

Assignment 4

Due: Monday, October 6, 2008 (before class)

Note: The Midterm is Thursday, October 9, 6:30-7:30 pm, in RAWL 1086.

Exercise exam questions (not to be handed in) will be posted Friday, October 3.

1) (16 pts.)

(i) Consider the randomized algorithm for the hiring problem described in class (sections 5.1 and 5.2). If the candidates are presented in random order, we showed that the expected number of hires is $\ln n + O(1)$. What is the probability that exactly one hire is made? What is the probability that exactly n hires are made? Explain your answers.

(ii) Consider the deterministic linear time selection algorithm described in class. Assume that instead of $n/5$ groups of size 5, the algorithm forms $n/7$ groups of size 7. Set up the recurrence solution for this modification and show that it results in an $O(n)$ time algorithm.

2) (17 pts.) Joe Isitbetter has designed a data structure for an application that requires the operations $\text{insert}(x)$ and $\text{search}(x)$. The insertion does not need to check if the element is already present as the application does not generate duplicates. Joe wants to use an array structure that allows binary search. Having taken CS251, he realizes that inserting an element into an array of size n costs $O(n)$ time (ignore any cost related to growing the array size). He proposed the following hybrid data structure:

- At any point in time, the current elements are stored in an array A of size n and a linked list L of size at most $\lceil \log n \rceil$.
- The **search** operation is implemented by first doing a binary search on array A and, if the search is unsuccessful, a linear scan of list L .
- The **insert** operation is implemented as follows: Assume list L currently contains k elements. If $k < \lceil \log n \rceil$, insert element x to list L . If $k = \lceil \log n \rceil$, create a sorted sequence of size $n + \lceil \log n \rceil$ from the elements currently in A and in L , make it the new array A , and set the linked list L to empty. Assume that Joe creates the merged array as efficiently as possible.

Joe Isitbetter conjectures that over a sequence of operations this hybrid structure gives a better asymptotic performance compared to using only array A .

(i) Analyze the performance of one insert and search operation in the hybrid structure in terms of n and k .

(ii) Assume we start off with array A containing n elements and list L being empty. Next, a total of $5pn$ operations take place: pn elements are inserted and $4pn$ searches are executed (order is unknown). Counting the time needed for all operations, make a comparison (in terms of n and

p) of the hybrid structure to using only a sorted array. Explain your analysis and whether Joe's conjecture is correct.

3) (17 pts.) The sorting and selection algorithms seen in class assume that given two elements x_i and x_j we can decide whether $x_i < x_j$, $x_i = x_j$, or $x_i > x_j$. Assume we can only decide whether $x_i = x_j$ is true. This holds, for example, when working with files of different types for which ordering has no meaning or when an ordering would reveal information to be kept hidden.

Consider the operation `At-least(k)` which checks whether in a sequence of n elements there exists an element that occurs at least k times. If such an element exists, return it and its multiplicity (ties can be broken in an arbitrary way). If no such element exists, return "none". You can assume $n = 2^r$, if convenient. A check for equality costs $O(1)$ time.

(i) Consider the case when $1 \leq k \leq n/2$. Describe and analyze an algorithm for `At-least(k)` for k in this range. Explain your time bound in terms of n and possibly k .

(ii) Consider the case of $k = n/2 + 1$. Describe and analyze an algorithm for `At-least(k)` for this value of k . Your algorithm should have a worst case running time of $O(n \log n)$ or better.

(iii) Can one use the algorithm for (ii) to handle all values of $k > n/2 + 1$? Explain your answer.