Examen Beginselen van Programmeren

27 januari 2024

Vincent Van Schependom

Vraag 1: gestructureerd programmeren

Door de staking in Hollywood moeten er zo veel mogelijk kosten bespaard worden bij het filmen van een nieuwe film. Daarom zoekt de regisseur van die film een zo optimaal mogelijk draaischema voor de opnames. De opnamedagen sluiten aan op elkaar en elke scène duurt exact 1 dag om te filmen. Dat wil dus zeggen dat er evenveel aansluitende filmdagen zijn als er scenes zijn die moeten gefilmd worden.

Elke acteur heeft een bepaalde loonkost per dag dat hij/zij/het werkt. Indien een acteur meer dan 3 dagen pauze heeft tussen twee opnames, heeft die acteur pech. Als er echter minder dan 3 dagen tussen twee opnames zitten, wordt de acteur in kwestie ook uitbetaald voor deze *brugdagen*. In Tabel 1 is een voorbeeld te zien van de gegevens.

Gevraagd:

- Een beschrijving van de voorstellingswijze van de gegevens en de oplossing van het probleem
- Een functie die als invoer de scènes - met bijhorende acteurs - en de loonkost per acteur neemt, en als uitvoer het optimale draaischema met minimale loonkost heeft.
- **Uitbreiding**: wat verandert er indien elke acteur enkel op opeenvolgende dagen op de set kan staan, en er dus geen gaten mogen zitten tussen twee opnamedagen voor elke acteur?

Scène	Acteurs
1	Acteur 1, Acteur 2
2	Acteur 1, Acteur 5, Acteur 7
3	Acteur 2, Acteur 4
...	...
10	Acteur 6, Acteur 2, Acteur 8

Acteur	Loonkost
Acteur 1	10.000
Acteur 2	8.000
Broke acteur	600
...	...
Acteur n	12.000

Tabel 1: de scenes en de benodigde acteurs voor die scenes, alsook de loonkosten voor elke acteur

Vraag 2: objectgericht programmeren

Op een veiling worden er verschillende items verkocht. De eenvoudigste items zijn de generieke items; die hebben enkel een naam en een beschrijving. Daarnaast zijn er ook kunstitems, die ook een kunstenaar hebben. Antieke kunstitems zijn kunstitems waarvan ook het tijdsvak en de stroming wordt bijgehouden. Van elektronische items wordt de staat bijgehouden. Het merendeel van de items die verkocht worden zijn auto's. Daarvan wordt het merk, het model en het bouwjaar bijgehouden.

Veilingen hebben altijd een verkoper. Die verkoper verkoopt dan 1 of meerdere items. Er moet altijd een startprijs opgegeven worden voor de biedingen en de verkoper kan optioneel ook een geheime grensprijs opgeven. Een verkoop is pas geklonken indien het hoogste bod hoger is dan de grensprijs. De winnaar van de veiling is dan de persoon die het hoogste bod heeft gedaan.

Er zijn twee types veilingen: open veilingen en gesloten veilingen. Bij open veilingen is het mogelijk om de boden en de hoogste bieder op te halen, bij gesloten veilingen wordt dit afgeschermd. Bij open veilingen wordt een nieuw bod enkel geregistreerd indien het hoger is dan het huidige hoogste bod, of als het hoger is dan de startprijs indien er nog geen bod werd gedaan. Bij gesloten veilingen worden alle boden geregistreerd die groter zijn dan het huidige hoogste bod.

Er zijn gewone gebruikers en premium gebruikers. Gewone gebruikers kunnen enkel boden uitbrengen, premium gebruikers kunnen daarnaast ook een veiling opstarten als verkoper. Premium gebruikers hebben als voordeel dat, indien er meerdere mensen het hoogste bod hebben, een premium gebruiker steeds de veiling zal winnen. Als er meerdere premium gebruikers het hoogste bod hebben, is de winnaar van de veiling de gebruiker die in het verleden al voor het hoogste bedrag iets heeft verkocht op een veiling. Indien er verder nog een ex aequo is, wordt er een willekeurige winnaar gekozen onder de hoogste bieders.

Gevraagd:

- Maak een klassendiagram dat de veiling voorstelt.
- Maak een functie die een gebruiker in staat stelt een bod te doen.
- Maak een functie die de winnaar van een veiling bepaalt.

Vraag 3: interpreteren van code en complexiteit

- Wat is de output van onderstaande code?
- Beschrijf in woorden wat de functie `waarde(.)` precies berekent.
- Wat is de complexiteit van de functie `waarde(.)`?
  Toon dit aan met een formele complexiteitsberekening.
- Schrijf zelf een efficiëntere functie en toon aan dat je eigen functie effectief beter is aan de hand van een complexiteitsberekening.

```
def waarde(p):
    i = len(p)-1
    r = []
    while i>=0:
        j = 0
        while j<=i:
            w = p[i]-p[j]
            r.append(w)
            j += 1
        i -= 1
    w = None
    for x in r:
        if w == None or abs(x) > w:
            w = abs(x)
    return w

print([1,2,3,4,5])
print([5,4,3,2,1])
print(range(0,100,2))
print(range(100,1000,50))
```