Fundamentals of Algorithms
CS502-Fall2011

ASSIGNMENT1

Deadline

Your assignment must be uploaded/submitted at or before **1st November 2011.**

Uploading instructions

Please view the **assignment submission process** document provided to you by the Virtual University to upload the assignment.

Rules for Marking

It should be clear that your assignment will not get any credit if:

- The assignment is submitted after due date.
- The submitted assignment does not compile or run.
- **The assignment is copied.**

Objectives

This assignment will help you to understand the concepts of Asymptotic Growth, making analysis of pseudo code, recurrence relation development, asymptotic function understanding and iterative solutions for recurrences.

Guidelines

RULES FOR CALCULATING TIME COMPLEXITY AND BIG-OH

Rule 00
Normally these formulas are very handy:

If $x^y = z$ then $y = \log_x z$
Also,

$$\sum_{i=1}^{n} a_i = \frac{n}{2}(a_1 + a_n) \qquad \sum_{i=1}^{n} i = \frac{n}{2}(n+1) \qquad \sum_{k=0}^{m} r^k = \frac{1-r^{m+1}}{1-r}$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \text{ (for n >=1)}$$

Rule 0

The condition that stops a loop executes ONE MORE time than the loop itself (the last time is when it is evaluated false)

Rule 1

for (i=0;i<n;i=i+k) Anything inside the loop will run approximately **n/k** times

Rule 2

for (i=n;i>0;i=i-k) Anything inside the loop will run approximately **n/k** times

Rule 3

for (i=1;i<n;i=i*k) Anything inside the loop will run approximately **$\log_k$n** times

Rule 4

```
for(i=1;i<=n;++i)
        for (j=1;j<=i;++j)
```

The above nested loop approximately runs **½ n(n+1)** times.
The variable j depends upon the value of i

Rule 5

```
for(i=1;i<=n;i=i*2)
        for (j=1;j<=i;++j)
```

The statements in the above nested loop approximately run **2n-1** times.
The variable j depends upon the value of i

Rule 6

If the loop variables are independent then the total times a statement inside a nested loop is executed is equal to the product of the times the individual loops run

e.g.

```
for (i=0;i<n;++i)
        for (j=0;j<m;++j)
```

A statement inside the above nested loop will run **n*m** times

Other Guidelines

While loop related information

Complexity of “while” loop depend upon the initial entrance condition if it remains true for “n” iterations it will be “n+1”; Note here “1” will be added for the last time check of the condition .Here this will be clear to you if the some logical conditions are checked other then counters then all complexity will be based on scenario of the problem and nature of the logical condition.

Function Growth rate concept

If some function $f_1(x)>f_2(x)$ for positive values of x then the function $f_1(x)$ is said to have greater growth rate then $f_2(x)$. For example $f_1(x)=x^5$ and $f_2(x)= x^6$ it is obvious that $f_2(x)$ has greater growth rate ($2^6 > 2^5$).This concept relate to complexity of algorithm ,an algorithm having greater growth rate function means the algorithm has greater complexity here $f_2(x)$ is more complex then $f_1(x)$.

Estimated Time 2.5 hour

For all parts of the question to understand maximum time is 1.25 hours and for solution maximum time is 1.25 hour. It all depends upon your sheer concentration.

Question **(4*5)**

Write piece of pseudo codes having the following time complexities:

a) $T_1(n) = n + \sum_{i=0}^{n} i$

Hints to develop the Code:

First you have require very simple loop starting from 1 to n as there is "+" operator in the function this loop will be independent .

Second expand the series this will help you to calculate the complexity.

$\sum_{i=0}^{n} i = 0+1+2+3+4+5+\ldots\ldots\ldots\ldots..+n = n(n+1)/2 = (n^2+n)/2 = O(n^2)$ here the summation formula is very well known. Basics idea is to develop the two nested loop in which inner loop runs as many times as the current counter of the outer loop. In short it is rule 4 above you just need to expand the series.

Note: Where symbols "//" have been used mean some elaboration comments for the code.

Now come to pseudo code:

```
for (i=0; i<n-1;i=i+1)
    {y=y+1;}
    // Here note "n-1" end condition  is to cater the rule 0 as the we need "n" not
    // "n+1" in complexity.

    for (i=0;i<=n;i=i+1)
     {   for (j=i;j<=i;++j)
        {
          s=s+1;
       }
 }
```

// this is completely rule 4 which gives the required complexity means n(n+1)/2 .

b) $T_2(n) = 6n(\log_{12} n) + 1 + (m/7)\log(m)$

Hints:

Nesting of loops will require for first part and rule 3 will help you to code this part as we need the outer loop runs "6n" times and the inner loop will run "$\log_{12} n$" at each iteration of outer loop and next "+" operator for "1" you need simple statement say "assignment of some variable" and next "+" operator very similar to its first part as here again you need the rule 3 for inner loop as here is "log" and for outer loop you can get help from rule 1 or rule 2 as you want to get division factor in iterations.

Now come to pseudo Code:

```
    for (i=0;i<=6n;i=i+1)
     {   for (j=1;j<=n;j=j*12)
         {
           s=s+1;
         }
}
// Here you may alter the inner and outer loop complexity will remain same
y=10;
// simple assignment statement will take time of O (1)

for (i=m;i>0;i=i-7)
    {    for (j=1;j<=m;j=j*2)
         {
           s=s+1;
         }
}

//Here again you may alter the sequence of loops and more outer loop can also
//be according to rule 1
```

c) O (z+m)

Hints:

This is very simple part as in complexity we are using simple addition we can take two different independent non-nested loops using the limit variable as in complexity notation.

Now come to pseudo Code:

```
    i=1;
    j=1;
    While(i<=z)
```

```
{
i++;

}
While(j<=m)
{
j++;

}
```

//Note here you can use the "for" loops as well this is not big issue ;what you required is to iterate as that many times as complexity function requires.

d) $T_3(n) = 5n\sum_{i=1}^{n}(1/i)$

As we have:

$$H_n = \sum_{i=1}^{n} \frac{1}{i}$$
$$= 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} \approx \ln n$$
$$= \Theta(\ln n)$$

Hints:

This is again very interesting case you need to just expand the series to get the complexity concept it is well known geometric series and approximate to natural "log" this is also mentioned in your lecture handouts and further here you need the nesting of loops as "ln(n)" will be multiplied by "5n" run outer loop "5n" times keeping the inner loop ln(n) time .
Note : Index step size is integer so we can take it as base

Note: Where the base of "log "has not been mentioned take it as base "2" if we consider the "e" as floor function and base "3" will also work and if some one take exact that will also be considered true.

Now come to pseudo Code:

```
e=2.718;
```

```
for (i=0;i<=5n;i=i+1)
 {   for (j=1;j<=n;j=j*e)
     {
       s=s+1;
    }
}
```
//Here again loops sequence can be altered and more in place of “e” one can use “2” or “3” it will be considered correct.