Bachelorscripties verdelen

Programmeeropdracht 4
Algoritmen en Complexiteit 2022

1 Probleembeschrijving

Aan het einde van de bachelor moet elke student een scriptie uitvoeren. Hier worden elk jaar een aantal onderwerpen voor uitgezocht.

Echter, sommige onderwerpen zijn populairder dan andere. Het kan best zijn dat er 10 studenten voorkeur hebben voor één bepaald onderwerp, maar een ander onderwerp kan wel eens door niemand graag uitgevoerd worden.

Jij bent degene die de scripties mag verdelen. Dit wil je echter op de meest eerlijke manier doen. Je hebt van elke student een lijstje van voorkeursscripties opgehaald. Voor sommige studenten zijn dit er maar een paar (of misschien zelfs nul), maar in principe mogen ze zo veel voorkeuren opgeven als ze willen. Er is geen onderlinge ordening tussen de voorkeursscripties die een student uitkiest (dus ze vinden elke scriptie binnen hun voorkeuren even interessant). Je kunt helaas alleen niet iedereen tevreden stellen. Hoe kun je de scripties zo verdelen dat zo veel mogelijk studenten een scriptie krijgen die hun voorkeur heeft?

Hints:

- Probeer dit probleem te representeren als een gerichte graaf waarbij je de studenten en de scripties representeert als knopen en een verbinding tussen een student en een scriptie maakt als deze student die scriptie als voorkeur heeft.
- Het probleem is te reduceren tot een netwerk-flow probleem dat je bijvoorbeeld met Ford-Fulkerson kunt oplossen. Hoe kun je in de graaf een source en een sinknode toevoegen en deze zodanig verbinden dat het vinden van de maximale flow in feite neerkomt op het vinden van de maximale aantal voorkeuren voor studenten? Hoe moet je in dit geval de capaciteiten kiezen?

2 Invoer/uitvoer

Op de eerste regel staat het aantal studenten s, het aantal bachelorscripties b die te kiezen zijn en het aantal voorkeuren v. Vervolgens volgen er v regels met op elke regel de voorkeur van een student (aangeduid met een nummer beginnend bij 1) voor een bepaalde bachelorscriptie (ook aangeduid met een nummer beginnend bij 1).

Als antwoord wordt verwacht hoeveel studenten maximaal een bachelorscriptie kunnen krijgen die hun voorkeur heeft als we de scripties optimaal verdelen.

In het voorbeeld hieronder is het antwoord hierop 5. Een mogelijke verdeling van scripties die dit mogelijk zou maken is bijvoorbeeld: $\{(1 \to 2), (3 \to 1), (4 \to 3), (5 \to 4), (6 \to 6)\}$ (dit hoeft je programma niet als uitvoer te geven).

2.1 Voorbeeldinvoer

```
6 6 8
1 2
3 1
1 3
3 4
4 3
5 3
5 4
6 6
```

2.2 Voorbeelduitvoer

```
5
```

3 Inleveren

Controleer goed of jouw Pythonversie Python 2 of Python 3 is en zorg dat je Python 3 gebruikt. Zorg er in ieder geval dat je altijd haakjes zet om je print statements (dus gebruik `print("hello world")` in plaats van `print "hello world"`). Dit is een van de dingen waar bij de vorige nakijksessie een aantal programma's op stuk liepen.

Let op: Deze programmeeropdracht wordt voor een groot deel automatisch nagekeken. Zorg dus dat je je precies aan onderstaande richtlijnen houdt, anders kan je code niet worden nagekeken.

Controleer of je code goed werkt voor het inleveren

Door `check.sh` uit te voeren in dezelfde map als je python- en input/output- files kun je controleren of je code de juiste uitvoer geeft op de kleine input (`small.input`) en het juiste inputformaat heeft.

Nakijkeisen

- De code dient geschreven te zijn in Python 3.

- De code leest de input van *stdin* en schrijft naar *stdout.* Om eenvoudig invoer te testen kun je gebruikmaken van input redirection:
  `python ex4.py < small.input`

  Hint: de volgende voorbeeldcode leest op de juiste manier van stdin en schrijft hetzelfde naar stdout.

```
import fileinput
for line in fileinput.input():
    # line bevat al een newline van zichzelf
    print(line, end="")
```

- Het codebestand zelf moet de naam `ex4.py` hebben.
- Lever **alleen** het python bestand met je code in en geef het de naam `ex4.py`. Lever dus geen zip/tar in of iets dergelijks.
- Je uitwerking dient binnen 10 seconden antwoord te kunnen geven op zowel de large als small input.

Hoe wordt de code nagekeken?

Er zijn twee inputfiles aan de hand waarvan je code wordt nagekeken:

- Bijgeleverd is `small.input` met de bijbehorende output `small.output`. Als je code dezelfde output geeft met `small.input` als input, wordt in principe het cijfer 6 toegekend.
  NB: Om deze score te behalen is het wel noodzakelijk dat je daadwerkelijk redelijkerwijs het algoritme hebt geïmplementeerd. Simpelweg altijd dezelfde output retourneren wordt niet goedgerekend.
- Daarnaast is ook voor elke student apart (in `large.input.zip`) een inputfile `large_{studentnummer}.input` bijgeleverd (zonder de bijbehorende output). Als je code ook de juiste output voor deze input geeft, wordt in principe het cijfer 10 toegekend.

De code wordt hiernaast gecontroleerd op plagiaat van anderen en code op het internet. Hierop is de algemene plagiaat- en frauderegeling van de UvA van toepassing.

Geen inputfile ontvangen / overige vragen

Mocht je studentnummer niet tussen de inputfiles staan of mocht je overige vragen/problemen hebben over het inputformaat, stel een vraag tijdens het werkcollege of stel je vraag op Canvas / via email.