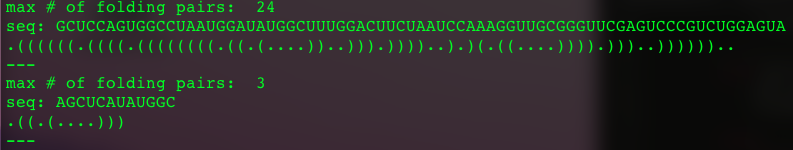
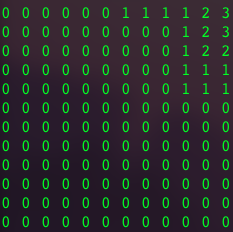
*CSE 417 HW 6 RNA FOLDING* YANG ZHANG 1030416 zhy9036@uw.edu

**RESULTS FOR 2 TEST CASE:**

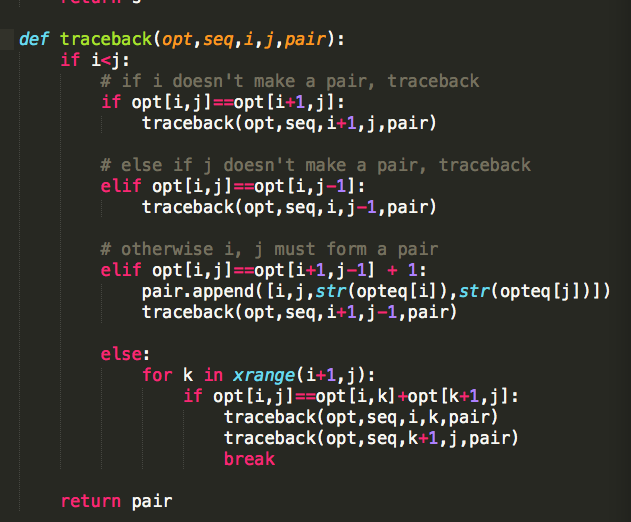
****

**OPT Matrix for small size test case:**

****

**Description of traceback algorithm**

**Code:**

****

The four input parameters stand for:

Opt => optimal count matrix (2d list)

Seq => List of RNA sequence

I => row index

J => column index

Pair => List of RNA pairs (initially empty)

Backtrace procedure:

This action can be separated into subcases

1. If opt[i,j] equals opt[i+1, j], which means the element in index **i** (the left boundary of current scope) of RNA sequence doesn’t form a pair with other elements in the rest of the sequence. Otherwise, the opt[i,j] should larger than opt[i+1, j]. Therefore in this case, we track back with smaller range, which the left boundary starts from **i+1**.
2. Else if opt[i,j] equals opt[i, j-1], which means the element in index **j** (the right boundary of current scope) of RNA sequence doesn’t form a pair with other elements in the sequence. Otherwise, the opt[i,j] should larger than opt[i, j -1]. Therefore in this case, we track back with smaller range, which the left boundary starts from **j-1**.
3. Else if opt[i,j] equals opt[i+1, j-1] + 1, we know that elements in **i** and **j** must form a pair with each other. The reason is that before this condition we already check the case that either i or j doesn’t form a pair. Therefore, in this case, we found a valid pair and appended this pair to the result list and then we trace back with smaller range **(i+1, j-1)**
4. Otherwise, the boundary elements at index **i** and **j** formed pairs but not with each other, which means the whole range are divided into two parts.

For example:

**i** **j**

(….)…(….)

Therefore, in this case, we loop over the current range. Keep moving the cursor **k,** until **opt[i, j] = opt[i, k] + opt[k+1, j],** then we know we found the boundary between the two sub set, and then we trace back with the two sub set.

**Run time analyze**

The run time for the build matrix step is **O(n2)**, because both for loop have n range and they are nested.

The run time for the trace back step is **O(n2)**, because for the worse case of trace back, every outer boundaries form a pair. I.e. for each pair in **n** total pairs, need move **n** steps to make sure its validity. Therefore, the run time is **O(n2)**

**Runtime Measurements**

|  |  |
| --- | --- |
| SIZE | TIME (ms) |
| 20 | **2.035** |
| 50 | **26.74** |
| 100 | **287.836** |
| 200 | **3263.463** |
| 500 | **97417.3219** |
| 1000 | **1447106.0359** |
| 2000 | **21687325.1345** |

**Graph of timing performance:**

From the graph, we can easily observe that the increament of time as increasing data size fits the polynomail tend line perfectly. Therefore, the runtime follows the **O(n2)**