EECE 7205: Introduction of Computer Engineering

Assignment 2

Jiayun Xin

NUID: 001563582

College of Engineering

Northeastern University Boston, Massachusetts

Fall, 2021

**Q1**

**Codes:**

#include <cstdlib>

#include <iostream>

#include <time.h>

#include <stdlib.h>

using namespace std;

int partition(int arr[], int low, int high) {

int pivot = arr[high];

int i = (low - 1);

for (int j = low; j <= high - 1; j++) {

if (arr[j] <= pivot) {

i++;

swap(arr[i], arr[j]);

}

}

swap(arr[i + 1], arr[high]);

return (i + 1);

}

int partition\_r(int arr[], int low, int high) {

srand(time(NULL));

int random = low + rand() % (high - low);

swap(arr[random], arr[high]);

return partition(arr, low, high);

}

void quickSort(int arr[], int low, int high) {

if (low < high) {

int pi = partition\_r(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

void printArray(int arr[], int size) {

int i;

for (i = 0; i < size; i++) printf("%d ", arr[i]);

printf("\n");

}

int main() {

int arr[100];

clock\_t start, finish;

double duration;

for (int i = 0; i < 100; ++i) {

arr[i] = i + 1;

}

int n = sizeof(arr) / sizeof(arr[0]);

start = clock();

quickSort(arr, 0, n - 1);

finish = clock();

duration = (double)(finish - start) / CLOCKS\_PER\_SEC;

printf("Sorted array: \n");

printArray(arr, n);

printf("\n");

printf("Running time: %f seconds\n", duration);

return 0;

}

Results:

**Table

Description automatically generated**

The screenshot shows the sorted array results and the running time of randomized quicksort for 5 times. The average running time is 0.0000508 seconds.

**Q2**

**Codes:**

#include <iostream>

#include <time.h>

using namespace std;

void heapify (int arr[], int n, int i) {

int largest = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

if (l < n && arr[l] > arr[largest])

largest = l;

if (r < n && arr[r] > arr[largest])

largest = r;

if (largest != i) {

swap(arr[i], arr[largest]);

heapify(arr, n, largest);

}

}

void heapSort (int arr[], int n) {

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

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

swap(arr[0], arr[i]);

heapify(arr, i, 0);

}

}

void printArray(int arr[], int n) {

for (int i = 0; i < n; ++i) cout << arr[i] << " ";

cout << "\n";

}

void changeValues (int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void shuffleRandom ( int arr[], int n ) {

srand ( time(NULL) );

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

int j = rand() % (i+1);

changeValues(&arr[i], &arr[j]);

}

}

int main() {

int arr\_size = 100;

int arr[arr\_size];

clock\_t start, finish;

double duration;

for (int i = 0; i < arr\_size; i++) {

arr[i] = i + 1;

}

shuffleRandom (arr, arr\_size);

cout << "Random permutation array is: \n";

printArray(arr, arr\_size);

cout << "\n";

start = clock();

heapSort(arr, arr\_size);

finish = clock();

duration = (double)(finish - start) / CLOCKS\_PER\_SEC;

cout << "Sorted array is: \n";

printArray(arr, arr\_size);

cout << "\n";

printf("Running time: %f seconds\n", duration);

}

**Reports:**

**A screenshot of a computer

Description automatically generated with low confidence**

**Q3**

**Codes:**

#include <iostream>

using namespace std;

void countSort(int array[], int size) {

int output[21];

int count[21];

int max = array[0];

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

if (array[i] > max)

max = array[i];

}

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

count[i] = 0;

}

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

count[array[i]]++;

}

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

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

}

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

output[count[array[i]] - 1] = array[i];

count[array[i]]--;

}

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

array[i] = output[i];

}

}

void printArray(int array[], int size) {

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

cout << array[i] << " ";

cout << endl;

}

int main() {

int array[] = {20, 18, 5, 7, 16, 10, 9, 3, 12, 14, 0};

int n = sizeof(array) / sizeof(array[0]);

clock\_t start, finish;

double duration;

start = clock();

countSort(array, n);

finish = clock();

cout << "Sorted array is: ";

printArray(array, n);

duration = (double)(finish - start) / CLOCKS\_PER\_SEC;

cout << "Running time: " << duration << "seconds" "\n";

}

**Results:**

**Text, letter

Description automatically generated**

**Q4**

**Codes:**

#include <iostream>

using namespace std;

int getMax(int arr[], int n) {

int mx = arr[0];

for (int i = 1; i < n; i++)

if (arr[i] > mx)

mx = arr[i];

return mx;

}

void countSort(int arr[], int n, int exp) {

int output[n];

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

for (i = 0; i < n; i++)

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

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

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

for (i = n - 1; i >= 0; i--) {

output[count[(arr[i] / exp) % 10] - 1] = arr[i];

count[(arr[i] / exp) % 10]--;

}

for (i = 0; i < n; i++)

arr[i] = output[i];

}

void radixsort(int arr[], int n) {

int m = getMax(arr, n);

for (int exp = 1; m / exp > 0; exp \*= 10)

countSort(arr, n, exp);

}

void print(int arr[], int n) {

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

cout << arr[i] << " ";

}

int main() {

int arr[] = {329, 457, 657, 839, 436, 720, 353};

int n = sizeof(arr) / sizeof(arr[0]);

clock\_t start, finish;

double duration;

start = clock();

radixsort(arr, n);

finish = clock();

printf("Sorted array is: ");

print(arr, n);

printf("\n");

duration = (double)(finish - start) / CLOCKS\_PER\_SEC;

cout << "Running time: " << duration << "seconds" "\n";

return 0;

}

**Results:**

**Text

Description automatically generated with medium confidence**