#cse2046

Decrease by a half algorithm

4.4.9. An array $A[0..n-2]$ contains $n-1$ integers from 1 to n in increasing order. (Thus one integer in this range is missing.) Design the most efficient algorithm you can to find the missing integer and indicate its time efficiency.

0	1	2	3	4	5	6	7	8
1	2	4	5	6	7	8	9	10

l $\quad$ l $\underline{3}$ m $\quad$ r $\quad$ $4\mathcal{E}$ $\underline{\underline{5}}$ $\quad$ r

$n = 10$

0	1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9	10

Algorithm MissingNumber$(A[0 \ldots n-2])$

$l \leftarrow 0, \quad r \leftarrow n-2$

while $l \le r$ do {

$m = \left\lfloor \frac{(l+r)}{2} \right\rfloor$

→ if $A[m] = m+1$

$l \leftarrow m+1$

→ else $\quad r \leftarrow m-1$

}

→ If $A[l] = l+1$ return $l+2$

→ else return $l+1$

$l \leftarrow 0, \ r = 8$

$0 < 8$

$m = \left\lfloor \frac{0+8}{2} \right\rfloor = 4$

$\underset{6}{A[4]} = \underset{5}{4+1}$

$r \leftarrow 4-1 \Rightarrow r = 3$

$m = \left\lfloor \frac{0+3}{2} \right\rfloor = 1$

$\underset{2}{A[1]} = 1+1$

$l \leftarrow 1+1 = 2$

$l < r$

$m = \left\lfloor \frac{2+3}{2} \right\rfloor = 2$

$\underset{4}{A[2]} = 2+1$

$r \leftarrow 2-1 = 1$

$\underset{2}{l} < \underset{1}{r}$

$\underset{4}{A[2]} = \underset{3}{2+1}$

return $l+1 = 2+1 = \underline{\underline{3}}$

$O(\log_2^n)$

Q.5.2. Apply quickselect to find the median of the list of numbers 9, 12, 5, 17, 20, 30, 8.

$n = 7$

$k = \lceil \frac{7}{2} \rceil = 4$

$k-1 = 3$

≤p	p	>p

s i

0 1 2 3 4 5 6

9 12 5 17 20 30 8

5

9 5 12 17 20 30 8

s s+1

9 5 8 17 20 30 12

8 5 9

17 20 30 12

17 20 30 12

s s+1

12 12 30 20

s

12 17 30 20

$A[3] = 12$

p	unknown

s i

p	<p	

s i

p	<p	

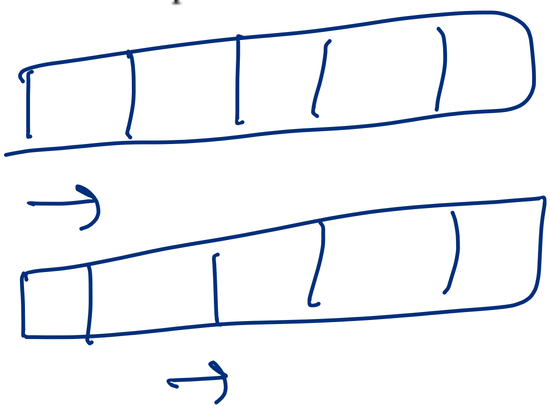
s i

≤p	p	>p

5.1.**2. a.** Write pseudocode for a divide-and-conquer algorithm for finding values of both the largest and smallest elements in an array of n numbers.

b. Set up and solve (for $n = 2^k$) a recurrence relation for the number of key comparisons made by your algorithm.

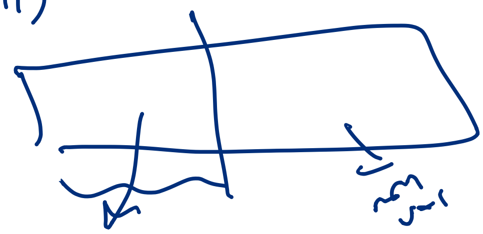
c. How does this algorithm compare with the brute-force algorithm for this problem?



minVal $\Theta(n)$

$+$ $2n$ in Brute

maxVal $\Theta(n)$



min val

min

max

Algorithm minMax(A, l, r)

if $l = r$

  minVal = A[l] , maxVal = A[l]

else if $r - l = 1$

  if $A[l] \le A[r]$

    minVal ← A[l]; maxVal ← A[r]

  else minVal ← A[r] ; maxVal ← A[l]

else // $r - l > 1$

minVal, maxVal = minMax(A, l, $\lfloor (l+r)/2 \rfloor$)

minVal2, maxVal2 = minmax(A, $\lfloor \frac{(l+r)}{2} \rfloor + 1$, r)

if minVal2 < minVal

  minVal ← minVal2

if maxVal2 > maxVal maxVal ← maxVal2

$C(n)$: # of element comparisons

$C(n) = 2C(n/2) + 2$ for $n > 2$ $\quad C(2) = 1$ ✓ $\quad C(1) = 0$

$n = 2^k$

$$C(2^k) = 2\,C(2^{k-1}) + 2$$
$$= 2[2C(2^{k-2}) + 2] + 2$$
$$= 2^2 C(2^{k-2}) + 2^2 + 2$$
$$= 2^2[2C(2^{k-3}) + 2] + 2^2 + 2$$
$$= 2^3 C(2^{k-3}) + 2^3 + 2^2 + 2$$
$$\vdots$$
$$= 2^{k-1} C(2^{k-k+1}) + 2^{k-1} + 2^{k-2} + \dots + 2$$
$$= 2^{k-1} + \underbrace{2^{k-1} + 2^{k-2} + \dots + 2}$$
$$2^{k-1} + \frac{2^{k-1+1} - 1}{2-1} = 2^{k-1} + 2^k - 1$$
$$= \frac{\boxed{2^k}}{2} + 2^k - 1$$
$$= \frac{n}{2} + n - 1$$
$$= \frac{3}{2}n - 2$$

c) For brute force, $2n$ key comparisons

For this, $1.5n$