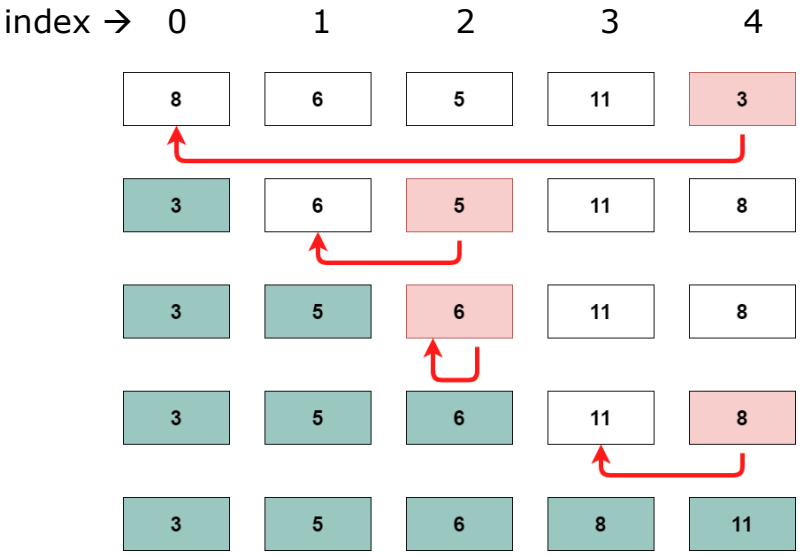
Oefen opdrachten standard algorithmen

Instructies
Afhankelijk van de opdrachten moet voor de **analyse** de volgende handelingen verrichten:

1. Bepaal wat de invoer moet zijn.
2. Bepaal wat de uitvoer moet zijn.
3. Bepaal de verwerking/het proces dat het programma moet uitvoeren.
4. Bepaal welke variabelen nodig zijn en welk type elk van deze moet hebben.
5. Bepaal de functies die nodig zijn. Beschrijf deze en bepaal de eventuele parameters die mee gegeven moeten worden en wat de mogelijke teruggeefwaarde is als er een nodig is.

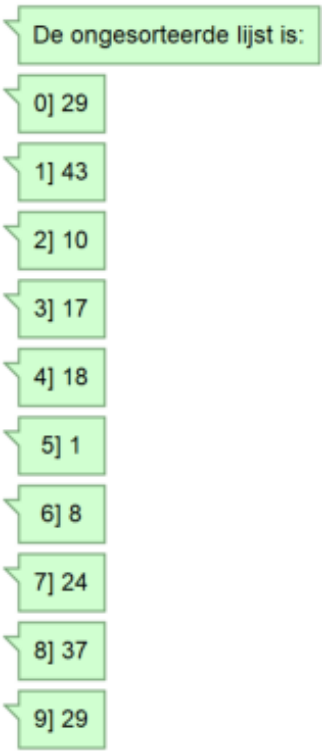
Opdracht 1 Selection Sort:
Dit is één van de eenvoudigste sorteeralgoritmen.

1. Zoek het kleinste element in een array
2. en vervang het door het eerste element.
3. Zoek het kleinste element in het ongesorteerde deel van een array
4. en vervang het door het tweede element.
5. Herhaal dit proces tot het einde en de array wordt aan het einde gesorteerd.

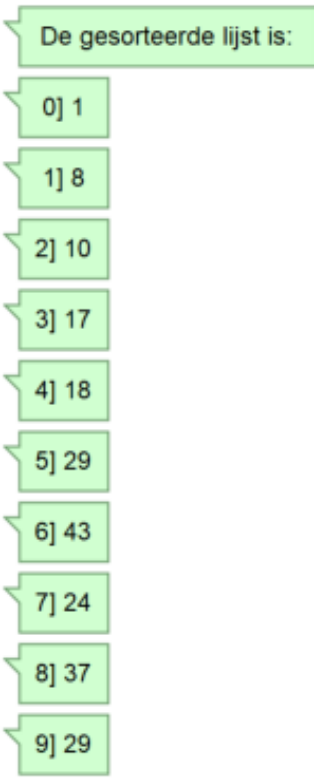


Oefen opdrachten standard algorithmen

Maak een programma dat een array vult met 10 willekeurige gehele getallen tussen 0 en 50 en deze met hun index-nummer weergeeft op het scherm.



Daarna doormiddel van het hierboven beschreven algoritme de array sorteert en opnieuw op het scherm weergeeft.



Opdracht 2 MergeSort:

Merge Sort (*Samenvoegsortering*) is een *verdeel-en-heers*-algoritme. Het is gebaseerd op het idee om de ongesorteerde array in verschillende subarrays te verdelen totdat elke subarray uit één enkel element bestaat en deze subarrays zo samen te voegen dat er een gesorteerde array ontstaat. De processtap van het samenvoegen sorteren kan als volgt worden samengevat:

- Verdelen
  - Verdeel de ongesorteerde array in verschillende subarrays totdat elke subarray slechts één element bevat.
- Samenvoegen
  - Voeg de sub-arrays samen op een manier die resulteert in een gesorteerde array en voeg ze samen totdat de originele array wordt bereikt.
- Samenvoegtechniek
  - het eerste element van de twee subarrays wordt overwogen en vergeleken. Bij het sorteren in oplopende volgorde wordt

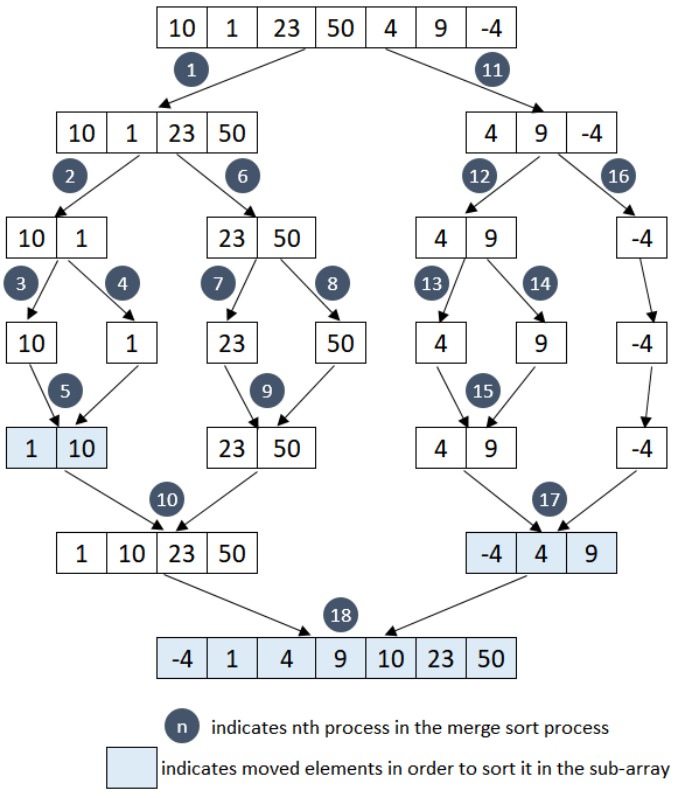
het element met de kleinere waarde uit de subarray gehaald en wordt het een nieuw element van de gesorteerde array. Dit proces wordt herhaald totdat beide subarrays zijn geleegd en de samengevoegde array een gesorteerde array wordt.

Voorbeeld:

Om **Merge Sort** te begrijpen, bekijken we een *ongesorteerde* array **[4, 9, -4]** (*array aan de rechterkant gemaakt na het 11e proces in het onderstaande diagram*) en bespreken we elke stap die is genomen om de array in oplopende volgorde te sorteren.

Bij **de eerste stap** wordt de array [4, 9, -4] verdeeld in *twee sub-arrays*. De *eerste sub-array* bevat [4, 9] en de *tweede sub-array* bevat [-4]. Omdat het aantal elementen in de eerste sub-reeks groter is dan één, wordt deze verder onderverdeeld in sub-reeksen die respectievelijk uit de elementen [4] en [9] bestaan. Omdat het aantal elementen in alle sub-reeksen één is, kan de verdere verdeling van de reeks niet plaatsvinden.

Tijdens het **samenvoegproces** worden de sub-arrays die in de laatste stap zijn gevormd, gecombineerd met behulp van het hierboven genoemde proces om een gesorteerde array te vormen. Eerst worden de sub-arrays [4] en [9] samengevoegd om een gesorteerde sub-array [4, 9] te vormen. Vervolgens worden de sub-arrays [4, 9] en [-4] samengevoegd om de uiteindelijk gesorteerde array [-4, 4, 9] te vormen.

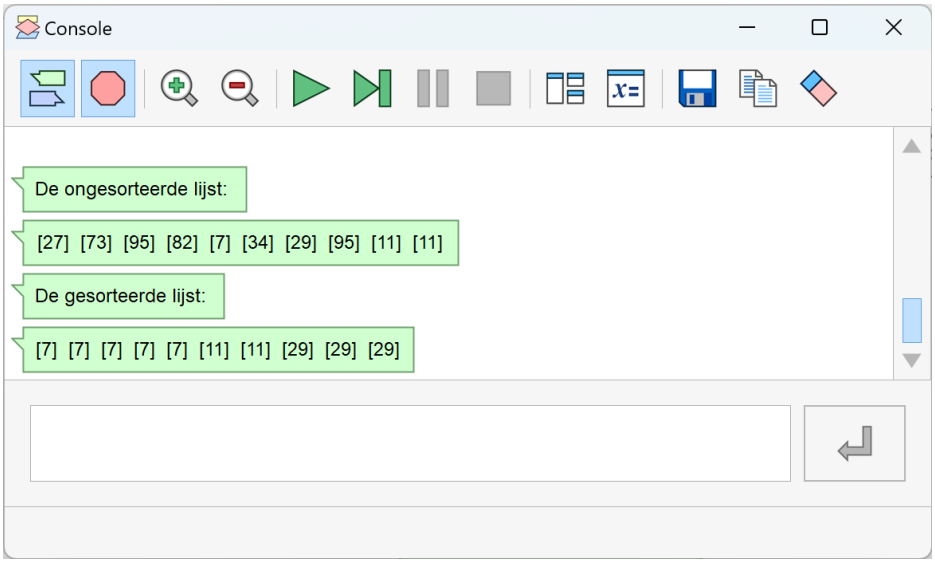


Oefen opdrachten standard algorithmen

Opdracht taken:

1. Maak het **MergeSort** functie in Flowgorithm die een array splits in een linker en rechter helft en de **Merge** functie aanroept om te *sorteren* en *samen te voegen*.
2. Maak een functie (**VulLijst**) die een lijst vult met *willekeurige* getallen tussen 0 en 100.
3. Maak een functie (**ToonLijst**) die de lijst op het scherm toont.

De uitvoer van het programma moet er als volgt uitzien:



Opdracht 3 QuickSort:

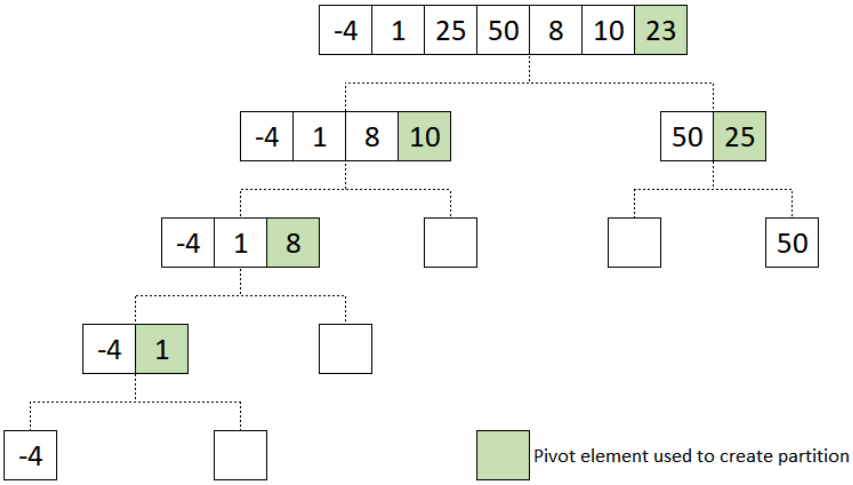
QuickSort (*Snel sorteren*) is een verdeel-en-heers-algoritme. Het is gebaseerd op het idee om één element als draaipunt te kiezen en de array rond het draaipunt te verdelen, waarbij de linkerkant van het draaipunt alle elementen bevat die kleiner zijn dan het draaipunt en de rechterkant van het draaipunt alle elementen bevat die groter zijn dan het draaipunt. Er zijn veel manieren om het draaipunt te kiezen. Enkele daarvan worden hieronder vermeld:

- Eerste element van de array
- Laatste element van de array
- Willekeurig element van de array
- Mediaan indexelement van de array

De snelle sortering heeft minder ruimtecomplexiteit in vergelijking met de **Merge Sort** (samengevoegde) sortering, omdat er geen hulparray wordt gebruikt.

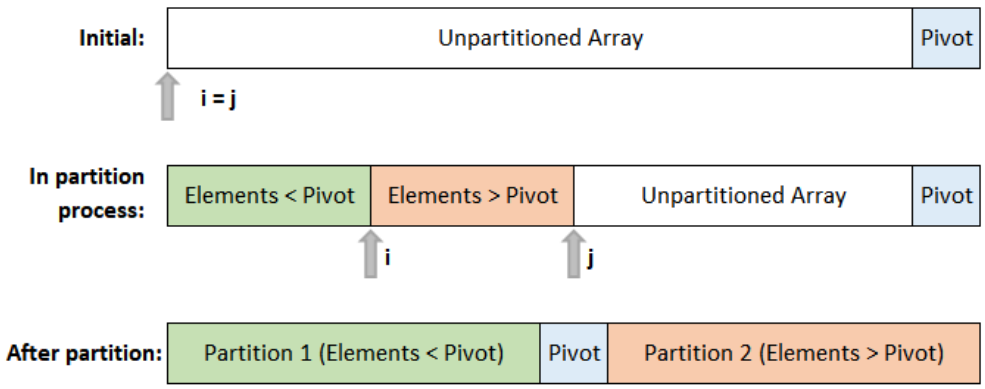
Oefen opdrachten standard algorithmen

Voorbeeld:
Laten we, om de snelle sortering te begrijpen, een ongesorteerde array [-4, 1, 25, 50, 8, 10, 23] bekijken en elke stap bespreken die is genomen om de array in oplopende volgorde te sorteren.



Bij de implementatie van dit voorbeeld wordt het laatste element van de array gekozen als **pivot** (*draai element, draaipunt*). Aan het begin van het partitieproces worden variabelen **i** en **j** aangemaakt, die aanvankelijk hetzelfde zijn. Naarmate de partitie vordert, wordt de waarde van **i** en *j* verschillend.

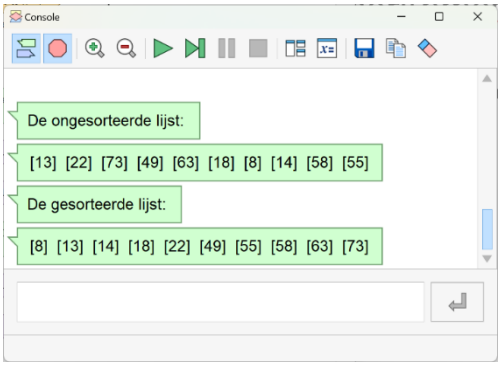
i geeft de grens aan tussen elementen kleiner dan het draaipunt en elementen groter dan het draaipunt.
j geeft de grens aan tussen elementen die groter zijn dan het draaipunt en de niet-gepartitioneerde array.



Na de partitie genereert het twee partities waarbij de linker-partitie elementen bevat die kleiner zijn dan het draaipunt en de rechter-partitie elementen bevat die groter zijn dan het draaipunt. Beide partities worden opnieuw

gepartitioneerd met dezelfde logica en dit proces gaat door totdat een partitie nul of één element bevat. Het uiteindelijke resultaat van dit proces zal een gesorteerde array zijn.

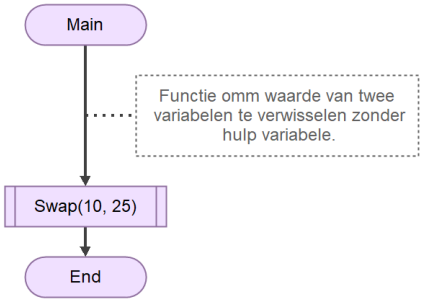
4. Maak het QuickSort programma in Flowgorithm.
5. Maak een functie (**VulLijst**) die een lijst vult met willekeurige getallen tussen 0 en 100.
6. Maak een functie (**ToonLijst**) die de lijst op het scherm toont.
7. Maak twee functies voor het Quicksort algoritme. Eén (**QuickSort**) die het sorteren start en één (**Partition**) die de partitie uitvoert.



De uitvoer van het programma moet er als volgt uitzien:

Opdracht 4: Swaps

Maak een Flowgorithm programma waarin een functie **Swap** gebruikt wordt die **twee gehele getallen als parameters** mee krijgt en de waarden van deze variabelen verwisseld zonder gebruik te maken van een dummy variabele.



Swap met een dummy variabele:

```
// Begin waarde variabelen
getalX = 10
getalY = 25

// Swap
dummy = getalX
getalX = getalY
getalA = dummy

// Eind waarde variabelen
getalX = 25
getalY = 10
```