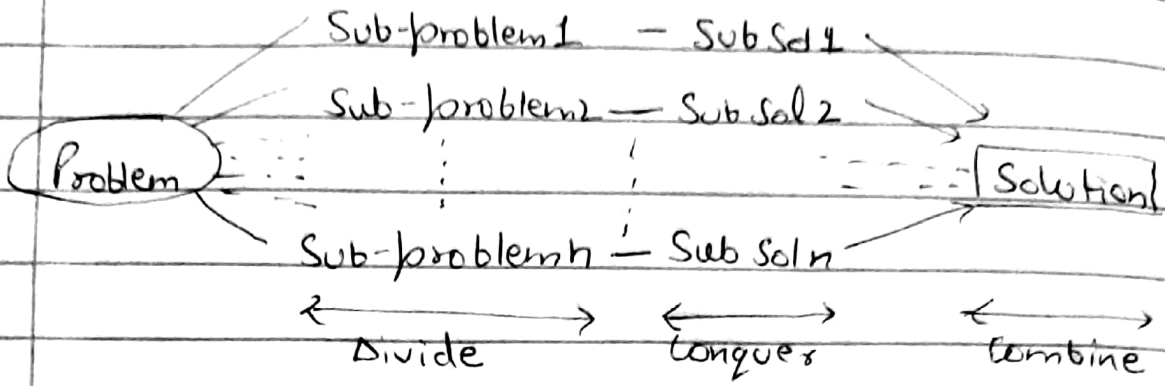
Divide and Conquer Approach

A divide and conquer method works by recursively breaking down a problem into two or more sub-problems of the same type, until these become simple enough to be solved directly.

In general, divide & conquer technique involves three steps:-

1) Divide
2) Conquer
3) Combine



Binary Search

A binary Search algorithm is a technique for finding a position of specified value within a sorted array.

Algorithm → T(n) low, mid, high } int x = Key element to be search

```
{  low = 1;
   high = hi
while (low + high)/2          — 1
{ if (A[mid] == x)            — 1
    return mid;               — 1
else if (x < A[mid])
    high = mid-1
else low = mid +1             — T(n/2)
}
return -1;
}
```

Example:

11	22	30	33	40	44	55	60	66	77	80	88	99
1	2	3	4	5	6	7	8	9	10	11	12	13

low ↓ mid ↓ high

x = 40

1) Initialize low = 1, high = 13

1 <= 13 True mid = $\frac{1+13}{2} = \frac{14}{2} = 7$

A[mid] = 55 ≠ 40

40 < 55 True mid = 7 − 1 = 6 ⇒ high

2) low = 1, high = 6

1 <= 6 True mid = $\frac{1+6}{2} = \frac{7}{2} = 3.5 = 3$

A[mid] = 30 ≠ 40

40 < 30 false

⇒ low = mid + 1; ⇒ low = 3 + 1 = 4

3) low = 4, high = 6

4 <= 6 True mid = $\frac{4+6}{2} = \frac{10}{2} = 5$

A[mid] = 40 = 40 True // search successful

return mid

** Hence, element found at index 5

Example:	10	20	30	40	50	60	70	80	90	100
index	1	2	3	4	5	6	7	8	9	10

find x = 90

1) Initialize low = 1 high = 10

1 <= 10 True mid = $\frac{1+10}{2} = \frac{11}{2} = 5.5 = 5$

A[mid] = 50 ≠ 90

90 < 50 false low = 5+1 = 6

2) low = 6, high = 10

6 <= 10 True mid = $\frac{6+10}{2} = \frac{16}{2} = 8$

A[mid] = 80 ≠ 90 false

90 < 80 false low = 8 + 1 = 9

3) low = 9, high = 10

9 <= 10 True mid = $\frac{9+10}{2} = \frac{19}{2} = 9.5 = 9$

A[mid] = 90 = 90

True // search Successful

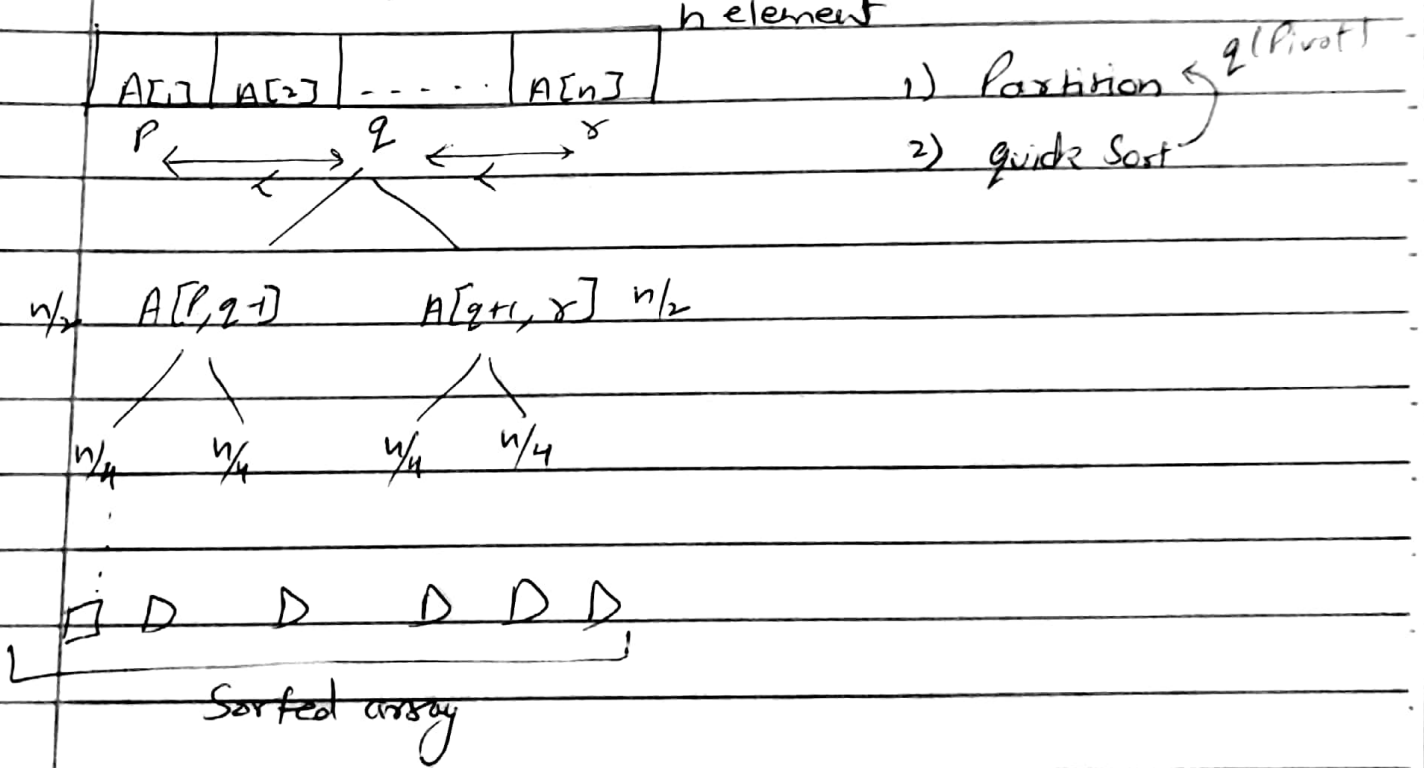
Hence, element found at index 9.

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

$$T(n) = T(n/2) + 1 \quad = ?$$

Quick Sort

Sorting is the process of arranging the given array of elements in either increasing or decreasing order.



```
Quicksort (A, P, r)
{ if (P < r)
  { q = Partition (A, P, r)
  quicksort (A, P, q-1);
  quicksort (A, q+1, r);
  }
}
Partition (A, P, r)
{  i = P-1;
  for (j = P; j <= r-1; j++)
{ if (A[j] <= A[r])
  { i++;
    exchange (A[i] & A[j])
  }
}
exchange (A[i+1] & A[r])
return (i+1);
}
```

	1	2	3	4	5	6	7	8
A	20	80	70	10	30	50	60	40
	20	10	70	80	30	50	60	40
	20	10	30	80	70	50	60	40
	20	10	30	40	70	50	60	80
	P		q-1	q	q+1			r
				Pivot				

A[1, 3] A[5, 8]

p=1, r=8

i=0 j=1, j<=r-1 => j<=8-1=7

j=1 A[1] <= A[8]

20 <= 40 True

i=1 A[1] & A[1]

j=2 A[2] <= A[8]

80 <= 40 false

j=3 A[3] <= A[8]

70 <= 40 false

j=4 A[4] <= A[8]

10 <= 40 True

i=2

A[2] & A[4]

j=5 A[5] <= A[8]

30 <= 40 True

i=3 A[3] & A[5]

j=6 A[6] <= A[8]

50 <= 40 false

j=7 A[7] <= A[8]

60 <= 40 false

exchange A[4] & A[8]

return i+1

3+1 = 4

A | 20 | 10 | 30 |
p r = q-1

A[1, 3]

20	10		30
			q

A[1, 2]

p = 1 , r = 2

i = 0 j = 1 , j <= 1

A[1] < A[2]

20 <= 10 false

exchange A[1] & A[2]

10	20
q	

return i+1;

return ~~or~~ 1;

10 20 30 40

p = 1 r = 3

i = 0

j = 1 , j <= 2

A[1] <= A[3]

20 <= 30 True

i = 1 A[1] & A[1]

j = 2

A[2] <= A[3]

10 <= 30 True

i = 2 A[2] ~~&~~ A[2]

exchange A[3] & A[3]

return 3;

	5	6	7	8
A	70	50	60	80
	p			r
				q

	5	6	7	
	70	50	60	80
				q

p = 5 , r = 8

i = 4

for(j = 5, j <= 7)

j = 5 A[5] <= A[8]

70 <= 80

i = 5 A[5] & A[5]

j = 6 A[6] <= A[8]

50 <= 80 True

i = 6 A[6] & A[6]

j=7 A[7] <= A[8]

60 <= 80 True

i=7 A[7] & A[7]

exchange A[8] & A[8]

return 8;

5	6	7
70	50	60

p r=q-1

p=5, r=7, i=4

j=5, j<=6

50	70	60

j=5 A[5] <= A[7]

70 <= 60 false

50	60	70

j=6 A[6] <= A[7]

50 <= 60 True

50	q	70
p,r	60	p,r

i=5 A[5] & A[6]

exchange A[6] & A[7]

return 6;

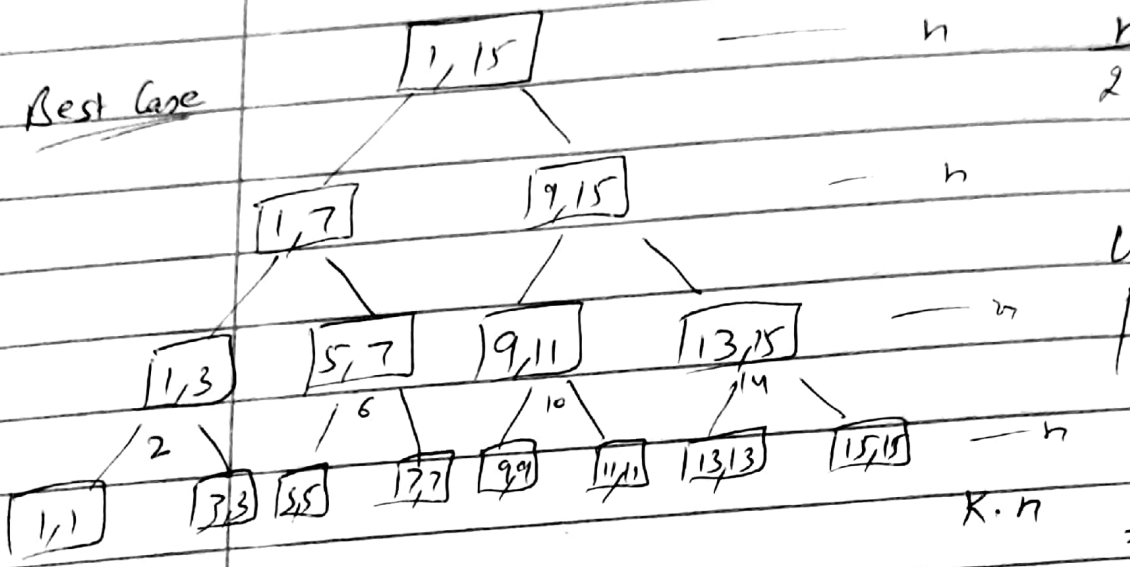
50	60	70	80

10	20	30	40	50	60	70	80

1 15

1 ←→ 7 8 9 ←→ 15

Best Case



— n $\frac{n}{2^k} = 1$

— n $n = 2^k$

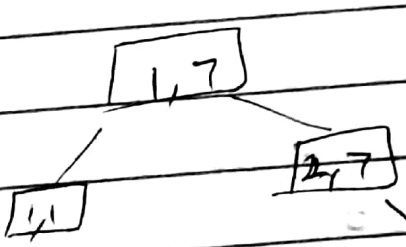
— n $\log n \quad k \log 2$

— n $k = \log_2 n$

$K \cdot n$

$\Rightarrow O(n \log n)$

worst Case



— n

$\frac{n(n+1)}{2}$

$O(n^2)$

3,7 — — (n-1)

4,7 — — (n-2)

5,7 — — (n-3)

6,7

7,7 1

$T(n) = T(n-1) + O(n)$