CS 321 - Algorithms - Fall 2018 *Justin Espiritu Robert Freiberg*

Assignment 6

Due BEFORE 8:00AM on Monday 10/15/2018

On time / 20% off / no credit

Total points: 40

You are allowed to work with a partner on this assignment. If you decide to form a pair, make sure to include both names above, but submit only one zip file to D2L.

This assignment will develop your algorithmic design skills. More specifically, it will make you more comfortable with the divide-and-conquer paradigm. You must write up your solutions to this assignment IN THIS FILE using LaTeX by filling in all of the boxes below. If your submitted `.tex` file does not compile, then you will receive 0 points. You should NOT add any LaTeX packages to your `.tex` file.

Since this assignment includes a programming component, you must also complete and submit the `SLIT.java` file.

Submission procedure:

1. Complete this file, called `a6.tex`, with your full name(s) and answers typed up below.
2. Compile this file to produce a file called `a6.pdf`. Make sure that this file compiles properly and that its contents and appearance meet the requirements described in this handout.
3. Create a directory called `a6` and copy exactly three files into this directory, namely:
   - `a6.tex` (this file with all of your answers added)
   - `a6.pdf` (the compiled version of the file above)
   - `SLIT.java` (your completed Java code file)
4. Zip up this directory to yield a file called `a6.zip`
5. Submit this zip file to the D2L dropbox for A6 before the deadline above.
6. BEFORE the beginning of class on the due date above, submit a single-sided, hard copy of your:
   - `a6.pdf` file
   - completed `SLIT.java` file, making sure that your name(s) are in the top documentation block and that you stated whether or not each algorithm works as expected. For FULL credit, your hard copy of the code MUST be properly indented, contain no lines that wrap around (i.e., with a length greater than the page width), and be fully legible and easily understandable.

Problem statement

In this assignment, you will implement two solutions to the following problem:

SLIT problem

Input: An $n \times n$ matrix m whose elements come from the set {'`A`', '`C`', '`G`', '`T`'}

Output: The size s of the largest imbalance in Thymine within any slit window in m, where:

- a slit window is a rectangular sub-matrix of m with height 2 and any positive width, and
- s is computed as the total number of '`T`' elements minus the total number of '`A`', '`C`', and '`G`' elements in the window.

The first solution you will implement is given to you in pseudocode format, as follows:

Algorithm 1 Brute-force SLIT algorithm

function COUNT(m, r, c, w)
 ▷ This is a helper function that returns the total number of non-T bases subtracted from the total number of T bases within the sub-matrix of m whose top-left corner is at $r \times c$, whose width is equal to w columns, and whose height is equal to two rows
end function
function BF_SLIT(m, n) ▷ m: 2D matrix
 ▷ n: # of top rows and left columns of m that are part of this problem instance
  $slit \leftarrow 0$
  for each $row \in [0..(n-2)]$ **do** ▷ row is the index of the row in the top-left corner of the slit
   for each $col \in [0..(n-1)]$ **do** ▷ col is the index of the column in the top-left corner of the slit
    for each $w \in [1..(n-col)]$ **do** ▷ w is the number of columns in the slit
     $slit \leftarrow max(slit, count(m, row, col, w))$
    end for
   end for
  end for
  return $slit$
end function

You will have to design another algorithm that has a better asymptotic running time than the brute-force algorithm given above. Here are the detailed requirements for this assignment.

1. **(5 points) What is the big-theta bound on the worst-case running time of the brute-force algorithm?**

   Answer: $\Theta(N^4)$

   Prove your answer by computing the exact worst-case running time.

$$
\begin{array}{lll}
T(N) & = \sum_{r=0}^{n-2}\sum_{c=0}^{n-‘1}\sum_{w=1}^{n-c}\sum_{i=1}^{2}\sum_{j=1}^{w} 1 & \text{taken from the SLIT algorithm} \\
 & = \sum_{r=0}^{n-2}\sum_{c=0}^{n-‘1}\sum_{w=1}^{n-c}\sum_{i=1}^{2} \frac{w(w+1)}{2} & \text{geometric series} \\
 & = \sum_{r=0}^{n-2}\sum_{c=0}^{n-‘1}\sum_{w=1}^{n-c} 2\frac{w(w+1)}{2} & \text{simplify via property of summation} \\
 & = \sum_{r=0}^{n-2}\sum_{c=0}^{n-‘1}\sum_{w=1}^{n-c}\sum_{i=1}^{2} w(w+1) & \text{simplify} \\
 & = \sum_{r=0}^{n-2}\sum_{c=0}^{n-‘1}\sum_{w=1}^{n-c}\sum_{i=1}^{2} w^2+w & \text{simplify} \\
 & = \sum_{r=0}^{n-2}\sum_{c=0}^{n-‘1}(\sum_{w=1}^{n-c} w^2 + \sum_{w=1}^{n-c} w) & \text{distribute summation} \\
 & = \sum_{r=0}^{n-2}\sum_{c=0}^{n-‘1}(\frac{(n-c)((n-c)+1)(2(n-c)+1)}{6} + \frac{(n-c)((n-c)+1)}{2}) & \text{geometric series} \\
 & = \sum_{r=0}^{n-2}\sum_{c=0}^{n-1} \frac{2(n-c)^3+6(n-c)^2+6(n-c)+3(n-c)^2+3(n-c)}{6} & \text{simplify} \\
 & = \sum_{r=0}^{n-2}\sum_{c=0}^{n-1} \frac{2n^3-c^3-3cn^2+3c^2n+9n^2-18cn+9c^2+9n-9c}{6} & \text{distribute and simplify} \\
 & = \sum_{r=0}^{n-2}\sum_{c=0}^{n-‘1} \frac{n^3}{3}-\frac{c^3}{6}-\frac{cn^2}{2}+\frac{c^2n}{2}+\frac{3n^2}{2}+\frac{3c^2}{2}+\frac{3n}{2}-\frac{3c}{2} & \text{distribute the 6} \\
 & = \sum_{r=0}^{n-2}(\frac{n^3}{3}-\frac{1}{6}\sum_{c=0}^{n-‘1} c^3 - \frac{n^2}{2}\sum_{c=0}^{n-‘1} c & \\
 & \quad +\frac{n}{2}\sum_{c=0}^{n-‘1} c^2 - 3n\sum_{c=0}^{n-‘1} c + \frac{3}{2}\sum_{c=0}^{n-‘1} c^2 & \\
 & \quad +\frac{3n}{2} - \frac{3}{2}\sum_{c=0}^{n-‘1} c) & \text{property of summations} \\
 & = \sum_{r=0}^{n-2}(\frac{n^3}{3}-\frac{1}{6}\frac{(n-1)^2n^2}{24}-\frac{n^2}{2}(\frac{(n-1)n}{2}) & \\
 & \quad +\frac{n}{2}(\frac{n(n-1)(2(n-1)+1)}{6}) - 3n(\frac{n(n-1)}{2}) + \frac{3}{2}(\frac{n(n-1)(2(n-1)+1)}{6}) & \\
 & \quad +\frac{3n}{2} - \frac{3}{2}(\frac{n(n-1)}{2})) & \text{simplify, property of summations} \\
 & = \sum_{r=0}^{n-2} \frac{n^3}{3} - \frac{n^4-2n^3-n^2}{24} - \frac{n^4-n^3}{4} & \\
 & \quad +\frac{2n^4-3n^3+n^2}{12} - \frac{3n^3-3n^2}{2} + \frac{6n^3-9n^2-3}{12} + \frac{3n}{2} - \frac{3n^2-3n}{4} & \text{simplify} \\
 & = \sum_{r=0}^{n-2} \frac{-3n^4-14n^3+3n^2+54n-6}{24} & \text{simplify} \\
 & = \frac{-3n^4-14n^3+3n^2+54n-6}{24} & \text{simplify, property of summations} \\
 & = \frac{-n^4}{8} - \frac{-7n^3}{12} + \frac{n^2}{8} + \frac{9n}{4} - \frac{1}{4} & \text{simplify} \\
\therefore & = \Theta n^4 & \text{simplify, property of Theta}
\end{array}
$$

2. **(6 points) Implement the brute-force algorithm, that is, the two methods called `count` and `algorithm1` in the file `SLIT.java`. For this question, you may NOT modify any other**

part of the code handout besides the BODY OF THESE TWO METHODS.

Your answer to this problem will be fully contained in the `SLIT.java` *file that you submit*

3. **(8 points) Design an algorithm whose worst-case running time is asymptotically better than your answer to the previous question. Your algorithm must use the divide-and-conquer approach. You must describe in precise and concise English prose the main idea(s) behind your algorithm by filling in the following four descriptions.**

   a. Divide step

   We are going through the n by n matrix and we are going through each row. Each row starting at row 0 until n - 2. Our divide step does not start until we call maxTRec. MaxTRec needs the matrix m, the start of the window l, the end of the window r and what row we are in called row.
   We start by looking at the window whose size is given by the arguments of maxTRec. The window starts at l and ends at r. We then split that window by 2. The first half of the windows starts at l and goes to (l+r)/2 which we call mid. The other array starts at mid+1 and then goes to r. We keep doing this recursively until we reach our base case of l=r. Once we reach our base we then call the helper method count which moves us to the conquer step.

   b. Conquer step

   Calculate the count of the window.

   c. Combine step

   At the combine step we have two counts the right w_1 *count and the left* w_2 *count. We then calculate the max count of the entire window. We then take the max of* w_1, w_2, *and the max count of* w_1+w_2.

   d. Base case(s)

   We have two base cases. The base case for Algorithm2 is when there is matrix of n by n where n < 2 we return 0. The other base case is of our helper function maxTRec. The base case is when l = r we then return the count of that window.

4. **(7 points) Fully specify your algorithm in pseudocode format using the LaTeX packages included in this file (just like in earlier assignments). Each recursive call of your algorithm must use** $O(1)$ **additional space.**

Algorithm 2 Our SLIT algorithm

```
function ALGORITHM2(m, n)
    if n<2 then return 0
    end if
    slit = 0
    for row = 0; row <= n-2; row++ do
        slit = Math.max(slit, maxTRec(m, 0, n, row)
    end for
end function
function MAXTREC(m, l, r, row)
    if l == r then return count(m, row, l, 1)
    end if
    mid = (l+r)/2
    maxL = maxTRec(m, l, mid, row)
    maxR = maxTRec(m, mid + 1, r, row)
    maxSumL = 0
    maxSumR = 0
    sum = 0
    for (i = mid; i >= l; i − −) do
        sum += count(m, row, i, 1)
        if sum > maxSumL then maxSumL = sum
        end if
    end for
    sum = 0
    for j = mid + 1; j < r; j++ do
        sum += count(m, row, j, 1)
        if sum > maxSumR then maxSumR = sum
        end if
    end for
    return return Math.max(Math.max(maxL, maxR), maxSumL + maxSumR)
end function
```

5. **(2 points) What is the FULL DEFINITION of the recurrence that describes your algorithm?**

   Answer: $T(N) = \sum_{r=0}^{n-2} 2T(\frac{nt2}{2}) + 2n$

6. **(2 points) What is the solution to the recurrence you gave as an answer to Question 5 above?**

   Answer: $\Theta(\boxed{N^2 \log N})$

7. **(8 points) Implement your algorithm, that is, the body of the method called `algorithm2` in the file `SLIT.java` as well as any helper methods that you need (these, MUST be**

added just below the `algorithm2` method). For this question, you may NOT modify any other part of the code handout. Recall that each recursive call of your algorithm must use $O(1)$ additional space.

Your answer to this problem will be fully contained in the `SLIT.java` *file that you submit*

8. **(2 points) Run the driver code provided in `A6.java` without modifying it. You MUST use the seed value of 1 in the output that you include below. You may interrupt the execution of the driver once the brute-force algorithm takes more than five minutes to complete. Include the output of this run below.**

Delete this box; then copy and paste the full output of your code below within a `verbatim` *environment.*

```
Populating the matrix...
Done
seed = 1
maxN = 8192
    32,   0.003,  22,       22,  0.001
    64,   0.015,  36,       36,  0.001
   128,   0.188,  76,       76,  0.002
   256,   3.542, 124,      124,  0.010
   512,  70.699, 196,      196,  0.042
```