INDU: Tåg

Kristoffer Sahlin

9 december 2021

Innehåll

Introduktion

I det här projektet ska du skriva ett Pythonprogram för att simulera tåg som kör fram och tillbaka på olika linjer (tänk tunnelbanan). Vi kommer tilla på en enkel "turn-based" (diskret) modell där angränsande stationer befinner sig en tidsenhet ifrån varandra. I varje "steg" (tidsenhet) i modellen befinner sig ett tåg vid en station. Från ett steg till ett annat kan två utfall ske för ett tåg: antingen har tåget nått nästa station, eller så har det blivit stående på samma station (försening). Kartan över tågstationer kan visualiseras som en graf där varje nod är en station och vaje kant beskriver att två stationer är anknutna med tågräls (exempel i fig. 1).

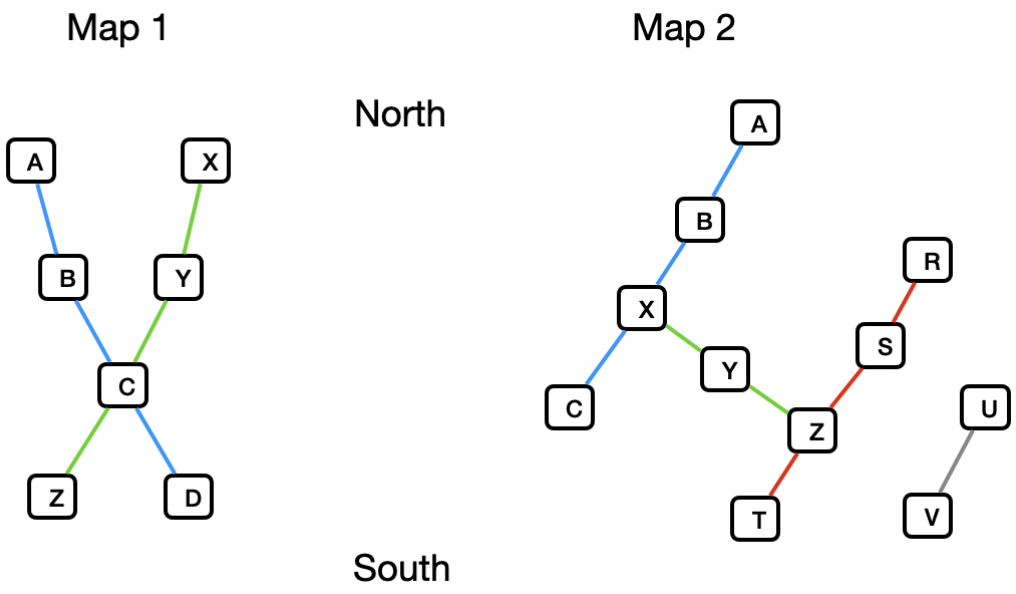
Modellen

Vi beskriver nu detaljerna av modellen. Varje tåg är bunden till en linje. Det finns endast räls mellan stationer på samma linje. När ett tåg når antingen den nordliga eller sydliga slutstation på linjen vänder de och åker tillbaka.

Vid start av programmet ska vi slumpa fram n tåg, där varje tåg slumpas till en station samt med en riktning (nordlig eller sydlig riktning). Om ett tåg slumpas till en slutstation, tex den nordliga slutstationen, kan de bara få en sydlig rikning.

Antag att tåget Bobby befinner sig på gröna linjen vid station x i Map1, fig. 1 åkandes i sydlig riktning i tidsenhet t. Det finns då två möjliga utfall för Bobby i tidsenhet $t + 1$: Antingen kom Bobby fram till station y, eller så blev Bobby stående på station x (försening). Eftersom station x har en sydlig granne (y) som inte är en slutstation behåller Bobby *sydlig riktning* i båda fallen.

Antag istället att Bobby befinner sig på gröna linjen vid station c i Map1, fig. 1 åkandes i sydlig riktning i tidsenhet t. Station c har en sydlig granne som är en slutstation. I det fallet skiljer sig de två möjliga utfallen för



Figur 1: Två kartor på möjliga tåglinjer. Tåglinjer är färgade. Varje tåg kör fram och tillbaka på endast en linje. En station kan knyta samman flera linjer (t ex Station C i Map 1 eller station X i map 2)

Bobby något från tidigare scenario i tidsenhet $t + 1$: Antingen kom Bobby fram till slutstation z och byter då riktning från sydlig till *nordlig riktning*, eller så blev Bobby ståendes på station c (försening) och behåller sydlig riktning.

Varje tågstation har en specifik sannolikhet p att det blir försening ($p \in [0, 1]$). Simuleringen stegar fram en tidsenhet i taget. Vid varje omgång ska användaren kunna fråga var tåg k, $k \in [1, n]$ befinner sig och vilken riktning de har.

Format på input

Input består av två filer. En kommaseparerad fil (stations.csv) innehållandes tågstationer (kolumn1) och sannolikhet för försening (kolumn2). Exempel för Map1 i fig. 1 visas nedan. Vi kan tolka från denna fil att gröna linjen har större sannolikhet för förseningar (tex signalfel) och att station c är värst drabbad.

```
a,0.001
b,0.001
c,0.2
d,0.001
x,0.1
y,0.1
z,0.1
```

Den andra filen är också en kommaseparerad fil (connections.csv) innehållandes tågstation1 (kolumn1), tågstation2 (kolumn2), linje (kolumn3), och riktning (kolumn4). S står för Syd och N för Norr. Denna fil beskriver vilka kanter som finns i grafen och deras linje. Till exempel beskriver första raden att: (i) det finns tågräls mellan a och b, (ii) de båda stationerna är angränsande stationer, (iii) att åka från a till b betyder att vi åker söderut.

```
a,b,blue,S
b,c,blue,S
c,d,blue,S
x,y,green,S
```

```
y,c,green,S
c,z,green,S
```

Uppgift 1

Vid varje steg i modellen ska programmet fråga användaren om ett val, där valen är att

1 Simulera en tidsenhet framåt i modellen utan att visa tåginfo.

2 Visa nuvarande plats på ett tåg.

q Avsluta.

Exempelkörning

Vi visar här ett exempel med både försening och byte av färdriktning. Programment förväntas ha föjande ungefärliga interaktion:

```
Enter locations of station file: stations.txt
Enter locations of connections file: connections.txt
Enter how many trains to simulate: 3

continue simulation [1], train info [2], exit [q].
Select an option: 2
which train [1 - 3]: 1

Train 1 on GREEN line is at station C heading in North direction

continue simulation [1], train info [2], exit [q].
Select an option: 1

continue simulation [1], train info [2], exit [q].
Select an option: 2
which train [1 - 3]: 1

Train 1 on GREEN line is at station C heading in North direction (DELAY)

continue simulation [1], train info [2], exit [q].
Select an option: 1

continue simulation [1],train info [2], exit [q].
Select an option: 2
which train [1 - 3]: 1

Train 1 on GREEN line is at station Y heading in North direction.

continue simulation [1], train info [2], exit [q].
Select an option: 1

continue simulation [1], train info [2], exit [q].
Select an option: 2
which train [1 - 3]: 1

Train 1 on GREEN line is at station X heading in South direction.
```

```
continue simulation [1], train info [2], exit [q].
Select an option: q
Thank you and Goodbye!
```

Krav uppgift 1

- Korrekt implementation av modellen. För högre betyg måste man även uppfylla de betygskriterier som beskrivs i de generella instruktionerna i *INDU—Individuell Uppgift i Programmering*.
- Uppgifterna ska lösas utan att importera icke-standard bibliotek (exempel på icke-standard bibliotek är bibliotek som måste installeras). Ni får använda vissa standardbibliotek, tex sys och random. Om ni är osäkra om ni får använda ett bibliotek så emaila mig och fråga. Alla använda bibliotek skall redogöras för i projektrapporten.
- Ni behöver inte förutse alla möjliga fell som kan ske i tex inputfiler (det är många) men ni ska visa att ni tänkt på *några* för att få poäng på god felhantering. Sedan finns det många hörnfall på möjliga kartor och parametrar, visa att ni tänkt på några för att få poäng på felhantering och testning.

Tips

Tips1: använd standardbiblioteket random för att slumpa, både förseningar och tåg. Tips2: Tänk på design. Vad som passar som objekt och vad som borde vara i funktioner?

Uppgift 2 (Avancerad)

Vi ska nu öka på programmet i uppgift 1. Programmet ska kunna svara på om det teoretiskt går att pendla mellan station p och q inom T tidsenheter. Valen är:

1 Simulera en tidsenhet framåt i modellen utan att visa tåginfo.

2 Visa nuvarande plats på ett tåg.

3 Svara på om vi kan nå station q från station p inom T tidsenheter (givet vår karta).

q Avsluta.

Alternativ 1 och 2 är från uppgift 1 och alternativ 3 är uppgift 2. Du ska inte anta förseningar eller inkludera tågens nuvarande platser i beräkningen. Du ska bara undersöka om station q ligger högst T stationer bort i kartan från station p. Om the inte finns några kanter mellan två stationer går det inte att åka mellan dem, oavsett T (se tex stationerna u och v till övriga stationer i Map2 i fig. 1).

Exempelkörning

Vi visar här ett exempel på användning av alternativ 3.

```
Enter locations of station file: stations.txt
Enter locations of connections file: connections.txt
Enter how many trains to simulate: 5

continue simulation [1], train info [2], route info [3], exit [q].
Select an option: 3
Select a start station: a
Select an end station: x
Select timesteps: 3

Station x is not reachable from station a within 3 timesteps.

continue simulation [1], train info [2], route info [3], exit [q].
```

```
Select an option: 3
Select a start station: a
Select an end station: z
Select timesteps: 3

Station z is reachable from station a within 3 timesteps.
Select an option: q
Thank you and Goodbye!
```

Krav uppgift 2

- Alla krav som gäller i uppgift 1 gäller för uppgift 2 också.
- I övrigt gäller även att om du gör båda uppgifterna ska du lämna in endast ett program. Dvs du ska utöka funktionaliteten i ditt program från uppgift 1 med uppgift 2. Dock är det bra att spara en kopia av uppgift 1 när du är klar med den innan du försöker implementera uppgift 2.

Betygssättning

Detaljer finns i de generella instruktionerna i *INDU—Individuell Uppgift i Programmering*. Om du bara gör uppgift 1 kan du fortfarande få A i detta projekt. Uppgift 2 ger 0-2 bonuspoäng att lägga till totalen av poäng. En korrekt löst uppgift 2 ger 2 poäng medan en nästan korrekt lösning kan ge 1 poäng.