Class DirectedGraph<E extends Edge>

- java.lang.Object
- - DirectedGraph<E>

- Type Parameters:
  `E` - en klass som utökar Edge

```
public class DirectedGraph<E extends Edge>
extends java.lang.Object
```

En klass som kan hitta kortaste vägen mellan två noder samt skapa ett minimalt uppspännande träd.

Author:
Erik Öhrn & Paula Eriksson Imable

- - ## Constructor Summary

Constructors

Constructor and Description
`DirectedGraph(int noOfNodes)` Konstruktor, initialerar en ny graf.

- - ## Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`void`	`addEdge(E e)` Lägger till en ny båge, dels i efterföljarlistan och dels i listan som innehåller alla bågar
`java.util.Iterator<E>`	`minimumSpanningTree()` Hittar ett minsta uppspännande träd hos grafen med hjälp av Kruskals algoritm Prioritetskön använder sig av CompKruskalEdge som jämförare.
`java.util.Iterator<E>`	`shortestPath(int from, int to)` Använder en variant av Dijkstras algoritm (en till en) för att hitta den kortaste vägen mellan två noder.

- - ### Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

- - ## Constructor Detail

- **DirectedGraph**

```
public DirectedGraph(int noOfNodes)
```

Konstruktor, initialerar en ny graf. Grafen använder sig av en efterföljarlisa för att spara bågarna som går från varje nod. Denna använder sig av ett fält av listor, där fältets storlek är antalet noder.

Parameters:
`noOfNodes` - Antalet noder i grafen
Throws:
`java.lang.IndexOutOfBoundsException` - Om antalet noder är mindre än ett

Method Detail

- **addEdge**

```
public void addEdge(E e)
```

Lägger till en ny båge, dels i efterföljarlistan och dels i listan som innehåller alla bågar

Parameters:
`e` - bågen som läggs till
Throws:
`java.lang.NullPointerException` - om e är null

- **shortestPath**

```
public java.util.Iterator<E> shortestPath(int from,
                                          int to)
```

Använder en variant av Dijkstras algoritm (en till en) för att hitta den kortaste vägen mellan två noder. Den sparar de besökta noderna i ett HashSet, mest för att uppslagning av contains() har komplexiteten O(1).

Parameters:
`from` - startnoden
`to` - slutnoden
Returns:
en iterator över vägen från startnoden till slutnoden, eller null om ingen väg hittas
Throws:
`java.lang.IndexOutOfBoundsException` - om from eller to är mindre än noll

- **minimumSpanningTree**

```
public java.util.Iterator<E> minimumSpanningTree()
```

Hittar ett minsta uppspännande träd hos grafen med hjälp av Kruskals algoritm Prioritetskön använder sig av CompKruskalEdge som jämförare. De olika listornas storlekar sparas inte som variabeln då detta efter test tycktes försvåra läsbarheten samtidigt oms .size() ändå på dessa går på O(1) i tidskomplexitet. Kräver en hel sammanbunden graf för att ge ett korrekt svar

Returns:
en iterator över bågarna som utgör trädet

- Detail:
- Field |
- Constr |
- Method