PROGRAMMING LOGIC AND TECHNIQUES

Algorithmic expressions: Sorting

Sorting

One of the most commonly used techniques in programming is **sorting**. There are several sorting techniques, and in this course we will start with sorting algorithms that are easy to use.

- Selection sort
- Bubble sort
- Insertion sort

Selection sort

The first sorting algorithm we will consider is the **selection sort**. The concept of this sorting algorithm is simple:

- We have to go through all the elements of the array in order to find the lowest value, which we wish to assign to the lowest index of the array.
- Once we have passed over the entire array and have found the smallest element, its content is swapped with that of the first array index.
- Then we iterate through the array for a second time, starting at the second array element, in order to find the second-smallest value. When it's found, it is swapped with the value at the second array index.
- This is repeated until all the elements of the array are sorted.

Selection sort

- https://www.youtube.com/watch?v=Ns4TPTC8whw

- What do you think of this algorithm?
  - In terms of ease of implementation?
  - In terms of efficiency?
  - In terms of memory management?

Bubble sort

The second sorting algorithm we will consider is the **bubble sort.** Its name is derived from the fact that when this sorting algorithm is executed, the larger values are pushed toward the end of the array, much like bubbles that rise to the surface of a fluid. It works as follows:

Bubble sort

- We begin by passing over the array for the first time, starting by comparing the first element with the second element. If the first element is larger than the second, they are exchanged.
- We continue this process until we are at the second-to-last element of the array.
- If an exchange occurred in the array during the first pass, we must repeat the previous steps, but stop one box earlier than the last time.
- This is repeated until we have passed through the array one full time without there being a single exchange.

Bubble sort

- https://www.youtube.com/watch?v=lyZQPjUT5B4

- What do you think of this algorithm?
  - In terms of ease of implementation?
  - In terms of efficiency?
  - In terms of memory management?

Insertion sort

- The objective of the **insertion sort** is similar to that of sorting cards in a certain card game. The principle is as follows:

- As though we were playing cards, we have a stack of cards on the table, and an empty hand. The stack of cards represents the unsorted array, and the hand represents an empty array of the same size.

Insertion sort

- We take the first card from the stack on the table and place it in our hand at the first array index.

- We take the second card from the stack on the table and compare it with the last card in our hand (with the highest element index). If the drawn card is smaller than the last card, we make space in our hand to place the drawn card before the last card. Otherwise, we place it after the last card.

Insertion sort

- We have now 2 cards in hand. We take a third card and compare it with the last card. If it is smaller, we continue to compare the drawn card with the next card, until we find a location where our drawn card is not smaller than the next element, or we reach the first (lowest index) element of the array.

Insertion sort

- What do you think of this algorithm?
  - In terms of ease of implementation?
  - In terms of efficiency?
  - In terms of memory management?

Exercises

- Can you think of a more efficient way to sort an array of numbers if you know that many of the values are repeated?
  - Ex. If the values are

1, 2, 1, 1, 2, 3, 3, 2, 2, 1, 3, 3, 2, 3, 3, 2, 1, 1, 2, 2, 3, 1, 1, 2, 3, 1, 1, 2, 3, 1, 3, 1, 2, 1, 1, 2, 3, 3, 2, 2, 1, 3, 3, 2, 3, 3, 2, 1, 1, 2, 2, 3, 1, 1, 2, 3, 1, 1, 2, 3, 1, 3

Exercises

- Write the pseudocode for selection sort, bubble sort and insertion sort

Comparison of Sorting Algorithms

https://miro.medium.com/max/1798/1*bPpvELo9_QqQsDz7CSbwXQ.gif

Comparison of Sorting Algorithms

SORTING ALGORITHM	TIME COMPLEXITY			SPACE COMPLEXITY
	BEST CASE	AVERAGE CASE	WORST CASE	WORST CASE
Bubble Sort	$\Omega(N)$	$\Theta(N^2)$	$O(N^2)$	$O(1)$
Selection Sort	$\Omega(N^2)$	$\Theta(N^2)$	$O(N^2)$	$O(1)$
Insertion Sort	$\Omega(N)$	$\Theta(N^2)$	$O(N^2)$	$O(1)$
Merge Sort	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N \log N)$	$O(N)$
Heap Sort	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N \log N)$	$O(1)$
Quick Sort	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N^2)$	$O(N \log N)$
Radix Sort	$\Omega(N k)$	$\Theta(N k)$	$O(N k)$	$O(N + k)$
Count Sort	$\Omega(N + k)$	$\Theta(N + k)$	$O(N + k)$	$O(k)$
Bucket Sort	$\Omega(N + k)$	$\Theta(N + k)$	$O(N^2)$	$O(N)$