. Společná matematika

1. Základy diferenciálního a integrálního počtu

1.1. Posloupnosti reálných čísel a jejich limity

1.1.1. definice, aritmetika limit

- **definice vlastní limity**: Posloupnost (x_n) má limitu $L \in \mathbb{R}$, když

$$\forall \varepsilon > 0 : \exists n_0 \in \mathbb{N} : \forall n : n > n_0 \Rightarrow |x_n - L| \leq \varepsilon$$

- **definice nevlastní limity**: Posloupnost (x_n) má limitu $L = \infty$, když

$$\forall H > 0 : \exists n_0 \in \mathbb{N} : \forall n : n > n_0 \Rightarrow x_n > H$$

- **definice nevlastní limity**: Posloupnost (x_n) má limitu $L = -\infty$, když

$$\forall H < 0 : \exists n_0 \in \mathbb{N} : \forall n : n > n_0 \Rightarrow x_n < H$$

- **aritmetika limit**: Nechť (a_n), (b_n) jsou posloupnosti s limitami $\lim_{n\to\infty} a_n = a$ a $\lim_{n\to\infty} b_n = b$
  - $\lim_{n\to\infty}(a_n + b_n) = a + b$, je-li výraz definován
  - $\lim_{n\to\infty}(a_n b_n) = ab$, je-li výraz definován
  - pokud $\forall n > n_0 : b_n \neq 0$, pak $\lim_{n\to\infty}\left(\frac{a_n}{b_n}\right) = \frac{a}{b}$, je-li výraz definován

1.1.2. věta o dvou policajtech, limity a uspořádání

- **věta o dvou policajtech**: Pro posloupnosti (a_n), (b_n) a (c_n), pokud platí, že

$$\exists n_0 \in \mathbb{N} : \forall n > n_0 : a_n \leq b_n \leq c_n$$

  a

$$\lim_{n\to\infty} a_n = \lim_{n\to\infty} c_n = L$$

  pak

$$\lim_{n\to\infty} b_n = L$$

- **věta o limitě a uspořádání**: Nechť (a_n), (b_n) jsou posloupnosti s limitami $\lim_{n\to\infty} a_n = a$ a $\lim_{n\to\infty} b_n = b$,
  (i) $a < b \Rightarrow \exists n_0 : n > n_0 \Rightarrow a_n < b_n$
  (ii) $\exists n_0 : n > n_0 \Rightarrow a_n \leq b_n \Rightarrow a \leq b$

1.2. Řady

1.2.1. definice částečného součtu a součtu

- **nekonečná řada** je výraz:

$$\sum_{n=1}^{\infty} a_n = a_1 + a_2 + a_3 + ...$$

- **částečný součet řady**: Pro řadu $\sum_{n=1}^{\infty} a_n$ je n-tý částečný součet

$$s_n = \sum_{i=1}^{n} a_i$$

- **definice součtu řady**: Řada $\sum_{n=1}^{\infty} x_i$ má součet s a je **konvergentní**, když

$$\lim_{n\to\infty}\sum_{i}^{n} a_i = s$$

nebo také

$$\lim_{n\to\infty} s_n = s$$

kde s_n je částečný součet řady.

1.2.2. geometrická řada, harmonická řada

- **geometrická řada** je řada:

$$\sum_{n=0}^{\infty} q^n$$

  - součet je $\frac{1}{1-q}$ pro $|q| < 1$
- **harmonická řada** je řada:

$$\sum_{n=0}^{\infty} \frac{1}{n}$$

1.3. Reálné funkce jedné reálné proměnné

1.3.1. limita funkce v bodě

- definice, aritmetika limit
  - **okolí bodu**: Okolí bodu $U(a,\delta) = (a-\delta, a+\delta)$
    - okolí nekonečen definujeme jako $U(+\infty,\delta) = \left(\frac{1}{\delta},\infty\right)$, $U(-\infty,\delta) = \left(-\infty,-\frac{1}{\delta}\right)$
  - **definice limity**: Funkce f má v bodě $h \in \mathbb{R}^*$ limitu $L \in \mathbb{R}^*$, když

$$\forall\varepsilon > 0 : \exists\delta > 0 : \forall x \neq h : d(x,h) \leq \delta \Rightarrow d(f(x),L) \leq \varepsilon$$

    u vlastních limit a analogicky s řadami u nevlastních nebo

$$\forall\varepsilon > 0 : \exists\delta > 0 : \forall x : x \in P(h,\delta) \Rightarrow f(x) \in U(L,\varepsilon)$$

    pro obojí díky definicím okolí pro nekonečna
  - **vztah s uspořádáním**: pro funkce f a g s limitami $\lim_{x\to\infty}$

    (i) $\lim_{x\to c} f(x) > \lim_{x\to c} g(x) \Rightarrow \exists\delta > 0 : f(x) > g(x) \forall x \in P(c,\delta)$

    (ii) $\exists\delta > 0 : f(x) \geq g(x) \forall x \in P(c,\delta) \Rightarrow \lim_{x\to c} f(x) \geq \lim_{x\to c} g(x)$

    (iii) $\exists\delta > 0 : f(x) \leq h(x) \leq g(x) \forall x \in P(c,\delta) \wedge \lim_{x\to c} f(x) = \lim_{x\to c} g(x) = A \in \mathbb{R}^* \Rightarrow$ $\lim_{x\to c} h(x) = A$ (ekv. věty o dvou policajtech)
  - **limita složené funkce**: Nechť $A, B, C \in \mathbb{R}^*$, $\lim_{x\to A} g(x) = B$ a $\lim_{x\to B} f(x) = C$, navíc, nechť je splněna jedna z podmínek:
    - $f(B) = \lim_{x\to B} f(x) = C$ (f je spojitá v B)

    nebo
    - $\exists\eta > 0 : B \notin g(P(A,\eta))$

    pak

$$\lim_{x\to A} f(g(x)) = C$$

1.3.2. funkce spojité na intervalu

- **spojitost na intervalu**: Funkce je spojitá na intervalu, je-li spojitá v každém vnitřním bodu a jednostraně spojitá v mezích
- **nabývání mezihodnot**: Funkce spojitá na intervalu nabývá všech hodnot mezi mezemi intervalu
- **nabývání maxima**: Funkce spojitá na intervalu má na tomto intervalu maximální hodnotu

1.4. Derivace a její aplikace

1.4.1. definice a základní pravidla pro výpočet

- **definice derivace**:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- **derivace mocniny**:

$$(x^n)' = nx^{n-1}$$

- **derivace součtu**:

$$(f+g)'(x) = f'(x) + g'(x)$$

- **derivace součinu**:

$$(fg)'(x) = g(x)f'(x) + f(x)g'(x)$$

  - (Right dLeft, Left dRight)

- **derivace podílu**:

$$\left(\frac{f}{g}\right)'(x) = \frac{g(x)f'(x) - f(x)g'(x)}{g^2(x)}$$

  - (Right dLeft, Left dRight)

- **derivace gon. funkcní**: $\sin'(x) = \cos(x)$, $\cos'(x) = -\sin(x)$
- **derivace složené funkce**: $(f(g(x)))' = f'(g(x))g'(x)$

1.4.2. l'Hospitalovo pravidlo

- pro funkce f a g s vlastní derivací v $P(h, \delta)$, pro které platí buď

$$\lim_{x \to h} f(x) = \lim_{x \to h} g(x) = 0$$

  nebo

$$\lim_{x \to h} g(x) = \pm\infty$$

  platí

$$\lim_{x \to h} \frac{f(x)}{g(x)} = \lim_{x \to h} \frac{f'(h)}{g'(h)}$$

1.4.3. vyšetření průběhu funkcí: extrémy, monotonie a konvexita/konkavita

- **extrémy**: $f'(x) = 0$
- **monotonie**: $f'(x) > 0 \Rightarrow$ rostoucí, $f'(x) < 0 \Rightarrow$ klesající
- **konvexita/konkavita**: $f''(x) > 0 \Rightarrow$ konvexní, $f''(x) < 0 \Rightarrow$ konkávní

1.4.4. Taylorův polynom (limitní forma)

- $T_n^{f,a}(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + ... + \frac{f^{(n)}(a)}{n!}(x-a)^n$
- platí, že $T^{(n)} = f^{(n)}$, $\left(T_n^{f,a}\right)' = T_{n-1}^{f',a}$

1.5. Integrály a jejich aplikace

1.5.1. primitivní funkce: definice a metody výpočtu

- **definice primitivní funkce**: Pokud pro funkci f definovanou na reálném intervalu $(a,b) \to \mathbb{R}$ existuje funkce $F : (a,b) \to \mathbb{R}$, jejíž derivace na intervalu (a,b) je rovna f, $F'(x) = f(x), \forall x \in (a,b)$ je funkce F primitivní funkcí k funkci f
- **substituce**:

$$\int f(g(x))g'(x)\,\mathrm{d}x = \left| \begin{array}{c} y = g(x) \\ \mathrm{d}y = g'(x)\,\mathrm{d}x \end{array} \right| = \int f(y)\,\mathrm{d}y$$

$$= F(y) + c = \Big| y = g(x) \Big| = F(g(x)) + c$$

$$\int_a^b f(g(x))g'(x)\,\mathrm{d}x = \left| \begin{array}{c} y = g(x) \\ \mathrm{d}y = g'(x)\,\mathrm{d}x \\ x = a \to y = g(a) \\ x = b \to y = g(b) \end{array} \right| = \int_{g(a)}^{g(b)} f(y)\,\mathrm{d}y$$

- **per partes**:

$$\int uv' = uv - \int u'v$$

1.5.2. Riemannův integrál: definice, souvislost s primitivní funkcí (Newtonovým integrálem)

- **dělení intervalu**: $D = (d_1 ... d_n)$ je dělení intervalu $I = [a,b]$, pokud platí, že

$$a = d_0 < d_2 < ... < d_n = b$$

- **dolní a horní Riemannova suma**: pro funkci f na intervalu $[a,b]$ a dělení D tohoto intervalu na intervaly $I_1 ... I_n$:

$$s(f,D) = \sum_{i=1}^{n} |I_i| \cdot \inf\{f(x) \mid x \in I_i\}$$

$$S(f,D) = \sum_{i=1}^{n} |I_i| \cdot \sup\{f(x) \mid x \in I_i\}$$

- **dolní horní Riemannův integrál**: pro funkci f na intervalu $[a,b]$:

$$\underline{\int_a^b} f = \sup\{s(f,D) \mid D \text{ je dělení } [a,b]\}$$

$$\overline{\int_a^b} f = \inf\{S(f,D) \mid D \text{ je dělení } [a,b]\}$$

- **Riemannův integrál**:

$$\underline{\int_a^b} f = \overline{\int_a^b} f = \int_a^b$$

- **1. základní věta analýzy**: Pro Riemannovsky integrovatelnou f je a F:

$$F(x) = \int_a^x f(t)\,\mathrm{d}t$$

  a platí
  1. F je spojitá na $[a, b]$
  2. v každém bodě $x \in [a, b]$ existuje $F'(x) = f(x)$

  tedy: Primitivní funkce lze spočítat Riemannovským integrálem
- **2. základní věta analýzy**: Pokud je f Newtonovsky i Riemannovsky integrovatelná, jsou si R. a N. integrály rovny.

1.5.3. aplikace

- odhady součtu řad (konečných i nekonečných)
  - pro neklesající f na intervalu $[1, n]$

$$\sum_{k=1}^{n-1} f(k) \leq \int_1^n f \leq \sum_{k=2}^{n} f(k)$$

    analogicky pro nerostoucí
  - pro nerostoucí $f : [1, \infty) \to [0, \infty)$ řada $\sum_{k=1}^{\infty} f(k)$ konverguje, když:

$$\lim_{b\to\infty} \int_1^b f < \infty$$

- obsahy rovinných útvarů – použij integrál
- délka křivky

$$\int_a^b \sqrt{1 + (f'(t))^2}\,\mathrm{d}t$$

- objemy a povrchy rotačních útvarů v prostoru

$$V = \pi \int_a^b f(t)^2\,\mathrm{d}t$$

$$S = 2\pi \int_a^b f(t)\sqrt{1 + (f'(t))^2}\,\mathrm{d}t$$

2. Algebra a lineární algebra

2.1. Algebraické struktury

2.1.1. grupy a podgrupy, permutace

- **grupa**: dvojice $(G, +)$:
  1. G je uzavřená na $+$
  2. G obsahuje neutrální prvek $0 : \forall x : x + 0 = x$
  3. pro každý prvek x existuje inverzní prvek $-x$ a platí $x + (-x) = 0$
  4. asociativita $+$
  5. (komutativita $+ \implies$ *Abelovská*)

- **podgrupa**: grupa $(P, +)$ je podgrupa $(G, +)$, pokud $P \subset G$
- **permutace**: bijektivní zobrazení $[n] \to [n]$?

2.1.2. tělesa a speciálně konečná tělesa

- **těleso**: trojice $(T, +, \cdot)$
  1. T uzavřené na $+$ a $\cdot$
  2. $(T, +, 0, -x)$ je Abelovská grupa
  3. $(T \setminus 0, \cdot, 1, x^{-}1)$ je Abelovská grupa
  4. $\cdot$ distributivní vůči $+$: $\forall a, b, c : a \cdot (b + c) = ab + ac$
- **konečné těleso**: těleso na konečné množině prvků: $\mathbb{Z}_p$, kde p je prvočíslo, $\mathrm{GF}(n)$, kde n je mocnina prvočísla

2.2. Soustavy lineárních rovnic

2.2.1. maticový zápis, elementární řádkové úpravy, odstupňovaný tvar matice

- soustavu:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ... + a_1 n x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_2 n x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n &= b_m \end{aligned}$$

  zapíšeme jako

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & ... & a_1 n & b_1 \\ a_{21} & a_{22} & ... & a_2 n & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & ... & a_{mn} & b_m \end{array}\right)$$

- **elementární řádkové úpravy**:
  1. vynásobení řádku nenulovým skalárem
  2. permutace řádků
  3. přičtení jednoho řádku k druhému
- **odstupňovaný tvar matice**:
  - v každém řádku n je pivot index p_n poslední nenulové číslice
  - pro $\forall n, m$ čísla řádků platí $n < m \Rightarrow p_n < p_m$

2.2.2. Gaussova a Gaussova-Jordanova eliminace, popis množiny řešení

- **Gaussova eliminace**: převedení elementárními úpravami do REF
  - postupně přidáváme nuly od leva a zhora dolů
- **Gaussova-Jordanova eliminace**: převádíme do RREF
  - pivoty odečteme od všech řádků nad nimi
- **rovnice má**:
  - 0 řešení, pokud $\mathrm{rank}(\mathbf{A}) < \mathrm{rank}(\mathbf{A}|\boldsymbol{b})$ (b obsahuje pivota)
  - 1 řešení, pokud $\mathrm{rank}(\mathbf{A}) = n$ (každý sloupec v $\mathbf{A}$ má pivota)
  - jinak nekonečně mnoho řešení

2.3. Matice

2.3.1. operace s maticemi a základní typy matic, hodnost matice

- typy matic
  - čtvercová, obdélníková

  - nulová $\mathbf{0} = \begin{pmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{pmatrix}$
  - jednotková $\mathbf{I} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}$
  - symetrická $\mathbf{A} = \mathbf{A}^T$
- operace
  - sčítání ($+ : \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$): $(A+B)_{ij} = A_{ij} + B_{ij}$
  - násobení ($\cdot : \mathbb{R}^{m\times n} \times \mathbb{R}^{n\times p} \to \mathbb{R}^{m\times p}$): $(AB)_{ij} = \sum_{k=1}^{n} A_{ik} \cdot B_{kj}$
  - násobení skalárem?
  - transpozice A^T
- hodnost matice $\text{rank}(\mathbf{A})$ = počet sloupců s pivoty v REF

2.3.2. regulární a inverzní matice

- regulární matice
  1. má inverzi
  2. $\mathbf{A} \in T^{n\times n} : \text{rank}(\mathbf{A}) = n$
  3. $\mathbf{A} \sim\sim \mathbf{I}$
  4. $\boldsymbol{Ax} = \mathbf{0}$ má pouze triv. řešení $\boldsymbol{x} = \mathbf{0}$
- inverzní matice A^{-1}

$$(\mathbf{A} \mid \mathbf{I}) \sim\sim (\mathbf{I} \mid \mathbf{A}^{-1})$$

2.4. Vektorové prostory

2.4.1. vektorový prostor, lineární kombinace, lineární závislost a nezávislost, lineární obal, systém generátorů

- vektorový prostor $(V, +, \cdot)$ nad tělesem $(T, +, \cdot)$
  1. $(V, +)$ je Abelovská grupa
  2. násobení skalárem je asociativní a distributivní
  3. násobení nulou a jedničkou dělá co má
- **podprostor**: neprázdná podmnožina V uzavřená na $+$ a $\cdot$
- **lineární kombinace**: Pro vektory $\mathbf{v}_1 ... \mathbf{v}_n$ je libovolná $\sum_i^n a_i \mathbf{v}_i$, kde $a_1 ... a_n \in T$
- **lineární obal**: $\text{span}(\{\mathbf{v}_1 ... \mathbf{v}_n\})$, průnik všech podprostorů, které obsahují $\{\mathbf{v}_1 ... \mathbf{v}_n\}$ $(\left\{\mathbf{w} : \exists(a_1 ... a_n) : \mathbf{w} = \sum_i^n a_i \mathbf{v}_i\right\})$
- **lineární nezávislost**: $\sum_i^n a_i \boldsymbol{b}_i = \mathbf{0}$ má pouze triviální řešení $a_1 = ... = a_n = 0$

2.4.2. Steinitzova věta o výměně, báze, dimenze, souřadnice

- **lemma o výměně**: v systému generátorů $\{\boldsymbol{c}_1 ... \boldsymbol{c}_n\}$ lze nahradit vektor $\boldsymbol{c}_k$ vektorem $\mathbf{v} = \sum_i^n a_i \boldsymbol{c}_i$, pokud $a_k \neq 0$
- **Steinitzova věta o výměně**: pro množinu vektorů B z vektorového prostoru V a množinu vektorů C, která generuje V můžeme sestrojit množinu D takovou, že $|D| = |C|$, $B \subset D$, D generuje V a $D \setminus B \subset C$
- **báze** vektorového prostoru V je lineárně nezávislá množina vektorů, které generují V
- **dimenze** vektorového prostoru V je velikost libovolné báze
- **vektor souřadnic** $[\mathbf{v}]_B$ vektoru v v bázi B obsahuje koeficienty lineární kombinace bázických vektorů B, která tvoří v

2.4.3. vektorové podprostory, zejména maticové (řádkový, sloupcový, jádro) a jejich dimenze

- **jádro** $\ker(\mathbf{A})$ matice $\mathbf{A}$ je množina řešení $\boldsymbol{Ax} = \mathbf{0}$
- **řádkový a sloupcový prostor** jsou prostory generované řádky a sloupci matice
- pro matici $\mathbf{A} \in T^{m \times n}$, $\dim(\ker(\mathbf{A})) + \operatorname{rank}(\mathbf{A}) = n$
- $\dim(R_{\mathbf{A}}) = \dim(S_{\mathbf{A}}) = \operatorname{rank}(\mathbf{A})$
- z toho plyne $\operatorname{rank}(\mathbf{A}) = \operatorname{rank}(\mathbf{A}^T)$

2.5. Lineární zobrazení

2.5.1. definice, maticová reprezentace lineárního zobrazení, matice složeného zobrazení

- **definice lineárního zobrazení**: U a V jsou vektorové prostory, $f : U \to V$ je lineární zobrazení, pokud:
  1. $f(\mathbf{u} + \mathbf{v}) = f(\mathbf{u}) + f(\mathbf{v})$
  2. $f(t \cdot \mathbf{v}) = t \cdot f(\mathbf{v})$
- **maticová reprezentace lineárního zobrazení**: $\mathbf{A} \in T^{m \times n}$, $f : T^n \to T^m$, $f(\mathbf{u}) = \boldsymbol{Au}$
- **matice linárního zobrazení vzhledem k bazím**: $f : U \to V$ k bazím B a C je

$$[f]_{B,C} = \begin{pmatrix} | & & | \\ [f(\boldsymbol{b}_1)]_C & \dots & [f(\boldsymbol{b}_n)]_C \\ | & & | \end{pmatrix}$$

  poté

$$[f(\mathbf{u})]_C = [f]_{B,C}[\mathbf{u}]_B$$

  nebo Hladíkovsky:

$$[f(\mathbf{u})]_C = {}_C[f]_B[\mathbf{u}]_B$$

- **matice složeného zobrazení**:
  - $\mathbf{A} \in T^{m \times n}, \mathbf{B} \in T^{n \times p}$
  - $f : T^m \to T^n, g : T^n \to T^p$
  - $f(\mathbf{u}) = \boldsymbol{Au}, g(\mathbf{u}) = \boldsymbol{Bu}$
  - $\Rightarrow g(f(u)) = \boldsymbol{BAu}$

2.5.2. obraz a jádro lineárních zobrazení

- **obraz lineárního zobrazení**: pro zobrazení $f : U \to V$, $\{f(\mathbf{u}) \mid u \in U\} \subset V$
- **jádro lineárního zobrazení**: pro zobrazení $f : U \to V$, $\{\mathbf{u} \in U \mid f(\mathbf{u}) = \mathbf{0}\}$

2.5.3. isomorfismus prostorů

- zobrazení $f : U \to V$ je isomorfismus prostorů U, V s bázemi B, V kdyžž $[f]_{B,C}$ je regulární

2.6. Skalární součin

2.6.1. skalární součin, norma indukovaná skalárním součinem

- **skalární součin**: V nad $\mathbb{C}$, $\langle a \mid b \rangle : V \times V \to \mathbb{C}$
  1. $\langle \mathbf{v} \mid \mathbf{v} \rangle \in \mathbb{R}_0^+$
  2. $\langle \mathbf{v} \mid \mathbf{v} \rangle = 0 \Leftrightarrow \mathbf{v} = \mathbf{0}$
  3. $\langle \mathbf{u} \mid \mathbf{v} \rangle = \overline{\langle \mathbf{v} \mid \mathbf{u} \rangle}$
  4. $\langle \mathbf{u} + \mathbf{v} \mid \mathbf{w} \rangle = \langle \mathbf{u} \mid \mathbf{w} \rangle + \langle \mathbf{v} \mid \mathbf{w} \rangle$

5. $\langle \alpha\mathbf{v} \mid \mathbf{u} \rangle = \alpha\langle \mathbf{v} \mid \mathbf{u} \rangle$
6. $\langle \mathbf{u} \mid \mathbf{v} \rangle = 0 \wedge \mathbf{u} \neq \mathbf{v} \Rightarrow$ vektory jsou *kolmé*

- **norma indukovaná skalárním součinem**: $\|\mathbf{v}\| = \sqrt{\langle \mathbf{v} \mid \mathbf{v} \rangle}$
- $\langle \mathbf{u} \mid \mathbf{v} \rangle = \|\mathbf{u}\| \cdot \|\mathbf{v}\| \cdot \cos\varphi$

2.6.2. Pythagorova věta, Cauchyho-Schwarzova nerovnost, trojúhelníková nerovnost

- vektory $\mathbf{u}, \mathbf{v}$ jsou **kolmé**, pokud $\langle \mathbf{u} \mid \mathbf{v} \rangle = 0$
- **Cauchy-Schwarzova nerovnost**: $|\langle \mathbf{u} \mid \mathbf{v} \rangle| \leq \|\mathbf{u}\| \cdot \|\mathbf{v}\|$ (intuice: $\cos\varphi \in [-1, 1]$)
- **trojúhélníková nerovnost**: $\|\mathbf{u}\| + \|\mathbf{v}\| \geq \|\mathbf{u} + \mathbf{v}\|$
- **Pythagorova věta**: pro kolmé vektory platí: $\|\mathbf{u} + \mathbf{v}\|^2 = \|\mathbf{u}\|^2 + \|\mathbf{v}\|^2$

2.6.3. ortonormální systémy vektorů, Fourierovy koeficienty, Gramova-Schmidtova ortogonalizace

- **ortonormální systém vektorů**: všechny vektory jsou vzájemně kolmé a délky 1
- **Grahamova-Schmidtova ortogonalizace**: pro bázi B
  1. $O \leftarrow \{\mathbf{b}_1\}$ odstraním $\mathbf{b}_1$ z B
  2. dokud není B prázdné:
     1. vezmu $\mathbf{b}$ z B
     2. $\mathbf{b}_p \leftarrow$ projekce b do O
     3. $\mathbf{b}_o \leftarrow \mathbf{b} - \mathbf{b}_p$
     4. $O \leftarrow O \cup \{\mathbf{b}_o\}$

  (orto**norm**alizace ještě přidávané vektory zkracuje na jednotkovou délku)
- **Fourierovy koeficienty**: pro ortonormální bázi $B = (\mathbf{b}_1 ... \mathbf{b}_n)$ prostoru V a pro každý vektor $\mathbf{v}$ z V platí:

$$\mathbf{v} = \langle \mathbf{v} \mid \mathbf{b}_1 \rangle \mathbf{b}_1 + ... + \langle \mathbf{v} \mid \mathbf{b}_n \rangle \mathbf{b}_n$$

2.6.4. ortogonální doplněk, ortogonální projekce, projekce jako lineární zobrazení

- **ortogonální projekce**: pro vektor $\mathbf{v}$ a bázi $B = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$:

$$\sum_i^n \frac{\langle \mathbf{v} \mid \mathbf{b}_i \rangle}{\|\mathbf{b}_i\|} \mathbf{b}_i$$

- **ortogonální doplňek**: podmnožiny V prostoru U: $V^\perp = \{\mathbf{u} \in U : \forall \mathbf{v} \in V : \mathbf{u} \perp \mathbf{v}\}$
- projekce je lineární zobrazení

2.6.5. ortogonální matice a jejich vlastnosti

- spíše „ortonormální“
- tvořená např. sloupci ortonormální báze
- unitární matice na $\mathbb{R}$
- uzavřené na součin
- $\mathbf{A}^H\mathbf{A} = \mathbf{I}$ ($\mathbf{A}^T\mathbf{A} = \mathbf{I}$)
- $\mathbf{A}^{-1} = \mathbf{A}^T$
- zobrazení dané ortogonálními maticemi je isometrie

2.7. Determinanty

2.7.1. definice a základní vlastnosti determinantu (multiplikativnost, determinant transponované matice, vztah s regularitou a vlastními čísly)

- S_n grupa permutací na množině $[n]$

- suma součinů diagonál matice vynásobených znamínkem permutace přes všechny permutace sloupců

$$\det \mathbf{A} = \sum_{p \in S_n} \operatorname{sgn} p \prod_{i=1}^{n} a_{i,p(i)}$$

- $\det \mathbf{A} = \det \mathbf{A}^T$
- permutace p sloupců mění znaménko determinantu podle $\operatorname{sgn}(p)$
  - to samé pro řádky
- dva stejné řádky $\Rightarrow \det \mathbf{A} = 0$ (permutace lze popárovat tak, aby pro páry platilo $q = p \circ (k, l)$, páry se liší pouze znaménkem a vzájemně se odečtou)
- vynásobení řádku skalárem vynásobí determinant skalárem

$$\begin{vmatrix} - & a & - \\ - & t \cdot b & - \\ - & c & - \end{vmatrix} = t \cdot \begin{vmatrix} - & a & - \\ - & b & - \\ - & c & - \end{vmatrix}$$

- sečteme-li řádek dvou matic lišících se o řádek, sečteme jejich determinanty

$$\begin{vmatrix} - & a & - \\ - & b_1 + b_2 & - \\ - & c & - \end{vmatrix} = \begin{vmatrix} - & a & - \\ - & b_1 & - \\ - & c & - \end{vmatrix} + \begin{vmatrix} - & a & - \\ - & b_2 & - \\ - & c & - \end{vmatrix}$$

- přičtením t-násobku jednoho řádku k druhému nezměníme determinant

$$\begin{vmatrix} - & a & - \\ - & b + t \cdot c & - \\ - & c & - \end{vmatrix} = \begin{vmatrix} - & a & - \\ - & b & - \\ - & c & - \end{vmatrix} + \overbrace{t \cdot \begin{vmatrix} - & a & - \\ - & c & - \\ - & c & - \end{vmatrix}}^{=0}$$

  - dá se dělat i se sloupci ($\det \mathbf{A} = \det \mathbf{A}^T$)
- $\det(\mathbf{AB}) = \det \mathbf{A} \det \mathbf{B}$
  - rozložení na elementární operace
  - součet řádků nemění determinant a $\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{vmatrix} = 1$
  - násobení řádku skalárem násobí determinant skalárem a $\begin{vmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{vmatrix} = 3$
- matice je singulární kdyžž $\det = 0$

2.7.2. Laplaceův rozvoj determinantu

- vybereme řádek i:

$$\det \mathbf{A} = \sum_{j=1} a_{ij} (-1)^{i+j} \det \mathbf{A}^{ij}$$

  kde $\mathbf{A}^{ij}$ je matice bez i-tého řádku a j-tého sloupce

$$\begin{vmatrix} 1 & 2 & 5 \\ 2 & 3 & 0 \\ 3 & 5 & 3 \end{vmatrix} = 2 \cdot (-1)^{2+1} \begin{vmatrix} 2 & 5 \\ 5 & 3 \end{vmatrix} + 3 \cdot (-1)^{2+2} \begin{vmatrix} 1 & 5 \\ 3 & 3 \end{vmatrix} + 0 \cdot (-1)^{2+3} \begin{vmatrix} 1 & 2 \\ 3 & 5 \end{vmatrix}$$

2.7.3. geometrická interpretace determinantu

- znamínkový obsah rovnoběžnostěnu daného vektory ve sloupci matice

2.8. Vlastní čísla a vlastní vektory

2.8.1. definice, geometrický význam a základní vlastnosti vlastních čísel, charakteristický polynom, násobnost vlastních čísel

- pro $f : V \to V$, **vlastní číslo** je $\lambda \in T : \exists \mathbf{v} \in V \setminus \mathbf{0} : f(\mathbf{v}) = \lambda\mathbf{v}$
- pro $f : V \to V$, **vlastní vektor** je $\mathbf{v} \in V \setminus \mathbf{0} : f(\mathbf{v}) = \lambda\mathbf{v}$
- množina vlastních čísel = **spektrum**
- **geometrická násobnost** λ: dimenze podprostoru vlastních vektorů
- diagonální matice – čísla na diagonále jsou λ, vektory kanonické báze $(1, 0, 0...)$ jsou vlastní
- vlastní vektory jednoho λ tvoří podprostor
- vlastní vektory různých λ jsou lineárně nezávislé
- matice může mít max. n vlastních čísel (max. počet podprostorů)

2.8.1.1. charakteristický polynom

- $p_{\mathbf{A}}(x) = \det(\mathbf{A} - x\mathbf{I})$
- kořeny $p_{\mathbf{A}}(x)$ jsou λ
  - $\mathbf{0} = \mathbf{A}\mathbf{v} - \lambda\mathbf{v} = (\mathbf{A} - \lambda\mathbf{I})v \Leftrightarrow \mathbf{A} - \lambda\mathbf{I}$ je singulární $\Leftrightarrow 0 = \det(\mathbf{A} - \lambda\mathbf{I}) = p_{\mathbf{A}}(\lambda)$
  - **algebraická násobnost** λ = kolikrát lze $p_{\mathbf{A}}(x)$ beze zbytku vydělit $(x - \lambda)$

2.8.2. podobnost a diagonalizovatelnost matic, spektrální rozklad

- **podobné matice** – matice stejného zobrazení, vůči jiné bázi
  - $[f]_{B,B} = [\mathrm{id}]_{C,B} \cdot [f]_{C,C} \cdot [\mathrm{id}]_{B,B}$
  - jinými slovy: $\mathbf{A} = \mathbf{R}^{-1}\mathbf{B}\mathbf{R}$
  - podobné matice mají stejná λ se stejnými geometrickými a algebraickými násobnostmi
- **algebraická násobnost** $\geq$ **geometrické násobnosti**
- **diagonalizovatelná matice** je podobná s diagonální maticí
  - $\Leftrightarrow$ vlastní vektory tvoří bázi T^n
- $\mathbf{A}^k = \mathbf{R}^{-1}\mathbf{D}^k\mathbf{R}$

2.8.3. symetrické matice, jejich vlastní čísla a spektrální rozklad

- pro symetrickou matici $\mathbf{A}$ existuje **ortogonální** $\mathbf{R}$ taková, že $\mathbf{R}^{-1}\mathbf{A}\mathbf{R}$ je diagonální
- zápis $\mathbf{A} = \mathbf{R}\mathbf{D}\mathbf{R}^T$ je **spektrální rozklad**

2.9. Positivně semidefinitní a positivně definitní matice

Definice a vlastnosti sloučeny, rozdíly značeny: positivně definitní a positivně semidefinitní

2.9.1. charakterizace a vlastnosti, vztah se skalárním součinem, vlastními čísly

- **definice**: symetrická $\mathbf{A}$ je P(S)D, když $\mathbf{v}^T\mathbf{A}\mathbf{v} \overset{>}{\geq} 0$
- P(S)D $\Rightarrow$ kladná / nezáporná diagonála
- **Gramova matice**: $\mathbf{v}^T\mathbf{A}^T\mathbf{u}^T = \langle \mathbf{u} \mid \mathbf{v} \rangle$
  - pro bázi $B = (\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_n)$, $a_{ij} = \langle \mathbf{b}_i \mid \mathbf{b}_j \rangle$
  - PD matice = Gramova matice nějakého V
- $\forall \lambda : \lambda \overset{>}{\geq} 0$
- $\exists$ regulární matice $\boldsymbol{U} : \boldsymbol{U}^T\boldsymbol{U} = \mathbf{A}$

2.9.2. Choleského rozklad (znění věty a praktické použití)

- PD $\Rightarrow$ jednoznačná $\boldsymbol{U} = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$ s $u_{ii} > 0$, tž. $\boldsymbol{U}^T\boldsymbol{U} = \mathbf{A}$ = **Choleského rozklad**
- $u_{11} = \sqrt{a_{11}}$, dále dopočítám $a_{12}, a_{13}, ..., a_{22}, ...$ jednoznačně, aby vycházel součin

3. Diskrétní matematika

3.1. Relace

- binární $R \subset X \times Y$, obecně $R \subset X \times Y \times ...$
- **reflexivita**: $\forall a : aRa$
- **symetrie**: $\forall a, b : aRb \Leftrightarrow bRa$
- **antisymetrie**: $\forall a, b : a \neq b \Rightarrow (aRb \Leftrightarrow b \not R a)$
- **transitivita**: $\forall a, b, c : aRb \wedge bRc \Rightarrow aRc$

3.2. Ekvivalence a rozkladové třídy

- **ekvivalence**: symetrická, reflexivní a transitivní relace
- **rozkladové třídy**: $R[x] \coloneqq \{x \mid xRy\}$
  - $R[x] = R[y] \Leftrightarrow R[x] \cap R[y] \neq \emptyset$
  - $\{R[x] | \forall x\}$ jednoznačně určuje R

3.3. Částečná uspořádání

- **uspořádání**: antisymetrická, reflexivní a transitivní relace
- x, y jsou **porovnatelné** $\Leftrightarrow xRy \vee yRx$
- **lineární uspořádání**: $\forall x, y : xRy \vee yRx$ (vs. **částečné**)
- **ostré uspořádání**: (není uspořádání z definice), přidáme ireflexivitu
- základní pojmy (minimální a maximální prvky, nejmenší a největší prvky, řetězec, antiřetězec)
- **minimální**: neexistuje menší ($\nexists y : y < x$)
- **nejmenší**: je menší než všichni ostatní ($\forall y : x \leq y$)
- **maximální**: neexistuje větší ($\nexists y : y > x$)
- **největší**: je větší než všichni ostatní ($\forall y : x \geq y$)
- **řetězec**: $Y \subset X : \forall x, y \in Y : xRy \vee yRx$
- **antiřetězec**: $Y \subset X : \forall x, y \in Y : \neg(xRy \vee yRx)$
- výška a šířka částečně uspořádané množiny a věta o jejich vztahu (o dlouhém a širokém)
  - **výška**: ω max. velikost řetězce
  - **šířka**: α max. velikost antiřetězce
  - $\alpha \cdot \omega \geq |X|$, $\max(\alpha, \omega) \geq \sqrt{|X|}$
  - **Erdős-Szekeres**: V posloupnosti $x_1 ... x_{n^2+1}$ různých čísel $\exists$ vybraná monotóní podposloupnost délky n

3.4. Funkce

- relace f je funkce, pokud $f : X \to Y, \forall x \in X \exists! y \in Y : xfy : f(x) = y$
- typy funkcí (prostá, na, bijekce)
- **prostá** (injektivní) $f : \forall x, y : f(x) = f(y) \Rightarrow x = y$
- **na** (surjektivní) $f : \forall y \exists x : f(x) = y$
- **biijektivní** prostá a na
- počty různých typů funkcí mezi dvěma konečnými množinami
  - **bijekce**: stejné velikosti množin, počet fcí je počet permutací, $|X|! = |Y|!$
  - **prosté**: pro každou z X vybírám jednu hodnotu z Y bez opakování, závisí na pořadí $|Y|^{\underline{|X|}}$
  - **na**: pro každou z Y vybíram jednu hodnotu z X s opakováním, závisí na pořadí

3.5. Permutace a jejich základní vlastnosti (počet a pevný bod)

- **permutace**: bijekce $\pi : [n] \to [n]$

- počet permutací: $n!$
- **pevný bod**: $i : \pi(i) = i$

3.6. Kombinační čísla a vztahy mezi nimi, binomická věta a její aplikace

- **kombinační číslo**:

$$\binom{n}{k} := \frac{n^{\underline{k}}}{k!} = \frac{n!}{k!(n-k)!} = \frac{n \cdot (n-1) \cdot ... \cdot (n-k+1)}{k \cdot (k-1) \cdot ... \cdot 1}$$

- počet k-prvkových podmnožin množiny velikosti n
- $\binom{n}{0} = 1, \binom{n}{n} = 1, \binom{n}{1} = n, \binom{n}{k} = \binom{n}{n-k}$
- $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$ $\#(|k| \subset A) = \#(|k| \subset A \setminus \{a\}) + \#((|k-1| \subset A \setminus \{a\})) \cup \{a\})$
- **binomická věta**:

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$$

- $2^n = (1+1)^n = \sum_{k=0}^{n} \binom{n}{k} \cdot 1 \cdot 1$
- $0 = (1-1)^n = \sum_{k=0}^{n} (-1)^k \binom{n}{k}$

3.7. Princip inkluze a exkluze

- **PIE**: pro množiny $A_1 ... A_n$

$$\left| \bigcup_{i=1}^{n} A_i \right| = \sum_{k=1}^{n} (-1)^{k+1} \sum_{I \in \binom{[n]}{k}} \left| \bigcap_{i \in I} A_i \right|$$

- **Dk1**: $A = \bigcup_{i=1}^{n} A_i$
  - kolikrát levá a pravá $\sum$ započítá libovolné $a \in A$?
  - levá $L = 1$
  - pravá: $t := \#i : a \in A_i$

$$\sum_{I \in \binom{[n]}{k}} \left| \bigcap_{i \in I} A_i \cap \{a\} \right| = \begin{cases} 0 \text{ pokud } k > t \\ \binom{t}{k} \text{ pokud } k \leq t \end{cases}$$

$$P = \sum_{k=1}^{t} (-1)^{k+1} \binom{t}{k}$$

$$P = -\sum_{k=1}^{t} (-1)^{k} \binom{t}{k}$$

$$P = -\left(\sum_{k=0}^{t} (-1)^{k} \binom{t}{k} - \binom{t}{0} \right)$$

$$P = -(0-1)$$

$$P = 1$$

- **Dk2**:

$$c_{A(x)} := \begin{cases} 0 \text{ pokud } x \notin A \\ 1 \text{ pokud } x \in A \end{cases}$$

$$1 - c_A \leftrightarrow \overline{A}$$

$$c_A \cdot c_B \leftrightarrow A \cap B$$

$$1 - (1 - c_A) \cdot (1 - c_B) \leftrightarrow \overline{\overline{A} \cap \overline{B}} = A \cup B$$

$$\prod_{i=0}^{n}(1-c_i) = \sum_{k=0}^{n}(-1)^k \sum_{I\in\binom{[n]}{k}} \prod_{i\in I} c_i$$

$$\prod_{i=0}^{n}(1-c_i) = 1 + \sum_{k=1}^{n}(-1)^k \sum_{I\in\binom{[n]}{k}} \prod_{i\in I} c_i$$

$$\prod_{i=0}^{n}(1-c_i) = 1 - \sum_{k=1}^{n}(-1)^{k+1} \sum_{I\in\binom{[n]}{k}} \prod_{i\in I} c_i$$

$$1-\prod_{i=0}^{n}(1-c_i) = \sum_{k=1}^{n}(-1)^{k+1} \sum_{I\in\binom{[n]}{k}} \prod_{i\in I} c_i$$

- použití (problém šatnářky, Eulerova funkce pro počet dělitelů, počet surjekcí)

3.7.1. problém šatnářky

- $\check{S}_n := \#\{\pi : [n] \to [n] \mid \forall i : \pi(i) \neq i\}$
- $\check{S}_n = \#\{\pi : [n] \to [n]\} - \#\{\pi : [n] \to [n] : \exists i_1 : \pi(i_1) = i_1\} + \#\{\pi : [n] \to [n] : \exists i_1, i_2 : i_1 \neq i_2 : \pi(i_1) = i_1, \pi(i_2) = i_2\} - ...$
- $$\check{S}_n = \sum_{k=0}^{n}(-1)^k \binom{n}{k}(n-k)! = \sum_{k=0}^{n}(-1)^k \frac{n!}{k!} = n!\sum_{k=0}^{n}\frac{(-1)^k}{k!} \approx \frac{n!}{e}$$

3.7.2. Eulerova funkce pro počet dělitelů

- ani náhodou

3.7.3. počet surejkcí

- $\#\{f : M \to N\} - \#\{f : M \to N \mid \exists n_1 : \forall m \in M : f(m) \neq n_1\} + ...$
- $$\sum_{k=0}^{n-1}(-1)^k \binom{n}{k}(n-k)^m$$
- $n-1$: $\{f : M \to N \mid \forall m \in M : f(m) \notin N\} = \emptyset$

3.8. Hallova věta o systému různých reprezentantů a její vztah k párování v bipartitním grafu

- **Hallova věta**: pro bipartitní graf $G = (V, E)$ s partitami A, B

  G má párování velikosti $|A| \Leftrightarrow \forall X \subseteq A : |\{y \in V \setminus X : \exists x \in X : \{x, y\} \in E\}| \geq |X|$
- princip důkazu a algoritmické aspekty (polynomiální algoritmus pro nalezení SRR) TODO

4. Teorie grafů

4.1. Základní pojmy teorie grafů

- **graf**: $G = (V, E), E \subset \binom{V}{2}$
- **orientovaný**: $E \subset V \times V$
- **isomorfismus grafů**: $G_1 = (V_1, E_1)$ a $G_2 = (V_2, E_2)$ jsou isomorfní kdyžž

  $\exists$ bijektivní $f : V_1 \to V_2 : \forall u, v \in V_1 : \{u, v\} \in E_1 \Leftrightarrow \{f(u), f(v)\} \in E_2$
- **podgraf**: $G_p = (V_p, E_p)$ grafu $G = (V, E)$: $V_p \subset V, E_p \subset E$
- **indukovaný podgraf**: $G_p = (V_p, E_p)$ grafu $G = (V, E)$: $V_p \subset V, E_p = E \cap \binom{V}{2}$
- **okolí vrcholu**: v v grafu $G = (V, E)$ je $\{u \in V \mid \{u, v\} \in E\}$
- **stupeň vrcholu**: v grafu $G = (V, E)$ $\deg(v) = |\{u \in V \mid \{u, v\} \in E\}|$
- **doplněk grafu**: $\overline{G} = (\overline{V}, \overline{E})$ grafu $G = (V, E)$: $\overline{V} = V, \overline{E} = \binom{V}{2} \setminus E$

- **bipartitní graf**: $\exists L, P : L \cap P = \emptyset, L \cup P = V, \forall e \in E : |e \cap L| = |e \cap P| = 1$

4.2. Základní příklady grafů

- **úplný graf**: $K_n = \left([n], \binom{[n]}{2}\right)$
- **úplný bipartitní graf**: $K_{m,n} = (L \cup P, \{\{l, p\} \mid l \in L, p \in P\}), |L| = m, |P| = n$
- **cesta**: $P_n([n]_0, \{\{i, i+1\} \mid 0 \leq i < n\})$
- **kružnice**: $C_n([n-1]_0, \{\{i, i+1 \bmod n\} \mid 0 \leq i < n\})$

4.3. Souvislost grafů, komponenty souvislosti, vzdálenost v grafu

- **relace dosažitelnosti**: $u \sim v$ kdyžž $\exists$ cesta mezi u a v v grafu G
- **souvislý graf**: $\forall u, v \in V : u \sim v$
- **komponenta souvislosti**: podgraf indukovaný třídami ekvivalence $\sim$
- **vzdálenost**: $d(u, v)$ délka nejmenší cesty mezi u a v

4.4. Stromy

- ekvivalentní charakteristiky stromů
  1. souvislý acyklický
  2. minimální souvislý
  3. maximální acyklický
  4. (**Eulerova formule**) souvislý a $|E| + 1 = |V|$
  5. $\forall u, v \in V \exists!$ cesta mezi u a v – jednoznačně souvislý
- **list** = vrchol $\deg(v) = 1$
- každý strom velikosti ≥ 2 má alespoň 1 list

4.5. Rovinné grafy

- **rovinné nakreslení**:
  - vrcholy jsou body v $\mathbb{R}^2$
  - hrany jsou oblouky $f : [0, 1] \to \mathbb{R}^2$ spojitá a prostá
  - hrany nemají společný bod
  - vrcholy se nachází pouze na těch hranách, které ukončují
- **rovinný graf** má rovinné nakreslení
- **rovinný na sféře** $\iff$ **rovinný** (bijekce přes polopřímky ze severního pólu (pokud je na severním pólu bod, lze sféru otočit o ε))
- **stěna**: oblast ohraničená hranami
  - specifické pro nakreslení, ne pro graf

4.6. Eulerova formule a maximální počet hran rovinného grafu (důkaz a použití)

- $v + f = e + 2$
- pro strom platí $(v + 1 = e + 2)$
- přidáním hrany přidáme i stěnu

4.7. Barevnost grafů

- **obarvení**: $c : V \to [k]$ t. ž. $\forall \{x, y\} \in E : c(x) \neq c(y)$
- **barevnost**: $\chi(G) \coloneqq \min\{k \mid \exists c : V \to [k], c \text{ je obarvení}\}$
- vztah barevnosti a klikovosti grafu
  - **klikovost**: $\omega(G) \coloneqq \max\{k \mid K_k \subseteq G\}$
  - $\chi(K_k) = k$
  - $\chi(G) \geq \omega(G)$

4.8. Hranová a vrcholová souvislost grafů

- **hranový řez**: množina $F \subset E$ taková, že $(V, E \setminus F)$ je nesouvislý
- **vrcholový řez**: množina $C \subset V$ taková, že podgraf indukovaný $V \setminus C$ je nesouvislý
- **hranová souvislost**: $k_{e(G)} = \min\{|F|; F \subset E, F \text{ je hranový řez}\}$
- **vrcholová souvislost**: $k_{v(G)} = \begin{cases} \min\{|C|; C \subset V, C \text{ je vrcholový řez}\} & \text{pro } G \ncong K_n \\ n-1 & \text{pro } G \cong K_n \end{cases}$
- hranová a vrcholová verze Mengerovy věty
  - **Mengerova hranová věta**: $k_{e(G)} \geq n \Leftrightarrow \forall u, v$: existuje n hranově disjunktních cest mezi u a v
  - **Mengerova vrcholová věta**: $k_{v(G)} \geq n \Leftrightarrow \forall u, v$: existuje n cest mezi u, v disjunktních až na u a v

4.9. Orientované grafy, silná a slabá souvislost

- **orientovaný**: $E \subset V \times V$ – hrany jsou uspořádané dvojice
- **silná souvislost**: mezi vrcholy existuje orientovaná cesta
- **slabá souvislost**: vrcholy jsou souvislé v podkladovém neorientovaném grafu

4.10. Toky v sítích

- **síť**: orientovaný graf (V,E), $c : E \to \mathbb{R}_0^+$ funkce kapacit, a vrcholy z zdroj a s stok
- **tok**: $f : E \to \mathbb{R}_0^+$
  1. $\forall e \in E : f(e) \leq c(e)$
  2. $\forall v \in V \setminus \{z, s\} : \sum_{u:uv \in E} f(uv) = \sum_{u:uv \in E} f(vu)$
  - $f^+(v) \coloneqq \sum_{u:uv \in E} f(uv)$ – přítok
  - $f^-(v) \coloneqq \sum_{u:uv \in E} f(vu)$ – odtok
  - $f^{\Delta(v)} \coloneqq f^+(v) - f^-(v)$ – přebytek
  - jinak řečeno 2. $\forall v \in v \setminus \{z, s\} : f^{\Delta(v)} = 0$
- existence maximálního toku (bez důkazu)
  - v každé síti existuje právě jeden nebo nekonečně mnoho maximálních toků – ze dvou lze vždy vytvořit třetí zprůměrováním hran, které se liší
  - velikost maximálního toku je rovna velikosti minimálního řezu
  - v celočíselné síti je max. tok celočíselný
- princip hledání maximálního toku v síti s celočíselnými kapacitami (například pomocí Ford-Fulkersonova algoritmu)
  - viz ADS

5. Pravděpodobnost a statistika

5.1. Pravděpodobnostní prostor, náhodné jevy, pravděpodobnost

- **pravděpodonostní prostor**: trojice $(\Omega, \mathcal{F}, P)$
  - Ω – množina elementárních jevů
  - $\mathcal{F} \subseteq \mathcal{P}(\Omega)$ – prostor jevů (typicky $\mathcal{P}(\Omega)$)
    1. $\emptyset \in \mathcal{F}, \Omega \in \mathcal{F}$
    2. $A \in \mathcal{F} \Rightarrow A^\complement \coloneqq \Omega \setminus A \in \mathcal{F}$
    3. $A_1, A_2 ... \in \mathcal{F} \Rightarrow \bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$
  - $P : \mathcal{F} \to [0, 1]$ – pravděpodobnost
    1. $P(\Omega) = 1$, $P(\emptyset) = 0$
    2. $P(\bigcup_{i=1}^{\infty} A_i = \sum_{i=1}^{\infty} P(A_i)$ pro po dvou disjunktní $A_1, A_2 ... \in \mathcal{F}$
- základní pravidla pro počítání s pravděpodobností
  - $P(A) + P(A^\complement) = 1$

- $A \subseteq B \Rightarrow P(B \setminus A) = P(B) - P(A) \Rightarrow P(A) \leq P(B)$
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- $P(A_1 \cup A_2 ...) \leq \sum_i P(A_i)$

- nezávislost náhodných jevů, podmíněná pravděpodobnost
  - jevy A a B jsou nezávislé, pokud $P(A \cap B) = P(A) \cdot P(B)$
  - $P(A \mid B) = \frac{P(A \cap B)}{P(B)}$ „pravděpodobnost A za předpokladu B“
- **Bayesův vzorec**: pro rozklad $\Omega = B_1 \cap B_2 ...$

$$P(B_j \mid A) = \frac{P(A \mid B_j) \cdot P(B_j)}{\sum_i P(A \mid B_i) \cdot P(B_i)}$$

5.2. Náhodné veličiny a jejich rozdělení

- **(diskrétní) náhodná veličina**: $X : \Omega \to \mathbb{R}$, pokud (Im($X$) je spočetná) a pro všechna x z $\mathbb{R}$ platí:

$$\{\omega \in \Omega : X(\omega) = x\} \in \mathcal{F}$$

5.2.1. popis pomocí funkcí

- **pravděpodobnostní funkce**: $p_X : \mathbb{R} \to [0, 1] : p_{X(x)} = P(\{X = x\})$ (pro d. n. v.)
- **distribuční funkce**: $F_X : \mathbb{R} \to [0, 1] : F_{X(x)} \coloneqq P(X \leq x) = P(\omega \in \Omega : X(\omega) \leq x)$
  1. F_X je neklesající
  2. $\lim_{x \to -\infty} F_{X(x)} = 0$
  3. $\lim_{x \to +\infty} F_{X(x)} = 1$
  4. zprava spojitá $F_{X(x_+)} = F_{X(x)}$
- **hustotní funkce:** $f_X : \mathbb{R} \to \mathbb{R}_0^+ : F_{X(x)} = \int_{-\infty}^{x} f_X(t)\,\mathrm{d}t$ (pokud existuje, n. v. je spojitá)
  1. $P(X = x) = 0 \quad \forall x \in \mathbb{R}$
  2. $P(a \leq X \leq b) = \int_a^b f_X(t)\,\mathrm{d}t \quad \forall a < b \in \mathbb{R}$
  3. F_X je spojitá

5.2.2. střední hodnota

- **střední hodnota diskrétní n. v.**:

$$\mathbb{E}(X) = \sum_{x \in \mathrm{Im}(X)} x \cdot P(X = x)$$

- **střední hodnota spojité n. v.**:

$$\mathbb{E}(X) = \int_{-\infty}^{\infty} x \cdot f_{X(x)}$$

- linearita střední hodnoty
  - $\mathbb{E}(X + Y) = \mathbb{E}(X) + \mathbb{E}(Y)$
  - $\mathbb{E}(a \cdot X + b) = a\mathbb{E}(X) + b$
- střední hodnota součinu nezávislých veličin
  - **diskrétní n. v. jsou nezávislé, pokud**: $P(X = x \wedge Y = y) = P(X = x)P(Y = y)$
  - pro **nezávislé** d. n. v. platí: $\mathbb{E}(X \cdot Y) = \mathbb{E}(X) \cdot \mathbb{E}(Y)$
- **Markovova nerovnost**: pro $X \geq 0$ a $a \in \mathbb{R}^+$:

$$P(X \geq a) \leq \frac{\mathbb{E}(X)}{a}$$

5.2.3. rozptyl

- **rozptyl:** $\operatorname{var}(X) = \mathbb{E}((X - \mathbb{E}X)^2)$
  - $= \mathbb{E}(X^2) - \mathbb{E}(X)^2$
- vzorec pro rozptyl součtu (závislých či nezávislých veličin)
  - $\operatorname{var}(aX + b) = a^2 \operatorname{var}(X)$

5.2.4. práce s konkrétními rozděleními

- **geometrické** (džbánové) – kolikrát musím dojít se džbánem pro vodu, než se ucho utrhne
  - $X \sim \text{Geom}(p)$ (p je pravděpodobnost, že se ucho utrhne při každé výpravě pro vodu)
  - $p_X(k) = (1-p)^{k-1} \cdot p$ ($(k-1)$-krát dojdu pro vodu bez utržení a jednou s utržením)
  - $\mathbb{E}(X) = \frac{1}{p}$
  - $\operatorname{var}(X) = \frac{1-p}{p^2}$
- **binomické** (multidžbánové) – počet džbánů, které spotřebuji pro n výprav pro vodu[1]
  - $X \sim \text{Binom}(n, p)$
  - $p_{X(k)} = \binom{n}{k} p^k (1-p)^{n-k}$ (k-krát dojdu pro vodu a ucho utrhnu, $n-k$-krát ho neutrhnu, existuje $\binom{n}{k}$ způsobů, jak posloupnost takových cest mohla vypadat)
  - $\mathbb{E}(X) = np$ (pokud jeden džbán vydrží v průměru $\frac{1}{p}$ cest, použiju $\frac{n}{\frac{1}{p}}$ džbánů)
  - $\operatorname{var}(X) = np(1-p)$
- **Poissonovo** (opravářovo) – kolik nosičů přijde za den k opraváři nechat přilepit ucho
  - $X \sim \text{Pois}(\lambda)$
  - $p_{X(k)} = \frac{\lambda^k}{k!} e^{-\lambda}$
  - $\mathbb{E}(X) = \lambda$
  - $\operatorname{var}(X) = \lambda$
  - intuice: $\lim_{n\to\infty} \text{Binom}\left(n, \frac{\lambda}{n}\right) = \text{Pois}(\lambda)$ (máme n nosičů, kteří mají n-krát kvalitnější džbány)
- **standartní normální**
  - $X \sim N(0,1)$
  - $f_{X(x)} = \varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$
  - $F_X = \Phi$
  - $\mathbb{E}(X) = 0$, $\operatorname{var}(X) = 1$
- **normální**
  - $X \sim N(\mu, \sigma)$
  - $N(\mu, \sigma) = N(0,1) \cdot \sigma + \mu$
  - $f_X = \frac{1}{\sigma} \varphi\left(\frac{x-\mu}{\sigma}\right)$
  - $\mathbb{E}(X) = \mu$, $\operatorname{var}(X) = \sigma^2$
- **exponenciální** (opravářovo čekací) – jak dlouho čeká opravář, než dorazí další nosič
  - $X \sim \text{Exp}(\lambda)$
  - $f_{X(x)} = \begin{cases} 0 & \text{pro } x \leq 0 \\ \lambda e^{-\lambda x} & \text{pro } x \geq 0 \end{cases}$
  - $F_{X(x)} = \begin{cases} 0 & \text{pro } x \leq 0 \\ 10 e^{-\lambda x} & \text{pro } x \geq 0 \end{cases}$
  - $\mathbb{E}(X) = \frac{1}{\lambda}$
  - $\operatorname{var}(X) = \frac{1}{\lambda^2}$

[1] vodu ve džbánu, jehož ucho se utrhlo, donesu domů (protože takový džbán se dá stále držet oběma rukama), ale je to otrava, takže si doma vezmu nový

5.3. Limitní věty

5.3.1. zákon velkých čísel

- stejně rozdělené $X_1 ... X_n$:
- $\overline{X_n} = \frac{1}{n}(X_1 + ... + X_n)$
- silný:

$$P\Big(\lim_{x\to\infty} \overline{X_n} = \mathbb{E}(X)\Big) = 1$$

- slabý:

$$\lim_{x\to\infty} P\Big(|\overline{X_n} - \mathbb{E}(X)| > \varepsilon\Big) = 0$$

5.3.2. centrální limitní věta

- stejně rozdělené $X_1 ... X_n$ s $\mathbb{E}(X) = \mu$ a $\text{var}(X) = \sigma^2$
- $$Y_n = \frac{(X_1 + ... + X_n) - n\mu}{\sqrt{n} \cdot \sigma}$$
- $$\lim_{n\to\infty} F_{Y_n}(x) = \Phi(x) \quad \forall x \in \mathbb{R}$$

5.4. Bodové odhady

- alespoň jedna metoda pro jejich tvorbu
- **výběrový průměr a rozptyl**
  - $\overline{X}_n = \frac{1}{n}\sum_i X_i$
    - konzistentní nestranný odhad μ
  - $\overline{S}_n = \frac{1}{n}\sum_i \left(X_i - \overline{X}_n\right)^2$
    - konzistentní asymptoticky nestranný odhad σ^2
  - $\hat{S}_n = \frac{1}{n-1}\sum_i \left(X_i - \overline{X}_n\right)^2$
    - konzistentní nestranný odhad σ^2
- **metoda momentů**
  - $m_r(\theta) = \mathbb{E}(X(\theta)^r)$
  - $\widehat{m_r} = \frac{1}{n}\sum_i X_i^r$
  - řešíme soustavu rovnic $m_{r(\theta)} = \widehat{m_r}$ pro různá r
- **maximum likelihood estimate**
  - pro realizaci náhodného výběru $x = (x_1 ... x_n)$ vybíráme hodnotu θ, pro kterou je pravděpodonost tohoto výběru největší
  - maximalizujeme $L(x;\theta)$ podle θ
    - $$L(x;\theta) = \begin{cases} p_{X(x;\theta)} = \prod_i p_{X_i}(x_i;\theta) \\ f_{X(x;\theta)} = \prod_i f_{X_i}(x_i;\theta) \end{cases}$$

5.5. Intervalové odhady

- **metoda založená na aproximaci normálním rozdělením**
  - odhadujeme $N(\theta, \sigma^2)$, známe σ^2, hledáme θ
    - $$\overline{X} \pm z_{\frac{\alpha}{2}} \cdot \frac{\sigma}{\sqrt{n}}$$
    - $z_{\frac{\alpha}{2}} = \Phi^{-1}(1 - \frac{\alpha}{2})$ – kvantilová fce $N(0,1)$
    - při dostatečně velkém n to platí díky CLV i pro libovolné rozdělení
  - odhadujeme $N(\theta, \sigma^2)$, neznáme σ^2 ani θ

–
$$\hat{\theta} \pm \boldsymbol{t}_{\frac{\alpha}{2}} \cdot \frac{\widehat{S_n}}{\sqrt{n}}$$

– $\frac{\overline{X}-\mu}{\frac{\widehat{S_n}}{\sqrt{n}}}$ už není normálně rozdělená n. v., ale má studentovo rozdělení s pdf $\Psi_{n-1}(x)$

– $t_{\frac{\alpha}{2}} = \Psi_{n-1}^{-1}(1-\frac{\alpha}{2})$ – kvantilová fce studentova rozdělení s $n-1$ stupni volnosti

5.6. Testování hypotéz

- základní přístup
  - $\boldsymbol{H_0}$ – nulová hypotéza
  - $\boldsymbol{H_1}$ – alternativní hypotéza
  - zamítáme či nezamítáme nulovou hypotézu
- chyby 1. a 2. druhu
  - **chyba 1. druhu** – trapas (hlásíme zajímavst, i když nenastala)
  - **chyba 2. druhu** – promarněná příležitost (nehlásíme zajímavst, i když bychom mohli)
- **hladina významnosti**
  - pravděpodobnost **chyby 1. druhu**, typicky $\alpha = 0.05$

6. Logika

6.1. Syntaxe

- znalost a práce se základními pojmy syntaxe výrokové a predikátové logiky (jazyk, otevřená a uzavřená formule, instance formule, apod.)
- **jazyk výrokové logiky** – množina všech prvovýroků, $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,)$
- **jazyk predikátové logiky** – signatura $\langle \mathcal{R}, \mathcal{F} \rangle$, $(=)$? , symboly volných proměnných, logické symboly
- **otevřená formule** – neobsahuje kvantifikátory
- **uzavřená formule** – neobsahuje volné proměnné
- **instance** – dosazení za volnou proměnnou
- **teorie** – libovolná spočetná (ne nutně konečná) množina výroků
- **term** – cokoliv co nevrací bool (proměnná, funkce nebo konstanta)

6.1.1. normální tvary výrokových formulí

- **PNF** – $(Q_1 x_1)(Q_2 x_2)...(Q_n x_n)\varphi$, kde φ je otevřená
  - převod vytýkáním
    1. $\neg(Qx)(\varphi) \sim \left(\overline{Q}x\right)(\neg\varphi)$
    2. $(Qx)(\varphi) \wedge \psi \sim (Qx)(\varphi \wedge \psi)$
    3. $(Qx)(\varphi) \vee \psi \sim (Qx)(\varphi \vee \psi)$
- **CNF** – konjunkce disjunkcí literálů $(p \vee \neg q \vee s) \wedge (\neg r) \wedge (\neg s \vee r)$
  - literál – $p, \neg p$
  - klauzule – disjunkce literálů
  - převod ekvivalentními úpravami
- **DNF** – disjunkce konjunkcí literálů $(p \wedge \neg q \wedge s) \vee (\neg r) \vee (\neg s \wedge r)$
  - převod ekvivalentními úpravami
- použití pro algoritmy (SAT, rezoluce)
  - rezoluce i SAT používají CNF

6.2. Sémantika

- **model teorie**
  - VL: ohodnocení prvovýroků, ve kterém platí všechny axiomy teorie

  - PL: struktura $\langle \mathcal{A}, \mathcal{R}^{\mathcal{A}}, \mathcal{F}^{\mathcal{A}} \rangle$ určující universum $\mathcal{A}$ a implementaci relačních a funkčních symbolů, ve které platí všechny axiomy teorie
- pravdivost, lživost, nezávislost formule vzhledem k teorii
  - formule je **pravdivá**, pokud platí ve všech modelech teorie (všechna ohodnocení, které splňují teorii, splňují i formuli)
  - formule je **lživá**, pokud neplatí v žádném modelu teorie (všechna ohodnocení, které splňují teorii, nesplňují formuli)
  - formule je **nezávislá**, pokud platí v některém modelu a v jiném ne
- splnitelnost, tautologie, důsledek
  - formule je **splnitelná**, pokud není lživá
  - formule je **tautologie**, pokud je pravdivá v logice ($p \vee \neg p$)
- analýza výrokových teorií nad konečně mnoha prvovýroky
  - prostě to zanalyzuješ ne? tablem? možná?

6.3. Extenze teorií

- **extenze** E teorie T je taková teorie, ve které jsou **pravdivé** alespoň ty modely, které jsou pravdivé v T
  - **konzervativní** – nemění množinu **důsledků** (pravdivých modelů) v původním jazyce (ale může přidávat nové důsledky nad novými symboly)
  - **jednnoduchá** – nemění jazyk
- schopnost porovnat sílu teorií
  - stronk
- skolemizace
  - PNF $\rightarrow$ otevřená formule
    - $(\forall x)(\varphi(x)) \sim \varphi(x)$
    - $(\exists x)(\varphi(x)) \sim \varphi(c)$ (nový konstatntí symbol)
    - $(\forall x)(\exists y)(\varphi(x, y)) \sim \varphi(x, f(x))$ (nový funkční symbol)

6.4. Dokazatelnost

- pojem formálního důkazu, zamítnutí
  - **důkaz** je mechanicky ověřitelný certifikát o tom, že φ platí, provedený na syntaktické úrovni
  - **zamítnutí** je důkaz sporu, protipříklad
- schopnost práce v některém z formálních dokazovacích systémů (např. tablo metoda, rezoluce, Hilbertovský kalkul)
  - tablo
    - $\varphi \wedge \psi \text{ — } \varphi \text{ — } \psi$
    - $\varphi \vee \psi \left\langle \begin{matrix} \varphi \\ \psi \end{matrix} \right.$
    - $(\forall x)\varphi(x) \text{ — } \varphi(x/t_i)$ – všechny existující termy
    - $(\exists x)\varphi(x) \text{ — } \varphi(x/c_i)$ – nový konstantní term

6.5. Věty o kompaktnosti a úplnosti výrokové a predikátové logiky

- znění a porozumění významu
- použití na příkladech, důsledky
- **věta o kompaktnosti**: teorie má model kdyžž každá její konečná část má model
- **větu o úplnosti** Bulínova skripta v tomto kontextu nezmiňují, v obecnosti úplnost znamená, že umíme dokázat všechná pravdivá tvrzení

6.6. Rozhodnutelnost

- teorie je **rekurzivně axiomatizovatelná** $\Leftarrow \exists$ algoritmus ověřující axiom $\varphi \in T$
- teorie je **rozhodnutelná**, $\Leftarrow \exists$ algoritmus, $\forall \varphi$ doběhne, ověří $T \vDash \varphi$
- teorie je **částečně rozhodnutelná**, $\Leftarrow \exists$ algoritmus $\forall \varphi$ ověří $T \vDash \varphi$, pro $T \nvDash \varphi$ nemusí doběhnout
- teorie je **kompletní**, pokud je bezesporná a každá uzavřená formule je pravdivá nebo lživá
- RA => částečně rozhodnutelná
- RA + kompletní => rozhodnutelná
- pojem kompletnosti a její kritéria, význam pro rozhodnutelnost
  - ω-kategorické kritérium kompletnosti
    - ani náhodou
- příklady rozhodnutelných a nerozhodnutelných teorií
  - nerozhodnutelné - Halting problem, Hilbertův desátý problém

. Společná informatika

1. Automaty a jazyky

1.1. Regulární jazyky

- deterministický a nedeterministický konečný automat
  - **DFA**: $(Q, \Sigma, \delta, q_0, F)$
    - Q – množina stavů
    - Σ – abeceda
    - $\delta : Q \times \Sigma \to Q$ – přechodová fce
    - $q_0 \in Q$ – počáteční stav
    - $F \subset Q$ – množina koncových stavů
  - **NFA**: $(Q, \Sigma, \delta, Q_0, F)$
    - Q – množina stavů
    - Σ – abeceda
    - $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ – přechodová fce
    - $Q_0 \subset Q$ – množina počáteční stavů
    - $F \subset Q$ – množina koncových stavů
- regulární gramatiky
  - pravá lineární $A \to \omega B$, $A \to \omega$, $A, B, \in V, \omega \in T^*$
- regulární výrazy $\epsilon, \emptyset, \mathtt{a}, \alpha + \beta, \alpha\beta, \alpha^*, (\alpha)$

1.2. Bezkontextové jazyky

- bezkontextové gramatiky, jazyk generovaný gramatikou
  - $A \to \omega, A \in V, \omega \in (V \cup T)^*$
- zásobníkový automat, třída jazyků přijímaných zásobníkovými automaty
  - **PDA:** $(Q, \Sigma, \Gamma, \delta, q_0, F)$
    - Q – množina stavů
    - Σ – vstupní abeceda
    - Γ – zásobníková abeceda
    - $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$ – přechodová fce
    - $q_0 \in Q$ – počáteční stav
    - $Z_0 \in \Gamma$ – počáteční zásobníkový symbol
    - $F \subset Q$ – množina koncových stavů

1.3. Rekurzivně spočetné jazyky

- gramatika typu 0
  - $\alpha \to \omega; \alpha, \omega \in (V \cup T)^*$
- **Turingův stroj**: $(Q, \Sigma, \Gamma, \delta, q_0, F)$
  - Q – množina stavů
  - Σ – vstupní abeceda
  - Γ – zásobníková abeceda
  - $\delta : (Q \setminus F) \times \Gamma \to Q \times \Gamma \times \{L, R\}$ – přechodová fce
  - $q_0 \in Q$ – počáteční stav
  - $B \in \Gamma \setminus \Sigma$ – symbol prázdné pásky
  - $F \subset Q$ – množina koncových stavů
- algoritmicky nerozhodnutelné problémy
  - Halting problem/diagonální jazyk
  - PCP

1.4. Chomského hierarchie

- schopnost zařazení konkrétního jazyka do Chomského hierarchie (zpravidla sestrojení odpovídajícího automatu či gramatiky)

2. Algoritmy a datové stuktury

2.1. Časová složitost algoritmů

- časová a prostorová složitost algoritmu
  - **časová složitost**: celkový počet kroků RAMu jako funkce délky vstupu $f : \mathbb{N} \to \mathbb{R}$
  - **prostorová složitost**: rozsah indexů použitých buněk paměti jako funkce délky vstupu $f : \mathbb{N} \to \mathbb{R}$
- měření velikosti dat
  - jednotková cena, neomezená čísla – dá se hackovat prostorová složitost kódováním do jednoho čísla
  - jednotková cena, konstantou omezená velikost čísel (a tím indexovatelné paměti)
  - jednotková cena, polynomem omezená velikost čísel (a tím indexovatelné paměti)
  - logaritmická cena, neomezená čísla – dobré, ale otrava s tím počítat
  - poměrná logaritmická cena, neomezená čísla – cena je poměr logaritmů velikosti vstupu a čísel, pro polynomy konstantní
- složitost v nejlepším, nejhorším a průměrném případě
- asymptotická notace
  - $f \in \mathcal{O}(g) \equiv \exists n_0, c : \forall n \geq n_0 f(n) \leq cg(n)$
  - $f \in \Omega(g) \equiv \exists n_0, c : \forall n \geq n_0 f(n) \geq cg(n)$
  - $f \in \Theta(g) \equiv f \in \mathcal{O}(g) \wedge f \in \Omega(g)$

2.2. Třídy složitosti

- třídy P a NP
  - **P** – třída rozhodovacích problémů řešitelných v polynomiálním čase
  - **NP** – $L \in$ NP kdyžž $\exists K \in$ P, polynom g, $\forall x L(x) = 1 \Leftrightarrow \exists y, |y| \leq g(|x|) : K(x, y) = 1$
- převoditelnost problémů, NP-těžkost a NP-úplnost
  - **převod** – platí $A \to B$, když $\exists$ funkce $f : \{0,1\}^* \to \{0,1\}^*$ taková, že $\forall x : A(x) = B(f(x))$ a f běží v polynomiálním čase
  - problém je **NP-těžký**, pokud na něj lze převést libovolný problém v NP
  - problém je **NP-úplný**, pokud je NP-těžký a v NP

- příklady NP-úplných problémů a převodů mezi nimi
  - SAT → 3-SAT – přidáváme nové literály, $(\alpha \vee \beta) \to (\alpha \vee x) \wedge (\neg x \vee \beta)$
  - 3-SAT → NzMna – klauzule tvoří trojúhélníky, spojujeme x s $\neg x$
  - 3-SAT → 3,3-SAT – proměnné rozdělíme a zařídíme stejně ohodnocení kolečkem implikací
  - 3,3-SAT → 3D-párování – proměnná se vyskytuje jen dvakrát – cyklus KDKD, Z čouhají, lze spárovat pouze Z naproti sobě, klauzule – KDZZZ

2.3. Metoda rozděl a panuj

- princip rekurzivního dělení problému na podproblémy
  - problém velikosti n rozdělíme na dva podproblémy velikosti $\frac{n}{2}$
- výpočet složitosti pomocí rekurentních rovnic
- Master theorem (kuchařková věta) (bez důkazu)
  - $T(n) = a \cdot T\left(\frac{n}{b}\right) + \Theta(n^c)$

$$= a^{\log_b n} + \Theta(n^c) \cdot \sum_{i=0}^{\log_b n} \left(\frac{a}{b^c}\right)^i$$

$$= n^{\log_b a} + \Theta(n^c) \cdot \sum_{i=0}^{\log_b n} \left(\frac{a}{b^c}\right)^i$$

  - $T(n) = \Theta(n^c \log n)$ pokud $\frac{a}{b^c} = 1$
  - $T(n) = \Theta(n^c)$ pokud $\frac{a}{b^c} < 1$
  - $T(n) = \Theta\left(n^{\log_b a}\right)$ pokud $\frac{a}{b^c} > 1$
- aplikace
  - Mergesort
  - násobení dlouhých čísel
    - $$x = \left(A \ll \frac{n}{2}\right) + B$$
      $$y = \left(C \ll \frac{n}{2}\right) + D$$
      $$xy = (AC \ll n) + \left(AD \ll \frac{n}{2}\right) + \left(BC \ll \frac{n}{2}\right) + BD$$
      $$(A+B)(C+D) = AC + AD + BC + BD$$
      $$xy = (AC \ll n) + ((A+B)(C+D) - AC - BD) \ll \frac{n}{2} + BD$$

2.4. Binarní vyhledávací stromy

- **vyhledávací strom**: klíč v každém vrcholu je větší než všechny klíče levého podstromu a menší než pravého
- operace s nevyvažovanými stromy – show, find, min, insert, delete
- **AVL strom**: rozdíl v hloubkách podstromů je max 1

2.5. Třídění

- primitivní třídicí algoritmy (Bubblesort, Insertsort)
- Quicksort – pivot p, $< p$, $> p$, rekurze
- dolní odhad složitosti porovnávacích třídicích algoritmů
  - $n \log n$

2.6. Grafové algoritmy

- prohledávání do šířky a do hloubky
  - **BFS**: dělá vrstvy, hledá nejkratší vzdálenost, používá frontu
  - **DFS**: zpětná hrana není nikdy most, xyyx – dopředná/stromová, yxxy – zpětná, yyxx – příčná, xxyy – nenastane
- topologické třídění orientovaných grafů
  - seřadím podle pořadí uzavírání v DFS, pořadí bude opačné
- nejkratší cesty v ohodnocených grafech (Dijkstrův a Bellmanův-Fordův algoritmus)
  - obecný relaxační algoritmus
  - **Dijkstra** – halda vzdáleností
  - **BF** – fronta podle pořadí otevíraných vrcholů – může otevírat už jednou zavřené vrcholy
- minimální kostra grafu (Jarníkův a Borůvkův algoritmus)
  - **Jarníkův** – přidáváme nejmenší hranu mezi kostrou a zbytkem grafu
  - **Borůvkův** – začínáme s n kostrami, ke každé přidáváme nejmenší hranu
  - **Kruskalův** – začínáme s n kostrami, vybíráme nejmenší hrany v celém grafu, přidáváme, netvoří-li cyklus
- toky v sítích (Ford-Fulkerson algoritmus)
  - hledáme zlepšující cesty, dostaneme maximální řez daný nasycenými cestami

3. Programovací jazyky

Některé následující body definují varianty požadavků pro různé individuální volby povinně volitelných předmětů. Vyžaduje se zvládnutí všech bodů bez označení Ⓥ a zvládnutí všech bodů s označením Ⓥ pro jeden z jazyků C#, C++ nebo Java.

- Koncepty pro abstrakci, zapouzdření a polymorfismus.
  - související konstrukty programovacích jazyků
    - třídy, rozhraní, metody, datové položky, dědičnost, viditelnost
    - Ⓥ zapouzdření poskytované moduly v Javě
  - (dynamický) polymorfismus, statické a dynamické typování
  - jednoduchá dědičnost
    - Ⓥ virtuální a nevirtuální metody v C++ a C#
  - vícenásobná dědičnost a její problémy
    - Ⓥ vícenásobná a virtuální dědičnost v C++
    - Ⓥ interfaces a defaultní metody v Javě
    - Ⓥ interfaces v C#
  - implementace rozhraní (interface)
  - vhodné použití uvedených konceptů
- Primitivní a objektové typy a jejich reprezentace.
  - číselné a výčtové typy
  - Ⓥ ukazatele a reference v C++
    - **& je lvalue, && je rvalue**
  - Ⓥ hodnotové a referenční typy v C#
    - **value**
      - int, float, bool, char
      - struct, enum, nullable
    - **reference**
      - class, interface, record, delegate (lambda)
      - dynamic, object, string
  - Ⓥ reference, imutabilní typy a boxing v C# a Javě

  - **boxing**: zabalení value typu do objectu
- Generické typy a funkcionální prvky (procedurálních programovacích jazyků).
  - Ⓥ šablony (templates) a statický polymorfismus v C++
  - Ⓥ generické typy v Javě a C# (bez omezení typových parametrů)
  - Ⓥ typy reprezentující funkce v C++, C#, nebo Javě
  - lambda funkce a funkcionální rozhraní
- Manipulace se zdroji a mechanizmy pro ošetření chyb.
  - správa životního cyklu zdrojů v případě výskytu chyb
    - Ⓥ RAII v C++
      - **Resource acquisition is initialization** – spojuje lifetime objektu a nějakého zdroje (mutexu, paměti)
    - Ⓥ using v C#
      - jako `with` v pythonu
    - Ⓥ try-with-resources v Javě
  - konstrukce pro obsluhu a propagaci výjimek
- Životní cyklus objektů a správa paměti.
  - alokace (alokace statická, na zásobníku, na haldě)
  - inicializace (konstruktory, volání zděděných konstruktorů)
  - destrukce (destruktory, finalizátory)
  - explicitní uvolňování objektů, reference counting, garbage collector
- Vlákna a podpora synchronizace.
  - reprezentace vláken v programovacích jazycích
  - specifikace funkce vykonávané vláknem a základní operace na vlákny
  - časově závislé chyby a mechanizmy pro synchronizaci vláken
- Implementace základních prvků objektových jazyků.
  - základní objektové koncepty v konkrétním jazyce
  - implementace a interní reprezentace primitivních typů
  - implementace a interní reprezentace složených typů a objektů
  - implementace dynamického polymorfismu (tabulka virtuálních metod)
- Nativní a interpretovaný běh, řízení překladu a sestavení programu.
  - reprezentace programu, bytecode, interpret jazyka
  - just-in-time (JIT) a ahead-of-time (AOT) překlad
  - proces sestavení programu, oddělený překlad, linkování
  - staticky a dynamicky linkované knihovny
  - běhové prostředí procesu a vazba na operační systém

4. Architektura počítačů a operačních systémů

- Základní architektura počítače, reprezentace čísel, dat a programů.
  - reprezentace a přístup k datům v paměti, adresa, adresový prostor
  - ukládání jednoduchých a složených datových typů
  - základní aritmetické a logické operace
- Instrukční sada, vazba na prvky vyšších programovacích jazyků.
  - Implementovat běžné programové konstrukce vyšších jazyků (přiřazení, podmínka, cyklus, volání funkce) pomocí instrukcí procesoru
  - Zapsat běžnou konstrukci vyššího jazyka (přiřazení, podmínka, cyklus, volání funkce), která odpovídá zadané sekvenci (vysvětlených) instrukcí procesoru
- Podpora pro běh operačního systému.
  - privilegovaný a neprivilegovaný režim procesoru

  - jádro operačního systému
- Rozhraní periferních zařízení a jejich obsluha.
  - Popsat roli řadiče zařízení při programem řízené obsluze zařízení (PIO), pro zadané adresy a funkce vstupních a výstupních portů implementovat programem řízenou obsluhu zadaného zařízení (myš, disk)
  - Popsat roli přerušení při programem řízené obsluze zařízení (PIO), na úrovni vykonávání instrukcí popsat reakci procesoru (hardware) a operačního systému (software) na žádost o přerušení
- Základní abstrakce, rozhraní a mechanizmy OS pro běh programů, sdílení prostředků a vstup/výstup.
  - neprivilegované (uživatelské) procesy
  - sdílení procesoru
    - procesy, vlákna, kontext procesu a vlákna
    - přepínání kontextu, kooperativní a preemptivní multitasking
    - plánování běhu procesů a vláken, stavy vlákna
  - sdílení paměti
    - Vysvětlit rozdíl mezi virtuální a fyzickou adresou a identifikovat, zda se v zadaném kontextu či fragmentu kódu používá virtuální nebo fyzická adresa
    - Na zadaném příkladu identifikovat a vysvětlit význam komponent virtuální a fyzické adresy (číslo stránky, číslo rámce, offset)
    - Pro konkrétní adresy a obsah jednoúrovňové stránkovací tabulky řešit úlohy překladu adres
    - Vysvětlit roli virtuálních adresových prostorů v ochraně paměti procesů a vláken
  - sdílení úložného prostoru
    - soubory, analogie s adresovým prostorem
    - abstrakce a rozhraní pro práci se soubory
- Paralelismus, vlákna a rozhraní pro jejich správu, synchronizace vláken.
  - časově závislé chyby (race conditions)
  - kritická sekce, vzájemné vyloučení
  - základní sychronizační primitiva, jejich rozhraní a použití
    - zámky
    - aktivní a pasivní čekání

. Systémové programování

1. Architektura počítačů

- Výkonnost počítače a procesoru, metriky a omezení
  - Vyjádřit a na příkladech demonstrovat vztah mezi dobou běhu programu a metrikami na úrovni architektury (CPI, IPC)
  - Popsat vliv instrukčního mixu na hodnoty metrik CPI a IPC, identifikovat typické hodnoty CPI a IPC
    - **některé instrukce jsou rychlejší než jiné**, aritmetika typicky 1 CPI, jumpy 3 CPI, load 5 CPI, ALE, často má vliv i paralelismus
  - Formulovat Amdahlův zákon a použít ho pro odhad limitů zrychlení díky paralelismu
    - $t_{\text{para}} = t_{\text{non-paralelizable}} + \frac{t_{\text{paralelizable}}}{n}$
- Zpracování instrukcí procesorem, paralelismus, predikce a spekulace
  - Na diagramu datové cesty procesoru vysvětlit postup zpracování základních instrukcí (add, load, store, branch)

- Popsat kroky vykonané procesorem pro obsluhu přerušení a obsluhu výjimek
  - **zastavení, identifikace příčiny, uložení stavu, skok na handler**
- Na příkladu kódu popsat zřetězené zpracování (pipelining), identifikovat datové závislosti a popsat jejich řešení, odhadnout zrychlení
- Popsat zřetězené zpracování (pipelining) za přítomnosti (také podmíněných a nepřímých) skoků, popsat a na příkladu identifikovat rozhodnutí základních prediktorů (saturating counter), popsat spekulativní vykonávání kódu a jeho vliv na výkon
- Popsat rozdíl mezi hardware cores a hardware threads a jeho vliv na výkon
  - **core** – více-méně oddělený procesor
  - **thread** – vlastní kontext (virtuální registry, PC), sdílená ALU, většina ostatního

- Architektura paměťového subsystému, architektura cache
  - Použít metriky hit či miss ratio a access latency pro odhad rychlosti přístupu do paměti
  - Pro zadanou architekturu (přímo mapovaná, množinově či plně asociativní) a relevantní parametry (velikost, stupeň asociativity, victim replacement policy) cache a zadanou sekvenci přístupů do paměti určit chování cache (hit/miss, 3C model, obsazení konkrétních cache lines)
    - **CCC** – cold, capacity, conflict
  - Diskutovat roli vrstev ve víceúrovňové architektuře cache a identifikovat obvyklé parametry vrstev
    - **L1** – 64K (per core), **L2** – 512K (per core or 2), **L3** – 32M (per CPU)
- Multi-core a multi-socket systémy, koherence cache
  - Na diagramu identifikovat UMA a NUMA architektury
    - **(non-)uniform memory access**
  - Popsat záruky poskytované mechanismy cache coherence při přístupu do paměti
  - Popsat a na příkladech ilustrovat fungování cache coherence protokolů (IV, MSI, MESI)
  - Popsat a na příkladech kódu ilustrovat false sharing

2. Operační systémy

- Spouštění procesů, dynamicky linkované knihovny, volací konvence
  - Použít systémové API (fork, exec, join …) pro spuštění a (počkání na) ukončení procesu
  - Použít pthread API (pthread_create, pthread_join …) pro spuštění a (počkání na) ukončení vlákna
    - `int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine) (void *), void *arg)`
    - `int pthread_join(pthread_t thread, void **retval);`
  - Identifikovat statické a dynamické linkování, použít obvyklé nástroje pro vytvoření staticky a dynamicky linkovaného programu
    - `gcc -static`
  - Použít obvyklé nástroje pro identifikaci linkovaných symbolů, identifikovat externí a exportované symboly
    - `ldd <binary>`
  - Porovnat použití a režii statického a dynamického linkování
    - **statické** – + dá se optimalizovat, - velké, když se nezoptimalizuje, musí se překompilovat při aktualizaci
    - **dynamické** – (potenciálně) malé binárky
  - Identifikovat na ukázkách kódu pozičně závislý a pozičně nezávislý program, použít překladač pro vytvoření takového programu
    - `gcc -no-pie`

- ▸ Identifikovat a vysvětlit použití absolutních adres pro adresaci proměnných na ukázkách kódu
- ▸ Identifikovat a vysvětlit použití relativních adres pro adresaci proměnných na ukázkách kódu
- ▸ Porovnat použití a režii absolutní a relativní adresace
- ▸ Popsat obvyklé umístění argumentů, proměnných a dalších informací na zásobníku programů
- ▸ Popsat obvyklé použití registrů procesoru uvnitř funkcí a při volání funkcí

- Paralelismus a synchronizace na multiprocesorech
  - ▸ Popsat a sestavit příklad race condition nad sdíleným čítačem s použitím interleaving sémantiky běhu programu
  - ▸ Popsat a sestavit příklad race condition nad sdíleným seznamem s použitím interleaving sémantiky běhu programu
  - ▸ Popsat sémantiku prostředků pro implementaci synchronizace na multiprocesorech (IPI, T&S, CAS, LL/SC)
    - **IPI** – interprocessor interrupt
    - **T&S** – test and set (atomické „zapiš hodnotu a vrať původní“) (lol mělo by to být spíš set and test)
    - **CAS** – compare and swap (atomické „zapiš hodnotu x, pokud je tam právě zapsané y“)
    - **LL/SC** – load-link/store-conditional (dvě instrukce, první načte hodnotu a druhá zapíše, pouze pokud na danou paměť nikdo od té doby nesáhl)
  - ▸ Implementovat spin lock s použitím zadaného prostředku (T&S, CAS, LL/SC)
  - ▸ Implementovat blokující zámek s použitím rozhraní futex
    - **futex** – fast userspace mutex, atomic ints in userspace, syscalls only when needed
- Rozhraní pro synchronizaci
  - ▸ Popsat sémantiku atomických typů, bariér, zámků, semaforů, condition variables jako nástrojů pro synchronizaci
  - ▸ Použít zadaný nástroj (atomické typy, bariéry, zámky, semafory, condition variables) pro odstranění race condition v ukázkách kódu
  - ▸ Použít zadaný nástroj (atomické typy, bariéry, zámky, semafory, condition variables) pro implementaci problému producent-konzument
- Správa paměti na multiprocesorech, alokátory, garbage collection
  - ▸ Na zadaném příkladu datových struktur heap alokátoru popsat jeho funkci a odhadnout prostorovou a časovou režii
  - ▸ Na zadaném příkladu obsahu heapu demonstrovat algoritmy mark and sweep a copying garbage collection
    - **mark and sweep** – označím a pak uklízím
    - **copying gc** – používané objekty rovnou kopíruju do druhé půlky paměti
  - ▸ Ovládat související terminologii (heap, mutator, collector, reachability, garbage)
- Rozhraní pro práci se soubory, paměťově mapované soubory
  - ▸ Použít systémové API (open, seek, read, write, close, readv, writev, mmap) pro práci se soubory
    - **readv / writev** – více sekvenčních čtení/zápisů do seznamu bufferů a velikostí (jeden syscall!)
  - ▸ Popsat interakci systému souborů a správy paměti při přístupu k souborům pomocí mapování (mmap)

  - `void *mmap(void addr[.length], size_t length, int prot, int flags, int fd, off_t offset);`
    - `addr` bude typicky `0`, kernel pickne
- Interní struktura základních systémů souborů
  - Vyjmenovat obvyklá metadata asociovaná se souborem
  - Na zadaném příkladu datových struktur systému souborů (FAT, inode, extent, dentry, bitmap) popsat realizaci operací se souborem (open, read, write)
- Princip komunikace mezi periferiemi a operačním systémem
  - Popsat a na příkladech kódu identifikovat jednotlivé kroky obsluhy žádosti o přerušení od zařízení
  - Popsat omezení nástrojů pro synchronizaci uvnitř kódu obsluhujícího žádosti o přerušení
  - Popsat jednotlivé kroky přenosu dat mezi zařízením a pamětí při použití DMA

3. Počítačové sítě

- Linková vrstva, adresace v Ethernetu
  - Vysvětlit základní vlastnosti a úkoly vrstvy síťového rozhraní
  - Vyjmenovat/identifikovat nejdůležitější položky ethernetové rámce, vysvětlit pojem MAC adresy a její význam
  - Propojovací zařízení na úrovni linkové vrstvy, přepínače (switch)
    - Popsat princip činnosti linkového rozhraní (přijímání, zpracování a odesílání rámců)
    - Popsat principy základní optimalizace síťového toku (filtrování, cílený forwarding, metoda zpětného učení)
- Síťová vrstva, adresace v IPv4 a IPv6, statické směrování, NAT
  - Vysvětlit základní vlastnosti a úkoly síťové vrstvy a její návaznost na vrstvy síťového rozhraní (hop-to-hop komunikace, přímé vs. nepřímé doručování)
  - Vymezení problému směrování (routing) a předávání (forwarding), směrovací tabulky
    - Vysvětlit na příkladu strukturu záznamů ve směrovacích tabulkách a význam jednotlivých položek
    - Na zadaném příkladu sítě (topologie, adresy, obsahy tabulek) popsat postup zpracování (doručení) konkrétního IP datagramu
  - IP adresy (v4 i v6)
    - Vysvětlit pojem IP adresa a síťová maska a jejich význam pro síťovou vrstvu (doručování datagramů)
    - Popsat způsob přidělování IP adres a speciální prostory adres (privátní adresy, multicast)
    - Popsat na konkrétním problému možnou aplikaci NA(P)T překladu a vysvětlit jeho užitečnost vs. nevýhody
    - Na příkladu dané sítě s použitím NA(P)T vysvětlit průchod datového paketu z vnitřní sítě do vnější a naopak
  - Protokol IPv4, základní vlastnosti, struktura IPv4 datagramu
    - Na příkladu identifikovat hlavní položky hlavičky (verze, time-to-live, protokol, kontrolní součet, adresy)
    - Na daném příkladu hlaviček datagramu (a konfigurace routeru) vysvětlit, jak bude datagram zpracován
    - Včetně atypických situací jako je např. zahození datagramu
    - Vysvětlit základní princip fungování nástroje traceroute
    - Popsat princip IPv4 fragmentace (s návazností na hlavičky IP datagramu) a navrhnout alespoň jeden postup, jak se fragmentaci vyhnout

  - Protokol ICMPv4, jeho účel a použití, základní typy zpráv, fungování nástroje Ping
  - Popsat protokol ARP a jeho význam pro interakci mezi síťovou a linkovou vrstvou
  - Popsat protokol DHCP (jak probíhá přidělování adres)
- Transportní vrstva, adresace v TCP a UDP, spolehlivost, řízení toku
  - Vysvětlit základní vlastnosti a úkoly transportní vrstvy a nejdůležitější protokoly a koncepty (TCP, UDP, porty)
  - Vysvětlit způsoby alokace portů, uvést příklady dobře známých portů
  - Na příkladu identifikovat a vysvětlit nejdůležitější položky UDP packetu a jejich souvislost s IP protokolem
  - Na příkladu identifikovat a vysvětlit nejdůležitější položky TCP packetu, vysvětlit souvislost tohoto paketu s přenášeným proudem dat
  - Popsat algoritmy navazování a ukončení spojení v TCP (three-way handshake)
  - Popsat princip řízení toku (flow control) a předcházení zahlcení (congestion control) v TCP, uvést konkrétní příklad, jak může dojít k zahlcení a jak se s tím TCP vypořádá
  - Vysvětlit, jak ovlivňuje dynamický překlad adres (NAPT) obsah hlaviček TCP a UDP paketů
- Aplikační rozhraní a abstrakce pro síťovou komunikaci
  - Napsat pseudo-kód části síťové aplikace (klient nebo server) řešící zadaný problém s použitím socket API (např. statický web server)
  - Na příkladu vysvětlit jednotlivé části URI/URL a jejich význam
  - Popsat princip fungování systému DNS a DNS protokolu (základní typy a význam záznamů, zóny, DNS servery, vyřizování DNS dotazů, bezpečnostní rizika)
  - Popsat jednotlivé (síťové) kroky (např. překlad doménového jména na IP adresu, navázání spojení na konkrétních portech, odeslání a příjem zpráv s konkrétním obsahem, …), které musí dobře známá síťová aplikace (prohlížeč, mailový klient, …) provést, když vykonává svou činnost (zobrazuje stránku, odesílá mail, …)
- Zabezpečení komunikace, autentizace, šifrování
  - Popsat jednoduché metody detekce poškozených bloků (kontrola parity, kontrolní součty) s uvedením konkrétních příkladů na různých vrstvách
  - Popsat principy a typy firewallů (co řeší a jak se konfigurují)
  - Na konkrétním příkladu problému (a konfigurace sítě) doporučit vhodné zabezpečení (nastavení firewallů, demilitarizovaná zóna, aplikační brány)
  - Popsat základní principy symetrického a asymetrického šifrování, význam použití certifikátů, zabezpečení síťové komunikace SSL/TLS

4. Překladače

- Lexikální analýza
  - Použít regulární výraz pro definici lexikálního elementu
  - Použít stavy lexikálního analyzátoru (např. flex) pro definici složitějších lexikálních elementů (např. řetězec s escape-sekvencemi)
- Syntaktická analýza
  - Transformovat opakování a alternativy (např. v syntaktických diagramech) na bezkontextovou gramatiku
  - Zapsat gramatikou typické konstrukce programovacích jazyků
  - Odstranit nejednoznačnosti v gramatice (např. if-then-else)
  - Odstranit LL(1) konflikty v gramatice (např. levá rekurze)
- Sémantická analýza
  - Definovat pomocí atributů propojení lexikální a syntaktické analýzy

- Použít syntetizované atributy (např. bison) pro předání mezivýsledků (např. při vyhodnocování výrazu)
- Použít atributovou gramatiku (včetně dědičných atributů) pro řešení složitějších konstrukcí
- Mezikód
  - Zkonstruovat control-flow graf pro danou proceduru (v C++ nebo v C#/Java)
  - Přepsat úsek zdrojového kódu (v C++ nebo v C#/Java) do zvoleného mezikódu

5. Pokročilé programovací jazyky

Některé následující body definují varianty požadavků pro různé individuální volby povinně volitelných předmětů. Vyžaduje se zvládnutí všech bodů bez označení Ⓥ a zvládnutí všech bodů s označením Ⓥ pro jeden z jazyků C#, C++ nebo Java. Dále tato specializace vyžaduje zvládnutí společných požadavků z informatiky v sekci Programovací jazyky ve stejné variantě.

- Pokročilé využití generických typů a metod
  - Ⓥ Vysvětlit fungování generických typů a metod s typovým omezením (constraints) v C# nebo Javě
  - Ⓥ Umět vhodně implementovat generické typy a metody s typovým omezením v C# nebo Javě
  - Ⓥ Umět implementovat šablony (templates) typů a funkcí, a vhodně využít konceptu duck-typingu v C++
  - Ⓥ Využít parciální specializaci na vhodných příkladech v C++
  - Ⓥ Aplikovat perfect-forwarding a variadické šablony v C++
- Funkcionální prvky
  - Lambda výrazy, closure, a variable capture
    - Ⓥ Vysvětlit fungování type inference pro parametry a návratovou hodnotu lambda výrazů v C#, C++ nebo Javě
    - Ⓥ Vysvětlit princip fungování a rozdíly mezi capture-by-copy, capture-by-reference, a no-capture sé- mantikami v C++
    - Na příkladu umět zvolit vhodný způsob variable capture
    - Ⓥ Vysvětlit princip fungování, výhody a nevýhody capture-by-“move-to-closure” sémantiky v C#
    - Umět ukázat nutnost ruční emulace capture-by-value
    - Ⓥ Vysvětlit princip fungování variable capture a potřebu final nebo “effectively final” zachycených proměnných v Javě
- Paralelní a asynchronní programování
  - Vysvětlit výhody a nevýhody thread a task-based concurrency
    - Vysvětlit princip thread poolu
    - Ⓥ Vhodně využít typů Task, Task<T> a ThreadPool místo vláken v C#
    - Ⓥ Vhodně využít tasků pomocí std::async místo vláken v C++
    - Ⓥ Vhodně využít exekutory a fork/join pool místo vláken v Javě
  - Ⓥ Vysvětlit koncepty future a promise a jejich využití v asynchronním programování v C#, C++ nebo Javě
    - Ⓥ Umět implementovat a využívat asynchronní metody vracející svůj budoucí výsledek ve formě future v C#, C++ nebo Javě
    - Ⓥ Umět implementovat asynchronní kód, který si drží informaci o slíbeném výsledku ve formě promise, a umí tento promise naplnit v C#, C++ nebo Javě
  - Vysvětlit koncept coroutine a možnosti jeho využití

  - Vysvětlit v kontextu coroutine princip a možnosti využití kooperativního multitaskingu a vysvětlit jeho výhody a nevýhody oproti preemptivnímu multitaskingu
  - Ⓥ Umět implementovat coroutine a použít v zadaném příkladu připravenou coroutine v C++ nebo Javě
  - Ⓥ Využít iterátorových metod pro implementaci coroutines a využít takové coroutines na příkladu v C#

```
public IEnumerator<T> GetEnumerator() {
    for (int i = 0; i < Count; i++) {
        yield return this[i];
    }
}

IEnumerator IEnumerable.GetEnumerator() {
    return GetEnumerator();
}
```

  - Ⓥ Využít asynchronní metody s await pro implementaci coroutines a využít takové coroutines na příkladu v C#
- Synchronizační primitiva
  - Definovat sémantiku a vysvětlit princip použití podmínkové proměnné, resp. monitoru
  - Ⓥ Využít podmínkové proměnné, resp. monitory, v C#, C++ nebo Javě pro řešení základních synchro- nizačních problémů (např. producent-konzument)
    -
```
lock (_obj) {
  while (unsafe_to_do_crimes) {
    Monitor.Wait(_obj);
  }

  Crimes.Do();
}
```
    -
```
Monitor.Enter(_obj);
while (unsafe_to_do_crimes) {
  Monitor.Wait(_obj);
}

Crimes.Do();
Monitor.Exit();
```
  - Definovat sémantiku a vysvětlit princip použití semaforu
  - Ⓥ Využít semaforů v C#, C++ nebo Javě pro řešení základních synchronizačních problémů (např. producent-konzument)
- Významné prvky standardních knihoven a jejich aplikace
  - Základní síťování pomocí socketů
    - Ⓥ Umět implementovat klientskou aplikaci pomocí TCP socketů v C#, C++ nebo Javě
      -
```
int main () {
  int server_socket = socket (AF_INET, SOCK_STREAM, 0); // always AF_INET,
SOCK_DGRAM for UDP
  struct sockaddr_in server_address;
  server_address.sin_family = AF_INET;
  server_address.sin_addr.s_addr = htonl (INADDR_ANY); // 0.0.0.0
  server_address.sin_port = htons (HTTP_PORT);
  bind (server_socket, (struct sockaddr *) &server_address, sizeof
(server_address));
```

```
        listen (server_socket, 0);
        while (true) {
          struct sockaddr_in client_address;
          socklen_t client_address_size = sizeof (client_address);
          int client_socket = accept (server_socket, (struct sockaddr *)
    &client_address, &client_address_size)
          char *page = "<HTML><BODY>Hello␣!</BODY></HTML>";
          write (client_socket, page, strlen (page));
          shutdown (client_socket, SHUT_RDWR);
          close (client_socket);
        }
      }
```

```
s = socket();
bind(s, addr, sizeof(addr));
listen(s, 0);
while (true) {
  accept(s, &client_addr, &caddr_size);
  /* do stuff */
  /* shutdown(s, SHUT_RDWR); */
  close(s);
}
```

  - Ⓥ Umět implementovat serverovou aplikaci pomocí TCP socketů v C#, C++ nebo Javě
  - Ⓥ Využít na jednoduchém příkladu UDP sockety pro komunikaci mezi aplikacemi v C#, C++ nebo Javě
- Ⓥ Využít koncept asynchronního programování v tvorbě síťových aplikací v C#, C++ nebo Javě

6. Návrh a tvorba software

6.1. Doporučené postupy

- Principy objektového návrhu
  - Abstrakce, zapouzdření, SOLID principy a jejich aplikace, dekompozice
    - **S** – single responsibility
    - **O** – open-closed – dá se rozšiřovat bez modifikace (implement changes by adding code)
    - **L** – Liskov substitution – pokud umím pracovat se zvířátky, měl bych umět pracovat i s kočkou, i když nevím, co to je kočka
      - expect no more, provide no less
    - **I** – interface segregation – interfacy by měly být specifické
    - **D** – dependency inversion – závislosti by měly procházet přes abstrakce, high-level nezávisí na lowlevelu, lowlevel si definuje abstrakci, oba závisí na abstrakci (interface--implementace)
  - Předpokládá se schopnost identifikovat porušení principů v návrhu a navrhnout změny vedoucí k nápravě
- Návrh API, tříd a metod
  - Rozhraní aplikací (API) a poskytovatelů služeb (SPI), evoluce rozhraní
  - Návrh tříd, dědičnost a kompozice, immutability
  - Návrh metod, zohlednění účelu/zodpovědnosti, funkční dekompozice
  - Testovatelnost tříd a metod, návrh pro testovatelnost
- Návrhové vzory

- **Strategy**, **Observer**, **Decorator**, **Factory**, **Adapter**, **Facade**
- Identifikace principů objektového návrhu v návrhových vzorech
- Struktura, záměr (intent) a aplikace základních návrhových vzorů

6.2. Programování v paralelním prostředí

- Paralelní programování, paměťový model.
- Atomické operace a neblokující datové struktury.

6.3. Nástroje pro vývoj software

- Správa verzí
  - Popsat účel a použití systémů pro správu verzí při vývoji rozsáhlého software a práci v týmech (typicky nabízené funkce těchto systémů a jejich použití pro řešení běžných situací)
  - Popsat běžné způsoby integrace verzovacích systémů s nástroji pro správu projektů
  - Vysvětlit hlavní koncepty (lokální a vzdálené repozitáře, pracovní kopie (working copy), commit, větve)
  - Popsat a použít operace clone, pull, push, add, diff, commit, branch, merge, log, blame nástroje git
  - Popsat koncept feature branch v kontextu procesu vývoje a udržování software
  - Vysvětlit konflikty mezi verzemi, důvody vzniku a způsoby řešení
- Systémy pro sestavování software
  - Popsat hlavní účel a typicky nabízené funkce
  - Vysvětlit hlavní koncepty (cíl, závislosti, akce)
- Nástroje pro testování software
  - Popsat hlavní koncepty (test, pokrytí, jednotkové testování, integrační testování, systémové testování, black box, white box)
  - Popsat a na ukázkách kódu identifikovat typickou strukturu testu (set up, tear down, execute and validate)
  - Vysvětlit omezení běžných testovacích postupů a základní možnosti řešení
  - Popsat způsoby automatizace testování