 constrains.
(b) With the same input, suggest an algorithm that in expected time $O(n)$ finds a value $\delta$ that minimizes
\[
\max |\delta - \alpha_i|
\]

10. Let $G(V, E)$ be a graph. Show an algorithm that assign to every vertex $v_i \in V$ a weight (a positive value) $w_i$ such that the sum of weights is as small as possible, for every edge $(v_i, v_j) \in E$ the sum of weights $w_i + w_j \geq 1$.

11. Let $A = \{a_1 \ldots a_n\}$ be a list of courses, and $B = \{b_1 \ldots b_n\}$ be a list of instructors. Also given a list $E$ of pairs $(a_i, a_j)$, such that an edge $(a_i, b_j)$ indicates that instructor $a_i$ is qualified to teach the course $b_j$. Not every instructor could teach every course. A course could be taught by more than one instructor, if they split the work between them (not necessarily equally). Your goal is to assign positive values $0 \leq w(a_i, b_j) \leq 1$ for every $(a_i, b_j) \in E$. We under this value by the portion of time that $a_i$ dedicates to course $b_j$. For example if $w(a_1, b_1) = 1$ we understand it that $a_1$ spends 100% of his or her time teaching course $b_1$.

(a) An instructor could taught multiple courses, but the total effort in all these course could not exceed 1 (that is, 100% of the instructor time). And in addition

(b) Each course is fully taught. That is, the total sum of weights of all instructors for this course is 100%.

![Diagram](image.png)

Figure 1: Instructor $a_1$ is capable to teach only the courses $b_1$ and $b_2$ but not $b_3$. The numbers near the edges indicate the effort that the instructor puts into the course. For example, $a_1$ spends 100% of his time teaching $b_1$ and 0% of his time teaching $b_2$. The course $b_3$ is taught by most $a_3$ but 10% by $a_2$.

This problem is known as the assignment problem.