**Program Name:** B.Sc.(H) Computer Science

**Semester:** 4th

**Title of the Paper:** Design and Analysis of Algorithms

**Unique Paper Code:** 32341401

**Name:** Shivam Verma

**College Roll No.:** 19HCS4048

**University Roll No.:** 19015570031

**Date of Submission:** 05th May 2021

Q1: Write a code for heap sort with following details.

- Create a random input of different sizes (100, 200, 300, 400, 500, 700, 800, 1000, 2000, 3000, 4000, 5000, 7000, 8000, 10000, 15000, 20000)

- For every input and every situation, count the number of comparisons made in the insertion sort.

- Create an excel with input size and number of comparisons made for each of the situations. (paste the screenshot of excel file in pdf).

CODE

/\* Given a set of positive integers and a sum value S, find out if there exists a subset in array whose sum is equal to given sum S using Dynamic Programming

Name: Shivam Verma

Course: B.Sc.(H) Computer Science

Semester: 4th

Roll No.: 19HCS4048

\*/

#include <bits/stdc++.h>

using namespace std;

int count=0;

void generateRandomArray(int \*array, int size)

{

    srand(time(nullptr));

    for(int index = 0; index < size; ++index)

        array[index] = (rand() % size) + 1;

}

void generateArrayAscendingOrder(int \*array, int size)

{

    for(int index = 0; index < size; ++index)

        array[index] = index;

}

void generateArrayDescendingOrder(int \*array, int size)

{

    int element = size;

    for(int index = 0; index < size; ++index)

        array[index] = element--;

}

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

    for(int index = 0; index < size; ++index)

        cout << array[index] << " ";

    cout << endl;

}

void maxHeapify(int \*array, int n, int index) {

    int largest = index;

    int left = 2 \* index + 1;

    int right= 2 \* index + 2;

    if(left < n && array[left] > array[largest]) {

        ::count++;

        largest=left;

    }

    if(right < n && array[right] > array[largest]) {

        largest = right;

        ::count++;

    }

    if(largest != index) {

        swap(array[largest],array[index]);

        maxHeapify(array, n, largest);

    }

}

void heapsort(int \*array, int n)

{

    for(int i = n / 2 - 1; i >= 0; i--)

        maxHeapify(array,n,i);

    for(int i = n - 1; i >= 0; i--) {

        swap(array[0],array[i]);

        maxHeapify(array,i,0);

    }

}

int main()

{

    char first\_choice, second\_choice;

    int size, \*array;

    cout << "\*\*\*\* Case Scenario \*\*\*\*\n";

    cout << "1. Random Case\n";

    cout << "2. Best Case\n";

    cout << "3. Worst Case\n";

    cout << "Enter your choice: ";

    cin >> first\_choice;

    do{

        cout << "Enter the size of array: ";

        cin >> size;

        array = new int[size];

        switch(first\_choice)

        {

            case '1':

                generateRandomArray(array, size);

                heapsort(array, size);

                break;

            case '2':

                generateArrayAscendingOrder(array, size);

                heapsort(array, size);

                break;

            case '3':

                generateArrayDescendingOrder(array, size);

                heapsort(array, size);

                break;

            default:

                cout << "Wrong choice of option!!!\n";

        }

        cout << "\nNumber of comparisons: " << ::count;

        ::count = 0;

        cout << "\nDo you wish to continue? (Y/N): ";

        cin >> second\_choice;

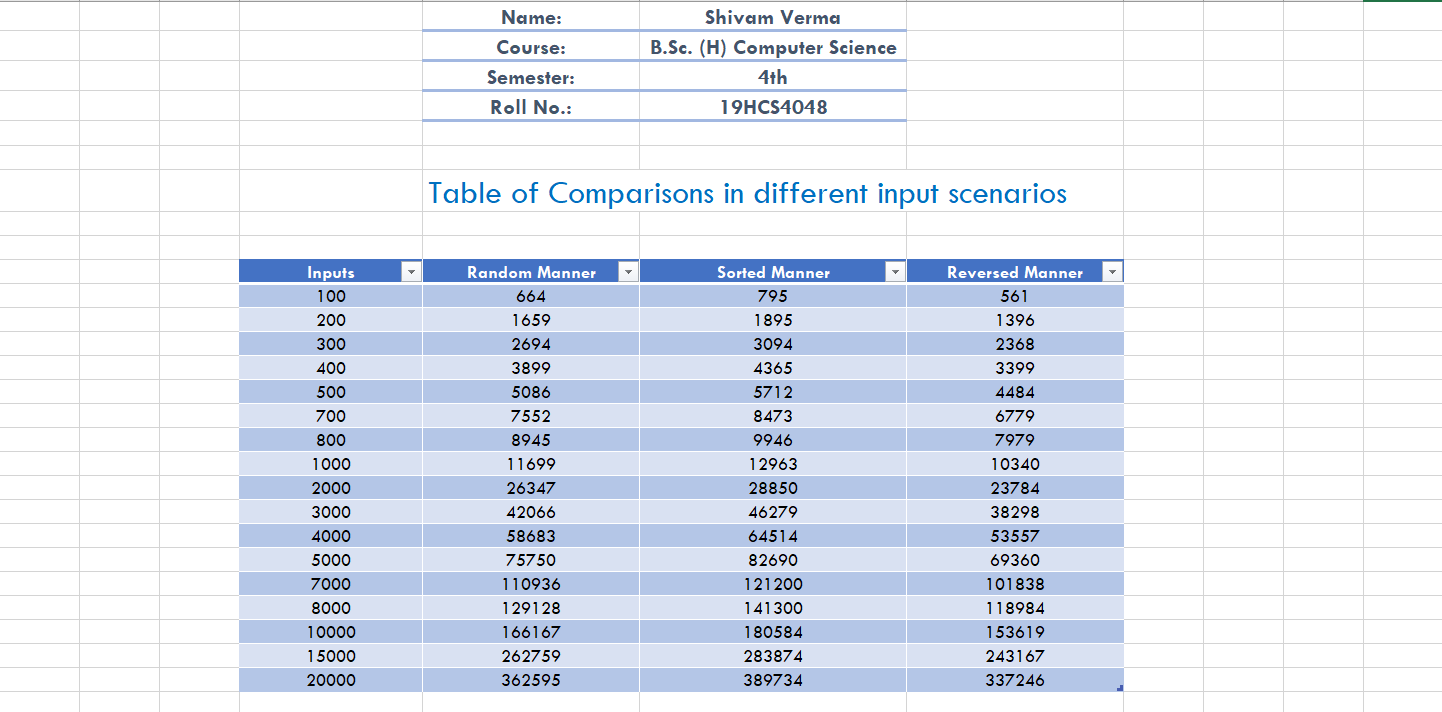
    } while(second\_choice == 'y'|| second\_choice == 'Y');

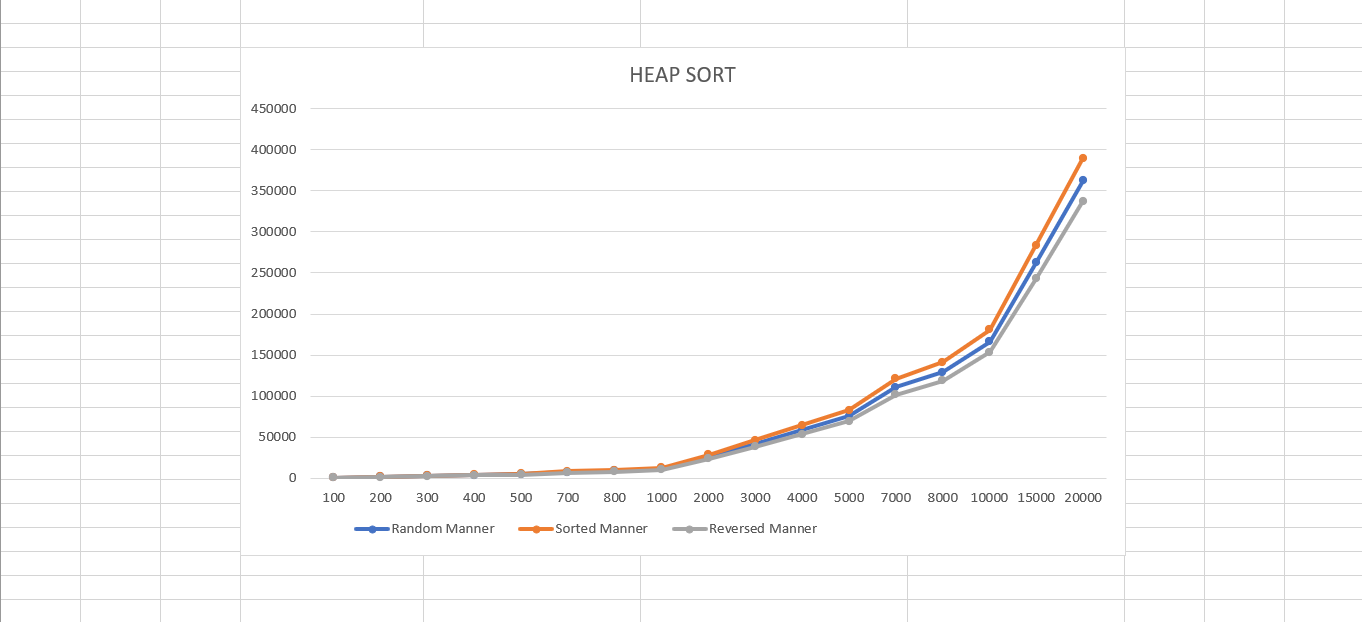
    delete[] array;

    return 0;

}

EXCEL SHEET





Q2. Given a set of positive integers and a sum value S, find out if there exists a subset in array whose sum is equal to given sum S using Dynamic Programming.

CODE

/\* Given a set of positive integers and a sum value S, find out if there exists a subset in array whose sum is equal to given sum S using Dynamic Programming.

Name: Shivam Verma

Course: B.Sc.(H) Computer Science

Semester: 4th

Roll No.: 19HCS4048

\*/

#include <bits/stdc++.h>

using namespace std;

bool\*\* dp;

void display(const vector<int>& v)

{

    for (int index = 0; index < v.size(); ++index)

        cout << v[index] << " ";

    cout << endl;

}

void printSubsets(int array[], int i, int sum, vector<int>& p)

{

    if (i == 0 && sum != 0 && dp[0][sum])

    {

        p.push\_back(array[i]);

          if (array[i] == sum)

              display(p);

        return;

    }

    if (i == 0 && sum == 0)

    {

        display(p);

        return;

    }

    if (dp[i - 1][sum])

    {

        vector<int> b = p;

        printSubsets(array, i - 1, sum, b);

    }

    if (sum >= array[i] && dp[i - 1][sum - array[i]])

    {

        p.push\_back(array[i]);

        printSubsets(array, i - 1, sum - array[i], p);

    }

}

void printAllSubsets(int arr[], int n, int sum)

{

    if (n == 0 || sum < 0)

       return;

    dp = new bool\*[n];

    for (int i = 0; i < n; ++i)

    {

        dp[i] = new bool[sum + 1];

        dp[i][0] = true;

    }

    if (arr[0] <= sum)

       dp[0][arr[0]] = true;

    for (int i = 1; i < n; ++i)

        for (int j = 0; j < sum + 1; ++j)

            dp[i][j] = (arr[i] <= j) ? dp[i - 1][j] || dp[i - 1][j - arr[i]] : dp[i - 1][j];

    if (dp[n - 1][sum] == false)

    {

        cout << "There are no subsets with sum " << sum;

        return;

    }

    vector<int> p;

    cout << "\nThe subsets with sum " << sum << " : \n";

    printSubsets(arr, n - 1, sum, p);

}

int main()

{

    int \*array, size, sum = 0;

    system("cls");

    cout << "Enter the size of Array: ";

    cin >> size;

    array = new int[size];

    cout << "Enter the elements in array...\n";

    for(int index = 0; index < size; ++index)

        cin >> array[index];

    cout << "Enter the target sum: ";

    cin >> sum;

    printAllSubsets(array, size, sum);

    return 0;

}

OUTPUT

