Assignment for Algorithms, Data Structures and Problem Solving

Esteban Rueda Martínez

[esrueda@ucm.es](mailto:esrueda@ucm.es)

Swedish Number: 19950317-T499

Algorithms chosen:

* Shell Sort (Medium)
* Gnome Sort (Easy)
* Selection Sort (Easy)

Shell Sort:

The Shell Sorting algorithm takes pairs of elements far apart from each other, and then it progressively reduces the gap between elements to be compared.

In the code, I create a gap which is half the length of the array, and make it by half for each time the for loop is finished.

Shell Sort Code:

/\* Esteban Rueda Martínez \*/

/\* esrueda@ucm.es \*/

/\* Swedish Number: 19950317-T499 \*/

/\* Libraries \*/

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\* Function Declaration \*/

void shellSort(int a[],int length);

void printArray(int a[],int length);

int main()

{

int a[] = {0, -3, 10, 3, 1, 4, 7, 1}; // array with given numbers

int length = 8; // length of the array

/\* For trying with 100.000 numbers\*/

/\*int length = 100000;

int a[100000];\*/

printArray(a,length); // we print the array as it is before any changes

shellSort(a,length); // we call the sorting function

}

/\* Shell Sort Algorithm \*/

void shellSort(int a[],int length){

int i,j;

int gap;

/\* To try with 100000 random numbers \*/

/\*srand(time(NULL));

for(i=0;i<100000;i++){

a[i]=(rand());

}

\*/

for( gap = length / 2 ; gap > 0; gap = gap / 2) // The gap is half as long as the array length.

{

for( j = gap; j < length; j++){

for(i = j - gap ; i >= 0 ; i = i-gap){

if(a[i + gap] >= a[i]){

break;

}

else{

int swap = a[i]; // We swap the values of a[i] with the values in a[i + gap]

a[i]= a[i + gap];

a[i + gap] = swap;

printArray(a,length); // We call the function printArray() to print in each for loop

}

}

}

}

}

/\* Printing the array \*/

void printArray(int a[],int length){

int i;

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

{

printf(" %d ",a[i]); // We print each value of the array one by one

}

printf("\n");

}

Shell Sort Output:

PS C:\Users\esRueda\Desktop\Algorithms> gcc -Wall -o Assignment2 Assignment2.c

PS C:\Users\esRueda\Desktop\Algorithms> ./Assignment2.exe

0 -3 10 3 1 4 7 1

0 -3 7 3 1 4 10 1

0 -3 7 1 1 4 10 3

0 -3 1 1 7 4 10 3

0 -3 1 1 7 3 10 4

-3 0 1 1 7 3 10 4

-3 0 1 1 3 7 10 4

-3 0 1 1 3 7 4 10

-3 0 1 1 3 4 7 10

Gnome Sort:

This is a very easy to understand description, mentioned in Wikipedia, about how the Gnome Sorting Algorithm works:

Here is how a garden gnome sorts a line of flower pots.

Basically, he looks at the flower pot next to him and the previous one; if they are in the right order he steps one pot forward, otherwise, he swaps them and steps one pot backward.

Boundary conditions: if there is no previous pot, he steps forwards; if there is no pot next to him, he is done.

Gnome Sort Code:

/\* Esteban Rueda Martínez \*/

/\* esrueda@ucm.es \*/

/\* Swedish Number: 19950317-T499 \*/

/\* Libraries \*/

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\* Function Declaration \*/

void gnomeSort(int a[],int length);

void printArray(int a[],int length);

int main()

{

int a[] = {0, -3, 10, 3, 1, 4, 7, 1};

int length = 8;

/\* For trying with 100.000 numbers\*/

/\*int length = 100000;

int a[100000];\*/

printArray(a,length);

gnomeSort(a,length);

}

/\* Selection Sort Algorithm \*/

void gnomeSort(int \* a, int length){

int pos = 0;

while (pos < length){

if (pos == 0 || a[pos] >= a[pos-1])

{

pos = pos + 1;

}

else{

int change = a[pos];

a[pos] = a[pos-1];

a[pos-1] = change;

pos = pos - 1;

printArray(a,length);

}

}

}

/\* Printing the array \*/

void printArray(int a[],int length){

int i;

for(i = 0; i < length; i++){

printf(" %d ",a[i]);

}

printf("\n");

}

Gnome Sort Output:

PS C:\Users\esRueda\Desktop\Algorithms> gcc -Wall -o Assignment Assignment.c

PS C:\Users\esRueda\Desktop\Algorithms> ./Assignment.exe

0 -3 10 3 1 4 7 1

-3 0 10 3 1 4 7 1

-3 0 3 10 1 4 7 1

-3 0 3 1 10 4 7 1

-3 0 1 3 10 4 7 1

-3 0 1 3 4 10 7 1

-3 0 1 3 4 7 10 1

-3 0 1 3 4 7 1 10

-3 0 1 3 4 1 7 10

-3 0 1 3 1 4 7 10

-3 0 1 1 3 4 7 10

Selection Sort:

The algorithm divides the input list into two parts: the sublist of items already sorted, which is built up from left to right at the front (left) of the list, and the sublist of items remaining to be sorted that occupy the rest of the list. Initially, the sorted sublist is empty and the unsorted sublist is the entire input list. The algorithm proceeds by finding the smallest (or largest, depending on sorting order) element in the unsorted sublist, exchanging (swapping) it with the leftmost unsorted element (putting it in sorted order), and moving the sublist boundaries one element to the right.

Selection Sort Code:

/\* Esteban Rueda Martínez \*/

/\* esrueda@ucm.es \*/

/\* Swedish Number: 19950317-T499 \*/

/\* Libraries \*/

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\* Function Declaration \*/

void selectionSort(int a[],int length);

void printArray(int a[],int length);

int main()

{

int a[] = {0, -3, 10, 3, 1, 4, 7, 1}; // array with given numbers

int length = 8;

/\* For trying with 100.000 numbers\*/

/\*int length = 100000;

int a[100000];\*/

printArray(a,length);

selectionSort(a,length);

}

/\* Selection Sort Algorithm \*/

void selectionSort(int \* a, int length){

int i,j;

int minimumNumber;

/\* To try with 100000 random numbers \*/

/\*srand(time(NULL));

for(i=0;i<100000;i++){

a[i]=(rand());

}

\*/

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

minimumNumber = i;

for (j = i + 1; j < length ; j++) // Finding the minimum number.

{

if(a[j] < a[minimumNumber])

{

minimumNumber = j;

}

}

if (minimumNumber != i)

{

int swap = a[i];

a[i] = a[minimumNumber];

a[minimumNumber] = swap;

printArray(a,length);

}

}

}

/\* Printing the array \*/

void printArray(int a[],int length){

int i;

for(i = 0; i < length; i++){

printf(" %d ",a[i]);

}

printf("\n");

}

Selection Sort Output:

PS C:\Users\esRueda\Desktop\Algorithms> gcc -Wall -o Assignment1 Assignment1.c

PS C:\Users\esRueda\Desktop\Algorithms> ./Assignment1.exe

0 -3 10 3 1 4 7 1

-3 0 10 3 1 4 7 1

-3 0 1 3 10 4 7 1

-3 0 1 1 10 4 7 3

-3 0 1 1 3 4 7 10

External Sources:

Create an array of random integers:

<https://stackoverflow.com/questions/13802399/generate-a-random-array-in-c>

Shell sort:

http://en.wikipedia.org/wiki/Shell\_sort

Gnome sort:

http://en.wikipedia.org/wiki/Gnome\_sort

Selection sort:

http://en.wikipedia.org/wiki/Selection\_sort