1. (15 points) Consider the closest pair of points problem discussed in the class (Section 5.4), but now suppose that the distance between two points \((x_i, y_i)\) and \((x_j, y_j)\) is measured as \(|x_i - x_j| + |y_i - y_j|\) instead of the Euclidean distance. Modify the algorithm discussed in the class to solve this problem.

2. (15 points) Consider an \(n \times n\) table where some cells are marked as forbidden. A frog is placed in the \((1,1)\)-cell (that is lowest and left most cell). At every step, the frog can jump from its current cell \((i, j)\) to any cell \((i', j')\) that satisfies \(i' + j' = i + j + 1\) provided that the cell is not forbidden. We assume that \((1,1)\) and \((n,n)\) are never forbidden.

Design and analyze an efficient algorithm that given the coordinates of the forbidden cells, finds the number of different paths that the frog can take to move from \((1,1)\) to \((n,n)\).

3. (15 points) Design and analyze a \(O(n^3)\) algorithm that given a sequence of \(n\) distinct numbers, finds the largest subsequence, such that each number is between the previous two numbers in the subsequence. In other words if \(a,b,c\) are three consecutive numbers in the subsequence, then \(\min(a,b) < c < \max(a,b)\). For example if the input is 1,3,9,8,5,7,6, then the output is 1,9,5,7,6.

4. (15 points) We are given a set of \(n\) jobs. The \(i\)-th job has a deadline \(d_i\), requires processing time of \(t_i\), and creates a profit of \(p_i\) only if it is finished before the deadline. The numbers \(d_i, t_i, p_i\) are all positive integers. We have a single processor (that can start processing at time 0), and we want to select a subset of jobs that will create the maximum possible profit. Design and analyze an algorithm that performs this task.

5. (20 points) We are given a directed graph with \(n\) nodes, and each node is labeled with the currency of a country (e.g. Canadian dollar, Yuan, Kyat, Euro, Ariary, etc). Every (directed) edge \(uv\) is labeled with a real number \(0 < \alpha_{uv}\) signifying the fact that 1 unit of the currency \(u\) can be exchanged with \(\alpha_{uv}\) units of the currency \(v\). We want to see if we can make money by just buying and selling currencies. In other words, we want to find a sequence of currencies \(u_1, \ldots, u_k\) such that, we can start from one unit of \(u_1\), and exchange it with \(\alpha_{u_1u_2}\) units of \(u_2\), and exchange that with \(\alpha_{u_2u_3}\alpha_{u_1u_2}\) units of \(u_3\), etc, and eventually exchange the amount of \(u_k\) that we obtained with \(u_1\), so that we will end up with more than one unit of \(u_1\).

Design and analyze a polynomial-time algorithm that tells us whether this is possible.

6. (20 points) Design and analyze an efficient algorithm that given an undirected graph \(G = (V, E)\), and two nodes \(s, t\) finds the number of shortest paths from \(s\) to \(t\).