Lista de Análisis y Diseño de Algoritmos (Sorting Algorithms)

Universidad de KENT

Parte 01, Sorting

1) Sort the following sequence using merge sort and quicksort.

22 80 18 9 90 12 22 57 86 36 32 88 20 6 62 22

For merge sort, show subsequences before and after merging. For quicksort, show the selected pivot element and the resulting partition.

2) For each of the following problems, give an algorithm that finds the desired numbers within the given amount of time.
   (a) Let S be an *unsorted* array of n integers. Give an algorithm that finds the pair $x, y \in S$ that *maximizes* $|x - y|$. Your algorithm must run in $O(n)$ worst-case time.
   (b) Let S be a *sorted* array of n integers. Give an algorithm that finds the pair $x, y \in S$ that *maximizes* $|x - y|$. Your algorithm must run in $O(1)$ worst-case time.
   (c) Let S be an *unsorted* array of n integers. Give an algorithm that finds the pair $x, y \in S$ that *minimizes* $|x - y|$, for $x \neq y$. Your algorithm must run in $O(n \log n)$ worst-case time.
   (d) Let S be a *sorted* array of n integers. Give an algorithm that finds the pair $x, y \in S$ that *minimizes* $|x - y|$, for $x \neq y$. Your algorithm must run in $O(n)$ worst-case time.

3) Give an efficient algorithm to compute the union of sets A and B, where $n = \max(|A|, |B|)$. fte output should be an array of distinct elements that form the union of the sets, such that they appear exactly once in the union. Assume that A and B are unsorted. Give an $O(n \log n)$ time algorithm for the problem.

Parte 02, Heaps

1) Sort the following sequence using heap sort.

22 80 18 9 90 12 22 57 86 36

Show each heap before extracting the maximum.

2) Describe an algorithm which deletes an item with index i from an n-element max-heap. Your algorithm should run in $O(\log n)$ time.

3) Show how to implement a first-in, first-out queue with a priority queue. Show how to implement a stack with a priority queue.

Parte 03, Lineal Time Sorting

1) Sort the following sequence using radix sort with base 10.

761	669	254	857	150	249	927	592
961	799	885	391	167	235	364	412

For each digit, show the array C used to count and order after sorting each digit.

2) Suppose that we have an array of n data records to sort and that the key of each record has the value 0 or 1. An algorithm for sorting such a set of records might possess some subset of the following three desirable characteristics:
   1. The algorithm runs in $O(n)$ time.
   2. The algorithm is stable.
   3. The algorithm sorts in place, using no more than a constant amount of storage space in addition to the original array.

   a) Give an algorithm that satisfies criteria 1 and 2 above.
   b) Give an algorithm that satisfies criteria 1 and 3 above.
   c) Give an algorithm that satisfies criteria 2 and 3 above.
   d) Can you use any of your sorting algorithms from parts (a)–(c) as the sorting method used in radix sort, so that radix sort sorts n records with b-bit keys in $O(bn)$ time? Explain how or why not.

3) You are given an array of integers, where different integers may have different numbers of digits, but the total number of digits over all the integers in the array is n. Show how to sort the array in $O(n)$ time.

Parte 04, Balanced Trees

1) Argue that since sorting n elements takes $\Omega(n \log n)$ time in the worst case in the comparison model, any comparison-based algorithm for constructing a binary search tree from an arbitrary list of n elements takes $\Omega(n \log n)$ time in the worst case.

2) You have given a sorted array A with n numbers. Describe a linear time algorithm that builds a balanced BST storing all elements in A.

3) What is the largest possible number of internal nodes in a red-black tree with black-height k? What is the smallest possible number?

Parte 05, Hashing

1) Insert the following numbers in an empty hash table of size 19.

$$22 \quad 80 \quad 18 \quad 9 \quad 90 \quad 12 \quad 22 \quad 57 \quad 86 \quad 36 \quad 32 \quad 88 \quad 20 \quad 6 \quad 62 \quad 24$$

   (a) Use chaining and the hash function $h(k) = k \bmod 19$.
   (b) Use open addressing with the hash function $h(k, i) = (k \cdot 181) + (i \cdot k \cdot 113) \bmod 19$.

2) You have given two arrays A and B. Create two lists D and S such that S contains all the singles (i. e., numbers only in A or only in B) and D contains all the doubles (i. e., numbers in A and in B). Your algorithm should run in linear time.

3) Your are given an array A of numbers and a number k. Note that A is not necessarily sorted. Find two numbers $x, y \in A$ such that $x + y = k$. Your algorithm should run in linear time.