Yao-Wen Chang

Practice Midterm #1

Instructions. This is a 150-minute test, with a total of 114 points available. There are four pages and eight problems. **Pace Yourself and Good Luck!**

Problem 1. (16 pts total) For each of the following recurrence relations, give the asymptotic growth rate of the solution **using the Θ notation**. Assume in each case that $T(n)$ is $\Theta(1)$ for $n \leq 10$.

(a) (5 pts) $T(n) = 5T(n/2) + n^3 \lg n$.

(b) (5 pts) $T(n) = T(\sqrt{n}) + \lg \lg n$.

(c) (6 pts) $T(n) = T(pn) + T(qn) + n$, where $p + q = 1$.

Problem 2. (20 pts total) Please give a brief answer for each of the following questions.

Q1. If $f(n)$ is asymptotically positive, is $f(n) + o(f(n)) = O(f(n))$? Why?

Q2. Can we have a priority queue in the comparison sorting model with both the properties: (1) EXTRACT-MIN runs in $O(\lg \lg n)$ time, and (2) BUILD-MAX-HEAP runs in $O(n)$ time? Why?

Q3. Suppose that an array contains n numbers, each of which is 0, 1, or 2. Then, can this array be sorted in linear time in the worst case? How to do it if it can, or why it is not possible?

Q4. Can we put the numbers 1, 2, ..., 10 in a tree such that it is both a valid max-heap and binary search tree at the same time? Why?

Problem 3. (12 pts total) Let A_0 be a numerical array of length n, originally sorted into ascending order. Assume that k entries of A_0 are overwritten with new values, producing an array A. Furthermore assume you have an array B containing n boolean values (0 or 1), where $B[i]$ is true if $A[i]$ is one of the k values that was overwritten, and false otherwise.

(a) (9 pts) Give a fast algorithm to sort A into ascending order, with time complexity better than $O(nk)$. (**Hint: Separate out A into two lists.**)

(b) (3 pts) Give the time complexity of your algorithm in big-O notation, as a function of n and k (the tighter, the better).

Problem 4. (16 pts total) Search trees.

(a) (5 pts) Give the binary search tree that results from successively inserting the keys 9, 10, 2, 1, 7, 6, 8 into an initially empty tree.

(b) (5 pts) Label each node in the tree with R or B denoting the respective colors RED and BLACK so that the tree is a legal red-block tree.

(c) (6 pts) Give the red-black tree that results from inserting the key 3 into the tree of (b). (**Hint:** The pseudocode for inserting a node in a red-black tree is given below.)

```
RB-Insert(T, x)
Tree-Insert(T, x);
color[x] ← RED;
while x ≠ root[T] and color[p[x]] = RED do
  if p[x] = left[p[p[x]]] then
        y ← right[p[p[x]]];
        if color[y] = RED then   /* uncle's color is red */
        /* Case 1 */
            color[p[x]] ← BLACK;
            color[y] ← BLACK;
            color[p[p[x]]] ← RED;
            x ← p[p[x]];
        else if x = right[p[x]] then   /* uncle's color is black */
        /* Case 2*/
                x ← p[x];
                Left-Rotate(T, x);
        /* Case 3*/
            color[p[x]] ← BLACK;
            color[p[p[x]]] ← RED;
            Right-Rotate(T, p[p[x]]);
  else (same as then clause with "right" and "left" exchanged)
color[root[T]] ← BLACK;
```

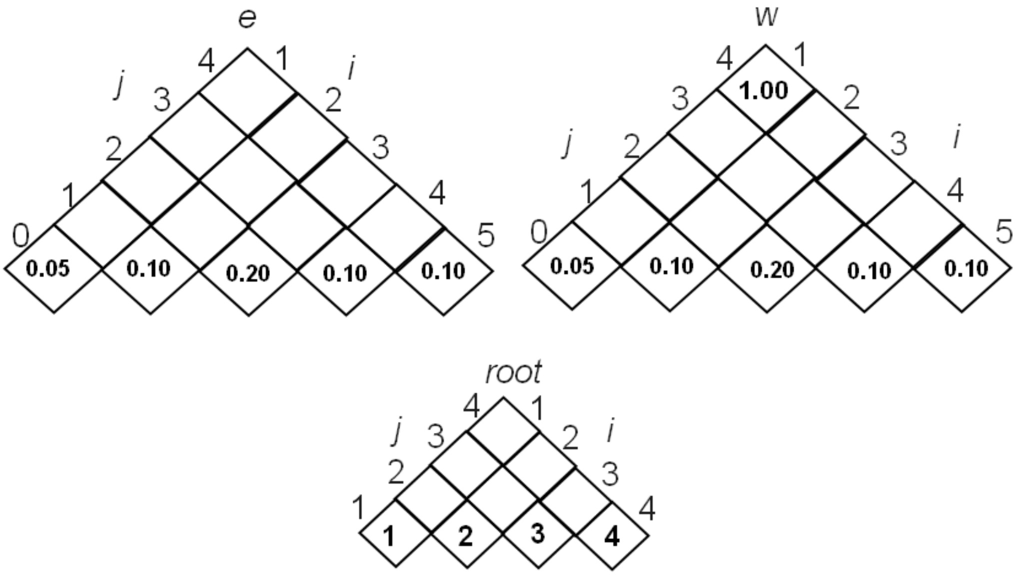
Problem 5. (16 pts total) You are asked to determine the cost and structure of an optimal binary search tree for a set $K =< k_1, k_2, k_3, k_4 >$ of $n = 4$ keys with the following probabilities:

i	0	1	2	3	4
p_i	-	0.10	0.10	0.20	0.05
q_i	0.05	0.10	0.20	0.10	0.10

a set of probabilities $P =< p_1, p_2, p_3, p_4 >$ for searching the keys in K and $Q =< q_0, q_1, q_2, q_3, q_4 >$ for unsuccessful searches, as discussed in class.

(a) (12 pts) Fill every field in the e, w, and $root$ tables, where $e[i, j]$ gives the expected cost of searching an optimal binary tree containing the keys $k_i, \ldots, k_j$, $w[i, j] = w[i, j-1] + p_j + q_j$, and $root[i, j]$ records the root of the subtree containing the keys $k_i, \ldots, k_j$.

(b) (4 pts) Find an optimal binary search tree of the given probabilities and give the expected search cost of the tree.



Problem 6. (12 pts total) Professor Chang plans to drive a car from Taipei to Tainan along the Formosa highway (Highway #3) for a meeting. His car's gas tank, when full, holds enough gas to travel n kilometers, and his map gives the distances between gas stations on his route. He wishes to make as few gas stops as possible along the way. Give an efficient algorithm to determine at which gas stations he should stop and prove the optimality of your algorithm.

Problem 7. (18 pts total) Given a log of wood of length k, Woody the woodcutter will cut it once, in any place you choose, for the price of k dollars. Suppose you have a log of length L, marked to be cut in n different locations labeled $1, 2, \ldots, n$. For simplicity, let indices 0 and $n+1$ denote the left and right endpoints of the original log of length L. Let the distance of mark i from the left end of the log be d_i, and assume that $0 = d_0 < d_1 < d_2 < \ldots < d_n < d_{n+1} = L$. The wood-cutting problem is the problem of determining the sequence of cuts to the log that will (1) cut the log at all the marked places, and (2) minimize your total payment to Woody.

(a) (4 pts) Give an example illustrating that two different sequences of cuts to the same marked log can result in two different costs.

(b) (8 pts) Let $c(i, j)$ be the minimum cost of cutting a log with left endpoint i and right endpoint j at all its marked locations. Suppose the log is cut at position m, somewhere between i and j. Define the recurrence of $c(i, j)$ in terms of i, m, j, d_i, and d_j. Briefly justify your answer.

(c) (6 pts) Using part (b), give an efficient algorithm to solve the wood-cutting problem. Use a table C of size $(n + 1) \times (n + 1)$ to hold the values $C[i][j] = c(i, j)$. What is the running time of your algorithm?

Problem 8. (4 pts total) Please give **two** of your opinion(s)/comment(s) on this course (e.g., strengths and weaknesses)? Any **specific** comments that may improve the quality of this course are very much welcome. Thank you.