Fundamentals of Algorithms
CS502-Spring 2011

SOLUTION ASSIGNMENT1

Deadline

Your assignment must be uploaded/submitted at or before **18th April 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 $f_1(x)$ 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 Q1 maximum time is 1.25 hour and for Q2 maximum time is 1.25 hour. It all depends upon your sheer concentration.

Question 1 (10)

Find the running time complexity of the following piece of code and show your working step by step.

```
1.  y=0;
2.  x=0;
3.  for(i=m; i>-2; i=i-2)
4.  { x++; }

5.  for (i=n; i>0;i=i-1)
6.  { y=y+1;}

7.  for (i=1;i<=n;i=i*5)
8.  {        for (j=1;j<=5n;++j)
9.  {
        a.  for(k=0;k<n; k=k+4)
                 i.  {
                ii.  x=x+5;
               iii.  }
10. }
11. }

12. While(i<=z)
13. {
14. i++;

15. }
```

Solution Q1

Code Line		**Time taken**	**Comments**
1	y=0	1	Constant time
2	x=0	1	
3	for (i=m; i>-2;i=i-2)	(m/2+1)+1	Outer independent loop1 ; apply rule 2 mentioned above in guidelines as here step size is “2” so value of k=2 and “m” will control the number of iterations and rule gives you m/2 and base limit is “-2” that’s why we added “1” more here. Note here”1” time more then “m/2+1”is for condition checking at end to exit from loop.
4	x=x+1	m/2+1	As entrance in loop is “m/2+1” time
5	for (i=n; i>0;i=i-1)	(n/1)+1=n+1	Outer independent loop2 ; again apply rule 2 here step size is “1” so value of k=1. Other logic is same as in above statement “3” only difference is of “m” and “n” .
6	{ y=y+1;}	n	As entrance in loop is

			“n” times
7	for(i=1;i<=n;i=i*5)	$\mathrm{Log}_5 n+1$	As step size is multiplied by five that’s why log to the base 5 (rule 3)
8	for(j=1;j<=5n;++j)	$5n(\log_5 n)+1$	As for each iteration of outer loop this loop will execute for 5n times
9 a	for(k=0;k<n;k=k+4)	$(n/4)5n(\log_5 n)+1=(5n^2 \log_5 n)/4+1$	As this is the inner most loop and for each above two loops it will execute n/4 times (rule 1)
9 a i	x=x+5;	$(n/4)5n(\log_5 n)=(5n^2 \log_5 n)/4$	This statement is under the second inner loop.
12	While(i<=z)	z+1	As this is simple independent loop with upper limit “z” and step size is increasing “1” each time.
14	i++;	z	As entrance in loop is “z” times

From above explanation over all time will be summed up to take final one.

$T(n)=1+1+(m/2+1)+1+m/2+1+n+1+n+\log_5 n+1+5n(\log_5 n)+1+(5n^2 \log_5 n)/4+1+(5n^2 \log_5 n)/4+z+1+z$

$=10+2m/2+2n+\log_5 n+5n(\log_5 n)+(10n^2\log_5 n)/4+2z \in \Theta(n^2\log_5 n+(m+z))$ as the leading term is $(10n^2 \log_5 n)/4$ and “m” and “z” also effect for large values so these terms will also consider; thus we can asymptotically bound the code run time by this. $\Theta(n^2\log_5 n+(m+z))$

<u>Question 2</u> (10)

Arrange the following in the **Most to Least** complexity order. Here “n “is the input size for the some complexity function and k< j and j & k are numbers greater than 2.Every function is separated by “comma” and note these are 20 functions to arrange.

$$n,\ n^{k/2},\ n^{j/2},\ n\lg n,\ n^{n},\ 1, 100,\ 2^{n},\ \lg n,\ n!, (n!n\sqrt{n})/\sqrt{n^2},$$

$$n/\sqrt{n}\ , n!n\log n/\sqrt{n^2}, n/\log n, 10000, n\ \ /\sqrt[4]{n}, n(\log n)\sqrt[8]{n},$$

$$n\ \ /\sqrt[5]{n}, n(\log n)\sqrt[6]{n}, 1000$$

<u>Hints to fast your logics to arrange:</u>

- Think other way around i.e. the function which is less complex and more efficient pick it first and go ahead and then and at end reverse the order to get final arrangement ;e.g. less complex function is 1, 100, etc(Note these are constant functions that’s why these are only judged on their constant value and they fall in same class because of having constant values) Other less complex and more efficient functions are simple “log” as these functions reduce the answer for large values of “n” ; then nth root means square root cube root etc ; then linear functions are efficient and so on the logic is build .
- If you feel difficulty for comparison then judge for lager inputs say 10^{100} etc.
- Note 2^{n} and n! are both complex but you can judge that n! is more complex by putting values of larger “n” and one more thing is n^{n} will be more complex then n! as by definition n! =n (n-1)(n-2)………..2.3.1.

But when you put the lager value in each place in above you get

n(n)(n)……………n= n^{n} which will obviously beat n!

Given Order	**Most to Least Complexity Arrangements**	**Hints**
n	n^n	
$n^{k/2}$	$(n!n\sqrt{n})/\sqrt{n^2} = n!\sqrt{n}$	**At reducing we get the value note here $\sqrt{n}$ is larger factor then "logn" so it will be more complex then very next in below cell**
$n^{j/2}$	$n!n\log n/\sqrt{n^2} = n!\log n$	
nlgn	n!	
n^n	2^n	
1	$n^{j/2}$	**As j>k and note here what ever the value of "j" or "k" will be taken it will become constant then the variable "n" for very large values of "n" you can compare the complexities of these with other functions.**
100	$n^{k/2}$	
2^n	$n(\log n)\sqrt[6]{n}$	
lgn	$n(\log n)\sqrt[8]{n}$	
n!	nlgn	
$(n!n\sqrt{n})/\sqrt{n^2}$	n	

$n/\sqrt{n}$	$n/\log n$	**Here is interesting point arise as when you divide by smaller values the answer becomes more so this is more complex then below one functions .Convince yourself by putting 10^8 in this function as well as in the following cells. Note one thing here "log "has base "2".**
$n!n\log n/\sqrt{n^2}$	$n\ /\sqrt[5]{n}$	
$n/\log n$	$n\ /\sqrt[4]{n}$	
10000	$n/\sqrt{n}$	
$n\ /\sqrt[4]{n}$	lgn	
$n(\log n)\sqrt[8]{n}$	10000	
$n\ /\sqrt[5]{n}$	1000	
$n(\log n)\sqrt[6]{n}$	100	
1000	1	

BEST OF LUCK

THINK BEYOND THE BOUNDRIES