CSE 417
Algorithms & Computational Complexity
PARTIAL DRAFT Assignment #7
Due: Wednesday, 3/6/19

Turnin: Gradescope again; @uw email and gradescope password as before. On-time turn-in deadline is 11PM.

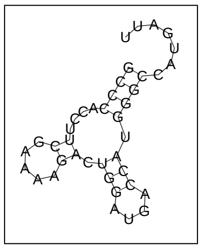
The following problems relate to RNA secondary structure prediction (aka "RNA folding"), as described in lecture and in section 6.5 of the text. In short, you will implement Nussinov's algorithm, together with its associate "traceback" routine.

Input: You should have a subroutine named "Nussinov" that has a single parameter which is a string of letters $x_1x_2 \ldots x_n$ from the 4 letter alphabet $\{$A,G,C,U$\}$ (all uppercase, for simplicity), as in line [1] below. Your main program should read a sequence of lines from "standard input" each containing one such string, and call "Nussinov" on each. Different lines may be of different lengths (different "n"). For timing purposes, you should also generate *random* sequences of length, say, $n = 2^k$ for $4 \leq k \leq 12$ or more, with A,C,G,U equally likely. Generate at least one sequence of each length, preferably several. Call "Nussinov" on each. Generating and processing these random sequences should be *optional*, and by default this option should be *"off"* so that our hard-working TAs don't have to endure the wait time, but it should be obvious how to enable it. E.g., your main program might look like:

```
while(!end_of_file(STDIN)){
  seq = read_a_line();
  Nussinov(seq);
}
if(FALSE){
  for(k=4; k<=12; k++){
    Nussinov(random_seq(2^k))
    Nussinov(random_seq(2^k))
    Nussinov(random_seq(2^k))
  }
}
```

Output: For each "Nussinov" call, print to standard out one line containing the input, like [1] below, a second line containing (one of) its optimal structure(s), formatted as in [2] below, plus a third line giving (i) the length of the input, (ii) the total number of pairs in that structure, and (iii) the time, in seconds (or fractions thereof) as in [3] (note that the structure/count shown are not optimal for this sequence). (iv) Additionally, for a length n input, if $n < 25$, print the $n \times n$ OPT matrix calculated by Nussinov's algorithm; print one line per row with n white-space-separated integer values per line, preferably keeping columns vertically aligned. (v) Follow all of this by one blank line. E.g.:

```
GCCCACCUUCGAAAAGACUGGAUGACCAUGGGCCAUGAUU                    [1]
((((....((.....)).(((....))).)))).......                    [2]
Length = 40, Pairs = 9, Time = 0.0001 sec                   [3]
                                                            [4]
```



I say "one of" since there may be different structures with equal numbers of pairs, often slight variants of each other. Giving any one of them is OK. The structure line will be a string of parens and dots, vertically aligned with the input string. A dot in the structure line means that the corresponding position in the RNA is unpaired; a left paren means it is paired with a position to its right, marked by a right paren. Furthermore, parens must be properly balanced/nested, so specific paired positions are marked by "matching" left/right parens. sequence).

Method: Use the Nussinov algorithm, described in section 6.5 of the text, and my slides. (Again note that the conventions used in the book and the slides differ; *clearly state which you are using*.)

What You Need To Do:

1. [30 points] Implement the Nussinov algorithm for calculating $\textsc{Opt}[i, j]$.
2. [20 points] Devise and implement an algorithm to construct and print the structure (i.e., the string of parens and dots). This is a "traceback," similar to ones we've seen with other dynamic programming algorithms. I strongly recommend that you look for a recursive algorithm to do this, but it is not required. You may (or may not) find it convenient to create auxiliary data structures while you're building $\textsc{Opt}$ to facilitate the traceback.

   As stated above, print the input, with the structure aligned vertically below it, and also print the number of pairs. Additionally, print the $\textsc{Opt}$ matrix if $n \leq 25$. All output should go to standard out.
3. [20 points] Write a description of your traceback algorithm, explaining how it works/why it is correct.
4. [10 points] Analyze (separately and collectively) the (big-O) run time of the algorithms in problems 1 and 2.
5. [10 points] Measure the actual run time of your algorithm (total time for both parts) on random RNA sequences of length 20–2000, say, plot them on a graph (e.g., Excel might be convenient, but is not required), and discuss how this compares to the theoretical performance predicted in problem 4. For some tips on how to do the timing, see the FAQ page.

Test Cases: TBD

Language: You may use C, C++, C#, Haskell, Java, Lisp, ML, Perl, Python, R, or Ruby; talk to me before beginning if you prefer something else.

What/How To Turn In: TBD