Genevieve Leach

CS 241 – 01

17 November 2016

Project 1

**Source Code****import** java.util.Arrays;  
  
**public class** LeachGp2 {  
  
 **private static int** *swapAvg*;  
 **private static int** *swapCount*;  
 **private static boolean** *print*;  
 **private static boolean** *avg*;  
 **private static int**[] *orig*;  
  
 **public static void** main(String[] args) {  
 *print* = **true**;  
 *avg* = **false**;  
 **int**[] oneInput = {12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25};  
 *orig* = Arrays.*copyOf*(oneInput, oneInput.**length**);  
 System.***out***.println(**"Part 1:"**);  
 System.***out***.println();  
 *heapSort*(oneInput,oneInput.**length**);  
 System.***out***.println();  
 System.***out***.println(**"Original Array: \n"** + Arrays.*toString*(*orig*));  
 System.***out***.println(**"Sorted Array: \n"** + Arrays.*toString*(oneInput));  
 **if**(*avg*) {  
 System.***out***.println(**"Number of Swaps: "** + *swapCount*);  
 }  
 System.***out***.println();  
  
 *print* = **false**;  
 *avg* = **true**;  
 **int**[] twoInput = **new int**[100];  
 System.***out***.println(**"Part 2:"**);  
 System.***out***.println();  
 **for**(**int** i = 0; i < 20; i++) {  
 System.***out***.println(**"Sort #"** + (i+1) + **": "**);  
 *genTwoInput*(twoInput);  
 *swapCount* = 0;  
 *heapSort*(twoInput,twoInput.**length**);  
 System.***out***.println(**"Number of swaps: "** + *swapCount*);  
 }  
 System.***out***.println();  
 System.***out***.println(**"Average number of swaps: "** + *swapAvg*/20);  
 }  
  
 **private static void** genTwoInput(**int**[] twoInput) {  
 **for** (**int** i = 0; i < 100; i++) {  
 twoInput[i] = 100 + (**int**) (Math.*random*() \* 800);  
 }  
 }  
  
 **public static void** reheap(**int**[]in, **int** i, **int** max) {  
 **boolean** done = **false**;  
 **int** orphan = in[i];  
 **int** left = 2\*i+1;  
 **while**(!done && (left<=max)) {  
 **int** larger = left;  
 **int** right = left+1;  
 **if**(right<=max && in[right] > in[larger]) {  
 larger = right;  
 }  
 **if**(orphan < in[larger]) {  
 in[i] = in[larger];  
 i = larger;  
 left = 2\*i+1;  
 } **else** {  
 done = **true**;  
 }  
 in[i] = orphan;  
 }  
 }  
  
 **public static void** heapSort(**int**[] in, **int** n) {  
 **for**(**int** i = n/2 -1; i>=0; i--) {  
 *reheap*(in, i, n-1);  
 }  
 *swap*(in, 0, n-1);  
 **for**(**int** max = n-2; max>0; max--) {  
 *reheap*(in, 0, max);  
 *swap*(in, 0, max);  
 }  
 }  
  
 **public static void** swap(**int** in[], **int** i, **int** j) {  
 **int** temp = in[i];  
 in[i] = in[j];  
 in[j] = temp;  
 **if**(*avg*) {  
 *swapAvg*++;  
 *swapCount*++;  
 }  
 **if**(*print*) {  
 System.***out***.println(**"Original Array: \n"** + Arrays.*toString*(*orig*));  
 System.***out***.println(**"After Swap: \n"** + Arrays.*toString*(in));  
 }  
 }  
}

**Sample Input/Output**

Part 1:

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[15, 28, 27, 24, 25, 26, 23, 21, 18, 20, 22, 12, 16, 14, 10, 17, 19, 11, 13, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[13, 25, 27, 24, 22, 26, 23, 21, 18, 20, 15, 12, 16, 14, 10, 17, 19, 11, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[11, 25, 26, 24, 22, 16, 23, 21, 18, 20, 15, 12, 13, 14, 10, 17, 19, 27, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[19, 25, 23, 24, 22, 16, 14, 21, 18, 20, 15, 12, 13, 11, 10, 17, 26, 27, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[17, 24, 23, 21, 22, 16, 14, 19, 18, 20, 15, 12, 13, 11, 10, 25, 26, 27, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[10, 22, 23, 21, 20, 16, 14, 19, 18, 17, 15, 12, 13, 11, 24, 25, 26, 27, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[11, 22, 16, 21, 20, 13, 14, 19, 18, 17, 15, 12, 10, 23, 24, 25, 26, 27, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[10, 21, 16, 19, 20, 13, 14, 11, 18, 17, 15, 12, 22, 23, 24, 25, 26, 27, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[12, 20, 16, 19, 17, 13, 14, 11, 18, 10, 15, 21, 22, 23, 24, 25, 26, 27, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[15, 19, 16, 18, 17, 13, 14, 11, 12, 10, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[10, 18, 16, 15, 17, 13, 14, 11, 12, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[12, 17, 16, 15, 10, 13, 14, 11, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[11, 15, 16, 12, 10, 13, 14, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[11, 15, 14, 12, 10, 13, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[13, 12, 14, 11, 10, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[10, 12, 13, 11, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[11, 12, 10, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

After Swap:

[10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]

Original Array:

[12, 19, 26, 13, 20, 27, 14, 21, 28, 15, 22, 29, 16, 23, 10, 17, 24, 11, 18, 25]

Sorted Array:

[10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]

Part 2:

Sort #1:

Number of swaps: 99

Sort #2:

Number of swaps: 99

Sort #3:

Number of swaps: 99

Sort #4:

Number of swaps: 99

Sort #5:

Number of swaps: 99

Sort #6:

Number of swaps: 99

Sort #7:

Number of swaps: 99

Sort #8:

Number of swaps: 99

Sort #9:

Number of swaps: 99

Sort #10:

Number of swaps: 99

Sort #11:

Number of swaps: 99

Sort #12:

Number of swaps: 99

Sort #13:

Number of swaps: 99

Sort #14:

Number of swaps: 99

Sort #15:

Number of swaps: 99

Sort #16:

Number of swaps: 99

Sort #17:

Number of swaps: 99

Sort #18:

Number of swaps: 99

Sort #19:

Number of swaps: 99

Sort #20:

Number of swaps: 99

Average number of swaps: 99