

## CSPP 55001 Algorithms — Autumn 2009

### Homework 4 (assigned October 21, due October 28)

*Reading:* CLRS chapters 8, 11, and 12.

*Written assignment:* Solve the following "Do" exercises and assigned problems. **Only solutions to the assigned problems should be turned in.**

Note: You are responsible for the material covered in **both** "Do" exercises and assigned problems.

**Note: If you work with others, indicate their names at the top of your homework paper. Everyone must submit their own independently written solutions.**

#### "Do" Exercises (*not* to be handed in):

1. Problem 10-1 on page 249.  
2nd Edition, Problem 10-1, page 217.
2. Exercises 11.2-1, 11.2-2 on page 261; Exercises 11.3-1, 11.3-3 on pages 268–269.  
2nd Edition, Exercises 11.2-1, 11.2-2 on pages 228–229; Exercises 11.3-1, 11.3-3 on page 236.
3. Exercises 12.2-1a, b, 12.2-5, 12.2-8 on pages 293–294; Exercises 12.3-4, 12.3-5 on page 299.  
2nd Edition, Exercises 12.2-1a, b, 12.2-5, 12.2-8 on pages 259–260; Exercises 12.3-4, 12.3-5 on page 264.

#### Problems (to be handed in):

1. (1) Suppose that a hash table with  $m$  slots contains a single element with key  $k$ ; the rest of the slots are empty. Suppose that you search  $t$  times in the table for various other keys not equal to  $k$ . Assuming simple uniform hashing, what is the probability that one of the  $t$  searches probes the slot containing the single element stored in the table? (5 points)  
(2) You are given a hash table with  $n$  keys and  $m$  slots, with the simple uniform hashing assumption, i.e., each key is equally likely to be hashed into each slot. Collisions are resolved by chaining. What is the probability that the first slot is empty? (5 points)
2. You are given a sorted array  $A[1..n]$  of  $n$  numbers. Describe in pseudocode an algorithm that constructs a binary search tree containing the same numbers. The tree should be roughly balanced (its height should be  $O(\lg n)$ ) and the running time of your algorithm should be  $O(n)$ . (10 points)
3. A certain string-processing language includes a primitive operation which breaks a string into two pieces. Because this operation copies the original string, it costs  $n$  units of time to break a string of length  $n$  into 2 pieces, regardless of the location of the cut. Suppose that you want to break a string into many pieces. The order in which the breaks are made can affect the total running time. For example, if you want to cut a 20-character string at positions 3 and 10, making the first cut at position 3 costs a total of  $20 + 17 = 37$ , while making the first cut at position 10 has a better cost of  $20 + 10 = 30$ .  
Give a dynamic programming algorithm that, given a string  $S$  of length  $n$  and an array  $L[1..m]$  containing the break points, computes the minimum cost of breaking the string into  $m + 1$  pieces. Define the entries of your output table (the "brain" of the solution) and give the formula for the recurrence and the base cases (the "heart" of the solution). Describe your algorithm in pseudocode. What is the running time of your algorithm? (20 points)

4. You are given an array  $A[1..n]$  of  $n$  integers. You want to determine whether or not they are all distinct, i.e., are there  $i$  and  $j, i \neq j$ , such that  $A[i] = A[j]$ ?
1. How can you use sorting to solve the above problem? (5 points)
  2. Show that any algorithm that solves the above problem by using only comparisons requires  $\Omega(n \lg n)$  comparisons. (Hint: show that you must have sorted the array.) (5 points)  
Observation: in other words, your solution to part (1) is asymptotically optimal.
  3. Suppose you have a good hash function, i.e., on a random input the expected number of collisions would be constant. Give an algorithm with expected running time  $O(n)$  to solve the above problem on a random input. Argue that the bound is in fact  $O(n)$ . (5 points)
  4. Suppose now that the keys are all integers in the range 1 to  $n^2$ . Give an  $O(n)$ -time algorithm to solve the above problem. Argue that  $O(n)$  is the bound. (5 points)
  5. *Challenge problem.* Assume that you can evaluate any polynomial of degree  $d$  in  $\sqrt{d}$  time. Give an  $O(n)$ -time algorithm to solve the above problem. (5 bonus points)
- 

*Gerry Brady and Janos Simon*

*Thursday October 22 15:47:38 CDT 2009*