Programmering 2: övningsuppgift 4

Version: 1.1
Uppdaterad: 2021-04-21, se dokumenthistoriken sist i dokumentet för info.
Författare: Patrick Wentzel

Detta är den fjärde övningsuppgiften på kursen PROG2 VT2021.

Uppgiften är giltig under kursomgången fram till uppsamlingstillfället i augusti 2021 och upphör därefter att gälla och kan komma att ersättas kursomgången 2022.

Innehållsförteckning

Inledning

För att öka försäljningen och förbättra logistiken har butiken investerat i grafteknologi och nu är det dags att den kommer till användning.

Den första uppgiften, som måste fixas, är att skapa en graf med data om vilka städer som butiken levererar skivor till och hur man kan resa dit. Som tur är finns informationen i en fil som borde vara lätt att läsa in.

Om du får tid över och har lust så skulle butiken också vilja pröva grafteknologin för att se om det går att förbättra rekommendationer om inköp som vi ger till folk som kommer in undrar vad de ska köpa.

Vi har en fil som innehåller namn på kunder och vad de har köpt och tänkte att det borde gå att använda för att bygga ett rekommendationssystem som kan svara på frågor som:

• Vilka är de mest populära skivorna?

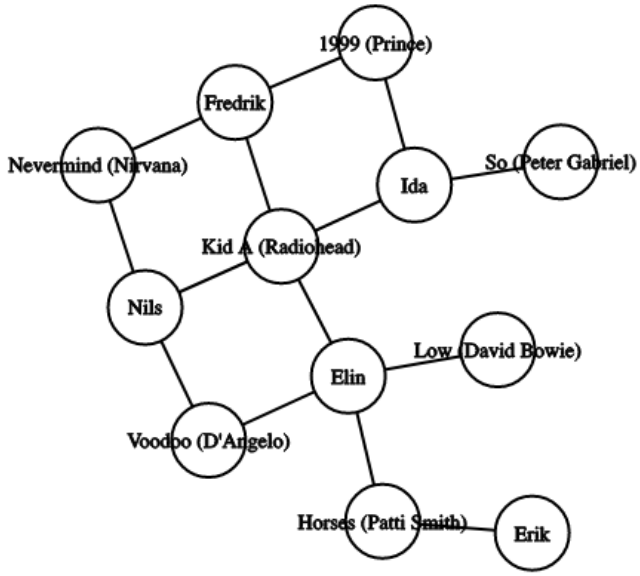
• Hur populär är en skiva?

• Vilka andra skivor är populära hos de som har köpt en viss skiva?

Tanken är att man borde kunna bygga en graf där både personer och skivor är noder och själva ägandeskapet utgör bågen mellan de och sedan kunna använda grafklassens olika metoder för att svara på frågorna.

I exempelgrafen nedanför finns det 5 personer som äger 7 olika skivor. Den populäraste skivan är Kid A (Radiohead) som ägs av 4 personer, och de minst populära är So (Peter Gabriel) och Low (David Bowie) som bara ägs av 1 person var.

Vi kan också se att folk som gillar Voodoo (D'Angelo) också gillar Kid A (Radiohead), Low (David Bowie) och Horses (Patti Smith).



Uppgiften

Uppgiften har en obligatorisk del och en frivillig del med extra uppgifter för den som vill utmana sig själv och se hur en enkel rekommendationsgraf[1] kan konstrueras.

Den obligatoriska uppgiften är att skriva en metod som läser in en datafil och bygger upp en graf (av typen ListGraph) utifrån innehållet i filen.

Den frivilliga delen är att läsa in en annan fil till en annan graf och med hjälp av grafens metoder lösa några uppgifter.

Uppgiften använder klassen ListGraph från inlämningsuppgiftens första del så den behöver vara klar och fungera innan någon del av del här uppgiften påbörjas.

Inlämningsdatum (deadline)

Rekommenderat inlämningsdatum är den 12/5. Uppgiften behöver vara avklarad innan slutuppgiften med deadline 5/6 kan lämnas in.

Nyckelord

Det finns inga delar specifika för uppgiften.

[1] Grafer av liknande slag, fast mycket mer komplexa, är det som t.ex. Facebook är byggt på och också kärnan i de flesta system för rekommendationer av närliggande platser/affärer/ restauranter/badplatser etc.

Klasserna

Exercise4

Klassen `Exercise4` är den klass som ska implementera gränssnittet Ex4 och där de olika uppgifterna ska lösas. Klassen har en instans av ListGraph specificerad för typen GraphNode (se sektionen Gränssnitten) som en medlemsvariabel som tilldelas i konstruktorn.

Klassen har inga andra metoder utöver konstruktorn och de som krävs från gränssnittet.

Edge

Klassen `Edge` representerar en båge i grafen och är en något modifierad variant av klassen från inlämningsuppgiften. Klassen ska inte förändras.

ListGraph

Klassen `ListGraph` representerar en graf och ska vara samma som i inlämningsuppgiften. Klassen tillhandahålls inte, men finns i VPL vid testning, och ska inte lämnas in. Klassen ska inte förändras.

Nodes

Klassen `Nodes` innehåller tre inre klasser som representerar olika typer av noder som används i de olika uppgifterna. Samtliga nedklassar implementerar gränssnittet GraphNode. Klassen ska inte förändras.

`LocationNode` används i den obligatoriska delen av uppgiften, övriga noder används i den frivilliga delen.

LocationNode

Klassen `LocationNode` representerar en plats. Klassen har attributen namn, x och y som alla sätts i konstruktorn. Klassen ska inte förändras.

PersonNode

Klassen `PersonNode` representerar en skiva. Klassen har attributet namn som sätts i konstruktorn. Klassen ska inte förändras.

RecordNode

Klassen `RecordNode` representerar en skiva. Titel och artist är ihopslaget till ett attribut ‘title’ som tilldelas i konstruktorn. Klassen ska inte förändras.

Gränssnitten

Till uppgiften hör tre olika gränssnitt. Inget av de ska förändras, och inget ska heller lämnas in.

Graph

Gränssnittet `Graph` är samma som i inlämningsuppgiften och innehåller de operationer som klassen `ListGraph` ska implementera.

GraphNode

Gränssnittet `GraphNode` som finns i samma fil som klassen `Nodes` är tomt och används enbart för att låta icke-relaterade klasser få samma typ[2].

Ex4

Gränssnittet `Ex4` innehåller de metoder som ska lösas i klassen Exercise4 enligt nedan:

- `loadLocationGraph` – den enda metod som **måste implementeras** tar emot ett filnamn och bygger upp en graf med hjälp av filens innehåll. Filformatet är dokumenterat i nästa sektion.

- `optionalLoadRecoGraph` – en frivillig metod som ska läsa in en fil och bygga upp en graf med hjälp av filens innehåll.
  Metoden har en default-implementation och behöver inte implementeras, men är nödvändig för de andra frivilliga metoderna.

- `optionalGetAlsoLiked` – en frivillig metod som tar emot en nod av typen `RecordNode` och returnerar en avbildning med andra skivor som personer som äger den aktuella skivan också äger. Avbildningen har popularitet som nyckel (sorterat fallande) och en sorterad mängd `RecordNode` som värde.

  Metoden är tänkt att svara på frågan "*vad gillar andra som gillar den här skivan*".

- `optionalGetPopularity` – en frivillig metod som tar emot en nod av typen `RecordNode` och returnerar ett värde som indikerar hur populär en skiva är, vilket är samma som antalet bågar/kanter ifrån noden[3].

  Metoden är tänkt att svara på frågan "*hur poppis är den här skivan*".

- `optionalGetTop5` – en frivillig metod som returnerar en avbildning med 5 (eller färre om det inte finns 5) samlingar av skivor ordnade efter popularitet i fallande ordning. Nyckel är popularitet och värde är en samling av `RecordNode`.

  Metoden är tänkt att svara på frågan "*vilka är de populäraste skivorna*".

---

[2] Det brukar kallas ett Marker Interface eller Tagging Interface

[3] Nodens "outdegree" i grafterminologi.

Exempel på resultat (från filen med testdata):

```
{
3=[The Miseducation of Lauryn Hill (Lauryn Hill)],

2=[Touch (Eurythmics), Nevermind (Nirvana), Licensed to Ill (Beastie Boys),
1999 (Prince), Kid A (Radiohead), Voodoo (D'Angelo)],

1=[Born to Run (Bruce Springsteen), Horses (Patti Smith), Stories From the
City, Stories From the Sea (PJ Harvey), Rust Never Sleeps (Neil Young & Crazy
Horse), Talking Book (Stevie Wonder), A Love Supreme (John Coltrane), So
(Peter Gabriel), Automatic for the People (R.E.M.), MTV Unplugged in New York
(Nirvana), Violator (Depeche Mode), The Velvet Underground & Nico (The Velvet
Underground), Low (David Bowie), Talking Heads: 77 (Talking Heads)]
}
```

Filerna

Fil för inläsning av graf för den obligatoriska delen

Filen `ex4location.graph` innehåller data om en graf med olika platser tänkta att representeras av klassen `LocationNode`.

Formatet på filen är som följer:

en inledande lång rad med semikolonseparerade uppgifter om noder, och därefter flera rader med uppgifter om förbindelserna, en förbindelse per rad.

Uppgifter om noder ska omfatta nodens namn, nodens x-koordinat och nodens y-koordinat, separerade med semikolon.

Raderna med uppgifter om förbindelserna ska för varje förbindelse innehålla namnet på från-noden, namnet på till-noden, namnet på förbindelsen och förbindelsens vikt, separerade med semikolon.

Ett exempel på en sådan fil ges här (obs att det inte finns några mellanslag efter semikolon):

```
Stockholm;469.0;242.0;Oslo;398.0;219.0;Warszawa;503.0;377.0
Stockholm;Oslo;Train;3
Stockholm;Warszawa;Airplane;2
Oslo;Stockholm;Train;3
Warszawa;Stockholm;Airplane;2
```

Fil för inläsning av graf för den frivilliga delen

Filen `ex4recosmall.graph` innehåller data om vilka skivor olika personer äger och är tänkt att användas i en graf där PersonNode representerar personen och RecordNode representerar skivan.

Varje rad innehåller två noder (person och skiva) separerade med semikolon och utgör dessutom bågen mellan noderna (som dock varken har namn eller vikt).

```
Ida;1999 (Prince)
Ida;So (Peter Gabriel)
Elin;Violator (Depeche Mode)
Elin;Voodoo (D'Angelo)
Elin;The Miseducation of Lauryn Hill (Lauryn Hill)
Elin;Low (David Bowie)
Fredrik;Licensed to Ill (Beastie Boys)
Fredrik;1999 (Prince)
```

Instruktion för inlämning

För att kunna lämna in behöver man ha valt grupp för inlämningen.

I VPL för övningsuppgiften laddas klassen Exercise4 upp. Inga andra filer ska lämnas in. De övriga filer som behövs (gränssnitten och klasser Edge, listGraph och Nodes) finns redan i VPL.

Koden testas automatiskt med hjälp och kommer att bli godkänd med automatik när alla tester uppfylls.

Överst i samtliga inlämnade filer ska det finnas en kommentar med namn och användarnamn för samtliga gruppmedlemmar. Exempelvis:

```
// PROG2 VT2021, Inlämningsuppgift, del 1
// Grupp 004
// Dewey Duck dedu1111
// Huey Duck hudu1234
```

Dokumenthistorik

Version	Datum	Ändringar
1.0	2021-04-13	Första version.
1.1	2021-04-21	Förtydligade optionalGetTop5.