Analysis of Algorithms, Spring 2007
Midterm ANSWERS

1	2	3	4	Total

Name:
Number:
Department:
Signature:

Write your answers in the space provided for them.
Write neatly.
Write your name on each sheet.

1) (20 points) Order the following functions by growth rate. Explain in detail how you decided on the ordering.

$n^{\log n}$ $\quad e^{\log n}$ $\quad \log(n+n^2)$ $\quad n^{5n}$ $\quad 2^{3n}$ $\quad 3^{2n}$ $\quad n^{1000}\log n$

ANSWER:

First of all some general properties of growth rates:

A) In terms of slower to faster grwoth rate:
Logarithms (logn)<polynomials(n^a)<exponentials with const base (a^n) <exponentials with base n (an^{bn})

B) n^a grows slower than n^b if a<b
a^n grows slower than b^n if a < b

C) Some simplifications using the properties of log() and the

$e^{\log n}=n$

$\log(n+n^2) = O(\log n^2) = O(2\log n) = O(\log n)$

$n^{1000}\log n = O(n^{1001})$

Therefore we can rewrite the functions we need to order as:

$n^{\log n}$ $\quad n$ $\quad \log(n+n^2)$ $\quad n^{5n}$ $\quad 2^{3n} = 8^n$ $\quad 3^{2n} = 9^n$ $\quad n^{1000}\log n$

Using properties in A)

$\log(n+n^2) < n < n^{1000}\log n < n^{\log n}, 8^n, 9^n < n^{5n}$

D) $8^n < 9^n$ because the base 8 is smaller than base 9.

In order to compare some items we will use their limits:

$$\lim \frac{n^{1000}\log n}{\log n} < \lim \frac{n^{1001}}{\log} = \lim \frac{1}{n-} \quad 1_0$$

$$=$$

$$= \infty$$

$n\to\infty$ $\quad n \quad$ $n\to\infty$ n $\quad n_n\to\infty$ $n^{\log}1001$

$1000\log <$ $\quad \log n$

E) Therefore: $n \quad n \quad n$

In order to compare some other items we can use their logs:

$\log \quad = \quad 2 \quad \Rightarrow \quad =$

$\log(n^n) \quad \log \ n \quad \log(\log^2 n) = 2 \log \log n \quad O \quad n$

$(\log \log \ )$

$\log(8^n) = n \log 8 \Rightarrow \log(n \log 8) \quad n$

$= \quad O(\log \ )$

F) Therefore: $n^{\log} n \quad < 8^n$

Using D, E, F we can complete the ordering:

$+ \quad 2 \quad < \quad n < \quad n^{1000} \quad < \quad \log n < 8^n < \quad 9^n \quad < \quad 5$

$\log(n\, n \quad) \log n \quad n \quad n^n$

2) (30 points) A problem on graph is to assign integers to the vertices of *G* such that if there is a directed edge from vertex i to vertex j, then i is less than j. One solution to the problem is to search for a vertex with no incoming edge, assign this vertex the lowest number, and delete it from the graph, along with all outgoing edges. Repeat this process on the resulting graph, assigning the next lowest number, and so on.

2a) [20points] Write the data structures and algorithm necessary to solve this problem.

A two dimensional integer array is necessary to represent the topology of the graph and another single dimensional array can be used to keep values assigned to vertices:

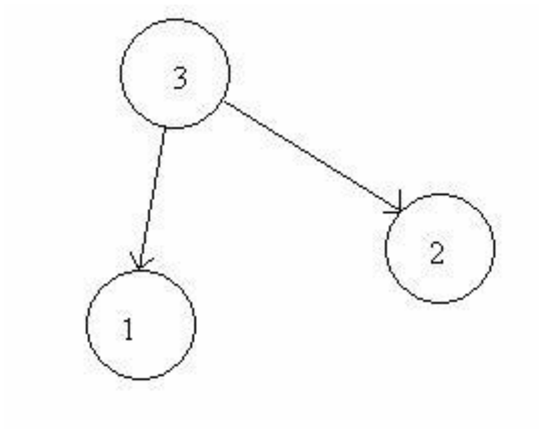
```
Type
Topology: Array [1..n,1..n] of integer {0/1};    {n represents the number of vertices}
Values: Array [1..n] of integer;

Function AssignNumbers (graph: Topology):Numbers: Values;

TempGraph: Topology    {TempGraph is used not to lose the original graph structure –
optional}
k: integer;

 TempGraph <- Graph
 k <- 1
 while k< (n+1)
begin
   find a column having no ‘1’s
   if no such column
   then return error
   else
     Numbers[j] <- k
     k++
     assign ‘0’ to all entries in row i
end {while}
return Numbers
```

2b) [5 points] What is the complexity of your algorithm in terms of the number of vertices n and/or the number of edges m?

$\theta(n^3)$, outher while loop is repeated n times. In the loop in order to locate a column having no ‘0’ entries one should check whole two dimensional array (worst case)

2c) [5points] Draw an example graph and show the relationship between your data structures and the graph.

	[1]	[2]	[3]
[1]	0	0	0
[2]	0	0	0
[3]	1	1	0


3
2
1

3) (20 points) Let g(*n*) be the number of ways to write a string of *n* zeros and ones so that there are never two successive zeros. For example, when *n*=1 the possible strings are 0 and 1, so g(1)=2; when *n*=2 the possible strings are 01, 11, and 10, so g(2)=3; when *n*=3 the possible strings are 010, 011, 101, 110, and 111, g(3)=5. Show that g(*n*)=f_{n+2} where f_{n+2} is (*n*+2)th Fibonacci number (f_0 = 0, f_1 = 1, ….).

n = 1 □ **0, 1** ;g(1) = 2

n = 2 □ 00 ; g(2) = 3
01
10
11

n= 3 □ 000 ; g(3) = 5
001
010
011
100
101
110
111

Let's take g(3)
101 -- These are the ones generated by g(2), we can put a '1' in front of them

110 --
111 --

However we cannot put a zero in front of the ones generated by g(2) since there can be double '0's. We can only put a zero in front the ones that start with a '1'. However, those are the ones generated g(1) and to the front of which '1's are placed to obtain the values in g(2). They are:
0*1*0
0*1*1

Recursively we can apply this to the other g(n)s and obtain:
g(n) = g(n-1) + g(n-2)