COMP 271 --- Spring 1999

MID-TERM EXAMINATION - March 20, 1999

Name: **Student ID**:

Lab Section (please circle): 1A 1B 2A 2B

<u>All questions should be answered within the space provided after each problem .</u>

1-24%] A divide and conquer algorithm is described by the following pseudo-code:

```
DC(i,j){
IF i<j THEN
        k=⌊(j-i+1)/3⌋
        DC(i,i+k-1)
        DC(i+k, i+2k-1)
        DC(i+2k,j)
        MERGE (i,j,k)
}
```

a-12%] Write the order of recursive calls for DC(0,8):

DC(0,8), DC(0,2), DC(0,0), DC(1,1), DC(2,2), MERGE(0,2,1), DC(3,5), DC(3,3), DC(4,4), DC(5,5), MERGE (3,5,1) DC(6,8), DC(6,6), DC(7,7), DC(8,8), MERGE(6,8,1), MERGE(0,8,3)

b-6%] What is the recurrence describing the cost of the algorithm as a function of the length n (=j-i) of the input assuming that the cost of merge is n^2.

$T(n)=3T(n/3)+ n^2$

c-6%] What is the complexity of the algorithm (hint: use master theorem).

$\Theta(n^2)$

2-26%] Give the pseudo-code for Dijkstra's algorithm. The pseudo-code *should also store information for computing the shortest path* (and not only its cost). Use the notation of the lecture notes: V is the set of nodes, v[0] is the initial node, S is the set of visited nodes, d(w,v) is the cost of edge (w,v) and D[w] is the minimum distance of node w from v[0].

```
Dijkstra(G, v[0])
S={v[0]}
FOR all other vertices v DO D[v]=d(v[0]),v)
WHILE (|S|<n) {
        select w from (V-S) with minimum D[W]
        add w to S;
        FOR each v in (V-S) DO
                cost=D[w]+d(w,v)
                IF cost < d[v] THEN
                        d[v]=cost;
                        P[v]=w;
}
```

3-30%] Consider the function *f(n,k)*= 0 of two non-negative integers *n* and *k* :
f(n,k)= 0, if *n*=0 or *k*=0 or *k*≥*n* and
f(n,k)=*f(n,k-1)*+ *f(n-1,k)* + *f(n-1,k-1)* +1, otherwise.

a-5%] Write a recursive algorithm that returns *f(n,k)*

function F(n, k)
IF n=0 or k=0 or k>=n THEN RETURN 0;
RETURN F(n,k-1)+ F(n-1,k)+ F(n-1,k-1)+1

b-10%] Write a dynamic programming algorithm that returns *f(n,k)*

```
function F(n, k)
FOR i=0 to n DO
        FOR j=0 to k DO
                IF i=0 or k=0 or k>=n
                                THEN F[i,j]=0
                                ELSE F[i,j]= F[i,j-1)+ F(i-1,j)+ F(i-1,j-1)+1
```

c-10%] Assuming that we want to compute *f(5,4)*, complete the following table

f(n,k)	k=0	k=1	k=2	k=3	k=4	k=5
n=0	0	0	0	0	0	0
n=1	0	0	0	0	0	0
n=2	0	1	0	0	0	0
n=3	0	2	4	0	0	0
n=4	0	3	10	15	0	0
n=5	0	4	18	44	60	0

d-5%] What is the time and space complexity of the dynamic programming algorithm as a function of *n* and *k*.

time and space complexity $\Theta(nk)$

4-20%] We use the dynamic programming algorithm (described in class) to find the optimal parenthesization for multiplying four matrices $A_1xA_2xA_3xA_4$ with dimensions 10x1, 1x20, 20x2 and 2x5 respectively.

a-15%] Complete the following tables with the intermediate costs and the optimal splittings.

m[i,j]	j=1	j=2	j=3	j=4
i=1	0	200	60	100
i=2		0	40	50
i=3			0	200
i=4				0

split[i,j]	j=1	j=2	j=3	j=4
i=1	0	1	1	1
i=2		0	2	3
i=3			0	3
i=4				0

b-5%] What is the optimal parenthesization?

$A_1x((A_2xA_3)\ xA_4)$