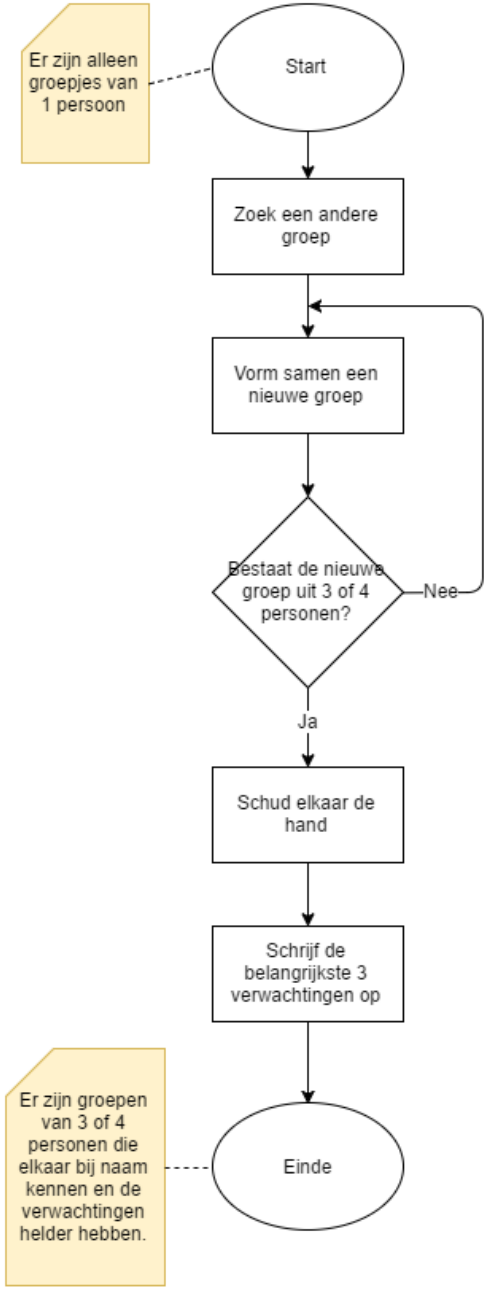
Nascholing
Algoritmisch Denken

Martin Bruggink
Renske Smetsers

Kennismaking en verwachtingen


Er zijn alleen groepjes van 1 persoon
Start
Zoek een andere groep
Vorm samen een nieuwe groep
Bestaat de nieuwe groep uit 3 of 4 personen?
Nee
Ja
Schud elkaar de hand
Schrijf de belangrijkste 3 verwachtingen op
Er zijn groepen van 3 of 4 personen die elkaar bij naam kennen en de verwachtingen helder hebben.
Einde

Doelen Cursus

- Inzicht in wat algoritmisch denken is:
  - Nieuwe examenpogrammen
- (Leren toepassen van) Algoritmisch denken in de les (verantwoord vanuit onderzoek)
  - Praktisch en zinvol voor docent & leerling
- Kennis opdoen van lesmateriaal
  - Alg. Denken met Greenfoot
  - Unplugged werkvormen

Uitgangspunten nascholing

- Theoretische kennis
- ‘Hands-on’ ervaring opdoen
- Tussentijds toepassen in de lespraktijk
- Terugkoppeling ervaringen
- Materiaal aanreiken waar je mee aan de slag kunt

Programma (Algemeen)

- Deel 1 (20 jan)
  - Theorie: AD en flowcharts
  - Praktijk: oefenen met lesmateriaal (AD in uitvoering)
- Deel 2 (23 feb)
  - Erik Barendsen: Relatie AD nieuw examenprogramma & onderzoek
  - Ervaring uitwisselen over toepassen van lesmateriaal
  - Praktijk: oefenen met lesmateriaal (AD in uitvoering)
- Deel 3 (23 maart)
  - Ervaringen uitwisselen over Alg. Denken zelf toepassen
  - Verkenning/verdieping in (nieuw) Greenfoot lesmateriaal

Studielast

De totale studielast is ongeveer 20 uur, inclusief de voorbereiding buiten de bijeenkomsten om. Opbouw studielasturen:

- 3x4 uur voor de bijeenkomsten
- 2x4 uur voor het tussentijds voorbereiden en toepassen in de eigen lespraktijk.

Programma eerste bijeenkomst

- Lesdoelen en kennismaken met AD
- Theorie: AD en flowcharts
- Diner (17:45)
- Praktijk: oefenen met lesmateriaal
- Afsluiting & Einde (20:00)

Uitdaging: Rondleiding

- Bedenk een oplossing:
  - Start & eindig bij hotel
  - Bezoek elk locatie precies 1 keer
- Noteer jouw algoritme / route
  - mbv pijltjes of nummertjes
  - Zodat jouw schoonzus het volgend weekend ook kan uitvoeren
- Evalueer of oplossing aan eisen voldoet

Uitdaging: Rondleiding

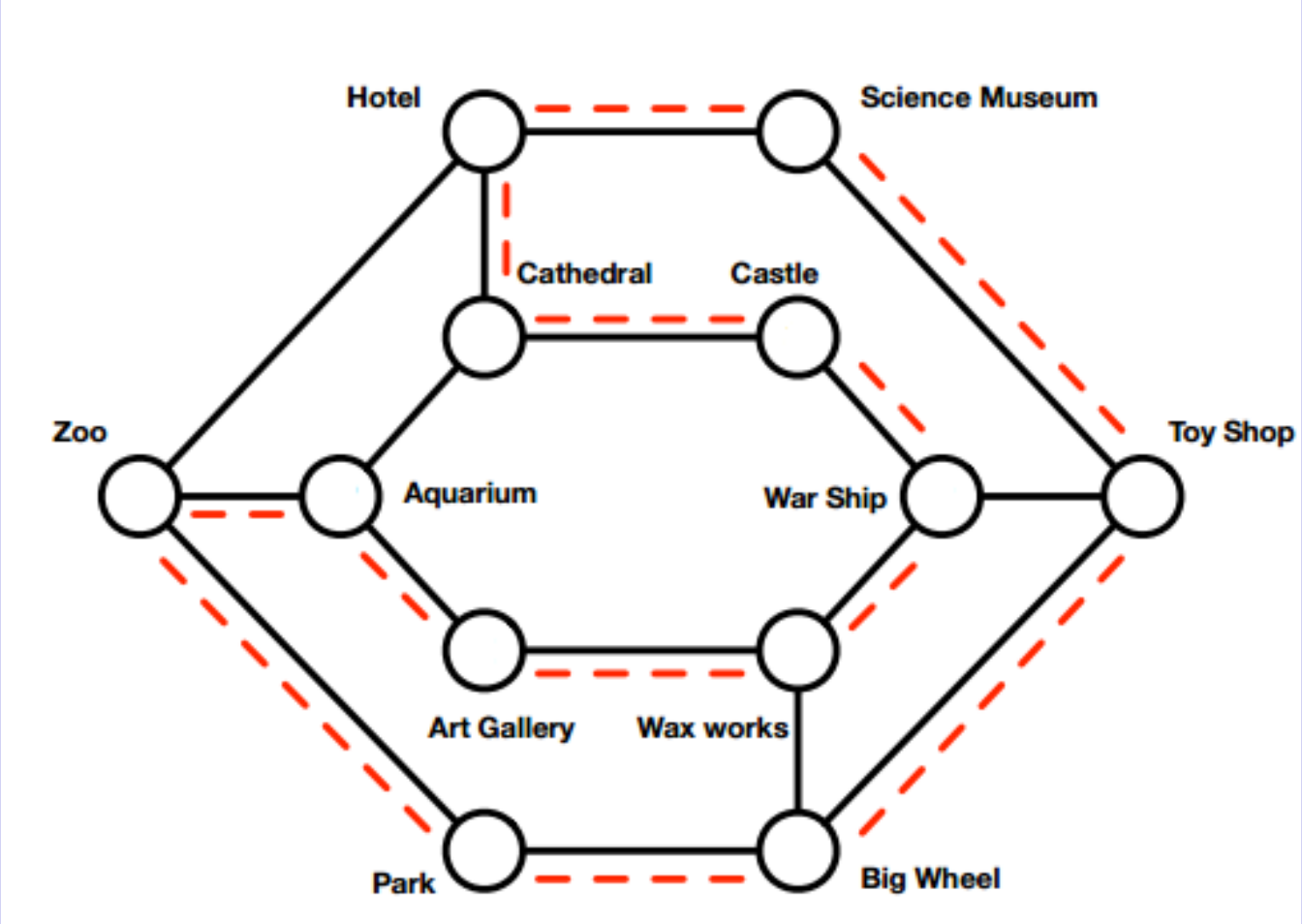
- Oplossing?
- Aanpak?
- Moeilijkheidsgraad?

Rondleiding: waar het om gaat

- Algoritmiek: opschrijven oplossing voor toekomstig bezoek (geluk gegarandeerd)
- Evaluatie: voldoet aan eisen?
- Abstractie & representatie: kaart als model (graaf), alleen relevante info

- *Verdieping: Hamiltonian cycle*

Rondleiding: een oplossing



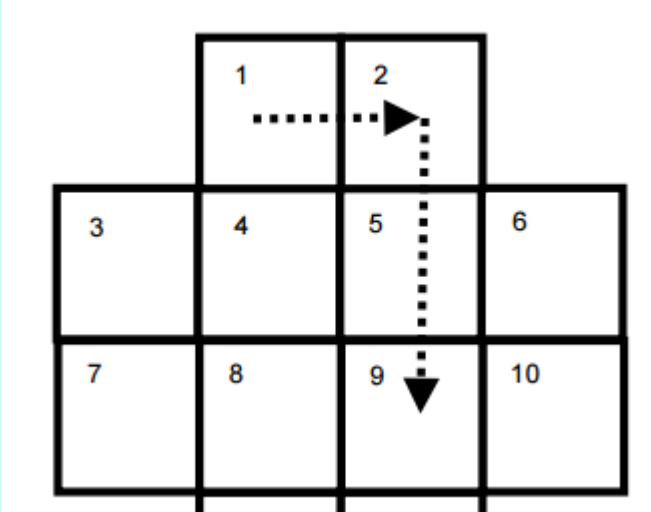
Uitdaging: Paardensprong

Zet een stuk in vakje 1

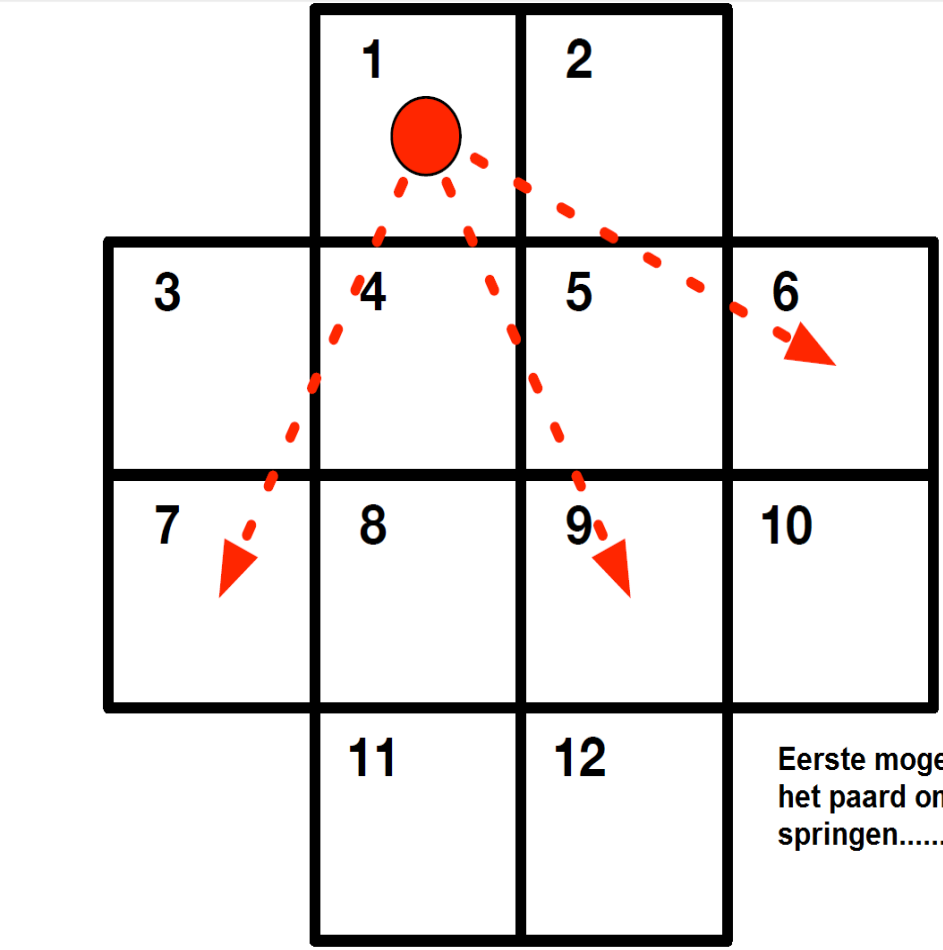
Vind een manier voor een paard om:

- elk vierkant te bezoeken
- precies één keer.

5’ in tweetallen



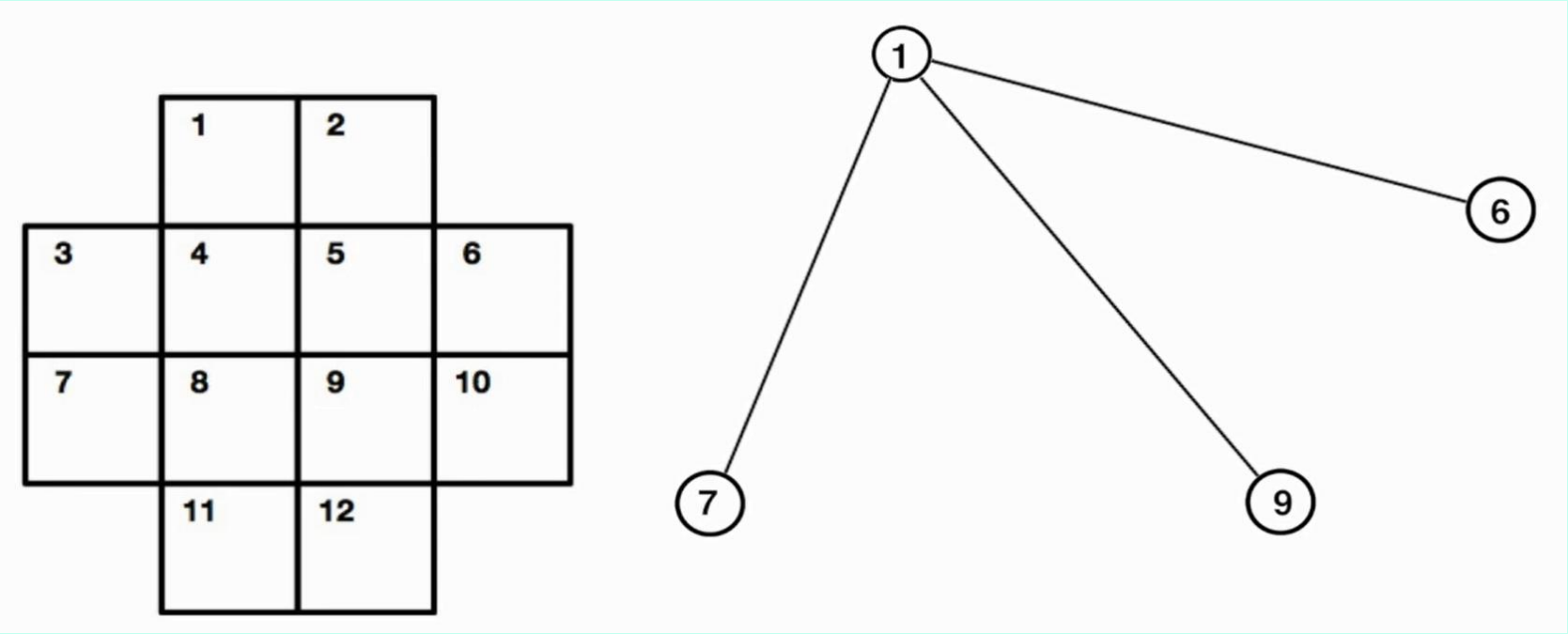
Paardensprong



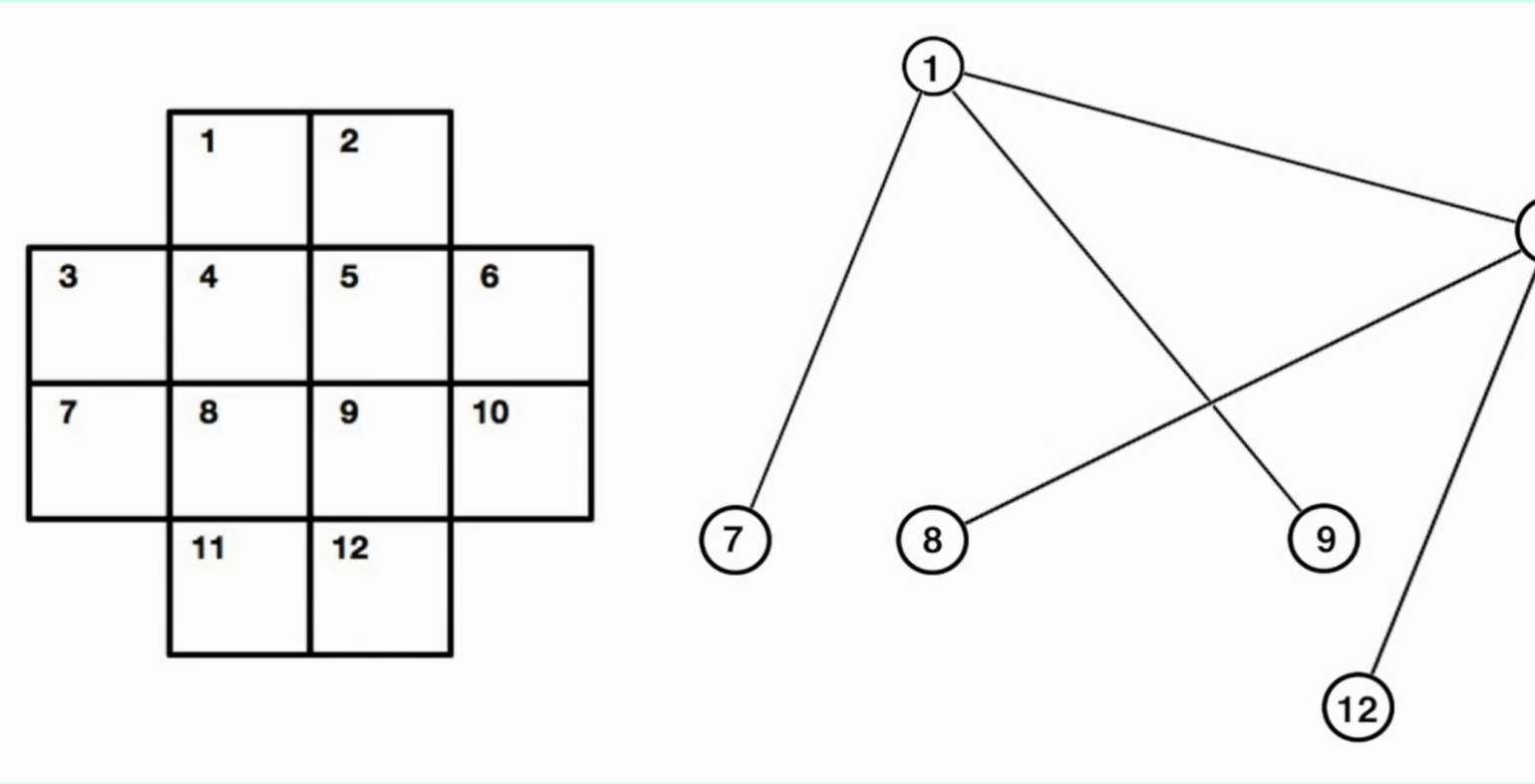
Uitdaging: Paardensprong

- Oplossingen?
- Aanpak?
- Moeilijkheidsgraad?

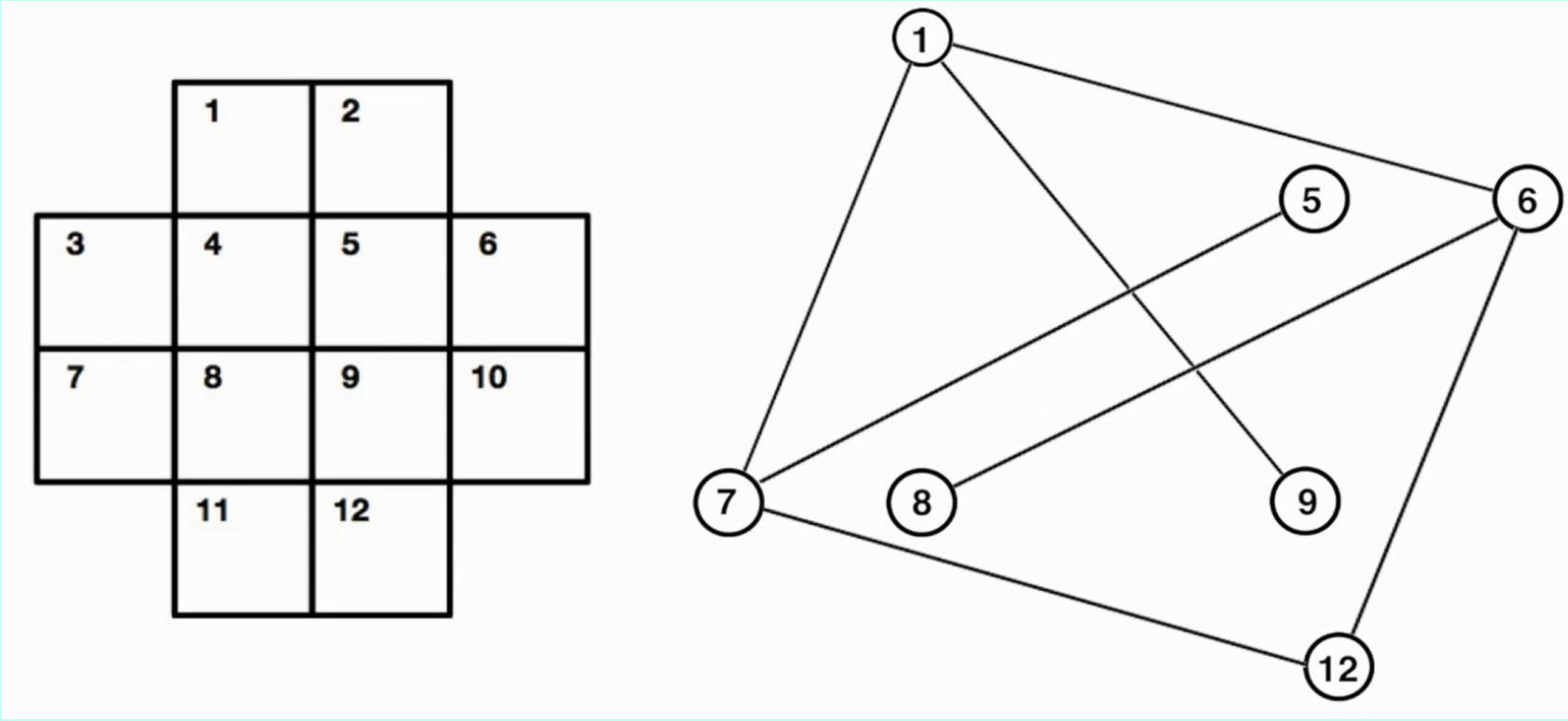
Paardensprong => Graaf



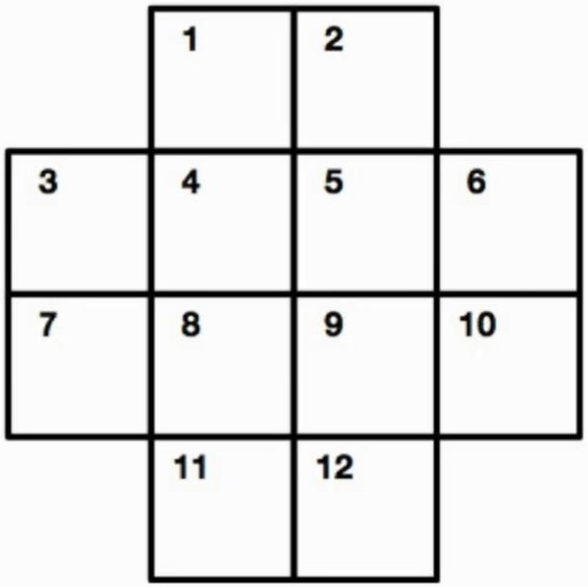
Paardensprong => Graaf

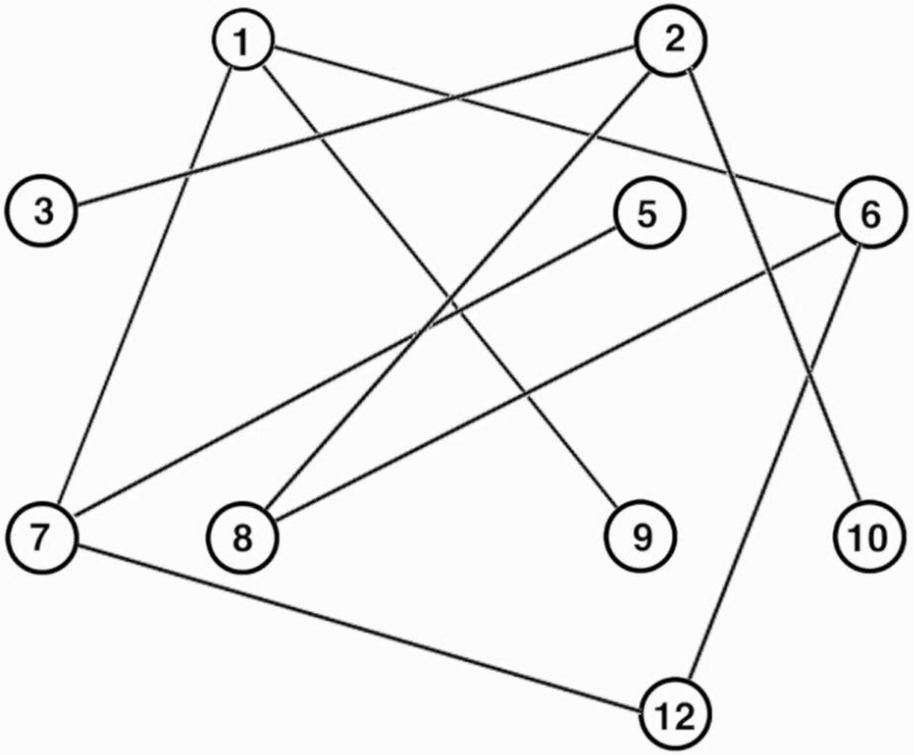


Paardensprong => Graaf

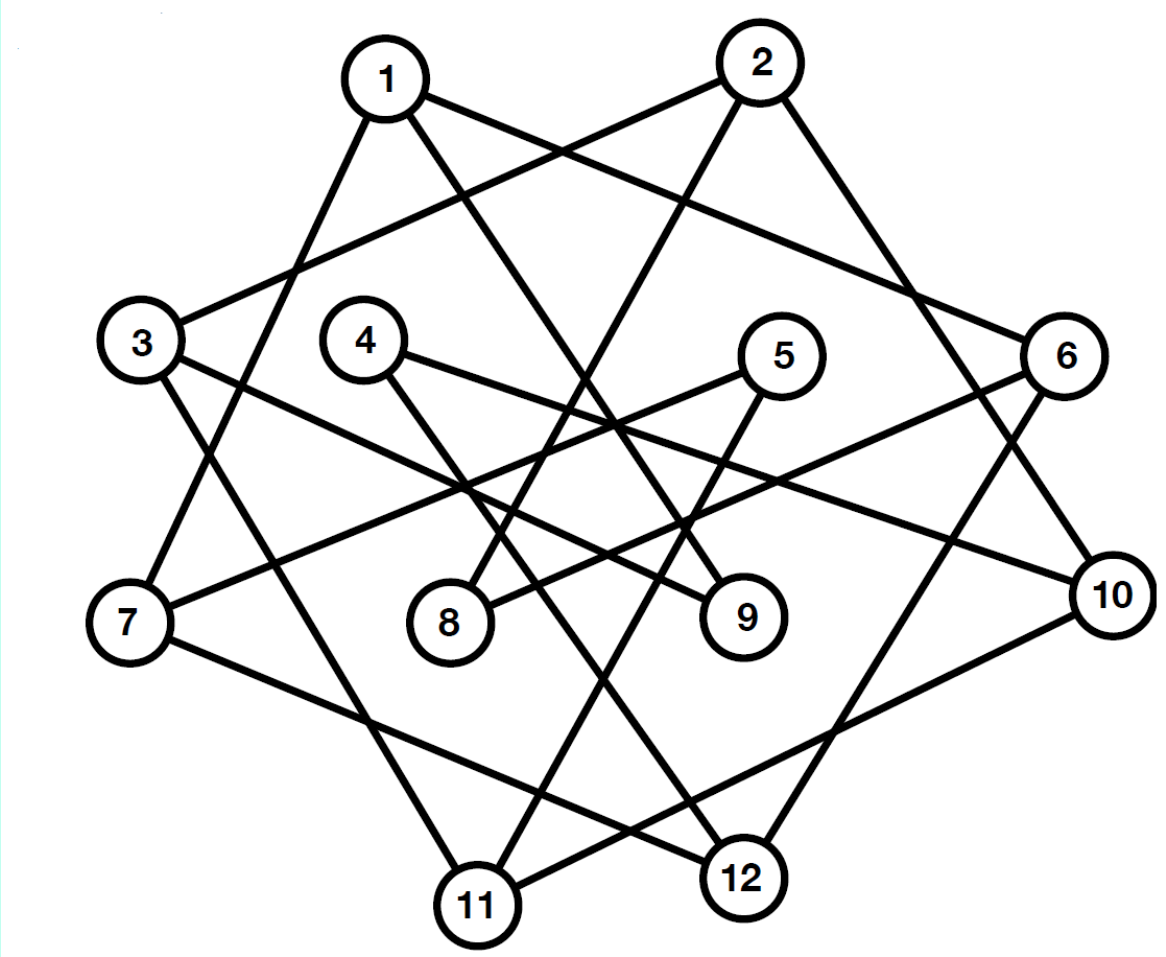


Paardensprong => Graaf

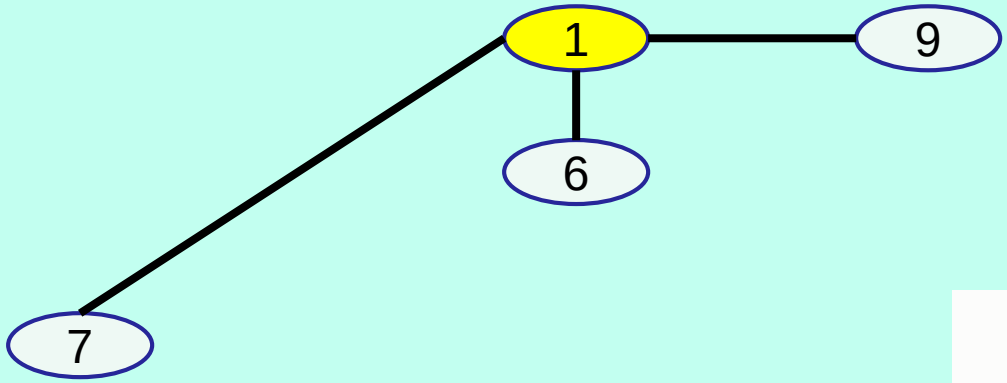


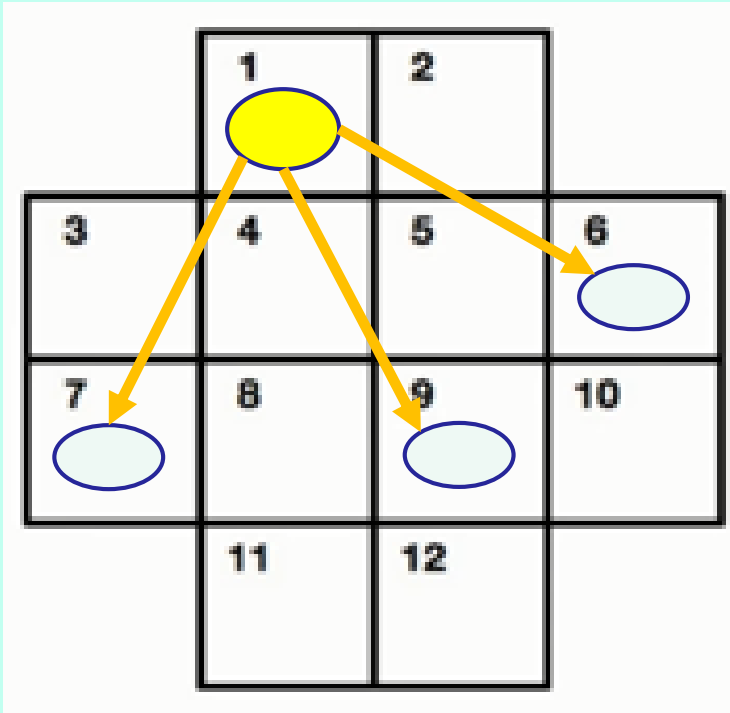


Paardensprong => Graaf

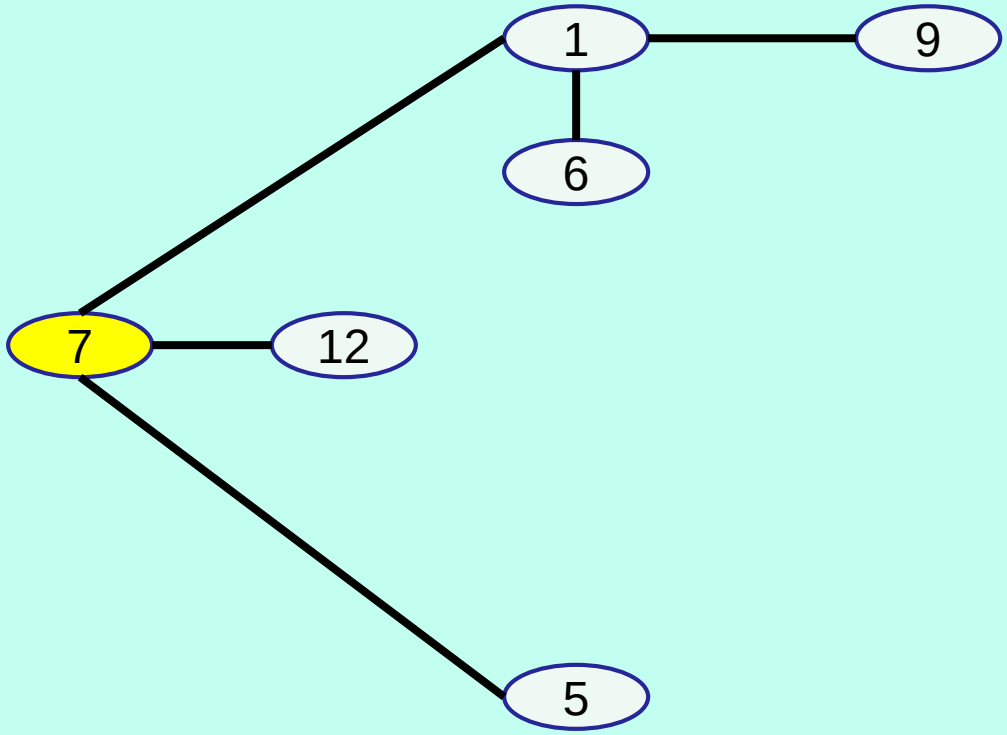


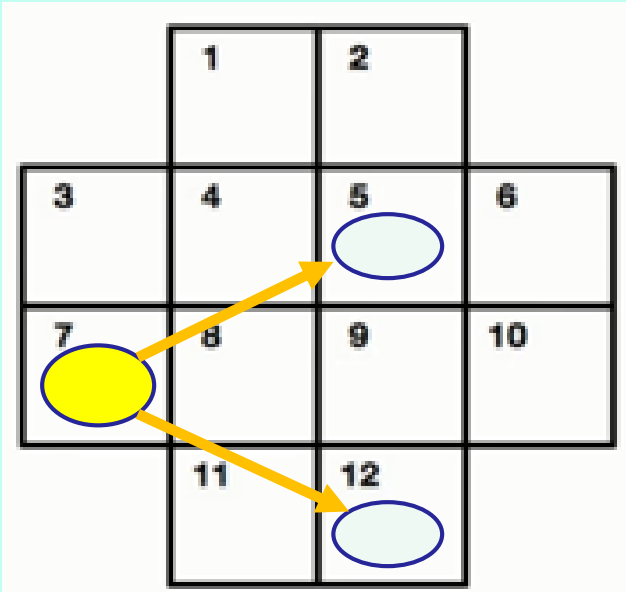
Paardensprong => Graaf



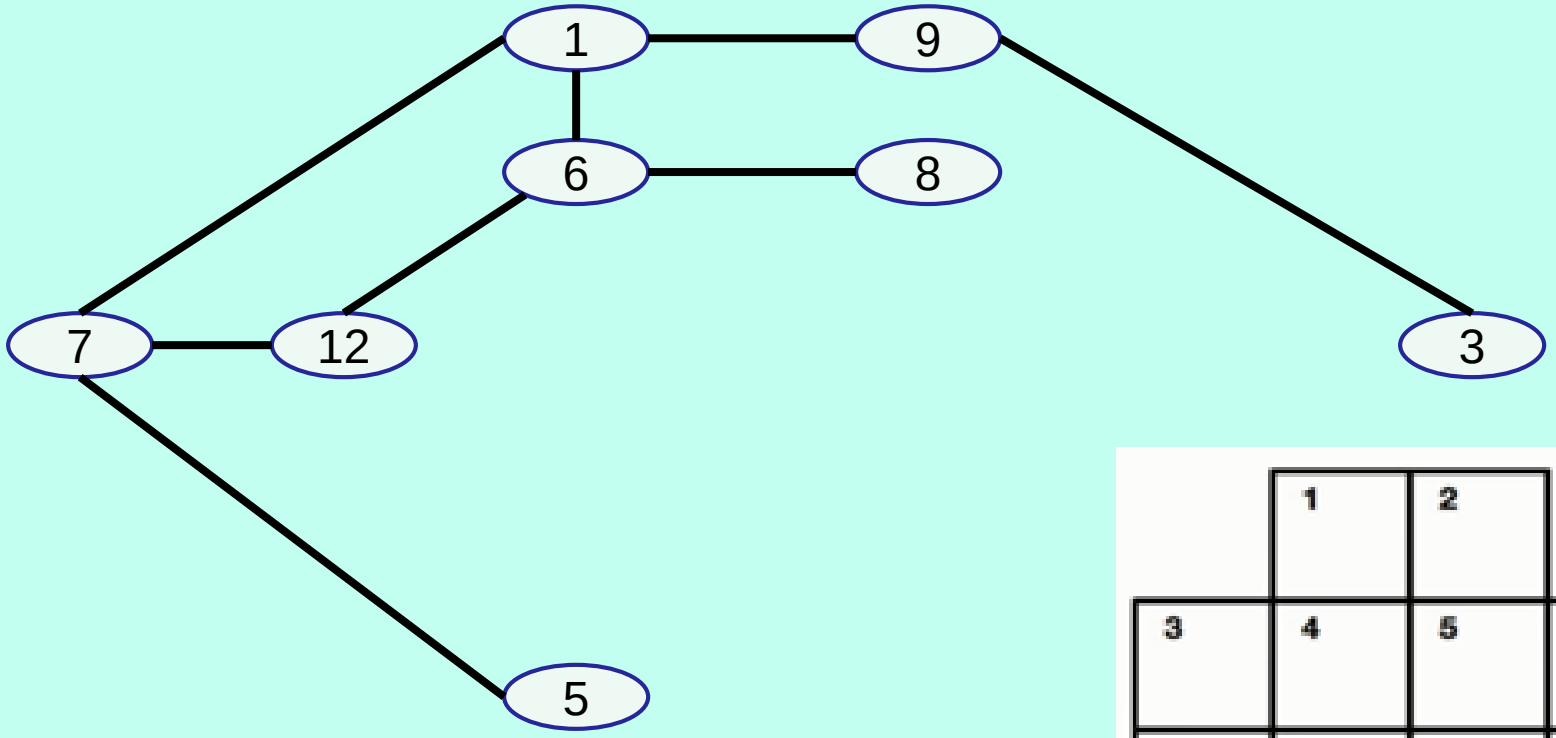


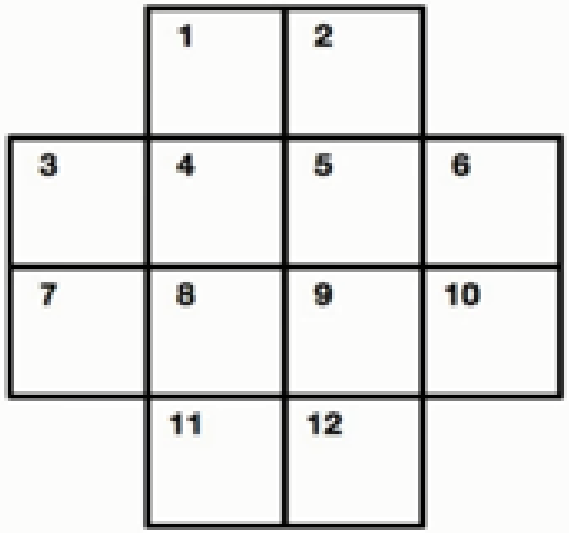
Ruiter => Graaf



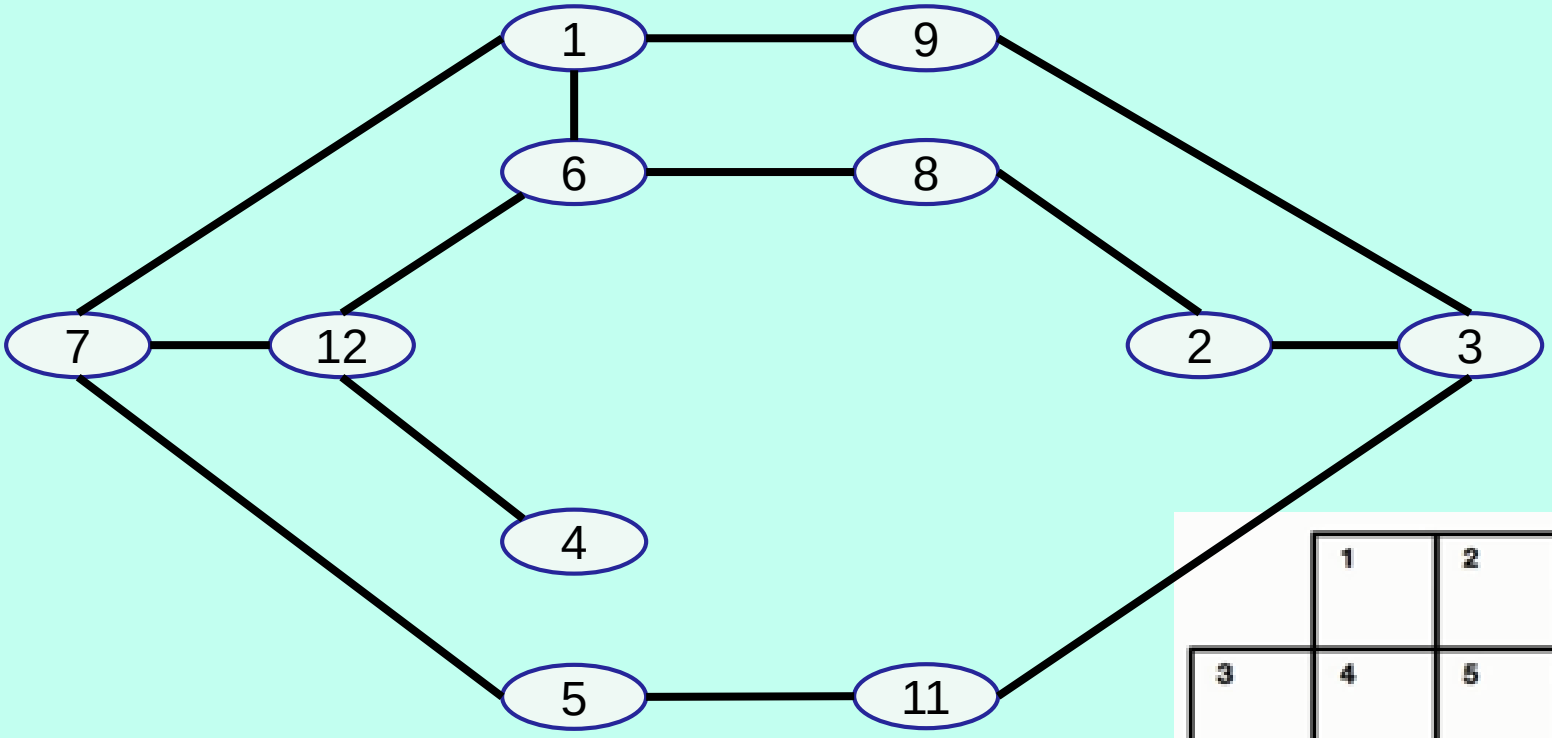


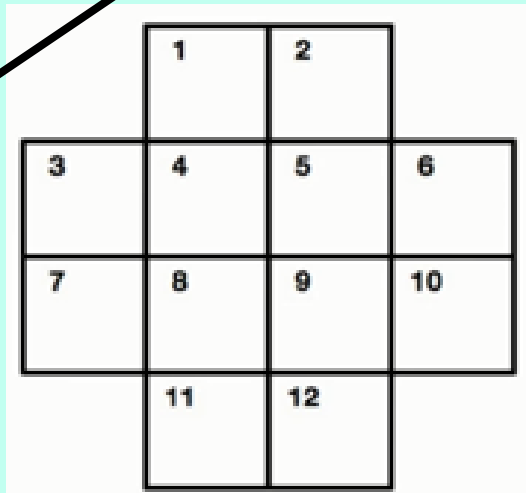
Ruiter => Graaf



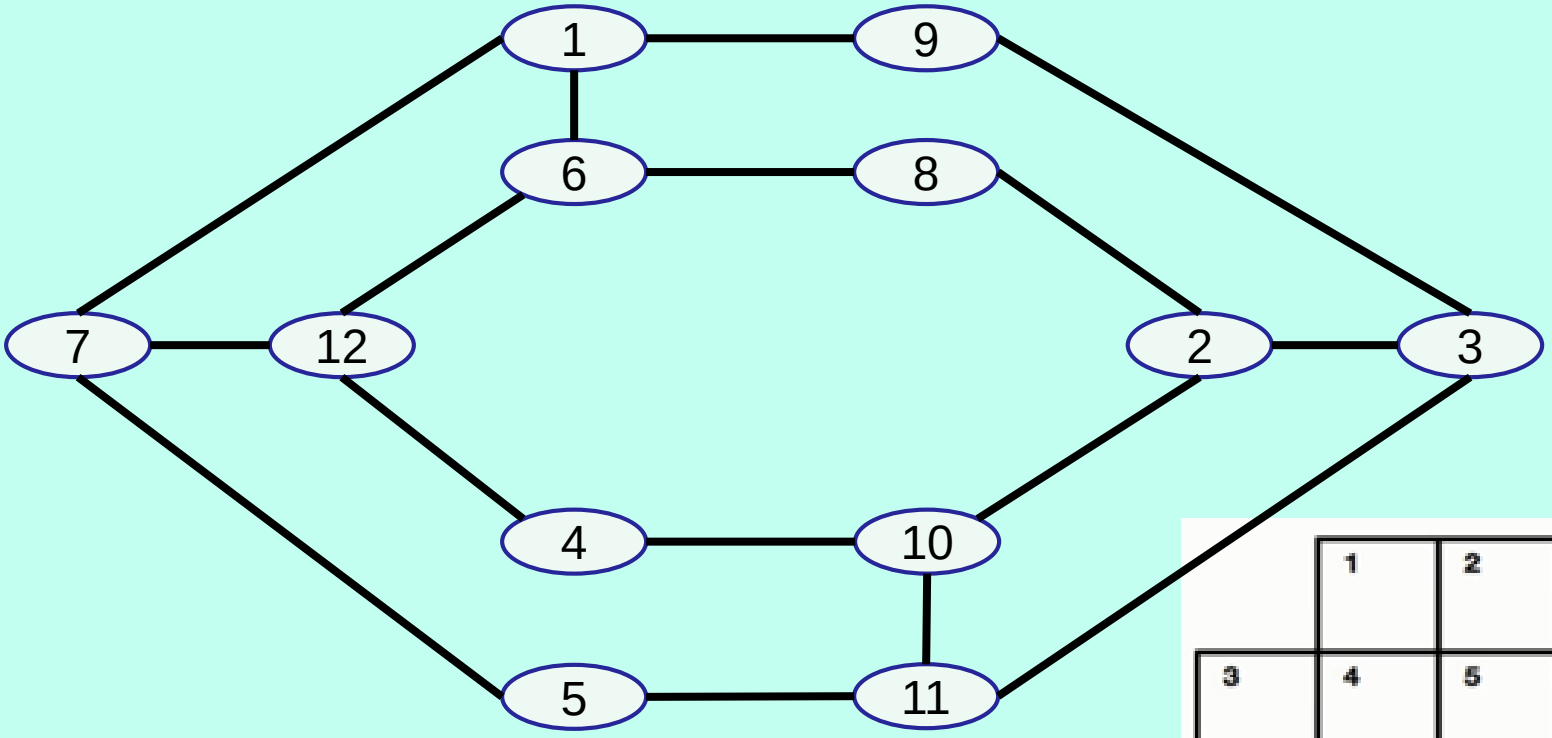


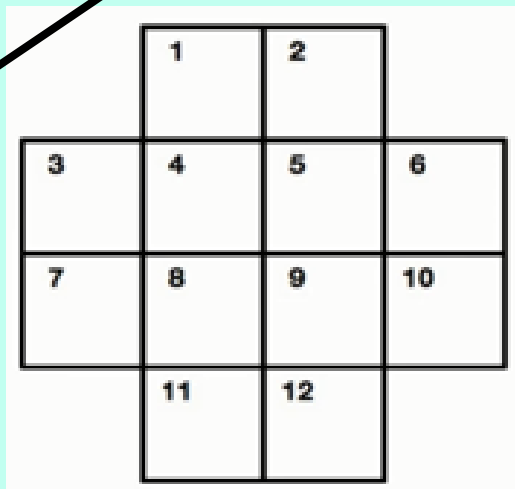
Ruiter => Graaf



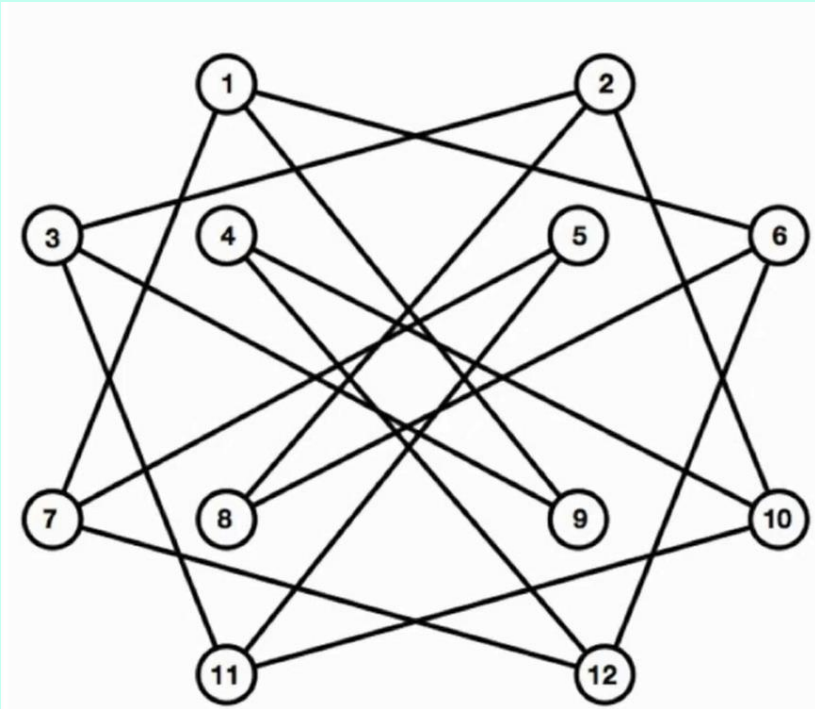


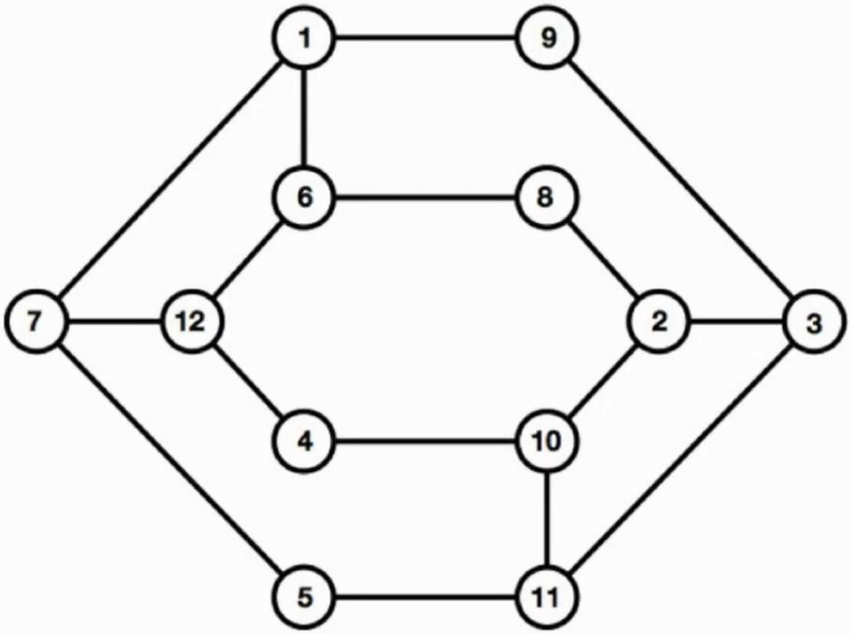
Ruiter => Graaf





Graaf anders getekend

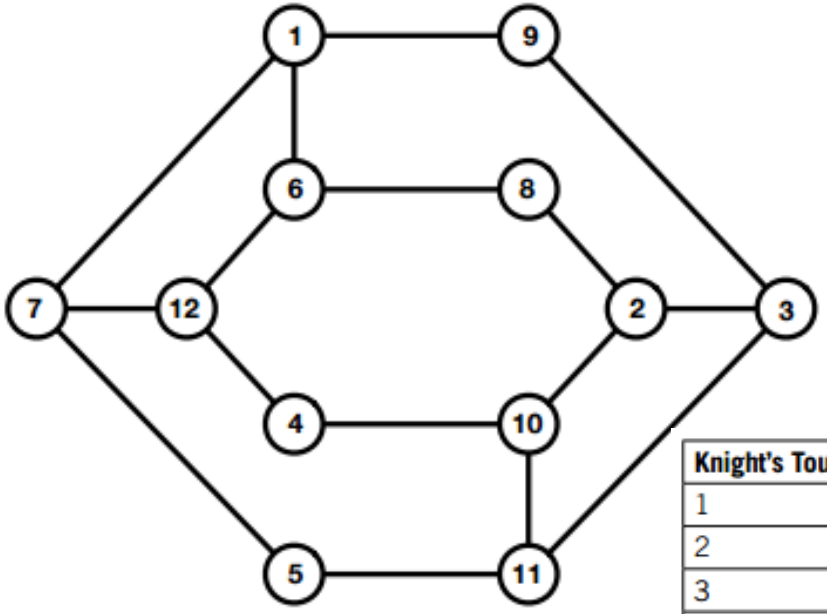




Rondleiding <=> Paardensprong

Paardensprong probleem	Rondleiding touristen attracties
1	Hotel
2	War ship
3	Toy Shop
4	Art Gallery
5	Park
6	Cathedral
7	Zoo
8	Castle
9	Science Museum
10	Wax Works
11	Big Wheel
12	Aquarium

Paardensprong: oplossing



1-9-3-11-5-7-12-4-10-2-8-6-1

Knight's Tour Square	Tour Guide Attraction
1	Hotel
2	War ship
3	Toy Shop
4	Art Gallery
5	Park
6	Cathedral
7	Zoo
8	Castle
9	Science Museum
10	Wax Works
11	Big Wheel
12	Aquarium

Paardensprong : waar het om gaat

Computational Thinking 'truc':

- Abstractie: details weglaten en handige representatie
- Representatie: graaf om probleem te vertegenwoordigen
- Generalisatie: bekende oplossingen gebruiken voor vergelijkbare problemen (transfer)

Paardensprong : waar het om gaat

- Niet alleen algoritmisch denken!

- De kracht abstractie!
- Keuze van representatie om probleem te vergemakkelijken
- Herkennen van problemen en oplossingsrichtingen en aan elkaar koppelen

Relevantie?

- Navigatiesystemen: steden en (1-richting verkeer) straten
- Indexeren webpagina's: sites zijn knopen (nodes), links zijn kanten (edges)
- User-interfaces: horloge of iPhone

- *Verdieping: depth-first / breadth first zoek algoritmes en (representatieve) efficientie*

Computational thinking (big 5):

- Algoritmiek: denken in regels en opeenvolgingen (van deeltaken)
- Abstractie: hoog niveau / weglaten details
- Decompositie: opdelen in deelproblemen
- Generalisatie: patronen & generieke opl.
- Evaluatie: voldoet oplossing aan eisen?
  had het beter/makkelijker gekund?

Kerndoel van Alg. Denken

"… first and foremost, it should teach those young people to THINK."

George Pólya (1887 – 1985)

Proces van Alg. Denken

bron: http://www.conradwolfram.com/home/anchoring-computational-thinking-in-todays-curriculum

Proces van Alg. Denken

1. Definieren van de vraag: kritisch denken**.**
Waar gaat het probleem precies om?
Kern van bijzaak scheiden.
Met wat voor een oplossing zou
ik het problem kunnen aanpakken?

bron: http://www.conradwolfram.com/home/anchoring-computational-thinking-in-todays-curriculum

Proces van Alg. Denken

2. Abstractie:

Vertalen van **oplossingsrichting** naar algoritme/flowchart
Belangrijke stap naar **precies** omschrijven van oplossing
(herkennen van) **Hergebruik** van bestaande oplossingen
Vaak **moeilijkste stap**:

creatieviteit, inzicht, conceptuele begrip, ervaring

=> begeleiding van docent nodig

bron: http://www.conradwolfram.com/home/anchoring-computational-thinking-in-todays-curriculum

DEFINE TRANSLATE COMPUTE INTERPRET

Proces van Alg. Denken

3. **Programmeren**:

Vertalen van algoritme/flowchart naar code

Testen of code voldoet aan opgestelde algoritme/flowchart

bron: http://www.conradwolfram.com/home/anchoring-computational-thinking-in-todays-curriculum

Proces van Alg. Denken

4. Evalueren / interpreteren:

Is oorspronkelijke probleem hiermee opgelost?

Het het beter/slimmer gekunt?

Wat zou ik volgende keer anders doen?

bron: http://www.conradwolfram.com/home/anchoring-computational-thinking-in-todays-curriculum

Waarom Alg. Denken?

- Probleemoplossend vermogen
- Complexere dingen maken
  - herbruikbaar, uitbreidbaar, efficiënt

- Meer (zelf)vertrouwen (in het resultaat)

… en het is uitdagend en leuk
(zowel voor docent als leerling)

The big 5:

- Algoritmiek: denken in regels en opeenvolgingen (van deeltaken)
- Abstractie: hoog niveau / weglaten details
- Decompositie: opdelen in deelproblemen
- Generalisatie: patronen & generieke opl.
- Evaluatie: voldoet oplossing aan eisen?
  had het beter/makkelijker gekund?

Wat moet je trainen?

- **Vertrouwen** in omgang met **complexiteit**
- **Doorzettingsvermogen** bij het werken aan **moeilijke** problemen
- Vermogen om met **open** problemen om te gaan
- Vermogen om te **communiceren** en werken **met anderen** om een gezamenlijke doel te bereiken

Bron: CAS computational thinking (Humphrey 2015)

Aanpak:

1) **Think first**
   - **Unplugged** enhances thinking-mood and growth mindset
   - Away from distracting computer
   - High-level **flowchart** (abstraction & focus op problem)

2) **Coding** (in Java)
   - Our material (using Greenfoot)
   - Practice meticulous precision & high detail
   - Verifying solution (visual feedback & motivation)

     -> tool for evaluation & reflection

1) Think First! Waarom?

- Flowcharts zijn intuïtief
- Unplugged: Weg van PC
  - nadenken over probleem
  - oplossingsrichtingen worden afgewogen
- Abstractie: niet verzanden in details
- Niet bezig met implementatiedetails en syntax
- Meteen succes ervaring:
  - Makkelijk begin maken
  - ‘iedereen’ kan hun oplossingen verwoorden

2) Coderen! Waarom?

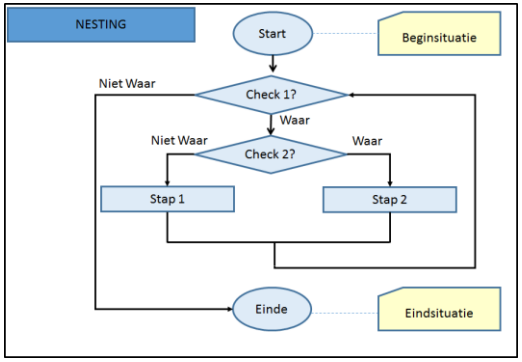
- Oefenen met:
  - Precies werken
  - Hoge mate van details
- Verifiëren van oplossing
  - Visuele feedback
  - Motivatie
  - Evaluatie en reflectie
- Resultaat:
  - Iets MAKEN

Challenges & problems

You must perform two aspects well:

1) Create a *problem-solving algorithm* (a disciplined and creative process)

We use a systematic approach



2) *Formulate* that algorithm *in terms of a programming language* (a disciplined and very precise process)

```
PrimeNumberGenerator.java
1  public class PrimeNumberGenerator {
2      static final int MAX_RANGE= 5000;
3      static final int DEFAULT= 50;
4
5      public static void main (String args[]) {
6          int inputNum= Integer.parseInt(args[0]);
7          if (inputNum < 1 || inputNum > MAX_RANGE) {
8              System.err.println("The number is outside the valid range: " +
9                                  "1 - " + MAX_RANGE);
10             System.err.println("Switching to default: " + DEFAULT);
11             inputNum= DEFAULT;
12         }
13
14         final boolean[] sieve= new boolean[inputNum];
15
16         //for each number between 2 and the square root of the maximum number
17         //inputed by the user, mark all multiples of the number as composite
18         for (int i= 2; i < Math.sqrt(inputNum) + 1; i++) {
19             for (int j = i+i; j < sieve.length; j+= i) {
20                 sieve[j]= true; //this number is composite of 'i'
21             }
22         }
23
24         //output the results
25         for (int i= 2; i < sieve.length; i++) {
26             //if a number has not been marked as composite it is prime
27             if (!sieve[i])
28                 System.out.println(i + " is prime.");
29         }
30     }
31 }
```

Always check that your algorithm is correct by running/testing the implementation!

Wat moet je oefenen?

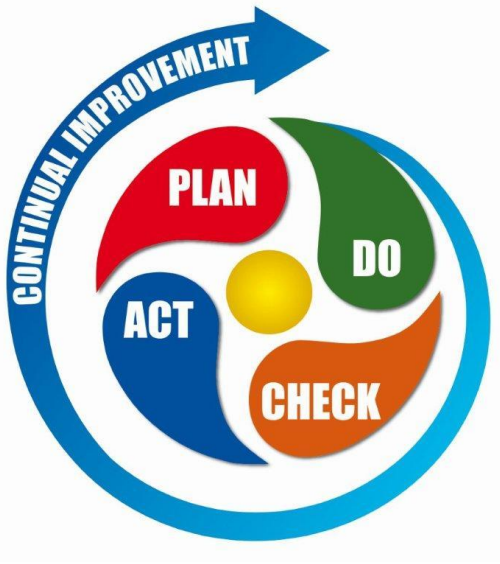
- **Gestructureerd werken**: deelproblemen identificeren en afzonderlijk oplossingen ontwerpen, implementeren en testen
- **Analyseren en redeneren** over kwaliteit van een oplossing (bv efficiëntie)
- **Reflecteren over gekozen oplossing en ontwikkelproces**
- **Generaliseren en hergebruik** van bestaande oplossingen (mogelijk maken), transfer

Gestructureerd werken

Per deelprobleem:

- Algoritme
- Flowchart
- Code implementatie
- Test resultaat
- Reflecteer

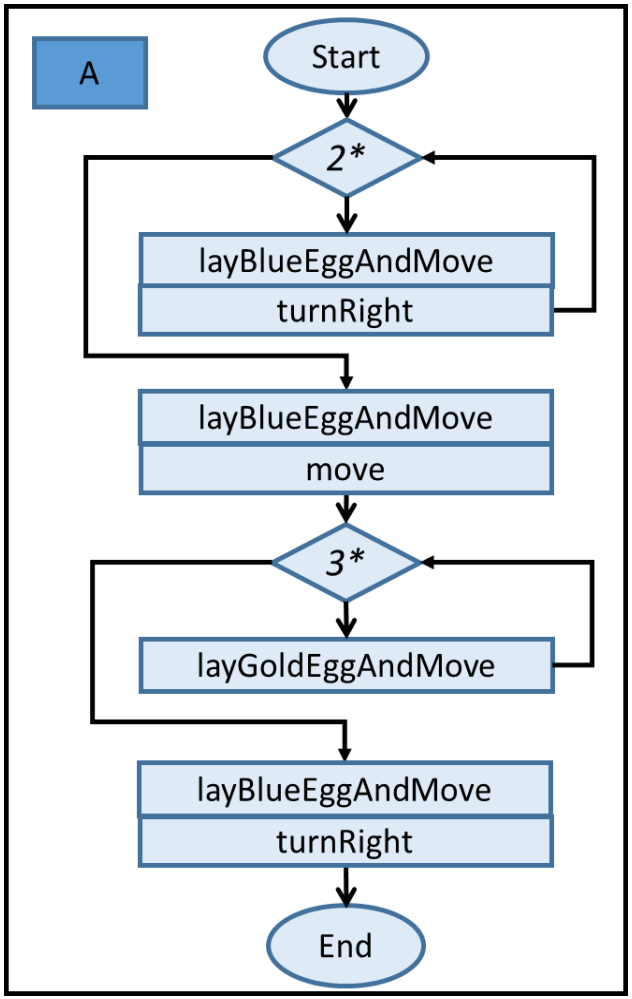
Kleine stapjes



Think first! mbv flowcharts

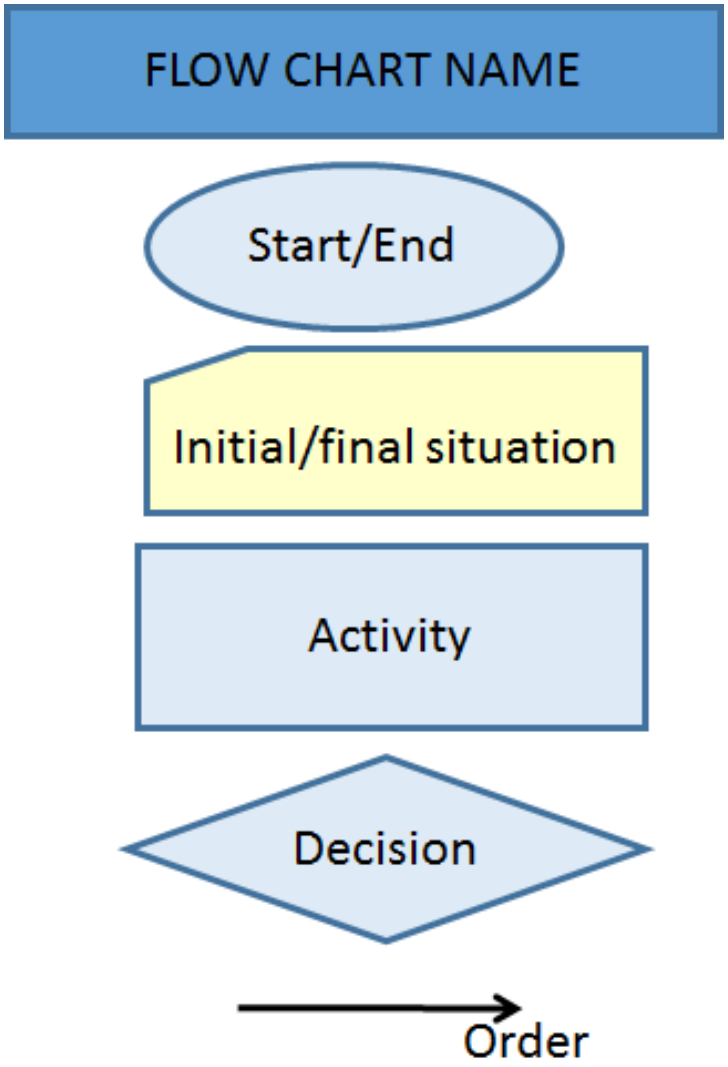
- Hoog niveau beschrijving van oplossing
  - ‘Schetsen’ van algoritme
  - Abstractie
  - Vroegtijdig feedback mogelijk docent-> ll.
- Tijdens implementatie & test:
  - Rode draad
  - Eenvoudig te vertalen naar code
  - Minder trail-and-error debuggen

Flowcharts vs. code



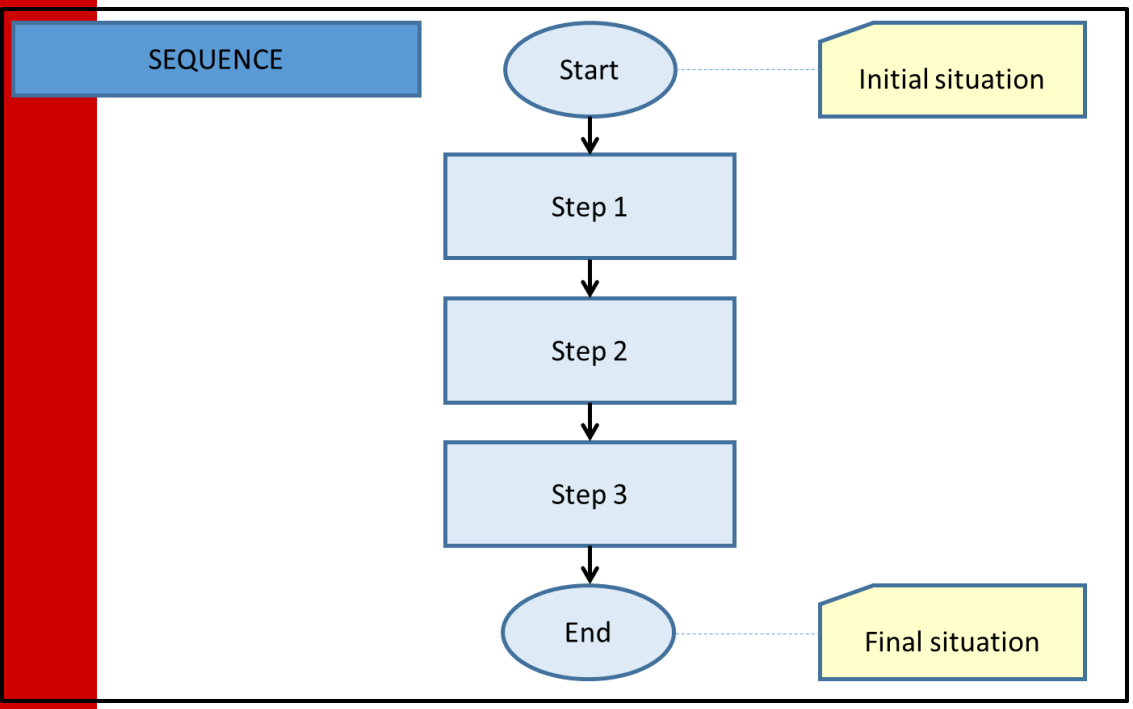

```
private void answerA(){
    for ( int i = 0; i < 2; i++) {
        layBlueEggAndMove();
        turnRight();
    }
    layBlueEggAndMove();
    move();
    for ( int i = 0; i < 3; i++) {
        layGoldEggAndMove();
    }
    layBlueEggAndMove();
    turnRight();
}
```

Flowchart onderdelen



Sequence

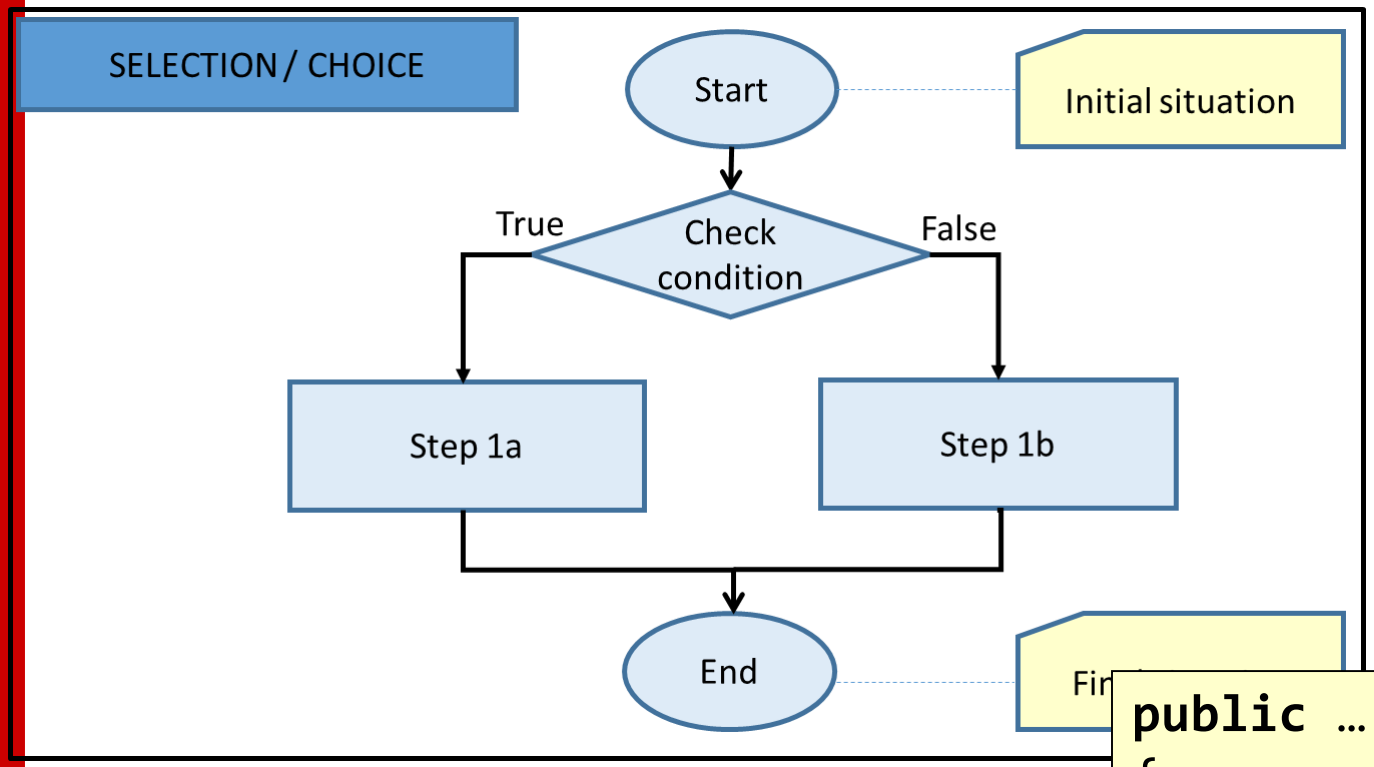
flowchart



code

```
public …
methodName( … ) {
   step1();
   step2();
   step3();
}
```

Selection (choice, if..then..else)

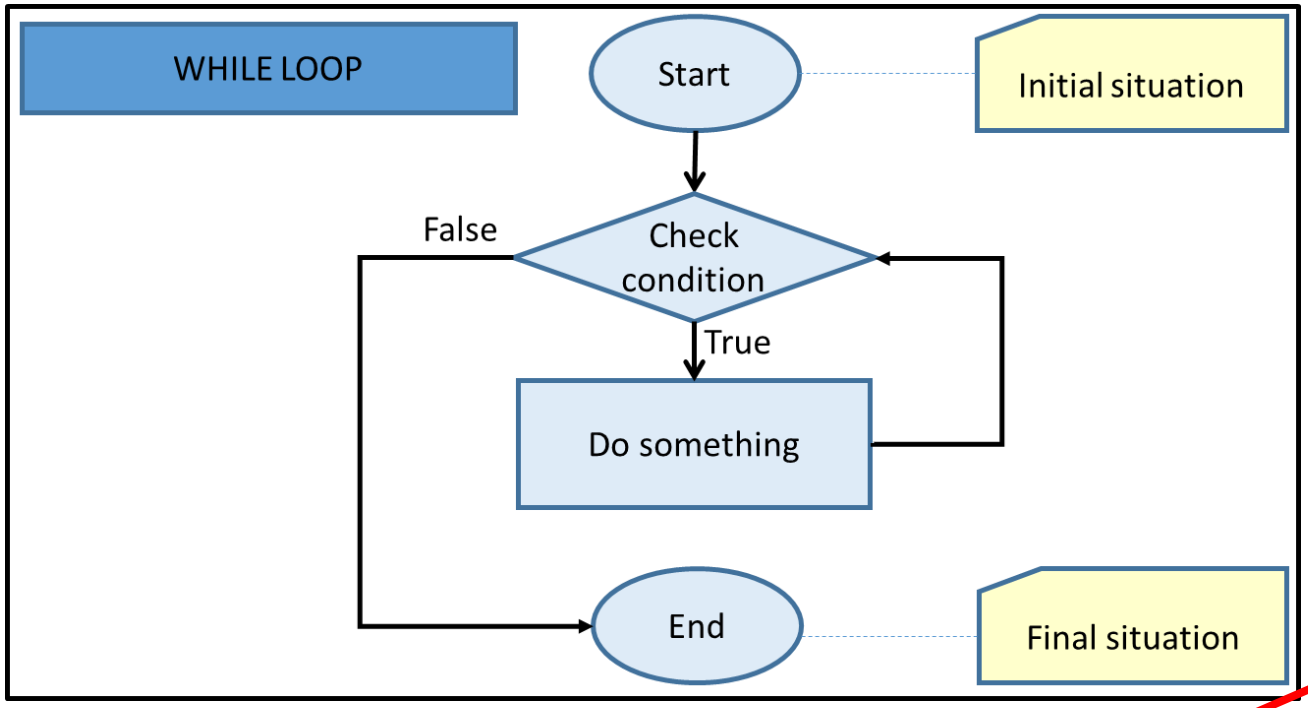


flowchart

code

```
public … methodName( … )
{
   if( check () ) {
      step1a();
   }else{
      step1b();
   }
```

Repetition (iteration, loop)



flowchart

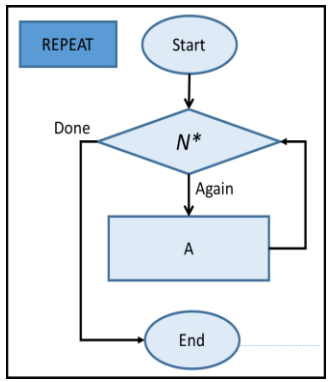
code

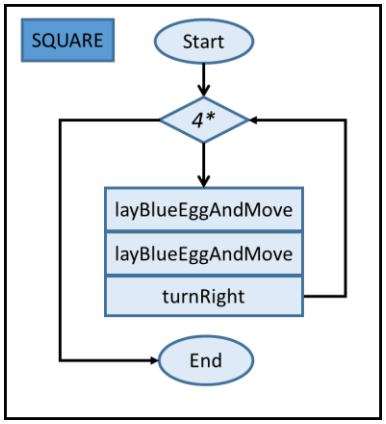
```
public … methodName( … )
{
   while ( check () ) {
      doSomething ();
   }
```

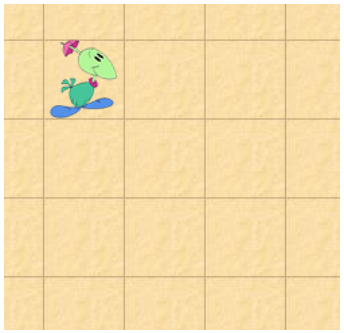
Ervaar het zelf!

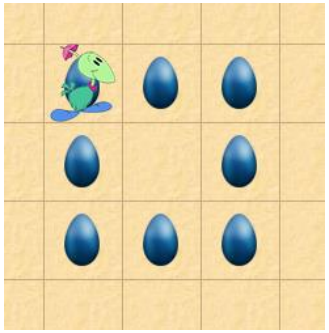
Method name	Method Description
layBlueEggAndMove	Lay a blue egg at current position, and then move forward one cell
layGoldEggAndMove	Lay a golden egg at current position, and then move forward one cell
move	Move forward one cell
turnRight	Turn clockwise 90 degrees

repeat A N times









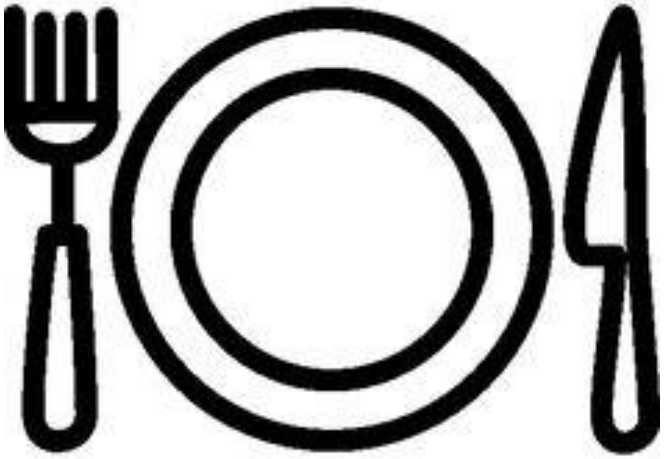
Lesmateriaal

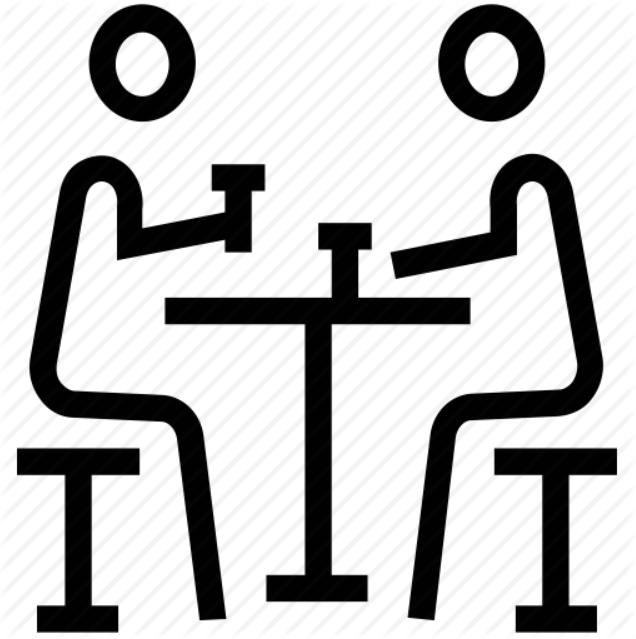
http://course.cs.ru.nl/greenfoot/docenten.html

Klik rechtsboven op 'nascholing'

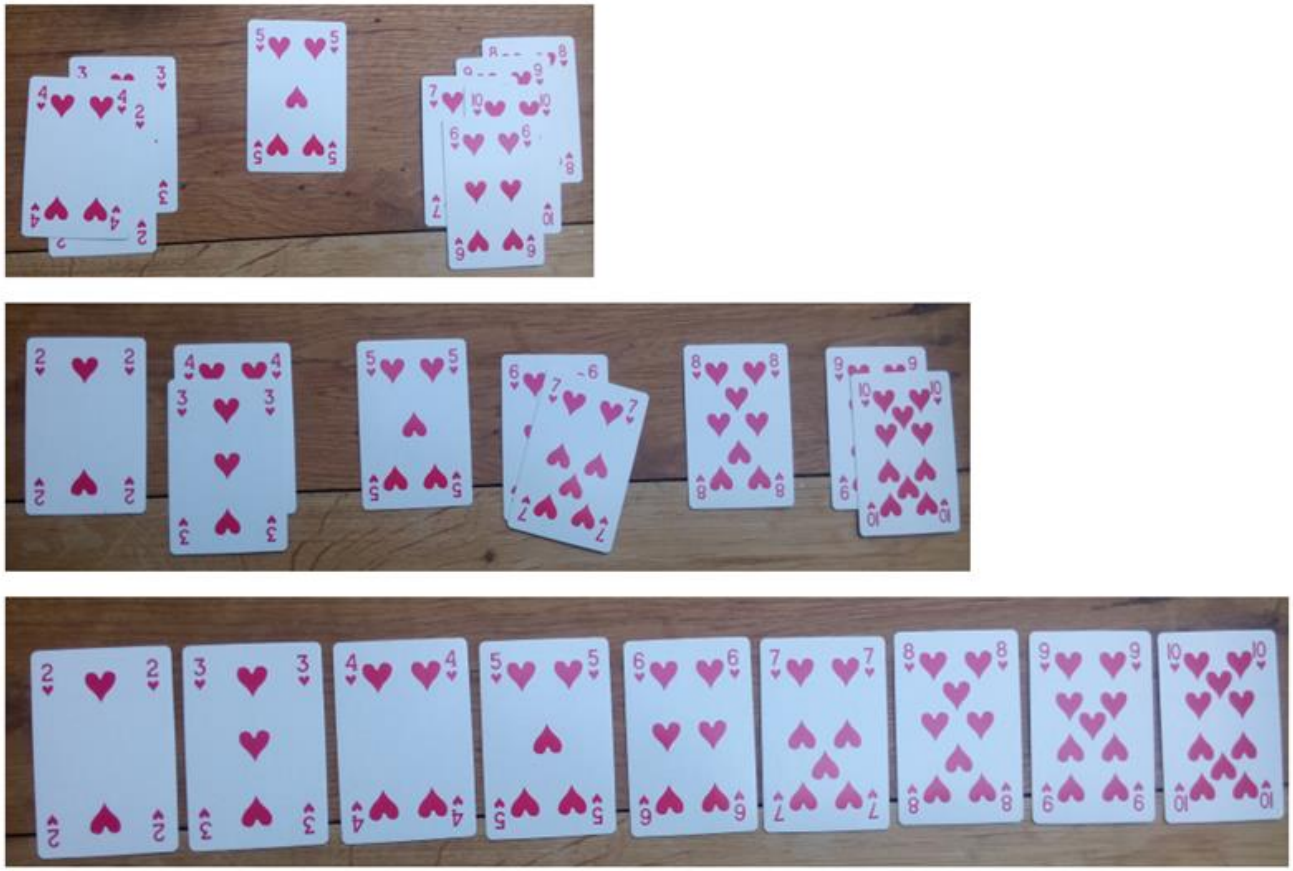
- Nascholing materiaal/bestanden
- Lesmateriaal voor leerlingen
- Lesmateriaal voor docenten

Pauze





Uitdaging: Sorteren



Reflectie

Unplugged sorteren in de klas

- Hoe ga je het brengen?
- Wat heb je daarvoor nog nodig?
- Wat moet je doen? Wat moet je vragen?
- Hoe kunnen wij elkaar ondersteunen?

Samenvattend AD in de les

- Eerst denken (weg van afleidende PC), dan bouwen
  - Algoritme bespreken (unplugged)
  - Algoritme omschrijven (flowchart)
  - Keuzes beoordelen (bespreken/overleggen)
  - Daarna implementeren
    - Submethodes uittesten
    - Geheel testen

Lesmateriaal

http://course.cs.ru.nl/greenfoot

Klik rechtsboven in: 'Naar nascholing'

- Nascholing materiaal/bestanden
- Lesmateriaal voor leerlingen
- Lesmateriaal voor docenten

Masterclass voor leerlingen

- Cursus Computational Thinking
- Doelgroep: 4/5 Vwo/Havo (met wisB)
- Start: 6 april (8 x op donderdag middag)
- Kosten: gratis

- Aanmelden en info:

http://www.ru.nl/pucofscience/scholieren/activiteiten/masterclasses-1617/

Afsluiting

- Opdracht voor de volgende keer: toepassen in de lespraktijk
  - Greenfoot Assignment 1
  - Unplugged sorteren met de leerlingen
  - Ervaringen/observaties vastleggen
- Vooruitblik volgende bijeenkomst:
  - AD en het nieuw examenprogramma (gastspreker Erik Barendsen)
  - Ervaringen terugkoppelen
  - BitParity opdracht

Vooruitblik naar 23 feb

- Theorie koppelen aan praktijk
- Hele ontwikkelcyclus doorlopen
  - Algoritme ontwerpen
  - Flowchart
  - Code
  - Reflectie
- Neem een laptop mee!

Vragen? Opmerkingen?

Tot de volgende keer!

- do 23 februari
  - inloop vanaf 15.30 uur,
  - start 16.00 uur tot 20.00 uur
- do 23 maart
  - inloop vanaf 15.30 uur
  - start 16.00 uur tot 20.00 uur