**Lab Description:**

\*\*Description in README

\*\*Not on Lab Write up

**Question C:**

**Main.cpp:**

#include <iostream> // cout, endl

#include <stdio.h> // printf

#include <math.h> // pow()

#include <time.h> // time functions

#include <chrono> // chrono

struct Heap {

int\* arr; // the underlying array

int length; // the size of the array

int heap\_size; // is the proportion of the array that is a valid heap

Heap(int\* a, int len) {

arr = a;

length = len;

heap\_size = 0;

}

~Heap() {

delete [] arr;

}

int& operator[](int i) {

return arr[i];

}

};

void MaxHeapify(Heap&, int); // Heapify function that will create a valid heap on a trio of elements

void BuildMaxHeap(Heap&); // Loop to create a valid heap throughout the array

void HeapSort(Heap&); // Creates the final sorting of the heap

void swap(int&, int&); // Swaps two elements taken by reference

int\* makeArray(int); // Creates a dynamic array of specified length

int\* makeArray2(int); // Creates a dynamic pre-sorted array

int\* makeArray3(int); // Creates a dynamic sorted descending array

void printArray(Heap&); // Prints an array in a line

void timeSort(void (\*sort)(Heap&), Heap&); // Calculates the time it takes to sort an array

bool isSorted(Heap&); // Takes a Heap struct and determines if array is sorted correctly

namespace Counters { // To get rid of ambiguity on a global swap counter variable

int count = 0;

}

int main() {

srand(time(NULL));

int sizesLen = 16;

int sizes[] = {10, 20, 50, 500, 700, 1000, 5000, 7500, 12000, 25000, 50000, 60000,

100000, 500000, 800000, 1000000};

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

std::cout << "Testing the print algorithm and a mini array for basic demonstration\n";

Heap heapTest(makeArray(15), 15);

printArray(heapTest);

HeapSort(heapTest);

printArray(heapTest);

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

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

std::cout << "Large Test cases of Heap sort: " << std::endl;

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

std::cout << "Sorting array of " << sizes[i] << " elements..." << std::endl;

Heap newHeap(makeArray(sizes[i]), sizes[i]);

timeSort(HeapSort, newHeap);

}

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

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

std::cout << "Large Test cases already sorted Heap sort: " << std::endl;

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

std::cout << "Sorting array of " << sizes[i] << " elements..." << std::endl;

Heap newHeap(makeArray2(sizes[i]), sizes[i]);

timeSort(HeapSort, newHeap);

}

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

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

std::cout << "Large Test cases sorted in descending Heap sort: " << std::endl;

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

std::cout << "Sorting array of " << sizes[i] << " elements..." << std::endl;

Heap newHeap(makeArray3(sizes[i]), sizes[i]);

timeSort(HeapSort, newHeap);

}

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

return 0;

}

/\*

\* timeSort Function:

\* Calculates the time it takes for heap sort to sort a given amount of elements

\*/

void timeSort(void (\*sort)(Heap&), Heap& sample) {

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

sort(sample);

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 " << Counters::count << std::endl;

if (isSorted(sample)) {

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

} else {

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

}

Counters::count = 0;

}

/\*

\* isSorted Function:

\* Takes a Heap struct and determines if the array contained is sorted correctly in ascending

\* order after a heap sort takes place

\*/

bool isSorted(Heap& arr) {

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

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

return false;

}

}

return true;

}

/\*

\* printArray Function:

\* Prints the array for function testing

\*/

void printArray(Heap& arr) {

for (int i = 0; i < arr.length; i++) {

for (int j = 0; j < pow(2, i) && j + pow(2, i) < arr.length; j++) {

std::cout << arr[j + pow(2, i) - 1] << " ";

}

std::cout << std::endl;

}

}

/\*

\* makeArray Function:

\* Takes a length and then makes a dynamic array that is the size of the length passed

\*/

int\* makeArray(int len) {

int\* rtn = new int[len];

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

rtn[i] = 1 + rand() & 50;

}

return rtn;

}

/\*

\* makeArray2 Function:

\* Takes a length and then makes a dynamic array that is the size of the length passed already sorted correctly

\*/

int\* makeArray2(int len) {

int\* rtn = new int[len];

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

rtn[i] = i / 2;

}

return rtn;

}

/\*

\* makeArray3 Function:

\* Takes a length and then makes a dynamic array that is the size of the length passed that is sorted in descending

\*/

int\* makeArray3(int len) {

int\* rtn = new int[len];

for (int i = 0, j = len / 2; i < len; i++, j--) {

rtn[i] = j;

}

return rtn;

}

/\*

\* MaxHeapify Function:

\* Heapify function that will create a valid heap on a trio of elements

\*/

void MaxHeapify(Heap& arr, int i) {

int Lchild = 2 \* i + 1;

int Rchild = 2 \* i + 2;

int largest = i;

if (Lchild < arr.heap\_size && arr[Lchild] > arr[largest]) {

largest = Lchild;

}

if (Rchild < arr.heap\_size && arr[Rchild] > arr[largest]) {

largest = Rchild;

}

if (largest != i) {

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

MaxHeapify(arr, largest);

}

}

/\*

\* BuildMaxHeap Function:

\* Loops on the array until the whole array is a heap

\*/

void BuildMaxHeap(Heap& arr) {

arr.heap\_size = arr.length;

for (int i = arr.length - 1; i >= 0; i--) {

MaxHeapify(arr, i);

}

}

/\*

\* HeapSort Function:

\* Makes sure that the array is a valid heap, then sorts the array correctly

\*/

void HeapSort(Heap& arr) {

BuildMaxHeap(arr);

for (int i = arr.length - 1; i > 0; i--) {

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

arr.heap\_size -= 1;

MaxHeapify(arr, 0);

}

}

/\*

\* swap Function:

\* Takes two elements by reference and swaps them in place

\*/

void swap(int& l, int& r) {

int temp = l;

l = r;

r = temp;

Counters::count++;

}

**Sample Output:**

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

Testing the print algorithm and a mini array for basic demonstration

2

50 50

18 50 50 50

18 2 18 16 32 50 32

2

2 16

16 18 18 18

32 32 50 50 50 50 50

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

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

Large Test cases of Heap sort:

Sorting array of 10 elements...

Finished at Sat Feb 15 13:55:52 2020

Elapsed time: 1.29e-06s

Amount of swaps 61

Sorted in correct order!

Sorting array of 20 elements...

Finished at Sat Feb 15 13:55:52 2020

Elapsed time: 3.352e-06s

Amount of swaps 71

Sorted in correct order!

Sorting array of 50 elements...

Finished at Sat Feb 15 13:55:52 2020

Elapsed time: 1.0211e-05s

Amount of swaps 226

Sorted in correct order!

Sorting array of 500 elements...

Finished at Sat Feb 15 13:55:52 2020

Elapsed time: 0.000153758s

Amount of swaps 3555

Sorted in correct order!

Sorting array of 700 elements...

Finished at Sat Feb 15 13:55:52 2020

Elapsed time: 0.000230409s

Amount of swaps 5365

Sorted in correct order!

Sorting array of 1000 elements...

Finished at Sat Feb 15 13:55:52 2020

Elapsed time: 0.000354335s

Amount of swaps 8079

Sorted in correct order!

Sorting array of 5000 elements...

Finished at Sat Feb 15 13:55:52 2020

Elapsed time: 0.00238644s

Amount of swaps 50684

Sorted in correct order!

Sorting array of 7500 elements...

Finished at Sat Feb 15 13:55:52 2020

Elapsed time: 0.00395578s

Amount of swaps 79416

Sorted in correct order!

Sorting array of 12000 elements...

Finished at Sat Feb 15 13:55:53 2020

Elapsed time: 0.01074s

Amount of swaps 133572

Sorted in correct order!

Sorting array of 25000 elements...

Finished at Sat Feb 15 13:55:53 2020

Elapsed time: 0.0295474s

Amount of swaps 301280

Sorted in correct order!

Sorting array of 50000 elements...

Finished at Sat Feb 15 13:55:53 2020

Elapsed time: 0.0660154s

Amount of swaps 646350

Sorted in correct order!

Sorting array of 60000 elements...

Finished at Sat Feb 15 13:55:53 2020

Elapsed time: 0.0678471s

Amount of swaps 786583

Sorted in correct order!

Sorting array of 100000 elements...

Finished at Sat Feb 15 13:55:53 2020

Elapsed time: 0.129755s

Amount of swaps 1375457

Sorted in correct order!

Sorting array of 500000 elements...

Finished at Sat Feb 15 13:55:54 2020

Elapsed time: 0.678618s

Amount of swaps 7877854

Sorted in correct order!

Sorting array of 800000 elements...

Finished at Sat Feb 15 13:55:55 2020

Elapsed time: 1.10244s

Amount of swaps 13091416

Sorted in correct order!

Sorting array of 1000000 elements...

Finished at Sat Feb 15 13:55:56 2020

Elapsed time: 1.19851s

Amount of swaps 16638731

Sorted in correct order!

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

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

Large Test cases already sorted Heap sort:

Sorting array of 10 elements...

Finished at Sat Feb 15 13:55:56 2020

Elapsed time: 1.634e-06s

Amount of swaps 29

Sorted in correct order!

Sorting array of 20 elements...

Finished at Sat Feb 15 13:55:56 2020

Elapsed time: 3.534e-06s

Amount of swaps 77

Sorted in correct order!

Sorting array of 50 elements...

Finished at Sat Feb 15 13:55:56 2020

Elapsed time: 1.0918e-05s

Amount of swaps 263

Sorted in correct order!

Sorting array of 500 elements...

Finished at Sat Feb 15 13:55:56 2020

Elapsed time: 0.000172719s

Amount of swaps 4359

Sorted in correct order!

Sorting array of 700 elements...

Finished at Sat Feb 15 13:55:56 2020

Elapsed time: 0.000266743s

Amount of swaps 6520

Sorted in correct order!

Sorting array of 1000 elements...

Finished at Sat Feb 15 13:55:56 2020

Elapsed time: 0.000443953s

Amount of swaps 9760

Sorted in correct order!

Sorting array of 5000 elements...

Finished at Sat Feb 15 13:55:56 2020

Elapsed time: 0.00246655s

Amount of swaps 60941

Sorted in correct order!

Sorting array of 7500 elements...

Finished at Sat Feb 15 13:55:56 2020

Elapsed time: 0.00417607s

Amount of swaps 95494

Sorted in correct order!

Sorting array of 12000 elements...

Finished at Sat Feb 15 13:55:56 2020

Elapsed time: 0.0146253s

Amount of swaps 161758

Sorted in correct order!

Sorting array of 25000 elements...

Finished at Sat Feb 15 13:55:56 2020

Elapsed time: 0.0312925s

Amount of swaps 363897

Sorted in correct order!

Sorting array of 50000 elements...

Finished at Sat Feb 15 13:55:56 2020

Elapsed time: 0.0626761s

Amount of swaps 777912

Sorted in correct order!

Sorting array of 60000 elements...

Finished at Sat Feb 15 13:55:56 2020

Elapsed time: 0.091777s

Amount of swaps 946578

Sorted in correct order!

Sorting array of 100000 elements...

Finished at Sat Feb 15 13:55:56 2020

Elapsed time: 0.132884s

Amount of swaps 1656183

Sorted in correct order!

Sorting array of 500000 elements...

Finished at Sat Feb 15 13:55:57 2020

Elapsed time: 0.766013s

Amount of swaps 9419865

Sorted in correct order!

Sorting array of 800000 elements...

Finished at Sat Feb 15 13:55:59 2020

Elapsed time: 1.48336s

Amount of swaps 15646389

Sorted in correct order!

Sorting array of 1000000 elements...

Finished at Sat Feb 15 13:56:01 2020

Elapsed time: 1.97536s

Amount of swaps 19812020

Sorted in correct order!

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

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

Large Test cases sorted in descending Heap sort:

Sorting array of 10 elements...

Finished at Sat Feb 15 13:56:01 2020

Elapsed time: 1.336e-06s

Amount of swaps 21

Sorted in correct order!

Sorting array of 20 elements...

Finished at Sat Feb 15 13:56:01 2020

Elapsed time: 3.161e-06s

Amount of swaps 62

Sorted in correct order!

Sorting array of 50 elements...

Finished at Sat Feb 15 13:56:01 2020

Elapsed time: 9.519e-06s

Amount of swaps 207

Sorted in correct order!

Sorting array of 500 elements...

Finished at Sat Feb 15 13:56:01 2020

Elapsed time: 0.000163132s

Amount of swaps 3676

Sorted in correct order!

Sorting array of 700 elements...

Finished at Sat Feb 15 13:56:01 2020

Elapsed time: 0.000238282s

Amount of swaps 5498

Sorted in correct order!

Sorting array of 1000 elements...

Finished at Sat Feb 15 13:56:01 2020

Elapsed time: 0.000399952s

Amount of swaps 8316

Sorted in correct order!

Sorting array of 5000 elements...

Finished at Sat Feb 15 13:56:01 2020

Elapsed time: 0.00743269s

Amount of swaps 53436

Sorted in correct order!

Sorting array of 7500 elements...

Finished at Sat Feb 15 13:56:01 2020

Elapsed time: 0.00853754s

Amount of swaps 84404

Sorted in correct order!

Sorting array of 12000 elements...

Finished at Sat Feb 15 13:56:01 2020

Elapsed time: 0.0222734s

Amount of swaps 143062

Sorted in correct order!

Sorting array of 25000 elements...

Finished at Sat Feb 15 13:56:01 2020

Elapsed time: 0.03432s

Amount of swaps 326586

Sorted in correct order!

Sorting array of 50000 elements...

Finished at Sat Feb 15 13:56:01 2020

Elapsed time: 0.0737035s

Amount of swaps 698892

Sorted in correct order!

Sorting array of 60000 elements...

Finished at Sat Feb 15 13:56:01 2020

Elapsed time: 0.0955604s

Amount of swaps 854794

Sorted in correct order!

Sorting array of 100000 elements...

Finished at Sat Feb 15 13:56:01 2020

Elapsed time: 0.174857s

Amount of swaps 1497434

Sorted in correct order!

Sorting array of 500000 elements...

Finished at Sat Feb 15 13:56:02 2020

Elapsed time: 1.03545s

Amount of swaps 8668450

Sorted in correct order!

Sorting array of 800000 elements...

Finished at Sat Feb 15 13:56:03 2020

Elapsed time: 1.14503s

Amount of swaps 14384464

Sorted in correct order!

Sorting array of 1000000 elements...

Finished at Sat Feb 15 13:56:05 2020

Elapsed time: 1.32517s

Amount of swaps 18333408

Sorted in correct order!

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