

UNIT II – Brute Force and Divide and Conquer

- **Brute Force Design Technique**
 - **Selection Sort**
 - **Bubble Sort**
 - Sequential Search
 - Closest pair and Convex hull problem
 - Travelling Salesman problem
 - Knapsack problem
 - Assignment problem

Brute Force Design Technique

- General problem solving technique
- Straight forward approach
- Every possibilities
- Test and error
- Example : 4 digit pattern lock

Try for all the possibilities – 0001,0002,0003,..... – in worst case 10^4

Selection Sort

Compares the 1st element with all the elements of list and finds the smallest element and swap

Process continues until the list is sorted

Example: $n=7$, i loop $\square$ $n-1=6$, j loop $\square$ $n-2=5$

	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
Pass	89	45	68	90	29	34	17
0	17	45	68	90	29	34	89
1	17	29	68	90	45	34	89
2	17	29	34	90	45	68	89
3	17	29	34	45	90	68	89
4	17	29	34	45	68	90	89
5	17	29	34	45	68	89	90

Selection Sort

Algorithm **ALGORITHM** *SelectionSort(A[0..n - 1])*
 //Sorts a given array by selection sort
 //Input: An array A[0..n - 1] of orderable elements
 //Output: Array A[0..n - 1] sorted in nondecreasing order
 for $i \leftarrow 0$ **to** $n - 2$ **do**
 $min \leftarrow i$
 for $j \leftarrow i + 1$ **to** $n - 1$ **do**
 if $A[j] < A[min]$ $min \leftarrow j$
 swap $A[i]$ and $A[min]$

Analysis

1. Input size - n
2. Basic operation – Key Comparison $A[j] < A[min]$
3. Count of basic operation – summation formulas - $O(n^2)$

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} [(n-1) - (i+1) + 1] = \sum_{i=0}^{n-2} (n-1-i).$$

1. No of Swap operations – $O(n)$
2. Efficiency – worst / Best / Average

Bubble Sort

- Compare the adjacent elements of list and swap if they are out of order
- Doing it repeatedly will bubble up largest element to the last position of the list
- **Example, $n=7$, $n-2=5$, $n-2-i$**

i	Pass j	89	45	68	90	29	34	17
0	0	45	89	68	90	29	34	17
	1	45	68	89	90	29	34	17
	2	45	68	89	90	29	34	17
	3	45	68	89	29	90	34	17
	4	45	68	89	29	34	90	17
	5	45	68	89	29	34	17	90

i	Pass j	45	68	89	29	34	17	90
1	0	45	68	89	29	34	17	90
	1	45	68	89	29	34	17	90
	2	45	68	29	89	34	17	90
	3	45	68	29	34	89	17	90
	4	45	68	29	34	17	89	90

i	Pass j	45	68	29	34	17	89	90
2	0	45	68	29	34	17	89	90
	1	45	29	68	34	17	89	90
	2	45	29	34	68	17	89	90
	3	45	29	34	17	68	89	90

i	Pass j	45	29	34	17	68	89	90
3	0	29	45	34	17	68	89	90
	1	29	34	45	17	68	89	90
	2	29	34	17	45	68	89	90

i	Pass j	29	34	17	45	68	89	90
4	0	29	34	17	45	68	89	90
	1	29	17	34	45	68	89	90

i	Pass j	29	17	34	45	68	89	90
5	0	17	29	34	45	68	89	90

Bubble Sort

Algorithm

```
ALGORITHM BubbleSort(A[0..n-1])
//Sorts a given array by bubble sort
//Input: An array A[0..n-1] of orderable elements
//Output: Array A[0..n-1] sorted in nondecreasing order
for i ← 0 to n-2 do
    for j ← 0 to n-2-i do
        if A[j+1] < A[j] swap A[j] and A[j+1]
```

Analysis

1. Input size ..
2. Basic operation – Key Comparison $A[j] < A[j+1]$
3. Count of basic operation – **summation formulas** - $O(n^2)$

$$\begin{aligned} C(n) &= \sum_{i=0}^{n-2} \sum_{j=0}^{n-2-i} 1 = \sum_{i=0}^{n-2} [(n-2-i) - 0 + 1] \\ &= \sum_{i=0}^{n-2} (n-1-i) = \frac{(n-1)n}{2} \in \Theta(n^2). \end{aligned}$$

1. No . of Swap is $O(n^2)$
2. Efficiency – worst / Best / Average

BUBBLE SORT

Count of basic operation

$$\begin{aligned}
 C(n) &= \sum_{i=0}^{n-2} \sum_{j=0}^{n-2-i} 1 \\
 &= \sum_{i=0}^{n-2} (n-2-i) - 0 + 1 \\
 &= \sum_{i=0}^{n-2} n-2-i+1 \\
 &= \sum_{i=0}^{n-2} (n-1)-i
 \end{aligned}$$

$$= (n-1) \sum_{i=0}^{n-2} 1 - \sum_{i=0}^{n-2} i$$

$$= (n-1) (n-2-0+1) - \sum_{i=0}^{n-2} i$$

$$= (n-1) (n-1) - \sum_{i=0}^{n-2} i$$

Summation formula

$$\sum_{i=1}^u 1 \Rightarrow u - 1 - 1 \quad (S_1)$$

$$\sum_{i=1}^u i \Rightarrow \frac{n(n+1)}{2} \quad (S_2)$$

$$= (n-1)^2 - \sum_{i=0}^{n-2} 1$$

$\Downarrow \frac{n(n+1)}{2}$ Here $n = n-2$

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

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

$$\frac{2(n^2 + 1 - 2n) - (n^2 - n - 2n + 2)}{2}$$

$$\frac{2\cancel{n^2} + 2 - 4n - \cancel{n^2} + n + 2n - 2}{2} \Rightarrow \frac{n^2 - n}{2} \Rightarrow \frac{n(n-1)}{2}$$

$\sim \frac{1}{2}n^2$

Insertion Sort

- Decrease and Conquer – Decrease the list and then arrange the elements
- Consider the list 1st element as sorted and the remaining elements as unsorted list
- Now arrange the elements between sorted and unsorted list
- **Example**

89		45	68	90	29	34	17					
45		89		68	90	29	34	17				
45		68		89		90	29	34	17			
45		68		89		90		29	34	17		
29		45		68		89		90		34	17	
29		34		45		68		89		90		17
17		29		34		45		68		89		90

Insertion Sort

Algorithm

ALGORITHM *InsertionSort*($A[0..n-1]$)

//Sorts a given array by insertion sort

//Input: An array $A[0..n-1]$ of n orderable elements

//Output: Array $A[0..n-1]$ sorted in nondecreasing order

for $i \leftarrow 1$ **to** $n-1$ **do**

$v \leftarrow A[i]$

$j \leftarrow i-1$

while $j \geq 0$ **and** $A[j] > v$ **do**

$A[j+1] \leftarrow A[j]$

$j \leftarrow j-1$

$A[j+1] \leftarrow v$

Insertion Sort

Analysis

1. Input size - n
2. Basic operation – Key Comparison $A[j] > v$
3. Count of basic operation – summation formulas - $O(n^2)$

$$C_{\text{worst}}(n) = \sum_{i=1}^{n-1} \sum_{j=0}^{i-1} 1 = \sum_{i=1}^{n-1} i = \frac{(n-1)n}{2} \in \Theta(n^2).$$

1. Efficiency – worst / Best / Average