Practica Datastructuren en Algoritmen (2019-2020)

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 zaterdag 21 december 2019. De laatste feedback wordt op maandag 16 december 2019 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 Quilt

Een quilt wordt gemaakt door een aantal basispatronen - mogelijk gedraaid - aan elkaar te naaien. Om dat te mechaniseren hebben we een machine ontworpen die een aantal bevelen om een quilt te maken uitvoert en als resultaat een tekening van die quilt maakt. Jij gaat die machine implementeren.

Een basispatroon is altijd een vierkant gevormd door 4 ASCII-tekens. De enig mogelijke ASCII-tekens zijn + - / \ |

Naai-Draai Een basispatronen is bijvoorbeeld:

```
+/
-\
```

Na een draai (90° in wijzerzin) krijg je

```
|+
/\
```

Dat zie je gemakkelijk door je hoofd schuin naar links te houden en te kijken naar het oorspronkelijke patroon.

Twee patronen worden aan elkaar genaaid als volgt:

```
+/    |+              +/|+
-\    /\              -\/\
```

Instructies Basispatronen worden geïdentificeerd door een getal (het volgnummer waarin het in de input voorkomt - begin te tellen bij 1). De naai-draai bevelen volgen het regime van een stapelmachine (initieel is de stapel leeg):

i : plaatst een lap met het i-de basispatroon bovenop de stapel,

`naai` : haalt de twee bovenste lappen van de stapel, naait ze aan elkaar (de bovenlapste lap links van de andere) en plaatst het resultaat bovenaan de stapel, en

`draai` : draait de lap bovenaan de stapel in wijzerzin.

Er zijn nog twee bevelen die de stapel niet veranderen:

`teken` : doet de huidige top verschijnen in de output en laat dan een regel open, en

`stop` : beëindigt het programma.

We garanderen dat alle instructies in de invoer uitgevoerd kunnen worden.

Invoer De eerste regel bevat het aantal opgaven t ($1 \leq t \leq 100$).

Een opgave bestaat uit $1 + 2 \times p + 1 + b$ regels. De eerste regel bevat het getal p ($1 \leq p \leq 10$), het aantal patronen. Daarna volgen de p patronen: elk patroon bestaat uit 2 maal 2 tekens. De gegeven patronen krijgen een nummer van 1 tot p. Daarna komt het getal b ($1 \leq b \leq 100$), het aantal bevelen, gevolgd door de b bevelen. Het laatste bevel is altijd `stop`.

Uitvoer Elk teken-bevel tekent de huidige gemaakte quilt, gevolgd door een lege regel.
Let op! Zorg ervoor dat je uitvoer geen overbodige tekens bevat, bijvoorbeeld een spatie op het eind van een regel of een lege regel op het eind van de uitvoer. Dat zorgt er immers voor dat je uitvoer als foutief wordt beschouwd.

Voorbeeld Invoer

De commentaar na % hoort niet bij de invoer.

```
1               % aantal opgaven
2               % aantal basispatronen
++              % eerste basispatroon
--
||              % tweede basispatroon
//
9               % aantal bevelen
2               % zet basispatroon twee op de stapel
draai
teken
1
naai
```

```
teken
draai
teken
stop
```

Uitvoer

```
\-
\-

++\-
--\-

|+
|+
//
||
```

2.3 Vals geld

In de eerste week van je nieuwe job bij de douane maak je deel uit van het team dat een groot transport van geldstukken onderschept heeft, waarvan geweten is dat er een belangrijk deel valse geldstukken tussen zitten. De geldstukken zijn verpakt per 26, en via een betrouwbare anonieme tip heeft jouw team kunnen achterhalen dat er telkens exact één vals geldstuk tussen zit.

De valsmunters zijn nauwkeurig te werk gegaan: de valse stukken kunnen niet op basis van grootte, kleur of reliëf onderscheiden worden van de echte stukken. Wel is er een klein verschil in gewicht.

Met behulp van een zeer nauwkeurige balansweegschaal is het jullie taak om de valse geldstukken van de echte te onderscheiden.

Schrijf een programma dat, aan de hand van de details van een aantal wegingen voor een pakketje van 26 geldstukken het valse geldstuk probeert te identificeren. De geldstukken worden geïdentificeerd aan de hand van de letters `a-z`. De invoer bestaat uit de details voor een aantal weegsessies. De eerste invoerlijn bevat één getal n, het aantal pakketten van 26 geldstukken waarvoor het valse geldstuk moet bepaald worden. Daarna volgt per pakket eerst een getal dat het aantal wegingen m_p voor het pakket p bepaalt, gevolgd door m_p lijnen die elk een weging van dat pakket voorstellen.

Elk van deze wegingen wordt voorgesteld door twee lettercodes, bestaande uit de letters `a-z`, gevolgd door een woord (`evenwicht`, `omhoog` of `omlaag`), telkens gescheiden door één spatie. De eerste lettercode geeft aan welke geldstukken er in de linker schaal van de balans gelegd worden, de tweede doet hetzelfde voor de rechterschaal. Het woord geeft aan of de *rechter*schaal van de balans naar omhoog gaat, naar omlaag gaat, of in evenwicht is met de linkerschaal. Je mag er hierbij van uitgaan dat er steeds evenveel munten links in de schaal liggen als rechts, dat er altijd minstens één munt in elke schaal ligt, en dat elke letter maximum één keer voorkomt in een weging.

De uitvoer bestaat uit n lijnen, één lijn per pakket van geldstukken. Als het valse geldstuk kon geïdentificeerd worden, dan wordt de boodschap

`Het valse geldstuk X is lichter.`

of

`Het valse geldstuk X is zwaarder.`

uitgeprint. Uiteraard wordt hierbij `X` vervangen door de juiste letter (`a-z`). Als het valse geldstuk niet eenduidig kan geïdentificeerd worden, dan wordt de boodschap

`Te weinig gegevens.`

uitgeprint. Tenslotte is het ook mogelijk dat de politieman de resultaten fout genoteerd heeft. Als de wegingen elkaar tegenspreken moet de boodschap

`Inconsistente gegevens.`

worden uitgeprint.

Zorg dat je programma zeker werkt voor onderstaande invoer (maar ook voor andere invoer).

```
6
2
abc efg evenwicht
a e omhoog
4
abcd efgh evenwicht
abci efjk omhoog
abij efgl evenwicht
mnopqrs tuvwxyz evenwicht
2
ab st omhoog
ef ab omlaag
3
cdefghijklmn opqrstuvwxyz evenwicht
a b omhoog
b c evenwicht
1
a b evenwicht
2
acx bdy omhoog
pst aeq omhoog
```

Voor bovenstaande invoer moet volgende uitvoer gegenereerd worden:

```
Inconsistente gegevens.
Het valse geldstuk k is lichter.
Te weinig gegevens.
Het valse geldstuk a is zwaarder.
Te weinig gegevens.
Inconsistente gegevens.
```

3 Recursie en backtracking

3.1 Foodfest

De laatste jaren zijn food trucks erg populair geworden: het zijn combi's of aanhangwagens waarin gerechten gemaakt worden, en die worden dan aan een kraam verkocht. Er zijn dan ook regelmatig food truck festivals, waar een groot aantal food trucks op n plaats gestationeerd zijn. Elke food truck

biedt meerdere gerechten aan, aan diverse, soms heel lage prijzen. Een bezoeker kan dan kiezen uit een groot aanbod en zo snel en goedkoop nieuwe gerechtjes ontdekken.

Aankopen in het food truck festival kan je alleen doen met bonnetjes die je op voorhand koopt. Een bonnetje kost 1 euro. De bonnetjes die je op het einde van het festival overhebt kan je niet inwisselen. Dat wil je dus vermijden. In deze opgave zijn we dus geïnteresseerd in budgetten die je helemaal kan uitgeven op het food truck festival. Een bijkomende vereiste is dat je exact 1 gerecht koopt bij elke food truck op het festival.

Invoer Alle getallen in de invoer die op dezelfde regel voorkomen, worden telkens gescheiden door één enkele spatie; alle regels worden beëindigd met één enkele newline.

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.

Elk geval bestaat uit een aantal regels met informatie. De eerste regel begint met een getal $2 \leq b \leq 5$ dat het aantal budgetten aangeeft. Op deze regel staan verder — gescheiden door spaties — b gehele getallen die budgetten voorstellen (budgetten gaan van 5 euro tot en met 100 euro): de budgetten staan in stijgende volgorde. De tweede regel bestaat uit n getal $2 \leq f \leq 10$ dat het aantal food trucks aangeeft. Deze wordt gevolgd door f regels die de prijzen van gerechten per food truck aangeven. Elke regel begint met een getal $1 \leq r \leq 10$, gevolgd door r prijzen, gescheiden door spaties (prijzen zijn gehele getallen, ze gaan van 1 euro tot en met 20 euro). Ook de prijzen staan in stijgende volgorde.

Een voorbeeld van mogelijke invoer is

```
2
4 5 10 14 20
2
5 1 2 3 4 5
5 6 7 8 9 10
2 20 30
2
2 8 9
3 5 6 7
```

Uitvoer De uitvoer bestaat uit n regels die voor elk geval aangeven welke budgetten aanleiding geven tot een combinatie van gerechten waar je geen bonnetjes over hebt. Elke regel begint met een volgnummer (dat begint bij 1 en verhoogt bij elk volgend geval), en dan de budgetten in stijgende volgorde, gescheiden door spaties. Indien je geen enkel van de gegeven budget exact kan uitgeven bestaat de regel enkel uit het volgnummer en de tekenreeks `GEEN`, gescheiden door een spatie.
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.

Voor de voorgaande invoer moet volgende uitvoer bekomen worden.

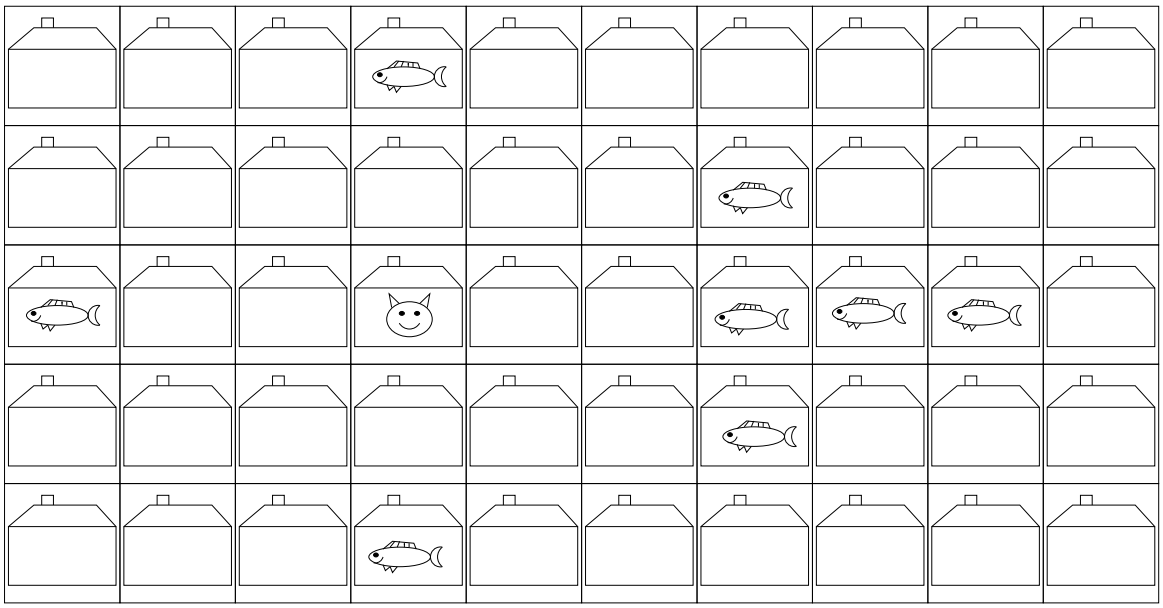
```
1 10 14
2 GEEN
```

3.2 Garfield

Garfield, de kat van Jon, is de laatste tijd nogal dik geworden van veel te eten en weinig te bewegen. Daarom heeft Jon Garfield op een strikt dieet gezet en het eten veilig opgeborgen. Garfield is ontevreden met de porties die hij krijgt maar kan thuis niets te knabbelen vinden. De huizen in de omgeving daarentegen zetten zorgvuldig eten klaar voor hun huisdieren. Garfield smeedt een snood plan...

Jon moet regelmatig het huis uit om inkopen te gaan doen, wat voor Garfield het perfecte moment is om uitstapjes te maken en het eten van andere huisdieren in de buurt te gaan verorberen. Het is natuurlijk belangrijk dat Garfield op tijd terug is, zodat Jon geen argwaan krijgt.

Tijdens zijn vele dutjes kan Garfield een overzicht maken van de buurt, met als doel om de route te kiezen waar hij bij de meeste huizen kan eten. De huizen liggen in een rechthoekig rooster, zoals op de tekening: sommige huizen bevatten eten (de vis) en je kan zien waar Garfield woont. In de andere huizen is geen voedsel te vinden. Omdat Garfield echt wel te dik is doet hij er telkens één minuut over om van huis naar huis te stappen (horizontaal of verticaal, niet diagonaal). Als er eten aanwezig is in een huis duurt het n minuut om alles op te eten. Het spreekt voor zich dat Garfield bij elk huis hoogstens één keer kan eten. Garfield kan zoveel eten als hij wil (en dat is ook het doel), maar hij moet terug thuis zijn voor Jon merkt dat zijn kat op strooptocht is. Concreet wil dit zeggen dat, als Jon op tijdstip 0 vertrekt, en T minuten weg is, Garfield ten laatste op tijdstip T terug thuis moet zijn. Als Garfield een huis passeert met eten, is het niet noodzakelijk dat hij dit opeet.



Invoer 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`.

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.

Elk geval bestaat uit een aantal regels met informatie. De eerste regel bevat drie getallen, gescheiden door een spatie. De eerste twee getallen $10 \leq b, h \leq 20$ geven de breedte b en hoogte h van de (rechthoekige) buurt aan. Het laatste getal geeft het aantal minuten $10 \leq t \leq 100$ aan voor Jon terug is.

Deze regel wordt gevolgd door h regels met elk b karakters, die de buurt voorstellen. Er is exact één veldje met een `G`, dit is de vertrek- en eindpositie van Garfield. Alle andere veldjes zijn ofwel

huizen met eten (`E`) of huizen zonder eten (`.`).

Uitvoer De uitvoer bestaat uit n regels. Elke regel bestaat uit het volgnummer van het geval, gevolgd door een spatie en een getal dat de maximale hoeveelheid eten voorstelt dat Garfield in de gegeven tijd kan opeten. Volgnummers beginnen bij 1 en verhogen voor elk geval met 1.
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
10 5 13
...E......
......E...
E..G..EEE.
......E...
...E......
9 2 18
EEE.G.EEE
...EEE...
```

De invoergegevens van het eerste geval, horen bij de figuur.
Uitvoer

```
1 3
2 6
```

3.3 Dwergen

Na een lange dag in de mijnen van Pirolog gaan dwergen graag uit eten bij de lokale kobold. Kobolds staan ervoor gekend dat ze nooit wisselgeld in de kassa hebben.

Alle betalingen door dwergen zijn gebaseerd op de uitwisseling van diamanten. De waarde van een diamant wordt uitgedrukt in karaat. Het grootste probleem bij betalingen is dat dwergen gehecht zijn aan hun diamanten. Diamanten zijn letterlijk een dwerg zijn beste vriend. Vreemd genoeg heeft de waarde in karaat geen invloed op de band tussen een dwerg en zijn diamant.

Wanneer dwergen samen uit gaan eten wordt het als beleefd beschouwd om de rekening te verdelen zodat elke dwerg evenveel diamanten uitgeeft. Bijvoorbeeld, een rekening van 100 karaat kan verdeeld worden onder drie dwergen zodanig dat de eerste dwerg één diamant van 70 karaat en één diamant van 1 karaat betaalt, de tweede dwerg 12 en 4 karaat en de derde dwerg 3 en 10 karaat. Dit is een beleefde transactie aangezien elke dwerg 2 diamanten betaalt. Merk op dat de rekening steeds exact betaald moet worden.

Invoer: De eerste regel van de invoer bevat het aantal testgevallen. Per testgeval volgen telkens de onderstaande regels.

- Eén regel bevat de waarde in karaat W van de rekening. W is een strikt positief geheel getal, kleiner dan 1000.

- Eén regel bevat het aantal dwergen D betrokken in de betaling. D is een strikt positief geheel getal, kleiner dan 20.
- Vervolgens komen er D keer 2 regels. De eerste van deze regels bestaat uit A, het aantal diamanten van een dwerg. A is steeds een geheel getal groter dan 1 en kleiner dan 20. De tweede regel bestaat uit A keer de waarde van een diamant in karaat. Deze waarden zijn ook gehele strikt positieve getallen. De opgesomde waarden worden gescheiden door spaties.

voorbeeldinvoer

```
3
100
3
5
5 40 5 10 110
6
10 30 10 5 5 20
3
5 5 12
100
2
6
10 10 10 10 20 10
6
10 10 10 10 10 10
121
2
3
1 1 111
3
2 121 1
```

uitvoer:

Per testgeval bestaat de uitvoer uit het volgnummer van het testgeval en het minimum aantal diamanten dat per dwerg betaald moet worden om een beleefde transactie te maken. Indien het niet mogelijk is om tot een beleefde transactie te komen dan wordt het volgnummer gevolgd door `ONMOGELIJK`. Het volgnummer en het resultaat worden steeds gescheiden door één spatie.

voorbeelduitvoer

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

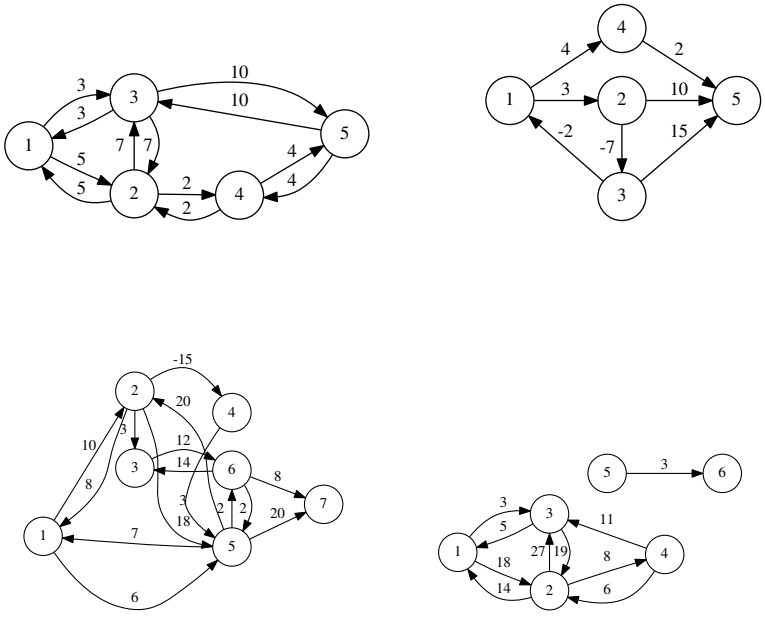
4 Grafen

4.1 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
```

4.2 Mijnenveld

Je bent de bevelvoerder van een oorlogspatrouille, waarmee je een mijnenveld moet oversteken. Dankzij de inlichtingendienst, die erin geslaagd is om het computernetwerk van de vijand te hacken, beschik je echter over een kaart van het gebied, die zeer nauwkeurig aangeeft hoeveel mijnen er in elke zone van het veld werden ingegraven. Om de verliezen in je patrouille zo laag mogelijk te houden, moet je de veiligste weg kiezen door het mijnenveld. Dit is de weg waarlangs zo weinig mogelijk mijnen liggen. Zijn er meerdere zulke wegen, dan moet je een van de kortste ervan nemen, om je manschappen niet teveel te vermoeien. De lengte van een weg is het aantal zones dat door die weg doorlopen wordt.

Het mijnenveld heeft de vorm van een rechthoek, en is opgesplitst in vierkantige zones, die elk even groot zijn. De kaart, waarover je beschikt, bestaat dus uit een rooster met r rijen en k kolommen. Voor elke zone in dat rooster geeft de kaart aan hoeveel mijnen in die zone liggen. Het is de bedoeling dat je een weg vindt die het mijnenveld van links naar rechts doorkruist (je weg bevat dus minstens één zone uit elke kolom). Deze weg kan bestaan uit stappen naar rechts, naar boven, naar onder, en naar links (zig-zag wegen zijn dus toegelaten). Een weg door het mijnenveld bestaat uit een reeks naburige zones, dat zijn zones die een gemeenschappelijke rand hebben (op de kaart liggen ze dus ofwel naast elkaar, ofwel onder elkaar).

Invoer:

De eerste regel van de invoer bestaat uit het aantal te verwerken mijnenvelden $1 \leq N \leq 1000$. Voor elk mijnenveld volgen daarna:

- een regel met twee gehele getallen r en k: het aantal rijen en het aantal kolommen (gescheiden door één spatie). $(1 \leq r, k \leq 100)$.
- gevolgd door r regels met elk k gehele getallen: het aantal mijnen in de overeenkomstige zone $(0 \leq \text{aantalMijnen} \leq 5000)$. Deze getallen worden telkens gescheiden door één spatie.

Een voorbeeld van invoer wordt hieronder gegeven:

```
3
4 3
0 8 6
2 6 5
3 9 4
4 5 10
```

```
3 3
1 1 3
4 2 1
2 1 2
6 6
1 1 1 1 7 8
3 7 4 1 8 4
8 1 1 1 9 2
10 1 7 9 10 8
12 1 1 1 8 12
4 8 3 1 1 1
```

Uitvoer:

Voor elk mijnenveld uit de invoer wordt een regel uitgeprint: die regel bevat drie gehele getallen (telkens gescheiden door één spatie):

- het nummer van het mijnenveld (van 1 tot N),
- de lengte (= het aantal zones dat doorlopen wordt) van een kortste veiligste pad, dat het mijnenveld van links naar rechts doorloopt,
- het totaal aantal mijnen langs dit pad;

Voor bovenstaande invoer moet dus de volgende uitvoer komen:

```
1 3 13
2 3 5
3 15 15
```

4.3 Nightcrawler

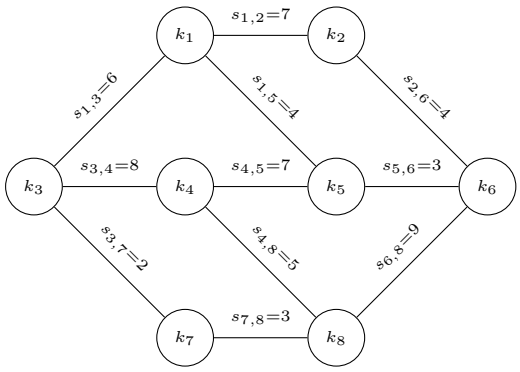
Jake de opportunistische Nightcrawler rijdt 's nachts rond in de stad. De politie afluisterend wacht Jake tot er een interessante gebeurtenis plaatsvindt, zoals een inbraak of een schietpartij. Van zodra dit wordt gesignaleerd snelt hij er naartoe om er videobeelden van op te nemen en deze te verkopen aan de meest biedende nieuwszender.

Jake merkt echter op dat hij steeds te laat aankomt. Hij besluit daarom een aantal mensen aan te nemen die strategisch gepositioneerd worden op wachtposten in de stad. Er wordt afgesproken dat bij elke melding van een interessante gebeurtenis de dichtstbijzijnde persoon er naartoe moet.

De vraag is nu, hoeveel leden moet het Nightcrawler team tellen opdat elk deel van de stad binnen een bepaalde tijdspanne kan bereikt worden?

Stad Een stad (K, S) wordt als volgt gedefinieerd:

- De stad wordt voorgesteld als een geheel van N_k kruispunten $K = \{k_1, k_2, \ldots, k_{N_\mathrm{k}}\}$ verbonden door N_s straten $S = \{s_{i,j}\}$, waarbij $s_{i,j}$ de lengte van de straat tussen kruispunten k_i en k_j voorstelt.
- Overal is tweerichtingsverkeer toegelaten, m.a.w. $s_{i,j} = s_{j,i}$.
- Tussen twee kruispunten is er maximaal één straat.



De kortste afstand tussen kruispunten k_i en k_j noteren we $d_{i,j}$.

Wachtpost Voor een gegeven stad willen we nu te weten komen op welke kruispunten men de Nightcrawlerleden moet positioneren opdat elke locatie in de stad maximaal een afstand M verwijderd is. We noemen een kruispunt waar een Nightcrawlerlid gestationeerd is een *wachtpost*.

M-Wachtpostselectie Voor een M-wachtpostselectie $\Sigma \subseteq K$ geldt dat elk kruispunt $k_i \in K$ maximaal een afstand M verwijderd is van de dichtstbijzijnde wachtpost k_j in Σ. Wiskundig uitgedrukt:

$$\forall\, k_i \in K \;\bullet\; \exists\, k_j \in \Sigma \;\bullet\; d_{i,j} \leq M$$

Bijvoorbeeld, elk kruispunt gebruiken als wachtpost ($\Sigma = K$) is altijd een M-wachtpostselectie omdat de afstand tot elk kruispunt 0 bedraagt.

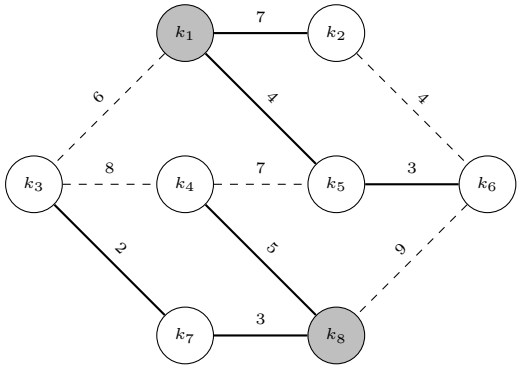
Metrieken We definiëren twee metrieken op M-wachtpostselecties.

- Het aantal wachtposten in de selectie Σ noemen we de *omvang*, genoteerd $\#\Sigma$.
- De som van alle minimale afstanden van elk kruispunt tot een wachtpost noemen we de *totale afstand*, genoteerd $\mathrm{D}(\Sigma)$.
$$\mathrm{D}(\Sigma) = \sum_{k_i \in K} \min_{k_j \in \Sigma}(d_{i,j})$$

Optimale M-wachtpostselectie Een *optimale M-wachtpostselectie* is een M-wachtpostselectie waarbij zowel de omvang als totale afstand geminimaliseerd zijn. M.a.w. om een optimale M-wachtpostselectie te vinden ga je als volgt te werk:

- Je zoekt alle M-wachtpostselecties.
- Hieruit selecteer je eerst die met de kleinste omvang.
- Hieruit selecteer je die met de kleinste totale afstand.

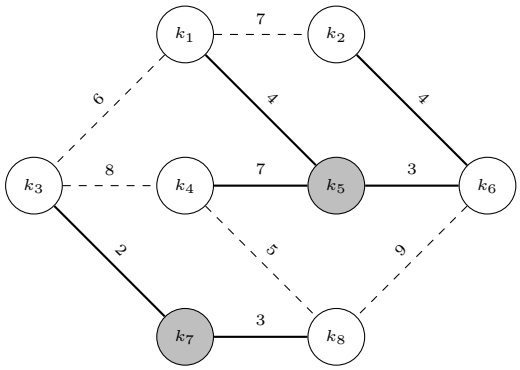
Voorbeeld We hernemen de voorbeeldstad waarvoor we optimale 10-wachtpostselecties wensen te vinden. Voorbeelden van 10-wachtpostselecties zijn $\{k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8\}$ en $\{k_1, k_4, k_7\}$. Deze hebben omvang 8 en 3, respectievelijk. We kunnen echter ook 10-wachtpostselecties vinden met omvang 2:



We rekenen per kruispunt de afstand tot de dichtstbijzijnde wachtpost uit.

$k_i \in K$	$k_j \in \Sigma$	pad	$d(i,j)$
k_1	k_1	$[k_1]$	0
k_2	k_1	$[k_2, k_1]$	7
k_3	k_8	$[k_3, k_7, k_8]$	$2 + 3 = 5$
k_4	k_8	$[k_4, k_8]$	5
k_5	k_1	$[k_5, k_1]$	4
k_6	k_1	$[k_6, k_5, k_1]$	$4 + 3 = 7$
k_7	k_8	$[k_7, k_8]$	3
k_8	k_8	$[k_8]$	0
		Totale afstand:	31

Er is echter een 10-wachtpostselectie met een kleinere totale afstand:



Dit geeft voor de totale afstand:

$k_i \in K$	$k_j \in \Sigma$	pad	$d(i,j)$
k_1	k_5	$[k_1, k_5]$	4
k_2	k_5	$[k_2, k_6, k_5]$	$4 + 3 = 7$
k_3	k_7	$[k_3, k_7]$	2
k_4	k_5	$[k_4, k_5]$	7
k_5	k_5	$[k_5]$	0
k_6	k_5	$[k_6, k_5]$	3
k_7	k_7	$[k_7]$	0
k_8	k_7	$[k_8, k_7]$	3
		Totale afstand:	26

Let op: voor andere steden is het mogelijk dat er meerdere optimale M-wachtpostselecties bestaan. Voor deze opgave moet je ze allemaal vinden.

Invoer:

De eerste regel bevat het aantal testgevallen. Per testgeval volgt

- Een regel met drie door één spatie gescheiden gehele getallen M, N_k en N_s met $0 \leq M$, $0 < N_\mathrm{k}$, $0 \leq N_\mathrm{s}$. Deze getallen stellen respectievelijk de maximum afstand, het aantal kruispunten en het aantal straten voor.
- N_s regels met telkens drie door één spatie gescheiden gehele getallen i, j, $s_{i,j}$ met $1 \leq i < j \leq N_\mathrm{k}$ en $0 \leq s_{i,j}$. Elk van deze regels stelt een straat voor gaande van k_i naar k_j met lengte $s_{i,j}$.

Voorbeeldinvoer:

```
9
10 8 11
1 2 7
1 3 6
1 5 4
2 6 4
3 4 8
3 7 2
4 5 7
4 8 5
5 6 3
6 8 9
7 8 3
10 2 1
1 2 2
1 2 1
1 2 2
100 3 0
100 3 3
1 2 1
1 3 1
```

```
2 3 1
100 4 2
1 2 1
3 4 1
10 5 4
1 2 5
2 3 5
3 4 5
4 5 5
10 10 9
1 2 5
2 3 5
3 4 5
4 5 5
5 6 5
6 7 5
7 8 5
8 9 5
9 10 5
1000 20 20
1 2 1
1 20 1
2 3 1
3 4 1
4 5 1
5 6 1
6 7 1
7 8 1
8 9 1
9 10 1
10 11 1
11 12 1
12 13 1
13 14 1
14 15 1
15 16 1
16 17 1
17 18 1
18 19 1
19 20 1
```

Uitvoer:

Elk testgeval heeft een eigen index, tellend vanaf 1. Per testgeval worden volgende regels uitgevoerd:

- Stel dat het testgeval K optimale M-wachtpostselecties telt. Er dienen dan K regels afgedrukt worden. Elke regel komt overeen met een optimale M-wachtpostselectie.

- Elke optimale selectie wordt omgezet naar een string als volgt: sorteer alle kruispuntindices van klein naar groot en voeg ze samen tot een enkele string gebruik makende van een enkele spatie als separator. Bijvoorbeeld, de optimale M-wachtpostselectie $\{k_4, k_7, k_9\}$ wordt voorgesteld door `4 7 9`.
- Deze strings moeten alfabetisch gesorteerd worden.
- Elk van deze strings wordt geprefixt door de index van het testgeval gevolgd door een spatie.

Bv. indien testgeval 54 als optimale selecties $\{k_4, k_5, k_8\}$ en $\{k_{21}, k_{35}, k_{36}\}$ heeft, dan worden voor dit testgeval de regels `54 21 35 36` en `54 4 5 8` afgeprint. Let op de alfabetische volgorde: 21 komt vóór 4!

Voorbeelduitvoer:

```
1 5 7
2 1
2 2
3 1 2
4 1 2 3
5 1
5 2
5 3
6 1 3
6 1 4
6 2 3
6 2 4
7 3
8 3 8
9 1
9 10
9 11
9 12
9 13
9 14
9 15
9 16
9 17
9 18
9 19
9 2
9 20
9 3
9 4
9 5
9 6
9 7
9 8
```

5 Dynamisch programmeren

5.1 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
```