OPGAVE A – Substrings

String A is een substring van string B wanneer string A verkregen kan worden uit string B door 0 of meer letters uit B "weg te strepen".

Voorbeeld : String A = SWEP
String B = SEP
String C = WESP

A en B zijn substrings van A. A en B zijn geen substrings van C. C is wel een substring van C, maar niet van A of B.

Matrix X is een nxn matrix van letters ($n \geq 1$). String B is een string "uit" matrix X wanneer string B een rij of een kolom is van matrix X.

Voorbeeld : Matrix X = A B C D E
F G H I J
K L M N O
P Q R S T
U V W X Y

B1 = KLMNO
B2 = EJOTY
B3 = EDCBA
B4 = UVW

B1 en B2 zijn strings uit matrix X. B3 en B4 zijn geen strings uit matrix X.

String A is een substring van matrix X als er een string B is zodanig dat A een substring is van B én B een string is uit matrix X.

Stel er zijn een matrix X en een string A gegeven. Op hoeveel manieren is deze string A een substring van matrix X ? Met andere woorden, op hoeveel manieren is A een substring van de strings uit matrix X.

De invoer bevat een aantal "runs". Elke run bestaat uit n+1 regels, waarbij de eerste n regels n karakters (A t/m Z) bevatten. Deze eerste n regels van een run vormen de matrix X. De laatste regel van een run bevat de string A. De laatste regel van de laatste run wordt afgesloten met een @. Dit karakter behoort niet tot de string A, maar geeft alleen aan dat er niet nog een run volgt. Alle letters uit het invoerfile zijn hoofdletters.

De uitvoer bevat 1 regel waarop de resultaten van de runs zijn afgedrukt, gescheiden door spaties.
Voorbeeld :

Invoer :

```
ABCD
AADD
BCBC
ABCD
AC
ABAB
BABA
ABAB
BABA
AB
ABCD
BADC
CDAB
DCBA
BAB@
```

Uitvoer : 3 16 0

OPGAVE B – Winnen met risk

Menigeen houdt van het spel Risk, een spel waarbij spelers met legers landen en werelddelen proberen te veroveren. Hierbij is het natuurlijk erg interessant om te weten hoeveel kans van slagen een aanval heeft. Een gevecht wordt uitgevochten door het gooien van 6-zijdige dobbelstenen.

In deze opgave bekijken we een kleine variant op het spel Risk:
· Een aanval wordt 'tot het bittere einde' uitgevoerd.
Er wordt net zolang door gevochten (gedobbeld) totdat de aanvaller nog maar 1 leger over heeft (de aanval is niet gelukt) òf de verdediger geen legers meer heeft (de aanval is gelukt).
· De spelers gooien altijd met het maximale aantal dobbelstenen dat toegestaan is.
De aanvaller moet met 3 dobbelstenen gooien, tenzij hij minder dan 4 legers over heeft. In dat geval moet hij met 1 dobbelsteen minder gooien dan het aantal legers dat hij heeft (er moet immers 1 leger overblijven om het veroverde gebied te bezetten).
De verdediger moet met 2 dobbelstenen gooien, tenzij hij nog maar 1 leger over heeft. In dat geval moet hij met 1 dobbelsteen gooien.

Een worp wordt als volgt beoordeeld:
Neem aan dat de aanvaller met x en de verdediger met y dobbelstenen gooit. Orden de beide worpen; dat wil zeggen laat a_1 t/m a_x de worp van de aanvaller ($a_i \geq a_{i+1}$) en b_1 t/m b_y de worp van de verdediger zijn ($b_i \geq b_{i+1}$).
Als $a_1 > b_1$ dan verliest de verdediger 1 leger
anders verliest de aanvaller 1 leger.
Als min(x,y) ≥ 2 dan
(
. als $a_2 > b_2$ dan verliest de verdediger 1 leger
anders verliest de aanvaller 1 leger.
)

(Steeds worden de hoogste ogen, en dan eventueel de op 1 na hoogste ogen (indien beide spelers met 2 of meer dobbelstenen wierpen) vergeleken. Degeen die het laagst gooide verliest 1 leger. Bij gelijke ogen verliest de aanvaller 1 leger.)

Opgave:
Schrijf een programma dat bepaalt hoe groot de kans is dat de aanvaller wint.

Invoer:
De invoer bestaat uit 1 of meer regels met op iedere regel 2 integers, gescheiden door een spatie. Deze integers stellen resp. het aantal legers van de aanvaller (minimaal 1) en het aantal legers van de verdediger (minimaal 0) voor. De combinatie 1 0 komt niet voor. De invoer wordt afgesloten met een regel bestaande uit 2 nullen.

Uitvoer:
De uitvoer bestaat uit een aantal regels met op iedere regel de kans (afgerond in hele procenten) dat de aanvaller wint, gevolgd door het procent-teken (%).

IBM

Voorbeeld:
Stel de invoer ziet er als volgt uit:

```
5 2
20 40
0 0
```

Dan moet de uitvoer er als volgt uitzien:

```
79%
2%
```

OPGAVE C - Go

Het spel Go wordt gespeeld op een vierkant bord met (gewoonlijk) 19 horizontale en vertikale lijnen. De spelers plaatsen witte en zwarte stenen op de 381 kruispunten van het bord en kunnen stenen van de tegenstander vangen.

U moet een programma schrijven dat, gegeven een aantal eindstellingen, de door wit en zwart behaalde score berekent. Deze score is gelijk aan het aantal gevangen stenen plus het aantal veroverde bordpunten (gebied). Een onbezet bordpunt behoort tot wits gebied als het, gaande via de lijnen over onbezette punten, wel met een witte maar niet met een zwarte steen verbonden is. Voor zwart geldt het omgekeerde. Alleen onbezette bordpunten tellen.

Voor de kenners: U hoeft dus geen rekening te houden met zogeheten 'wilde' stenen.

Een stelling wordt in de input gerepresenteerd door k regels van ieder k karakters, $1 \leq k \leq 19$, waarbij onbezette bordpunten door een punt (.), witte stenen door een 0 en zwarte door een 1 weergegeven worden. Hierna volgt een regel met de aantallen door wit en zwart gevangen stenen (gescheiden door een spatie).

De invoerfile bevat 1 of meer van zulke stellingen zonder verdere scheiding. De uitvoer dient voor iedere stelling een regel met een volgnummer en het totaal aantal punten van wit en van zwart te bevatten.

Voorbeeld:
Stel de invoer ziet er als volgt uit:

```
00100
11.1.
3 2
1.
0 0
```

Dan moet de uitvoer er als volgt uit zien:

```
Stelling 1. Wit: 7, zwart: 6.
Stelling 2. Wit: 0, zwart: 3.
```

OPGAVE D – Beschrijving Mini Produktie Systeem

In de wereld van kunstmatige intelligentie wordt voor het programmeren van zoekprocessen dikwijls gebruik gemaakt van een produktie systeem. Een produktie systeem bestaat uit de volgende componenten :

- Een verzameling regels of produkties. Iedere produktie bestaat uit een conditie en een actie. De conditie geeft aan wanneer de regel toegepast kan/mag worden; de actie beschrijft de uit te voeren verandering –het weghalen of toevoegen van feiten– wanneer de regel wordt toegepast. Een regel waarvan de conditie waar is, noemen we enabled. Het toepassen van een regel in de context noemen we het afvuren van die regel.
- Een beschrijving van de huidige toestand van de zoekruimte, in de vorm van een verzameling feiten, de context. De feiten drukken een hoeveelheid kennis uit; de regels geven aan hoe je, uitgaande van bepaald kennis, nieuwe kennis kunt afleiden.
- Een controle strategie specificeert in welke volgorde de regels worden vergeleken met de feiten verzameling en hoe conflicten moeten worden opgelost wanneer meerdere regels tegelijk matchen (conflict resolutie).

Een voorbeeld van een produktie systeem is het volgende :

De regels zijn :

[1] als heeft_haar dan is_zoogdier
[2] als geeft_melk dan is_zoogdier
[3] als heeft_veren dan is_vogel
[4] als heeft_vinnen en niet is_zoogdier dan is_vis
[5] als is_zoogdier en heeft_hoeven dan is_hoefdier
[6] als is_zoogdier en eet_vlees dan is_carnivoor
[7] als is_carnivoor en kleur_bruin dan is_leeuw
[8] als is_hoefdier en kleur_bruin dan is_paard
[9] als is_zoogdier en heeft_vinnen dan is_walvis
[10] als is_vis en kleur_bruin dan is_paling

>De verzameling feiten :

heeft_haar, heeft_hoeven, kleur_bruin

>De controle strategie :

herhaal de volgende opdrachten totdat er geen regels meer afvuren : verzamel alle regels die enabled zijn en kies daaruit de regel met het laagste nummer waarvan de actie –het weghalen of toevoegen van het genoemde feit– werkelijk effect heeft, en voer de regel uit.

Gegeven de genoemde verzameling feiten, gedraagt het produktie systeem zich als volgt :

Context	:	{heeft_haar, heeft_hoeven, kleur_bruin}
Enabled regels	:	1
Voer uit	:	1, voeg feit "is_zoogdier" toe
Context	:	{heeft_haar, heeft_hoeven, kleur_bruin, is_zoogdier}
Enabled regels	:	1, 5; conflict resolutie, alleen 5 heeft effect
Voer uit	:	5, voeg "is_hoefdier" toe
Context	:	{heeft_haar, heeft_hoeven, kleur_bruin, is_zoogdier, is_hoefdier}
Enabled regels	:	1, 5, 8; conflict resolutie, alleen 8 heeft effect
Voer uit	:	8, voeg "is_paard" toe
Context	:	{heeft_haar, heeft_hoeven, kleur_bruin, is_zoogdier,is_hoefdier, is_paard}
Enabled regels	:	1, 5, 8; conflict resolutie, geen enkele regel heeft effect
Stop, we zijn klaar		

Opgave

Schrijf een programma dat een willekeurig aantal produktie systemen kan verwerken. De invoer bestaat uit een aantal "runs". Iedere run bevat de beschrijving van een produktie systeem. De produktie systemen worden afgesloten door een regel waarop als eerste en enige character een '#' staat. Voor deze opgave geldt voor ieder produktie systeem dat er maximaal 16 produktie regels en maximaal 64 feiten zijn. Iedere produktie regel en ieder feit staat op 1 invoerregel, die maximaal 80 characters lang kan zijn.

Formeel is de invoer met syntaxregels als volgt te beschrijven:

invoer	::= {produktiesysteem} <eof>
produktiesysteem	::= produkties <eoln> context eindesysteem
produkties	::= {regel <eoln>}
context	::= {feit <eoln>}
regel	::= als conditie dan actie
conditie	::= test {en test}
actie	::= update {en update}
test	::= [niet] feit
update	::= [niet] feit
feit	::= letter {letter|cijfer|underscore}
letter	::= <een hoofdletter of een kleine letter>
cijfer	::= <een cijfer>
eindesysteem	::= '#' <eoln>
underscore	::= '_'
als	::= <empty>
dan	::= '=>'
niet	::= '!'
en	::= ','

Voor mensen die niet vertrouwd zijn met syntaxregels, of die de in deze opgave gebruikte notatie niet kennen, volgt hier een korte beschrijving van deze notatie.

In syntaxregels mag het symbool aan de linkerkant van '::=' steeds vervangen worden door de symbolen aan de rechterkant. Symbolen die vervangen kunnen worden (non-terminals) zijn altijd identifiers, symbolen die niet vervangen kunnen worden (terminals) staan tussen enkele quotes indien ze als string op te schrijven zijn, of tussen '<' en '>' indien ze alleen te omschrijven zijn (bv. <eof>).
De omschrijving <empty> aan de rechterkant betekent dat de non- terminal aan de linkerkant door de lege string vervangen kan worden, m.a.w. kan worden weggelaten.

Aan de rechterkant van een syntaxregel kunnen een aantal commandotekens gebruikt worden.

[symbool]	symbool 0 of 1 keer
{symbool}	symbool 0 of meer keer
symbool1|symbool2	symbool1 of symbool2

De regels in de invoer zijn ongenummerd; beschouw de eerste produktie regel van ieder produktie systeem als regel nummer 1 van dat produktie systeem, en nummer de resterende n ($n \geq 0$) produktie regels van hetzelfde systeem met de nummers 2 t/m n+1. Een conditie evalueert tot true als alle tests in de conditie slagen. De test "feit" slaagt dan en slechts dan als "feit" aanwezig is in de context; de test "niet feit" slaagt dan en slechts dan als "feit" niet aanwezig is in de context. De update "feit" heeft effect als "feit" nog niet aanwezig is; het effect bestaat uit het toevoegen van "feit" aan de context. De update "niet feit" heeft effect als "feit" aanwezig is in de context; het effect is het verwijderen van "feit" uit de context. Een actie heeft effect als minimaal één der updates effect heeft. Het effect van een actie bestaat uit alle effecten van de updates in die actie.

De uitvoer per ingelezen produktie systeem bestaat uit de produktie-regelnummers van de regels die afgevuurd zijn -in volgorde van afvuren en één nummer op één uitvoer regel-, een lege regel en alle feiten uit de context -één feit per uitvoer regel- in alfabetische volgorde. Tussen uitvoer van de verschillende produktie systemen staat een lege regel.

Loops

Een probleem met bovenbeschreven produktie systemen is het in een "oneindige loop" raken van het produktie mechanisme. Een oneindige loop ontstaat wanneer, na 1 of meer produkties, de context weer gelijk wordt aan een eerdere context. Bij de invoer

```
aap=>!aap
!aap=>aap

aap
```

zullen, indien er geen maatregelen getroffen worden, regel 1 en 2 afgewisseld worden afgevuurd en zal het produktie mechanisme niet stoppen. Dergelijke oneindige loops zijn te detecteren door elke nieuwe context te vergelijken met elke voorgaande context. Indien er een context opnieuw voorkomt, moet het produktie mechanisme "LOOP" afdrukken en vervolgens de feiten in de context afdrukken. De uitvoer bij bovenstaande invoer wordt dus

```
1
2
LOOP

aap
```

IBM

Een voorbeeld

Invoer :

```
heeft_haar=>is_zoogdier
geeft_melk=>is_zoogdier
heeft_veren=>is_vogel
heeft_vinnen,!is_zoogdier=>is_vis
is_zoogdier,heeft_hoeven=>is_hoefdier
is_zoogdier,eet_vlees=>is_carnivoor
is_carnivoor,kleur_bruin=>is_leeuw
is_hoefdier,kleur_bruin=>is_paard
is_zoogdier,heeft_vinnen=>is_walvis
is_vis,kleur_bruin=>is_paling

kleur_bruin
heeft_haar
heeft_hoeven
#
```

Uitvoer :

```
1
5
8

heeft_haar
heeft_hoeven
is_hoefdier
is_paard
is_zoogdier
kleur_bruin
```

OPGAVE E – Expressies uitrekenen

In deze opgave zijn we geïnteresseerd in het aantal verschillende manieren waarop we expressies zoals (1+2)*3 en 2+3*(4+5*6) kunnen uitrekenen. Bij het uitrekenen van dergelijke expressies mogen we uitsluitend de volgende regels toepassen:

a)
```
1+1 -> 2
1+2 -> 3
2+5 -> 7
```

b)
```
1*1 -> 1
1*2 -> 2
2*5 -> 10
```

c)
```
x*(y1+y2)-> x*y1+x*y2
x*(y1+y2+y3)-> x*y1+x*y2+x*y3
(x1+x2)*y-> x1*y+x2*y
(x1+x2+x3)*y-> x1*y+x2*y+x3*y
```

Met de eerste groep regels kunnen we ieder tweetal getallen optellen. De regels in de tweede groep stellen ons in staat het produkt van ieder tweetal getallen te berekenen. Met de derde groep regels kunnen we haakjes wegwerken. Voor de x, y1, y2, ..., y, x1, x2, ... in deze regels mogen we willekeurige expressies invullen, dus niet uitsluitend getallen. De expressie (1+2)*3 kunnen we nu op verschillende manieren uitrekenen. Om te beginnen kunnen we 1+2 vervangen door 3. We krijgen dan 3*3. Dit wordt vereenvoudigd tot 9. Als eerste stap kunnen we ook de regel (x1+x2)*y -> (x1*y+x2*y) toepassen (met 1 voor x1, 2 voor x2 en 3 voor y ingevuld). De resulterende expressie 1*3+2*3 kunnen we op twee manieren verder vereenvoudigen, afhankelijk van de volgorde waarin we 1*3 en 2*3 door respectievelijk 3 en 6 vervangen. De expressie (1+2)*3 kunnen we dus op 3 manieren uitrekenen:

```
(1+2)*3 -> 3*3 -> 9
(1+2)*3 -> 1*3+2*3 -> 3+2*3 -> 3+6 -> 9
(1+2)*3 -> 1*3+2*3 -> 1*3+6 -> 3+6 -> 9
```

We zijn er stilzwijgend vanuitgegaan dat * sterker bindt dan +. De regel 3+2 -> 5 is dus niet toepasbaar op de expressie 3+2*3. Bovendien nemen we aan dat + en * associatief zijn. Dit betekent dat we geen onderscheid maken tussen de expressies 1+((2*3)*4+5) en (1+2*(3*4))+5.

LET OP: Soms moet je zelf haakjes plaatsen. Als je bijv. in de expressie 1*(2+3)*4 regel c) toepast, zou dit vereenvoudigen tot 1*2+1*3*4, maar dit is incorrect. In dit geval moet je zelf haakjes plaatsen zodat de expressie 1*(2+3)*4 vereenvoudigt tot (1*2+1*3)*4.

De opgave is nu een programma te schrijven dat voldoet aan onderstaande specificatie.

INVOER: Een willekeurige expressie opgebouwd uit positieve gehele getallen, +'en, *'en en haakjes. Je mag aannemen dat de invoer syntactisch correct is (een expressie kan niet leeg zijn). Bovendien bevat de expressie zo min mogelijk haakjes. (De expressie 1+((2*3)*4+5) zou dus als 1+2*3*4+5 aan het programma aangeboden worden.)

UITVOER: Het aantal manieren waarop de expressie uitgerekend kan worden met de gegeven regels.

VOORBEELD:

invoer	uitvoer
(1+2)*3	3
1+2+3	2

OPGAVE F – Een postzegelprobleem

De staatsdrukker van een land heeft de opdracht gekregen om een nieuwe serie postzegels te gaan drukken. Hij mag zelf de waarden van die zegels kiezen. De serie bestaat uit n verschillende waarden ($1 \leq n \leq 10$). Nu is de drukker gevraagd de waarden zó te kiezen, dat er een zo groot mogelijk aaneengesloten rij van mogelijke porto-waarden (te beginnen bij waarde 1) ontstaat als maximaal m postzegels ($1 \leq m \leq 25$) op een brief mogen worden geplakt. Hierbij is nog de restrictie dat de maximale porto- waarde in dat land 1000 bedraagt.

Opgave: Schrijf een programma dat bij gegeven n en m bovenstaand probleem oplost. Dat wil zeggen: vindt, bij gegeven n en m, de grootst mogelijke aaneengesloten rij van porto-waarden 1,2,3,...,max die gemaakt kan worden. Druk het maximum van deze rij af.

Invoer: De invoer bestaat uit 1 of meer regels, met op iedere regel 2 integers gescheiden door een spatie. Deze integers stellen resp. n en m voor. De invoer wordt afgesloten met een regel bestaande uit 2 nullen.

Uitvoer: De uitvoer bestaat uit een aantal regels met op iedere regel de gevraagde waarde.

Voorbeeld:

Stel de invoer ziet er als volgt uit:

```
3 2
2 4
0 0
```

Dan moet de uitvoer zijn:

```
8
10
```

OPGAVE G - Een kortste pad probleem

Stel we hebben een vierkant bord (een soort van schaakbord) van N velden. Op dit bord is een begin- en een eindveld gegeven. Verder hebben we te maken met een soort van motorisch gestoord schaakstuk dat een aantal (gegeven) bewegingen mag maken. We willen nu weten wat de lengte van een kortste pad is dat het stuk van het beginveld naar het eindveld kan "bewandelen". De lengte van een pad is het aantal stappen dat gedaan wordt.

Specificatie :

- De velden van het bord zijn genummerd van onder naar boven, van links naar rechts.

- Er zitten "gaten" in dit bord, d.w.z. er zijn velden die niet in een pad mogen voorkomen. Het stuk mag er wel "overheen springen".

- De bewegingen die een stuk mag maken worden gegeven door een rij getallen gescheiden door spaties. Wanneer deze rij uit n getallen bestaat, kan het stuk n-1 bewegingen maken en wel van het veld dat aangeduid wordt door het eerste getal naar de velden die aangeduid worden door de overige n-1 getallen. De spiegelingen en rotaties van een beweging zijn niet toegestaan (tenzij expliciet gegeven).

Voorbeeld :

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

Neem een 4*4 bord. Wanneer de toegestane bewegingen gegeven worden door de getallenrij 11 3 14 16, dan betekent dit dat het stuk 3 bewegingen mag maken :

1) Van veld 11 naar veld 3 (2 naar beneden)
2) Van veld 11 naar veld 14 (1 omhoog en 1 naar links)
3) Van veld 11 naar veld 16 (1 omhoog en 1 naar rechts)

- Het "van-veld" en de "naar-velden" zijn velden die op het bord liggen. De waarden zijn dus minimaal 1 en maximaal N.

- De invoer bevat een aantal "runs". Elke run bestaat uit 4 regels getallen. Op de eerste regel is het aantal velden van het bord gegeven (niet kleiner dan 1 en niet groter dan 900). De tweede regel bevat de bewegingen die het stuk kan maken. Op de derde regel zijn de veldnummers gegeven van de "verboden" velden. Tenslotte wordt op de vierde regel het beginveld en het eindveld gegeven (gescheiden door een spatie). Na de laatste run staat in het invoerfile een 0.

- Per run moet de lengte van een kortste pad op één regel worden afgedrukt (het aantal regels in de uitvoerfile is dus gelijk aan het aantal runs). Wanneer er geen pad gevonden kan worden moet de mededeling 'Geen oplossing' afgedrukt worden.

Voorbeeld :

Invoer

```
100
55 45 64 66

1 99
16
11 3 14 16
9
5 1
0
```

Uitvoer

```
11
Geen oplossing
```

OPGAVE H - Manhattan

Op de planeet Repyh storen de bewoners zich niet aan de voor ons gebruikelijke drie dimensies, maar reizen en bouwen ze in willekeurig veel (maximaal 1000) dimensies. De grootste stad van de planeet is Manhattan. Deze stad is hetzelfde opgebouwd als het Manhattan-district in New York: alle straten staan loodrecht op elkaar met een Central Park in het midden. Het enige verschil is dat het straat-patroon en het park uit willekeurig veel dimensies kunnen bestaan.

Eén van de inwoners van Manhattan wil zich van het ene kruispunt naar het andere verplaatsen en vraagt zich af wat de kortste afstand tussen deze twee punten is. De afstand wordt in stappen gerekend. Een stap is een verplaatsing van een kruispunt naar een aangrenzend kruispunt. Het is niet mogelijk om door Central Park te lopen, maar men mag wel over de buitenste rand van het park lopen.

Opgave:
Schrijf een programma dat de kortste afstand tussen twee kruispunten bepaald, waarbij niet door Central Park mag worden gelopen. Druk deze kortste afstand af.

Invoer:
De invoer bestaat uit een aantal "runs". Iedere run bestaat uit een aantal regels:

- Het aantal dimensies in Manhattan (minstens 2 en hoogstens 1000).
- Een hoekpunt van Central Park .
- Een ander hoekpunt van Central Park (het park ligt tussen deze twee punten opgespannen).
- Het startpunt van de reis.
- Het eindpunt van de reis.

Na de laatste run volgt een dimensie 0.

N.B. Ieder punt bestaat uit een rij integers. Deze integers worden gescheiden door een spatie of een <eoln>.

Uitvoer:
De uitvoer bestaat uit een aantal regels, waarbij op elke regel de kortste afstand van een run is afgedrukt. De uitvoer bestaat dus uit evenveel regels als het aantal runs. Als het startpunt of het eindpunt in Central Park ligt, moet de boodschap 'Geen oplossing' worden afgedrukt.

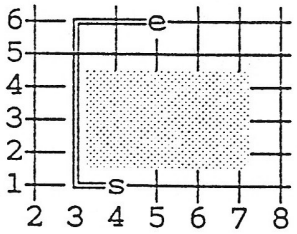
Voorbeeld:
Stel de invoer ziet er als volgt uit:

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

Dan moet de uitvoer er als volgt uit zien:

```
8
Geen oplossing
15
```

Het plaatje dat bij de eerste run hoort ziet er als volgt uit:



Met s = startpunt en e = eindpunt.