Summatieve Huiswerkopdracht 4

**Onderdeel 1A:**

**Pseudocode 1 Insertion Sort:**

1. Call insert to insert the element that starts at index 1 into the sorted subarray in index 0.
2. Call insert to insert the element that starts at index 2 into the sorted subarray in indices 0 through 1.
3. Call insert to insert the element that starts at index 3 into the sorted subarray in indices 0 through 2.
4. …
5. Finally, call insert to insert the element that starts at index n−1*n*−1 into the sorted subarray in indices 0 through n−2*n*−2.

Bron:

Khan Academy; <https://www.khanacademy.org/computing/computer-science/algorithms/insertion-sort/a/insertion-sort-pseudocode>

**Pseudocode 2 Insertion Sort:**

for i from 1 to N

key = a[i]

j = i - 1

while j >= 0 and a[j] > key

a[j+1] = a[j]

j = j - 1

a[j+1] = key

Bron:

Algorithmist; <http://www.algorithmist.com/index.php/Insertion_sort>

Het verschil tussen de twee codes is de vorm van uitleg. In de eerste situatie is een stappenplan gegeven van hoe de code gecodeerd zou moeten worden. In de tweede situatie is een soort code gegeven van hoe het geschreven moet worden, alleen moet die code nog omgeschreven worden naar Python-code. Omdat bij de tweede situatie alleen vertaald hoeft te worden hoef je minder werk te verrichten.

**Onderdeel 1B:**

Voor de implementatie van de Insertion Sort in Python, bekijk de code op mijn GitHub.

**Rechtstreekse link naar de opdracht:** <https://github.com/JaimeyHolm/AnalyticalSkills/blob/master/Insertion%20Sort.py>

**Onderdeel 1C:**

De Best Case Scenario is wanneer er zo min mogelijk waarden in de lijst zitten. Hierdoor is de tijd die het programma erover doet om de waarden te sorteren het kortst. De Worst Case Scenario is wanneer er meer waarden in de lijst zitten en de kleinste waarden achteraan de lijst staan. Als de kleinste waarden achteraan staan, moet per waarde een groot deel van de lijst vergeleken worden met elke voorgaande waarde, totdat de waarde op de juiste plaats staat. Dit kost erg veel tijd, wat het dus een Worst Case Scenario maakt.

**Onderdeel 2A:**

Voor de implementatie van Lineair Zoeken, bekijk de code op mijn GitHub.

**Rechtstreekse link naar de opdracht:**

<https://github.com/JaimeyHolm/AnalyticalSkills/blob/master/Lineair%20Zoeken.py>

**Onderdeel 2B:**

Voor de implementatie van Binair Zoeken, bekijk de code op mijn GitHub.

Rechtstreekse link naar de opdracht:

<https://github.com/JaimeyHolm/AnalyticalSkills/blob/master/Binair%20Zoeken.py>

GitHub-link naar alle opdrachten:

<https://github.com/JaimeyHolm/AnalyticalSkills>