Connor Steware

# COMP 3804 Assignment 1:

① $\log(\log(n)) < \frac{\log(n)}{\log(\log n)} < \log n < (\log n)^3 < n^{2/3} < 2^{\log n} < n^{3/2} < 2^n < n! < n^n$

Smallest → Largest

② $\log(n!) = \log(1 \cdot 2 \cdot 3 \cdots n) = \log(1) + \log(2) + \log(3) + \ldots + \log(n-3) + \log(n-2) + \log(n-1) + \log(n)$

$\rightarrow \leq \log(n) + \log(n) \ldots \log(n) = n\log(n)$

to understand the above:

- $\log(n!)$ = the result of the factorial of all n's logged
- This results in: $\log(1) + \log(2) \ldots \log(n)$
- The upperbound of n minus an Integer is always n, thus the upperbound of the factorial is the $\log(n)$ added to itself n times
- Multiplications can be thought of as repeated additions, we can represent $\log(n)$ added to itself n times as $n\log(n)$

$\therefore \log(n!) \leq O(n\log(n))$ for all $n > 0$

$\log(n!) = \log(1 \cdot 2 \cdot 3 \cdots n) = \log(1) + \log(2) + \log(3) + \ldots + \log(n-2) + \log(n-1) + \log(n)$

$= \log(1) + \log(2) + \log(3) + \ldots + \log(n/2) + \ldots + \log(n-2) + \log(n-1) + \log(n)$

$\geq \log(n/2) + \log(\frac{n+1}{2}) + \ldots + \log(n-1) + \log(n) = n/2 \log(n/2)$

thus $\log(n!) \geq \frac{n}{2}\log(\frac{n}{2})$ by replacing all terms w/ their lower bounds

$\log(n!) \geq \frac{n}{2}\log(\frac{n}{2})$

$= \frac{n}{2}(\log(n) - \log(2))$ [log Rule: $\log_2 \frac{M}{N} = \log_2 M - \log_2 N$]

$= \frac{n}{2}(\log(n) - 1)$

$= \frac{n}{2}\log(n) - \frac{n}{2}$ → Prove: $\frac{n\log n - n}{2} \geq c n \log n$

This proof is omitted for clarity:

For $n \geq 4$:

$2 \leq \log n$ [$\log_2(4) = 2$]

$\frac{1}{2} \leq \frac{\log n}{4}$ [divide by 4]

$\frac{n}{4} \leq \frac{n\log n}{4}$ [multiply by n]

$0 \leq \frac{n\log n - 2n}{4}$ [Add $\frac{n}{4}$]

$\frac{n\log n}{4} \leq \frac{n\log n - n}{2}$ [add $\frac{n\log n}{4}$]

$c = 1/4$

thus we can see: $n\log n \geq \log(n!) \geq \frac{n\log n}{4}$

Proves: $\frac{n\log n - n}{2} \geq \frac{n\log n}{4}$

I'm showing $\frac{n\log n - n}{2}$ is greater than a multiple of $n\log n$:

thus: $c n\log n \leq \frac{n\log n - n}{2}$

Since: $\frac{n\log n - n}{2} < \frac{n\log n}{2}$, a multiple of $\frac{1}{4}$ is chosen to allow: $\frac{n\log n - n}{2} \geq \frac{n\log n}{4}$

Base Case: $n = 1$ (factorial can't have $n = 0$)

$\log(n!) = \Theta(n\log n)$

$\log(1!) = \Theta((1)\log(1))$

$\log(1) = \Theta(\log(1))$

$0 = \Theta(0)$ ∴ the base case is true

$\therefore$

$\log(n!) \geq \frac{n}{2}\log\left(\frac{n}{2}\right)$ [from earlier]

$= \frac{n}{2}(\log n - \log 2)$

$= \frac{n}{2}(\log n - 1)$

$= \frac{n\log n - n}{2}$ [Algebra]

$\frac{n\log n - n}{2} \geq \frac{n\log n}{4}$ [from limit Comparison]

$\frac{n\log n}{4} \Rightarrow \Omega(n\log n)$ [Drop Constants]

thus $\log(n!) = \Omega(n\log n)$

Since $\log(n!) = O(n\log n)$ & $\Omega(n\log n)$

$\log(n!) = \Theta(n\log n)$

(3)

Proove: $\sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1}$

Base Step:

Let $n=1$:

$\frac{1}{i(i+1)} = \frac{n}{n+1} \rightarrow \frac{1}{1+1} = \frac{1}{1+1} \Rightarrow \frac{1}{2} = \frac{1}{2}$

Assume True for $f(n-1)$: when $n>1$ $f(n) = \sum_{i=1}^{n} \frac{1}{i(i+1)}$

$$\sum_{i=1}^{n} \frac{1}{(i+1)(i)} = \frac{1}{2} + \frac{1}{6} + \frac{1}{12} + \ldots + \frac{1}{(n-2)(n-1)} + \frac{1}{(n-1)(n)} + \frac{1}{n(n+1)}$$

$$\left[\text{Value of } n-1\right] \leftarrow = \sum_{i=1}^{n-1} \frac{1}{(i+1)(i)} + \frac{1}{n(n+1)} \rightarrow \left[\text{Value of } n \text{ in the Sum}\right]$$

Inductive Hypothesis: for any value less then $n$, assume true for $n>1$:

$$\frac{n-1}{n} = \sum_{i=1}^{n-1} \frac{1}{(i+1)(i)}$$

thus: $\frac{n-1}{n} + \frac{1}{n(n+1)} = \frac{n-1(n+1)+1}{n(n+1)} = \frac{n^2+n-n-1+1}{n(n+1)} = \boxed{\frac{n}{n+1}}$

$\therefore$ Its true that $\sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1}$ for all $n \geq 1$

(4) a) $2n^2-5n-12 = O(n^3)$

Show: $2n^2-5n-12 \leq cn^3$, $\forall n > d$ where $c>0$ & $d$

$2n^2-5n-12 \leq 2n^3 \rightarrow \forall n > 1$

$\therefore c=2$ & $d=1$ its true

$\therefore$ its (true) that $O(n^3)$ is big Oh for $2n^2-5n-12$

b) $3n^2\log n - 2n\log n = \Omega(n^2)$

$3n^2\log n - 2n\log n = cn^2$

$3\log n - \frac{2\log n}{n} \geq c$ [definition of $\Omega$]

$\lim_{n\to\infty} \frac{3\log n}{1} - \lim_{n\to\infty} \frac{2\log n}{n} = \infty - \lim_{n\to\infty} \frac{2\log n}{n} \overset{H}{=} \infty - \lim_{n\to\infty} \frac{1/n\ln(2)}{1}$

$= \infty - \lim_{n\to\infty} \frac{1}{n\ln(2)} = \infty - \frac{1}{\infty} = \infty > c=1 \therefore \lim_{n\to\infty} \frac{f(x)}{g(x)} > 0$

$\therefore$ Its (true) that $\Omega(n^2)$ is the lower bound for $f(n)$

c) $2^n = \Omega(2^{n+1})$

Must Show that:

$2^n \geq c2^{n+1} \quad \forall n \geq d$

$\frac{2^n}{2^n} \geq c\frac{2^{n+1}}{2^n} \quad \forall n \rightarrow 1$ $\left[2^{n+1} = 2^n 2^1 = 2(2^n)\right]$

$1 \geq 2c \rightarrow \frac{1}{2} \geq c$

$\therefore c=\frac{1}{2}$ & $d=1$ its true

$\therefore$ Its (True) that $\Omega(2^{n+1})$ is big-omega for $2^n$

d) $n\log n = O(n^2)$ $\lim_{n\to\infty} \frac{f(x)}{g(x)} < \infty$ $\frac{n\log n}{n^2} \leq c$

Must Show that:

$\lim_{n\to\infty} \frac{n\log n}{n^2} \overset{H}{=} \lim_{n\to\infty} \frac{(\log(n)+1)/\log(2)}{2n} = \lim_{n\to\infty} \frac{\log n + 1}{\log(2)2n} \overset{H}{=} \lim_{n\to\infty} \frac{1/n}{2\log(2)} = \lim_{n\to\infty} \frac{1}{n2\log(2)}$

$= 0 < c=1$

$0 < \infty \therefore$ True

Its (true) that $O(n^2)$ dominates $n\log n$.

e) $n^2\log n = \Omega(n^3)$ $\therefore \frac{n^2\log n}{n^3} \geq c$

$$\lim_{n\to\infty} \frac{n^2\log n}{n^3} = \lim_{n\to\infty} \frac{\log(n)}{n} \overset{H}{=} \lim_{n\to\infty} \frac{1/n\log(2)}{1} = \lim_{n\to\infty} \frac{1}{n\log(2)}$$

$$= \frac{1}{\infty} = \boxed{0} > c = 1$$

$0 < \infty$ ∴ False as $0 < c \leq \infty$

∴ Its **False** that $\Omega(n^3)$ dominates $n^2\log n$

⑤ a)

$$T(n) \leq \begin{cases} 1 & \text{if } n=1 \\ 1+2T(\frac{n}{3}) & \text{if } n\geq 3 \end{cases} \quad a=2, b=3, d=0 \quad a\geq 1,\ b>1,\ d\geq 0$$

∴ **Use** Masters Theorm

$\log_3(2) \geq 0$ $T(n) = \boxed{O(n^{\log_3(2)})}$

b)

$$T(n) \leq \begin{cases} 1 & \text{if } n=2 \\ 1+T(\sqrt{n}) & \text{if } n\geq 4 \end{cases} \quad a=1, b=1, d=0 \quad a\geq 1,\ b\geq 1,\ d\geq 0$$

∴ **do NOT use Masters**

$T(n) = 1 + T(\sqrt{n})$

Assume $n$ is a Power of a Power 2 : $n = 2^{2^K}$ & $K = \log\log n$

$\sqrt{n} = n^{1/2} = (2^{2^K})^{1/2} = 2^{\frac{2^K}{2}} = 2^{2^{K-1}}$ ; for $n>1$

thus: $\sqrt{n} = 2^{2^{K-1}}$ can be used in the unfoldment

thus:

$T(n) = 1 + T(\sqrt{2^{2^K}}) = T(2^{2^{K-1}}) + 1$

↳ now: let $V = 2^K$; $V \in \mathbb{R}$

$n^{1/4} = (2^{2^K})^{1/4} = 2^{(\frac{2^K}{4})} = 2^{(2^K \cdot 2^{-2})} = \boxed{2^{2^{K-2}}}$

we can say:

$n^{1/V} = (2^{2^K})^{1/V} = 2^{(\frac{2^K}{V})} = 2^{(2^K \cdot 2^{-V})} = 2^{2^{K-V}}$

this describes the recurence. V is defined in here: it equal to $V = 2^K$; $V \in \mathbb{R}$

thus:

$T(n) = 1 + T(\sqrt{2^{2^K}}) = T(2^{2^{K-1}}) + 1 = T(2^{2^{K-2}}) + 2$

$= (1 + T(2^{2^{K-3}}) + 3) = \ldots = (K + T(2^{2^0}))$

$= T(2) + K$

∴ $O(\log\log n)$

then:

$T(2) = T(\sqrt{2}) + 1 = T(2) + 1 = 2$ ∴ $T(2^{2^K}) = K+2$

thus if $T(2^{2^K}) = K+2$ & & $n$ is a power of a power of two then:

$2^{2^K} = n$ & $K = \log\log n$ thus:

$T(2^{2^K}) = T(n)$ ∴ $T(n) = (\log\log n) + 2$ when $n>1$

$T(n) = O(\log\log n) + 2$

c) $T(1) = 1$

$T(n) = 5T(\frac{n}{7}) + n$ for $n \geq 2$

Assume $n = 7^k$ for some $k > 0$ [Assignment Specifications]

$$T(n) \leq \begin{cases} 1 & \text{if } n=1 \\ 5T(\frac{n}{7})+n & \text{if } n \geq 2 \end{cases} \quad a=5, b=7, d=1$$

$a \geq 1, b > 1, d \geq 0$

$\therefore$ Use Masters Theorm

- $d \leftrightarrow \log_b a$
- $1 > \log_7(5)$
- $1 > \frac{\log_{10} 5}{\log_{10} 7} \Rightarrow 1 > \sim 0.8271$

thus: $d > \log_b a \quad \therefore \boxed{T(n) = O(n^d) = O(n)}$

the big oh of the given recurrence is $O(n)$.

d)

$$T(n) = \begin{cases} 1 & \text{if } n=1 \\ T(n-1)+2n & \text{if } n \geq 2 \end{cases} \quad a=1, b=0, d=2$$

$a \geq 1, b \not> 1, d \geq 2$

$\therefore$ Do not use Masters Theorm

$T(n) = T(n-1) + 2n$

$T(n-1) = T(n-2) + 2(n-1)$

$T(n-2) = T(n-3) + 2(n-2)$

$T(n) = T(n-1) + 2n$

$= T(n-2) + 2(n-1) + 2n = T(n-2) + 4n - 2$

$= T(n-3) + 2(n-2) + 2(n-1) + 2n =$

$= T(n-4) + 2(n-3) + 2(n-2) + 2(n-1) + 2n$

if we substitute n for n-1 then: index value:

| | | |
|---|---|---|
| $T(n-1) = T(n-2) + 2(n-1)$ | 2 | increase by 1 element, $\therefore$ theirs a recurrence |
| $T(n-2) = T(n-3) + 2(n-2)$ | 3 | |
| $T(n-3) = T(n-4) + 2(n-3)$ | 4 | |
| $T(n-4) = T(n-5) + 2(n-4)$ | 5 | |

the recurrence can be expressed as follows: for $k > 1$

$T(n) = T(n-k) + 2(n-k+1) + 2(n-k+2) + \ldots + 2n - k$

finding Base case w/ $T(n-k)$; let $n=k$ such that $T(0)$, Solve for $k$

$T(n) = T(n-n) + 2(n-n+1) + \ldots + 2n - n$

$= (0 + 2 + 4 + 6 + \ldots + 2n) - n$

$= \sum_{i=0}^{n} 2i - n$ [Convert $0+2+4+\ldots+2n$ to a Summation]

$= \frac{2n(n+1)}{2} - n = n^2 + n - n = \boxed{O(n^2)}$

(6) a) this problem can be modeled as a recurrence relation where $a=5, b=2, \& d=1$ such that $5T(\frac{n}{2})+n$ represents the recurrence relation.

$$\begin{cases} 5T(\frac{n}{2})+n & \text{if } n \geq 0 \end{cases}$$

$a=5, b=2, d=1$ & $a \geq 1, b > 1, d \geq 1$

∴ use Masters Theorm

$d \rightarrow \log_b a$

$d < \log_2 5$

$1 < \log_2 5$

$1 < \frac{\log_{10} 5}{\log_{10} 2} \Rightarrow 1 < \approx 2.322 \quad \therefore T(n) = O(n^{\log_b a})$

$T(n) = O(n^{\log_2(5)})$

b) This can be represented as a recursive function where $2T(n-2)+2$ represents the recurrence relation.

$b=1$ ∴ Masters Theorm can't be used, as $b \not> 1$

To Solve via. Algebra:

$T(n) = 2T(n-1)+2$

$T(n-1) = 2T(n-2)+2$

$T(n-2) = 2T(n-3)+2$

$T(n-3) = 2T(n-4)+2$

$T(n) = 2T(n-1)+2$

$2(2T(n-2))+2+2$

$2(2(2T(n-3)))+2+2+2$

if we substitute for $n \rightarrow n-1$ then:

| | index value: | |
|---|---|---|
| $T(n-1) = 2T(n-2)+2$ | 2 | increase by 1 element ∴ recurence |
| $T(n-2) = 2T(n-3)+2$ | 3 | |
| $T(n-3) = 2T(n-4)+2$ | 4 | |

Thus the recurence can be expressed as follows: for $k \geq 1$ & $r > 1$

$T(n) = 2^r T(n-k) + \underbrace{2+2+2+\ldots+2}_{\text{Add } n \text{ many times}}$

$= 2^r T(n-k) + 2\sum_{i=0}^{n-1} 2^i = 2^r T(n-k) + 2(2^k-1)$

finding base case w/ $T(n-k)$: see $n=k$ such that $T(0)$, Solve for $k$

$T(n) = 2^n T(n-n) + 2(2^n-1)$

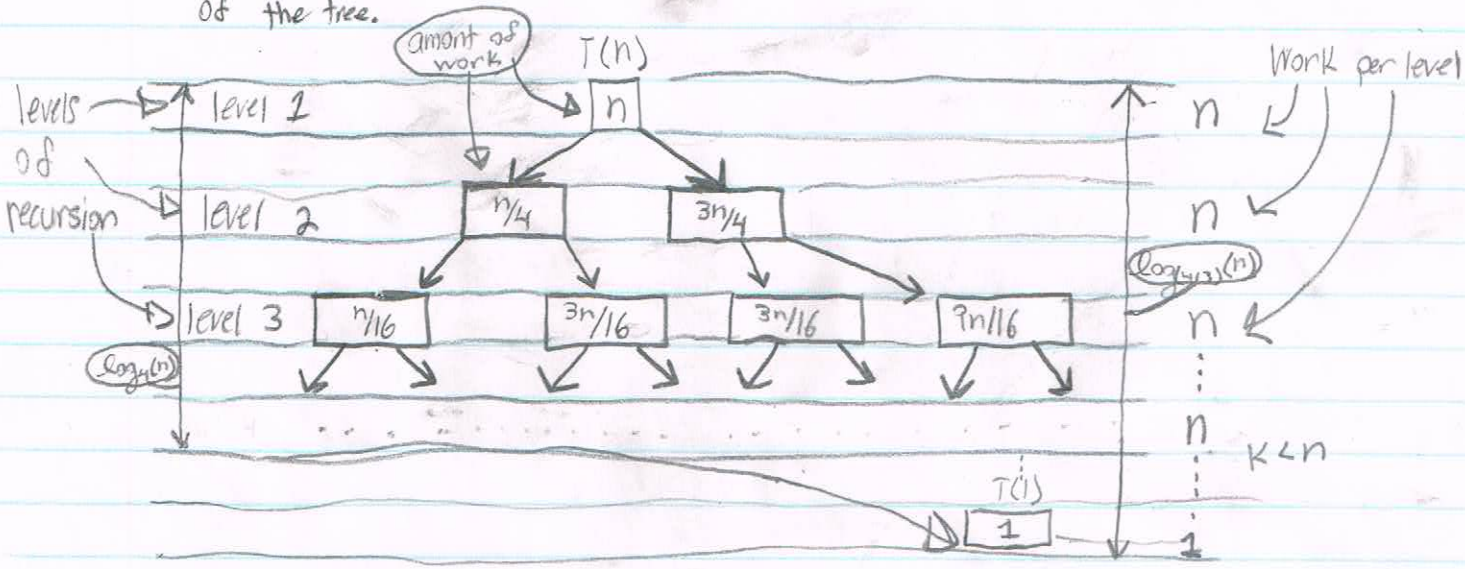
$= 2^n(T(0)+2)-2$

$= O(2^n)$ Simplified

C) This problem can be modeled as a recursive relation where $b_1 = \frac{n}{4}$ & $b_2 = \frac{3n}{4}$ added and joined in $O(n)$ time.

$T(\frac{n}{4}) + T(\frac{3n}{4}) + O(n)$ represents the recurrence relation

Using the recursion tree method, we can determine the time complexity of the tree.




$n\log_4(n) \leq T(n) \leq n\log_{4/3}(n)$

$T(n) = \Theta(n\log n)$ $\therefore O(n\log n)$

- Explenation: For the left side of the tree
  - We Assume: $n = 4^k$
  - We make it to a leaf in the tree when were at $T(1)$, the terminating step
  - To get to $t_1$, we must divide $n$ by $4^k$ such that it equals one
    - $\frac{n}{4^k} = 1$ thus: $n = 4^k$ thus $k = \log_4(n)$

- For the Right side of the tree: we do the same as above; $n = \frac{1}{3}$
  - $n(3/4)^k = 0 \rightarrow \log_{4/3}(n)$

- Thus we still get the estimate of: $\Theta(n\log n)$, but since the left half of the tree is shorter, we can't proove $\Omega(n\log n)$

- Since the final solution is combined in $O(n)$ time (defined by question), there is no deterministic $\Omega$, $\therefore O(n\log n)$ is the running time.
  - Thus its possible to get lower lower bound then $n\log n$, thus theres no $\Theta$, only $O$.

- Out of the Algorithms from parts A, B, & C from question 6, I would choose to use Algorithm [C].
- This is because the big Oh of Alg. C is the lowest, the runtimes for the Alg. is as follows:

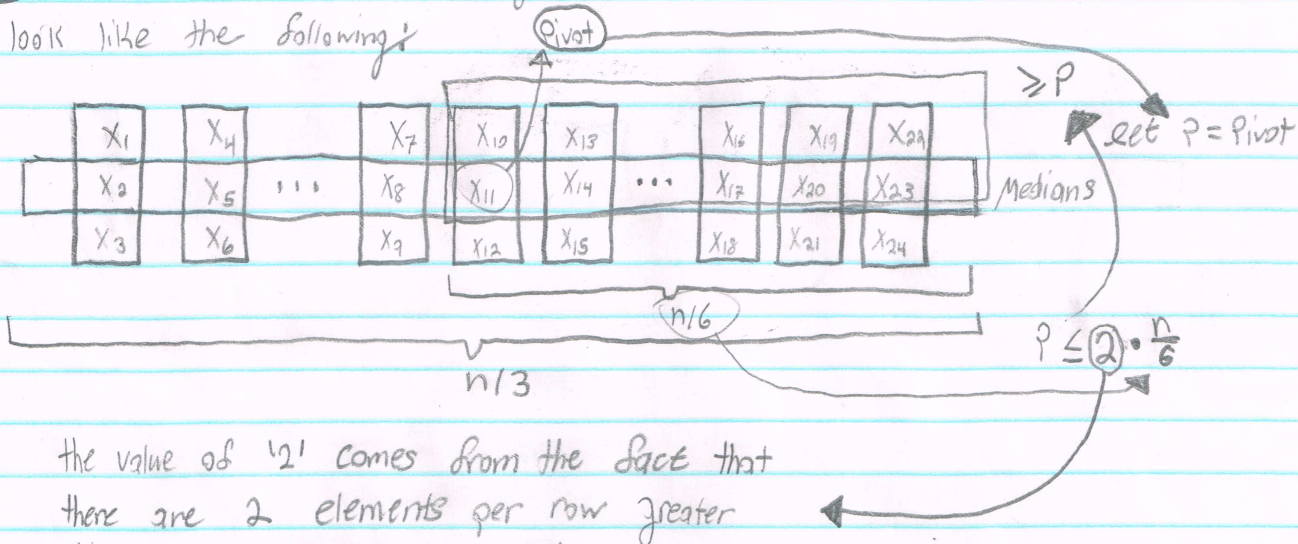
  A: $O(n^{\log_2(5)})$

  B: $O(2^n)$ Simplified

  C: $O(n \log n)$

Thus Alg. C has the lowest time Complexity w/ a big Oh of $O(n \log n)$, so I would use [Algorithm C].

(7) If we Partitioned the array into n/3 medians of size 3, it would look like the following:




the value of '2' comes from the fact that there are 2 elements per row greater then or equal to the current pivot.

a) As seen above, $P \leq 2 \cdot \frac{n}{6}$; thus the elements [Smaller] then the pivot would be the remainder of this value: Let $P_S$ = elements smaller than the Pivot

$2 \cdot \frac{n}{6} = \frac{2n}{6} = \frac{n}{3} \rightarrow n - \frac{n}{3} = \frac{2n}{3}$

$\therefore$ [$P_S < \frac{2n}{3}$] elements (may) be Smaller then the Pivot

The number of elements (guaranteed) to be larger then the pivot are:

$$P_L \geq 2 \cdot \left(\frac{1}{2}\left(\frac{n}{3}\right) - 1\right) + 2 \geq \frac{2n}{6} - 2$$

→ the # of elements included/excluded from equation

2 : the number of elements per median-group which is equal to or greater then the pivot

1 : the number of elements not greater then the pivot, which includes only the main Pivot itself

↳ unsimplified for clarity

$\frac{n}{6}$: the number of elements in the list, each element is a median-group of size 3

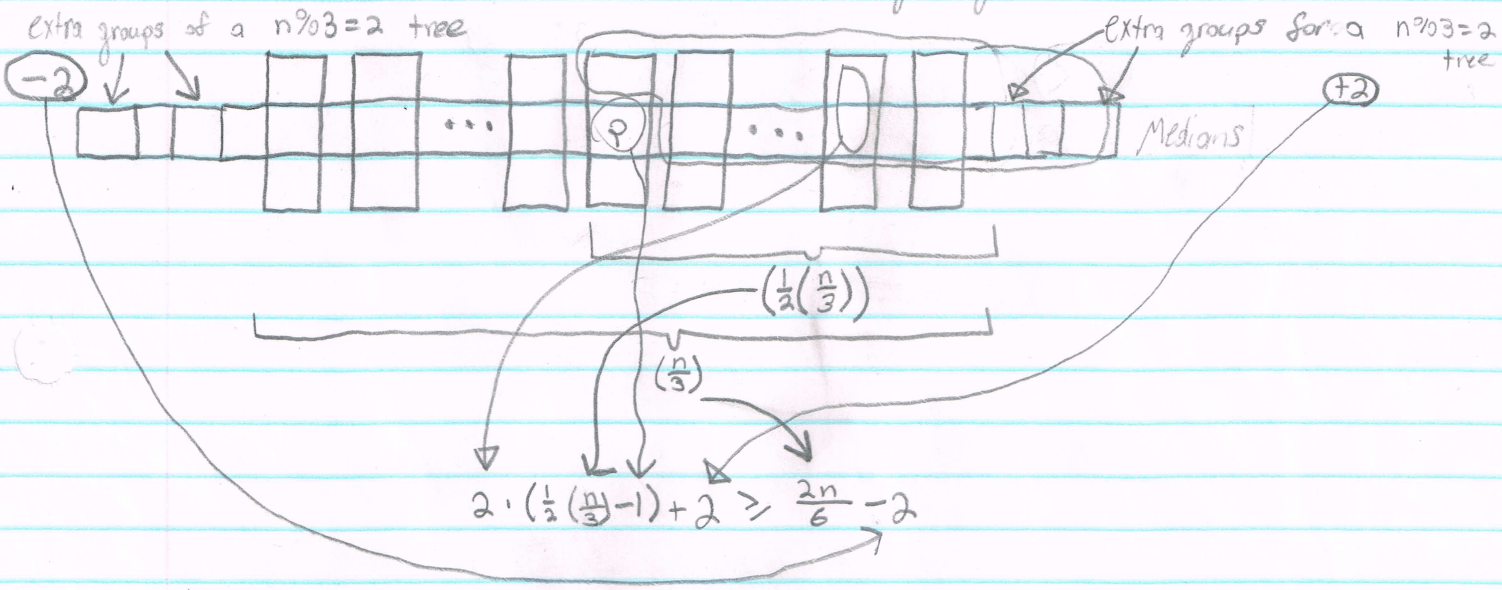
$\frac{2n}{6}$: the number of elements greater then or equal to the pivot, unsimplified for clarity.

thus: the guarenteed elements to be larger (or smaller) then the pivot is within the range of: let $P_L$ = elements larger then the Pivot / $P_S$ = elements Smaller then the Pivot

$$P_L \geq 2 \cdot \left(\tfrac{1}{2}\left(\tfrac{n}{3}\right) - 1\right) + 2 \geq \tfrac{2n}{6} - 2$$

$$P_S \geq 2 \cdot \left(\tfrac{1}{2}\left(\tfrac{n}{3}\right) - 1\right) + 2 \geq \tfrac{2n}{6} - 2$$

this can be best illustrated w/ the following diagram




b) The recurence for the running time for the Alg. can be broken into four steps:

Step 1: Divide S into $n/3$ groups, each of Length 3

↳ $O(n)$ time, we must iterate through the list 1 time inorder to accomplish this

Step 2: For $i = 1, 2, 3, \ldots, n/3$ groups; Compute the median of the $i^{th}$ group, & let this median be $m_i$

↳ $O(n)$ time, we must iterate through $n/3$ groups of 3, which is exactly $n/3 \cdot 3 = n$ iterations

Step 3: Compute the median p of $m_1, m_2, \ldots, m_{n/3}$

↳ $T(n/3)$ recurences, we can simply recursivly compute the $\frac{n}{6}$/th Smallest element of the sequence $m_1, m_2, \ldots, m_{n/3}$ & call Select($\{m_1, m_2, \ldots, m_{n/3}\}$, $\frac{n}{6}$) which takes $T(n/3)$ time.

↳ let Select be the algorithm discribed in this question (Q7)

Step 4: With p as the Pivot, use the Select Algorithm: Select(S, K) (S: Set; K: half of the list of medians)

↳ $O(n)$ time to iterate through the elements in a list of size $\frac{n}{3}$

\+ $T(\frac{2n}{3})$ recurrences to search through the longest list, the list of elements guaranteed to be smaller than the pivot.

↳ thus: $O(n) + T(\frac{2n}{3})$

thus: the recurrence for the running time is

$T(n) = \underbrace{O(n)}_{\text{Step 1}} + \underbrace{O(n)}_{\text{Step 2}} + \underbrace{T(\frac{n}{3})}_{\text{Step 3}} + \underbrace{O(n)+T(\frac{2n}{3})}_{\text{Step 4}}$

$= 2O(n) + T(\frac{n}{3}) + T(\frac{2n}{3})$

$= O(n) + T(n/3) + T(\frac{2n}{3})$ [$2O(n)$ is just $O(n)$]

c) we can find the running time of the Algorithm in terms of n via. induction:

Claim: $T(n) \leq cn$ for some constant n

Proof:

Base Case: Chose large c of for small n

let $c=10$ & $n=1$

$T(1) = 0$

$0 \leq 10n$

Inductive Hypothesis:

Let n be "large", assume $T(n') \leq cn'$ for all $1 \leq n' < n$

Inductive Step:

$T(n) = O(n) + T(n/3) + T(2n/3)$

$d > 0$ $\leq dn + \frac{cn}{3} + \frac{2cn}{3}$

$= dn + cn$

$\leq cn$ if $dn \leq 0cn \rightarrow dn \leq 0$

↳ this is a contridiction, the list must be greater then 0 elements!

Contridiction, $O(n)$ is not the runtime

Next, lets try nlogn:

Assume $T(n) = O(n \log n)$

Claim: $T(n) \leq cn\log n$ for some some constant $n$

Proof:

use the previouse base case

Inductive hyp.:

let $n$ be large, assume $T(n') \leq cn'\log n'$ for all $1 \leq n' < n$

Inductive Step:

$$T(n) = O(n) + T(n/3) + T(2n/3)$$
$$\leq dn + \frac{cn}{3} + \frac{2cn}{3}$$
$$= dn + cn$$
$$\leq cn\log n \text{ if } dn \leq c\log n \text{ iff } c \geq \frac{dn}{\log n}$$

$\therefore$ the $k^{th}$ Smallest element in a Sequence of $n$ #'s can be Computed in $O(n\log n)$ time.

We can use Masters Theorm to Support this Claim:

$T(n) = T(n/3) + T(2n/3) + O(n)$ $\quad a \geq 1$ & $b > 1$, $d \geq 0$

is Similar to $\Rightarrow G(n) = 3G(n/3) + O(n)$ Where $a=3, b=3, d=1$ $\quad \therefore$ Masters Applies

- $d = \log_b(a)$ then $T(n) = O(n^1 \log n) = O(n\log n)$
- $1 = \log_3(3)$
- $1 = 1$
- $\therefore T(n) = O(n\log n)$

Therefore, the running time is $O(n\log n)$ through Masters Theorm & induction.

(8) Given an unsorted array of n numbers, the Second Smallest number can be found in $n+(\log n)-2$ comparisons;

- To find the Second Smallest element, we must also find the Smallest.
- we can build a tree & find the Smallest element in $O(n)$ time: use tournement Comparison
  - ☐ we must pair all elements into groups of two & Compare them we compare $i_{2n}$ w/ $i_{2n+1}$ for list $0 \leq n < \frac{n-1}{2}$, if $i_{2n} > i_{2n+1}$ interchange the elements.

iteration: We can draw the following pattern: $\forall\, 0 \leq K < (n-2^{K-2})(2^{-K})$

1: if $i_{2n} > i_{2n+1}$ change
2: if $i_{4n} > i_{4n+1}$ change
⋮
K: if $i_{2^K K} > i_{2^K(K+2^{K-1})}$ change

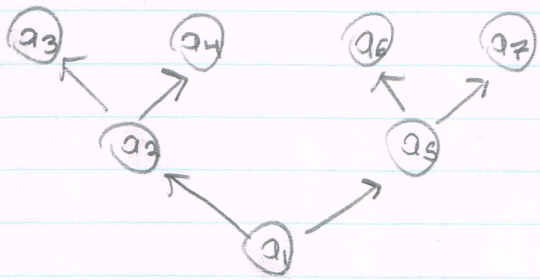
Swap elements if true

⇓

this runs for $n \leq 2^{K-1}$ trials, thus if the last Pass is the $K^{th}$ Pass then we must make 1 fewer then the number of elements. this is because we must compare every number in the Set except the Smallest:

$\therefore$ (n-1) Comparisons

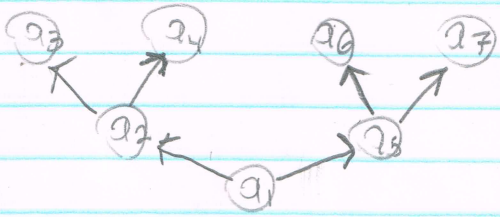
To illistrate:




Can result in n-1 Comparisons if every elements Compared & Swapped. The Smallest # in the Set will become the first element, & every other element will have been Compared. Hense: n-1

- the set is now in a sorted tree. We can simply iterate through the trees elements which have least direct comparisons to the smallest element.
  - Let $z$ be the number of comparisons made
  - there were a total of $2^{z-1} < n \leq 2^z$ comparisons to make the tree, thus the upperbound for $n$ can be $n = 2^z$.
    $\therefore z = \log n$
  - thus the tornement-tree can be serched through in $z$ ($\log n$) time
  - we already have the smallese element, thus there are only $0, 1, 2, \ldots, z-1$ comparisons that were smaller then the smallest element.
  - Worst case: $z-1$ comparisons to find the second smallest when $z$ is the time to serch the tornement-tree, when $z = \log n$
  - $\therefore$ $\log n - 1$ comparisons
    ↳ Assuming we use the floor of $\log n$ as the heap may be unbalenced.

this can be illustrated as:




Such that the elements $a_i$ are all sorted from step one.

Thus we have a tree, i.e. we know the smallest element
$\therefore$ 1 less comparison

- In total, building the tree via comparison & finding the smallest element is: $n-1$

Serching the tree minus the smallese element is:

$\log n - 1$

$\therefore$ the total serches are:

$(n-1) + (\log n - 1)$

$= n + \log n - 2$