Exam in Algorithms and Data Structures 1 (1DL210)

Department of Information Technology
Uppsala University
2015–10–22
Lecturers: Mohamed Faouzi Atig, Tuan Phong Ngo, Othmane Rezine and Jari Stenman
Location: Polacksbacken, skrivsalen
Time: 14:00 - 19:00
No books or calculator allowed

Directions:

1. Do not write on the back of the paper
2. Write your anonymous code on each sheet of paper
3. **Important** Unless explicitly stated otherwise, justify you answer carefully! Answers without justification do not give any credits.

Good Luck!

Problem 1 (7p)

Order these functions in order of increasing asymptotic growth rate [1]. If two of them have the same asymptotic growth rate, state that fact. No justification is needed.

$$4\,log(n) \qquad n\,log(n) \qquad log(n^3) \qquad log(n) \qquad log(2^n) \qquad (\frac{1}{4})^n \qquad (\frac{1}{2})^{2n}$$

Problem 2 (6p)

State whether the following statements are true or false. No explanation is needed.

(i) If $f(n) = \Omega(g(n))$ and $g(n) = \Omega(h(n))$ then $f(n) = \Omega(h(n))$

(ii) If $f(n) = \mathcal{O}(g(n))$ then $g(n) = \Omega(f(n))$

(iii) The worst case running time for Heapsort is $\Omega(n)$

[1] Here, *log* denotes the binary logarithm.

(iv) The best case running time for Merge Sort is $O(n\,log(n))$

(v) The *maximal* height (i.e. the maximum number of levels) of a complete tree (or a heap) with n elements is $log(n)$.

(vi) Consider the recurrence

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T\left(\frac{n}{2}\right) \;+\; 2n & \text{if } n > 1 \end{cases}$$

Is it the case that $T(n) = \Omega(n)$?

Problem 3 (6p)

Consider the algorithm SameValueEveryWhere, given below.

```
SameValueEveryWhere(A)
n ← A.length
for j ← 1 to n − 1
      do s ← j
         for i ← j + 1 to n
             do if A[i] ≠ A[s]
                   then return FALSE
return TRUE
```

a) What does SameValueEveryWhere do ? No justification is needed

b) Give a tight asymptotic upper bound for the worst case asymptotic running time of SameValueEveryWhere.

c) Propose a different way of doing the same thing that is asymptotically faster in the worst case.

Problem 4 (6pt) Assume that you are giving an array S on integers. Describe a $O(nlog_2(n))$-time algorithm that determine whether or not there exists a pair of elements x and y in the array S such that $x = y + 1$.

Problem 5 (10p)

Give the max-heap (i.e., priority queue) that results when the keys

`10 12 1 15 8 19 11 13 6 7 13 9 6`

are inserted (using the function MAX-HEAP-INSERT) into an initially **empty** max-heap (i.e., priority queue) in the order they are listed (first 10, then 12, and so on).

Problem 6 (5p) Assume that we have a max-heap H and an integer x such that x is **strictly larger** than any other key appearing in H. Assume also that we construct the heaps H_1 and H_2 in the following way:

- Let H_1 be the resulting heap after executing MAX-HEAP-INSERT(H, x).
- Let H_2 be the result of executing HEAP-EXTRACT-MAX(H_1).

Under the assumption that all the keys appearing in H are **different**, is it the case that $H_2 = H$? If your answer is yes, justify it. If your answer is no, give a counterexample by providing concrete values for H and x and drawing H_1 and H_2.

Problem 7 (10p)

Assume you have the set $S = \{1, 8, 15, 22, 29, 36\}$ and you want to insert them into a hash table T of size at most 10, using chaining to resolve collisions.

a) Provide a size of T and a suitable hash function h, such that

   - the distribution of elements in T by using h would be **good** for random input
   - h performs **badly** for the elements in S

b) Provide a size of T and a suitable hash function h, such that

   - the distribution of elements in T by using h would be **bad** for random input
   - h performs **well** for the elements in S

Problem 8 (18p)

a) Consider inserting the following keys into a hash table of length $m = 13$, in the order they are listed (first 152, then 44, and so on):

$$152 \;\; 44 \;\; 39 \;\; 22 \;\; 134 \;\; 53 \;\; 144 \;\; 131 \;\; 0 \;\; 135$$

The auxiliary hash function is given by $(k \mod m)$. Draw the resulting hash table if we use chaining to resolve collisions.

b) Consider a hash table H of a given size $n > 0$. Does increasing the size of H to $3n$ necessarily imply that the probability of collisions decreases by approximately one third?

Problem 9 (16p)

a) Suppose that we start from an empty binary search tree and insert the following elements: 10,11,9,8,15,16,4,3,20,5. Then, we delete the elements: 3 and 10. Show the tree you obtain after each insertion/deletion.

b) Suppose that we have numbers between 1 and 1000 in a binary search tree and want to search for the number 363. Which of the following sequences could not be the sequence of nodes examined?

- 2, 25, 400, 329, 310, 134, 397, 363.
- 924, 200, 11, 24, 89, 258, 362, 363.
- 925, 202, 11, 124, 12, 245, 363.
- 2, 309, 307, 19, 266, 382, 31, 278, 363.
- 935, 78, 47, 21, 299, 392, 358, 363.
- 100, 145, 200, 202, 239, 300, 330, 363.

Problem 10 (16p)

a) Suppose that we first insert an element x into a binary search tree that does not already contain x. Suppose that we then immediately delete x from the tree. Will the new tree be identical to the original one? If *yes* give the reason. If *no* give a counter-example. Draw pictures if necessary.

b) Is the operation of insertion in a binary search tree commutative in the sense that inserting x and then y from a binary search tree leaves the same tree as inserting y and then x? Argue why it is so or give a counter-example.