Please enter your name and uID below.

Name:

uID:

Collaborators, if any, and how you collaborated:

Submission notes

- Due at 11:59 pm on Thursday, October 20th.
- Solutions must be typeset using one of the template files. For each problem, your answer must fit in the space provided (e.g. not spill onto the next page) *without* space-saving tricks like font/margin/line spacing changes.
- Upload a PDF version of your completed problem set to Gradescope.
- Teaching staff reserve the right to request original source/tex files during the grading process, so please retain these until an assignment has been returned.
- Please remember that for this problem set, you are allowed to collaborate in detail with your peers, as long as you cite them. However, you must write up your own solution, alone, from memory. If you do collaborate with other students in this way, you must identify the students and describe the nature of the collaboration. You are not allowed to create a group solution, and all work that you hand in must be written in your own words. Do not base your solution on any other written solution, regardless of the source.

1. (Basic arithmetic expression, 45pts)

   A ***basic arithmetic expression*** is composed of characters from the set $\{1, +, \times\}$ and parentheses. Almost every integer can be represented by more than one basic arithmetic expression. For example, all of the following basic arithmetic expression represent the integer 14:

$$1+1+1+1+1+1+1+1+1+1+1+1+1+1$$
$$((1+1)\times(1+1+1+1+1))+((1+1)\times(1+1))$$
$$(1+1)\times(1+1+1+1+1+1+1)$$
$$(1+1)\times(((1+1+1)\times(1+1))+1)$$

   Describe and analyze an algorithm to compute, given an integer n as input, the minimum number of 1s in a basic arithmetic expression whose value is equal to n. The number of parentheses doesn't matter, just the number of 1s. For example, when $n = 14$, your algorithm should return 8, for the final expression above. The running time of your algorithm should be bounded by a small polynomial function of n.

   By *describe*, we mean the same requirements as for PS3. To solve this question, you should follow the same process we have for all other dynamic programming problems so far. Namely, if you're trying something more complicated and clever than "try every choice, recurse on subproblems, and return the best", you should reconsider your approach!

   You should start by asking yourself: "If I wanted to find the minimum number of 1s in a basic arithmetic expression (BAE) with value of n, how could I write it as an expression in terms of BAEs for $n-1$ through 1? What *choices* could I make in how to write the expression for n that would correspond to smaller subproblems? For this problem, what would it mean to 'try all possibilities'?"

2. (Square peg in a round hole, 35pts) Determine a greedy algorithm for solving the PS4 Programming problem, and then, using complete sentences, describe it, and prove its correctness using an inductive exchange argument.

   You can assume your algorithm has input $N[1..n]$ (a list of circular plots each with radius $N[i]$), $M[1..m]$ (a list of circular houses each with radius $M[i]$), and $K[1..k]$ (a list of square houses with side length $K[i]$).

   Note that a square house with side length s will fit into a circular plot of radius r if and only if $\frac{s}{\sqrt{2}} < r$. You should start by simplifying your problem by converting your square plots to their equivalent circular sizes.

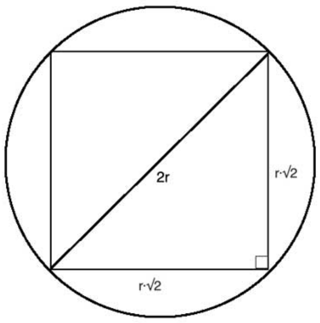

Figure 1: From https://geometryhelp.net/square-inscribed-in-a-circle

   Hint: Once you have an idea for a greedy algorithm, your correctness proof should be similar to that of scheduling (posted in-detail on Piazza). However, the notation for representing the matching of houses to plots can be a bit trickier. You are welcome to copy the structure below, but you must replace the parts in italics.

   First argue that you can convert your square houses into equivalent circular houses, $S[1..k]$.

   We can combine our two house lists into one, $X[1..m + k] = \mathit{Append}(M, S)$.

   Describe your greedy algorithm here.

   For notational simplicity, we can sort the lists N, X from smallest to largest – this will not affect the correctness of our proof. Assume, for the sake of contradiction, that our greedy algorithm is not optimal. This means that for some input, our greedy algorithm matches p houses to plots, yet there is some different algorithm that matches $q > p$ houses to plots. We can represent the output of a particular algorithm by listing the plot indices it chooses for each house, or -1 if it did not match the house.

   So, these two outputs can be represented with two sequences,

$$\langle g_1, \ldots, g_j, \ldots, g_{k+m} \rangle$$
$$\langle o_1, \ldots, o_j, \ldots, o_{k+m} \rangle$$

   Where g has a total of p non-negative plot indices in the whole list and o has a total of q.

   The bulk of your proof should go here. Point out that since the algorithm outputs differ, there must be some first difference between these lists of plot indices. Then, argue that no matter what that difference was, you can do an exchange without making the optimal algorithm worse off. You will likely need to use facts about how exactly your greedy algorithm works, and also consider exchanges that involve a -1 plot index. Lastly, you should argue why, after all exchanges are done via induction, the result is contradictory.