

IV122 Matematika a programování

Úvod kurzu

Radek Pelánek

Cíle předmětu

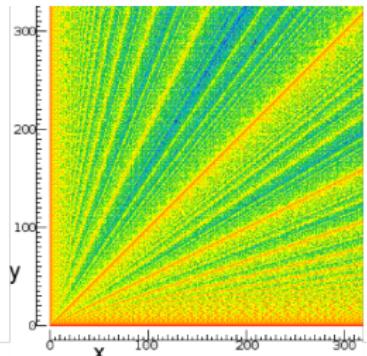
- Lepší pochopení matematických pojmu a metod skrze praktické programování.
- Procvičení programátorských schopností; trénink přechodu od teoretického algoritmu k funkční implementaci.
- Porozumění souvislostem mezi pojmy z různých oblastí.
- Radost z objevování, ocenění elegance, experimentování, informatická kreativita.

O čem předmět není

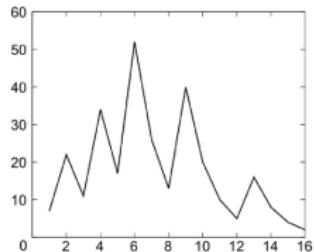
- Úvodní výuka matematických pojmů.
- Úvodní výuka programování.
- Výuka syntaxe konkrétního programovacího jazyka, technické triky, detailly vývojového prostředí, ...

Obsah

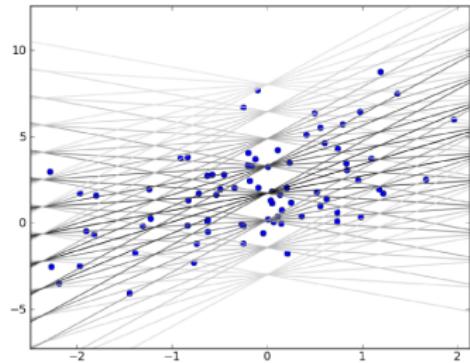
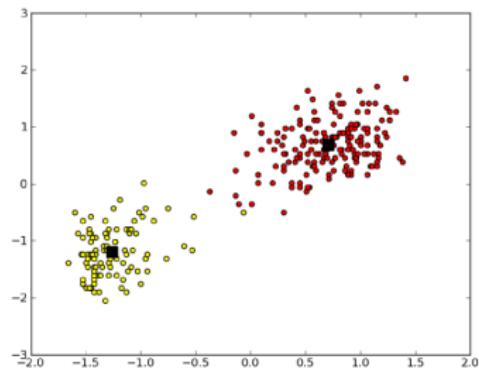
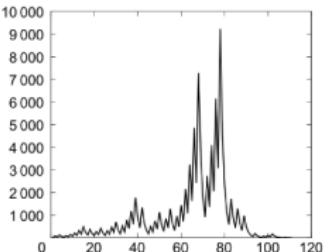
- teorie čísel, kombinatorika
- geometrie, lineární algebra
- fraktály, chaos
- pravděpodobnost, statistika
- grafy, bludiště

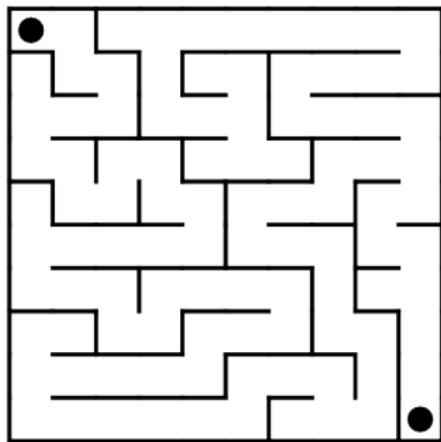
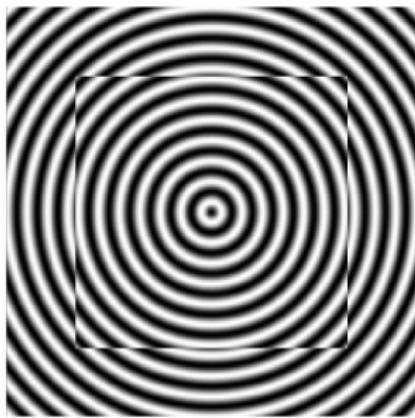
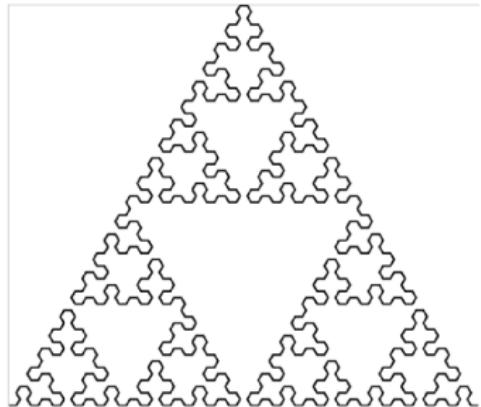
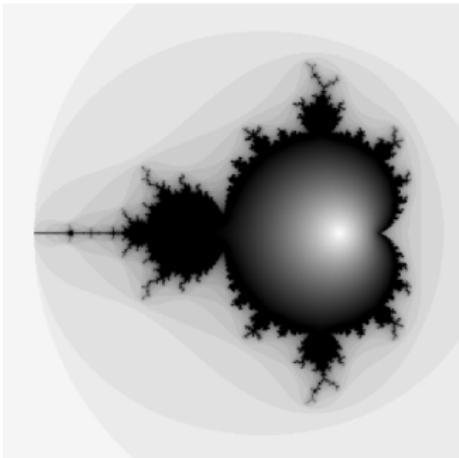


začínající číslem 7



začínající číslem 27





- volně provázaná „přednáška“ a „cvičení“
 - připomenutí matematických pojmu
 - vysvětlení principů použitých v příkladech
 - rady ke struktuře programu, k vhodnému pohledu na problém
 - upozornění na typické chyby
- samostatná práce

důraz na řešení příkladů

Hodnocení předmětu

- domácí úlohy
- účast a aktivita v hodinách
- zkouška
 - „tužka & papír“ – příklady na matematické pojmy
 - programování – variace na úlohy z průběhu semestru
 - diskuze o domácích úlohách

především „slovní“ hodnocení, bez explicitního bodování

Domácí úlohy

- řešeny částečně již v průběhu semináře
- každý týden jedno zadání (většinou obsahuje několik podčástí)
- hodnocení na konci, průběžná orientační kontrola v průběhu semestru

Portfolio domácích úloh

- obsah:

- všechny výtvory relevantní k předmětu – programy z hodin, domácí úkoly
- zdrojové kódy, výsledky experimentů, vytvořené obrázky, komentáře, ...

- realizace:

- vlastní webová stránka – IS / Můj web / IV122, tj.
<http://is.muni.cz/www/UC0/IV122>
- může být kdekoliv jinde (např. blog, autorizovaný web, github) ⇒ na uvedenou adresu dát odkaz

Kritéria hodnocení – domácí úlohy

- nezbytné:
 - z každého zadání alespoň jedna podúloha
 - celkově vypracováno alespoň 75 %
 - alespoň jednou výraznější bonus: vlastní rozšíření úlohy, netypické pojetí, ...
- hodnocení A: ± vše a kreativita

Opisování

- zcela samostatně – žádné přebírání kódu od kolegů
- inspirace na webu
 - obecně nedělat, cíl je psát zcela samostatně
 - výjimečně může být smysluplné část kódu převzít a rozšířit ⇒ jasně označit převzatý kód

očekávám jednotnost zpracování (např. volba jazyka) – výjimky osvětlit

Matematika – předpoklady a poznámky

- znalosti na úrovni bakalářské matematiky na FI
- bude stručné připomenutí pojmu
- dílčí neznalosti je reálné doučit se za běhu
- zdroje např: knihovna, Wikipedia, Khan Academy

Matematika – příklady potřebných pojmů

- prvočísla, dělitelnost, mod
- log, exp, sin, cos
- vektor, matice, affinní transformace
- pravděpodobnost, normální distribuce
- graf, cesta, kostra

Programovací jazyk

- můžete pracovat v libovolném jazyce a prostředí
- problémy se syntaxí musíte zvládnout samostatně
- žádné sofistikované knihovny – budeme sami implementovat „základy“
- doporučuji Python (reálné i pokud ho zatím neznáte)

Kultura programování

... se bere v potaz, např:

- volba reprezentace dat
- názvy funkcí a proměnných
- smysluplné komentáře
- dekompozice problému na funkce
- nepoužívání „copy & paste kódu“

Samostatnost, prezentace

- kreativita a samostatnost vítána (a alespoň trochu očekávána)
 - rozšíření zadaných úloh
 - **vlastní drobné variace** (ne ulehčení úlohy, ale kreativní prozkoumání)
 - hrátky s barvami, experimenty s parametry, pokusy „co se stane když“, alternativní algoritmy, ...
- prezentace výtvorů
 - přehlednost
 - výstižné popisky
 - **dobré vyjádření hlavní myšlenky**

Náročnost předmětu

- předmět není těžký, ale je časově náročný
- **velmi** vhodné řešit úlohy průběžně

- **vysoko-úrovňový** – velká míra abstrakce, „spustitelný pseudokód“
- **interpretovaný** – pomalejší než komplikovaný, ale větší volnost
- **pedagogický** – byl tak navržen
- **moderní a široce používaný**
- volně a snadno **dostupný** na všech platformách

Python: velmi rychlé uvedení

- dynamicky typovaný
- bílé znaky důležité – vyznačení bloků
- Python 2.7 vs 3
- zdroje pro naučení např.:
 - <http://python.cz/>
 - <http://howto.py.cz/index.htm>

Grafika

Mnoho příkladů vede na grafické znázornění, budou nám stačit základní operace:

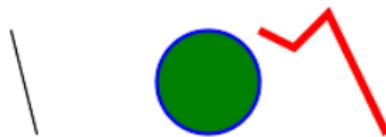
- vektorová grafika
 - `line(x1, y1, x2, y2)`, `circle(x, y, r)` + barvy, výplň
 - doporučeno SVG
- bitmapová grafika
 - `put_pixel(x, y, color)`
 - Python: knihovna Image
 - ostatní jazyky: canvas, vhodná knihovna
- vykreslování základních grafů: liniový, bodový (scatter), histogram

Scalable Vector Graphics (SVG)

- vektorový formát založený na XML
- snadný způsob vytváření obrázků v jakémkoliv jazyce (generujeme prostý text)
- prohlížení: např. webový prohlížeč
- ruční editování: např. Inkscape
- převod na bitmapu: např. convert (ImageMagick)

SVG příklad

```
<svg xmlns="http://www.w3.org/2000/svg">
<line x1="15" y1="20" x2="30" y2="80"
      stroke="black" stroke-width="1"/>
<circle cx="130" cy="50" r="30" stroke="blue"
       stroke-width="2" fill="green" />
<polyline fill="none" stroke="red" stroke-width="4"
           points="160,20 180,30 200,10 234,80"/>
</svg>
```

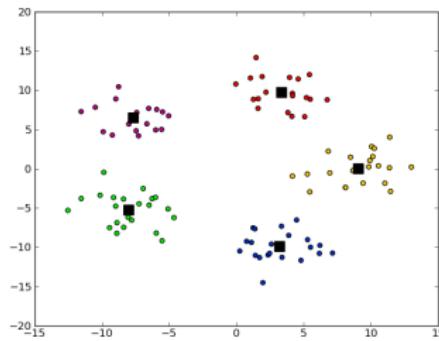
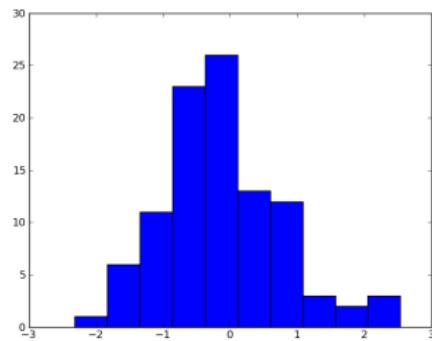
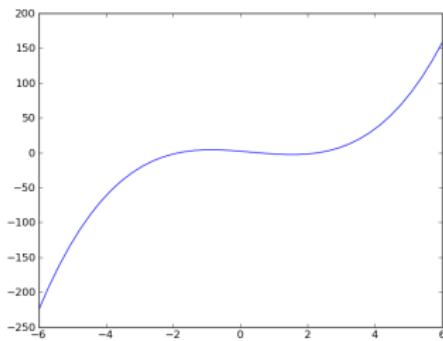


Bitmapová grafika: příklad Python

```
from PIL import Image

def disc(size=150, r=50):
    im = Image.new("RGB", (size, size))
    for x in range(size):
        for y in range(size):
            if (x-size/2)**2 + (y-size/2)**2 < r**2:
                im.putpixel((x, y), (0, 0, 0))
            else:
                im.putpixel((x, y), (255, 255, 255))
    im.show()
```

Vykreslování základních grafů



Python, NumPy, matplotlib

```
import pylab as plt
import numpy as np

def demo_hist(n):
    values = np.random.randn(n)
    plt.hist(values)
    plt.show()

def demo_lines():
    x = np.linspace(-6, 6, 40)
    plt.plot(x, x**3 - x**2 - 4*x + 2)
    plt.show()
```

Rozcvička: počítání s čísly

Radek Pelánek

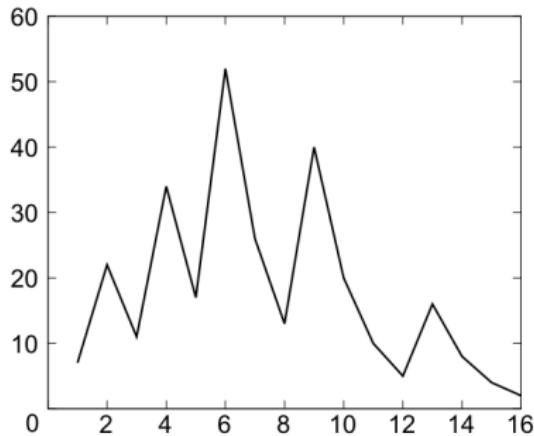
IV122

Collatzova posloupnost

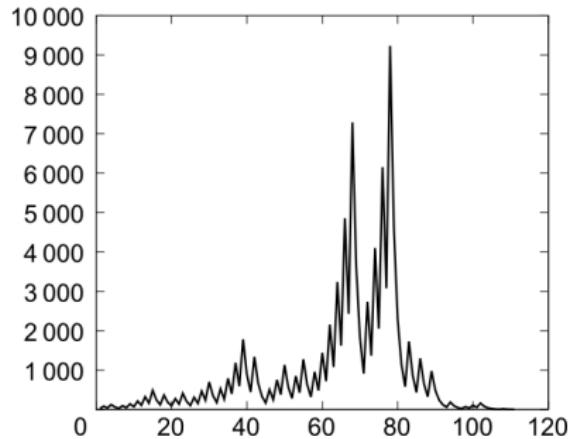
- vezmi přirozené číslo:
 - pokud je sudé, vyděl jej dvěma
 - pokud je liché, vynásob jej třemi a přičti jedničku
- tento postup opakuj, dokud nedostaneš číslo jedna

Collatzova posloupnost: příklady graficky

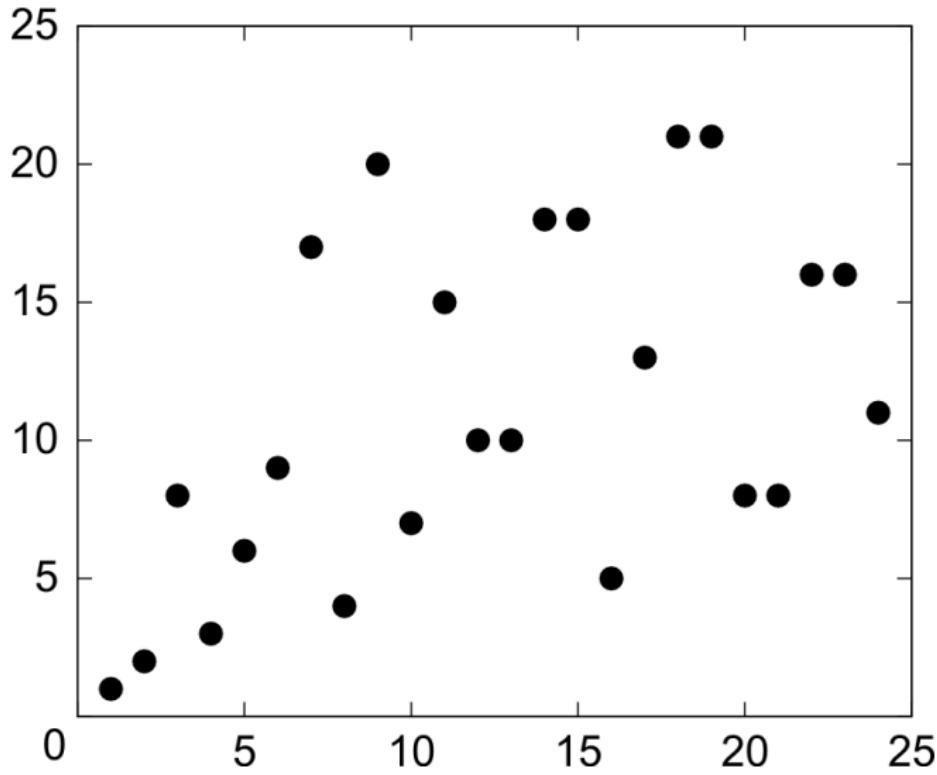
začínající číslem 7



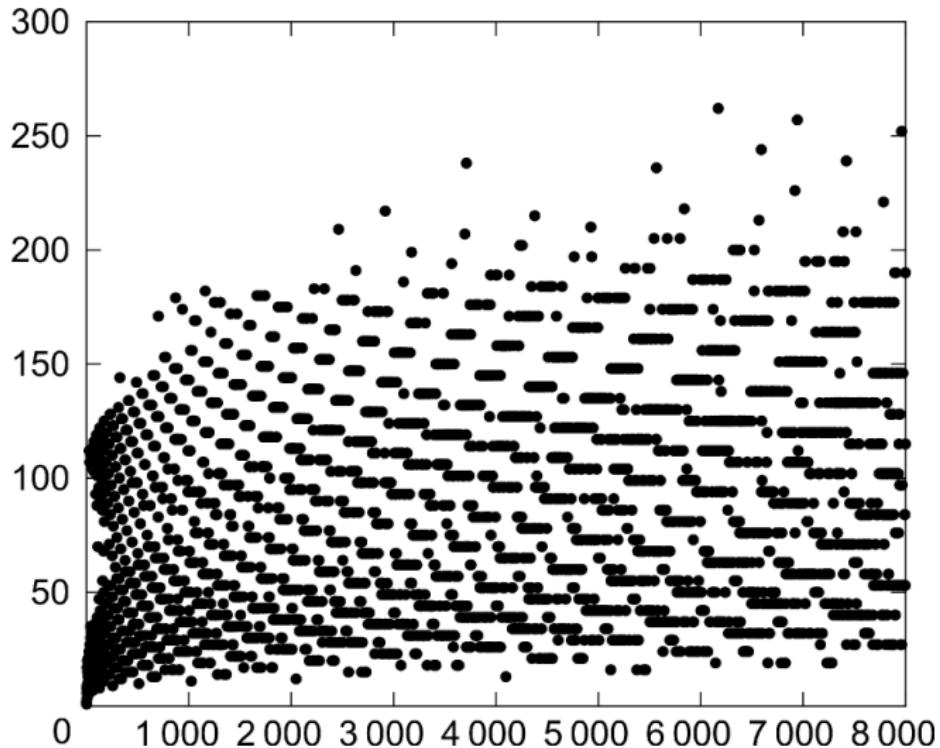
začínající číslem 27



Collatzova posloupnost: počty kroků I



Collatzova posloupnost: počty kroků II



Collatzova hypotéza

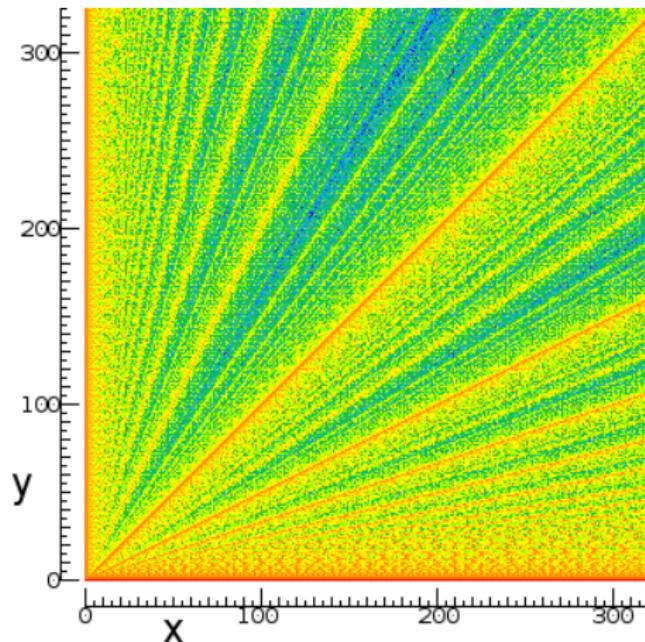
- platí, že pro každé počáteční číslo n , narazí posloupnost na číslo 1?
- experimentálně ověřeno pro velká n ($\sim 10^{18}$)
- důkaz není znám

Největší společný dělitel

- Euklidův algoritmus
- $NSD(a, b) = NSD(a - b, b)$
- $NSD(a, b) = NSD(a \bmod b, b)$

```
def nsd(a,b):  
    if b == 0:  
        return a  
    else:  
        return nsd(b, a % b)
```

Euklidův algoritmus: vizualizace



http://en.wikipedia.org/wiki/Euclidean_algorithm

Úkol: Vizualizace NSD, Euklidův algoritmus

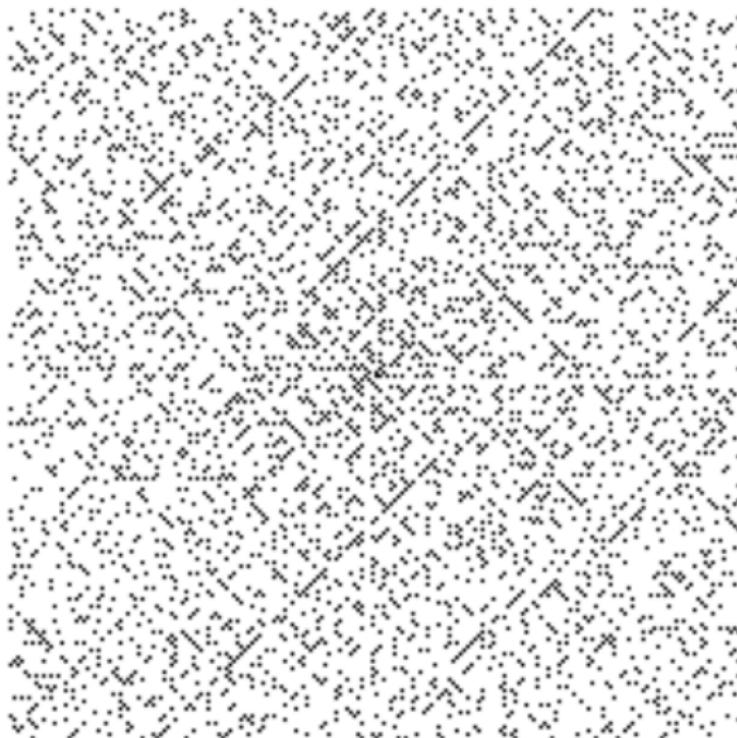
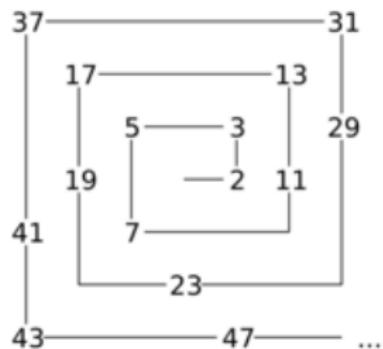
A) Program vygeneruje obrázek vizualizující největší společné dělítel.

B) Program vygeneruje obrázek vizualizující délku běhu Euklidova algoritmu:

- počet kroků algoritmu – odčítací varianta
- počet kroků algoritmu – efektivní modulo varianta
- různé způsoby barevného znázornění (např. kombinace obou předchozích do jednoho obrázku)

Ulamova spirála

37–36–35–34–33–32–31
38 17–16–15–14–13 30
39 18 5–4–3 12 29
40 19 6 1–2 11 28
41 20 7–8–9–10 27
42 21–22–23–24–25–26
43–44–45–46–47–48–49...



Ulamova spirála – variace

- Jak to dopadne, když místo prvočísel budeme do spirály zakreslovat čísla dělitelná k ?
- Jaká jiná kritéria můžeme použít? Barevné barvování?
- Co když použijeme jiný tvar mřížky?

Kombinatorika, výpočty

Radek Pelánek

IV122

Styl

- jednoduché výpočty s čísly
- zatím spíše opakování + pár dílčích zajímavostí
- užitečný trénink programování (rekurze)
- experimentování

Kombinace, permutace, variace

Daná množina M s n prvky

- ① permutace = ...
- ② k prvkové kombinace = ...
- ③ k prvkové kombinace s opakováním = ...
- ④ k prvkové variace = ...
- ⑤ k prvkové variace s opakováním = ...

Kombinace, permutace, variace

Daná množina M s n prvky

- ① **permutace** = uspořádání zadaných prvků do fixního pořadí
- ② k prvkové **kombinace** = všechny možné výběry k prvků ze zadané množiny
- ③ k prvkové **kombinace s opakováním** = všechny možné výběry k prvků ze zadané množiny, přičemž prvky se mohou opakovat
- ④ k prvkové **variace** = všechny možné uspořádané výběry k prvků ze zadané množiny
- ⑤ k prvkové **variace s opakováním** = všechny možné uspořádané výběry k prvků ze zadané množiny, přičemž prvky se mohou opakovat

Kombinace, permutace, variace – příklady

Úloha	Vstup	Výstup
permutace	A, B, C	ABC, ACB, BAC, BCA, CAB, CBA
kombinace $k = 2$	A, B, C, D	AB, AC, AD, BC, BD, CD
kombinace s opakováním $k = 2$	A, B, C, D	AA, AB, AC, AD, BB, BC, BD, CC, CD, DD
variace $k = 2$	A, B, C, D	AB, AC, AD, BA, BC, BD, CA, CB, CD, DA, DB, DC
variace s opakováním $k = 2$	A, B, C	AA, AB, AC, BA, BB, BC, CA, CB, CC

Kombinace, permutace, variace – počty prvků

Počet všech

- permutací n prvků = ...
- k prvkových kombinací z n = ...
- k prvkových kombinací s opakováním z n prvků = ...
- k prvkových variací z n prvků = ...
- k prvkových variací s opakováním z n prvků = ...

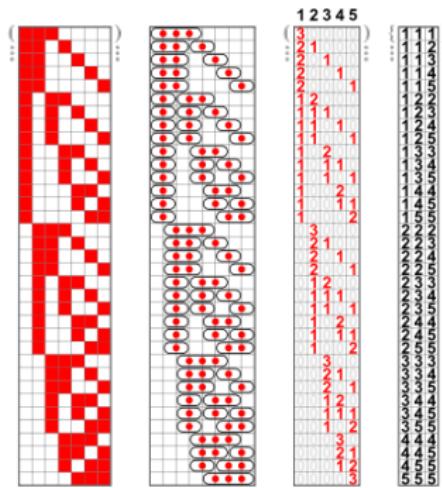
Kombinace, permutace, variace – počty prvků

Počet všech

- permutací n prvků = $n!$
- k prvkových kombinací z n = $\binom{n}{k} = \frac{n!}{(n-k)!k!}$
- k prvkových kombinací s opakováním z n prvků = $\binom{n+k-1}{k}$
- k prvkových variací z n prvků = $\frac{n!}{(n-k)!}$
- k prvkových variací s opakováním z n prvků = n^k

Kombinace s opakováním – ilustrace

3 prvkové kombinace s opakováním z 5 prvků $\sim \binom{5+3-1}{3} \sim \binom{7}{3}$



<https://en.wikipedia.org/wiki/Combination>

Úkol: generování kombinací, permutací, variací

- Vstup: množina (seznam) a případně k
 - Výstup: (uspořádaný) výpis všech permutací/kombinací/variací (s opakováním)
-
- vede na přirozené využití **rekurze**
 - myšlenkově podobné \Rightarrow programy by měly být podobné

Výpočet kombinačního čísla

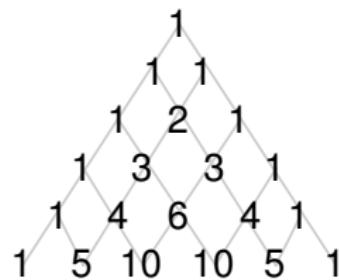
$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

```
def comb_number(n, k):
    if k == 0 or k == n:
        return 1
    else:
        return comb_number(n-1, k-1) + \
            comb_number(n-1, k)
```

Výpočet kombinačního čísla

- neefektivní – opakované výpočty
- podobné jako klasická ukázka neefektivního použití rekurze u Fibonacciho čísel
- efektivněji:
 - explicitní vztah
 - počítání „od spodu“

Pascalův trojúhelník



$$\begin{aligned} & \binom{0}{0} \\ & \binom{1}{0} \quad \binom{1}{1} \\ & \binom{2}{0} \quad \binom{2}{1} \quad \binom{2}{2} \\ & \binom{3}{0} \quad \binom{3}{1} \quad \binom{3}{2} \quad \binom{3}{3} \end{aligned}$$

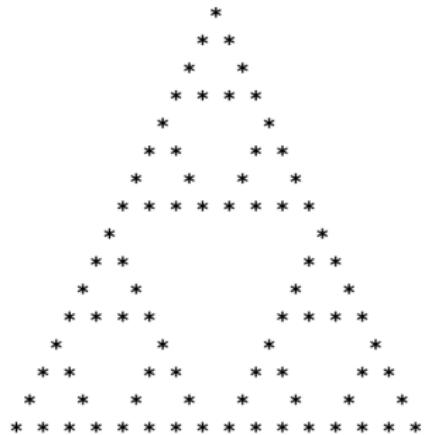
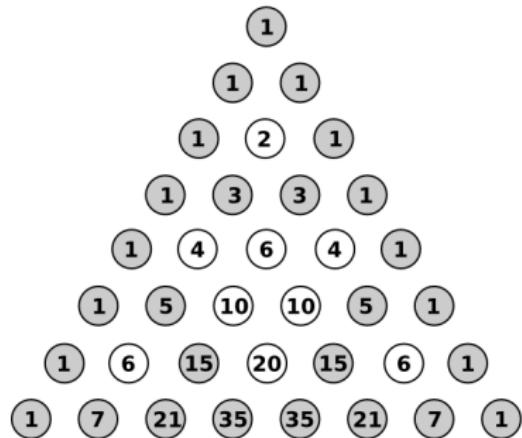
Explicitní vzorec

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}$$

Rekurzivní vztah

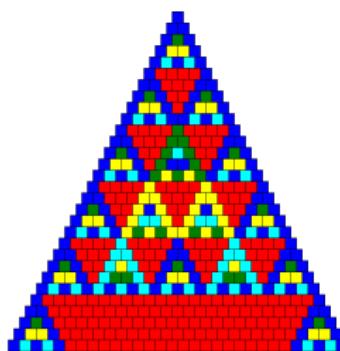
$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

Pascalův trojúhelník a Sierpiňského fraktál



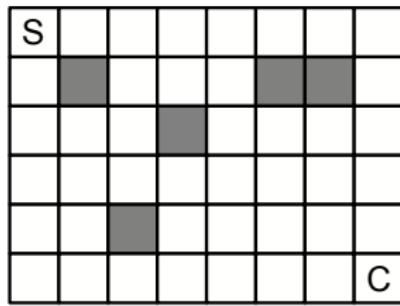
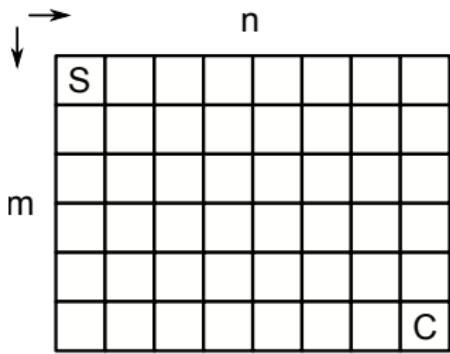
Obarvování čísel: Pascal a Ulam

- video Vi Hart: Sick Number Games
http://www.youtube.com/watch?v=Yhlv5Aeuo_k
- obarvování Pascalova trojúhelníku modulo k
- vztah k jednorozměrným buněčným automatům



Rada: pozor na „přetečení“

Počítání cest



Umocňování

$$x^y$$

- x, y – kladná čísla (ne nutně celá)
- např. $3, 45^{2,3}$
- co to vlastně znamená?
- jak vypočítat?
přibližná hodnota, jen pomocí základních aritmetických operací

Umocňování: úkol

$$x^y$$

- vypočítat přibližnou hodnotu, jen pomocí základních aritmetických operací
- stačí jednoduché metody
- experimentálně prozkoumat chování: rychlosť, přesnosť

Efektivní umocňování

$$a^n \bmod k$$

- a, n, k – přirozená čísla
- n může být „velké“ (stovky cifer)
- jak vypočítat efektivně? (lépe než lineárně vzhledem k n)
- aplikace např. v kryptologii

Výpočet π

- $\pi = 3,141592653589793\dots$
- iracionální číslo
- známé s přesností na miliardy cifer
- jak se určuje hodnota π ?
- zmíníme jen velmi naivní metody – přímočaré cvičení na „experimentální porovnání“

Výpočet π

Gregoryho-Leibnizova řada (součet je π):

$$4 \cdot \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \dots$$

(Důkaz: $\arctan(1)$, integrál)

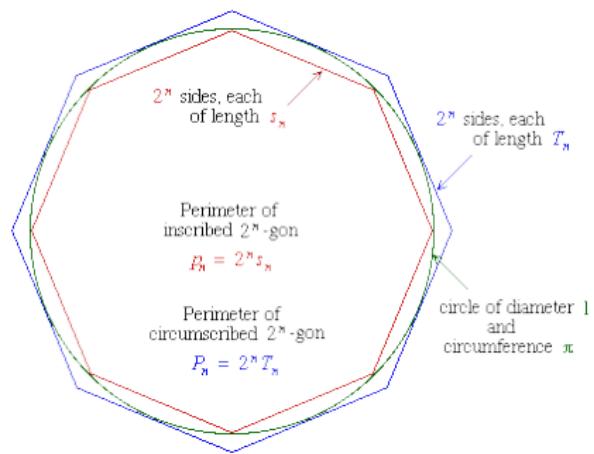
Výpočet π

Archimedova metoda (dvě posloupnosti a_n, b_n společně konvergující k π)

$$a_0 = 2\sqrt{3}; b_0 = 3$$

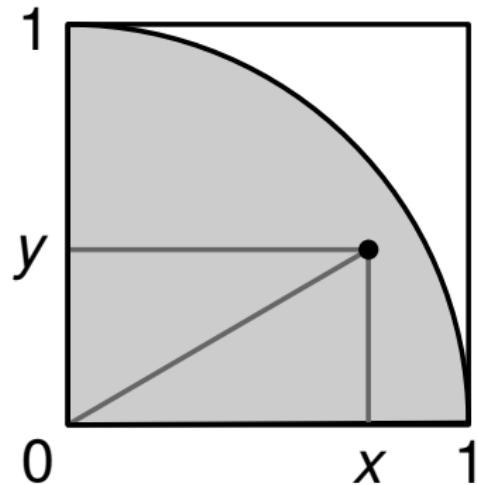
$$a_{n+1} = \frac{2a_n b_n}{a_n + b_n}$$

$$b_{n+1} = \sqrt{a_{n+1} b_n}$$



<http://personal.bgsu.edu/~carother/pi/Pi3a.html>

Výpočet π – Monte Carlo



- obsah čtvrtdisku: $\pi/4$
- obsah čtverce: 1

Úkol: Výpočet π

- implementujte uvedené metody
- (najděte další metody a implementujte je)
- experimentálně vyhodnoťte a porovnejte jednotlivé metody
- co je férové porovnání?

Úkol: Výpočet π

- implementujte uvedené metody
- (najděte další metody a implementujte je)
- experimentálně vyhodnoťte a porovnejte jednotlivé metody
- co je férové porovnání?
- jaké přesnosti jsou schopny dosáhnout během 1 vteřiny?

Umocňování: rady

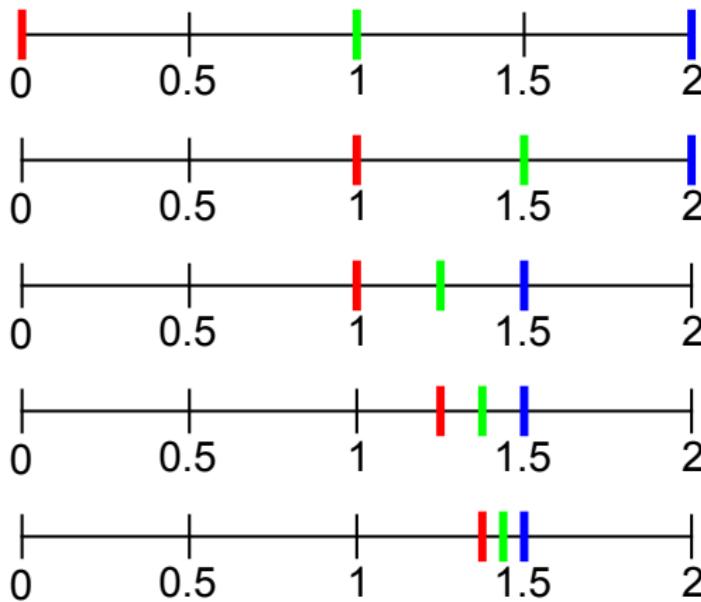
$$x^{a/b} = \sqrt[b]{x^a}$$

výpočet odmocniny:

- vstup: číslo x
- výstup: přibližná hodnota \sqrt{x}
- základní metoda: binární půlení (rozhodně ne nejvíce efektivní)

Výpočet odmocniny: binární půlení

spodní odhad střed horní odhad



Umocňování a Taylorova řada

Taylorova řada:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!} (x - x_0)^n$$

Pro $f(x) = x^k$ a $x_0 = 1$ lze snadno vypočítat.

Geometrie 1: úhly, goniometrické funkce, želví grafika

Radek Pelánek

IV122

Vektorová grafika

dnes jediná základní operace: vykreslit úsečku

minimalistická realizace v SVG:

```
<svg xmlns="http://www.w3.org/2000/svg">  
  <line x1="15" y1="20" x2="30" y2="80" stroke="black"/>  
</svg>
```

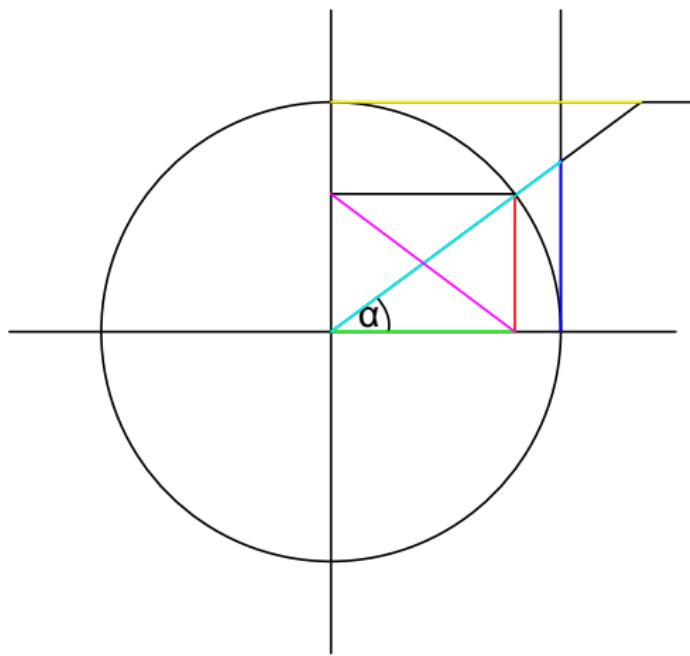
Jednotková kružnice a goniometrické funkce

$\sin \alpha$

$\cos \alpha$

$\tan \alpha$

$\cotan \alpha$



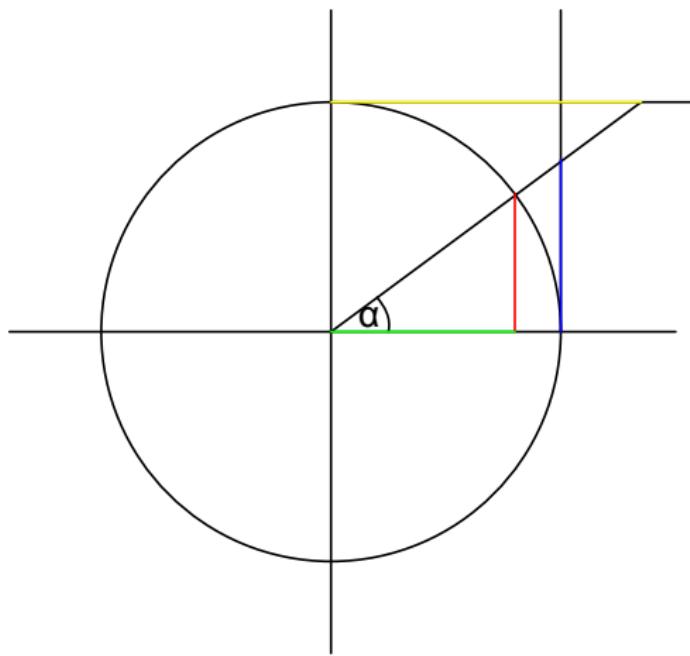
Jednotková kružnice a goniometrické funkce

$\sin \alpha$

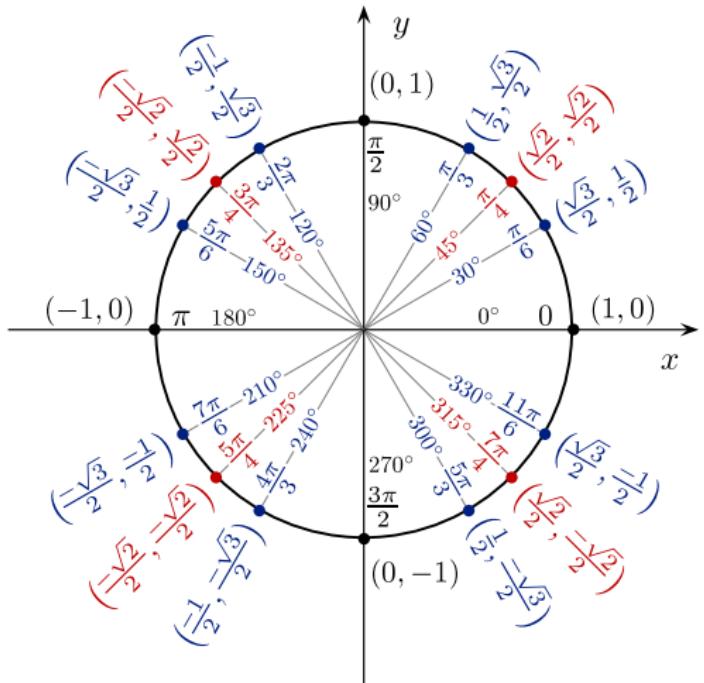
$\cos \alpha$

$\tan \alpha$

$\cotan \alpha$

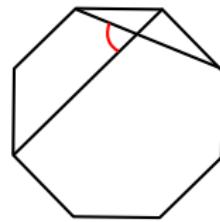
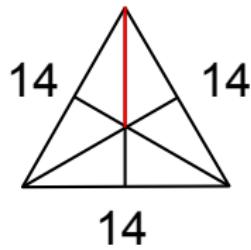
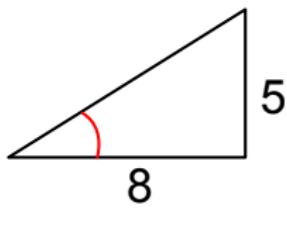


Jednotková kružnice



Zdroj: Wikipedia

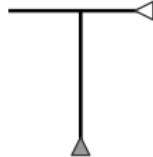
Určete úhly, délku



Želví grafika

- forward, back – posun dopředu a dozadu o zadanou vzdálenost
- left, right – otočení o zadaný úhel
- penup, pendown – zvednutí a položení pera

```
forward 80  
left 90  
forward 40  
back 80
```



```
forward 80  
right 135  
penup  
forward 30  
pendown  
forward 30
```



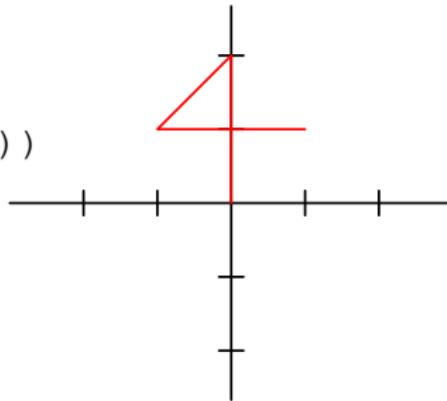
Úkol: Knihovna pro želví grafiku

Vytvořte vlastní knihovnu pro práci s želví grafikou.

- přirozená objektová realizace (třída Turtle)
- metody pro pohyb: `forward(step)`, `right(angle)`,
`penup()`, ...
- uložení obrazce do SVG souboru

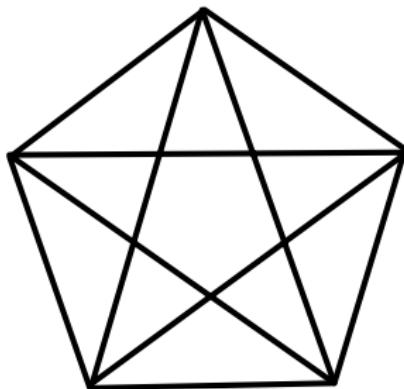
Relativní vs. absolutní vykreslování

```
forward(2)
left(135)
forward(sqrt(2))
left(135)
forward(2)
```

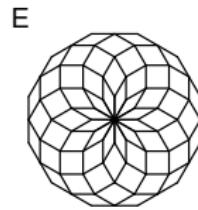
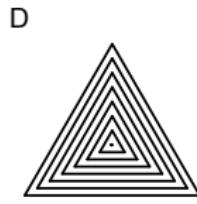
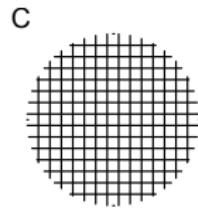
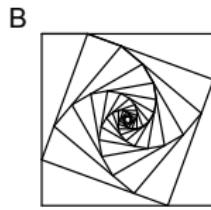
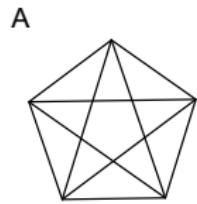


```
line(0,0,0,2)
line(0,2,-1,1)
line(-1,1,1,1)
```

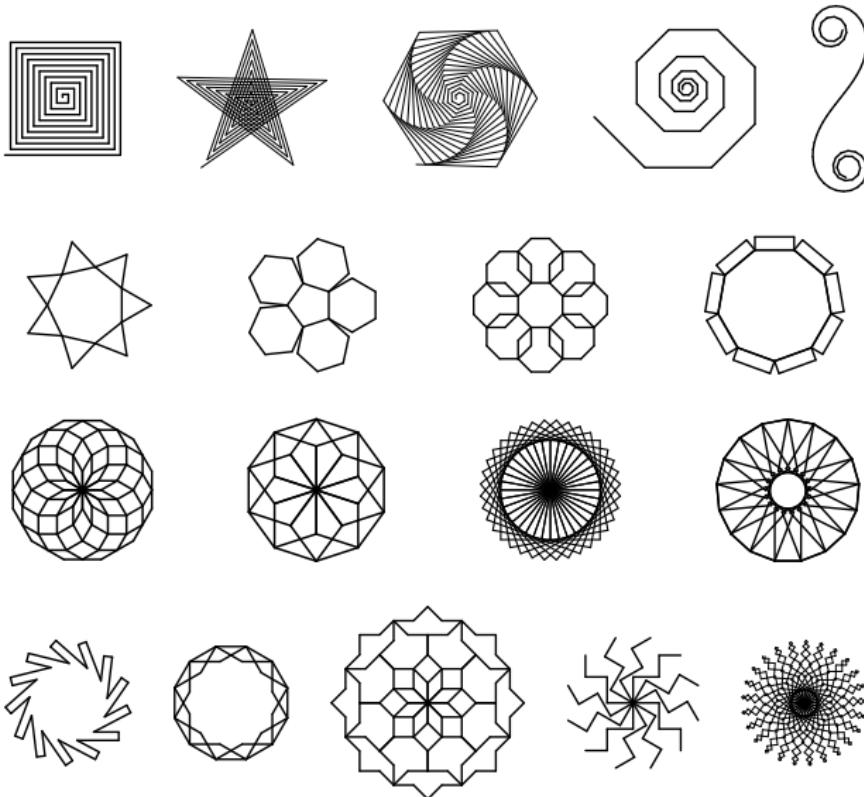
Relativní vs. absolutní vykreslování



Relativní vs. absolutní vykreslování

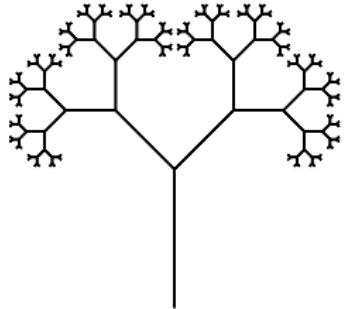


Želví grafika – inspirace

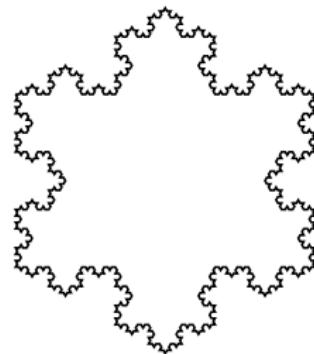


Želví grafika – fraktály

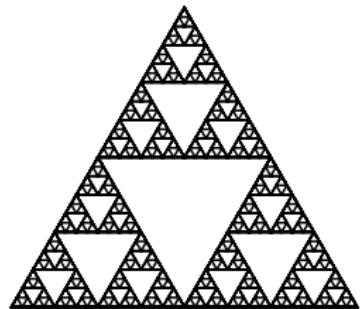
Ker



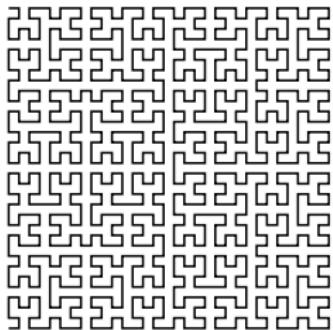
Kochova vločka



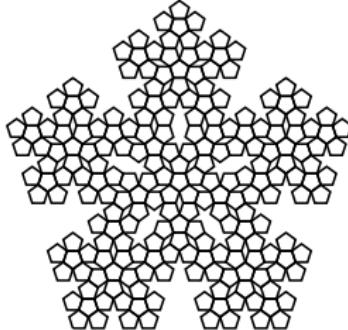
Sierpińského trojúhelník



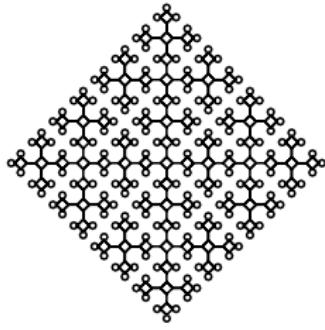
Hilbertova křivka



Pětiúhelníková vločka



Krishna Anklet



Implementační poznámky

- minimalizujte „kouzelné konstanty“ uprostřed kódu, používejte spíše:
 - pojmenované parametry (length)
 - konstanty (pi)
 - symbolický zápis ($\sqrt{3}/2$, $\cos(\pi/6)$ vs. 0.866)
- 3 a více velmi podobných řádků \Rightarrow zobecnit (typicky vhodný for cyklus)

Geometrie 2: bitmapová grafika a rovnice

Radek Pelánek

IV122

Implicitní a parametrická rovnice

- implicitní rovnice:
 - $f(x, y) = 0$
- parametrické rovnice:
 - $x = f_1(t)$
 - $y = f_2(t)$

Implicitní a parametrická rovnice

- přímka:
 - implicitní rovnice: $ax + by + c = 0$
 - parametrické rovnice:
 - $x = t$
 - $y = at + b$
- kružnice (střed v bodě (a, b) , poloměr r):

Implicitní a parametrická rovnice

- přímka:
 - implicitní rovnice: $ax + by + c = 0$
 - parametrické rovnice:
 - $x = t$
 - $y = at + b$
- kružnice (střed v bodě (a, b) , poloměr r):
 - implicitní rovnice: $(x - a)^2 + (y - b)^2 = r^2$
 - parametrické rovnice ($t \in [0, 2\pi]$):
 - $x = a + r \cos(t)$
 - $y = b + r \sin(t)$

Rovnice a vykreslování

přímočaré vykreslování:

- parametrická rovnice:

```
for t in suitable interval:  
    x = fx(t), y = fy(t)  
    putpixel(x, y)
```

- implicitní rovnice:

```
forall x:  
    forall y:  
        if f(x, y) == 0:  
            putpixel(x, y)
```

Implicitní rovnice

- polorovina
- čtverec
- trojúhelník
- kruh
- elipsa

Bitmapová grafika

- černo-bílé vykreslování základních obrazců (kruh, trojúhelník, elipsa ...)
 - implicitní / parametrické rovnice
- optické efekty
 - využití logické funkce XOR
- vykreslování obrázků barevně / s odstíny (vlny, ...)
 - využití goniometrických funkcí

Základní tvary

čtverec



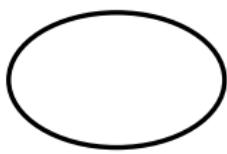
trojúhelník



kruh



elipsa



spirála



Základní tvary barevně

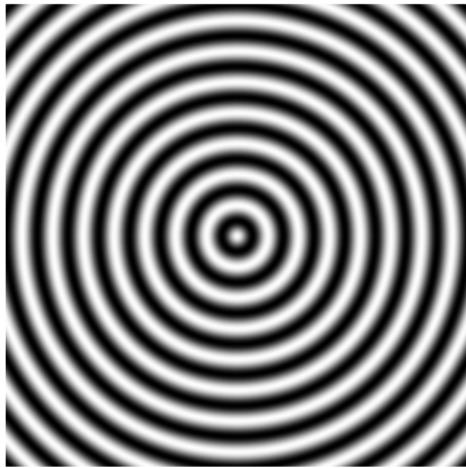
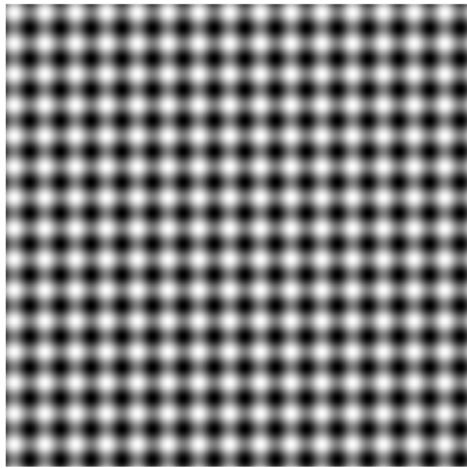


Vykreslení mnohoúhelníku

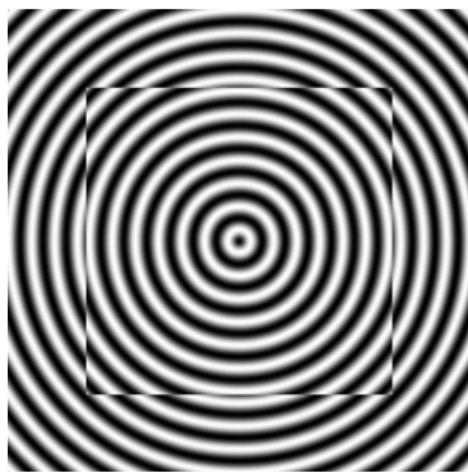
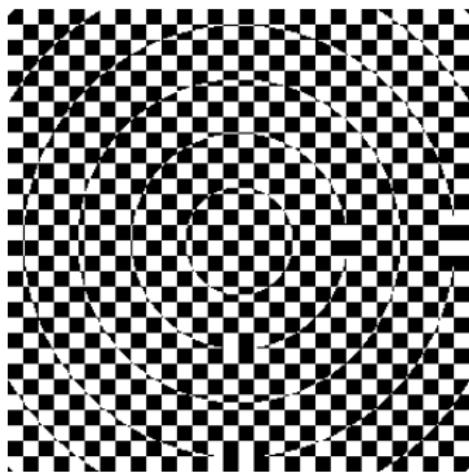
```
[(10, 10), (180, 20), (160, 150), (100, 50), (20, 180)]
```



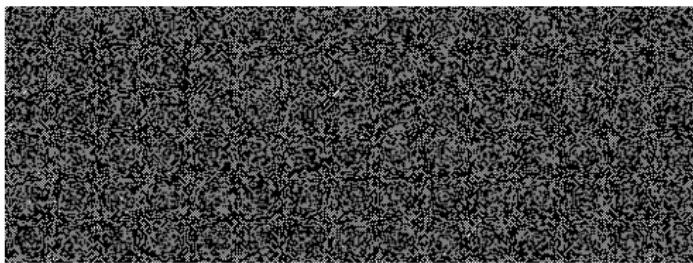
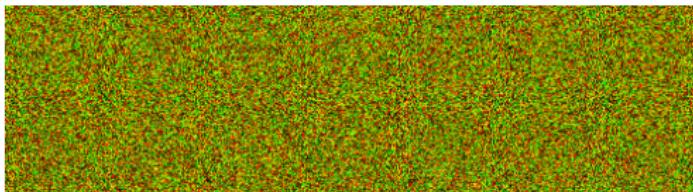
Odstíny: vlny, pruhy



Efekty



Skrývačky v bitmapové grafice



Základní geometrické algoritmy

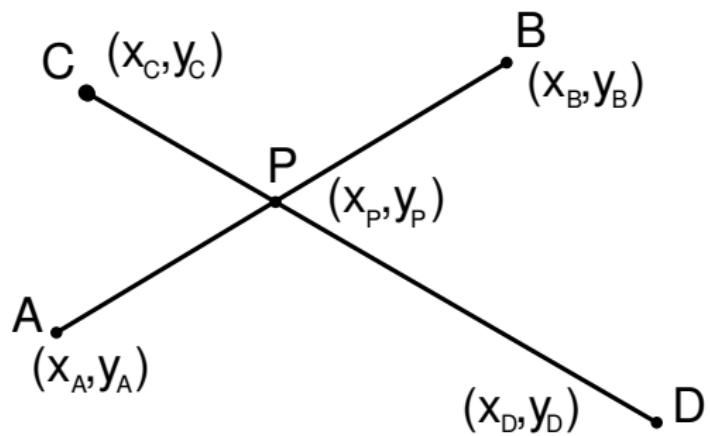
Radek Pelánek

IV122

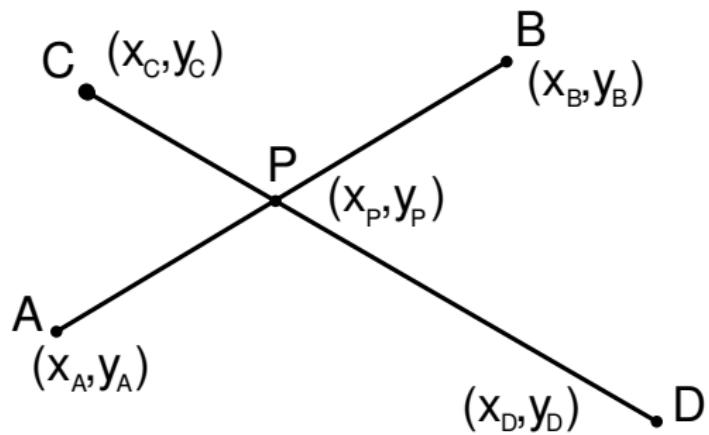
Geometrické algoritmy a speciální případy

- geometrické algoritmy často vyžadují ošetření speciálních případů (vedou na dělení nulou apd), např.:
 - rovnoběžné přímky
 - kolmé přímky
 - průsečík v koncovém bodě
- pro zjednodušení budeme zde ignorovat (při náhodně generovaných vstupech nastává s velmi malou pravděpodobností)

Hledání průsečíku



Hledání průsečíku



$$x_P = \frac{(x_A y_B - y_A x_B)(x_C - x_D) - (x_A - x_B)(x_C y_D - y_C x_D)}{(x_A - x_B)(y_C - y_D) - (y_A - y_B)(x_C - x_D)}$$

Hledání průsečíku

The intersection P of line L_1 and L_2 can be defined using determinants.

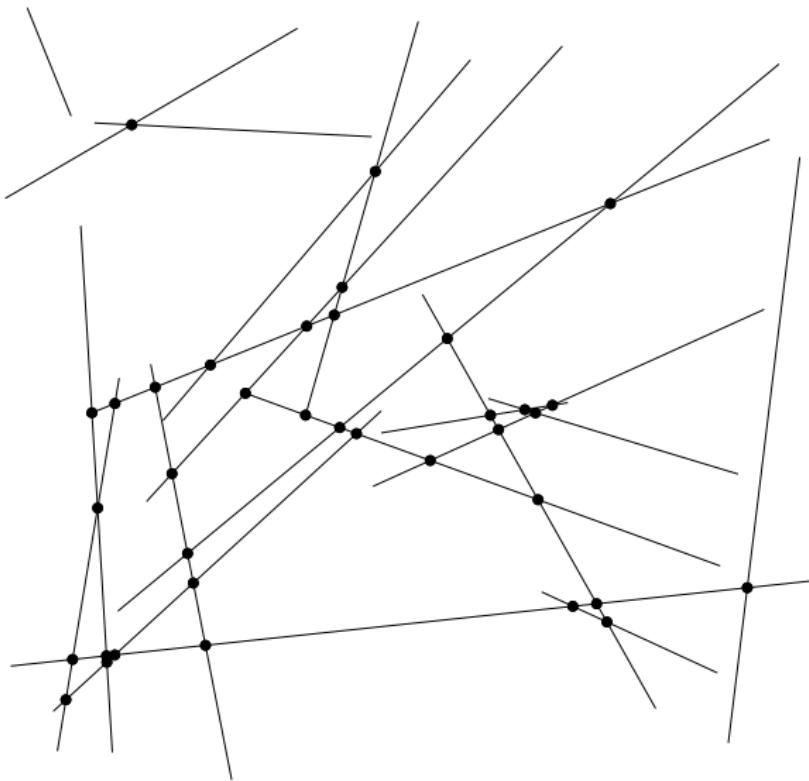
$$P_x = \frac{\begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix}}$$
$$P_y = \frac{\begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix}}{\begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix}}$$

The determinants can be written out as:

$$(P_x, P_y) = \left(\frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}, \right. \\ \left. \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \right)$$

http://en.wikipedia.org/wiki/Line-line_intersection

Hledání všech průsečíků



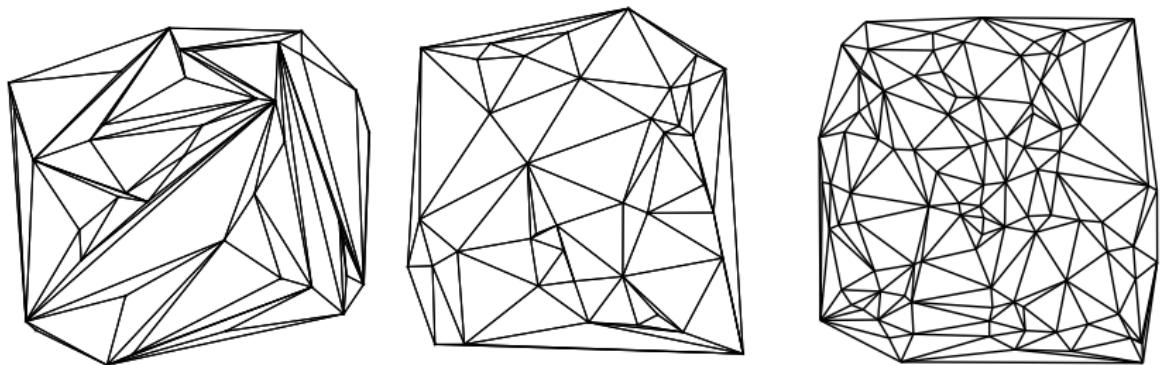
Hledání všech průsečíků

- vstup: N úseček
- výstup: seznam všech průsečíků
- algoritmus:
 - přímočarý: $O(N^2)$
 - „sweep line“: $O((N + k) \log N)$, kde k je počet průsečíků

Triangulace

- rozdelení dvojrozměrného objektu/prostoru na trojúhelníky
- motivace: s trojúhelníky se snadno pracuje
- častý první krok u složitějších geometrických operací

Triangulace



Jak sestrojit triangulaci?

Co je „pěkná“ triangulace?

Triangulace: základní algoritmus

```
def triangulace(body):  
    inicializuj prázdný výběr  
    pro všechny úsečky U mezi body:  
        pokud se U neprotíná s žádnou úsečkou ve výběru:  
            přidej U do výběru  
    return výběr
```

Minimální triangulace konvexního mnohoúhelníku

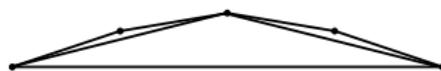
- vstup: konvexní mnohoúhelník
- výstup: minimální triangulace
- kritérium: minimální součet délek hran
- hladový algoritmus:
 - v každém kroku bereme nejkratší hranu, která neprotíná žádnou z již přidaných hran
 - není optimální – najděte protipříklad
- optimální řešení – dynamické programování

Minimální triangulace konvexního mnohoúhelníku

hladová triangulace

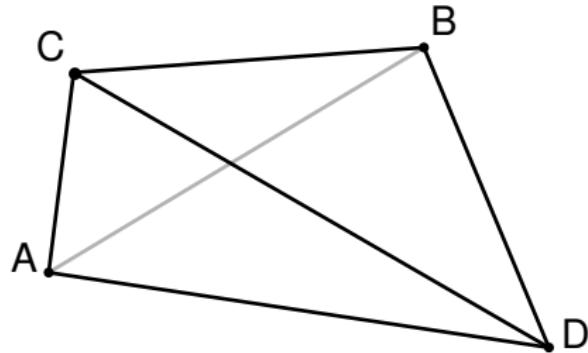


minimální triangulace

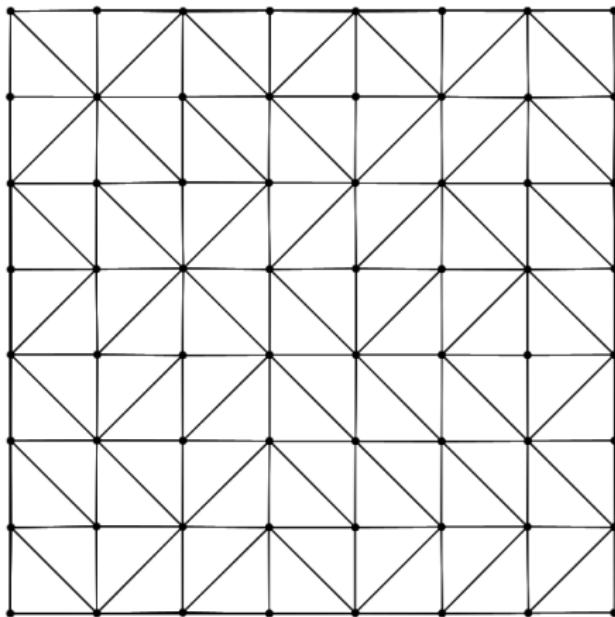


Delaunayova triangulace

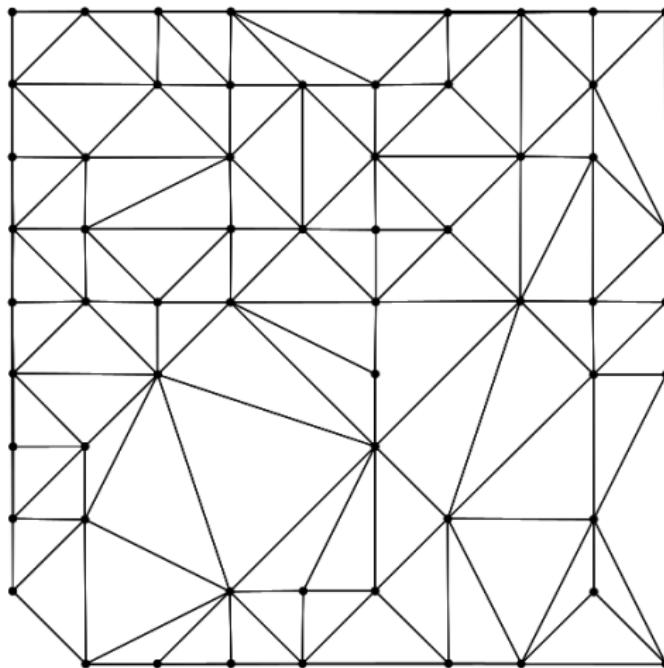
- kritérium: maximalizace minimálního úhlu
- alternativně: žádný bod uvnitř kružnice opsané trojúhelníku v triangulaci
- spojitost Voronei diagram
- algoritmus: prohazování hran



Hrátky s triangulací



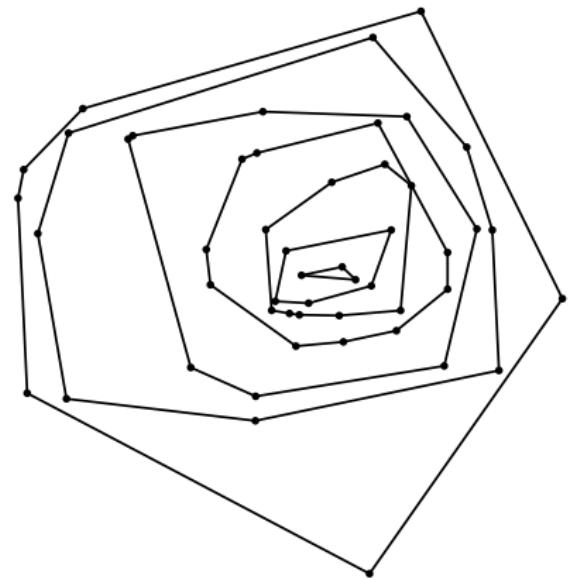
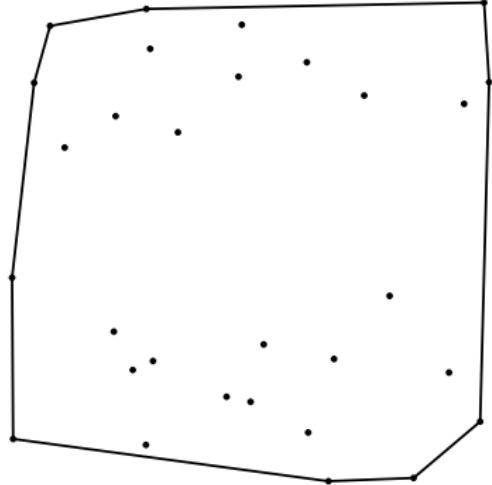
Hrátky s triangulací



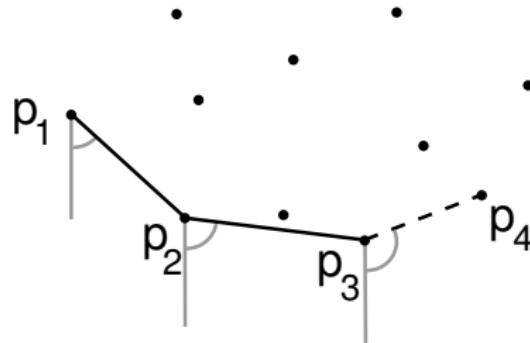
Konvexní obal

- Množina M je konvexní, pokud pro každé dva body z této množiny platí, že všechny body na jejich spojnici leží v M .
- Konvexní obal množiny bodů je nejmenší konvexní množina, která obsahuje všechny dané body.

Konvexní obal

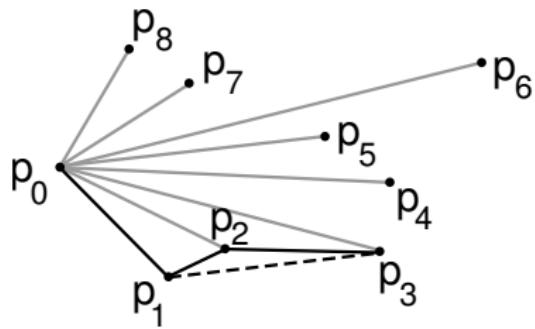


Konvexní obal: Jarvisův algoritmus



časová složitost: $O(nh)$, kde n je celkový počet bodů a h je počet bodů tvořících konvexní obal

Konvexní obal: Grahamův algoritmus



časová složitost: $O(n \log n)$

Fraktály a chaos I

Radek Pelánek

IV122

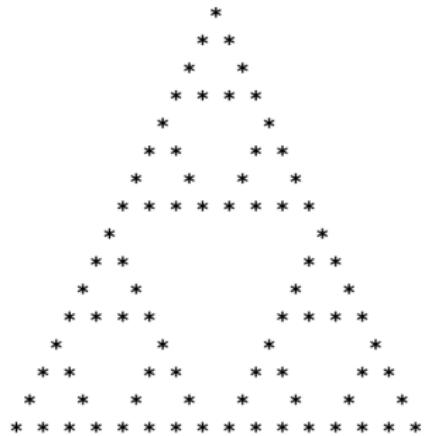
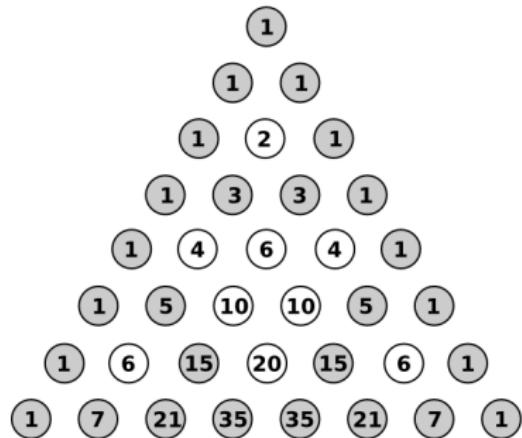
- (netriviální) sobě-podobnost
- rekurze
- reálné příklady (viz slidy P. J.)
- matematické principy: iterované systémy, rekurentní rovnice, L-systémy, podivné atraktory, ...

často vede na „jednoduché programy, které dělají zajímavé věci“ – dnes několik typických ukázek

Sierpińskiho fraktál



Sierpińskiho fraktál a Pascalův trojúhelník



Sierpińskiho fraktál: „chaos game“

- zvolíme 3 body A, B, C tvořící rovnostranný trojúhelník
- vybereme náhodný bod X uvnitř trojúhelníku
- opakujeme následující postup:
 - vyber náhodně jeden z bodů A, B, C
 - přesuň X do poloviny mezi X a zvoleným bodem
 - vykresli X

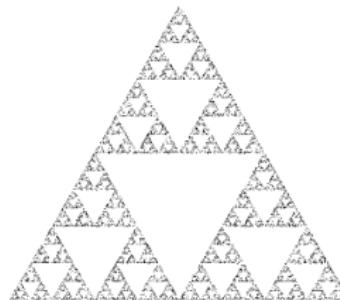
Sierpińskiho fraktál: „chaos game“

E

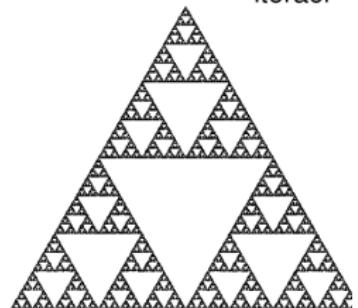
1 000
iterací



10 000
iterací

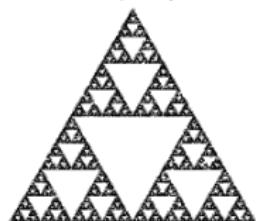


100 000
iterací



„Chaos game“: další fraktály

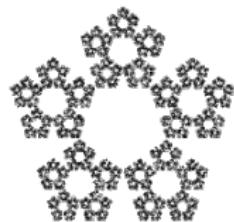
$n = 3, r = 1/2$



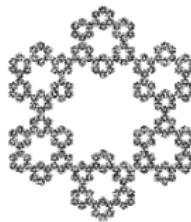
$n = 5, r = 1/3$



$n = 5, r = 3/8$



$n = 6, r = 1/3$



<http://mathworld.wolfram.com/ChaosGame.html>

Hlavní myšlenka

Malé změny v iniciálních podmínkách mohou způsobit velké změny při dlouhodobém chování.

- Mávnutí křídel motýla v Amazonském pralese může způsobit bouři v Texasu.
- Můžeme dostat **zdánlivě náhodné** chování i pro **deterministický** systém.

60. léta, Lorenz, jednoduchý model počasí, ...

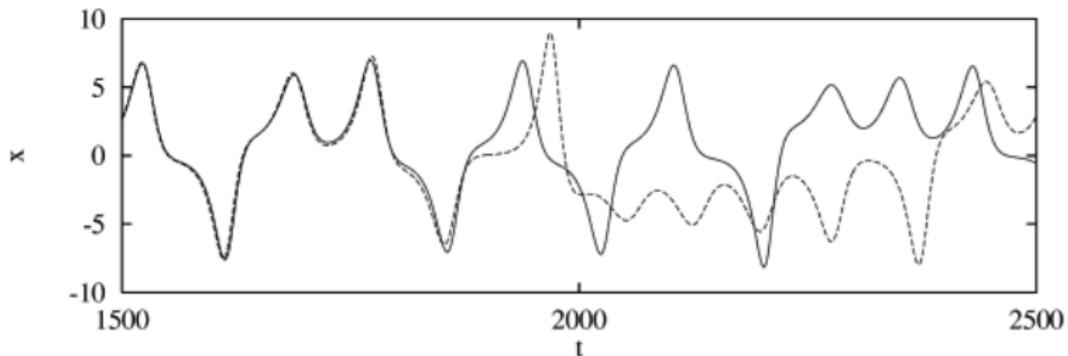
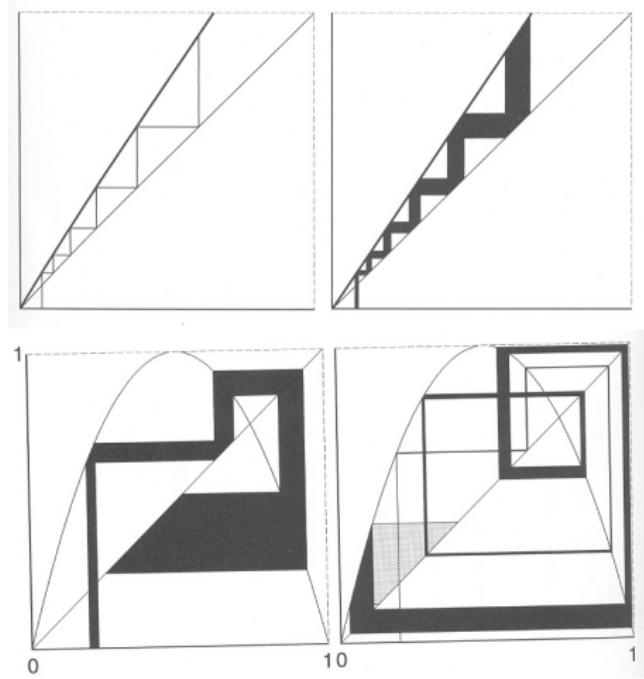


Figure 11.7 Two time evolutions of x with an infinitesimal initial difference

Figure from *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation*. Copyright © 1998-2000 by Gary William Flake. All rights reserved. Permission granted for educational, scholarly, and personal use provided that this notice remains intact and unaltered. No part of this work may be reproduced for commercial purposes without prior written permission from the MIT Press.

Lineární a nelineární systémy



Logistická rovnice

$$x_{t+1} = 4 \cdot r \cdot x_t \cdot (1 - x_t)$$

- $r \in [0, 1]$, $x_0 \in [0, 1]$
- možný význam: velikost populace
- jednoduchý příklad ilustrující základní koncepty chaosu

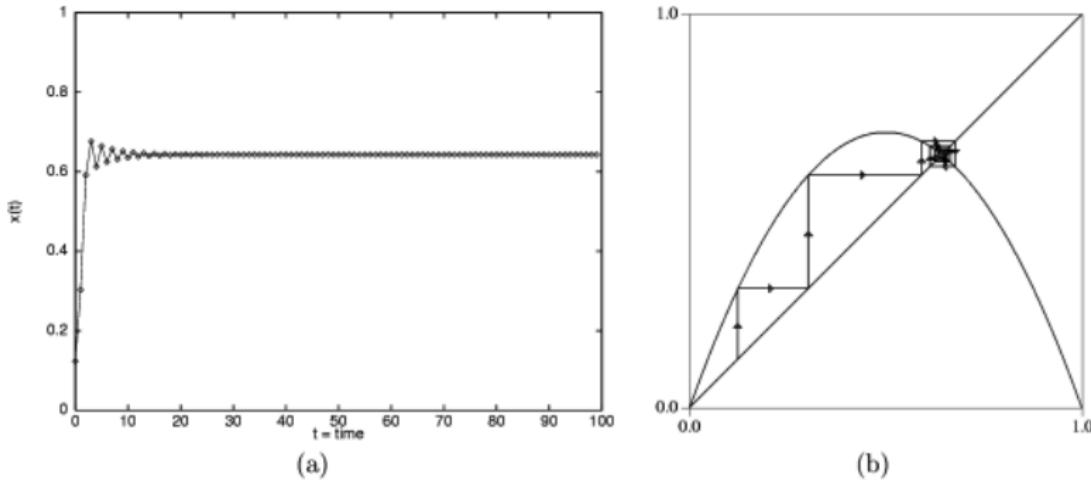
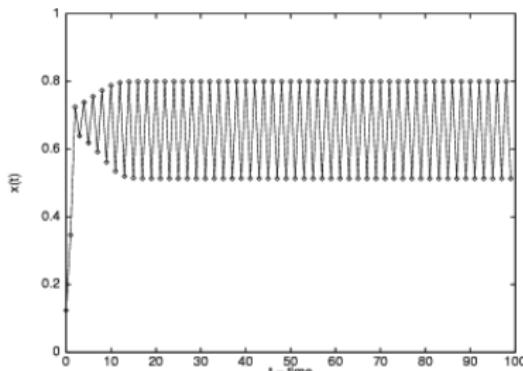
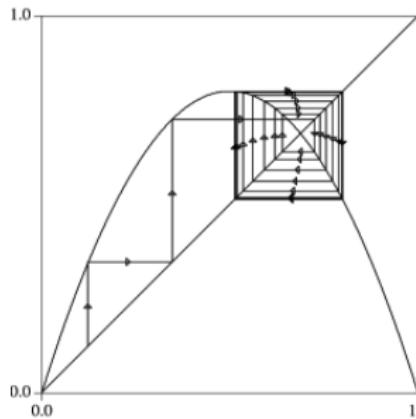


Figure 10.2 Logistic map with $r = \frac{7}{10}$: (a) The time series quickly stabilizes to a fixed point. (b) The state space of the same system shows how subsequent steps of the system get pulled into the fixed point.

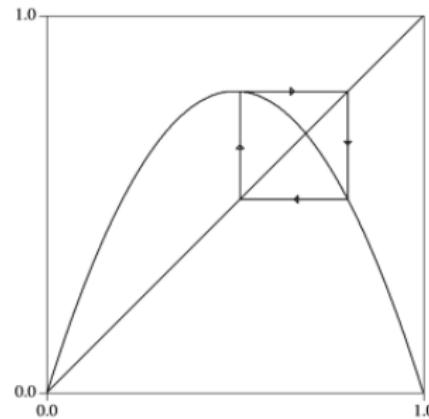
Figure from *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation*. Copyright © 1998–2000 by Gary William Flake. All rights reserved. Permission granted for educational, scholarly, and personal use provided that this notice remains intact and unaltered. No part of this work may be reproduced for commercial purposes without prior written permission from the MIT Press.



(a)

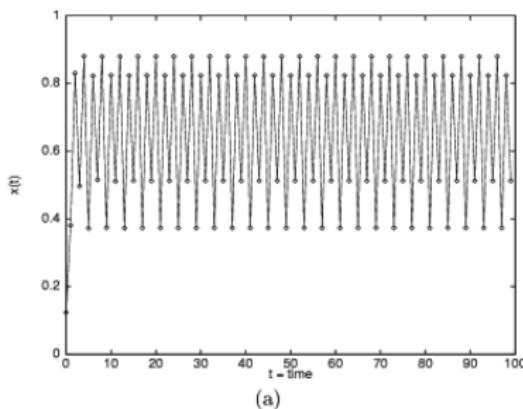


(b)

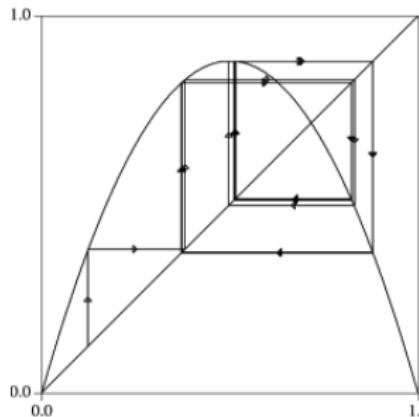


(c)

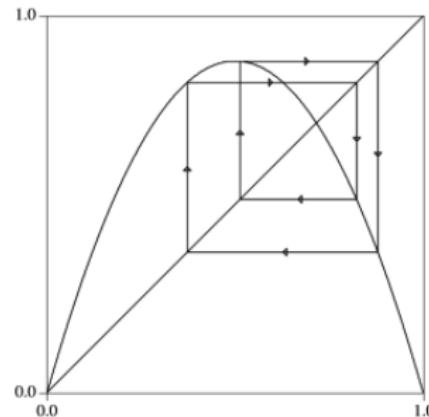
Figure 10.4 Logistic map with $r = \frac{8}{10}$: (a) The time series quickly stabilizes to a period-2 limit cycle. (b) The state space of the same system shows how subsequent steps of the system get pulled into the limit cycle. (c) The state space of the same system but with only the converged values for x , plotted as to clearly show the limit cycle's location.



(a)

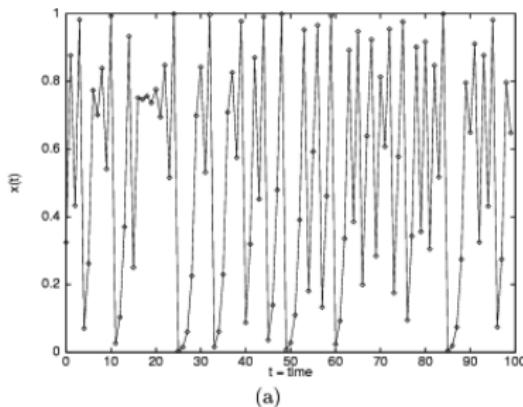


(b)

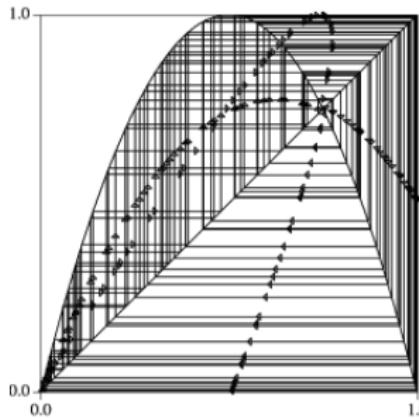


(c)

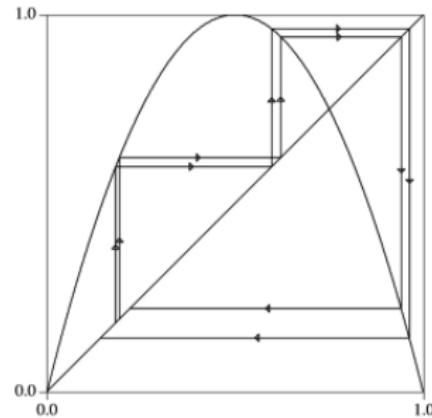
Figure 10.5 Logistic map with $r = \frac{88}{100}$: (a) The time series quickly stabilizes to a period-4 limit cycle. (b) The state space of the same system. (c) The state space of the same system but with only the converged values for x_t plotted.



(a)

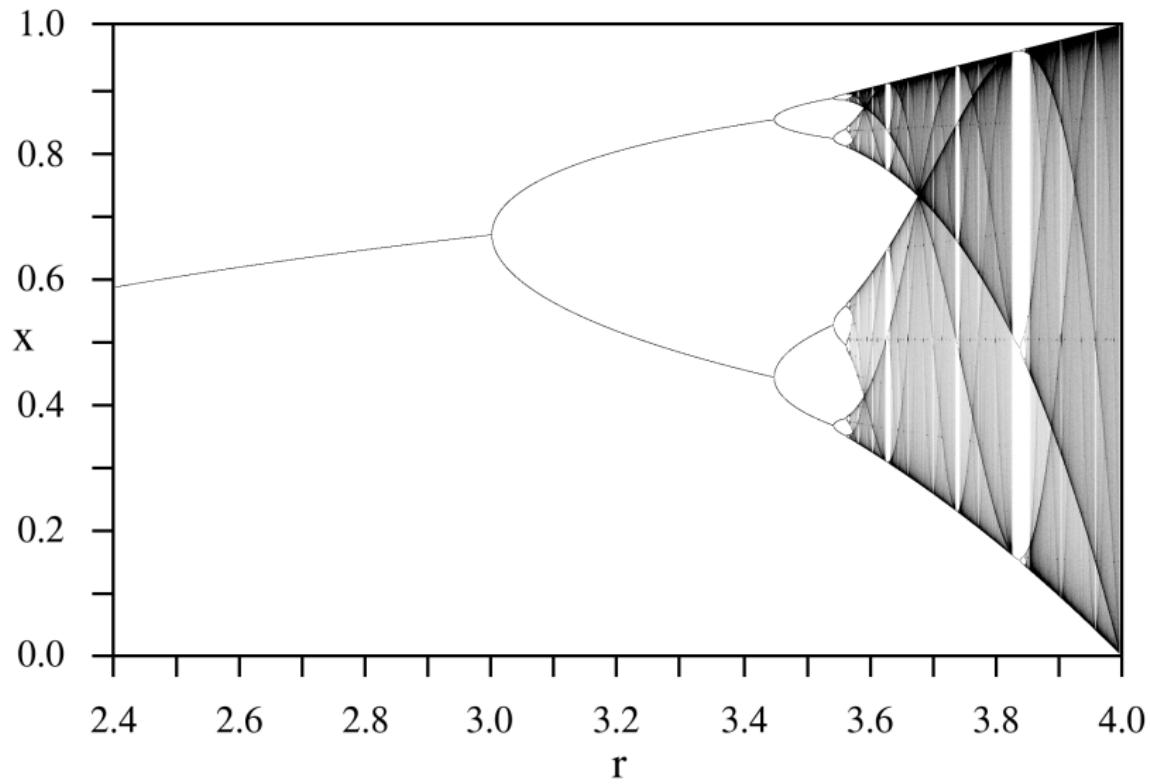


(b)



(c)

Figure 10.6 Logistic map with $r = 1$: (a) The time series is chaotic and has the appearance of noise. (b) The state space of the same system, which illustrates how the system's trajectory visits every local region. (c) The state space of the same system with only four steps plotted, so as to show how small differences turn into larger differences.



Feigenbaumův diagram

- pro hodnoty r simulovat 200 kroků, prvních 100 zahodit, ostatní zanést na y -ovou osu
- Feigenbaumův bod: přechod od řádu k chaosu
- bifurkační body, Feigenbaumova konstanta 4.6692
- soběpodobnost
- vztah reálné věci: tok (přímý, turbulence), srdce (pravidelně, fibrilace)

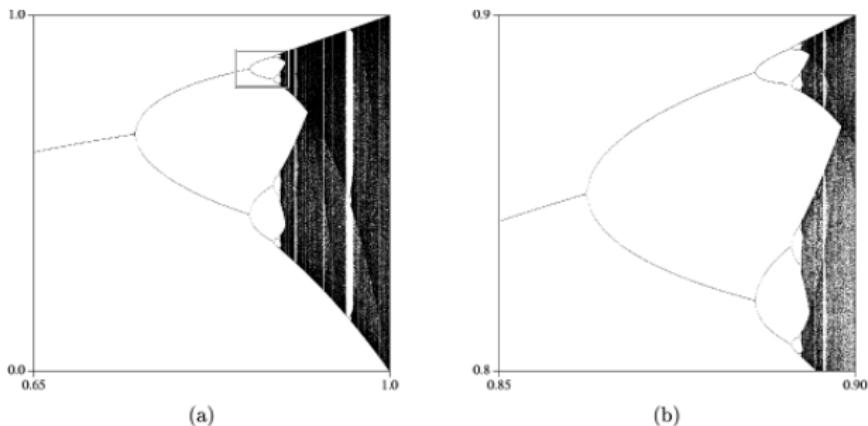
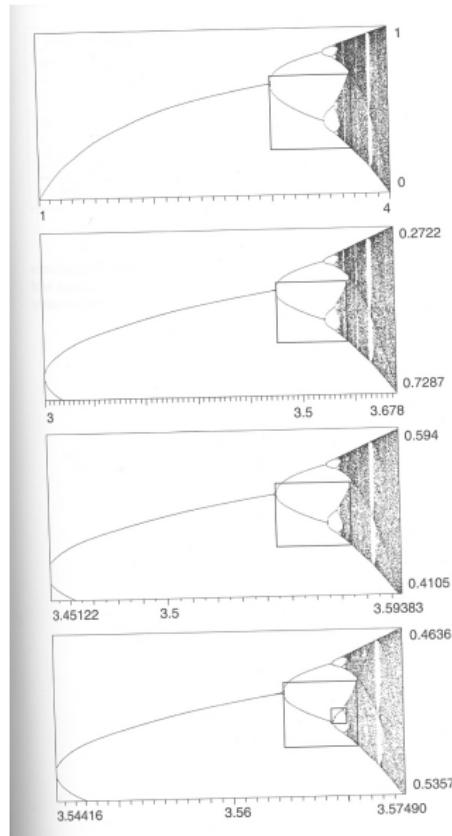
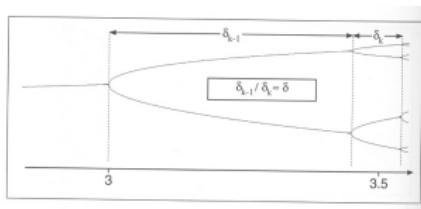


Figure 10.7 Bifurcation diagrams for the logistic map: (a) This image has values of r such that fixed points, limit cycles, and chaos are all visible. (b) This image shows the detail of the boxed section of (a).

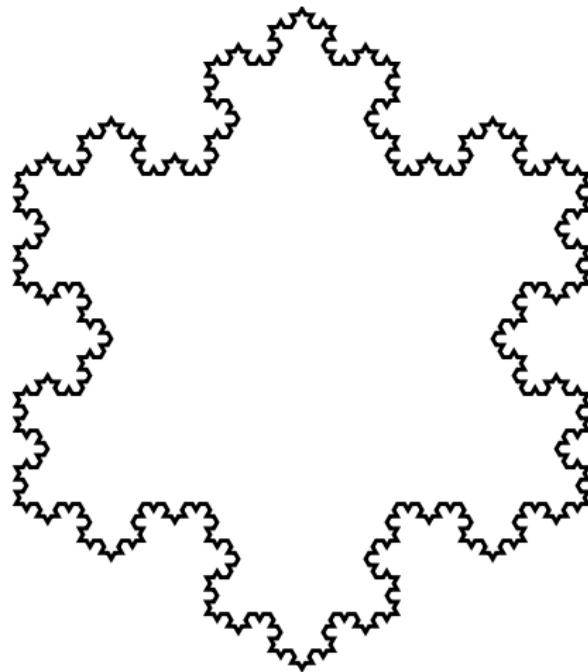
Figure from *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation*. Copyright © 1998-2000 by Gary William Flake. All rights reserved. Permission granted for educational, scholarly, and personal use provided that this notice remains intact and unaltered. No part of this work may be reproduced for commercial purposes without prior written permission from the MIT Press.



Feigenbaumova konstanta



Kochova vločka



L-systém

- Lindenmayerův systém
- modelování růstu rostlin, viz např. *The Algorithmic Beauty of Plants*
<http://algorithmicbotany.org/papers/abop/abop.pdf>
- paralelní přepisovací gramatika:
 - axiom
 - přepisovací pravidla, aplikována paralelně
- přirozená interpretace želví grafikou

Kochova vločka – L-systém

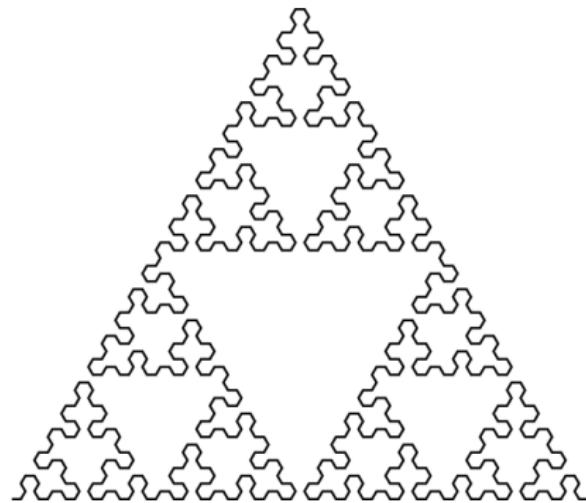
- symboly: F, -, +
 - axiom: F--F--F
 - přepisovací pravidlo systému je $F \Rightarrow F+F--F+F$
 - interpretace: F = forward(10), + = right(60), - = left(60)

$$F \uparrow\uparrow F+F--F+F \uparrow\uparrow$$

$$F+F-F+F+F-F-F+F-F+F-F+F-F+F \Rightarrow$$

F+F--F+F+F--F+F--F+F--F+F+F+F--F+F+F+F--F+F+F+F--
F+F--F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F+F+F--
F+F--F+F+F+F--F+F+F+F--F+F+F+F--F+F+F+F--F+F+F+F--

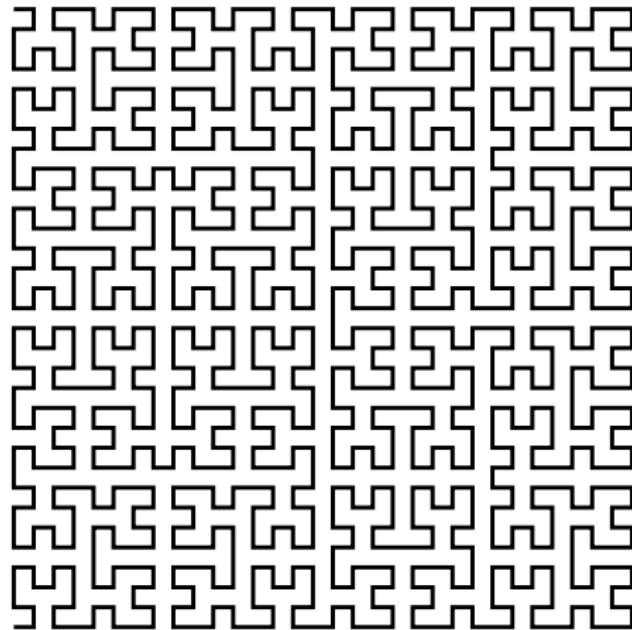
Sierpińskiho fraktál – L-systém



$A \Rightarrow B-A-B$

$B \Rightarrow A+B+A$

Hilbertova křivka



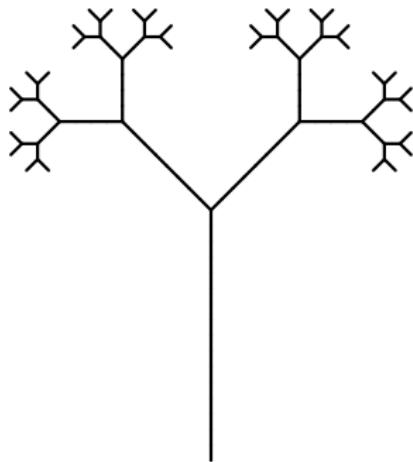
A \Rightarrow - B F + A F A + F B -
B \Rightarrow + A F - B F B - F A +

- prostor vyplňující křivka
- další podobné:
Peanova křivka

Rozšíření

- [– push, uložení polohy želvy na zásobník
-] – pop, obnovení polohy želvy ze zásobníku

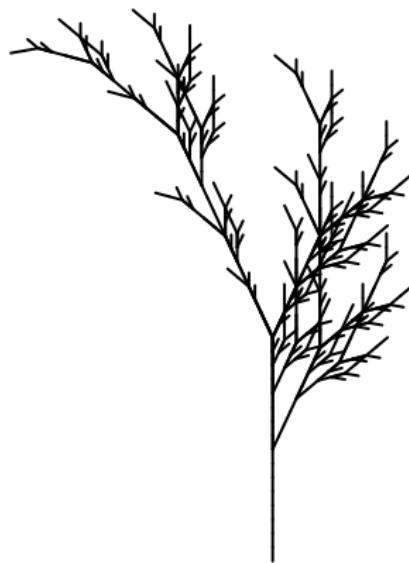
Strom



$A \Rightarrow F [+ A] - A$

$F \Rightarrow F F$

Strom II



$A \Rightarrow F - [[A] + A] + F [+ F A] - A$

$F \Rightarrow F\ F$

úhel 25°

Barevné rostliny

úhel: 25°

$F \Rightarrow FF [+F-FF]$
[-F+F+F]



Stochastický L-systém

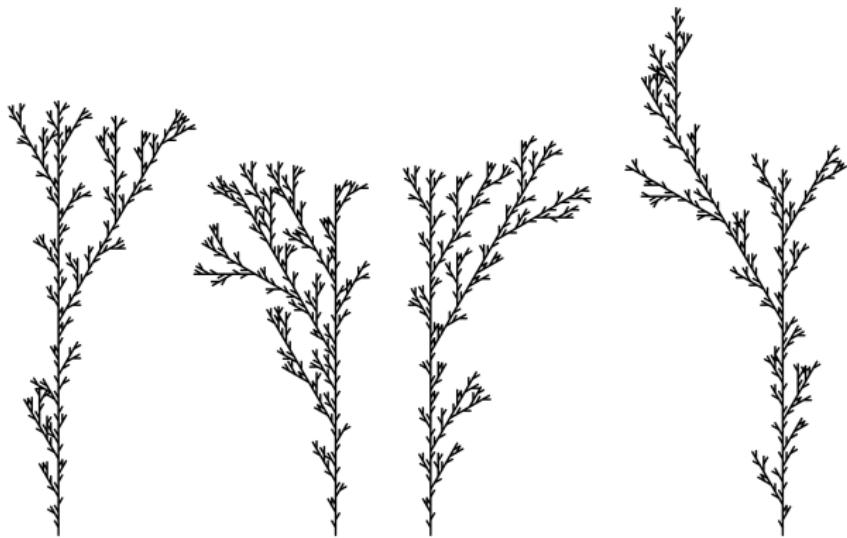
vybíráme náhodnostně jedno z pravidel

úhel: 30°

$F \Rightarrow F[+F]F[-F]F$

$F \Rightarrow F[+F]F$

$F \Rightarrow F[-F]F$



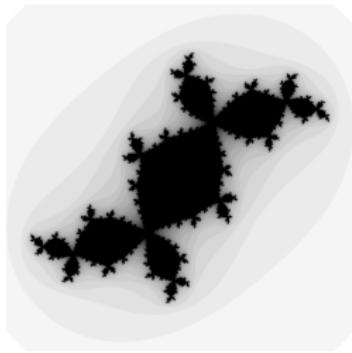
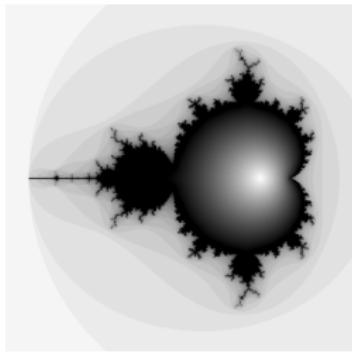
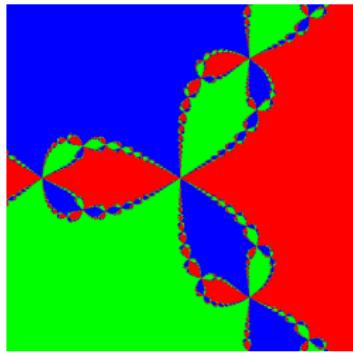
Úloha

- důraz na kompaktnost a eleganci implementace
 - žádný „copy & paste“ kód
 - oddělení „dat“ a „obecného principu“
 - stručně zapsaná pravidla (řetězce) ⇒ obrázek
- experimentování s pravidly – neopisujte pouze pravidla, zkuste vlastní variace!

Fraktály a komplexní čísla

Radek Pelánek

IV122



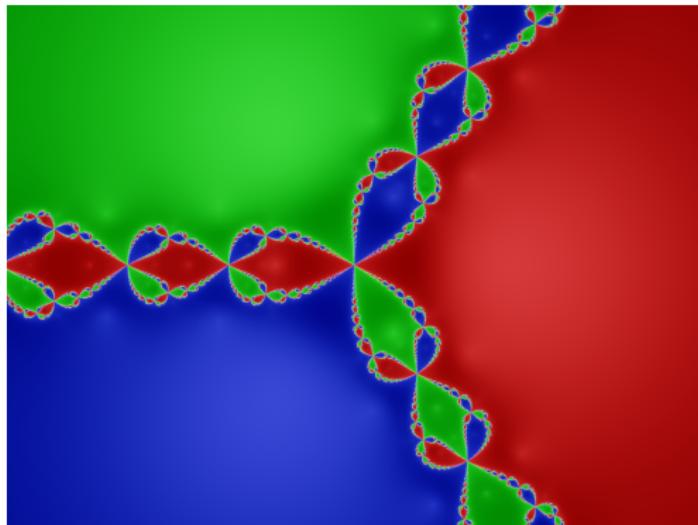
Komplexní čísla: připomenutí

- imaginární číslo $i = \sqrt{-1}$
- komplexní číslo: $x + yi$
- polární souřadnice: $r(\cos \varphi + i \sin \varphi) = re^{i\varphi}$
- sčítání: $(a + bi) + (c + di) = (a + c) + (b + d)i$
- násobení: $(a + bi) \cdot (c + di) = (ac - bd) + (bc + ad)i$
- velikost čísla: $\sqrt{x^2 + y^2}$

Komplexní čísla: test

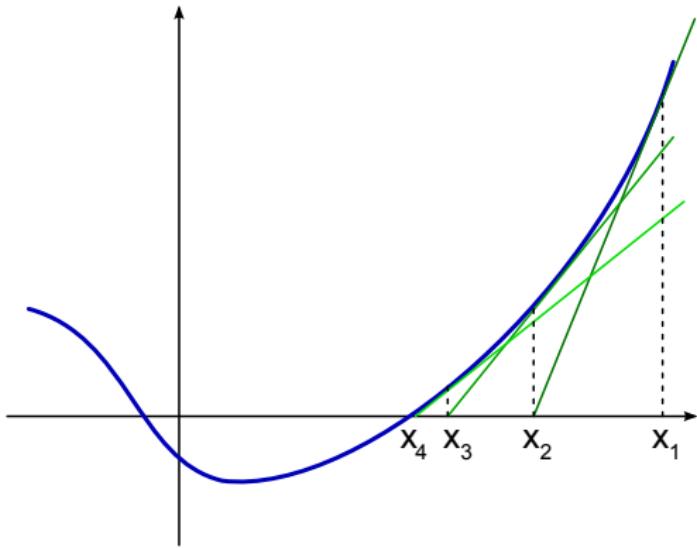
- $(3 - i) + (1 + 2i) = \dots$
- $(1 + i)^2 = \dots$
- $i^{39} = \dots$
- $|5 + i| = \dots$
- $(4 + 2i)/(2i) = \dots$
- $\sqrt{i} = \dots$
- $e^{\pi i} = \dots$

Newtonův fraktál



Zdroj: Wikipedia

Newtonova metoda



$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

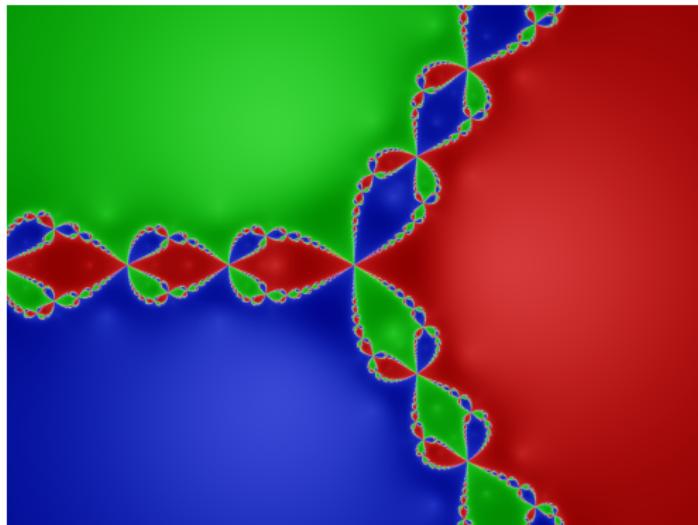
Newtonův fraktál

- $z^3 = 1$
- $z^3 - 1 = 0$
- jaká jsou řešení (v oboru komplexních čísel)?

Newtonův fraktál

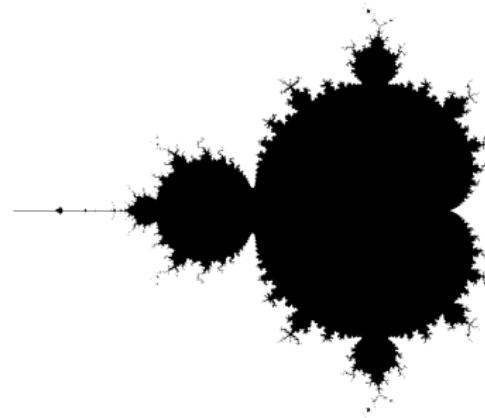
- $z^3 = 1$
- $z^3 - 1 = 0$
- jaká jsou řešení (v oboru komplexních čísel)?
- $1, -0.5 + \frac{\sqrt{3}}{2}i, -0.5 - \frac{\sqrt{3}}{2}i$
- Newtonova metoda: $z_{n+1} = z_n - \frac{z_n^3 - 1}{3z_n^2}$
- pro iniciální bod $z_0 = x + yi$, ke kterému řešení konverguje?
- prakticky: 20 iterací, ke kterému řešení je 20. krok nejblíž?

Newtonův fraktál



Zdroj: Wikipedia

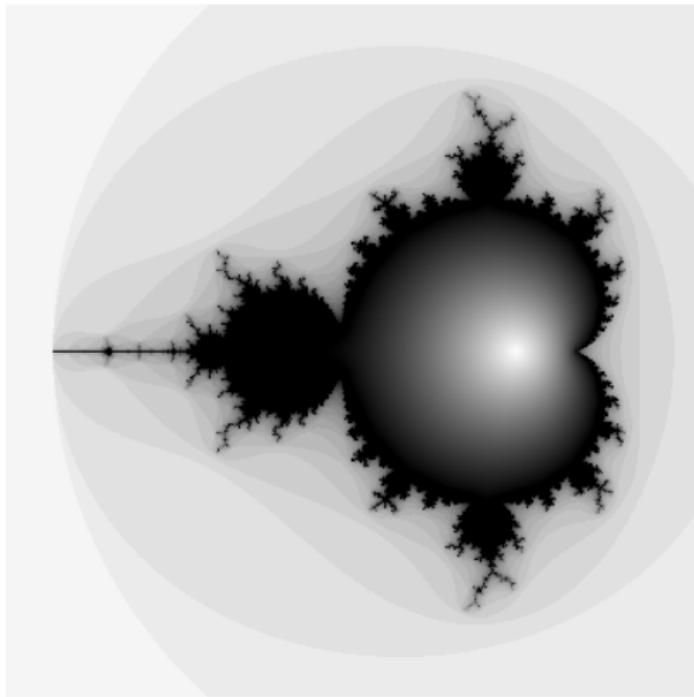
Mandelbrotova množina



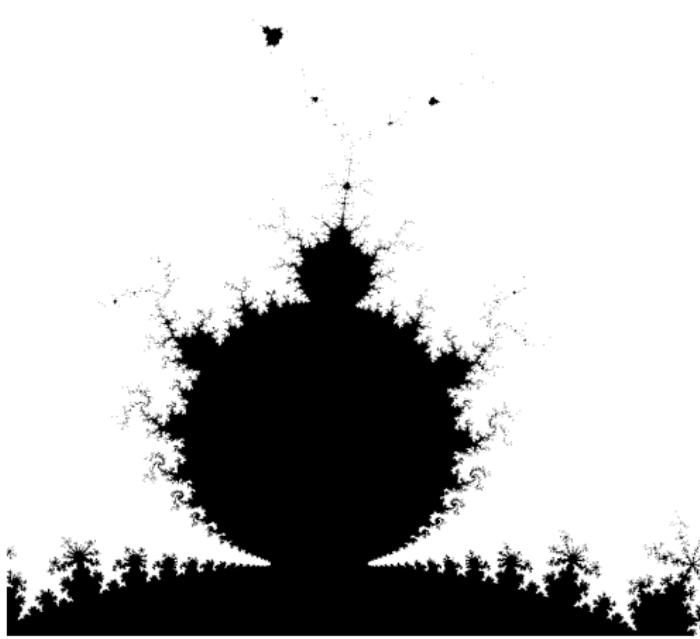
YouTube: Mandelbrot Zoom, např.

<https://www.youtube.com/watch?v=PD2XgQ0yCCk>

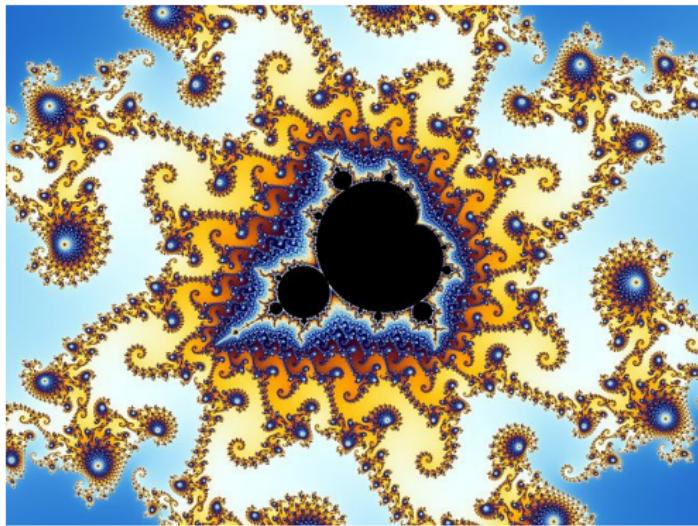
Mandelbrotova množina



Mandelbrotova množina

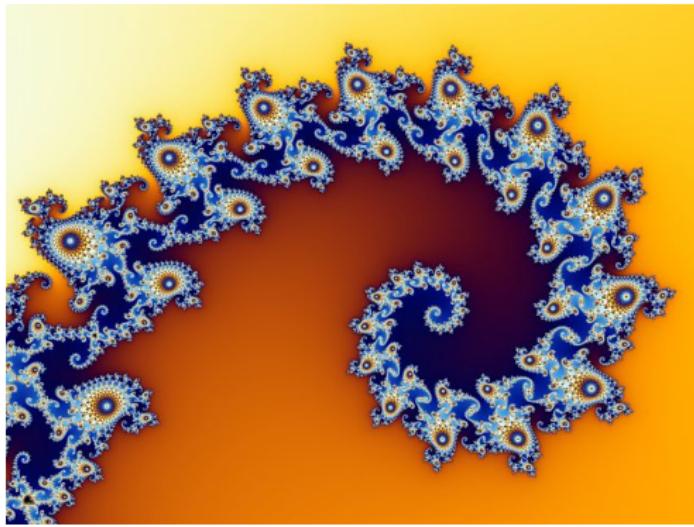


Mandelbrotova množina



Zdroj: Wikipedia

Mandelbrotova množina



Zdroj: Wikipedia

Mandelbrotova množina

- $z_1 = 0$, $c = x + yi$ je konstanta (komplexní číslo)
- definujeme posloupnost

$$z_{n+1} = z_n^2 + c$$

- c patří do Mandelbrotovy množiny \Leftrightarrow tato posloupnost je omezená

Alternativní definice přes reálné posloupnosti:

$$x_{n+1} = x_n^2 - y_n^2 + c_x$$

$$y_{n+1} = 2x_n y_n + c_y$$

Mandelbrotova množina

Příklady:

- $c = 1$

0, 1, 2, 5, 26, ...

není ohrazená

číslo 1 nepatří do Mandelbrotovy množiny

- $c = i$

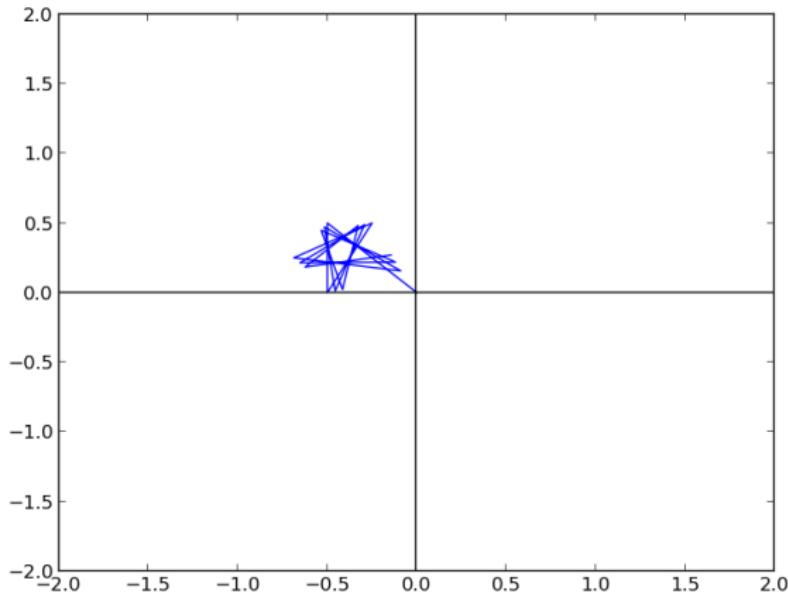
0, i , $(-1 + i)$, $-i$, $(-1 + i)$, $-i$, ...

je ohrazená

číslo i patří do Mandelbrotovy množiny

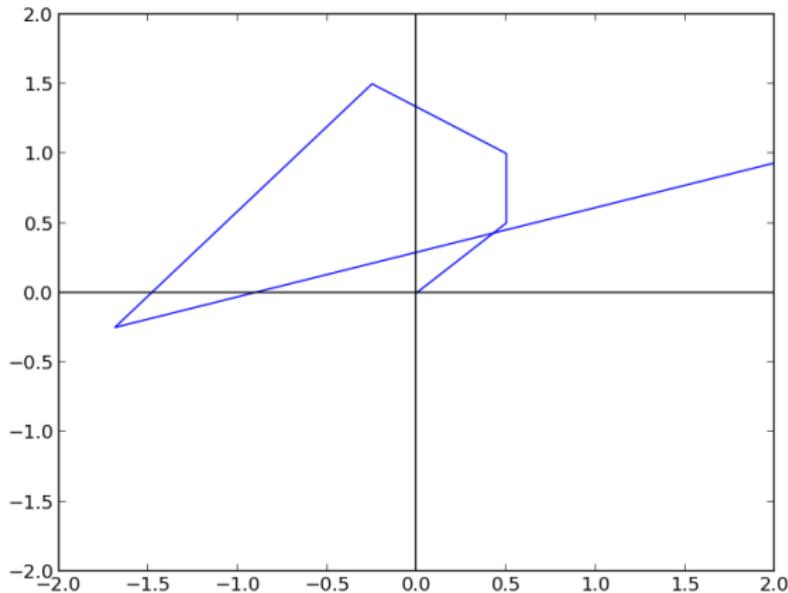
Mandelbrotova množina: ukázka posloupnosti

$$c = -0.5 + 0.5i$$



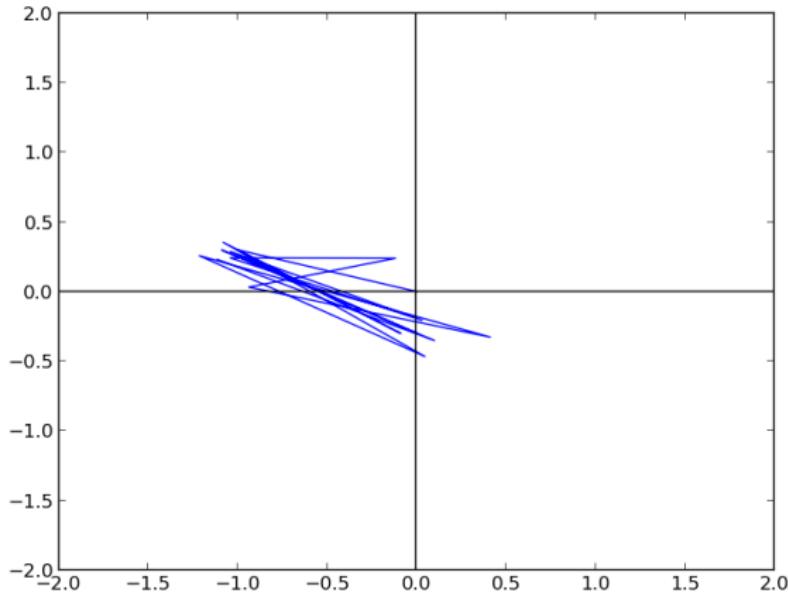
Mandelbrotova množina: ukázka posloupnosti

$$c = 0.5 + 0.5i$$



Mandelbrotova množina: ukázka posloupnosti

$$c = -1 + 0.3i$$

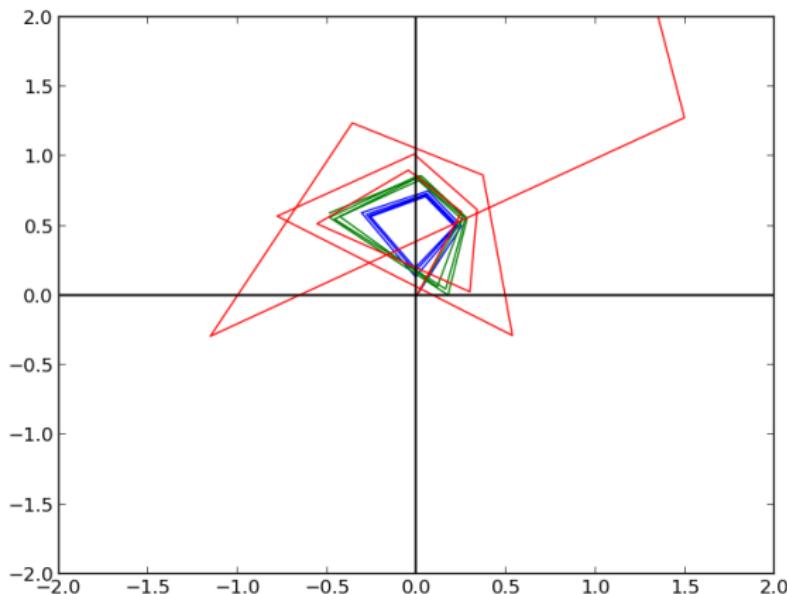


Mandelbrotova množina: ukázka posloupnosti

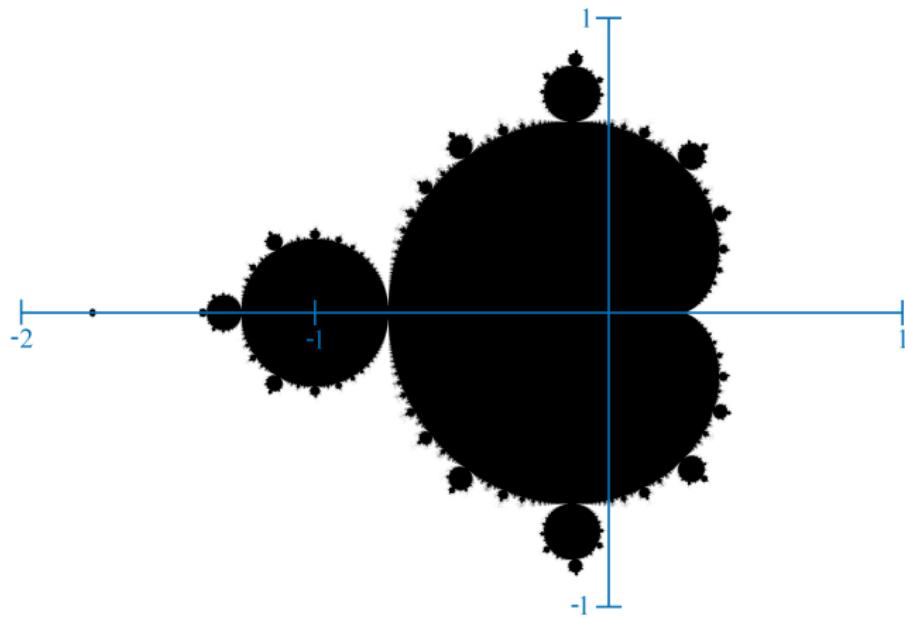
$c = 0.25 + 0.5i$ (modrá)

$c = 0.25 + 0.55i$ (zelená)

$c = 0.25 + 0.6i$ (červená)



Mandelbrotova množina



Zdroj: Wikipedia

Mandelbrotova množina: heuristická metoda

- uděláme 30 iterací
- c dáme do množiny \Leftrightarrow poslední člen má velikost ≤ 2

Mandelbrotova množina – zdrojový kód

```
=  (
    255,
    lambda
        V      ,B,c
        :c    and Y(V*V+B,B, c
                    -1)if(abs(V)<6)else
        (           2+c-4*abs(V)**-0.4)/i
        ) ;v,      x=1500,1000;C=range(v*x
        );import struct;P=struct.pack,M,\
j = '<QIIHHHH',open('M.bmp','wb').write
for X in j('BM'+P(M,v*x*3+26,26,12,v,x,1,24))or C:
    i ,Y=_;j(P('BBB',*(lambda T:(T*80+T**9
        *i-950*T **99,T*70-880*T**18+701*
        T **9 ,T*i**((1-T**45*2)))(sum(
        [          Y(0,(A%3/3.+X/v+(X/v+
            A/3/3.-x/2)/1j)*2.5
            /x -2.7,i)**2 for \
            A      in C
            [:9]]))
            /9)
        ) )
```

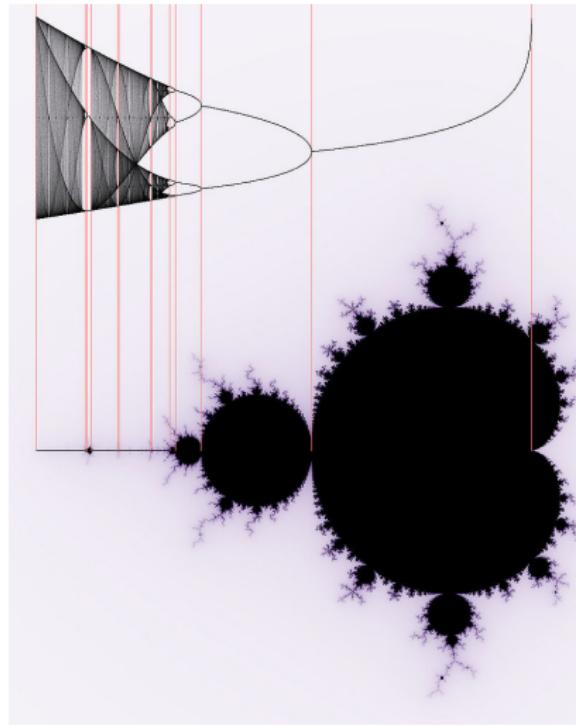
<http://preshing.com/20110926/high-resolution-mandelbrot-in-obfuscated-python/>

Mandelbrotova množina: obarvení

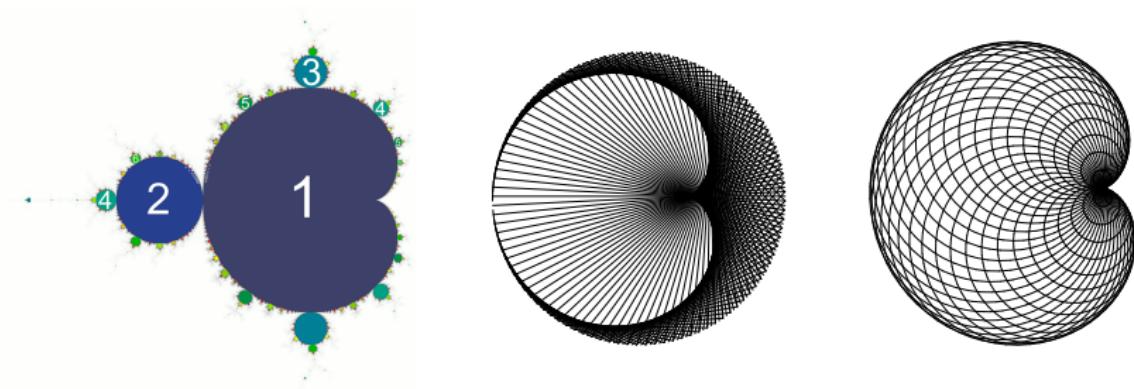
jednoduché metody:

- vnější body: počet iterací potřebných na překročení velikosti 2
- vnitřní body: průměrná vzdálenost od $(0,0)$ v průběhu iterací

Mandelbrotova množina a bifurkace



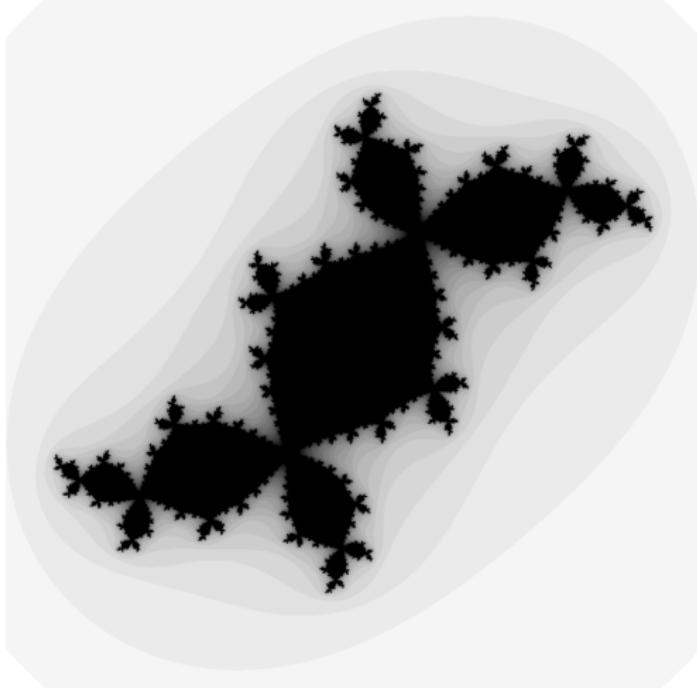
Mandelbrotova množina a kardioida



<https://www.youtube.com/watch?v=qhbukbxJsk8>

Juliova množiny

Juliova množina pro $c = -0.13 + 0.75i$



Juliovy množiny

- opět stejná rovnice $z_{n+1} = z_n^2 + c$
- jedno **fixní c**
- zkoumáme, pro které **iniciální body** $z_1 = x + yi$ je posloupnost ohraničená
- Juliova množina pro hodnotu c je souvislá $\Leftrightarrow c$ patří do Mandelbrotovy množiny.

https://www.mathmarks.org/visualization/julia_sets/

- pozn. pojem Juliova množina použit zjednodušeně

Lineární algebra, transformace v rovině, fraktály

Radek Pelánek

IV122

Lineární algebra – pojmy

- skalár, vektor, matice
- sčítání, násobení, transpozice, inverze
- diagonální matice

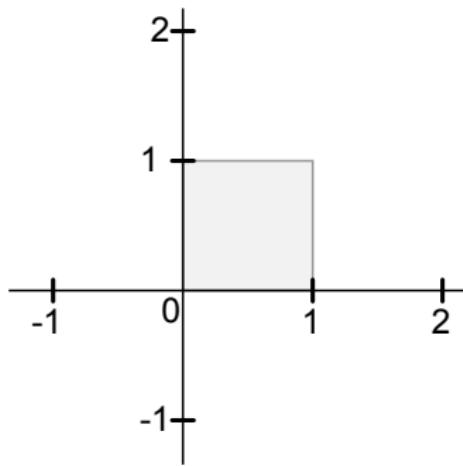
$$\begin{pmatrix} 2 & -0,5 \\ -1 & 3 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 2 \end{pmatrix} =$$

$$\begin{pmatrix} 2 & -0,5 \\ -1 & 3 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 2 \end{pmatrix} =$$

Geometrická interpretace?

Rozcvička: geometrická interpretace

$$\begin{pmatrix} 2 & -0,5 \\ -1 & 3 \end{pmatrix}$$



Lineární a affinní transformace

- lineární transformace:
 - $f(a + b) = f(a) + f(b)$
 - $f(k \cdot a) = k \cdot f(a)$
- affinní transformace: lineární transformace + posun

Lineární a affinní transformace v rovině

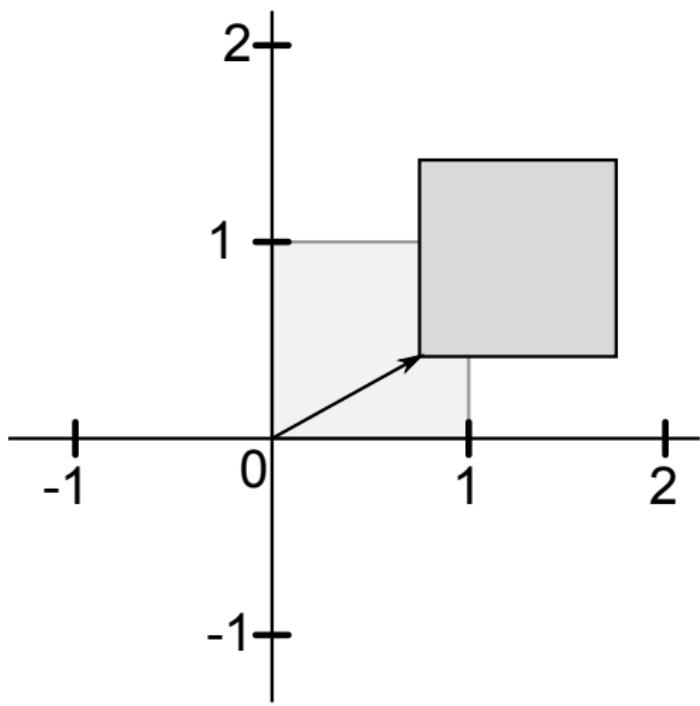
- posunutí
- překlopení
- rotace
- změna velikosti

Jak zapsat pomocí vektorů a matic?

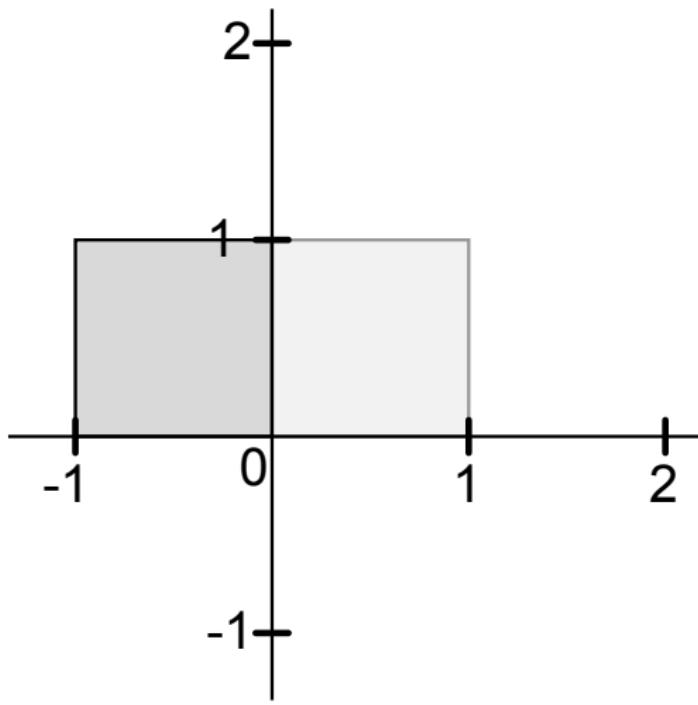
Lineární a affinní transformace v rovině

- lineární transformace \sim násobení maticí 2×2
 - sloupce matice \sim „kam se zobrazí body $[1, 0]$ a $[0, 1]$ “
- affinní transformace \sim násobení maticí 2×2 + přičtení vektoru délky 2

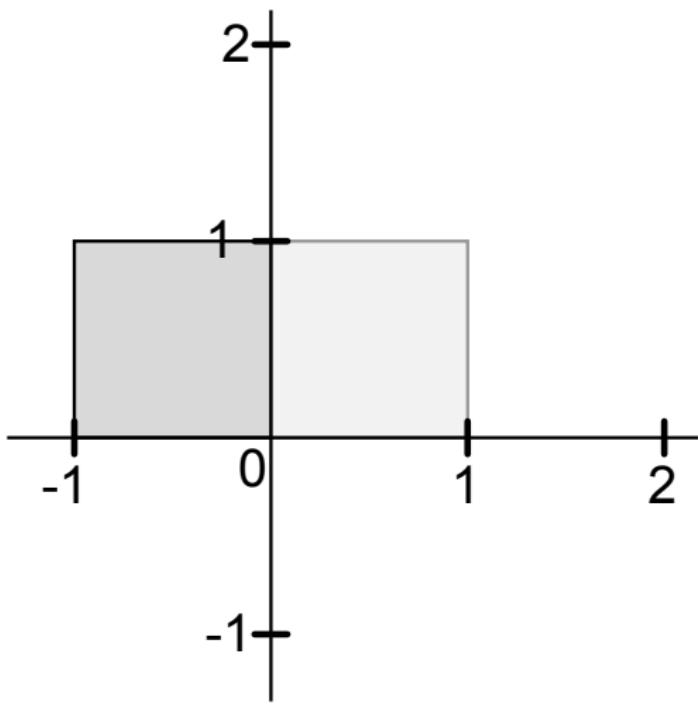
Posunutí (translation)



Překlopení (reflexion)

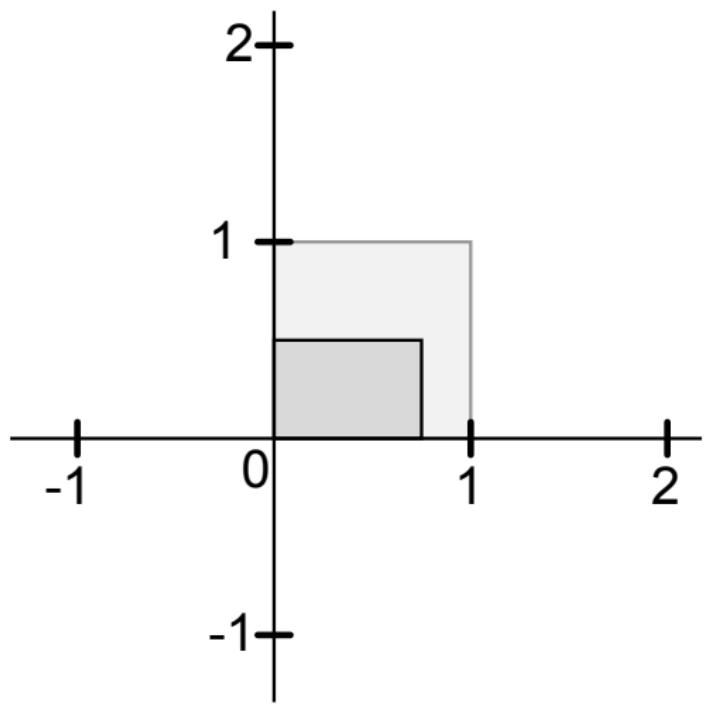


Překlopení (reflexion)

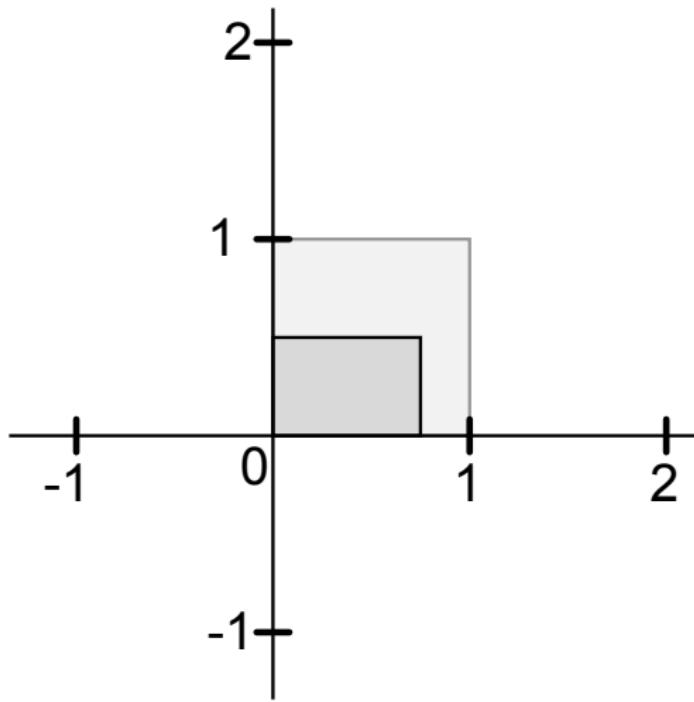


$$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

Změna velikosti (scaling)

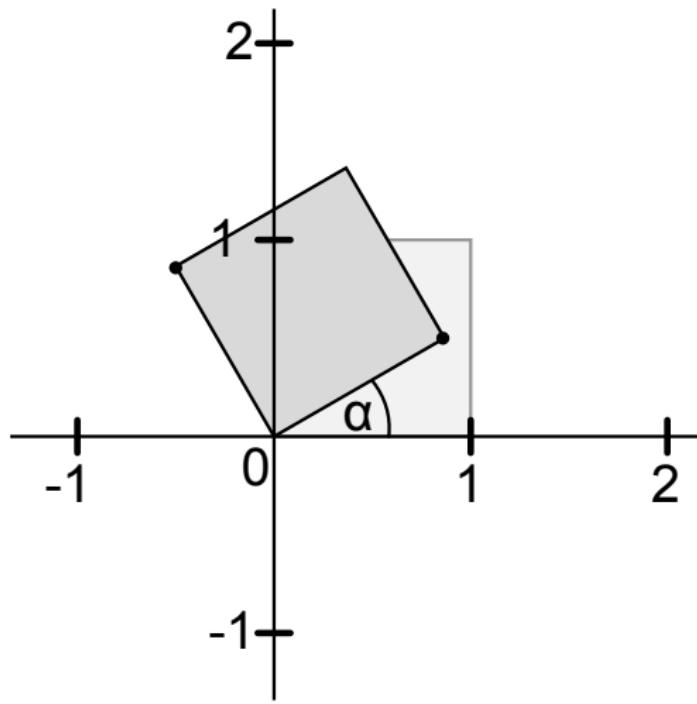


Změna velikosti (scaling)

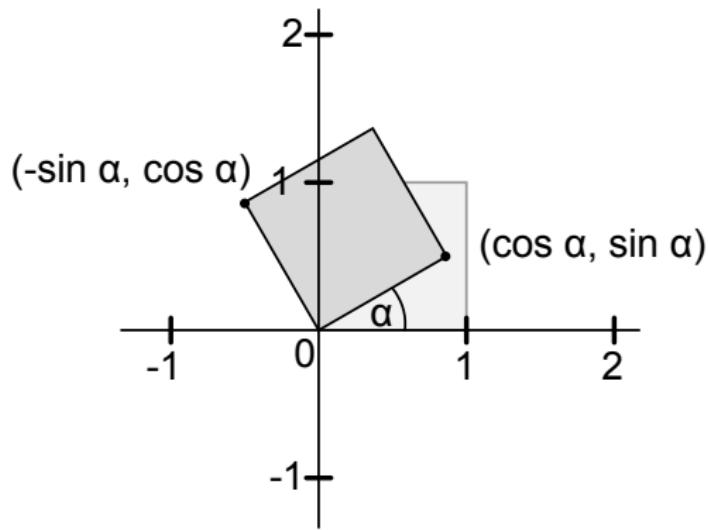


$$\begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix}$$

Rotace (rotation)

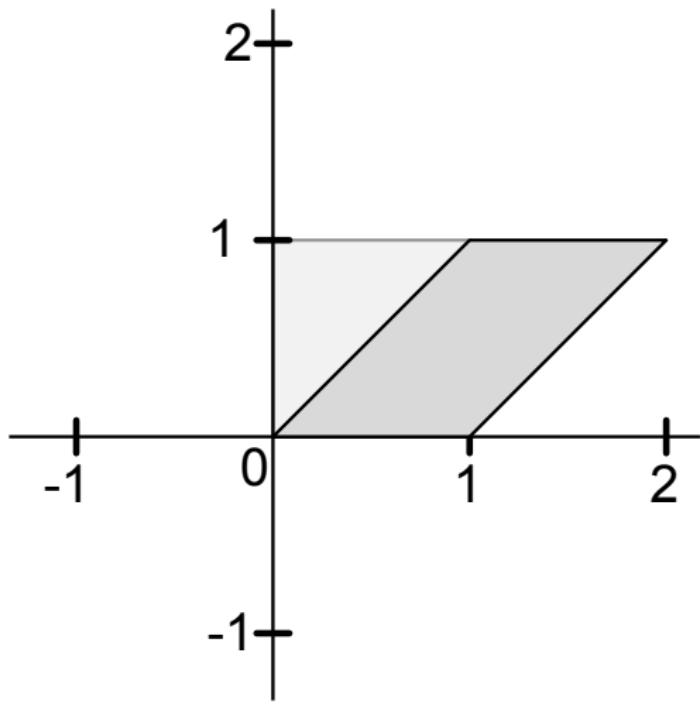


Rotace (rotation)



$$\begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

Shear



$$\begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}$$

Homogenní souřadnice

- reprezentace affinních transformací pomocí matic 3×3
- bod (x, y) reprezentujeme vektorem $(x, y, 1)$
- skládání transformací = násobení matic
(pozor na pořadí násobení)

Klasické souřadnice

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}$$

Homogenní souřadnice

$$\begin{pmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Homogenní souřadnice: příklady

Rotace

$$\begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Posunutí

$$\begin{pmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{pmatrix}$$

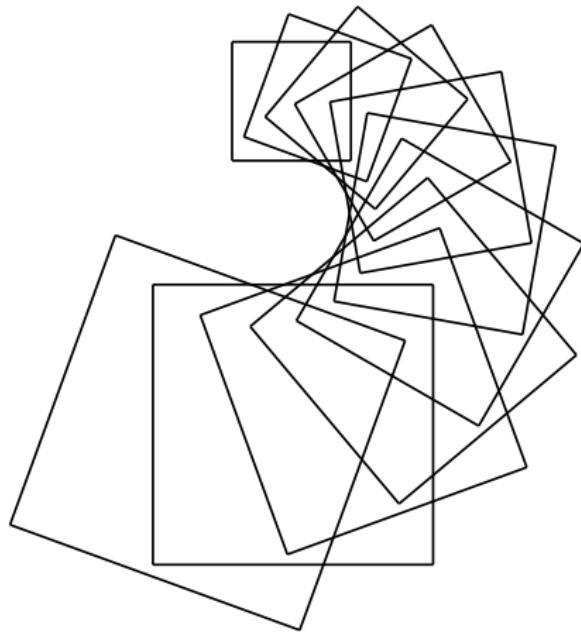
Úkol: implementace transformací

- transformace reprezentujte maticí 3×3
- zvolte vhodnou reprezentaci obrazce v rovině (např. seznam úseček)
- implementujte:
 - generování základních transformací, např. `rotation(angle)`, `scaling(sx, sy)`
 - skládání transformací
 - aplikaci transformace na obrazec
- otestujte

- funkcionální spíše než objektový styl programování
- transformace \sim matice \sim pole 3×3 (není potřeba nic navíc)
- funkce pro generování, skládání, aplikaci

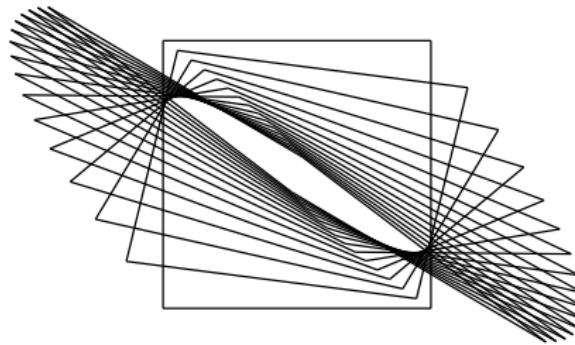
Ukázka 1

Repeat 10: rotation(20), scaling(1.1, 1.1), translation(5, 10)



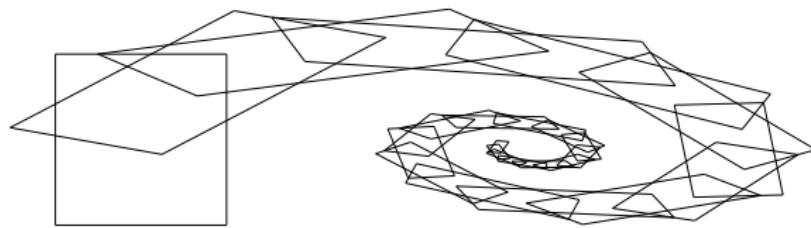
Ukázka 2

Repeat 15: rotation(10), scaling(1.1, 0.8)



Ukázka 3

Repeat 25: shear(1.3), rotation(10), scaling(0.9,0.9),
translation(50, 50)



Poznámka

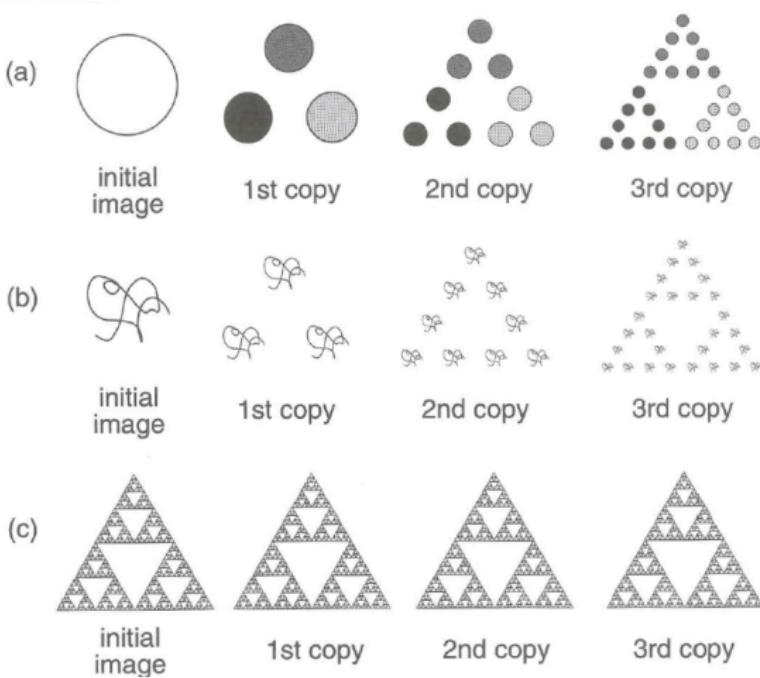
- uvedené příklady nemají vyznačené osy
- pro stejnou sekvenci operací můžete tedy dostat jiný výstup

(rozmyslete si, vyzkoušejte)

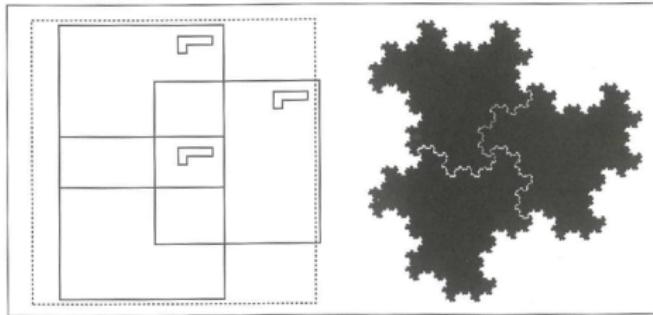
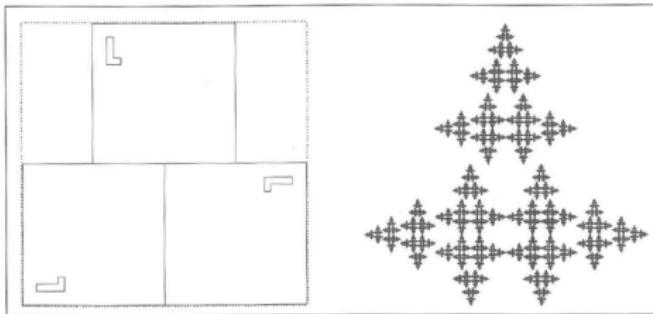
Multiple Reduction Copy Machine (MRCM)

- speciální případ konceptu *deterministic iterated function system*
- iterovaně provádíme operaci:
„nahrad' obrazec několika zmenšenými kopiami“
- iniciální obrázek není důležitý
- „atraktor“ operace (pevný bod, invariant) – typicky fraktál
- definice „zmenšených kopií“ pomocí affinních transformací

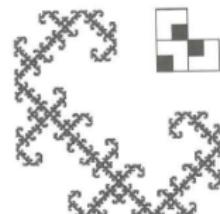
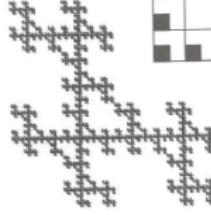
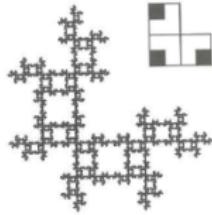
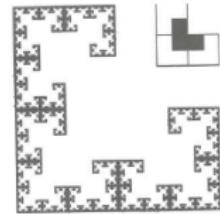
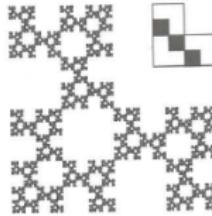
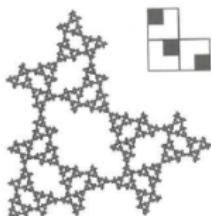
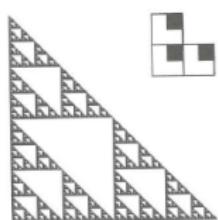
MRCM: princip



MRCM: příklady

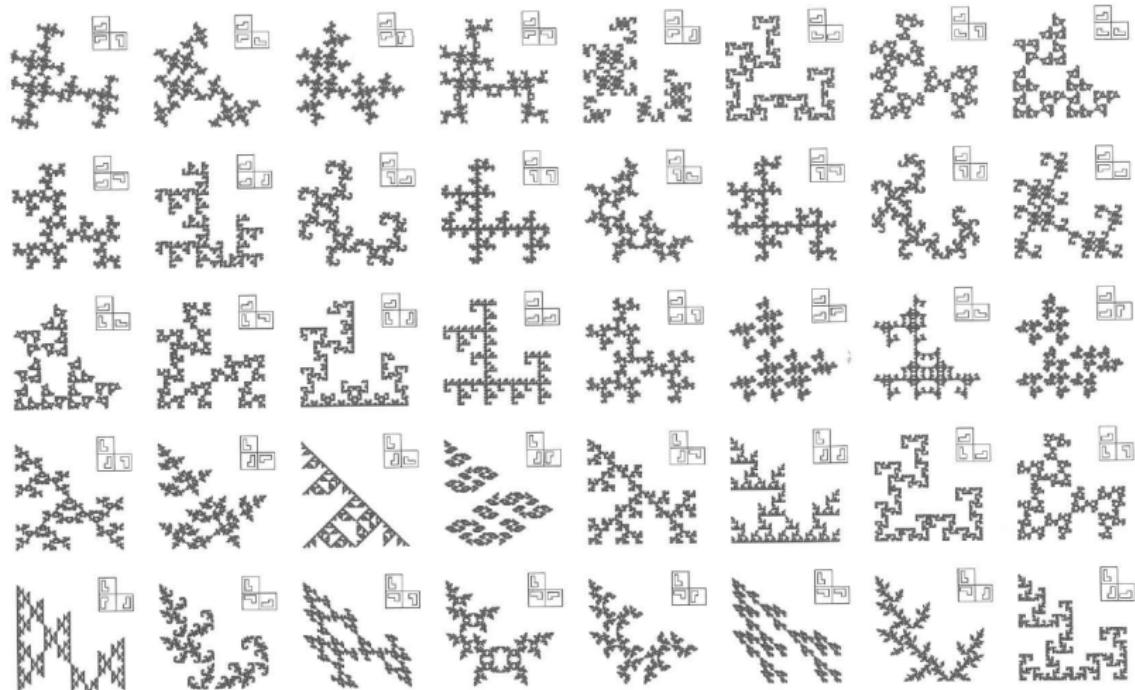


Sierpińskiho příbuzní



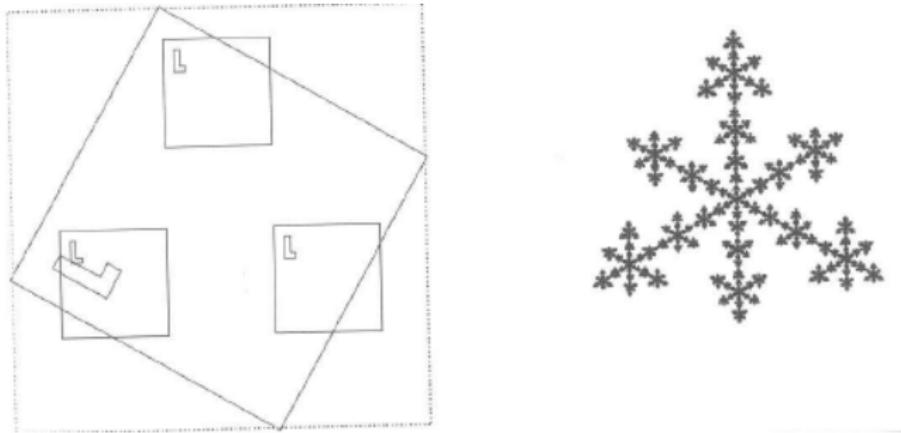
Peitgen, Jurgens, Saupe. *Chaos and Fractals*

Sierpińskiho příbuzní



Peitgen, Jurgens, Saupe. *Chaos and Fractals*

Hvězda

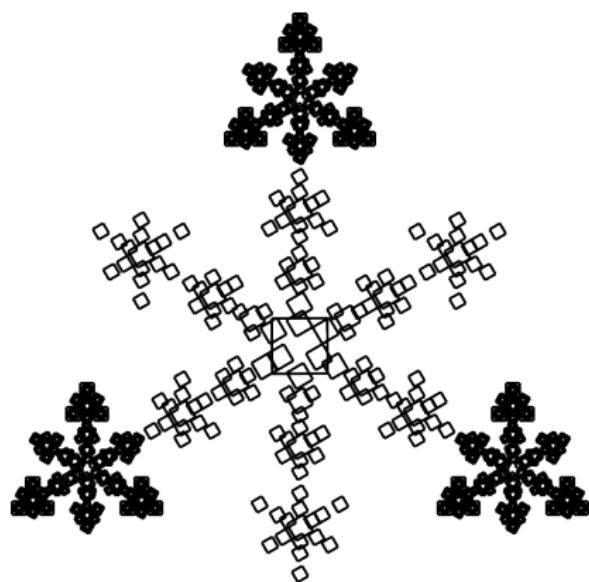


a	b	c	d	e	f
0.255	0	0	0.255	0.3726	0.6714
0.255	0	0	0.255	0.1146	0.2232
0.255	0	0	0.255	0.6306	0.2232
0.370	-0.642	0.642	0.370	0.6356	-0.0061

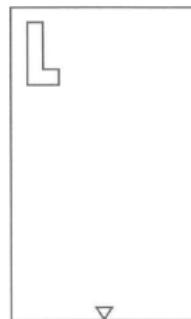
Interpretace uvedených konstant

$$\begin{pmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{pmatrix}$$

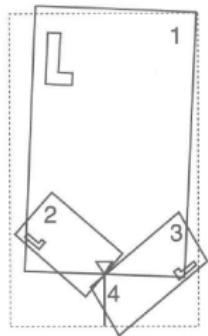
Hvězda – přímočaré generování



Kapradí (Barnsley fern)



Initial Image



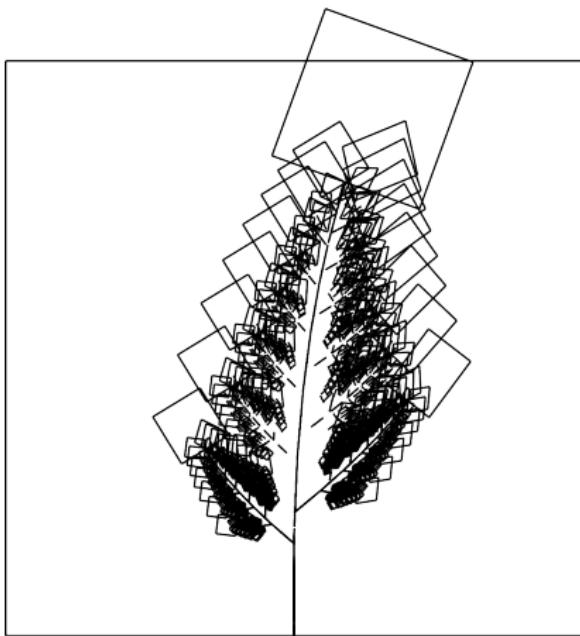
Stage 1

Peitgen, Jurgens, Saupe. *Chaos and Fractals*

Kapradí (Barnsley fern)

a	b	c	d	e	f
0.849	0.037	-0.037	0.849	0.075	0.183
0.197	-0.226	0.226	0.197	0.4	0.049
-0.15	0.283	0.26	0.237	0.575	0.084
0	0	0	0.16	0.5	0

Kapradí – přímočaré generování



Souvislosti

- princip MRCM souvisí s „chaos game“ (generování Sierpińského trojúhelníku za využití náhody, bitmapově)
- zkuste se nad souvislostmi zamyslet a využít princip chaos game třeba pro generování kapradí

Pravděpodobnost, náhoda, kostky

Radek Pelánek

IV122

Výhled

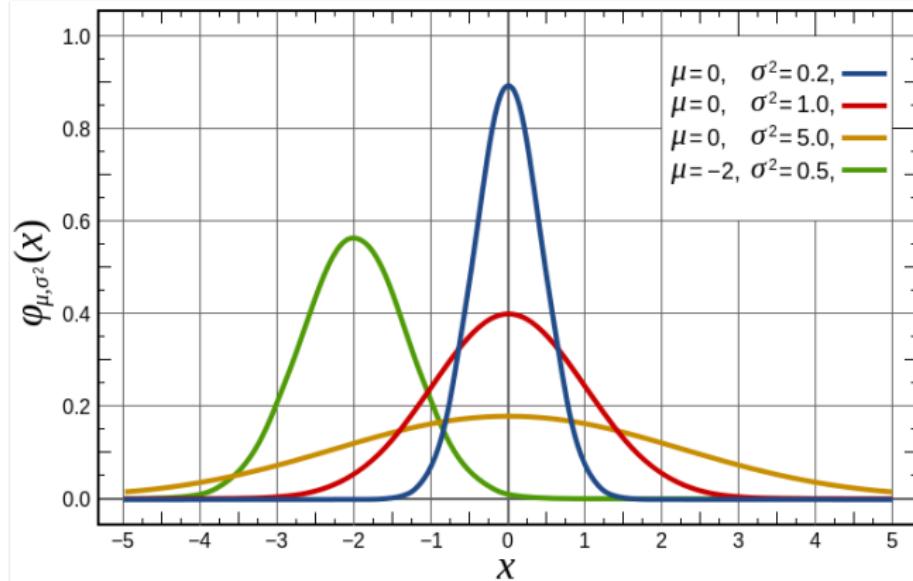
- pravděpodobnost
- náhodná čísla
- lineární regrese
- detekce shluků

- lehce nesourodá směs úloh souvisejících s pravděpodobností
- připomenutí, souvislosti
- krátké programy, realizovatelné i v tabulkovém editoru
- základní myšlenka: využití jednoduchých simulací a analýz pro lepší pochopení abstraktních matematických pojmu
- „ kostky“

Pojmy

- pravděpodobnost, podmíněná pravděpodobnost, nezávislost
- střední hodnota, rozptyl, směrodatná odchylka
- distribuční funkce
- normální distribuce

Normální distribuce



Wikipedia

Normální distribuce

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- μ – průměr
- σ – standardní odchylka

Monty Hall Problem

- troje dveře, za jedněmi z nich je poklad, cílem je najít poklad
- vyberete jedny dveře
- já otevřu jedny z nevybraných dveří, za kterými není poklad
- vy nyní můžete zůstat u své volby nebo změnit své rozhodnutí
- co je rozumné udělat?
 - zůstat u své volby
 - změnit rozhodnutí
 - je to úplně jedno (můžeme se rozhodnout náhodně)

Monty Hall Problem: řešení

- je výhodnější změnit rozhodnutí:
 - zůstat u své volby: 33 %
 - změnit rozhodnutí: 66 %
 - rozhodnout se náhodně: 50 %
- problém známý tím, že i mnoho matematiků se v něm snadno splete

Využití simulace

- pro vybudování intuice (lepší pochopení) se hodí simulace
- Monty Hall – velmi jednoduché využití simulace
- užitečný obecný princip

Monty Hall: experimentálně

- implementujte simulátor hry
- vyzkoušejte strategie „zůstat při původním rozhodnutí“, „změnit rozhodnutí“, „náhodně měnit rozhodnutí“
- experimentálně vyhodnotte úspěšnost strategií v dlouhém běhu

alternativa: jiný, podobný úkol

Náhodná čísla

- aplikace:
 - počítačové hry, loterie
 - kryptografie
 - vědecké výpočty, simulace
- zdroje:
 - „pseudonáhodná čísla“ – běžné `random()`, „deterministické s chaotickým chováním“
 - „opravdová náhoda“ – např. atmosférický tlak, www.random.org

Co to jsou náhodná čísla?

„Házení kostkou“ – čísla 1-6

Která z následující posloupností je více pravděpodobná?

- 1 1 2 2 3 3 4 4 5 5 6 6
- 1 5 2 3 4 6 2 3 3 1 2 4

Co to jsou náhodná čísla?

„Házení kostkou“ – čísla 1-6

Která z následující posloupností je více pravděpodobná?

- 1 1 2 2 3 3 4 4 5 5 6 6
- 1 5 2 3 4 6 2 3 3 1 2 4

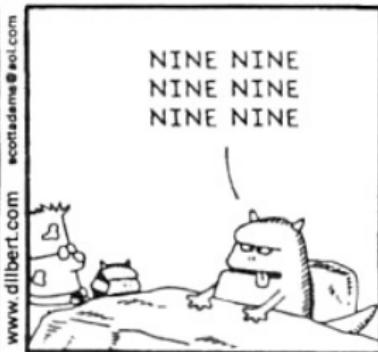
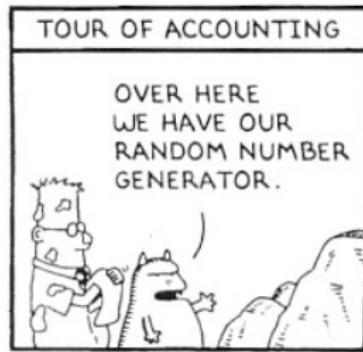
Obě mají stejnou pravděpodobnost $(\frac{1}{6})^{12}$

Úkol: (ne)náhodné posloupnosti

- máte k dispozici několik posloupností čísel „hody kostkou“ ~ celá čísla 1 až 6
- určete, které z nich jsou „nenáhodné“ a proč
- co to znamená, že posloupnost je „náhodná“?

Testování náhodnosti

DILBERT By SCOTT ADAMS



Testování náhodnosti

- nenáhodná posloupnost:
 - predikovatelná – dokážete předpovědět další číslo (lépe než náhodným tipem)?
 - zdroje nenáhodnosti např. zkreslení, korelace, vzory, periodicita
- existují rozsáhlé sady testů náhodnosti
- vztah statistické testy

Testování náhodnosti: frekvence

Frekvence čísel ve 300 hodech

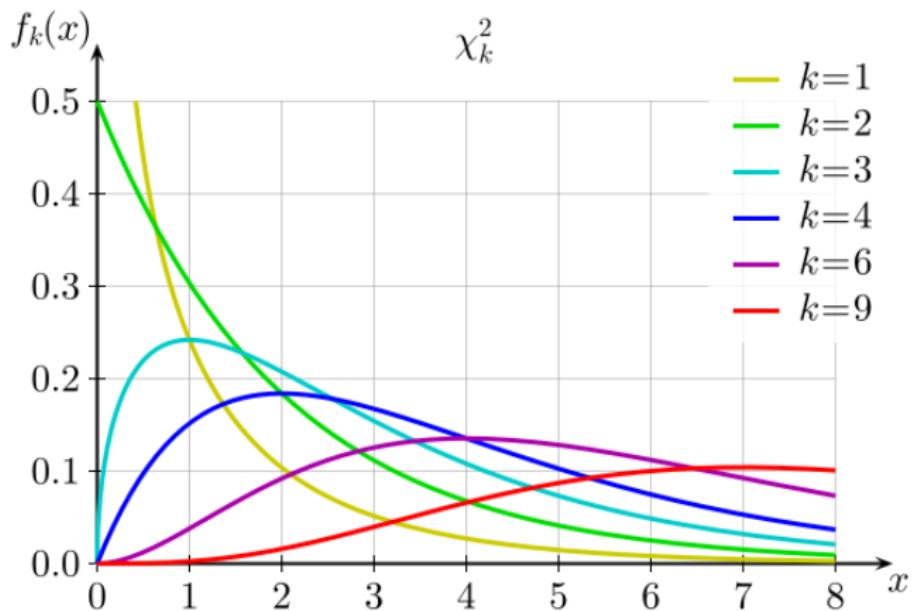
	1	2	3	4	5	6
očekávané	50	50	50	50	50	50
série 1	49	50	48	51	52	50
série 2	56	45	43	62	44	50
série 3	52	71	66	34	30	48

Odpovídá to náhodnému generování?

Testování náhodnosti: Chí kvadrát test

- O_i – očekávaný počet
- P_i – pozorovaný počet
- $S = \sum_{i=1}^6 \frac{(P_i - O_i)^2}{P_i}$
- S – pro velké n má přibližně χ^2 -rozložení o 5 stupních volnosti
- $\chi^2(k) = \sum_{i=1}^k Z_i^2$
kde Z_i má standardní normální rozdělení
- test: určíme p-hodnotu $\chi^2(5)$ pro S , pokud příliš malá – zamítnout

Chí kvadrát



Wikipedia

Centrální limitní věta

Centrální limitní věta

zjednodušeně:

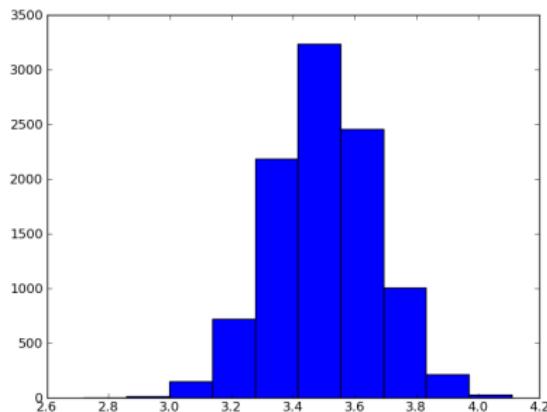
- nezávislé a identicky rozložené proměnné
- vzorky velikosti n
- pro velké n je průměr vzorku přibližně normálně rozložen

Centrální limitní věta: příklad

hody (férovou) kostkou
vzorky velikosti 100
z nich vypočítám průměr
počet vzorků 10000
jak vypadá distribuce průměrů?

Centrální limitní věta: příklad

hody (férovou) kostkou
vzorky velikosti 100
z nich vypočítám průměr
počet vzorků 10000
jak vypadá distribuce průměrů?



Centrální limitní věta: poznámky

- umožňuje modelovat mnoho „neznámých vlivů“ pomocí normální distribuce
- typický příklad – šum v datech (chyba měření):
 - předpokládáme, že šum je výsledkem mnoha dílčích vlivů
 - modelujeme pomocí normální distribuce
- pozor na:
 - předpoklad „nezávislé a identicky rozložené“
 - platí pro aritmetický průměr („aditivní“ veličiny)
 - rychlosť konvergencie závisí na výchozí distribuci

Centrální limitní věta: příklady kostky

K_a = zatížená kostka, která preferuje vyšší čísla
(pravděpodobnost úměrná počtu teček)

K_b = inverzně zatížená kostka

Jak to dopadne (rozmyslete „teoreticky“, udělejte simulaci):

- hody kostkou K_a
- pro každý hod náhodně vybereme jednu z kostek K_a , K_b
- náhodně vybereme jednu z kostek K_a , K_b a tou házíme všechna čísla ve vzorku

Věnujte pozornost tvaru výsledné distribuce, průměru i směrodatné odchylce.

Bayesova věta

pojmy:

- Bayesova věta
- prior, posterior
- likelihood – věrohodnost
- Bayesovská analýza dat

Bayesova věta

- $P(A|B)$ – podmíněná pravděpodobnost
- Bayesova věta

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayesova věta

- D – pozorovaná data
- H_i – hypotézy o vzniku dat
- $P(H_i)$ – „prior“, odhad pravděpodobnosti H_i předtím, než jsme viděli data
- $P(D|H_i)$ – pravděpodobnost dat při dané hypotéze
- $P(H_i|D)$ – „posterior“, odhad pravděpodobnosti H_i korigovaný daty
- Bayesova věta

$$P(H_i|D) = \frac{P(D|H_i)P(H_i)}{P(D)}$$

- $P(D) = \sum_i P(D|H_i)P(H_i)$ – pravděpodobnost dat

Bayesova věta – klasický příklad

- předpokládejme
 - výskyt AIDS: 6 z 1000
 - spolehlivý test na AIDS:
 - správný výsledek 99,9 % pro ty, co mají AIDS
 - 99 % pro ty, co nemají AIDS
- výsledek testu osoby X je pozitivní
- jaká je pravděpodobnost, že X má AIDS?

Bayesova věta – klasický příklad

hypotézy: $A = \text{AIDS}$, $N = \text{nemá AIDS}$

data: $V = \text{pozitivní výsledek}$

$$\begin{aligned} P(A|V) &= \frac{P(V|A)P(A)}{P(V|A)P(A)+P(V|N)P(N)} \\ &= \frac{0.006 \cdot 0.999}{0.006 \cdot 0.999 + 0.994 \cdot 0.01} \sim 0.38 \end{aligned}$$

Bayesova věta – příklad kostky

- 100 kostek, 1 falešná (samé 6), ostatní poctivé
- náhodně vytáhnu jednu kostku, 3 krát hodím, vždy padne šestka
- jaká je pravděpodobnost, že jde o poctivou kostku?

Vypočítejte:

- vzorcem (Bayes)
- simulací

Analýza dat: lineární regrese, detekce shluků

Radek Pelánek

IV122

Úvodní poznámky

- princip „simulovaná data“
- rozbor dvou konkrétních technik
 - lineární regrese
 - detekce shluků (k -means)
- průběžně ilustrace obecných principů z analýzy dat, pravděpodobnosti, strojového učení, ...

Simulovaná data – jednoduchý příklad

- zvolíme parametry μ, σ , počet dat n
- simulovaná data = vygenerujeme n bodů z normálního rozdělení s průměrem μ a směrodatnou odchylkou σ
- na základě dat odhadneme parametry m, s

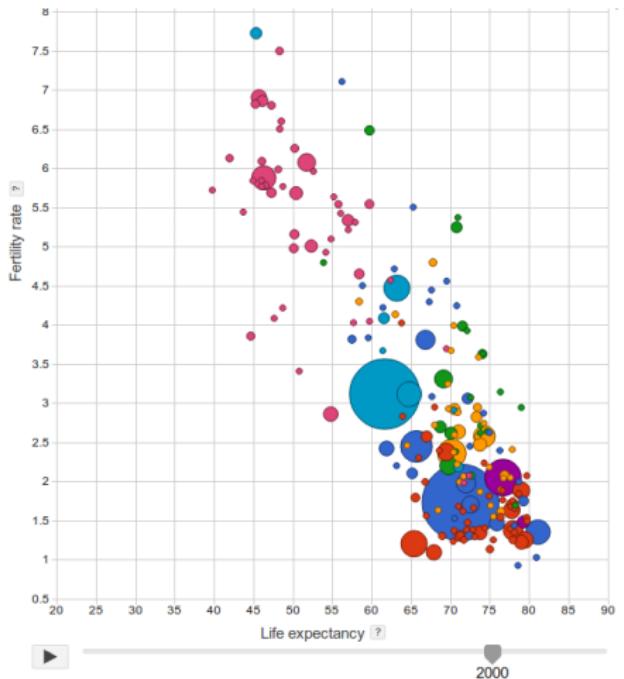
Simulovaná data – jednoduchý příklad

- zvolíme parametry μ, σ , počet dat n
- simulovaná data = vygenerujeme n bodů z normálního rozdělení s průměrem μ a směrodatnou odchylkou σ
- na základě dat odhadneme parametry m, s

co z toho:

- ujasnění metod pro odhad parametrů
- kontrola implementace
- intuitivní výhled do vztahu mezi n a přesností odhadnutých parametrů
- u složitějších modelů i „přidané“ výsledky, které nelze (snadno) získat analyticky

Reálná data: délka života, porodnost

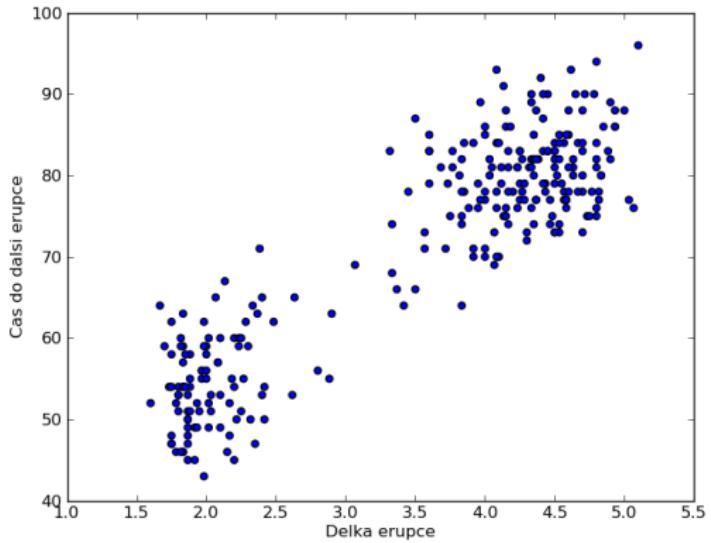


Google Public Data / World Bank

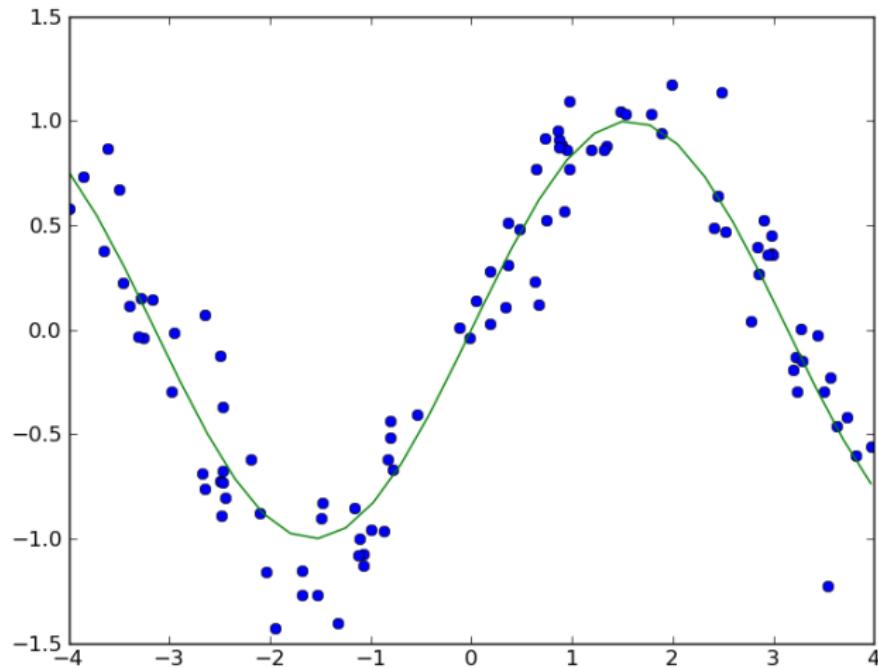
Reálná data: Old Faithful



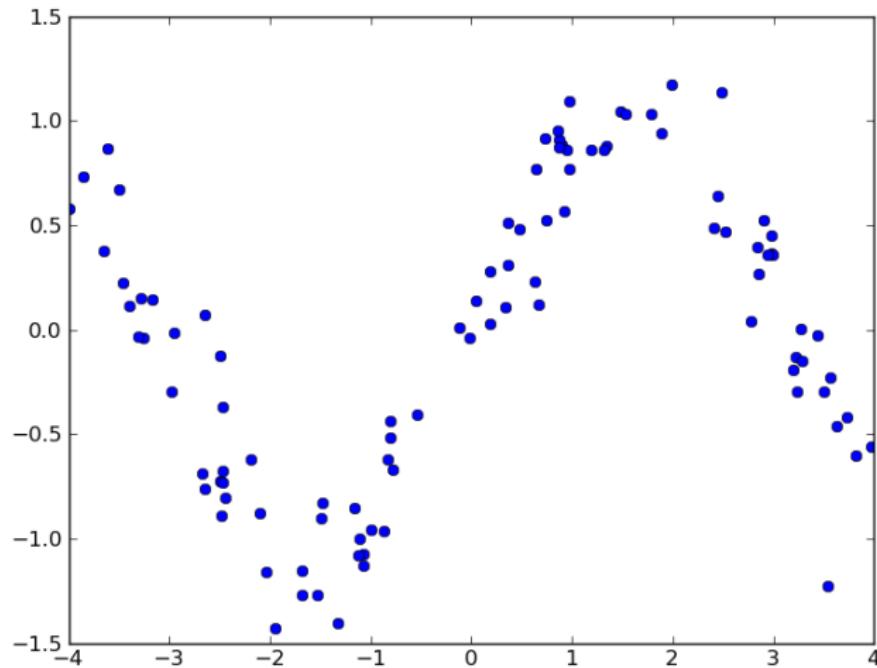
Zdroj: Wikipedia



Simulovaná data: generování



Simulovaná data: vstup pro analýzu



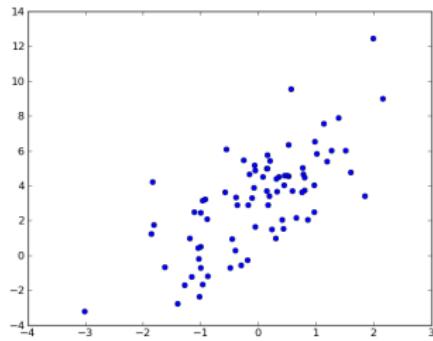
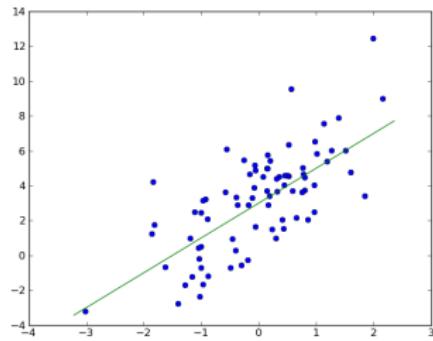
Simulovaná data

též „syntetická“ data

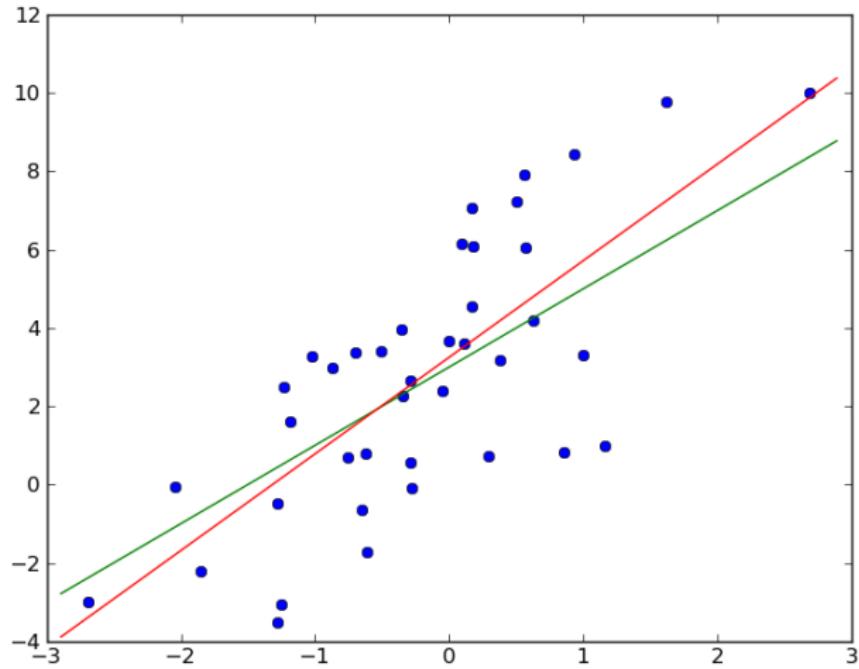
- zvolíme „správné řešení“
- vygenerujeme data: „správné řešení“ + náhodný šum
- náhodný šum \sim normální rozdělení (většinou)
- algoritmu pro analýzu dat dáme pouze vygenerovaná data
- výsledek algoritmu můžeme porovnat se správným řešením

užitečný přístup z mnoha hledisek: pochopení, ladění implementace, nastavení parametrů

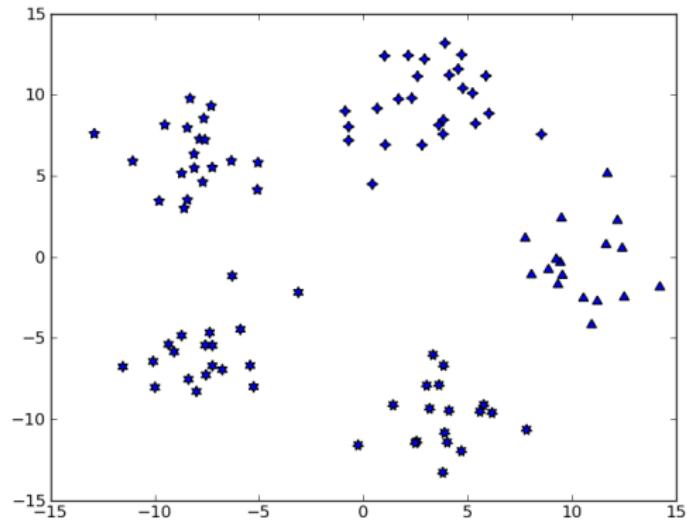
Simulovaná data: lineární regrese



Lineární regrese



Simulovaná data: Detekce shluků



Úkol

- k dispozici data pro lineární regresi a detekci shluků
- zkuste najít „co nejlepší“ přímku / rozdělení na shluky
 - ➊ co to znamená „co nejlepší“?
 - ➋ jak hledat?
- zkuste vymyslet ...
žádný Google, Wikipedie, studijní materiály

Která přímka je nejlepší?

- hledáme co nejlepší přímku $ax + b$
- minimalizace „sumy čtverců chyb“ (sum of squared error)

$$SSE = \sum_{i=1}^n (y_i - (ax_i + b))^2$$

- proč zrovna tato funkce?

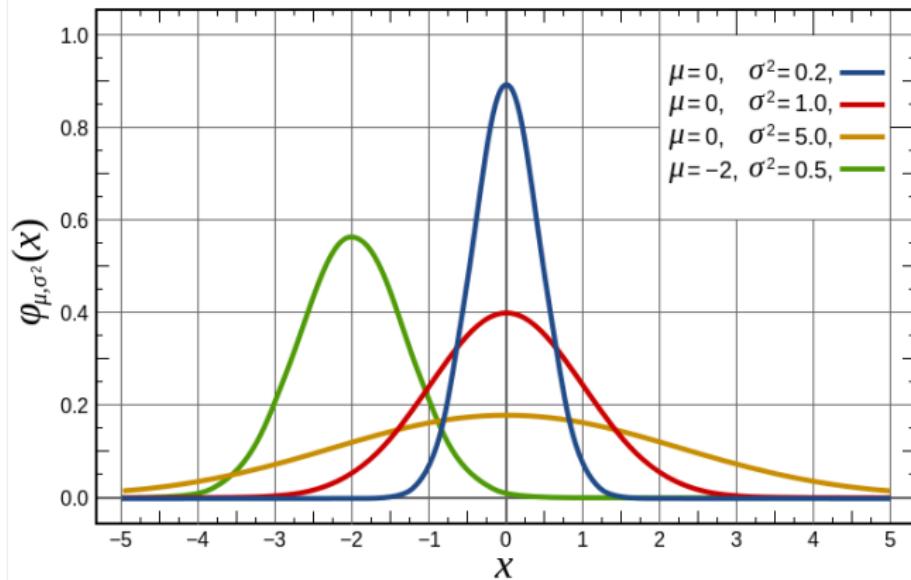
Která přímka je nejlepší?

- hledáme co nejlepší přímku $ax + b$
- minimalizace „sumy čtverců chyb“ (sum of squared error)

$$SSE = \sum_{i=1}^n (y_i - (ax_i + b))^2$$

- proč zrovna tato funkce?
- pragmaticicky: dobře se s tím pracuje
- teoreticky: nejlepší vysvětlení dat při předpokladu normálního šumu

Normální rozdělení



Wikipedia

Normální rozdělení

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- μ – průměr
- σ – standardní odchylka

Metoda maximální věrohodnosti

maximum likelihood estimation

- jaká je věrohodnost (likelihood) dat, pokud jsou generována přímkou $ax + b$?

$$L = \prod_i p(x_i, y_i) = \prod_i \mathcal{N}(ax_i + b, \sigma^2)(y_i)$$

- hledáme a, b tak, abychom maximalizovali
- vezmeme logaritmus (monotónní operace, zachovává maximum)
- maximalizovat L je to stejné jako minimalizovat sumu čtverců:

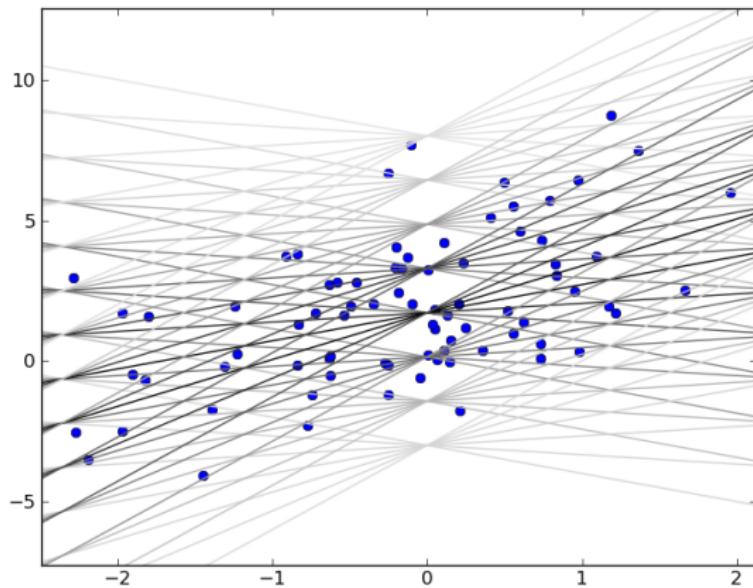
$$SSE = \sum_{i=1}^n (y_i - (ax_i + b))^2$$

Jak najít přímku minimalizující SSE?

- analytické řešení „vzorečkem“ – ideální řešení, tady funguje, u složitějších problémů však nikoliv
- pro ilustraci:
 - „grid search“ – hrubá síla
 - gradient descent – postupné vylepšování

Grid search

8 hodnot b , 7 hodnot a ; stupeň šedi \sim SSE



Analytické řešení

$$SSE = \sum_{i=1}^n (y_i - (ax_i + b))^2$$

- hledáme minimum vzhledem k a, b
- parciální derivace musí být 0

$$\frac{\partial SSE}{\partial a} = 2 \sum_{i=1}^n -y_i x_i + ax_i^2 + x_i b = 0$$

$$\frac{\partial SSE}{\partial b} = 2 \sum_{i=1}^n -y_i + ax_i + b = 0$$

Analytické řešení

Po algebraických úpravách dostaneme:

$$a = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x_i)^2} = r_{xy} \frac{s_y}{s_x}$$

$$b = \bar{y} - a\bar{x}$$

r_{xy} – korelační koeficient

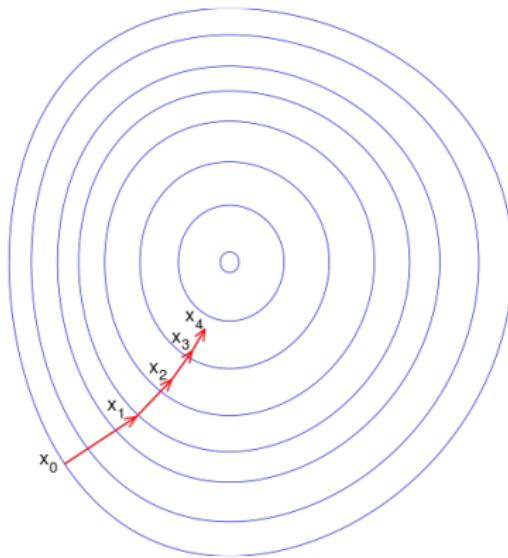
s_x, s_y – standardní odchylka x, y

Metoda největšího spádu

gradient descent

- „hladová“ metoda
 - začneme s iniciálním odhadem parametrů
 - iterativně zlepšujeme
- snažíme se o co největší lokální zlepšení = úprava hodnot parametrů ve směru spádu (gradient)
- parametr „learning rate“: velikost skoku ve směru gradientu
 - příliš malý – pomalé
 - příliš velký – nestabilní (nekonverguje)

Gradient descent: intuice



Wikipedia

Gradient descent pro lineární regresi

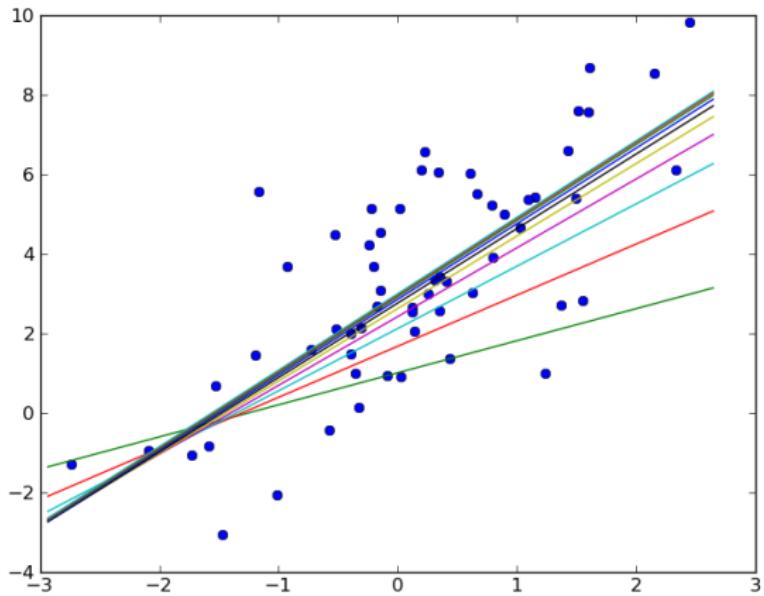
$$SSE = \frac{1}{2} \sum_{i=1}^n (y_i - (ax_i + b))^2$$

gradient:

$$\frac{\partial SSE}{\partial a} = - \sum_{i=1}^n x_i(y_i - (ax_i + b))$$

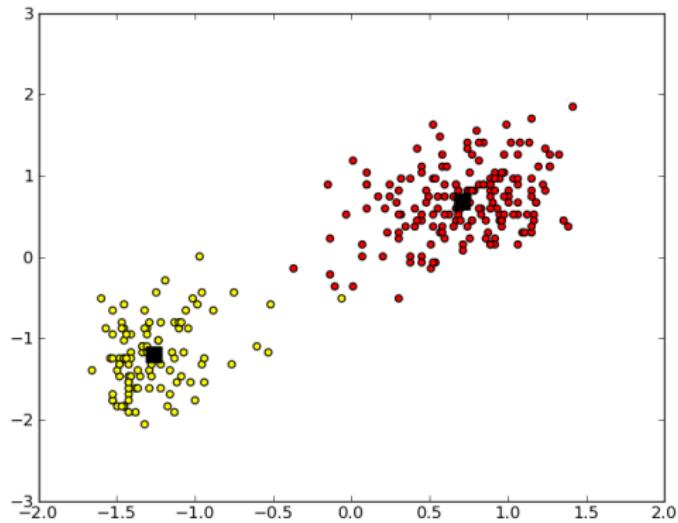
$$\frac{\partial SSE}{\partial b} = - \sum_{i=1}^n (y_i - (ax_i + b))$$

Gradient descent demo



Detekce shluků (clustering)

Old Faithful



Cíl shlukování

shlukování obecně:

- minimalizovat vzdálenosti v rámci shluku
- maximalizovat vzdálenosti mezi shluky

konkrétně např: minimalizace sum čtverců vzdáleností od středů shluků

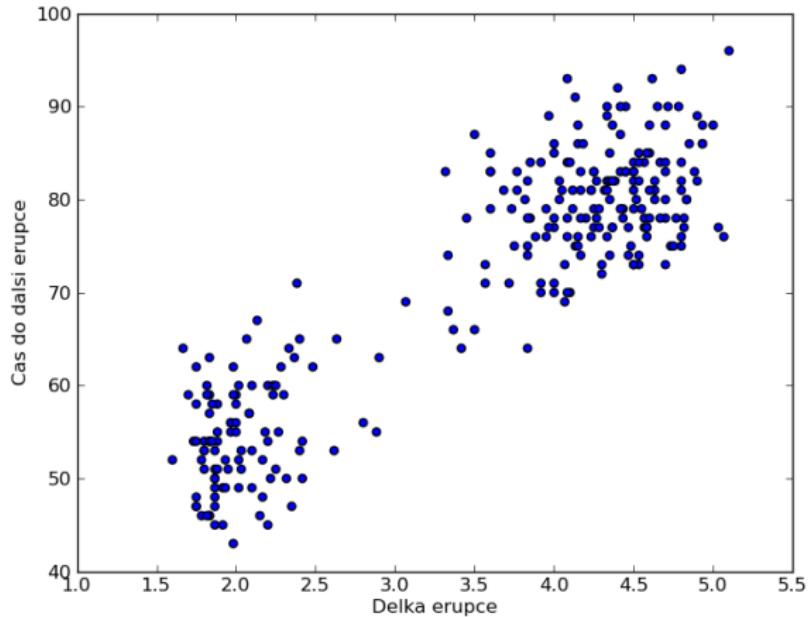
Normalizace

klíčový praktický krok: normalizace (standardizace)

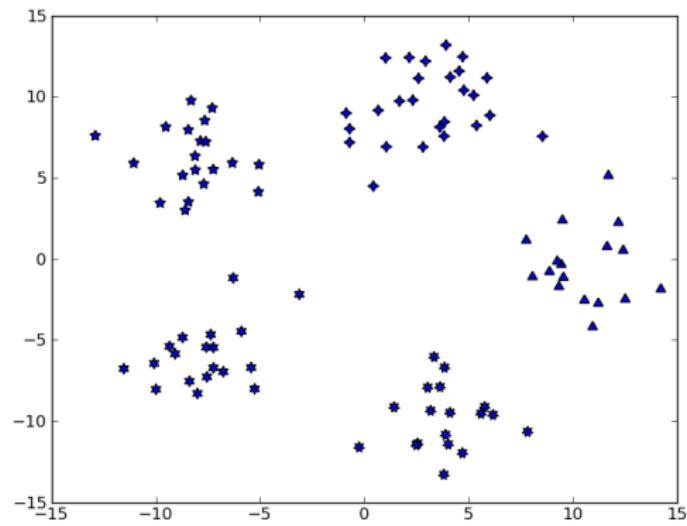
- potřebujeme data dostat na stejnou „škálu“, jinak bude dominovat jedna dimenze
- z-skóre
 - odečíst průměr
 - podělit standardní odchylkou

Význam normalizace

Old Faithful data



Detekce shluků – simulovaná data

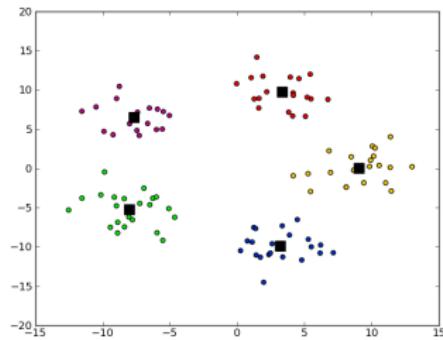
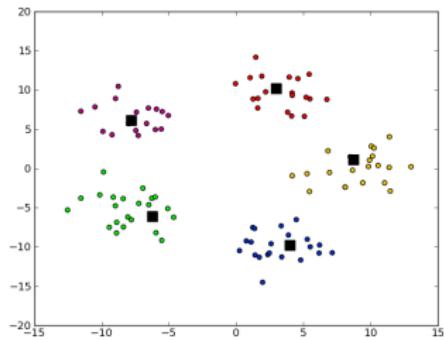
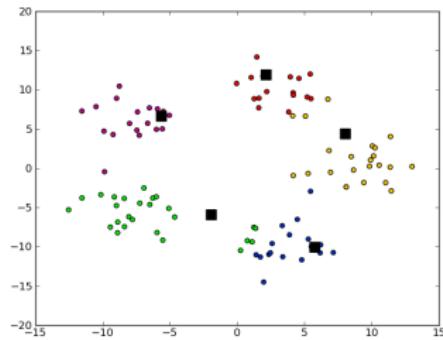
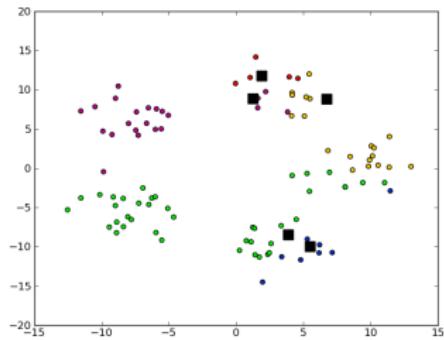


Algoritmus k -means

- vyber k „středů shluků“
- opakuj:
 - každý bod přiřad' do toho shluku, jehož střed je nejblíž
 - aktualizuj polohu středů – těžiště bodů přiřazených do shluku

<http://www.naftaliharris.com/blog/visualizing-k-means-clustering/>

Algoritmus k -means: ukázka



Algoritmus k -means – poznámky

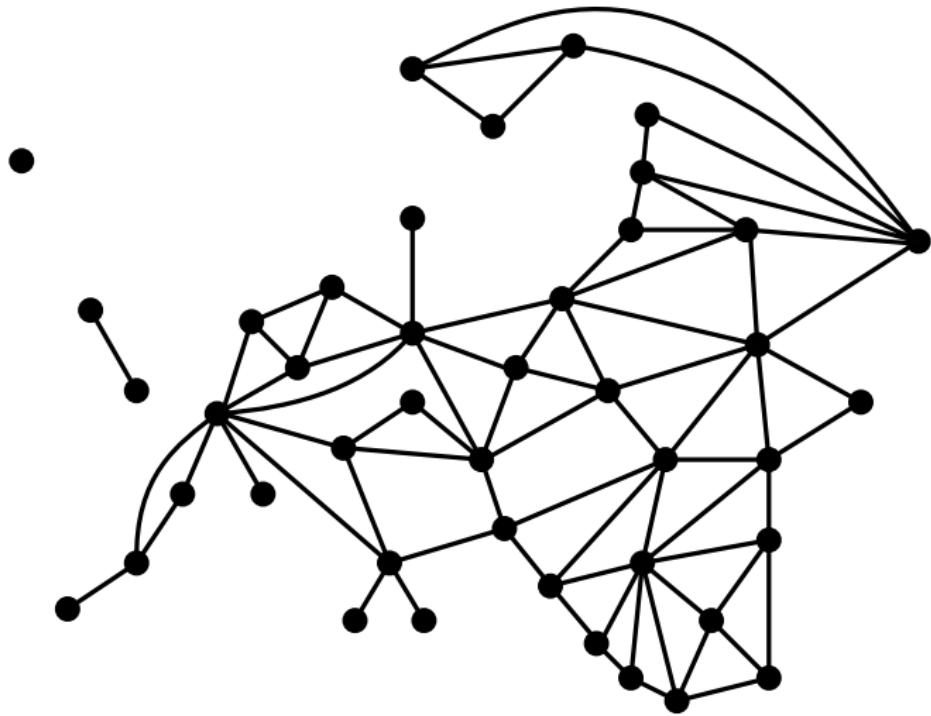
- hladová metoda
- lokální optima
- role inicializace
- opakované spuštění

Grafy a bludiště

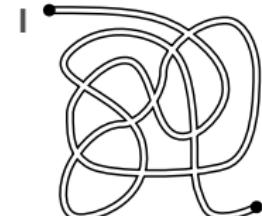
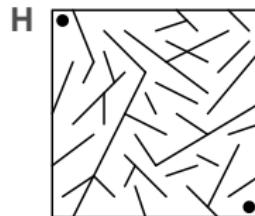
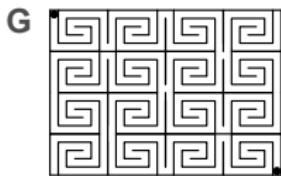
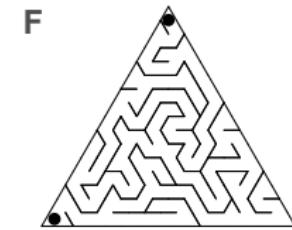
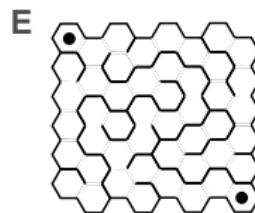
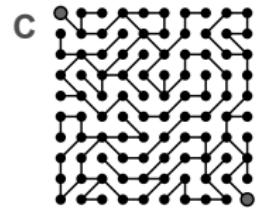
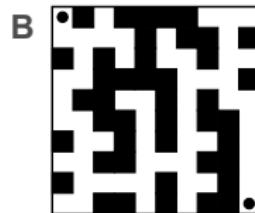
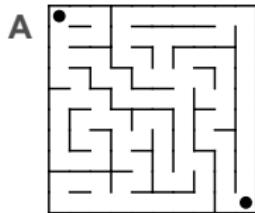
Radek Pelánek

IV122

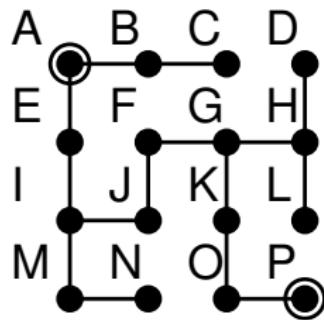
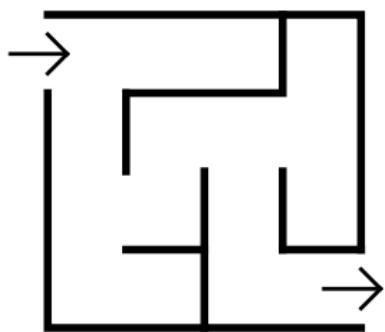
Graf



Bludiště



Bludiště a grafy



Grafy a bludiště

dnes:

- aplikace grafových pojmů na bludištích
- řešení bludišť

příště:

- generování bludišť
- netradiční „mřížky“ – aplikace geometrických pojmů

Grafy – pojmy

- vrchol, hrana, stupeň, cesta, vzdálenost
- procházení grafu: do šířky (BFS), do hloubky (DFS)
- kostra grafu
- komponenta, silně souvislá komponenta

Grafový kvíz

Lze následující problémy řešit efektivně? Jakým algoritmem?

- ① nejkratší cesta v neváženém grafu
- ② nejkratší cesta ve váženém grafu
- ③ detekce cyklu v orientovaném grafu
- ④ eulerovský tah (každá hrana právě jednou)
- ⑤ hamiltonovská kružnice (každý vrchol právě jednou)
- ⑥ nejlevnější kostra grafu

Illustrace grafových algoritmů

Pěkná interaktivní ukázka grafových algoritmů (hledání cesty):

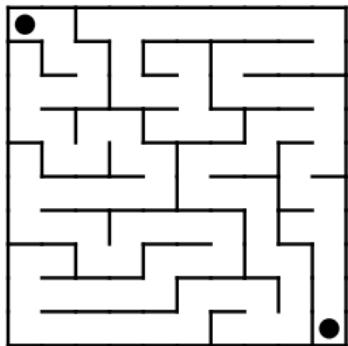
<http://qiao.github.io/PathFinding.js/visual/>

Bludiště – pojmy, typy

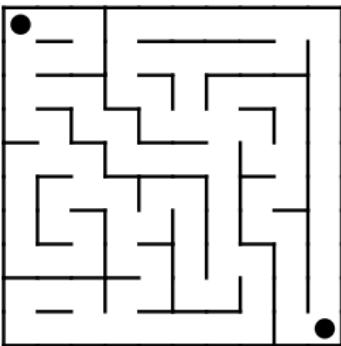
- struktura:
 - unicursular = labyrinth
 - perfektní
 - braid
 - semi-braid
- podkladová mřížka: čtvercová, trojúhleníková, „kruhová“, nepravidelná, ...
- starty a cíle: 1:1, 1:N, N:N, ...
- speciální podmínky: mosty, žebříky, zákazy zatáčení, střídání barev, ...

Typy bludišť: příklady

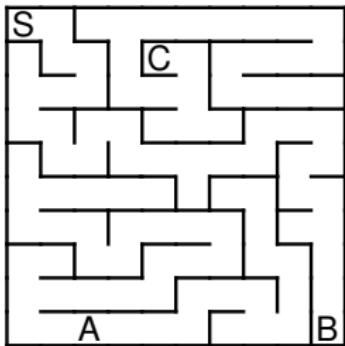
Perfektní bludiště



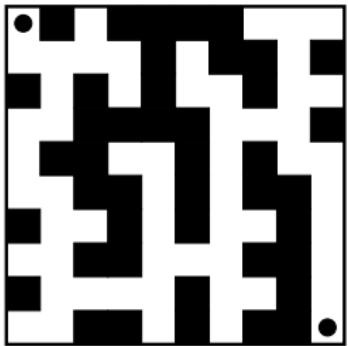
Pletenec



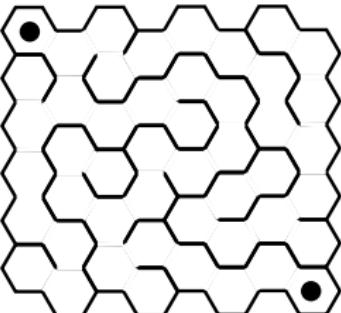
Více cílů



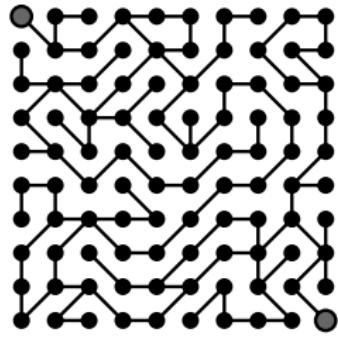
Bludiště se zdmi



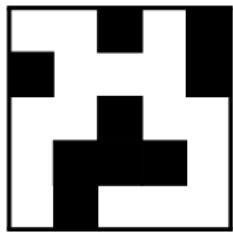
Šestiúhelníková mřížka



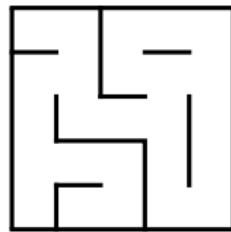
Uzly a spojnice



Reprezentace



..#.#
#. . #
..#..
.###.
. # ...



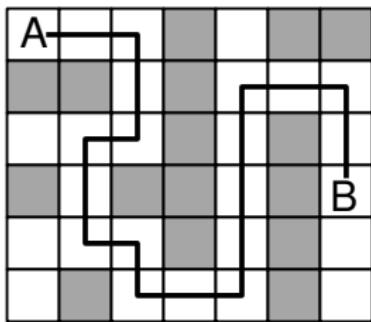
```
++++++  
|...|....|  
+-+-+,-+-+  
|...|.....|  
+-+-+,-+-+  
|...|.....|  
+-+-+,-+-+  
|.|.....|.|  
+-+-+,-+-+  
|.....|.|.|  
+-+-+,-+-+  
|.|...|...|  
+-+-+,-+-+
```

Řešení bludišť

- algoritmy „pro lidi“ (agenty v bludišti)
 - pravidlo pravé ruky – nefunguje vždy, protipříklad?
 - algoritmus se značkováním – funguje vždy
- algoritmy „pro počítače“ (agenty nad bludištěm)
 - standardní procházení grafu – BFS
 - bludišťové variace (ale i mnohé další problémy):
 - většinou správně: vymyslet vhodnou transformaci na graf, použít BFS
 - většinou možné a nevhodné: použít přímočarý graf a vymýšlet speciální grafový algoritmus

Základní bludiště

Základní zadání



Textový zápis
zadání

```
A . . # . ##  
## . # . . .  
. . . # . # .  
# . ## . # B  
. . . # . # .  
. # . . . # .
```

Textový zápis
řešení

```
A x x # . ##  
## x # x x x  
. x x # x # x  
# x # # x # B  
. x x # x # .  
. # x x x # .
```

Bludištové variace

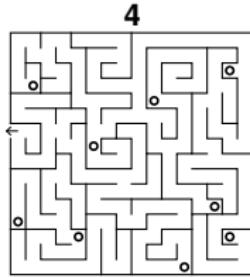
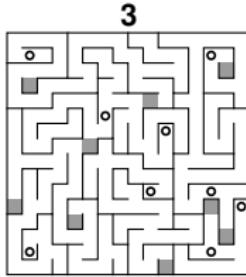
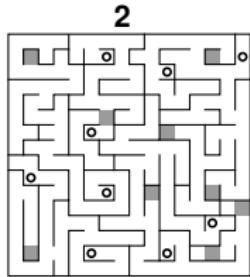
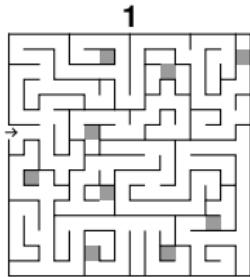
- Trojrozměrné bludiště
- Robot v bludišti
- Bludiště a dynamit
- 3 lampy v bludišti
- Zákaz zatáčení vlevo
- Bludiště s kuličkou
- Sokoban
- Číselné bludiště
- Barevné bludiště
- Kvantové bludiště

Jak formulovat jako grafový problém?

Trojrozměrné bludiště

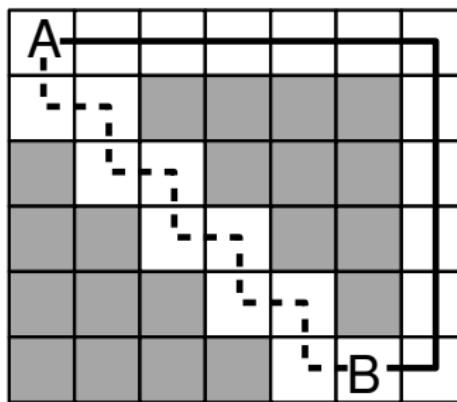
šedé pole = žebřík nahoru

kolečko = žebřík dolů



Robot v bludišti

Akce robota: krok, otočení o 90 stupňů
 $A \rightarrow B$, minimalizovat počet akcí

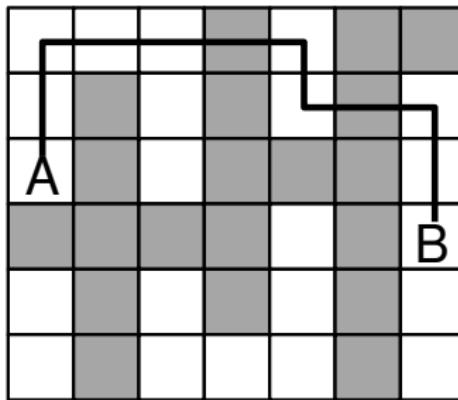


Optimální řešení: plná čára

Bludiště a dynamit

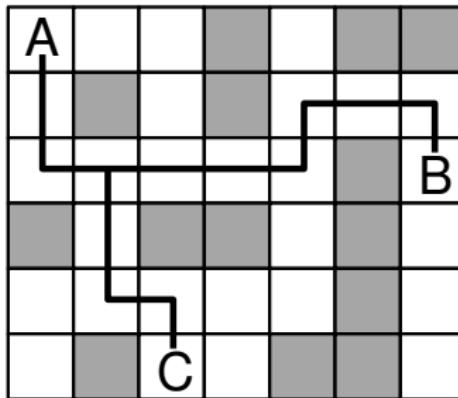
$A \rightarrow B$, minimalizovat:

- primárně počet „odpalených“ zdí
- sekundárně počet kroků



3 lampy v bludišti

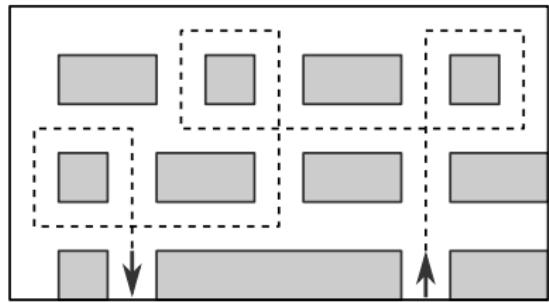
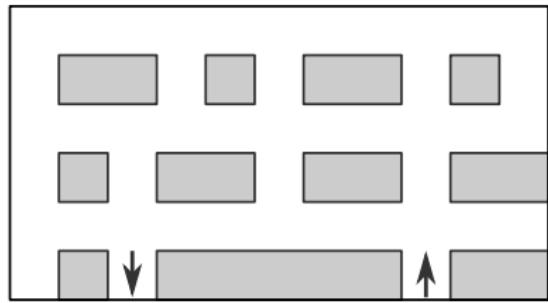
Spojit lampy drátem (co nejkratším)



Co když je lamp n ?

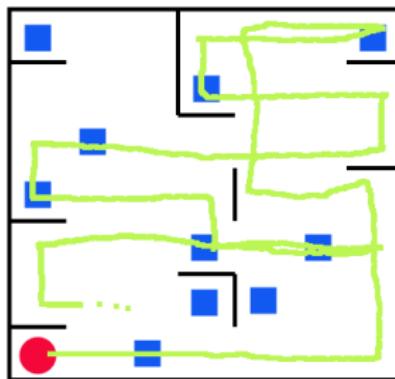
Zákaz zatáčení vlevo

„No left turn maze“



Bludiště s kuličkou

kulička: pohyb „k zarážce“
posbírat všechny značky

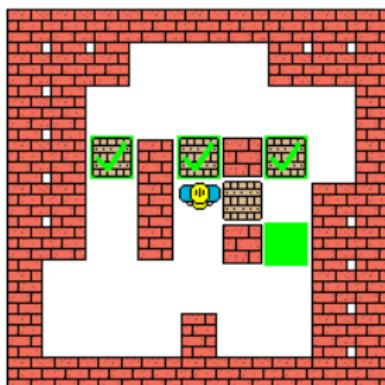
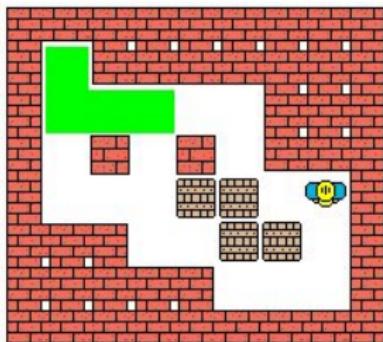


K vyzkoušení: umimematiku.cz

Sokoban

Dostat bedny na vyznačená pole.

Panáček může pouze tlačit, vždy jen 1 bednu.



K vyzkoušení: umimematiku.cz

Číselné bludiště

pravý horní roh → levý dolní roh

skoky vertikálně a horizontálně, číslo = délka skoku

2	4	4	3	3
2	3	3	2	3
3	2	3	1	3
2	2	3	2	1
1	4	4	4	★

3	4	3	2	2
4	4	2	2	4
4	3	1	4	2
2	1	1	3	3
1	4	2	3	★

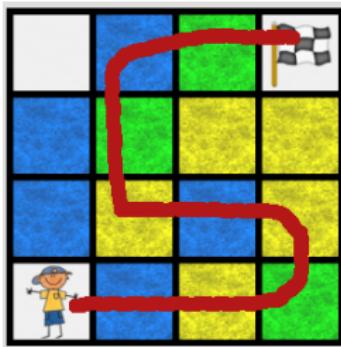
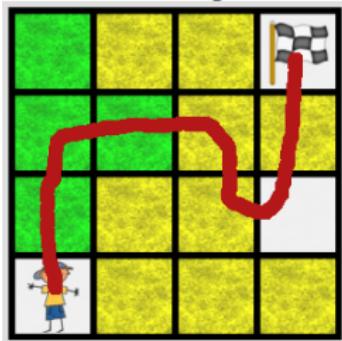
1	4	2	1	1
2	4	2	3	2
2	3	4	2	4
3	3	3	3	2
2	4	2	2	★

Jak ověřit, zda má úloha jednoznačné nejkratší řešení?
K vyzkoušení: umimematu.cz

Barevné bludiště

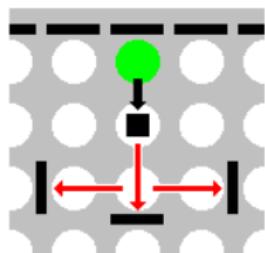
levý dolní roh → pravý horní roh

stejný počet polí od každé barvy (bílá libovolně)



The quantum maze

Bludiště, které se generuje podle pohybu...



<http://www.clickmazes.com/quantum/ixquantum.htm>

<https://www.fi.muni.cz/~xpelanek/sendvic13/blx87ac/>

Úkoly k zamyšlení

formulujte úlohu jako grafový problém:

- graf:
 - vrcholy
 - hrany – orientované? vážené?
- algoritmus:
 - prohledávání do šířky (BFS)
 - jiný algoritmus
- odhadněte velikost grafu
- Lze řešit efektivně? Je nutné použít heuristiky?

Úloha, technický tip

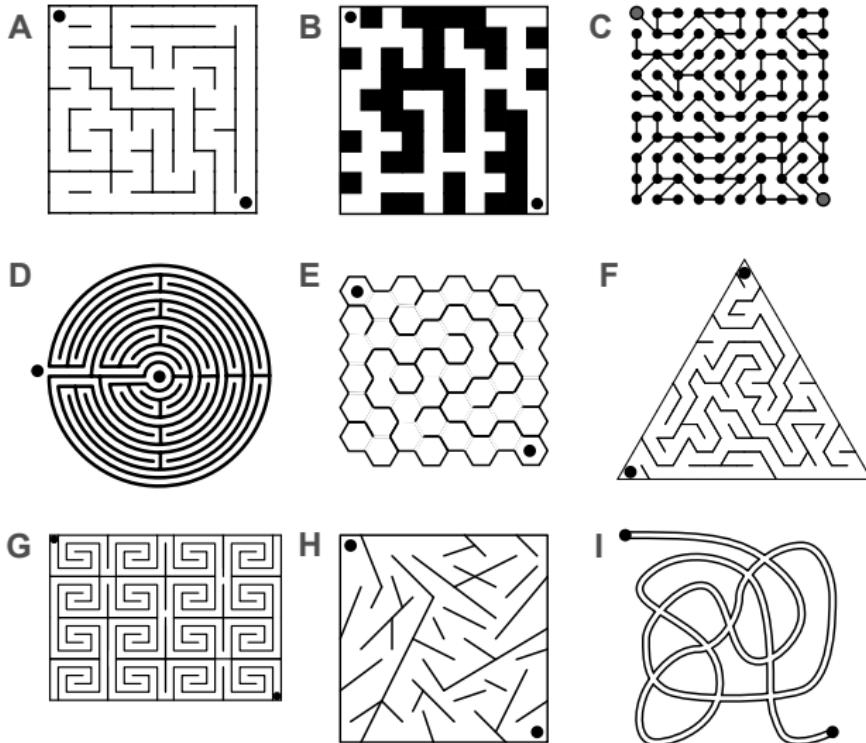
- úloha: řešič některé z bludišťových variant
- znázornění řešení: textová grafika, obrázek (SVG), animace
- jak snadno udělat animaci:
 - jednotlivé kroky řešení: obrázky v SVG
 - convert
 - avconv

Generování bludišť

Radek Pelánek

IV122

Bludiště



Bludiště – pojmy, typy

- struktura:
 - unicursular = labyrinth
 - perfektní
 - braid
 - semi-braid
- podkladová mřížka: čtvercová, trojúhelníková, „kruhová“, nepravidelná, ...
- starty a cíle: 1:1, 1:N, N:N, ...
- speciální podmínky: mosty, žebříky, zákazy zatáčení, střídání barev, ...

Generování bludišť

Cíl: vygenerovat bludiště

- zadané parametry (např. velikost, obtížnost)
- náhodnosteně (každý běh jiný výstup)

Nápady?

Generování bludišť – základní přístupy

- bourání zdí
 - začneme s vyplněnou mřížkou
 - prokopáváme chodby
 - kontrolujeme, abychom to s prokopáváním nepřehnali (např. cykly)
- přidávání zdí
 - začneme s prázdnou mřížkou
 - přidáváme zdi
 - kontrolujeme souvislost

Generování perfektních bludišť

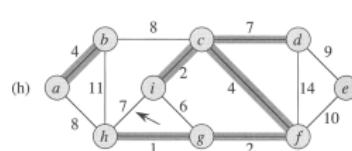
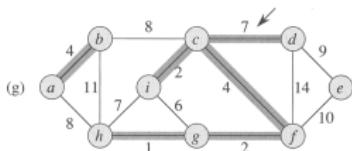
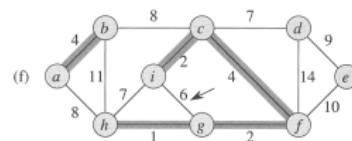
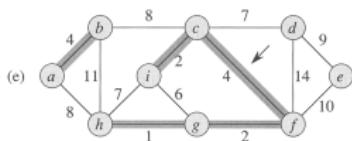
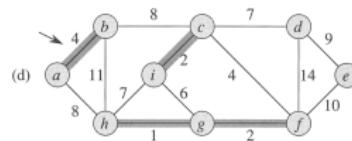
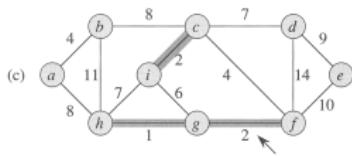
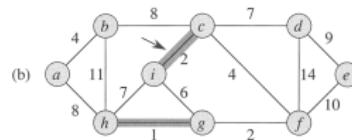
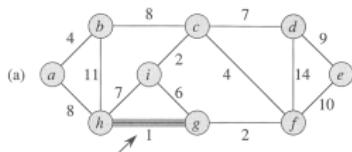
základní problém:

- čtvercová mřížka
- perfektní bludiště
- 1 start, 1 cíl

základní algoritmy:

- náhodnostní DFS
- náhodnostní generování kostry – Prim, Kruskal

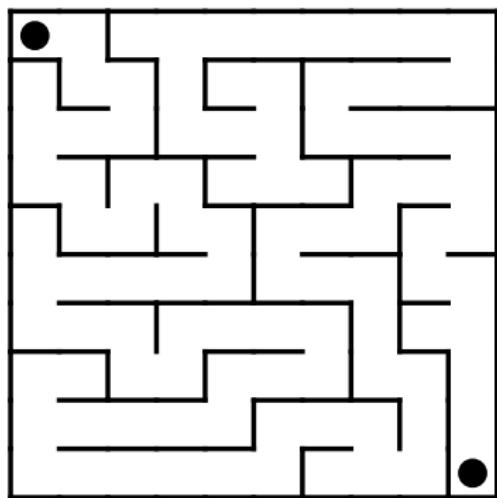
Připomenutí: Kruskalův algoritmus



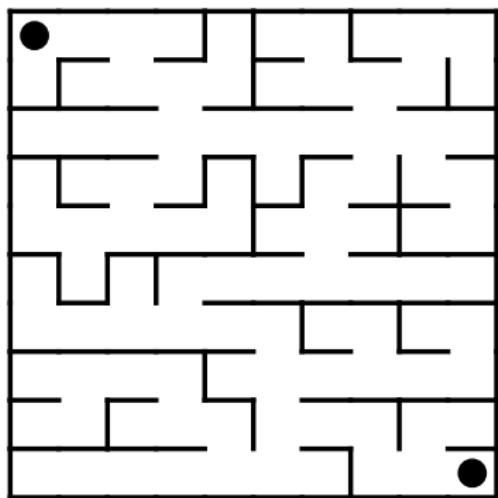
T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: Introduction to Algorithms.

Generování perfektních bludišť

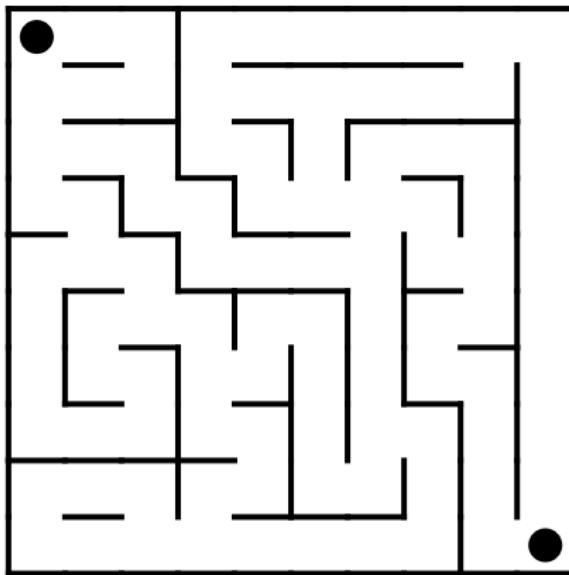
Prohledávání do hloubky



Kruskalův algoritmus



Generování bludišť: „braid“



Generování „braid“ bludišť

základní myšlenka:

- v náhodném pořadí přidávat zdi
- kontrolovat, zda nevznikají slepé cesty

komplikace: vznik „náměstí“

řešení: v prvním kroku přidat ke každému bodu alespoň 1 zed'

Další algoritmy a zdroje

další algoritmy: rekurzivní půlení, hunt and kill, Eller, ...

Pro zájemce:

- <http://www.astrolog.org/labyrnth/algrithm.htm>
- DP Algoritmy pro generování a řešení bludišť, Petr Matějka
- DP Automated Maze Generation and Human Interaction, Martin Foltin

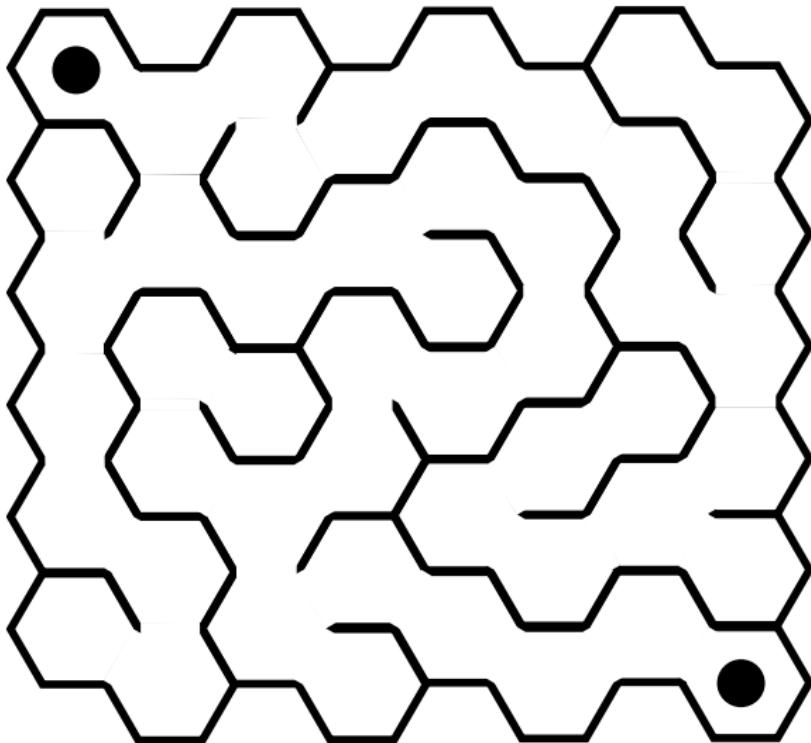
Jiné mřížky

- trojúhelníková
- šestiúhelníková
- kruhové bludiště
- další „dlaždění“ (tilings)
- nepravidelné, „roztřesené“

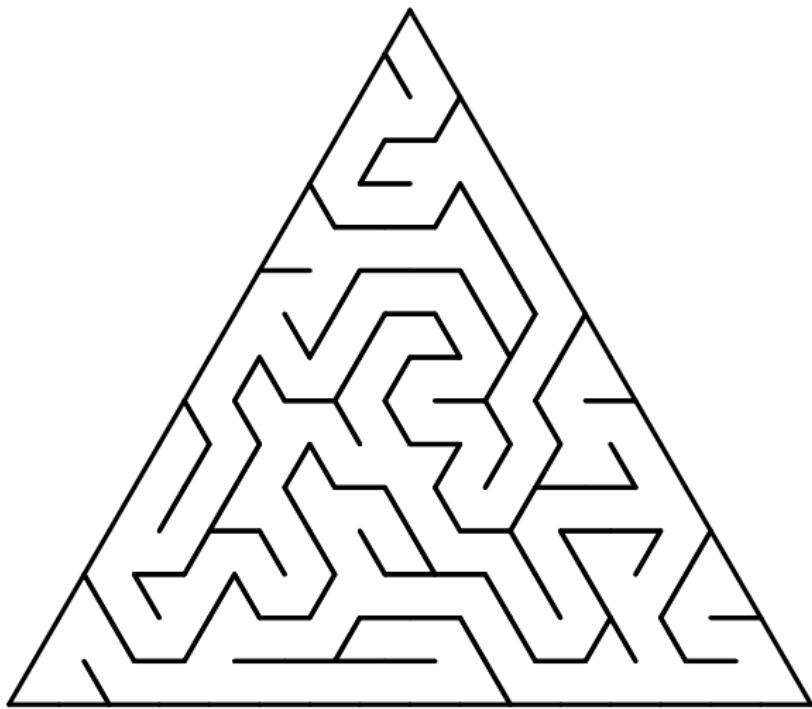
Jak určíme polohu bodů? Kdo jsou sousedi v grafu?

Tip: možnost využití lineárních transformací

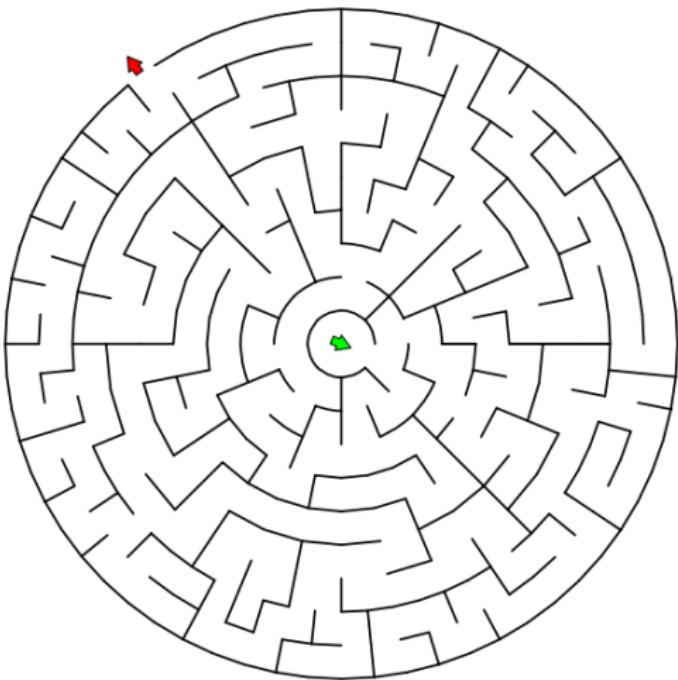
Šestiúhelníková mřížka



Trojúhelníková mřížka



Kruhové bludiště

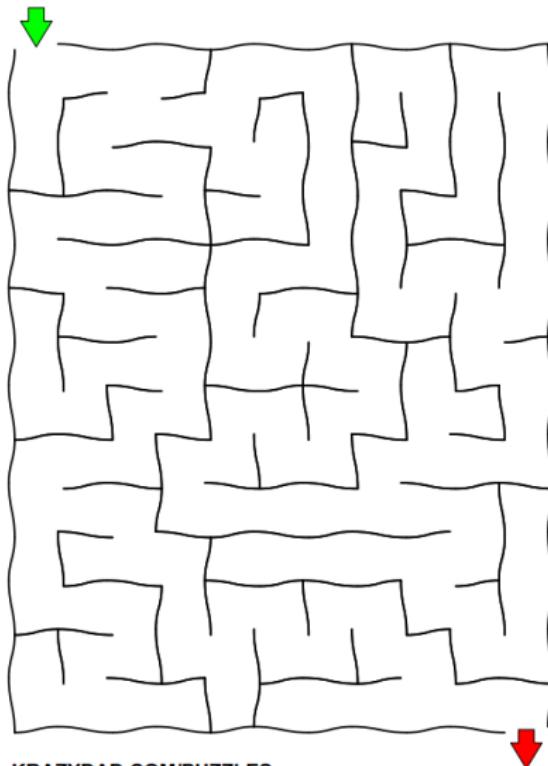


<http://krazydad.com>

Čtvercová mřížka + posuny

Easy Mazes by KrazyDad, Book 1

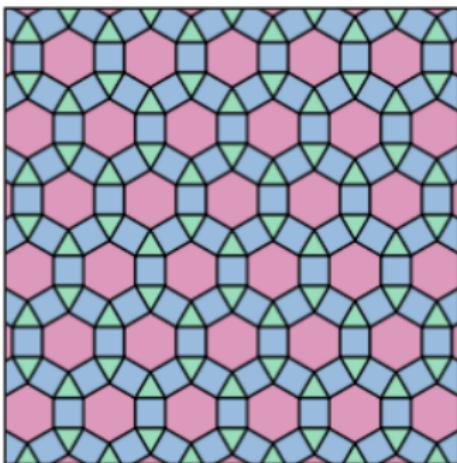
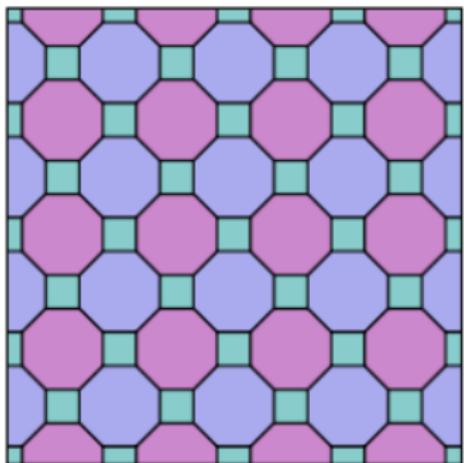
Maze #2



KRAZYSNAD.COM/PUZZLES
Need the answer? <http://krazysdad.com/mazes/answers>

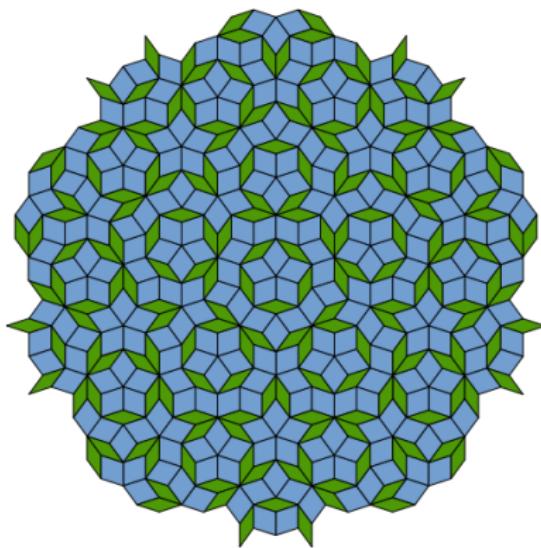
© 2010 KrazyDad.com

Další pravidelná „dlaždění“



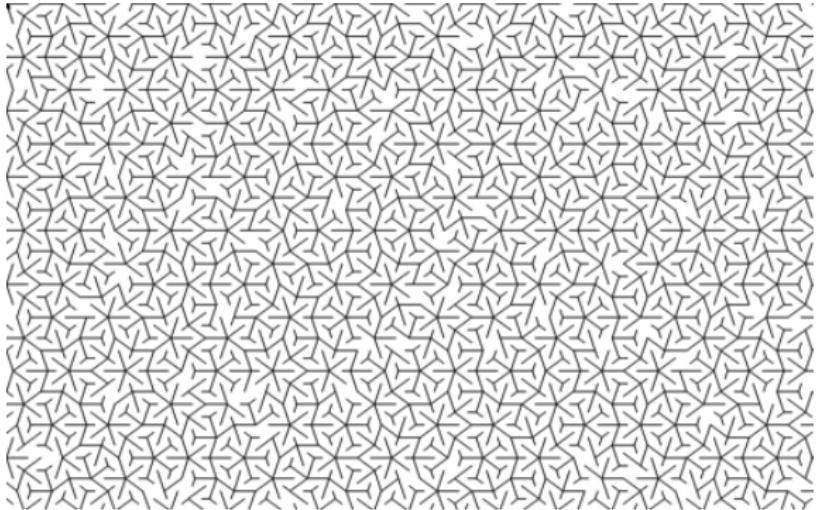
http://en.wikipedia.org/wiki/Tiling_by_regular_polygons

Penrose – neperiodické dláždění



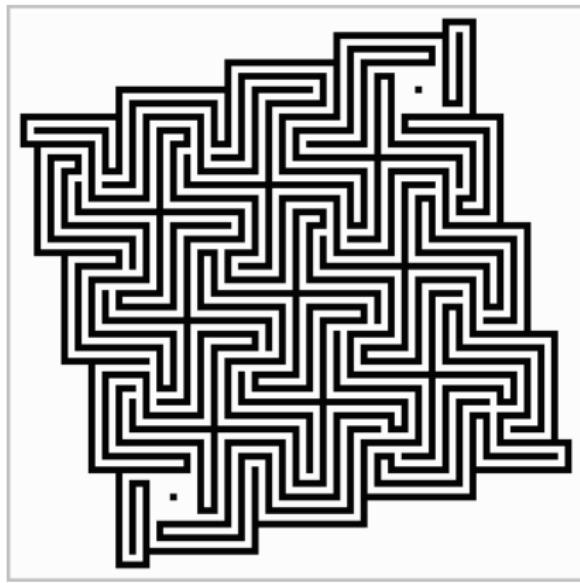
http://en.wikipedia.org/wiki/Penrose_tiling

Penrosovo bludiště



<http://justinpombrio.net/code/penrose-maze/>

Další náměty: pravidelná struktura



<http://www.clickmazes.com>

Další náměty: Bludištové variace

- Bludištové úlohy z minule
- Jak generovat (zajímavá) zadání?

Úkol, postup

- domácí úkol – volba:
 - perfektní bludiště na jiné než čtvercové mřížce
 - bludiště typu „braid“
 - složitější variace
- doporučený postup
 - nejprve perfektní bludiště na čtvercové mřížce
 - zapsat kompaktně (lze pod 50 řádků v Pythonu)