Central Washington University
College of the Sciences
Department of Computer Science

CS-301 Data Structures Fall 2016

Lab Practice 02

This practice is about algorithms and its characteristics, and practice methods for estimating running times. We are going to learn the use of a linux utility, to compute some run times.
Normally you, will find the source and data files in `/home/cs-301/Labs/Lab02`

1. For this we are going to run in a linux environment, to be able to use the `time` command (do `man time` for details). Consider the following function.

```
public long  pythagoric (long  n ) {
 long ret;
 for  (int x  = 0   ; x < n ;  ++x )
 for  (int y  = 0   ; y < n ;  ++y )
 for  (int z  = 0   ; z < n ;  ++z )
     ret =x*y*z + (y-1)/3 ;
 return ret;
}
```

   a. How many multiplications $M(n)$ and divisions $D(n)$. If we use as units the totals of those of operations, what is is $worstTime(n)$. In big-O notation?

   b. Implement the function and run it for $n = 250, 500, 1000, 2000$. Use the `time` command. Are the values increasing as expected ?

   c. The `System` class in package `java.util` provides a method measuring the number of nanoseconds ($10^{-9}s$) since the start of the JVM. Use the method in your program, to compute the running times as in (b). How do the numbers compare with `time`.

2. Can the algorithm be rewriten with a slight improvement. **Hint:** try to reduce $D(n)$

3. This will help understand what is the difference beteen the running time of an algorithm and the *time complexity* of a problem.

   a. Consider the problem: Given an array $C_1,\ C_2,\ \ldots C_{n-1},\ C_n$ of numbers compute

$$tel(n) = \sum_{k=1}^{n-1} c_{k+1} - c_k$$

   Write a Java function to compute it using a loop.

```
public   double  tel ( double [] c )
```

What is $worstTime(n)$ in terms of addition/subtraction operations?. In terms if big-O notation?

b. Directly implementing the sum above resulted in too much computation. Write out all the terms of the sum, and see if you can write a *much better* algorithm for the same function. What is the complexity of *the problem*?. Use big-O notation.

4. What is $worstTime(n, m)$ in the following program. What is the big-O of it (in terms of m and n)? Use comparisons as units.

```
public void  simple (int   n , int m) {
int r= m * n;
int k = r/2;
while ( k < r) {
     if ( k % 7 == m % 5 )
       break;
     else {
        s += k;
        k += 2
     }
}
```

   Cand you see a simple way to improve the program?

5. What is approximately $worstTime(n)$ for the following algorithm in terms of divisions operations?

```
j = n;
while  ( j > 1 )  {
   z = 0;
   while ( z  <= n-5) {
       a[z] = a[z] / a[z+1]
        z++;
    }
    j /= 3;
 }
```

6. The following solves a very practical problem, which you may have already encounter in CS-111.

```
public static void whatisit (int[ ] x) {
       for (int i = 1; i < x.length; i++)
           for (int k = i; k > 0 && x [k -1] > x [k]; k--)
            swap (x, k, k -1);
         }
```

Using as units the calls to `swap`, what is $worstTime(n)$?
Can you provide conditions under which the running time is 0?
Can you provide conditions under which the running time is $n(n-1)/2$

7. Consider a method firstandsecond (int a[]) that computes the two largest numbers in the array. Give a simple argument why $worstTime(n)$ is $O(n)$. There is not need to write a program.