Programmering 2: övningsuppgift 4

Version: 2.2
Uppdaterad: 2024-03-25, se dokumenthistoriken sist i dokumentet för info.
Författare: Patrick Wentzel, modifierad av Jozef Swiatycki

Detta är den fjärde övningsuppgiften på kursen PROG2 VT2024.

Ett förslag till lösning kommer att presenteras via Zoom fredagen 2024-05-03,
som alltså utgör rekommenderad deadline. Obs att lösningsförslagets kod inte kommer att publiceras.

Uppgiften kan testas med VPL i iLearn eller med det tillhandahållna JUnit-testet.

Uppgiften förutsätter att del 1 av inlämningsuppgiften är klar (den används i denna uppgift). Obs att funktionaliteten i `ListGraph<>`-klassen (som del 1 av inlämningsuppgiften går ut på) är deklarerad i gränssnittet `Graph<>`, som finns i iLearn under Inlämningsuppgiften/Filer för inlämningsuppgiften, så de metoder för graf-hantering som ska användas i lösning av denna övning är de som finns i `Graph<>`-gränssnittet, och man kan koda mot detta gränssnitt även om implementering av `ListGraph<>`-klassen inte är helt klar.

Innehållsförteckning

Inledning

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

Den första uppgiften, som måste fixas är ett program som skapar 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. Programmet ska alltså läsa in filen och skapa grafen ifrån den.

Dessutom vill butiken pröva graftekniken för att se om det går att förbättra rekommendationer om inköp som ges till folk som kommer in och undrar vad de ska köpa.

Det finns en fil som innehåller namn på kunder och vad de har köpt och vi tänker att detta borde gå att använda för att bygga ett rekommendationssystem som kan svara på frågor som:

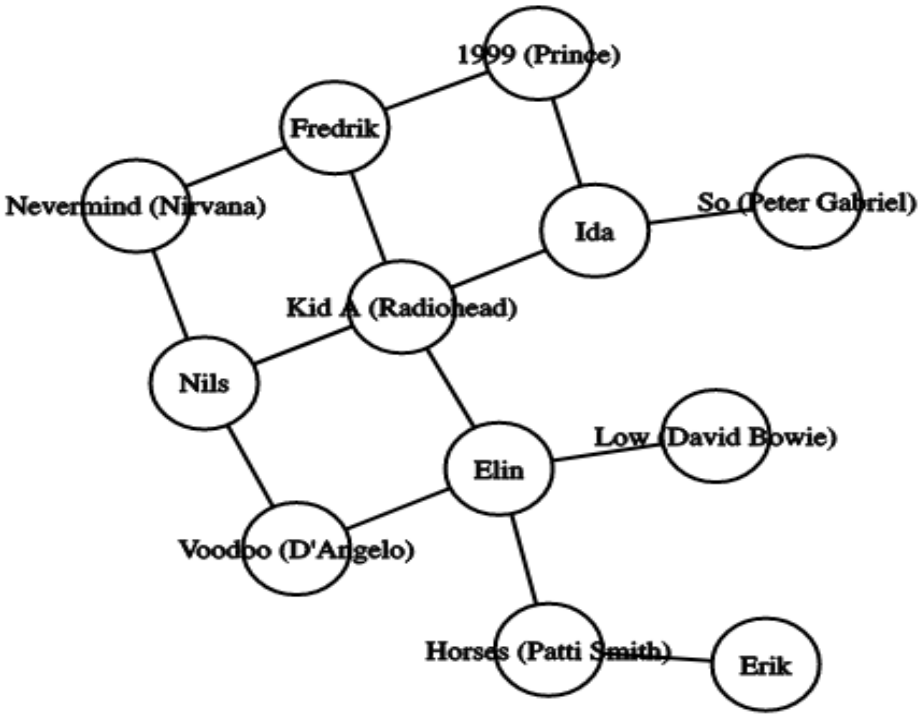
- Vilka är de mest populära skivorna?

- 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 ägandet utgör kanten mellan dem, och sedan kunna använda grafklassens olika metoder för att svara på frågorna.

I exempelgrafen nedan finns det fem personer som äger sju olika skivor.
Den mest populära skivan är Kid A (Radiohead) som ägs av fyra personer, och de minst populära är So (Peter Gabriel) och Low (David Bowie) som bara ägs av en 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 alltså två delar: en del där man läser in data om en graf och bygger upp grafen (och som liknar en operation som kommer att behöva göras i del 2 av inlämningsuppgiften) och en del där man kan se hur en enkel rekommendationsgraf[1] kan konstrueras.

Den första delen går ut på att skriva en metod som läser in en textil med data och bygger upp en graf (av typen `ListGraph`) utifrån innehållet i filen.

Den andra delen är att läsa in en annan textfil 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.

Klasser och gränssnitt

Exercise4

Klassen `Exercise4` är den klass där metoder som utgör denna uppgift ska implementeras. Metoden `loadLocationGraph()` utgör första delen av uppgiften och är inte implementerad alls.

- `loadLocationGraph` – den enda metod som inte alls är implementerad tar emot ett filnamn och bygger upp en graf med hjälp av filens innehåll. Filformatet är dokumenterat i nästa sektion. Metoden bör skriva ut grafen efter inläsning, med användning av `ListGraph`s `toString`-metode

De övriga metoderna hör till den andra delen av uppgiften (rekommendationssystemet) och är implementerade i så liten grad som möjligt för att man ska kunna testa programmet utan att implementera dem:

- `loadRecommendationGraph` – en metod som ska läsa in en textfil och bygga upp en graf med hjälp av filens innehåll (se nedan för filformatet).

- `getAlsoLiked` – en metod som tar emot en nod av typen `Record` och returnerar en avbildning (`SortedMap`) med andra skivor som personer som äger den skivan också äger. Avbildningen har popularitet som nyckel (sorterat fallande) och en sorterad mängd av `Record` som värde.
  Metoden är tänkt att svara på frågan "*vad gillar andra som gillar den här skivan*".

- `getPopularity` – en metod som tar emot en nod av typen `Record` och returnerar ett heltal som anger hur populär en skiva är, vilket är samma som antalet kanter ifrån noden[2].
  Metoden är tänkt att svara på frågan "*hur poppis är den här skivan*".

- `getTop5` – en metod som returnerar en avbildning (`SortedMap`) med fem (eller färre om det inte finns fem) samlingar av skivor ordnade efter popularitet i fallande ordning. Nyckel är popularitet och värde är en samling av `Record`.
  Metoden är tänkt att svara på frågan "*vilka är de mest populära skivorna*".

[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.
[2] Nodens förgreningsfaktor, utgrad (*outdegree*) i grafterminologi.

Node

Gränssnittet `Node` implementeras av alla nedanstående nodklasserna och är egentligen bara till för att klasserna `Person` och `Record` ska ha en gemensam supertyp, så objekt av båda kan ingå i samma graf. Eftersom alla tre nedanstående nodklasserna har ett `String`-attribut `name` så deklareras även metoden `getName()` i `Node`-gränssnittet. Gränssnittet ska inte förändras.

Location

Klassen `Location` representerar en plats och används som nodklass i första delen av uppgiften, den implementerar gränssnittet `Node`. Klassen har attributen `name` (`String`) och koordinater x och y (`double`) som alla sätts i konstruktorn, samt metoden `getName()`. Klassen ska inte förändras.

Person

Klassen `Person` representerar en person och implementerar gränssnittet `Node`. Klassen har attributet `name` som sätts i konstruktorn och metoden `getName()`. Klassen ska inte förändras.

Record

Klassen `Record` representerar en skiva och implementerar gränssnittet Node. Den har attributen `name` (titel) och `artist` som tilldelas i konstruktorn samt get-metoder för dem. Klassen ska inte förändras.

Filerna

Fil för inläsning av Location-grafen

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

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 omfattar nodens namn, nodens x-koordinat och nodens y- koordinat, separerade med semikolon. Koordinaterna är angivna med decimaler.

Raderna med uppgifter om förbindelserna innehåller för varje förbindelse namnet på från-noden, namnet på till-noden, namnet på förbindelsen och förbindelsens vikt (ett heltal), separerade med semikolon.

Exempel på en sådan fil (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 grafen för rekommendationer

Filen `ex4reco.graph` innehåller data om vilka skivor olika personer äger och är tänkt att användas i en graf där Person representerar personen och Record representerar skivan.

Varje rad innehåller namnet på en person, titeln på en skiva och namnet på skivans artist, separerade med semikolon. De utgör dessutom förbindelsen mellan personen och skivan (förbindelsen har dock varken 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
```

Dokumenthistorik

Version	Datum	Ändringar
1.0	2022-04-13	Första version.
2.0	2023-04-14	Större förändring.
2.1	2024-03-17	Ändrat regler och datum
2.2	2024-03-25	Ändrat datum