CSE100 Algorithm Design and Analysis — Homework set #2
Fall semester 2018 — Miguel Á. Carreira-Perpiñán

Total possible marks: 100. You have to submit this homework to a TA due Oct. 17 at 8 pm or at your own lab session (also at TA's office hours if it is before the deadline). Homeworks must be done in groups of up to 3 students (one submission per group, listing materials consulted). This set covers chapters 6–12 of the textbook *Introduction to Algorithms*, 3rd. ed., by Cormen et al.

Exercise 1: heaps (5 points). (Exercise 6.5-8 in the book.) The operation HEAP-DELETE(A, i) deletes the item in node i from heap A. Give an implementation of HEAP-DELETE that runs in $O(\lg n)$ time for an n-element max-heap.

Exercise 2: quicksort (21 points). (Combination of exercises 7.1-2, 7.2-2 and problem 7.2 in the book.) Consider an array A with n elements, and refer to the pseudocode for the algorithms PARTITION (p. 171), QUICKSORT (p. 171) and RANDOMIZED-QUICKSORT (p. 179).

1. (3 points) What value of q does PARTITION return when all elements in the array $A[p\,..\,r]$ have the same value? Modify PARTITION so that $q = \lfloor (p+r)/2 \rfloor$ when all elements in the array $A[p\,..\,r]$ have the same value. *Hint*: this can be done by adding a short piece of pseudocode just before the **return** statement in PARTITION. You may give just this piece of pseudocode as the solution (no need to write the rest).

2. (3 points) What is the running time of QUICKSORT when all elements of array A have the same value? Give the recurrence explicitly.

3. (2 points) What is the running time of RANDOMIZED-QUICKSORT when all elements of array A have the same value? Explain.

4. (7 points) The PARTITION(A, p, r) procedure returns an index q such that each element of $A[p\,..\,q-1]$ is less than or equal to $A[q]$ and each element of $A[q+1\,..\,r]$ is greater than $A[q]$. Modify the PARTITION procedure to produce a procedure PARTITION'(A, p, r), which permutes the elements of $A[p\,..\,r]$ and returns two indices q and t, where $p \leq q \leq t \leq r$, such that

   - all elements of $A[q\,..\,t]$ are equal,
   - each element of $A[p\,..\,q-1]$ is less than $A[q]$, and
   - each element of $A[t+1\,..\,r]$ is greater than $A[q]$.

   Like PARTITION, your PARTITION' procedure should take $\Theta(r-p)$ time. Explain briefly how your PARTITION'(A, p, r) works.

5. (3 points) Give a loop invariant for PARTITION' that would allow one to prove its correctness. You don't have to prove correctness, just state the invariant.

6. (3 points) Modify the RANDOMIZED-QUICKSORT procedure to call PARTITION', and name the new procedure RANDOMIZED-QUICKSORT'. Then modify the QUICKSORT procedure to produce a procedure QUICKSORT'(A, p, r) that calls RANDOMIZED-PARTITION' and recurses only on partitions of elements not known to be equal to each other.

Exercise 3: decision tree for comparison sorts (15 points). Consider the SELECTION-SORT(A) algorithm (exercise 2.2-2 in the book), using the following pseudocode:

```
SELECTION-SORT(A)
1  n = A.length
2  for j = 1 to n − 1
3       smallest = j
4       for i = j + 1 to n
5            if A[i] < A[smallest]
6                 smallest = i
7       exchange A[j] with A[smallest]
```

1. (5 points) Draw the decision tree corresponding to SELECTION-SORT when running on an array of $n = 3$ elements $A = \langle a_1, a_2, a_3 \rangle$ as in fig. 8.1 (keep "$\leq$" on the left child and "$>$" on the right child).

2. (2 points) Mark the execution path followed for the array $A = \langle 6, 4, 2 \rangle$, as in fig. 8.1.

3. (2 points) For the tree you drew, what is the depth for the best and worst cases? Comment on the result.

4. (3 points) For a tree corresponding to an array with n elements, what would be the depth for the best and worst cases? Comment on the result.

5. (3 points) For the tree you drew, how many leaves does it have? Compare this with $n!$ and comment on the result.

Exercise 4: hash tables (27 points). Consider a hash table with $m = 11$ slots and using the hash function $h(k) = \lfloor m(kA - \lfloor kA \rfloor) \rfloor$ where $A = (\sqrt{5} - 1)/2$ and k is a natural number. Consider the keys $k = 3, 6, 14, 71, 74, 60, 51, 7, 1$ (in that order).

1. (3 points) Give $h(k)$ for each of those keys.

Now, consider inserting those keys in the order given above into the hash table. Show the final table in these two cases:

2. (12 points) Chaining using as hash function $h(k)$.

3. (12 points) Open addressing using linear probing and the same hash function $h(k)$.

Exercise 5: hash tables (10 points). We have a hash table T_1 that uses chaining to resolve collisions; it has m slots and contains currently n elements. We have another hash table T_2 that uses open addressing; it contains currently n elements and occupies exactly the same amount of memory as T_1. Now, we execute a search for a key k which is in neither table. On average, in which of the two tables does the search take fewer operations? *Hint*: assume that pointers and keys occupy each one word of memory, and that an unsuccessful search requires on average $1 + \alpha(T_1)$ operations for T_1 and $\frac{1}{1-\alpha(T_2)}$ operations for T_2, where α is the load factor.

Exercise 6: binary search trees (22 points).

1. (10 points) Starting from an empty binary search tree, draw the final tree resulting from the insertion of the following keys: $14, 10, 20, 6, 15, 2, 5, 30, 12, 9$ (in that order).

2. (2 points) Do the inorder tree walk, printing the resulting keys.

3. (10 points) Starting from the tree obtained in the former question, draw the tree resulting from the removal of the following keys: $10, 12$ (in that order).

Bonus exercise: bucket sort (20 points). (Exercise 8.4-4 in the book.) We are given n points in the unit circle, $p_i = (x_i, y_i)$, such that $0 < x_i^2 + y_i^2 \leq 1$ for $i = 1, 2, \ldots, n$. Suppose that the points are uniformly distributed; that is, the probability of finding a point in any region of the circle is proportional to the area of that region. Design an algorithm with an average-case running time of $\Theta(n)$ to sort the n points by their distances $d_i = \sqrt{x_i^2 + y_i^2}$ from the origin. *Hint*: design the bucket sizes in BUCKET-SORT to reflect the uniform distribution of the points in the unit circle.