/*
Jason Wilson
Thursday, February 13th 2020
CMPT435L 111 20S
*/

Assignment 2

Please read turn-in checklist at the end of this document before you start doing exercises.

Section 1: Pen-and-paper Exercises

1. Analyze the following code and provide a "Big-O" estimate of its running time in terms of n. Explain your analysis.

```
int i = 1;
while (i <= n)
      some O(1) time statements;
      i = i*2;
end while
```

Note: Credit will not be given only for answers - show all your work:
(3 points) steps you took to get your answer.

```
int i = 1;                                  // 1
while (i <= n)
      some O(1) time statements;            // 1 * n
      i = i*2;                              // 2 * n
end while
```

Iteration 1:
i = 1 $2^0 = 1$

Iteration 2:
i = 2 $2^1 = 2$

Iteration 3:
i = 4 $2^2 = 3$

Iteration 4:
i = 8 $2^3 = 4$

Iteration 5:
i = 16 $2^4 = 5$

Iteration k:

$i = 2^{k-1} = n$

$\log(2^{k-1}) = \log(n)$

$k - 1 = \log(n)$

$k = \log(n) + 1$

$k = \log(n)$

(2 points) your answer.

O(logn)

2. Analyze the following code and provide a "Big-O" estimate of its running time in terms of n. Explain your analysis.

```
for( int i = n;  i > 0;  i /= 2 ) {
  for( int j = 1;  j < n;  j *= 2 ) {
    for( int k = 0;  k < n;  k += 2 ) {
         ... // constant number of operations
    }
  }
}
```

Note: Credit will not be given only for answers - show all your work:
(5 points) steps you took to get your answer.

Iteration 1:
i = n $n/2^0$
j = 1 2^0
k = 0   n/2

Iteration 2:
i = n/2 $n/2^1$
j = 2 2^1
k = 2   n/2

Iteration 3:
i = n/4 $n/2^2$
j = 4 2^2
k = 4   n/2

Iteration 4:
i = n/8 $n/2^3$
j = 8 2^3
k = 6   n/2

Iteration k:
i = n/8 $n/2^{k-1}$
j = 1 2^{k-1}

I3: $i = \frac{n}{4}$ $\frac{n}{2^2}$
$j = 4$ 2×2 2^2
$K = 4$

I4: $i = \frac{n}{8}$ $\frac{n}{2^3}$
$j = 8$ 2^3
$K = 6$

IK: $i = \frac{n}{2^{k-1}}$
$j = 2^{k-1}$
$K = \frac{n}{2}$

$n = 2^{k-1}$
$\log(n) = \log_2 2$
$\log(n)+1 = K$

$2^{k-1} = n$
$(k-1)(\log_2 2) = \log n$
$k-1 = \log n$
$k = \log n + 1$

$(\log n + 1)(\log n)(n/2)$
$\frac{(\log n + 1) * (\log n) * n}{2}$
$\boxed{O(n \log n)}$

k = 0 n/2

(2 points) your answer.

O(nlog n)

3. Analyze the following code and provide a "Big-O" estimate of its running time in terms of n. Assume that n = 2^m. Explain your analysis.

```
for( int i = n;  i > 0;  i-- ) {
  for( int j = 1;  j < n;  j *= 2 ) {
    for( int k = 0;  k < j;  k++ ) {
         ... // constant number C of operations
    }
  }
}
```

Note:

$$2^0 + 2^1 + \cdots + 2^k = 2^{k+1} - 1$$

Note: Credit will not be given only for answers - show all your work:

(5 points) steps you took to get your answer.

#3

Iteration 1

I 1:

$i = n$

$j = 1 \quad 2^0$

$k = 0 \quad k-1$

Iteration 2

I 2)

$i = n-1$

$j = 2 \quad 2^1$

$k = 1 \quad k-1$

Iteration 3

I 3:

$i = n-2$

$j = 4 \quad 2^2$

$k = 2 \quad k-1$

Iteration k:

I k:

$i = \boxed{n-(k-1)} = n \Rightarrow k-1 = 2n \Rightarrow k =$

$j = \boxed{2^{k-1}} = \log n \Rightarrow k-1 = \log n \Rightarrow k = \log n$

$k = \boxed{k-1} = n \Rightarrow k = n+1 \Rightarrow \phi(n)$

n

$n-(k-1) * 2^{k-1} * k-1 =$

$n-(k-1)^2 * 2^{k-1}$

$k = 2 \quad \cancel{k-1}$

Iteration k:

I k:

$i = \boxed{n-(k-1)} = n \Rightarrow k-1 = 2n \Rightarrow k = 2n+1 = \boxed{O(n)}$

$j = \boxed{2^{k-1}} = \log n \Rightarrow k-1 = \log n \Rightarrow k = \log n + 1 = \boxed{O(\log n)}$

$k = \boxed{k-1} = n \Rightarrow k = n+1 \Rightarrow \boxed{O(n)}$

$n * n * \log n$

$n^2 * \log n$

$\boxed{O(n^2)}$

$n-(k-1) * 2^{k-1} * k-1 =$

$n-(k-1)^2 * 2^{k-1}$

(2 points) your answer.

O(n^2)

4. Analyze the following code and provide a "Big-O" estimate of its running time in terms of n. Explain your analysis.

```
j = 1, i = 0;
while (i < n)
{
   i = i + j;
   j++;
}
```

Note: The loop variable ‘i’ is incremented by 1, 2, 3, 4, … until i becomes greater than or equal to n.
Note: Credit will not be given only for answers - show all your work:
(5 points) steps you took to get your answer.

#4

```
j = 1, i = 0;      // 2

while (i<n)
{
   i = i + j;      // 2 * √2n
   j++;            // 1 * √2n
}
```

Iteration 1:
I1:
$i = 0+1$
$j = 2$

Iteration 2:
I2:
$i = 0+1+2$
$j = 3$

Iteration 3:
I3:
$i = 0+1+2+3$
$j = 4$

Iteration K:
IK:
$i = 0+1+2+\ldots+K$
$j = K+1$

Work:

$\frac{K*(K+1)}{2} = n \Rightarrow K*(K+1) = 2n$

$\Rightarrow K^2 \cancel{+K} = 2n$

$\Rightarrow K = \sqrt{2n}$

$2 + 2*\sqrt{2n} + \sqrt{2n}$

$\boxed{O(\sqrt{n})}$

(2 points) your answer.

$O(\sqrt{n})$

5. Arrange the following functions in ascending order of growth rate (8 points):

n^4	$\sum_{i=1}^{n} 1$	$\log\log n$	2010	$\sum_{i=1}^{n} i$
2^n	$\sqrt{n}$	$\log n$	n^2	$n \log n$
n^n	$\sum_{i=1}^{n} \frac{1}{i}$	$n!$	e^n	n

You are NOT required to justify your ordering.
Note:
In this problem, you are asked to identify if f1(n) < f2(n) for a "sufficiently large" input size n. However, for small values of n this is not always true.

$$2010 \le loglogn \le logn \le \sum_{i=1}^{n} \frac{1}{i} \le \sqrt{n} \le n \le \sum_{i=1}^{n} 1 \le nlogn \le n^2 \le \sum_{i=1}^{n} i \le n^4 \le e^n \le n! \le n^n$$

6. Given a positive integer x, find square root of it. If x is not a perfect square, then return floor (round down).

Examples:

Input: x = 4
Output: 2
Input: x = 11
Output: 3

Outline an algorithm for finding square root of x. Expected in O(log n) time.

Full credit (10 points) will be awarded for an algorithm that is O(log n). Algorithms that are O(n) or slower will be scored out of 5 points.

Note: You should NOT use existing functions like math.sqrt() to obtain the square root of x. Create your own function. Solutions that use existing functions will receive 0 points.

(i) **describe the idea behind your algorithm in English (2 points);**

1. Find the middle of the search area and square it
2. Check to see if the number is a prefect square
3. If the mid number squared is greater than x, set the end of the search area to the midpoint munis one
4. Else if the mid number squared is smaller than x, set the beginning of the search area to the mid point + 1

(ii) provide pseudocode (5 points);

Input x

Int start = 1 , int end = x, int mid = 0, int midSquare, int result = 0

while [start <= end]

    mid = start + (end - start) / 2
    midSquare = mid * mid

    if (midSquare == x)

        return mid

    end if

    else if (midSquare > x)

        end = mid - 1

    end if

    else

    start = mid + 1
    result = mid

    end else

end while

Output result

(iii) analyze its running time (3 points).

```
public static int squareroot(int x)
    {

  int start = 1;      //1
  int end = x;        //1
  int mid = 0;        //1
  int midSquare;   //1
  int result = 0;     //1

  while (start <= end) {

   mid = start + (end - start) / 2;  // 4
   midSquare = mid * mid;           // 2

   if (midSquare == x) {       // 1
```

```
        return mid;             // 1

      } else if (midSquare > x) { // 1

        end = mid - 1; // 2

      } else {

        start = mid + 1; //2
        result = mid; //1

      }
    }

    return result; //1
```

Ending condition = sz

Iteration 1:

Sz = n $\qquad \frac{n}{2^0}$

Iteration 2:

Sz = $\frac{n}{2}$ $\qquad \frac{n}{2^1}$

Iteration 3:

Sz = $\frac{n}{4}$ $\qquad \frac{n}{2^2}$

Iteration k:

Sz = $\frac{n}{2^{k-1}}$ = 1

Steps:

$$n = 2^{k-1}$$
$$logn = k - 1$$
$$k = logn + 1$$

Big O time complexity:

$$O(logn)$$

Section 2: Java Implementation

7. Implement problem 6 in Java (30 points).
   Note:
   Find a file called Problem6.java in assignment 2 folder.

Complete the method of squareroot().
Test your method in the main method provided.
Programs that are O(n) or slower will be scored out of 10 points.
Programs that use existing functions like math.sqrt() will receive 0 points.
Important: In all of the assignments of this course, when you are asked to implement an algorithm for a problem, your code will be evaluated based on:

5 points - Execution
Each file must run without error or warning on valid input described in the main method provided.

5 points - Within Code Documentation
Is the code documented for obvious understanding of the use, preconditions, and postconditions of each function?

20 points - Correctness
Is the algorithm implemented correctly? Does your method pass the test?

TURN-IN CHECKLIST:

1. **Answers to Section 1 (.doc/.txt), and to Section 2 (all your source Code (.java files)). Remember to include your name, the date, and the course number in comments near the beginning of your code/report.**

2. **Create a folder and name it 'FirstName_LastName_assignment_2'. In the newly created folder copy and paste your files (.doc/.txt/.java files). Then compress the folder, and push it to iLearn.**