Abstract Data Types

Datastructuren & Algoritmen

Arjen Wiersma

NOVI Hogeschool

17 februari 2022

Welkom

Wees bewust

Deze bijeenkomst wordt opgenomen.

Overview

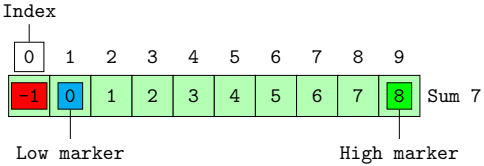
1. Bespreken Opdracht 1
2. Abstract Data Types
3. Veel gebruikte ADT
4. Afronding

Bespreken Opdracht 1

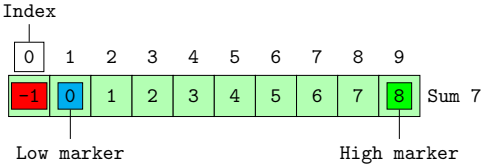
Opsomming van 3 getallen

Wie had een goede oplossing?

Som van 3

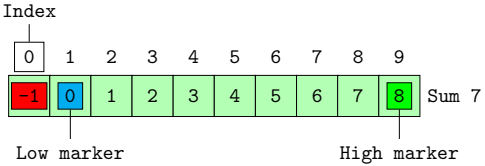


Som van 3



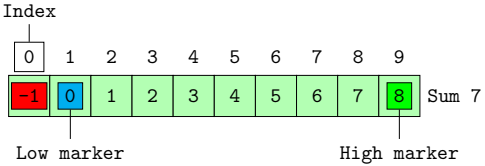
- Start bij index **0**

Som van 3



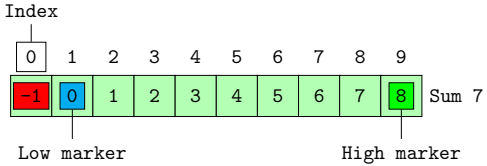
- Start bij index **0**
- Low marker op index **1**, high marker op **laatste** index

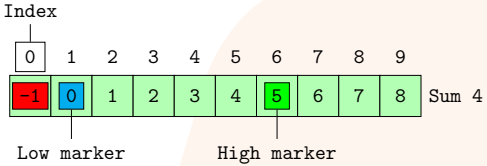
Som van 3



- Start bij index **0**
- Low marker op index **1**, high marker op **laatste** index
- Som te hoog? **high marker** naar beneden, te laag? **low marker** omhoog

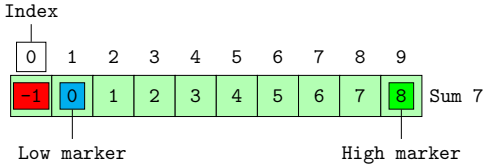
Som van 3

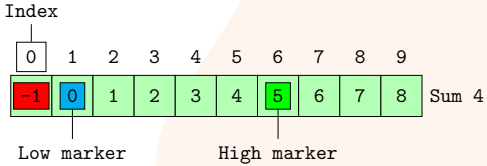




- Start bij index **0**
- Low marker op index **1**, high marker op **laatste** index
- Som te hoog? **high marker** naar beneden, te laag? **low marker** omhoog

Som van 3

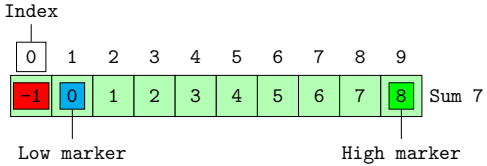


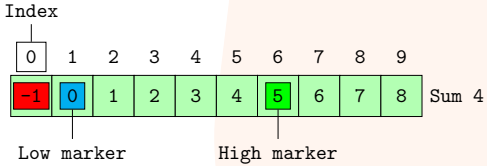


- Start bij index **0**
- Low marker op index **1**, high marker op **laatste** index
- Som te hoog? **high marker** naar beneden, te laag? **low marker** omhoog

- Gaat door totdat de som gevonden is

Som van 3

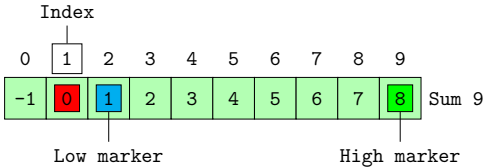




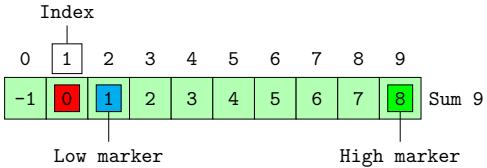
- Start bij index **0**
- Low marker op index **1**, high marker op **laatste** index
- Som te hoog? **high marker** naar beneden, te laag? **low marker** omhoog

- Gaat door totdat de som gevonden is
- Of de **low marker** is hetzelfde als de **high marker**

Som van 3

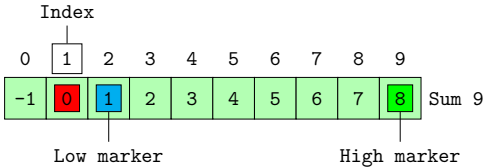


Som van 3



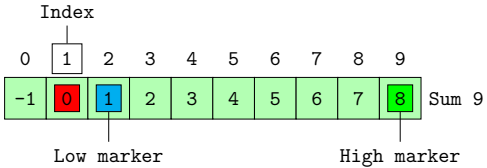
- Het algoritme verhoogt de **index**

Som van 3



- Het algoritme verhoogt de **index**
- Reset de **low** (**index** + 1) en **high** markers

Som van 3



- Het algoritme verhoogt de **index**
- Reset de **low** (**index** + 1) en **high** markers
- Totdat **low** en **high** marker hetzelfde zijn

Wie kan de **sidequest** uitleggen?

Abstract Data Types

Oefening

Verschillende inputs

```
1  (()(()[])[[(())()]])
2  [[[([])[[]])]]]
```

Oefening

- **PARENS** is een programmeertaal die bestaat uit expressies

Verschillende inputs

```
1  (()(()[])[[(())()]])
2  [[[([])[[]])]]]
```

Oefening

- **PARENS** is een programmeertaal die bestaat uit expressies
- Elke expressie bestaat uit een paar **()** of **[]**, en daartussen een instructie

Verschillende inputs

```
1  (()(()[])[[(())()]])
2  [[[(([])[[]])]]]
```

Oefening

- **PARENS** is een programmeertaal die bestaat uit expressies
- Elke expressie bestaat uit een paar **()** of **[]**, en daartussen een instructie
- De instructies zijn niet belangrijk voor nu, ik wil enkel weten of de haakjes gebalanceerd zijn

Verschillende inputs

```
1  (()(()[])[[(())()]])
2  [[[((])[[]])]]
```

Oefening

- **PARENS** is een programmeertaal die bestaat uit expressies
- Elke expressie bestaat uit een paar **()** of **[]**, en daartussen een instructie
- De instructies zijn niet belangrijk voor nu, ik wil enkel weten of de haakjes gebalanceerd zijn
- Een expressie zoals **([)** is incorrect, er mist een **]**.

Verschillende inputs

```
1  (()(()[])[[(())()]])
2  [[[([])[[]])]]]
```

Oefening

- **PARENS** is een programmeertaal die bestaat uit expressies
- Elke expressie bestaat uit een paar **()** of **[]**, en daartussen een instructie
- De instructies zijn niet belangrijk voor nu, ik wil enkel weten of de haakjes gebalanceerd zijn
- Een expressie zoals **([)** is incorrect, er mist een **]**.
- Neem max **10 minuten** om een aanpak te bedenken

Verschillende inputs

```
1  (()(()[])[[(())()]])
2  [[[([])[[]])]]]
```

Oplossingen?

Abstracte Data Typen (ADT)

"A data type whose implementation is hidden from the client"

ADTs in Java

Module java.base
Package java.util

Class AbstractCollection<E>

java.lang.Object
    java.util.AbstractCollection<E>

All Implemented Interfaces:

`Iterable<E>`, `Collection<E>`

Direct Known Subclasses:

`AbstractList`, `AbstractQueue`, `AbstractSet`, `ArrayDeque`, `ConcurrentLinkedDeque`

```
public abstract class AbstractCollection<E>
extends Object
implements Collection<E>
```

This class provides a skeletal implementation of the `Collection` interface, to minimize the

To implement an unmodifiable collection, the programmer needs only to extend this class
and `next`.)

To implement a modifiable collection, the programmer must additionally override this clas

ADTs in Java

- ADT vind je veelvuldig in Java

Module java.base
Package java.util

Class AbstractCollection<E>

java.lang.Object
  java.util.AbstractCollection<E>

All Implemented Interfaces:

Iterable<E>, Collection<E>

Direct Known Subclasses:

AbstractList, AbstractQueue, AbstractSet, ArrayDeque, ConcurrentLinkedDeque

---

```
public abstract class AbstractCollection<E>
extends Object
implements Collection<E>
```

This class provides a skeletal implementation of the Collection interface, to minimize the

To implement an unmodifiable collection, the programmer needs only to extend this class
and next.)

To implement a modifiable collection, the programmer must additionally override this clas

ADTs in Java

- ADT vind je veelvuldig in Java
- Ook wel bekend als `Interface` of `Abstract Classes`

Module java.base
Package java.util

Class AbstractCollection<E>

java.lang.Object
  java.util.AbstractCollection<E>

All Implemented Interfaces:

Iterable<E>, Collection<E>

Direct Known Subclasses:

AbstractList, AbstractQueue, AbstractSet, ArrayDeque, ConcurrentLinkedDeque

---

```
public abstract class AbstractCollection<E>
extends Object
implements Collection<E>
```

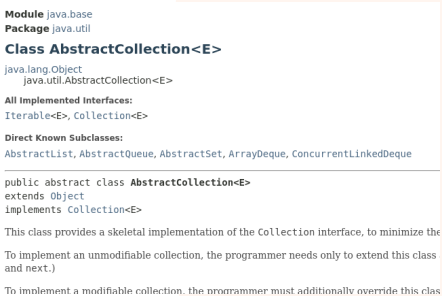
This class provides a skeletal implementation of the Collection interface, to minimize the

To implement an unmodifiable collection, the programmer needs only to extend this class
and next.)

To implement a modifiable collection, the programmer must additionally override this clas

ADTs in Java

- ADT vind je veelvuldig in Java
- Ook wel bekend als `Interface` of `Abstract Classes`
- In andere talen zie je hetzelfde concept van een gedeeld **contract**



Module java.base
Package java.util

Class AbstractCollection<E>

java.lang.Object
  java.util.AbstractCollection<E>

All Implemented Interfaces:

Iterable<E>, Collection<E>

Direct Known Subclasses:

AbstractList, AbstractQueue, AbstractSet, ArrayDeque, ConcurrentLinkedDeque

---

```
public abstract class AbstractCollection<E>
extends Object
implements Collection<E>
```

This class provides a skeletal implementation of the Collection interface, to minimize the

To implement an unmodifiable collection, the programmer needs only to extend this class
and next.)

To implement a modifiable collection, the programmer must additionally override this clas

Implementeren van een ADT

Implementeren van een ADT

- Niet heel anders dan een klasse die een `Interface` implementeert

Implementeren van een ADT

- Niet heel anders dan een klasse die een `Interface` implementeert
- De functies van de `Interface` moeten worden geïmplementeerd

Implementeren van een ADT

- Niet heel anders dan een klasse die een `Interface` implementeert
- De functies van de `Interface` moeten worden geïmplementeerd
- Hoe de data wordt opgeslagen is aan de programmeur

Implementeren van een ADT

- Niet heel anders dan een klasse die een `Interface` implementeert
- De functies van de `Interface` moeten worden geïmplementeerd
- Hoe de data wordt opgeslagen is aan de programmeur
- We koppelen data dus met de *functie implementatie*, maar verbergen de *representatie* van de data.

Waarom een ADT gebruiken

Focus op functies in de API

Encapsulatie

Oefening

Lees de volgende `Interface` omschrijvingen door:

- List
- Map
- Queue
- Set

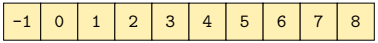
Welk type zou het beste passen bij de *checker* uit de eerste oefening, waarom?

> Ik zal niet de `Interface` functies benoemen bij de bespreking van de typen, deze kun je vinden op de Javadoc pagina waar je nu bent

Veel gebruikte ADT

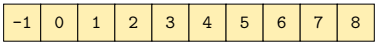
Lists

ArrayList



Lists

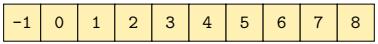
ArrayList



- Een List variatie waarbij de lengte **beperkt** is (`Integer.MAX_VALUE`)

Lists

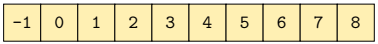
ArrayList



- Een List variatie waarbij de lengte **beperkt** is (`Integer.MAX_VALUE`)
- Interne opslag op basis van Arrays

Lists

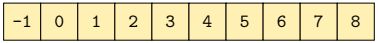
ArrayList



- Een List variatie waarbij de lengte **beperkt** is (`Integer.MAX_VALUE`)
- Interne opslag op basis van Arrays
- Moeilijk om items te verwijderen

Lists

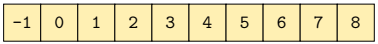
ArrayList



- Een List variatie waarbij de lengte **beperkt** is (`Integer.MAX_VALUE`)
- Interne opslag op basis van Arrays
- Moeilijk om items te verwijderen
- Gemakkelijk om naar een specifieke index te gaan

Lists

ArrayList

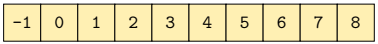


- Een List variatie waarbij de lengte **beperkt** is (`Integer.MAX_VALUE`)
- Interne opslag op basis van Arrays
- Moeilijk om items te verwijderen
- Gemakkelijk om naar een specifieke index te gaan

LinkedList

Lists

ArrayList



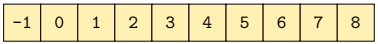
- Een List variatie waarbij de lengte **beperkt** is (`Integer.MAX_VALUE`)
- Interne opslag op basis van Arrays
- Moeilijk om items te verwijderen
- Gemakkelijk om naar een specifieke index te gaan

LinkedList

- Een List variatie waarbij de lengte **erg** groot kan worden (geheugen max)

Lists

ArrayList



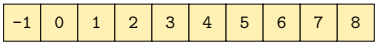
- Een List variatie waarbij de lengte **beperkt** is (`Integer.MAX_VALUE`)
- Interne opslag op basis van Arrays
- Moeilijk om items te verwijderen
- Gemakkelijk om naar een specifieke index te gaan

LinkedList

- Een List variatie waarbij de lengte **erg** groot kan worden (geheugen max)
- Elk element kent de buur elementen

Lists

ArrayList



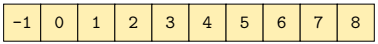
- Een List variatie waarbij de lengte **beperkt** is (`Integer.MAX_VALUE`)
- Interne opslag op basis van Arrays
- Moeilijk om items te verwijderen
- Gemakkelijk om naar een specifieke index te gaan

LinkedList

- Een List variatie waarbij de lengte **erg** groot kan worden (geheugen max)
- Elk element kent de buur elementen
- Gemakkelijk om items te verwijderen

Lists

ArrayList

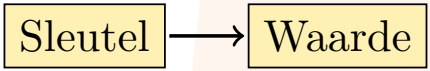


- Een List variatie waarbij de lengte **beperkt** is (`Integer.MAX_VALUE`)
- Interne opslag op basis van Arrays
- Moeilijk om items te verwijderen
- Gemakkelijk om naar een specifieke index te gaan

LinkedList

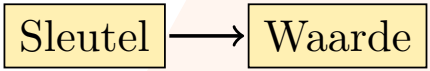
- Een List variatie waarbij de lengte **erg** groot kan worden (geheugen max)
- Elk element kent de buur elementen
- Gemakkelijk om items te verwijderen
- Moeilijk om naar een specifieke index te gaan

Maps



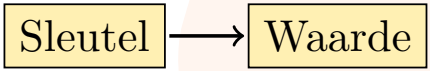
Maps

- `Map` slaan *waarden* op bij *sleutels*



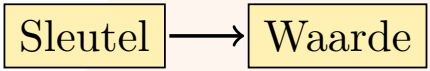
Maps

- `Map` slaan *waarden* op bij *sleutels*
- Variatie in de manier waarop *waarden* zelf worden opgeslagen



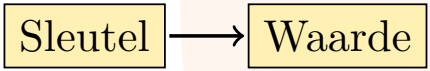
Maps

- `Map` slaan *waarden* op bij *sleutels*
- Variatie in de manier waarop *waarden* zelf worden opgeslagen
- Op basis van de *HASH* van een *waarde* (`HashMap`)



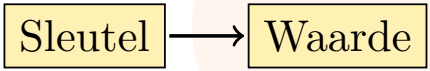
Maps

- `Map` slaan *waarden* op bij *sleutels*
- Variatie in de manier waarop *waarden* zelf worden opgeslagen
- Op basis van de *HASH* van een *waarde* (`HashMap`)
- Op basis van sortering (`Comparator`) van de *waarde* (`TreeMap`)

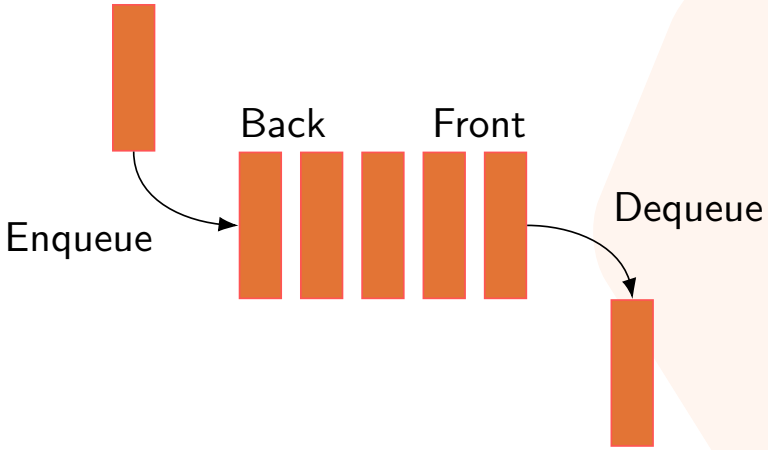


Maps

- `Map` slaan *waarden* op bij *sleutels*
- Variatie in de manier waarop *waarden* zelf worden opgeslagen
- Op basis van de *HASH* van een *waarde* (`HashMap`)
- Op basis van sortering (`Comparator`) van de *waarde* (`TreeMap`)
- Over het algemeen is `HashMap` het type wat je wil hebben



Queueing



Queue variaties

- Queue (FIFO)

Queue variaties

- Queue (FIFO)
- Deque (Double Ended Queue)

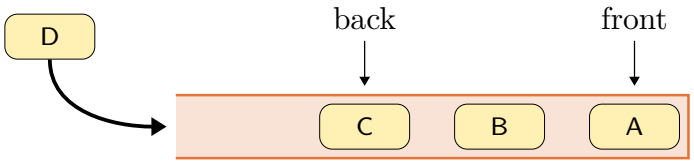
Queue variaties

- Queue (FIFO)
- Deque (Double Ended Queue)
- Stack (LIFO)

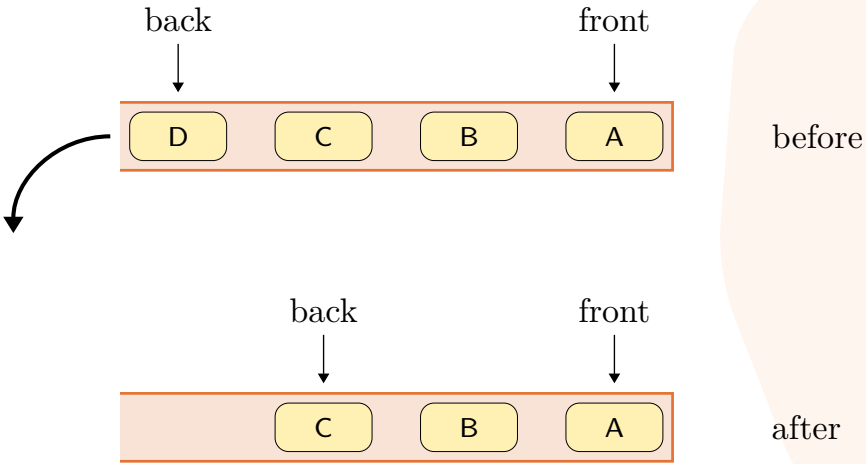
Queue variaties

- Queue (FIFO)
- Deque (Double Ended Queue)
- Stack (LIFO)
- Priority Queue

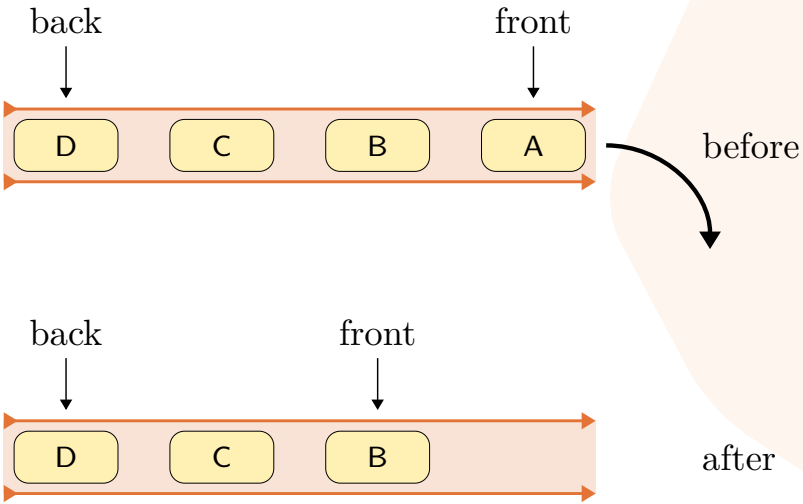
FIFO vs LIFO



LIFO



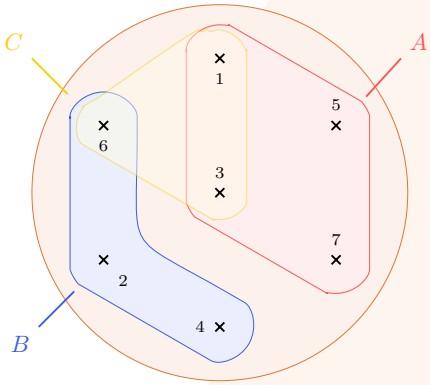
FIFO



Oefening

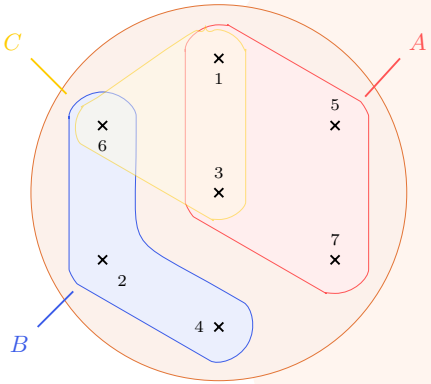
Implementeer de *checker* van eerder met een Queue (Stack) (15 minuten)

Sets



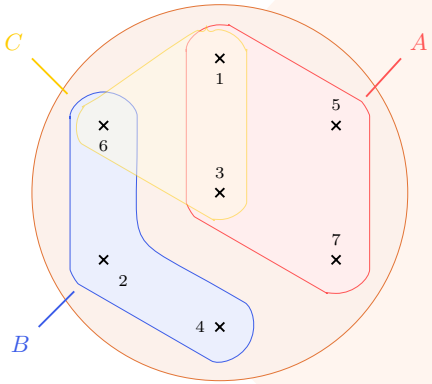
Sets

- bevat een **unieke lijst** aan waarden



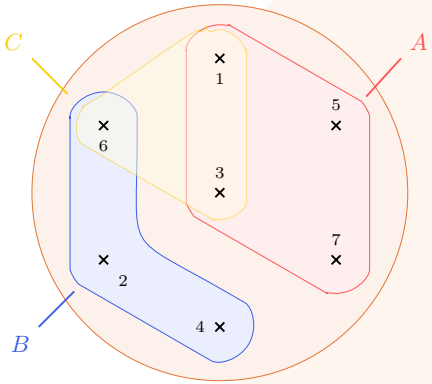
Sets

- bevat een **unieke lijst** aan waarden
- Formeel: contains no pair of elements `e1` and `e2` such that `e1.equals(e2)`



Sets

- bevat een **unieke lijst** aan waarden
- Formeel: contains no pair of elements `e1` and `e2` such that `e1.equals(e2)`
- Als iets dubbel wordt toegevoegd heeft het geen effect



Afronding

Huiswerk

- In Teams zet ik zo een (vrijwillige) huiswerkopdracht (zie `opdracht.pdf`)
- Inleveren kan tot de volgende bijeenkomst, zet je uitwerking in Teams

Volgende bijeenkomst

- **25 Maart**
- Onderwerp: *Algoritme Analyse*