DAA_Ass2

Name: Vishal Sule

Roll No:234

PRN:0120190064

//Code

```
#include <algorithm>
#include <chrono>
#include <iostream>
#include <vector>
using namespace std;
using namespace std ::chrono;
int partition(int a[], int l, int h)
{
    int i = l, j = h;
    int pivot = a[l];
    int temp;
    while (i < j)
    {
        do
        {
            i++;
        } while (a[i] <= pivot);
        do
        {
            j--;
        } while (a[j] > pivot);
        if (i < j)
        {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
        }
    }
    temp = a[l];
    a[l] = a[j];
    a[j] = temp;
    return j;
}
void quicksort(int arr[], int l, int h)
{
    if (l < h)
    {
        int j = partition(arr, l, h);
        quicksort(arr, l, j);
        quicksort(arr, j + 1, h);
    }
}
int findMedian(vector<int> vec)
{
    // Find median of a vector
    int median;
```

```
    size_t size = vec.size();
    median = vec[(size / 2)];
    return median;
}
int findMedianOfMedians(vector<vector<int>> values)
{
    vector<int> medians;
    for (int i = 0; i < values.size(); i++)
    {
        int m = findMedian(values[i]);
        medians.push_back(m);
    }
    return findMedian(medians);
}
void selectionByMedianOfMedians(const vector<int> values, int k)
{
    // Divide the list into n/5 lists of 5 elements each
    vector<vector<int>> vec2D;
    int count = 0;
    while (count != values.size())
    {
        int countRow = 0;
        vector<int> row;
        while ((countRow < 5) && (count < values.size()))
        {
            row.push_back(values[count]);
            count++;
            countRow++;
        }
        vec2D.push_back(row);
    }
    cout << endl
         << endl
         << "Printing 2D vector : " << endl;
    for (int i = 0; i < vec2D.size(); i++)
    {
        for (int j = 0; j < vec2D[i].size(); j++)
        {
            cout << vec2D[i][j] << " ";
        }
        cout << endl;
    }
    cout << endl;
    // Calculating a new pivot for making splits
    int m = findMedianOfMedians(vec2D);
    cout << "Median of medians is : " << m << endl;
    // Partition the list into unique elements larger than 'm' (call this
 // those smaller them 'm' (call this sublist L2)
vector<int> L1, L2;
for (int i = 0; i < vec2D.size(); i++)
{
    for (int j = 0; j < vec2D[i].size(); j++)
    {
        if (vec2D[i][j] > m)
        {
            L1.push_back(vec2D[i][j]);
        }
        else if (vec2D[i][j] < m)
```

```
        {
            L2.push_back(vec2D[i][j]);
        }
    }
}
// Checking the splits as per the new pivot 'm'
cout << endl
     << "Printing L1 : " << endl;
for (int i = 0; i < L1.size(); i++)
{
    cout << L1[i] << " ";
}
cout << endl
     << endl
     << "Printing L2 : " << endl;
for (int i = 0; i < L2.size(); i++)
{
    cout << L2[i] << " ";
}
// Recursive calls
if ((k - 1) == L1.size())
{
    cout << endl
         << endl
         << "Answer :" << m;
}
else if (k <= L1.size())
{
    return selectionByMedianOfMedians(L1, k);
}
else if (k > (L1.size() + 1))
{
    return selectionByMedianOfMedians(L2, k - ((int)L1.size()) - 1);
}
}
void printArray(int *arr, int len)
{
    for (int i = 0; i < len; i++)
        cout << arr[i] << " ";
    cout << endl;
}
void merge(int arr[], int left, int middle, int right)
{
    int n1 = middle - left + 1;
    int n2 = right - middle;
    int L[n1], R[n2];
    for (int i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[middle + 1 + j];
    int i = 0;
    int j = 0;
    int k = left;
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
```

```
            i++;
        }
        else
        {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
    while (i < n1)
    {
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j < n2)
    {
        arr[k] = R[j];
        j++;
        k++;
    }
}
void mergeSort(int arr[], int left, int right)
{
    if (left >= right)
    {
        return;
    }
    int mid = left + (right - left) / 2;
    mergeSort(arr, left, mid);
    mergeSort(arr, mid + 1, right);
    merge(arr, left, mid, right);
}
int main()
{
    int choice;
    do
    {

        cout << "1) Merge Sort \n";
        cout << "2) Quick sort \n";
        cout << "3) Median of median\n";
        cout << "4) Exit\n";
        cout << " Select your choice : ";
        cin >> choice;
        switch (choice)
        {
        case 1:
        {
            int n, ch;
            auto start = high_resolution_clock::now();
            auto stop = high_resolution_clock::now();
            auto duration = duration_cast<microseconds>(stop - start);
            cout << "\nHow many elements you want to sort?";
            cin >> n;
            int arr[n];
            for (int i = 0; i < n; i++)
            {
```

```
            arr[i] = (rand() % n) + 1;
        }
        int arr_size = sizeof(arr) / sizeof(arr[0]);
        cout << "\nOriginal array: ";
        printArray(arr, arr_size);
        start = high_resolution_clock::now();
        mergeSort(arr, 0, arr_size - 1);
        stop = high_resolution_clock::now();
        duration = duration_cast<microseconds>(stop - start);
        cout << "\nArray after sorting: ";
        printArray(arr, arr_size);
        cout << "\nTime taken by Merge Sort:" << duration.count() << "microseconds\n ";
            break;
    }
    case 2:
    {
        int n;
        cout << "\nHow many elements you want to sort?";
        cin >> n;
        int arr[n];
        cout << "\nOriginal array: ";
        for (int i = 0; i < n; i++)
        {
            arr[i] = (rand() % n) + 1;
        }
        for (int i = 0; i < n; i++)
        {
            cout << arr[i] << " ";
        }
        cout << endl;
        auto start = high_resolution_clock::now();
        cout << "\nArray after sorting: ";
        quicksort(arr, 0, n);
        for (int i = 0; i < n; i++)
        {
            cout << arr[i] << " ";
        }
        cout << endl;
        auto stop = high_resolution_clock::now();
        auto duration = duration_cast<microseconds>(stop - start);
        cout << "\nTime taken by Quick Sort: " << duration.count() << "microseconds " << endl;
            break;
    }
    case 3:
    {
        int size;
        cout << "enter the size off arr:";
        cin >> size;
        int values[size];
        for (int i = 0; i < size; i++)
            values[i] = rand() % 10000;
        vector<int> vec(values, values + 25);
        cout << "The given array is : " << endl;
        for (int i = 0; i < vec.size(); i++)
        {
            cout << vec[i] << " ";
        }
        int k = (size - 1) / 2;
```

```
            selectionByMedianOfMedians(vec, k);
            break;
        }
        case 4:
        {
            exit(0);
        }
        default:
        {
            cout << "\nWrong Choice";
            break;
        }
        }
    } while (choice != 7);
    return 0;
}
```

//Output:

```
input
1) Merge Sort
2) Quick sort
3) Median of median
4) Exit
Select your choice : 1

How many elements you want to sort?10

Original array: 4 7 8 6 4 6 7 3 10 2

Array after sorting: 2 3 4 4 6 6 7 7 8 10

Time taken by Merge Sort:1microseconds
1) Merge Sort
2) Quick sort
3) Median of median
4) Exit
Select your choice : 2

How many elements you want to sort?10

Original array: 3 8 1 10 4 7 1 7 3 7

Array after sorting: 1 1 3 3 4 7 7 7 8 10

Time taken by Quick Sort: 9microseconds
1) Merge Sort
2) Quick sort
3) Median of median
4) Exit
Select your choice : 3
Enter the size off arr:10
The given array is :
5211 5368 2567 6429 5782 1530 2862 5123 4067 3135 406510049 22081 1
0 0 0 0 10 0 0 0 10 0 0 0 10
```

```
Printing 2D vector :
5211 5368 2567 6429 5782
1530 2862 5123 4067 3135
406510049 22081 10 0 0
0 10 0 0 0
10 0 0 0 10

Median of medians is : 10

Printing L1 :
5211 5368 2567 6429 5782 1530 2862 5123 4067 3135 406510049 22081

Printing L2 :
0 0 0 0 0 0 0 0 0

Printing 2D vector :
5211 5368 2567 6429 5782
1530 2862 5123 4067 3135
406510049 22081

Median of medians is : 5123

Printing L1 :
5211 5368 6429 5782 406510049 22081

Printing L2 :
2567 1530 2862 4067 3135

Printing 2D vector :
5211 5368 6429 5782 406510049
22081

Median of medians is : 22081

Printing L1 :
```

```
Printing L2 :
5211 5368 6429 5782

Printing 2D vector :
5211 5368 6429 5782

Median of medians is : 6429

Printing L1 :


Printing L2 :
5211 5368 5782

Printing 2D vector :
5211 5368 5782

Median of medians is : 5368

Printing L1 :
5782

Printing L2 :
5211

Printing 2D vector :
5782

Median of medians is : 5782

Printing L1 :


Printing L2 :
```

//Graph:

ize	Quick So	Merge Sc
10	15	1
11	5	1
12	9	1
13	6	1
14	5	2