22C:31 Homework 1

Due in class on Tuesday, Feb 9th

This homework will be graded out of 60 points and it is worth 6% of your grade. The Teaching Assistant will grade some 6 out of the 7 problems, with each problem worth 10 points.

1. Consider the following (informally described) *greedy* algorithm for the TRAVELING SALESPERSON PROBLEM (TSP).

   > Find a closest pair (p, q) of points from the input set. Points p and q (in any order) form the first two points in your sequence. Now repeatedly pick from among the "not yet picked" points, a point that is closest to the last point in the sequence (with ties broken arbitrarily). Append the newly picked point to the sequence. Do this until all points are picked.

   (a) The intuition for this algorithm is that it tries to get to the next city by traveling as short a distance as possible. However, this greedy algorithm does not always produce an optimal traveling salesperson's tour. Demonstrate this by constructing a *counterexample*, i.e., an input set of points for which there exists a tour that is shorter in length than the tour constructed by the above greedy algorithm. For your counterexample, clearly show what the greedy algorithm produces and also identify a shorter tour.

   (b) Now suppose that I provide to you a data structure, call it `X`, that can maintain a set of points in $\mathbb{R}^2$ under the following operations:

   - `X.delete`(p): This operation deletes point p from the current set of points.
   - `X.closest`(p): This operation returns a point closest to point p from among the current set of points.

   Further suppose that both operations run in $O(\sqrt{n})$ time each, where n is the number of points in the current set. Describe using pseudocode an implementation of the greedy algorithm shown above that runs in $o(n^2)$ (i.e., strictly less than quadratic) time, where n is the number of points in the input. Analyze the running time of your implementation and show that it indeed runs in $o(n^2)$ time.

2. Consider the *weighted interval scheduling* problem described in Chapter 1, Page 14. There is a comment towards the end of the discussion on this problem that says "There appears to be no simple greedy rule that walks through the intervals, one at a time, making the correct decision in the presence of arbitrary values." "Check" this comment by providing simple counterexamples to the following simple greedy algorithms.

   (a) Repeatedly consider an interval with highest value, add it to the solution and delete it and all other intervals that overlap with it from the set of intervals being processed.

   (b) Define the *credit* of an interval as the ratio of its value to the number of other intervals it overlaps. Intuitively, it seems like intervals with high credit should be picked up quickly to be in our solution because they provide a lot of value, yet do not rule out many other intervals. Here is the corresponding greedy algorithm. Repeatedly consider an interval with highest credit, add it to the solution and delete it and all other intervals that overlap with it from the set of intervals being processed.

   In both cases, assume that ties are broken arbitrarily.

3. Faced with a great deal of difficulty in designing efficient, correct algorithms for problems such as the *Maximum Independent Set (MIS)* problem (Chapter 1, Page 6), researchers have resorted to using approximation algorithms. Let c be a constant such that $0 < c < 1$. Then a *c-approximation algorithm* for the MIS problem is a polynomial-time algorithm that, for any input graph $G = (V, E)$, outputs a subset $S \subseteq V$ whose size is at least c times the size of a largest independent set. For example, if $c = 0.75$, then a c-approximation algorithm would be guaranteed to yield an independent set whose size is at least 75% of the size of the largest independent set.

   This problem guides you towards an example that shows that a simple and natural greedy algorithm for MIS is not a c-approximation for any constant c. Consider the greedy algorithm for MIS that repeatedly considers a vertex with smallest degree in the graph, adds it to the solution, and deletes it and all its neighbors from the graph.

   (a) Consider a graph with vertex set $\{a, b_1, b_2, b_3, c_1, c_2, c_3\}$. Connect vertex a to each b_i, $i = 1, 2, 3$, connect every b_i, $i = 1, 2, 3$ to every c_j, $j = 1, 2, 3$, and finally connect c_1, c_2, and c_3 to each other. For this graph show the independent set produced by the greedy algorithm and also the largest possible independent set.

   (b) Generalize this example, to come up with a graph for every positive integer $k \geq 3$ in which the degree-based greedy algorithm produces an independent set of size 2, whereas the graph contains an independent set of size k for any positive

   (c) Argue why this family of graphs shows that the degree-based greedy algorithm is not a c-approximation for MIS for any constant c, $0 < c < 1$.

4. There are two algorithms called Alg1 and Alg2 for a problem. On an input of size n, Alg1 runs in n^2 microseconds and Alg2 runs in $100n \ln n$ microseconds. Alg1 can be implemented using 4 hours of programmer time and Alg2 requires 15 hours of programmer time. If programmers are paid 20 dollars per hour and CPU time costs 50 dollars per minute, how large a problem instance must we be solving in order to justify the extra programming cost for Alg2? Show all your calculations carefully.

5. Analyze the running time of the following code fragments. Express your answers in an asymptotically tight manner, i.e., in the form $\Theta(f(n))$, where $f(n)$ is some function of n.

   (a)
```
for $i \leftarrow$ 1 to $n$ do
    $j \leftarrow n$
    while ($j \geq 1$) do
        $j \leftarrow j/2$
```

   (b)
```
for $i \leftarrow$ 1 to $n$ do
    $j \leftarrow i$
    while ($j \geq 1$) do
        $j \leftarrow j/2$
```

6. Problem 6, Chapter 2, Pages 68-69.

7. Problem 8, Chapter 2, pages 69-70. This is a fun problem and the hint for part (a) is that $f(n) = \sqrt{n}$.