MO417 – Complexidade de Algoritmos
Segundo Semestre de 2008
Quarta Lista de Exercícios

1 HeapSort

1. Where in a heap might the smallest element reside, assuming that all elements are distinct?
2. Is an array that is in sorted in decresing order a heap?
3. The code for HEAPIFY is quite efficient in terms of constant factors, except possibly for the recursive call, which might cause some compilers to produce inefficient code. Write an efficient HEAPIFY that uses an iterative control construct (a loop) instead of recursion.
4. Why do we want the loop of BUILD-HEAP to decrease from $\lfloor length[A]/2 \rfloor$ to 1 rather than increase from 1 to $\lfloor length[A]/2 \rfloor$?
5. What is the running time of heapsort on an array A of length n that is already sorted in increasing order? What about decreasing order?
6. Give an $\mathrm{O}(n \lg k)$-time algorithm to merge k sorted lists into one sorted list, where n is the total number of elements in all the input lists.
7. You are given a sequence of n elements to sort. The input sequence consists of n/k subsequences, each containing k elements. The elements in a given subsequence are all smaller than the elements in the succeeding subsequence and larger than the elements in the preceding subsequence. Thus, all that is needed to sort the whole sequence of length n is to sort the k elements in each of the n/k subsequences. Show an $\Omega(n \lg k)$ lower bound on the number of comparisons needed to solve this variant of the sorting problem.

2 QuickSort

8. What value of pivot does PARTITION return when all elements in the array A[begin..end] have the same value? Modify PARTITION so that pivot = (begin+end)/2 when all elements in the array A[begin..end] have the same value.
9. Suppose that the splits at every level of quicksort are in the proportion $(1 - \alpha)$ to α, where $0 < \alpha \leq 1/2$ is a constant. Show that the minimum depth of a leaf in the recursion tree is approximately $(-\lg n / \lg \alpha)$ and the maximum depth is approximately $(-\lg n / \lg (1 - \alpha))$. Don't worry about integer round-off.
10. The running time of quicksort can be improved in practice by taking advantage of the fast running time of insertion sort when its input is "nearly" sorted. When quicksort is called on a subarray with fewer than k elements, let it simply return without sorting the subarray. After the top-level call to quicksort returns, run insertion sort on the entire array to finish the sorting process. Argue that this sorting algorithm runs in O(nk + n lg(n/k)) expected time. How should k be picked, both in theory and in practice?

3 Ordenação em Tempo Linear

11. Describe an algorithm that, given n integers in the range 0 to k, preprocesses its input and then answers any query about how many of the n integers fall into a range $[a..b]$ in $\mathrm{O}(1)$ time. Your algorithm should use $\Theta(n + k)$ preprocessing time.

12. Show how to sort n integers in the range 0 to $n^2 - 1$ in $\mathrm{O}(n)$ time.

13. What is the worst-case running time for the bucket-sort algorithm? What simple change to the algorithm preserves its linear expected running time and makes its worst-case running time $\mathrm{O}(n \lg n)$?