

Fikiina

Statnicová

Sněška

Ni-spol

1. Teorie grup: Grupoidy, pologrupy, monoidy a grupy. Podgrupy, cyklické grupy a jejich generátory.

NI-MPI

- **Grupoid** – uspořádaná dvojice (M, \circ)
 - o M = libovolná neprázdná množina
 - o \circ = binární operace nad M (tzv. kule)
 - **Pologrupa** – grupoid, pro který je \circ asociativní
 - **Monoid** – pologrupa, ve které existuje neutrální prvek $e: \forall a \in M: e \circ a = a \circ e = a$
 - o V monoidu existuje právě jeden neutrální prvek
 - **Grupa** – monoid, ve kterém ke každému $a \in M$ existuje inverzní prvek $a^{-1}: a^{-1} \circ a = a \circ a^{-1} = e$
 - o V grupě má každý prvek právě 1 inverzní prvek
 - **Abelovská grupa** – grupa, kde \circ je komutativní
- Uzavřená \rightarrow grupoid \rightarrow asociativní \rightarrow pologrupa $\rightarrow e \rightarrow$ monoid \rightarrow inverze \rightarrow grupa \rightarrow komutativní \rightarrow Abel. grupa
- **Podgrupa grupy** $G = (M, \circ)$ je $H = (N, \circ): N \subseteq M, (N, \circ)$ je grupa
 - o V každé grupě s alespoň 2 prvky existují alespoň 2 podgrupy
 - Triviální podgrupy $(\{e\}, \circ), G = (M, \circ)$
 - Ostatní podgrupy jsou **vlastní podgrupy**
 - o **Průnik** podgrup je podgrupa
 - o **Kritérium** podgrupovosti: $H = (N, \circ)$ je podgrupa G , právě když $\forall a, b \in N: a \circ b^{-1} \in N$
 - o **Neutrální prvek** podgrupy je roven neutrálnímu prvku grupy
 - o **Inverze** prvku v podgrupě je stejná, jako inverze stejného prvku v grupě
 - **Řád grupy** $G = (M, \circ)$ – počet prvků množiny M – $\#G$
 - o Podle řádu se dělí na **konečné a nekonečné grupy** (nekonečná M)
 - o **Lagrangeova věta** – buď H podgrupa konečné grupy G , potom řád H dělí řád G
 - o **Sylowova věta** – buď G grupa konečného řádu n a číslo p prvočíselný dělitel čísla n . Pokud p^k dělí n (pro k přirozené), pak grupa G obsahuje podgrupu řádu p^k
 - $G = (M, \circ), N \subset M \neq \emptyset \Rightarrow \langle N \rangle := \cap \{H: H \text{ je podgrupa grupy } G \text{ obsahující } N\}$ je podg. G obsah. N
 - **Grupa generovaná množinou** – podgrupa $\langle N \rangle$ grupy $G = (M, \circ), N \subseteq M$
 - o Množina N je **generující množina** grupy N
 - o Pro jednoprvkovou množinu $N = \{a\}$ zavádíme značení $\langle a \rangle := \langle \{a\} \rangle, a = \text{generátor } \langle a \rangle$
 - o $\langle N \rangle$ je nejmenší podgrupa G obsahující množinu N
 - o Všechny prvky $\langle N \rangle$ lze získat pomocí „**grupového obalu**“ $\langle N \rangle = \{a_1^{k_1} \circ a_2^{k_2} \circ \dots \circ a_n^{k_n} : n \in \mathbb{N}, k_i \in \mathbb{Z}, a_i \in N\}$
 - o **Generátor** = prvek, jehož **mocněním** dostaneme všechny prvky grupy
 - Grupa \mathbb{Z}_n^+ je rovna $\langle k \rangle, k \in \mathbb{Z}_n^+$ právě když k a n jsou nesoudělná čísla
 - **Cyklická grupa** – existuje prvek $a \in M: \langle a \rangle = G, a = \text{generátor}$ cyklické grupy G
 - **Řád prvku** – buď g prvek grupy G . Pokud existuje $\in \mathbb{N}^+: g^m = e$, pak **nejmenší** m s touto vlastností je **řád prvku** g . Pokud takové m neexistuje, řád prvku je nekonečno
 - o Řád prvku g $\text{ord}(g)$ je roven řádu grupy $\langle g \rangle: \text{ord}(g) = \# \langle g \rangle$
 - o \mathbb{Z}_n^\times je cyklická, právě když $n = 2, 4, p^k, 2p^k$, kde p je liché prvočíslo a $k \in \mathbb{N}^+$
 - **Jak najít všechny generátory**
 - o Je-li (G, \circ) cyklická grupa řádu n a a nějaký její generátor, potom a^k je také generátor tehdy, a jen tehdy, když k a n jsou nesoudělná ($\text{gcd}(k, n) = 1$)
 - o V cyklické grupě řádu n je počet generátorů roven $\varphi(n)$
 - φ = **Eulerova funkce** – každému $n \in \mathbb{N}$ přiřazuje počet přirozených čísel menších než n , které jsou s ním nesoudělná
 - Takže pro prvočíslo p je \mathbb{Z}_p^\times cyklická grupa řádu $p - 1$ a má $\varphi(p - 1)$ generátorů,
 - Libovolná podgrupa cyklické grupy je opět cyklická grupa
 - **Malá Fermatova věta** – pro libovolné p a libovolné $1 \leq a < p: a^{p-1} \equiv 1 \pmod{p}$
 - o Z důsledku Lagrangeovy věty: $\forall a \in M: a^n = e$, kde e je neutrální prvek

2. Tělesa a okruhy: Základní definice a vlastnosti. Konečná tělesa. Okruhy polynomů, ireducibilní polynom.

NI-MPI

- Okruh – $R = (M, +, \cdot)$, kde M je neprázdná množina a $+$, \cdot binární operace na ní a platí:
 1. $(M, +)$ je **abelovská grupa** = **aditivní grupa** okruhu R
 - Neutrální prvek = **nulový prvek** – značí se 0
 - Inverzní prvek vůči $+$ k $a \in M$ značíme $-a$
 - Lze definovat odčítání: $a - b = a + (-b)$
 2. (M, \cdot) je monoid = **multiplikativní monoid** okruhu R
 - Je-li \cdot komutativní, je R komutativní okruh
 - Neutrální prvek = **jednička**, značení 1
 3. Platí **distributivní zákon**: $\forall a, b, c \in M: (a(b + c) = ab + ac \wedge (b + c)a = ba + ca)$
- Základní vlastnosti okruhu
 - o Násobení nulovým prvkem dává nulový prvek
 - o Levý i pravý distribuční zákon pro odečítání: $c(b - a) = cb - ca$
- **Obor integrity** – okruh, ve kterém neexistují dělitelé nuly
 - o **Dělitelé nuly** = nenulové prvky $a, b \in M: a \cdot b = b \cdot a = 0$
- **Těleso** – okruh $T = (M, +, \cdot)$, kde $(M \setminus \{0\}, \cdot)$ je abelovská grupa
 - o Tuto grupu nazýváme **multiplikativní grupou** tělesa T
 - o Pokud pro a, b z tělesa T platí $ab = 0$, potom $a = 0$, nebo $b = 0$
 - Každé těleso je oborem integrity
- **Zobrazení h z okruhu/tělesa R do okruhu/tělesa S** je **homomorfismus** těchto okruhů/těles, jestliže je h homomorfismem příslušných aditivních a multiplikativních monoidů/grup a platí $h(1_R) = 1_S$
 - o Je-li navíc **h bijekce** (prosté a na), jedná se o **izomorfismus** těchto okruhů/těles
 - o Tělesa T a K nazýváme **izomorfní**, právě když existuje izomorfismus $T \rightarrow K$. V tomto případě je těleso T izomorfní s tělesem K
- **Konečné těleso** – těleso, které má konečný počet prvků
- **Řád tělesa** = počet prvků tělesa
- Základní příklad konečného tělesa – množina $\mathbb{Z}_p = \{0, 1, \dots, p - 1\}$ s operacemi **modulo prvočíslo p**
 - o $(\mathbb{Z}_p, +, \cdot)$ – aditivní grupa \mathbb{Z}_p^+ , multiplikativní grupa \mathbb{Z}_p^\times
 - \mathbb{Z}_p^+ – řád p
 - Každý nenulový prvek je její **generátor**
 - Je grupou i pro neprvočíselné p
 - \mathbb{Z}_p^\times – řád $p - 1$ – není prvočíslo
 - Je **cyklická**
 - Počet generátorů závisí na řádu, je roven $\varphi(p - 1)$
- **Řád konečného tělesa** musí být mocnina prvočísla p^n , kde p je prvočíslo a n je kladné celé číslo
 - o Všechna tělesa řádu p^n jsou **navzájem izomorfní**
- **Galois field** – těleso s p^n prvky – **$GF(p^n)$**
 - o Každé konečné těleso je Galois Field
 - o Prvočíslo p = charakteristika tělesa $GF(p^n)$
 - o $GF(p^n)$ – aditivní grupa
 - Řád p^n
 - Neutrální prvek $0 = 00 \dots 0 = 0^n$
 - Pro $n > 1$ není cyklická
 - o $GF(p^n)$ – multiplikativní grupa
 - Řád $p^n - 1$ – zas vynechávám 0, všechno by požrala
 - Neutrální prvek: $00 \dots 1 = 0^{n-1}1$
 - Inverzi lze nalézt pro každý prvek s REA v polynomiálním čase
 - Je vždy cyklická

- **Polynom nad okruhem**

$$P(x) = \sum_{i=0}^n a_i x^i$$

- Nad okruhem R ; $a_i \in R$; $i = 0, 1, \dots, n$
- $a_i \dots$ koeficienty polynomu $P(x)$ – o ty nám jde
- $x \dots$ **formální proměnná** polynomu $P(x)$
- Pokud pro $P(x)$ existuje $k \in \{0, 1, \dots, n\}$: $a_k \neq 0$, pak největší z k = **stupeň polynomu** $P(x)$, značeno $\deg(P(x))$
- $P(x) = 0 \dots$ nulový polynom – nedefinovaný stupeň
- Abychom mohli dělat operace s polynomy, potřebujeme je umět s jejich koeficienty – lze vybudovat okruh polynomů nad libovolným okruhem (i tělesem)

- **Okruh polynomů** – množina všech polynomů nad okruhem R spolu s operacemi **sčítání** a **násobení** definovanými předpisy

$$\sum_{i=0}^n a_i x^i + \sum_{i=0}^n b_i x^i = \sum_{i=0}^n (a_i + b_i) x^i$$

$$\left(\sum_{i=0}^n a_i x^i \right) \cdot \left(\sum_{i=0}^m b_i x^i \right) = \sum_{i=0}^{n+m} \left(\sum_{j+k=i} a_j b_k \right) x^i$$

kde $a_i, b_i \in R$, tvoří okruh polynomů nad okruhem $R - R[x]$

- **Násobení** polynomů: buď T těleso a $f(x), g(x) \in T[x]$ nenulové polynomy. Platí
 $\deg(f(x)g(x)) = \deg(f(x)) + \deg(g(x))$
- **Dělení** polynomů: buď T těleso a $f(x), g(x) \in T[x]$ nenulové polynomy. Pak existují jednoznačně určené polynomy $q(x), r(x) \in T[x]$ takové, že
 $f(x) = q(x)g(x) + r(x)$
 kde $r(x)$ je buď nulový, nebo má stupeň ostře menší než stupeň $g(x)$
- **Bézoutova rovnost** pro polynomy: Buďte $f(x)$ a $g(x)$ nenulové polynomy nad tělesem T . Pak existují polynomy $u(x), v(x) \in T[x]$ tak, že:
 $\gcd(f(x), g(x)) = u(x)f(x) + v(x)g(x)$
- Buď T těleso a $p(x) \in T[x]$ polynom stupně n . Prvek $\xi \in T$ je **kořen polynomu** p právě tehdy, když:
 $p(\xi) = (x - \xi)g(x)$

kde $g(x) \in T[x]$ je stupně $n - 1$

- **Ireducibilní polynom** – buď $P(x) \in K[x]$ stupně alespoň 1. Řekneme, že $P(x)$ je ireducibilní nad okruhem K , jestliže $\forall A(x), B(x) \in K[x]$:

$$A(x)B(x) = P(x) \implies (\deg(A(x)) = 0 \vee \deg(B(x)) = 0)$$

- Mějme celé $n > 1$ a prvočíslo p . Označme $N(p, n)$ počet monických polynomů stupně n ireducibilních nad \mathbb{Z}_p . Potom

$$N(p, n) = \frac{1}{n} \sum_{d|n} \mu(d) p^{n/d} \geq \frac{1}{n} \left(p^n - \sum_{q, q \text{ prvoč.}} p^{n/q} \right)$$

- **Monický polynom** – má za koeficient u nejvyšší mocniny jedničku
- μ – **Möbiova funkce** definovaná pro celé $n > 0$:

$$\mu(n) = \begin{cases} 1 & n \text{ neobsahuje čtverec prvočísla a má sudý počet prvočíselných faktorů} \\ -1 & n \text{ neobsahuje čtverec prvočísla a má lichý počet prvočíselných faktorů} \\ 0 & \text{obsahuje čtverec prvočísla} \end{cases}$$

3. Funkce více proměnných: gradient, Hessián, definitnost matic, extrémy funkcí více proměnných bez omezení a s rovnostními omezeními.

NI-MPI

Parciální derivace

- **Norma** na vektorovém prostoru V je zobrazení $\|\cdot\|: V \rightarrow \mathbb{R}_0^+$ splňující:

1. $\|x\| = 0 \Rightarrow x = 0$
2. $\|\alpha x\| = |\alpha| \cdot \|x\|$
3. $\|x + y\| \leq \|x\| + \|y\|$ (trojúhelníková nerovnost)
 - o Pro každé $x, y \in V$ a všechny skaláry α
 - o Euklidovská norma na \mathbb{R}^n (\mathbb{C}^n): $\|x\|_2 = (\sum_{i=1}^n |x_i|^2)^{1/2}$

- Reálná **funkce více proměnných** – zobrazení $D_f \rightarrow \mathbb{R}, D_f \subset \mathbb{R}^n$

- o D_f ... definiční obor, $f(D_f)$... obor hodnot
- o **Graf funkce** = množina:

$$\Gamma_f = \{(b_1, b_2, \dots, b_n, f(b_1, b_2, \dots, b_n)) : (b_1, b_2, \dots, b_n) \in D_f\} \subset \mathbb{R}^{n+1}$$

- o **Okolí bodu** = buď $x \in \mathbb{R}^n$ a $\delta \in \mathbb{R}^+$, δ -okolí bodu x je množina $H_\delta(x) = \{b \in \mathbb{R}^n : \|x - b\| < \delta\}$
- o **Hromadný bod** = $x \in \mathbb{R}^n$ je hromadným bodem M ($M \subset \mathbb{R}^n$), pokud $\forall r > 0: H_r(x) \cap M \neq \emptyset$
 - Bod $x \in M$, který není hromadný, je izolovaný

- **Limita funkce více proměnných** – funkce $f: D_f \rightarrow \mathbb{R}, D_f \subset \mathbb{R}^n$, má limitu $L \in \mathbb{R}$ v hromadném bodě b množiny D_f , pokud:

$$\forall H(L) \quad \exists H(b) \quad x \in (D_f \cap H(b)) \setminus \{b\} \Rightarrow f(x) \in H(L)$$

- o Značení:

$$\lim_{x \rightarrow b} f(x) = L$$

- o **Limita posloupnosti** – posl. $(x_i)_{i=0}^{+\infty}$ má limitu $L \in \mathbb{R}^n$, pokud $\forall \varepsilon > 0 \exists N \forall n > N \quad x_n \in H_\varepsilon(L)$
- o Mějme funkci $f: D_f \rightarrow \mathbb{R}, D_f \subset \mathbb{R}^n$. Funkce F má v bodě b limitu $L_\varepsilon \Leftrightarrow \forall (x_i)_{i=0}^{+\infty} \quad x_i \neq b$:

$$\lim_{n \rightarrow +\infty} x_n = b \quad \Rightarrow \quad \lim_{n \rightarrow +\infty} f(x_n) = L$$

- o **Spojitosť funkce** – funkce $f: D_f \rightarrow \mathbb{R}, D_f \subset \mathbb{R}^n$ je spojitá v bodě $x_0 \in D_f$, pokud:

$$\forall \varepsilon > 0 \quad \exists \delta > 0: \quad x \in (D_f \cap H_\delta(x_0)) \Rightarrow f(x) \in H_\varepsilon(f(x_0))$$

- Funkce je spojitá, pokud je spojitá ve všech bodech definičního oboru
- Formulace pomocí limity – f je spojitá, pokud pro všechny neizolované body $x_0 \in D_f$:

$$\lim_{x \rightarrow x_0} f(x) = f(x_0)$$

- **Parciální derivace** funkce f ve směru (podle) x_i v bodě $b \in D_f, \exists H(b) \subset D_f$:

$$\lim_{h \rightarrow 0} \frac{f(b_1, b_2, \dots, b_i + h, \dots, b_n) - f(b_1, b_2, \dots, b_i, \dots, b_n)}{h} = L$$

pokud tato limita existuje

- o Je to směrnice tečny ke grafu funkce f ve směru osy x_i

- **Gradient funkce f** v bodě $b \in D_f$ je vektor

$$\nabla f(b) = \left(\frac{\partial f}{\partial x_1}(b), \frac{\partial f}{\partial x_2}(b), \dots, \frac{\partial f}{\partial x_n}(b) \right)$$

- **Derivace ve směru** – $v \in \mathbb{R}^{n,1} = \mathbb{R}^n, \|v\| = 1$, pak derivace f ve směru v v bodě $b \in D_f, \exists H(b) \subset D_f$ je:

$$\nabla_v f(b) = \lim_{h \rightarrow 0} \frac{f(b + hv) - f(b)}{h}.$$

- o Pokud existuje gradient f v bodě b (všechny parc. derivace f jsou na nějakém okolí b spojitě), pak:

$$\nabla_v f(b) = \nabla f(b) \cdot v$$

- **Parciální derivace 2. řádu** v bodě b :

$$\frac{\partial^2 f}{\partial x_j \partial x_i}(b) = \frac{\partial}{\partial x_j} \left(\frac{\partial f}{\partial x_i} \right)(b)$$

- o $i \neq j \rightarrow$ smíšená 2. parciální derivace

- $i = j \rightarrow \frac{\partial^2 f}{\partial x_i^2}(\mathbf{b})$
- Opět zobrazení z podmnožiny D_f
- **Hessova matice** – existují-li všechny 2. parciální derivace v bodě \mathbf{b} , zaznamenávají se do Hessovy matice (=Hessián):

$$\nabla^2 f(\mathbf{b}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(\mathbf{b}) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(\mathbf{b}) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(\mathbf{b}) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(\mathbf{b}) \end{pmatrix}$$

- Zaměnitelnost 2. parciálních derivací
 - Pokud jedna existuje a ta funkce je v \mathbf{b} spojitá, potom druhá existuje, platí:
- $$\frac{\partial^2 f}{\partial x \partial y}(\mathbf{b}) = \frac{\partial^2 f}{\partial y \partial x}(\mathbf{b})$$
- **2. derivace ve směru** – lze derivovat ve směru \mathbf{v} v první derivaci ve směru \mathbf{v} – druhá parciální derivace f ve směru \mathbf{v} v bodě \mathbf{b} : $\nabla_{\mathbf{v}}(\nabla_{\mathbf{v}} f)(\mathbf{b})$
 - Existuje-li Hessova matice (existuje okolí takové, že má f na něm spojitě všechny druhé parciální derivace), potom:

$$\nabla_{\mathbf{v}}(\nabla_{\mathbf{v}} f)(\mathbf{b}) = \mathbf{v}^T \cdot \nabla^2 f(\mathbf{b}) \cdot \mathbf{v}$$

- **Jacobiho matice** – matice prvních derivací
 - Máme $\Psi: \mathbb{R}^n \rightarrow \mathbb{R}^n$, $\Psi(\mathbf{v}) = (\Psi_1(\mathbf{v}), \dots, \Psi_n(\mathbf{v}))$
 - Jacobiho matice funkce ψ (psi) je zobrazení $\mathbb{R}^n \rightarrow \mathbb{R}^{n,n}$:

$$J_{\Psi} = \begin{pmatrix} \frac{\partial \Psi_1}{\partial v_1} & \cdots & \frac{\partial \Psi_1}{\partial v_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \Psi_n}{\partial v_1} & \cdots & \frac{\partial \Psi_n}{\partial v_n} \end{pmatrix}$$

pokud všechny derivace existují

- Má na řádcích **složky gradientů** jednotlivých složek ψ
- **Geometrický význam gradientu**
 - Gradient ukazuje směr (v definičním oboru) nejvyššího růstu funkce f
 - V místech, kde je gradient nulový (stacionární body) hledáme extrémy
 - **Tečná nadrovina** – sjednocení tečen ve všech směrech (v bodě \mathbf{b}) – tečná nadrovina f v \mathbf{b}
 - Musí existovat gradient f v \mathbf{b}
 - Rovnice nadroviny:

$$z = \frac{\partial f}{\partial x_1}(\mathbf{b})(x_1 - b_1) + \frac{\partial f}{\partial x_2}(\mathbf{b})(x_2 - b_2) + \cdots + \frac{\partial f}{\partial x_n}(\mathbf{b})(x_n - b_n) + f(\mathbf{b})$$

- Normálový vektor:

$$\left(\frac{\partial f}{\partial x_1}(\mathbf{b}), \frac{\partial f}{\partial x_2}(\mathbf{b}), \dots, \frac{\partial f}{\partial x_n}(\mathbf{b}), -1 \right)$$

Lokální extrémy – bez omezení

- Reálná funkce f má v bodě $\mathbf{b} \in D_f$:
 - **Lokální minimum**, pokud:

$$\exists \delta > 0, \forall \mathbf{x} \in (D_f \cap H_{\delta}(\mathbf{b})), f(\mathbf{x}) \geq f(\mathbf{b})$$
 - **Ostré lokální minimum**, pokud:

$$\exists \delta > 0, \forall \mathbf{x} \in (D_f \cap H_{\delta}(\mathbf{b})) \setminus \{\mathbf{b}\}, f(\mathbf{x}) > f(\mathbf{b})$$
 - **Globální minimum**, pokud:

$$\forall \mathbf{x} \in D_f, f(\mathbf{x}) \geq f(\mathbf{b})$$
- Je-li D_f omezená a uzavřená, pak má spojitá funkce $f: D_f \rightarrow \mathbb{R}$ globální minimum a globální maximum
- **Nutná podmínka existence lokálního extrému** – nechť má $f: D_f \rightarrow \mathbb{R}$, $D_f \subset \mathbb{R}^n$ v bodě \mathbf{b} parciální derivaci podle i -té proměnné. Pokud má f v bodě \mathbf{b} lokální extrém, pak:

$$\frac{\partial f}{\partial x_i}(\mathbf{b}) = 0$$

- Pokud existuje gradient f v \mathbf{b} , pak existence lokálního extrému implikuje, že $\nabla f(\mathbf{b}) = \mathbf{0}$

- **Stacionární body** = body $b \in D_f$ splňující $\nabla f(b) = 0$
- V úloze hledání extrémů jsou stacionární **body podezřelé z extrému** = **kritické body**
 - Počítají se mezi ně i body, kde ∇f neexistuje
- **Definitnost matic** – $A \in \mathbb{R}^{n,n}$. Řekneme, že A je:
 - **Pozitivně semidefinitní**, pokud $x^T A x \geq 0 \forall x \in \mathbb{R}^{n,1}$
 - **Negativně semidefinitní**, pokud $x^T A x \leq 0 \forall x \in \mathbb{R}^{n,1}$
 - **Pozitivně definitní**, pokud $x^T A x > 0 \forall x \in \mathbb{R}^{n,1}, x \neq 0$
 - **Negativně definitní**, pokud $x^T A x < 0 \forall x \in \mathbb{R}^{n,1}, x \neq 0$
 - **Indefinitní**, pokud není pozitivně ani negativně semidefinitní
 - Matice A je indefinitní $\Leftrightarrow \exists x, y \in \mathbb{R}^n, x^T A x > 0$ a $y^T A y < 0$
- Pokud je $A \in \mathbb{R}^{n,n}$ **symetrická** matice, potom:
 - A je **pozitivně semidefinitní** \Leftrightarrow **nezáporná** všechna její vlastní čísla
 - A je **pozitivně definitní** \Leftrightarrow **kladná** všechna její vlastní čísla
 - A je **negativně semidefinitní** \Leftrightarrow **nekladná** všechna její vlastní čísla
 - A je **negativně definitní** \Leftrightarrow **záporná** všechna její vlastní čísla
 - A je **indefinitní** \Leftrightarrow alespoň jedno **kladné** a alespoň jedno **záporné** vlastní číslo
- **Sylvestrovo kritérium**: $A \in \mathbb{R}^{n,n}$ je symetrická matice. Pro $A_1 \dots A_n: A_k \in \mathbb{R}^{k,k}$ je čtvercová matice v levém horním rohu A
 - A je **pozitivně definitní** \Leftrightarrow **determinant** všech matic $A_1 \dots A_n$ je **kladný**
 - A je **negativně definitní** \Leftrightarrow **determinant** A_k je **záporný** pro k **liché** a **kladný** pro k **sudé**
- Jak poznat **indefinitnost** – pokud má $A \in \mathbb{R}^{n,n}$ na diagonále 2 prvky s **různým znaménkem**, pak je indefinitní
- Postačující podmínka existence extrému a sedlového bodu – nechť $b \in D_f$ je stacionární bod funkce $f: D_f \rightarrow \mathbb{R}, D_f \subset \mathbb{R}^n$. Nechť existuje okolí bodu b takové, že f má na okolí spojitě všechny 2. parciální derivace, potom:
 - Je-li $\nabla^2 f(b)$ **pozitivně definitní**, pak b je **ostré lokální minimum**
 - Je-li $\nabla^2 f(b)$ **negativně definitní**, pak b je **ostré lokální maximum**
 - Je-li $\nabla^2 f(b)$ **indefinitní**, pak b je **sedlový bod**
- **Nutná podmínka existence lokálního extrému** – nechť $b \in D_f$ je stacionární bod funkce $f: D_f \rightarrow \mathbb{R}, D_f \subset \mathbb{R}^n$. Nechť existuje okolí bodu b takové, že f má na okolí spojitě všechny 2. parciální derivace, potom:
 - Je-li b **lokální minimum**, pak $\nabla^2 f(b)$ je **pozitivně semidefinitní**
 - Je-li b **lokální maximum**, pak $\nabla^2 f(b)$ je **negativně semidefinitní**
 - Tvrzení nelze obrátit
- Postup **analytického hledání extrémů**:
 1. Najít **body podezřelé z extrému** = stacionární body a body, kde alespoň jedna parciální derivace neexistuje
 2. Nalézt **Hessovu matici** v bodě b podezřelém z extrému, pokud ta je
 - a. **Pozitivně definitní**, pak je bod b bodem **ostrého lokálního minima**
 - b. **Negativně definitní**, pak je bod b bodem **ostrého lokálního maxima**
 - c. **Indefinitní**, pak je bod b **sedlovým bodem** (takže není extrém)
 - d. V ostatních případech je třeba rozhodovat jiným způsobem

Lokální extrémy – rovnostní omezení

- **Úloha vázaného extrému** (minima) s rovnostní podmínkou = minimalizuj $f(x)$ za podmínky $g_j = 0, j \in \hat{m}$
 - $\hat{m} = \{1 \dots m, m \in \mathbb{N}\}$
 - f ... objektivní / účelová / minimalizovaná / optimalizovaná funkce
 - g_j ... rovnostní podmínka / vazba
 - Jsou-li všechny funkce lineární, je to úloha lineárního programování
- **Množina přípustných řešení**:

$$\mathcal{M} = \{x \in D: (\forall j \in \hat{m})(g_j(x) = 0) \wedge (\forall k \in \hat{p})(h_k(x) \leq 0)\}$$
 - Máme úlohu hledání $m^* = \min\{f(x): x \in \mathcal{M}\}$ a bodů $x^* \in \mathcal{M}$, pro které $m^* = f(x^*)$
 - Hledáme body lokálního minima vzhledem k množině \mathcal{M} : (pro nějaké okolí $H(x^*)$)

$$\forall x \in (H(x^*) \cap \mathcal{M}) \quad f(x^*) \leq f(x)$$

- Lagrangeova funkce $L: M \times \mathbb{R}^m \rightarrow \mathbb{R}$ pro danou úlohu:

$$L(\mathbf{x}; \lambda) = f(\mathbf{x}) + \sum_{j=1}^m \lambda_j g_j(\mathbf{x})$$

- o Lagrangeovy multiplikátory ... $\lambda = (\lambda_1, \dots, \lambda_m)$

- Postačující podmínka existence ostrého lokálního minima pro rovnostní vazby – necht $f, g, j \in \hat{m}$ mají spojitě všechny 2. parciální derivace na nějaké otevřené nadmnožině $\tilde{M} \supset M$.

Pokud dvojice $(x^*, \lambda^*) \in \mathbb{R}^n \times \mathbb{R}^m$ splňuje podmínky:

- i. (0. derivace) $x^* \in M$

- ii. (1. derivace) $\forall i, \frac{\partial L}{\partial x_i}(x^*; \lambda^*) = 0$

- iii. (2. derivace) pro každý (sloupcový) vektor $0 \neq v \in \mathbb{R}^n$ splňují

$$\nabla g_j(x^*) \cdot v = 0, \quad \text{pro } \forall j \in \{1, \dots, m\} \text{ platí:}$$

$$v^T \cdot \nabla_x^2 L(x^*; \lambda^*) \cdot v > 0$$

kde ∇_x^2 je Hessova matice funkce L vzhledem k proměnným $x = (\dots)$

potom je x^* bodem **ostrého lokálního minima**

- o Body i. a ii. jsou ekvivalentní rovnosti $\nabla L(x^*, \lambda^*) = 0$

4. Integrál funkcí více proměnných (Darbouxova konstrukce).

NI-MPI

Teorie 1D a 2D integrálu

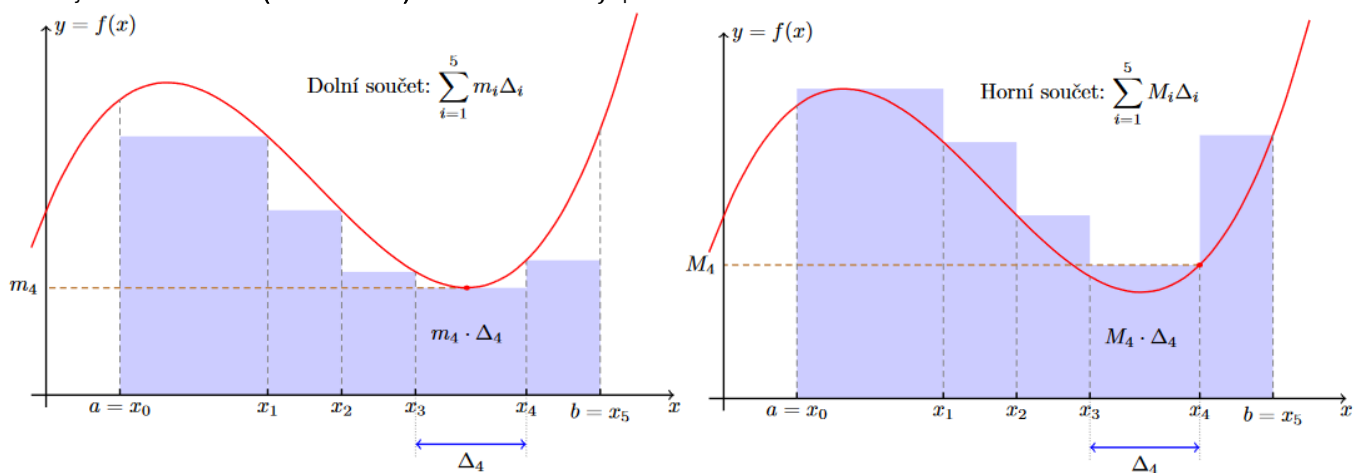
- **Integrál** = nástroj pro výpočet obsahu pod grafem nějaké funkce
- **Rozdělení intervalu** $[a, b]$ = konečná množina $\sigma = \{x_0 \dots x_n\}$ taková, že $a = x_0 < \dots < x_n = b$
 - o $x_k, k = 1, 2, \dots, n-1$... **dělicí body** intervalu $[a, b]$
 - o $v(\sigma) = \max\{\Delta_k, k = 1, \dots, n\}$, $\Delta_k = x_k - x_{k-1}$... **norma rozdělení** σ
- **Darbouxův součet** (horní/dolní) – necht f je definovaná na $[a, b]$ a $\sigma = \{x_0 \dots x_n\}$ je jeho rozdělení. Označme:

$$M_i = \sup_{x \in [x_{i-1}, x_i]} f(x) \quad \text{a} \quad m_i = \inf_{x \in [x_{i-1}, x_i]} f(x)$$

Potom

$$S_f(\sigma) = \sum_{i=1}^n M_i \Delta_i \quad \text{a} \quad s_f(\sigma) = \sum_{i=1}^n m_i \Delta_i$$

je horní a dolní (Darbouxův) součet funkce f při rozdělení σ



- o Horní Darbouxův integrál (f na $[a, b]$):

$$D_f = \inf \left\{ S_f(\sigma) : \sigma \text{ je rozdělení } [a, b] \right\}$$

- o Dolní darbouxův integrál (f na $[a, b]$):

$$d_f = \sup \left\{ s_f(\sigma) : \sigma \text{ je rozdělení } [a, b] \right\}$$

- o Pokud $D_f = d_f$, nazveme tuto hodnotu **Darbouxovým integrálem** f na $[a, b]$:

$$\int_a^b f(x) dx = D_f = d_f$$

- f je (Darbouxovsky) **integrabilní** na $[a, b]$

- o Posloupnost rozdělení σ_n je **normální**, pokud pro její normy platí:

$$\lim_{n \rightarrow \infty} v(\sigma_n) = 0$$

- o Buď f **spojitá** na $[a, b]$. Potom existuje $\int_a^b f(x) dx$. Je-li σ_n normální posloupnost rozdělení, potom

$$\lim_{n \rightarrow \infty} s_f(\sigma_n) \quad \text{a} \quad \lim_{n \rightarrow \infty} S_f(\sigma_n)$$

existují a jsou rovny $\int_a^b f(x) dx$

- o **Aditivita integrálu:**

$$\int_a^b (f+g)(x) dx = \int_a^b f(x) dx + \int_a^b g(x) dx$$

- o **Multiplikativita integrálu:**

$$\int_a^b (cf)(x) dx = c \int_a^b f(x) dx$$

- **Primitivní funkce** – nechť funkce f je definována v intervalu (a, b) , kde $-\infty \leq a < b \leq +\infty$. Funkci F splňující podmínku

$$F'(x) = f(x) \quad \forall x \in (a, b)$$

nazýváme primitivní funkcí k funkci f v intervalu (a, b)

- **2D integrál nad obdélníkovou oblastí**

- o Máme 2 rozdělení $\sigma_x [a, b], \sigma_y [c, d]$. $\sigma = \sigma_x \times \sigma_y$ je rozdělením $D = [a, b] \times [c, d]$
- o Označme:

$$M_{i,j} = \sup \left\{ f(x, y) : (x, y) \in [x_{i-1}, x_i] \times [y_{j-1}, y_j] \right\} \text{ a } m_{i,j} = \inf \left\{ f(x, y) : (x, y) \in [x_{i-1}, x_i] \times [y_{j-1}, y_j] \right\}$$

- o **Horní Darbouxova suma** f vzhledem k σ :

$$S_f(\sigma) = \sum_{i=1}^n \sum_{j=1}^m M_{i,j} (x_i - x_{i-1}) (y_j - y_{j-1})$$

- o **Dolní Darbouxova suma** f vzhledem k σ :

$$s_f(\sigma) = \sum_{i=1}^n \sum_{j=1}^m m_{i,j} (x_i - x_{i-1}) (y_j - y_{j-1})$$

- o **Horní Darbouxův integrál** (funkce f na D):

$$D_f = \inf \left\{ S_f(\sigma) : \sigma \text{ je rozdělení } D \right\}$$

- o **Dolní Darbouxův integrál** (funkce f na D):

$$d_f = \sup \left\{ s_f(\sigma) : \sigma \text{ je rozdělení } D \right\}$$

- o **(dvojitý) Darbouxův integrál**:

$$\iint_D f(x, y) dx dy = D_f = d_f$$

- f je Darbouxovsky integrovatelná na D

- o Posloupnost rozdělení je normální, pokud jsou obě původní rozdělení normální

- Bud' $f(x, y)$ integrovatelná funkce na $D = [a, b] \times [c, d]$. Pokud existuje jeden z integrálů

$$\int_a^b \left(\int_c^d f(x, y) dy \right) dx \quad \text{nebo} \quad \int_c^d \left(\int_a^b f(x, y) dx \right) dy$$

potom je roven dvojnému integrálu $\iint_D f(x, y) dx dy$

- o **Výpočet dvojného integrálu** – funkci nejdřív zintegrujeme vzhledem k jedné proměnné a druhou považujeme za konstantu, výsledek (získaný pomocí Newtonovy formule) potom závisí už jen na jedné proměnné, vzhledem ke které se provede 2. integrace

- **Vlastnosti dvojného integrálu**

- o Množina **míry nula** – je pro hodnotu integrálu zanedbatelná
- o Pokud má průnik D_1 a D_2 míru nula, pak $\iint_D f = \iint_{D_1} f + \iint_{D_2} f$
- o Pokud $f_1(x, y) \leq f_2(x, y)$, pak $\iint_D f_1 \leq \iint_D f_2$
- o Pro reálné c a integrovatelní f :

$$\iint_D c \cdot f(x, y) dx dy = c \cdot \iint_D f(x, y) dx dy$$

Metody pro výpočet 1D a 2D integrálu

- **Newtonova formule**:

$$\int_a^b f(x) dx = \lim_{x \rightarrow b-} F(x) - \lim_{x \rightarrow a+} F(x)$$

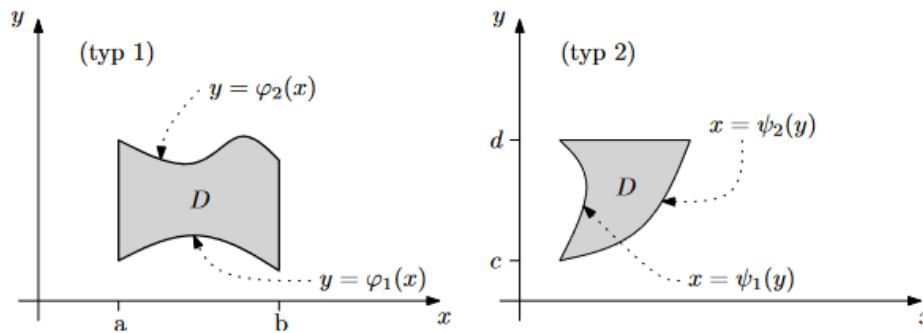
- **Per partes** pro určitý integrál:

$$\int_a^b f(x) g(x) dx = [f(x) G(x)]_a^b - \int_a^b f'(x) G(x) dx$$

- **Substituce** – funkce f, φ , φ a její derivace spojitě na $[\alpha, \beta]$, f spojitá na $\varphi([\alpha, \beta])$. Potom

$$\int_{\alpha}^{\beta} f(\varphi(t)) \cdot \varphi'(t) dt = \int_{\varphi(\alpha)}^{\varphi(\beta)} f(x) dx$$

- Výpočet dvojného integrálu nad obecnou oblastí



- 2 typy oblastí:
 - **Typ 1** – x z intervalu $[a, b]$, y omezené spojitými fncemi $\varphi_1(x)$ a $\varphi_2(x)$ splň. $\varphi_1(x) \leq \varphi_2(x)$
 - **Typ 2** – y z intervalu $[c, d]$, x omezené spojitými fncemi $\psi_1(y)$ a $\psi_2(y)$ splň. $\psi_1(y) \leq \psi_2(y)$
- Pokud dané integrály existují, platí pro oblast D :

- Je-li D typu 1, pak:

$$\iint_D f(x, y) dx dy = \int_a^b \left(\int_{\varphi_1(x)}^{\varphi_2(x)} f(x, y) dy \right) dx.$$

- Je-li D typu 2, pak:

$$\iint_D f(x, y) dx dy = \int_c^d \left(\int_{\psi_1(y)}^{\psi_2(y)} f(x, y) dx \right) dy.$$

- Substituce ve dvojném integrálu

- **Jacobiho matice** – matice prvních derivací
 - Máme $\Psi: \mathbb{R}^n \rightarrow \mathbb{R}^n$, $\Psi(\mathbf{v}) = (\Psi_1(\mathbf{v}), \dots, \Psi_n(\mathbf{v}))$
 - Jacobiho matice funkce ψ (psí) je zobrazení $\mathbb{R}^n \rightarrow \mathbb{R}^{n,n}$:

$$J_\Psi = \begin{pmatrix} \frac{\partial \Psi_1}{\partial v_1} & \dots & \frac{\partial \Psi_1}{\partial v_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \Psi_n}{\partial v_1} & \dots & \frac{\partial \Psi_n}{\partial v_n} \end{pmatrix}$$

pokud všechny derivace existují

- Má na řádcích **složky gradientů** jednotlivých složek ψ : $J_\Psi = \begin{pmatrix} \nabla \Psi_1 \\ \vdots \\ \nabla \Psi_n \end{pmatrix}$
 - Jsou to jednotlivé funkce

- **Věta o substituci** – nechť D je omezená uzavřená množina na \mathbb{R}^n . Nechť $\psi: \mathbb{R}^n \rightarrow \mathbb{R}^n$ má spojitě všechny parciální derivace (všech složek) na nějaké otevřené nadmnožině množiny D a skoro všude na D platí, že:

- ψ je **bijekce**
- $\det J_\psi$ je **nenulový**

Potom pro každou spojitou funkci $f: D \rightarrow \mathbb{R}$ platí:

$$\int_{\psi(D)} f(\mathbf{x}) d\mathbf{x} = \int_D f(\Psi(\mathbf{v})) |\det J_\Psi(\mathbf{v})| d\mathbf{v}$$

kde $\mathbf{x} = (x_1 \dots x_n)$

- Transformace do jiného souřadnicového systému – například z kruhové výseče dělám obdélník
 - Pokud nemám ani oblast typu 1, ani oblast typu 2
- Determinant je tam, protože transformace může změnit objem – nutno kompenzovat

5. Numerická matematika: reprezentace čísel v počítači, chyby vznikající při výpočtech s pohyblivou řádovou čárkou, podmíněnost a stabilita numerických algoritmů.

NI-MPI

Strojová čísla

- **Standard IEEE-754** – strojové číslo lze reprezentovat znaménkem s a celými kladnými čísly e a m
- Pro celá čísla x – vědecký zápis čísel v binární bázi: $x \pm m \cdot 2^e$
 - o m ... **mantisa** – pevný počet cifer – platné cifry
 - o e ... **exponent** – pevný počet cifer
- Omezení reprezentovatelných čísel podle přesnosti:

přesnost	délka m	$d = \text{délka } e$	parametr b
poloviční (binary16, <i>half precision</i>)	10	5	15
jednoduchá (binary32, <i>single precision</i>)	23	8	127
dvojitá (binary64, <i>double precision</i>)	52	11	1023
čtyřnásobná (binary128, <i>quadruple prec.</i>)	112	15	16383

- Určení reprezentované hodnoty x :
 - o Pokud $e = 2^d - 1$ a $m \neq 0$, pak $x = NaN$
 - o Pokud $e = 2^d - 1$ a $m = 0$, pak $x = (-1)^s \cdot Inf$
 - o Pokud $0 < e < 2^d - 1$ a $m \neq 0$, pak $x = (-1)^s \cdot (1.m_2)_2 \cdot 2^{e-b}$... **normalizovaná čísla**
 - o Pokud $e = 0$ a $m \neq 0$, pak $(-1)^s \cdot (0.m_2)_2 \cdot 2^{1-b}$... **subnormální čísla**
 - o Pokud $e = 0$ a $m = 0$, pak $(-1)^s \cdot 0$
- **Skrytá jednička** – pro normalizované číslo x se uloží o 1 platnou cifru více, než kolik je délka m
 - o Nemusí se reprezentovat, protože normalizovaná čísla vždycky začínají na 1
- Čísla, která lze takto reprezentovat = **strojová čísla**
- **Množina strojových čísel** $F \equiv F(|m|, |e|, b)$ – počet strojových čísel je závislý na m a e
 - o $b = \text{bias} = \text{offset}$
 - o F charakterizováno pomocí **strojové přesnosti** ε_F (machine epsilon) = vzdálenost čísla $1 = +1 \cdot 2^0$ od nejbližšího většího čísla v F :

$$\varepsilon_F = (1.0 \dots 01)_2 \cdot 2^0 - (1.0 \dots 00)_2 \cdot 2^0.$$
 - Pro jednoduchou přesnost platí $\varepsilon_F = 2^{-23}$
 - o **Vzdálenost** libovolného normalizovaného čísla $x \in F$ od jeho nejbližších sousedů z F je nejméně $\varepsilon_F \frac{|x|}{2}$ a nejvíce $\varepsilon_F |x|$
- Pokud o reprezentaci čísel mimo rozsah \rightarrow **přetečení / podtečení**

Chyby při výpočtech s pohyblivou řádovou čárkou

- **Typy chyb**
 - o Chyba **modelu** – mat. model úlohy je nějak zjednodušený, nebo se používají průměrné místo aktuálních hodnot
 - o Chyba **dat** – data z měření bez absolutní přesnosti
 - o Chyba **algoritmu** – algoritmus nemusí najít v konečném počtu kroků přesné řešení
 - o **Zaokrouhlovací chyba** – při samotném výpočtu dochází např. k chybám v aritmetických operacích
- **Zaokrouhlovací chyby** - $\alpha \in F$ je přibližnou hodnotou čísla $a \in \mathbb{R}$
 - o **Absolutní chyba** reprezentace a pomocí $\alpha = |\alpha - a|$
 - o **Relativní chyba** pro $a \neq 0$ reprezentace a pomocí $\alpha = \frac{|\alpha - a|}{|a|}$
- **Zaokrouhlovací jednotka** – meze pro relativní chybu - $u = 2^{-23}$ (pro jednoduchou přesnost)
 - o Zaokrouhlování směrem k 0 – usekneme bity přeskakující délku mantisy
- **Aritmetické operace** a chyba při jejich provádění
 - o $x, y \in F$; \odot značí operaci sčítání, násobení, odčítání nebo dělení
 - o Pokud nedojde k přetečení nebo podtečení, platí:

$$fl(x \odot y) = (x \odot y)(1 + \delta), \quad \text{kde } |\delta| \leq u$$
 - o Operací \odot nemusíme dostat strojové číslo – obecně reálné číslo, které je třeba zaokrouhlit

- **Ztráta platných cifer**
 - Velké problémy – **krácení** – lze se mu vyhnout:
 - Přeformulováním problému tak, aby nedocházelo k odčítání
 - Použitím rozvoju funkcí do řad
 - Použitím jiných rovností
 - **Odečítání** – nechť x a y jsou normalizovaná strojová čísla a $x > y > 0$. Pokud $2^{-p} \leq 1 - \frac{y}{x} \leq 2^{-q}$ pro nějaká kladná celá p a q , tak platí, že nejvíce p a nejméně q platných binárních bitů je ztraceno při provedení odčítání $x - y$
- **Původy zaokrouhlovacích chyb** – zaokrouhlovací chyby jednotlivých operací a jejich **kumulace, krácení**

Numerické algoritmy

- **Přímá metoda** – počítá řešení v konečném počtu kroků tak, že v teoretické absolutní přesnosti dává (přesné) řešení
 - Příklady:
 - GEM (Gaussova eliminační metoda)
 - Hledání inverze matice pomocí GEM
 - Hledání vlastních čísel
 - Není „samoopravující“ – když v jednom kroku vznikne chyba, už se jí nemusíme zbavit
- **Iterační metody** – hledají přibližná řešení matematických problémů tak, že konstruují posloupnost přibližných řešení x_0, x_1, x_2, \dots
 - Každé další přibližné řešení je odvozeno z předchozího: $x_k = T(x_{k-1})$ pro $k > 0$ a zobrazení T
 - Zobrazení T je voleno tak, aby posl. $(x_k)_{k=0}^{\infty}$ měla limitu, která je skutečným řešením dané úlohy
 - **Stacionární metoda** = pokud je T neměnné pro všechny iterace k
 - Příklady:
 - Richardsova metoda
 - Jacobiho metoda
 - Gauss-Seidelova metoda / SOR

Podmíněnost numerických algoritmů

- V ... numerický algoritmus
- $V^*(d)$... teoretický (přesný) výstup V
- d ... vstupní data
- $V(d)$... výsledek v konečné (strojové) aritmetice
- **Dopředná / přímá chyba** - $\Delta v = V^*(d) - V(d)$... **odchylka** spočítaného řešení od přesného řešení
 - Chyba výsledku
- **Zpětná chyba** - $V^*(d + \Delta d) = V(d)$, Δd je nejmenší číslo v normě, které to splňuje
 - Co musím zadat na vstup, abych dostala to, co jsem dostala
 - Promítnutí chyby algoritmu V do jeho vstupu
- Pokud je pro všechny vstupy d zpětná chyba relativně malá (podle kontextu), algoritmus je **zpětně stabilní**
- **Podmíněnost úlohy / algoritmu** – vyjadřuje závislost změny výstupu na změně vstupních dat
 - **Relativní číslo podmíněnosti:**

$$C_r = \lim_{\epsilon \rightarrow 0^+} \sup_{\substack{d + \delta d \in D \\ \|\delta d\| \leq \epsilon}} \frac{\frac{\|V^*(d + \delta d) - V^*(d)\|}{\|V^*(d)\|}}{\frac{\|\delta d\|}{\|d\|}} \quad wtf$$

- D ... zkoumaný definiční obor V/V^*
- $C_r \approx 1 \rightarrow$ úloha je dobře podmíněná – změna na vstupu málo změní výsledek
- C_r velké \rightarrow úloha je špatně podmíněná – při malé změně se výsledek změní hodně
- **Soustavy lineárních rovnic** - $n \in \mathbb{N}$ lineárních rovnic pro n neznámých
 - Zápis $Ax = b$

- $A \in \mathbb{R}^{n,n}$... regulární matice soustavy
- $b \in \mathbb{R}^{n,1}$... vektor pravých stran
- $x \in \mathbb{R}^{n,1}$... hledané řešení
- **Norma na vektorovém prostoru** V = zobrazení $\|\cdot\|: V \rightarrow \mathbb{R}_0^+$ splňující (pro každé $x, y \in V, \alpha$ skalár atd.):
 - i. $\|x\| = 0 \Rightarrow x = 0$
 - ii. $\|\alpha x\| = |\alpha| \cdot \|x\|$
 - iii. $\|x + y\| \leq \|x\| + \|y\|$
- **Přidružená maticová norma** matice $A \in \mathbb{R}^{n,n}$ k vektorové normě $\|\cdot\|$ na $\mathbb{R}^n \equiv \mathbb{R}^{n,1}$:

$$\|A\| := \sup \{ \|Ax\| : x \in \mathbb{R}^{n,1} \text{ a } \|x\| = 1 \}$$
 - Definice suprema neprázdné omezené množiny M :

$$\sup M = \min \{ y \in \mathbb{R} : \forall x \in M : x \leq y \}$$
 - $\|A\|$ je norma a platí pro ni (pro všechny $A, B \in \mathbb{R}^{n,n}$):
 - $\|E\| = 1$ (E ... jednotková matice)
 - $\|Ax\| \leq \|A\| \cdot \|x\|$... konzistence normy
 - $\|AB\| \leq \|A\| \cdot \|B\|$... submultiplikativita
- **Podmíněnost úlohy:**

$$\frac{\|\delta x\|}{\|x\|} \leq \|A\| \cdot \|A^{-1}\| \cdot \frac{\|\delta b\|}{\|b\|}$$
 - „relativní chyba v řešení soustavy $Ax = b$ je menší, než relativní chyba pravé strany b vynás. $\kappa(A)$ “
 - $\kappa(A) = \|A\| \cdot \|A^{-1}\|$... číslo podmíněnosti matice A
 - Čím větší, tím horší podmíněnost a hrozí větší chyba
 - Takhle to dopadne, když je b lehce změněna o perturbaci δb
 - Změna v řešení $x = A^{-1}b$ označena δx , potom platí:

$$Ax = b \quad \text{a} \quad A(x + \delta x) = Ax + A\delta x = b + \delta b$$
 - Z toho se dá odvodit ta nerovnost výše přes pravidla pro normu, ale vypisovat to nechcu

Iterační numerické metody

- Cílem je algoritmus, který konstruuje posl. vektorů x_0, x_1, \dots , která se „blíží“ přesnému řešení $Ax = b$
- Postup:
 1. Startovací vektor x_0 zvolíme náhodně
 2. Zvolíme **regulární matici** Q (podle metody)
 3. x_0, x_1, \dots počítáme podle:

$$Qx_k = (Q - A)x_{k-1} + b, \quad \forall k > 0$$

$$x_k = Q^{-1}((Q - A)x_{k-1} + b)$$
- Kdyby byla posloupnost $(x_k)_{k=0}^\infty$ konvergentní s limitou x^* , potom je x^* hledané řešení
- **Konvergence – volba Q**
 - Rovnost $x_k = Q^{-1}((Q - A)x_{k-1} + b)$ dosadíme:

$$\begin{aligned} x_k - x &= Q^{-1}((Q - A)x_{k-1} + b) - x \\ &= (E - Q^{-1}A)x_{k-1} - x + Q^{-1}b \\ &= (E - Q^{-1}A)x_{k-1} - (E - Q^{-1}A)x \\ &= (E - Q^{-1}A)(x_{k-1} - x), \end{aligned}$$
 - x je vektor splňující $Ax = b$ a E je jednotková matice
 - $W = E - Q^{-1}A$
 - **Vektor chyby:** $e_k = x_k - x$
 - Potom platí:

$$e_k = We_{k-1} = W^2e_{k-2} = \dots = W^ke_0$$
 - Potřebujeme W , pro které $\lim_{k \rightarrow \infty} W^k = 0$
 - Chceme, aby se e_k blížilo k 0
 - **Spektrální poloměr** matice M : $\rho(M) \geq 0$ = absolutní hodnota největšího vlastního čísla (abs):

$$\rho(M) := \max\{|\lambda| : \lambda \text{ je vlastním číslem } M\}.$$
 - Nechť $M \in \mathbb{C}^{n,n}$. Potom platí

$$\lim_{k \rightarrow +\infty} M^k = 0 \Leftrightarrow \rho(M) < 1$$

- Máme zajištěnou konvergenci iterační metody, právě tehdy, když $\rho(W) < 1$
- Všechna vlastní čísla $W = E - Q^{-1}A$ jsou v absolutní hodnotě < 1

- **Ukončení iterace** – v kroku k , dosáhne-li x_k požadované přesnosti $\|W\| < 1 \rightarrow$ posloupnost $(\|e_k\|)$ je ostře klesající a iteraci lze zastavit, když nastane:

$$\|e_k\| = \|x_k - x\| < \epsilon$$

- ϵ ... uživatelem zadaný parametr
- Nepraktické – nemáme x
- V kroce k napočítáme reziduum $Ax_k - b$, získáme **kritérium konvergence**:

$$\|Ax_k - b\| < \epsilon$$

- Někdy místo rezidua méně náročné kritérium $\|x_{k+1} - x_k\| < \epsilon$
- V praxi – maximální počet iterací, po překročení metoda selže
- Tady vzniká chyba algoritmu

- **Konkrétní metody – volby Q**

- $x_k = Q^{-1}((Q - A)x_{k-1} + b)$
- $a_{i,j}$ prvky matice A , definujeme matice L, D jako:

$$L := \begin{pmatrix} 0 & 0 & \cdots & 0 \\ a_{2,1} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n-1} & 0 \end{pmatrix} \quad \text{a} \quad D := \begin{pmatrix} a_{1,1} & 0 & \cdots & 0 \\ 0 & a_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{n,n} \end{pmatrix}$$

- $U = A - L - D$, takže platí

$$A = L + D + U$$

- L = lower (pod diagonálou), D jakože diagonála, U = upper (nad diagonálou)
- **Richardsonova metoda:** $Q = E$

- Iterace:

$$x_k = Q^{-1}((Q - A)x_{k-1} + b) = (E - A)x_{k-1} + b$$

- Konvergenci kontroluje matice $W = E - Q^{-1}A = E - A$
 - A musí být blízko E , aby byl třeba rozdíl pod 1

- **Jacobiho metoda:** $Q = D$

- Iterace:

$$x_k = Q^{-1}((Q - A)x_{k-1} + b) = D^{-1}(-L - U)x_{k-1} + D^{-1}b$$

- Konvergence kontrolována maticí $W = E - Q^{-1}A = D^{-1}A$
- Postačující podmínka – A je **diagonálně dominantní** \rightarrow Jacobiho metoda konverguje pro všechny volby x_0
- Matice je diagonálně dominantní, pokud pro každý řádek platí, že součet absolutních hodnot prvků vyjma diagonálního je menší než absolutní hodnota diagonálního prvku

- **SOR / Gauss-Seidel metoda:**

- **Gauss-Seidel:** $Q = D + L$

- **SOR:** $Q = \frac{1}{\omega}D + L$

- ω ... k urychlení konvergence, nenulové

- Iterace:

$$\left(\frac{1}{\omega}D + L\right)x_k = \left(\frac{1}{\omega}D + L - A\right)x_{k-1} + b = \left(\left(\frac{1}{\omega} - 1\right)D - U\right)x_{k-1} + b$$

- SOR konverguje, pokud $0 < \omega < 2$ a A je symetrická a pozitivně definitní s kladnými prvky na diagonále

6. Testování statistických hypotéz. T-testy, testy nezávislosti, testy dobré shody.

NI-VSM

- **Náhodný výběr** z rozdělení $F = n$ -tice stejně rozdělených nezávislých náhodných veličin (iid) $x_1 \dots x_n$ s distribuční funkcí F
- **Realizace náhodného výběru** = n -tice konkrétních pozorovaných čísel $x_1 \dots x_n$
- **Kroky statistického uvažování**
 - o **Odhad tvaru** rozdělení
 - o **Odhad parametrů** rozdělení
 - **Bodový odhad**
 - **Intervalový odhad**
 - o **Testování hypotéz** – ověření správnosti modelu
 - **Testy dobré shody** – ověřujeme hypotézy o tvaru pravděpodobnostního rozdělení
 - „má veličina normální rozdělení?“
 - **Parametrické testy** – tvoříme hypotézu o parametru θ a na základě dat se snažíme rozhodnout, zda je možné hypotézu zamítnout
 - „ $\theta = 0$ “

Intervaly spolehlivosti

- Zajímá nás interval, ve kterém leží skutečná hodnota parametru s danou pravděpodobností $1 - \alpha$
- Interval (L, U) určený statistikami $L \equiv L(x_1 \dots x_n)$ a $U \equiv U(x_1 \dots x_n)$, splňující:

$$P(\theta \in (L, U)) = P(L < \theta < U) = 1 - \alpha$$

se nazývá **oboustranný $100 \cdot (1 - \alpha)\%$ interval spolehlivosti** (konfidenční interval)

- Interval $(L, +\infty)$, resp. $(-\infty, U)$ určený statistikou L/U , splňující:

$$P(\theta \in (L, +\infty)) = P(\theta \in (-\infty, U)) = 1 - \alpha$$

se nazývá **horní/dolní (jednostranný) $100 \cdot (1 - \alpha)\%$ interval spolehlivosti**

- L/U = **dolní/horní mez** intervalu spolehlivosti
- $(1 - \alpha)$ = **hladina** spolehlivosti
- Pro oboustranný interval spolehlivosti volíme L, U tak, aby platilo

$$P(\theta < L) = \frac{\alpha}{2} \wedge P(U < \theta) = \frac{\alpha}{2}$$

- Interval spolehlivosti pro **střední hodnotu při známém rozptylu**:

$$\left(\bar{X}_n - z_{\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}}, \bar{X}_n + z_{\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}} \right)$$

- o $z_{\alpha} = \Phi^{-1}(1 - \alpha)$... **kritická hodnota** standardního normálního rozdělení $N(0, 1)$

- Interval spolehlivosti pro **střední hodnotu při neznámém rozptylu σ^2**

- o Neznáme $\sigma^2 \rightarrow$ odhadujeme ho pomocí **výběrového rozptylu s_n^2**
- o Oboustranný interval spolehlivosti:

$$\left(\bar{X}_n - t_{\frac{\alpha}{2}, n-1} \frac{s_n}{\sqrt{n}}, \bar{X}_n + t_{\frac{\alpha}{2}, n-1} \frac{s_n}{\sqrt{n}} \right)$$

- o $t_{\alpha, n-1}$ = kritická hodnota **studentova rozdělení t_{n-1}** s $n - 1$ stupni volnosti

- Interval spolehlivosti pro **rozptyl**

- o Náhodný výběr z **normálního rozdělení $N(\mu, \sigma^2)$**
- o Využijeme **výběrový rozptyl s_n^2**
- o Oboustranný interval spolehlivosti

$$\left(\frac{(n-1)s_n^2}{\chi_{\frac{\alpha}{2}, n-1}^2}, \frac{(n-1)s_n^2}{\chi_{1-\frac{\alpha}{2}, n-1}^2} \right)$$

- o $\chi_{\alpha, n-1}^2$ = **kritická hodnota rozdělení χ^2** s $n - 1$ stupni volnosti na hladině α
- o Na rozdíl od předchozích platí **POUZE pro normální rozdělení**

Hypotézy a jejich testování

- Náhodný vektor $X = (x_1 \dots x_n)^T$ s nějakým rozdělením
 - o Tvrzení o tomto rozdělení s neznámou platností = **hypotéza**
- Testování hypotéz – ověřování platnosti na základě pozorování hodnot X
 - o Nulová hypotéza H_0 = tvrzení, o kterém chceme rozhodovat
 - o Alternativní hypotéza H_A/H_1 = opačné tvrzení, které v rozhodovacím procesu stavíme proti H_0
 - o Rozhodovací proces je založen na hod. X , na jehož základě **zamítneme/nezamítneme** hypotézu H_0
- Chyby při testování hypotéz
 - o Chyba 1. druhu – zamítneme H_0 , i když platí
 - o Chyba 2. druhu – nezamítneme H_0 , i když neplatí
- Výsledek – testujeme H_0 proti H_A na hladině významnosti α
 - o (ne)zamítáme H_0 ve prospěch H_A
- Kritický obor W_α = množina realizací X , pro které testování na hladině α skončí zamítnutím H_0
 - o $x \in W_\alpha \Leftrightarrow$ zamítáme H_0 na hladině α
 - o $x \notin W_\alpha \Leftrightarrow$ nezamítáme H_0 na hladině α
- P-hodnota = minimální hladina významnosti \hat{p} , na které lze hypotézu H_0 zamítnout
$$\hat{p} \equiv \hat{p}(x) = \inf\{\alpha \mid x \in W_\alpha\}$$
 - o Je-li p-hodnota menší než naše α , zamítáme H_0
 - o P-hodnota je **horní mez** pro pravděpodobnost, s jakou bude při platnosti nulové hypotézy další realizace x' stejně příznivá zamítnutí, jako ta aktuální - $\hat{p} \geq P_\theta(x' \in W_{\hat{p}})$
 - P_θ ... možné rozdělení X

Parametrické testy

- Určen náh. výběr $X = (x_1 \dots x_n)^T$ z rozdělení určeno par. $\theta \in \Theta \subset \mathbb{R}$, n. v. $x_1 \dots x_n$ jsou iid s tímto rozd.
- Chceme testovat jednoduchou parametrickou hypotézu proti **oboustranné alternativě**:
 - o $H_0: \theta = \theta_0$ proti $H_A: \theta \neq \theta_0$ pro konkrétní hodnotu θ_0
- $(L(x), U(x))$ oboustranný $100 \cdot (1 - \alpha)\%$ interval spolehlivosti pro parametr θ sestavený na základě náhodné veličiny $X \rightarrow \forall \theta \in \Theta: P_\theta(\theta \in (L, U)) = 1 - \alpha$
 - o Zamítneme H_0 , pokud $\theta_0 \notin (L, U)$
 - o Nezamítneme H_0 , pokud $\theta_0 \in (L, U)$
- Parametrické testy proti **jednostranné alternativě**:
 - o $H_0: \theta \leq \theta_0$ proti $H_A: \theta > \theta_0$
- **Jednostranný interval** spolehlivosti typu odpovídajícího alternativní hypotéze – horní interval spolehlivosti $(L, +\infty) \rightarrow \forall \theta \in \Theta: P_\theta(\theta \in (L, +\infty)) = P_0(\theta > L) = 1 - \alpha$
 - o Zamítneme H_0 , pokud $\theta_0 \notin (L, +\infty)$
 - o Nezamítneme H_0 , pokud $\theta_0 \in (L, +\infty)$
- Pro $H_0: \theta \geq \theta_0$ proti $H_A: \theta < \theta_0$ analogicky

Testy o parametrech normálního rozdělení

- $x_1 \dots x_n$ náhodný výběr z $N(\mu, \sigma^2)$
- Test $H_0: \mu = \mu_0$ proti alternativě $H_A: \mu \neq \mu_0$ na hladině významnosti α
 - o Známy rozptyl $\sigma^2 \rightarrow H_0$ zamítneme, pokud μ_0 neleží v intervalu $(\bar{X}_n - z_{\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}}, \bar{X}_n + z_{\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}})$
 - o Neznámý rozptyl $\sigma^2 \rightarrow H_0$ zam., pokud μ_0 neleží v intervalu $(\bar{X}_n - t_{\frac{\alpha}{2}, n-1} \frac{s_n}{\sqrt{n}}, \bar{X}_n + t_{\frac{\alpha}{2}, n-1} \frac{s_n}{\sqrt{n}})$
- Test $H_0: \sigma^2 = \sigma_0^2$ proti alternativě $H_A: \sigma^2 \neq \sigma_0^2$ na hladině významnosti α
 - o H_0 zamítneme, pokud σ_0^2 neleží v intervalu $(\frac{(n-1)s_n^2}{\chi_{\frac{\alpha}{2}, n-1}^2}, \frac{(n-1)s_n^2}{\chi_{1-\frac{\alpha}{2}, n-1}^2})$
- **Jednostranné analogicky**
 - o $H_0: \mu \leq \mu_0 \wedge H_A: \mu > \mu_0 \wedge$ známý rozptyl: $(\bar{X}_n - z_{\alpha} \frac{\sigma}{\sqrt{n}}, +\infty)$
 - o $H_0: \mu \leq \mu_0 \wedge H_A: \mu > \mu_0 \wedge$ neznámý rozptyl: $(\bar{X}_n - t_{\alpha, n-1} \frac{s_n}{\sqrt{n}}, +\infty)$
 - o $H_0: \sigma^2 \leq \sigma_0^2 \wedge H_A: \sigma^2 > \sigma_0^2: ((n-1)s_n^2/\chi_{\alpha, n-1}^2, +\infty)$

Testové statistiky

- **Testová statistika** = sestrojíme statistiku $T \equiv T(x) =$ funkci náhodného vektoru X , u které při platnosti nulové hypotézy známe její rozdělení
- V oblasti možných hodnot T vybereme podmnožinu S_α , pro kterou:

$$\sup_{\theta \in \Theta_0} P_\theta(T \in S_\alpha) \leq \alpha$$

= při platnosti H_0 má T hodnoty v S_α s pravděpodobností nejvýše α

- Zároveň obvykle chceme, aby $\forall \theta \in \Theta_A: P_\theta(T \in S_\alpha) > \sup_{\theta \in \Theta_0} P_\theta(T \in S_\alpha)$
- Testování hypotéz: **Zamítneme** H_0 , jestliže $T \in S_\alpha$, **nezamítneme** H_0 , jestliže $T \notin S_\alpha$

Jednovýběrové testy o střední hodnotě a rozptylu

- **Testy o střední hodnotě normálního rozdělení**

- o $x_1 \dots x_n$ náhodný výběr z $N(\mu, \sigma^2)$ + předpoklad, že známe σ^2
→ Pro testy o hodnotě μ porovnávané s μ_0 uvažujeme testovou statistiku

$$T = \frac{\bar{X}_n - \mu_0}{\sigma/\sqrt{n}}$$

- $T \sim N(u, 1)$, kde $u = \frac{\mu - \mu_0}{\sigma/\sqrt{n}}$
- o Pro test $H_0: \mu = \mu_0$ proti $H_A: \mu \neq \mu_0$ na hladině α : $S_\alpha = (-\infty, -z_{\frac{\alpha}{2}}] \cup [z_{\frac{\alpha}{2}}, +\infty)$
→ $T \in S_\alpha \Leftrightarrow |T| \geq z_{\frac{\alpha}{2}}$
- o Pro test $H_0: \mu \leq \mu_0$ proti $H_A: \mu > \mu_0$ na hladině α : $S_\alpha = [z_\alpha, +\infty)$
→ $T \in S_\alpha \Leftrightarrow |T| \geq z_\alpha$
→ podmínka na zamítnutí H_0 je stejná jako při testu na konfidenčním intervalu
- Je tady jen α , ne $\alpha/2$, pozor

- **Testy o parametrech normálního rozdělení**

- o $x_1 \dots x_n$ náhodný výběr z $N(\mu, \sigma^2)$
- o **Test o střední hodnotě** – testová statistika a kritické obory při známém rozptylu σ^2 :

H_0	H_A	testová statistika T	kritický obor
$\mu = \mu_0$	$\mu \neq \mu_0$	$T = \frac{\bar{X}_n - \mu_0}{\sigma} \sqrt{n}$	$ T \geq z_{\alpha/2}$
$\mu \leq \mu_0$	$\mu > \mu_0$		$T \geq z_\alpha$
$\mu \geq \mu_0$	$\mu < \mu_0$		$T \leq -z_\alpha$

- o **Test o střední hod.** – test. statistika a krit. obory při neznámém rozptylu σ^2 (**jednovýběrový t-test**):

H_0	H_A	testová statistika T	kritický obor
$\mu = \mu_0$	$\mu \neq \mu_0$	$T = \frac{\bar{X}_n - \mu_0}{s_n} \sqrt{n}$	$ T \geq t_{\alpha/2, n-1}$
$\mu \leq \mu_0$	$\mu > \mu_0$		$T \geq t_{\alpha, n-1}$
$\mu \geq \mu_0$	$\mu < \mu_0$		$T \leq -t_{\alpha, n-1}$

- o **Testy o rozptylu** na hladině významnosti α :

H_0	H_A	testová statistika T	kritický obor
$\sigma^2 = \sigma_0^2$	$\sigma^2 \neq \sigma_0^2$	$T = \frac{(n-1)s_n^2}{\sigma_0^2}$	$T \leq \chi_{1-\alpha/2, n-1}^2 \vee T \geq \chi_{\alpha/2, n-1}^2$
$\sigma^2 \leq \sigma_0^2$	$\sigma^2 > \sigma_0^2$		$T \geq \chi_{\alpha, n-1}^2$
$\sigma^2 \geq \sigma_0^2$	$\sigma^2 < \sigma_0^2$		$T \leq \chi_{1-\alpha, n-1}^2$

Párový t-test

- Náhodný výběr $(X_1, Y_1)^T \dots (X_n, Y_n)^T$ z nějakého 2D rozdělení s neznámým vektorem středních h. $(\mu_1, \mu_2)^T$
- Testujeme hypotézu $H_0: \mu_1 = \mu_2$ proti $H_A: \mu_1 \neq \mu_2$
- $Z_i = X_i - Y_i \rightarrow Z_i$ jsou iid se střední hodnotou $\mu_\Delta = \mu_1 - \mu_2$
 - o Předpoklad, že $Z_i \sim N(\mu, \sigma^2)$ kde σ^2 neznáme

→ Lze převést na **párový t-test**

= Jednovýběrový t-test hypotézy $H_0: \mu_\Delta = 0$ proti $H_A: \mu_\Delta \neq 0$

H_0	H_A	testová statistika T	kritický obor
$\mu_1 = \mu_2$	$\mu_1 \neq \mu_2$	$T = \frac{\bar{Z}_n}{s_Z} \sqrt{n}$	$ T \geq t_{\alpha/2, n-1}$
$\mu_1 \leq \mu_2$	$\mu_1 > \mu_2$		$T \geq t_{\alpha, n-1}$
$\mu_1 \geq \mu_2$	$\mu_1 < \mu_2$		$T \leq -t_{\alpha, n-1}$

- o $s_Z^2 \dots$ výběrový rozptyl veličiny Z

Dvouvýběrový t-test

- Náhodné výběry $X_1 \dots X_n \sim N(\mu_1, \sigma_1^2)$ a $Y_1 \dots Y_m \sim N(\mu_2, \sigma_2^2)$ (nezávislé)
- Testujeme hypotézu $H_0: \mu_1 = \mu_2$ proti $H_A: \mu_1 \neq \mu_2$
- Test na základě stat., která má při plat. $\mu_1 = \mu_2$ **studentovo rozdělení** s určitým počtem stupňů volnosti
 - o Záleží, jestli $\sigma_1^2 = \sigma_2^2$ (homoskedasticita)
 - o **Stejné rozptyly** $\sigma_1^2 = \sigma_2^2$:

H_0	H_A	testová statistika T	kritický obor
$\mu_1 = \mu_2$	$\mu_1 \neq \mu_2$	$T = \frac{\bar{X}_n - \bar{Y}_m}{s_{12}} \sqrt{\frac{n \cdot m}{n + m}}$	$ T \geq t_{\alpha/2, n+m-2}$
$\mu_1 \leq \mu_2$	$\mu_1 > \mu_2$		$T \geq t_{\alpha, n+m-2}$
$\mu_1 \geq \mu_2$	$\mu_1 < \mu_2$		$T \leq -t_{\alpha, n+m-2}$

$$s_{12} = \sqrt{\frac{(n-1)s_X^2 + (m-1)s_Y^2}{n+m-2}}$$

- o **Různé rozptyly** $\sigma_1^2 \neq \sigma_2^2$:

H_0	H_A	testová statistika T	kritický obor
$\mu_1 = \mu_2$	$\mu_1 \neq \mu_2$	$T = \frac{\bar{X}_n - \bar{Y}_m}{s_d}$	$ T \geq t_{\alpha/2, n_d}$
$\mu_1 \leq \mu_2$	$\mu_1 > \mu_2$		$T \geq t_{\alpha, n_d}$
$\mu_1 \geq \mu_2$	$\mu_1 < \mu_2$		$T \leq -t_{\alpha, n_d}$

$$s_d = \sqrt{\frac{s_X^2}{n} + \frac{s_Y^2}{m}} \quad \text{a} \quad n_d = \frac{s_d^4}{\frac{1}{n-1} \left(\frac{s_X^2}{n}\right)^2 + \frac{1}{m-1} \left(\frac{s_Y^2}{m}\right)^2}$$

F-test rovnosti rozptylů

- Nezávislé náhodné výběry $X_1 \dots X_n \sim N(\mu_1, \sigma_1^2)$ a $Y_1 \dots Y_m \sim N(\mu_2, \sigma_2^2)$
- Chceme testovat hypotézy porovnávající σ_1^2 a σ_2^2

H_0	H_A	testová statistika T	kritický obor
$\sigma_1^2 = \sigma_2^2$	$\sigma_1^2 \neq \sigma_2^2$	$T = \frac{s_X^2}{s_Y^2}$	$T \leq F_{1-\alpha/2, n-1, m-1} \vee T \geq F_{\alpha/2, n-1, m-1}$
$\sigma_1^2 \leq \sigma_2^2$	$\sigma_1^2 > \sigma_2^2$		$T \geq F_{\alpha, n-1, m-1}$
$\sigma_1^2 \geq \sigma_2^2$	$\sigma_1^2 < \sigma_2^2$		$T \leq F_{1-\alpha, n-1, m-1}$

- $F_{1-\alpha, n-1, m-1} \dots$ kritická hod. Fisher-Snedecorova F-rozdělení s $n-1$ a $m-1$ stupni volnosti, které splňuje

$$F_{1-\alpha, n-1, m-1} = 1/F_{\alpha, n-1, m-1}$$
- Citlivý na normalitu X a Y – při nejistotě např. Levenův test

Testy dobré shody

- D.n.v. X nabývající hodnot $1 \dots k$ s pravděpodobnostmi $p_1 \dots p_k$ – rozdělení $x: p = (p_1 \dots p_k)^T$
- **Multinomické rozdělení**
 - o Provedeme náhodný výběr $X_1 \dots X_n$ z rozdělení p – můžeme výsledek až na pořadí zaznamenat pomocí **četností**, s jakými jednotlivé hodnoty nastaly
 - Dostaneme náhodné veličiny $N_1 \dots N_k$, $N_i = |\{j \mid X_j = i\}|$
 - o **Multinomické rozdělení** = rozdělení tohoto náhodného vektoru $N = (N_1 \dots N_k)^T$
 - Značení $M(n, p)$, určeno psmi:

$$P(N_1 = n_1, \dots, N_k = n_k) = \frac{n!}{n_1! \dots n_k!} p_1^{n_1} \dots p_k^{n_k},$$

- o $k = 2 \dots$ **binomické** rozdělení
- o Vlastnosti multinomického rozdělení
 - Buď $N \sim M(n, p)$
 - Podmíněná rozdělení podmnožin složek n při fixovaných hodnotách zbylých složek jsou opět multinomická
 - Marginální rozdělení jsou binomická: $N_i \sim \text{Binom}(n, p_i)$
 - $EN_i = np_i \quad \forall i$
 - $\text{var} N_i = np_i(1 - p_i) \quad \forall i$
 - $\text{cov}(N_i, N_j) = -np_i p_j \quad \forall i \neq j$

- **Pearsonova statistika** – buď $N \sim M(n, p)$. Pak Pearsonova statistika

$$\chi^2 = \sum_{i=1}^k \frac{(N_i - np_i)^2}{np_i} = \sum_{i=1}^k \frac{N_i^2}{np_i} - n$$

má při $n \rightarrow +\infty$ asymptoticky rozdělené χ^2_{k-1}

- o N_i ... **naměřené četnosti**
- o np_i ... **teoretické četnosti**
- **Test χ^2 při známých parametrech**
 - o Testování **shodnosti diskretních rozdělení**
 - o Náhodný výběr $X = X_1 \dots X_n$ o velikosti n z diskretního rozdělení p'
 - Četnosti $N_1 \dots N_k$ hodnot X mají multinomické rozdělení $M(n, p')$
 - o Testujeme hypotézu H_0 , že skutečné hodnoty pravděpodobností $p_1 \dots p_k$
 - o Provedení testu:

H_0	H_A	testová statistika χ^2	kritický obor
$p' = p$	$p' \neq p$	$\chi^2 = \sum_{i=1}^k \frac{(N_i - np_i)^2}{np_i}$	$\chi^2 \geq \chi^2_{\alpha, k-1}$

- $p = (p_1 \dots p_k)^T$
- $\chi^2_{\alpha, k-1}$... kritická hodnota χ^2 rozdělení s $k - 1$ stupni volnosti
- o Test χ^2 je asymptotický, takže lze použít jen pro dostatečně velký rozsah výběru n : $\forall i: np_i \geq 5$
- **Test χ^2 při neznámých parametrech**
 - o Obecná situace, kdy:
 - H_0 : „náhodný výběr $X_1 \dots X_n$ pochází z rozdělení F_θ , které může záviset na neznámé hodnotě nějakého parametru θ “
 - H_A : náhodný výběr pochází z jiného rozdělení (mimo parametrickou třídu F_θ)
 - o **Převod na test hypotézy pro multinomiální rozdělení:**
 - Rozklad \mathbb{R} do k intervalů $I_1 = (-\infty, b_1], I_2 = (b_1, b_2], \dots, I_k = (b_{k-1}, +\infty)$
 - Četnosti $N_1 \dots N_k$ naměřených hodnot v jednotlivých intervalech $I_1 \dots I_k$, $N_i = |\{j | X_j \in I_i\}|$ mají multinomické rozdělení $M(n, p')$, kde $p'_1 = P(X_1 \in I_i)$
 - H_0 : skutečné hodnoty pravděpodobností jsou $p_1 \dots p_k$ a mohou záviset na **neznámém m-rozměrném parametru** $\theta = (\theta_1 \dots \theta_m)^T$, jehož hodnotu při testování také odhadujeme

$$\chi^2(\theta) = \sum_{i=1}^k \frac{(N_i - np_i(\theta))^2}{np_i(\theta)}$$

- $\hat{\theta}$ = hodnota θ minimalizující
- Bodový odhad θ = **odhad metodou minimálního χ^2**
 - Pro něj má statistika $\chi^2(\hat{\theta})$ asymptoticky χ^2_{k-m-1} rozdělení

H_0	H_A	testová statistika χ^2	kritický obor
$p' = p$	$p' \neq p$	$\chi^2 = \sum_{i=1}^k \frac{(N_i - np_i)^2}{np_i}$	$\chi^2 \geq \chi^2_{\alpha, k-m-1}$

- o Provedení testu (při $p_i = p_i(\hat{\theta})$)
 - Počet stupňů volnosti = # chlůvků = # odhadovaných parametrů - 1
 - Zás platí $np_i \geq 5$

Test nezávislosti v kontingenčních tabulkách

- Máme náhodný vektor $X = (Y, Z)^T$ s diskretním rozdělením Y hodnoty $1 \dots r$, Z hodnoty $1 \dots c$
- Sdružené a marginální pravděpodobnosti:

$$p_{ij} = P(Y = i, Z = j), \quad p_{i\bullet} = \sum_j p_{ij}, \quad p_{\bullet j} = \sum_i p_{ij}$$

- Ještě máme náhodný výběr z X o velikosti n
 - o N_{ij} počet výsledků, kdy nastala dvojice (i, j) , $N_{ij} = |\{k | Y_k = i, Z_k = j\}|$
 - o Náhodné veličiny N_{ij} mají sdružené multinomiální rozdělení s parametrem n a pravděpodobnostmi p_{ij}

- **Kontingenční tabulka** = náhodná matice N rozměru $r \times c$ se složkami N_{ij}

- o **Marginální četnosti:**

$$N_{i\bullet} = \sum_j N_{ij}, \quad N_{\bullet j} = \sum_i N_{ij}$$

Kontingenční tabulka

Y	Z			Σ
	1	...	c	
1	N_{11}	...	N_{1c}	$N_{1\bullet}$
...
r	N_{r1}	...	N_{rc}	$N_{r\bullet}$
Σ	$N_{\bullet 1}$...	$N_{\bullet c}$	n

Matice pravděpodobností

Y	Z			Σ
	1	...	c	
1	p_{11}	...	p_{1c}	$p_{1\bullet}$
...
r	p_{r1}	...	p_{rc}	$p_{r\bullet}$
Σ	$p_{\bullet 1}$...	$p_{\bullet c}$	1

- o Pro **celkový počet** platí:

$$n = \sum_i N_{i\bullet} = \sum_j N_{\bullet j} = \sum_{i,j} N_{ij}$$

- o Chceme testovat nezávislost veličin Y a Z

$$H_0 : p_{ij} = p_{i\bullet} p_{\bullet j} \quad \text{pro každé } i, j$$

- Pravděpodobnosti p_{ij} funkcemi marginálních pravděpodobností $p_{i\bullet}, p_{\bullet j}$
- Počet nezávislých parametrů je $m = (c - 1) + (r - 1)$, protože $\sum_i p_{i\bullet} = \sum_j p_{\bullet j} = 1$

- **Test nezávislosti** → test χ^2

- o $H_0 : p_{ij} = p_{i\bullet} p_{\bullet j}$, s $m = c + r - 2$ neznámými parametry $p_{i\bullet}, p_{\bullet j}$

- o Odhady metodou min. χ^2 :

$$\hat{p}_{i\bullet} = \frac{N_{i\bullet}}{n} \quad \text{a} \quad \hat{p}_{\bullet j} = \frac{N_{\bullet j}}{n}$$

- o Statistika χ^2 má asymptoticky χ^2 rozd. s $k - m - 1 = rc - (c + r - 2) - 1 = (r - 1)(c - 1)$ stupni volnosti

- o Provedení testu:

H_0	H_A	testová statistika χ^2	kritický obor
$p_{ij} = p_{i\bullet} p_{\bullet j}$	$p_{ij} \neq p_{i\bullet} p_{\bullet j}$	$\chi^2 = \sum_{i=1}^r \sum_{j=1}^c \frac{\left(N_{ij} - \frac{N_{i\bullet} N_{\bullet j}}{n} \right)^2}{\frac{N_{i\bullet} N_{\bullet j}}{n}}$	$\chi^2 \geq \chi_{\alpha, (r-1)(c-1)}^2$

- Počet **stupňů volnosti** (degrees of freedom dF) = $(\# \text{ řádků} - 1)(\# \text{ sloupců} - 1)$

- o Alternativní výpočet statistiky χ^2 :

$$\chi^2 = n \sum_{i=1}^r \sum_{j=1}^c \frac{N_{ij}^2}{N_{i\bullet} N_{\bullet j}} - n$$

7. Základy teorie informace a kódování, entropie.

NI-VSM

Entropie

- **Entropie** = „míra neuspořádanosti“
- Uvažujeme **diskrétní náhodné veličiny**
 - o Množina hodnot $X \dots \mathcal{X}$
 - o Pravděpodobnostní funkce $X \vee x \dots p(x) = P(X = x)$
 - o Pravděpodobnostní rozdělení $\dots p$
 - o Argument pravděpodobnostní funkce – o kterou veličinu se jedná
- **Entropie $H(X)$ diskrétní náhodné veličiny:**

$$H(X) = - \sum_x p(x) \log p(x)$$

- o Log o základu 2, $0 \log 0 = 0$
- o Entropie závisí pouze na rozdělení p veličiny X
- o Je invariantní vůči transformacím: $H(X) = H(g(x))$
- **Jednotky entropie** – báze $b > 1$ logaritmu – $H_b(X)$, b označuje jednotky entropie
 - o $b = 2 \dots$ bit
 - o $b = 10 \dots$ digit
 - o $b = e \dots$ nat
 - o Přechod mezi bázemi: $H_b(X) = (\log_b a) \cdot H_a(X) = (\log_b 2) H(X)$
- Entropie **jako očekávaná míra neurčitosti**
 - o $H(X)$ lze chápat jako **střední hodnota**:

$$H(X) = -E \log p(x) = EI(X)$$

$$I(X) = -\log p(x)$$

- $I(X)$ = **vlastní informace** = míra neurčitosti $X \in \mathcal{X}$
→ entropie je očekávaná míra neurčitosti X

- o Míra neurčitosti je vždy nezáporná a pro jisté jevy 0
- o Méně pravděpodobný jev → vyšší míra neurčitosti
- o $I(X)$ se při pozorování nezávislých jevů sčítá

Vlastnosti entropie

- o **Nezápornost** entropie: $H(X) \geq 0$
- o Entropie je konkávní funkcí rozdělení
- o V deterministických případech je entropie 0
- o Maximální $H(X)$ → rovnoměrné rozdělení – nejvyšší neurčitost

- **Sdružená entropie $H(X, Y)$ diskrétních náhodných veličin X, Y se sdruženým rozdělením $p(x, y)$:**

$$H(X, Y) = - \sum_x \sum_y p(x, y) \log p(x, y)$$

- o Sdružená entropie diskrétního náhodného vektoru X se sdruženým rozdělením $p(x)$:

$$H(X) = - \sum_x p(x) \log p(x)$$

- o Alternativně $H(X, Y) = -E \log p(X, Y)$

- **Podmíněná entropie $H(Y|X)$ diskrétních náhodných veličin X, Y se sdruženým rozdělením $p(x, y)$:**

$$H(Y|X) = - \sum_x \sum_y p(x, y) \log p(y|x)$$

- o $p(y|x) = \frac{p(x, y)}{p(x)}$

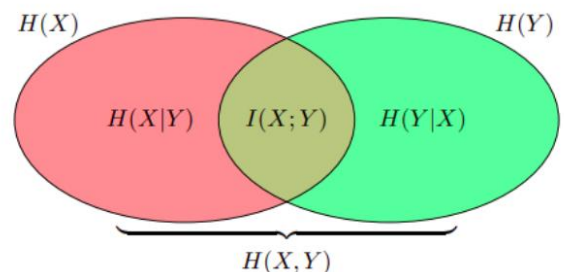
- o Alternativně $H(X, Y) = -E \log H(Y|X)$

- $Z p(x, y) = p(y|x) \cdot p(x)$: $(Y|X) = \sum_x p(x)(Y|X = x) = E_x H(Y|X = x)$
→ $H(Y|X = x) = - \sum_y p(y|x) \log p(y|x)$

- **Řetězové pravidlo:**

$$H(X, Y) = H(X) + H(Y|X)$$

→ $H(Y|X) = H(X, Y) - H(X)$... určuje, která část informace je ve veličině Y navíc oproti tomu, co je v X



- **Relativní entropie** = Kullback-Leiblerova vzdálenost $D(p||q)$:

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

- Pokud $\exists x: p(x) > 0, q(x) = 0: D(p||q) = +\infty$
- Je to „vzdálenost“ – nezáporná a 0, jen pokud $p = q$
 - Ale ne opravdová, neplatí $D(q||p)$ ani trojúhelníková nerovnost
 - Alternativně:

$$D(p||q) = E_p \log \frac{p(x)}{q(x)}$$

- **Vzájemná informace** $I(X; Y)$:

$$I(X; Y) = \sum_x \sum_y p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

= relativní entropie skutečného sdruženého rozdělení a rozdělení nezáv. veličin se stejnými marginálami:

$$I(X; Y) = D(p(x, y)||p(x)p(y))$$

- Symetrie: $I(X; Y) = I(Y; X)$
- Z nezápornosti relativní entropie: $I(X; Y) \geq 0$

- **Vztah vzájemné informace a entropie**

$$I(X; Y) = H(Y) - H(Y|X) = H(X) - H(X|Y)$$

- Odvození přes věty o logaritmech
- $I(X; Y) = H(X) + H(Y) - H(X, Y)$
- $I(X; X) = H(X)$ – vlastní informace

- **Informační nerovnost:** $p(x), q(x)$ možná rozdělení X : $D(p||q) \geq 0$

- Rovnost pouze pokud $p(x) = q(x) \forall x \in \mathcal{X}$
- Důsledky:
 - **Nezápornost vzájemné informace** – pro dvě d.n.v. X, Y : $I(X; Y) \geq 0$
 - Pokud rovnost, pak jsou nezávislé
 - I je číselná charakteristika sdruženého rozdělení, která je schopná poznat nezávislost
 - **Maximalizace entropie** – pro d.n.v. X s hodnotami z \mathcal{X} : $H(X) \leq \log |\mathcal{X}|$
 - $|\mathcal{X}|$... počet prvků množiny \mathcal{X} – rovnost, pokud rovnoměrné rozdělení
 - Entropie je maximalizována rovnoměrným rozdělením
 - **Podmiňování redukuje entropii:** $H(Y|X) \leq H(X)$
 - Rovnost, pokud jsou X a Y nezávislé
 - „informace neublíží“ – znalost n.v. Y může v průměru pouze redukovat neurč. v X
 - Pouze v průměru, samotné $H(X|Y = y)$ může být pro nějaké y větší než $H(X)$, ale:

$$H(X|Y) = \sum_y p(y) H(X|Y = y) \leq H(X)$$

Teorie kódování

- Jak zapsat zdrojovou zprávu, tak, aby následný přenos byl co nejefektivnější
- **D-ární abeceda** – abeceda \mathcal{D} obsahující D přenositelných symbolů
- **Zpráva** $x_1 \dots x_n$ je posloupnost znaků z \mathcal{X}
- Chceme co nejkratší zakódovanou zprávu
- Zobrazení $C: \mathcal{X} \rightarrow \mathcal{D}^*$ z množiny \mathcal{X} do množiny \mathcal{D}^* konečných řetězců symbolů D -ární abecedy \mathcal{D} nazýváme **kód** diskretní náhodné veličiny X
 - Obraz $C(x)$ = **kódové slovo** příslušného prvku x a jeho délku značíme $l(x)$
 - $\mathcal{D}^* = \bigcup_{k=1}^{\infty} \mathcal{D}^k$
 - \mathcal{D}^k ...řetězec symbolů z \mathcal{D} délky k
- **Střední délka** $L(C)$ kódu C náhodné veličiny X s rozdělením $p(x)$:

$$L(C) = \sum_x l(x)p(x)$$

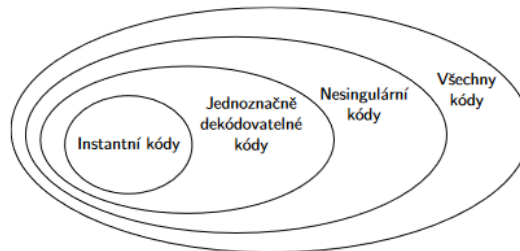
- $l(x)$...**délka kódového slova** příslušejícího k prvku $x \in \mathcal{X}$:
 $\rightarrow L(C) = E l(x)$

- Typy kódů

- **Nesingulární kód** C d.n.v. X – pokud je C prosté zobrazení

$$\forall x, x' \in X: x \neq x' \Rightarrow C(x) \neq C(x')$$
 - Dostačující pro schopnost rekonstruovat z kódových slov jednotlivé hodnoty X
 - Není dostačující pro dekódování posloupnosti hodnot X (celých zpráv)
- **Jednoznačně dekódovatelný kód** C – pokud je C^* nesingulární
 - C^* = rozšíření kódu C – zobrazení X^* do \mathcal{D}^* :

$$C^*(x_1 \dots x_n) = C(x_1) \dots C(x_n)$$
 - Zápis jednotlivých kódových slov po sobě
 - Jsme schopni jednoznačně dekódovat libovolnou přijatou zprávu
- Kód je **instantní** (prefixový), pokud žádné kódové slovo není prefixem jiného kódového slova
- Hierarchie kódů:



- **Kraftova nerovnost** – pro libovolný instantní kód nad D -ární abecedou musí délky kódových slov $l_1 \dots l_n$ splnit nerovnost:

$$\sum_i D^{-l_i} \leq 1$$

Navíc, ke každé n -tici délek, které splní tuto nerovnost, existuje instantní kód s kódovými slovy těchto délek

- Pro jednoznačně dekódovatelné kódy analogicky (McMillanova věta)
 - Ke každému jednoznačně dekódovatelnému kódu lze sestavit instantní kód, který má stejně dlouhá kódová slova

- Optimální kódy

- **Střední délka** $L(C)$ instantního D -árního kódu C d.n.v. X je:

$$L(C) \geq H_D(X)$$
 - Rovnost, právě když $D^{-l_i} = p_i \ \forall i = 1 \dots |X|$
 - $p_i = p(x_i)$
- **Optimální kód** = kód o nejmenší střední délce
- Uvažujme optimální instantní kód C^*

$$H_D(X) \leq L(C^*) < H_D(X) + 1$$
 - Optimálním kódem se od dolní meze dané entropie můžeme vzdálit maximálně o 1

- Huffmanovo kódování

- Algoritmus na sestavení binárního Huffmanova kódu:
 1. Spojíme 2 nejméně pravděpodobné hodnoty → nové rozdělení s o 1 menším počtem hodnot
 2. Opakujeme, dokud nezůstane 1 hodnota → prázdný řetěz jako kódové slovo
 3. Zpětným chodem zkonstruujeme kódová slova všech původních hodnot
 4. Pro hodnotu X , která vznikla spojením u a v vytvoříme kódové slovo méně pravděpodobné hodnoty připojením 1 za kódové slovo $C(u)$ a analogicky kódové slovo více pravděpodobné hodnoty připojením 0 za $C(v)$
 - Tzn. Pokud $\{u, v\} \mapsto x$ a $p(u) \leq p(v)$, tak $C(u) = C(x)1$ a $C(v) = C(x)0$
- Huffmanův kód je **optimální** – je-li C^* Huffmanův kód a C' libovolný jednoznačně dekódovatelný kód, potom $L(C^*) \leq L(C')$
- Algoritmus sestavení je hladový algoritmus, který **lokálně agreguje** 2 nejméně pravděpodobné hodnoty

8. Markovské řetězce s diskretním časem. Jejich limitní vlastnosti.

NI-VSM

- **Náhodný proces** – buďte (Ω, \mathcal{F}, P) pravděpodobnostní prostor a $T \subseteq \mathbb{R}$ indexová množina. Náhodná proces je systém náhodných veličin

$$X = \{X_t | t \in T\}, X_t: \Omega \rightarrow \mathbb{R}$$
- **Množina T** lze chápat jako **čas** → časová souslednost dána uspořádáním T
 - o **Diskrétní čas** – je-li T nejvýše spočetná
 - o **Spojité čas** – je-li T nespočetná
- **Množina stavů S** = minimální podmnožina \mathbb{R} , pro kterou platí, že $\forall t \in T P(X_t \in S) = 1$
 - o S = společná množina hodnot X_t (diskrétní/kontinuum)
- **Trajektorie náhodného procesu**
 - o **Náhodný proces** $X = \{X_t | t \in T\}$ = zobrazení z Ω do prostoru funkcí $X: \Omega \rightarrow \{f: T \rightarrow S\}$
 - o $X_t(w)$...hodnota funkce $X(t, w)$ proměnných t a w (w je elementární jev)
 - o **Trajektorie/realizace** náhodného procesu X = funkce $f: T \rightarrow \mathbb{R}: f(t) = X_t(w)$
- Spočetná množina $S: S = \mathbb{N}, S = \{1, \dots, |S|\}$
- Diskrétní čas: $T = \mathbb{N}_0: X = \{X_n | n = 0, 1, 2, \dots\}$
- Rozdělení v čase $n \in \mathbb{N}_0$ char. pravděpodobnostní funkcí: $p_i(n) = P(X_n = i), p(n) = (p_1(n), p_2(n), \dots)$
- **Matice pravděpodobnostního přechodu** za čas mezi n a $m \geq n$:

$$P_{ij}(n, m) = P(X_m = j | X_n = i) \quad P(m, n) = (P_{ij}(n, m))_{i, j \in S}$$

Markovský řetězec

- Náhodný proces $\{X_n | n \in \mathbb{N}_0\}$ s nejvýše spočetnou množinou stavů S nazýváme **markovský řetězec s diskretním časem**, pokud pro každý stav s splňuje **markovskou podmínku**:

$$P(X_n = s | X_{n-1} = s_{n-1}, \dots, X_1 = s_1, X_0 = s_0) = P(X_n = s | X_{n-1} = s_{n-1})$$
- Následující podmínky jsou **ekvivalentní markovské podmínce**:

$$P(X_{n+m} = s | X_m = s_m, \dots, X_1 = s_1, X_0 = s_0) = P(X_{n+m} = s | X_m = s_m)$$

$$P(X_{n_k} = s_k | X_{n_k-1} = s_{k-1}, \dots, X_{n_0} = s_0) = P(X_{n_k} = s_k | X_{n_k-1} = s_{k-1})$$
- **Ekvivalentní definice markovského řetězce** – náhodný proces s nejvýše spočetnou množinou stavů S je markovský řetězec právě tehdy, když pro každé k, n, s :

$$P(X_{n_0} = s_0, \dots, X_{n_k} = s_k) =$$

$$= p_{s_0}(n_0) \cdot P_{s_0 s_1}(n_0, n_1) \cdot P_{s_1 s_2}(n_1, n_2) \cdot \dots \cdot P_{s_{k-1} s_k}(n_{k-1}, n_k)$$
- **Chapman-Kolmogorova rovnice** – pro matice přechodu markovského řetězce platí $\forall n \leq m \leq r \in \mathbb{N}_0$:

$$P(n, r) = P(n, m)P(m, r)$$

Homogenní markovský řetězec

- Markovský řetězec je **homogenní**, pokud $\forall n \in \mathbb{N}, \forall i, j \in S$:

$$P(X_{n+1} = j | X_n = i) = P(X_1 = j | X_0 = i)$$
- Pro homogenní markovský řetězec platí $\forall m, n \in \mathbb{N}_0$:

$$P(m, m+n) = P(0, n) = P^n$$
- Pro homogenní m. ř. definujeme (jednokrokovou) **matici přechodu**:

$$P = P(0, 1) = (P(X_1 = j | X_0 = i))_{i, j \in S}$$
 - o Značení $P(n) = P(0, n) = P^n$
 - o **Pravděpodobnost přechodu** ze stavu i do j během n kroků:

$$P(X_n = j | X_0 = i) = P_{ij}(n) = (P^n)_{ij}$$
 - o Ch-K. rovnice pro homo. m. ř.:

$$P(n+m) = P(n)P(m)$$
 - Jiný zápis pro $P^{n+m} = P^n P^m$

- Rozdělení náhodné veličiny X_n

- Pro $n > m$ platí, že:

$$\begin{aligned} p_j(n) = P(X_n = j) &= \sum_{i \in S} P(X_m = i) P(X_n = j | X_m = i) \\ &= \sum_{i \in S} p_i(m) P_{ij}(m, n). \end{aligned}$$

- Maticový zápis:

$$p(n) = p(m)P(m, n) = p(0)P(0, n)$$

- Tím pádem pro h. m. ř. platí:

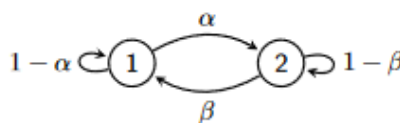
$$p(n) = p(m)P^{n-m} = p(0)P^n$$

- Stochastické matice – matice přechodu je stochastická:

- P má **nezáporné prvky**: $P_{ij} \geq 0 \forall i, j \in S$
- Součet řádků P je roven 1**: $\sum_{j \in S} P_{ij} = 1 \forall i \in S$
- Součin stochastických matic je opět stochastická matice
- K libovolné čtvercové stochastické matici P existuje homogenní markovský řetězec s diskrétním časem takový, že P je jeho maticí přechodu

- Příklad diagramu přechodu a matice přechodu:

$$P = \begin{pmatrix} 1 - \alpha & \alpha \\ \beta & 1 - \beta \end{pmatrix}$$



Stacionární rozdělení

- Počáteční rozdělení $p(0)$ – může být libovolný vektor $q \in \mathbb{R}^{|S|}$ splňující:

$$\forall i \in S : q_i \geq 0, \quad \text{a} \quad \sum_{i \in S} q_i = 1$$

- Maticí přechodu P definujeme třídu náhodných procesů $\{X_n \mid n \in \mathbb{N}_0\}$ definovaných na prostorech (Ω, F, P_q) , pro které platí:

$$P_q(X_0 = i) = q_i, \quad P_q(X_n = i) = (qP^n)_i$$

- Stacionární rozdělení – buď $\{X_n \mid n \in \mathbb{N}_0\}$ homogenní markovský řetězec s maticí přechodu P . Pokud existuje vektor π takový, že:

- $\forall i \in S : \pi_i \geq 0$
- $\sum_{i \in S} \pi_i = 1$

pro který platí $\pi P = \pi$, nazýváme jej stacionárním rozdělením řetězce

- Existuje-li stacionární rozdělení, pak

$$p(0) = \pi \implies p(n) = \pi P^n = \pi P^{n-1} = \dots = \pi P = \pi$$

- Stacionární rozdělení má požadovanou vlastnost $P_\pi(X_n = i) = \pi_i$

Klasifikace stavů markovského řetězce

- Stav i nazveme **trvalý (rekurentní)**, pokud:

$$P(\exists n \in \mathbb{N} : X_n = i | X_0 = i) = 1$$

- Trvalost** = každá trajektorie začínající v i se někdy vrátí do i skoro jistě

- Stav i nazveme **přechodný (transientní)**, pokud není trvalý, tj:

$$P(\exists n \in \mathbb{N} : X_n = i | X_0 = i) < 1$$

- Přechodnost** = existuje hodně trajektorií, které se do i už nikdy nevrátí

- Čas první návštěvy stavu $i \in S$:

$$\tau_i = \min\{n \in \mathbb{N} | X_n = i\}$$

je-li množina neprázdná, a $\tau_i = +\infty$, je-li množina prázdná

- f_{ij} ...pravděpodobnost, že **řetězec někdy navštíví j** , startoval-li v i
- $f_{ij}(n)$...pst, že **1. návštěva j při startu z i nastane v n -tém kroku**
- Z toho vyplývá:

- Stav je **trvalý** $\iff f_{ii} = P(\tau_i < +\infty | X_0 = i) = 1$
- Stav je **přechodný** $\iff f_{ii} = P(\tau_i < +\infty | X_0 = i) < 1$

- Střední doba návratu do $i \in S$:

$$\mu_i := E(\tau_i | X_0 = i) = \begin{cases} \sum_{n=1}^{\infty} n f_{ii}(n) & \text{je-li } i \text{ trvalý,} \\ +\infty & \text{je-li } i \text{ přechodný} \end{cases}$$

- o Trvalý stav i = nenulový, pokud je střední doba návratu konečná: $\mu_i < +\infty$
 - Jinak je stav nulový

- Perioda stavu $i \in S$:

$$d(i) = \gcd\{n \in \mathbb{N} \mid P_{ii}(n) > 0\}$$

= největší společný dělitel časů, kdy se řetězec vrátí do stavu i

- o Periodický stav = $d(i) > 1$
- o Aperiodický stav = $d(i) = 1$
- o $P_{ii}(n) = P_{ii}^n$, je-li stav periodický s $d(i)$, pak $\forall n \in \mathbb{N}_0$:

$$n \notin \{k \cdot d(i) \mid k \in \mathbb{N}_0\} \implies P_{ii}(n) = 0$$
- o Pokud existují $n, m \in \mathbb{N}$ takové, že $P_{ij}(n) > 0$ a $P_{ji}(m) > 0$ = stavy jsou vzájemně dosažitelné, pak:

$$d(i) = d(j)$$

- Klasifikace stavů pomocí matice přechodu P – stav i m.ř. je:

- o Přechodný $\Leftrightarrow \sum_{n=0}^{\infty} P_{ii}(n) < +\infty$, potom $P_{ji}(n) \rightarrow 0$
- o Trvalý nulový $\Leftrightarrow \sum_{n=0}^{\infty} P_{ii}(n) = +\infty$ a $\lim_{n \rightarrow \infty} P_{ii}(n) = 0$, potom $P_{ji}(n) \rightarrow 0$
- o Trvalý nenulový aperiodický $\Leftrightarrow \lim_{n \rightarrow \infty} P_{ii}(n) = 1/\mu_i > 0$, potom $P_{ji}(n) \rightarrow f_{ji}/\mu_i$
- o Trvalý nenulový periodický \Leftrightarrow má periodu $d(i)$ a $\lim_{k \rightarrow \infty} P_{ii}(kd(i)) = d(i)/\mu_i > 0$

Rozklad množiny stavů

- Dosažitelný stav j ze stavu i ($i \rightarrow j$), pokud se lze dostat z i do j v konečném čase = $\exists n \in \mathbb{N}_0: P_{ij}(n) > 0$
- Stavy i a j jsou vzájemně dosažitelné ($i \leftrightarrow j$), pokud $i \rightarrow j$ a $j \rightarrow i$
 - o $i \neq j: i \rightarrow j \Leftrightarrow f_{ij} > 0$
 - o Relace \leftrightarrow je ekvivalence
 - o Pokud $i \leftrightarrow j$, jsou oba stejného typu

- Uzavřená množina $C \subseteq S: \forall i \in C, \forall j \notin C: P_{ij} = 0$

- o Uspořádejme stavy tak, aby C byly na konci ($S = (C', C)$):

$$P = \begin{matrix} & \begin{matrix} C' & C \end{matrix} \\ \begin{matrix} C' \\ C \end{matrix} & \begin{pmatrix} A & B_1 \\ 0 & C \end{pmatrix} \end{matrix}, \quad P^2 = \begin{matrix} & \begin{matrix} C' & C \end{matrix} \\ \begin{matrix} C' \\ C \end{matrix} & \begin{pmatrix} A^2 & B_2 \\ 0 & C^2 \end{pmatrix} \end{matrix}, \quad P^n = \begin{matrix} & \begin{matrix} C' & C \end{matrix} \\ \begin{matrix} C' \\ C \end{matrix} & \begin{pmatrix} A^n & B_n \\ 0 & C^n \end{pmatrix} \end{matrix}$$

- o Z uzavřené množiny řetězec neuteče: $\forall i \in C \text{ a } i \rightarrow j \implies j \in C$
- o Je-li uzavřená množina C tvořena jediným stavem $C = \{S\}$, pak tento stav = pohlcující (absorbční)
- Rozložitelnost – množina stavů $C \subseteq S$ nerozložitelná (ireducibilní), pokud $\forall i, j \in C$ platí $i \leftrightarrow j$
 - o Markovský řetězec je nerozložitelný, pokud S je nerozložitelná
- Jednoznačný rozklad – množina stavů S lze jednoznačně rozložit do tvaru $S = T \cup C_1 \cup C_2 \cup \dots$
 - o T ... množina všech přechodných stavů
 - o C_1, C_2, \dots vzájemně disjunkt ní nerozložitelné uzavřené množiny trvalých stavů
 - o Matice přechodu po uspořádání $S = (T, C_1, C_2, \dots)$ má tvar:

$$P = \begin{matrix} & \begin{matrix} T & C_1 & C_2 & \dots \end{matrix} \\ \begin{matrix} T \\ C_1 \\ C_2 \\ \vdots \end{matrix} & \begin{pmatrix} T & R_1 & R_2 & \dots \\ 0 & C_1 & 0 & \dots \\ 0 & 0 & C_2 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \end{matrix}$$

- T ... čtvercová matice přechodů mezi přechodnými stavy v T
- C_i ... čtvercová matice přechodů mezi trvalými stavy v C_i
- R_i ... matice přechodů z množiny přechodných stavů do množiny trvalých stavů C_i

- Je-li stav $i \in S$ trvalý a $i \rightarrow j$, pak $j \in S$ je také trvalý a $j \rightarrow i$

- V řetězci s konečně mnoha stavy $|S| < +\infty$
 - o Nemohou být všechny stavy přechodné (takže **alespoň 1 trvalý**)
 - o Neexistují stavy trvalé nulové (takže trvalé stavy **vždy nenulové**)
 - o Tím pádem alespoň 1 **uzavřená nerozložitelná množina** trvalých nenulových stavů (bude jich konečně mnoho), zbylé stavy přechodné
- Mějme konečnou množinu stavů S a $i \in S$. $S_i = \{j \in S \mid i \rightarrow j\}$ množina stavů dosažitelných z i . Pak
 - o Pokud $\forall j \in S_i: j \rightarrow i$, stav i je **trvalý nenulový**
 - o Pokud $\exists j \in S_i: j \nrightarrow i$, stav i je **přechodný**

Existence stacionárního rozdělení

- Pro rozložitelný markovský řetězec platí:
 - o Jsou-li všechny stavy **přechodné nebo trvalé nulové**, stacionární rozdělení **neexistuje**
 - o Jsou-li všechny stavy **trvalé nenulové**, stacionární rozdělení π existuje a je **jediné**. Jsou-li navíc všechny stavy aperiodické, platí $\forall i, j \in S: \pi_j = \lim_{n \rightarrow \infty} P_{ij}(n) = 1/\mu_j > 0$, tedy
$$\pi = \lim_{n \rightarrow \infty} p(n) \text{ pro libovolné } p(0)$$
- Je-li množina stavů konečná, pak stacionární rozdělení existuje
- **Počet stacionárních rozdělení**
 - o Pro každou množinu C_r trvalých nenulových stavů, $r \in I$, existuje **stacionární rozdělení** $\tilde{\pi}^{(r)}$ splňující:
$$\tilde{\pi}^{(r)} \cdot C_r = \tilde{\pi}^{(r)}$$
 - o Pak platí, že vektor $\pi^{(r)} := (0, \dots, 0, \tilde{\pi}^{(r)}, 0, \dots, 0)$ řeší rovnici
$$\pi^{(r)} \cdot P = \pi^{(r)}$$
 - o Z toho plyne, že máme celkem tolik lineárně nezávislých **stacionárních rozdělení** $\pi^{(r)}$, $r \in I$, kolik je (nenulových) množin C_r
 - o Pak libovolná konvexní kombinace

$$\sum_{r \in I} \lambda_r \pi^{(r)}, \quad \lambda_r \geq 0, \quad \sum_{r \in I} \lambda_r = 1$$

je stacionárním rozdělením procesu s maticí přechodu P

Pravděpodobnosti pohlcení

- Množinu stavů S lze jednoznačně rozložit do tvaru $S = T \cup C_1 \cup C_2 \cup \dots$, T přechodné, C_r nerozložitelné trvalé
- S konečná \rightarrow alespoň 1 neprázdná C_r , všechny stavy jsou nenulové
- $i \in T$ přechodný $\rightarrow \forall j \in S$:

$$\lim_{n \rightarrow \infty} P_{ji}(n) = 0 \quad \wedge \quad \sum_{n=1}^{+\infty} P_{ii}(n) < +\infty$$

- Je-li S konečná, bude řetězec v konečné době pohlcen jednou z množin C_r
- **Čas absorpce** = náhodná veličina $\tau_A: \Omega \rightarrow \{0, 1, \dots, +\infty\}$:

$$\tau_A(\omega) := \min\{n \in \mathbb{N}_0 \mid X_n(\omega) \notin T\}$$

- o Je-li množina neprázdná, jinak $\tau_A(\omega) = +\infty$
- o Je-li množina stavů S konečná, pak $P(\tau_A = +\infty \mid X_0 = i) \forall i \in T$
- Označme U_{ij} **pravděpodobnost, že řetězec startující v $i \in T$ opustí T přechodem do stavu $j \in C$:**

$$U_{ij} = P(X_{\tau_A} = j \mid X_0 = i)$$

- o $U = (U_{ij})_{i \in T, j \in C} \in \mathbb{R}^{|T| \times |C|}$
- o Pravděpodobnost pohlcení v množině C :

$$P(X_{\tau_A} \in C_r \mid X_0 = i) = \sum_{j \in C_r} U_{ij}$$

- o Pokud nás nezajímá, **kudy se řetězec do uzavřené množiny C_r dostal**, je výhodné sloučit stavy $j \in C_r$ do jednoho „superstavu“:

- Matice přechodu P :

$$P = \begin{matrix} & \begin{matrix} T & C \end{matrix} \\ \begin{matrix} T \\ C \end{matrix} & \begin{pmatrix} \mathbf{T} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \end{matrix}$$

- Matice pravděpodobností pohlcení U je řešením rovnice

$$U = R + TU$$

- o Řešení rovnice:

$$U = R + TU \Leftrightarrow R = U - TU = (I - T)U$$

- o Je-li matice $(I - T)$ regulární, existuje jediné řešení:

$$U = (I - T)^{-1}R$$

- o Buď A čtvercová matice taková, že $A^n \rightarrow 0$ při $n \rightarrow \infty$. Pak matice $I - A$ je regulární a platí $(I - A)^{-1} = I + A + A^2 + \dots = \sum_{k=0}^{\infty} A^k$

- o Pro matici pravděpodobnosti pohlcení platí $U = (I - T)^{-1}R$

- Počet návštěv stavu $j \in C =$ náhodná veličina W_j :

$$W_j := \sum_{n=0}^{\infty} \mathbf{1}_{\{X_n=j\}}$$

- o Pro $j \in C$ trvalý tedy platí $W_j = +\infty$ skoro jistě: $P(W_j = +\infty) = 1$

- o Pro $i \in T$ přechodný naopak $P(W_j < +\infty) = 1$

- o $E \mathbf{1}_{\{X_n=j\}} = 0 \cdot P(X_n \neq j) + 1 \cdot P(X_n = j) = P(X_n = j)$

- $N_{ik} =$ střední počet návštěv stavu $k \in T$, jestliže řetězec startuje v $i \in T$

$$N_{ik} = E(W_k | X_0 = i)$$

- o Pro matici $N = (N_{ik})_{i,k \in T} \in \mathbb{R}^{|T| \times |T|}$:

$$N = (I - T)^{-1}$$

- o N ... fundamentální matice řetězce

- Střední doba do pohlcení při startu v $i \in T$:

$$E(\tau_A | X_0 = i) = \left[(I - T)^{-1} \cdot \mathbf{1} \right]_i$$

- Limita matice C^n

- o Struktura matice C odpovídající trvalým stavům

$$C = \begin{matrix} & \begin{matrix} c_1 & c_2 & \dots \end{matrix} \\ \begin{matrix} c_1 \\ c_2 \\ \vdots \end{matrix} & \begin{pmatrix} C_1 & \mathbf{0} & \dots \\ \mathbf{0} & C_2 & \dots \\ \vdots & \vdots & \ddots \end{pmatrix} \end{matrix} \Rightarrow C^n = \begin{matrix} & \begin{matrix} c_1 & c_2 & \dots \end{matrix} \\ \begin{matrix} c_1 \\ c_2 \\ \vdots \end{matrix} & \begin{pmatrix} C_1^n & \mathbf{0} & \dots \\ \mathbf{0} & C_2^n & \dots \\ \vdots & \vdots & \ddots \end{pmatrix} \end{matrix}$$

- o Předpoklad, že stavy trvalé jsou aperiodické. Pak $\forall C_r$:

$$(C_r)^n \xrightarrow{n \rightarrow +\infty} \tilde{C}_r, \quad \text{kde} \quad (\tilde{C}_r)_{i,j} = \tilde{\pi}_j^{(r)} \text{ pro } i, j \in C_r$$

- o \tilde{C}_r má v řádcích stacionární rozdělení C_r :

$$C^n \xrightarrow{n \rightarrow +\infty} \tilde{C} := \begin{matrix} & \begin{matrix} c_1 & c_2 & \dots \end{matrix} \\ \begin{matrix} c_1 \\ c_2 \\ \vdots \end{matrix} & \begin{pmatrix} \tilde{C}_1 & \mathbf{0} & \dots \\ \mathbf{0} & \tilde{C}_2 & \dots \\ \vdots & \vdots & \ddots \end{pmatrix} \end{matrix}$$

- Limita matice P^n – buď $S = T \cup C$ konečná množina stavů, T přechodné, C trvalé aperiodické. Pak

$$\lim_{n \rightarrow +\infty} P^n = \lim_{n \rightarrow +\infty} \begin{matrix} & \begin{matrix} T & C \end{matrix} \\ \begin{matrix} T & C \end{matrix} & \begin{pmatrix} T^n & R_n \\ \mathbf{0} & C^n \end{pmatrix} \end{matrix} = \begin{matrix} & \begin{matrix} T & C \end{matrix} \\ \begin{matrix} T & C \end{matrix} & \begin{pmatrix} \mathbf{0} & U\tilde{C} \\ \mathbf{0} & \tilde{C} \end{pmatrix} \end{matrix}$$

9. Markovské řetězce se spojitým časem. Souvislost s Markovskými řetězci s diskrétním časem a s Poissonovým procesem.

NI-VSM

- **Čítací proces** = náhodný proces $\{N_t \mid t \geq 0\}$, jehož trajektorie jsou nezáporné, celočíselné a neklesající:
 - o $N_t \geq 0$
 - o $N_t \in \mathbb{Z}$
 - o $s \leq t \Rightarrow N_s \leq N_t$
 - o Pro $s < t$ udává přírůstek $N_t - N_s$ počet událostí, které nastaly během časového intervalu
 - o **Binomický proces** = čítací proces takový, že časy mezi událostmi jsou nezávislé a geometricky rozdělené
 - o Spojitý ekvivalent geometrického rozdělení je exponenciální
 - o **Poissonův proces** = čítací proces s nezávislými exponenciálně rozdělenými časy mezi událostmi

Poissonův proces

- Budte $\{X_j \mid j \in \mathbb{N}\}$ iid náhodné veličiny s rozdělením $Exp(\lambda)$. Definujeme $\{T_n \mid n \in \mathbb{N}\}$:

$$T_0 = 0, \quad T_n = T_{n-1} + X_n = \sum_{j=1}^n X_j, \quad n \geq 1$$

pak náhodný proces $\{N_t \mid t \in [0, +\infty)\}$, kde:

$$N_t(\omega) := \max\{n \in \mathbb{N}_0 \mid T_n(\omega) \leq t\}$$

nazveme **Poissonovým procesem**

- Modeluje **příchody událostí**, které jsou na sobě **nezávislé**. Mezi přicházejícími událostmi není žádná interakce
- Řekneme, že proces $\{N_t \mid t \in [0, +\infty)\}$ je **Poissonův proces**, pokud:
 - $N_0 = 0$ skoro jistě
 - $N_t - N_s \sim \text{Poisson}(\lambda(t-s)) \quad \forall t > s \geq 0$
 - $\{N_t\}$ má nezávislé přírůstky: $\forall k \in \mathbb{N}$ a pro všechny $0 \leq t_0 < t_1 < \dots < t_k$

$$N_{t_1} - N_{t_0}, N_{t_2} - N_{t_1}, \dots, N_{t_k} - N_{t_{k-1}} \text{ nezávislé}$$
- **Binomický x Poissonův proces**
 - o Liší se pouze rozdělením času mezi událostmi

Binomický	Poissonův
$X_j \sim \text{Geom}(p)$	$X_j \sim \text{Exp}(\lambda)$
$T = \mathbb{N}_0$	$T = [0, +\infty)$
$Y_n \sim \text{Binom}(n, p)$	$N_t \sim \text{Poisson}(\lambda t)$
$E Y_n = pn$	$E N_t = \lambda t$

- o Geometrické a exponenciální rozdělení mají společnou vlastnost – **bezpečnost**
- o Poisson lze chápat jako spojitou variantu binomického
- o Bezpečnost klíčová pro MŘ se spojitým časem
- **Exponenciální rozdělení $T \sim \text{Exp}(\lambda)$:**

Hustota pravděpodobnosti

$$f_T(t) = \begin{cases} \lambda e^{-\lambda t} & t \geq 0, \\ 0 & t < 0. \end{cases} \quad \text{Funkce přežití}$$

$$P(T > t) = \begin{cases} e^{-\lambda t} & t \geq 0, \\ 1 & t < 0. \end{cases}$$

Distribuční funkce

$$F_T(t) = \begin{cases} 1 - e^{-\lambda t} & t \geq 0, \\ 0 & t < 0. \end{cases} \quad \text{Momenty}$$

$$E T = \frac{1}{\lambda}, \quad \text{var } T = \frac{1}{\lambda^2}, \quad E T^k = \frac{k!}{\lambda^k}$$

- **Bezpečnost exponenciálního rozdělení** – buď $T \sim \text{Exp}(\lambda)$ exponenciálně rozdělená náhodná veličina. Pak $\forall t, s \geq 0$:

$$P(T > t + s \mid T > t) = P(T > s)$$

- o Pokud jsme na bus čekali t minut, pak pst, že budeme čekat dalších s minut je stejná, jako kdybychom vůbec nečekali

- **Silná bezpaměťovost exponenciálního rozdělení** – buď $T \sim \text{Exp}(\lambda)$ a buď A spojitá nezáporná náhodná veličina nezávislá na T . Pak $\forall s \geq 0$:

$$P(T > A + s | T > A) = P(T > s)$$

- o Pokud jsme na bus 143 čekali do příjezdu 180, pak pst, že budeme čekat dalších s minut je stejná, jako kdybychom vůbec nečekali
- Budte X_1, X_2, \dots i.i.d., $X_j \sim \text{Exp}(\lambda)$. Pak $T_n := X_1 + X_2 + \dots + X_n \sim \text{Ga}(\lambda, n)$, tj.

$$f_{T_n}(t) = \frac{\lambda^n}{(n-1)!} e^{-\lambda t} t^{n-1}, \quad t \geq 0$$

- Náhodná veličina N_s má Poissonovo rozdělení s parametrem λ_s
- **Bezpaměťovost** – buď $s \geq 0$ pevné. Pak náhodný proces $\{N_{t+s} - N_s | t \geq 0\}$ je Poissonův proces s intenzitou λ . Navíc $N_{t+s} - N_s$ je nezávislé na N_r pro $0 \leq r \leq s$
 - o Když přijdu k běžícímu Poiss. procesu v čase s , je to stejné, jako by se proces restartoval
→ má **nezávislé přírůstky**

Markovský řetězec se spojitým časem

- **Markovský řetězec se spojitým časem** = náhodný proces $\{X_t | t \geq 0\}$ s nejvýše spočetnou množinou stavů S , který splňuje markovskou podmínku: $\forall k \in \mathbb{N}, \forall 0 \leq t_0 < t_1 < \dots < t_k \in \mathbb{R}_0^+$, a $\forall s_0, \dots, s_k \in S$

$$P(X_{t_k} = s_k | X_{t_{k-1}} = s_{k-1}, \dots, X_{t_0} = s_0) = P(X_{t_k} = s_k | X_{t_{k-1}} = s_{k-1})$$

- Rozdělení v čase $t \in [0, +\infty)$. Pro $i \in S$:

$$p_i(t) := P(X_t = i), \quad p(t) := (p_1(t), p_2(t), \dots)$$

- Matice pravděpodobností přechodu za čas mezi s a $t \leq s$:

$$P_{ij}(t, s) := P(X_s = j | X_t = i), \quad P(t, s) := (P_{ij}(t, s))_{i,j \in S}$$

- Náhodný proces $\{X_t | t \geq 0\}$ s nejvýše spočetnou množinou stavů S je markovský právě tehdy, když $\forall k \in \mathbb{N}, \forall 0 \leq t_0 < t_1 < \dots < t_k \in \mathbb{R}_0^+$, a $\forall s_0, \dots, s_k \in S$

$$P(X_{t_0} = s_0, \dots, X_{t_k} = s_k) = p_{s_0}(t_0) \cdot P_{s_0 s_1}(t_0, t_1) \cdot P_{s_1 s_2}(t_1, t_2) \cdot \dots \cdot P_{s_{k-1} s_k}(t_{k-1}, t_k)$$

- **Chapman-Kolmogorova věta** – pro matice přechodu markovského řetězce platí $\forall t \leq s \leq r \in [0, +\infty)$:

$$P(t, r) = P(t, s) \cdot P(s, r)$$

- **Homogenní markovský řetězec** - $\forall t, s \geq 0$ platí:

$$P(t, t+s) = P(0, s) = P(s)$$

- o **Chapman-Kolmogorova věta** pro homogenní MŘ:

$$P(t+s) = P(t)P(s) = P(s)P(t)$$

- o Rozdělení v $t \geq 0$:

$$p(t) = p(0) \cdot P(t)$$

- **Matice skokových intenzit**

- o Pokud $\forall i, j \in S, i \neq j$ existují konečné limity

$$Q_{ii} = \lim_{h \rightarrow 0^+} \frac{P_{ii}(h) - 1}{h}, \quad Q_{ij} := \lim_{h \rightarrow 0^+} \frac{P_{ij}(h)}{h}$$

nazveme $Q = (Q_{ij})_{i,j \in S}$ maticí (skokových) intenzit

$$P(0) = I \Rightarrow Q := \lim_{h \rightarrow 0^+} \frac{1}{h} (P(h) - I) = \lim_{h \rightarrow 0^+} \frac{1}{h} (P(h) - P(0))$$

- Předpokládáme, že limity existují a jsou konečné. Pak:

$$Q = \lim_{h \rightarrow 0^+} \frac{1}{h} (P(h) - P(0)) = \left. \frac{d}{dt} P(t) \right|_{t=0} = P'(0)$$

- o Vlastnosti matice skokových intenzit:

- Neboť $0 \leq P_{ij}(t) \leq 1$, platí pro $i \neq j$:

- Neboť $P_{ii}(t) = 1 - \sum_{j \neq i} P_{ij}(t)$, platí:

$$Q_{ii} = \lim_{h \rightarrow 0^+} \frac{1}{h} \left(1 - \sum_{j \neq i} P_{ij}(h) - 1 \right) = - \sum_{j \neq i} \lim_{h \rightarrow 0^+} \frac{P_{ij}(h)}{h} = - \sum_{j \neq i} Q_{ij}$$

- Skok z i do j : $P_{ij}(h) = Q_{ij} \cdot h + o(h)$
- Skok z i pryč: $1 - P_{ii}(h) = -Q_{ii} \cdot h + o(h)$
- Simulace procesu pomocí skokových intenzit
 - Buď τ_i čas do výskoku ze stavu i , tj. $X_{\tau_i} \neq i, X_s = i$ pro $s \in [0, \tau_i)$. Pak:
 - i. Čas do výskoku z i je exponenciální s $\lambda_i = -Q_{ii}$
 $= \tau_i \sim \text{Exp}(-Q_{ii})$
 - ii. Pravděpodobnost, že řetězec skočí z i do j je dána poloměrem intenzit Q_{ij} , tj.

$$P(X_{\tau_i} = j \mid X_0 = i) = \frac{Q_{ij}}{\lambda_i} = \frac{Q_{ij}}{-Q_{ii}} = \frac{Q_{ij}}{\sum_{k \neq i} Q_{ik}}$$
- Princip simulace:
 1. Začínám v $i \in S$
 2. Generuji náhodný čas $\tau_i \sim \text{Exp}(-Q_{ii})$ a „posunu“ hodiny o τ_i
 3. Změním stav z i na j s pravděpodobností $\frac{Q_{ij}}{-Q_{ii}}$
 4. Opakuji od 2.

Konstrukce řetězců se spojitým časem

- Homogenní MŘ se spojitým časem (shrnutí)
 - Markovská podmínka

$$P(X_{t_k} = s_k \mid X_{t_{k-1}} = s_{k-1}, \dots, X_{t_0} = s_0) = P(X_{t_k} = s_k \mid X_{t_{k-1}} = s_{k-1})$$
 - Homogenita

$$P(t, t+s) = P(0, s) := P(s)$$
 - Chapman-Kolmogorova rovnice

$$P(s+t) = P(s)P(t) = P(t)P(s)$$
 - Matice skokových intenzit

$$Q = P'(0) = \lim_{h \rightarrow 0_+} \frac{P(h) - I}{h}$$
 - Vlastnosti matice Q

$$Q_{ij} \geq 0, i \neq j, \quad Q_{ii} = -\sum_{k \neq i} Q_{ik} \leq 0$$
- Diskrétní čas časovaný Poissonovým procesem
 - Buď $\{N_t \mid t \geq 0\}$ Poissonovský proces s intenzitou λ . Buď $\{Y_n \mid n \in \mathbb{N}_0\}$ homogenní MŘ s diskrétním časem mající matici přechodu D . Pak proces $\{X_t \mid t \geq 0\}$ definovaný jako

$$X_t := Y_{N_t}, \quad \text{tj.} \quad X_t(\omega) := Y_{N_t(\omega)}(\omega)$$
 je homogenní markovský proces se spojitým časem
 - Dynamika procesu $\{X_t\}$
 - $X_0 = i$
 - Bez ohledu na minulost má čas do další události rozdělení $\text{Exp}(\lambda)$
 - Po uplynutí tohoto času přeskóčí řetězec z i do j s pravděpodobností D_{ij}
 - Čas do další události je opět exponenciální a nezávislý na minulosti
 - Matice přechodu
 - Pravděpodobnost, že za čas t nastane právě n událostí je $P(N_t = n)$
 - Za podmínky, že nastalo n událostí, je pst přeskoku z i do j dána n -krokovou pravděpodobností přechodu řetězce $\{Y_n\}$:

$$P(X_t = j \mid X_0 = i, N_t = n) = (D^n)_{ij}$$
 - Navíc platí:

$$P(X_t = j \mid X_0 = i) = \sum_{n=0}^{+\infty} P(N_t = n) P(X_t = j \mid X_0 = i, N_t = n)$$
 - Matice přechodu řetězce $X_t = Y_{N_t}$ mají tvar:

$$P_{ij}(t) = \sum_{n=0}^{+\infty} \frac{(\lambda t)^n}{n!} e^{-\lambda t} (D^n)_{ij}$$

- Prvky matice přechodu pro $i \neq j$ a $h \rightarrow 0_+$:

$$P_{ij}(h) = \lambda h(D_{ij}) + o(h)$$

- Prvky Q pro $i \neq j$:

$$Q_{ij} = \lambda D_{ij}$$

- Prvky Q pro $i = j$:

$$Q_{ii} = \lambda(D_{ii} - 1)$$

- Konstrukce diskrétního řetězce pomocí Q :

- $Q = \lambda(D - I) \Rightarrow D = I + \frac{1}{\lambda}Q$
- Aby byla D stochastická, musí být $\lambda \geq Q_{ij} \Rightarrow \sup_{i \in S} (-Q_{ii}) < +\infty$
 - Vždy splněno, je-li S konečná

- Simulace pomocí diskrétního řetězce – mám Q a chci simulovat trajektorii $\{X_t\}$

1. $\lambda = \sup_{i \in S} (-Q_{ii})$
2. Matice přechodu: $D = I + \frac{1}{\lambda}Q$
3. Vygeneruji trajektorii Poissonova procesu $\{N_t\}$
4. Vygeneruji trajektorii $\{Y_n\}$ pomocí matice D
5. Trajektorie $\{X_t\}$: $X_t = Y_N$

Kolmogorovy rovnice

- Výpočet matic přechodu pomocí matice intenzit

- MŘ se spojitým časem definovaná pomocí matice intenzit přeskočků Q_{ij}
- Chceme znát vývoj rozdělení v čase $t \geq 0$:

$$p(t) = p(0) \cdot P(t)$$

- K tomu potřebujeme matice přechodu $P(t)$
- $P(t)$ je řešením soustavy diferenciálních rovnic:

$$\mathbf{P}'_{ij}(t) = F_{ij}(\mathbf{P}_{kl}(t); k, \ell \in S), \quad i, j \in S$$

- Maticově:

$$\mathbf{P}'(t) := (\mathbf{P}'_{ij}(t))_{ij \in S}, \quad \mathbf{P}'(t) = \mathbf{F}(\mathbf{P}(t))$$

- Kolmogorovy rovnice – buď $\{X_t \mid t \geq 0\}$ markovský řetězec s maticí intenzit Q . Pak pro matice přechodu $P(t)$ platí:

- Kolmogorova dopředná rovnice: $P'(t) = P(t) \cdot Q$
- Kolmogorova zpětná rovnice: $P'(t) = Q \cdot P(t)$

- Rozdělení $p(t)$ je řešením soustavy diferenciálních rovnic

$$p'(t) = p(t) \cdot Q, \quad p(0) = p_{\text{initial}}$$

- Po složkách:

$$\underbrace{p'_i(t)}_{\text{změna psti } i} = \sum_{j \in S} p_j(t) Q_{ji} = \underbrace{\sum_{j \neq i} p_j(t) Q_{ji}}_{\text{zisk psti } i} - \underbrace{\sum_{k \neq i} p_i(t) Q_{ik}}_{\text{ztráta psti } i}$$



- Řešení Kolmogorových rovnic – pro matice přechodu platí $P(t) = e^{Qt}$, kde

$$e^{Qt} := \sum_{n=0}^{+\infty} \frac{(Qt)^n}{n!}$$

Stacionární rozdělení

- Buď $\{X_t \mid t \geq 0\}$ MŘ s pravděpodobnostmi přechodu $P(t)$. Pak vektor π nazvu **stacionárním rozdělením**, pokud $\forall t \geq 0$:

$$\pi \cdot P(t) = \pi$$

- Vektor π je **stacionárním rozdělením**, právě když $\pi \cdot Q = 0$

- **Limitní vlastnosti**

- o MŘ $\{X_t \mid t \geq 0\}$ je **nerozložitelný (irreducibilní)**, pokud se z každého stavu $i \in S$ mohu dostat do libovolného stavu $j \in S$ pomocí konečně mnoha přeskoků
= pokud existuje $n \in \mathbb{N}$ a stavy $i = s_0, \dots, s_n = j \in S$ takové, že $Q_{s_k, s_{k+1}} > 0$ pro $k = 0, \dots, n-1$
- o Buď $\{X_t \mid t \geq 0\}$ **nerozložitelný MŘ se spojitým časem**:
 - i. Existuje-li stacionární rozdělení π , pak je jednoznačné a $\forall i, j \in S$:

$$\lim_{t \rightarrow +\infty} P_{ij}(t) = \pi_j$$

- ii. Pokud stacionární rozdělení neexistuje, pak $\forall i, j \in S$:

$$\lim_{t \rightarrow +\infty} P_{ij}(t) = 0$$

- Je-li množina stavů S **konečná**, pak stacionární rozdělení markovského řetězce se spojitým časem existuje

- **Detailní rovnováha**

- o Stacionární rozdělení splňuje

$$0 = \frac{d}{dt} \pi_i = \sum_{j \in S} \pi_j Q_{ji} = \sum_{j \neq i} \pi_j Q_{ji} - \sum_{k \neq i} \pi_i Q_{ik}$$

- o Lze přepsat na:

$$0 = \sum_{j \neq i} (\pi_j Q_{ji} - \pi_i Q_{ij})$$

- o Pokud rozdělení π splňuje detailní rovnováhu:

$$\pi_j Q_{ji} = \pi_i Q_{ij}$$

pak je stacionárním rozdělením

10. Systémy hromadné obsluhy a jejich limitní vlastnosti. Souvislost s Markovskými řetězci se spojitým časem.

NI-VSM

Exponenciální závody a Markovské řetězce

- Model hromadné obsluhy



o Princip:

- Požadavky na server **přichází** náhodně s **intenzitou λ**
- Je-li server zaneprázdněn vyřizováním požadavku, zařadí se nový do **fronty**
- Server **vyřizuje** požadavky s **intenzitou μ**

- Rozdělení minima – budte $T \sim \text{Exp}(\lambda)$ a $S \sim \text{Exp}(\mu)$ nezávislé. Pak

$$Z := \min\{T, S\} \sim \text{Exp}(\lambda + \mu)$$

→ bud' $T_1 \dots T_n$ nezávislé veličiny, $T_j \sim \text{Exp}(\lambda)$. Pak

$$\min\{T_1, \dots, T_n\} \sim \text{Exp}(\lambda_1 + \dots + \lambda_n)$$

o Z toho plyne, že:

$$E \min\{T, S\} = \frac{1}{\mu + \lambda}, \quad E \min\{T_1, \dots, T_n\} = \frac{1}{\lambda_1 + \dots + \lambda_n}$$

o Platí, že $\max\{T, S\} = T + S - \min\{T, S\}$. Potom

$$E \max\{T, S\} = ET + ES - E \min\{T, S\} = \frac{1}{\lambda} + \frac{1}{\mu} - \frac{1}{\lambda + \mu}$$

o Pro libovolné nezávislé náhodné veličiny X, Y platí

$$\{\max\{X, Y\} \leq t\} = \{X \leq t\} \cap \{Y \leq t\}$$

$$\{\min\{X, Y\} \leq t\} = \{X \leq t\} \cup \{Y \leq t\}$$

- Vítěz závodů – budte $T \sim \text{Exp}(\lambda)$ a $S \sim \text{Exp}(\mu)$ nezávislé. Pak

$$P(T < S) = \frac{\lambda}{\lambda + \mu}, \quad P(S < T) = \frac{\mu}{\lambda + \mu}$$

→ bud' $T_1 \dots T_n$ nezávislé veličiny, $T_j \sim \text{Exp}(\lambda)$. Pak

$$P(T_i = \min\{T_1, \dots, T_n\}) = \frac{\lambda_i}{\lambda_1 + \dots + \lambda_n}$$

- Konstrukce matice intenzit

i. Přeskok o $k \geq 2$: $Q_{n, n+k} = 0$

ii. Přeskok o 1: $Q_{n, n+1} = \lambda$

$$Q_{n, n-1} = \mu$$

iii. Setrvání: $Q_{n, n} = -(\lambda + \mu)$

- Proces $\{X_t \mid t \geq 0\}$ je MŘ se spojitým časem \Leftrightarrow mezi jednotlivými stavy probíhají **exponenciální závody**

1. Začínám v $i \in S$

2. Generuji náhodný čas $\tau_i \sim \text{Exp}(\lambda_i)$, kde $\lambda_i = \sum_{k \neq i} \lambda_{ik}$ a "posunu hodiny" o τ_i

3. Změním stav z i na j s pravděpodobností $\frac{\lambda_{ij}}{\lambda_i}$

4. Opakuji od 2.

o Pak se jedná o markovský řetězec s maticí intenzit Q takovou, že:

$$Q_{ij} = \lambda_{ij}, \quad i \neq j, \quad Q_{ii} = -\lambda_i$$

- Řetězec se spojitým časem lze popsat **diagramem** – váhy hran určeny intenzitou přeskoku $Q_{ij} = \lambda_{ij}$ mezi jednotlivými stavy

Systémy hromadné obsluhy

- Model hromadné obsluhy

- λ [zákazníků za časovou jednotku] ... **intenzita příchodů**
- A_i ... náhodný čas mezi příchozem $(i - 1)$ -ního a i -tého zákazníka,

$$A_i \sim F_A, EA_i = \frac{1}{\lambda}$$

- μ [zákazníků za časovou jednotku] ... **intenzita obsluhy 1 serveru**
- S_j ... čas obsluhy j -tého zákazníka

$$S_j \sim F_S, ES_j = \frac{1}{\mu}$$

- Veličiny $A_1, A_2, \dots, S_1, S_2, \dots$ jsou nezávislé
- Server obsahuje c nezávislých obslužných míst

- Proces hromadné obsluhy $X = \{X_t \mid t \geq 0\}$ = proces, který zaznamenává počet zákazníků v systému hromadné obsluhy (= serveru a frontě) v čase t

- Konečněrozměrná rozdělení procesu jsou jednoznačně určena rozděleními F_A a F_S
- Intenzita příchodů je λ
- Intenzita obsluhy zákazníků je nejvýše $c\mu$

$$\rho = \frac{\lambda}{c\mu}$$

- $\rho > 1 \rightarrow$ počet zákazníků v systému poroste nad všechny meze
- $\rho < 1 \rightarrow$ systém se ustálí na stabilním rovnovážném rozdělení

- Kendallova notace $A \mid S \mid c \mid K \mid N \mid D$

- A ... rozdělení časů příchodu F_A
- S ... rozdělení časů obsluhy F_S
- c ... počet obslužných míst
- K ... kapacita systému (neuvedeno = $+\infty$)
- N ... velikost populace (neuvedeno = $+\infty$)
- D ... typ obsluhy (neuvedeno = FIFO)

- Rozdělení A, S jsou značena:

- $M, M(\lambda)$ – **exponenciální** rozdělení (markovské)
- $D, D(d)$ – **degenerované rozdělení** soustředěné v hodnotě d
- G – **obecné rozdělení**, neznámé nebo známé „neexponenciální“

- Systém $M \mid M \mid 1$

- Proces zrodu a zániku s parametry:

$$\lambda_n = \lambda, \quad \mu_m = \mu$$

- Matice intenzit:

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & \dots \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ \vdots \end{matrix} & \begin{pmatrix} -\lambda & \lambda & 0 & 0 & \dots \\ \mu & -(\lambda + \mu) & \lambda & 0 & \dots \\ 0 & \mu & -(\lambda + \mu) & \lambda & \dots \\ 0 & 0 & \ddots & \ddots & \ddots \end{pmatrix} \end{matrix}$$

- Pokud $\lambda < \mu$, pak existuje **stacionární rozdělení** ve tvaru:

$$\pi_n = \left(1 - \frac{\lambda}{\mu}\right) \left(\frac{\lambda}{\mu}\right)^n$$

$$P(X_t = n) \rightarrow \pi_n = (1 - \rho)\rho^n$$

- Pokud $\rho \geq 1$, stacionární rozdělení neexistuje a $P(X_t = n) \rightarrow 0$

- Stacionární vlastnosti $M \mid M \mid 1$

- **Střední počet** zákazníků v systému: $EN = E_{\pi}X_t = \frac{\rho}{1-\rho}$
- **Střední počet** zákazníků na serveru: $EN_S = 1 - \pi_0 = \rho$
- **Střední počet** zákazníků ve frontě: $EN_f = EN - EN_S = \frac{\rho^2}{1-\rho}$

- Doba W čekání zákazníka ve frontě:

- $P(W = 0) = \pi_0 = 1 - \rho \rightarrow P(W > 0) = 1 - \pi_0 = \rho$
- $P(W > S) = \rho e^{-(\mu-\lambda)S} \rightarrow (W \mid W > 0) \sim \text{Exp}(\mu - \lambda)$

- Systémy $M | M | \infty$

- o Nekonečno obslužných míst – každý zákazník **okamžitě obsluhován**
- o Proces zrodu a zániku s parametry:

$$Q_{n,n+1} = \lambda_n \equiv \lambda, \quad Q_{n,n-1} = \mu_n = n \cdot \mu$$

- o Matice intenzit:

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & \dots \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ \vdots \end{matrix} & \begin{pmatrix} -\lambda & \lambda & 0 & 0 & \dots \\ \mu & -(\lambda + \mu) & \lambda & 0 & \dots \\ 0 & 2\mu & -(\lambda + 2\mu) & \lambda & \dots \\ 0 & 0 & \ddots & \ddots & \ddots \end{pmatrix} \end{matrix}$$

- o **Stacionární rozdělení** vždy existuje a je Poissonovo rozdělení s parametrem $\frac{\lambda}{\mu}$, takže pro $n \in \mathbb{N}_0$:

$$P_{\pi}(X_t = n) = \pi_n = \frac{1}{n!} \left(\frac{\lambda}{\mu} \right)^n e^{-\frac{\lambda}{\mu}}.$$

- Systém $M | M | c$:

- o $1 < c < +\infty$ nezávislých obslužných míst \rightarrow existuje **fronta**
- o Proces zrodu a zániku s intenzitami:

$$Q_{n,n+1} = \lambda_n \equiv \lambda, \quad Q_{n,n-1} = \mu_n = \min\{c, n\} \cdot \mu = \begin{cases} n \cdot \mu & n \leq c, \\ c \cdot \mu & n > c. \end{cases}$$

- o Matice intenzit:

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & \dots & c+1 & c+2 & \dots \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ \vdots \\ c \\ c+1 \\ \vdots \end{matrix} & \begin{pmatrix} -\lambda & \lambda & 0 & 0 & \dots & 0 & 0 & \dots \\ \mu & -(\lambda + \mu) & \lambda & 0 & \dots & 0 & 0 & \dots \\ 0 & 2\mu & -(\lambda + 2\mu) & \lambda & \dots & 0 & 0 & \dots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots \\ 0 & 0 & 0 & c\mu & -(\lambda + c\mu) & \lambda & 0 & \dots \\ 0 & 0 & 0 & 0 & c\mu & -(\lambda + c\mu) & \lambda & \dots \\ 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots \end{pmatrix} \end{matrix}$$

- o **Stacionární rozdělení** existuje pro $\rho < 1$:

$$\pi_n = \begin{cases} \frac{1}{n!} \left(\frac{\lambda}{\mu} \right)^n \pi_0 & n \leq c, \\ \frac{c^c}{c!} \left(\frac{\lambda}{c\mu} \right)^{n-c} \pi_0 & n > c. \end{cases}$$

- **Littleho věta** – buď $\{X_t \mid t \geq 0\}$ striktně stacionární proces hromadné obsluhy. Budte dále:

- EN ... střední **počet zákazníků** v systému
- ET ... střední **doba** strávená zákazníkem v systému
- λ ... intenzita procesu příchodů

Jsou-li všechny střední hodnoty konečné, pak $EN = \lambda \cdot ET$

- Systém $G | G | 1$

- o **Obecné rozdělení** příchodů i obsluhy (při zachování stacionarity)
- o λ ... intenzita příchodů
- o μ ... intenzita obsluhy

$$\text{Střední doba obsluhy} = ES = \frac{1}{\mu}$$

- o $T_k = W_k + S_k$... **doba strávená k -tým zákazníkem** v systému
 - W_k ... doba čekání ve frontě
 - S_k ... doba obsluhy

- o **Littleho věta** pro:

- Celý **systém**: $EN = \lambda ET_k = \lambda EW_k + \lambda ES_k$
- Samotnou **frontu**: $EN_f = \lambda EW_k$

- o Z toho vyplývá: $\pi_0 = 1 - \frac{\lambda}{\mu}$

Nehomogenní Poissonův proces

- Buď $\lambda(r)$ funkce integrabilní na libovolném konečném intervalu, pak proces $\{N_t \mid t \geq 0\}$ nazvu nehomogenním Poissonovým procesem s intenzitou $\lambda(r)$, pokud
 - i. $N_0 = 0$ skoro jistě
 - ii. $\{N_t\}$ má nezávislé přírůstky
 - iii. Pro $t > s$:

$$N_t - N_s \sim \text{Poisson} \left(\int_s^t \lambda(r) dr \right)$$

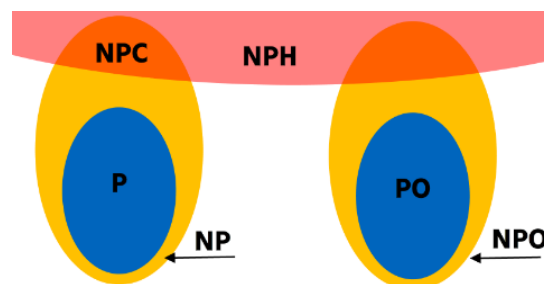
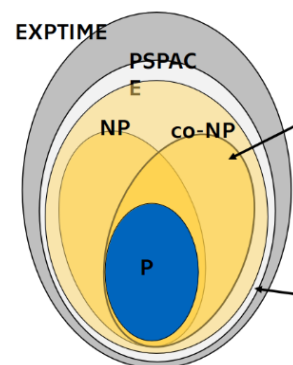
- Modelujeme situaci, kdy se **intenzita příchodu** zákazníka **mění s časem**: $\lambda = \lambda(t)$
- **Systém $M \mid G \mid \infty$**
 - o M ... Poissonovské příchody s intenzitou λ
 - o G ... **obecné rozdělení** časů obsluhy se střední hodnotou $ES = \frac{1}{\mu}$ a distribuční funkci G
 - o ∞ ... neomezený počet serverů (**žádná fronta**)
 - o Systém v $t = 0$ prázdný \rightarrow požadavek, který přišel v čase s je v čase $t > s$ **stále v systému s pravděpodobností**:
$$P(\text{doba obsluhy} > t - s) = 1 - G(t - s)$$
 - Z pohledu t tedy událost, která přišla v čase $s < t$ **přijmu s pravděpodobností**
$$p(s) = 1 - G(t - s)$$
 - Poissonovo rozdělení s parametrem $\lambda \cdot ES$
 - o Počet zákazníků v systému má z dlouhodobého pohledu ($t \rightarrow +\infty$) Poissonovo rozdělení s parametrem $\frac{\lambda}{\mu} = \lambda \cdot \frac{1}{\mu} = \lambda \cdot ES$
 - o **Poissonovo rozdělení**:
$$P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}, \quad k = 0, 1, 2, \dots, \quad EX = \text{var } X = \lambda$$
 - Pravděpodobnost, že v systému bude k požadavků z dlouhodobého hlediska

11. Význam tříd NP a NPH pro praktické výpočty.

NI-KOP

Kombinatorický problém

- **Kombinatorický problém** – charakterizují ho:
 - o Vstupní proměnné – seznam prvků
 - o Výstupní proměnné – pořadí prvků (popis výstupu)
 - Hodnota, kterou chceme optimalizovat – například u TSP délka cesty, kterou minimalizujeme
 - o Konfigurační proměnné – pořadí prvků (popis konfigurace)
 - Proměnná, kterou můžeme měnit v rámci řešení problému – např. u TSP pořadí měst
 - o Omezení – každý prvek právě jednou
 - o Optimalizační kritérium – nejmenší
- **Proměnné** – konečný počet, konečné domény
- **Instance a řešení problému**
 - o **Instance I** – ohodnocení vstupních proměnných
 - o **Konfigurace Y** – ohodnocení konfiguračních proměnných (během řešení instance)
 - o **Řešení instance** – ohodnocení konfiguračních proměnných splňujících omezení
 - Omezení $R(I, Y)$ říká, jestli je Y řešením instance I
 - o **Optimální řešení** – řešení s nejlepší hodnotou optimal. Kritéria
 - o **Suboptimální řešení** – řešení s vyhovující hodnotou opt. Kritéria
- Typy problémů
 - o **Rozhodovací problém**: Existuje řešení? Nebo platí omezení pro všechny konfigurace?
 - o **Konstruktivní problém**: Sestrojte řešení (konfiguraci) takovou, aby platila omezení
 - o **Enumerační problém**: Sestrojte všechna řešení, pro které platí omezení
 - o Všechny typy problémů lze převést na optimalizační verze
 - o Měření složitosti pomocí asymptotické složitosti
- **Turingův stroj** – původně pro rozhodnutelnost problému
 - o Neomezená páska, čtecí hlavice, řídící jednotka (stav)
 - o Zastavení = přechod do koncového stavu
 - o Q – konečná neprázdná množina stavů
 - o Σ – konečná vstupní abeceda
 - o G – konečná neprázdná pracovní abeceda
 - o δ – přechodová funkce
 - o q_0 – počáteční stav
 - o B – prázdný symbol
 - o F – množina koncových stavů



Třídy složitosti

- **Třída P** – rozhodovací problém patří do třídy P, jestliže pro něj existuje program pro deterministický Turingův stroj, který jej řeší v čase $O(n^k)$, kde n je velikost instance a k konečné číslo
- **PSPACE** – existuje program pro deterministický Turingův stroj, který jej řeší v paměti $O(n^k)$, kde n je velikost instance a k konečné číslo
- **EXPTIME** – existuje program pro deterministický Turingův stroj, který jej řeší v čase $O(2^{P(n)})$, kde $P(n)$ je polynom ve velikosti instance n
- **NP** – problém patří do NP, pokud existuje program pro NTS, který každou ANO-instanci řeší v polynomiálním čase $O(n^k)$, kde n je velikost instance a k konečné číslo
 - o NP
 - ANO – ověření probíhá jako existence odpovědi – certifikát v P
 - NE – musíme zkontrolovat všechny možnosti
 - o **co-NP** – doplněk k NP problémům

- NE – ověření jako existence 1 odpovědi (ne, není pravda, že nemůžeme najít řešení – certifikát v P)
 - ANO – ano, nemůžeme najít řešení – potřeba kontroly všech
- **NPH** – problém Π je NPH, pokud se efektivní (polynomiální) řešení všech problémů v NP dá zredukovat na řešení Π
 - Nejtěžší problém je takový, na který se dají převést všechny ostatní problémy
 - Je nejméně tak těžký, jako problémy, jež na něj byly převedeny
 - **NPC** – pokud Π je zároveň NPH a NP
 - **Cookova věta** – všechny NP problémy lze převést na SAT
- **NPO** – optimalizační problém patří do třídy NPO, jestliže splňuje podmínky:
 - Velikost výstupu instance omezena polynomem ve velikosti instance
 - Problém, zda daná konfigurace je řešením, patří do P
 - Existuje program pro TS, který vypočítá hodnotu optimalizačního kritéria pro každé řešení každé instance v polynomiálním čase
- Problémy různých složitostí:
 - P – součet dvou čísel
 - NP – faktorizace čísel
 - NPC – SAT
 - NPH – halting problem
- **Redukce problémů**
 - **Karpova redukce** – problém Π_1 je karp-redukovatelný na Π_2 , pokud existuje polynomiální algoritmus na DTS, který provede každou instanci I_1 problému Π_1 na instanci I_2 problému Π_2 tak, že jejich výstup je shodný
 - Platí tranzitivita redukovatelnosti problémů
 - **Turingova redukce** – problém Π_1 je turing-redukovatelný na Π_2 , pokud existuje algoritmus na DTS, který řeší každou instanci I_1 problému Π_1 tak, že používá program M_2 pro problém Π_2 jako podprogram
 - Karpova redukce je speciální případ

Význam pro praktické výpočty

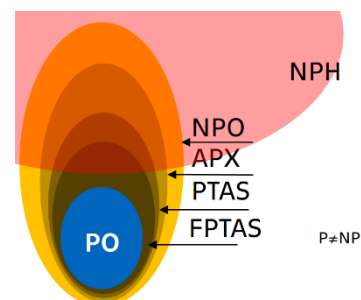
- NP a NPH problémy nelze efektivně řešit – řešení je exponenciální na DTS (normálním počítači), které pro větší instance může trvat neakceptovatelně dlouho - $O(n^k)$ – využití v kryptografii, hashování
- Neznámé problémy můžeme **redukcí převádět na známé problémy** a řešit je dle zvyku
 - Např. převod na SAT problém splnitelnosti
- V některých případech nepotřebujeme přesné řešení, stačí **přibližné/suboptimální**
- **Zrychlení řešení NPO úloh**
 - **Metoda redukce stavového prostoru** – např. branch and bound – nepoužitelné výsledky ignorovány
 - **Pseudopolynomiální algoritmy** – počet kroků algoritmu závisí na velikosti instance, ale závisí dále na parametru, který s velikostí nemá nic společného - $O(nM)$
 - **Aproximace** – FPTAS, PTAS, APX – jednodušší heuristika pro přibližné řešení – spokojenost s určitou hladinou chybovosti algoritmu
 - **Randomizace** – algoritmy založené na náhodné volbě
- **Aproximativní algoritmy**
 - $C(S)$... hodnota optimalizačního kritéria řešení S
 - $APR(I)$... aproximované řešení instance I
 - $OPT(I)$... optimální řešení instance I
 - **Relativní chyba:**

$$\varepsilon \geq \max_{\forall I} \left\{ \frac{|C(APR(I)) - C(OPT(I))|}{\max\{C(OPT(I)), C(APR(I))\}} \right\}$$

- **Relativní kvalita:**

$$R \geq \max_{\forall I} \left\{ \frac{C(APR(I))}{C(OPT(I))}, \frac{C(OPT(I))}{C(APR(I))} \right\}$$

- Chyba a kvalita jsou spolu spojené: $\varepsilon = 1 - 1/R$



- **APR** – algoritmus APR pro problém Π je R -aproximativní (ε -aproximativní), jestliže každou instanci Π vyřeší v polynomiálním čase s relativní kvalitou $R(\varepsilon)$
 - $R(\varepsilon)$ je aproximační prah problému
- **PTAS** – algoritmus APR, který pro každé $1 > \varepsilon > 0$ vyřeší každou instanci I problému Π s relativní chybou nejvýše ε v čase polynomiálním v $|I|$ nazýváme polynomiální aproximační schéma problému Π
- **FPTAS** – polynomiální aproximační schéma APR, jehož čas výpočtu závisí polynomiálně na $1/\varepsilon$ nazýváme plně polynomiální aproximační schéma
- **Nejtypičtější NPH úlohy**
 - SAT, 3SAT – problém splnitelnosti booleovské formule
 - Problém batohu, obchodní cestující, hamiltonovská kružnice
 - Existence zadání
 - Offline – mám všechna data
 - Online – postupně dostávám údaje a upravuji řešení
 - Některé úlohy lze řešit ne lokálními, ale globálními metodami (dekompozice, dynam. prog., ...)

12. Experimentální vyhodnocení algoritmů, zejména randomizovaných.

NI-KOP

Experimentální vyhodnocení

- **Analytické odpovědi o algoritmech**
 - o Nejhorší případ, asymptotické meze
 - Významná oblast může být mimo zájem
 - Konstanty jsou nepraktické
 - Nejhorší případ je vzácný a nezajímavý, analýza je příliš složitá
 - o Průměrný případ
 - Analýza je proveditelná pro jednoduché případy
 - Analýza závisí na statistických vlastnostech vstupu, které nemusí být známy
- Co nás zajímá
 - o **Složitost** (teoretická x z hlediska nasazení)
 - o **Kvalita řešení**
 - o **Porozumění** – proč to blbne na určitých instancích
 - Závislost něčeho na něčem (výpočetní čas na velikosti instance, kvalita řešení na param.)
- **Experiment:**

co chci zjistit → plán experimentu → provedení experimentu → sběr dat → interpretace výsledku → odpověď

- **Metriky** – metrika vstupu → závislost → metrika výstupu → interpretace
 - o Abstrahujeme vlastnosti vstupu a výstupu do kvantitativních veličin = metrik
 - o Zkoumáme závislost mezi metrikami vstupu a výstupu
 - o Interpretace – zobecnění podle jednotlivých chodů
- **Metriky výstupu – složitost**
 - o Metriky vypovídající o algoritmu (např. počet zkoumaných konfigurací)
 - o Metriky vypovídající o implementaci (např. čas CPU)
 - o Chceme metriku, která nezávisí na detailech implementace (v praxi náročné vyhodnocení optimalizačního kritéria)
- **Metriky vstupu**
 - o Ostatní metriky vstupu – známe a udržíme konstantní při generování, nebo vůbec o nich nevíme
 - Tlak, teplota okolí, ... - snažíme se udržet konstantní
- **Příklad metrik a závislostí (3-SAT)**
 - o Závislost počtu kroků procedury DP (backtracking, výstupní metrika) na poměru počtu klauzulí k počtu proměnných (vstupní metrika)
 - o Fázový přechod – pst splnitelnosti v závislosti na poměru počtu klauzulí k počtu proměnných
- **Neznámé metriky**
 - o **Generování instancí** – pro každou hodnotu nastavované metriky vygenerujeme náhodné instance tak, aby každá instance se zadanou metrikou byla stejně pravděpodobná
 - o Sběr instancí – seberu co nejvíce instancí, které má nasazený algoritmus zpracovávat
 - o Více instancí je zdroj variance
 - o Pro každou hodnotu zadané metriky provádím měření na více instancích
 - o Statistické zpracování umožní potlačit varianci
- **Statistika pro jednu hodnotu zadané metriky**
 - o Nejoblíbenější – průměr, medián
 - o Pochybnější – kontrola **statistického rozložení** – nemusí být uniformní ani normální
 - o Parametry rozložení (střední hodnota, variance, ...): komprese získaných dat
 - o Příprava pro důležitou kvalitativní interpretaci – identifikace a charakterizace statistického rozložení (proč je to mix dvou rozdělení? Proč jsou některé instance o tolik těžší?)
- **Srovnání dvou algoritmů A, B**
 - o Na základě parametrů rozložení:
 - A má lepší všechny parametry → A je lepší
 - Jinak nevíme

- **Na základě dominance:**
 - Pro každou instanci (zadané metriky) je A lepší nebo stejně dobrý → A dominuje
- Nevíme → hlubší analýza, kde a proč je A nebo B lepší
- **Odvozené metriky**
 - **Primární metriky** – přímo měřené hodnoty
 - **Vizualizace, vzhled** – histogramy apod.
 - **Kvantitativní srovnání – sekundární metriky:**
 - Průměr, medián
 - Ad hoc kombinace primárních metrik (pokuty za nevyřešení)
 - Úplné charakteristiky statistického rozložení
 - Distribuční funkce, korigované distribuční funkce
 - Křížení distribučních funkcí
- **Randomizovaný algoritmus**
 - **Vyhodnocení na jedné instanci** – jedna instance → algoritmus → statistika, interpretace
 - **Srovnání na jedné instanci**
 - Počet kroků algoritmu jako náhodná proměnná
 - Histogramy úspěšných běhů
 - **Distribuční funkce** – Estimated Cumulative Distribution Function – ECDF
 - Pro každý krok pravděpodobnost, že algoritmus skončil nejvýše v tomto kroku
 - Počítá se z úspěšných běhů – jinak je na konci skok
 - Korekce na úspěšnost – jak do CDF promítnout, že algoritmus v nějakém počtu případů neuspěl
 - Pro každý krok pst, že algoritmus **úspěšně** skončil nejvýše v tomto kroku
 - Stačí přenásobit CDF úspěšností
 - **Vyhodnocení na sadě instancí**
 - Generátor instancí – všechny instance se zadanou metrikou jsou stejně pravděpodobné
 - Algoritmus – všechny hodnoty generátoru náhodných čísel jsou stejně pravděpodobné
 - Statistika 1 – potlačení variance z randomizace
 - Statistika 2 – potlačení variance v instancích
 - Interpretace
 - **Dva zdroje variance** – dva stupně zpracování
 - Protože statistické rozložení vůči char. vstupní instance nemusí být stejné jako rozložení vůči RNG
 - Vlastně statistika ze statistik
 - Přes hodnoty RNG – pro každou instanci tolik spuštění algoritmu, až dostaneme spolehlivá data
 - Přes instance – jako bez randomizace
- **Metriky 2. fáze**
 - Potlačení variance od instancí
 - Statistika ze statistik
 - Co je důležité pro odpověď z experimentu – co vypovídá nejvíc
 - Charakteristiky rozložení
 - Dominance obecně a dominance v zajímavých oblastech – např. malé instance nebo krátké časy nejsou důležité
 - Úspěšnost do určitého počtu kroků, atd.
- **Robustnost heuristiky**
 - Zpřeházení pořadí proměnných ve formuli (stejná booleova funkce) a SAT solver dá jiné řešení v jiném čase
 - Zpřeházením pořadí deklarací proměnných v programu a ten funguje rychleji
 - Závislost práce heuristiky na popisu instance (na rozdíl od instance samotné)
 - Např. lexikální uspořádání dat, které nemění význam
 - Lexikální uspořádání může být důsledkem
 - Reprezentace množiny vektorem
 - Výběru první konfigurace z množiny stejně dobrých

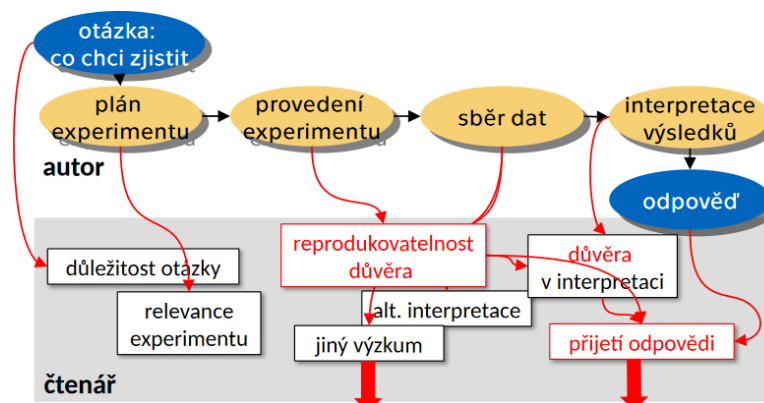
- Náhodný výběr místo postupného provádění → randomizace
- **Měření robustnosti**
 - Před algoritmem – náhodná perturbace (všechny jsou stejně pravděpodobné)
 - Ve statistice měření variance vyvolané perturbací

- Interpretace experimentu

- **Data** – chování na konkrétních instancích, s konkrétními parametry
 - Kvantitativní, pohled zvenčí, mnoho jednotlivých dat
- **Interpretace** – obecný (obecně užitečný) závěr
 - Nutná extrapolace – obecně málo spolehlivá – vyžaduje důvěru
 - Nejčastěji kvalitativní
 - Často vyžaduje porozumění
 - Pohled zevnitř
 - Jednoduchá formulace

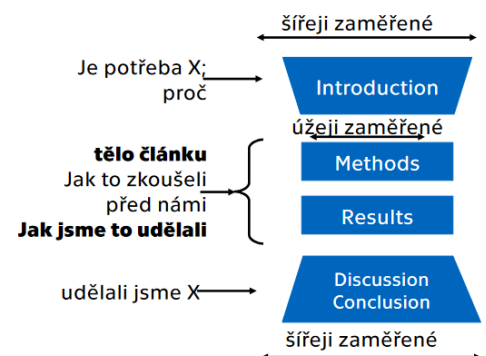
Prezentace výsledků

- Experiment



- IMRaD

- **Introduction** – kontext výzkumu, proč byla studie provedena, jaká otázka, jaká hypotéza, proč experimentální přístup
 - Vybuduje výchozí bod pro další výklad
 - Důvod zahrnuje chybějící nebo neúplné znalosti, problémy
- **Methods** – kdy, kde, jak provedeno, proč takové metody, co se podniklo k ověření korektnosti (RNG, ...), jaký experimentální materiál, proč takový materiál
- **Results** – jaké výsledky, jaká odpověď, hypotéza potvrzena?
 - Všechna relevantní data a pouze relevantní data
- **Discussion** – co odpověď může znamenat, vztah k dosavadnímu výzkumu, perspektivy pro budoucí výzkum
 - Všechno, co podporuje tvrzení článku, všechno, co se nezdá podporovat tvrzení článku



- Přesvědčení a data

- Tabulky bez grafů – nepřesvědčí, grafy bez tabulek – nejdou použít pro další výzkum
- Chyby: Chybějící sloupce v tabulkách, grafy bez srozumitelného sdělení, nevýstižné, matoucí popisky, grafy/obrázky vyžadující dlouhý popis – složité sdělení, neúmyslné lži (lexikografické pořadí instancí, oslí můstky, posunuté počátky pokud to nedává smysl)

13. Princip lokálních heuristik, pojem globálního a lokálního minima, obrana před uváznutím v lokálním minimu.

NI-KOP

Stavový prostor

- **Stav** – jedno ohodnocení proměnných
- **Stavový prostor** – dvojice (S, Q), S = množina stavů, Q = množina operátorů
- **Akce** – aplikace operátoru na stav
- **Graf** stavového prostoru algoritmu – orientovaný graf, kde hrany odpovídají akcím
- **Okolí** stavu – množina stavů dosažitelných ze stavu aplikací některé operace
- **K-okolí** stavu – množina stavů dosažitelných ze stavu aplikací nejméně jedné a nejvýše k operací
- **Sousední** stavy – stavy z okolí stavu
- **Inverzní operátory** – libovolnou vloženou věc lze vyjmout, v grafu lze libovolně dlouho bloudit
- **Stavový prostor TSP/HC:**
 - o Konfigurace – podgraf
 - o Uzel stavového grafu = podgraf
 - o Operátor = např. dvojjáměna na hranách
- Pokud je úkol nalézt cestu, je množina stavů posloupností akcí
- **Prohledávání stavového prostoru:**
 - o **Úplná strategie** – navštíví všechny stavy kromě těch, o kterých víme, že nedávají (optimální) řešení
 - o **Systematická strategie** – úplná strategie, která navštíví každý stav nejvýše jednou

Heuristiky

- **Heuristika** = přístup k řešení používající metody, které negarantují nalezení optimálního řešení
- Požadavky na heuristiku:
 - o **Použitelnost v praxi** – metoda musí pracovat na praktických instancích, s přijatelnou přesností a přijatelným výpočetním časem
 - o **Omezení a optimalizace** – nejlepší čas pro potřebnou přesnost, nejlepší přesnost v časovém limitu
 - o **Přesnost** – přibližné a pro nás uspokojující řešení
- **Druhy heuristik**
 - o **Konstruktivní heuristika** – začne z triviální konfigurace a postupnými kroky konstruuje řešení
 - o **Iterativní heuristika** – začne z nějakého (i neplatného) řešení a to postupně vylepšuje
 - o **Dvojfázové heuristiky** – první fáze slouží k získání řešení (konstruktivně, náhodné řešení), ve druhé fázi iterativní vylepšování
- **Druhy heuristických metod**
 - o **Lokální metody** – práce vždy jen s aktuálním stavem, může uvažovat i blízké okolí, např. sousedy
 - o **Globální metody** (ne heuristiky) – dekompozice instance na menší instance téhož problému
 - **Rozdělení na panuj** – možnost rekurze
- **Heuristická funkce**
 - o Stav s je přiřazena hodnota (odhad)
 - o Preferovány jsou stavy s vyššími/nížšími hodnotami
 - o Při shodné hodnotě u více stavů lze vybrat náhodně
- **Hladové heuristiky** – rozhodují ve prospěch lokálního optima
 - o Doufáme, že nalezneme globální optimum

Lokální heuristiky

- U náročnějších problémů je prohledávání neúnosné
- Jednoduché heuristiky – jen jeden aktuální stav, koukají na sousedy (okolí)
- Cyklus heuristiky:
 - o Kontrola ukončující podmínky
 - o Pokud není ukončeno, výběr stavu z okolí
 - o Pokud je lepší, označí se jako nejlepší
 - o Pokud není, vrací se do prvního kroku bez nalezení nejlepšího

- Náhodná procházka
 - o Posun na jakéhokoliv souseda, i horšího
 - o Není úplná ani systematická
- **Best-only** – metoda nejrychlejšího sestupu/vzestupu
 - o Pokud je nový stav lepší, tak se nahradí za momentální
 - Cyklus přes Q, takže se opravdu vybere nejlepší ze všech možností
 - o Vráť prázdný stav, pokud neexistuje lepší soused
 - o Pořadí procházení množiny Q neovlivní výsledek, pokud nejlepších stavů není více
- **First improvement**
 - o Znovu cyklus přes Q, ale jakmile se nalezne lepší řešení, tak return = první zlepšení
 - o Vráť prázdný stav, pokud neexistuje lepší soused
 - o Pořadí procházení množiny Q ovlivní výsledek – randomizace
- **Návrh heuristiky** a jejího stavového prostoru
 - o Chceme mnoho jednoduchých, ale rychlých akcí
 - o Chceme mnoho akcí, které nemění konfiguraci drasticky (občas menší množství radikálních akcí)
- Okolí heuristik **Kernighan-Lin** – heuristiky pro TSP a bisekci grafu
 - o Aplikace daného operátoru opakovaně až do stop podmínky bez ohledu na optimalizační kritérium nebo heuristickou funkci
 - o Z takto získaného okolí výběr nejlepšího stavu jako příštího
- **Pohyb v prohledávacím prostoru**
 - o Strategie mohou, ale nemusí být systematické
 - o Typický krok prohledávání: výběr proměnné, výběr hodnoty proměnné
 - o Prořezávání se vztahuje na oblast stavového prostoru
 - o Možnost odvolat nastavení proměnné (backtracking, ...)
- **Práce s heuristikou**
 - o **White box fáze** – omezená sada instancí, detailní měření, vzhled, porozumění, modifikace heuristiky
 - o **Black box fáze** – plná sada instancí, měření výsledků, ověření kvality a výkonu, žádné modifikace heuristiky

Lokální minima

- **Lokální minimum** – všechny sousední stavy mají horší hodnotu optimalizačního kritéria, ale není to ta nejnižší hodnota v celém SP
- **Globální optimum** – stav, ve kterém je hodnota optimalizačního kritéria nejlepší mezi všemi možnými stavy
- **Uváznutí v lokálních minimech** – kvalita výsledného řešení silně závisí na kvalitě počátečního řešení
- **Řízení úniku z lokálních minim**
 - o **Diverzifikace** – rovnoměrný průzkum stavového prostoru
 - Velká ochota připustit akci, která vede k horšímu řešení
 - o **Intenzifikace** – konvergence k finálnímu řešení
 - Malá ochota připustit akci, která vede k horšímu řešení
- **Řešení diverzifikace**
 - o Zvětšení okolí – pravidelné k-okolí – exponenciální velikost s k
 - Okolí Kernigham-Lin
 - Dynamické řízení okolí
 - o Pokročilejší heuristiky:
 - Přípuštění akce, která **zhorší řešení** (simulované ochlazování)
 - Nutné řízení – intenzifikace vs. Diverzifikace
 - Práce s **více stavy/konfiguracemi** (genetické algoritmy)
 - Různé způsoby interakce mezi současně zpracovávanými stavy
 - Spuštění z více různých náhodných počátečních stavů (randomizované algoritmy)
 - **Mapování stavového prostoru** (částečné), modelování stavového prostoru, učení
 - Backtracking – Algoritmus vycouvá z minima návratem – může být časově náročné
 - Restriktivní opatření, kterým stavům se vyhnout (tabu search)

14. Princip genetických algoritmů, význam selekčního tlaku pro jejich funkci.

NI-KOP

- Simulovaná evoluce

- **Více stavů najednou** – snížení možnosti uváznutí v jediném minimu, zpravidla konstantní počet
- Operátory – unární x binární
- Definice operátorů – nad reprezentacemi, problémově nezávislé
- Konfigurace = **jedinec** (fenotyp)
- Kódování konfigurace – genetická reprezentace jedince (**genotyp**, chromozom)
- Proměnná kódování – **gen**
- Hodnota proměnné – alela
- Aktuální množina reprezentací konfigurací – **generace**, populace
- Unární operátor – **mutace**
- Binární operátor – **křížení**
- Optimalizační kritérium – **zdatnost (fitness)**
- Rozšíření kvalitní konfigurace – **konvergence**
- Rozšíření konfigurace uváznulé v lokálním minimu – **degenerace**
- Biodiverzita – diverzita populace

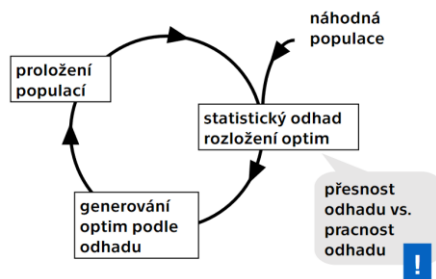


- Explorace = diverzifikace, exploitace = intenzifikace
- **Genetické algoritmy**
 - Kódování – **binární řetězec**
 - Binární operátory – křížení – 1-bodové, 2-bodové, uniformní
 - Unární operátory – mutace, inverze
 - Řízení populace – nová generace nahradí původní
- **Evoluční strategie**
 - Kódování – **vektor reálných čísel**
 - Strategické parametry – mutace, interakce mezi složkami
 - Operátory – mutace (dominuje) – přičtení Gaussova rozložení
 - Křížení – diskriminující, průměrující
 - Řízení populace – z x rodičů a y potomků se vybere x členů nové generace, nebo náhrada vcelku
- **Genetické programování**
 - Kódování – **rozkladový strom** výrazu
 - Operátory – křížení, mutace, definice stavebního bloku, editace
 - Řízení populace – nová populace nahradí starou
- **Evoluční programování**
 - Kódování – **automat**
 - Operátory (pouze unární)
 - Změna výstupního symbolu
 - Změna přechodu
 - Přidání/vypuštění stavu
 - Změna počátečního stavu
 - Řízení populace – z x rodičů a y potomků se vybere x členů nové generace
- Společné rysy
 - více individuí

- prostředky diverzifikace – mutace, ...
 - prostředky intenzifikace – selekce pro rekombinaci, selekce pro další generaci
- charakteristické rysy
 - jaká reprezentace
 - jaké unární, binární operátory
 - jaká selekce pro rekombinaci a konstrukci následující generace
- **Genetické algoritmy**
 - **Binární řetězec**
 - Vektor proměnných obecně různých domén
 - Permutace řetězce z dané abecedy
 - Genetické operátory – nastavení pravděpodobnost mutace, pravděpodobnost křížení
 - **Křížení** (rekombinace) – jednobodové (náhodný bod), dvoubodové (dva náhodné body), uniformní (vektor náhodných 0/1)
 - **Inverze** – zpřeházet pořadí bitů, ale ponechat význam proměnných
 - **Mutace** – náhodně vybraný bod z celé populace
 - Někteří jedinci mohou být mutováni vícekrát, někteří vůbec
 - **Selekce** – účel je způsobit, aby početní zastoupení jedince v populaci odpovídalo jeho zdatnosti
 - Řízení selekčního tlaku – převod informace ze zdatnosti na početnost
 - Selekční tlak = pravděpodobnost výběru nejlepšího jedince
 - Velký – nebezpečí degenerace
 - Malý – pomalá konvergence
 - Šum vnesený mutací může převážit nad pomalou konvergencí – divergence
 - Regulace selekčního tlaku se liší podle výběrového mechanismu
 - **Ruletový výběr** – rozevření políčka rulety odpovídá žádané pravděpodobnosti výběru
 - Provedeme m náhodných voleb úhlu, odtud m prvků
 - Jeden prvek může být vybrán vícekrát
 - **Řízení selekčního tlaku pro ruletový výběr** – přímá úměra mezi zdatností a poměrnou pravděpodobností výskytu často zdegeneruje populaci – je třeba nadřžovat slabším
 - Přepočítání zdatnosti lineární funkcí – scaling – Lineární škálování
 - Použití pořadí ve zdatnosti místo zdatnosti (ranking)
 - **Turnajový výběr** – náhodně vybrat r jedinců (turnaj) a z něj nejlepšího
 - Opakovat až do naplnění populace
 - Selekční tlak se řídí velikostí turnaje
 - „turnaj“ s jedním jedincem – žádný selekční tlak
 - Turnaj přes celou populaci – jistota výběru nejlepšího jedince
 - **Zkrácený výběr** – z populace M jedinců se vybere pM nejzdatnějších, každý vstupuje do rekombinace 1/p krát
 - Informace obsažená ve zdatnosti je redukována na jediný bit
 - Variace – několik stejně velkých skupin podle zdatnosti, pro každou skupinu četnost výběru, redukce informace na typicky 2 bity
 - **Řízení populace**
 - Náhrada en bloc – nová generace vzniklá křížením a mutací nahradí starou
 - Částečná náhrada – z x rodičů a y potomků se vybere x členů nové generace
 - Ustálená populace – po každém křížení potomek nahradí nejslabšího jedince
 - Přechodové formy mezi en bloc a ustálenou populací
 - **Elitismus** – několik málo nejlepších jedinců přejde ze staré populace do nové (pozor na degeneraci)
 - Velikost populace
 - Malá (10) – nebezpečí ztráty diverzity
 - Méně obtížné problémy – od 30 jedinců, lineární růst
 - Obtížné problémy – 100 jedinců
 - Velké instance – sublineární růst s velikostí populace, technické důvody

- podmínky ukončení – pevný počet generací, příznaky konvergence
 - změna průměrné zdatnosti mezi generacemi
 - Rozložení zdatnosti v generaci
- **Omezující podmínky** – co když výsledek genetického operátoru není řešení
 - **Relaxace** – převod omezujících podmínek na penalizaci
 - Velikost penalizace – každé řešení lepší než ne-řešení, minimální penalizace, která zabrání aby bylo ne-řešení vybráno jako optimální
 - Způsob penalizace – vzdálenost od řešení (odhad ceny opravy, počet porušených podmínek)
 - **Trest smrti** – zahodit výsledek
 - Zmarněná práce – výpočet optimalizačního kritéria, genetický operátor
 - Snížená dostupnost stavového prostoru
 - **Doménové reprezentace a operátory** – viz permutační operátory
 - **Dekodéry** – volba reprezentace tak, aby každý genotyp odpovídal řešení
 - Každé řešení musí být poskytnuto
 - Každý výstup dekodéru je řešením
 - Každé řešení reprezentováno tímž počtem genotypů
 - Lokalita – malá změna genotypu působí malou změnu řešení
 - Rychlý výpočet
- **Schémat a teorie stavebních bloků**
 - Reprezentace stavebních bloků – schémata – co mají chromozomy společného?
 - **Implicitní paralelismus** – každý genotyp o n genech obsahuje 2^n schémat
 - Každý výpočet zdatnosti získává údaj o průměrné zdatnosti všech schémat
 - Zpracování populace o velikosti M znamená zpracování $O(M^3)$ schémat
 - Jiný pohled na GA:
 - Populace jako množina schémat
 - Vývoj populace jako vývoj zastoupení schémat
 - GA jako zpětnovazební dynamický systém – ve smyčce obíhají schémata, převod zdatnosti schémat na jejich zastoupení
 - Věta o schématech – selekce s pravděpodobností úměrnou zdatnosti, jednobodové křížení
 - Krátká schémata malého řádu přežívají ve smyčce snáze
 - Větší schémata potřebují lepší zdatnost
 - Mutace zkracuje životnost
 - **Linkage problém**
 - Životnost a vývoj schémat závisí na jejich délce, nikoliv pouze na řádu
 - Uniformní křížení zpracovává správně pouze schémata řádu 1
 - Řešení – dynamické přeuspořádání, explicitní práce se stavebními bloky
 - Zavádějící, klamné funkce – kombinace podobných funkcí – optimalizační kritérium testovacích úloh
 - Reálné úlohy mohou skrytě mít podobný charakter
 - **Kompetentní GA**
 - Klasický GA nezachází optimálně se schématy vyšších řádů, protože o nich neví
 - Řešení:
 - Práce se stavebními bloky explicitně jako s oddělenými fragmenty genetické informace – messy GA
 - Práce s pravděpodobnostním modelem vazeb mezi hodnotami proměnných, které vytvářejí stavební bloky – bayesovská optimalizace
- **Fast messy GA**
 - Fragment genetické informace – někde nedospecifikovaný, někde přespecifikovaný
 - Hodnocení schémat – referenční jedinec (kompetitivní šablona) poskytne informaci pro nedospecifikovaná místa

- Bereme v úvahu první výskyt hodnoty pro dané místo genotypu
- **Generování schémat**
 - Deterministicky – vygenerovat všechny a vyhodnotit (messy GA)
 - Stochasticky – vygenerovat schémata většího řádu s rovnoměrnou pravděpodobností
 - Při filtraci využít selekce a implicitního paralelismu (fast messy GA)
- **Filtrace schémat** – generovaná schémata mají řád $k < l$
 - Selekce turnajovým výběrem
 - Velikost turnaje 2, první účastník zvolen náhodně
 - Druhý účastník musí mít společnou určitou míru informace
 - Soutěží nápady na podobné téma
 - Zůstává zdatnější
 - Zkrácení schématu vyjmutím náhodného genu
 - Zkrácená schémata musí být znovu vyhodnocena
- **Rekombinace schémat**
 - Rozdělení (cut) obou rodičů v náhodných bodech
 - Spojení (splice) rozdělených fragmentů
 - Obě operace mají nezávisle nastavenou pravděpodobnost
- Algoritmy založené na statistických modelech závislostí



-
- **Bayesovská optimalizace** – po odhadu modelování bayesovskou sítí – optimalizační algoritmus a metrika
- **Bayesovská síť** – každá proměnná reprezentace modelována uzlem orientovaného acyklického grafu
- Užití bayesovských sítí
 - Najít nejpravděpodobnější vysvětlení pozorovaných pravděpodobností
 - Zjistit pravděpodobnosti hodnot proměnných v závislosti na vstupních pravděpodobných
- Optimalizace bayesovských sítí
 - Metrika – jak dobře vystihuje měřená síť současnou populaci
 - Algoritmus – pouze nejlepší nebo jiná lokální heuristika
 - Operace – přidání hrany, obrácení hrany, odebrání hrany
- **Paralelizace evolučních algoritmů**
 - **Granularita**
 - Jedna aktuální generace v paralelním systému
 - Sdílená paměť
 - Výpočet zdatnosti, křížení, mutace se snadno škálují s počtem procesorů
 - Minimální a maximální zdatnost, ranking – nutná komunikace, logaritmická složitost
 - Jedna aktuální generace na procesor – nutná občasná výměna genetického materiálu (např. se sousedními procesory)
 - Napomáhá zachovat diverzitu genetického materiálu
- Island parallel GA

15. Princip simulovaného ochlazování, význam parametrů a způsoby jejich řízení.

NI-KOP

Simulované ochlazování

```
T ← počáteční teplota;  
best ← ∅;  
while (!frozen (T, ...)) {  
  while (!equilibrium (...)) {  
    state = try(state);  
    if (state.better(best)) best ← state;  
  }  
  T = cool (T, ...);  
}
```

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state  
  current ← problem.INITIAL  
  for t = 1 to ∞ do  
    T ← schedule(t)  
    if T = 0 then return current  
    next ← a randomly selected successor of current  
    ΔE ← VALUE(current) – VALUE(next)  
    if ΔE > 0 then current ← next  
    else current ← next only with probability  $e^{-\Delta E/T}$ 
```

- Algoritmus iterativní optimalizace

- o Frozen, equilibrium a cool určují průběh teploty – rozvrh ochlazování
- o Princip algoritmu:
 - Náhodně se zvolí počáteční řešení, volba počáteční teploty
 - Náhodný výběr souseda
 - Pokud je lepší, přechod
 - Pokud ne, zjistí se, jak moc je horší
 - o Pokud je dostatečně vysoká teplota a zhoršení dostatečně malé, přejdeme
 - Po x krocích snižování teploty (ne pokaždé, aby to nebylo moc rychle)
 - x = **délka ekvilibria**
 - Ovlivňuje pravděpodobnost přechodu do horšího stavu
 - Po dostatečném snížení teploty konec

- Vlastnosti simulovaného ochlazování

- o **Diverzifikace** – rovnoměrný průzkum stavového prostoru
 - Velká ochota připustit akci, která vede k horšímu řešení
- o **Intenzifikace** – konvergence k finálnímu řešení
 - Malá ochota připustit akci, která vede k horšímu řešení
- o **Počáteční stav** – řešení z jiné heuristiky nebo náhodná řešení
- o **Vysoké teploty** – převaha diverzifikace
 - Vysoká pravděpodobnost přijetí horšího řešení
- o **Nízké teploty** – převaha intenzifikace, konvergence k minimu

- Ochlazovací funkce:

$$t_k = \frac{c}{\log(1 + k)}$$

- o k ... číslo kroku
- o c ... hloubka lokálního minima
- o Proces po nekonečném počtu kroků skončí v globálním minimu – asymptotická konvergence

- Parametry

- o **Equilibrium a koeficient ochlazování**
 - Teplota se vynásobí parametrem mezi 0 a 1. většinou 0.8 – 0.999
 - Správná míra ochlazování lze za běhu řídit délkou ekvilibria (konstantní nebo proměnná délka)
 - Příliš velký koeficient – výpočet trvá dlouho bez zlepšení
 - Příliš nízký – malé prohledávání, riziko uvážnutí v lokálním minimu
 - **Délka ekvilibria** – ovlivňuje počet okolních stavů, které prohledáváme v každém teplotním skoku
 - Správná hodnota závisí na míře ochlazování
 - Příliš vysoká – dlouhý výpočet bez zlepšení
 - Příliš nízká – uvážnutí v lokálním minimu
- o **Počáteční teplota**
 - Známe hloubku lokálních minim → nastavíme teplotu tak, aby pravděpodobnost úniku z minima byla např. 0.5

- **Zpětnovazební řízení**
 - Rychle zvyšujeme teplotu
 - Sledujeme četnost přijatých změn k horšímu
 - Zaznamenáme teplotu pro pravděpodobnost např. 0.5
 - Vrátime původní stav a nastavíme teplotu
- Příliš vysoká – přijmutí i velkých zhoršení
- Příliš nízká – riziko uváznutí v lokálním minimu v důsledku nedostatečného prohledávání stavového prostoru
- **Frozen** – ukončovací podmínka
 - Četnost změn (jakýchkoliv) klesla pod nastavenou mez
 - Četnost změn k lepšímu klesla pod nastavenou mez
 - Pevná mez teploty
 - Koncová teplota
 - Příliš vysoká – riziko uváznutí v lokálním minimu
 - Příliš nízká – výpočet trvá dlouho bez dalšího zlepšení výsledku
- **Cenová funkce**
 - Lze spočítat, i když konfigurace není řešením (výpočet optimalizačního kritéria)
 - Navádí optimalizaci stále blíže k řešení (hodnocení „špatnosti“ konfigurace)
 - Možnosti implementace
 - Konfigurace, která není řešením, má **mnohem horší cost** než jakékoliv řešení
 - Konvergence vždy k řešení
 - 2 úlohy, 2 fáze – nalezení řešení, vylepšení řešení
 - Nebudou se střídat
 - Mohou potřebovat jiná řešení
 - Konfigurace, která není řešením, může mít **stejný cost** jako nějaké řešení
 - Lepší vlastnosti stavového prostoru
 - Konvergence k „ne-řešení“ je možná
- **Počáteční řešení**
 - **Náhodná počáteční řešení**
 - Vícenásobné spuštění
 - Měření iterativní síly
 - Dobře aplikované simulované ochlazování není závislé na počátečním řešení – těžiště práce v iteracích
 - **Konstruktivní počáteční řešení**
 - Chytrá konstruktivní fáze vede k hlubokému lokálnímu minimu
 - Je to aspoň nějaké minimum
- **Omezující podmínky**
 - Např. vyhození nevalidních sousedů, relaxace je vhodnější
 - **Relaxace** – náhrada omezujících podmínek přírážkou
 - **Omezující podmínky** – co když nemohu zabránit, aby akce převedla řešení na konfiguraci, která řešením není?
 - Relaxace, zahození kandidátní konfigurace, oprava kandidátní konfigurace
- **Vývoj heuristiky**
 - **White box evaluation** – omezená sada instancí, detailní měření, vzhled, porozumění, modifikace heuristika a adaptačních mechanismů
 - **Black box evaluation** – plná sada instancí, měření výsledků, ověření kvality a výkonu

16. Výkonnostní měřítka paralelních algoritmů, PRAM model, APRAM model, škálovatelnost.

NI-PDP

Výkonnostní měřítka paralelních algoritmů

- **Paralelní časová složitost $T(n, p)$** – závisí nejen na n , ale i na počtu procesorů/jader p
 - o $p = \# \text{ procesorů} = \# \text{ jader} = \# \text{ vláken}$
 - o **Paralelní čas $T(n, p)$** = čas, který uplynul od začátku paralelního výpočtu do okamžiku, kdy poslední (nejpomalejší) procesor skončil výpočet
 - o $T(n, p)$ závisí na architektuře paralelního výpočtu → hodnocení par. Algoritmu musí vždy brát v úvahu architekturu počítače
 - o $T(n, p)$ je měřen čítáním:
 - Výpočetních kroků – aritmetické, logické, paměťové operace
 - Komunikačních kroků – přenos a výměna dat mezi procesory
- **Paralelní cena $C(n, p)$** – $C(n, p) = p \times T(n, p)$
 - o Většinou statická alokace výpočetních jader
 - o Výpočet začíná vytvořením vláken a ty jsou použita k výpočtu až do konce, i když některá mohou být nějakou dobu neaktivní (idle)
 - o Měřením kvality je součin procesory – čas = paralelní cena
 - o Sekvenční složitost = $SU(n) \rightarrow C(n, p) = \Omega(SU(n))$
 - o **Cenová optimalita** – paralelní algoritmus má optimální cenu, pokud $C(n, p) = O(SU(n))$
 - Cena je optimální právě tehdy když $C(n, p) = \theta(SU(n))$
- **Paralelní zrychlení $S(n, p)$** – $S(n, p) = \frac{SU(n)}{T(n, p)} \leq p$, nebo asymptoticky $S(n, p) = O(p)$
 - o Paralelní zrychlení je lineární, právě když $S(n, p) = \Omega(p) \rightarrow S(n, p) = \theta(p) (\leq p)$
 - o **Lineární zrychlení** = nejvyšší cíl paralelního programu – jestliže p stoupne k -krát, chceme, aby $T(n, p)$ klesnul k -krát – obtížně splnitelné
 - o **Superlineární zrychlení** – výjimečně dosažitelné
 - a. Sekvenční algoritmus je paměťově náročnější, než kapacita paměti a souhrnná kapacita pamětí paralelního systému je dostatečná a při paralelním výpočtu ušetříme swapování mezi hlavní pamětí a diskem
 - Sekvenční algoritmus znevýhodněn tím, že běží za jiných HW podmínek než paralelní
 - b. Anomálie při prohledávání kombinatorického stavového prostoru
- **Paralelní efektivnost $E(n, p)$** – relativní vytížení jader dedikovaných paralelnímu výpočtu během výpočtu
 - o Vždy $< 100\%$ (komunikační a synchronizační režie)
 - o $E(n, p) = \frac{SU(n)}{C(n, p)} \leq 1$
 - o $E(n, p)$ = zrychlení na jádro
 - o Konstanta $0 < E_0 < 1$
 - o Paralelní algoritmus má konstantní efektivnost, jestliže $E(n, p) \geq E_0 \rightarrow E(n, p) = \Omega(1)$
- **Paralelní optimalita výkonnosti** – z definic plyne, že paralelní algoritmus je cenově optimální \Leftrightarrow má lineární zrychlení \Leftrightarrow má konstantní efektivnost

PRAM model

- **PRAM** = paralelní RAM = výpočetní model
 - o Množina p procesorů
 - 1 procesor \rightarrow vlastní lokální (soukromá) paměť + index i
 - o M sdílených paměťových buněk (pole)
 - o Každý p může přistoupit do jakékoliv buňky sdílené paměti v $O(1)$ čase
 - o Řešení konfliktů – explicitní ošetření
- **PRAM algoritmus**
 - o Vstup = n položek v (obvykle prvních) n buňkách sdílené paměti
 - o Výstup = n' položek v n' buňkách sdílené paměti

- Procesy provádí synchronně 3 typy instrukcí:
 - o **READ** – čtení sdílené buňky
 - o **LOCAL** – lokální operace
 - o **WRITE** – zápis do buňky sdílené paměti
- Komunikace procesorů = READ/WRITE sdílené buňky
- PRAM algoritmus lze zapsat regulárními výrazy
- **Jednotkový** model → R/L/W trvá čas 1
- **Globální** model → L trvá 1 a R/W trvají konstantní čas $d > 1$
- Ošetřování konfliktů při přístupech do sdílené paměti
 - o **EREW** – Exclusive Read Exclusive Write
 - Žádné 2 procesory nesmějí číst nebo zapisovat tutéž sdílenou pam. Buňku současně
 - o **CREW** – Concurrent Read Exclusive Write
 - Současná čtení 1 b. povolena, ale v 1 okamžiku může jen 1 proces zkoušet zapsat do dané buňky
 - o **CRCW** – Concurrent Read Concurrent Write
 - Povoleny současné čtení a zápisy téže buňky
 - **Priority-CRCW-PRAM**
 - Procesy mají pevné priority – dokončení zápisu povoleno procesu s nejvyšší prioritou
 - **Arbitrary CRCW-PRAM**
 - Dokončení zápisu povoleno náhodně vybranému procesu (algoritmus nesmí činit žádné předpoklady, který proces byl vybrán)
 - **Common-CRCW-PRAM**
 - Všechny procesy smí dokončit zápis, pokud jsou všechny zapisované hodnoty stejné
 - o Každý a. musí zajistit splnění podmínky
 - o Jinak a. není správný a stav PRAM není definován

APRAM model

- **APRAM** = asynchronní PRAM
 - o Procesy pracují asynchronně, neexistují centrální hodiny
 - o READ, WRITE, LOCAL jako PRAM
 - o Nutná explicitní synchronizace – **bariérová synchronizace**
 - o Doba přístupu do sdílené paměti není jednotková
- **APRAM výpočet** = posloupnost globálních fází, ve kterých procesory pracují asynchronně, oddělených bariérovou synchronizací
- Dva + procesory nemohou přistupovat do téže buňky v téže globální fázi, pokud jeden z nich do ní zapisuje
- Výkonnostní parametry:
 - o Lokální operace – 1
 - o Globální READ/WRITE: d
 - o K po sobě jdoucích globálních R/W: $d+k-1$
 - o Bariérová synchronizace: $B(p)$
 - $2 \leq d \leq B(p) \leq p$
- 2 možné implementace bariéry
 - o **Centrální čítač**
 - Inicializovaný na 0 a na příchozí fázi, procesy přistupují ve vzájemném vyloučení
 - $B(p) = \theta(dp)$
 1. Proces dorazí k bariéře, zkontroluje, zda je v příchozí fázi a inkrementuje čítač
 2. Je-li čítač $< p$, proces se deaktivuje
 3. Jinak nastaví bariéru do odchozí fáze a aktivuje ostatní procesy
 4. Poslední aktivovaný proces nastaví bariéru do příchozí fáze

- **Binární redukční strom**
 - $B(p) = \theta(d \log p)$
 1. Každý proces dorazí k bariéře a zkontroluje, zda je v příchozí fázi
 2. Čeká, až skončí redukce v jeho podstromu
 3. Po jejím skončení pošle signál rodiči
 4. Kořen stromu počká na redukci z obou podstromů a přepne do odchozí fáze

Škálovatelnost

- **Amdahlův zákon saturace paralelizace**

- Každý sekvenční algoritmus A s časem $T_A(n)$ nad daty o velikosti n se proporčně skládá z
 1. Inherentně sekvenčního podílu f_s , který může provést pouze 1 vlákno $0 < f_s < 1$
 2. Paralelizovatelného podílu $1 - f_s$
- Nechť A je paralelizován pro pevné n pomocí $p > 1$ vláken
- Pak pro zrychlení A platí při p vláknech ideálně:

$$S(n, p) = \frac{1}{f_s + \frac{1-f_s}{p}} \leq \frac{1}{f_s}$$

- Nezávisle na tom, kolik vláken bylo použito, nemůže zrychlení přesáhnout $\frac{1}{f_s}$
- Po jisté hranici nemá přidávání procesů už smysl, bo pro něj není dost paralelní práce
- Problém fixní velikosti poskytuje omezené množství paralelismu a tudíž při provedení i omezuje použitelný počet paralelních vláken/jader

- **Gustafsonův zákon**

- S rostoucím p máme úměrně navyšovat i velikost problému n
- Pak sekvenční část trvá konstantní čas t_{seq} nezávisle na p (V/V operace, inicializace), kdežto inherentně paralelní část $t_{par}(n, p)$ bude lineárně škálovat s p v čase

$$S(n, p) = \frac{t_{seq} + t_{par}(n, 1)}{t_{seq} + t_{par}(n, p)}$$

- $\lim_{n \rightarrow \infty} S(n, p) = p$ pro monotónně rostoucí $SU(n)$
- **Paralelní škálovatelnost** – schopnost par. Počítače se zvětšit, pokud narůstá velikost řešeného problému
 - **Silná** – schopnost p. a. pro fixní n dosáhnout lineárního zrychlení s rostoucím p
 - Měří pokles efektivnosti, pokud p roste a n se nemění
 - **Slabá** – definuje, jak se mění par. Čas s p pro fixní n/p
 - Měří růst n takový, že při rostoucím p zůstává efektivnost stejná
 - **Škálovatelnost** = schopnost p. a. držet paralelní optimalitu, pokud oba p a n rostou/klesají

- **Izoefektivní funkce ψ_1, ψ_2**

- $\psi_1(p)$ = asymptoticky minimální funkce taková, že

$$\forall n_p = \Omega(\psi_1(p)) : E(n_p, p) \geq E_0$$
- $\psi_2(p)$ = asymptoticky maximální funkce taková, že

$$\forall n_p = O(\psi_2(p)) : E(n, p_n) \geq E_0$$

- Z Amdahlova zákonu vyplývá:

$$p = \omega(\psi_2(n))$$

- Abychom si udrželi konstantní efektivnost, musí být procesorů alespoň $\psi_2(p)$

- Z Gustafsonova zákonu vyplývá, že když velikost problému roste s p vztahem

$$n = \Omega(\psi_1(p)),$$

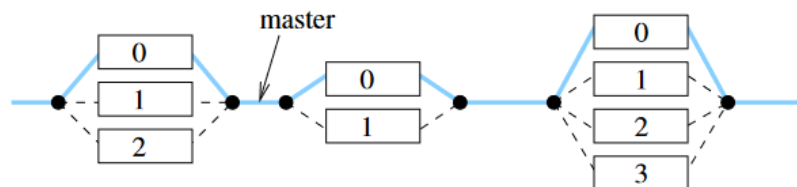
efektivnost nebude klesat

17. Programování nad sdílenou pamětí, programový model OpenMP, datový a funkční paralelismus, synchronizace vláken, vícevláknové algoritmy (násobení polynomů, násobení matic, řazení).

NI-PDP

OpenMP

- **OpenMP** – explicitní model paralelního výpočtu, kdy má programátor plnou kontrolu a zodpovědnost za paralelní výpočet
- **Paralelní regiony** = části původně sekvenčního kódu
 - o V nich pomocí fork-join mechanismu vytvářena, prováděna a ukončována paralelní vlákna



- Mimo par. regiony pouze 1 hlavní (master) vlákno
- Podpora pro iterační i funkční model paralelismu
- Pomocí OpenMP direktiv v kódu paralelní regiony, ve kterých bude výpočet prováděn více paralelně běžícími vlákny nebo paralelně běžícími úlohami, kdy je každá úloha prováděna 1 vláknem
- Zákaz skákat z paralelního regionu ven či dovnitř
- Kompilace s -fopenmp
- Tvorba paralelních regionů – **direktiva parallel**
 - o Možné klauzule direktivy:
 - **If(podmínka)**: podmínka paralelizace regionu
 - **Num_threads(výraz)**: počet vláken v paralelním regionu
 - **Vlastnosti(seznam proměnných)**: OpenMP vlastnosti proměnných v paralelním regionu
 - o Na konci p. r. je implicitní bariéra
 - o Po jejím provedení jsou nová vlákna ukončena a dál pokračuje jen hlavní vlákno 0
 - o Pokud je 1 vlákno předčasně ukončeno, jsou ukončena všechna vlákna i celý program
- Vlastnosti proměnných v paralelním regionu
 - o **Shared** – daná skalární proměnná (ne pole, ne struktura) je sdílená všemi vlákny
 - o **Private** – daná proměnná je lokální ve vláknech – každé vlákno má nezávislou minimalizovanou instanci této proměnné
 - o **Firstprivate** – proměnná je lokální ve vláknech, každé vlákno ji má inicializovanou na hodnotu, kterou měla před vstupem do p. r.
 - o **Lastprivate** (pouze v paralelních cyklech) – p. je lokální ve vláknech, ale hodnota ze sekvenčně poslední iterace se po skončení p. cyklu přepokopíruje do proměnné hlavního vlákna procesu
 - o **Default** – určuje, jakou z předchozích vlastností budou mít implicitně všechny proměnné použité v paralelním regionu
 - o **Reduction** – určuje, že daná sdílená proměnná je lokálně nakopírovaná do každého vlákna a že po skončení par. Regionu se všechny lokální instance této proměnné zredukují pomocí zadaného redukčního operátoru a výsledek bude zapsán do původní sdílené proměnné
 - Musí to být skalární proměnná
 - Redukční operátory: +, *, -, &, ^, |, &&, ||
 - Nelze kombinovat s direktivou task
 - o **Threadprivate** – def. Globální platnost hodnot lokálních proměnných vláken v rámci celého programu napříč všemi paralelními regiony
 - o Počet vláken ve všech regionech musí být stejný, proměnné si „drží“ hodnoty při přestupech mezi p. r.

Datový a funkční paralelismus

- **Direktiva for** – přidělení jednotlivých iterací for cyklu uvnitř par. Regionu jednotlivým vláknům
 - Na konci par. Cyklu implicitně bariéra
 - Možné klauzule:
 - o **Schedule()**: upřesňuje způsob přidělení iterací cyklu vláknům
 - o **Collapse()**: upřesňuje paralelizaci vnořených cyklů (implicitně for jen na nejvyšší úrovni)
 - o **Ordered()**: pořadí provádění iterací je stejné jako při sekvenčním provádění
 - o **Nowait()**: vlákna po dokončení svých iterací nečekají na bariéře
 - **Klauzule schedule** – schedule(typ) × schedule(typ, chunk-size)
 - o Typy klauzulí:
 - **Static** – buď jsou vláknům přiděleny staticky cyklicky bloky (=chunks) o velikosti chunk-size, nebo se přidělí rovnoměrně (n/p)
 - **Dynamic** – dynamicky přiřazuje chunky po sobě jdoucích iterací velikosti chunk-size nebo 1
 - **Guided** – vláknům dynamicky přiděleny bloky x iterací, kde
$$x = \max ([\#dosud\ nepřidělených\ iterací/p], chunk_size)$$
 - **Runtime** – způsob přiřazení zvolen v okamžiku spuštění dle systémové proměnné OMP_SCHEDULE
 - **Auto** – přidělení it. Necháno kompilátoru/běhovému prostředí
 - o **Efektivita**:
 - Schedule(**static**[,k])
 - Nejmenší režie
 - Rovnoměrné rozdělení iterací
 - Ideální, pokud mají všechny iterace stejnou výpočetní dobu
 - K ovlivňuje promíchání iterací
 - Schedule(**dynamic**[,k])
 - Vyšší režie kvůli synchronizaci
 - Vyšší k snižuje režii
 - Výhodné při kolísavé době iterací
 - Schedule(**guided**[,k])
 - Vyšší režie (synchr.)
 - Vyšší k režii snižuje
 - Výhodné při postupně rostoucí době iterací
 - **Paralelizace 2-úrovňového for cyklu**
 - o **Statické přidělení vláken** (pro jednoduchost)

```
# parallel for nebo #parallel for collapse(2)
For(...)
    For(...)
        Funkce()
```
 - o Paralelizace **pouze vnitřního cyklu**
 - ```
For(...)
 #...parallel for
 For(...)
 Funkce()
```
      - ```
# ... parallel
    For(...)
        #...for
        For(...)
            Funkce()
```
- Nutné, je-li vnitřní smyčka datově závislá na vnější smyčce
- **Task** = úloha
 - Podporuje složitější funkční paralelismus s větší režii – vhodné i pro rekurzivní algoritmy (zapouzdření kódu i dat)
 - **Přidělování úloh** – typ producent-konzument
 - o Vlákna jsou producenti i konzumenti
 - **Úloha** = jednotka par. Výpočtu, obsahuje:
 - o Ukazatel na začátek svého kódu (k provedení)

- Vstupní data
- Dat. Strukturu, do které vloží svůj identifikátor vlákno, jakmile danou úlohu začne provádět jeho konzument (=vlastnické vlákno)
- **#pragma omp task** způsobí:
 - Vlákno – producent vygeneruje novou úlohu a vloží ji do zásobárny úloh (=task pool)
 - Úloha čeká, než ji volné vlákno – konzument vyzvedne a provede
- **Podmíněné spouštění par. Úloh**
 - If(...) – efektivní řízení task par. Rekursivních kódů, kdy rekurze závisí na splnění podmínky
 - Splněno – synovská úloha do task poolu
 - Nesplněno – pozastavení rodičovské úlohy a odložení do zásobárny úloh, ihned provedení nové synovské úlohy, po dokončení vyzvednutí rodiče a dokončení
 - #pragma omp **taskwait**
 - Rodičovská úloha čeká na dokončení všech synovských úloh
 - Stromová rekurze
- Volání task direktivy musí být uvnitř paralelního regionu
- Rekurzi startuje jediné vlákno
 - #pragma omp parallel num_threads(...){

pragma omp single

Funkce()

}

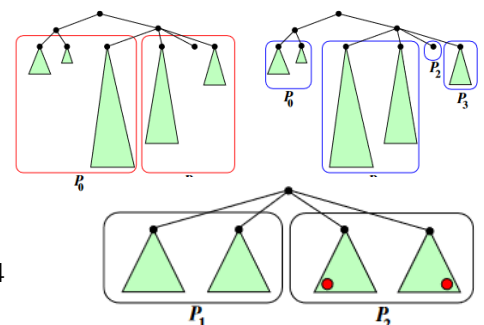
Synchronizace vláken

- **Synchronizační direktivy**
 - **Barrier** – místo, kam par. Vlákna daného p. r. musí dorazit a čekat na ostatní
 - **Master** – daný blok kódu smí provést pouze hlavní vlákno
 - **Single** – daný blok kódu smí provést pouze 1 libovolné vlákno
 - **Critical** – vytvoření kritické sekce
 - **Atomic** – operace nad paměťovou buňkou bude provedena jednovláknově a nepřerušitelně
 - **Flush** – propsání aktuálních hodnot daných sdílených proměnných do sdílené paměti
 - **Taskwait** – synchronizace synovských úloh s rodičovskou v task paralelismu
- **Bariéra a serializace v par. Regionu**
 - # pragma omp **single** – následující blok se smí provést pouze jednou – ostatní vlákna čekají na implicitní bariéře za single blokem
 - # pragma omp **master** – následující blok smí provést pouze hlavní vlákno – ostatní pokračují hned kódem, který je za tím
 - # pragma omp **barrier** – synchronizační bod, vlákna uspávána a probouzena, až dorazí všechna vlákna
 - Implicitně na konci par. Regionu a single
- **Kritická sekce**
 - Jedna/více částí kódu par. Regionu, které lze v 1 okamžiku provádět pouze 1 vláknem
 - Direktiva # pragma omp critical – anonymní kritická sekce
 - Několik kritických sekcí – vzájemné vyloučení vstupu vláken platí globálně pro všechny její výskyty
 - Direktiva #pragma omp critical name – pojmenovaná krit. sekce taky může být víckrát, platí to samé
- **Direktiva atomic a její použití**
 - Přístup do pam. Místa se skalárním datovým typem (integer, floating-point, ...) bude atomická operace
 - Nepřerušitelná R/W/RMW
 - Bude deterministický výsledek
 - Read, write, update, capture
 - Inkrementace - #... atomic update
 - Capture – rozšiřuje update o získání hodnoty dané pr. Před/po modifikaci
- Uživatelsky řízené předčasné ukončení par. Regionu – **direktiva cancel**
 - Provedením vydá vlákno ostatním signál k ukončení – přejde na bariéru

- Další vlákna, která později narazí na cancel provedou totéž
- Vlákna, která už poslední volání cancel minuly, standardně dokončují
- # pragma omp cancel construct[if(expr)]
 - Construct \in {parallel, for, taskgroup, sections}

Vícevláknové algoritmy

- **Prohledávání kombinatorického SP**
 - NPH úloha najít určitý 1 stav ve velkém SP
 - Vstupní proměnné, stavové proměnné, výstupní proměnné, omezení, optimalizační kritérium
 - Rozhodovací vs konstruktivní vs enumerační
 - **SB-DFS** – přípustný koncový stav bez optimalizace
 - **BB-DFS** – diskrétní optimalizační problém
 - Přípustný koncový stav s max./min. cenou
 - **PP-DFS** – prohledávání v iteracích se zvyšující se hloubkou SP (např. lineární prohlubování)
 - BB-DFS do hloubky L, pokud nenašlo řešení, prohloubí se
- **Paralelní algoritmy pro PKPS**
 - Čas. Složitost PKPS je superpolynomiální
 - Paralelní prohledávání může mít anomální chování
 - Základní podmínka úspěšného par. PKPS:
 - Jádra by měla být pokud možno stále vytížena prohl. pokud možno disjunktních částí SP
- **Statické rozdělení SP**
 - Nerozlišujeme mezi procesem a vláknem
 - P CPU jader, každé p_i provádí v 1 okamžiku jedno vlákno π_i
 - Základní postup statického rozdělení výpočtu:
 - Master vlákno τ_0 – sekvenční BFS – vygeneruje p odlišných stavových prostorů s cca stejným počtem nastavených stavových proměnných
 - Prohledávání stavových podprostorů přiděleno vláknům π_i
 - Každé vlákno τ_i (včetně hlavního) provede sekvenční DFS přiděleného SPp pomocí lokálního zásobníku
 - Výsledky lokálních PKSP předají hlavnímu vláknem τ_0 – globální řešení
- **Problémy efektivnosti statického rozdělení SP**
 - P jader by mělo mít podobný výkon a parametry paměti
 - Stejně rozsáhlé podprostory se stejnou sekvenční časovou náročností
 - ALE: navracení (=backtracking) je silně datově závislé
 - Výpočet vláken se může dost lišit
 - Některá pak budou neúčinná – neefektivní
- **Statické rozdělení SP a anomální chování**
 - V případě prohledávání celého SP rozděleného pouze staticky může vzniknout anomální chování:
 - 4-vláknové řešení vpravo pomalejší než 2-vláknové vlevo
 - 2. příklad:
 - V případě FSB-DFS:
 - Paralelní DFS s 2 vlákny trvá stejně jako se 4 vlákny
 - Anomálie – přidáním jader může DFS
 - Superlineárně zrychlit
 - Zpomalit
 - Pro efektivní PKSP:
 - Jemnější statická dekompozice v modelu dynamického Master-Slave
 - Doplnění o dynamické vyvažování zátěže
- **Dynamické vyvažování zátěže**
 - τ_0 generuje podprostory a přiřadí je vláknům



- τ_i – DFS pomocí lokálního zásobníku (= je aktivní)
- Aktivní τ_i vyprázdní zásobník, ale nenajde řešení
 - Stává se nečinným, ale žádá jiná vlákna o přidělení neprozkoumaných částí jejich SP
- τ_j se stane dárce - τ_i = příjemce
- Půlení zásobníku (rozdělování na k částí – požadavků) - $2^{\lceil \log k \rceil}$ částí
 - Neexp. stavy blízko dna/vrcholu zásobníku skrývají pravděpodobně větší/menší části SP
 - Položky nad řeznou výškou H se nepředávají
- Paralelní algoritmus dyn. **Master-Slave DFS**
 - Hlavní vlákno = Master, $p - 1$ dalších vláken = Slaves
 - M – sekvenční BFS → podprostory pro vlákna S
 - M pošle každému S 1 podprostor z množiny
 - S po přijetí podprostoru jede sekvenční DFS, nikdy se nevrací za počáteční stav svého zásobníku
 - S neúspěšně ukončí lok. PKSP → požádá M o další podprostor (definován lok. Zásobníkem)
 - M má nepřidělené podprostory → přidělí S → další lokální PKSP
 - M odpoví negativně → požádá S o ukončení aktivity
 - FSB-DFS – S nalezne řešení → informuje M → M oznámí všem S konec
- **Klasifikace efektivně paralelizovatelných algoritmů**
 - **Výpočetně intenzivní algoritmy** – čas procesoru strávený výpočtem nad daty je větší než čas nutný na přesun dat z paměti do CPU (PKSP pro NPH úlohy, ...)
 - **Paměťově intenzivní algoritmy** – čas procesoru strávený nad výpočtem je menší, než čas nutný na přesun dat z paměti do CPU
 - Počet výpočetních operací na přenesený bajt/prvek je příliš malý
 - Typicky a. s lineární výpočetní složitostí, kde data přen. Z paměti do CPU použita jen k -krát, kde $k \geq 1$ je malá konstanta
 - Skalární součin, dynamické programování, Fourierovy transformace
 - O smysluplnosti paralelizace rozhoduje typ úlohy
 - Nutná podmínka = teoretické zrychlení
 - Rozhodující je řád výpočetní složitosti nebo multiplikativní konstanta (u lin. Složitosti)
- **Zdroje neefektivity OpenMP kódů**
 - **Nevyvážená výpočetní zátěž** pro jednotlivá vlákna – bariéra → čekající vlákna → nevyužitá jádra
 - Příliš **těsná synchronizace** – velký počet bariér/krit. sekcí
 - **Omezený paralelismus** – # iterací $\text{for} < \#$ vláken
 - **Vysoká režie** správy vláken – častá tvorba/zánik, schedule(dynamic)
 - **Významná sekvenční část** – z Amdahlova zákonu
 - **Neefektivní využívání keš paměti** – falešné sdílení, častý zápis
- **Falešné sdílení** – různá vlákna zapisují na různé adresy, které jsou ale natolik blízké, že jsou namapovány do stejného bloku keš paměti
 - U datového paralelismu typické
 - Zabránění vede na protichůdný požadavek, než je požadavek přístupu se třídou 1 u jednovláknových aplikací
- **Snížení dopadu falešného sdílení**
 - Vhodnější rozdělení iterací cyklu nad dostatečně velkým polem mezi vlákna je blokově rovnoměrné – schedule(static)
 - Vhodná **chunk-size** při statickém/dynamickém přidělování bloků iterací vláknům
 - Pole A začíná na adrese dělitelné `cache_line_size`, čili pole A je v paměti zarovnáno stejně jako bloky keše
 - Umělé navýšení velikosti zapisované datové struktury připojením jalové výplně (dummy data)
 - Např. každý prvek pole navýšen na velikost bloku keše + podm. Zarovnání
 - Dobré pro malá sdílená pole, kde má každé vlákno vyhrazené místo pro zápis svého lok. Výsledku velikosti ≤ 1 blok keše

Paralelizace násobení polynomů, násobení matic, řazení

- **Násobení polynomů** – vstup = 2 polynomy A, B, výstup = C = A × B, sekvenční složitost $O(nm)$
 - o Sekvenční algoritmus – for cyklus pro init C, potom dvojitý for cyklus pro $C[i + j] = A[i] * B[j]$
 - o **Paralelizace vnějšího cyklu** – každé vlákno počítá násobení 1 členu 1. polynomu – kritické zápisy
 - Všechna vlákna čtou postupně všechny prvky polynomu B
 - Oblasti v poli C (zápis) nejsou disjunktní – potřeba vzájemného vyloučení (*atomic update*)
 - $C[i + j] += A[i] * B[j]$ uvnitř dvojitého for cyklu
 - o **Paralelizace vnitřního j-cyklu** – vlákna sdílejí index i – najednou čtou A[i] (paralelní vnitřní for cyklus)
 - Paralelní zápis je disjunktní, ale + režie opak. Rozdělení m iterací vnitřního cyklu nově vytvořeným p vlákny + režie synchronizace → $n * T_{bar}$
 - Pro 1 člen jsou spočítány násobky
 - 2. možnost: + použití *omp parallel* před i-cyklem – vlákna se vytvoří jen jednou → v iteracích se rozděluje B → na konci každé iterace bariéra
 - Ale vždy vzájemné zneplatnění keše, neodstranitelné falešné sdílení
 - o **Paralelizace polynomu C** – nerozdělujeme násobení, ale výsledné členy – každý se počítá zvlášť
 - Pouze paralelní čtení, žádné kolize zápisu, potřeba vyvažování výpočetní zátěže
 - Minimum falešného sdílení – vlákna provedou maximum výpočtů lokálně – zápis pouze výsledku do sdílené paměti
- **Násobení hustých matic** – 2 čtvercové $n \times n$ matice A, B, výstup = C = A × B
 - o Sekvenčně – středoškolský algoritmus – n^2 skalárních součinů ř. A a sloupců B – n^3 násobení
 - 2 for cykly pro iteraci přes A a B a třetí for cyklus ($k < n$) pro skalární součin
 - o **Paralelizace i-cyklu** – každý řádek matice C = 1 vlákno, zapisovatelné oblasti disjunktní
 - Pouze 1 implicitní bariéra na konci par. regionu
 - o **Paralelizace j-cyklu s blokovaným přidělením** oblastí – větší synchronizace – $n * T_{barr}$, bez kolizí
 - Pro velké n/p není falešné sdílení
 - Každé vlákno zapisuje do souvislé části daného řádku C o velikosti n/p prvků
 - o **Paralelizace j-cyklu s cyklickým přidělením** iterací – jako předtím synchron. $n * T_{barr}$ + bez kolizí
 - Místo *schedule(static)* je *schedule(static, 1)*
 - Falešné sdílení – různá vlákna zapisují současně do sousedních prvků stejné řádky matice C – do stejného keš bloku
 - o **Přidělení iterací j-cyklu v paralelním regionu** – stejné jako 2 kromě synchronizace
 - Před for($i < n$) *omp parallel* – vytvoření p vláken pouze jednou
 - o **Paralelizace k-cyklu – parallel for schedule(static) reduction(t: s)**
 - Skalární součin paralelní redukcí (1 řádek A a 1 sloupec B) – obrovská režie synchronizace
 - Možnost přidat *fork – join (omp master)* před zapsání C – 1x tvorba vláken, ale rychlejší
- **Násobení řídké matice vektorem** – $y = Ax$ (A $n \times n$, počet nenulových prvků N), algoritmus SpMVM
 - o **Formáty pro uložení řídkých matic**
 - **Souřadnicový (COO)** – A reprezentována 3 poli: indexy řádků nenulových prvků, indexy sloupců nenulových p., hodnoty nenulových p. – pořadí typicky po řádcích
 - **Paralelizace:** 2x parallel for, výpočet jako tmp – přičtení za tím jako atomic update
 - Nepřímá indexace – **falešné sdílení** – neefektivní
 - o Možno, aby víc vláken přičítalo ke stejnému prvku vektoru y současně
 - **Komprimované řádky (CSR)** – A repr. 3 poli: (položky podle indexů ř. a sl.) indexy do pole A.Collnd, od kterých jsou v něm uloženy indexy sloupců nenulových prvků jednotlivých řádků A; indexy sloupců nenulových prvků; hodnoty nenulových prvků A
 - **Paralelizace:** *Parallel for schedule([static[, c]] | dynamic[, X])* před 1. for
 - o *Schedule(static)* – řádky distribuovány blokově
 - Nevyvážená zátěž, falešné sdílení sousedních vláken
 - o *Schedule(static, 1)* – řádky přidělovány cyklicky
 - Taký nevyvážená zátěž, horší falešné sdílení (všechna vlákna)
 - o *Schedule(dynamic, X)* – dyn. přidělování – lepší vyváženost, ale vyšší režie
 - Matice s nepravidelným rozdělením nenulových prvků

- **Vyvažující metoda** – matice v CSR formátu na p řádkových pásů (=bands), které mají přibližně stejný počet nenulových prvků, každé vlákno 1 pás
 - Pomocí `omp parallel` a `omp barrier` (iterace po pásech)
- **Paralelizace QuickSortu**
 - **QuickSort** – varianta Lomuto: pivot → třídění prvků → L a R
 - Funkční paralelismus → *omp task* – rekurzivní volání na L a R
 - Velké množství malých tasků → velká režie, zlepšení:
 - Tail call optimization – odstranění koncové rekurze (tail recursion), resp. Nahrazení 1 ze 2 rekurzivních volání iterací
 - Prahování – zavedení prahu počtu prvků pro vytváření nových OpenMP úloh
 - Paralelizace rozdělování řetězce – varianta Hoare
 - **QuickSort – varianta Hoare**
 - Pole se souběžně prochází zleva i zprava a přitom se porovn. navštívené prvky s pivotem
 - L menší než pivot a R větší rovno pivot → oba na správných pozicích; L i R menší než pivot → L je správně; L i R větší než pivot → R je správně; L větší a R menší → prohození
 - Iterace, až se průchody zleva a zprava potkají = neutralizace – v každé i. neutr. min. 1 prvek
 - **Rozdělení řetězce pro rekurzi lze paralelizovat** – indexy průchodů = sdílené proměnné
 - Každé vlákno chce získat unikátní hodnoty těchto proměnných tak, aby mohlo k rozdělení přispět svou prací na disjunktních dvojicích prvků pole
 - Na konci zbyde max. 1 špatně zařazený prvek – levný a rychlý úklid
 - Vyžaduje aktivovaný vnořený OpenMP paralelismus - *omp_set_nested(1)*
 - Vede na strom rekurzivních volání → par. vlákna se rozvětvují do par. vláken
 - Operace s indexy pomocí *omp atomic capture*
- **Paralelizace MergeSortu: MergeSort** → standardní binární rozdělování, problém je s Merge fází
 - **Klasický task** paralelismus je **moc pomalý** – slučování moc krátkých částí, falešné sdílení
 - Zlepšení: Prah počtu prvků; místo 2 nových úloh pouze 1 pro levou část → pravá zpracována sekvenčně; paralelizace Merge
 - **Paralelizace Merge** – u sekvenčního průchodu 2 seřazenými poli zleva doprava a porovn. počátečních prvků → menší do výsledního pole = 2-cestné slučování
 - Binární matice, kde seř. pole A jsou řádky a B sloupce, pokud $A[i] > B[j]$, pak $M[i,j]=0$, jinak 1
 - Tlustá lomená čára dělí oblast 0 a 1
 - M proložíme $p-1$ vedlejšími diagonálami v diagonální matici vzdálenosti $n/2p$ od sebe → označíme průsečíky s hranicí $(x_0 \dots x_p)$ → oblasti jsou ohraničené pomocí průsečíků
 - Dělení na pásy podle # CPU, průsečíky = oblast čísel rozdělených podle 1 ekviv. počtu
 - Budou se všemi vlákny zapisovat do seříděného pole
 - Každé vlákno pak sekv. sloučí bloky A_i, B_i , na konci zřetězení slouč. pos. od každého vlákna
 - $p - 1$ vláken počítá $p - 1$ průsečíků za $O(\log n)$, možné falešné sdílení, rychlejší p -cestné
- **Paralelizace p -cestného MergeSortu – PMWMS = parallel multi-way MergeSort**
 - p vláken rozdělí vstupní neseř. pole o velikosti n na p částí velikostí n/p , vlákna seřadí sekv. svoje s
 - Po bariéře každé vlákno $i > 0$ vypočte vektor svých p tzv. **rozdělovačů** *split_vec[i]* = p indexů do p seřaz. částí pole $S[p][n/p]$ takových, že součet délek disjunktních úseků určených v $S[p][n/p]$ dvěma po sobě jdoucími vektory rozdělovačů *split_vec[i]* a *split_vec[i+1]* má vel. n/p
 - Po bariéře vlákno i vyřízne podle svých p rozdělovačů svých p disjunktních seřaz. úseků délky n/p
 - Úseky sloučí pomocí sekvenčního p -cestného slučování do seřazeného pole velikosti n/p
 - Výsledný seřazený úsek o velikosti n/p vloží na připravené disjunktní místo $B[i]$ výstupního pole
 - **Splitters_by_Rank** – vlákno vezme sdílené pole p seřazených polí $s[0] \dots s[p-1]$ a pořadí rank svého 1. čísla na výstupu a vygeneruje pole p rozdělovačů těchto polí takových, že počet prvků vlevo od nich sečtený přes všechna pole $S[i]$ se rovná přesně rank
 - Po seřazení jednotlivých částí a výpočtu oddělovačů musí být bariéry
 - **Části:** Sekvenční řazení n/p čísel, $\log n$ provedení p hledání v polích velikosti n/p , sekvenční p -cestné slučování p polí celkové délky n/p
 - N je velké, p malé – bude dominovat první člen

18. Programování nad distribuovanou pamětí, programový model MPI (vícevláknové procesy, komunikátory, 2-bodové blokující a neblokující komunikační operace, kolektivní operace), paralelní násobení hustých matic, paralelní mocninná metoda.

NI-PDP

- MPI = systém zasílání zpráv mezi procesy
- Komunikace procesů/vláken
 - o **OpenMP** – pomocí čtení/zápisů z/do sdílené paměti, podpora pro redukci
 - o **MPI** – procesy nesdílí paměť – komunikace zasíláním zpráv, všechny proměnné privátní
 - Redukce pro všechny procesy najednou: **MPI_Allreduce**
- Využití sdílené paměti:
 - o **Jen MPI** – na každém jádru 1 či několik MPI procesů – nedělí se o vlákna
 - o **MPI + OpenMP** – výpočetní uzel → MPI proces(y) → pomocí OpenMP dělení na několik vláken, běžících na jádrech
 - o **Hybrid** – 1 OpenMP vlákno na jádro
- Kombinace MPI + OpenMP
 - o Inicializace → **MPI_Init_thread**: Vrací v proměnné zaručenou míru spolupráce MPI s vlákny
 - o Požadovaná míra spolupráce MPI s vlákny:
 - **MPI_THREAD_SINGLE** – pouze MPI, procesy se nedělí na vlákna
 - **MPI_THREAD_FUNNELED** – vícevláknové procesy s omezením, že pouze hlavní vlákno může zavolat funkce MPI = jednoportový model
 - **MPI_THREAD_SERIALIZED** – vícevláknové procesy s om., že v daném okamžiku smí funkce MPI volat pouze 1 vlákno (volání MPI funkcí je kritická sekce) = jednoportový model
 - **MPI_THREAD_MULTIPLE** – vícevláknové procesy, kde všechna vlákna mohou volat funkce MPI bez omezení = všeporťový model
- **Komunikátor** – určuje množinu procesů, v rámci níž probíhá komunikace
- **Intra-komunikátor** – asociovaný s konkrétní skupinou procesů
- **MPI_COMM_WORLD** – předdef. Intra-kom. Pro všechny MPI procesy
- **Inter-komunikátor** – 2 různé skupiny procesů
- **MPI_Comm_rank** – číslo procesu, **MPI_Comm_size** – počet procesů
- **Komunikační MPI operace**: 2-bodové (komunikace mezi 2 procesy), kolektivní – (komunik. mezi všemi p.)
- **Základní 2-bodová kom.** – zdrojový p. - **MPI_Send** – určí cílový p., cílový p. - **MPI_Recv** – určí zdrojový p.
- **Blokující komunikační operace** – příslušná MPI funkce je ukončena teprve po dosažení určitého stavu dané komunikační operace
- **MPI_Send**
 - o Buf – ukazatel na posílaná data
 - o Count – počet posílaných položek
 - o Datatype – dat. Typ posílaných dat
 - o Dest – číslo cíl. Procesu
 - o Tag – značka zprávy
 - o Comm – MPI komunikátor
- **MPI_Recv**
 - o Source – číslo zdrojového procesu
 - o Status – ukazatel na stavový objekt
 - o Zbytek stejně jako u Send
- **Typ přenášených dat**
 - o Parametr datatype – typ **MPI_Datatype**
 - o Základní datové typy - **MPI_CHAR**, **MPI_INT**, ...
 - o Složitější - **MPI_Type_create_struct**
- **Množství přenášených dat** – Lze najednou posílat víc prvků, ale stejného dat. Typu a uložené za sebou v paměti, Parametr count
- **Zdrojový a cílový proces** – Cíl – parametr dest, Zdroj – parametr source

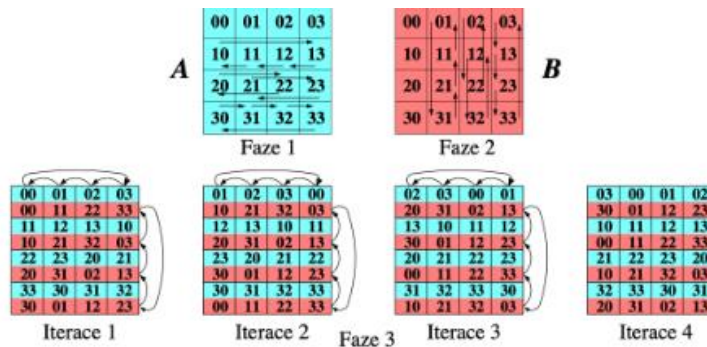
- Přijetí od 1 konkrétního zdroje × od libovolného zdroje - *MPI_ANY_SOURCE*
- **Značky přenášených dat – Tag** – rozeznání sémantického významu zpráv
 - Přijetí konkrétní značky × libovolné značky - *MPI_ANY_TAG*
- **Stavový objekt** – Proměnná typu *MPI_Status*
 - Můžeme ignorovat – *MPI_STATUS_IGNORE*
 - Struktura s položkami:
 - *MPI_SOURCE* – číslo zdroj. Procesu zprávy
 - *MPI_TAG* – značka přijaté zprávy
 - Pomocí *MPI_Get_count* velikost zprávy
- ***MPI_Send*** je blokující – ukončena až když lze modifikovat vstupní buffer
 - Realizuje standardní mód – návrat z funkce nastane, když jsou data:
 - Odeslána cílovému procesu
 - Překopírována do dočasného systémového bufferu pro pozdější odeslání
 - Kvůli odesílání je to nelokální operace
- ***MPI_BSend*** – realizuje Buffered mode, návrat zaručeně nezávisí na připravenosti příjemce přijímat data, lokální operace
 - Pokud příjem nebyl iniciován, MPI musí odesílaná data uložit do bufferu, který si musí uživatel předtím připravit pomocí *MPI_Buffer_attach*
- ***MPI_SSend*** – Synchronous mode – není návrat, dokud není inicializace přijetí dat, nelokální operace
- ***MPI_RSend*** – Ready mode – pokud při volání není init příjmu, vrátí chybu, nelokální operace
- **Standardní mód** – MPI rozhodne, jestli použít Buffered/Synchronous – Uživatel to neovládá
 - MPI – stanovisko, že korektní MPI program není na systémovém bufrování závislý
- Tyhle sendy jsou blokující ve smyslu, že po návratu z nich můžeme buffer odesílaných dat přepsat
- Recv je blokující ve smyslu, že po jejich ukončení jsou přijatá data uložena v bufferu a lze je číst
- Neblokující funkce ***MPI_Isend, MPI_Ibsend, MPI_Issend, MPI_Irsend*** iniciují odeslání dat a skončí
 - ***MPI_Irsend*** může začít, až když příjemce iniciuje příjem, ostatní libovolně
- Buffer vstupních dat nelze modifikovat, dokud není dokončení komunikační operace explicitně otestováno
- Neblokující funkce ***MPI_Irecv*** iniciuje příjem dat
- Buffer není možné použít, dokud není dokončení operace příjmu dat explicitně otestováno
- Všechny neblokující funkce mají dodatečný parametr – ukazatel na proměnnou typu ***MPI_Request***
 - Vstupní arg. Funkcí, které slouží pro testování/čekání na dokončení těchto komunikačních operací
 - **Testování dokončení - *MPI_Test***
 - Neblokující, vrátí okamžitě *MPI_SUCCESS* nebo chybu
 - **Čekání na dokončení - *MPI_Wait***
 - Blokující, vrátí až tehdy, když jsou data skutečně obdržena
- U neblokujícího příjmu **stavový objekt** až z funkcí Test/Wait, NE z Irecv
 - Parametry *MPI_Request* a stavový objekt *MPI_Status*
- Neblokující operace důležité, bo umožňují překrývání volání komunikačních párů – není nutná serializace
- ***MPI_Testany/Waitany*** – dokončení libovolné operace
- ***MPI_Testall/Waitall*** – dokončení všech operací z množiny
- **Komunikační módy** neblokujících operací
 - Vrací okamžitě nezávisle na splnění dané podmínky
 - Na splnění podm. závisí návrat z funkcí čekání na dokončení neblokujících operací *MPI_Wait, ...*
- ***MPI_Iprobe, MPI_Probe*** – testují příchod zprávy, aniž by zpráva byla přijata
 - (*int source, int tag, MPI_Comm comm, int * flag, MPI_Status * status*)
- ***MPI_IProbe*** – neblokující lokální funkce
 - *flag = true*, pokud existuje zpráva, kterou lze přijmout a která odpovídá parametrům *source, tag, comm*
 - Pak vrátí argumentu *status* stejnou hodnotu, jakou by vrátila operace *MPI_Receive*
 - Jinak vrátí *flag = false* a *status* je nedefinován
- *source* může být *MPI_ANY_SOURCE* a *tag* může být *MPI_ANY_TAG*
- Sondovaná zpráva nemusí být přijata po té, co byla sondována a danou zprávu lze tedy sond. opakovaně

- **MPI_Probe** – blokující nelokální funkce
 - o Vráti až poté, co existuje zpráva, kterou lze přijmout a která odpovídá parametrům *source, tag, comm* a ve výstupním argumentu *status* vrátí stejnou hodnotu, jakou by vrátila operace *MPI_Receive*
- **Požadavky na implementaci MPI_Iprobe a MPI_Probe** – měly by garantovat následující:
 - o Je-li zavolán *MPI_Probe* jedním procesem a jiný proces zavolá *Send* s kompatibilními parametry, pak se volání *MPI_Probe* úspěšně vrátí KROMĚ případů, kdy zprávu přijme konkurenční funkce *MPI_Receive*
 - o Pokud proces aktivně čeká pomocí *MPI_Iprobe* a odpovídající zpráva byla vyslána, pak volání *MPI_Iprobe* v konečném čase vrátí *flag = True*, pokud
 - Kompat. zprávu nepřijme konkurenční *MPI_Receive* provedená jiným vl. téhož procesu
 - Taková zpráva nebyla sond. konkurenční operací *Probe* provedenou jiným vl. téhož proc.
- Volání *Iprobe/Probe* det. zprávu, kterou by byla přij. ve stejném místě vol. f. *MPI_Recv* se stejnými arg.
- Ve vícevláknových procesech je seznam příchozích zpráv sdílen a může docházet k soupeření vláken o přijetí zpráv kompatibilních s těmi, které vysondovaly předchozími voláními *Probe/Iprobe*
- **Sondování s rezervací pro budoucí Receive** – pro zajištění větší korektnosti a efektivnosti soupeření
 - o Neblokující *MPI_Improbe(source, tag, comm, flag, MPI_Message * message, status)*
 - Oproti *Iprobe* vrátí v případě, že zpráva existuje, v hodnotě argumentu *message* message handle na vysondovanou zprávu
 - o Message je vstupem volání funkce *MPI_Mrecv(buf, count, datatype, MPI_message * message, status)*
 - Před návr. z volání *MPI_Mrecv* se message handle reset. Na *MPI_MESSAGE_NO_PROC*
 - Volání *MPI_Mrecv* s takovouto hodnotou argumentu message nic nepřijme
- Využití funkcí pro **testování příchodu zpráv**
 - o Příchod „volitelných“ zpráv – předčasné uk. výpočtu při nalezení optim. řešení jiným procesem
 - o Příjem zprávy neznámé velikosti – zjištění velikosti zprávy pomocí *MPI_Probe* a *MPI_Get_count*, alokace bufferu, příjem dat pomocí *MPI_Recv*
- MPI neposkytuje mechanismy pro řešení chyb komunikačního systému
- Chyby způsobené voláním MPI funkce s chybným argumentem, nedostatek zdrojů, ...
- **Návratová hodnota MPI funkce** – úspěch/neúspěch
 - o Úspěch → *MPI_SUCCESS*, neúspěch → chybový kód
- **Chybový kód** = základ pro obsluhu chyby (=error handler) dané MPI funkce, která se při výskytu chyby zavolá před návratem
- 3 předdefinované obsluhy chyb:
 - o **MPI_ERRORS_FATAL** – chyba → násilně ukončen celý pr., k návratu chyb. funkce nedojde
 - Procesy interně zavolají *MPI_ABORT*
 - Implicitně asociovaná s *MPI_COMM_WORLD*
 - o **MPI_ERRORS_RETURN** – neukončí program, vrátí chybový kód funkce
 - Stav MPI výpočtu není po chybě MPI standardem definován
 - Pro diagnostiku stavu a výpis chybového hlášení
 - o **MPI_ERRORS_ABORT** – násilné ukončení procesů spoj. s daným komunikátorem, ale ne všech
- Funkce pro **vytvoření kódu obsluhy chyby**, její navázání na komunikátory, testování vazeb a jejich zrušení:
 - o *MPI_Comm_create_errhandler, MPI_Comm_set_errhandler, MPI_Comm_get_errhandler, MPI_errhandler_free* → uživatelsky naprogramované volání obsluhy

Násobení hustých matic

- **Násobení hustých matic:** Předpokládám klasický školní algoritmus na násobení matic a blokově-šachovnicové mapování matic
- **Naivní algoritmus:** Každý procesor potřebuje odpovídající submatice pomocí AAG
 - o Na závěr se provede lokální vynásobení, časová náročnost: $\Theta(N/p \cdot (\sqrt{p} + \sqrt{N}))$, paměťově neefektivní (nevleze se to do paměti jednoho procesoru)

- Cannonův systolický algoritmus:



- Rozděluje výslednou matici C do které zapisují mezivýsledky na P pod-matic, podle toho P procesů – algoritmus zabere \sqrt{P} kroků (matice A a B mají každá celkem P buněk)
- Krok 1:
 - Cyklický posuv řádků A a sloupců B (*MPI_Sendrecv*)
 - Matice A: posouvám řádky doleva – 1. řádek o 0, druhý řádek o 1, 3. řádek o 2 atd.
 - Matice B: posouvám sloupce nahoru – 1. sloupec o 0, 2. sloupec o 1 atd.
 - Proces P_{ij} spočítá součin $A_{ij} \cdot B_{ij}$ – jedna buňka matice C
- Krok 2:
 - Cyklický posuv A a B (tentokrát vždy o jedno políčko)
 - Každý řádek A o 1 doleva, každý sloupec B o 1 nahoru
 - Proces P_{ij} updatuje příslušnou buňku C (takže se zase násobí buňky A a B a potom se to k C přičte)
- Krok 2 pak opakuji, dokud se celkově neposunu o \sqrt{P} políček (udělá se \sqrt{P} -krát podle velikosti matice $\sqrt{P} \cdot \sqrt{P}$)
- Výsledná složitost

$$T_{\text{Cannon}}(N, p) = O(t_s \sqrt{p}) + O\left(\frac{N}{\sqrt{p}} t_m\right) + O\left(\frac{\sqrt{N^3}}{p}\right)$$

- Kvůli těm točkám jsou na to optimální 2D toroidy

- Foxův algoritmus – Broadcast-Multiply-Roll:



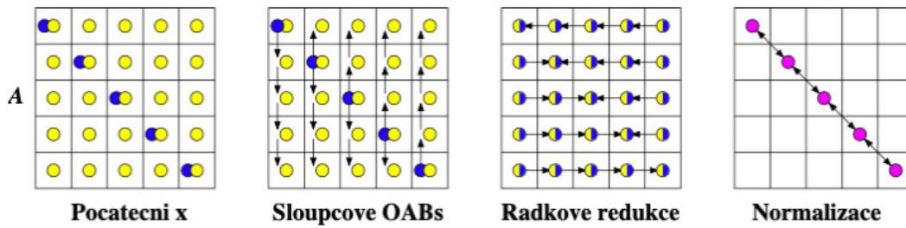
- Zase P procesů a \sqrt{P} kroků, každý proces p zpracovává jedno políčko – 1 součin $A_{ij} \cdot B_{ij}$
- Krok 1:
 - Nastavíme $k = 0$ (to je jako nějaké číslo posuvu)
 - Broadcast (jeden všem) od políčka A_{ii} (na diagonále) na každém řádku i (celému řádku) – *MPI_BCast*
 - Podle toho se inicializuje matice C – P různých hodnot B se násobí \sqrt{P} hodnotami A v P políčkách (procesech): $A_{ii} \cdot B_{ij} = C_{ij}$
- Krok 2:
 - Vybírám nová políčka A k broadcastu – „posun doprava“ – cyklický posuv o 1 (na obrázcích modře mezi iteracemi), nastavuju $k += 1$
 - Broadcastovat se budou políčka $A_{i, (i+k) \bmod \sqrt{P}}$
 - Sloupce B se posunou o 1 nahoru
- Krok 2 se zase opakuje do kroku \sqrt{P}
- Časová složitost, škálovatelnost podobná jako u Cannonova algoritmu

- **Násobení matice vektorem**
 - o **Řádkové mapování** – každý proces má jeden řádek a příslušný prvek vektoru x
 - Broadcastuje ho všem
 - Všichni mají nově celý vektor x a počítají 1 y prvek – rychlost závisí na broadcastu
 - o **Sloupcové mapování**
 - Každý p. má sloupce a příslušný prvek z x – může spočítat vlastní příspěvek do vektoru y , který je zatím ve formě matice bez operace sčítání
 - Máme matici sčítanců v řádcích, co jsou po sloupcích v procesech
 - Řádková redukce přes několik procesů s operací sčítání – složitost jako u řídké matice
 - o **Šachovnicové mapování** – každá buňka je proces
 - Každý sloupec potřebuje prvek x se stejným indexem, jako je sloupec
 - Jako při řádkovém – distribuce dle sloupce
 - Jako při sloupcovém – tvorba příspěvků a pak redukce s operací sčítání

Paralelní mocninná metoda

- **Mocninná metoda:** hledá iterativně největší vlastní číslo, vhodné pro velmi řídkou matici, využití: Google PageRank
 - o Opakovaně se násobí matice vektorem: $y = Ax$
- **Algoritmus:**
 1. Inicializace – vytvořím nenulový počáteční vektor (typicky $x = (1, 1, 1, 1, \dots)$)
 2. Násobení matice-vektor (MVM) – vynásobím A vektorem x , vznikne vektor $y = Ax$ (nějaký algoritmus pro násobení řídké matice vektorem)
 3. Normalizace – spočtu normu α vektoru y
 4. Aktualizace vektoru – nahradím x normalizovaným $y = x/\alpha$ (paralelní redukce)
 5. Vyhodnotíme kritérium konvergence, pokud není splněno, opakuji od kroku 2
 - Pokud je splněno, α je aproximací dominantního (největšího) vlastního čísla
- **Implementace v MPI:**
 - o Předpokládáme řídkou matici, předem neurčená struktura
 - o Procesory provádí lokální násobení, dílčí výsledky redukují (MPI_Allreduce)
 - o Máme komunikační skupiny pro řádky, sloupce a diagonálu
 - **MPI_Bcast** – provede sloupcový broadcast
 - **MPI_Reduce** – zpětně namapuje hodnoty x na diagonálu
 - **MPI_Allreduce** – zjistí normalizaci přes diagonálu
- **Náhodné mapování matice:**
 - o Každý proces potřebuje celý vektor x a y pro výpočet své části y
 - o Na konci každé iterace je **MPI_Allreduce** pro synchronizaci vektoru x a y mezi procesy
 - o Po provedení algoritmu má každý proces celý vektor y a α
 - o Složitost: $O(n)$ paměť, kde $n = \sqrt{N}$
- **Řádkové mapování matice:**
 - o Každý procesor potřebuje celý vektor x , ale vektor y si již můžou rovnoměrně rozdělit
 - o Matice rozdělena do p horizontálních pásů velikosti n/p
 - o Získání vektoru x pomocí **MPI_Allgather**, potom každý proces lokálně vypočte svou část y
 - o Normalizace se provede lokálně a vektor y se vyřeší normalizovaným výsledkem
 - o Rychlejší (nepotřebujeme kopírovat y do x),
 - o Složitost: $x - O(n)$, $y - O(n/p)$
- **Šachovnicové mapování matice:**
 - o Procesy tvoří virtuální 2D mřížku $M(n, n)$
 - o Každý procesor potřebuje jen část vektoru x (menší paměťové nároky)
 - o Vektor x je namapován na hlavní diagonálu
 - o Procesy na diagonále posílají hodnoty vektoru po sloupcích (sloupcový broadcast)
 - o Paralelně všechny procesory spočítají svou část submatice (lokální výpočty)
 - o Provede se nejdřív redukce zpět na diagonálu, a potom redukce diagonály

- Složitost: $x - O\left(\frac{n}{p}\right)$, $y - O\left(\frac{n}{p}\right)$



- Rozdělení komunikátorů:
 - Potřebujeme provést paralelní redukci jen ve virtuálních řádcích matice procesů
 - **MPI: *MPI_Comm_split*** – rozdělí komunikátor podle pole „barev“ (řádková souřadnice)
 - Redukci tak můžeme provádět ve všech řádcích nezávisle na sobě
 - ***MPI_Comm_split(MPI_Comm, int color, int new_rank, MPI_Comm * newComm)***
 - Potřebujeme ale taky komunikátor pro diagonální procesy, nejefektivnější časově i paměťově


Kolektivní operace

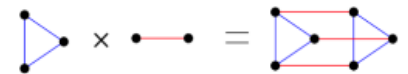
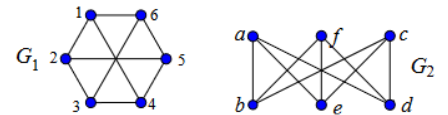
- Operace prováděné nad všemi MPI procesy – blokující i neblokující verze
 - ***MPI_Bcast*** – OAB (one to all broadcast) – root proces vysílá tutéž zprávu všem ostatním
 - ***MPI_Gather(v)*** – AOG (all to one gather) – root proces posbírání data ode všech, včetně sebe
 - Stejná velikost informací u všech, jinak s variabilitou funkce ***MPI_Gatherv***
 - ***MPI_Allgather*** – AAG/AAB (all to all gather) – všichni ode všech, včetně sebe, sbírají data
 - ***MPI_Scatter(v)*** – OAS (one to all scatter) – distribuuje části včetně sebe samého
 - Scatter na rozdíl od broadcastu neposílá všem to samé, ale každému různý chunk informace
 - ***MPI_Alltoall(v)*** – AAS (all to all scatter) – symetrická distribuce všichni všem

19. Přímé ortogonální a hyperkubické propojovací sítě paralelních počítačů (definice, vlastnosti, vnořování).

NI-PDP

Základní definice a vlastnosti

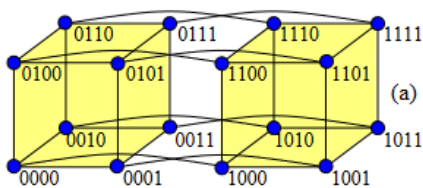
- **Přímé propojovací sítě** – lze je popsat souvislými grafy
 - o Vrchol = výpočetní uzel = procesory, lokální paměť a směrovač
 - o Hrana = komunikační linka
 - o Nepřímé sítě – tvořeny přepínači
 - o Přímé sítě vlastně propojují jednotlivé směrovače mezi sebou
- **Vlastnosti**
 - o **Stupeň uzlu** $\deg_G(u)$ = počet sousedů uzlu u
 - o **Bisekční šířka** $bw_e(G)$ = velikost nejmenšího hranového řezu grafu na 2 poloviny
 - o **Uzlová symetrie** = pro libovolnou dvojici uzlů u a v existuje automorfismus, který zobrazí u na v (obrázek)
 - o **Souvislost** = minimální počet uzlů/hran, jejichž odebrání způsobí rozpojení grafu G
 - o **Bipartita** = existuje obarvení vrcholu dvěma barvami tak, že koncové vrcholy každé hrany mají odlišnou barvu
 - o **Kartézský součin** – k. s. množin vrcholů, a hrana vede tehdy, pokud vedla původně v jednom nebo druhém grafu
 - o **Regularita** – stupeň každého uzlu je roven konstantě k
- **Požadavky na propojovací sítě**
 - o **Malý a konstantní stupeň uzlu** – technologický požadavek
 - **Řídké grafy** = stupeň je omezený konstantou, počet hran $\Theta(N)$
 - **Hustá topologie** = stupně uzlů jsou rostoucí funkcí
 - Konstantní stupeň nutnou podmínkou pro uzlovou symetrii
 - o **Malý průměr a malá průměrná vzdálenost** – pro snížení režie komunikačních operací
 - V protikladu s požadavkem na nízký stupeň uzlů
 - Průměr N -uzlové sítě s konstantním stupněm je $\Omega(\log N)$
 - o **Symetrie** – zjednodušený návrh algoritmů
 - o **Škálovatelnost**
 - **Inkrementálně škálovatelná topologie** = definovaná pro jakékoli N
 - **Částečně škálovatelná topologie** = není definovaná pro jakékoliv N
 - **Efektivně škálovatelná topologie** = pro vytvoření $(N + k)$ -uzlové instance z N -uzlové instance je třeba odebrat pouze $O(k)$ hran a pak $O(k)$ hran přidat
 - Chceme-li zmenšit nebo zvětšit danou síť o k uzlů, musíme v původní síti přepojit pouze $O(k)$ uzlů
 - 2D mřížka: částečně škálovatelná, ne efektivně: 
 - o Obrázek – zvětšení uzlu o 1, ale přepojení 4 hran
 - o **Hierarchická rekurzivita** – graf má h. r. topologii, jestliže obsahuje menší instance sebe sama
 - Množina stejně definovaných grafů různých dimenzí, kde nižší dimenze jsou podgrafy vyšších dimenzí
 - Obvykle částečně škálovatelné
 - Dobrá pro realizaci obvodů a induktivní návrhy a mapování paralelních rekurzivních algo.
 - o **Vysoká souvislost a odolnost vůči poruchám** – kvůli spolehlivosti a robustnosti sítí
 - V případě výpadku uzlů nebo spojů by měla síť nabídnout náhradní krátké spoje
 - Chceme malý chybový průměr a malou chybovou průměrnou vzdálenost
 - o **Velká bisekční šířka** – pro algoritmy typu binární rozděl a panuj
 - Rozdělení problému na dva stejně velké podproblémy
 - Rekurzivní řešení v obou polovinách paralelně, možná výměna mezivýsledků



- Sloučení výsledků z obou polovin do konečného výsledku
- Podpora pro směřování a kolektivní komunikační operace – chceme minimální směřování
- Vnořitelnost jiných a do jiných topologií
 - Vnoření = má-li komunikační graf par. algoritmu jinou strukturu, odlišnou od topologie propojovací sítě, je třeba efektivně zobrazit graf procesů do topologie sítě
- Existence hamiltonovských kružnic či cest a existence 2-barvení
 - Hamiltonovská kružnice = vnoření N -uzlové kružnice – speciální případ vnořitelnosti
 - Kritická vlastnost – zjednodušení návrhu algoritmů, kde jsou procesory označeny čísly 1 ... p a komunikace je posun dat ve směru indexování, např. třídící algoritmy

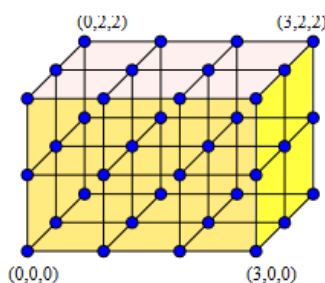
Striktně ortogonální (mřížkové) topologie

- n -rozměrná binární hyperkrychle Q_n



$V(Q_n) = \mathcal{B}^n$	$ V(Q_n) = 2^n$
$E(Q_n) = \{(x, \text{neg}_i(x)); x \in V(Q_n), 0 \leq i < n\}$	$ E(Q_n) = n2^{n-1}$
$\phi(Q_n) = n$	$\deg(Q_n) = \{n\}$
$\text{bw}_e(Q_n) = 2^{n-1}$	

- Uzly = binární řetězce
 - Sousední uzly = liší se právě v 1 bitu → každý uzel má n sousedů a hyperkrychle je **regulární**
- Stupeň není konstantní, roste logaritmicky – **hustá topologie**
- Vzdálenost dvou uzlů je Hammingova vzdálenost řetězců (počet odlišných stejnohlých bitů)
- Počet uzlů vzdálenosti i od libovolného uzlu vždy $\binom{n}{i}$ → průměrná vzdálenost $n/2$
- Uzlově i hranově **symetrická**
- Částečná škálovatelnost – velikost pouze mocniny 2
- Efektivně škálovatelná – 2 podkrychle Q_{n-1} apod. – dobré pro řešení problému přidělování procesorů v hyperkubickém počítači
- Optimální souvislost (rovna stupni uzlu – n), největší možná bisekční šířka bw_e
- Vyvážený bipartitní graf, hamiltonovský graf
- Základní minimální směrovací algoritmus – **e-cube**
 - Cesta (u, v) : porovnají se bitové řetězce a dimenze hran, ze kterých se cesta skládá, odpovídají zprava doleva souřadnicím, ve kterých se u a v liší
- Simuluje efektivně téměř jakoukoli jinou známou topologií – většina topologií je do ní **optimálně vnořitelná**
- Její hamiltonovské kružnice = **Grayovy kódy**
- Testovací architektura pro paralelizaci problémů
- n -rozměrná mřížka o velikosti stran $z_1 \dots z_n, M(z_1 \dots z_n)$

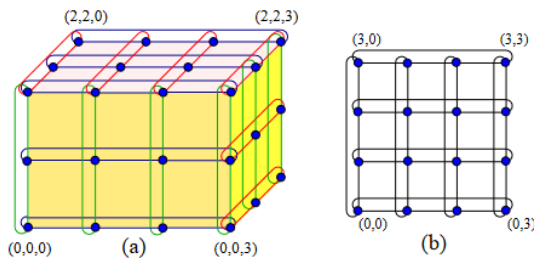


$V(M(\dots)) = \{(a_1, a_2, \dots, a_n); 0 \leq a_i \leq z_i - 1 \forall i \in \{1, \dots, n\}\}$	
$E(M(\dots)) = \{(\langle \dots, a_i, \dots \rangle, \langle \dots, a_i + 1, \dots \rangle); 0 \leq a_i \leq z_i - 2\}$	
$ V(M(\dots)) = \prod_{i=1}^n z_i$	$ E(M(\dots)) = \sum_{i=1}^n (z_i - 1) \prod_{j=1, j \neq i}^n z_j$
$\phi(M(\dots)) = \sum_{i=1}^n (z_i - 1) = \Omega(\sqrt[n]{ V(M(\dots)) })$	
$\deg(M(\dots)) = \{n, \dots, n + j\}, j = \{z_i; z_i > 2\} $	
$\text{bw}_e(M(\dots)) = \begin{cases} (\prod_{i=1}^n z_i) / \max_i z_i & \text{jestliže } \max_i z_i \text{ je sudé,} \\ \Omega((\prod_{i=1}^n z_i) / \max_i z_i) & \text{jinak.} \end{cases}$	

- Sestavena **kartézským součinem** mřížek nižších dimenzí:
 - $M(z_1 \dots z_n) = M(z_1 \dots z_{n-1}) \times M(z_n) = M(z_1) \times \dots \times M(z_n)$
- 1-D mřížka je vlastně řada procesorů
- Uzly mřížky jsou značeny n -znakovými k -árními řetězci
- Dva uzly jsou sousední, právě když se liší v jedné souřadnici o jedničku
- **Není regulární ani uzlově symetrická** – stupeň uzlu závisí na jeho poloze
- **Nejsou hranově symetrické** (kvůli tomu tak debilní vzorce)

- 2-D a 3-D mřížky – topologie s **velkým průměrem**
- **Částečně škálovatelná** (líp než hyperkrychle) – N součin n čísel větších než 1
- Je **hierarchicky rekurzivní** – obsahuje podmřížky stejné dimenze, ale menších délek stran
- Má **optimální souvislost** – počet disjunktních cest mezi 2 uzly je roven minimu ze stupňů koncových uzlů
- **Bipartitní**, ne nutně vyvážená, vždy má hamiltonovské cesty a jestliže má aspoň 1 strana sudou délku, má i hamiltonovskou kružnici
- Algoritmus pro minimální směrování – **dimenzionálně uspořádané směrování**
 - Při konstrukci cesty mezi lib. 2 uzly se hrany směřjí používat pouze v jediném pořadí dimenzí

- **n -rozměrný toroid o velikosti stran $z_1 \dots z_n, T(z_1 \dots z_n)$**

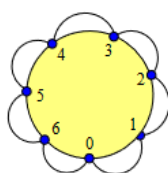
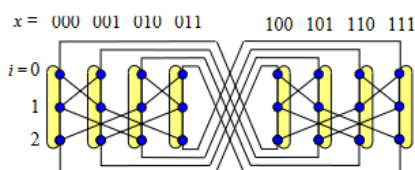


$V(T(\dots)) = V(M(\dots))$	
$E(T(\dots)) = \{ \langle (\dots, a_i, \dots), (\dots, a_i \oplus_{z_i} 1, \dots) \rangle; 0 \leq a_i < z_i \}$	
$ E(T(\dots)) = n \times \prod_{i=1}^n z_i$	$\phi(T(\dots)) = \sum_{i=1}^n \lfloor z_i/2 \rfloor$
$\deg(T(\dots)) = \{2n\}$	$\text{bwe}(T(\dots)) = 2 \text{bwe}(M(\dots))$

- „**zabalená mřížka**“ – od mřížky se liší jen tak, že každá **lineární řada je uzavřena do kružnice** přidáním hrany (=obalující hrana), která spojí první a poslední uzel
 - Jednorozměrný toroid je **kružnice** = prstenec
- Dva uzly jsou sousední, právě když se liší v jedné souřadnici o jedničku **modulo velikost strany** v dané dimenzi
 - Díky tomu je toroid **regulární** (stupeň $2n$) a **uzlově symetrický**
 - Automorfismus = přeložení
- Průměr poloviční oproti stejné velké mřížce
- Částečně škálovatelné a hierarchicky rekurzivní, dekompozice na kartézský součin stejně jako u mřížek: $M(z_1 \dots z_n) = M(z_1) \times \dots \times M(z_n)$
- Algoritmus pro minimální směrování – dimenzionálně uspořádané směrování
 - Kvůli kružnicím může dojít k zablokování
- Bipartitní, pokud jsou všechny délky stran sudé (kvůli kružnicím), hamiltonovský
- Komerčně populární pro masové paralelní počítače

Hyperkubické topologie

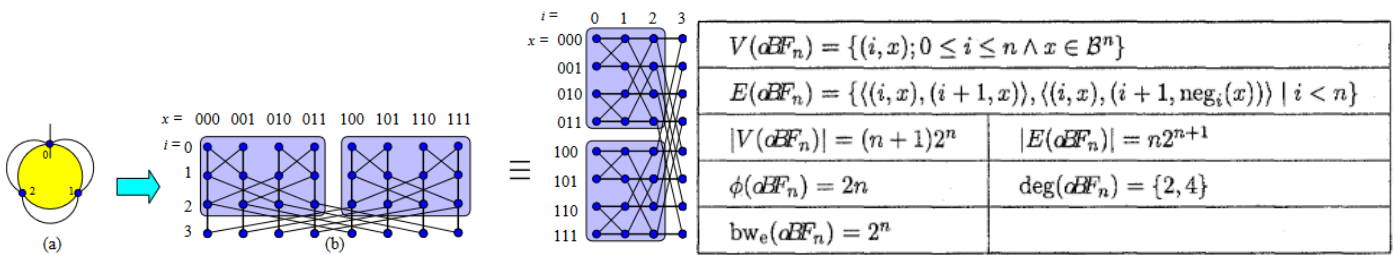
- Nedostatek hyperkrychle – logaritmicky rostoucí stupeň uzlu
 - Proto topologie odvozené z hyperkrychle, které mají její dobré vlastnosti, ale konstantní stupeň
- O motýlcích a spol. obecně platí:
 - Vzniknou **rozvinutím** každého uzlu hyperkrychle **na více uzlů**
 - Kvůli tomu zhoršená částečná škálovatelnost N hodnoty $n2^n$ apod.
 - Optimální z hlediska průměru – dosahují logaritmický průměr při konstantním stupni uzlů
 - Dobrá bisekční šířka - $\Omega(N/\log N)$
- **Zabalený motýlek dimenze n, wBF_n**



$V(wBF_n) = \{(i, x); 0 \leq i < n \wedge x \in B^n\}$	
$E(wBF_n) = \{ \langle (i, x), (i \oplus_n 1, x) \rangle, \langle (i, x), (i \oplus_n 1, \text{neg}_i(x)) \rangle \mid (i, x) \in V(wBF_n) \}$	
$ V(wBF_n) = n2^n$	$ E(wBF_n) = n2^{n+1}$
$\phi(wBF_n) = n + \lfloor \frac{n}{2} \rfloor$	$\deg(wBF_n) = \{4\}$
$\text{bwe}(wBF_n) = 2^n$	

- Vznikne rozvinutím Q_n do $K(n)$ – kružnice n nových uzlů
- Vrcholy kružnic jsou indexovány $0, \dots, n-1$
- Dva druhy hran – **hyperkubické a kružnicové** (proto není hranově symetrický)
 - Kružnicová (motýlková) hrana – v rámci jedné kružnice

- Vrchol i v cyklu x je spojen hyperkubickou hranou s vrcholem $i \oplus 1$ v cyklu $neg_i(x)$
 - **Hyperkubické hrany** spojují sousední uzly vlevo a vpravo – každý uzel má dva sousedy ve své kružnici a dva sousedy v sousedních kružnicích
 - Takhle vznikají křížové hrany, které spolu s kružnicovými hranami tvoří základní motýlky
- **Regulární, uzlově symetrický**
- Není hierarchicky rekurzivní - $K(n)$ neobsahuje podkružnice
- Kvůli kružnicím je **vyvážený bipartitní**, pokud je n sudé
- Vždy existují hamiltonovské kružnice
- Optimální průměr, řídký graf
- **Obyčejný motýlek dimenze n , oBF_n**

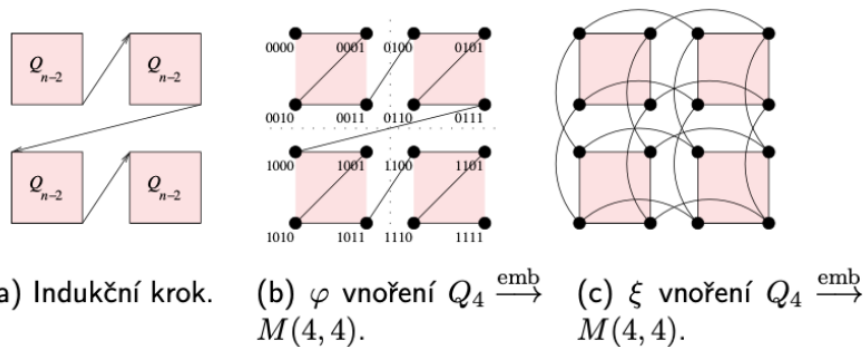


- Obrázek: rozřezání wBF_3 na wBF_3
- Vznikne **ze zabaleného motýlka**, tak, že **rozřízneme každou kružnici v wBF_n v uzlu na pozici 0** tak, že tento uzel se rozdvojí a vzniklé poloviny se podělí o hyperkubické hrany
- Místo n -uzlových kružnic vzniknou $(n+1)$ -uzlové **lineární řady**
- oBF_n se má k wBF_n podobně jako mřížka k toroidu
- Není symetrický, není regulární
- Dva druhy hran – **přímé a křížové** (hyperkubické)
- Uzly oBF_n organizovány do **řad** $0 \leq x \leq 2^n - 1$ a do **sloupců** (stupňů) $0 \leq i \leq n$
 - Hrany spojující sloupec i se sloupcem $i+1$ = hrany úrovně i
- **Hierarchicky rekurzivní** - oBF_n obsahuje dva podgrafy oBF_{n-1}
 - Odebrání uzlů ve sloupci 0 nebo ve sloupci n
- **Bipartitní** (střídání barvy po sloupcích), není hamiltonovský
- Směrování – pouze jedna nejkratší cesta, e-cube směrování
- Využití – minimální permutační síť – levná náhrada křížového přepínače

- Vnořovací problém

- **Statické vnořování** – máme **graf procesů** a **graf fyzického propojení** výpočetních uzlů, chceme namapovat procesy a uzly tak, aby mohly efektivně komunikovat a nevytěžovat síť
 - Simulační mechanismus, který dovolí, aby se počítač s topologií 1 choval jako počítač s topologií 2 a nebyly potřeba algoritmické změny
 - Uvažujeme **zdrojový graf G** s vrcholy $V(G)$ a hranami $E(G)$ a **cílovou sítí H** s uzly $V(H)$ a linkami $E(H)$. Pak (**statické**) vnoření G do H je dvojice zobrazení (φ, χ) , kde:
 - $\varphi: V(G) \rightarrow V(H)$
 - $\chi: E(G) \rightarrow \mathcal{P}(H)$
 - $\mathcal{P}(H)$... množina všech **cest** v síti H
 - Uspořádaná dvojice zobrazení – procesní uzly se mapují na výpočetní uzly, hrany mezi procesy se však musí namapovat na cesty ve výpočetních uzlech
- **Měřítka kvality vnoření**
 - **Maximální zatížení cílového uzlu** – maximální počet zdrojových vrcholů namapovaných na jeden cílový uzel
 - Maximální počet procesů, který bude přidělen 1 procesoru
 - Chceme stejnoměrné zatížení (liší se max o 1)
 - **Expanze vnoření** – poměr velikosti cílové sítě (= počet výpočetních uzlů) a velikosti zdrojového grafu (=počet procesů)
 - Větší expanze implikuje dražší simulace

- Chceme blízkou 1 při jedničkovém zatížení a snižujeme úměrným rovnoměrným zvětšením zatížení uzlů
- **Maximální dilatace zdrojových hran v cílové síti** – maximální délka obrazů zdrojových hran v cílové síti
 - Po jak dlouhých cestách budou muset v cílovém počítači putovat zprávy posílané mezi procesy, které jsou v zdrojovém grafu sousední
 - Sledujeme, pokud máme přepínání citlivé na vzdálenosti, jinak průměrná dilatace
- **Maximální zahlcení cílové hrany**
 - **Linkové zahlcení** – maximální počet obrazů zdrojových hran procházejících skrz cílové linky
 - **Uzlové zahlcení** – maximální počet obrazů zdrojových hran procházejících skrz cílové uzly
 - Spíš sledujeme průměrné zahlcení
- **Kvaziizometrická topologie** – síť G a H jsou kvaziizometrické, pokud G může být vnořen do H a naopak s konstantními měřítky vnoření
 - G a H jsou výpočetně ekvivalentní, pokud jedna může simulovat druhou s konstantním zpomalením (implikováno kvaziizometrií)
- **Vnoření hyperkrychle do nízkorozměrných mřížek**



- Jde vlastně o mapování logické funkce
 - **Svobodovy a Karnaughovy mapy** – Svoboda lexikograficky, Karnaugh Grayův kód
- Máme hyperkrychlický algoritmus a chceme, aby běžel na 2D mřížce
- Hyperkrychle i mřížka jsou **rekurzivní**
- Využití **Mortonovy křivky** – jednotlivé hyperkubické souřadnice se mapují rekurzivně střídavě ve směrech x a y :

$$\varphi(b_{2k-1}b_{2k-2} \dots b_0) = [b_{2k-1}b_{2k-3} \dots b_1, b_{2k-2}b_{2k-4} \dots b_0]$$
 - Alternativně **lexikografické mapování po řádcích/sloupcích**
 - Po řádcích – horní půlka bitů do 1 dimenze, dolní do druhé, sloupce analogicky jako transpozice
- Tímhle dostaneme **4 podkrychle**, které se mapují do 2D mřížky rekurzivně v tzv. **Z-fraktálu**
- Hyperkrychli rozdělíme na 4 podkrychle, mřížku na 4 kvadranty
- Uděláme **vnoření podkrychlí do kvadrantů**
 - Děláme rekurzivně do doby, než se dojde na mřížku 2×2
 - Poté lexikograficky uděláme cestu – Z tvar, který se skládá z malých Z – fraktální křivka

20. Paralelní algoritmy pro redukci, prefixový součet a segmentový prefixový součet na PRAM, v ortogonálních, hyperkubických a obecných topologiích, aplikace.

NI-PDP

Paralelní redukce

- Dáno vstupní pole $X = \{x_0 \dots x_{n-1}\}$ prvků množiny D a binární operace \oplus na D
- Cíl = vypočítat hodnotu $S = x_0 \oplus \dots \oplus x_{n-1}$
- Postačující podmínka pro paralelizovatelnost – **asociativnost** operace \oplus
- $SL(n) = \Omega(n), SU(n) = O(n)$
- Optimální triviální algoritmus:

```

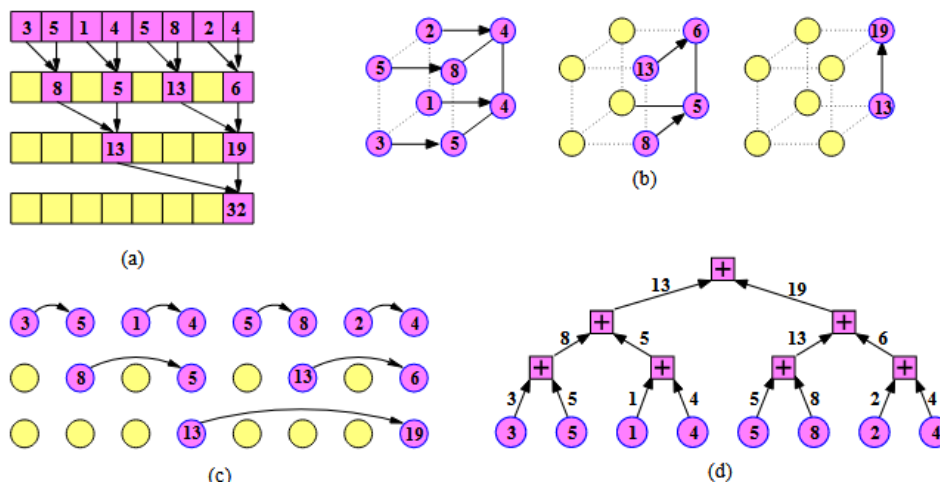
Algorithm Reduction(in:  $X[0, \dots, n-1]$ ; out:  $C$ ;)
{  $i := 0$ ;  $sum := X[i]$ ;
  while ( $i < n-1$ ) do {  $i := i+1$ ;  $sum := sum \oplus X[i]$  };
   $C := sum$  }

```

- **Paralelní výpočet** – operace \oplus se musí aplikovat na co nejvíc nezávislých párů vstupních hodnot
- Pokud je \oplus na D asociativní bin. Operace, potom je paralelní redukce pole X o velikosti n hodnot z D na p procesorech proveditelné na **EREW PRAM**, **přímých a nepřímých stromech**, **hyperkrychlích a hyperkubických topologiích**, **WH mřížkách a toroidech** s následujícími vlastnostmi:

$T(n, p) = O(n/p + \log p)$	$C(n, p) = O(n + p \log p)$	$E(n, p) = \Theta(n/(n + p \log p))$
$\psi_1(p) = p \log p$	$\psi_2(n) = n / \log n$	$\psi_3(n) = n / \log n$

- Paralelní redukce na (a) EREW PRAM, (b) hyperkrychli, (c) 1-D WH mřížce, (d) nepřímém stromu



- o Bruh je to vlastně součet vrcholů no já se poseru

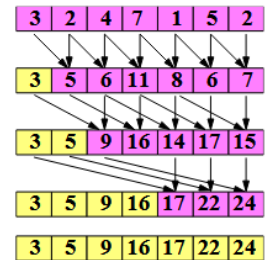
- Redukce v MPI - **MPI_Reduce**
- **Hyperkrychle** – předávání po binomiální kostře
- **Binární strom** nebo motýlek – z listů redukce
- Wormhole (WH) 1D **mřížka** – simulace hyperkrychle
- **PRAM** – redukce každého 2^n -tého prvku

Prefixový součet

- **Prefixový součet** – zobecnění redukce – taky pole X z množiny D a asociativní binární operace \oplus nad D (říkáme jí součet, ale nemusí to být součet, může to být libovolná asociativní operace)
- **Výstup** není jedna hodnota (redukce X), ale **pole Y stejně velké jako původní pole**
 - o y_i je rovno **redukci počátečních i hodnot z X**
 - o Cíl = vypočítat **pole prefixů** pole X : $Y = y_0, \dots, y_{n-1}, y_i = x_0 \oplus \dots \oplus x_i$

- Paralelní prefixový součet (PPS) na EREW PRAM

Algorithm PRAM_PPS(in: $X[0, \dots, n-1]$; out: $M[0, \dots, n-1]$);
 for all $i := 0, \dots, n-1$ do in_parallel
 $M[i] := X[i]$;
 for $j := 0, \dots, \lceil \log n \rceil - 1$ do sequentially
 { for all $i := 2^j, \dots, n-1$ do in_parallel
 $y_i := M[i] + M[i - 2^j]$;
 for all $i := 2^j, \dots, n-1$ do in_parallel
 $M[i] := y_i$ }



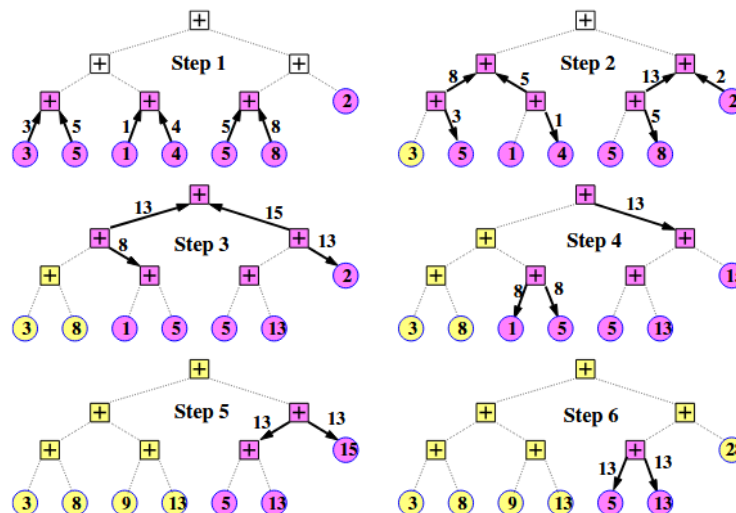
- o $O(\log(n))$ jako paralelní redukce
- o V každém n kroku seřazeno 2^n prvků
- o Pro tvorbu n -tého stavu se přičítají všechny prvky s posunem o 2^n prvků

- PPS na nepřímém stromu/motýlku

- o Pořadí indexace vstupu dáno pořadím listů při průchodu do hloubky
- o **Vnitřní uzly** nevlastní vstupní hodnoty, ale **realizují výpočet**
- o PPS n vstupních hodnot v listech binárního stromu T výšky $h(T)$ lze vypočítat v $2h(T)$ krocích. Je-li T úplný, PPS potřebuje $O(\log n)$ kroků
- o Výpočet má podobu vzestupné vlny, kdy každý vnitřní uzel čeká na hodnoty z obou podstromů, které sečte a pošle svému rodiči, současně ale **předá hodnotu z levého podstromu do pravého**
- o Vzestupná vlna iniciuje menší sestupné vlny, až dorazí do kořene, tam iniciuje sestupnou vlnu v pravém podstromu
- o Sestupující hodnoty se pouze kopírují do všech listů podstromu, kde jsou přidány k mezivýsledku

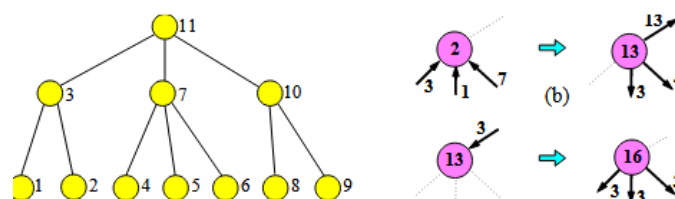


- o Celkový počet kroků = nejvýše dvojnásobek výšky stromu



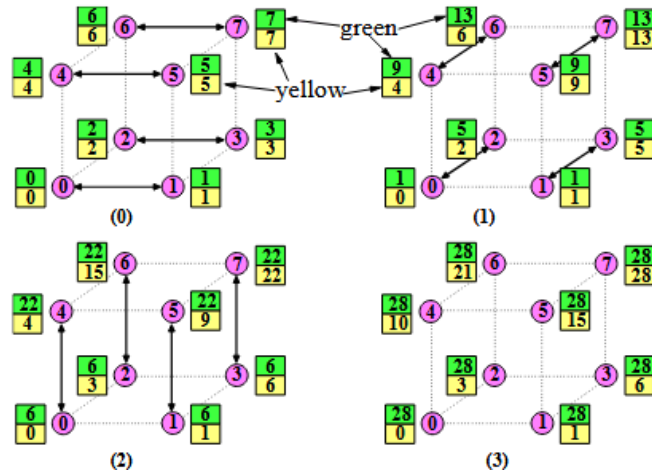
- PPS na **přímém stromu** – modifikace předchozího algoritmu pro p. strom s omezeným větvením

- o Každý uzel stromu **vlastní počáteční hodnotu**
- o Strom není pole, takže musí proběhnout **lineární indexace**
 - Číslování **postorder** – průchod stromu od listů v podstromu zleva
- o Vnitřní uzel při vzestupné vlně čeká na hodnoty ze všech podstromů
- o K nim přidá svoji hodnotu a výsledek pošle rodiči
- o Hodnotu z daného podstromu pošle do všech podstromů vpravo
- o Při sestupné vlně si hodnotu shora započte pro sebe a předá podstromům
- o PPS v $2h(T)$ krocích



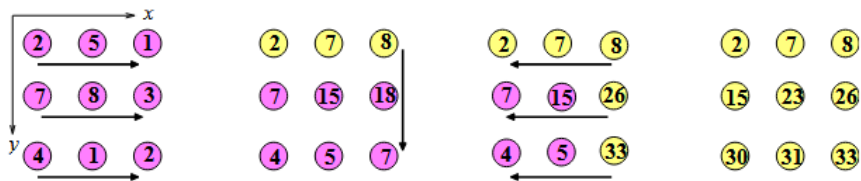
- PPS na **hyperkrychli**

- o Rozšíření vysílání všichni-všem v SF modelu, trvá r paralelních kroků pro Q_r
- o Indexace – zase potřeba **linearizace** – buď **lexikografická**, nebo podle **Grayova kódu**
- o Každý procesor P_i má **2 pomocné registry zelený a žlutý**
 - Ve **žlutém** procesy akumulují pouze to, co je zajímavá z hlediska prefixového součtu (dáno indexem)
 - V **zeleném** akumulují vše včetně hodnot, které je nezajímají, ale které mají jako prostředníci předat procesorům, které je potřebují
- o PPS na hyperkrychli je **normální hyperkubický algoritmus** – lze efektivně implementovat na jakékoliv hyperkubické či posuvné síti



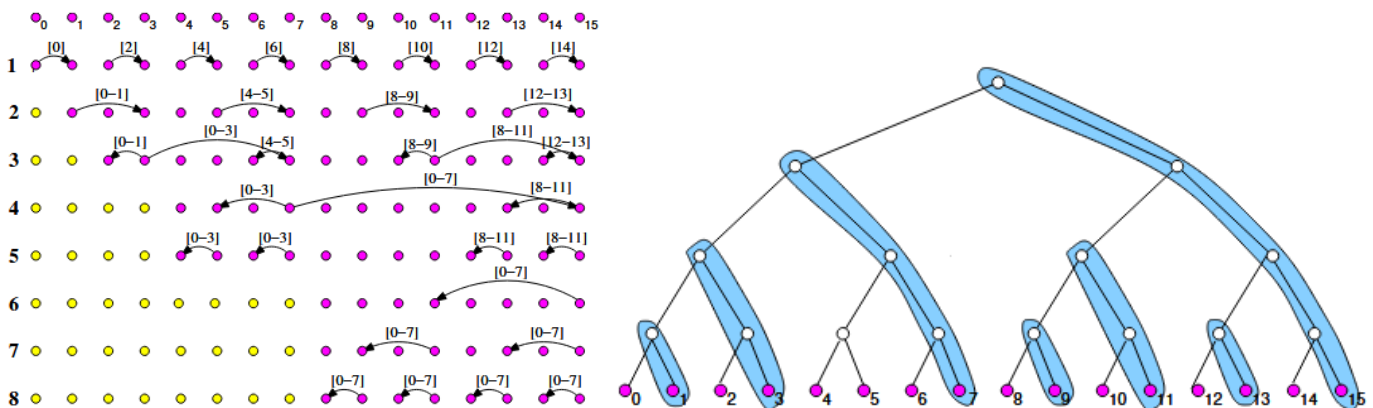
- PPS na ortogonální mřížce

- o SF
 - Mapování vstupního pole na mřížku – podle řádku, sloupce, zig-zag, diagonálně, náhodně
 - Po řádku – 3 fáze:
 - Fáze 1: horizontální PPS s jednotlivými řádky – uzly nejvíc vpravo drží součet řádku
 - Fáze 2: vertikální PPS se sloupcem nejvíc vpravo – takže uzly nejvíc vpravo budou mít správné globální hodnoty prefixového součtu
 - Fáze 3: uzly nejvíc vpravo pošlou horizontálně globální hodnoty prefixového součtu do jejich řádků tak, aby všechny uzly mohly globalizovat svůj výsledek



o WH

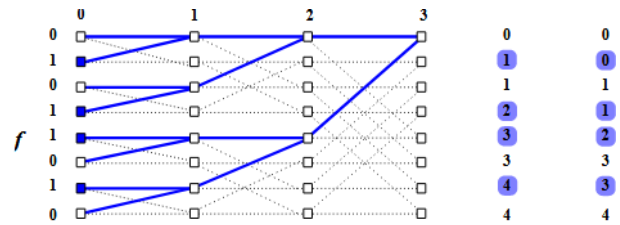
- Simulace na 1-D WH mřížce je v podstatě PPS na nepřímém úplném binárním stromu, který se zploští do lineárního pole
- Oblouky zleva doprava odpovídají vzestupným vlnám, oblouky zprava doleva sestupným vlnám



- Aplikace prefixového součtu

o Zhušťovací problém (packing problem)

- Podmnožina z N procesorů připojených ke vstupům nepřímé vícecestné sítě (nD motýlek) má paket, který je potřeba dopravit na 2. výstupní stranu sítě tak, aby i -tý paket odshora skončil na i -tém výstupním vodiči shora
- Máme množinu procesů, každý má příznak 0 nebo 1 (1 = patří do distribuované množiny)
- Chci určit pořadí procesů označených 1 (např. dle pořadí se rozhoduje místo pro zápis)

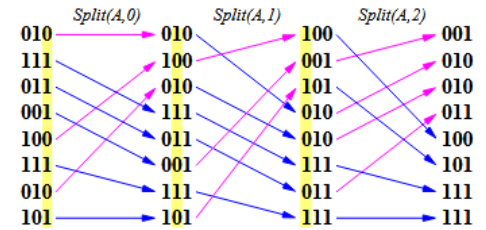


o Paralelní RadixSort – řazení v lexikografickém pořadí

- Nejdřív řadí podle jednotek, desítek apod.
- Zhušťování dle každého lexikografického symbolu a přehazování se zachováním pořadí

o Paralelní sčítačka s predikcí přenosu

- Aby operace netrvala $O(n)$
- 2-bitová čísla – paralelně sčítáním – sečteme jako $0 + 0 = s, 0 + 1 = p, 1 + 1 = g$
- g = určitě carry, s = ne, p = možná carry – řetězec spg doplníme s zprava a vyměníme p za g – z g se stávají 1, z s 0, posuneme o jedno místo $\ll 2$ – máme C
- Sčítámé X, Y, C



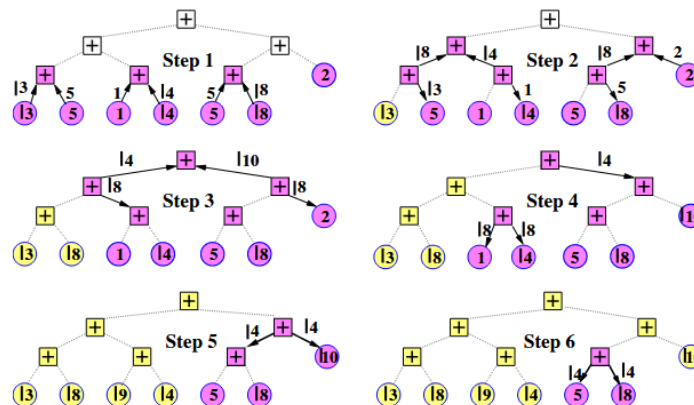
Y:	0	1	1	0	1	0	0	1
B:	s	g	p	p	g	s	p	g
B':	s	g	g	g	g	s	g	g
C:	0	1	1	1	1	0	1	1
Z:	1	1	0	0	0	1	0	0

- Segmentový prefixový součet – SPSS

- Zobecnění PPS pro případ, kdy je vstupní pole rozděleno do různě velkých segmentů
- Cíl = vypočítat všechny prefixové součty uvnitř segmentů izolovaně, rovnoměrné zatížení procesorů
- SPSS se provádí jako globální PPS nad celým polem, ale s modifikovanou operací \oplus , jejíž tabulka se odvodí z tabulky \oplus :

\oplus	b	$ b$
a	$a \oplus b$	$ b$
$ a$	$ a \oplus b$	$ b$

- Vertikální čáry = dané číslo je na levé hranici nějakého segmentu, nebo vlevo od něj už jsou pouze levé hraniční prvky
- Operace je stejná jako ta původní, ale navíc si všimá, zda její operandy jsou levými hraničními prvky segmentů
- Pokud ano, žádná hodnota zleva se nesmí za tuto hranici dostat
- Aplikace \oplus současně značku levé hranice posouvá doprava
- Je-li \oplus asociativní, pak je \oplus taky asociativní



- Aplikace segmentového prefixového součtu – paralelní segmentový QuickSort

- Rozdělení pole podle procesů (rovnoměrně)
- Vstupní posloupnost A je rovnoměrně rozdělena mezi $p < n$ procesorů
- V 1 iteraci hlavní smyčky je každý segment S rozdělen na 3 podsegmenty $S_<, S_=>, S_>$
- Pivot pro daný segment je jeho první prvek zleva