

CS432, Algorithms

Names:

Some problems for Chap. 2 lectures

Answer the questions below before the end of class and turn it in. The results you give will not count toward your grade in the class. The purpose of this test is for me to get a feel for how well the class overall understands this material, **and** for **you** to see how well you understand this material.

- 1) Set up the formula for calculating the average case complexity for Sequential Search (AKA Linear Search) on an unsorted array. Assume that the "target" element is definitely in the array, but that there is a 50% chance it is in the first position (and equally likely in all other positions). Just set up the formula - you don't have to solve it.
For position 1, we do one comparison so $T(i) = 1$. $P(i) = 0.5$.
For each position i from 2 to n , we $T(i) = i$ and $P(i)$ is $0.5/(n-1)$.

$$\text{So } A(n) = 1 \times 0.5 + \text{Sum}_{\{i=2 \text{ to } n\}} (0.5 \times i / (n-1))$$

- 2) Argue that one cannot find the minimum element in an unsorted array of size n without doing at least $n-1$ comparisons.

To find the minimum you must show that the other $n-1$ items are not the minimum, which implies that they win when compared to some other item. One comparison produces exactly one winner, so you need $n-1$ comparisons.

- 3) Selection sort works by finding the largest element in an array from position 0 to position $n-1$, moving it to the end of the array, and then repeating this process with the sub-array from 0 to $n-2$, etc.

(A) Given what you know about FindMax, how many comparisons does selection sort do? $\text{Sum}_{\{i=1 \text{ to } n-1\}} (n-i) = \text{Sum}\{i=1 \text{ to } n-1\} i = n(n-1)/2$

(B) FindMax is optimal: does this imply selection sort is optimal? **No!**

- 4) Show that any polynomial grows more slowly than any exponential (see the slide) using the method shown in class using limits and L'Hopital's rule.

This is problem O5 of HW1. See solution posted for that. (Doh! I didn't mean to put this hard a problem on this quiz! I meant you to compare $\log n$ to n here.)

- 5) How often is x incremented in the following code?

```
for i = 1 to n
  for j = i downto 1
    x = x + 1
Sum_{i=1 to n} (i) = n(n+1)/2
```

- 6) How often is x incremented in the following code?

```
j = n
while (j >= 1) {
  for i = 1 to j
    x = x + 1
  j = j / 2
}
```

See page 49-50 in the textbook. If n is a power of 2, this is just $2(n-1)$