Practica Datastructuren en Algoritmen (2021-2022)

dr. ir. Annemie Vorstermans

1 Inleiding

De code voor onderstaande oefeningen wordt ingediend via Toledo. Feedback over de ingediende oefeningen gaat ook via toledo en zal minstens 1 maal per week gegeven worden (tenzij overmacht). De laatste feedback wordt op maandag 13 december 2021 gegeven. De uiterste datum (harde deadline) om de oefeningen in te dienen is maandag 27 december 2021. Oefeningen die daarna worden ingediend worden niet meer beschouwd voor EP1 tenzij er een grondige reden is. Het niet-maken van de oefeningen leidt tot een NA voor deze opleidingsactiviteit. De bovenstaande datum is de uiterste indiendatum. Het is sterk aangeraden om zeker de richtdata (soft deadlines) hieronder te respecteren. Maak de oefeningen zo snel mogelijk zodat je niet in tijdsnood komt.

- oefeningen reeks 1: ten laatste zondag 17 oktober 2021
- oefeningen reeks 2: ten laatste zondag 7 november 2021
- oefeningen reeks 3: ten laatste zondag 28 november 2021
- oefeningen reeks 4: ten laatste zondag 19 december 2021

Er zijn 3 online labozittingen (via Toledo Collaborate) gepland in de lessenroosters, maar wacht niet tot dan als je een probleem hebt met een oefening. Contacteer me via mail of tijdens de les om een afspraak (fysiek of online) te maken voor meer uitleg of zend een mail met een duidelijke beschrijving van het probleem en jouw Java-code. Mijn emailadres is Annemie.Vorstermans@kuleuven.be

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 maximaal 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 punten die je via toledo krijgt houden enkel rekening met de correctheid en de snelheid. Voor de berekening van de eindscore wordt ook rekening gehouden met het aantal ingediende pogingen en de leesbaarheid van de code.

De oefeningen worden individueel gemaakt! Gekopiëerde code (= fraude) zorgt voor een 0 op de opleidingsactiviteit, ook voor de bron.

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 Wachten

Je wil tickets kopen voor je favoriete muziekfestival. Je staat in de rij om tickets te kopen maar deze rij wordt op een speciale manier verwerkt. Per minuut wordt er precies één ticket verkocht aan de persoon die vooraan in de rij staat. Als deze persoon meer dan 1 ticket nodig had sluit deze opnieuw achteraan aan. In het andere geval verlaat deze persoon natuurlijk de rij.

Je bent ergens midden in de rij terechtgekomen en wil graag berekenen hoe lang het nog duurt voor je jouw gewenste aantal tickets hebt gekocht.

Stel bv. dat de rij er als volgt uitziet:

$$3, 1, \mathbf{2}, 3$$

Jij bent de derde persoon in de rij, en zoals je ziet wil je twee tickets kopen. De twee personen voor je willen respectievelijk 3 en 1 tickets kopen. De persoon achter je wil er drie kopen.

De rij wordt als volgt verwerkt:

aantal minuten	**rij**
0	3, 1, **2**, 3
1	1, **2**, 3, 2
2	**2**, 3, 2
3	3, 2, **1**
4	2, **1**, 2
5	**1**, 2, 1
6	2, 1

Na 6 minuten heb je al je tickets gekocht en heb je de rij verlaten.

invoer De eerste regel bevat het aantal testgevallen. Per testgeval volgen er twee regels. De eerste regel bevat twee getallen a en p die van elkaar gescheiden worden door een spatie. Het getal a stelt het aantal personen in de rij voor terwijl p jouw initiële positie voorstelt (tellend vanaf 1). Dan volgt een tweede regel met a positieve gehele getallen die van elkaar gescheiden worden door een enkele spatie. Deze getallen geven aan hoeveel tickets elke persoon in de rij wil kopen.

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

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

uitvoer Per testgeval druk je één regel af. Deze regel bevat twee getallen gescheiden door één spatie:

- Het eerste getal stelt de index van het testgeval voor. Het eerste testgeval heeft index 1.
- Het tweede getal is het aantal minuten dat het duurt vooraleer je al je gewenste tickets hebt gekocht.

```
1 24
2 129
3 49
4 30
5 36
6 22
7 18
8 4
9 29
10 82
```

2.3 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
```

3 Recursie en backtracking

3.1 Naomees

In de Grote Oceaan ligt ten zuiden van Kiribati, ten noorden van Fiji en ten oosten van de Salomonseilanden een miniscuul eilandje met de naam Naom. De inwoners van Naom spreken een wel heel bijzondere taal. Na jarenlang onderzoek stelden taalwetenschappers vast dat de woordenschat van het Naomees op de volgende manier is opgebouwd.

1. De basiswoorden zijn `ba`, `di` en `du`.
2. Door verdubbeling van een woord ontstaat een nieuw woord, bijvoorbeeld `baba`.
3. Een nieuw woord kan ook gevormd worden door een bestaand woord voor en na een ander bestaand woord te plaatsen, bijvoorbeeld `baduba`.

Voor deze opgave moet je bepalen of een gegeven woord een geldig Naomees woord voorstelt.

Invoer De eerste regel bevat het aantal testgevallen t. Daarna volgt de beschrijving van t testgevallen. Elk tekstgeval wordt omschreven door vijf regels tekst: van elk van die vijf regels bepaal je of ze Naomees of onzin zijn.

```
1
dudubabadudubabadudu
didudubadududi
dudubadibadibadu
dididudidibadibadididudidi
babadidudidibadi
```

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

- de index van het testgeval, startend vanaf 1;
- één enkele spatie;
- voor elk van de vijf woorden de tekst `naomees` of `onzin` al naargelang het overeenkomstige woord een geldig Naomees woord voorstelt of niet: die stukjes tekst worden gescheiden door één enkele spatie

```
1 naomees naomees onzin naomees onzin
```

3.2 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 één 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 één 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
```

3.3 Tokio

Net zoals in vele steden met een metro moet je in Tokio je kaart scannen bij het binnenkomen en bij het buitengaan: het bedrag dat van je prepaid kaart afgaat hangt enkel af van je start- en eindstation, niet van welke stations je onderweg nog aandeed. Je mag inderdaad gelijk welke route volgen, bijvoorbeeld om je favoriete sushi-stand te bezoeken, zolang je het station maar niet verlaat. De prepaid kaarten in Tokio zijn naamloos, en, echt op zijn Belgisch, kwamen we direct op het idee voor een app die een groep mensen in staat stelt wat minder te betalen, namelijk door onderweg kaarten te wisselen.

Hier is een extreem geval: Kenji moet 's morgens van station Shibuya naar station Asakusa; dat kost 4 eenheden. Terzelfdertijd moet Ronin van Asakusa naar Shibuya, ook 4 eenheden. Samen betalen ze 8 eenheden. Maar als ze mekaar onderweg ontmoeten, bijvoorbeeld in Shimbashi, en daar dan hun kaarten wisselen, dan komt elke kaart aan waar die vertrok, en is de totale kost 0, of de totale winst is 8. 's Avonds reizen ze omgekeerd, wisselen weer van kaart en iedereen is gelukkig, behalve de uitbaters van de metro.

De app die jullie gaan ontwikkelen heeft een inschrijfmodule voor een bepaald tijdsslot: daarin geven een aantal mensen hun begin- en eindstation op. De app berekent vervolgens hoe de kaarten moeten gewisseld worden en toont de totale winst. Uiteraard mag het niet zo zijn dat sommige individuen verlies lijden: zie daarvoor de invoer later.

Stations worden gegeven als een positief getal, van 1 tot N, het aantal stations. De prijs tussen twee stations komt in de vorm van een matrix, geïndexeerd door de stations. Deze matrix krijg je helemaal: hij is telkens symmetrisch met nullen op de diagonaal en elders enkel strikt positieve gehele getallen.

Als uitvoer verwachten we de maximale totale winst die geboekt kan worden door kaarten te wisselen zonder dat er iemand verlies lijdt.

Invoer De eerste regel stelt het aantal testgevallen voor. Per testgeval volgt

- een regel met het aantal stations N; $N > 1$
- de kostenmatrix: N regels met telkens N positieve getallen gescheiden door een blanco; enkel op de diagonaal staan nullen
- een regel met het aantal personen P dat inschreef op de app: die personen zijn genummerd van 1 tot P; $P > 0$

- dan komt één regel met de P vertrekstations van de P personen, gescheiden door een blanco
- dan komt één regel met de P eindstations gescheiden door een blanco

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

Uitvoer Als uitvoer verwachten we per testgeval één regel met daarop het volgnummer van het testgeval, een blanco, en dan de maximale totale winst die kan gemaakt worden zonder dat iemand verlies lijdt.

```
1 8
2 0
```

In eerste testgeval kunnen persoon 1 en 3 hun kaart wisselen en elk 4 eenheden winst maken: persoon 2 wisselt niet, want daardoor wordt het niet globaal beter.

In het tweede testgeval is de totale kost zonder verwisselen van kaarten 8. Door de kaarten te wisselen wordt de totale kost 6 en dat is beter, maar de kaart van de eerste persoon wordt in dit scenario met 6 gedebiteerd i.p.v. 4 en dat wil die natuurlijk niet. Hier kan geen winst gemaakt worden.

4 Grafen

4.1 Arctische verbindingen

Het Amerikaanse Department of National Defense (DND) is van plan om verschillende onderzoekscentra op Antarctica met elkaar te verbinden door middel van een draadloos netwerk. Hiertoe worden twee verschillende technologiëen gebruikt: elk onderzoekscentrum zal beschikken over een radio-ontvanger, en sommige onderzoekscentra zullen daarnaast ook beschikken over een satelietverbinding.

Elk onderzoekscentrum dat beschikt over een dergelijke satelietverbinding kan communiceren via de sateliet, onafhankelijk van zijn positie. Communicatie via de radioverbinding tussen twee

onderzoekscentra is echter enkel mogelijk indien de afstand tussen beide onderzoekscentra voldoende klein is. De maximale afstand waarover een stabiele radioverbinding kan opgezet worden hangt af van de kracht van de radio-ontvangers; een krachtiger toestel kan over een langere afstand communiceren, maar is meteen ook een stuk duurder. Om korting te kunnen krijgen bij de gezamenlijke aankoop van de radio-ontvangers, en ook om onderhoudsredenen, is het noodzakelijk dat alle radio-ontvangers van hetzelfde type zijn.

Bepaal de minimale kracht K van de radio-ontvangers, zodat er minstens één mogelijke manier van communiceren mogelijk is tussen elk mogelijk paar onderzoekscentra. Ook indirecte communicatie, waarbij een of meerdere onderzoekscentra boodschappen doorspelen, is aanvaardbaar, gezien de grote afstanden in dit gebied. De kracht K is een geheel getal, en stemt overeen met de afstand waarover betrouwbare communicatie met de radio-ontvanger mogelijk is.

De eerste lijn bevat N, het aantal test cases. Voor elke test case volgen dan een aantal lijnen met informatie. De eerste lijn van een test case bevat twee getallen, namelijk:

- het aantal satelietverbindingen S, met $1 \leq S \leq 100$; en
- het aantal onderzoekscentra P, met $S < P \leq 500$

Daarna volgen P lijnen, die de coördinaten voor elk onderzoekscentrum aanduiden. De coördinaten zijn waarden tussen 0 en 10000 (inclusief), gescheiden door een komma, en werden bepaald relatief ten opzichte van een referentiepunt in het uiterste zuidwesten van het gebied.

De uitvoer bestaat uit één lijn per test case, die de minimale kracht K van de radio-ontvangers weergeeft. De minimale kracht wordt uitgedrukt in afstand, en is steeds een geheel getal.

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

```
2
1 2
100 100
500 500
2 4
0 100
0 300
0 600
150 750
```

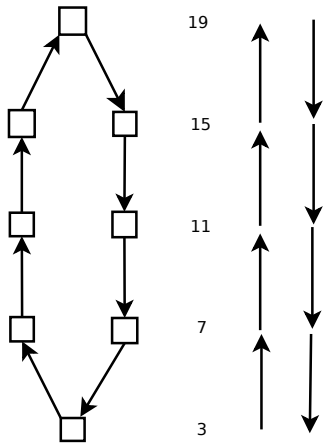
Voor bovenstaande invoer moet volgende uitvoer gegenereerd worden:

```
566
213
```

4.2 Radliften

In het jaar 2500 leven de meeste mensen in superwolkenkrabbers zodat alle vruchtbare land voor de voedselvoorziening gebruikt kan worden. Mensen wonen en werken in éénzelfde enorm gebouw en moeten massaal naar een andere verdieping. De huidige liftsystemen zijn veel te traag voor gebouwen van honderd verdiepingen en honderden mensen die van de lift gebruik willen maken. Dhr R. Adlift heeft een nieuw soort lift ontwikkeld: de radlift. Het idee is afkomstig van de kermis waar de uitvinder een rad aan het werk zag. Een radlift bestaat uit een aantal kooien (evenveel als het aantal stopplaatsen van de radlift, voorgesteld door vierkantjes in figuur 1 links). Voor de gebruikers

komt een radlift overeen met twee liftkokers: één waarin de liftkooien altijd naar boven bewegen, en één waar de kooien altijd naar beneden gaan (zie figuur 1 rechts).



Figuur 1: Voorstelling van een radlift (links) en het equivalent van 2 richtingen rechts.

Stel dat zo´n lift op elke verdieping zou stoppen, dan kan hij heel veel mensen tegelijk verplaatsen, maar iemand die op verdieping 0 start en op verdieping 328 moet uitstappen is dan wel heel lang onderweg want de meeste tijd staat de lift stil zodat de mensen kunnen in- en uitstappen. Daarom zijn er in de superwolkenkrabber meerdere liften en stoppen niet alle liften op alle verdiepingen. Er is bijvoorbeeld een lift die start op verdieping 0 en enkel stopt op verdiepingen 20, 40, 60, 80 en 100 (en terug: startend op 100 en dan naar 80, 60, 40, 20 en 0). Een andere lift start bijvoorbeeld op 8, stopt op 20 en gaat in stappen van 2 (dus 8, 10, 12, 14, 16, 18 en 20 en natuurlijk terug: 20, 18, 16, 14, 12, 10, 8). De lift in de figuur start op verdieping 3, eindigt op verdieping 19 en stopt om de 4 verdiepingen (en analoog naar beneden).

We hebben reeds vermeld dat de gebruikers van radliften op een verdieping, bijvoorbeeld 11 in figuur 1, voor elke radlift 2 liftkokers zien: één waarin de kooien naar boven bewegen en één waarin de kooien naar beneden bewegen. Wij hebben de radliften genummerd van 1 tot n, en de kokers die ermee overeenkomen van 1 tot 2n.

Radlift i komt dan overeen met de kokers $2i - 1$ en $2i$ waarbij in de eerste de kooien naar boven gaan en in de tweede de kooien naar beneden gaan.

Stel dat de radlift in figuur 1 het nummer 4 draagt, dan ziet de gebruiker in plaats van die ene radlift twee liftkokers: een koker met nummer 7 (met kooien die naar boven bewegen) en een koker met nummer 8 (met kooien die naar beneden bewegen). Op de onderste verdieping kan men natuurlijk niet verder naar beneden en op de bovenste kan men niet verder naar boven, dus daar is er per radlift maar één koker (met een oneven nummer onderaan en een even nummer bovenaan). In liftkoker 7 kunnen mensen instappen op verdiepingen 3, 7, 11, 15 en uitstappen op een hogere verdieping uit de reeks 7, 11, 15, 19. In liftkoker 8 kunnen mensen instappen op verdiepingen 19, 15, 11, 7 en uitstappen op een lagere verdieping uit de reeks 15, 11, 7, 3.

De kooien bewegen synchroon: ze starten en stoppen allemaal op hetzelfde ogenblik. Elke kooi blijft een vaste tijd op een verdieping staan zodat de mensen tijd hebben om eventueel van radlift te wisselen. De tijd die een kooi nodig heeft om naar de volgende verdieping te gaan is verwaarloosbaar.

De tijd die een persoon nodig heeft om van verdieping A naar verdieping B te gaan is daarom enkel afhankelijk van het aantal verdiepingen waar de kooien die hij daarvoor neemt, zullen stoppen.

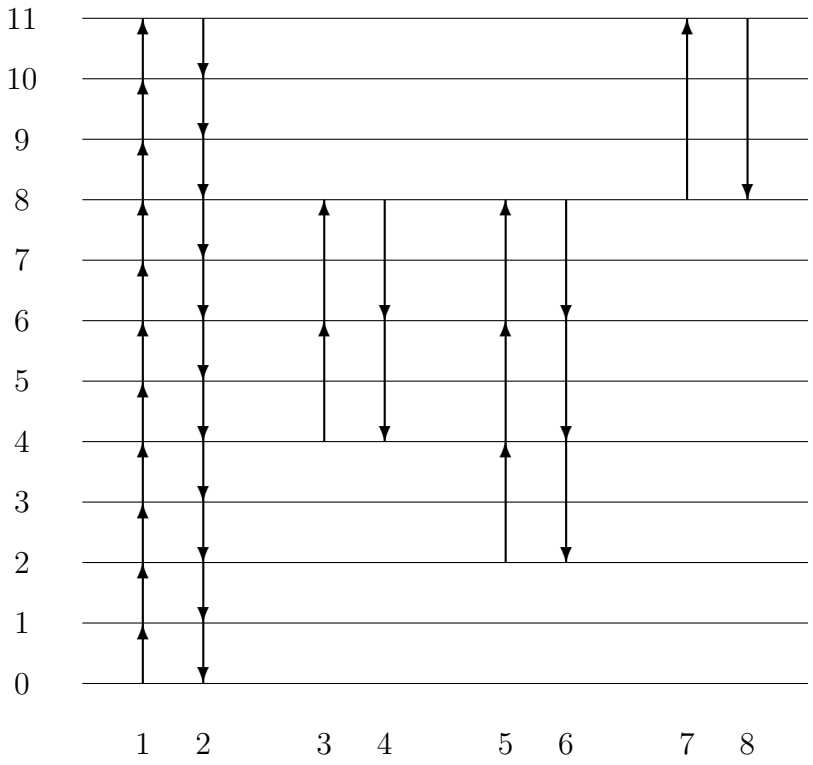
Schrijf nu een programma dat

- gegeven een aantal verdiepingen (als er N verdiepingen zijn, dan zijn ze genummerd van 0 tot (N-1)) en
- gegeven een aantal radliften (met onderste en bovenste verdieping en grootte van de tussenstap) en
- gegeven de beginverdieping en eindverdieping van een verplaatsing

bepaalt welke sequentie van kooinummers die persoon moet nemen (samen met de verdiepingen waar de persoon die kooi moet verlaten) om zo snel mogelijk (dus met zo weinig mogelijk tussenstops) de eindverdieping te bereiken. We veronderstellen dat die peroon steeds plaats vindt in de kooi die hij moet nemen. Er is steeds een oplossing mogelijk, d.w.z. dat er steeds een pad bestaat van de beginverdieping naar de eindverdieping.

Het kan zijn dat er meerdere sequenties eenzelfde minimum aantal stopplaatsen geven. Je schrijft dan de kortste sequentie (= kleinst aantal te nemen liftkooien) uit. Indien er meerdere zijn met dezelfde lengte, kies dan deze met de kleinste sequentie van kokernummers (kleinste nummer van eerst genomen koker, als gelijk: kleinste nummer van tweede genomen koker, ...). Als ook deze gelijk zijn, neem dan diegene met de kleinste sequentie van uitstapverdiepingen (overstapverdiepingen).

Dit wordt geïllustreerd aan de hand van het 3de voorbeeld (visualisatie zie figuur 2).



Figuur 2: Visualisatie van het 3de invoervoorbeeld

Stel dat ik van verdieping 4 naar verdieping 8 wil. Ik neem dan een kooi in de koker met nummer 3 omdat 3 kleiner is dan 5 (verplaatsing met zelfde kost) en omdat er in dit geval geen snellere weg is als ik kokers van meerdere radliften gebruik.

Als ik van verdieping 2 naar verdieping 8 wil, neem ik een kooi in de koker met nummer 5. Ik zou ook eerst een kooi uit de koker met nummer 5 kunnen nemen en op verdieping 4 of 6 overstappen naar een kooi uit de koker met nummer 3 (zelfde kost), maar dan is het aantal te nemen kokers groter. Dus doe ik dit niet.

Invoer De eerste regel bevat een enkel geheel getal dat het aantal superwolkenkrabbers (= aantal testgevallen) voorstelt. Elk testgeval (= elke superwolkenkrabber) wordt als volgt opgegeven:

- een regel met het aantal verdiepingen ($2 \leq V \leq 300$) en het aantal radliften ($1 \leq L \leq 50$)
- L regels met telkens drie gehele getallen: onderste verdieping OV, bovenste verdieping BV en stapgrootte SG van elke radlift (uiteraard is (BV-OV) altijd een veelvoud van SG).
- een regel met de begin- en eindverdieping van een verplaatsing.

Opgelet, zoals reeds vermeld moet elke radlift vertaald worden naar 2 kokernummers. Radlift i geeft aanleiding tot een koker met nummer $2 * i - 1$ voor de kooien die omhoog gaan en een koker met nummer $2 * i$ voor de kooien die omlaag gaan. In het 3de voorbeeld zijn er 4 radliften, dus zijn er 8 kokers met nummers: 1 en 2, 3 en 4, 5 en 6 en tot slot 7 en 8.

Tussen de getallen die zich op één regel bevinden, staat er steeds één spatie.

Uitvoer Per superwolkenkrabber wordt er 1 outputregel gegenereerd met de paren (kokernummer, uitstapverdieping) van het snelste pad steeds voorafgegaan door het volgnummer van de testcase (superwolkenkrabber) en een spatie. Tussen en na de haakjes staan er geen spaties.

Let op! Zorg ervoor dat je uitvoer geen overbodige tekens bevat, bijvoor- beeld 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*

```
3
5 3
0 4 2
1 3 1
1 4 3
0 1
5 3
0 4 2
1 3 2
0 4 1
1 4
12 4
0 11 1
4 8 2
2 8 2
```

```
8 11 3
0 11
```

Uitvoer

```
1 (1,2)(4,1)
2 (3,3)(5,4)
3 (1,2)(5,8)(7,11)
```

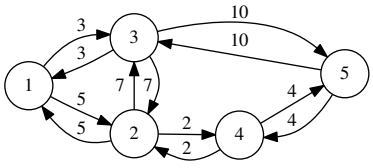
In de uitvoer voor het eerste testgeval stelt (1,2)(4,1) het gebruik van twee liftkokers voor, namelijk een met nummer 1 waarbij men overstapt op verdieping 2 en dan de koker met nummer 4 waarbij men uitstapt op verdieping 1.

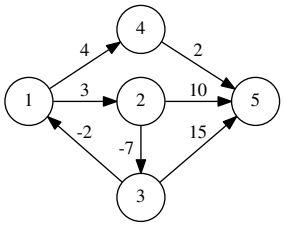
Bij het 3de voorbeeld vertrekken we op verdieping 0 en moeten we naar verdieping 11. We nemen eerst een kooi in de koker met nummer 1 en stappen uit op verdieping 2. Dan nemen we een kooi in koker 5 om op verdieping 8 uit te stappen. We stappen NIET uit op verdieping 4 om een kooi met nummer 3 te nemen want het pad (1,2)(5,8) is korter dan het pad (1,2)(5,4)(4,8). Tenslotte nemen we een kooi in koker 7 om onze bestemming (verdieping 11) te bereiken.

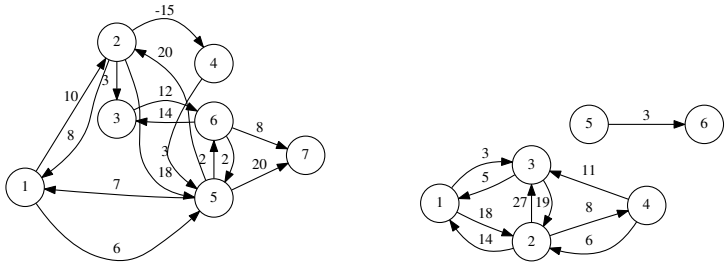
4.3 Jediparcours

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
```

5 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 $k > I$.

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 Kroegentocht

De VPW-juryleden willen op kroegentocht in VPW-stad. Deze stad bevat veel kroegen waarbij de VPW-mocktail op de kaart staat. De prijs van deze mocktail kan in elke kroeg verschillend zijn. Elke kroeg heeft ook een gezelligheidsscore die door de klanten van die kroeg werd toegekend. Aangezien ons budget beperkt is, willen we een kroegentocht uitstippelen die de totale gezelligheidsscore gaat maximaliseren. Er zijn wel een paar voorwaarden. In de eerste plaats drinken we in elke kroeg op de tocht één VPW-mocktail. Ten tweede mogen we ons budget niet overschrijden, d.w.z. de som van

[1] www.diedelphi.org

de prijzen bij de kroegen op de tocht moet kleiner of gelijk zijn aan het budget. Natuurlijk mag elke kroeg maar één keer bezocht worden op de kroegentocht. Voor het gemak worden de prijzen en het budget uitgedrukt in cent (gehele getallen).

Invoer De eerste regel stelt het aantal testgevallen voor. Per testgeval volgt er eerst een regel met twee positieve geheel getallen gescheiden door 1 spatie: het budget B ($0 \leq B \leq 10000$) en het aantal kroegen N ($0 \leq N \leq 1000$). Daarna volgen N regels, 1 regel per kroeg, met de prijs van de mocktail in die kroeg ($0 < P_i \leq 1000$) en de gezelligheidsscore van die kroeg ($0 \leq S_i \leq 100$). Dit zijn telkens 2 gehele getallen gescheiden door 1 spatie.

```
2
10 4
5 10
4 40
6 30
3 50
100 1
150 74
```

Uitvoer Per testgeval moet je één regel afdrukken met de maximale totale gezelligheidsscore. Dit moet voorafgegaan worden door het volgnummer van het testgeval gevolgd door één spatie.

```
1 90
2 0
```