10 (100 pts.) There is a war between the underworld and the heavens.

[The following question is long, but should be relatively easy (and quick) to do, as the subparts are pretty short/straightforward.]

Let $A$ and $B$ be two sets of $n$ lines in the plane in general position (i.e., no line in $A \cup B$ is vertical, no pair of lines are parallel, and no three lines meet in a common point). Our purpose here is to compute efficiently a point $p$ that lies above all the lines of $A$ and below all the lines of $B$ — such a point is feasible. Formally, a point $p$ lies above a line $\ell$, of $p$ is on $\ell$, or lies vertically above it. Similarly, $p$ is below $\ell$, if $p$ is on $\ell$, or lies vertically below it.

A vertex is the intersection point of two lines.

10.A. (10 pts.) Consider the following functions

\[ L(t) = \min_{\ell \in B} \ell(t) \quad \text{and} \quad U(t) = \max_{\ell \in A} \ell(t), \]

where $\ell(x)$ denote the $y$ coordinate of the point on $\ell$ with $x = t$. The function $L(t)$ is the lower envelope of $B$, and it is concave. The function $U(t)$ is the upper envelope of $A$, and it is a convex function. See Figure ??.

\[ L \]
\[ U \]
\[ L \]
\[ (A) \]
\[ (B) \]
\[ (C) \]

(a)
(b)
(c)

Figure 1: (A) The set $B$ of lines, and its lower envelope $L$. (B) The set $A$ of lines, and its upper envelope $U$. (C) The feasible region is sandwiched between $L$ and $U$.

Consider the function $f(t) = L(t) - U(t)$. Argue that the function $f$ is concave. Show how to compute the value of $f(t)$, for a specified value of $t$, in $O(n)$ time. Show how to compute the left and right derivatives of $f$ at $t$ (denoted by $f'_-(t)$ and $f'_+(t)$) in $O(n)$ time.

10.B. (10 pts.) Using the above, describe an algorithm that for a given $t$, outputs a feasible point in $O(n)$ time, if there exits such a point with $x$ coordinate equal to $t$ (hint: compute the value $f(t)$).

10.C. (10 pts.) You are given value of $t$, such that $f(t) < 0$. Argue that if one of the following holds, then there is no feasible point:
(i) \( f'_-(t) = 0 \) or \( f'_+(t) = 0 \).
(ii) \( f'_-(t) > 0 \) and \( f'_+(t) < 0 \).

10.D. (10 Pts.) You are given a value of \( t \), such that \( f(t) < 0 \). Argue that the following two things hold:
(i) If \( f'_-(t) < 0 \), and \( f'_+(t) < 0 \), then if there is a feasible point \((x, y)\), it must be that \( x < t \).
(ii) If \( f'_-(t) > 0 \), and \( f'_+(t) > 0 \), then if there is a feasible point \((x, y)\), it must be that \( x > t \).

10.E. (10 Pts.) You are given a value \( t \), and two lines \( \ell_1, \ell_2 \in A \) that intersects at a point \((x', y')\), such that \( x' < t \), and all the feasible points of \( A \cup B \) have \( x \) coordinate strictly larger than \( t \). Show how to compute in \( O(1) \) time a line \( \ell_i \) (\( i = 1 \) or \( i = 2 \)), such that a point \( p \) is feasible for \( A \cup B \) if and only if it is feasible for \( A \cup B \setminus \{\ell_i\} \). (A similar algorithm holds if \( x' > t \), and all the feasible points must have \( x \) coordinates smaller than \( t \). Or if the two lines belong to \( B \).)

10.F. (50 Pts.) Imitating the algorithm seen in class, describe a linear time algorithm, using the above (in detail), describe a linear time algorithm that decides if \( A \cup B \) has a feasible point, and if so outputs it.

11 (100 Pts.) Maximum submatrix.
The input is a matrix \( M[1 \ldots n][1 \ldots n] \) of real numbers (potentially positive and negative). The \textit{value} of a submatrix \( M[b \ldots c][d \ldots e] \) is
\[
v(M[b \ldots c][d \ldots e]) = \sum_{i=b}^{c} \sum_{j=d}^{e} M[i][j].
\]
Describe an algorithm, as fast as possible, that computes the maximum value submatrix of \( M \).

12 (100 Pts.) Find the sink.
You are given an implicit DAG \( G \) defined over the set of vertices \( V = [n]^2 \), where \( [n] = \{1, \ldots, n\} \).
There are weights of the vertices, and you can retrieve the weight of any vertex \( v \), by calling a given function \( f(v) \). Such a call takes constant time. You can assume all the weights on the vertices are distinct.

There is an edge \((v, v')\) between two vertices \( v = (i, j) \) and \( v' = (i', j') \) in this DAG, if and only if \( |i - i'| + |j - j'| = 1 \) and \( f(v) < f(v') \). (Namely, any two adjacent vertices in the grid \([n]^2\) are connected by an edge, with the direction of the edge is determined by the value of \( f \).)

Note, that the input here is just the function \( f \), and the number \( n \). Describe a \textit{recursive} algorithm, as fast as possible, that computes a sink in this DAG. What is the running time of your algorithm? How many calls to \( f \) does it perform?