Name:-Aakash Bhatt
Sap-id:- 500124633

Data Structure

Lab-10

Submitted by:

Aakash Bhatt

500124633

(2023-2024)

Submitted to:

Pankaj Sir

Name:-Aakash Bhatt
Sap-id:- 500124633

WAP to populate an array of 'n' elements using a random function. Share the time complexity for all the experiments, n should be large enough to see the difference in execution.

1. Implement Insertion sort in the above data set.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// Function to generate random elements in the array
void populateArrayRandomly(int arr[], int n)
{
    srand(time(NULL)); // Seed for random number generation
    for (int i = 0; i < n; i++)
    {
        arr[i] = rand() % 100; // Generate random numbers between 0 and
99
    }
}

void insertionSort(int arr[], int n)
{
    int i, j;
    for (i = 1; i < n; i++)
    {
        int temp = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > temp)
        {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = temp;
    }
}

int main()
{
    int n = 10; // Choose a sufficiently large value for n
    int arr[n];

    // Populate array with random elements
    populateArrayRandomly(arr, n);

    printf("Original array: ");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
```

Name:-Aakash Bhatt
Sap-id:- 500124633

```
    }

    // Perform insertion sort
    insertionSort(arr, n);

    printf("\nSorted array using Insertion Sort algorithm: ");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }

    printf("\n");

    return 0;
}
```

```
PS D:\MCA\MCA-DSA\LAB-10> gcc .\Question1.c
PS D:\MCA\MCA-DSA\LAB-10> .\a.exe
Original array: 1 17 26 50 87 82 63 83 42 45
Sorted array using Insertion Sort algorithm: 1 17 26 42 45 50 63 82 83 87
PS D:\MCA\MCA-DSA\LAB-10>
```

Implement Selection sort in the above data set.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

void selectionSort(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n - 1; i++)
    {
        int min = i;
        for (j = i + 1; j < n; j++)
        {
            if (arr[j] < arr[min])
            {
                min = j;
```

Name:-Aakash Bhatt
Sap-id:- 500124633

```
            }
        }
        // Swap the elements
        if (min != i)
        {
            swap(&arr[i], &arr[min]);
        }
    }
}

// Function to generate random elements in the array
void populateArrayRandomly(int arr[], int n)
{
    srand(time(NULL)); // Seed for random number generation
    for (int i = 0; i < n; i++)
    {
        arr[i] = rand() % 100; // Generate random numbers between 0 and
99
    }
}

int main()
{
    int n = 10; // Choose a sufficiently large value for n
    int arr[n];

    // Populate array with random elements
    populateArrayRandomly(arr, n);

    printf("Original array: ");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }

    // Perform selection sort
    selectionSort(arr, n);

    printf("\nSorted array using Selection Sort algorithm: ");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }

    printf("\n");

    return 0;
}
```

Name:-Aakash Bhatt
Sap-id:- 500124633

```
PS D:\MCA\MCA-DSA\LAB-10> gcc .\Question2.c
PS D:\MCA\MCA-DSA\LAB-10> .\a.exe
Original array: 20 64 10 60 35 29 21 63 6 8
Sorted array using Selection Sort algorithm: 6 8 10 20 21 29 35 60 63 64
PS D:\MCA\MCA-DSA\LAB-10>
```

Implement Quicksort in the above data set.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

int partition(int a[], int low, int high)
{
    int pivot = a[low];
    int i, j;
    i = low;
    j = high;
    while (i < j)
    {
        do
        {
            i++;
        } while (a[i] <= pivot);

        do
        {
            j--;
        } while (a[j] > pivot);

        if (i < j)
        {
            swap(&a[i], &a[j]);
        }
    }
    swap(&a[low], &a[j]);
    return j;
}

void quicksort(int a[], int low, int high)
```

Name:-Aakash Bhatt
Sap-id:- 500124633

```
{
    if (low < high)
    {
        int j = partition(a, low, high);
        quicksort(a, low, j);
        quicksort(a, j + 1, high);
    }
}

// Function to generate random elements in the array
void populateArrayRandomly(int a[], int n)
{
    srand(time(NULL)); // Seed for random number generation
    for (int i = 0; i < n; i++)
    {
        a[i] = rand() % 100; // Generate random numbers between 0 and 99
    }
}

int main()
{
    int n = 10; // Choose a sufficiently large value for n
    int arr[n];

    // Populate array with random elements
    populateArrayRandomly(arr, n);

    printf("Original array: ");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }

    // Perform quick sort
    quicksort(arr, 0, n - 1);

    printf("\nSorted array using Quick Sort algorithm: ");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }

    printf("\n");

    return 0;
}
```

Name:-Aakash Bhatt
Sap-id:- 500124633

```
PS D:\MCA\MCA-DSA\LAB-10> gcc .\Question3.c
PS D:\MCA\MCA-DSA\LAB-10> .\a.exe
Original array: 4 17 0 17 36 18 94 69 61 48
Sorted array using Quick Sort algorithm: 0 4 17 17 18 36 61 69 94 48
PS D:\MCA\MCA-DSA\LAB-10>
```

Implement Merge Sort using the above dataset.

```c
#include <stdio.h>

void merge(int a[], int lb, int mid, int ub)
{
    int i, j, k;
    i = lb;
    j = mid + 1;
    k = lb;
    int b[ub + 1];

    while (i <= mid && j <= ub)
    {
        if (a[i] < a[j])
        {
            b[k] = a[i];
            i++;
        }
        else
        {
            b[k] = a[j];
            j++;
        }
        k++;
    }

    if (i > mid)
    {
        while (j <= ub)
        {
            b[k] = a[j];
            j++;
            k++;
        }
    }
    else
    {
        while (i <= mid)
        {
            b[k] = a[i];
            i++;
            k++;
```

Name:-Aakash Bhatt
Sap-id:- 500124633

```
        }
    }

    for (k = lb; k <= ub; k++)
    {
        a[k] = b[k];
    }
}
void mergeSort(int a[], int lb, int ub)
{
    if (lb < ub)
    {
        int mid = (lb + ub) / 2;
        mergeSort(a, lb, mid);
        mergeSort(a, mid + 1, ub);
        merge(a, lb, mid, ub);
    }
}

int main()
{
    int arr[] = {15, 5, 24, 8, 1, 3, 16, 10, 20};
    int n = sizeof(arr) / sizeof(arr[0]);

    printf("Original array is: \n");
    for (int i = 0; i < n; i++)
    {
        printf("%d, ", arr[i]);
    }
    mergeSort(arr, 0, n - 1);

    printf("\n\nSorted array using Merge Sort algorithm is: \n");
    for (int i = 0; i < n; i++)
    {
        printf("%d, ", arr[i]);
    }
    return 0;
}
```

```
PS D:\MCA\MCA-DSA\SearchingAndSorting> gcc .\mergeSort.c
PS D:\MCA\MCA-DSA\SearchingAndSorting> .\a.exe
Original array is:
15, 5, 24, 8, 1, 3, 16, 10, 20,

Sorted array using Merge Sort algorithm is:
1, 3, 5, 8, 10, 15, 16, 20, 24,
```

Name:-Aakash Bhatt
Sap-id:- 500124633