Data Structure & Algorithms

Nilesh Ghule

Binary Search

input space = O(n) - array & key

aux space = l, r, i, m = k → recursive $\frac{\log n}{\log 2}$ times

S = log(n) → stack space

0	1	2	3	4	5	6	7	8
11	22	33	44	55	66	77	88	99
	r	l						

m

$2^{itr} = n$

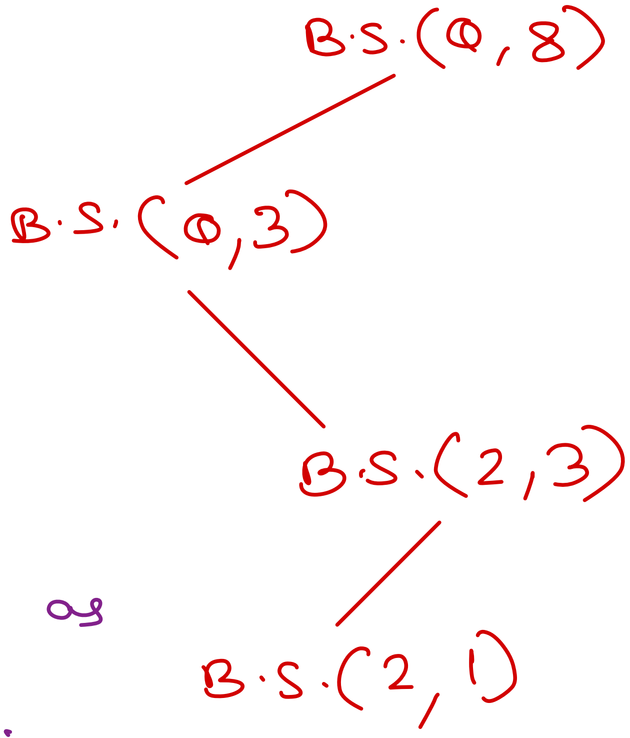
$itr = \frac{\log n}{\log 2}$

$T \propto \log n$

$O(\log n)$

T.C. is same as loop code.

But running time is little more than loop code.



invalid partition
↓
elem not cond.

```
binSearch(left, right, arr, key) {
    if(left > right)
        return -1;
    mid = (left + right)/2;
    if(key == arr[mid])
        return mid;
    if(key < arr[mid])
        i = binSearch(left, mid-1, arr, key);
    else
        i = binSearch(mid+1, right, arr, key);
    return i;
}
```

Selection Sort

5 6 3 8 2 4

i=0 2	j=1 6	5	8	3	j=5 4	→ pass 1 (5)
2	i=1 3	j=2 6	8	5	j=5 4	→ pass 2 (4)
2	3	i=2 4	j=3 8	6	j=5 5	→ pass 3 (3)
2	3	4	i=3 5	j=4 8	j=5 6	→ pass 4 (2)
2	3	4	5	i=4 6	j=5 8	→ pass 5 (1)

total = 15

input space = $O(n)$

aux space = $O(1)$

0	1	2	3	4	5
5	6	3	8	2	4

```
for(i=0; i<n-1; i++) {
    for(j=i+1; j<n; j++) {
        if (a[i] > a[j])
            swap(a[i], a[j]);
```

$itr = (n-1) + (n-2) + (n-3) + \dots + 1$

$itr = \frac{n(n-1)}{2}$

$T \propto \frac{n(n-1)}{2}$

$T \propto n^2 - n$

$T \propto n^2$

$\boxed{O(n^2)}$

$n >> 1$, $n^2 >>>> n$

lower order terms are negligible (can be ignored)

↳ theory of approximation

3

3

Bubble Sort

5 6 3 8 2 4

j=0	j=1	j=2	j=3	j=4		
5	3	6	2	4	8	pass 1 (5)
3	5	2	4	6	8	pass2 (5)
3	2	4	5	6	8	pass 3 (5)
2	3	4	5	6	8	pass 4(5)
2	3	4	5	6	8	pass 5(5)

total : 25

0	1	2	3	4	5
5	6	3	8	2	4

```
for(i=0; i<n-1; i++){
    for(j=0; j<n-1; j++){
        if(a[j] > a[j+1])
            swap(a[j], a[j+1]);
    }
}
```

$itr = (n-1)*(n-1)$

$itr = n^2 - 2n + 1$

$T \propto n^2 - 2n + 1$

$T \propto n^2$

$\boxed{O(n^2)}$

$n >> 1$

$n^2 >>>> n$

Bubble Sort - improved

5 6 3 8 2 4

j=0	j=1	j=2	j=3	j=4		
5	3	6	2	4	8	pass 1 (5)
j=0 3	j=1 5	j=2 2	j=3 4	6	8	pass 2 (4)
j=0 3	j=1 2	j=2 4	5	6	8	pass 3 (3)
j=0 2	j=1 3	4	5	6	8	pass 4 (2)
j=0 2	3	4	5	6	8	pass 5 (1)

total : 15

0	1	2	3	4	5
5	6	3	8	2	4

$itr = (n-1) + (n-2) + \dots + 1$

$itr = n(n-1)/2$

$T \propto n^2 - n$

$T \propto n^2$

$n >> 1$

$n^2 >>>>> n$

$\boxed{O(n^2)}$

```
for(i=0; i<n-1; i++){
    for(j=0; j<n-1-i; j++){
        if(a[j] > a[j+1])
            swap(a[j], a[j+1]);
    }
}
```

Bubble Sort - further improved

3 4 5 6 7 8 pass1 (5)

3 4 5 6 7 8 pass 2 (4)

total = 9

~~3 4 5 6 7 8~~

0	1	2	3	4	5
3	4	5	6	8	7

if in any pass, no swapping is done; it means array is sorted. So further passes are not required.

2 3 4 5 6 7 → pass 1 (5)

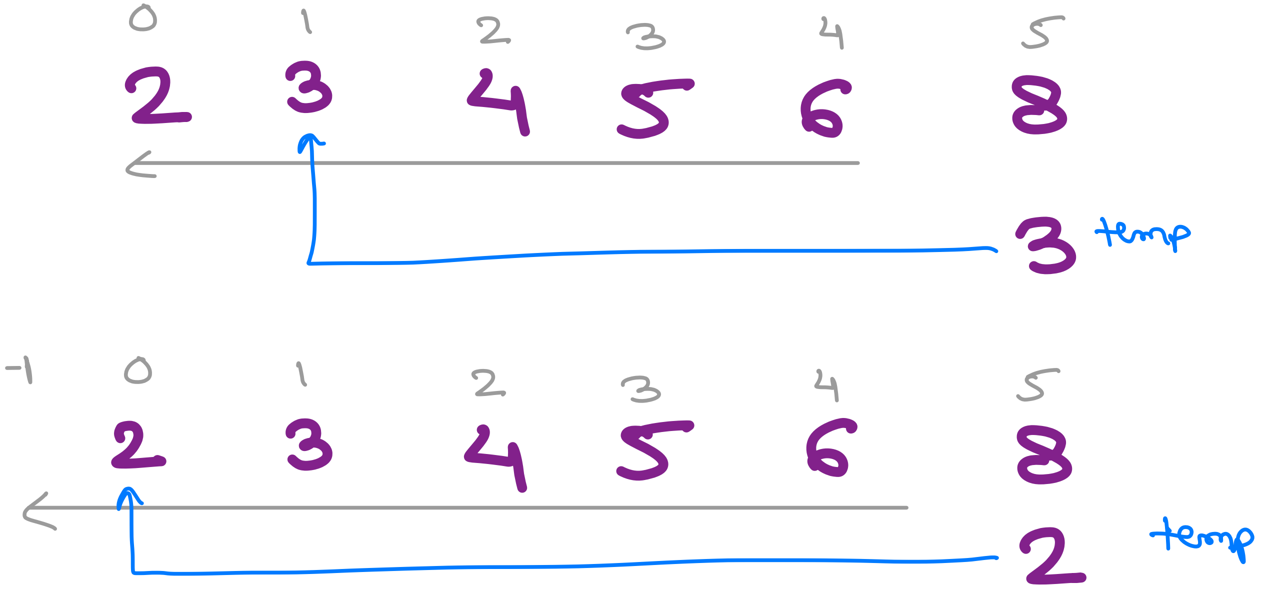
Best case : itr = n-1

$T \propto n-1$

$O(n)$

Insertion Sort

6 5 3 8 2 4



0	1	2	3	4	5
6	5	3	8	2	4

-1	0	1	2	3	4	5
	5	i=1 6				
	3	5	i=2 6			
	3	5	6	i=3 8		
	2	3	5	6	i=4 8	
	2	3	4	5	6	i=5 8

```
for(i=1; i<n ; i++){
    temp= a[i];
    for(j=i-1; j>=0 && a[j] > temp; j--)
        a[j+1] = a[j];
    a[j+1] = temp;
}
```

H.W.

① Calculate time complexity - general case

② Calculate time complexity - best case

1, 2, 3, 4, 5, 6

SUNBEAM

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>