Midterm Review Session: :)

3. $n^{1/2} + n^{1/3}$

2. poly-time

$n \log n$

n^2

$(n+4)(n-6)$ $[O(n^2)]$

n^{2^6}

$\sqrt{n} = n^{1/2}$

exp-time

$2^n \log n$

$n!$

$2^{n/2}$

2^{n^6}

$(n-2)!$

$n^{1/2}$, $n \log n$, $n^2 + 8n\log n \approx (n+4)(n-6)$, n^{2^6}

$2^{n/2}$ $2^n \log n$ 2^{n^6} $(n-2)!$ $n!$

$\cdot (n-1) \cdot n$

$\cdot \approx n^2$

A1: $n^2 + 8n\log n$

$(n-2)!$, $n!$

A2: $(n+4)(n-6) = n^2 - 6n + 4n - 24$

$$\frac{n. \quad n\log n}{n^2. \quad n\log^{50}(n)}$$

$$n! = (n-2)! \cdot \underbrace{n \cdot n-1}_{n^2}$$

$(n-5)!$, $(n-6)!$

$$\lim_{n\to\infty} \frac{(n-5)!}{(n-6)!} = \frac{(n-5)\cdot\cdot\cancel{(n-6)}\cdot\cdots 1}{\cancel{(n-6)}\cdot\cancel{(n-7)}\cdots\cdots 1}$$

$$\lim_{n\to\infty} (n-5)$$

$$2^n \quad \cdot \quad 2\cdot\cdots 2$$

$\underbrace{\qquad}_{n}$

$n! = n \cdot n-1 \cdots n-2 \cdots$

n^n exp or poly?

n^c constant polynomial

11.

A n integers not sorted

A_s n integers sorted increasing order

A_h n integers min-heap.

	X in A?	X occurs at least $3/4^n$ times?	X < smallest element in array?
A (not sorted)	$O(n)$	$O(n)$	$O(n)$ (lin. time selection)
A_s (sorted)	$O(\log n)$	$O(\log n)$	$O(1)$
A_h (min-heap)	$O(n)$	$\Theta(n)$	$O(1)$

14.

A array containing n distinct integers

input: i, j. $0 < i < j \leq n$

output: all elements of A with ranks from i to j

$A = [41, 20, 10, 34, 98, 64, 72]$

$A_{sorted} = [10, 20, \overset{i=3}{34}, 41, \overset{j=5}{64}, 72, 98]$

$i=3 \quad j=5$

$\Rightarrow 34, 41, 64$

sol. 1

Sort $\rightarrow$ go to indices & output in between

$O(n\log n)$

Sol. 2 $O(n)$

1. Run Lin. time selection to find element w/ rank i. $O(n)$

2. Run Lin time selection to find element w/ rank j. $O(n)$

3. Scan through array and check each entry to be greater than $A[i]$ & less than $A[j]$ $O(n)$

$\Rightarrow$ Runtime $O(n)$

Correctness:

From the correctness of LTS (shown in class) Step 1 gives us the i^{th} rank element of A.

" j^{th} rank element.

Now we know that elements with rank k for $i \leq k \leq j$ will have value greater than

i^{th} rank element and less than j^{th} rank element.

Our scan only outputs these elements.

Runtime:

3 steps alg.

Step 1: We run Linear time selection once. From the course, we know this runs in $O(n)$ time.

Step 2: Same as step 1. Again, we run LTS once, resulting in runtime $O(n)$.

Step 3: We perform one scan of the array and to check at each point in the array is a constant time ($O(1)$) check.

Summing over the runtimes of our steps, gives us

$O(n) + O(n) + O(n) \Rightarrow O(n)$ total runtime.