CSE100 Algorithm Design and Analysis — Homework set #2
Fall semester 2020 — Miguel Á. Carreira-Perpiñán

Total possible marks: 100. This set covers chapters 6–12 of the textbook *Introduction to Algorithms*, 3rd. ed., by Cormen et al.

Exercise 1: heapsort (26 points). We consider HEAPSORT to sort an array in *decreasing* order by using min-heaps.

1. (3 points) State the min-heap property.

Then, using the same notation as in the textbook, write pseudocode for the following functions (where A is the min-heap):

2. (3 points) MIN-HEAPIFY(A, i), which assumes that the binary trees rooted at LEFT(i) and RIGHT(i) are min-heaps, but that $A[i]$ may be larger than its children. MIN-HEAPIFY lets the value of $A[i]$ float down so that the subtree rooted at i obeys the min-heap property.
3. (2 points) BUILD-MIN-HEAP(A), which builds a min-heap on the input array A (overwriting it).
4. (2 points) HEAPSORT(A), which sorts the array A in decreasing order, based on MIN-HEAPIFY and BUILD-MIN-HEAP.

Finally:

5. (10 points) State a loop invariant for HEAPSORT and use it to prove its correctness. Assume MIN-HEAPIFY and BUILD-MIN-HEAP are correct.
6. (6 points) Give the runtime for MIN-HEAPIFY, BUILD-MIN-HEAP and HEAPSORT as a function of the array size n. Explain your answers.

Exercise 2: decision tree for comparison sorts (15 points). Consider the BUBBLESORT(A) algorithm (problem 2-2 in the book), using the following pseudocode:

```
BUBBLESORT(A)
n = A.length
for i = 1 to n − 1
    for j = n downto i + 1
        if A[j] < A[j − 1]
            exchange A[j] with A[j − 1]
```

1. (5 points) Draw the decision tree corresponding to BUBBLESORT 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 3: hash tables (27 points). Consider a hash table with $m = 10$ slots and using the hash function $h(k) = (2k + (3 * A))\%m$ where $A = (\sqrt{9} - 1)/2$ and k is a natural number. Consider the keys $k = 4, 5, 15, 72, 84, 70, 41, 8, 2$ (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 4: counting sort (10 points). Write the pseudocode of an algorithm that, given an array A of n integers in $\{0, \ldots, k\}$, preprocesses A in $\Theta(n + k)$ time and then answers any query about how many integers fall into the range $[a, b]$ in $\Theta(1)$ time.

Exercise 5: 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: $15, 10, 13, 36, 25, 17, 16, 35, 12, 37$ (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.