Lebanese International University

Department of Computer Science

CSCI 440 – Algorithms
Assignment 1

Solution by Bassam Kaddoura (31730588)

Question 1:

a. Write a recursive method that will compute the sum of even digits of an integer. If n=12314, the method returns 6 (4+2)
b. Write the recurrence relation and determine the big-Oh complexity of the method.
c. Will the iterative solution of the problem give a better complexity? Why?

```
public static int evenDigits(int n)
  {
    if (n != 0)        //stop if n hits zero
    {
       int d = n % 10;

       if (d%2 == 0)  return (d + evenDigits(n/10));

       else return evenDigits(n/10);

    } else return 0;
  }
```

b) If n = 1, the method runs once → $C_1 = 1$

If we multiply it by 10 (n=10), it runs twice $C_{10} = 2$ and so on

Big-Oh of **$\log_{10} n$**

c)

```
public static int evenDigits(int n)
   {
      int s = 0;
      for(int i = 1; i < n; n/= 10)
      {
         int d = n%10;
         if(d%2 == 0) s += d;
      }
      return s;
   }
```

The program stops when n hits 0, and n is being divided by 10 for every iteration, so it will loop through the problem at $\log_{10} n$

Question 2:

Given two algorithms A1 and A2, which have a running time of $0.001n^3$ and $0.5n^2\sqrt{n}$, which of two algorithms should be chosen for a problem size greater than $2x10^5$? Explain in details your analysis.

For $n = 2*10^5$:

$0.001n^3 = 8*10^{12}$
$0.5n^2\sqrt{n} = \sim 8.944*10^{12}$
➔ $0.5n^2\sqrt{n} > 0.001n^3$ (for $n = 2*10^5$)
Thus we use algorithm A2.
But as n grows larger, we will notice that A1 will become more efficient.

$0.001n^3 > 0.5n^2\sqrt{n}$
$0.001n > 0.5\sqrt{n}$
$0.001 > (0.5\sqrt{n})\ /n$
$0.002 > \sqrt{n}\ /n$
$0.002^2 > n/n^2$
$1/n < 0.002^2$
$n > 1/0.002^2$
$n > 2.5*10^5$

If $n > 2.5*10^5$ we start using A1, else we use A2.

Question 3:

Given the following recursive method:

```
public static String HR(String s)
{
        int N = s.length();
        if (N <= 1)
                return s;
        String left = s.substring(0, N/2);
        String right = s.substring(N/2, N);
        return HR(right) + HR(left);
}
```

a. What does this method do?
b. What is the time complexity of this method?

a) Prints the input string backwards.

The program is going through every letter of the input string, and for each recurrence, the initial string is halved in length, in addition to the final iteration when the length becomes <= 1.
So $T(N) = T(N/2) + T(N/2) + 1$
$T(N) = 2T(N/2) + 1$ but $T(N/2) = 2T(N/4) + 2$
$T(N) = 4T(N/4) + 3$ and so on
$T(N) = kT(N/k) + (k-1)$ where $k = 2^x$
$x = \log_2 N$
➔ $NT(1) + N - 1$ ➔ $2N - 1$

Complexity = O(N)

Question 4:

We need to Implement a method rotate(ar[], d, n) that rotates an array (ar[]) of size n by d elements. For example if the array is: 1,2,3,4,5,6,7

A rotation of 2 elements makes the array as follows: 3,4,5,6,7,1,2

a. Describe an algorithm as a pseudo-code (the java code is not required) for the method rotate with a time complexity of O(N*d)
b. Describe an algorithm as a pseudo-code (the java code is not required) for the method rotate with a time complexity of O(N)

```
public static void rotate(int[] arr, int d, int n )
  {
     for(int i = 0; i < d; i++)                    //d loop
     {

        int temp = arr[0];
        for(int j = 0; j < n-1; j++)               //n loop
        {
           arr[j] = arr[j+1];
        }
        arr[n-1] = temp;
     }
  }
```

Complexity = d*n O(n*d)

```
public static void rotate(int[] arr, int d, int n )
   {
     int [] temp = new int[d];
     for(int i = 0; i < d; i++)                    //d
     {
        temp[i] = arr[i];
     }

     for(int j = 0; j < n-2; j++)                  //n
     {
        arr[j] = arr[j+2];
     }

     for(int k = 0; k < d; k++)                    //d
     {
        arr[n-k-1] = temp[1-k];
```

```
    }

  }
```

2d + n but the worst case is when we rotate by n times → 2n + n = 3n
→ O(n)

Question 5:

Solve the following recurrence relations - – Show the necessary details:

a. $x(n) = x(n-1) + 5$ for $n > 1$, $x(1) = 0$
b. $x(n) = 3x(n-1)$ for $n > 1$, $x(1) = 4$
c. $x(n) = x(n-1) + n$ for $n > 0$, $x(0) = 0$

a)

$x(n) = x(n-1) + 5$

$x(n-1) = x(n-2) + 10$

$x(n-k) + 5k$

for $k = n-1$

$x(n-n+1)+5(n-1)$

$x(1) + 5(n-1)$ $x(1) = 0$
$5(n-1) = 5n - 5$ →→ $O(n)$

b)

$x(n) = 3x(n-1)$

$3^2x(n-2)$

$3^3x(n-3)$

… $3^kx(n-k)$
for $k= n-1$

$3^{n-1}x(1)$

$3^{n-1}.4$

⇨ $O(3^n)$

Question 6:

What is the time complexity of each of the following Java code?

```
for (int a = 0; a < m; a=a+2){                  m/2
     for (int t= 1;t <= m; t=t*2){              log2m
             System.out.println(a + t);         total: m/2(log2m)
               }
}
for (int i = 1; i <= n; i++){                   n
     for (int j = 1; j <= n; j++){              n
          for (int k = 1; k <= 3; k++){         3
                   System.out.println(i+j+k);total: n*n*3
            }
     }
}
```

For the first loop:
$m/2(\log_2 m) \rightarrow O(m(\log_2 m))$
For the second loop:
$3n^2 \rightarrow O(n^2)$

Question 7:

Using the formal definition of Big-Oh notation, prove that $f(x) = x^3 + 100x + 34$ is $O(x^3)$

For all $x >= 1$

$x^3 + 100x + 34 <= kx^3$
$(x^3 + 100x^3 + 34x^3)/x^3 <= k$ (up the degree)
$1 + 100(1) + 34 <= k$ (sub by lowest positive x)
➔ $k = 135$

$135x^3$ → $f(x)$ is $O(x^3)$ with $k = 135$ and $x >= 1$

Assignment Guidelines:

- This assignment is handed on Wednesday, 7 November and is due **by e-mail** on Sunday, 11November (before midnight).
- If you submit one day late you will get a zero.
- A "PDF" document containing the questions and their corresponding "CLEAR" answers should be sent to: mohammad.chamseddine@liu.edu.lb
- Do not send files with the .docx, doc or .txt extension and do not send the answers inside the body of the message.
- **Your name should be clearly stated inside the document**.
- Refer to the syllabus for the policies on cheating and grade distribution.