

CMPS 201 Fourth Homework, Fall 04
3 problems, 30 pts, Due Tuesday, October 26

1. (12 pts) For this problem you will show that using groups of three does *not* lead to a linear time median-of-medians algorithm.

Recall the median of median algorithm presented in class. Consider a modified algorithm that divides the input is divided into groups of 3 rather than groups of 5. Let $f(n)$ be the number of comparisons done by this groups-of-three algorithm assuming that all partitions are maximally un-even (with the pivot element being the median of medians), and that the desired key is the sub-array with as many keys as possible. Analyze the groups-of-three algorithm to obtain a recurrence relation $T(n)$ that *lower bounds* $f(n)$ (i.e. the value defined by your recurrence should be an underestimate of $f(n)$). Include floors and ceilings in your $T(n)$ recurrence and for each term in the recurrence indicate what part of the algorithm it corresponds to.

Next prove that the function defined by your recurrence is in $\omega(n)$ (i.e. grows faster than linearly). This can be done by using an induction to prove that that the solution to $T(n)$ is at least some function $g(n) \in \omega(n)$ (consider using $g(n) = cn \log n$). Although not mathematically correct, you may drop the floors and/or ceilings in the induction to simplify the algebra. You may also assume that the algorithm uses heap sort when n is small ($n < 40$), so the recurrence relation is only used for large n .

2. (6 pts) Consider the problem of storing n different positive integer keys and an associated data pointer for each key in a dictionary (perhaps the keys are student IDs and the associated data is a pointer to the students record).
 - a. First, assume that all n keys (and their pointers) are available ahead of time (so you can do whatever preprocessing you want) and that there will be no insertions or deletions. Describe a way to store the keys and pointers so that the worst case lookup time is $O(\log n)$ and no extra storage is needed.
 - b. Next, assume you implement a hash table with m slots and chaining to hold the keys and data. How many additional pointers are required (including null pointers)?
 - c. Now assume we want to implement the hash table using open addressing. Assuming uniform hashing, how big does the table size have to be so that the expected number of probes for an unsuccessful lookup is at most $\lg n$?
3. (12 pts) Prove that at least $\lceil \frac{3n}{2} \rceil - 2$ comparisons are required to find both the maximum and minimum of n elements. (Hint: use an adversary argument which keeps track of four sets of elements – those that can be the largest, those that can be the smallest, those that can be both, and those that can be neither.)