

## Worrying about $L$ in BSelect

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

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

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

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

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

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

We use sorting in this case.

$$T(n) \leq c \quad \text{for } n \leq 9$$

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

$$\begin{aligned} \text{proof } T(n) &\leq c \cdot n + T(\lfloor n/5 \rfloor) + T(n - 3 \lfloor n/10 \rfloor) \\ &\leq c \cdot n + d \lfloor n/5 \rfloor + d(n - 3 \lfloor n/10 \rfloor) \\ &\leq c \cdot n + d n/5 + d(n - 3(n/10 - 1)) \\ &= c \cdot n + d n/5 + d \cdot 7n/10 + 3d \\ &= c \cdot n + d \cdot 9/10 n + 3d \\ &= d n + \underbrace{c n - d/10 n + 3d} \end{aligned}$$

if  $n \geq 31$  and  $d \geq 310c$

this is  $< 0$

# Lower and Upper bounds on Problem Complexity (28)

Given a problem  $P$  (for example,  $P = \text{Sorting}$ )

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  $C_{\text{SORT}}(n)$  is:  
is defined to be

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

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

Any comparison based algorithm can be modeled as a decision tree.

To sort correctly the tree must have  $\geq n!$  leaves

$\Rightarrow$  must make  $\geq c_1 n \lg n$  comparisons

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

# Complexity of Finding the Median

(29)

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

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

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

$c = \text{constant}$

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

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

①  $C_{\text{MEDIAN}}(n) \geq \lg n$  [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  $C_{\text{MEDIAN}}(n) \geq n$  Any correct algorithm must look at all of the input.

Why? If input is 3 3 3 3  $\leftarrow \leftarrow \leftarrow$   
 algorithm doesn't need to look at

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

Suppose input is  $X = [1 \ 2 \ 3 \ \dots \ n]$

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  $X' = [1 \ 2 \ 3 \ \dots \ i-1 \ \textcircled{q} \ i+1 \ \dots \ n]$

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  $\lfloor \frac{n+1}{2} \rfloor - 1$

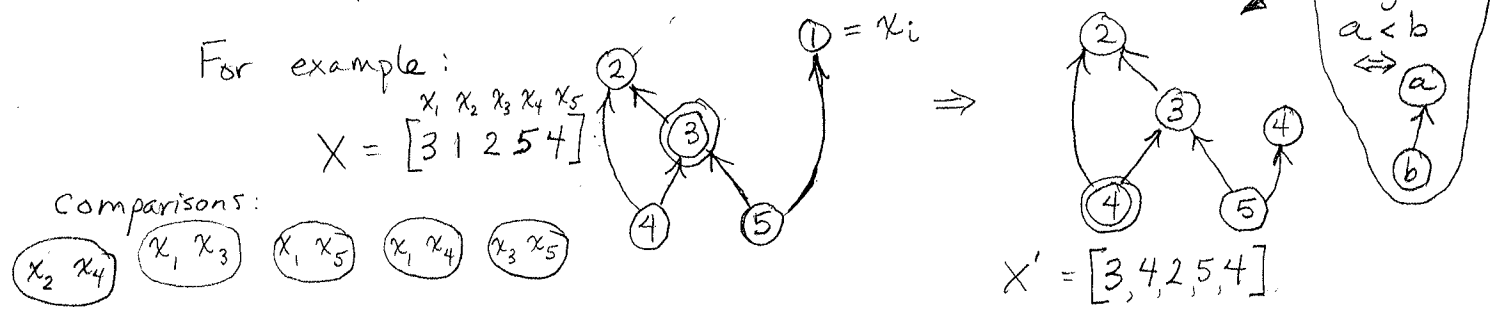
The point is the algorithm cannot tell the difference between input  $X$  and input  $X'$  (they only differ in location  $i$ , which the algorithm doesn't look at) so the algorithm produces the same output for both and must be wrong for either input  $X$  or input  $X'$  or both.

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

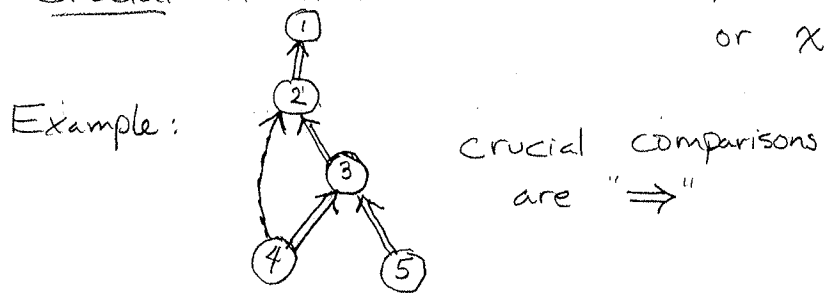
Claim Any correct algorithm must establish relation ( $>$  or  $<$ ) between every element and median.

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  $X$ .



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



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.

(like a cheating player of 20 questions)

"Adversary" creates input

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.

So ... ① Start by labelling all elements "?"

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

This is an Adversary strategy

+  $\frac{n-1}{2}$  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

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		
	M	L		
	S	M		
				etc.

\* 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
* $x_1 \leq x_2$	$x_1=?$ $x_2=?$	<	$x_1=S$ $x_2=L$
* $x_1 \leq x_3$	$x_1=S$ $x_3=?$	<	$x_3=L$
(Now we've used up L labels)			
$x_4 \leq x_5$	$x_4=?$ $x_5=?$	<	$x_4=S$ $x_5=M$
(Now all elements are labelled)			
$x_2 \leq x_3$	$x_2=L$ $x_3=L$	<	
$x_2 \leq x_5$	$x_2=L$ $x_5=M$	>	
$x_1 \leq x_5$	$x_1=S$ $x_5=M$	<	

\* = non crucial comparison

Other comparisons are crucial

