arr = {0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610} (-1)

target = 55 O(n) ⟷ log(n) = 4 = √16

length = 16 so, n = 16, steps = √n

Normally in binary search we divide by 2. Find the mid.

In JUMP SEARCH, we take √n steps.

step = 4, prev = 0, index = 3 (89) < 55

arr[3] < 55 arr[7] < 55 arr[11]

arr = {0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610}

arr[8] 55 < 89, 144, 233, 377, 610

21 < 55, ptr = 9, arr[9] 34 < 55, step = 4, 10 < 55, step = 0

arr[min(n, step) - 1] arr[4-1] arr[3] = 2 < 55

arr[11] 89 > 55, prev = 8

arr[7] 13 < 55

arr = {5, -8, 1, 2, -1, 4} -6

Maximum Subarray Sum :→ 6

Kadane's Algorithm O(n)

cmax = arr[0] = 5 gmax = arr[0] = 5

arr[1] = 4 cmax = 6 gmax = 6

for (i = 1; i < size; i++) {

cmax = max(arr[i], cmax + arr[i])

gmax = max(cmax, gmax);

}

return gmax;

* Merge two sorted arrays:

a1 = [1, 3, 5, 7, 9]

a2 = [2, 4, 6]

O(n)

Two pointer approach.

if a1[i] < a2[j]

arr[k] = a1[i] i++

a2[j] < a1[i]

arr[8] = {1 2 3 4 5 6 7 9}

k = 0 0 1 2 3 4 5 6 7

Merge Sort Algorithm :→

Divide & Conquer (Recursion)

l = m+1 r = e-m

= 3-0 mid

Split = 3+1 = s+e/2

log(n) = 6/2

Single ele = 3

Sorted ⇒ n

SA1 SA2

S+M = n log n

1 2 3 4 4 5 6 9 → O(n)

Important Interview Question on Strings :→

* Why are Strings in Java called immutable? Original string is unchanged.

* How can we create mutable strings in Java?

StringBuffer & StringBuilder

java.lang Which is better?

StringBuilder is more efficient when we have larger Manipulations & it is also not Thread Safe.

It can run independently in any "thread" → the smallest unit of processing.