**1**

**Logika** - veda o spravnom usudzovani; symbolicky charakter; ide o formu usudzovania nie o obsah

- znalost zakladou logiky nam umoznuju jednoznacne a zrozumitelne sa vyjadrovat a argumentovat

- **druhy**: a) ***klasicka*** - 2 pravd. Hodnoty (0 a 1) a kl. spojky

b) ***neklasicka*** - pouziva aj ine ako kl. spojky a zaobera sa inymi aspektami

a) ***modalna*** - „je mozne, ze“

b) ***epistemicka*** - „vie sa, ze“

c) ***fuzzy*** - ma ine prav. hodnoy ako 0 a 1 (medzi 0 a 1)

d) ***logika casu*** - tvrdenia, v kt. hraje rolu cas

- **informatika** - podmienky (if... then... else...)

- log. programovanie, analyza dat, AI, etc.

**Vyroky** - tvrdenie, ktore je 0 alebo 1; z jednoduchych sa vytvaraju zlozite pouzitim log. spojek

1. ***atomicky vyrok*** - bez prav. spojek; prav. hodnota zavisi od ex. zdroja
2. ***zlozeny vyrok*** - prav. spojky; prav. hodnota zavisi od prav. spojek

**Log. spojky** - specialne jazyk. vyrazy – „a“ ; „alebo“ ; „ak, potom“ ; „prave ked“ ; „nie je pravda, ze“

- konjukcia, disjunkcia, implikacia, ekvivalencia, negacia

**Kvantifikatory** – operator mat. logiky urcujuci poctu predmetov, kt. pripisujeme nejaku vlastnost alebo vztah

1. ***obecne*** - „pre kazde x plati, ze“
2. ***existencne*** – „existuje x, pre kt. plati, ze“

**Pravdivostna hodnota vyroku** – pravda 1, nepravda 0;

**2**

**Jazky vyr. logiky** - sklada sa z vyrok. symbolov (p,q,r,...), symbolov log. spojek a pomoc. symbolov (),[],...

- zo symbolov jazyka sa vytvaraju **formule VL** - konecna postupnost symbolov VL

**Pravd. ohodnotenie** – zobrazenie vyrokovych symbolov daneho jazyka VL do mnoziny {0, 1}, t.j kazdemu vyrok. symbolu sa priradzuje 0 alebo 1

**Pravd. ohodnotenie formule** - ak je vyrok symbol, tak ||p||*e*=e(p)

- ak je to zlozeny vyrok, tak

||￢||*e*=⇁||||*e*, ||⇔||*e*=||||*e* ⬄||||*e*,

||∧||*e*=||||*e* ∧||||*e*, ||⇒||*e*=||||*e* →||||*e*,

||∨||*e*=||||*e*∨||||*e*,

a) ***tautologia*** - pri kazdom ohodnoteni je 1

b) ***kontradikcia*** - pri kazdom ohodnoteni je 0

c) ***splnitelna*** - 1 aspon pri jednom ohodnoteni

**3**

**Tabulk. metoda** - jednoduchy sposob, ako prehladne zapisat pravd. hodn. danej formule pri kazdom ohodnoteni

- sluzi k vypisaniu (tabelacii) hodnot zadanych formulou

- tabulka ma riadkov a n+m stlpcov (n=pocet vyr. symbolov, m=pocet log. spojok)

- ***sem. ekvivalencia*** -  su sem. ekv., ake=e pre kazde ohodnotenie e

**semanticke vyplyvanie** - formula sem. plynie z mnoziny *T* formuly (*T* |=akje  pravdive pri kazdom ohodnoteni, pri kt. su vsetky formule z T pravdive

Majme formule 1; ...;n (n>=0). Formula  **semanticky plynie** z formuli 1; ...;n

(1; ...;n |= ), ak ||e=1 pre kazde ohodnotenie e take, ze ||1e=1; ...; ||ne=1.

**Pr.:** Zistite, ci z mn. *T* = {*p*⇒￢*q*,*q*,￢(((*p* ⇒￢*q*)∨*r* )⇔(*r* ∧￢*q*))} sem. plynu formule: *r* ∧￢*q*; *r* ; (*p* ⇒￢*q*)∨*r*.

**4**

**Booleovske funkcie s n-argumantami** - lubovolne zobrazenie, ktore kazdej n-tici hodnot 0 alebo 1 priradi 0

**(n-arne booleovske funkcie)** alebo 1 f : {0, 1}n → {0, 1}

- existuje n-arnych bool. f-cii

- kazdu f-ciu f(x1, ..., xn) mozeme zapisat pomocou tabulky

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vs. bool. f-cie 1 premennej | | | | |
| x1 | f1 | f2 | f3 | f4 |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |

(V=mn. vyr. symbolov)

**Literal** nad V je lubovolny vyr. symbol z V alebo jeho negacia.

**UEK** nad V je lubovolna konjukcia literalov, v kt. sa kazdy vyr. symbol z V vyskytuje prave v jednom literale.

**UED** nad V je lubovolna disjunkcia literalov, v kt. sa kazdy vyr. symbol z V vyskytuje prave v jednom literale.

**UKNF** nad V je konjukcia UED nad V.

**UDNF** nad V je disjunkcia UEK nad V.

**Pr.:** (p ⇒ q)∧(p ⇒ r)

**5**

**Uplny system spojek** - mn. bool. f-cii {f1, ..., fk} je **funkcne uplna**, ak kazdu bool. f-ciu f : {0, 1}n → {0, 1} mozeme vyjadrit ako zlozenie niektorych f-cii z {f1, ..., fk}

- mn. vyr. spojek je **uplna** (**tvori uplny sys. spojek**) ak je mnozina im odpovedajucim bool. f-cii funkcne uplna

- **baza** - kazdy uplny min. system spojek VL

- specialny vyznam maju Sheffer a Pierce, ktore samy o sebe tvoria uplny sys. spojek

a) **Sheffer** ↑ - „ak, potom neplati“ – negacia konjukcie (NAND) a↑b = ¬(a∧b)

b) **Pierce (Nicod)** ↓ - „ani, ani“ – negacia disjunkcie (NOR) a↓b = ¬(a∨b)

a∨b = ¬(¬a∧¬b) ¬a = ¬ (a∧a) = (a↑a)

a∧b = ¬(¬a∨¬b) (a∧b) = ¬¬(a∧b) = ¬(a↑b) = ((a↑b)↑(a↑b)) - podobne pre ↓

a∧b = ¬(a⇒¬b) (a∨b) = ¬¬(a∨b) = ¬(¬a∧¬b) = (¬a↑¬b) = ((a↑a)↑(b↑b))

**6**

**Množiny** – pojem mnozina je matematickym naprotivkom pojmu subor (zoskupenie)

- objekt, ktory sa sklada z inych objektov (prvky) tej mnoziny; x∈A - x je v mn. A

- je jednoznacne dana svojimi prvkamy, teda tym, kt. prvky sa v nej vyskytuju a kt. nie, pricom na ich pocte a poradi nezalezi

- **prazdna mnozina** ∅ - neobsahuje ziaden prvok, t.j. pre kazde x plati, ze x !∈ ∅

- **zapisovanie mnozin** - a) ***vyctom prvkov*** - A = {2, 4, 6, 8}

b) ***charakteristickym znakom*** - prvky, kt. splnaju vlastnost (x) - {x | (x)}

B = {x∈R; (x mod 2)=0} - parne cisla

- **system mnozin** – mnozina mnozin; mnozina, ktorej prvkami su mnoziny

- **univerzum** - mnozina, ktora je nadradena vsetkym mnozinam, s kt. pracujeme

- **potencna mnozina** P(X)/2X - mnozina, ktorej prvkami su prave vsetky podmnoziny danej mnoziny X

P(X)={A | A⊆X}

napr. B={2,3,4} ; 2|B|=|23|=8 ; 2B={∅,{2},{3},{4},{2,3},{2,4},{3,4},{2,3,4}}

**Vztahy medzi mnozinami** – a) **rovnost** - A=B (A⊆B∧B⊆A) 🡺{x | x∈A ∧ x∈B} – mn. A je rovna mn. B

b) **inkluzia** - A⊆B 🡺 {x | x∈A ⇒ x∈B} - mn. A je podmnozinou mn. B

A⊆B ∧ B⊆C ⇒ A⊆C

**Operacie s mnozinami** - a) **prienik**∩- A∩B={x | x∈A a x∈B}

b) **zjednotenie** ∪ - A∪B={x | x∈A alebo x∈B}

c) **rozdiel** − (alebo \) - A−B={x | x∈A a x!∈B}

- (**navzajom) disjunktne mnoziny** – A∩B=∅

- **doplnok** – X=univerzum ; A⊆X ; doplnok A = X−A

- **Vennove diagramy**

**7**

**Konecna mnozina** – ma konecny pocet prvkov

- **n** je pocet prvkov mn. A a oznacujeme ho |A|, tj. |A|=n

- patri tu aj prazdna mnozina

**Nekonecna mnozina** - mn. A je nekonecna, ak nie je konecna; ma nekonecne vela prvkov |A|=

**Spocetna mnozina** - existuje bijekcia f: A→N ; deli sa na ***konecne spoc. mnoziny*** a ***nekonecne spoc. mnoziny***

**Nespocetna mnozina** – prave ked je nekonecna a nie je spocetna

**8**

**Zobrazenie (funkcia)** - matematicky naprotivok pojmu priradenie

f: X→Y - mozeme to chapat ako mn. dvojic <x,y>, kde y je objekt priradenia x

- binarna relacia medzi mn. X objekov, ktorym su priradovane objekty a mnozinou Y objektov, ktore su priradovane objektom z mn. X

- relacia R medzi mnozinami X a X sa nazyva zobrazenim mn. X do mn. Y, prave ked pre ∀x∈X existuje y∈Y tak, ze

<x,y>∈R

a pre ∀x∈X a y1,y2∈Y plati, že

<x,y1>∈R a <x,y2>∈R implikuje y1=y2

napr. X={a,b} ; Y={1,2,3}

relacia R={<a,1>,<a,3>} nie je zobrazenie X do Y

relacia S={<a,1>,<b,2>,<a,3>} nie je zobrazenie X do Y

relacia T={<b,1>,<a,1>} je zobrazenie X do Y

**Typy zobrazeni** - a) **injektivne**(proste) - pre ∀x1,x2∈X, pre x1!=x2 plynie f(x1)!=f(x2)

- kazdy prvok mn. Y je obrazom najviac 1 prvku mn. X

b) **surjektivne** - zobrazenie mn. X **na** mn. Y(surjektivnu), prave ked pre

∀y∈Y existuje x∈X tak, ze f(x)=y

- zobrazenie na urcitu mnozinu; na kazdy prvok z cielovej mn. priraduje aspon 1 prvok z vychodiskovej mnoziny

c) **bijektivne** - zobrazenie, kt. je naraz injektivne aj surjektivne

napr. X={1,2,3} ; Y={a,b,c,d} ; Z={°,#,|}

f:XY={<1,a>,<2,b>,<3,d>} - injektivne, nesurjektivne

g:YZ={<a,°>,<b,#>,<c,°>,<d,|>} - surjektivne, neinjektivne

f ° g: A→C = {<1,°>,<2,#>,<3,|>} – injektivne, surjektivne = bijektivne

**9**

**Relacia** - matematicky naprotivok pojmu vztah

- je urcena aritou relacie a mnozinami, ktorych prvky do vztahu vstupuju

- **arita -** cislo udavajuce pocet objektov, ktore do vztahu vstupuju

- relacia je mnozina pozostavajuca z n-tic prvkov prislusnych mnozin; obsahuje tie n-tice, ktore maju medzi sebou zamyslany vztah

- **kartezsky sucin** - kart. sucin mn. X1,...,Xn je mnozina X1×...×Xn definovana predpisom

X1×...×Xn = {<x1,...,xn> | x1∈X1,...,xn∈Xn}

ak X1=...=Xn=X potom X1×...×Xn=Xn  - n-ta kartezska mocnina mn. X

napr. A={{a},b} ; B={1} ; A×B={<{a},1>,<b,1>}

A=∅ ; B={a,b} – neexistuje A×B

- ak X1,...,Xn su mnoziny; relacia medzi X1,...,Xn je lubovolna podmnoznina ich kartezskeho sucinu

- **specialne relacie:** a) ***prazdna relacia*** ∅

b) **relacia identity** X={<x,x> | x∈X}

c) **kartezsky stvorec** X=X×X

**10**

**Operacie s (bin.) relaciami** - relacie su mnoziny, preto s nimi mozeme prevadzat mnozinove operacie {∩∪−} a mozeme na nic aplikovat vztah inkluzie⊆

- **inverzia** - R⊆X×Y 🡪 R-1⊆Y×X={<y,x> | <x,y>∈R}

- **skladanie** - R⊆X×Y a S⊆Y×Z

R°S⊆X×Z={<x,z> | ex. y∈Y : <x,y>∈R a <y,z>∈S}

- **ine sposoby skladania:** ◁ , ▷ ; □

R⊆X×Y a S⊆Y×Z

R◁S={<x,z>|∀y∈Y : <x,y>∈R ⇒ <y,z>∈S}

R▷S={<x,z>|∀y∈Y : <y,z>∈S ⇒ <x,y>∈R}

R□S={<x,z>|∀y∈Y : <x,y>∈R ⇔ <y,z>∈S}

□=◁∩▷

(R-1)-1=R ; (R°S)-1=S-1°R-1  ; R°(S°T)=(R°S)°T

**Znazornovanie bin. relacii** - a) **maticou (tabulkou)** - matica M typu m×n (m-riadky, n-stlpce) ma na kazdom mieste so suradnicami m,n nejaku hodnotu

- zakladny sposob reprezentacie binarnych relacii, nevyhodou je velka pamat. narocnost

- relaciu R reprezenujeme maticou(tab) tak, ze na mieste odpovedajucom riadku j a stlpci i oznacime, ci sa dvojica <xi,yj> nachadza v R (X/1-ano, \_/0 – nie)

-MR pre R={x1,...,xm}⊆{y1,...,yn} mij=1 ak <xi,yj>∈R

mij=0 ak <xi,yj>!∈R

-**operacie:** M, N typu m×n ; K typu n×k

M∨N=P, pij=max{mij,nij}

M∧N=P, pij=min{mij,nij}

M-N=P, pij=max{0,mij-nij}

M•K=P, pij=max{mil•kil; l=1,...,n}

MT, mijT=mji

+ operacie s relaciami

b) **grafom** - kazdy prvok x∈X znazornime v rovine kruzkom s oznacenim daneho prvku; ak <x,y>∈R, potom nakreslime z x do y ciaru (so sipkou)

c) **zoznamom zoznamov** – tuto reprezentaciu tvori hlavny (spojovy) zoznam, v kt. su vsetky prvky mn. X, z kazdeho x∈X vedie zoznam obsahujuce tie y∈X, pre kt. plati <x,y>∈R

**Pr.:** X={a,b,c,d} ; Y={1,2,3}

**11**

**Bin. relacie na mnozine** - su matematickym naprotivkom vztahov medzi 2 prvkami

**-** binarna relacia R na mnozine X != ∅ je podmnozina X×X, tj. R⊆X×X

**Vlastnosti** - a) **reflexivita** – ak pre ∀x∈X | <x,x>∈R

b) **irreflexivita** – ak pre ∀x∈X | <x,x>!∈R

c) **symetria** – ak pre ∀x,y∈X | <x,y>∈R ⇒ <y,x>∈R

d) **antisymetria** – ak pre ∀x,y∈X | (<x,y>∈R ∧ <y,x>∈R) ⇒ x=y

e) **asymetria** – ak pre ∀x,y∈X | <x,y>∈R ⇒ <y,x>!∈R

f) **tranzitivita** – ak pre ∀x,y,z∈X | (<x,y>∈R ∧ <y,z>∈R) ⇒ <x,z>∈R

g) **uplnost** – ak pre ∀x,y∈X | <x,y>∈R ∨ <y,x>∈R

**ekvivalencia** – relacia, ktora je zaroven reflexivna, symetricka a tranzitivna

**usporiadanie** – relacia, ktora je zaroven reflexivna, antisymetricka a tranzitivna

**∅ relacia** – tranzitivna, antisymetricka, symetricka, nie je reflexivna

**identita** – reflexivna, symetricka, antisymetricka a tranzitivna (ekvivalencia a usporiadanie zaroven)

**kartezsky stvorec** – reflexivna, symetricka, tranzitivna (ekvivalencia)

- ak |X|=1 - antisymetricka

**12**

**Uzavery relacii** – ku kazdej relacii dokazeme stanovit jej reflex., sym. alebo tranz. uzaver, t.j najmensiu reflex., sym. alebo tranz. relaciu na danej mn., kt. obsahuje vychodziu relaciu

- pre bin. relaciu R na X definujeme Ref(R) (Sym(R)/Tra(R)) na X tak, ze reflex. (sym/tra) relacia Ref(R) je reflexivna relacia obsahujuca R a pre kazdu reflex relaciu R´ na X, kde R⊆R´, mame Ref(R)⊆R´

Ref(R)=R∪X

Sym(R)=R∪R-1

Tra(R)=

**13**

**Ekvivalencia –** bin. relacia; matematicky naprotivok nerozlisitelnosti na X

- reflexivna, symetricka a tranzitivna relacia

- **tolerancia** – reflexivna a symetricka relacia ; tranz. tolerancia = ekvivalencia

- ekv. E na X definujeme pre ∀x∈X mnozinu [x]E = {y∈X | <x,y>∈E}

[x]E – obsahuje tie prvky z X, ktore nedokazeme od x rozlisit ekvivalenciou E

- X je najmensia ekvivalencia na X a X je najvacsia evk. na X

Napr. na X= {*a*,*b*,*c*} existuje 5 navzajom roznych ekvivalencii

X = {<a,a>, <b,b>, <c,c>},

E1 = {<a,a>, <a,b>, <b,a>, <b,b>, <c,c>},

E2 = {<a,a>, <b,b>, <b,c>, <c,b>, <c,c>},

E3 = {<a,a>, <b,b>, <c,c>, <a,c>, <c,a>},

X = {<a, a>, <a, b>, <a, c>, <b, a>, <b, b>, <b, c>, <c, a>, <c, b>, <c, c>}.

**Rozklad** – matematicky naprotivok zhlukovania nerozlisitelnych prvkov

- rozklad Π na X je system neprazdnych podmnozin Y tak, ze kazde 2 rozne podmnoziny Y1,Y2∈Π, Y1!=Y2 su disjunktne (Y1∩Y2=∅) a zjednotenie vsetkych mnozin Π je X

X!= ∅ ; Π⊆2X splnuje:

∀Y∈Π | Y!=∅ ∧ ∀Y1,Y2∈Π | Y1∩Y2 !=∅ ⇒ Y1=Y2 ∧ ∪Π = X

- rozklad na X je disjunktne pokrytie X (∪Π = X)

- na X existuju 2 medzne rozklady: a) ∀x∈X | [x]Π={x} – vsetky triedy rozkladu su 1-prvkove

b) ∀x∈X | [x]Π=X - Π={X} – obsahuje 1 triedu rovnu celemu X

Rozklad a ekvivalencia su vo vzajomne jednoznacnej korespondencii:

**Faktorova mnozina X podla E** – rozklad na mnozine X prislusny ekvivalencii E; X/E=ΠE

- obecna metoda zmensenia vychodzej mnoziny

- predstavuje „zjednoduseny pohlad“ na vychodziu mn. X, pri ktorom sme stotoznili tie prvky X, ktore od seba neboli rozlisitelne ekvivalenciou E

- pre konecne X a E!=X plati |X/E|<|X| (fakt. mn. < vychodzia mn.)

- vyuziva sa pri analyze dat alebo zhlukovani

**Surjektivne zobrazenie** – „zjednoduseny pohlad“ na vychodziu mn. X

**-** ak je zobrazenie f:X→Y surjektivne, potom mozeme obraz f(x) prvku x chapat ako „prvok x nahradime (zjednodusime) prvkom f(x)“

**-** pre f:X→Y definujeme Ef na X predpisom: <x,y>∈Ef ⇔ f(x)=f(y)

Ef – **ekvivalencia indukovana zobrazenim f**

- ak g: X→Y je zobrazenie, existuje injektivna funkcia h:X/Eg→Y tak, ze g=fEg°h

- ak je g surjektivne, tak h je bijekcia

**14**

**Usporiadanie** - reflexivna, antisymetricka a tranzitivna relacia R na X

- da sa interpretovat ako relacia urcujuca hierarchiu, popr. zavislost dat; vyuziva sa pri triedeni

-uplne usporiadanie sa nazyva ***retazec*** (***linerane usporiadanie***)

- <X, ≤ > sa nazyva **usporiadana mnozina**

- vyuziva sa oznacenie ≤ - usporiadanie; <x,y>∈≤ = x≤y

- ak x≤y a zaroven x!=y, potom x<y

- **nezrovnatelne prvky** – x,y∈X pre kt. plati x||y , tj. ani x≤y ani y≤x

- ak je ≤ linearne usporiadanie na X, tak ≤ je uplna relacia a kazde 2 prvky su zrovnatelne

- ≤-1 = ≥

- **antiretazec** – usporiadanie relacie identity X

- najmensie usporiadanie, pretoze kazde ≤ na X obsahuje X

- ak ≤ = X, potom pre kazde 2 rozne x,y∈X mame x||y

- **znazornenie** – maticou, orintovanym grafom, Hasseove diagramy

- **pokrytie** ≺ - x≺y ⇔ x<y a pre ∀z∈X | x≤z≤y ⇒ z∈{x,y}

- ≺⊆≤ - relacia pokrytia je obsiahnuta v usporiadani

- je irreflexivna, asymetricka (vlastnosti <) a nie je tranzitivna

**Hasseove diagramy** – su zalozene na relacii pokrytia

- skladaju sa z uzlov reprezentujucich prvky z X a hran, ktore znazornuju relaciu ≺ prislusnu danemu usporiadaniu ≤ na X

- uzly (body) znazornujeme v rovine, takze ak x>y, potom x je vyssie ako y

- 2 body x,y∈X spojime useckou ak x≺y

**15**

**Specialne prvky usporiadania** – ak <X, ≤ > je usporiadana mnozina, prvok x∈X moze byt:

a) minimalny – ak y∈X | y≤x ⇒ x=y

b) najmensi – ak y∈X | x≤y

c) maximalny – ak y∈X | y≥x ⇒ x=y

d) najvacsi – ak y∈X | x≥y

- ak <X, ≤ > je usporiadana mnozina, potom plati:

1. v <X, ≤ > existuje najviac 1 najmensi a 1 najvacsi prvok

2. ak je x∈X najvacsi (najmensi) prvok, tak je aj maximalny (minimalny)

3. ak je ≤ linearne usporiadanie, potom je x∈X najvacsi (najmensi) prvok, prave ked je maximalny (minimalny)

- **dolny kuzel v <X, ≤ >** – L(Y) = {x∈X | x≤y plati pre ∀y∈Y}

- obsahuje vsetky prvky z X, kt. su <= prvkom z Y

- **horny kuzel** **v** **<X, ≤ >** – U(Y) = {x∈X | x≥y plati pre ∀y∈Y}

- obsahuje vsetky prvky z X, kt. su >= prvkom z Y

- **infimum** – inf(Y) - najvacsi prvok L(Y)

- **supremum** – sup(Y) - najmensi prvok U(Y)

**Polozvaz** – <X, ≤ > je usp. mn. – a) **priesekovy polozvaz** – ak pre ∀x,y∈X existuje inf(x,y)

b) **spojovy polozvaz** – ak pre ∀x,y∈X existuje sup(x,y)

**Zvaz** – ak <X, ≤ > je zaroven priesekovy aj spojovy polozvaz, potom sa jedna o zvaz

- usporiadana mnozina, kde pre kazde 2 prvky existuje infimum a supremum

- kazdy konecny zvaz ma najmensi a najvacsi prvok, u nekonecnych zvazov to neplati

**16**

**Prirodzene cisla N** – cele cisla 0< = {1,2,3,4,5,6,...}

**Cele cisla Z** – cele cisla, kladne a zaporne – 0,1,-1,2,-2,3,-3,...

**Racionalne cisla Q** – cisla, ktore dokazeme vyjadrit v tvare zlomku (m-cele cislo, n-prirodzene cislo)

- napr. , ,

- zapisujeme ich aj pomocou **desatinneho rozvoja** - ;

**Realne cisla R** – vsetky cisla nachadzajuce sa na ciselnej ose

- okrem Q obsahuju aj iracional. cisla, tj. cisla, kt. nevieme vyjadrit v tvare zlomku (napr. ), π

**Komplexne cisla C** – mnozina vsetkych usporiadanych dvojic real. cisel <a,b>, ktore obvykle zapisujeme v tvare a+bi, kde i je **imaginarna jednotka** (i2 = -1)

- kazde cislo z=a+bi dokazeme vyjadrit v goniometrickom tvare z=r(cos + sin)

- r = – **absolutna hodnota**

-  - argument komplexneho cisla

**17**

**Princip indukcie** – umoznuje dokazovat tvrdenia v tvare „pre kazde prirodzene cislo n plati V(n)“, kde V(n) je tvrdenie, kt. zavisi na n

- pre ∀n∈N je dane V(n), potom predpokladame, ze plati:

1. V(1) – ***indukcny predpoklad***

2. pre ∀n∈N : z V(n) vyplyva V(n+1) – ***indukcny krok***

Potom ∀n∈N | V(n)

**Dokaz principu mat. indukcie** – predpokladame, ze kazda neprazdna mnozina K⊆N ma najmensi prvok. (\*) Princip indukcie dokazeme sporom, tj. predpokladame, ze neplati, tj. existuju tvrdenia V(n), n∈N, ktore splnaju princip indukcie, ale existuje n´, pre ktore V(n´) neplati. Mnozinu vsetkych takych n´ oznacime K={n´∈N |V(n´) neplati}, K!=∅, pretoze n´∈K. K ma najmensi prvok k!=1 (\*) (pretoze 1 !∈ K podla ind. predpokladu). Potom (k-1) !∈ K, teda V(n-1) plati. Z ind. kroku plynie, ze plati aj V(k), teda k !∈ K, co je spor s k∈K.

**Pr.:** Dokazte mat. indukciou .

**18**

**Delitelnost** – pre m,n∈Z plati , ked existuje k∈Z tak, ze

- jednoznacnost delenia so zbytkom – pre m,n∈Z existuju jednoznacne urcene k,r∈Z tak, ze

a 0<=r<b

r – ***zbytok*** po deleni cisla m cislom n (m mod n = r)

**Prvocisla –** prirodzene cislo n sa nazyva prvocislo, ak n != 1 a n je delitelne iba cislami 1 a n (napr. 2,3,11,29,...)

**Ciselne sustavy** – sposob, akym sa zapisuju cisla pomocou znakov

a) **pozicne** –hodnota kazdej cislice je dana jej poziciou v sekvencii symbolov

- klucovou jednotkov je zaklad (prir. cislo >1), ktoreho vahy jednotlivych cislic su jeho mocninami (dvojkova, osmickova, desiatkova,...)

b) **nepozicne** – sposob reprezentacie cisel, v ktorom nie je hodnota cislice dana jej umiestnenim v danej sekvencii cisel (rimske cisla, egyptske cisla,...)

**19**

**Kombinatorika** – zaobera sa urcovanim poctu moznosti, ktore mozu nastat za urcitych podmienok

- je uzko spata s bezpecnostou pocitacovych systemov, so sifrovanim a desifrovanim

- zakladnymi kombinatorickymi pravidlami su pravidlo suctu a sucinu

- **pravidlo suctu** – ak mozeme ulohu **A vykonat *m* sposobmi** a ulohu **B *n* sposobmi**, pricom ziaden z m sposobov **nie je totozny** so ziadnym z n sposobov, potom ulohu A **alebo** B mozeme vykonat sposobmi

**Pr.:** V kniznici je 4 knih o autora A, 6 knih od autora B a 2 knihy od autora C. Kolkymi sposobmi mozeme vybrat knihu od A alebo B alebo C? (4+6+2=12)

- **pravidlo sucinu** – ak vykonat ulohu C znamena najprv vykonat A **a potom** B, pricom **A vykoname *m*** **sposobmi** a **B *n* sposobmi**, potom vykonat ulohu C vieme sposobmi

**Pr.:** V meste vedu zo skoly do nemocnice 4 rozne trasy a z nemocnice k banke 3 rozne trasy. Kolkymi roznymi sposobmi sa vieme dostat zo skoly do banky? (4\*3=12)

**Permutacie** P(n) – permutacia n (navzajom roznych objektov) je lubovolne zoradenie tychto objektov

**Dokaz:** permutacia je specialny pripad variacie, kde k=n, cize:

Podla pravidla sucinu:

**Pr.:** Kolkymi roznymi sposobmi mozeme do radu postavit 10 ludi? ()

**Permutacie s opakovanim** P´(n) – permutacia n prvkov, pricom niektore prvky su rovnake

- mame danych **n objektov** rozdelenych do **k skupin**, ktore maju n1,...nk objektov, tj. spolu n1+...+nk=n objektov. Objekty v kazdej skupine su od seba **navzajom nerozlisitelne**. Kazde zoradenie tychto n objektov sa nazyva permutacia s opakovanim (pocet znacime P(n1,...,nk)).

**Pr.:** Kolkymi sposobmi mozeme rozsadit 9 ludi, pricom 3 maju modre nohavice, 4 cierna a 2 cervene? (

**20**

**Variacie** V(n,k) – je danych ***n*** navzajom odlisnych objektov a ***k***. **Variacie k-tej triedy z n objektov** (k<=n) je

lubovolny vyber k objektov z danych n objektov, pri ktorom zalezi na poradi

**Dokaz:**

**Pr.:** Kolkymi sposobmi mozu byt obsadene prve 3 miesta na futbalovom turnaji, ak sa ho zucastni 8 tymov? ()

**Variacie s opakovanim** V´(n,k) – su dane objekty ***n*** roznych typov, objektov kazdeho typu je neobmedzene

mnoho a su navzajom neodlisitelne. **Variacia k-tej triedy z n objektov s opakovanim** sulubovolny vyber k objektov z danych n objektov, pri ktorom zalezi na poradi.

**Dokaz:** 1. prvok vyberieme n sposobny, 2. taktiez, ..., k-ty prvok vyberieme n sposobmi.

**Pr.:** Kolkymi sposobmi mozeme zvolit 5 miestny kod pre trezor, ak na lubovolnom mieste mozu byt cisla 0-9? (

**Kombinacie** C(n,k) – je danych n navzajom odlisnych objektov a k (k<=n). **Komb. k-tej triedy z n objektov**

je lubovolny vyber k objektov z danych n objektov, pricom nezalezi na poradi vyberanych prvkov. Pocet takychto kombinacii znacime () – **kombinacne cislo**.

**Dokaz:** 🡺 🡺 ,

**Pr.:** 6 osob si navzajom poda ruku (kazdy s kazdym). Kolko bude podani ruky?

()

**Kombinacie s opakovanim** C´(n,k) – su dane objekty n roznych typov, objektov kazdeho typu je neobmedzen

mnoho a su navzajom neodlisitelne. **Kombinacie k-tej triedy z n prvkov s opakovanim** su lubovolny vyber k objektov z danych n objektov, pricom nezalezi na poradi vyberanych prvkov.

**Pr.:** Kolko roznych sucinov 2 cinitelov mozeme vytvorit z cisel 2,5,11,18?

**21**

**Binomicka veta** – matematicka veta, vychadzajuca z kombinatoriky, vdaka ktorej mozeme n-tu odmocninu 2 scitancov rozlozit na sucet n+1 scitanov

- pre realne cisla ***a,b*** a nezaporne cele cisla ***n*** je:

**Princip inkluzie a exkluzie** – Pre mnoziny plati:

- pocet prvkov prieniku mnoziny, ktorych index patri do *I*

tj. napr. pre *I*={2,3,4} je to

ak *I* obsahuje neparny pocet prvkov, inak

**22**

**Pravdepodobnost** P(A) – jedna z aplikacii kombinatorickeho pocitania

- predstavme si, ze sa kona pokus, ktory konci vysledkami (***elementarne javy***). Predpokladame, ze kazdy z vysledkov ma rovnaku sancu, tj. elementarne javy su rovnako pravdepodobne. ***Jav*** je kazda podmnozina . , teda pocet ***priaznivych javov*** ku poctu vsetkych moznych vysledkov. P(A)moze mat hodnotu od 0 do 1 (0 – ***nemozny jav***, 1 – ***isty jav***)

**Pocitanie pravdepodobnosti**: 1. urcime mnozinu E vs. elem. javov a overime, ze su vs. rovnako pravdepodobne

2. urcime jav A⊆E, kt. odpoveda danej udalosti

3. urcime pocet prvkov mn. E, tj. |E|

4. urcime |A|

**Pr.:** Hadzeme modoru a cervenou kockou. Aka je pravdepodobnost, ze na modrej kocke padne parne cislo a na cervenej neparne?

E={1,2,3,4,5,6} 🡺 – hracia kocka

A(m)={2,4,6} ; A(c)={1,3,5} 🡺

**23**

**Algoritmus** – predpis skladajuci sa z krokov a ktory zabezpeci, ze na zaklade vstupnych dat su poskytnute pozadovane vystupne data

- napr. vyriesenie kvadratickej rovnice, Dijkstrov algoritmus, ...

- **obsahuje** – hodnoty vstupnych dat, predpis pre riesenie a vystupne data

- **vlastnosti**: a) ***konecnost*** – kazdy alg. musi skoncit v konecnom pocte krokov

b) ***jednoznacnost*** – kazdy krok alg. musi byt jednoznacne a presne definovany

c) ***obecnost*** – algoritmus riesi celu triedu problemov, nie len 1 problem

d) ***rezultativnost*** – alg. pri zadani vstupnych dat musi vratit nejaky vystup

e) ***korektnost*** – vysledok algoritmu musi byt spravny

f) ***opakovatelnost*** – pri zadani rovn. vstup. dat musi alg. vratit vzdy rovnake vystupy

- **druhy algoritmov** – a) ***rekurzivny*** – volaju sami seba

b) ***„rozdel a panuj“*** – alg. deli problem na mensie, az trivialne, problemy, pricom tieto problemy vyriesi a vhodnym sposobom zluci

c) ***pravdepodobne*** – predpovedaju niektore rozhodnutia nahode alebo pseudonahodne

d) ***paralelne*** – rozdeluju ulohy napr. medzi viac pocitacov

e) ***heuristicke*** – hladaju vhodne priblizne riesenie, ak sa nedaju vyuzit exaktne algoritmy

...

**Rekurzia** (sebaopakovanie) – opakovane vnorene, volanie rovnakej funkcie

- musi obsahovat podmienku na zastavenie rekurzie, ak je podmienka nesplnena, nastava rekurzia

- po kazdom vnoreni musi nastat zjednodusenie problemu

- kazdy rekurzivny alg. sa da napisat v nerekurzivnom stave

- **typy** – **1. typ** – a) ***priama rekurzia*** – podprogram vola priamo sam seba

b) ***nepriama*** – funkcia A vola funkciu B, ktora vola funkciu A

**2. typ** – a) ***linearna*** – funkcia pri vykonavani ulohy vola samu seba iba raz

- vytvara sa linearna struktura postupne volanych f-cii

b) ***nelinearna*** – funkcia sa v ramci 1 vykonavania vola viackrat

- vzniknuta struktura sa znazornuje ako strom

**24**

**Konecne automaty** – teoreticky vypoct. model pouzivany v studiu vycislitelnosti a obecne formalnych jazykov

- popisuje jednoduchy pocitac, ktory moze byt v jednom z niekolkych stavov, medzi kt. prechadza na zaklade symbolov z vstupov

- mn. stavov je konecna

- nema inu pamat okrem aktualneho stavu

- **deterministicky konecny automat** – a) T – konecna mn. **vstupnych symbolov** (abeceda)

b) Q – konecna mn. **stavov**

c) **prechodova f-cia** popisujuca pravidla priechodu medzi stavami  : Q×T →Q

d) q0 – **pociatocny stav**, q0∈Q

e) F – mn. **koncovych stavov**, F⊆Q

- **reprezentacia** – tabulkou, prechodovou f-ciou , modifikovanym orientovanym grafom

**Cinnost KA** – 1. na zaciatku je automat v definovanom stave q0

2. v kazdom kroku precita jeden symbol na vstupe a prejde do stavu daneho prechod. f-ciou

3. opakuje 2. krok

4. ak skonci automat po precitani celeho vstupu v konc. stave, tak ***vstup prijal***, inak ho ***neprijal***

**Retazec** - retazec  prvkov mn. V je lubovolna konecna postupnost tejto mnoziny

**-** pocet prvkov mnoziny je **dlzka retazca** | |

**-** **pradzny retazec** - | | = 0

**Pozitivny uzaver** V+ - mn. vsetkych neprazdnych retazcov prvkov mn. V

- uzaver V\* je mn. V+ {}

- ak V je abeceda, potom V\* je ***formalny jazyk*** nad V

**25**

**Zlozitost algoritmu** – funkcia zavisla na velkosti vstupnych dat algoritmu

**-** a) **pamatova narocnost** – poziadavok na velkost pamate pocitaca, ktora je potrebna na vykonanie vypoctu

b) **casova narocnost** – cas potrebny na vypocet (obvykle pocet elem. krokov algoritmu)

- **radova velkost** – zlozitosti lisiace sa konstantnym nasobkom berieme ako rovnake

- **polynomicka cas. zlozitost** – ak existuje ***polynom p*** tak, ze f(n)<=p(n) pre ∀n∈N (n3+5)

- v praxi pouzivane alg. maju vacsinou cas. zlozitost **O(logN), O(N), O(NlogN), O(N2), O(N2logN), ..., O(2N), O(N!),** pricom stupen polynomu je nizky

**Radove porovnavanie funkcii**- ak f ,g : N→R su funkcie, potom:

a) f(n) = O(g(n)) ⇔ ex. c1>0, ex. n0>0, ze pre ∀n>=n0 | f(n)<=c1.g(n)

- **funkcia f rastie radovo najvys ako funkcia g**

b) f(n) = Ω(g(n)) ⇔ ex. c2>0, ex. n0>0, ze pre ∀n>=n0 | f(n)>=c2.g(n)

- **funkcia f rastie radovo aspon ako funkcia g**

c) f(n) = Θ(g(n)) ⇔ f(n) = O(g(n)) a f(n) = Ω(g(n))

- **funkcie f a g su asymptoticky ekvivalentne**

**Pr.:** Dokazte, ze plati: a)

b)

**26**

**Triedy zlozitosti:**

a) **polynomialna uloha** P – uloha (algoritmus), ktorej casovu zlozitost mozeme zhora ohranicit polynomom (napr. triedenie na halde - )

r∈U – r je riesenie ulohy U

**Uloha U1 je redukovatelna na ulohu U2**, ak existuje ***polynomialny algoritmus M***, ktory ***riesenie ulohy r*** prevadza na M(r) tak, ze r∈U1 prave ked M(r)∈U2. Oznacujeme U1◁U2.

b) **nedeterministicky polynomialna uloha** NP

– uloha, pre ktoru existuje nedeterministicky algoritmus, ktory ju riesi v polynom. case

**-** jediny deterministicky sposob riesenia uloh triedy NP je postupne prejst vsetky moznosti riesenia

**-NP-uplne ulohy** – **uloha U je NP-uplna**, ak na nu mozeme redukovat lubovolnu ulohy z triedy NP, tj. pre ∀U´∈NP plati U´◁U. Triedu NP-uplnych uloh znacime NPC

- vzhladom k cas. narokom povazujeme NP-uplne ulohy za neriesitelne

- vsetky NP-uplne ulohy su medzi sebou prevoditelne v polynomialnom case (su na seba redukovatelne); existuje ich viac ako 2000

- k ziadnemu NPC problemu nie je alg., kt. ho riesi v polynomickom case (preto ), v praxi sa tieto ulohy riesia iba priblizne (heuristicke alg., ...)

Predpoklada sa, ze , tj. .

**Priklady NP-uplnych problemov** – a) splnitelnost bool. f-cii

b) **uloha batohu** – mame mn. prirodzenych cisel a1,...an a cislo A. Ulohou je vybrat z cisiel podmnozinu, ktorej sucet bude A, teda ai1+ai2+...+aik=A.

-**nedeterministicky** – zvolim X=0, pre i=1,2,..,n bud ai pricita k X alebo nie. Ak X=A – splnene.

c) **chromaticke cislo grafu** – urcite najmensi pocet farieb, ktorymi musime vyfarbit vrcholy grafu G, ak 2 susedne vrcholy musia mat inu farbu; (G)

d) hladanie najkratsej (najdlhsiej) cesty v orientovanom ohodnotenom grafe z bodu x do bodu y

e) **najdenie Hamiltonovej kruznice** (obsahuje vsetky vrcholy grafu)

e) **obchodny cestujuci** – moze obchodnik prejst vsetky mesta tak, ze kazde navstivy prave raz a vrati sa do vychodzieho miesta tak, ze dlzka jeho cesty bude mensia ako K? (Exustuje v orientovanom grafe s ohodnotenymi hranami Hamiltonova kruznica, v ktorej je sucet ohodnoteni hrat ?)

**27**

**Graf** – graficke vyjadrenie vztahu medzi nejakymi objektami, ktore su reprezentovane kruzkami (***vrcholy grafu***)

- vztah medzi 2 vrcholmi grafu je vyznaceny ***hranou***, ktora tieto 2 uzly spaja

- **slucka** – hrana, ktorej oba konce vychadzaju z rovnakeho uzlu

- hrany mozu byt **orientovane** alebo **neorientovane**

a) **orientovany graf** – dvojica G=<V,E>, kde mnozina vrcholov a je usporiadana mnozina dvojic vrcholov (orientovanych hran)

b) **neorientovany graf** - dvojica G=<V,E>, kde mnozina vrcholov a je mnozina dvojprvkovych mnozin vrcholov (neorientovanych hran)

- **suvisly neo. graf** – pre ∀u,v existuje sled z u do v

- **strom** – suvisly neo. graf bez kruznic

- **podgraf** – neo. graf <V1,E1> je podgrafom grafu <V2,E2>, ak a

- **kostra neo. grafu** – podgraf grafu G, ktory obsahuje vsetky jeho vrcholy a hrany bez kruznic

- graf ma kostu prave ked je suvisly

**Sled** – v neorient. grafe G=<V,E> je to postupnost v0,e1,v1,e2,v2,...,en,vn kde vi∈V su vrcholy a ej∈E su hrany, pricom plati pre j={1,2,...,n}.

- **dlzka sledu** n

- **druhy** – a) **uzavrety** – v0=vn

b) **tah** – ziadna hrana sa neopakuje ( ⇒)

c) **cesta** – ziaden vrchol sa neopakuje ( ⇒)

d) **kruznica** – ak v0=vn , tak s vynimkou v0 a vn sa ziadne vrcholy neopakuju

**Vzdialenost** – z vrcholu u do v je to najmensia dlzka z u do v

**Hranove ohodnotenie** pre graf G=<V,E> s mnozinou hodnot D je funkcia w : E →D.

**28**

**Dijkstrov algoritmus** – hladanie najkratsej cesty z daneho vrcholu

**VSTUP**: neorientovany graf G=<V,E>, hranove ohodnotenie w : E →R+, vrchol s∈V

**VYSTUP**: hodnota d(v) pre ∀v∈V; d(v) je hodnota najkratsej cesty z *s* do *v*

**PREMENNE**: f-cia d : V →R+, cislo m∈R0+, mnoziny

**ALGORITMUS:**

1. A:=V ; d(s):=0 ; pre v∈V-{s}: d(s)=

2. ak neexistuje v∈A tak, ze , skonci

3. m:=min{d(v)|v∈A} ; N:={v∈A|d(v)=m} ; A:=A-N

4. pre ∀v∈A, u ∈ A take, ze {v,u}∈E: ak d(v)+w({v,u}) < d(u), potom

d(u) := d(v)+w({v,u}) ; pokracuje od 2. bodu

**29**

**Kruskalov algoritmus** – hlada minimalnu kostru suvisleho grafu G=<V,E>

**VSTUP**: suvisly graf G=<V,E>, |V|=n, |E|=m, hranove ohodnotenie w : E →R+

**VYSTUP**: minimalna kostra grafu, tj. kostra grafu s min. suctom ohodnoteni jeho hran

**ALGORITMUS**:

1. usporiadaj hrany z E podla vahy: w(e1)<= w(e1)<=...<= w(em)

2. vytvor prazdnu mn. hran E0

3. ak pridanim hrany ei vznikne kruznica, ponechaj Ei=Ei-1, inak Ei=Ei-1∪{ei}

4. opakuj 3. krok, pokym i<m a Ei este nema n-1 hran.