CS 787
Fall 2024
Homework #1

Due in class Friday, October 4, 2024

Rules for Homework.

i. Everyone must do his or her own work. Use of any sources other than class notes and recommended readings should be accompanied by a citation. In any case, there should be significant "value added" by the student's work.

ii. Starred problems have research potential. These are optional, as their difficulty is unknown (at least to me).

1. The lecture on 9/13/24 presented a "collision" algorithm to solve $x^2 + y^2 = n$ in integers. With some improvements suggested by the class, the algorithm is: In parallel, enumerate x^2 for $0 \le x \le B$ and $n - y^2$ for $B \ge y \ge 0$. (B denotes $\lfloor \sqrt{n} \rfloor$.) By merging these two lists, identify suitable pairs (x, y). Note that both lists are generated in increasing order, so that space to store them is not required.
   a) If (x, y) is a solution, so is (y, x). We can therefore restrict attention to pairs with $x \le y$. Modify the algorithn to incorporate this idea. You should get a constant-factor improvement in the number of operations used.
   b) As the algorithm is stated above, the bulk of the work will be $O(\sqrt{n})$ multiplication operations. Modify so that it uses $O(\sqrt{n})$ addition/subtraction operations. You are allowed $O(1)$ more expensive operations, such as multiplication, division, etc.

2. Suppose that the data for a longest increasing subsequence problem are random. (E.g. X_i i.i.d. uniform on (0,1), or the X_i are a random permutation of $1..n$.)
   a) Write down an exact formula for the expected number of increasing subsequences of $X_1 \dots X_n$.
   b) Show that this grows more quickly than any polynomial in n. Thus, if you use only the first pruning rule that Pruhs mentions, it will not be sufficient to get a polynomial time algorithm, even on average.

3. Recall the min-cost set cover problem: Given positive-cost $S_1, \dots, S_n \subset X = \{1, \dots, m\}$, find a minimum-cost collection of these subsets whose union is X. In class, we studied a greedy algorithm that solves this approximately.
   a) Let $y_1, \dots, y_m$ be the prices paid by the greedy algorithm. Show that if $(y_1, \dots, y_m)$ is dual feasible, then the greedy solution is optimal. (Note that the converse is not true.)
   b) Give examples to show that this is possible, for every $m \ge 1$.
   c) Explain why a minimal cover found as in a) has a short, easily checkable proof of optimality.
   d) (*) How likely is it that such "greedy prices" are dual feasible?