Mission 14: Searching and Sorting III

Start date: 19 October 2017
Due: 21 October 2017, 23:59

Readings:

- None

IMPORTANT WARNING:

For this mission, we will be using an external program to aid in the visualisation of our solution. As a result, there are some additional constraints applied to the solutions for this mission. In this mission, we do not permit the declaration of any other functions other than the one already provided by the template solution. Furthermore, you must use the predefined `swap` function for this mission. Failure to do so could result in unexpected behavior from the visualiser program.

Background

Safety is one of Source Academy's number one priority. The recent crash landing incident further highlights the importance of being capable of securing an escape route to safety. Since we cannot always rely on our ability to send SOS signals, nor our ability to recall the SOS dictionary, Grandmaster Hartin Menz has decided to host a special safety course to equip cadets with the necessary skills for securing their own escape routes.

This mission has **two** task. Please remember to press "*Save*" button before you submit this mission.

Task 1:

In most crash settings, it is commonplace to find rubble of all sizes strewn all over the incident site. By cleverly manipulating these rubbles, we can reconfigure them into a sorted order, hence forming a flight of stairs for our escape route.

Unfortunately, most crash victims are often in a state of extreme mental and physical fatigue. Therefore, we need to prioritise simplicity over efficiency in these circumstances. The **Bubblesort** algorithm is one of the easiest sorting algorithms to understand and implement. Hence, it is the perfect candidate for these situations.

Bubblesort comes from a simple idea, that if we were to bubble the largest element to the last position in an array, the second largest element to the second last position, and so on until we bubble all the elements to their correct position, then we will end up with a sorted array.

Consider the following array `[1, 3, 4, 2]`.

```
// First iteration of bubbling, assume all elements not sorted
[1, 3, 4, 2] // Check 1 and 3, correct order, next
[1, 3, 4, 2] // Check 3 and 4, correct order, next
[1, 3, 4, 2] // Check 4 and 2, out of order, swap
[1, 3, 2, 4] // No more pairs to check, done
```

After our first iteration, we know that the last element must contain the largest element. Thus, the last element is in sorted order. Hence, we can ignore the last element and continue bubbling the remaining unsorted elements.

```
// Second iteration of bubbling, last element is sorted
[1, 3, 2, 4] // Check 1 and 3, correct order, next
[1, 3, 2, 4] // Check 3 and 2, out of order, swap
[1, 2, 3, 4] // No more pairs to check, done
```

After our second iteration, we know that the last element and the second last element must contain the largest and second largest element, respectively. Thus, the last 2 elements are in sorted order. Doing the same for the remaining 2 presumed-unsorted elements.

```
// Third iteration of bubbling, last 2 elements are sorted
[1, 2, 3, 4] // Check 1 and 2, correct order, next
[1, 2, 3, 4] // No more pairs to check, done
```

After our third iteration, we know that the last three elements must be in sorted order. We are left with 1 presumed-unsorted element. Since a single element is trivially sorted, we can stop here. Thus, we have obtained a sorted array from bubble sort.

For this task, complete the template `sort` function, by implementing the bubble sort algorithm for arrays.

Task 2:

Note: This task is not graded.

After implementing the algorithm, it is time to test it out in the simulation room. To do that, download either the Simulation Program for Mac or the Simulation Program for Windows.

The mouse can be used to change the direction of the character's view. You may toggle this behavior using the `Tab` key. Use the `W, A, S, D` or the `Up, Down, Left, Right` keys to navigate around the room. Once you have orient yourself in front of the command panel, copy and paste the body of your `sort` function into the panel. Next, click the `Submit` button, followed by the `confirm` button to see the reconfiguration in action.

Once you have form your flight of stairs, navigate yourself to the base of the stairs to begin climbing it. Use the `Space` button to jump. When you have reached the top, take a commemorative photo to upload and share.

Note: Cadets have been known to fall off the simulation platform while scaling the thin steps. Fortunately, this is but a simulation. In the event that you fall off the platform or find yourself stuck between walls, you may use the `R` button to reset your position.

Submission

Submit your mission on the Source Academy.

IMPORTANT: Make sure that everything for your programs to work is on the left hand side (Editor) and not in the Side Content! This is because only the programs in the Editor are preserved when your Avenger grades your submission.