Algorithms & complexity 2014 EE10-C 1

1

a) $T(n) = 4\,T(n/2) + n^3$

$a = 4$; $b = 2$ $d = 3$.

$\log_b a = \log_2 4 = 2 < d = 3$.

Hence $T(n) = O(n^3)$

b) $T(n) = 17\,T(n/4) + n^2$

$a = 17$ $b = 4$ $d = 2$

$\log_b a = \log_4 17 > \log_4 16 = 2 > d$

Hence $T(n) = O\left(n^{\log_4 17}\right)$

c) $T(n) = 9\,T(n/3) + n^2$

$a = 9 \quad b = 3 \quad d = 2$

$\log_b a = \log_3 9 = 2 = d = 2$

$\Rightarrow T(n) = O(n^2 \log n).$

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

Let $n = 2^k$

$T(2^k) = T(2^{k \cdot 1}) + 1$

$= O(k) = O(\log n).$

2

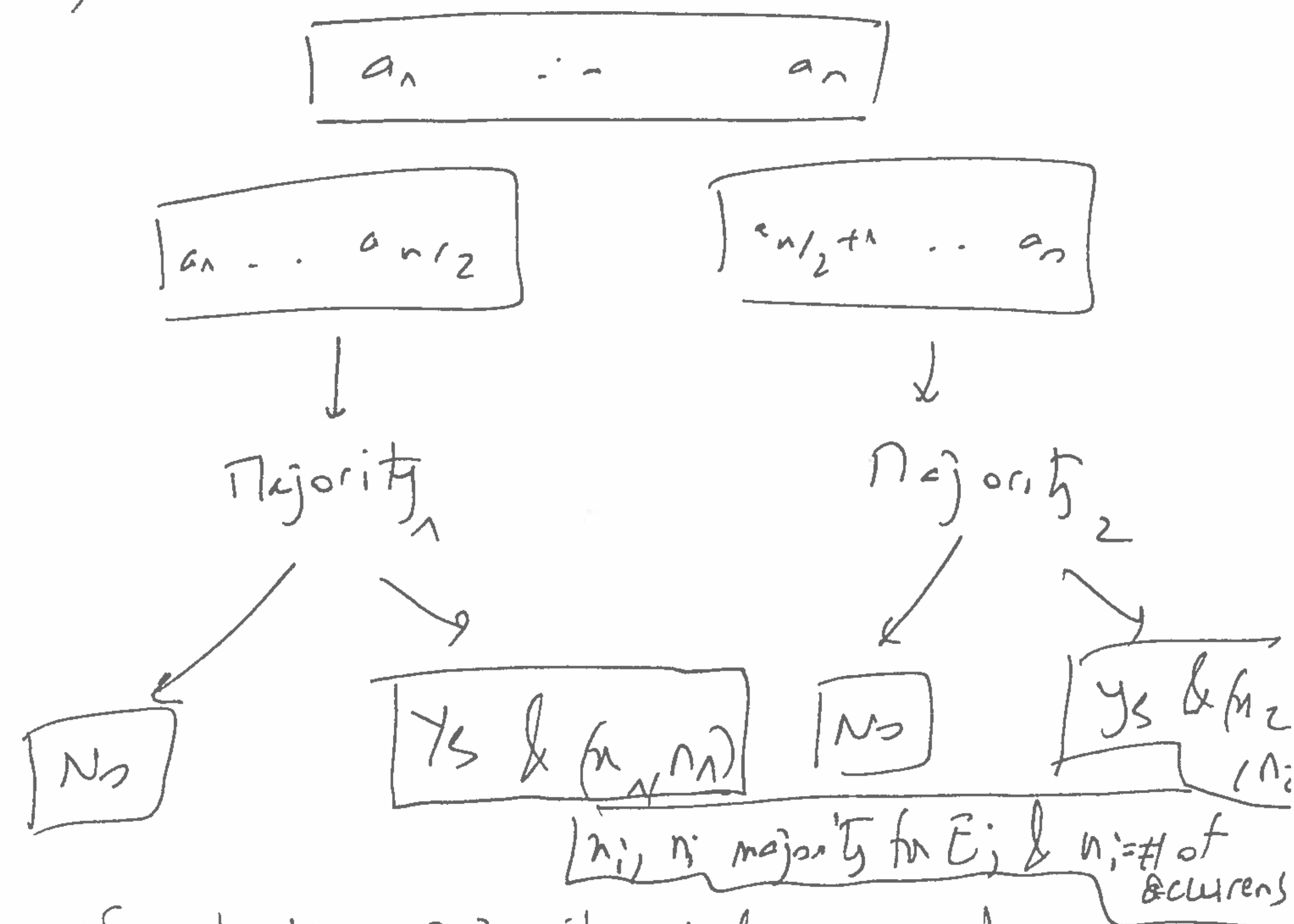
a) For each a_i the for loop on j counts the number of times a_i appears in the list.

For each a_i it then check whether it appears more than $n/2$ times.

The algorithm then either returns "there is no majority element" if none of the a_is appears more than $n/2$ times or returns the UNIQUE element that appears more than $n/2$ times.

The algorithms runs two for loops on i & j from 1 to n So its complexity is $O(n^2)$.

b)



+ if both Majority 1 & 2 return no then there is no majority number since no says that all a_i occur fewer than $n/4$

times in each of the lists E_1 & E_2 5

& no fewer than $n/2$ times in total.

* if Majority 1 returns (x_1, n_1)
  & Majority 2 returns no.

  Here we need the number of occurrences of x_1 in E_2, n_2, if $n_1 + n_2 > n/2$ then yes x_1 is a majority element. Otherwise no.

* Similarly if Majority 2 returns (x_2, n_2) & Majority 1 returns no.

* ~~Finally~~ if Majority 1 & Majority 2 return (x, n_1) & (x, n_2) then clearly x is the majority element.

* Finally if Majority 1 returns (x_1, n_1) (6)

& Majority 2 returns (x_2, n_2) $x_1 \neq x_2$

we need to look at the occurence

of x_1 in $E_2 = m_{12}$

& x_2 in $E_1 = m_{21}$

then see whether $n_1 + m_{12}$ or

$n_2 + m_{21}$ is larger than $n/2$.

Given a list and a number let

$Num(E, x) = \#$ appearency of x in E.

Majority $(a_1 \dots a_n)$ 7

Let $E_1 = a_1 \dots a_{n/2}$; $Maj_1 = Majority(E_1$

$E_2 = a_{n/2+1} \dots a_n$; $Maj_2 = Majority(E_2$

If $Maj_1 = (*, 0)$ and $Maj_2 = (*, 0)$

then $(*, 0)$

else

If $Maj_1 = (x_1, n_1)$ & $Maj_2 (*, 0)$.

then
if $n_1 \leftarrow n_1 + Num(x_1, E_2) > n/2$

then (x_1, n_1)

else $(*, 0)$.

else if $Maj_2 = (x_2, n_2)$ & $Maj_1 = (*, 0)$

then if $n_2 \leftarrow n_2 + Num(x_2, E_1) > n/2$

then (x_2, n_2)

else $(*, -)$

else if $Maj_1 = (x_1, n_1)$ & Maj_2 (x_2, n_2)

then if $x_1 = x_2$ then

$(x_1, n_1 + n_2)$.

else

if $n_1 \leftarrow n_1 + Num(x_1, E_2) > n/2$

then (x_1, n_1)

else

if $n_2 \leftarrow n_2 + Num(x_2, E_1) > n/2$

then (x_2, n_2)

else $(*, 0)$.

D

REMARK: Students can either provide text or a pseudocode $\boxed{\text{BONUS } (+2)}$ if they provide both

Complexity.

$$C(n) = 2\left[C(n/2) + n/2\right]$$

call for E_1 E_2

needs to count the number of time an element appears in one of the sublists E_1 & E_2

$$C(n) = 2C(n/2) + n$$

Master theorem $\rightarrow$ $C(n) = O(n \log n)$.

3

a) We have to look at all possible subwords in both words $2 \cdot 2^n$.

b)

i) $a_i \neq b_j$ this means that a_i & b_j cannot belong to the same subword.

Hence either a_i belongs to the common subword & $p(i,j) = p(i, j-1)$

or b_j & $p(i,j) = p(i-1, j)$

ii) $a_i = b_j = x$ then either x belongs to the common subword in which case $p(i,j) = p(i-1, j-1) + 1$

or not & $p(i,j) = p(i-1, j-1)$.

iii) Putting all the pieces together we have

$$p(i,j) = \max\left(p(i,j-1),\ p(i-1,j),\ p(i-1,j-1) + \delta_{a_i b_j} \right).$$

iv) either i or j decreases at each step and we have at most mn steps yielding a complexity of $O(mn)$.

c/

b	0	1	2	2	3	4	4
a	0	1	2	2	3	3	3
d	0	1	2	2	2	3	3
b	0	1	1	2	2	3	3
c	0	1	1	2	2	2	2
b	0	1	1	1	1	2	2
a	0	0	0	0	1	1	1
$\emptyset$	0	0	0	0	0	0	0
	$\emptyset$	b	d	c	a	b	a.

Hence b d a b