Q1 Partition(){

while l < r

if arr[j] < pvt

arr = swap(arr, i, j)

i++

j++

arr = Swap(arr, i, r)

return i;

if (l <= r) do

pivot

pivot = random() % n

arr = Swap(arr, pivot+1, r)

PI = (arr, l, r)

if (PI == k)

b = arr[PI]

if (a != -1)

return minValue

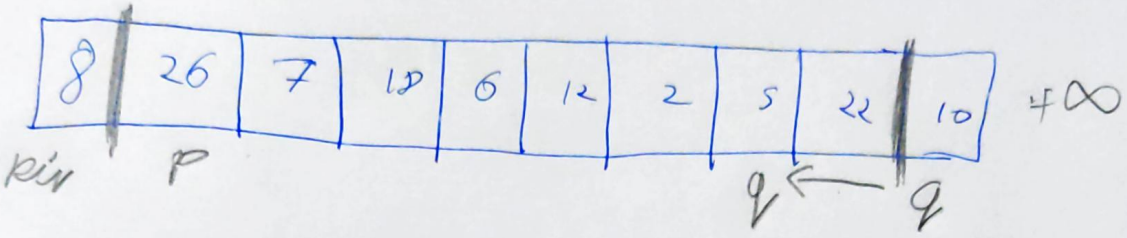
return median

Time Complexity : O(n)

The time complexity of the original partition algo. is also $O(n)$; same as the new one.

$A = \{8, 26, 7, 18, 6, 12, 2, 5, 22, 10\}$

a)



b) 8 5 7 18 6 12 2 26 22 10

Piv P → P q ← q

c) 8 5 7 2 6 12 18 26 22 10

Piv P → P q q

d) 6 5 7 2 8 12 18 26 22 10

Piv P → P Piv P q

e) 6 5 2 7 8 12 10 26 22 18

q ← q P → P q q

f) 2 5 6 7 8 10 12 26 22 18 $+\infty$

P → P q

g) 2 5 6 7 8 10 12 18 22 26

Now the array is sorted

no. of swapping = 6

no. of comparisons = 15

Q2 The average case time complexity of Quick Sort is - $\Theta(n)$

The Worse Case time complexity of Quick Sort is - $O(n^2)$

The Worse Case Space Complexity of Quick Sort is = $O(\log n)$

Though the worse case time complexity of Bubble, selection and insersion sort is the same as Quick Sort, for sorting larger array Quick Sort is prefered as Quick Sort follows divide & conquore strategy, it divides the problem into sub parts while sorting the array which requires comparitively less time than others.

Q3 MergeSort(A, p, r)

```
if p < r
    then q = ⌊(p+r)/2⌋
         MergeSort(A, p, q)
         MergeSort(A, q+1, r)
         Merge(A, p, q, r)
```

Merge (A, p, q, r)

$n_1 = q - p + 1$

$n_2 = r - q$

for $i = 1$ to n_1

do $L[i] = A[p + i - 1]$

for $j = 1$ to n_2

do $R[j] = A[q + j]$

$L[n_1 + 1] = \infty$

$R[n_2 + 1] = \infty$

$i = 1$, $j = 1$

for $k = p$ to r

if $L[i] \geq R[j]$

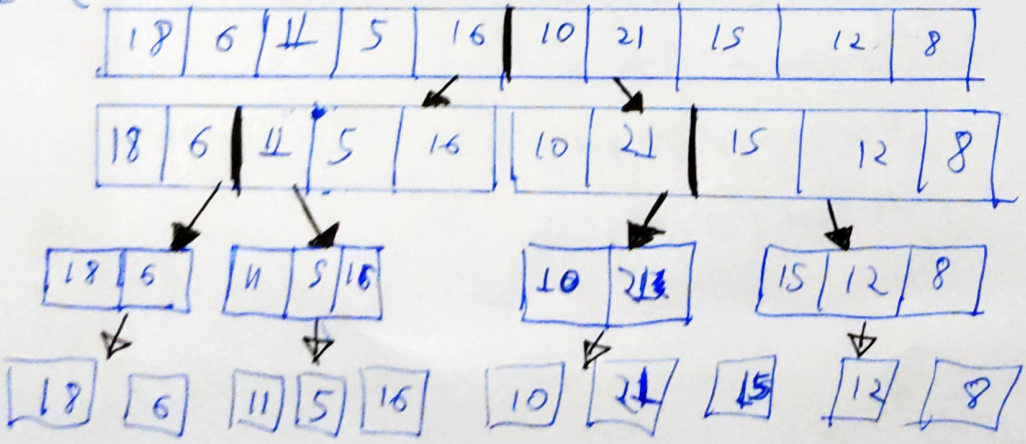
then $A[k] = R[j]$

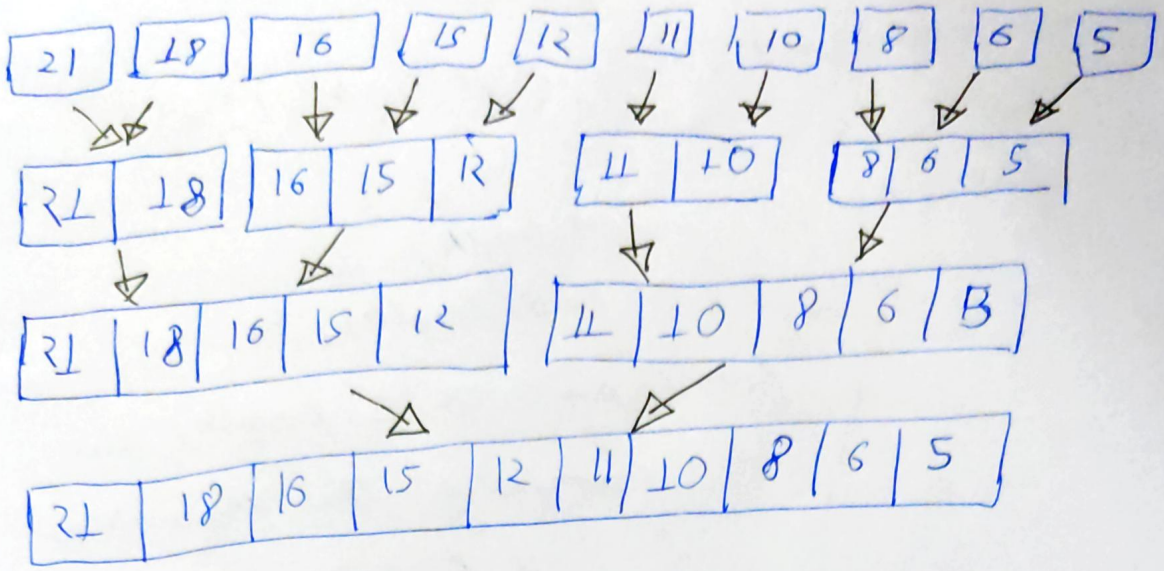
$j = j + 1$

else $A[k] = L[i]$

$i = i + 1$

$A = \{18, 6, 11, 5, 16, 10, 21, 15, 12, 8\}$





The array is sorted

Q4

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}
A	7	10	12	8	2	1	6	3	5	4	11	9
B	3	1	7	11	8	2	6	9	5	12	4	10
C	6	2	4	11	10	12	3	8	1	5	9	7

Degree of dissimilarity of A = 5

Degree of dissimilarity of B = 3

Degree of dissimilarity of C = 4

∴ 'A' has the highest degree of dissimilarity from others, so, 'A' has the chance to win the prize.

Reference Comparision (A[], a, B[], b, C[], c) {

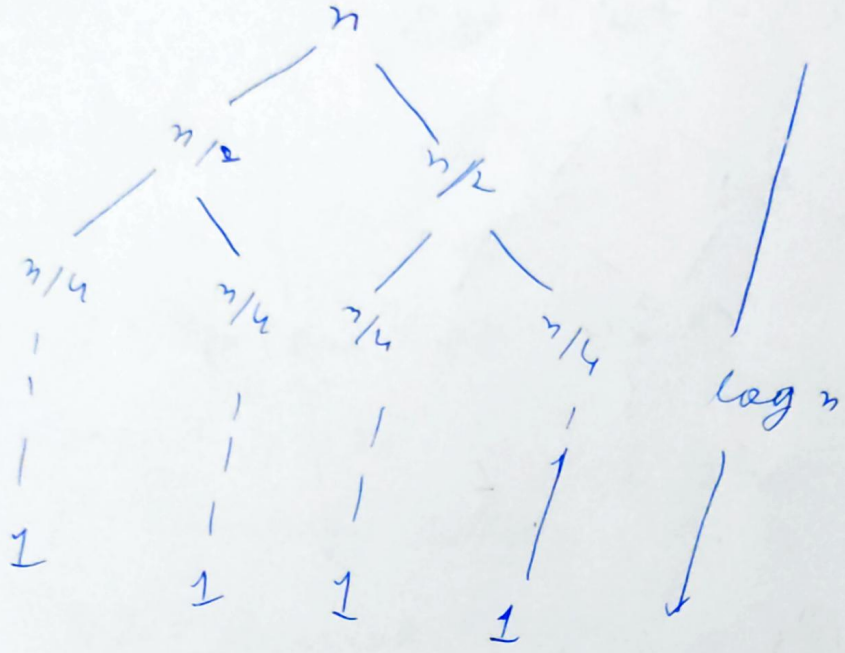
1. a = b = c = 0
2. for (i = 0 to A.length) do
3. for (j = 0 to B.length) do
4. for (k = 0 to C.length) do
5. Compare the three arrays.
6. if A[i] < B[j] & C[k]
7. a ++
8. else if B[j] < A[i] & C[k]
9. b ++
10. else if C[k] < A[i] & B[j]
11. C ++
12. return a, b, c

Q5

i) ClosestPoint () {

1. S1 = closestPoint (left half)
2. min = d
3. S2 = closestPoint (Right half)
4. Array.sort (strip, 0, size)
5. for i = 0 to (i < size)
6. for j = (i+1) to (j < size & strip[j] < min)
7. if (distance between strip[i] & strip[j] < min)
8. min = distance between (strip[i], strip[j])
9. return min.

$T(n) = 2T(n/2) + O(n \log n)$



$\therefore T(n) = T(n \times \log n \times \log n)$

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

(ii) Closest Pair $(P_1, P_2 - P_m)$ {

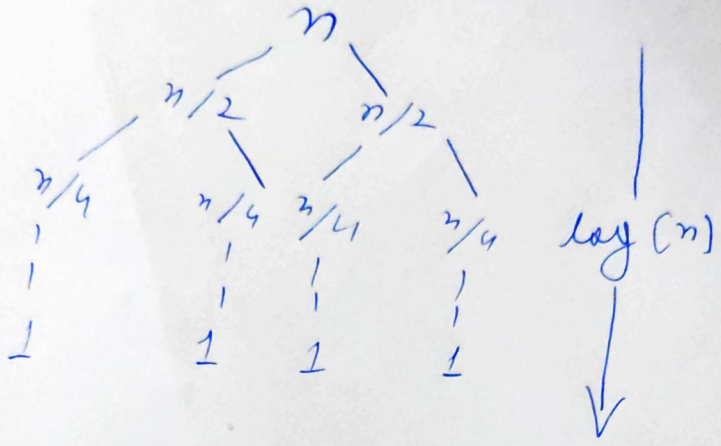
1. compute separation line L such that half the pts are on one side & half on other
2. δ_1 = ClosestPair (left half)
3. δ_2 = ClosestPair (Right half)
4. $\delta = \min(\delta_1, \delta_2)$
5. delete all pts further than δ from L
6. Sort remaining points & merge them
7. Scan points & compare distance between each pair and then

8. if there is any distance less than δ

9. update δ

10. return δ

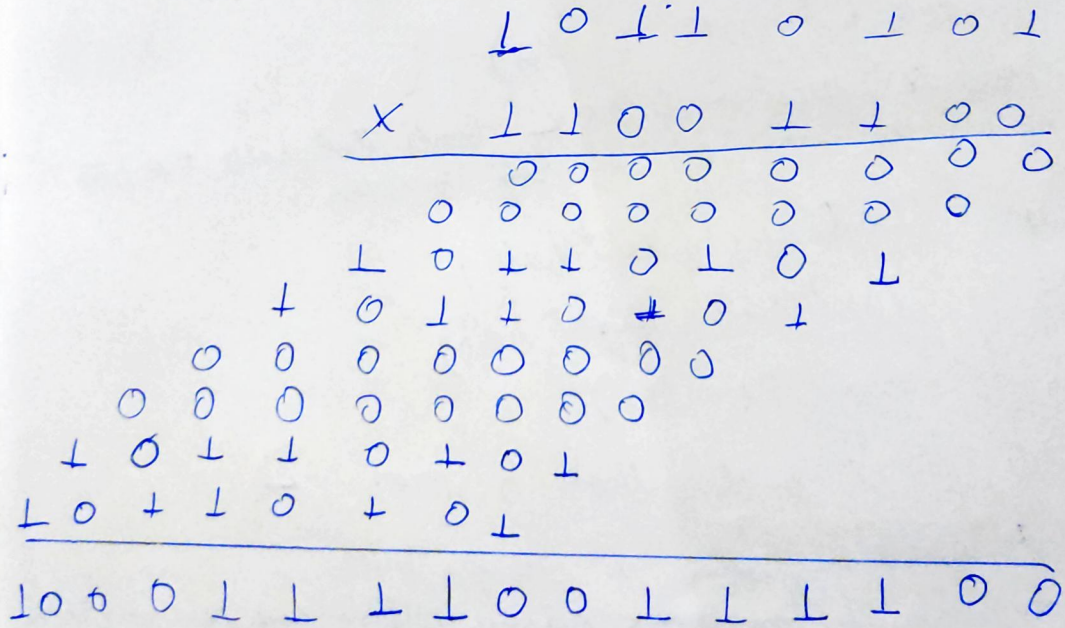
$T(n) = 2T(n/2) + O(n \log n)$



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

Q6 a = 10110101

b = 11001100



The product of a and b is

1 0 0 0 1 1 1 1 0 0 1 1 1 1 0 0

Recursive Multiply (x, y) :

1. $x = x_1 \cdot 2^{n/2} + x_0$
2. $y = y_1 + 2^{n/2} + y_0$
3. Compute $x_1 + x_0$ and $y_1 + y_0$
4. $P :=$ Recursive Multiply $(x_1 + x_0, y_1 + y_0)$
5. $x_1 y_1 =$ Recursive Multiply (x_1, y_1)
6. $x_0 y_0 =$ Recursive Multiply (x_0, y_0)
7. return $x_1 y_1 \cdot 2^n + (P - x_1 y_1 - x_0 y_0) 2^{n/2} + x_0 y_0$

Time Complexity

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + T(1 + \lceil n/2 \rceil) + \Theta(n)$$

$$T(n) = O(n^{\log 3})$$

$$T(n) = O(n^{1.585}) \approx \underline{O(n^2)}$$

Q8 $x = 10$

$P(10) = 21034$

$Q(10) = 352$

$P(x) = [4\ 3\ 0\ 1\ 2]$ (Thousands, Hundreds, Tens, Ones)

$Q(x) = [2\ 5\ 3\ 0\ 0]$

$\Rightarrow [4\ 3\ 0\ 1\ 2]\ [2\ 5\ 3\ 0\ 0]$

$\Rightarrow [2\ 0\ 0] \times [4\ 3\ 0\ 1\ 2] + [5\ 0] \times [4\ 3\ 0\ 1\ 2] + [3] \times [4\ 3\ 0\ 1\ 2]$

$\Rightarrow [0\ 0\ 2\ 0\ 4\ 3\ 6] + [0\ 0\ 7\ 1\ 5\ 0\ 1] + [8\ 6\ 0\ 2\ 4\ 0\ 0]$

$[8\ 6\ 9\ 3\ 0\ 4\ 7]$ (Ones, Tens, Hundreds, Thousand)

$= 7,403,968$