Foundations of computing: Algorithms and Data Structures

Rasmus Pagh

October 23, 2011

Priority queues

1. Suppose you are given a list of n integers and a number k, and want to report the k smallest integers in the list in sorted order.

   (a) Give a solution that uses at most $c(n+k\log n)$ comparisons, for some constant c.

   (b) What is the time to sort all elements?

   (c) For what values of k is the time linear in n?

2. Imagine you are given a sequence of n numbers $x_1, \ldots, x_n$ and want to find *outliers*. This is an example of a *data mining* task. For some parameter k we define the ith number (x_i) as an outlier if $x_i \leq (x_{j+1} + \cdots + x_{j+k})/(2k)$, for some integer j such that $i \in \{j+1, \ldots, j+k\}$. A brute-force algorithm solves this in time proportional to nk. (How?)

   Unless k is small we can find such outliers more efficiently using a normal queue plus an addressable priority queue:

   (a) Put the k first numbers in the priority queue, and the corresponding handles in the queue.

   (b) Compute the sum of the k numbers in the queue.

   (c) See if the minimum of the priority queue is an outlier (remove it if it is, and repeat).

   (d) Put x_{k+1} in the queue and priority queue, remove x_1 in the same places (using the handle), and update the sum.

   (e) ... Iterate as in steps (c) and (d) to consider all intervals of length k.

   Analyze the worst-case running time of this algorithm when an addressable priority queue with logarithmic time per operation is used. How would you extend the algorithm to also keep track of large outliers (defined as being larger than twice the average)?