CSE 417 HW 7

Zachary McNulty (zmcnulty 1636402)

March 2019

1 Problem 5

Traceback Algorithm

My traceback algorithm works by creating an auxillary datastructure, $BACKLINKS$, which is a hash table that stores which subproblems were used to calculate $OPT[i, j]$. For example, if we found that $OPT[i, j] = OPT[i, j-1]$, case 2 of the Nussinov algorithm where no pair occurs at base j, then:

$$BACKLINKS[(i, j)] = [(i, j-1)]$$

If we instead have the other important case, case 3 of the Nussinov algorithm where base t forms a pair with base j, we would have:

$$BACKLINKS[(i, j)] = [(i, t-1), (t+1, j-1)]$$

As is evident above, we use (i, j) as the key and the list of the subproblems as the value. This data structure allows us to quickly run the traceback. Starting with a sequence of all dots, we add left and right parentheses where appropriate as we recurse down through all the subproblems. Below is a more specific procedure for how this process is performed.

1. Initialize the pattern that will represent the base-pairing relationships with n dots where there are n bases in the full RNA sequence.

   $$pattern = ...$$

2. Start with $i = 0$ and $j = n - 1$, i.e. the $OPT[i, j]$ value corresponding to the optimal value within the full-length of the sequence.

3. With $OPT[(i, j)]$, use $BACKLINKS$ to determine which of the following is true:

   (a) $BACKLINKS[(i, j)] = [(i, j-1)]$

   (b) $BACKLINKS[(i, j)] = [(i, t-1), (t+1, j-1)]$

4. If case a) is true, there is no base pair at j so leave *pattern* unchanged and recursively repeat step 3 with the subproblem $OPT[i, j-1]$.

5. If case b) is true, then bases t and j form a pair: set $pattern[t] =$ "(" and $pattern[j] =$ ")" as t is strictly less than j. Recursively apply step 3 to both the subproblems $OPT[i, t-1]$ and $OPT[t+1, j-1]$

6. If there is no banklink for (i, j), you have reached the base case where $j - 4 < i$ and can recurse upwards.

Essentially, the algorithm just finds the spots where we split into case 3 when calculating OPT as those splits were the points where we decided a pair between base t and base j was optimal.

2 Problem 6

Complexity Analysis: Calculating OPT

There are $O(N^2)$ entries in our OPT table as there are N possible starting points i of the subsequence/subproblem of interest and N ways to choose the end. Even though we can ignore many of these because the only reasonable choices of i, j have $i \leq j$ and more specifically $i < j - 4$ due to the "no tight turns" condition, we still see that:

$$\sum_{i=0}^{N-1} N - 5 - i = N^2 - 5N - \sum_{i=1}^{N-1} i = N^2 - 5N - \frac{N(N-1)}{2} \in O(N^2)$$

Futhermore, each entry of this table requires an amount of work roughly proportional to the length of the subsequence as we have to find the maximum across all possible split points t. Thus, a logical way in which we can consider all subsequences is to consider them in order of their size k. For a given k, there are $n - k + 1$ starting points in the sequence we could choose. Choosing the endpoint j to be possibly part of a pair there are $k - 5$ other bases in the subsequence that j could pair with. For each of these, we perform only a few constant time operations (check if t, j are base compatible, array lookups, arithmetic, comparisons, hashing to the $BACKLINKS$ dictionary) so overall our complexity is roughly proportional to:

$$\sum_{k=5}^{N} \sum_{i=0}^{N-k} k - 5 = \sum_{k=5}^{N} (N - k + 1)(k - 5) = \frac{1}{6}(N^3 - 12N^2 + 47N - 60) = O(N^3)$$

Thus, overall the main algorithm is $O(N^3)$

Complexity Analysis: Traceback

The $BACKLINKS$ data structure is calculated within the OPT routine above at no extra cost, so we will not factor that into our analysis of the traceback. The traceback algorithm breaks the problems into three cases:

1. $i > j - 4$ in which case no further base pairs can occur so we can leave *pattern* as it is.
2. $OPT[i, j]$ got its value from $OPT[i, j - 1]$ because it was not optimal to place a base pair at j.
3. $OPT[i, j]$ got its value from $1 + OPT[i, t - 1] + OPT[t + 1, j - 1]$ for some $i \leq t < j - 4$ because it was optimal to place a new pairing between base j and base t. In this case, we update *pattern* to reflect the new base pair: $pattern[t] =$ "(" and $pattern[j] =$ ")".

In case one, we clearly have the recursive relationship $T(n) = T(n-1)$ as all that happens is that the interval shrinks in length by 1. In case 2, we have the problem split into two subproblems recursively, $OPT[i, t-1], OPT[t+1, j-1]$. Thus, we see that our traceback algorithm runs in (where n is the length of the interval n = j - i):

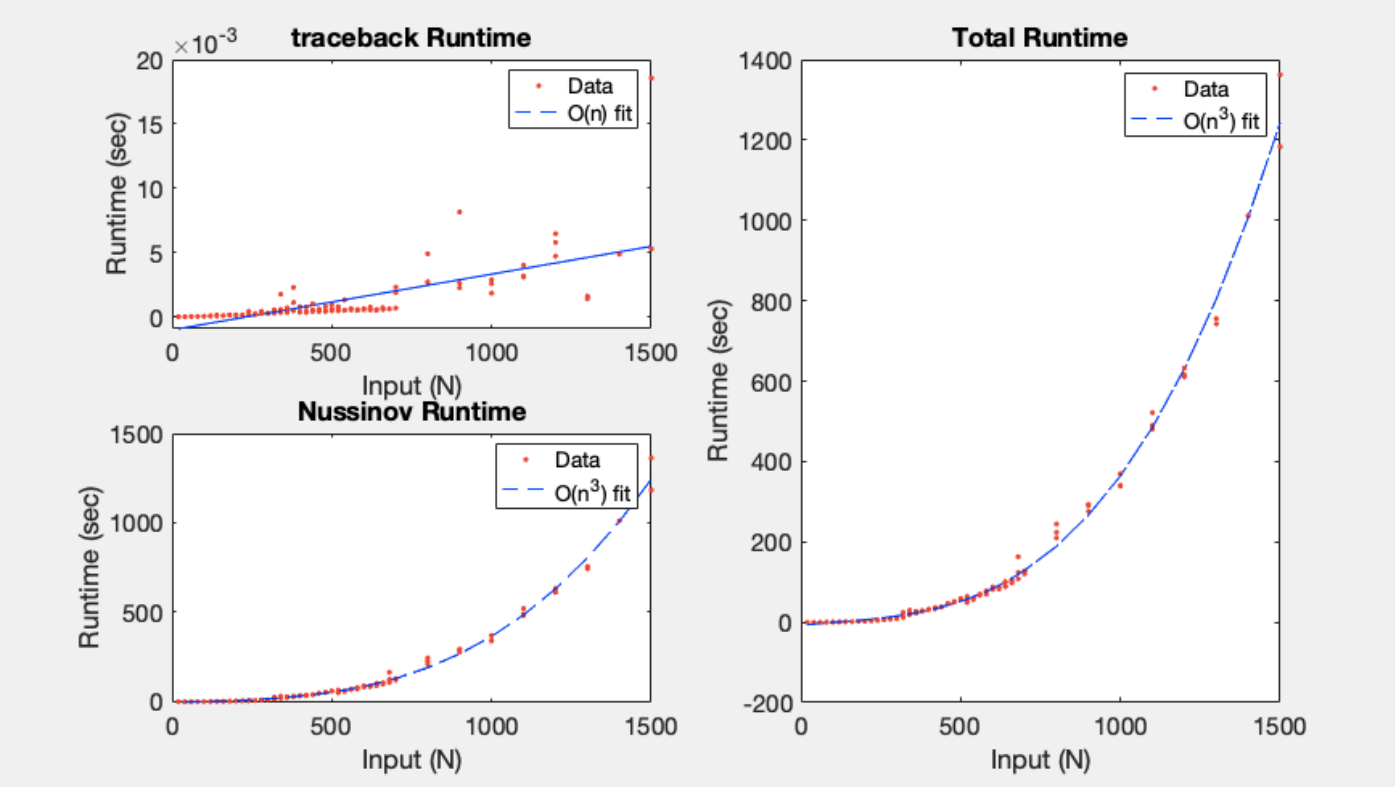
$$T(n) = \begin{cases} O(1) & i > j - 4 \\ T(n-1) + O(1) & OPT[i,j] = OPT[i, j-1] \\ T(k) + T(n-k-2) + O(1) & OPT[i,j] = 1 + OPT[i, t-1] + OPT[t+1, j-1] \end{cases}$$

where k is the length of the first subinterval in the third case. If there were no base pairs and we only went into the second case, our worst runtime would be $O(N)$. If we were only to go into the third case, as there is only constant work at each level it is clear the majority of the work will come from the recursive work. Worst case scenario would be $k = 1$ always as we get the worst division of work and the most possible leaf nodes, but again this only yields a worst case scenario that is $O(N)$ rather than the $O(log(N))$ we could get from a perfect binary division. Thus, overall we see that the traceback runs in $O(N)$.

Collective Complexity

The time complexity of the main algorithm is $O(N^3)$ which far outwieghs the runtime of the traceback which is merely $O(N)$. Thus, overall the algorithm is still $O(N^3)$.

Problem 7



Here we have the actual runtimes for the Nussinov algorithm (calculating OPT table), the traceback, and the total runtime. Above we can see that the actual runtimes behave fairly close to our theoretical approximations. The traceback part of the code barely contributes to the total runtime at all, running in less than a second. Although we might not have enough data points to truly say whether or not the traceback algorithm is linear, it certainly appears to be that way and that idea is supported by our theoretical analysis. Furthermore, it is certainly faster than the main part of the Nussinov algorithm which we theoretically predicted at O(n^3). Here, we see that the Nussinov Algorithm is well fit by a third-order polynomial, giving support to our theoretical claim. Similarly, the overall runtime is also seemingly cubic. Again, the traceback does not contribute much to the runtime, so the overall runtime will be the same order as the runtime of the main part of the Nussinov Algorithm.