# *CSE331: Data Structures and Algorithms*

***Quick, Counting and Radix Sort Lab Report***

Name: Anthony Amgad Fayek

Program: CESS

ID: 19P9880

**The Full Project is in a GitHub Repository Below**

 Here are the used libraries and definitions:

#include <iostream>

#include <fstream>

#include <ctime>

#include <cstdlib>

#define LENGTH 10000

using namespace std;

int step = 0;

Part 1:

Writing a C++ function to generate 10,000 random numbers between 1 and 10,000 and save them in a file (the full generated txt is in the GitHub repository linked below):

void createRandFile() {

ofstream mfile("unsortedFile.txt");

srand(time(0));

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

mfile << ((rand() % LENGTH) + 1) << endl;

}

}

Part 2:

Writing the Heap sort functions (this includes the counter (the variable “step”) that is required in Part 3:

int partition(int arr[], int s, int l) {

int temp;

int x = arr[l];

int i = s - 1;

for (int j = s; j < l; j++) {

if (arr[j] <= x) {

i++;

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

step += 4;

}

}

temp = arr[l];

arr[l] = arr[i + 1];

arr[i + 1] = temp;

step += 5;

return i + 1;

}

void quickSort(int arr[], int s, int l) {

if (s < l) { int m = partition(arr, s, l);

quickSort(arr, s, m - 1);

quickSort(arr, m + 1, l);

step += 2;}

}

Writing a function to write the resultant array into a file (the full generated txt is in the GitHub repository linked below):

void createSortedFile(int arr[]) {

ofstream mfile("sortedFile.txt");

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

mfile << arr[i] << endl;

}

}

Part 3:

Creating the main function which reads n items using another function from the file generated and executes the heap algorithm with step 50 and writes a file that includes pairs of n and f(n) (“step”) (the full generated txt is in the GitHub repository linked below):

void readFile(int arr[], int l) {

ifstream mfile("unsortedFile.txt");

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

mfile >> arr[i];

}

}

int main() {

int arr[LENGTH];

createRandFile();

readFile(arr, LENGTH);

ofstream aFile("quickStepFile.txt");

ofstream bFile("radixStepFile.txt");

ofstream cFile("countingStepFile.txt");

int x[LENGTH];

int y[LENGTH];

int z[LENGTH];

for (int i = 10; i < 10000; i += 50) {

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

x[j] = arr[j];

y[j] = arr[j];

z[j] = arr[j];

}

step = 0;

quickSort(x, 0, i - 1);

aFile << i << ',' << step << endl;

step = 0;

radixSort(y,i);

bFile << i << ',' << step << endl;

step = 0;

countingSort(z, i);

cFile << i << ',' << step << endl;

}

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

x[j] = arr[j];

y[j] = arr[j];

}

step = 0;

quickSort(x, 0, LENGTH - 1);

aFile << LENGTH << ',' << step << endl;

step = 0;

radixSort(y, LENGTH);

bFile << LENGTH << ',' << step << endl;

step = 0;

countingSort(arr, LENGTH);

cFile << LENGTH << ',' << step << endl;

createSortedFile(arr);

system("pause");

return 0;

}

Part 4:

The “quickStepFile.txt” created in the main function is then imported into excel with an added column of (nlg(n)). Then a generated Graph from the excel is created:

The same steps are then repeated but with Counting Sort. It requires the following function:

void countingSort(int arr[], int len) {

step += 3;

int outarr[LENGTH];

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

outarr[j] = 0;

step++;

}

int count[LENGTH+1];

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

count[j] = 0;

step++;

}

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

count[arr[i]] = count[arr[i]] + 1;

step++;

}

for (int i = 1; i <= LENGTH; i++) {

count[i] = count[i] + count[i - 1];

step++;

}

for (int i = len - 1; i >= 0; i--) {

outarr[count[arr[i]]-1] = arr[i];

count[arr[i]] = count[arr[i]] - 1;

step += 2;

}

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

arr[i] = outarr[i];

step++;

}

}

The “countingStepFile.txt” created in the main function is then imported into excel with an added column of (n + k). Then a generated Graph from the excel is created:

The same steps are then repeated but with Radix Sort. It requires the following functions:

void countingRadSort(int arr[], int d, int len) {

int count[10] = { 0,0,0,0,0,0,0,0,0,0 };

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

count[(arr[i] / d) % 10]++;

step++;

}

for (int i = 1; i < 10; i++) {

count[i] += count[i - 1];

step++;

}

int outarr[LENGTH];

step += 2;

for (int i = len - 1; i >= 0; i--) {

outarr[--count[(arr[i] / d) % 10]] = arr[i];

step++;

}

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

arr[i] = outarr[i];

step++;

}

}

void radixSort(int arr[], int len) {

int max = arr[0];

step++;

for (int i = 1; i < len; i++) {

if (max < arr[i]) {

max = arr[i];

step++;

}

}

for (int d = 1; max / d > 0; d \*= 10) {

countingRadSort(arr, d, len);

}

}

The “radixStepFile.txt” created in the main function is then imported into excel with an added column of (d(n + k)). Then a generated Graph from the excel is created:

GitHub Repository:

<https://github.com/Anthony-Amgad/CSE331QuickCountRadixSort19P9880>