Midterm review
Announcements

Midterm:
Open: book/notes/e-book/e-notes
Closed: friends/internet/”others”

Programming assignment 1 up now
Q 1

Suppose we had a sequence, S, of numbers in the range 1-100

How could we find the mode of S?

If we didn't know the range, how could we find the mode with using o(|S|) memory
A 1

Know range: Use first part of counting sort, find max $O(n)$ runtime

Unknown range: sort using $n \lg(n)$
sort which takes $O(1)$ memory
(quicksort or heapsort)
Find longest sequence of repeated numbers. $O(n \lg n + n) = O(n \lg n)$
Alice has a list with a billion entries. All her entries are in the range 1-100. She needs to remove all duplicates, in her original list. What should she do?
Create an array of length 100 to store boolean values of whether or not a value has been seen.

Go through the list, for value v, if a[v] true: discard v. If a[v] false: set a[v] true and keep v.
Q 3

What is the point of zip codes?
A 3

Allows the postal system to do bin-sorting (based on the zip-code)
Q 4

What is the running time (avg, worst) for the following algorithm? (Different than normal quicksort?)

Quick-mod(A,p,r)
q = Select(A,p,r,|A|/2) // median
partition(A,p,r) with pivot q
Quick-mod(A,p,q-1)
Quick-mod(A,q,r)
2 subproblems, half the size...
selection takes $O(n)$ avg and worst...
$T(n) = 2 \cdot T(n/2) + O(n)$

$a=b=2$, $n$ is $\Theta(n^{\log_b a})$
so by Master's theorem:
$O(n \log n)$ in worst and average
Differs as QS has $O(n^2)$ worst case
Heapify this array (max heap):

[1, 5, 2, 6, 10, 5, 92, -1, 4]
A 5

Using build-max-heap:

[92, 10, 5, 6, 5, 1, 2, -1, 4]
How can you build a heap using a recursive divide and conquer method?

(D&C-build-heap(A,i), builds heap rooted at A[i])

Does this effect the runtime or memory requirements?
D&C-build-heap(A,i)
if no children: do nothing
else
   D&C-build-heap(A,left(i))
   D&C-build-heap(A,right(i))
   m = arg max \{i, left(i), right(i)\}
   swap i and m
if i \neq m: D&C-build-heap(A,m)
A 6

Runtime: (Master's theorem again)

\[ T(n) = 3 \ T(n/2) + O(1) \text{ (single core)} \]
\[ T(n) = 3 \ T(n/2) + O(n) \text{ (multi core)} \]

\( a=3, \ b=2, \ n \text{ is } o(n^{\log_b a}) \)

so by Master's theorem, worst case:
\[ O(n^{\log_2 3}) = O(n^{1.58}) \]