Academiejaar: 2019–2020

Opleiding: Bachelor in de Toegepaste informatica

Fase: 1

Examinator: G. Jongen, T. Eversdijk, J. Van Hee

OPO: MBI72 – Bomen & Grafen

OLA: MBI72a – Bomen & Grafen

Hulpmiddelen: Eigen laptop, beperkt wifi netwerk xtoledo

Datum: 11 juni 2020

Beginuur: 13:00

Tijdsduur: 2 uur + evt. 30 % extra tijd

Student:...................................................... Nr: ....................

Deze opgaven vormen het tweede deel van het examen. Dit deel bevat twee vragen. Je krijgt het pas na het inleveren van het schriftelijk deel. Beide opgaves programmeer je op je eigen laptop. Je mag hierbij alles gebruiken wat op je laptop staat.

Je mag enkel verbinding maken met het Wifi-netwerk “xtoledo” of “xtoledo5G”. Het wacht- woord is “examen001”.

Dit is het enige toegelaten netwerk. Er is enkel toegang tot x.toledo.ucll.be. We gebruiken de xtoledo omgeving enkel om je wat code te bezorgen en de code van je oplossing af te geven.

Doorloop nu volgende zeven stappen:

1. Voor dit examen maak je een nieuw java-project aan in IntelliJ. De naam van je project is “FamilienaamVoornaam”.
2. Je maakt in je src map een package domain en een package ui aan. Elke examenvraag (van de twee) geeft een nieuwe klasse in domain en een bijhorende driverklasse in ui. Je project zal dus bestaan uit twee javabestanden in domain en twee javabestanden in ui.
3. Als je klaar bent, exporteer je je project in IntelliJ als een zip bestand. Geef de zip dezelfde naam als je project (dus “FamilienaamVoornaam.zip”). Als je een lege zip afgeeft, kan je ook geen punten krijgen. Controleer m.a.w. of je zip de juiste bestanden bevat (door het bestand eens te downloaden, te ontzippen en de inhoud te bekijken).
4. Log in op x.toledo.ucll.be.

Je vindt er het examenvak “2020\_2\_11/06/2020\_nm\_MBI72a\_Bomen en grafen”.

1. Je vindt op xToledo het bestand templates.zip. Deze zip bevat broncode die je van ons krijgt.
2. Dien je examen officieel in via de knop “verlaat examen”, links onder het cursusmenu.
3. Laat het zip-bestand onaangeroerd op je laptop staan tot na de examens zodat je een bewijs hebt als er iets technisch fout zou lopen.

Veel succes!

# Vraag 1. Binaire zoekbomen (BST) (6 punten)

De werking van een BST is gekend, in deze opgave gaan we een stap verder en gaan we een Ternary Search Tree (TST) implementeren.

# Een voorbeeld

Onderstaande figuur is een voorbeeld van een TST en wordt bekomen door de volgende knopen in de respectievelijke volgorde toe te voegen: 2 - 5 - 1 - 8 - 3 - 10 - 4 - 9.

Zoals duidelijk in de figuur wordt een niet bestaand item voorgesteld door NULL.



# AddNode

Bij een ternary search tree kan iedere knoop 3 kinderen hebben en bevat iedere knoop 1 of 2 items. Afhankelijk van het aantal items gebeurt het volgende tijdens het toevoegen van een nieuw item in de zoekboom.

* + Boom is leeg: Maak een wortel voor de boom met daarin het nieuwe item.
	+ Wortel bevat 1 item: Het nieuwe item zal in deze knoop worden toegevoegd.
	+ Wortel bevat 2 items:
		- Het nieuwe item is kleiner dan de 2 bestaande waardes in de knoop. De nieuwe knoop wordt aan de linkerdeelboom toegevoegd.

→

* + - Het nieuwe item is groter dan de 2 bestaande waardes in de knoop. De nieuwe knoop wordt aan de rechterdeelboom toegevoegd.

→

* + - Het nieuwe item ligt tussen de 2 bestaande waardes in de knoop. De nieuwe knoop wordt in de middendeelboom toegevoegd.

→

# Opgave

Pas de methode addNode() van lesweek 3 (BST) aan zodat deze werkt voor een Ternary Search tree.

Je krijgt van ons de domainklasse TernarySearchTree en bijhorende driver TernarySearchTreeDriver (zie templates.zip op xToledo).

# Vraag 2. Grafen (6 punten)

* + Kopieer de klasse Graph van week 7 (directory week07\_grafen\_BFS) naar je IntelliJ project.
	+ Gebruik het BFS algoritme om dit probleem op te lossen (methodes
	+ findAncestors(int start, int destination) en
	+ findPath(int start, int destination)).

Een stad wordt voorgesteld als gerichte graaf. Elk kruispunt is een knoop. De straten zijn de verbindingen tussen de knopen. Sommige straten zijn eenrichtingsverkeer.

An bevindt zich op kruispunt A; Bernard op kruispunt B. Ze lopen naar mekaar toe. Daarbij tellen ze hoeveel verbindingen tussen kruispunten ze afleggen. De som van het aantal verbindingen dat ze afgelegd hebben moet zo klein mogelijk zijn. Bovendien mogen ze mekaar niet ontmoeten op een van de startkruispunten m.a.w. ze moeten beiden minstens één verbinding gevolgd hebben.

# Opgave

Schrijf de methode ontmoeting(startplaatsAn:int, startplaatsBernard: int): int dat het knooppunt teruggeeft waar An en Bernard mekaar zullen ontmoeten.

* + Tip: bereken voor elke knoop het aantal stappen dat An en Bernard moeten zetten om die knoop te bereiken.

Op de volgende pagina vind je een voorbeeld.



**Figuur 1** Netwerk voor ontmoeting van A en B

In het netwerk van figuur 1 moeten An en Bernard samen 4 verbindingen afleggen om mekaar te ontmoeten. De ontmoetingsplaats zelf hangt af van je implementatie (dus er zijn meerdere correcte antwoorden, bijv. 3, 7, 6).

De verbindingsmatrix van dit netwerk is gegeven in het bestand voorbeeldGrafen.txt in het gegeven zip-bestand:

int data[][]= {

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| // A | 2 | 3 | 4 | 5 | 6 | 7 | B |  |
| {0, | 1, | 0, | 1, | 0, | 0, | 0, | 0}, | //A |
| {1, | 0, | 1, | 0, | 0, | 0, | 0, | 0}, | //2 |
| {0, | 0, | 0, | 1, | 0, | 0, | 1, | 1}, | //3 |
| {1, | 0, | 1, | 0, | 1, | 1, | 0, | 0}, | //4 |
| {0, | 0, | 0, | 0, | 0, | 0, | 1, | 1}, | //5 |
| {1, | 0, | 0, | 1, | 0, | 0, | 0, | 0}, | //6 |
| {0, | 0, | 1, | 0, | 0, | 1, | 0, | 0}, | //7 |

{0, 0, 0, 0, 0, 0, 1, 0}};//B