**Saad iqbal**

**Bee-6B**

**Reg# 32903**

**CODE**

#include<iostream>//libraries

#include<vector>

#include<time.h>

#include<fstream>

#include<algorithm>

using namespace std;//defining std

int g = 0;

//finding median using first,middle and last value

int median(vector<int>&X, int first\_index, int last\_index){//calculate median by first,middle,last value in data

int A[] = { first\_index, last\_index, (first\_index + last\_index) / 2 };

for (int i = 2; i < 0; i--){

int max = A[i];

for (int j = 0; j < i; j++){

if (X[A[j]]>X[max])

max = A[j];

}

swap(A[i], A[max]);

}

return A[1];

}

int parting(vector<int>&A, int first, int last){//place pivot at ite location+on left side are smaller and on right side are greater

g++;

int x, y, z;

/\*x = median(A, first, last);

swap(A[x], A[first]);\*/

x = first;

y = first + 1;

z = last;

while (1){

int d = 1;

for (; A[z] >= A[x] && z > first; z--);

for (; A[y] <= A[x] && y <= last && d; y++){

if (y == last){

d--;

y--;

}

}

if (y < z){

swap(A[y], A[z]);

}

else

{

swap(A[x], A[z]);

return y;

}

}

}

void Quick\_sort(vector<int> &A,int first\_index,int last\_index){//Quick sort(part handling recursive call in quick sort)

if (first\_index < last\_index){

int q = parting(A, first\_index, last\_index);

Quick\_sort(A,first\_index,q-1);

Quick\_sort(A, q, last\_index);

}

}

vector<int> random(int y){//random number generation

vector<int>a;

for (int i = 0; i <y; i++)

a.push\_back(rand() % 100 + 1);

return a;

}

int main(){//main

int number = 10;

ofstream filed;//output file stream

filed.open("file1.csv");

while (number <= 100000){//loop for analysing algo for more inputs

filed << number << "inputs" << endl;

int a, b;

srand(time(NULL));

vector<int> x = random(number);

cout << "using Quick Sort" << endl;

a = clock();//clock function

Quick\_sort(x,0,x.size()-1);

b = clock();

filed << "Quick Sort" << "," << b - a << "msec" <<","<<g<<"times"<< endl;

g = 0;

number \*= 10;//increment

}

filed.close();

system("pause");

return 0;

}

**Output**

|  |  |  |
| --- | --- | --- |
| Quick Sort | Time | Call for parting in code |
| 10inputs | 0msec | 9times |
|  |  |  |
| 100inputs | 0msec | 99times |
|  |  |  |
| 1000inputs | 0msec | 999times |
|  |  |  |
| 10000inputs | 78msec | 9999times |
|  |  |  |
| 100000inputs | 5674msec | 99999times |

**Task1 running time analysis for random pivot**

|  |  |  |
| --- | --- | --- |
| Quick Sort | Time | Call for parting in code |
| 10inputs | 0msec | 9times |
|  |  |  |
| 100inputs | 0msec | 99times |
|  |  |  |
| 1000inputs | 0msec | 999times |
|  |  |  |
| 10000inputs | 94msec | 9999times |
|  |  |  |
| 100000inputs | 6828msec | 99999times |

**Task2 running time analysis for median as pivot**

**In best case:**

Time complexity is **nlogn.**

**In worse case:**

Time complexity is **n2**.

And In Quick sort no additional memory is required by machine.

Quick sort called parting function n-1 times in both cases and reason for greater time in task#2 is that function of median also called n-1 times.