

CSE3358 Problem Set 5
02/11/05
Due 02/18/05

Problem 1: Young tableau

An $m \times n$ Young tableau is an $m \times n$ matrix such that the entries of each row are in sorted order from left to right and the entries of each column are in sorted order from top to bottom. Some of the entries of a Young tableau may be ∞ , which we treat as nonexistent elements. Thus a Young tableau can be used to hold $r \leq mn$ numbers.

Here's an example of a 4x4 Young tableau containing the elements $\{9, 16, 3, 2, 4, 8, 5, 14, 12\}$. Note that this is not unique.

2	4	9	∞
3	8	16	∞
5	14	∞	∞
12	∞	∞	∞

- (a) (5 points) Argue that an $m \times n$ Young tableau Y is empty if $Y[1, 1] = \infty$. Argue that Y is full (contains mn elements) if $Y[m, n] < \infty$.
- (b) (5 points) Argue that the minimum element of a Young tableau Y is always in $Y[1, 1]$.
- (c) (10 points) Give an algorithm to implement EXTRACT-MIN on a nonempty $m \times n$ Young tableau that runs in $O(m + n)$ time. Your algorithm should use a recursive subroutine that solves an $m \times n$ problem by recursively solving either an $(m - 1) \times n$ or an $m \times (n - 1)$ subproblem. Define $T(p)$, where $p = m + n$, to be the maximum running time of EXTRACT-MIN on any $m \times n$ Young tableau. Give and solve a recurrence for $T(p)$ that yields the $O(m + n)$ time bound.
- (d) (10 points) Show how to insert a new element into a nonfull $m \times n$ Young tableau in $O(m + n)$ time.
- (d) (10 points) Using no other sorting method as subroutine, show how to use an $n \times n$ Young tableau to sort n^2 numbers in $O(n^3)$ time. Based on this, describe a sorting algorithm that has an $O(n^{1.5})$ worst-case running time.
- (e) (0 points) I will not ask: given n distinct elements, how many Young tableaux can you make?

Problem 2: Sometimes you just have to count

(10 points) 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 $O(1)$ time. Your algorithm should use $\Theta(n + k)$ preprocessing time. For full credit you should

- Describe your algorithm in english
- Provide a pseudocode
- Provide arguments for the correctness of your algorithm
- Provide arguments for the running times of the preprocessing and the query operations

Problem 3: Sorting revisited

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 properties:

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) (5 points) Describe an algorithm that possesses properties (1) and (2).

(b) (5 points) Describe an algorithm that possesses properties (2) and (3).

(c) (10 points) Describe an algorithm that possesses properties (1) and (3).

For full credit you should in each case:

- Describe your algorithm in english
- Provide a pseudocode, unless you are describing an algorithm for which we have seen the pseudocode in class
- Provide arguments that your algorithm possesses the desired properties