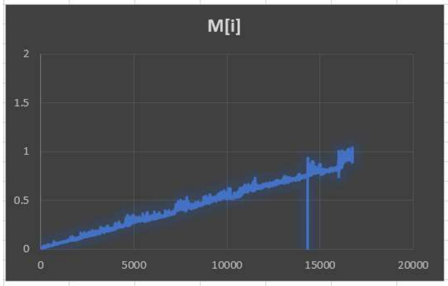
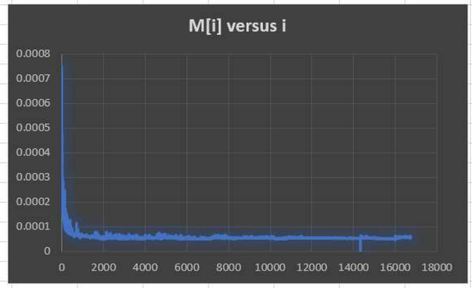
Submitted by: Wilbur Rotoni and Genji Nakano

CPSC-3283-A01

Module 6 Programming Assignment Report

The following are the plots of M[i] and "M[i] versus i".





Interpretation:

We perform a time complexity analysis on the **RECURSIVE-ACTIVIY-SELECTOR** algorithm.

RECURSIVE-ACTIVIY-SELECTOR(*s, f,* k, n)

1 m = k + 1
2 while $m \leq n$ and s[m] < f[k]
3 $\quad m = m + 1$
4 if $m \leq n$
5 $\quad$ **return** $\{a_m\}$ U RECURSIVE-ACTIVIY-SELECTOR(s, f, m, n)
6 **else return**

We already know from Dr. Biaz' Module 6 lecture that: The total cost of operation of the for-loop in lines 2 to 3 is Θ(n). The recursive call in line # 5 also costs Θ(n). The rest of the lines in the pseudocode have a constant time cost, Θ(1). Thus, the **RECURSIVE-ACTIVIY-SELECTOR** has a time cost of Θ(n + n + 1) = Θ(2n + 1) = Θ(n). (Excluding the sorting).

Next, we perform a time complexity analysis on the **GREEDY-ACTIVIY-SELECTOR** algorithm.

GREEDY-ACTIVIY-SELECTOR(*s, f*)

1 $n = s.length$
2 $A = \{a_1\}$
3 $k = 1$
4 **for** m = 2 **to** n
5 $\quad$ **if** $s[m] \geq$ f$[k]$
6 $\quad\quad A = A$ U $\{a_m\}$
7 $\quad\quad k = m$
8 **return** A

The total cost of operation of the for-loop in lines 4 to 7 is Θ(n). The rest of the lines in the pseudocode have a constant time cost, Θ(1). Thus, the **GREEDY-ACTIVIY-SELECTOR** has a time cost of Θ(n + 1) = Θ(n). (Excluding the sorting).

We can see that the time complexities of the **RECURSIVE-ACTIVIY-SELECTOR** and **GREEDY-ACTIVIY-SELECTOR** algorithms are **asymptotically the same** (i.e. Θ(n)). However, we can see that in terms of their coefficients, the **RECURSIVE-ACTIVIY-SELECTOR** has a higher coefficient (i.e. ***2n*** versus ***n***). Hence, theoretically, the $\frac{TimeRecursive}{TimeIterative}$ ratio is: $\frac{2n}{n} = 2$. This is consistent with what we observe in our **M[i]** graph above that the $\frac{TimeRecursive}{TimeIterative}$ ratio is upper- bounded by 2.

Furthermore, we can see in our **"M[i] versus i"** graph above that in the beginning (***i.e.*** the first 20 or so samples), the iterative algorithm appears to be more efficient than the recursive one (i.e. TimeRecursive > TimeIterative); however, as the number of samples significantly increase, we can see that $\frac{M[i]}{i}$ significantly decreases (*approaches a very low number, i.e. <<<<<1*). Again, this is consistent with our expectation as we have already shown in our time complexity analysis above that TimeRecursive and TimeIterative are **asymptotically the same**, ***i.e.*** Θ(n).

In addition:

- Yes, the program works.
- The program is written in Java using Auburn University's jGrasp IDE. Therefore, simply open the ***ProgrammingAssignment1.java*** file in jGrasp -> then Compile -> and then Run .

  The file F is a *.txt file but has been formatted like a *.csv file. This way, it can be conveniently opened (and plotted) in excel.

  Notes:

  1. In line # 34 of the code (***M6ProgrammingAssignment.java***), the user should change the location where the log file (F) should be saved. Currently, it is set to the programmer's local disk path/folder. **It is very strongly recommended that the user selects the same folder where he/she saved the source code *M6ProgrammingAssignment.java* file.**
  2. In line # 133, the user should make sure that **the file name in line # 34 and line # 153 should exactly match**.