Vaibhav
Bhaskar
11912060
IT

* Bubble sort *

Algorithm

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

1) **Best case**

$n=5$ 1, 2, 3, 4, 5

Iterations $= 4 \times 1 = 4$

$T \approx O(n)$

2) **worst case**

$n=5$, 5, 4, 3, 2, 1

Iterations $= (4+3+2+1)$ $T \approx O(n^2)$

3) **Average case**

$n=5$ A = 3, 4, 2, 1, 5

$T \approx O(n^2)$

Since no extra Element is required, Space complexity $\approx O(1)$

$\Rightarrow$ Inplace Algorithm

* Quick sort *

Algorithm

```
qs(arr, l, h)
{ if (l≤h) return;
  p= partition (arr, l, h)
  qs(arr, l, p-1)
  qs(arr, p+1, h)
}
```

$T(n) = T(k) + T(n-k-1) + cn$

Best case : when array is divided into two subarrays of equal size.

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}-1\right) + Cn$$

when $n \to \infty$, $\frac{n}{2} - 1 \approx \frac{n}{2}$

$$T(n) = 2T\left(\frac{n}{2}\right) + Cn$$

By Master theorem

$$\boxed{T(n) \approx O(n \log n)}$$

worst case: Array in ascending or descending order.

$k = 0$

$$T(n) = T(n-1) + Cn$$
$$T(n-1) = T(n-2) + C(n-1)$$
$$\vdots$$
$$T(n) = C(n + n-1 + n-2 + \cdots 1)$$
$$\boxed{T_n \approx O(n^2)}$$

Average case: $T(n) \approx O(n \log_b n)$ $b > 1$

Space complexity : There is no extra space in terms of variables or arrays; constant. But stack activation records consume memory.

Avg. case: $\approx O(\log n)$ **Best case**: $\approx O(\log n)$

worst case $\approx O(n)$ → Tree is skewed

Even though, it consumes stack space, MFT press has considered Merge sort as **Inplace Algorithm**

Time complexity Analysis

	Best	worst	Average
Merge	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
quick	$O(n \log n)$	$O(n^2)$	$O(n \log n)$
Bubble	$O(n)$	$O(n^2)$	$O(n^2)$
Insertion	$O(n)$	$O(n^2)$	$O(n^2)$