**Saad iqbal**

**Bee-6B**

**Reg# 32903**

**Task no#01**

#include<iostream>//libraries

#include<vector>

#include<cmath>

#include<time.h>

#include<queue>

#include<fstream>

using namespace std;//defining std

int K\_value(vector<int>x){//counting digits of greatest number-1

int temp = -1;

for (int i = 0; i < x.size();i++)

if (temp < x[i])

temp = x[i];

if (temp>0)

return log10(temp);

return -1;

}

vector<int> Radix(vector<int> &data){//nlogn sorting algo "Radix" implementation......non negative integers data + zero is not greatest number in data

int x, y, z, divider, k;

const int radix = 10;

k = K\_value(data);

if (k == -1)

return data;

queue<int>temp[radix];

divider = 1;

for (x = 0; x <= k; x++, divider \*= radix){

for (y = 0; y < data.size(); y++)

temp[(data[y] / divider) % divider].push(data[y]);

for (y = z = 0; y < radix; y++)

while (!temp[y].empty()){

data[z++] = temp[y].front();

temp[y].pop();

}

divider = 1;

}

return data;

}

void Merge(vector<int> &x, int n1, int mid, int n2){//Merging two sorted vectors

int a = n1, b = mid, c = n1;

vector<int> B = x;

while (a < mid && b <= n2){

if (x[a] < x[b])

B[c++] = x[a++];

else

B[c++] = x[b++];

}

while (a < mid)

B[c++] = x[a++];

while (b <= n2)

B[c++] = x[b++];

for (a = n1; a <= n2; a++)

x[a] = B[a];

}

void Merge\_sort(vector<int> &x,int first,int last){//nlogn sorting algo "Merge" implimentation

if (first < last){

int mid = (first + last) / 2;

Merge\_sort(x, first, mid);

Merge\_sort(x, mid + 1, last);

Merge(x, first, mid + 1, last);

}

}

int main(){//main

vector<int> y, x = y = { 13, 62, 73, 04, 33, 63, 39, 42, 41, 24, 56, 46, 40, 52, 45 ,45, 22, 35, 41, 81 };//random data

cout << "Initially random data" << endl;//printing random data on screen

for (int i = 0; i < x.size(); i++)

cout << x[i] << " ";

cout << endl;

cout << "Using Merge Sort" << endl;

Merge\_sort(x, 0, x.size() - 1);//Merge sort

for (int i = 0; i < x.size(); i++)

cout << x[i] << " ";

cout << endl;

Radix(y);

cout << "Using Radix Sort" << endl;

for (int i = 0; i < y.size(); i++)//radix sort

cout << x[i] << " ";

cout << endl;

system("pause");

return 0;

}

**Task no#02**

#include<iostream>//libraries

#include<vector>

#include<cmath>

#include<time.h>

#include<queue>

#include<fstream>

using namespace std;//defining std

int K\_value(vector<int>x){//counting digits of greatest number-1

int temp = -1;

for (int i = 0; i < x.size();i++)

if (temp < x[i])

temp = x[i];

if (temp>0)

return log10(temp);

return -1;

}

vector<int> Radix(vector<int> &data){//nlogn sorting algo "Radix" implementation......non negative integers data + zero is not greatest number in data

int x, y, z, divider, k;

const int radix = 10;

k = K\_value(data);

if (k == -1)

return data;

queue<int>temp[radix];

divider = 1;

for (x = 0; x <= k; x++, divider \*= radix){

for (y = 0; y < data.size(); y++)

temp[(data[y] / divider) % divider].push(data[y]);

for (y = z = 0; y < radix; y++)

while (!temp[y].empty()){

data[z++] = temp[y].front();

temp[y].pop();

}

divider = 1;

}

return data;

}

void Merge(vector<int> &x, int n1, int mid, int n2){//Merging two sorted vectors

int a = n1, b = mid, c = n1;

vector<int> B = x;

while (a < mid && b <= n2){

if (x[a] < x[b])

B[c++] = x[a++];

else

B[c++] = x[b++];

}

while (a < mid)

B[c++] = x[a++];

while (b <= n2)

B[c++] = x[b++];

for (a = n1; a <= n2; a++)

x[a] = B[a];

}

void Merge\_sort(vector<int> &x,int first,int last){//nlogn sorting algo "Merge" implimentation

if (first < last){

int mid = (first + last) / 2;

Merge\_sort(x, first, mid);

Merge\_sort(x, mid + 1, last);

Merge(x, first, mid + 1, last);

}

}

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 = 100;

ofstream filed;//output file stream

filed.open("file.csv");

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

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

int a, b;

srand(time(NULL));

vector<int> y,x=y = random(number);

cout << "using Radix Sort" << endl;

a = clock();//clock function

Radix(x);

b = clock();

filed << "Radix\_Sort" << "," << b - a << "msec" << endl;

cout << "using Merge sort" << endl;

a = clock();

Merge\_sort(y,0,y.size()-1);

b = clock();

filed << "Merge sort" << "," << b - a << "msec" << endl;

number \*= 10;//increment

}

filed.close();

system("pause");

return 0;

}

**Task no#03**

#include<iostream>//libraries

#include<vector>

#include<cmath>

#include<time.h>

#include<queue>

#include<fstream>

using namespace std;//defining std

int K\_value(vector<int>x){//counting digits of greatest number-1

int temp = -1;

for (int i = 0; i < x.size();i++)

if (temp < x[i])

temp = x[i];

if (temp>0)

return log10(temp);

return -1;

}

vector<int> Radix(vector<int> &data){//nlogn sorting algo "Radix" implementation......non negative integers data + zero is not greatest number in data

int x, y, z, divider, k;

const int radix = 10;

k = K\_value(data);

if (k == -1)

return data;

queue<int>temp[radix];

divider = 1;

for (x = 0; x <= k; x++, divider \*= radix){

for (y = 0; y < data.size(); y++)

temp[(data[y] / divider) % divider].push(data[y]);

for (y = z = 0; y < radix; y++)

while (!temp[y].empty()){

data[z++] = temp[y].front();

temp[y].pop();

}

divider = 1;

}

return data;

}

void Merge(vector<int> &x, int n1, int mid, int n2){//Merging two sorted vectors

int a = n1, b = mid, c = n1;

vector<int> B = x;

while (a < mid && b <= n2){

if (x[a] < x[b])

B[c++] = x[a++];

else

B[c++] = x[b++];

}

while (a < mid)

B[c++] = x[a++];

while (b <= n2)

B[c++] = x[b++];

for (a = n1; a <= n2; a++)

x[a] = B[a];

}

void Merge\_sort(vector<int> &x,int first,int last){//nlogn sorting algo "Merge" implimentation

if (first < last){

int mid = (first + last) / 2;

Merge\_sort(x, first, mid);

Merge\_sort(x, mid + 1, last);

Merge(x, first, mid + 1, last);

}

}

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 = 100;

ofstream filed;//output file stream

filed.open("file.csv");

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

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

int a, b;

srand(time(NULL));

vector<int> y,x=y = random(number);

sort(x.begin(), x.end());//sort AS

sort(y.begin(), y.end());//sort AS

reverse(x.begin(), x.end());//reverse

reverse(y.begin(), y.end());//reverse

cout << "using Radix Sort" << endl;

a = clock();//clock function

Radix(x);

b = clock();

filed << "Radix\_Sort" << "," << b - a << "msec" << endl;

cout << "using Merge sort" << endl;

a = clock();

Merge\_sort(y,0,y.size()-1);

b = clock();

filed << "Merge sort" << "," << b - a << "msec" << endl;

number \*= 10;//increment

}

filed.close();

system("pause");

return 0;

}

**Task#2 reasoning**

|  |  |
| --- | --- |
| 100inputs | Time |
| Radix\_Sort | 0msec |
| Merge sort | 0msec |
| 1000inputs |  |
| Radix\_Sort | 31msec |
| Merge sort | 47msec |
| 10000inputs |  |
| Radix\_Sort | 110msec |
| Merge sort | 2031msec |
| 100000inputs |  |
| Radix\_Sort | 2219msec |
| Merge sort | 21002msec |

**Task2 running time analysis**

Result is nearly as same we imagining, for first 100 inputs, time is less than 1msec.

Radix sort, check n x log10(n) times and there is no comparison.

Merge sort, check n x log10(n) times and there is insertion for ~2n-1.

**Task#3 reasoning**

|  |  |
| --- | --- |
| 100acscending inputs | Time |
| Radix\_Sort | 0msec |
| Merge sort | 0msec |
| 1000acscending inputs |  |
| Radix\_Sort | 15msec |
| Merge sort | 31msec |
| 10000acscending inputs |  |
| Radix\_Sort | 125msec |
| Merge sort | 2000msec |
| 100000acscending inputs |  |
| Radix\_Sort | 2562msec |
| Merge sort | 19049msec |

**Task3 running time analysis for ascending data**

|  |  |
| --- | --- |
| 100descending inputs | Time |
| Radix\_Sort | 0msec |
| Merge sort | 15msec |
| 1000descending inputs |  |
| Radix\_Sort | 31msec |
| Merge sort | 47msec |
| 10000descending inputs |  |
| Radix\_Sort | 109msec |
| Merge sort | 2016msec |
| 100000descending inputs |  |
| Radix\_Sort | 2406msec |
| Merge sort | 22785msec |

**Task3 running time analysis for descending data**

Radix->same for random, best and worst because there is no comparison. Its complexity depends on the digits of large number.

Merge->In best case, there is no insertion but in worst case there is 2n-1 time insertions.