Výroková logika   
1. úvod do logiky, výroky, logické spojky, kvantifikátory, pravdivostní hodnota výroku

* Logika je věda o správném usuzování, **jde v ní o formu a ne obsah**
* Klasická – dvě pravdivostní hodnoty, pravdivostní spojky / Neklasická
  + Modální (neklasické spojky „je množné že“)
  + Temporální (roli hraje čas)
  + Fuzzy
  + Epistemická (neklasické spojky „věří se, že“)
* **Výrok** je tvrzení, které muže být pravdivé nebo nepravdivé.
* Z jednodušších výroků se vytvářejí složitejší výroky pomocí tzv. logických spojek. Logické spojky jsou speciální jazykové výrazy jako např. A pak, když, nebo a...
* Že je výrok V pravdivý, resp. nepravdivý, zapisujeme následovne ||v|| = 1
* **Atomický výrok** = neobsahuje logické spojky, např „Prší“
* Jak zjistíme **pravdivostní hodnotu?** U atomických výrazů díky externímu ohodnocení e(Vi), jinak z významu logických spojek, které spojují atomické výrazy
* Výrazy obsahující promenné, které se po dosazení hodnot za promenné stanou výroky, se nazývají **výrokové formy.**
* Obecný (Pro každé x) a Existenční (existuje x tak, že platí)

2. jazyk, formule a pravdivostní hodnota formule výrokové logiky

* **Jazyk se skládá** z
  + Výrokových symbolů, p, q...
  + Výrokových pomocných symbolů, ()...
  + Logických spojek
* **Formule** = posloupnost výrokových symbolů
  + Každý výrokový symbol je formule
  + Pokud je ϕ a ψ, pak i ¬ϕ, (ϕ ⇒ ψ)... jsou formule
* **Pravdivostní ohodnocení** : libovolné zobrazení výrokových symbolů do intervalu {0, 1}
  + tj. ohodnocení každému symbolu přiřazuje buď 0 nebo 1
* **Pravdivostní hodnota formule**
  + Jestliže je formule ϕ symbol p, tak pravdivostní hodnota je e(p)
  + Je-li ϕ složená formule, tj. je jednoho z tvarů ¬ψ, ψ ∧θ, ψ ∨θ, ψ ⇒ θ, ψ ⇔ θ, pak

**||**¬ψ||e = ¬|| ψ ||e atd…

3. tabulková metoda, sémantické vyplývání ve výrokové logice

* Formule ϕ **sémanticky vyplývá** z množiny T formulí, jestliže ϕ je pravdivá při každém ohodnocení, při kterém jsou pravdivé všechny formule z T.
* **Tabulka** má 2n řádků, kde n je počet symbolů.
* Poznámka: Formule ψ1,...,ψn z předchozí definice někdy nazýváme predpoklady , formuli ϕ sémantický důsledek formulí ψ1,...,ψn.

4. booleovské funkce, úplná konjunktivní a disjunktivní normální forma

* **Booleovská funkce** s n argumenty (nekdy n-ární booleovská funkce) je libovolné zobrazení, které každé uspořádané n-tici hodnot 0 nebo 1 přiřadí hodnotu 0 nebo 1.
* Existuje 2(2 n ) booleovských funkcí s n argumenty.
* Funkce:
  + OR (nebo, logický součet), výrok je pravdivý, jeli aspoň jeden symbol pravdivý
  + AND (a, logický součin), výrok je pravdivý, jsouli oba symboly pravdivé
  + Implikace (a pak, výrok je nepravdivý, jeli první pravdivý a druhý nepravdivý, jinak pravdivý)
  + Ekvivalence (Právě když, výrok je pravdivý jsou li oba symboly ekvivalentní)
  + NOT, NAND (sheffer), NOR (pierce), NORekvivalence
* UKNF = konjunkce všech elementárních disjunkcí
* UDNF = diskunkce všech elementárních konjunkcí
* **literál** nad V je libovolný výrokový symbol z V nebo jeho negace
* **úplná elementární konjunkce** nad V je libovolná konjunkce literálů, ve které se každý výrokový symbol z V vyskytuje práve v jednom literálu

5. vyjadřování spojek jinými spojkami, úplné systémy spojek

* Úplné systémy:
  + {¬, ∧, ∨} ({¬,∧}, {¬,∨}, {¬,→}, {sh}, {pe})
  + Např. ¬p ⇒ q = disjunkce
* Sheffer a pierce

Množiny, zobrazení   
6. pojem množiny, zapisování množin, vztahy mezi množinami, operace s množinami

* **Množina** je matematický protějšek souboru (seskupení)
* Množina je objekt, který se skládá z jiných objektů (prvků množiny), je jednoznačně dána svými prvky a nezáleží na pořadí prvků
* Prázdná množina
* Zápis
  + Výčtem prvků A = {a1, … an}
  + Charakteristickou vlastností {x | x ∈ A a ϕ(x)}
* **Vztahy**:
  + Rovnost = A = B znamená, že pro každý x : x ∈ A, práve když x ∈ B
  + Inkluze = A ⊆ B znamená, že pro každý x : jestliže x ∈ A, pak x ∈ B
  + Všimneme si, že A = B platí, práve když je zároven A ⊆ B a B ⊆ A.
* Potenční množina = množina všech podmnožin dané množiny X,
  + Tedy 2X = {A|A ⊆ X}
  + B = {a,b,c} je |2B| = 23 = 8
* **Operace**:
  + Průnik A∩B = {x|x ∈ A a x ∈ B},
  + Sjednocení = A∪B = {x|x ∈ A nebo x ∈ B}
  + Rozdíl = A\B = {x|x ∈ A a x ∈/ B}.
  + Množiny A a B nazýváme (navzájem**) disjunktní**, práve když A∩B = 0.
* Univerzum
* Doplněk, vennovy diagramy

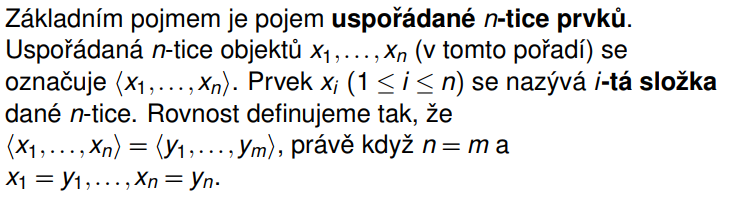
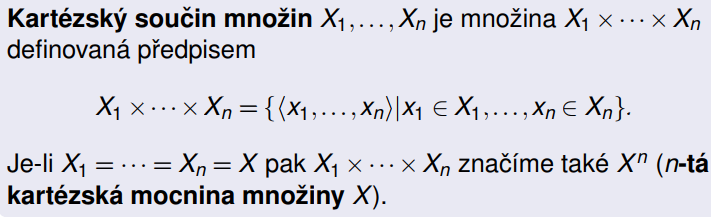
7. konečné, spočetné a nespočetné množiny

* Množina A se nazývá konečná , práve když existuje přirozené číslo n tak, že prvky této množiny lze jednoznacně očíslovat čísly 1 ,2,...,n. nebo když je prázdná
* {1, 2, 3, 4} |A|= 4
* |A| = n. n = počet prvků množiny
* Množina A se nazývá nekonečná, není li konečná např. Množina všech přirozených čísel
* Množina A se nazývá spočetná , práve když existuje bijekce f : A → N. Množina A se nazývá nespocetná , práve když je nekonecná a není spocetná.

8. pojem zobrazení, typy zobrazení (surjektivní, injektivní, bijektivní)

* Zobrazení je matematický protějšek přiřazení
* Relace R mezi X a Y se nazývá zobrazení (funkce) množiny X do množiny Y, právě když pro každé x ∈ X existuje právě jedno y ∈ Y tak, že <x,y> ∈ R.
* Typy:
  + prosté (**injektivní**), právě když pro každé x1,x2 ∈ X, pro x1 /= x2 plyne f(x1) /= f(x2),
  + zobrazení množiny X na množinu Y (**surjektivní**), právě když pro každé y ∈ Y existuje x ∈ X tak, že f(x) = y,
  + Bijektivní, když je injektivní a surjektivní zároveň

Relace   
9. pojem relace, arita relace

* Relace je matematický protějšek vztahu (Dva členové rodiny mohou být ve vztahu "být sourozencem".)
* Relace je dána **aritou** (číslo udávající počet objektů, které do relace vstupují) a množinami, jejichž prvky vstupují do relace
* Kartézský součin množin je množina všech uspořádáných n-tich prvků z těchto množin.
*  
* Necht’ X1,...,Xn jsou množiny. Relace R mezi X1,...,Xn je libovolná podmnožina kartézského soucinu X1 ×··· ×Xn.
* Má ústřední roli v relačním databázovém systému

10. operace s binárními relacemi, binární relace a jejich reprezentace

* Relace jsou množiny. Proto s nimi lze aplikovat stejné operace a vztahy jako u množin
  + Průnik, sjednocení, rozdíl, inkluze, rovnost
  + Navíc je **inverze** R −1 = {<y,x>| <x,y> ∈ R}.
  + Skládání relací:
  + Další operací je tzv. skládání. Je-li R relací mezi množinami X a Y a S relací mezi množinami Y a Z, pak složením relací R a S je relace R ◦S mezi X a Z definovaná předpisem **R ◦S = {<x,z>|∃y ∈ Y : <x,y> ∈ R a <y,z> ∈ S}.**
  + R S = {<x, z> | pro každé y ∈ Y : pokud <x, y> ∈ R, pak <y, z> ∈ S},
* **Reprezentace:** 
  + Tabulkou
  + Matice mij = 1 je-li <xi ,yj> ∈ R, mij = 0 je-li <xi ,yj> ∈/ R.
  + Pro binární matice můžeme zavést operace, které odpovídají operacím s relacemi.
    - M ∨ N = P, pij = max{mij,nij};
    - M ∧ N = P, pij = min{mij,nij};
    - M −N = P, pij = max{0,mij −nij};
  + Matice má ale nevýhodu: Kdybychom měli matici, kde je 1000x1000 políček a pouze 3000 by byly v relaci, tak se zbytečně uchovává 997 000 nul
  + Grafem
  + seznam seznamů

11. binární relace na množině (vlastnosti, příklady)

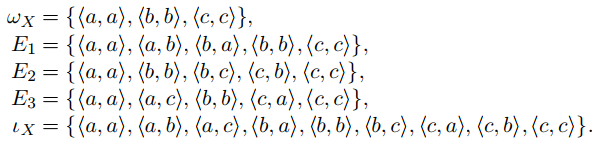
* speciální relace jsou prázdná relace, relace identity ωX = {<x,x>|x ∈ X} a kartézský čtverec  
   ιX = X ×X.
* reflexivní, pokud pro každé x ∈ X platí <x,x> ∈ R
  + každý prvek je v relaci „sám se sebou“
  + v grafu je u každého vrcholu smyčka
  + v binární matici jsou na diagonále samé jedničky
* symetrická, pokud pro každé x,y ∈ X platí <x,y> ∈ R ⇒ <y,x> ∈ R
  + matice je symetrická dle hlavní diagonály
  + v grafu musí vést hrany oběma směry, nebo nevedou vůbec
* antisymetrická, pokud pro každé x,y ∈ X platí (<x,y> ∈ R ∧ <y,x> ∈ R) ⇒ x = y
  + v grafu je buď jedna hrana nebo žádná
  + Když dvě různá pole matice souměrná podle diagonály neobsahují dvě jedničky
* tranzitivní, pokud pro každé x,y,z ∈ X platí (<x,y> ∈ R ∧ <y,z> ∈ R) ⇒ <x,z> ∈ R
* Relace, která je reflexivní, symetrická a tranzitivní se nazývá ekvivalence
* Relace, která je reflexivní, antisymetrická a tranzitivní se nazývá uspořádání
* Další vlastnosti:
  + Irreflexivní – opak reflexivity
  + Asymetrická – opak symetrie
  + Uplná <x,y> ∈ R ∨ <y,x> ∈ R.

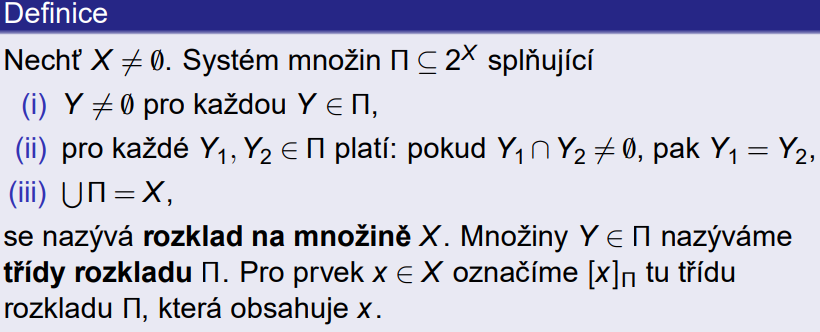
12. uzávěry relací

* Reflexivní, symetrický a tranzitivní uzávěr = nejmenší R nebo S nebo T relace na dané množině, která obsahuje danou relaci
* Tra(R), Sym(R), Ref(R)
* Ref(R) = R ∪ ωX
* Sym(R) = R ∪ R −1
* Tra(R) = U∞ n=1 R n , kde R 1 = R a R n = R ◦R n−1
* Tranzitivní uzávěry se používají v informatice např. v teorii automatů

13. ekvivalence a rozklady na množině, ekvivalence a surjektivní zobrazení

* Ekvivalence je binární relace, která je R S T, matematický protějšek nerozlišitelnosti
* Pro ekvivalenci E na množine X definujeme pro každý x ∈ X množinu [x]E = {y ∈ X | <x,y> ∈ E}, kterou nazýváme třída ekvivalence prvku x.
* Třída ekvivalence [x]E je dle definice množina těch prvků y ∈ X, ktere´ jsou E-ekvivalentnı´ x. Jiny´mi slovy, [x]E obsahuje právě ty prvky z X, které nelze od x rozlišit ekvivalencı´ E.
* Příklad: pro {a,b c} existuje právě 5 ekvivalencí

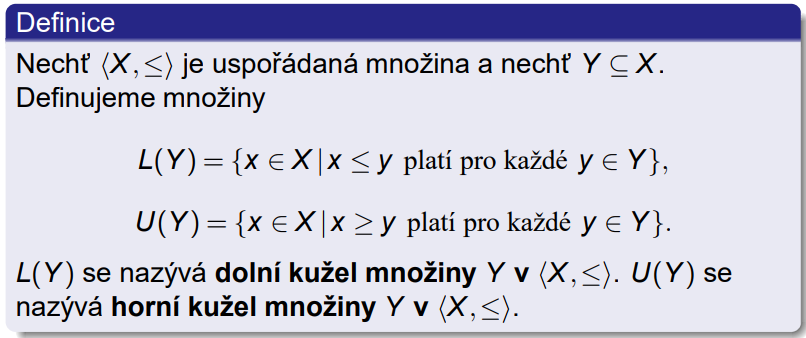


* Rozklad :
* 
* Na X = {a,b,c} existuje pět vzájemně různých rozkladů
  + {{a},{b},{c}}, {{a,b},{c}}, {{a,c},{b}}, {{a},{b,c}}, {{a,b,c}}.
  + {{a},{b},{c}} – mezní případ jednoprvkové rozklady
  + {{a,b,c}} – mezní případ jediná třída rozkladu, která obsahuje všechny prvky
* Necht’Π je rozklad na X. Pak binární relace EΠ na X definovana´ <x, y> ∈ EΠ, právě když [x]Π = [y]Π je ekvivalence.
  + X je ekvivalentní Y právě když se rovnají jejich třídy rozkladu
* Ekvivalence EΠ se nazývá ekvivalence příslušná rozkladu Π.
* Necht’ E je ekvivalence na X. Pak systém množin ΠE ⊆ 2X definovaný ΠE = {[x]E | x ∈ X} je rozklad na množině X.
* Rozklad ΠE se nazývá rozklad príslušný ekvivalenci E.
* Poznámka: Rozklad na množině X příslušný ekvivalenci E oznacujeme běžně X/E místo ΠE a nazýváme jej faktorová množina X podle E.
* Pokud je zobrazení f : X → Y surjektivní, pak lze obraz f(x) prvku x chápat jako vyjádření: "prvek x nahradíme (zjednodušíme) prvkem f(x)".
* Pro zobrazení f : X → Y definujme binární relaci Ef na X předpisem <x,y> ∈ Ef práve když f(x) = f(y). Zřejmě Ef je ekvivalence, tzv. ekvivalence indukovaná zobrazením f .

14. uspořádání, Hasseovy diagramy

* R A T binární relace R na X se nazývá uspořádání
* Úplné uspořádání se nazývá lineární uspořádání = řetězec
* Pokud je R uspořádání na X, pak se <X, R> nazývá uspořádaná množina
* Značíme x≤y
* Jeli <X, ≤ >, pak můžou existovat x, y ∈ X, kde neplatí ani x≤y ani y ≤ x, říkáme, že dva prvky jsou **NESROVNATELNÉ**, značí se i x || y, jinak jsou **SROVNATELNÉ**
* Jeli ≤ lineární uspořádání na X, pak je to úplná relace, což znamená, že každé dva prvky jsou srovnatelné.
* Každá relace identity je **ANTIŘETĚZEC**
* Jeli ≤ na X antiřetězec, pak každé dva prvky jsou nesrovnatelné
* Množinová inkluze je také uspořádání
* Princip duality = Nechť ≤ je uspořádání na X, pak ≤-1  je uspořádání na X, které označujeme ≥.
* Ke každému uspořádání ≤ na X můžeme uvažovat odvozenou RELACi **pokrytí** =
  + x ≺ y právě když x < y a pro ∀z ∈ X: pokud x ≤ z ≤ y, pak z ∈ {x, y};
  + je irreflexivní, asymetrická a není tranzitivní
* HASSEOVY DIAGRAMY

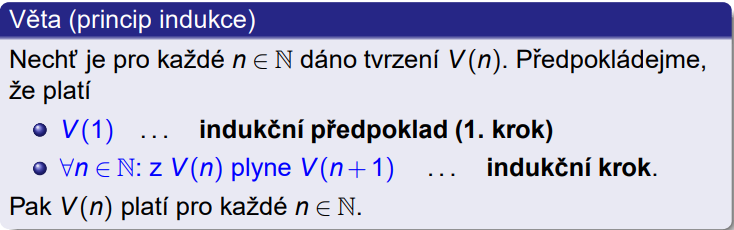
15. speciální prvky uspořádaných množin, polosvazy, svazy

* Prvek je
  + Minimální, jestliže ∀y ∈ X : y ≤ x, pak y = x
  + Nejmenší ∀y ∈ X: x ≤ y
  + Maximální jestliže ∀y ∈ X : x ≤ y, pak y = x
  + Největší ∀y ∈ X: y ≤ x
* Věty:
  + V relaci uspořádání existuje jen jeden největší a nejmenší prvek
  + Jeli prvek nejmenší (největší) pak je také minimální (maximální)
  + pokud je ≤ lineární uspořádání, pak je x ∈ X největší (nejmenší) prvek, právě když je maximální (minimální).
  + Příklad: Potencní množina 2U uspořádaná množinovou inkluzí ⊆ má nejmenší a zároveň minimální prvek, kterým je „prázdná množina“ a dále má i nejvetší a zároveň maximální prvek, kterým je množina U.
* 
* INFIMUM = největší prvek dolního kužele
* SUPREMUM = nejmenjší prvek horního kužele
* Pokud pro každé x, y ∈ X existuje infimum, pak množinu nazveme průsekový polosvaz
* Pokud pro každé x, y ∈ X existuje supremum, pak množinu nazveme spojový polosvaz
* Pokud existuje infimum i supremum, pak množinu nazveme svaz
* Svaz je tedy uspořádaná množina, kde ke každým dvěma prvkům existuje jejich infimum i supremum.

Čísla a číselné obory   
16. přirozená, celá, racionální, iracionální, reálná a komplexní čísla

* **Přirozená**: Jsou to čísla 1,2,3,4,5,6,.... Množinu všech přirozených čísel označujeme N
* **Celá**: Jsou to čísla 0, 1 ,- 1, 2, -2...., Množinu označujeme Z
* **Racionální**: Jsou to čísla, která lze vyjádřit zlomkem , kde m je celé číslo a n je přirozené číslo, množinu označujeme Q,
* **Reálná:** všechna čísla na číselné ose
* **Iracionální:** čísla, která nelze vyjádřit ve tvaru zlomku, např √ 2, √ 3, π, e.
* **Komplexní:** množina dvojic reálných čísel ve tvaru a + bi, kde i je imaginární jednotka, pro kterou platí, že i2= -1, značíme C, znázorňují se v rovině
  + Každé komplexní číslo můžeme znázornit i v goniometrickém tvaru (Moiverova věta)
    - n(cos x +i sin x)

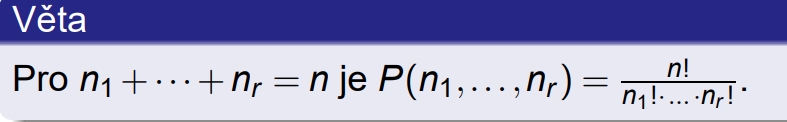
17. princip indukce

* 
* Princip dobrého uspořádání: předpokládáme, že každá neprázdná podmnožina přirozených čísel má nejmenší prvek

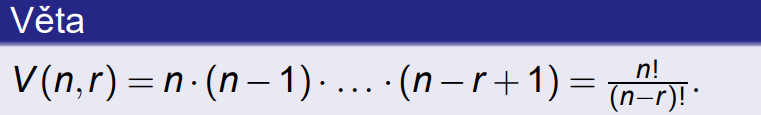
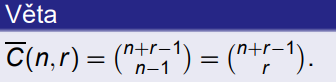
18. dělitelnost, prvočísla, číselné soustavy

* Pro m,n ∈ Z říkáme, že m dělí n, píšeme m | n, právě když ∃k ∈ Z tak, že m · k = n.
* Jestliže a|b a b|c tak a|c
* Pro a,b ∈ Z existují jednoznačně určená q,r ∈ Z tak, že a = bq +r a 0 ≤ r < b.
  + Císlo r se nazývá zbytek po celocíselném dělení císla a císlem b. Píšeme také (a mod b) = r.
* Přirozené číslo n se nazývá prvocíslo , jestliže n /= 1 a jestliže n je delitelné jen čísly 1 a n
* Existuje nekonečně mnoho prvočísel
* Každé přirozené číslo větší než jedna se dá rozepsat jako součin prvočísel

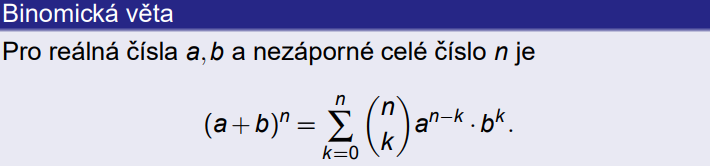
Kombinatorika, pravděpodobnost   
19. pravidla součtu a součinu, permutace a permutace s opakováním

* Lze-li úkol A provést m způsoby a lze-li úkol B provést n způsoby, přičemž žádný z m způsobů provedení úkolu A není totožný s žádným z n způsobů provedení úkolu B, pak provést úkol A nebo úkol B lze m +n způsoby.
* Lze-li úkol C rozložit na po sobe následující úkoly A a B (tj. provést C znamená provést nejdříve A a potom B) a lze-li úkol A provést m způsoby a úkol B lze provést n způsoby, pak lze úkol C provést m · n způsoby.
* Permutace nějakých prvků je jejich seřazení
* Permutace navzájem různých prvků je jejich libovolné seřazení od prvního k n-tému členu. Značíme P(n)
* Nemůžou se opakovat
* Záleží na pořadí
* Seřazujeme-li objekty z nichž některé jsou stejné = permutace s opakováním
* Je dáno n objektů rozdelených do r skupin, které mají po řadě n1,...,nr objektů, tj. n1 +···+nr = n. Objekty v každé ze skupin jsou navzájem nerozlišitelné. Každé seˇrazení techto ˇ n objekt˚u se nazývá permutace s opakováním. Pocet takových ˇ permutací znacíme ˇ P(n1,...,nr).
* 

20. variace a variace s opakováním, kombinace a kombinace s opakováním

* U variací záleží na pořadí a nesmí se opakovat
* Vybírám kátice z n prvků, kde k <= n, V(k, n)
* 
* Variace s opakováním: V(n,r) = n r .
* Kombinace = nezáleží na pořadí C(n, r)
* Kombinace s opakováním = nezáleží na pořadí a mohou se prvky opakovat
* 
* Dirichletův princip =
  + Je-li aspoň r + 1 objektů rozděleno do r šuplíků, pak musí existovat šuplík s nejméně dvěma objekty

21. binomická věta, princip inkluze a exkluze



* Pascalův trojúhelník
* udává počet prvků sjednocení několika množin pomocí počtu prvků průniku jednotlivých množin.

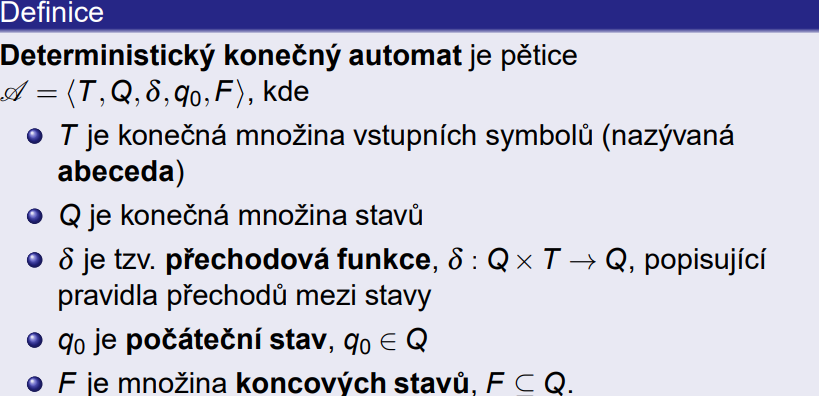
22. počítání pravděpodobnosti

* představme si pokus, který končí jedním z výsledků e1 ... en – těmto výsledkům, se říká elementární jevy (např. když házíme kostkou – 1 2 3 4 5 6)
* JEV je každá podmnožina A ⊆ E
* P(A) = |A| / |E|
* Pravděpodobnost nabývá hodnot 0 až 1, kde
  + 0 = jev je nemožný
  + 1 = jev je jistý

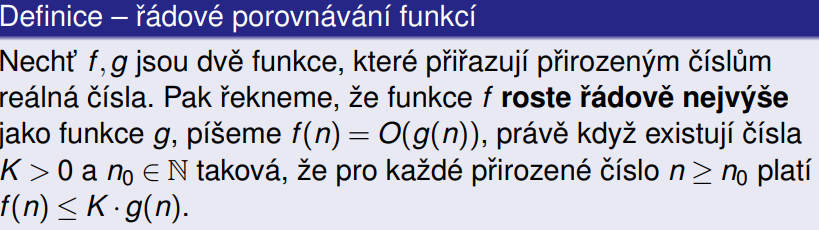
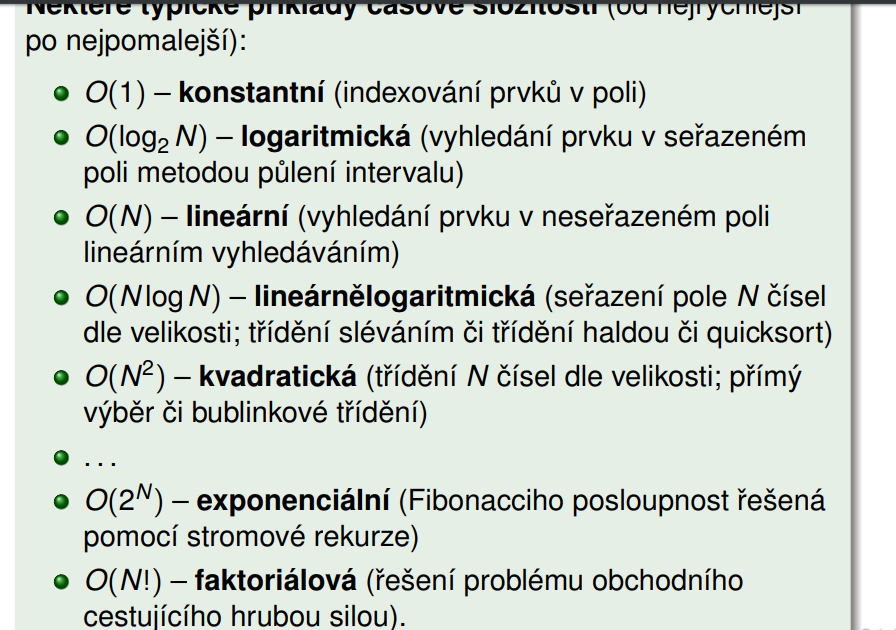
Algoritmy, složitost algoritmů   
23. pojem algoritmu (vlastnosti, druhy), rekurze

* Algoritmus = předpis pro řešení nějakého problému
* Obsahuje: vstupní data, předpis pro řešení, výstupní data
* Vlastnosti:
  + Konečnost (finitnost) = každý algoritmus musí skončit po konečném počtu kroků (nemůže být loop, který poběží milion let)
  + Jednoznačnost (determinovanost) = každý krok v algoritmu musí být předem definován, musí být jasné, jaký krok se provede a jak bude algoritmus pokračovat
  + Obecnost (hromadnost) = neřeší jen jeden problém, ale dá se aplikovat na obecně každý problém stejného typu
  + Rezultativnost = algoritmus při zadání vstupních dat vždy vrátí nějaký výsledek
  + Korektnost = výsledek algoritmu musí být správný
  + Opakovatelnost = při zadání vstupních dat musí algoritmus vždy vracet stejný výsledek
* Příklady algoritmů: třídící algoritmy, euklidův, dijkstrův
* Druhy:
  + Rekurzivní (volají semi sebe)
  + Hladový = k řešením se propracovávájí po jednotlivých rozhodnutích, která jsou nevratná – Kruskalův algoritmus
  + Rozděl a panuj = dělí problém na menší podproblémy až po triviální podproblémy, které lze vyřešit přímo = Quicksort
  + Pravděpodobnostní = provádí některá rozhodnutí náhodně
  + Paralelní = rozdělení úlohy mezi více počítačů
  + Genetické = napodobují evoluční procesy postupným pěstováním řešení pomocí mutací a křížení
  + Dynamické programování = řeší problémy od nejjednoduších k nejsložitějším tím, že pak výsledky z jednoduchých používají k řešení těch složitějších
  + Heuristické – nehledají přesné řešení, ale jen přiblížení, používají se v situacích, kdy dostupné zdroje nejsou dostačující k využití exaktních algoritmů
* REKURZE =
  + Musí obsahovat koncovou podmínku, která rekurzi ukončí
  + Musí se redukovat problém na menší problém, tak dlouho, dokud nenastane koncová podmínka
  + Když nenastane koncová podmínka, vykoná se rekurzivní krok
  + Každý algoritmus rekurzivního tvaru lze přepsat do nerekurzivního tvaru pomocí zásobníku
* TYPY REKURZE:
  + Přímá = pokud podprogram volá sám sebe
  + Nepřímá = situace, kdy vzájemné volání podprogramů vytváří kruh: Ve funkci A voláme funkci B, ve funkci b voláme funkci C a ve funkci C voláme funkci A
  + Lineární = podprogram volá sám sebe pouze jednou – vytváří se lineární struktura postupně volaných podprogramů
  + Stromově = podprogram se volá vícekrát – znázornění stromu
* Nerekurzivní řešení bývá efektivnější, ale méně přehledné
* Mechanické použití rekurze často vede k exponenciální složitosti algoritmu = pomalé

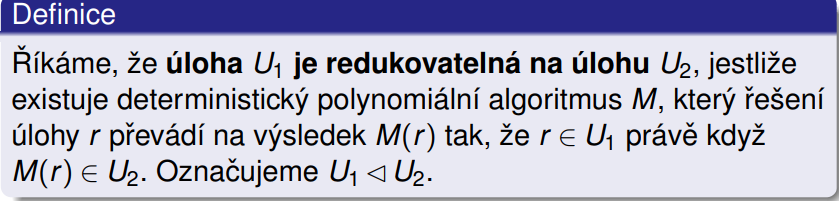
24. konečné automaty

* Teoretický výpočetní proces používaný v informatice pro studium vyčíslitelnosti a obecně formálních jazyků
* Popisuje velice jednoduchý počítač, který může být v jednom z několika stavů, mezi kterýma přechází na základě symbolů, které čte ze vstupů
* Množina stavů je konečná
* Nemá žádnou další pamět kromě informace o aktuálním stavu
* Používají se pro zpracování regulárních výrazů např. v překladačích
* 
* Existují i nedeterministické automaty, které mají stejnou výpočetní sílu
* Konečné automaty lze reprezentovat
  + Přechodovou funkcí
  + Tabulkou
  + Upraveným orientovaným grafem, kde stavy jsou vrcholy, přechody jsou hrany, které jsou ohodnocené vstupními symboly, koncové stavy jsou vyznačené dvojitým kroužkem
* ŘETĚZEC alfa prvků množiny V je libovolná konečná posloupnost prvků této množiny, je dán svojí délkou |alfa|, speciální případ je prázdný řetězec
* ABECEDA V je konečná množina symbolů
* Pozitivní uzávěr V+ konečné množiny V je množina všech neprázdných řetězců prvků množiny V.
* Uzávěr = V\* je pak V+ sjednoceno se všemi prázdnými řetězci
* Necht’ je dána abeceda V. Pak libovolná podmnožina uzávěru V ∗ je formální jazyk nad V.
* Množina všech slov příjmaných konečným automatem se nazývá jazyk rozpoznávaný tímto automatem

25. pojem složitosti, řádové porovnávání funkcí, příklady časové složitosti

* Složitost může být studována buď z hlediska pamětové složitosti (jak je třeba velká pamět k provedení výpočtu) nebo časové složitosti (kolik času je potřeba k provedení výpočtu)
* Čas se neměří v časových jednotkách, ale v počtu elementárních kroků algoritmu.
* Složitost je funkce závislá na velikosti vstupních dat algoritmu
* Uvažujeme pouze řádovou složitost = funkce lišící se v konstantě bereme za stejné
* 
* Rekneme, že algoritmus má **polynomickou časovou složitost**, práve když existuje polynom p takový, že f(n) ≤ p(n) pro všechna n ∈ N.
* Existují algoritmicky neřešitelné problémy, pro kreté nemá smysl algoritmy konstruovat
  + Např. algoritmus, který by dokázal posoudit, zdali daný algoritmus skončí po konečném počtu kroků či nikoli
* Algoritmům se složitostí O(N) říkáme lineární, se složitostí O(N2 ) kvadratické, se složitostí O(N3 ) kubické.
* f(n) = O(g(n)) = „funkce f roste řádově nejvýše jako funkce g“
* f(n) = Ω(g(n)) = „funkce f roste řádově aspoň jako funkce g“
* f(n) = Θ(g(n)) = „funcke f roste řádově stejně jako funkce g“
* 

26. třídy složitostí P a NP, NP-úplné úlohy (příklady)

* Algortmus nazveme řešitelný v polynomiálním čase, jestliže jeho časovou složitost můžeme shora ohraničit polynomem.
* Třídu s polynomiální složitostí označujeme P.
* Úlohy třídy P označujeme jako řešitelné (v přiměřeném čase)
* Například heap-sort je úloha třídy P neboť má složitost nlogn ≤ n2
* NP = úlohu nazveme nedeterministicky polynomiální, jestliže existuje nedeterministický algoritmus, který ji řeší v polynomiálním čase
* Vzhledem k casovým nárokům považujeme NP-úplné problémy za neřešitelné pro n ≥ 70.
* NP – úloha : ÚLOHA BATOHU
  + Máme batoh, ve kterém jsou všechna přirozená čísla
  + Z batohu máme vybrat čísla tak, aby jejich součet dal dohromady číslo A, které si zvolíme
  + Nejprve si nastavíme X na 0
  + Pak procházíme jednotlivá čísla a buď je přičteme k X nebo nepřičteme
  + Pokud se X rovná A, úloha je splněna
  + Velmi časově náročná, protože složitost je 2n
* Při řešení NP úloh, je jediný známý deterministický způsob projít všechny možnosti
* Skutečnost, že r je řešením úlohy U označíme r ∈ U
* 
* Úlohu U nazveme NP- úplnou, jestliže na ni lze redukovat libovolnou úlohu z třídy NP.
* NP-úplné problémy patří mezi ty nejsložitější z třídy NP
* PŘÍKLADY:
  + Úloha batohu
  + Chromatické číslo grafu = určení nejmenšího počtu barev, kterými můžeme obarvit vrcholy grafu tak, že žádné dva vedle sebe nemají stejnou barvu
  + Problém splnitelnosti booleovských formulí (cookova věta)
  + Nalezení hamiltonovy kružnice = kružnice, která obsahuje všechny vrcholy grafu
  + Problém kliky = stanovení, zda má daný neorientovaný graf kliku o K vrcholech
  + Problém obchodního cestujícího = Může obchodní cestující projet všechna města tak, aby každé navšítil právě jednou, na závěr se vrátil do výchozího města a přitom urazil vzdálenost menší než K?
* V praxi se NP-úplné úlohy řeší jen přibližně
* Využití v šifrování:
  + Metoda RSA = šifrování s veřejným klíčem
  + Je založena na využití součinu velkých prvočísel

Teorie grafů   
27. neorientované a orientované grafy, základní pojmy teorie grafů

* Grafické vyjádření vztahu mezi objekty
* Objekty jsou reprezentovány kroužky = VRCHOLY (UZLY)
* Vztahy mezi nimi pak znázorňují hrany, každý z obou konců hrany musí vycházet z vrcholu
* Pokud oba konce hrany vychází se stejného vrcholu = SMYČKA
* Neorientovaný graf je dvojice <V, E>, kde V je neprázdná množina všech vrcholů a E je množina neorientovaných hran (množina dvouprvkových množin vrcholů)
* orientovaný graf je dvojice <V, E>, kde V je neprázdná množina všech vrcholů a E je množina orientovaných hran (množina uspořádaných dvojíc vrcholů)
* SLED = posloupnost vrcholů a hran, n je délka sledu
  + Uzavřený = jeli v0 = vn
  + CESTA = když se neopakuje žádný vrchol
  + TAH = když se neopakuje žádná hrana
  + KRUŽNICE = jeli v0 = vn a s výjimkou v0 a vn jsou každé dva vrcholy různé
  + VZDÁLENOST z vrcholu u do v je délka cesty z u do v, která je nejmenší ze všech cest z u do v
* Hranové ohodnocení grafu G =<V,E> s množinou hodnot D je funkce w : E → D.

28. hledání cest (Dijkstrův algoritmus)

* K nalezení nejkratší cesty grafu
* Vstup:
  + Neorientovaný graf G {V, E}
  + Hranové ohodnocení E - > R+
  + s ∈ V = počáteční vrchol, ze kterého budeme vycházet
* Výstup:
  + Hodnota d(v) pro každý vrchol, hodnota znázorňuje nejkratší cestu z vrcholu s do daného vrcholu
* Algoritmus si pro každý vrchol pamatuje nejkratší cesty, kterými se k němu dá dostat
* Množiny Z a N:
  + N = dosud nenavštívené vrcholy
  + Z = navštívené vrcholy
  + Algoritmus pracuje tak dlouho, dokud není N prázdná
  + V každém průchodu cyklu se přidá jeden vrchol vmin z N do Z, a to takový, který má nejmenší hodnotu d[v] ze všech vrcholů v z N.
* POSTUP:
  + 1) nastav vrchol s = 0; pro každý vrchol kromě vrcholu s nastav hodnotu nekonečno
  + 2) pokud neexistuje vrchol, který by měl hodnotu nekonečna, algoritmus skončí
  + 3) nastaví se vmin na momentální nejmenší d(v)
  + 4) Pokud je momentální nejmenší d(v) + délka cesty do daného vrcholu < d(daný vrchol), pak se do daného vrcholu zapíše hodnota, pokračuj bodem 2.

29. hledání minimální kostry grafu (Kruskalův algoritmus)

* Neorientovaný graf se nazývá souvislý, právě když pro každé u, v existuje sled z u do v
* Strom je neorientovaný graf bez kružnic
* Neorientovaný graf <V1,E1> je podgrafem grafu <V2,E2>, právě když V1 ⊆ V2 a E1 ⊆ E2.
* Kostra neorientovaného grafu G je jeho podgraf, který je stromem a obsahuje všechny vrcholy grafu G
* Graf má kostru právě, když je souvislý
* VSTUP:
  + Neorientovaný graf G {V, E}
  + Hranové ohodnocení E - > R+
  + |V| = n, |E| = m,
* VÝSTUP = minimální kostra grafu = kostra grafu s minimálním součtem ohodnocení jejích hran
* Algoritmus postupně prochází hrany a vytváří z nich množiny E0 .... E2.. z nichž poslední je minimální kostra grafu
* Všechny hrany si seřadíme podle velikosti (vzestupně - od hrany s nejmenší váhou).
* Hranu s nejmenší váhou použijeme jako první hranu kostry.
* Pokud jsme tím už vytvořili kostru (graf měl jen dva vrcholy), končíme. V opačném případě vezmeme hranu s druhou nejmenší váhou.  
  POZOR! Pokud by nám v grafu vznikla [kružnice](http://teorie-grafu.cz/zakladni-pojmy/kruznice-cyklus.php), hranu nepoužijeme.
* Opakujeme minulý krok, dokud vznikající kostra nespojí všechny vrcholy grafu.
* PODLE SLIDŮ:
  + Uspořádej hrany z E dle váhy od nejmenší po největší
  + Vytvoř prázdnou množinu hran E0
  + Pokud přidáním ei vznikne kružnice, ponechej Ei = E i-1, jinak Ei := Ei−1∪ {ei}
  + Opakujeme dokud i < m a dokud E ještě neobsahuje n – 1 hran