

Homework 3 (due before class, Monday, October 8, 2001)

1 Lower Bounds, Linear Time Sorting [Chapter 8 in CLRS]

1. [CLRS 8.1-3] Show that there is no comparison sort whose running time is linear for at least half of the $n!$ inputs of length n . What about a fraction of $\frac{1}{n}$ of the inputs of length n ? What about a fraction $\frac{1}{2^n}$?
2. [CLRS 8-1] Do this problem.
3. The sorting lower bound applies only if the input array can be any permutation; it does not apply if the input array is restricted in any way. Consider an array that is a concatenation of at most k increasing sequences. For example, the array [2, 4, 1, 7, 6, 5] is an array with 4 increasing sequences, namely [2, 4], [1, 7], [6], and [5].
 - Derive the lower bound $k^{n-k}k!$ for the minimum number of leaves required in any decision tree for sorting this array. You may want to require particular elements to occur in particular sequences.
 - Describe (and show the correctness of) an algorithm that requires $O(\lg(k^{n-k}k!))$ time to sort an array of this kind with n elements. Conclude that your algorithm runs in $\Theta(n \lg k)$ time.

2 Search trees [Chapter 13 in CLRS]

4. [CLRS 13-2] Do this problem.
 5. Consider a binary search tree in which each node v contains a key as well as an additional value called 'addend'. The addend of node v is implicitly added to all keys in the subtree rooted at v (including v). Let (key, addend) denote the contents of any node v .

Part 1:

Let h be the height of such a tree. Describe how to perform the following operations in $O(h)$ time:
- $Find(x, T)$: return YES if element x is stored in tree T
 - $Insert(x, T)$: insert element x in tree T
 - $Push(x, k, T)$: add k to all elements in tree T which are greater than or equal to x (x is not necessarily in T)

Part 2:

Describe how it can be insured that $h = O(\log n)$ during the above operations. (Note: You don't need to go to the rotation level of detail on this problem. What things can you show to prove that the rotations work?)