1. (20 points) Please answer the following questions regarding binary search trees:

   (a) Calculate the number of different binary trees with 6 nodes. (10 points)

   (b) Insert the following elements into an initially empty binary search tree with the specified order: 15, 29, 68, 52, 21, 7, 2, 60, 59. Draw the binary search tree after all elements are inserted. (5 points)

   (c) Delete 7 and 29 from the binary search tree in the previous question. Draw the binary search tree after these 2 deletions. (5 points)

2. (15 points) Please answer the following questions regarding hash tables:

   (a) Consider inserting the keys 7, 14, 16, 8, 40, 85, 75, 19, 5, 27, 36 into an initially empty hash table of length 7 in the specified order. Each slot in the hash table can store 2 keys. Collisions are resolved by using linear probing. Assume that the hash function is defined as $h(x) = x \mod 7$. Draw the hash table after all keys are inserted. (5 points)

   (b) Assume that we have a hash table with collisions resolved by chaining. Attempting to improve the performance of this implementation, we use a red-black tree for each slot instead of an unsorted linked list, resulting in a hash table with collisions resolved "by red-black trees". Assume that a simple uniform hashing function is utilized and the load factor of the hash table is α. What is the running time for unsuccessful searches and insertions for our new implementation, respectively? (10 points)

3. (15 points) Please answer the following questions regarding sorting algorithms:

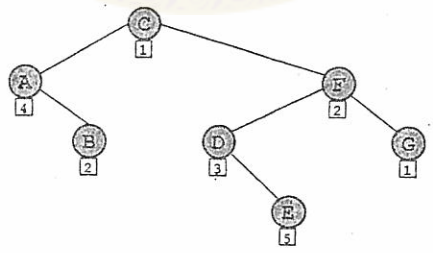
   (a) Prove that any comparison sort algorithm requires $\Omega(n \log n)$ comparisons in the worst case. (10 points)

   (b) The worst-case behavior for quicksort occurs when the partitioning routine produces one subproblem with $n-1$ elements and one with 0 elements. In this case, the recurrence for the running time is $T(n) = T(n-1) + T(0) + \Theta(n)$. Use the substitution method to determine the solution of $T(n)$. (5 points)

4. (16 points) Given n search keys, $K_1 < K_2 < \ldots < K_n$ and their associated frequencies, we often want to construct an optimal binary search tree for better search performance. Here, the cost of a binary search tree is defined as $\sum_{i=1}^{n} f(K_i) \cdot d(K_i)$, where $f(K_i)$ is the key K_i's frequency and $d(K_i)$ is K_i's depth in the tree. For example, given the following keys and their frequencies,

key	A	B	C	D	E	F	G
frequency	4	2	1	3	5	2	1

   the cost of the following binary search tree is $4 \cdot 2 + 2 \cdot 3 + 1 \cdot 1 + 3 \cdot 3 + 5 \cdot 4 + 2 \cdot 2 + 1 \cdot 3 = 51$.



   (a) Let $C(i,j)$ be the cost of the optimal binary search tree for $K_i, \ldots K_j$. It can be written as the following formula.

$$C(i,j) = \begin{cases} \underline{\quad 4A \quad} & \text{if } i > j, \\ \underline{\quad 4B \quad} & \text{otherwise.} \end{cases}$$

   Please complete the above formula? (6 points)

   (b) What is the optimal binary search tree for the above example and what is its cost? (10 points)

見背面

5. (16 points) The string matching problem can be formulated as the following. Given a text $T[1,\ldots,n]$ (n characters) and a pattern $P[1,\ldots,m]$ ($m \leq n$) (both of which are strings over the same alphabet), find all occurrences of P in T. We say that P occurs in T with shift s if $P[1,\ldots,m] = T[s+1,\ldots,s+m]$.

   (a) What is the lower bound of any string matching algorithm and why is it? (4 points)

   (b) Knuth-Moris-Pratt (KMP) algorithm is an efficient string matching algorithm which requires a preprocessing. What is its running time for preprocessing and matching respectively? (4 points)

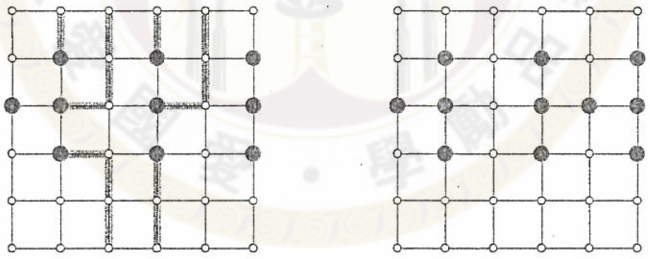
   (c) During preprocessing, the KMP algorithm computes a prefix function $\pi : \{1,\ldots,m\} \to \{0,\ldots,m-1\}$. What is the prefix function given the following text and pattern? (4 points)

   (d) Find all occurrence shifts. (4 points)

   T: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
   a a b a b b a b a b b b a b a b b a a a b b

   P: 1 2 3 4 5 6 7 8
   a b a b b a a a

6. (18 points) The *escape problem* is defined as the following. An $n \times n$ gird is an undirected graph consisting of n rows and n columns of vertices The following Figures show 6×6 grids. We denote the vertex in the ith row and jth column by (i, j). All vertices in a grid have exactly four neighbors, except for the boundary vertices, which are the vertices (i, j) for which $i = 1$, $i = n$, $j = 1$ or $j = n$. Given $m \leq n^2$ starting vertices in the grid, the escape problem is to determine whether or not there are m **vertex-disjoint** paths from the starting vertices to any m different vertices on the boundary. Vertex-disjoint paths mean that each vertex can be used at most once in the escape. The following figures give examples of a grid with an escape (on the left) and a grid without (on the right). Black vertices are the starting vertices and the shaded paths show the escape. Show how to convert the escape problem into the maximum flow problem. It is enough to give the conversion procedure. It is not required to show the correctness of your procedure.



試題隨卷繳回