Practica Datastructuren en Algoritmen (2018-2019)

dr. ir. Annemie Vorstermans

1 Inleiding

De code voor onderstaande oefeningen wordt ingediend via Toledo. De uiterste datum om de oefeningen in te dienen is vrijdag 21 december 2018. De laatste feedback wordt op maandag 17 december 2018 gegeven. Het niet-maken van de oefeningen leidt tot een NA voor deze opleidingsactiviteit.

Voor elke oefening moet er 1 Java-file worden ingediend die Main.java moet noemen. De klasse met de main-methode noem je Main. De andere klassen kunnen ook in dezelfde file worden opgenomen als je de public voor die klassen weglaat. Je mag geen packages gebruiken. Per oefening kan je max 5 maal een oplossing indienen. Correctheid en snelheid zijn de belangrijkste punten, maar ook de leesbaarheid is belangrijk (alle namen in het Nederlands zoals de opgave!, zinvolle commentaar).

De oefeningen worden individueel gemaakt! Gekopiëerde code (= fraude) zorgt voor een 0 op de opleidingsactiviteit, ook voor de bron. Extra uitleg/hulp kan gevraagd worden tijdens de contactmomenten of na afspraak (Annemie.Vorstermans@cs.kuleuven.be).

2 Lineaire datastructuren

2.1 Braille

Braille is een reliëfalfabet, waarbij letters (en andere tekens) door middel van puntjes in een drager (zoals papier) gedrukt zijn: voor elke letter is er een unieke combinatie van puntjes in een 2 op 3 raster. Braille leest men door met de vingers over die puntjes te glijden en te voelen welke letter er staat. Braille omzetten naar *zwartschrift*, dat is de opdracht hier. Als invoer krijg je eerst de codering die gebruikt wordt voor ons alfabet van 26 tekens. Daarna krijg je een aantal teksten in braille die je moet omzetten naar ons alfabet.

invoer De invoer bestaat uit:

- drie regels met op elke regel 52 tekens, alle een punt (`.`) of een letter `x`: die stellen per twee kolommen de braille-encodering van de letters A tot en met Z voor;
- één regel met het aantal testgevallen;
- per testgeval drie regels die samen een tekst in braille voorstellen, ook weer per twee kolommen één teken.

```
x.x.xxxxx.xxxxx..x.xx.x.xxxxx.xxxxx..x.xx.x..xxxxxx.
..x....x.xx.xxxxx.xx..x....x.xx.xxxxx.xx..x.xx...x.x
...................x.x.x.x.x.x.x.x.x.x.xxxx.xxxxxxx
4
x.
..
..
x.xx.x
x.x.xx
xxx..x
x.x.x.x.xx.xx.xxx.x.xxx.x.xxxxx.x.x..xx.xx.x.xx..x.xxx
x.x.......x..xx.xx.xxxxx.......x.xxxxx.x.xx.xxxxx.xx.x
xxx.....x.x...x.x.x...x...x.x.....x..x....x.x.x......
x.x.xxxxx.xxxxx..x.xx.x.xxxxx.xxxxx..x.xx.x..xxxxxx.
..x....x.xx.xxxxx.xx..x....x.xx.xxxxx.xx..x.xx...x.x
...................x.x.x.x.x.x.x.x.x.x.xxxx.xxxxxxx
```

uitvoer Voor elk geval antwoord je met één enkele regel. Deze bevat, gescheiden door één spatie, volgende informatie:

1. het volgnummer van het geval. Dit begint bij 1 en wordt telkens verhoogd bij elk volgend geval;
2. de overeenkomstige alfabetische tekst van dit geval.

```
1 A
2 VPW
3 VLAAMSEPROGRAMMEERWEDSTRIJD
4 ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

2.2 Cluedo

Als leid(st)er van de lokale jeugdbeweging organiseer je een spelletjesavond voor de leden. Jij bent toezichter bij een spelletje Cluedo, waar vier spelers samenwerken om een moord op te lossen: de spelers zoeken uit wie de moordenaar is, waar de moord is gepleegd en met welk wapen. Elke speler begint het spel met een aantal kaartjes die elk één aanwijzing bevatten die een bepaalde dader, locatie of moordwapen uitsluit. Jij probeert het spel mee te spelen door aan de hand van de vragen en antwoorden van spelers te achterhalen wie welke kaartjes heeft.

Elke beurt mag een speler vragen of iemand een kaartje heeft voor een persoon OF een locatie OF een wapen. De eerste speler — in wijzerzin — die een van deze kaartjes heeft moet dit gedekt tonen aan de vragende speler: als een speler meer dan één van de gevraagde aanwijzingen heeft kiest hij zelf welke hij laat zien, maar hij toont er juist één. Het kan gebeuren dat geen enkele speler een kaartje kan tonen op een bepaalde vraag.

Stel je voor dat speler 1 een vraag stelt voor persoon **3**, locatie **E** en wapen **a**. Speler 2 en 3 passen, maar speler 4 toont een kaartje aan speler 1. Hieruit kan je afleiden dat spelers 2 en 3 geen enkel van de kaartjes {**3**, **E**, **a**} hebben, en dat speler 4 juist wel één (of meerdere!) van deze kaartjes heeft. Als speler 1 later opnieuw een vraag stelt, deze keer voor persoon **3**, locatie **E** en wapen **c** en speler 2 antwoordt wel, dan weet je dat deze speler het kaartje voor wapen **c** heeft, want deze speler moest passen voor de eerste vraag met **3**, **E** en **a**.

Invoer De invoer bestaat uit een aantal gevallen. Elk geval wordt voorgesteld door een aantal lijnen. De eerste lijn bestaat uit drie getallen $2 \leq p, l, w < 10$, welke het aantal mogelijke personen, locaties en wapens voorstellen. Eén persoon, locatie en wapen zijn de gezochte oplossing voor het spel, en de overige $(p-1)+(l-1)+(w-1)$ kaartjes worden gelijkmatig over vier spelers verdeeld zodat elke speler evenveel kaartjes heeft. De tweede lijn n van het geval geeft aan hoeveel vragen (en antwoorden) gegeven zijn. De resterende n lijnen bestaan uit drie componenten, gescheiden door een spatie:

1. Het volgnummer van de speler die de vraag stelt.
2. De vraag, bestaande uit drie karakters. Het eerste karakter is de persoon, voorgesteld door een cijfer uit de set $\{1, 2, \ldots\}$. Het tweede karakter is de locatie, voorgesteld door een hoofdletter uit de set $\{\texttt{A}, \texttt{B}, \texttt{C}, \ldots\}$. Het derde karakter is het wapen, voorgesteld door een kleine letter uit de set $\{\texttt{a}, \texttt{b}, \texttt{c}, \ldots\}$.
3. Het volgnummer van de speler die antwoordt, of `X` als niemand antwoordt.

```
6
4 4 3
16
1 1Aa 2
2 2Db 4
3 4Ab 2
4 4Ac 2
1 1Cb X
2 4Cb 3
3 3Cc 4
4 2Bb X
1 2Da 2
2 3Ac 3
3 2Aa 2
4 3Cb 1
1 3Dc 3
2 4Ba 3
3 1Aa 1
4 2Dc X
2 3 2
8
1 2Aa X
2 2Ab 4
3 2Ab 4
4 2Bb 1
1 1Ab 3
2 1Aa 3
3 1Ca 2
4 2Ca 2
2 3 2
8
```

```
1 1Ca 3
2 1Aa 4
3 1Ab X
4 1Cb 3
1 2Ba 2
2 2Aa 4
3 1Ba 4
4 2Cb 1
2 3 2
12
1 1Ca 3
2 2Aa 1
3 2Ca 1
4 2Cb 3
1 2Ca 3
2 1Aa 4
3 1Aa 4
4 1Ab X
1 1Bb 2
2 2Ba 1
3 2Aa 1
4 1Cb 3
8 6 5
32
1 2Da 2
2 2Cc 4
3 7Ce 4
4 8Bd 1
1 3Fc 3
2 6Bc 4
3 1Ca 4
4 8Ee 2
1 6Ea 4
2 1Be 1
3 1Bd 1
4 3Aa 3
1 5Dd 2
2 5Ce 4
3 2Fb 1
4 3Dd 2
1 8Fa 3
2 8Da 3
3 1Ae 4
4 4Fc 3
1 8De 2
2 1Fe 1
3 2Cd 4
```

```
4 5Ab 2
1 2Bd 3
2 6Ba 1
3 6Dd 1
4 5Cc X
1 7Ec 2
2 6Ad 3
3 6Dc 4
4 6Cc 1
7 9 7
40
1 2Ea 3
2 6Ae 4
3 7Ig X
4 6Ge 2
1 2Cf 2
2 7Ce 3
3 7Ce 1
4 2Id 2
1 1Hf 2
2 2Fd 4
3 5Ca 4
4 7Hg 2
1 7He 2
2 3Ab X
3 1Fa 4
4 7Gc 1
1 2Fc 4
2 2Ig 3
3 1Da 4
4 1Ed 2
1 2Aa 2
2 1Cd 3
3 1Bb 1
4 2Ff 2
1 5Fe 2
2 7Bf 3
3 3Aa 1
4 7Db 3
1 6Cc 4
2 5Ia 3
3 4Hg 1
4 6Ba 1
1 1Ie 2
2 6Cg 3
3 2Da 4
4 1Gb 3
```

```
1 6Ie 2
2 4Gc 1
3 2Hc 4
4 3Be 1
```

Uitvoer Voor elk geval antwoord je met een enkele lijn. Deze bevat — gescheiden door spaties — volgende informatie:

1. Het volgnummer van het geval. Dit begint bij 1 en hoogt op voor elk volgend geval.
2. De kaartjes van speler 1 tot en met 4, telkens lexicografisch gesorteerd (eerst personen, dan locaties, dan wapens). Dit komt overeen met de ingebouwde sort-functionaliteit van de meeste programmeertalen.

```
1 1C Aa 34 Dc
2 B C 1 b
3 2 B C a
4 a B C 1
5 126B 7Dbe 348d ACEc
6 4BCac AHdef 17EIg 256DF
```

2.3 Achtbaan

Het is tijd om reclame te maken voor de pretparken en al hun achtbanen. In de brochure komt een mooi zijaanzicht van elke achtbaan, samen met de naam, de lengte, G-krachten, enz.

Deze opgave bestaat eruit om een zijaanzicht te tekenen van een achtbaan, gegeven een lijst van de spoorsegmenten die gebruikt werden. Dit zijaanzicht beschouwt de achtbaan vanuit het zuiden. Elke achtbaan begint met een station op grondniveau, met vertrek naar het oosten (in zijaanzicht dus naar rechts).

Je krijgt een aantal segmenten en moet aan de hand van die segmenten de achtbaan weergeven. Elk segment is even lang. We definiëren volgende segmenten aan de hand van een teken:

`S` Station, op- en afstapplaats.

`V` 1 vakje gewoon rechtdoor.

`U` 1 vakje diagonaal omhoog.

`D` 1 vakje diagonaal omlaag.

`L` 90° bocht naar links.

`R` 90° bocht naar rechts.

Deze laatste twee segmenten veranderen de rijrichting in de spanne van 1 vakje. Met andere woorden, als je een vakje binnenrijdt in oostelijke richting zorgt een `L`-bocht ervoor dat je buitenrijdt in het noorden. Het volgende segment begint dan 1 vakje ten noorden van het originele vakje.

Elk karakter van de tekening is het *voorste* segment gezien vanuit het zuiden. Het laatste stuk van de eerste voorbeeld-achtbaan loopt bijvoorbeeld achter de vertrekhalte. Merk ook op dat een

diagonaal omlaag segment reeds 1 vakje lager getekend wordt. Voor sommige segmenten hangt het karakter dat je moet tekenen af van de oriëntatie van het segment:

Segment	N	O	Z	W
Station		=		
Rechtdoor	_	_	_	_
Omhoog	#	/	#	\
Omlaag	#	\	#	/
Linksaf	_	_	_	_
Rechtsaf	_	_	_	_

(Noot: slechts 1 oriëntatie mogelijk)

Invoer De eerste regel van de invoer bevat een geheel getal $1 \leq n \leq 1000$ dat het aantal testgevallen aangeeft. Per geval volgt dan een enkele regel. Alle regels worden beëindigd met een enkele newline \n.

Elk geval bestaat uit een regel met het aantal segmenten $6 \leq s \leq 500$, een spatie, en dan s tekens van segmenten als een aaneengesloten tekenreeks.

Elke achtbaan is gecertifieerd als veilig, dit wil zeggen dat het traject een lus vormt en zichzelf niet kruist op eenzelfde hoogte.

```
3
8 SVLLDULL
24 SSVUUVVDLVLVVVVVVVVDLVLV
58 SSSSSVVUUUUVVDDDDUULVLVVVVVVVVVVVVUUUVVVDDDVVVLDDVLVVVVLVR
```

Uitvoer De uitvoer voor elk geval bestaat uit een aantal lijnen. Elke lijn begint met het volgnummer (dat begint bij 1 en ophoogt voor elk opeenvolgend geval) en een spatie. De resterende karakters van de lijn zijn ofwel een puntje (.) als er op die locatie geen stuk van de achtbaan is, ofwel het karakter dat overeenkomt met het *voorste* segment zichtbaar op die locatie. De uitvoer is de kleinst mogelijke rechthoek die het zijaanzicht van de achtbaan geheel bevat. Met andere woorden, er mag geen rij of kolom in de uitvoer staan die uitsluitend uit puntjes bestaat.
Let op! Zorg ervoor dat je uitvoer geen overbodige tekens bevat, bijvoorbeeld een spatie op het einde van een regel of een lege regel op het einde van de uitvoer. Dat zorgt er immers voor dat je uitvoer als foutief wordt beschouwd.

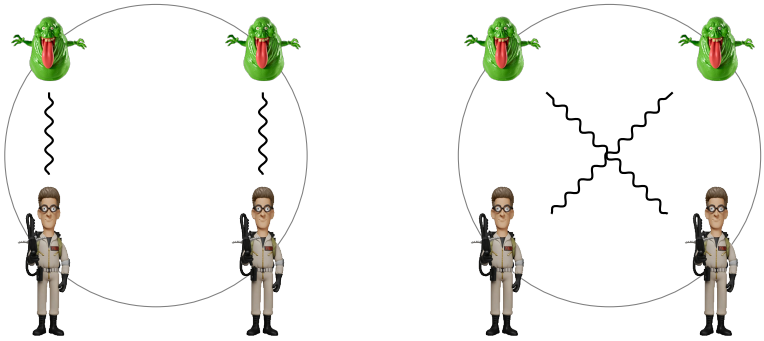
```
1 _=__
1 .\/.
2 ........__..
2 ..____/__\_
2 __==_/.....
3 ........___................
3 ......./...\......__.......
3 ...../.....\..../..\......
3 ____/.......\__/____\_____
3 #............./......\../.
3 ______=====__/........\/..
```

3 Recursie en backtracking

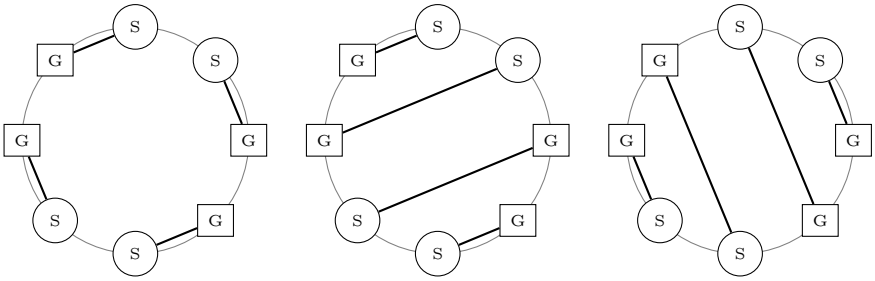
3.1 Ghostbusters

N ghostbusters en N spoken staan in een cirkel. Elke ghostbuster moet op exact één spook schieten, en geen twee ghostbusters mogen schieten op eenzelfde spook. Deze 1-op-1 associatie noemen we "schietconfiguratie". Hoeveel mogelijke schietconfiguraties zijn er, rekening houdend met het feit dat de stralen elkaar niet mogen snijden?

Hieronder staat een illustratie voor $N = 2$. Enkel de linkerfiguur geeft een geldige schietconfiguratie weer: in de rechterfiguur snijden de stralen elkaar.



De spoken en ghostbusters kunnen elkaar willekeurig afwisselen. Bijvoorbeeld, gebruik makende van een abstractere visualisering, waarbij spoken en ghostbusters worden voorgesteld door S en G, respectievelijk:



De positionering van spoken en ghostbusters wensen we voor te stellen als een string. Hiervoor kiezen we een willekeurige beginpositie en rotatiezin en gaan we vervolgens de cirkel rond. Een spook stellen we voor met `S`, een ghostbuster met `G`. De positionering uit bovenstaand voorbeeld kunnen we encoderen als `GSSGGSSG`, maar tevens met bijvoorbeeld `GGSSGGSS` of `GGSSGGSS`.

invoer De eerste regel bevat een positief geheel getal dat het aantal testgevallen voorstelt. Hierop volgt één regel per testgeval. Deze regel bevat een string bestaande uit `G`'s en `S`'sen die de positionering van ghostbusters en spoken voorstelt. De string bevat altijd evenveel `G`'s als `S`'sen.

```
9
GS
SG
GSGS
GSGGSS
GSSGGSSG
```

```
GSGSGS
SGGSSGGS
GGSGGSSSSSGGGSGGGSSS
SGSSSSSSGGSGSGSGSGGGGSSGGSGSSSSGSSGGGGGG
```

uitvoer Per testgeval voer je één regel uit. Deze bevat twee getallen gescheiden door één enkele spatie.

- Het eerste getal stelt de index van het testgeval voor, waarbij het eerste testgeval index 1 heeft.
- Het tweede getal stelt het aantal mogelijke schietconfiguraties voor.

```
1 1
2 1
3 2
4 2
5 3
6 5
7 3
8 20
9 12894
```

3.2 Satelliet

De signalen van een satelliet van een concurrerende natie worden opgepikt. De scan van de signalen over een zekere tijdspanne wordt voorgesteld door een rij `1`'en en `0`'en. Een voorbeeld van zulk een rij staat hieronder:

```
110100
```

Na analyse van de scans is gebleken dat de satelliet dikwijls dezelfde opeenvolging van signalen uitstuurt. Deze opeenvolgingen worden voorgesteld aan de hand van lettercodes. Hieronder staat een aantal voorbeelden van lettercodes:

```
A    10
B    0101
C    11
D    1
E    0
```

De scan uit het bovenstaande voorbeeld `110100` kan dus aan de hand van de lettercodes verkort beschreven worden als `CEDEE`.

Gevraagd wordt om een algoritme op stellen dat de omzetting van een scan naar een zo kort mogelijke beschrijving automatiseert.

invoer De eerste regel van de invoer bevat het aantal testgevallen. Per testgeval volgt

- Eén regel met de scan van de signalen van de satelliet. Je mag er vanuit gaan dat de scan maximaal 10000 keer een `1` of een `0` bevat.
- Een getal met het aantal lettercodes. Dit getal ligt tussen 1 en 20.
- Per lettercode: een regel met een hoofdletter en een regel met de opeenvolging van signalen. De opeenvolging van de signalen bestaat uit maximaal 20 keer een `1` of een `0`.

Een voorbeeld van de invoer wordt hieronder getoond.

```
31
0110000
4
C
000
U
1
I
01
V
10
00010000
2
G
000
J
10
001001001
3
P
010
U
001
G
11
1001101
4
A
1
E
01
N
10
Z
101
00110110
4
```

E
010
Y
10
S
001
G
101
11101010
2
H
10
V
11
110111111110
3
U
111
G
110
W
1
110110
5
Y
1
P
0
M
1011
C
00
U
01
110110000000
3
G
110
V
01
C
000
0100011010
3
F
1
B

001
U
010
01101011
2
K
011
Q
01
101111101101111
2
L
111
D
101
010010
3
R
111
I
0
N
01
0001000010
3
P
01
Z
00
H
10
1100
6
F
1
Z
01
G
011
W
0
X
000
V
00
01010
5

G
001
X
00
Z
111
C
0
Y
01
0000000
4
X
000
L
11
W
10
Z
00
010101010010011
5
C
010
L
011
X
01
P
10
J
101
1011101101101
3
Q
101
A
011
B
1
10001100100
3
M
011
L
100
N

01
11110110
3
X
101
P
10
Z
111
010000
4
W
00
U
01
V
10
Q
1
111010010111
3
B
001
Q
111
V
010
01001010100001
3
A
00
C
01
L
10
101010
3
V
10
Q
010
R
0
0000100100
2
N
00

X
100
0011010101011010
4
F
10
M
010
C
001
P
101
100100011011011011100100
3
F
011
S
1
Z
100
010110111
6
H
1
S
0
O
11
L
01
E
101
U
00
100011011100001
5
B
001
X
11
R
100
F
010
P
011
110101

```
6
R
10
M
110
E
1001
V
00
F
1
0
01
```

uitvoer Per testgeval wordt het volgnummer van het testgeval gevolgd door de kortste notatie afgedrukt. Mochten er meerdere notaties de kortste zijn wordt alfabetisch gezien de eerste afgedrukt. Mocht het algoritme niet in staat zijn om een verkorte voorstelling te genereren aan de hand van de gegeven lettercodes dan wordt het volgnummer van het testgeval gevolgd door 'ONMOGELIJK' afgedrukt.

```
1 IVC
2 GJG
3 UUU
4 NEZ
5 SGY
6 VHHH
7 GUUG
8 YMP
9 GGCC
10 UBFU
11 KQK
12 DLDDL
13 NINI
14 ZPZZH
15 FFV
16 YYC
17 XZZ
18 CJCCL
19 BAQQQ
20 LNLL
21 ZXP
22 UWW
23 QVVQ
24 CALLLAC
25 VVV
26 NNXX
27 CPMPFF
```

```
28 ZZFFFFZZ
29 LLEO
30 RPPRB
31 MFO
```

3.3 Cupcakes

Dankzij diverse tv-programma's zijn cupcakes razend populair, en winkels kunnen nauwelijks aan de vraag voldoen. Er zijn twee soorten klanten: degenen die aangeven *ik wil een vol doosje van 6 stuks, en eentje van 13 stuks* (op voorwaarde dat er zulke doosjes bestaan natuurlijk), maar anderen bestellen gewoon *19 cup cakes*. Het is dan aan de winkelier om te beslissen in welke doosjes hij die 19 stuks levert. Het kan voor hem voordeliger zijn om (zeg maar) 7 doosjes voor 3 stuks te gebruiken (waarbij één van de doosjes slechts 1 cup cake bevat) dan enige andere combinatie van doosjes.

Je krijgt een rij doosjes, met voor elk doosje zijn maximale capaciteit (de maat van de doos) en kost - van elke soort zijn er onbeperkt veel. Je krijgt ook het aantal bestelde cupcakes. Je bepaalt de goedkoopste manier om alle bestelde cupcakes in doosjes te steken. Bijvoorbeeld:

```
maat  kost
  1    3
  2    5
  3    7
  5    9

bestelling 4
```

De bestelling kan in doosjes met maat 1 en 3 (kost = 3+7 = 10) of 2 doosjes van 2 (kost = 10) of in 1 doosje van 5 (kost = 9). De laatste is de goedkoopste. Je drukt 9 af.

Er bestaat niet altijd een doosje met maat 1. In de opgaven zal altijd een doos met grotere maat strikt meer kosten dan een doos met kleinere maat - anders zou die kleinere doos overbodig zijn.

Invoer: De eerste regel van de invoer bevat een geheel getal $1 \leq n \leq 1000$ dat het aantal testgevallen aangeeft. Per geval volgen dan een aantal regels met informatie: op de eerste regel staat het aantal bestelde cupcakes; op de tweede regel staat het aantal soorten dozen; dan volgt een regel voor elke soort doos, met daarop voor die soort de maat en de kost (beide zijn gehele getallen). De grootte van de dozen is strikt stijgend, en de kost ook.

Alle getallen in de invoer die op dezelfde regel voorkomen, worden gescheiden door 1 enkele spatie; alle regels worden beëindigd met een enkele newline `\n`.

Uitvoer: De uitvoer bestaat uit n regels met daarop eerst het volgnummer van het testgeval (begin te tellen bij 1, oplopend tot n), daarna een spatie en daarna voor het testgeval de minimale kost om de bestelling te verpakken.

Let op! Zorg ervoor dat je uitvoer geen overbodige tekens bevat, bijvoorbeeld een spatie op het einde van een regel of een lege regel op het einde van de uitvoer. Dat zorgt er immers voor dat je uitvoer als foutief wordt beschouwd.

Voorbeeld:

Invoer:

```
2
4
4
```

```
1 3
2 5
3 7
5 9
11
4
1 28
3 35
5 55
10 100
```

Uitvoer:

```
1 9
2 125
```

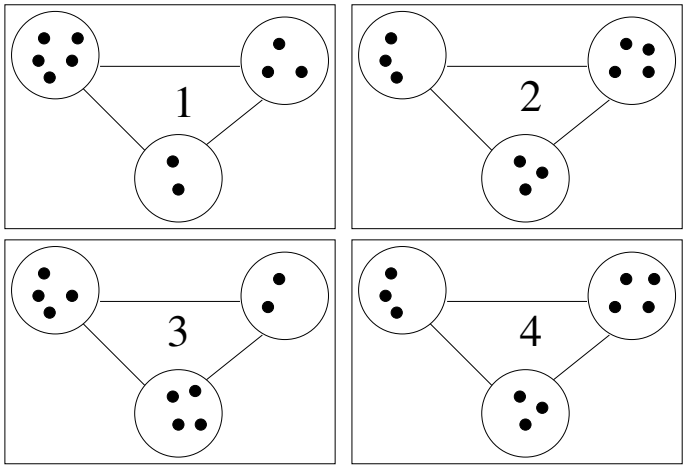
4 Grafen

4.1 Keigraaf

Een keigraaf is een niet-gerichte graaf met in elke knoop een aantal keien. Een keigraaf evolueert. Bij elke tik van de klok geeft elke knoop zoveel mogelijk keien aan al zijn buren, maar wel aan alle buren even veel: de knoop houdt de rest. Bijvoorbeeld: als een knoop 7 keien heeft, en 3 buren, dan geeft hij aan elke buur 2 keien en houdt er 1. Het is mogelijk dat een knoop geen keien kan weggeven. Bijvoorbeeld: als een knoop 8 buren heeft en slecht 4 knopen, dan houdt hij gewoon alles.

Vanuit een beginsituatie zal na een tijdje een bepaalde configuratie herhaaldelijk optreden. Het aantal kloktikken tussen twee identieke configuraties is de periode van de overeenkomstige beginsituatie. Jullie schrijven een programma dat die periode berekent.

Hieronder staat een voorbeeld. De graaf heeft drie knopen en de keien zijn getekend als volle zwarte cirkeltjes. De 4 configuraties zijn genummerd volgens hun kloktik. Je ziet de evolutie en ook dat de configuratie 2 weer optreedt 2 tikken later. De periode is dus 2. Merk ook op dat het niet de beginconfiguratie is die herhaald wordt in dit voorbeeld, maar dat kan natuurlijk wel in andere gevallen.



Invoer De eerste regel van de invoer bestaat uit het aantal testgevallen $1 \leq n \leq 1000$. Daarna volgen de regels voor de n testgevallen. Elk testgeval bestaat uit de volgende regels.

- een regel met het aantal knopen $1 \leq k \leq 50$ van de graaf, gevolgd k regels – 1 voor elke knoop in de graaf. De i-de regel bevat een enkel geheel getal dat het aantal keien $0 \leq k_i \leq 100$ voorstelt voor de i-de knoop. Merk op dat het aantal keien gedurende de uitvoering natuurlijk wel groter dan 100 kan worden.
- een regel met een enkel geheel getal $0 \leq b \leq 1500$ dat het aantal bogen in de graaf voorstelt, gevolgd door b regels — 1 regel voor elke boog in de graaf. Een dergelijke regel bevat twee gehele getallen gescheiden door een enkele spatie. Deze getallen stellen de knopen voor die door de boog verbonden worden. De knopen zijn vanzelfsprekend genummerd van 1 t.e.m. k.

Een testgeval wordt dus omschreven door $k + b + 2$ regels.

Uitvoer De uitvoer bestaat uit n regels met de periodes van de overeenkomstige testgevallen.

Voorbeeld Voor de keigraaf in de bovenstaande tekening is de invoer:
Invoer

```
1
3
5
3
2
3
1 2
2 3
1 3
```

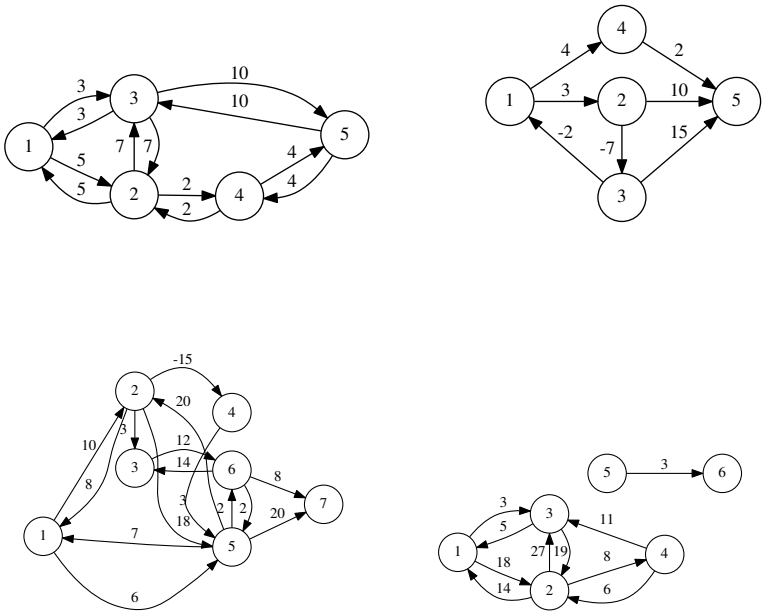
Uitvoer
De uitvoer voor dat voorbeeld is

```
2
```

4.2 Jediparcours

Voor de evaluatie van de Jedi apprentices op Shedu Maad is er een parcours aangelegd met opdrachten. Er zijn meerdere wegen (met dan ook mogelijk andere opdrachten) om van het startpunt naar het eindpunt te geraken. De Jedi apprentice kent het parcours met de opdrachten en kan dus inschatten welke score hij bij elke opdracht zal krijgen. Scores kunnen positieve (strafpunten) of negatieve (bonuspunten) gehele getallen zijn.

Schrijf nu een programma om de Jedi apprentices te helpen bepalen wat de laagst mogelijke totale score is die ze kunnen behalen. Het is mogelijk dat het eindpunt niet bereikbaar is. Het kleinst mogelijk aantal strafpunten is dan gelijk aan plus oneindig. Als het eindpunt bereikbaar is, kan het ook zijn dat het parcours een lus bevat waarvan de som van de scores negatief is. Als dit op een pad van startpunt naar eindpunt ligt is het kleinst mogelijk aantal strafpunten gelijk aan min oneindig. Ligt de negatieve lus in een onbereikbaar deel (van start en eindpunt) dan heeft deze natuurlijk geen invloed en is er een normale score.



Invoer De eerste regel bevat het aantal testgevallen. Per testgeval volgt

- een regel met het aantal knooppunten (startpunt en eindpunt inbegrepen) gevolgd door een spatie gevolgd door het aantal verbindingen met opdrachten tussen deze knooppunten.
- aantal verbindingen regels met per regel: beginknooppuntnummer spatie eindknooppuntnummer spatie strafpunten

De knooppunten zijn steeds genummerd van 1 (altijd het startpunt) tot en met het aantal knooppunten (altijd het eindpunt). Het aantal knoopunten ligt steeds in $[3, 100]$. De strafpunten (bonuspunten zijn negatieve strafpunten) liggen in het interval $[-20, 50]$

Hieronder vindt men een voorbeeld van invoer passend bij de figuren.

```
4
5 12
1 2 5
2 1 5
1 3 3
3 1 3
2 3 7
3 2 7
2 4 2
4 2 2
4 5 4
5 4 4
3 5 10
5 3 10
5 7
1 2 3
```

```
3 1 -2
1 4 4
2 3 -7
2 5 10
3 5 15
4 5 2
7 15
1 2 10
2 1 8
2 4 -15
2 5 18
5 2 20
4 5 3
1 5 6
5 1 7
2 3 3
3 6 12
6 3 14
5 6 2
6 5 2
5 7 20
6 7 8
6 10
1 2 18
2 1 14
1 3 3
3 1 5
2 3 27
3 2 19
2 4 8
4 2 6
4 3 11
5 6 3
```

Uitvoer Per testgeval dien je één regel uit te voeren. Deze regel bestaat uit

- de index van het testgeval, beginnende bij 1;
- één spatie;
- het aantal strafpunten of *plus oneindig* in het geval het eindpunt niet bereikbaar is, of *min oneindig* in het geval er een negatieve lus is.

Voor bovenstaande invoer wordt de volgende uitvoer gegenereerd.

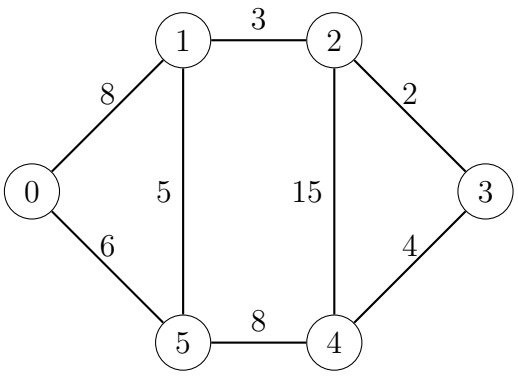
```
1 11
2 min oneindig
3 8
4 plus oneindig
```

4.3 Pizzakoerier

In een stad zijn er meerdere pizzeria's: zij posten de bestellingen op het net en jij kan als zelfstandig pizzakoerier kiezen welke leveringen je voor je rekening wenst te nemen. Uiteraard wil je je keuze zo te maken dat je je winst maximaliseert. Daarvoor gebruik je een app waarvan nu een stukje geïmplementeerd moet worden ...

De stad kan je je voorstellen als een gewogen bidirectionele graaf.

- Elke locatie (klant of pizzeria) komt overeen met een node en krijgt een uniek geheel getal als identificatiecode.
- Wegen tussen locaties worden voorgesteld door bogen.
- Het gewicht van een boog tussen nodes X en Y stelt de tijd voor die je nodig hebt om je te verplaatsen van X naar Y of van Y naar X: er is geen sprake van eenrichtingsverkeer.



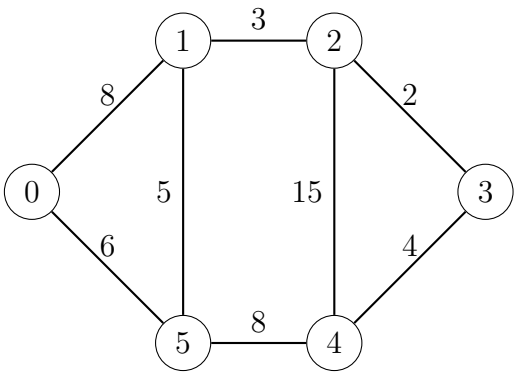
Klanten geven hun bestelling op tijd in. Een bestelling wordt voorgesteld door een tuple (a, b, t, w):

- a is de locatie van de pizzeria waar de klant pizza('s) heeft besteld.
- b is het leveringsadres.
- t is het tijdstip waarop de pizza exact moet geleverd worden. Pizza's mogen te vroeg noch te laat geleverd worden.
- w is de winst die je maakt voor de levering.

Je begint elke werkshift thuis, voor het gemak aangeduid als locatie 0 en tijdstip 0. Je bent vrij te kiezen welke leveringen je uitvoert: de overblijvende leveringen zullen wel door een andere koerier gedaan worden - of niet, who cares.

Je kan telkens maar één bestelling per keer afhandelen. Het is dus niet toegelaten om twee of meer verschillende bestellingen op te halen (in dezelfde of meerdere pizzeria's) en deze dan te leveren aan twee of meer klanten. M.a.w. je rijdt altijd alternerend tussen pizzeria en klant.

Voorbeeld



Beschouw de bestellingen $(2, 4, 20, 10)$ en $(5, 4, 35, 15)$. We noteren je huidige toestand als (ℓ, t), waarbij ℓ je locatie en t het huidige tijdstip voorstelt.

Stel, je voert de eerste bestelling uit: je begint op $(\ell = 0, t = 0)$, je rijdt $(0, 0) \to (1, 8) \to (2, 8 + 3 = 11)$. Je weet dat je op locatie 4 moet zijn tegen tijdstip 20, wat enkel mogelijk is indien je $2 \to 3 \to 4$ rijdt. Deze weg duurt 6 minuten, dus je wacht tot $t = 14$ om de bestelling op te pikken, en je rijdt $(2, 14) \to (3, 16) \to (4, 20)$ waar je de pizza's aan de gelukkige klant overhandigt. Je hebt nu 10 euro winst gemaakt.

Je kan echter de tweede bestelling niet meer tijdig afhandelen: het is reeds $t = 20$ en je moet $4 \to 5 \to 4$ rijden, wat je minimaal 16 tijdseenheden kost. Je kan de tweede bestelling onmogelijk geleverd krijgen voor $t = 36$ terwijl de klant wenst deze om $t = 35$ te ontvangen.

De optimale aanpak bestaat er dus in om de eerste bestelling aan een andere koerier over te laten: je rijdt $(0, 0) \to (5, 6)$, wacht tot $t = 27$, je pikt de pizza's op, je rijdt $(5, 27) \to (4, 35)$ en je verdient 15 euro.

invoer De eerste regel bevat een positief geheel getal dat het aantal testgevallen voorstelt. Per testgeval volgt hierop:

- Een eerste regel bevat twee door één spatie gescheiden gehele getallen N en M. N stelt het aantal nodes en M het aantal bogen voor van de stadsgrafe.
- Hierop volgen M regels met op elke regel drie door één spatie gescheiden gehele getallen x, y en d, met $0 \leq x < y < N$ en $0 \leq d$. Deze regel betekent "Je kan je verplaatsen tussen x en y in d tijdseenheden."
- Vervolgens is er een regel met een positief geheel getal K. Dit stelt het aantal bestellingen voor.
- Ten slotte volgen er K regels met op elke regel vier door één spatie gescheiden gehele getallen a, b, t en w, met $0 \leq a < N$, $0 \leq B < N$, $0 \leq t$ en $0 \leq w$. Deze regel betekent "Een klant heeft bij pizzeria op locatie a een bestelling geplaatst die moet geleverd worden op locatie b op tijdstip t. Je maakt er w winst op."

```
9
6 8
0 1 8
1 2 3
```

```
2 3 2
3 4 4
4 5 8
0 5 6
1 5 5
2 4 15
2
2 4 20 10
5 4 35 15
2 1
0 1 5
1
0 1 10 5
2 1
0 1 5
2
0 1 10 5
0 1 19 5
2 1
0 1 5
2
0 1 10 5
0 1 20 5
2 1
0 1 5
3
0 1 10 5
0 1 20 5
0 1 21 50
4 4
0 1 5
1 2 20
0 3 5
2 3 20
1
1 2 30 10
4 4
0 1 5
1 2 20
0 3 5
2 3 20
2
1 2 30 10
3 2 30 20
4 4
0 1 5
1 2 20
```

```
0 3 5
2 3 20
2
1 2 25 10
3 2 65 20
3 1
1 2 20
1
1 2 25 10
```

uitvoer Per testgeval voer je één regel uit met twee gehele getallen gescheiden door één spatie. Het eerste getal is de index van het testgeval. Het eerste testgeval heeft index 1. Het tweede getal stelt de maximale winst voor.

```
1 15
2 5
3 5
4 10
5 55
6 10
7 20
8 30
9 0
```

5 String en dynamisch programmeren

5.1 Dictee

Om het verbeteren van dictees minder tijdrovend en consistenter te maken moet er een programma geschreven worden dat automatisch het aantal foutpunten berekent. Het mappen van de ingediende zin met de correcte zin kan vaak op meerdere manieren gebeuren. Het programma moet steeds het beste resultaat voor de leerling (dus het minst aantal fouten) geven. Het aanrekenen van fouten wordt geïllustreerd aan de hand van voorbeelden. We starten met de correcte zin *kat*. Als er ook *kat* wordt ingediend dan is het aantal foutpunten 0 want dit is volledig correct. Was er echter em Kat ingediend, dan wordt er 1 foutpunt toegekend: het verwisselen van een kleine en een hoofdletter kost 1 punt. Dit geldt enkel als het over dezelfde letter gaat. Een L in de plaats van een k is een gewone verwisseling zoals het geval hierna. Een gewone verwisseling, bv *kot* geeft 2 foutpunten. Het toevoegen van een letter *kast* of het weglaten van een letter *at* heeft ook steeds 2 foutpunten als resultaat. De verwisseling a - o kan ook aanzien worden als het weglaten van de a en het toevoegen van de o maar dat kost veel meer en is dus niet in het voordeel van de leerling.

invoer De eerste lijn van de invoer bevat het aantal te verbeteren zinnen. Daarna volgen per geval telkens 2 lijnen. De eerste lijn bevat de ingediende zin en de tweede lijn bevat de correcte zin.

```
7
kat
kat
```

```
Kat
kat
kot
kat
kast
kat
at
kat
plaast
plaats
mogelike drangken zijn water cola fruitsap
Mogelijke dranken zijn: water, cola, fruitsap.
```

uitvoer Voor elk geval antwoord je met een enkele lijn. Deze bevat, gescheiden door 1 spatie, volgende informatie:

1. Het volgnummer van het geval. Dit begint bij 1 en wordt telkens verhoogt bij elk volgend geval.
2. het minimum aantal foutpunten voor dit geval

```
1 0
2 1
3 2
4 2
5 2
6 4
7 13
```

5.2 Delphi

Een beetje luguber, maar ook realistisch...

Je wil graag weten hoe lang je nog te leven hebt - in aantal jaren, nauwkeuriger hoeft niet. Het orakel van Delphi - niet perfect, maar op dit ogenblik het beste orakel [1] en bovendien op haar tournee door Europa één dag in pretpark Bilawi - kan in de toekomst kijken en weet wanneer je het tijdelijke voor het eeuwige verwisselt. Een nadeel: je kan aan het orakel enkel vragen stellen van de vorm: *overleef ik de eerstvolgende k jaren?* Daarop antwoordt het orakel natuurlijk enkel met een *ja* of een *nee*.

Er is van tevoren (willekeurig) bepaald hoeveel vragen je mag stellen waarop een *nee* als antwoord volgt: dat staat op je toegangsticket voor het pretpark. We stellen dat aantal verder voor door N (de N van Nee). Vragen met een *ja* als antwoord mag je zoveel stellen als je wil, maar die kosten je telkens een vaste bijdrage (1 Griekse euro) aan het pensioenfonds van het orakel: ook het orakel moet aan haar toekomst denken! Je wil de N gratis vragen zo efficiënt mogelijk gebruiken zodat je zo weinig mogelijk van je zuurverdiende euro's moet bijdragen aan de orakelkas. Gelukkig houdt je interesse verband met een investering die beperkt is in tijd: je bent enkel geïnteresseerd in de eerstvolgende I jaren. Je stelt dus nooit een vraag van de vorm *overleef ik de eerstvolgende k jaren?* met $k > I$.

[1] www.diedelphi.org

Wat is het minimaal aantal ja-vragen waarmee je zeker kan te weten komen wanneer (en of) je binnen de eerste I jaar de pijp aan Maarten geeft, gelijk wanneer dat juist is?

invoer De invoer bestaat uit een aantal gevallen. Elk geval wordt voorgesteld door 2 gehele getallen: de waardes van N en I. Die twee getallen zijn gescheiden door een blanco, en elk geval staat op een nieuwe lijn. N en I zijn ≥ 1.

```
5
12 10
3 9
5 5
9 18
11 12
```

uitvoer Voor elk geval geef je als uitvoer het aantal ja-vragen dat je moet stellen. Elk geval staat op een andere lijn die moet beginnen met het volgnummer van het geval: begin te tellen bij 1.

```
1 1
2 2
3 1
4 2
5 2
```

5.3 Samen reizen

Alice heeft haar reis uitgestippeld van Brussel naar Barcelona: ze gaat met de fiets heen, en met de trein terug. Ze is vooral geïnteresseerd in cultuur en heeft haar reis zo gepland dat ze in Mons ..., in Parijs, ... Bordeaux ... overnacht (in jeugdherbergen) en telkens tijd heeft om één en ander te bezoeken. Voor haar ligt de reis helemaal vast. Bob is haar vriend. Zijn reis ligt nog niet vast, maar hij heeft andere interesses dan Alice: hij wil een aantal voetbalstadiums bezoeken en die liggen niet altijd in de cultuursteden van Alice. Hij wil wel op dezelfde dag als Alice vertrekken en samen met haar op de trein van Barcelona terugkeren. Verder probeert hij zoveel mogelijk samen met Alice te reizen, al was het maar om uitvoerig verslag uit te brengen van zijn voetbalstadiumbelevenissen. Concreet betekent het dat hij probeert zoveel mogelijk nachten in dezelfde jeugdherberg als Alice door te brengen. Alice overnacht niet in Barcelona: ze neemt de avondtrein terug.

Je krijgt als gegevens

- startplaats (in tekst hiervoor Brussel)
- terugkeerplaats (Barcelona)
- de steden waarin Alice overnacht, in de volgorde waarin dat gebeurt
- de steden met een voetbalstadium die Bob wil bezoeken
- een wegennetwerk

Het wegennetwerk bestaat uit koppels van twee steden die op één dagreis (per fiets) van elkaar liggen: in die steden is een jeugdherberg, en je kan er overnachten. Een reis bestaat dus uit een rij van steden, maar het is mogelijk dat Alice of Bob één (of meer) rustdag(en) nemen en dus ter plaatse blijven. Bovendien mag een reis meerdere keren langs dezelfde stad passeren. Steden met een voetbalstadium hebben ook een jeugdherberg, zodat Bob niet onder de blote hemel moet slapen.

Gevraagd wordt hoeveel nachten Bob maximaal in dezelfde jeugdherberg kan slapen als Alice, met als beperking dat (1) hij al zijn voetbalstadiums bezocht moet hebben, (2) hij op tijd in de terugkeerplaats is om samen met Alice terug te keren. Dat aantal schrijf je uit. Als (1) en (2) niet samen mogelijk zijn, dan schrijf je `onmogelijk` uit.

Invoer De eerste regel van de invoer bevat een geheel getal $1 \leq n \leq 1000$ dat het aantal testgevallen aangeeft. Per geval volgen dan een aantal regels met informatie. Daarbij is het goed te weten dat steden voorgesteld worden met een geheel getal van 1 tot het aantal steden.

```
Regel 1: aantal steden
Regel 2: startplaats
Regel 3: terugkeerplaats
Regel 4: het aantal C overnachtingen van Alice in cultuursteden
Regel 5: C getallen die die cultuursteden voorstellen in de volgorde
         waarin ze bezocht worden door Alice
Regel 6: aantal V voetbaltempels die Bob wil bezoeken
Regel 7: V getallen die die voetbaltempels voorstellen
Regel 8: aantal L links in het wegennetwerk

Dan volgen L regels die telkens twee getallen bevatten: de twee
uiteinden van een verbinding tussen twee steden
```

Alle getallen in de invoer die op dezelfde regel voorkomen, worden gescheiden door 1 enkele spatie; alle regels worden beëindigd met een enkele newline `\n`.

Uitvoer De uitvoer bestaat uit n regels. Op een regel staat eerst het volgnummer van het testgeval (begin te tellen bij 1, oplopend tot n), daarna een spatie en dan een getal dat aangeeft hoeveel nachten Alice en Bob in dezelfde plaats overnachten (0 kan ook !) of `onmogelijk` als de hele onderneming niet kan.

Let op! Zorg ervoor dat je uitvoer geen overbodige tekens bevat, bijvoorbeeld een spatie op het einde van een regel of een lege regel op het einde van de uitvoer. Dat zorgt er immers voor dat je uitvoer als foutief wordt beschouwd.

Voorbeeld invoer

```
2
7
1
7
5
2 3 4 5 6
2
```

```
2 5
6
1 2
2 3
3 4
4 5
5 6
6 7
4
1
3
1
2
1
4
3
1 2
2 3
1 4
```

Voorbeeld uitvoer

```
1 5
2 onmogelijk
```