

CST 370 Programming Assignment (Sorting Algorithm 2)

In this assignment you will implement a sorting algorithm, which has not been covered in class. We will call this sorting algorithm CoolSort(). We will take advantage of insertion sort for designing this sorting algorithm. The description of the sorting algorithm is as follows.

We will first choose a decreasing sequence of numbers that ends at 1. For example, let us consider the sequence of step sizes $H = 5, 3, 1$. You can choose any decreasing sequence. Note that the first element of the sequence is less than the number of elements in the array.

For each H , sort sub-arrays that start at arbitrary element and include every H th element using insertion sort. For example, consider the following array

$a = [62\ 83\ 18\ 53\ 07\ 17\ 95\ 86\ 47\ 69\ 25\ 28]$.

An example run of CoolSort with gaps 5, 3 and 1 is shown below.

$a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7\ a_8\ a_9\ a_{10}\ a_{11}\ a_{12}$

Input 628318530717958647692528 $H=5$ 172818470725838653696295 $H=3$
170718472825696253838695 $H=1$ 071718252847536269838695

The first pass, 5-sorting, performs insertion sort on separate subarrays (a_1, a_6, a_{11}) , (a_2, a_7, a_{12}) , (a_3, a_8) , (a_4, a_9) , (a_5, a_{10}) . For instance, it changes the subarray (a_1, a_6, a_{11}) from $(62, 17, 25)$ to $(17, 25, 62)$. The next pass, 3-sorting, performs insertion sort on the subarrays (a_1, a_4, a_7, a_{10}) , (a_2, a_5, a_8, a_{11}) , (a_3, a_6, a_9, a_{12}) . The last pass, 1-sorting, is an ordinary insertion sort of the entire array (a_1, \dots, a_{12}) .

As the example illustrates, the subarrays that CoolSort operates on are initially short; later they are longer but almost ordered.

Though unintuitive, it can be shown that the above algorithm has a runtime of $O(N^{3/2})$ in comparison to selection sort which has a runtime of $O(N^2)$. That is why this algorithm is cool!

Sample Test Case 1

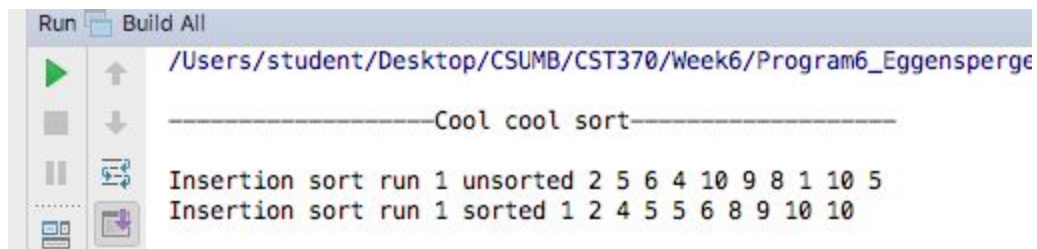
Input = [2, 5, 6, 4, 10, 9, 8, 1, 10, 5] and H = [5, 3,1] Output = [1, 2, 4, 5, 5, 6, 8, 9, 10, 10]

Given the test cases provided and their implementation into the program, we receive a mirror example.

```
// Const array sizes for main array and sequence array
const int X_ARRA_SIZE_RUN1 = 10;
const int SEQUENCE_ARRA_SIZE_RUN1 = 3;
const int X_ARRA_SIZE_RUN2 = 10;
const int SEQUENCE_ARRA_SIZE_RUN2 = 3;

// Test Run 1 array
int H_Run1[X_ARRA_SIZE_RUN1] = {5, 3, 1};
int sortArr_Run1[X_ARRA_SIZE_RUN1] = {2, 5, 6, 4, 10, 9, 8, 1, 10, 5};

// Test Run 2
int H_Run2[X_ARRA_SIZE_RUN2] = {5, 2, 1};
int sortArr_Run2[X_ARRA_SIZE_RUN2] = {2, 5, 9, 4, 10, 7, 8, 1, 11, 5};
```

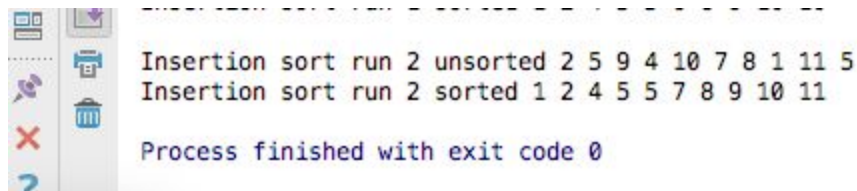


```
Run Build All
/Users/student/Desktop/CSUMB/CST370/Week6/Program6_Eggensperge

-----Cool cool sort-----
Insertion sort run 1 unsorted 2 5 6 4 10 9 8 1 10 5
Insertion sort run 1 sorted 1 2 4 5 5 6 8 9 10 10
```

Sample Test Case 2

Input = [2, 5, 9, 4, 10, 7, 8, 1, 11, 5] and H = [5, 2,1] Output = [1, 2, 4, 5, 5, 7, 8, 9, 10, 11]



```
Insertion sort run 2 unsorted 2 5 9 4 10 7 8 1 11 5
Insertion sort run 2 sorted 1 2 4 5 5 7 8 9 10 11

Process finished with exit code 0
```