**Lab Description:**

\*\*In README File

\*\*Not on lab write up

**Graph:**

**Main.cpp:**

#include <iostream> // cout, endl

#include <stdio.h> // printf

#include <time.h> // time functions

#include <chrono> // chrono

void quickSort(int\*, int, int); // sorts array using Quick Sort algorithm

int partition(int\*, int, int); // partitions array for quickSort function

void mergeSort(int\*, int, int); // sorts array using Merge Sort algorithm

void merge(int\*, int, int, int); // returns a sorted array to the mergeSort function

int\* makeArray(int); // creates a dynamically allocated array with passed limit

void swap(int&, int&); // swaps two integers

int\* makeArray2(int); // Creates an array made for worst case

bool isSorted(int\*, int); // checks to see if array is sorted

void timeSort(void (\*sort)(int\*, int, int), int\*, int, int); // calculates the time it takes to sort an array

int count = 0;

int main() {

srand(time(NULL)); // Seed the random number generator

int sizesLen = 15; // Length of test case array

int sizes[] = {10, 20, 50, 500, 700, 1000, 5000, 7500, 12000, 25000, 50000, 60000, 100000, 500000}; // Amount of integers test cases

std::cout << "===================================\n" << std::endl;

std::cout << "Sorting in Ascending order!" << std::endl;

std::cout << "Merge Sort: " << std::endl;

for (int i = 0; i < 14; i++) {

int \*arr = makeArray(sizes[i]);

int len = sizes[i];

timeSort(mergeSort, arr, 0, len - 1);

}

std::cout << "===================================\n" << std::endl;

std::cout << "Quick Sort: " << std::endl;

for (int i = 0; i < 14; i++) {

int \*arr = makeArray(sizes[i]);

int len = sizes[i];

timeSort(quickSort, arr, 0, len - 1);

}

std::cout << "===================================\n" << std::endl;

std::cout << "Sorting in Worst case!" << std::endl;

std::cout << "Quick Sort: " << std::endl;

for (int i = 0; i < 10; i++) {

int \*arr = makeArray2(sizes[i]);

int len = sizes[i];

timeSort(quickSort, arr, 0, len - 1);

}

std::cout << "===================================\n" << std::endl;

return 0;

}

int\* makeArray2(int len) {

int \*rtn = new int[len];

for (int i = len - 1, j = 0; j < len; j++, i--) {

rtn[j] = i;

}

return rtn;

}

/\*

\* timeSort Function:

\* Calculate the time it takes for specific sorting algorithms to run on and

\* how long they take to sort

\*/

void timeSort(void (\*sort)(int\*, int, int), int\* a, int l, int r) {

auto start = std::chrono::system\_clock::now();

sort(a, l, r);

auto end = std::chrono::system\_clock::now();

std::chrono::duration<double> elapsed\_seconds = end - start;

std::time\_t end\_time = std::chrono::system\_clock::to\_time\_t(end);

std::cout << "Finished at " << std::ctime(&end\_time) << "Elapsed time: " << elapsed\_seconds.count() << "s\n";

std::cout << "Amount of swaps " << count << std::endl;

if (isSorted(a, r)) {

std::cout << "Sorted in correct order!" << std::endl;

} else {

std::cout << "Not sorted correctly!" << std::endl;

}

count = 0;

delete [] a;

}

/\*

\* makeArray Function:

\* Creates and fills an array based off of a specified limit

\*/

int\* makeArray(int len) {

int\* rtn = new int[len];

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

rtn[i] = 1 + rand() % 100;

}

return rtn;

}

/\*

\* isSorted Function:

\* Takes a dynamic array and a length of that array and scans the array to test

\* if it is sorted in the correct ascending order and then determines if the array

\* is sorted correctly

\*/

bool isSorted(int\* arr, int length) {

for (int i = 0; i < (length - 1); i++) {

if (arr[i] > arr[i + 1]) {

return false;

}

}

return true;

}

/\*

\* mergeSort Function:

\* Sorts an array of integers using the Merge Sort algorithm

\*/

void mergeSort(int\* arr, int l, int r) {

if (l < r) {

int p = (l + r) / 2;

mergeSort(arr, l, p);

mergeSort(arr, p + 1, r);

merge(arr, l, p, r);

}

}

/\*

\* merge Function:

\* Merges two sorted arrays into one array and then returns it

\*/

void merge(int\* arr, int l, int m, int r) {

int i, j, k;

int n1 = m - l + 1;

int n2 = r - m;

int L[n1], R[n2];

for (int i = 0; i < n1; i++) {

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

}

for (int j = 0; j < n2; j++) {

R[j] = arr[m + j + 1];

}

i = j = 0;

k = l;

while (i < n1 && j < n2) {

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

count++;

} else {

arr[k] = R[j];

j++;

count++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

count++;

}

while (j < n2) {

arr[k] = R[j];

j++;

k++;

count++;

}

}

/\*

\* swap Function:

\* Takes two array elements by reference and swaps them

\*/

void swap(int& prev, int& next) {

int temp = prev;

prev = next;

next = temp;

count++;

}

/\*

\* quickSort Function:

\* Sorts and array of integers using the Quick Sort algorithm

\*/

void quickSort(int\* arr, int l, int r) {

if (l < r) {

int p = partition(arr, l, r);

quickSort(arr, l, p);

quickSort(arr, p + 1, r);

}

}

/\*

\* partition Function:

\* Returns a partition that will be used for the quickSort function after sorting

\*/

int partition(int\* arr, int l, int r) {

int p = arr[r - 1];

int i = l - 1;

for (int j = l; j <= r - 2; j++) {

if (arr[j] < p) {

i++;

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

}

}

swap(arr[r - 1], arr[i + 1]);

return i + 1;

}

**Sample Output:**

===================================

Sorting in Ascending order!

Merge Sort:

Finished at Fri Feb 7 19:58:05 2020

Elapsed time: 0.0113984s

Amount of swaps 34

Sorted in correct order!

Finished at Fri Feb 7 19:58:05 2020

Elapsed time: 8.253e-05s

Amount of swaps 88

Sorted in correct order!

Finished at Fri Feb 7 19:58:05 2020

Elapsed time: 0.000234249s

Amount of swaps 286

Sorted in correct order!

Finished at Fri Feb 7 19:58:05 2020

Elapsed time: 0.00270347s

Amount of swaps 4488

Sorted in correct order!

Finished at Fri Feb 7 19:58:05 2020

Elapsed time: 0.00429266s

Amount of swaps 6676

Sorted in correct order!

Finished at Fri Feb 7 19:58:05 2020

Elapsed time: 0.0134942s

Amount of swaps 9976

Sorted in correct order!

Finished at Fri Feb 7 19:58:05 2020

Elapsed time: 0.0762269s

Amount of swaps 61808

Sorted in correct order!

Finished at Fri Feb 7 19:58:05 2020

Elapsed time: 0.110439s

Amount of swaps 96808

Sorted in correct order!

Finished at Fri Feb 7 19:58:05 2020

Elapsed time: 0.154142s

Amount of swaps 163616

Sorted in correct order!

Finished at Fri Feb 7 19:58:06 2020

Elapsed time: 0.225519s

Amount of swaps 367232

Sorted in correct order!

Finished at Fri Feb 7 19:58:06 2020

Elapsed time: 0.450202s

Amount of swaps 784464

Sorted in correct order!

Finished at Fri Feb 7 19:58:07 2020

Elapsed time: 0.526407s

Amount of swaps 954464

Sorted in correct order!

Finished at Fri Feb 7 19:58:08 2020

Elapsed time: 0.917984s

Amount of swaps 1668928

Sorted in correct order!

Finished at Fri Feb 7 19:58:14 2020

Elapsed time: 5.09085s

Amount of swaps 9475712

Sorted in correct order!

===================================

Quick Sort:

Finished at Fri Feb 7 19:58:14 2020

Elapsed time: 0.00260755s

Amount of swaps 14

Sorted in correct order!

Finished at Fri Feb 7 19:58:14 2020

Elapsed time: 5.1388e-05s

Amount of swaps 42

Sorted in correct order!

Finished at Fri Feb 7 19:58:14 2020

Elapsed time: 0.000175185s

Amount of swaps 196

Sorted in correct order!

Finished at Fri Feb 7 19:58:14 2020

Elapsed time: 0.00203724s

Amount of swaps 2214

Sorted in correct order!

Finished at Fri Feb 7 19:58:14 2020

Elapsed time: 0.0107297s

Amount of swaps 4011

Sorted in correct order!

Finished at Fri Feb 7 19:58:14 2020

Elapsed time: 0.0166865s

Amount of swaps 5677

Sorted in correct order!

Finished at Fri Feb 7 19:58:14 2020

Elapsed time: 0.0663078s

Amount of swaps 22682

Sorted in correct order!

Finished at Fri Feb 7 19:58:14 2020

Elapsed time: 0.139378s

Amount of swaps 37484

Sorted in correct order!

Finished at Fri Feb 7 19:58:14 2020

Elapsed time: 0.188735s

Amount of swaps 59626

Sorted in correct order!

Finished at Fri Feb 7 19:58:15 2020

Elapsed time: 0.461957s

Amount of swaps 108962

Sorted in correct order!

Finished at Fri Feb 7 19:58:16 2020

Elapsed time: 1.54238s

Amount of swaps 239666

Sorted in correct order!

Finished at Fri Feb 7 19:58:19 2020

Elapsed time: 2.2218s

Amount of swaps 296484

Sorted in correct order!

Finished at Fri Feb 7 19:58:25 2020

Elapsed time: 5.73937s

Amount of swaps 548055

Sorted in correct order!

Finished at Fri Feb 7 20:00:38 2020

Elapsed time: 132.695s

Amount of swaps 2715134

Sorted in correct order!

===================================

Sorting in Worst case!

Quick Sort:

Finished at Fri Feb 7 20:00:38 2020

Elapsed time: 2.4996e-05s

Amount of swaps 25

Sorted in correct order!

Finished at Fri Feb 7 20:00:38 2020

Elapsed time: 7.0688e-05s

Amount of swaps 100

Sorted in correct order!

Finished at Fri Feb 7 20:00:38 2020

Elapsed time: 0.00039942s

Amount of swaps 625

Sorted in correct order!

Finished at Fri Feb 7 20:00:38 2020

Elapsed time: 0.0744401s

Amount of swaps 62500

Sorted in correct order!

Finished at Fri Feb 7 20:00:38 2020

Elapsed time: 0.123423s

Amount of swaps 122500

Sorted in correct order!

Finished at Fri Feb 7 20:00:38 2020

Elapsed time: 0.204924s

Amount of swaps 250000

Sorted in correct order!

Finished at Fri Feb 7 20:00:42 2020

Elapsed time: 3.9049s

Amount of swaps 6250000

Sorted in correct order!

Finished at Fri Feb 7 20:00:51 2020

Elapsed time: 8.63583s

Amount of swaps 14062500

Sorted in correct order!

Finished at Fri Feb 7 20:01:13 2020

Elapsed time: 21.8774s

Amount of swaps 36000000

Sorted in correct order!

Finished at Fri Feb 7 20:02:44 2020

Elapsed time: 90.9781s

Amount of swaps 156250000

Sorted in correct order!

===================================