91.404 Homework 8 Solution

1. Exercise 9.3-5 (page 223) (20 points)

We assume that we are given a procedure MEDIAN that takes as parameters an array A and subarray indices p and r, and returns the value of the median element of $A[p\,..\,r]$ in $O(n)$ time in the worst case. Given MEDIAN, here is a linear-time algorithm SELECT_ for finding the i th smallest element in $A[p\,..\,r\,]$. This algorithm uses the deterministic PARTITION algorithm that was modified to take an element to partition around as an input parameter.

```
SELECT_(A, p, r, i )
if p = r
   then return A[p]
x ←MEDIAN(A, p, r )
q ←PARTITION(x)
k ← q − p + 1
if i = k
   return A[q]
elseif i < k
    return SELECT_(A, p, q − 1, i )
else return SELECT_(A, q + 1, r, i − k)
```

Because x is the median of $A[p\,..\,r\,]$, each of the subarrays $A[p\,..\,q-1]$ and $A[q+1\,..\,r\,]$ has at most half the number of elements of $A[p\,..\,r\,]$. The recurrence for the worst-case running time of SELECT_ is $T(n) \le T(n/2) + O(n) = O(n)$, based on Master Theorem.

2. Problem 9-1, page 224. (30 points)

We assume that the numbers start out in an array.

a. Sort the numbers using merge sort or heapsort, which take $\Theta(n \lg n)$ worst-case time. (Don't use quicksort or insertion sort, which can take $\Theta(n^2)$ time.) Put the i largest elements (directly accessible in the sorted array) into the output array, taking $\Theta(i)$ time.

The total worst-case running time is: $\Theta(n \lg n + i) = \Theta(n \lg n)$ (because $i \le n$).

b. Implement the priority queue as a heap. Build the heap using BUILD-HEAP, which takes $\Theta(n)$ time, then call HEAP-EXTRACT-MAX i times to get the i largest elements, in $\Theta(i \lg n)$ worst-case time, and store them in reverse order of extraction in the output array. The worst-case extraction time is $\Theta(i \lg n)$ because i extractions from a heap with $O(n)$ elements takes $i \cdot O(\lg n) = O(i \lg n)$ time, and half of the i extractions are from a heap with $\ge n/2$ elements, so those $i/2$ extractions take $(i/2)\Omega(\lg(n/2)) = \Omega(i \lg n)$ time in the worst case.
Total worst-case running time: $\Theta(n + i \lg n)$.

c. Use the SELECT algorithm of Section 9.3 to find the i th largest number in $\Theta(n)$ time. Partition around that number in $\Theta(n)$ time. Sort the i largest numbers in $\Theta(i \lg i)$ worst-case time (with merge sort or heapsort).

Total worst-case running time: $\Theta(n + i \lg i)$.
Note that method (c) is always asymptotically at least as good as the other two methods, and that method (b) is asymptotically at least as good as (a). (Comparing (c) to (b) is easy, but it is less obvious how to compare (c) and (b) to (a). (c) and (b) are asymptotically at least as good as (a) because n, $i \lg i$, and $i \lg n$ are all $O(n \lg n)$. The sum of two things that are $O(n \lg n)$ is also $O(n \lg n)$.)

3. Consider inserting keys 3,4,2,5,1 in the order given into a hash table of length m= 5 using hash function $h(k) = k^2 \bmod m$. (20 points)

a) Using $h(k)$ as the primary hash function, illustrate the result of inserting these keys using open addressing with linear probing.

h(3)=4
h(4)=1
h(2)=4
h(5)=0
h(1)=1

m	k
1	4
2	5
3	1
4	3
0	2

b) Using $h(k)$ as the primary hash function, illustrate the result of inserting these keys using open addressing with quadratic probing, where $c1=1$ and $c2=2$. $h(k,i)=(h(k)+i+2i^2) \bmod m$

m	k
1	4
2	2
4	3
0	5

c) Using $h(k)$ as the hash function, illustrate the result of inserting these keys using chaining. Compute the load factor α for the hash table resulting from the insertions.

m	K	
0	5	
1	4	1
2		
3		
4	3	2

load factor α=5/5=1

d) What different values can the hash function $h(k) = k^2 \bmod m$ produce when m= 11? Carefully justify your answer in detail.

The hash function can produce 6 different values: 0,1,3,4,5,9.
For any integer k, it can be written as $k = 11n + m$, $m = 0,1,2,...,10$.
Consider k^2, $k^2 = (11n)^2 + 22nm + m^2$. Obviously remainder can only come from m^2. Since m = 0,1,2,...,10, from m^2 mod 11, we can only get possible remainders 0,1,3,4,5,9.