CIS 575. Introduction to Algorithm Analysis
Material for March 1, 2024

Priority Queues and Their Implementation

The topic of this note is covered in *Cormen*'s Section 6.5.

1 Priority Queues

It is often convenient to maintain a *priority queue* which is a collection of records, each having a (not necessarily unique) **key**; the queue is interactive in that one may add records, or select (and also remove) the record with the highest **priority** (as determined by the key).

Operations An implementation of priority queues must offer the operations

INSERT which adds a record

FINDMAX which returns a record with maximum key (several such may exist)

DELETEMAX which removes a record with the maximum key.

We have assumed that records with high keys are more important than records with low keys; in a symmetric way, we can consider records with low keys to be the most important.

Implementations Two naive approaches come to mind; we shall now describe them and also analyze their running times (as a function of n, the current number of records):

Arrays (unsorted) where we insert a new record at the first available spot; thus INSERT runs in constant time but FINDMAX requires a linear search and thus in the worst case runs in time $\Theta(n)$. We could make FINDMAX run in constant time by keeping track of the location of a maximal element, but that information would need to be recomputed after DELETEMAX which would then always run in linear time.

Sorted Arrays where we have the largest key at the end of the array; thus FINDMAX and DELETEMAX both run in constant time but INSERT will in the worst case run in time $\Theta(n)$: for even though we can use binary search to find (in time $\Theta(\lg n)$) the proper location to put the new record, all records with larger keys will have to be shifted.

We have seen implementations where at least one operation has a worst-case running time that is linear in the number of records. We shall go for something better, where all operations have worst-case running times that are (at most) **logarithmic**.