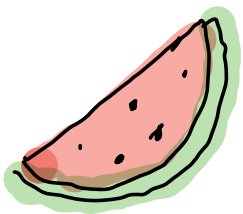
Sorting & Searching



Sorting
↳ Merge sort Algo
Searching
↳ Binary search

Sorting : Ordering in a particular order

1, 2, 3, 4, 5 → Asc
5, 4, 3, 2, 1 → desc

7, 2, 4, 9, 6 →
2, 2, 3, 3, 4

→ Selection Sort →
Insertion sort → $O(n^2)$
Bubble sort →
Quick sort
✓ Merge sort → $O(n \log n)$
Count sort

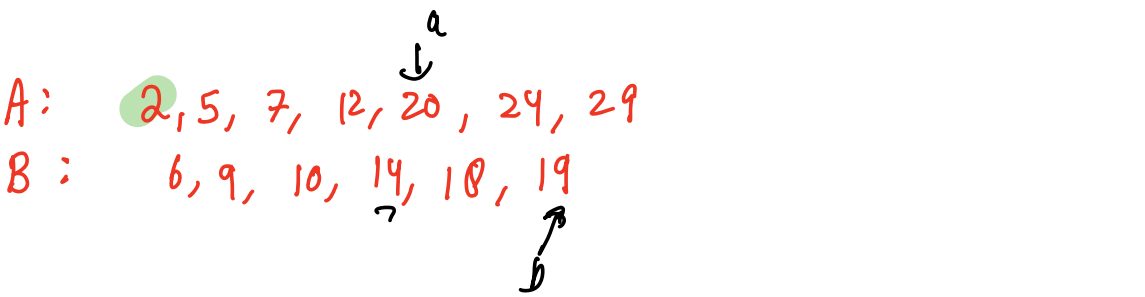
Q) Amazon WhatsApp

Given 2 sorted arrays of size N & M. Merge both & return new sorted Array

A: 2, 5, 7, 12, 20, 24, 29 ← N

B: 6, 9, 10, 14, 18, 19 ← M

C: 2, 5, 6, 7, 9, 10, 12, 14, 18, 19, 20, 24, 29 ← N+M

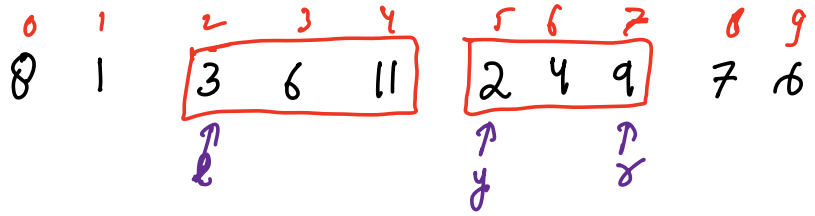


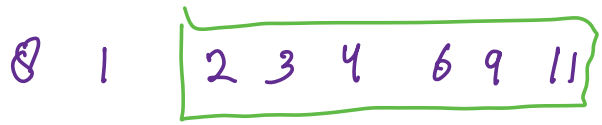

```
int[] merge ( A[] , N, B[] , M)
{
    C[N+M] → new array
    a=0, b=0, c=0
    while( a<N && b<M )
    {
        if ( A[a] < B[b] )
        {
            C[c] = A[a];
            a++;
        }
    else
```

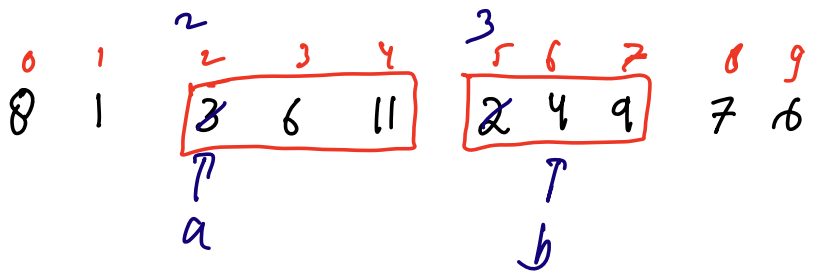
O(n+m)

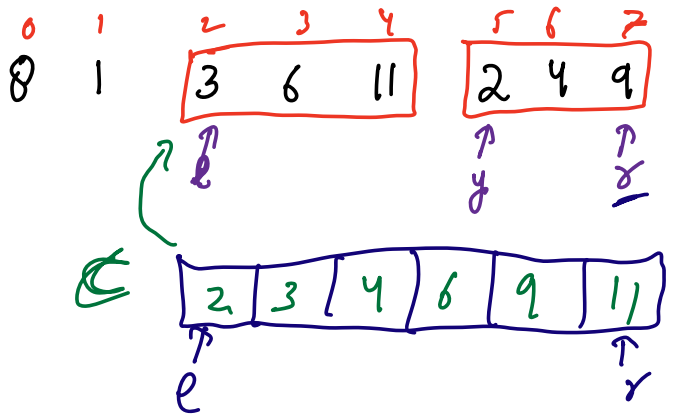
```
        }
            C[c] = B[b]
            b++;
        }
        c++;
    }
while (a < N)
    {
        C[c] = A[a];
        a++;
        c++;
    }
    while (b < M)
    {
        C[c] = B[b];
        b++;  c++;
    }
    ret C;
}
```

Q⇒ Given an array of size N & 3 indexes l, y, r.
Sort array from l → r

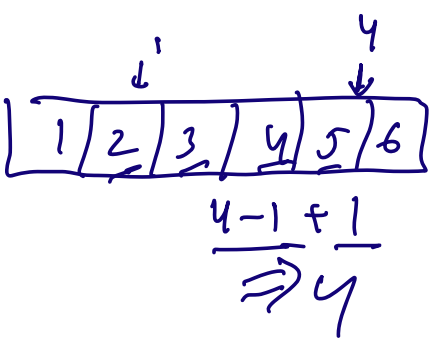








r-l+1



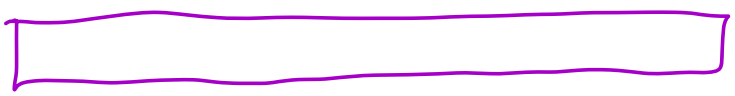

```
int[]  merge ( A[] , l, y, r )
{
    C[r-l+1]  →  new array
```

```
a = l , b = y, c = 0
while ( a < y && b ≤ r )
  {
    if ( A[a] < A [b] )
      {
        C [c] = A[a];
        a++;
      }
    else
      }
        C [c] = A [b]
        b ++;
      }
    C++;
  }
while  (a < y)
  {
    C [a]  = A[a];
    a++;
    c++;
  }
```

O(n+m)

```
while ( b <= o )
  {
    C [c] = B [b];
     b++;   c++;
  }
for( i=0;  i < ( r-l+1) ; i++)
  {
     A [i+ l] =  C [i]
  }
}
```

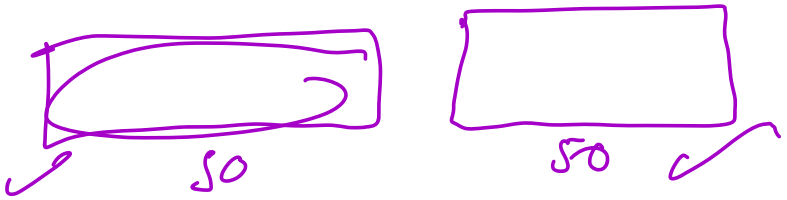
$O(n^2)$



100

$(100)^2$

$\Rightarrow 10000$ ✓



50

$(50)^2$

2500

50

$(50)^2$

2500 $= 5000$

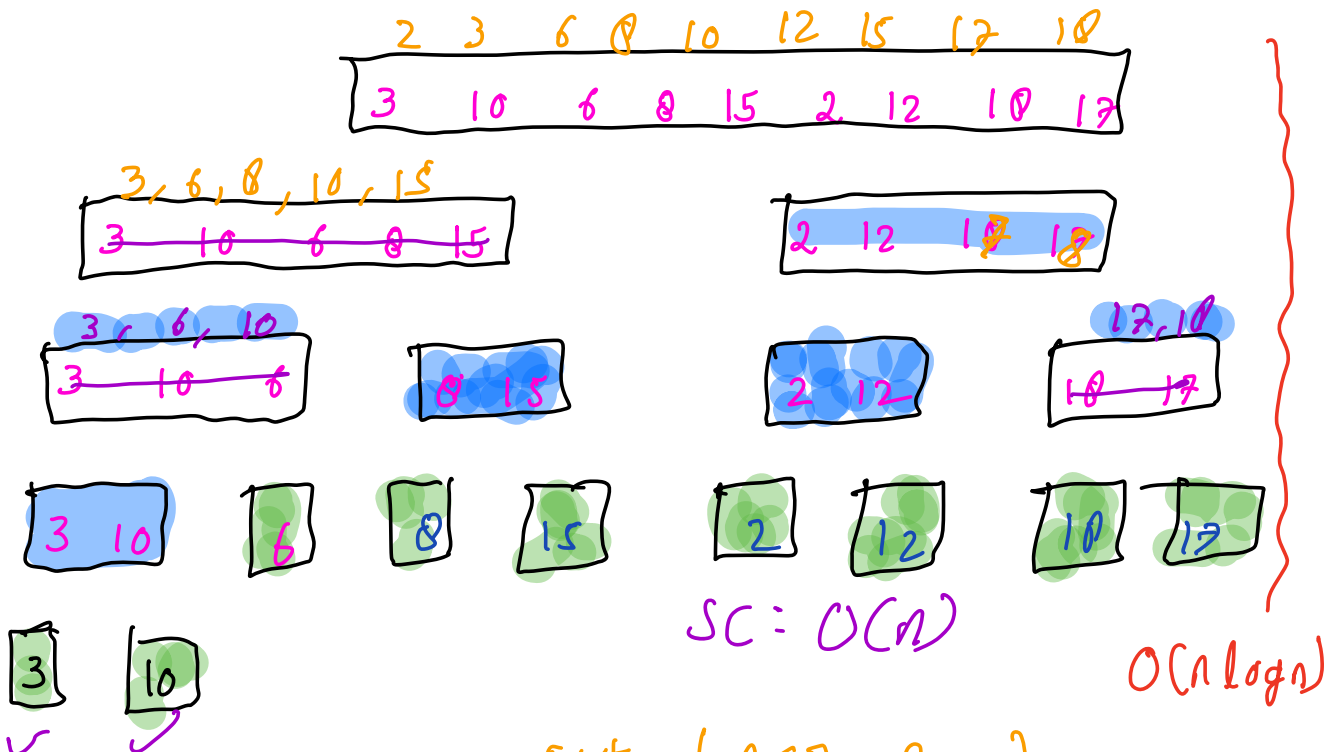
$+100$

5100 ✓

$4 * [25]^2$

$2500 \quad + 200$

2700



SC : O(n)

O(n log n)

```
mergesort (A[], l, r)
{
    if ( l == r ) ret;

    mid = (l + r) / 2;
T(n/2) → mergesort( A, l, mid);
T(n/2) → mergesort( A, mid+1, r);
O(N) → merge( A, l, mid+1, r);
```

}

$T(N) = 2\,T\left(\frac{n}{2}\right) + O(N)$ HW

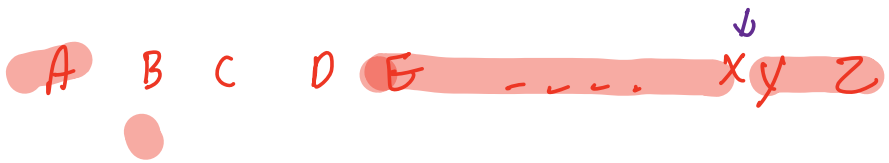
$Tc: O(n \log n)$

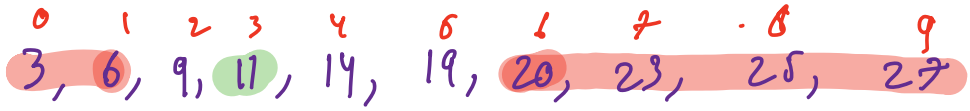
Break: 10:3

Searching : Target
Search Space

- search a word in a newspaper
  search a word in a dictionary

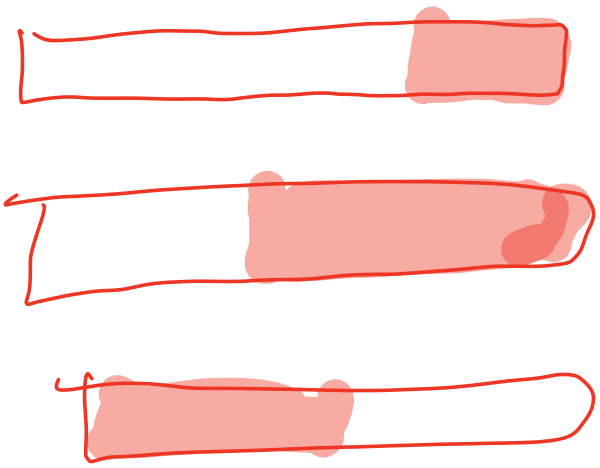
Search ph no of contact in a hand written diary
Search ph no of contact in a directory





target = 11

s	e	Rand Elem
0	9	6
0	5	1
2	5	3



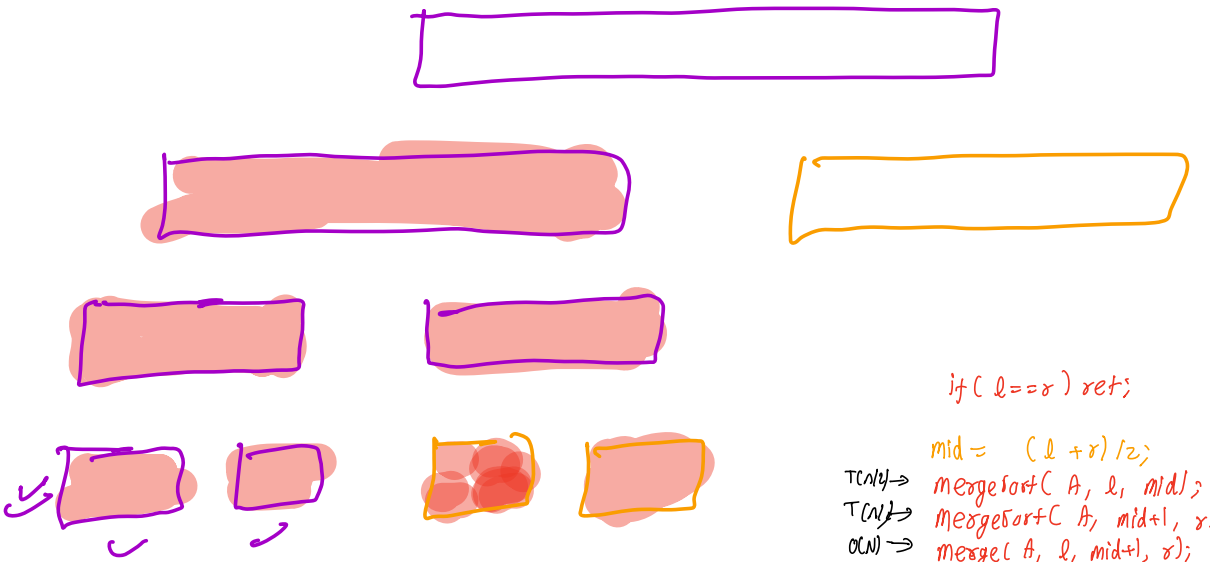
	0	1	2	3	4	5	6	7	8	9
A:	3,	6,	9,	11,	14,	19,	20,	23,	25,	27

Target	i	e	mid	A[mid]
9	0	9	4	14
12	0	3	1	6
	2	3	2	9 ✓
	3	3	3	11
	4	3		

```
Binary search ( A[] , K)
{
    l = 0 ,   r = N-1
    while ( l <= r )
    {
        mid = (l+r)/2;
        if ( A[mid] == K)
            return mid,
```

O[log n]
O(1)

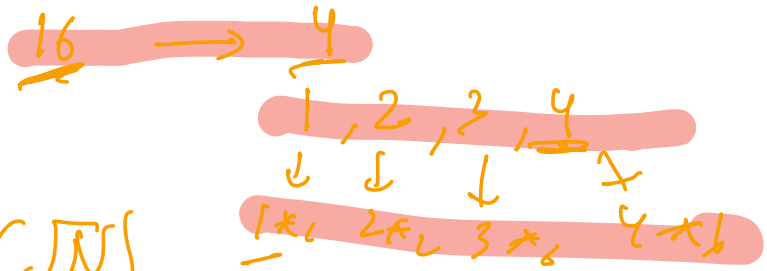
```
else if (A[mid] < K)
    l = mid+1;
else
    r = mid-1;
}
ret -1;
}
```



```
if(l==r) ret;
mid = (l+r)/2;
T(N/2) → mergesort(A, l, mid);
T(N/2) → mergesort(A, mid+1, r);
O(N) → merge(A, l, mid+1, r);
}
```

2^{2N} $O(2^{KN})$

N



$O(N)$

$O(\log N)$

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

Aggressive cuts