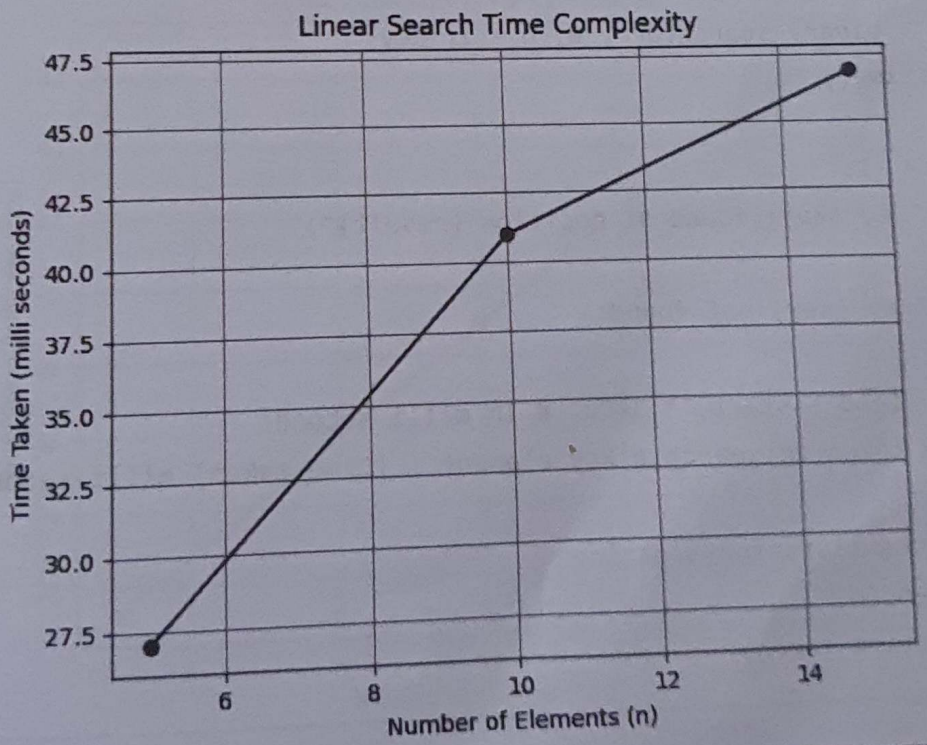
Output

```
Enter the number of runs: 3
Enter the number of elements: 15
Enter the elements of an array: 150 140 130 120 110 100 90 80 70 60 50 40 30 20 10

Enter the key element to be searched: 10
Key 10 found at position 14
Time taken to search a key element = 46.87952995300293 milli seconds

Enter the number of elements: 10
Enter the elements of an array: 100 90 80 70 60 50 40 30 20 10
Enter the key element to be searched: 10
Key 10 found at position 9
Time taken to search a key element = 40.95292091369629 milli seconds

Enter the number of elements: 5
Enter the elements of an array: 50 40 30 20 10
Enter the key element to be searched: 10
Key 10 found at position 4
Time taken to search a key element = 26.984214782714844 milli seconds
```



Output

Run 1:

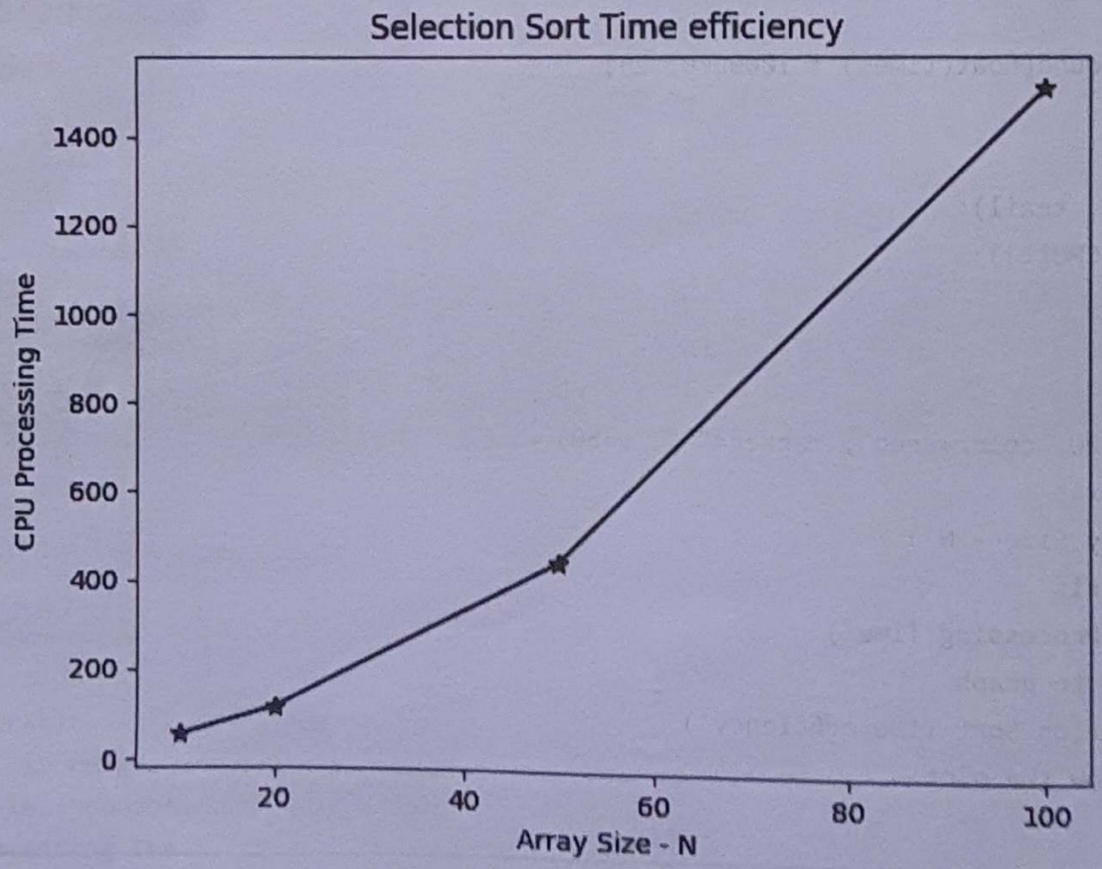
```
Enter the number of disks: 2
Sequence is:
Move Disk 1 S->T
Move Disk 2 S->D
Move Disk 1 T->D
The Number of Moves: 3
```

Run 2:

```
Enter the number of disks: 3
Sequence is:
Move Disk 1 S->D
Move Disk 2 S->T
Move Disk 1 D->T
Move Disk 3 S->D
Move Disk 1 T->S
Move Disk 2 T->D
Move Disk 1 S->D
The Number of Moves: 7
```

Output

```
Enter no. of trails: 4
-----> TRAIL NO: 1
Enter number of elements: 10
Sorted Array:
[6, 7, 15, 17, 27, 29, 32, 36, 39, 46]
-----> TRAIL NO: 2
Enter number of elements: 20
Sorted Array:
[6, 8, 8, 11, 13, 13, 14, 18, 19, 20, 21, 28, 29, 32, 33, 35, 36, 44, 44, 44]
-----> TRAIL NO: 3
Enter number of elements: 50
Sorted Array:
[2, 3, 5, 5, 6, 6, 6, 7, 8, 9, 9, 10, 10, 11, 14, 15, 17, 17, 20, 20, 20, 21, 22, 22, 24, 25,
26, 27, 28, 28, 29, 32, 33, 36, 38, 38, 39, 40, 40, 41, 41, 41, 42, 43, 43, 44, 44, 46, 49, 49]
-----> TRAIL NO: 4
Enter number of elements: 100
Sorted Array:
[1, 1, 2, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 9, 9, 9, 12, 12, 13, 13, 14, 15, 15, 16, 16, 16, 18, 18,
19, 20, 20, 20, 21, 21, 21, 21, 22, 22, 23, 23, 24, 24, 24, 24, 24, 25, 25, 25, 26, 26, 26, 27,
27, 30, 30, 31, 31, 32, 32, 33, 34, 34, 34, 34, 35, 36, 36, 36, 36, 37, 37, 38, 39, 39, 39, 39,
39, 40, 40, 40, 41, 41, 41, 42, 42, 42, 43, 44, 44, 44, 44, 45, 45, 46, 46, 47, 47, 48, 49, 49]
N CPU
10 54.7
20 117.5
50 446.2
100 1509.9
```



Output	

```
Enter the value of a and n: 2 8
Result using brute force: 256
Result using divide and conquer: 256
```

Output

```
Enter the value of n: 5
Enter the value of k: 2
Binomial Coefficient (Brute Force): 10
Binomial Coefficient (Dynamic Programming): 10
```

Output	Note : Refer Chapter 7. P.No 7.34, Let us considert Example 1 for Inputs.

```
Enter the number of vertices: 4
Enter the cost matrix row by row (space-separated):
[Enter 99999 for Infinity]
[Enter 0 for cost(i,i)]
0 99999 2 99999
3 0 99999 99999
99999 5 0 1
6 99999 99999 0
The following matrix shows the shortest distances between every pair of vertices
      0      7      2      3
      3      0      5      6
      7      5      0      1
      6     13      8      0
```

```
[Let us evaluate value of 2x^3 - 6x^2 + 2x - 1 for x = 3, co-efficients are {2, -6, 2, -1}]
Enter the degree of the polynomial: 3
Enter the coefficients from highest degree to lowest:
2
-6
2
-1
Enter the value of x: 3
Brute force result: 5.00, time used: 0.000000 seconds
Horner's rule result: 5.00, time used: 0.000000 seconds
```

Output

```
Run 1:
Enter the text: SKYWARDPUBLISHERS
Enter the pattern: PUB
Found pattern at index 7

Run 2:
Enter the text: MALAYALAM
Enter the pattern: LA
Found pattern at index 2
Found pattern at index 6
```

Output

[**Note** : We will consider the directed acyclic graph given in Chapter 4, Page No 4.9, Example 1. The adjacency cost matrix for that graph will be given as an input to this program to get the topological ordering of vertices]

Run 1:

```
Enter the number of vertices: 5
Enter the cost matrix (row by row):
0 0 1 0 0
0 0 1 0 0
0 0 0 1 1
0 0 0 0 1
0 0 0 0 0
The topological order is:
  1  2  3  4  5
```

[**Note** : We will consider the directed acyclic graph given in Chapter 4, Page No 4.13, Example 2. The adjacency cost matrix for that graph will be given as an input to this program to get the topological ordering of vertices]

```
Enter the number of vertices: 7
Enter the cost matrix (row by row):
0 0 1 0 0 0 0
0 0 0 1 0 1 0
0 0 0 0 0 0 0
0 0 1 0 0 0 0
0 0 1 0 0 0 0
0 0 0 0 0 0 0
1 0 0 0 0 1 0
The topological order is:
  2  4  5  7  1  3  6
```

Note : Every directed acyclic graph may have one or more topological orderings.

Output

[**Note** : We will consider the directed graph given in Chapter 7, Page No 7.26, The adjacency cost matrix for that graph will be given as an input to this program to get the Transitive closure of Directed graph]

```
Enter the number of vertices: 4
Enter the adjacency cost matrix:
0 1 0 0
0 0 0 1
0 0 0 0
1 0 1 0
The transitive closure of the graph is:
1 1 1 1
1 1 1 1
0 0 0 0
1 1 1 1
```