CS180 Homework 4

Due: June 5, 11:59pm

1. (25 pt) (Lucas) A palindrome is a string that reads the same from left to right and from right to left. Design an algorithm to find the minimum number of characters required to make a given string to a palindrome if you are allowed to insert characters at any position of the string. For example, for the input "aab" the output should 1 (we'll add a 'b' in the beginning so it becomes "baab").

   The algorithm should run in $O(n^2)$ time if the input string has length n.

   Solution
   OPT(str,n)

   ```
   table← $[n, n]$ zero array
   $r, c$, idx ← 0
   for idx in range(1,n) do
     r = 0
     for c in range(idx, n) do
       if str[r] == str[c] then
         table[r][c] = table[r + 1][c - 1]
       else
         table[r][c] = min (table[r][c - 1] + 1, table[r + 1][c]) + 1)
       end if
       r += 1
     end for
   end for
   return table[0][n - 1];
   ```

   Run OPT(string, len(string))

   Essentially all we are doing is looking at all sub-strings of length 1, then 2, etc. up until length n. We store the minimum number of characters required to make each sub-string a palindrome and then use that information from smaller sub-strings to calculate the answer for longer sub-strings until we get to the entire string itself.

   Adding character to r or c, (last on left or right don't match, have to add to one side or the other)

2. (25 pt) (Xuanqing) Consider the problem of "Approx-3SAT": The input is the same as 3-SAT, which is a boolean expression $C_1 \wedge C_2 \wedge \cdots \wedge C_n$ where each C_i is an "or" of three literals (each literal can be one of $x_1, \ldots, x_m, \neg x_1, \ldots, \neg x_m$). Note that we assume a clause cannot contain duplicate literals (e.g., $(x_1 \vee x_1 \vee x_2)$ is not allowed). Instead of determining whether there's a truth assignment on $\{x_i\}_{i=1}^m$ that satisfies this boolean expression (which means it satisfies all the clauses), now we want to determine whether there's an assignment that satisfies at least $n-1$ clauses. Prove that 3-SAT can be polynomial time reduced to Approx-3-SAT.

   solution:

   Given a 3-SAT, add a set of dummy clauses that there's always exact one clause not satisfiable. For example, we can add $(z_1 \vee z_2 \vee z_3)$, $(z_1 \vee z_2 \vee \bar{z}_3)$, $(z_1 \vee \bar{z}_2 \vee z_3)$, $(z_1 \vee \bar{z}_2 \vee \bar{z}_3)$, $(\bar{z}_1 \vee z_2 \vee z_3)$, $(\bar{z}_1 \vee z_2 \vee \bar{z}_3)$, $(z_1 \vee \bar{z}_2 \vee z_3)$, $(z_1 \vee \bar{z}_2 \vee \bar{z}_3)$, so there will be exactly one violation and the 7 of them are satisfied for any truth assignment. Therefore, this newly constructed instance (by adding those dummy clauses) will be yes instance of Approx-3SAT if and only if the original instance is a yes instance for 3-SAT.

★ Homework assignments are due on the exact time indicated. Please submit your homework using the Gradescope system. Email attachments or other electronic delivery methods are not acceptable. To learn how to use Gradescope, you can:

  - 1. Watch the one-minute video with complete instructions from here:
    https://www.youtube.com/watch?v=-wemznvGPfg
  - 2. Follow the instructions to generate a PDF scan of the assignments:
    http://gradescope-static-assets.s3-us-west-2.amazonaws.com/help/submitting_hw_guide.pdf
  - 3. **Make sure you start each problem on a new page.**

★ We recommend to use LaTeX, LyX or other word processing software for submitting the homework. This is not a requirement but it helps us to grade the homework and give feedback. For grading, we will take into account both the correctness and the clarity. Your answer are supposed to be in a simple and understandable manner. Sloppy answers are expected to receiver fewer points.

★ Unless specified, you should justify your algorithm with proof of correctness and time complexity.