Name: Mashir Kalban (ent547)

CS5463 Fundamentals of Software - Fall 2020

Assignment 1: Function Runtimes Table

Due 9/10/20 by 11:59pm

Completing the Program (15 points)

This program prints a table of runtimes (these are displayed in seconds) for given functions on arrays. The program tests different array sizes to establish a relationship between input size and runtime. It tests each array size multiple times and then takes an average of the times. Here are example calls to the timing functions:

```
int[] sizes1= { 1000, 2000, 4000, 8000, 16000};

fRT = timeAlgorithm("Insertion Sort", 10, 5, sizes1, "insertionSortInitial" );
printRuntimeTable(fRT);

fRT = timeAlgorithm("quicksort", 10, 5, sizes1, "quickSortOptInitial" );
printRuntimeTable(fRT);
```

This results in following table:

Insertion Sort

Test size	Test #0	Test #1	Test #2	Test #3	Test #4	Test #5	Test #6	Test #7	Test #8	Test #9	Average
1000	0.001	0.001	0.001	0.001	0.001	0.002	0.003	0.002	0.001	0.001	0.001
2000	0.003	0.003	0.003	0.005	0.004	0.004	0.003	0.005	0.003	0.005	0.004
4000	0.008	0.009	0.011	0.011	0.011	0.010	0.010	0.011	0.008	0.008	0.010
8000	0.028	0.026	0.026	0.030	0.037	0.028	0.028	0.025	0.027	0.029	0.028
16000	0.084	0.086	0.084	0.090	0.091	0.117	0.130	0.130	0.122	0.100	0.103

quicksort

Test size	Test #0	Test #1	Test #2	Test #3	Test #4	Test #5	Test #6	Test #7	Test #8	Test #9	Average
1000	0.006	0.001	0.004	0.002	0.001	0.001	0.002	0.003	0.002	0.002	0.002
2000	0.002	0.003	0.002	0.003	0.003	0.005	0.005	0.002	0.003	0.002	0.003
4000	0.004	0.006	0.007	0.005	0.006	0.005	0.005	0.005	0.006	0.006	0.006
8000	0.013	0.018	0.015	0.010	0.011	0.011	0.011	0.009	0.010	0.011	0.012
16000	0.019	0.019	0.022	0.019	0.018	0.019	0.019	0.019	0.018	0.020	0.019

Note your runtimes may vary since the test data is randomly generated.

The runtimes are stored in a functionRuntimes class. You are completing a program to create and fill data in this class, print the data of this class.

You are given a partial implementation in the files "MysteryRuntime.java", "functionRuntimes.java", and "ArrayAlgs.java". The portions of code that you need to write have been marked with the text "TODO".

Using the Program (5 points)

After you have the program completed, you should use it to help determine the

asymptotic runtimes of the three mystery functions (i.e., mysteryRuntime1, mysteryRuntime2, mysteryRuntime3).

Be sure to also examine the code of the mystery functions to confirm your estimations.

Fill in the following table with your runtimes:

```
/*
Give your asymptotic estimates for the runtimes of the following 3 functions:

mysteryRuntime1:  O( 1 )
mysteryRuntime2:  O( n )
mysteryRuntime3:  O( n^2 )
*/
```

1. Longest Sorted Subarray (4 points)

Consider the following problem:

Input: An array $A[1 \dots n]$ of integers
Output: The largest integer m such that the array $A[1 \dots n]$ has subarray of length m which is in sorted order (i.e, increasing order).

The following pseudocode finds the length of the longest of the given array $A[1 \dots n]$ by considering all possible subarrays:

Algorithm 1 longestSubArray(int $A[1 \dots n]$)

```
1:  k = n;
2:  while ( true ) do
3:      //(I) The longest increasing subarray of A has length ≤ k
4:      low = 1
5:      high = k
6:      while ( high ≤ n ) do        n (n-1) (n-2) .... 1 = n! (runtime)
7:          if ( isIncreasing(A[low ... high]) ) then
8:              return k;
9:          end if
10:         low++
11:         high++
12:     end while
13:     k--;
14: end while
```

The following code checks if an array is increasing (i.e., each number is smaller than the next in the array).

Example: longestSubArray($[2, 4, 3, 8, 5, 6, 7, 9, 0, 1]$) returns 4

Algorithm 2 isIncreasing(int $C[a \ldots b]$)

```
1: i = a;
2: while i < b do
3:     if ( C[i] ≥ C[i + 1] ) then
4:         return false;
5:     end if
6:     i++;
7: end while
8: return true;
```

n (runtime)

Justification: $[\cancel{2,4,3,8},5,6,7,9,\cancel{0,1}] = [5,6,7,9]$ which is a longest increasing subarray of the original array.

(1) (2 points) Consider running longestSubArray on the array:

$$[119, 100, 112, 114, 125, 113, 110, 129, 130, 140, 142, 115, 120]$$

What does longestSubArray return and what is the longest sorted subarray of A?

(2) (1 point) Give the best-case runtime of longestSubArray in asymptotic (i.e., O) notation as well as a description of an array which would cause this behavior.

(3) (1 point) Give the worst-case runtime of longestSubArray in asymptotic (i.e., O) notation as well as a description of an array which would cause this behavior.

(4) (0 points) Is this an efficient algorithm for finding the longest sorted subarray? Can you find a better algorithm for computing this?

2. Asymptotic Notation (4 points)

Show the following using the definitions of O, Ω, and Θ.

(1) (2 points) $2n^3 + n^2 + 4 \in \Theta(n^3)$

(2) (2 points) $3n^4 - 9n^2 + 4n \in \Theta(n^4)$
(**Hint:** careful with the negative number)

Deliverables:

Your solution should be submitted as "MysteryRuntime.java" as well as a .pdf file with your answers to written part of the assignment. Be sure to fill in the table of runtimes described above:

Upload this file to Blackboard under Assignment 1. **Do not zip your file**.

To receive full credit, your code must compile and execute. You should use valgrind to ensure that you do not have any memory leaks.

Remember:

The program you submit should be the work of only you. Cheating will be reported to SCCS. Both the copier and copiee will be held responsible.

Longest Sorted Subarray

1. LongestSubArray Returns 5

   [110, 129, 130, 140, 142]

2. Best case runtime O(n)

   [100, 110, 112, 113, 114, 115, 119, 120, 125, 129, 130, 140, 142]

3. Worst case runtime O(n! · n)

   [142, 140, 130, 129, 125, 120, 119, 115, 114, 113, 112, 110, 100]

4. Better algorithm

```
LongestSubArray (arr, n)
Declare max = 1, len = 1
  for i = 1 to n-1
  if arr[i] > arr[i-1]
     len++
  else
     If max < len
        max = len
     len = 1
```

if max < len

max = len

return max

Asymptotic Notation

1) $2n^3 + n^2 + 4 \quad \Theta \; n^3$

$2n^3 + n^2 + 4$

$\leq 2n^3 + n^2 \cdot n$

$= 3n^3$

$c = 3, \quad n_0 = 1$

thus $2n^3 + n^2 + 4 \in \Theta(n^3)$

2) $3n^4 - 9n^2 + 4n \quad \Theta \; n^4$

$3n^4 - 9n^2 + 4n$

$\geq 3n^4 - n^2 \cdot n^2$ (true for $n \geq 9$)

$= 2n^4$

$c = 2, \quad n_0 = 9$

thus $3n^4 - 9n^2 + 4n \quad \Theta \; n^4$