EE608: Final Exam for Spring 1998

Name: ____________________________________

Login: ____________________________________

Student ID: ____________________________________

1.	________________	**(20 points)**
2.	________________	**(20 points)**
3.	________________	**(20 points)**
4.	________________	**(20 points)**
5.	________________	**(20 points)**
Total:	________________	**(100 points)**

This test is closed book, closed note. You may not use a calculator or any other resources other than the formulae and algorithm handout provided by Professor Harper. If you are not a member of the class, you must obtain permission to take the exam.

1. Answer the following questions concerning algorithm basics. **(20 points total)**

   (a) Consider the following code segment. Determine as tight a bound as you can on the number of iterations for the **while** loop. **(5 points)**

```
1. $r \leftarrow 1$
2. $t \leftarrow n * n * n$
3. while $r < n$
4.         do $r \leftarrow r + r + r$
```

(b) Solve the following recurrence in closed form. **(5 points)**

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ T(n-1) + n^2 & n > 1. \end{cases}$$

(c) Prove or disprove the following. Let $f(n)$ and $g(n)$ be two monotonically increasing positive functions such that $f(n) = o(g(n))$ and the limit as n goes to infinity of $f(n)$ is infinity, then $\lg(f(n)) = o(\lg(g(n)))$. **(5 points)**

(d) Suppose that you have two algorithms A and B such that the worst-case running time of A is $3\,n^2 + 3n$ and the worst-case running time of B is $20\,n\lg n + 3n + 6$. Describe what factors would impact your choice of algorithm. **(5 points)**

2. Answer the following questions concerning dynamic set operations and data structures (e.g., hash tables, binary search trees, red-black trees). **(20 points total)**

   (a) Describe the concept of **universal hashing**. Explain all of the requirements of a universal collection of hash functions. **(5 points)**

   (b) Let T be a binary search tree, x be a leaf node, and y be x's parent. Show that $key[y]$ is either the smallest key in T larger than $key[x]$ or the largest key in T smaller than $key[x]$. **(5 points)**

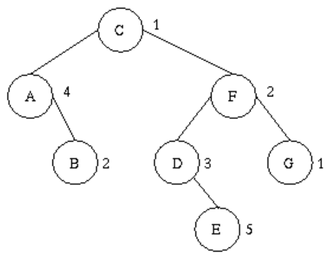
(c) Describe two dynamic set operations for which using a balanced binary search tree is more time efficient than the same operations on a heap. Explain your choices. Under what circumstances is a heap preferred to a balanced binary search tree? **(5 points)**

(d) How is tree imbalance in red-black trees detected during the insertion of new nodes into the tree? What operations are used by red-black trees to restore tree balance? Why is it that these operations restore tree balanace? Be precise and to the point. **(5 points)**

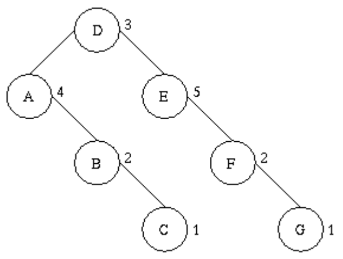
3. Consider the problem of constructing an optimal binary search tree, which is a binary search tree constructed in such a way as to minimize the expected number of comparisons required to search for keys based on their expected frequency of occurrence in the search process. For example, suppose out of every 18 searches in a tree, we expect the keys to appear with the following frequencies:

INDEX	**Key**	**Frequency**
1	A	4
2	B	2
3	C	1
4	D	3
5	E	5
6	F	2
7	G	1
Total		18

The two binary search trees below meet the binary search tree property with $A < B < C < D < E < F < G$, but tree (b) is a lower cost tree given the key frequencies specified in the above table. Tree (a) requires 51 comparisons if the 18 keys (i.e., 4 A's, 2 B's, etc.) are searched for in that tree (as shown in the table below). Tree (b), on the other hand, requires only 41 comparisons; hence, the tree has a lower cost than Tree (a). The table below shows how the number of comparisons for Tree (a) and (b) are calculated.



(a)



(b)

INDEX	**Key**	**Frequency**	**Comparisons for (a)**	**Comparisons for (b)**
1	A	4	8	8
2	B	2	6	6
3	C	1	1	4
4	D	3	9	3
5	E	5	20	10
6	F	2	4	6
7	G	1	3	4
Total		18	51	41

The goal is to create an optimal binary search tree over $key[1] \ldots key[n]$. If we know what the root of the optimal binary search tree is, say $key[k]$, then we also know that $key[1] \ldots key[k-1]$ are stored in the left subtree and $key[k+1] \ldots key[n]$ are stored in the right subtree. This problem clearly has the optimal substructure property; optimal binary search trees are made up of optimal binary search trees.

Below is a dynamic program for creating an optimal binary search tree. Note that key is an array such that $key[1] \leq key[2] \leq \ldots \leq key[n]$. Note also that $freq[i]$ corresponds to the expected frequency of $key[i]$. When determining the cost of a search tree, it is necessary to sum the cost of each subtree together with an additional amount that represents the cost of adding 1 more comparison for the elements in the subtrees and the root (that is what is calculated in *node_cost* in the code).

```
TREE-COMPUTE($key, freq$)
1.  $n \leftarrow length[key]$
2.  for $i \leftarrow 1$ to $n$
3.      do $C[i,i] \leftarrow freq[i]$
4.  for $i \leftarrow 1$ to $n+1$
5.      do $C[i,i-1] \leftarrow 0$
6.  for $l \leftarrow 1$ to $n-1$
7.      do for $i \leftarrow 1$ to $n-l$
8.             do $j \leftarrow i+l$
9.                $node\_cost \leftarrow 0$
10.               for $k \leftarrow i$ to $j$
11.                   do $node\_cost \leftarrow node\_cost + freq[k]$
12.               $C[i,j] \leftarrow \infty$
13.               for $k \leftarrow i$ to $j$
14.                   do $cost \leftarrow C[i,k-1] + C[k+1,j] + node\_cost$
15.                      if $cost < C[i,j]$
16.                        then $C[i,j] \leftarrow cost$
17.                             $root[i,j] \leftarrow k$
18. return $C$ and $root$
```

Answer questions concerning this code segment below. **(20 points total)**

(a) What does $C[i,j]$ represent in the above code? **(3 points)**

(b) What is an expression denoting the cost of the optimal binary search tree over all of the n keys? **(2 points)**

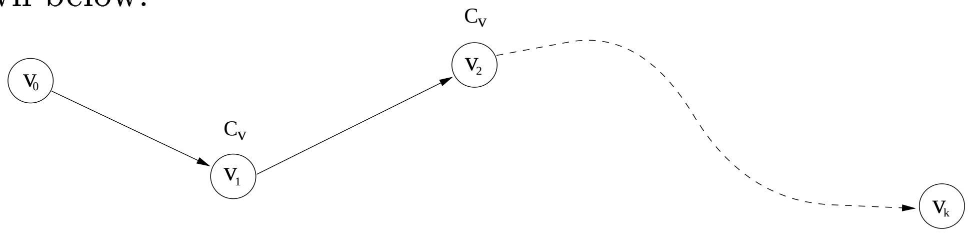
(c) What does $root[i, j]$ represent in the above code? **(2 points)**

(d) What are the space requirements of the above program. Explain. **(3 points)**

(e) Analyze the running time of the above algorithm using summations which you evaluate. **(5 points)**

(f) Can we extract the optimal binary search tree based on information computed by the above code? If so, explain how. If not, explain what needs to be added. **(5 points)**

4. Single-source shortest path with vertex costs. Let $G(V, E)$ be a weighted directed graph, where $w(u, v)$ is the cost of the edge from vertex u to v. All edges in G have non-negative cost, i.e. $w(u, v) \geq 0$ for all u, v. In many real life scenarios (computer networks, time for a plane flight), the cost of a path P is equal to the sum of the edge costs of P plus a fixed cost C_v for each intermediate vertex in P. Thus, the cost of the path $(v_0, v_1, \ldots, v_k)$ is $w(v_0, v_1) + w(v_1, v_2) + \cdots + w(v_{k-1}, v_k) + (k - 1) * C_v$ as shown below.



Throughout this problem, assume $d[v]$ represent the shortest known distance so far from the source vertex s to vertex v. Let S be the set of vertices to which we have found the shortest paths and let u be the last vertex added to S.

(20 points total)

(a) Show how to modify Dijkstra's algorithm to find the shortest path from a source vertex to all other vertices in V according to the above cost definition. Give high-level pseudo code. **(10 points)**

(b) Prove your algorithm produces the correct result. To prove the correctness you need to show that for each $u \in V$, $d[u] = \delta(s, u)$ at the time u is inserted into S, and that the equality is maintained thereafter, with $\delta(s, u)$ modified to include the intermediate node costs. **(10 points)**

5. The following questions concerns KING ARTHUR'S PROBLEM (KAP), which is defined as follows:

   INSTANCE: n knights and a list of pairs of knights that are enemies (note that if a is an enemy of b, then b is an enemy of a).

   QUESTION: Is it possible to arrange the knights around a round table such that no pair of knights who are enemies sit side by side? (Note that the table is filled by all of the knights with each knight seated next to two individuals in a circle.) **(20 points total)**

   (a) Show that KING ARTHUR'S PROBLEM is in NP.

   (b) Show that KING ARTHUR'S PROBLEM (KAP) is NP-hard by HAMILTONIAN CIRCUIT PROBLEM $\propto$ KAP. **(10 points)**

(c) Suppose that you later discover that KING ARTHUR'S PROBLEM is a member of P. What can you conclude about the relationship between P and NP? Explain. **(5 points)**