LAB 03 Sorting:

Task-1:

Create an array of 50 random integers, and apply the bubble sort algorithm. Do the following

activities:

Note: You can generate random numbers using (Random Number List 1-100 | Number Generator).

1. Display the sorted array

2. Display the number of comparisons performed by the inner array

3. Now change the size of the array to 100 random numbers and repeat point numbers 1 and

2.

4. Take the completely sorted array of 50 numbers and repeat point numbers 1 and 2.

5. Can we reduce the number of comparisons for the sorted array? How? Write a code for it.

and repeat point numbers 1 and 2.

**Key Points**:

Bubble Sort, the two successive strings arr[i] and arr[i+1] are exchanged whenever arr[i]> arr[i+1]. The larger values sink to the bottom and hence called sinking sort. At the end of each pass, smaller values gradually “bubble” their way upward to the top and hence called bubble sort.

Task-2:

Use the same array of 50 random integers as you did in task 1, and apply the selection sort

algorithm. Do the following activities:

1. Display the sorted array

2. Display the number of comparisons performed by the inner array.

3. Now use the same array of 100 random integers as you did in task 1, and repeat point

numbers 1 and 2.

4. Check for your understanding, which algorithm is performing better: bubble sort or

selection sort?

5. Take the completely sorted array of 50 numbers and repeat point numbers 1 and 2.

6. Is it possible to reduce the number of comparisons for the sorted array?

**Key Points**:

void selectionSort (int \*array, int size) {

Find the smallest element in the array and exchange it with the element in the first position.

Find the second smallest element in the array and exchange it with the element in the second position.

Continue this process until done.

}

Task -3:

Use the same array of 50 random integers as you did in task 1, and apply the insertion sort

algorithm. Do the following activities:

1. Display the sorted array

2. Display the number of comparisons performed by the algorithm.

3. Now use the same array of 100 random integers as you did in task 1, and repeat point

numbers 1 and 2.

4. Check for your understanding, which algorithm is performing better: bubble sort, selection

sort, or insertion sort?

5. Now generate partially sorted array of 50 numbers, and repeat point numbers 1 and 2.

6. Take the completely sorted array of 50 numbers and repeat point numbers 1 and 2

7. Does insertion sort take advantage of already sorted data?

**Key Points**:

void insertion Sort (int \*array, int size) {

Choose the second element in the array and place it in order with respect to the first element.

Choose the third element in the array and place it in order with respect to the first two elements.

Continue this process until done.

Insertion of an element among those previously considered consists of moving larger elements one position to the right and then inserting the element into the vacated position

}

Task 4:

Given an array arr [ ] of length N consisting cost of N toys and an integer K the amount with

you. The task is to find maximum number of toys you can buy with K amount.

Example 1:

Input:

N = 7

K = 50

arr [] = {1, 12, 5, 111, 200, 1000, 10}

Output: 4

Explanation: The costs of the toys

you can buy are 1, 12, 5 and 10.

Example 2:

Input:

N = 3

K = 100

arr [] = {20, 30, 50}

Output: 3

Explanation: You can buy all toys

Task 5:

Given a string s, sort it in decreasing order based on the frequency of the characters. The frequency of a character is the number of times it appears in the string. Return the sorted string. If there are multiple answers, return any of them.

Example 1:

Input: s = "tree"

Output: "eert"

Explanation: 'e' appears twice while 'r' and 't' both appear once. So 'e' must appear before both 'r' and 't'. Therefore "eetr" is also a valid answer.

Example 2:

Input: s = "cccaaa"

Output: "aaaccc"

Explanation: Both 'c' and 'a' appear three times, so both "cccaaa" and "aaaccc" are valid answers. Note that "cacaca" is incorrect, as the same characters must be together.

Example 3:

Input: s = "Aabb"

Output: "bbAa"

Explanation: "bbaA" is also a valid answer, but "Aabb" is incorrect. Note that 'A' and 'a' are treated as two different characters.

Task 6:

Imagine you're managing a library, and you need to arrange books based on their due dates. In this scenario, what sorting algorithm might you consider to efficiently organize the books in ascending order of their due dates?

Task 7:

Picture yourself adding books to a shelf one by one. As each book is placed, you want to ensure they are in order based on their due dates. What sorting strategy could you employ to achieve this organized arrangement as you add each book?