國立清華大學試卷

記			分		
1		2			
3		4			
5		6			
7		8			
9		10			
11		12			
13		14			
15		16			
17		18			
19		20			
總 分					

所系 CS

科目 Algo.

學號 110062619

姓名 楊淨富

日期

index	1	2	3	4	5	3 is missing
value	1	2	4	5	6.	return 3.

1.

(a). Algo: Use binary Search

① First, check if $\lfloor \frac{n}{2} \rfloor == A[\lfloor \frac{n}{2} \rfloor]$ if so., binary search on range: $\lfloor \frac{n}{2} \rfloor + 1, n$

else.

do binary search on $1, \lfloor \frac{n}{2} \rfloor$.

② Recursively call step ① until we find the first number at index i such that $i != A[i]$, return i

Correctness: Because it is a sorted array, so we can do binary search. And each round, we move closer to find the first number that its index doesn't match its value which means its index is the missing number.

Time: Each round, we reduce problem size to half of the previous one.

So, $T(n) = T(\frac{n}{2}) + 1$,

$= O(\log n)$.

(b).

Algo: Do binary search to find the minimum number. Min

At any location p

if $A[1] > A[p]$, Min is on p's left.

else $A[1] < A[p]$, Min is on p's right.

Correctness: Since the all the numbers in A left are larger than all the numbers in A right, we can easily use above binary search to find Min.

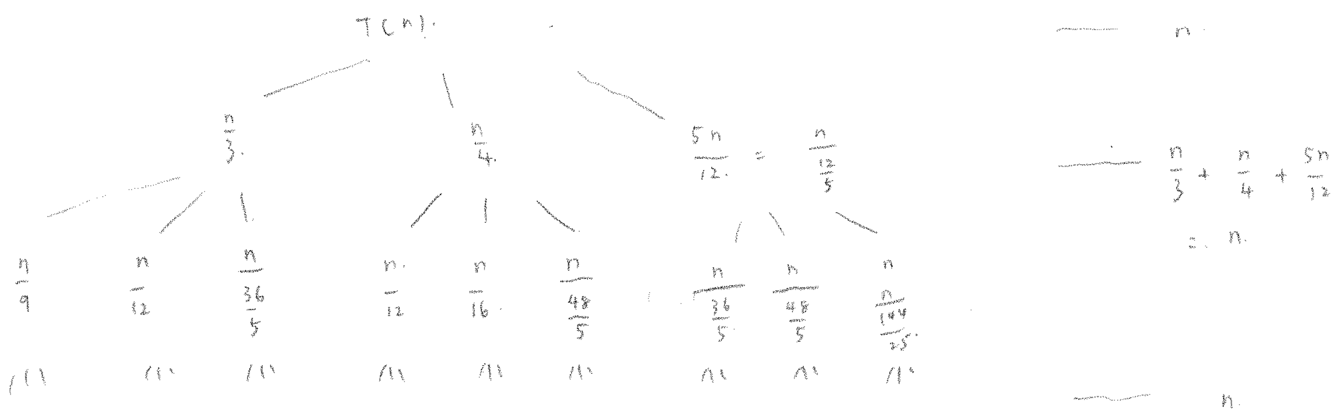
Time: Each step, we reduce problem size to half of the previous one.

$\therefore O(\log n)$

$\uparrow$

$T(n) = T(\frac{n}{2}) + 1$.

2. By recursion tree.



$\frac{n}{4^{k_1}} = 1$, $k_1 = \log_4 n$ $\Rightarrow$ Recursion tree's min height

$\frac{n}{(\frac{12}{5})^{k_2}} = 1$, $k_2 = \log_{\frac{12}{5}} n$ $\Rightarrow$ Recursion tree's max height

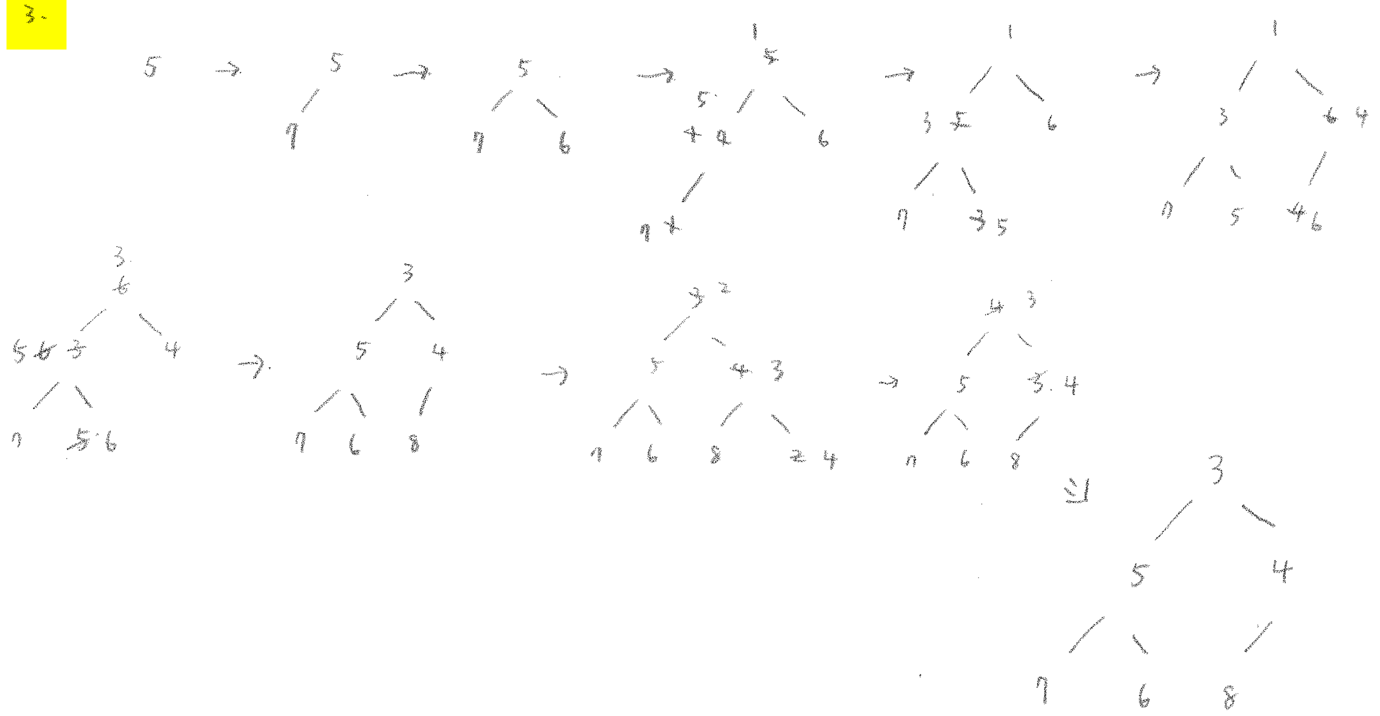
$T(n) = \text{tree height} \times O(n)$

$k_1 \cdot n \leq T(n) \leq k_2 \cdot n$

$n \log_4 n \leq T(n) \leq n \cdot \log_{\frac{12}{5}} n$

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

3.



4. 1 3^* $3'$ 3 2

k=1. j=1.
j=2. 1 $3'$ 3^* 3 2.

1 $3'$ 3 3^* 2.

1 $3'$ 3 2 3^*

k=2.
j=2 1 3 $3'$ 2 3^*

3. 1 3 2 $3'$ 3^*

4 1 3 2 3^* $3'$

k=3.
j=2. 1 2 3 3^* $3'$

3 1 2 3^* 3 $3'$

4 1 2 3^* $3'$ 3

k=4.
j=1.
j=2. → 1 2 $3'$ 3^* 3
j=3.

j=4 1 2 3^* $3'$ 3.

Reason for "NOT STABLE": line 3 : if A[j] ≥ A[j+1]

⇒ so if A[j] == A[j+1], two same number will be rearranged and thus it causes unstable order.

Change : Revise line 3 to if A[j] > A[j+1]

5.
(a) 4 distinct numbers ⇒ 4! = 24 results.

⇒ By decision tree, we can infer that

$24 \leq 2^h$

$h \geq \log_2 24 = 4.xxx.$

$h = 5.$ ← the minimum height of decision tree so we must have at least 5 comparisons

(b). Algo: Use merge sort.

a_1 a_2 ① a_3 a_4 ②

a_1 a_2 a_3 a_4.

① We compare (a_1, a_2), (a_3, a_4), # comparisons = 2.

② Merge step : At most, we will compare $n-1 = 4-1 = 3$. times.

So total number of comparisons is $2+3 = 5$ (at most).

follow 5(b).

Correctness: First step, we make (a_1, a_2), (a_3, a_4) sorted. # comparisons = 2.

Second step, we merge two sorted list by just keep comparing the first number of two sorted list and even in the worst case, we only need $(n-1)$ times comparisons which is 3 in this case. So total # comparisons = 2+3 = 5.

Time: Each step we at most spend $O(n)$ for merge.

And tree height is $O(\log n)$.

Total time consumes $O(n \log n)$.

6.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

① Round 1: Compare the last character by ~~lexi~~ 字典排序.

1	2	7	18	23	24
SEA, TEA	TAB, MOB	RUG, DOG, BIG, DIG	TAR, EAR, BAR	ROW, COW, NOW	BOX, FOX

→ output: TEA → SEA → MOB → TAB → DIG → BIG → DOG → RUG → BAR → EAR → TAR → NOW → COW → ROW → FOX → BOX

② Round 2: Compare the 2nd last character from round 1's output by 字典排序.

1	5	9	15	21
TAR, EAR, BAR, TAB	SEA, TEA	BIG, DIG	BOX, FOX, ROW, COW, NOW, DOG, MOB	RUG

→ output: TAB → BAR → EAR → TAR → TEA → SEA → DIG → BIG → MOB → DOG → NOW → COW → ROW → BOX → FOX → RUG

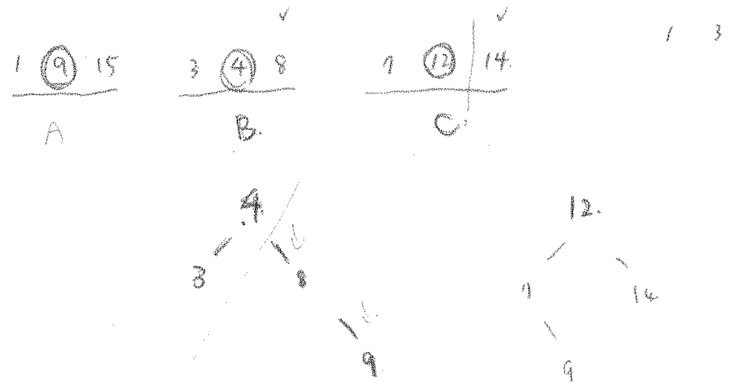
③ Round 3: Compare the 3rd last character from round 2's output by 字典排序

2	3	4	5	6	13	14	18	19	20
BOX, BIG, BAR	COW	DOG, DIG	EAR	FOX	MOB	NOW	ROW, RUG	SEA	TEA, TAR, TAB

output:

AR → BIG → BOX → COW → DIG → DOG → EAR → FOX → MOB → NOW → ROW → RUG → SEA → TAB → TAR → SEA

7.



1 3 | 4 7 8 | 9 12 14 15

3个数比9小　　2个数比9大.　　∴ 9 is not median with 3n numbers

Idea: 拿A的median去跟B./C作Binary search (如上圖).

目的是要看"比A大"跟"比A小"的數.是否相同.

由於B.C是Sorted list. 所以可以知道有多少數字 { 比A之median大 / 比A之median小

每次Binary search.

即B.S.T.對稱

所以每一次決定往左or往右時,是可以推算出

Time: 由於皆為Binary search之operation ∴ $O(\log n) * 2 = O(\log n)$

8.

By counting sort, we can sort number in $O(n+k)$, k is value range,

count $[1 \cdots x]$

1	2	3	4	...	x
0	1	0	1		1

which is x in this case. Since x is independent of n. we take it as a constant. So total running time will be $O(n+k)$

$$= O(n+c)$$

$$= O(n).$$

Detail for counting sort:

We create an array count $[1 \cdots x]$, and for each item, we put it into its correspond index. e.g. an item is 10 kg then count $[10] = 1$.

which means increment the value.

Also, we can have a start array to record the start position of items.

e.g. count [5] =>

1	2	3	4	5
0	1	0	1	1

start [5] =>

1	2	3	4	5
0	1	0	2	3

Finally, we can have an output array for output sorted numbers.

e.g. output[3]

1	2	3
2	4	5