$T(n) = 4T(n/2) + n^2$ $\quad aT(n/b) + \Theta(n^k \log^p n)$

$a = 4,\ b = 2,\ k = 2,\ p = 0$ $\quad \log_2 4 = \log_2 2^2 = 2\log_2 2 = 2$

$a\ b^k$

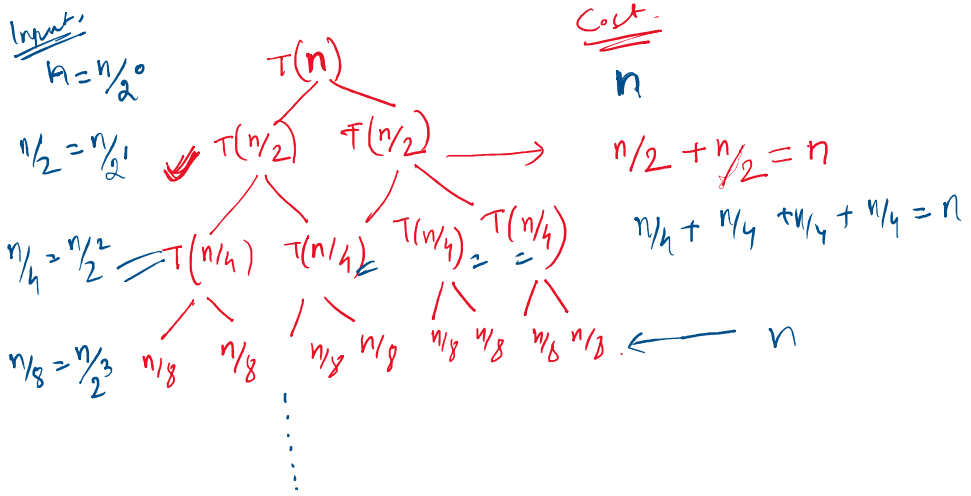
$4 = 2^2$ Cond^n-2nd (a) $\rightarrow$ $T(n) = O(n^{\log_b a} \log^{p+1} n)$

$= O(n^{\log_2 4} \cdot \log^{0+1} n)$

$= O(n^2 \cdot \log n)$

Recursion tree method –

$$T(n) = \begin{cases} 2T(n/2) + n, & n > 1 \\ 1, & n = 1 \end{cases}$$



Total cost =

Steps (0 to k) $\equiv (k+1)$ steps

$n \times (k+1)$

$= nk + n$

$= (n\log_2 n + n)$

$= n\log_2 n.$

Recursion Tree $\rightarrow$ $\left(\frac{n}{2^k}\right) = 1$ $\quad$ from question.

Recursion stops when input size $\Rightarrow \left(\frac{n}{2^k}\right) = 1$

Recursion stops when input size $\Rightarrow \left(\frac{n}{2^k}\right) = 1$

$\Rightarrow n = 2^k$

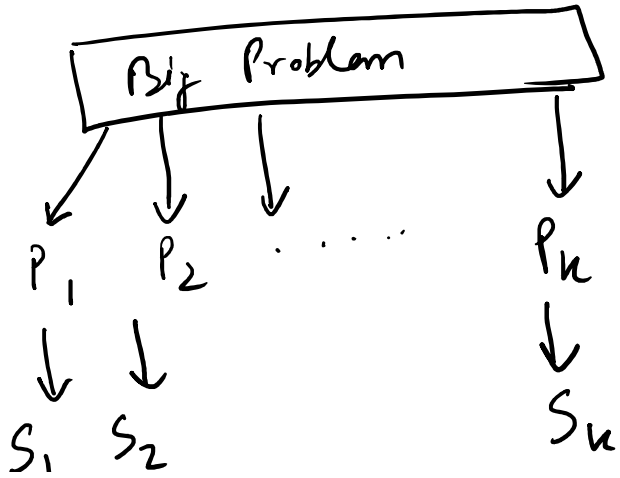
$\Rightarrow \log_2 n = \log_2 2^k = k \log_2 2 = k$

$\Rightarrow k = \log_2 n$

$\log_a a = 1$

H/W: $T(n) = \begin{cases} 2T(n/2) + c, & n > 1 \\ 1, & n = 1 \end{cases}$ Using Recursion Tree.

Divide & Conquer method. —— Technique to design algorithm.



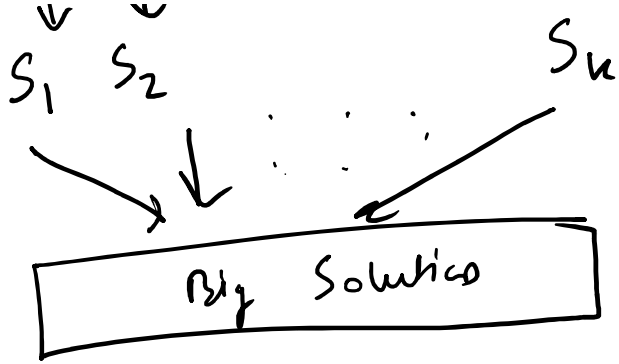
(Small sub-problems)

(Solution of small sub-problems)

Eg: $1+2+3+4$

$= 10$

$(1+2)+(3+4)$



(sub-problems)

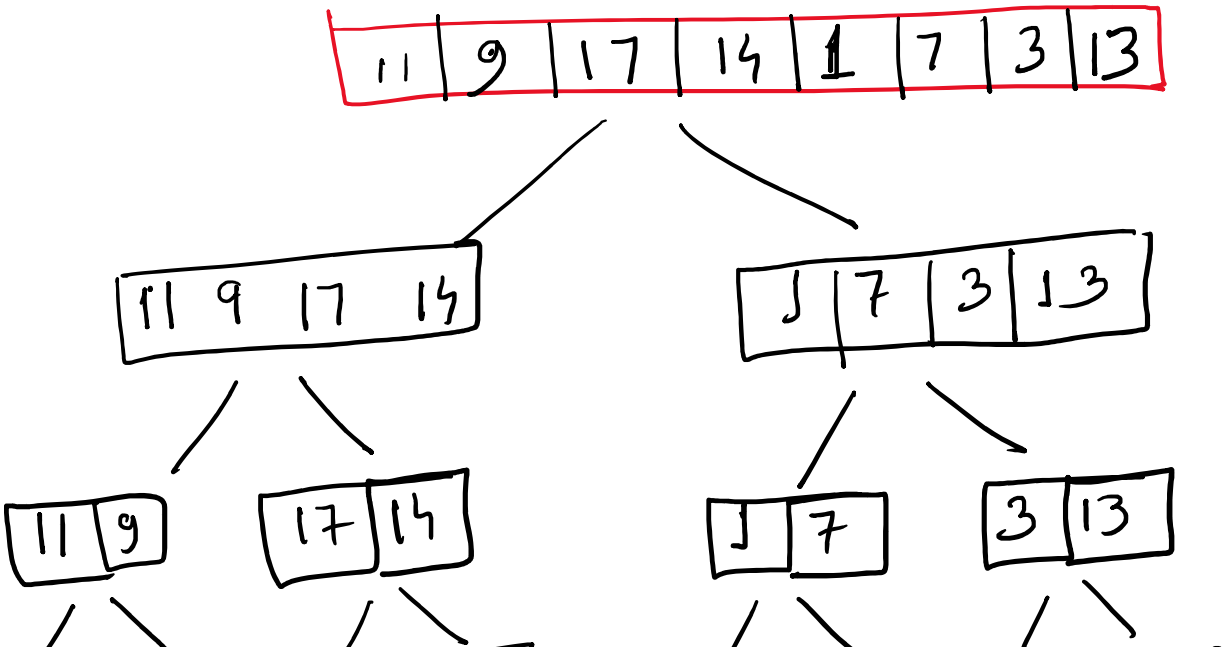
(1+2)+(3+4)

P1 P2

S1 = 3 S2 = 7

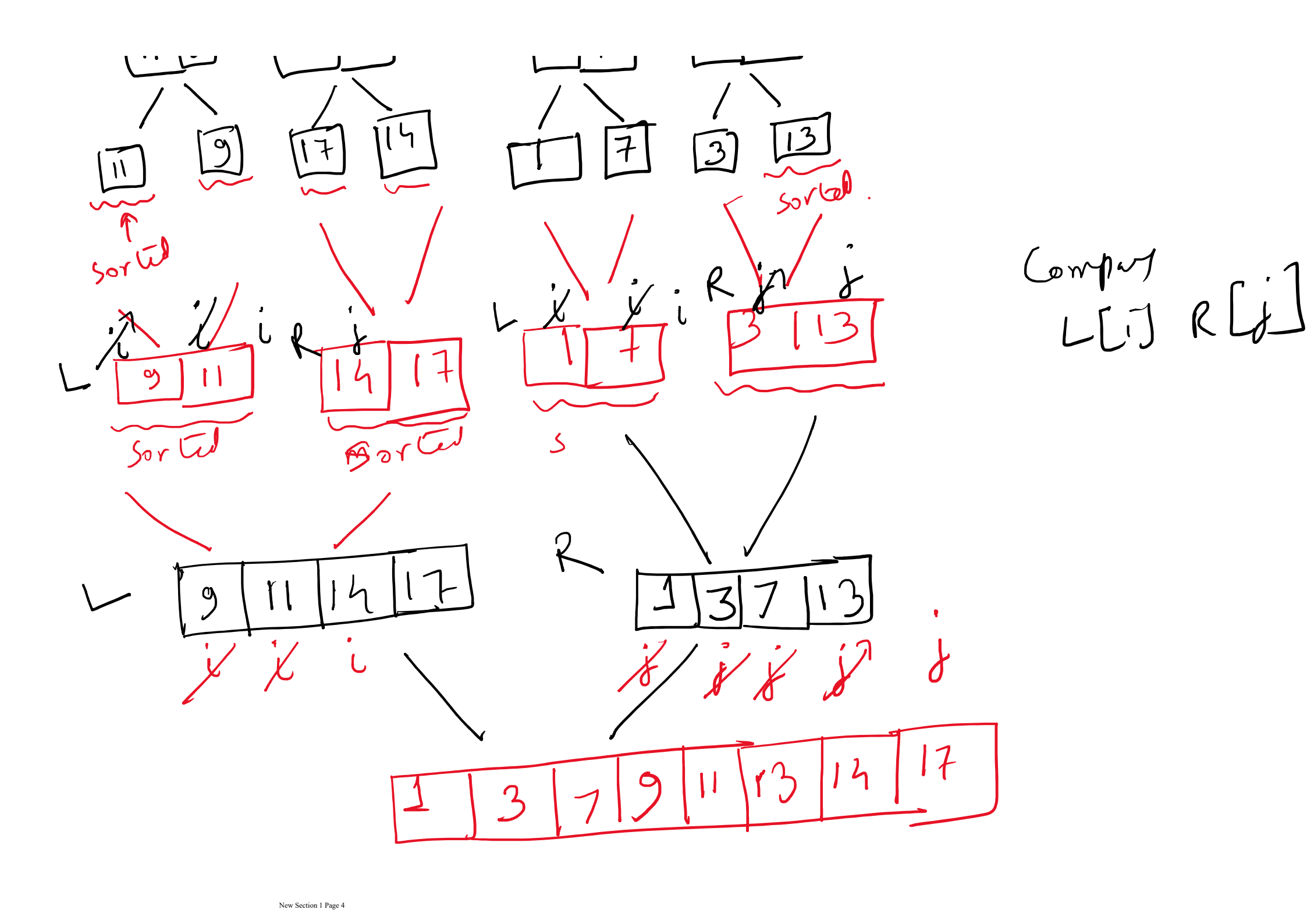
10

Merge Sort - Based on divide & conquer method.



Note -

1. Repeatedly divide the array into 2 equal parts.
2. Merge 2 sorted arrays into 1 sorted array.


11
9
17
14
1
7
3
13
sorted
sorted
L
R
i
j
9
11
14
17
1
7
3
13
Compare
L[i] R[j]
1
3
7
9
11
13
14
17