Algoritmen en Datastructuren 2: Minimale Spanningsbossen

Youri Coppens
Youri.Coppens@vub.be

De implementatie van de verscheidene MST-algoritmes vind je terug in `a-d/graph-algorithms/undirected/mst.rkt`.

1 Minimale Spanningsboom (of -bos) als nieuwe graaf

In dit hoofdstuk hebben we drie verschillende algoritmen gezien om op basis van **gewogen, ongerichte grafen** zogenaamde **minimale spanningsbomen of -bossen** te genereren. Dit zijn de algoritmen van **Kruskal**, **Prim-Jarník** en **Borůvka**. In de huidige implementatie hebben deze algoritmen echter geen uniforme uitvoer. Als eerste oefening zullen we de implementaties van deze drie algoritmen aanpassen zodat ze een uniforme uitvoer genereren in de vorm van een nieuwe, gewogen graf die de minimale spanningsboom (of -bos) voorstelt. Met andere woorden: dit is een graaf die alleen de gekozen bogen bevat. Je vindt de drie algoritmen terug in `Vraag1.rkt`.

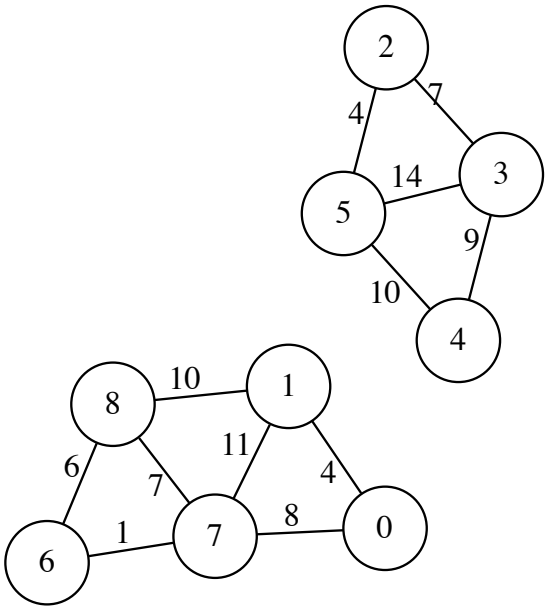
2 Het algoritme van Kruskal

De implementatie van het algoritme van Kruskal geeft alle bogen van een MST, samen met hun gewicht, terug. Om te weten tot **welke** MST van een geconnecteerde component van de graaf ze behoren, moet je echter zelf naar de graaf kijken om dit uit te vissen. We gaan de implementatie van het algoritme uitbreiden zodat deze informatie ook in het resultaat weergegeven wordt.

1. Pas de implementatie aan zodat voor elke boog in het resultaat ook bijgehouden wordt tot welke geconnecteerde component van de graaf de boog behoort. Dit is hetzelfde als bijhouden tot welke minimale spanningsboom in het minimale spanningsbos de boog behoort. **Tip:** een uniek nummertje per MST-component bijhouden is genoeg. Waar kan je al een uniek nummer per MST-component vinden?

2. Welke worst-case performantie heeft je aangepaste versie van Kruskal?

Je kan deze vraag maken in de file `Vraag2.rkt`. Gebruik hiervoor de graaf `cormen571split`. Deze staat in de file `undirected-weighted.rkt`, **meegeleverd met de opgave**. Een visuele voorstelling van deze graaf is gegeven in Figuur 1.



Figuur 1: `cormen571split`

3 Het algoritme van Prim-Jarník

1. Doe hetzelfde als Vraag 2 voor het algoritme van Prim-Jarník. Merk op dat je geen gebruik meer kan maken van hetzelfde trucje als bij het algoritme van Kruskal!

2. Welke worst-case performantie heeft je aangepaste versie van Prim-Jarník?

Je kan deze vraag maken in de file `Vraag3.rkt`.

4 Het algoritme van Borůvka

1. Doe hetzelfde als Vraag 2 voor het algoritme van Borůvka. **Tip:** de aanpassing die nodig is, is gelijkaardig als die voor Kruskal.

2. Welke worst-case performantie heeft je aangepaste versie van Borůvka?

Je kan deze vraag maken in de file `Vraag4.rkt`.