

$\{ -5, -8, 1, 2, -1, 4 \}$ Kadane Algorithm $O(n)$

* What is the maximum subarray sum of this array?

$c_{max} = arr[0] = -8$ $c_{max} = m(-8, -3)$
 $g_{max} = arr[0] = -8$ $g_{max} = m(-8, -3)$

```

for (int i=1; i<n; i++) {
    c_{max} = max(arr[i], c_{max} + arr[i]);
    g_{max} = max(c_{max}, g_{max});
}
return g_{max};
    
```

Important Interview Questions on Binary Search & its applications: $\rightarrow (\log N)$

- * 1) Square root of a number (integral part).
 - * 2) Missing element in a sorted array.
 - * 3) First, Last & Total Occurrences of an Element in a Sorted Array.
 - 4) Peak in a Mountain Array. $[1, 2, 3, 4, 3, 2, 0]$
 - * 5) Search in a 2D Matrix
 - 6) Aggressive Cows
 - 7) Painter's Partition Problem
 - 8) Book Allocation Problem
- Square root using Binary Search ($\log N$)
- ```

int squareRoot(int n) {
 int s = 0, e = n;
 int mid = (s + e) / 2;
 while (s <= e) {
 if (mid * mid == n)
 return mid;
 if (mid * mid < n)
 s = mid + 1;
 else
 e = mid - 1;
 mid = (s + e) / 2;
 }
 return -1;
}

```

First, Last, Total

|                                         |              |                           |
|-----------------------------------------|--------------|---------------------------|
| $arr = [1, 2, 3, 3, 3, 3, 4, 5]$        | $key = 3$    | $F_O = 2$                 |
| $m = \frac{s+e}{2} = \frac{1+7}{2} = 4$ | $L_O = 5$    | $T_O = 5-2+1 = 4$         |
| $if (arr[mid] == key)$                  | $ans = mid;$ | $ans = 4$                 |
| $else if (arr[mid] > key)$              | $e = m-1;$   | $mid = \frac{s+e}{2} = 3$ |
| $else$                                  | $s = m+1;$   | $ans = 4$                 |

|                                         |              |                           |
|-----------------------------------------|--------------|---------------------------|
| $arr = [1, 2, 3, 3, 3, 3, 4, 5]$        | $key = 3$    | $F_O = 2$                 |
| $m = \frac{s+e}{2} = \frac{1+7}{2} = 4$ | $L_O = 5$    | $T_O = 5-2+1 = 4$         |
| $if (arr[mid] == key)$                  | $ans = mid;$ | $ans = 4$                 |
| $else if (arr[mid] > key)$              | $e = m-1;$   | $mid = \frac{s+e}{2} = 3$ |
| $else$                                  | $s = m+1;$   | $ans = 4$                 |

Missing Element

$arr = \{1, 2, 3, 4, 5, 6, 7\}$

$if [element = index + 1]$

$mid = \frac{s+e}{2}$

$if (arr[mid] != mid + 1) = 2$

$left \{ if (mid == 0 || arr[mid-1] == mid) = 2$

$mid = 3$

$arr[2] == 2+1 \quad e = m-1;$

$arr[2] == 3 \quad s = m+1;$

$28 - arr\_sum = 28 - 24 = 4$

Radix Sort:  $1^s, 10^s, 100^s$  Bucket Sort

M D N       $325 | 246 | 019 | 008 | 021$

① Max  $\rightarrow 325$  CS       $102 | 325 | 246 | 008 | 019 \rightarrow 11$

3 passes       $102 | 019 | 022 | 325 | 246 \rightarrow 11$

② 0-9 10 buckets       $008 | 019 | 022 | 246 | 325 \rightarrow 11$

$O(n + max)$        $0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$

\* How is the no of iterations (passes) (rounds) in the radix sort algo controlled?

(max) = 325  $\rightarrow$  no of digits = 3  $\rightarrow$  3 passes?

exp = 1000  $\rightarrow$  4  $\rightarrow$  4 passes.

for (int exp = 1; max/exp > 0; exp \*= 10)

    = 325  
 $325/10 = 32$   
 $325/100 = 3$   
 $325/1000 = 0$

Standard Template Library C++      Collections Framework Java

#include <lib-name> package java.util.lib-name;

\* Stack (vector.push-back(), queue.push-back(), priority-queue, max-heap)

\* Queue (vector.pop-front(), queue.pop-front(), max-heap)

\* List (list, DLL, forward-list, SL)

\* Map (unordered-map, map, set, multimap, multiset, map, set, multimap, multiset)

\* Set (set, multiset, unordered-set, map, set, multimap, multiset)

\* Vector

Feedback: bigbittraining.com

Session Code: 11575