FINAL

CIS 102 - Spring 03

Warmuth

NAME: SOLUTIONS

Student ID:__________

This exam is closed book and closed notes. Show partial solutions to get partial credit.

If your answers are not written legibly, you won't get full credit.

Clarity and succinctness will be rewarded.

Question 1:	(out of 15)
Question 2:	(out of 10)
Question 3:	(out of 15)
Question 4:	(out of 15)
Question 5:	(out of 15)
Question 6:	(out of 15)
Question 7:	(out of 15)

Total: (out of 100)

1. Run the following three algorithms on the below graph:
   Fig. 23.1, p. 562, of CLRS Prim's algorithm for finding the minimum spanning tree;
   Kruskal's algorithm for finding the minimum spanning tree;
   and Dijkstra's algorithm for the single source shortest path problem starting at vertex a).

   In each case show in what order the vertices or edges are added and draw the resulting minimum spanning tree or shortest path tree.

   Prim: AB, AH, HG, GF, FC, CI, CD, DE
   (or AB, BG, CI, CF, FG, GH, CD, DE)
   Drawing of tree

   Kruskal: HG, CI, GF (last two may be swapped),
   AB, CF (last two may be swapped),
   CD, AH (or BC), DE
   Drawing of tree

   Dijkstra: AB, AH, HG, GF, BC, CI, CD, FE
   Drawing of tree

2. Given an adjacency matrix A of a directed graph G. Let B be $A \cdot A$, where $\cdot$ represents Boolean matrix multiplication (which uses $\vee$ instead of addition and $\wedge$ instead of multiplication). Let C be $A * A$, where now $*$ stands for the regular matrix multiplication.

   What is the meaning of $B[i, j]$ and $C[i, j]$?
   Give reasons for your answers.

   $B[i, j] = true$ **iff there is a path of length two from** i **to** j**:**
   $B[i, j] = \bigvee_{k=1}^{n} (A[i, k] \wedge A[k, j])$. **This expression is true if there exist an intermediate vertex fro which both** $A[i, k]$ **and** $A[k, j]$ **is true, i.e. there is a path of length two that goes via** k **from** i **to** j**.**

   $C[i, j]$ **= number of paths of length two from** i **to** j**:**
   $C[i, j] = \sum_{k=1}^{n} A[i, k]A[k, j]$. **Each path of length two via an intermediate vertex** k **(i.e.** $A[i, k]$ **and** $A[k, j]$ **are one) contributes one to the sum.**

3. Short questions:

   (a) What is the running time of the standard matrix multiplication algorithm for multiplying two $n \times n$ matrices?

   $$O(n^3)$$

   What is the running time of Strassen's matrix multiplication algorithm? (If you don't know the exact exponent, then give an approximate value of it.)

   $$O(n^{\log_2 7}) = 2.81$$

   (b) Name two sorting algorithms that require $O(n \log n)$ comparisons in the worst case.

   Heapsort and Mergesort

   (c) What are the two key ideas of the efficient union-find data structure?

   Pathcompression and weighted union
   (root of tree with less node is pointing to root of tree with more nodes; ties are broken arbitarily)

   (d) Give a definition of Binary Search Trees.

   Binary tree (i.e. each node has at most two children) with one key in each node. Inorder traversal produces keys in sorted order.

   (e) What three operations can be done in $O(\log n)$ time using Red Black Trees?

   Insert, Delete, Find.

4. For n distinct elements $x_1, x_2, \dots, x_n$ with positive weights $w_1, w_2, \dots, w_n$ such that $\sum_{i=1}^{n} w_i = 1$, the *weighted median* is the element x_k satisfying

$$\sum_{x_i < x_k} w_i < \tfrac{1}{2} \text{ and } \sum_{x_i > x_k} w_i \leq \tfrac{1}{2}.$$

Example:

i	1	2	3	4	5	6	7
x_i	8	3	6	-7	0	19	-4
w_i	.2	.1	.3	.05	.1	.05	.2
rank	6	4	5	1	3	7	2
median		*					
w. median			*				

(a) Argue that the median for $x_1, x_2, \dots, x_n$ is the weighted median of the x_i when the weights w_i are all equal $\frac{1}{n}$.

(b) Show how to compute the weighted median of n elements in $O(n \log n)$ worst-case time using sorting.
Reason your time bound!

(c) Show how to compute the weighted median in $\Theta(n)$ worst-case time using a linear time median algorithm as a subroutine. Give a recurrence and reason your time bound.

See Homework 6 for the solution!

5. Given an n by n array whose entries are either '.' or '+'. A legal path from position (1,1) (which is the lower left corner) to position (n,n) (upper right corner) is any path that always goes right from the current array point or up from the current point but avoids entries which contain a '+'. If we denote a path using (x, y) coordinates then going along a legal path must always either increase the x or the y coordinate by one. The goal of this exercise is to compute the number of legal paths from $(1, 1)$ to (n, n).

   Examples of paths:

$$
{\scriptstyle (1,1)}\begin{pmatrix}
. & & + & & . & \rightarrow & . & \rightarrow & . \\
 & & & & \uparrow & & & & \uparrow \\
. & & + & & . & & + & & . \\
 & & & & \uparrow & & & & \uparrow \\
. & \rightarrow & . & \rightarrow & . & & . & \rightarrow & . \\
\uparrow & & & & & & \uparrow & & \\
. & & + & & + & & . & & . \\
\uparrow & & & & & & \uparrow & & \\
. & \rightarrow & . & \rightarrow & . & \rightarrow & . & & .
\end{pmatrix}^{(n,n)}
$$

   Let $T(i, j)$ be the number of legal paths from $(1, 1)$ to (i, j).

   a) Give a recurrence for $T(i, j)$.
      $T(i, j)$ equals zero of entry $(i, j) = +$ and $T(i - 1, j) + T(i, j - 1)$ otherwise.

   b) How should the array be initialized. **Add a row zero and column zero with + entries.**

   c) In what order should the array be evaluated.
      Execute above recurrence ($0 \leq i \leq n, 0 \leq j \leq n$) in row major or column major order.

   d) and what is the time bound of the dynamic programming algorithm.
      $O(1)$ per entry and thus $O(n^2)$ altogether.

   Alternate for b):
   For $1 \leq i \leq n$, $T(0, i) := T(i, 0) := 0$.

   Now execute recurrence for $1 \leq i, j \leq n$ in row or column major order.

6. Suppose you are given three strings of characters:

$$X = x_1x_2...x_m,\ Y = y_1y_2...y_n,\ \text{and}\ \ Z = z_1z_2...z_{n+m}$$

Z is said to be a *shuffle* of X and Y if Z can be formed by interspersing the characters from X and Y in a way that maintains the left-to-right ordering of the characters from each string. For example, *cagotat* is a shuffle of *cat* and *goat*, but *cogatt* is not. Devise a dynamic programming algorithm that takes as input X, Y, Z, m, and n and determines whether Z is a shuffle of X and Y.

Hint: Construct an m by n 2-dimensional table. Let $T(i, j)$ be true iff $z_1...z_{i+j}$ is a shuffle of $x_1...x_i$ and $y_1...y_j$.

Give a recurrence! In what order do you evaluate your table? How do you initialize the table? What is the running time of your algorithm.

		j: 0	1 g	2 o	3 a	4 t
	0	T	F	F	F	F
	1 c	T	F	F	F	F
i	2 a	T	T	T	F	F
	3 t	F	F	T	T	T

1	2	3	4	5	6	7
c	a	g	o	t	a	t

Recurrence:
$T(i, j) = (T(i-1, j) \wedge x_i = z_{i+j}) \vee (T(i, j-1) \wedge y_j = z_{i+j})$

Row major or column major.

Initialization: Row -1 and column -1 filled with T.
Or: $T(0, 0) = T$ **and T whenever recurse on entry** $T(i, j)$ **for which** i **or** j **is negative.**

$O(1)$ **per entry,** $O(nm)$ **in total.**

7. Modify the following DFSsweep to detect a cycle in a directed graph G.

```
DFSsweep(G)
   for each vertex u ∈ V(G)
      do color[u]:=white
   for each vertex u ∈ V(G)
      do if color[u]=white
         then DFS(u)

DFS(u)
   color[u]= gray                       • White vertex u has just been discovered
   for each adjacent v in the edge list of u          • Explore edge (u, v)
      do if color[v]=white
         then DFS(v)
         else if color[v]=gray then PRINT CYCLE and goto exit
   color[u]=black                       • Blacken u since its finished

PRINT NO CYCLE
exit:
```

Simply add some statements to the above algorithm.
What is the running time of your cycle detection algorithm?

Added in bold above. Time is same as DFSsweep: $O(n + e)$.