

2

Binaire bomen

Inleiding


Download het bestand BG_les2.zip van Toledo en unzip het. Om de broncode van de oefeningen in IntelliJ te krijgen: map NIET openen, wel code IMPORTEREN (File > New > Project from Existing Sources of bij beginscherm: “import project”).

Nu moet je misschien nog aangeven dat de code in het mapje src de source is. Kies daarvoor in het menu File > project structure en duid bij het blad Modules de src map aan als Sources. De Java bestanden krijgen nu als icoontje een blauwe cirkel met een letter ‘c’ erin.

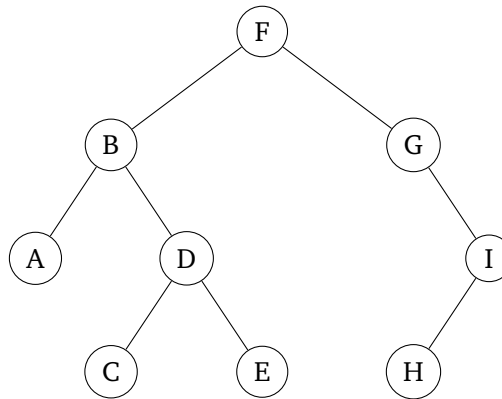
Je herkent de klassieke structuur met domain en ui. Je zal vooral methodes toevoegen in het java bestand binaryTree in domain en dan code uittesten in ui.

We maken eerst twee oefeningen op bomen op papier (). Vanaf oefening 2.3 schrijf je nieuwe methodes ().

Oefening 2.1

 Bekijk de boom uit figuur 2.1 op pagina 6

- Is dit een binaire boom?
- Wat is de wortel van deze boom?
- Wat is de diepte van deze boom?
- Is deze boom compleet?
- Geef alle bladeren van deze boom.
- Geef alle interne knopen van deze boom.
- Hoeveel knopen bevat de linkersubboom? Teken deze subboom.



Figuur 2.1 Een (binaire?) boom

- h) Schrijf de knopen van de gegeven boom (zie fig. 2.1) op in de volgorde waarin ze bezocht worden bij een pre-order wandeling.
- i) Schrijf de knopen van de gegeven boom (zie fig. 2.1) op in de volgorde waarin ze bezocht worden bij een in-order wandeling.
- j) Schrijf de knopen van de gegeven boom (zie fig. 2.1) op in de volgorde waarin ze bezocht worden bij een post-order wandeling.

Oefening 2.2



Voor deze oefening duiken we in de code die je in IntelliJ geïmporteerd hebt.

- a) Bestudeer de implementatie van `BinaryTree<E>`. Stel vragen als er onduidelijkheden zijn.
- b) Bestudeer de `main` functie in de `BinaryTreeDriver` klasse. Hierin wordt een boom aangemaakt waarmee de functie `printPreorder` wordt geïllustreerd. Deze functie is een recursieve implementatie om de waarden van de knopen in pre-order volgorde af te printen.
- c) Teken de boom uit de driver klasse op papier.
- d) Schrijf op papier de verwachte output bij het doorlopen van de boom in pre-order.
- e) Test je verwachte output door de driver-klasse te runnen.
- f) Vervang de code in de driver-klasse door de constructie van de boom uit oefening 2.1.
- g) Run de code en controleer op die manier je antwoord op vraag h) van oefening 2.1.

Oefening 2.3



Van pre-order naar in-order ...

- Implementeer volledig analoog aan `printPreorder` een recursieve implementatie `printInorder` om de waarden van de knopen in in-order volgorde af te printen.
- Controleer je implementatie met behulp van je testvoorbeeld. Ga ook na of de uitvoer overeenkomt met je antwoord op vraag i) van oefening 2.1.

Oefening 2.4



... en naar post-order.

- Implementeer volledig analoog aan `printPreorder` en `printInorder` een recursieve implementatie `printPostorder` om de waarden van de knopen in post-order volgorde af te printen.
- Controleer je implementatie met behulp van je testvoorbeeld. Ga op die manier ook na of de uitvoer overeenkomt met je antwoord op vraag j) van oefening 2.1.

Oefening 2.5



Het doel van deze opdracht bestaat erin om een recursieve implementatie te schrijven voor het bepalen van het aantal knopen van een boom.

- Ga voor de boom uit oefening 2.1 na dat het aantal knopen kan gevonden worden als volgt: 1 + als er een linkersubboom is: het aantal knopen van de linkersubboom + als er een rechtersubboom is: het aantal knopen van de rechterSubboom.
- Gebruik het vorige idee om een recursieve implementatie `countNodes` te schrijven om het aantal knopen van een binaire boom te bepalen.
- Controleer je implementatie van `countNodes` met behulp van je testvoorbeeld in de `main` functie.

Oefening 2.6



Schrijf een recursieve implementatie voor het bepalen van de diepte van een gegeven binaire boom.

- Ga voor de boom uit oefening 1 na dat zijn diepte kan gevonden worden als 1 + het maximum van de diepte van de linker- en rechtersubboom van de boom.

2 Binaire bomen

- b) Gebruik het vorige idee om een recursieve implementatie `getDepth` te schrijven om de diepte van een binaire boom te bepalen.
- c) Controleer je implementatie van `getDepth` met behulp van je testvoorbeeld in de `main` functie.

Oefening 2.7



Schrijf een functie `isLeaf` om na te gaan of een boom een blad is. Dit wil zeggen dat de linkerdeelboom en de rechterdeelboom leeg zijn.

Oefening 2.8



Het doel van deze opdracht bestaat erin om een recursieve implementatie te schrijven voor het bepalen van het aantal bladeren van een boom.

- a) Ga voor de boom uit oefening 1 na dat het aantal bladeren kan gevonden worden als volgt: 1 voor een boom die enkel 1 blad bevat en anders de som van het aantal bladeren van de linkersubboom als er een linkersubboom is en het aantal bladeren van de rechtersubboom als er een rechtersubboom is.
- b) Gebruik het vorige idee om een recursieve implementatie `countLeaves` te schrijven om het aantal bladeren van een binaire boom te bepalen.
- c) Controleer je implementatie van `countLeaves` met behulp van je testvoorbeeld in de `main` functie.

Oefening 2.9



In deze oefening schrijf je een recursieve implementatie `getDataLeaves` voor het bepalen van een lijst van datavelden van de bladeren van een boom. Controleer je implementatie met behulp van je testvoorbeeld en vergelijk de uitvoer met je antwoord op vraag e) uit oefening 2.1. De volgorde zal natuurlijk afhangen van het soort wandeling dat je gebruikt.

Oefening 2.10



Programmeer een recursieve implementatie voor het bepalen of een gegeven data-veld in de binaire boom voorkomt. Schrijf een methode `contains` die gegeven een data-veld `true` teruggeeft indien de boom een knoop bevat met het gegeven data-veld en `false` anders. Probeer

in je `main` deze functionaliteit uit door vier keer de functie op te roepen met respectievelijk “D”, “H”, “F” en “Q” als parameter.