1. (15 points) Let $G = (V, E)$ be a directed graph and let $w : E \longrightarrow \mathbb{R}_{>0}$ be a function that assigns to every edge of G a *width*. For a directed path $p = \langle e_1, \ldots, e_k \rangle$ we define its *width* as $w(p) = \min\{w(e_i) \mid 1 \leq i \leq k\}$.

   For two nodes $s, t \in V$ we define the *accessibility of t from s* as the maximum width of any directed path from s to t. (If G models a road network and $w(e)$ models the width of a road, then the accessibility tells you the widest vehicle that you can send from s to t over the road network.)

   Design an efficient algorithm for computing the accessibility of t from s given s and t. Give an argument why you algorithm is correct. Analyze its running time.

2. (10 points) In class you saw the following universal set of hash funtions: The key universe was $U = \{0, \ldots, t-1\}^d$ for some $d > 1$. For $a = (a_1, \ldots, a_d) \in U$ we defined the function

$$h_a(x) = \sum_{1 \leq i \leq d} a_i \cdot x_i \bmod t$$

which maps any $x = (x_1 \ldots, x_d) \in U$ to some value in $\{0, \ldots, t-1\}$. Finally we proved that the set of functions

$$\mathcal{H} = \{h_a \mid a \in U\}$$

was universal, **provided that t is a prime number.**

Is this last condition really necessary? Is the set $\mathcal{H}$ also universal if t is a power of 2?

3. (15 points) In class you saw an algorithm that given a set S of n points in the plane computes the convex hull of S. To be more exact, the algorithm returns a circular list containing the points of S that constitute corners of the convex hull in order as they appear counter-clockwise around the convex hull.

   The algorithm(s) that you saw have running time $O(n \log n)$. Is this best possible?

   Consider the following argument why no improvement should be possible:

   Here is an algorithm for sorting a set A of n real number:

   Sort(A)
   1. Compute the set of planar points $S = \{(a, a^2) \mid a \in A\}$
   2. Compute the convex hull of S
   3. From the circular list returned in 2. recover the sorted order of A

   (a) Show that step 3 of the above algorithm can indeed be correctly realized, and that this is possible in time $\Theta(n)$.

   (b) Step 1 of the above algorithm clearly takes time $\Theta(n)$. Thus if $f(n)$ denotes the worst case running time of step 2, the total running time of this sorting algorithm is $f(n) + \Theta(n)$.

   In class we showed that any comparison-based sorting algorithm has worst case running time $\Omega(n \log n)$.

   Is it now correct to conclude that computing that $f(n)$ must be $\Omega(n \log n)$, i.e. computing convex hull of n points in the plane must take time $\Omega(n \log n)$ in the worst case?

4. (10 points) We define the *upper hull* of a planar point set S to be the chain of those edges of the convex hull of S that bound the convex hull from above. Let us denote this chain by $UH(S)$.

   We are interested in maintaining $UH(S)$ under insertions of points into S. In particular we want to maintain the point set S so that

   (i) the call S.ENUMERATE() lists the corners along $UH(S)$ in time proportional to their number, and

   (ii) the call S.INSERT(p : point) incorporates point p into S in logarithmic *amortized time.*

   (a) Explain the representation that you will use.
   (b) Explain how using your representation you will realize the two operations ENUMERATE and INSERT.
   (c) Prove that a call to INSERT only takes logarithmic amortized time.

   You may assume non-degeneracy meaning no two points encountered have the same x-ccordinate.

5. (10 points) Let S be a finite point set in the plane. In class, we showed how to compute the convex hull of S by sweeping. A triangulation of S is a division of the convex hull of S into triangles none of which contains a point of S in its interior. Show how to compute a triangulation of S by sweeping.

6. (10 points) Consider an $N \times M$ checker board from which some of the squares are missing. We want to decide whether the remaining squares can be covered by dominos so that no two dominos overlap (a domino covers two adjacent squares, one black and one white). Describe an algorithm. What is its running time?

7. (20 points) Let $G = (V, E)$ be a directed graph, let cap be a nonnegative capacity function on the edges of G and let c be a nonnegative cost function on the edges of G. Let s and t be two designated nodes.

   (a) What is the definition of a flow from s to t and what is its cost? What is the value of the flow?

   (b) Given a flow f from s to t. How can you check whether the flow is maximum?

   (c) Given a flow f from s to t. How can you check whether f is a minimum cost flow of value $val(f)$.

   (d) Assume the capacity of some edge is increased by one. Show that the value of the maximum flow can grow by at most one. Given a maximum flow for the old network, show how to compute a maximum flow for the modified network. What is the running time of your solution?

   (e) Given a maximum flow of minimum cost for the old network, show how to compute a maximum flow of minimum cost for the modified network. What is the running time of your solution.

   For questions (d) and (e) you may assume that capacities and costs are integral and that the old flow is integral.

8. (15 points) A truck is on a trip through cities 1, 2, to n. There is a request for transporting r_{ij} units from city i to city j. For each unit transported, the truck driver will receive a fare of f_{ij} Euros. The truck has a capacity of L and therefore the driver will not be able to satisfy all requests. Design an algorithm that computes an optimal schedule, i.e., a schedule maximizing revenues.