Worrying about $\lfloor \rfloor$ in BSelect

#elts smaller than $p \geq 3\left\lfloor \left\lfloor n/5 \right\rfloor / 2 \right\rfloor - 1 = 3\lfloor n/10 \rfloor - 1$

$\Rightarrow |S| \geq 3\lfloor n/10 \rfloor - 1$

$\Rightarrow |L| \leq n - 3\lfloor n/10 \rfloor$

Symmetrically $|S| \leq n - 3\lfloor n/10 \rfloor$

So... $T(n) \leq c \cdot n + T(\lfloor n/5 \rfloor) + T(n - 3\lfloor n/10 \rfloor)$

If $n \leq 9$, $n - 3\lfloor n/10 \rfloor = n$ so the subproblem might not decrease!

We use sorting in this case.

$T(n) \leq c$ for $n \leq 9$

Claim $T(n) \leq d \cdot n$ for $n \geq 31$ $\quad d = 310c$

Proof

$$\begin{aligned} T(n) &\leq c \cdot n + T(\lfloor n/5 \rfloor) + T(n - 3\lfloor n/10 \rfloor) \\ &\leq c \cdot n + d\lfloor n/5 \rfloor + d(n - 3\lfloor n/10 \rfloor) \\ &\leq c \cdot n + dn/5 + d(n - 3(n/10 - 1)) \\ &= c \cdot n + dn/5 + d \cdot 7n/10 + 3d \\ &= c \cdot n + d\,9/10\, n + 3d \\ &= dn + \underbrace{cn - d/10\, n + 3d}_{\text{if } n \geq 31 \text{ and } d \geq 310c \text{ this is } < 0} \end{aligned}$$

Lower and Upper bounds on Problem Complexity

Given a problem P (for example, P = Sorting)

An optimal algorithm for P is an algorithm with the lowest complexity among all algorithms that solve P.

for example "worst case running time" ← we usually mean this
or "worst case space usage"
or "average case running time" ← what's the difference?
or "expected running time" ←

The complexity of P is the complexity of an optimal algorithm for P.

The complexity of many problems is unknown exactly

Suppose $T_A(n)$ is the worst case running time of a sorting algorithm A on inputs of size n.

The (worst case running time) complexity of Sorting on inputs of size n, call it $C_{SORT}(n)$ is:

is defined to be

$$C_{SORT}(n) \triangleq \min_{\substack{\text{all correct} \\ \text{comparison based} \\ \text{sorting algorithms } A}} T_A(n)$$

$$c_1 \cdot n \lg n \leq C_{SORT}(n) \leq c_2 n \lg n$$

Any comparison based algorithm can be modeled as a decision tree.
To sort correctly the tree must have $\geq n!$ leaves
$\Rightarrow$ must make $\geq c_1 \cdot n \lg n$ comparisons

$\min_A T_A(n) \leq T_{MergeSort}(n) \leq c_2 n \lg n$

Complexity of Finding the Median

↑ $\lfloor \frac{n+1}{2} \rfloor^{th}$ smallest

Let $T_A(n)$ = worst case running time of a median finding algorithm A on inputs of size n.

$C_{MEDIAN}(n) \triangleq \min_{\substack{\text{all correct} \\ \text{comparison-based} \\ \text{median finding} \\ \text{algorithms } A.}} T_A(n)$

c = constant

$? \leq C_{MEDIAN}(n) \leq c \cdot n$

↖ because $T_{BSELECT}(n) \leq c \cdot n$

(1) $C_{MEDIAN}(n) \geq \lg n$ [decision tree lower bound]

n possible outputs $\Rightarrow \geq n$ leaves in decision tree

$\Rightarrow \geq \lg n$ depth

$\Rightarrow \geq \lg n$ comparisons in worst case

(2) $C_{MEDIAN}(n) \geq n$ Any correct algorithm must look at all of the input.

Why? If Input is 3 3 3 3 _ _ _ algorithm doesn't need to look at

However, for worst case input, algorithm <u>does</u> need to look at all of the input.

Suppose input is : $X = [1\ 2\ 3 \ldots n]$ Ex: 1 2 3 4 5

then median is $\lfloor \frac{n+1}{2} \rfloor$ median is 3

Suppose algorithm doesn't look at x_i for some i

If algorithm says $m \neq \lfloor \frac{n+1}{2} \rfloor$ is median then it is wrong on this input.

If it says $\lfloor \frac{n+1}{2} \rfloor$ is median then it must say $\lfloor \frac{n+1}{2} \rfloor$ is median of $X' = [1\ 2\ 3 \ldots i-1\ (q)\ i+1 \ldots n]$

where $q = n$ if $i \leq \lfloor \frac{n+1}{2} \rfloor \Rightarrow$ median is $\lfloor \frac{n+1}{2} \rfloor + 1$

$q = 1$ if $i > \lfloor \frac{n+1}{2} \rfloor \Rightarrow$ median is $\lfloor \frac{n+1}{2} \rfloor - 1$

The point is the algorithm cannot tell the difference between input X and input X' (they only differ in location i, which the algorithm doesn't look at) so the algorithm produces the same output for both and must be wrong for either input X or input X' or both.

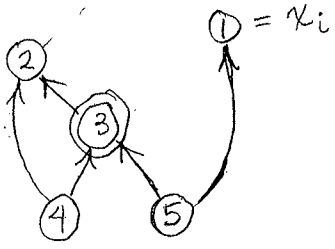
(3') $C_{MEDIAN}(n) \geq n - 1 + \frac{n-1}{2}$

Claim Any correct algorithm must establish relation ($>$ or $<$) between every element and median.

Proof Suppose on input X with median x_m the algorithm doesn't know if $x_i \leq x_m$ for some x_i
If alg. outputs "x_j is median" where $x_j \neq x_m$ then alg is wrong on input X.
If alg. outputs "x_m is median" then we can create an input X' by changing the values of some elements (including x_i) so x_m is not the median but the outcome of the comparisons made by the alg. is the same as for input X.

For example:

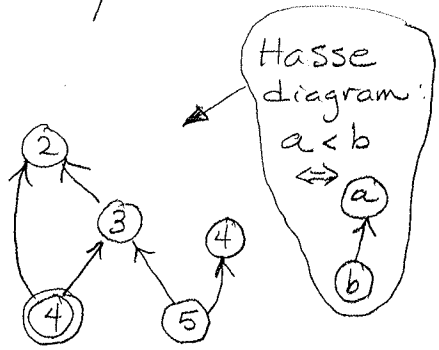
$X = [3\ 1\ 2\ 5\ 4]$ ($x_1\ x_2\ x_3\ x_4\ x_5$)



Comparisons:

(x_2, x_4) (x_1, x_3) (x_1, x_5) (x_1, x_4) (x_3, x_5)

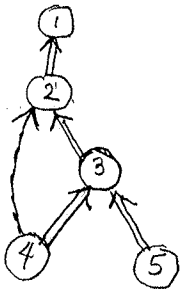
$\Rightarrow$



$X' = [3, 4, 2, 5, 4]$

Def A comparison involving element x is crucial if it is the first comparison $x > y$ where $y \geq$ median or $x < y$ where $y \leq$ median

Example:



crucial comparisons are "$\Rightarrow$"

Fact Alg. must make $n-1$ crucial comparisons.
(from previous claim)

Idea Construct input for the algorithm on-demand to force many non crucial comparisons.

"Adversary creates input"

(like a cheating player of 20 questions)

Call an element "Large" if it is bigger than median
"Small" " " smaller than median.

A comparison between a Large and Small element is non-crucial.

This is an Adversary strategy

So ...

① Start by labelling all elements "?"

② If algorithm asks "$x_i \lessgtr x_j$?" then:

non-crucial*	x_i	x_j	answer	action (until use up labels)
✓	?	?	"<"	label x_i "S", label x_j "L"
✓	S	?	"<"	label x_j "L"
✓	?	L	"<"	label x_i "S"
✓	S	L	"<"	nothing
✓	?	S	">"	label x_i "L"
✓	L	?	">"	label x_j "S"
	S	S	$<$ if $i<j$, $>$ if $i>j$	nothing
	L	L	$<$ if $i<j$, $>$ if $i>j$	nothing
	M	L		
	S	M		
	etc.			

+ $\frac{n-1}{2}$ labels of each kind "S" or "L"

+ one "M" label

If an algorithm stops before all elements are labelled then it is incorrect on some input.

⇓ why?

Alg makes at least $\frac{n-1}{2}$ noncrucial comparisons

⇓

Any correct median-finding algorithm must make $\geq n-1+\frac{n-1}{2}$ comparisons

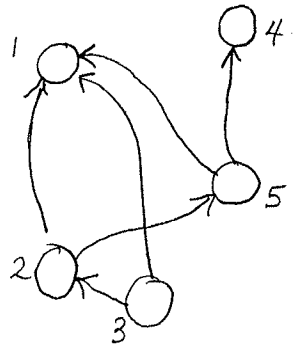
* Note: If we use up "S" labels (or "L" labels) the result of further comparisons is forced by what labels remain. These comparisons may be crucial then.

Example: Input x_1 x_2 x_3 x_4 x_5

Labels ? ? ? ? ?

S L L S M

Algorithm asks:		answer	relabel
✱ $x_1 \lessgtr x_2$	x_1=? x_2=?	<	x_1=S x_2=L
✱ $x_1 \lessgtr x_3$	x_1=S x_3=?	<	x_3=L
(Now we've used up L labels)			
$x_4 \lessgtr x_5$	x_4=? x_5=?	<	x_4=S x_5=M
(Now all elements are labelled)			
$x_2 \lessgtr x_3$	x_2=L x_3=L	<	
$x_2 \lessgtr x_5$	x_2=L x_5=M	>	
$x_1 \lessgtr x_5$	x_1=S x_5=M	<	



✱ = non crucial comparison

Other comparisons are crucial