VII. kapitola. Informatika

In: Bohdan Zelinka (author): Matematika hrou i vážně. (Czech). Praha: Mladá fronta, 1979. pp. 168–182.

Persistent URL: http://dml.cz/dmlcz/403955

VII. KAPITOLA

INFORMATIKA

Mluvíme-li o matematice, nemůžeme jistě vynechat to, co dnes doslova hýbe světem — samočinné počítače. Nebudeme zde mnoho mluvit o samotných počítačích. Různé zajímavosti o tom, co všechno počítač umí, popřípadě humorné historky týkající se vztahu počítače a člověka, můžete nalézt v jiné literatuře. Zde si povíme něco o matematickém oboru, který s počítači souvisí; nazývá se informatika.

Název informatika není starý. Sama nauka ovšem také není stará, ale její název je podstatně mladší. Donedávna se pro ni z nedostatku vhodného českého výrazu užívalo názvu „computer science", což značí anglicky „věda o počítačích". Zahrnuje problematiku související s matematickou stránkou provozu počítačů; elektrotechnické záležitosti do ní nepatří. Nezaměňujme rovněž informatiku s kybernetikou; kybernetika je širší pojem než informatika — počítače jsou jen jednou stránkou její náplně.

O některých odvětvích informatiky si povíme v dalších odstavcích.

TEORIE ALGORITMŮ

Asi si pamatujete, že jsme v II. kapitole mluvili o jistém středověkém matematikovi s těžko zapamatovatelným

jménem. Byl to al-Chvárizmí. Jak to, že jsme se od počítačů dostali náhle do „temného" středověku?

Al-Chvárizmí ve své knize podal návod k řešení algebraických rovnic. Podle jeho polatinštěného jména nazýváme podobné návody algoritmy; podobnost se slovem logaritmus je čistě náhodná.

Aby však návod k nějakému matematickému postupu byl algoritmem, musí splňovat dvě základní podmínky: musí být použitelný pro řešení poměrně rozsáhlé třídy úloh a musí být popsán tak, aby v každém okamžiku postupu bylo jednoznačně určeno, jak se má pracovat dál.

Příkladem algoritmu je Eukleidův algoritmus pro nalezení největšího společného dělitele dvou přirozených čísel. Nechť jsou dána dvě přirozená čísla; větší z nich označme a, menší b. Chceme najít jejich největšího společného dělitele d. Postupujeme tak, že číslo a dělíme se zbytkem číslem b; zbytek označíme r_1. Je-li $r_1 = 0$, pak $d = b$. Je-li $r_1 \neq 0$, opakujeme postup tak, že místo a vezmeme b a místo b vezmeme r_1. Zbytek při dělení čísla b číslem r_1 označíme r_2. Je-li $r_2 = 0$, je $d = r_1$; jinak opakujeme postup tak, že číslo r_1 dělíme číslem r_2. Dostáváme postupně čísla $r_1, r_2, r_3, \ldots$. Všechna tato čísla jsou nezáporná a každé následující je menší než předcházející, tedy pro nějaké n musíme dostat $r_n = 0$; v tom případě $d = r_{n-1}$.

Ukažme si to pro $a = 2520$, $b = 322$. Dělíme $2520 : 322 = 7$, zbytek 266. Dále $322 : 266 = 1$, zbytek 56. Potom $266 : 56 = 4$, zbytek 42. Pak $56 : 42 = 1$, zbytek 14. Nakonec $42 : 14 = 3$, zbytek 0. Máme $d = 14$. Přehlednou tabulku vidíme na obr. VII.1.

Program pro samočinný počítač se tvoří ve formě algoritmu. Na takovýto program se totiž kladou právě ty požadavky, o nichž jsme mluvili. Pokud by určitý

návod k postupu byl použitelný pouze pro malý počet úloh, tedy například pouze pro nalezení největšího společného dělitele čísel 2520 a 322, nemělo by smyslu jej programovat pro počítač; bylo by jednodušší si to vypočítat na papíře. A protože počítač není člověk, nemůžeme chtít, aby samostatně uvažoval; chceme-li něco od něho, musíme mu přesně popsat postup, aby v každém okamžiku práce věděl, co má dělat dál.

DĚLENEC	DĚLITEL	ZBYTEK
2520	322	266
322	266	56
266	56	42
56	42	14
42	14	0

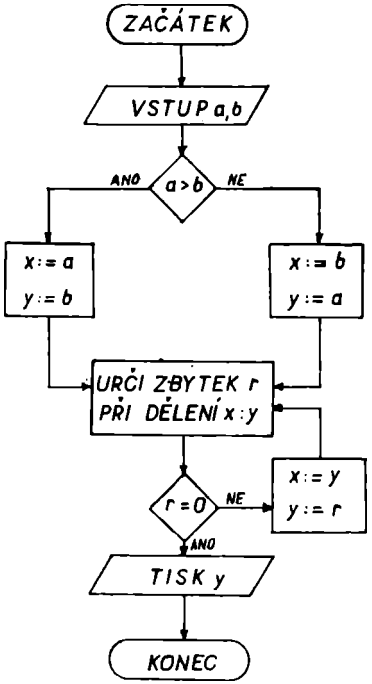
Obr. VII.1

Algoritmus se přehledně zobrazuje vývojovým diagramem. Je to vlastně orientovaný graf, jehož uzly jsou jednotlivé pracovní etapy a jehož hrany uvádějí následnost těchto etap. Vývojový diagram Eukleidova algoritmu vidíme na obr. VII.2.

Algoritmy pro počítač je nutno vypracovat tak, aby byly efektivní, to jest aby výpočet podle nich netrval příliš dlouho. I když počítač vykonává jednotlivé početní úkony ve zlomcích sekundy, při nevhodném programu by mohl pracovat příliš dlouho. A cena za jednu hodinu práce počítače se uvádí ve stovkách korun.

Nejde však jen o sestavování algoritmů, ale často i o určování, zda vůbec lze algoritmus sestavit. Ono to totiž vždy nejde; jsou úlohy, o nichž říkáme, že jsou

algoritmicky neřešitelné. Nebudeme si žádné z nich uvádět; stačí, když budete vědět, že o některých úlohách lze tvrdit, že je sebedokonalejší počítač nikdy nevyřeší. Zde teorie algoritmů těsně souvisí s matematickou logikou. S těmito problémy souvisí pojem Markovova algoritmu a Turingova stroje, což je jakýsi idealizovaný počítač. Nelze jej ve skutečnosti sestrojit,



Obr. VII.2

protože by musel obsahovat jistou nekonečnou pásku, ale lze o něm provádět teoretické úvahy, které se pak aplikují na skutečné počítače. (Mohli bychom jej přirovnat k ideálnímu plynu, který také neexistuje, ale pomocí něhož lze provádět úvahy potřebné při zkoumání skutečných plynů.)

JAK SE DOMLUVIT S POČÍTAČEM

Jak jsme se už zmiňovali, pro počítač je nutno vypracovat program, to znamená sdělit mu, co od něho chceme. Program lze napsat ve strojovém kódu, který je ovšem pro každý typ počítače jiný a vždy je těžko zapamatovatelný. Nelze s ním pracovat bez nahlížení do příslušných tabulek. Existují však také takzvané univerzální programovací jazyky. Jsou to určité systémy symbolů, kterým se programátor může naučit tak, jako se učí cizímu jazyku. Programu napsanému v takovémto jazyce mohou rozumět různé počítače; každý z nich si jej prostě přeloží do svého kódu.

Ukažme si, jak vypadá jednoduchý program v jazyce ALGOL. Je to jazyk, který se skládá z matematických symbolů a některých anglických slov. Chceme-li tedy, aby nám počítač Eukleidovým algoritmem určil největšího společného dělitele čísel a a b, můžeme mu to „říci" takto:

begin integer a, b;
if $a > b$ **then** $x := a; y := b$ **else** $x := b; y := a$;
A: $z := x \div y; r := x - y \times z$;
if $r = 0$ **then** *print* (y) **else** $x := y; y := r$; **go to** A
end

Nebudeme tento program blíže rozbírat. Je docela zají-

mavé vidět třeba turecký překlad některé známé české básně, i když turečtině nerozumíme. Zde je situace podobná; víme, o co jde, i když nerozumíme jednotlivým slovům. Ostatně kdo umí anglicky, pro toho to tak docela „turečtina" nebude.

Možná, že se vám zdá divné, že něco takového nazýváme jazykem. Souvisí to však s jistým odvětvím aplikované matematiky, které se nazývá matematická lingvistika. V něm je základním pojmem formální jazyk. Mějme jistou množinu A základních symbolů (říká se jí základní množina nebo abeceda). Množina A^* všech konečných posloupností symbolů z A (říká se jim slova) včetně prázdné posloupnosti (neobsahující žádný symbol) je monoid (viz II. kapitola) vzhledem k operaci zřetězení (například zřetězením slov *aba* a *bcda*, kde a, b, c, d jsou prvky množiny A, je slovo *ababcda*). Říká se mu volný monoid nad množinou A. Formální jazyk (A, L) je pak libovolná podmnožina tohoto monoidu; tento formální jazyk se pak zkoumá matematickými metodami. Lze toho užít ke studiu přirozených jazyků jako například češtiny — zde je základní množinou množina všech slov příslušného jazyka a „slovy" jsou věty z těchto slov utvořené. Jazyk pak je chápán jako množina všech gramaticky správných (byť třeba významově nesmyslných) vět příslušného přirozeného jazyka. Ovšem jsou formální jazyky, které mají význam čistě matematický — například jazyk nad abecedou $\{a, b\}$ složený ze všech slov, kde se na začátku n-krát vyskytuje symbol a a za ním n-krát symbol b, pro libovolné n.

Matematická lingvistika zkoumá obecné zákonitosti jazyků, čímž pomáhá jak jazykovědě, tak tvorbě programovacích jazyků, a zasahuje i do dalších oblastí informatiky.

Nepůjde zde o automat, který po vhození koruny zahraje „O sole mio", ani o místo, kde lze vstoje u stolku sníst párek s hořčicí. Automaty, o nichž bude řeč, jsou určité matematické objekty, které mohou popsat činnost určitého automatického zařízení (třeba nakonec i toho hudebního automatu). Ukážeme si to na příkladě.

Představme si, že bychom v nějaké tlusté knize měli určit počet všech slov, která začínají písmenem K a končí písmenem L. Je to práce galejnická, proto bychom ji rádi přenechali stroji. Takovýto automat by četl jednotlivá písmena textu, přičemž mezera mezi slovy by se rovněž považovala za písmeno. Jakmile by přečetl slovo zmírněného tvaru, napsal by čárku.

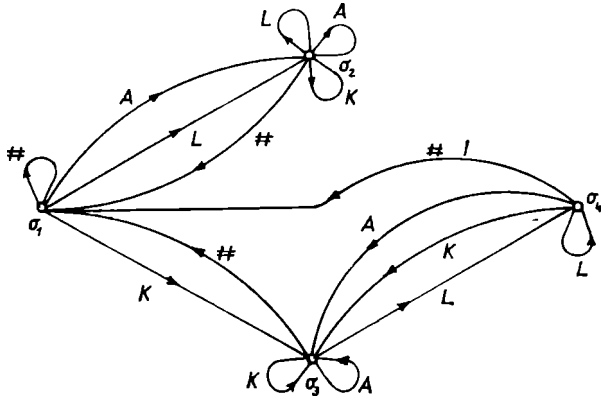
Teorie konečných automatů umožňuje popsat princip činnosti takovéhoto stroje. Vžijme se do jeho situace; představme si, že jsme žáky první třídy základní školy, že už známe abecedu, ale dosud nedovedeme vnímat napsané slovo jako celek, musíme je číst po jednotlivých písmenech. Chceme-li provést požadovaný úkon, musíme si při čtení každého písmene pamatovat vždy právě jednu z těchto věcí:

1. Teď přijde nové slovo.
2. Slovo, které právě čtu, nezačíná písmenem K.
3. Slovo, které právě čtu, začíná písmenem K, ale písmeno, které jsem četl naposledy, není L.
4. Slovo, které právě čtu, začíná písmenem K, a písmeno, které jsem četl naposledy, je L.

Označme si tyto jednotlivé případy σ_1, σ_2, σ_3, σ_4; říkáme jim stavy vnitřní paměti automatu nebo krátce stavy automatu. Je jich konečný počet, proto mluvíme o konečném automatu.

Tyto stavy se mění podle toho, jaké písmeno automat

přečte čili jaký vstupní symbol je do něho vložen. Rozlišujeme, zda automat čte K, L, nějaké jiné písmeno (můžeme je vždy značit A) nebo mezeru (značíme #). Je-li například automat ve stavu σ_1 a čte K, přejde do σ_3, protože začal číst slovo začínající písmenem K. Čte-li L nebo A, přejde do σ_2, čte-li #, zůstane v σ_1.



Obr. VII.3

Bude psát čárku neboli vydá výstupní symbol | právě tehdy, byl-li ve stavu σ_4 a následuje-li #; znamená to, že právě přečetl hledané slovo.

Můžeme si to znázornit orientovaným grafem (vlastně multigrafem), jehož uzly budou stavy automatu a hrany budou znázorňovat přechod z jednoho stavu do druhého při určitém vstupním symbolu (ten symbol je vždy u hrany označen). Ke hraně se symbolem # z σ_4 do σ_1 připíšeme ještě čárku; značí to, že v tomto případě automat píše čárku. Graf vidíme na obr. VII.3.

Není zde pravopisná chyba, protože nejde o byt s ústředním topením ani bez něho, ale o jednotku množství informace. Je to zkratka anglického „binary digit" čili „dvojková číslice".

V paměti počítače musejí být uloženy různé informace; jsou tam uloženy ve formě posloupností nul a jedniček. Proč tomu tak je, víme už z II. kapitoly, z odstavce o číselných soustavách. Nejmenší počet takových symbolů, které jsou potřebné pro zakódování určité informace, je právě počet bitů, který tato informace má. Tedy jako délky měříme na metry a hmotnost na kilogramy, měříme množství informace v bitech.

Má-li nějaká informace *n* bitů, znamená to také, že *n* je nejmenší počet otázek, který musíme položit, abychom tuto informaci s jistotou získali, můžeme-li dostávat pouze odpovědi „ano" nebo „ne".

Představme si, že náš kamarád vytáhl z mariášové hry jednu kartu a my se chceme dovědět, která to je. Pokud bychom se rovnou zeptali: „Je to zelená osmička?" a odpověď by byla „ano", víme to ovšem hned. Pokud by byla odpověď „ne", nevěděli bychom o mnoho více než předtím. Můžeme však položit určitých pět otázek tak, že žádanou informaci zcela jistě získáme.

Příklad:

„Představuje barva této karty nějakou část rostliny?"
„Ne."
„Je to červená karta?" „Ano."
„Je to karta od desítky níže?" „Ne."
„Je to jedna z nejvyšších dvou karet?" „Ano."
„Je to eso?" „Ne."
„Je to červený král!"

Méně než pěti otázkami to s určitostí zjistit nemůžeme.

Informace o určité kartě z mariášové hry má 5 bitů. Je to pochopitelné; počet karet je 32, tedy 2^5. Kdybychom tyto karty očíslovali a čísla zapsali ve dvojkové soustavě, dostali bychom čísla pětimístná.

Podobnými problémy se zabývá teorie informace; je rovněž součástí informatiky a od ní informatika dostala své jméno. Má význam i pro spojovací techniku. Lze pomocí ní například určit, jak máme nějakou důležitou zprávu zakódovat, aby byla co nejstručnější, ale aby na druhé straně chyba v některém znaku nezpůsobila podstatné zkreslení zprávy. Teorie informace souvisí i s lingvistikou.

Větší jednotkou než bit je byte (čti „bajt"). Je to 8 bitů. Proč právě osm a ne třeba deset nebo sto, pochopíte opět z odstavce o číselných soustavách. Další násobky této jednotky jsou kilobyte a megabyte. Kilobyte nemá přesně tisíc, ale $2^{10} = 1024$ bytů. Megabyte má 1024 kilobytů.

TEORIE HER

V VI. kapitole jsme se zmiňovali o některých hrách, kde výsledek závisel na náhodě (hra v kostky, ruleta). Samozřejmě kromě ryze hazardních her jsou i hry ušlechtilejší, v nichž výhra závisí pouze na schopnostech hráče (šach) nebo částečně na schopnostech a částečně na náhodě (běžné karetní hry). Není divu, že matematika nezanedbává ani tyto hry.

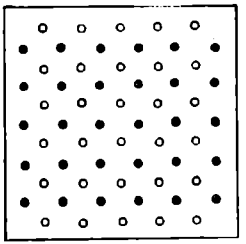
Šach, dáma, mlýn a podobné deskové hry patří mezi takzvané hry s úplnou informací — každý hráč přesně zná situaci hry, nikdo nic nezatajuje. Naproti tomu ka-

retní hry jsou v převážné většině hry s neúplnou informací — žádný hráč nezná karty svých soupeřů.

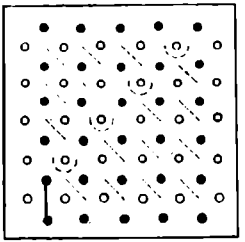
Řada her, mezi nimi i šach, může být vyjádřena pomocí orientovaných grafů. Uzly grafu jsou jednotlivé pozice hry spolu s údajem, který hráč je na tahu. (Tedy každé pozici odpovídají dva různé uzly, každý z nich znázorňuje situaci, kdy je určitý hráč na tahu.) Z každého uzlu vedou hrany do uzlů znázorňujících pozice, které mohou následovat bezprostředně po odpovídající pozici. Můžeme si potom představovat, že se hra místo na šachovnici nebo jiné hrací desce hraje na grafu, a to s jediným hracím kamenem. Hráči střídavě tahají kamenem po orientovaných hranách grafu. Prohrává ten hráč, který je na tahu a přitom nemůže dále táhnout (neexistuje hrana vycházející z příslušného uzlu).

Nakreslit takovýto graf pro šach by byla ovšem nadlidská práce. A stejně je zatím nejen nad síly člověka, ale i nad síly samočinných počítačů vůbec provést nějakou dokonalou analýzu této hry. Počítače sice hrají šach, ale zdaleka ne zázračně; hrají prostě jako dobří šachisté. Zda je možné, aby jednou počítač porážel velmistry, o tom se názory (mezi matematiky i šachisty) různí. (Zmiňme se v této souvislosti o tom, že exmistr světa v šachu M. Botvinnik je matematikem a zabývá se právě matematickou teorií šachu.) Ještě větší potíže než šach dělá počítačům bridž a japonská desková hra go. Šachisté mohou být rádi — kdyby byl vypracován přesný postup, který by jednoho z hráčů mohl za všech okolností vést k výhře (vyhrávající strategie) nebo alespoň k remíze, šach jako hra by přestal existovat. Teoreticky takováto strategie musí existovat, ale je nad síly lidí i počítačů ji popsat. Jsou však hry, u nichž je takový postup popsán. Některé jednoduché jsou uvedeny ve cvičeních. Zde si všimneme hry, která je trochu

složitější. Je to „bridge-it", v doslovném překladu „přemosti to". (Nezaměňovat s karetní hrou bridž.) Hrají dva hráči, bílý a černý. Hráči mají papír s nákresem z obr. VII.4. Tahají střídavě, začíná černý. Tah každého hráče spočívá v tom, že hráč spojí dva sousední body své barvy vodorovnou nebo svislou úsečkou. Úsečky se



Obr. VII.4



Obr. VII.5

nesmějí protínat. Vyhrává hráč, který první spojí lomenou čarou dva body své barvy na protilehlých stranách obrazce. (Pokud by se to nepodařilo nikomu a jeden z hráčů by už nemohl nakreslit další úsečku, tento hráč by prohrál.) Jak ukázal O. Gross, černý může v této hře vždy vyhrát. První úsečku musí vést tak, jak je znázorněno na obr. VII.5. Jestliže bílý vede úsečku procházející koncovým bodem některé z úseček nebo oblouků označených na tomto obrázku čárkovaně, černý musí vést úsečku procházející druhým koncovým bodem. (Tato úsečka je určena jednoznačně.) Vede-li bílý úsečku, která žádným popsaným bodem neprochází, černý může vést libovolnou úsečku. Jestliže černý takto postupuje, zaručeně vyhraje.

Zmiňme se ještě o maticových hrách. Patří mezi ně

například takové hry, v nichž oba hráči konají tah současně nezávisle jeden na druhém, jako v této hře:

Hrají dva hráči A a B. Každý položí korunovou minci na stůl a zakryje ji, aby ji druhý neviděl. Pak se obě mince současně odkryjí. Jsou-li obě mince vzhůru pannou, platí hráč A hráči B čtyři koruny, jsou-li obě vzhůru lvem, platí dvě koruny. Je-li mince hráče A vzhůru lvem a mince hráče B vzhůru pannou, platí hráč B hráči A dvě koruny; je-li mince hráče A vzhůru pannou a mince hráče B vzhůru lvem, platí hráč B hráči A čtyři koruny.

Ukazuje se, že nejvhodnější stratégie hráče A je taková: hodit si hrací kostkou a obrátit minci pannou vzhůru tehdy, padne-li číslo dělitelné třemi; v opačném případě obrátit minci vzhůru lvem. Hráč B by to měl dělat zase obráceně. Zde ovšem hraje svou roli náhoda a tedy i teorie pravděpodobnosti.

Zdálo by se, že teorie her slouží pouze kratochvíli. Není tomu tak. Hra ve smyslu této teorie nemusí být jen společenská hra, ale zahrnují se do tohoto pojmu i velmi vážné věci. Může to být válka nebo diplomatická jednání, ale také třeba plánování výživy obyvatelstva. To je „hra proti přírodě", obdobná oné hře s mincemi. Příroda na nás neočekávaně posílá mrazy, vedra, deště, sněhy, krupobití a podobné „tahy" a národohospodáři musejí vést „hru" tak, aby od přírody „vyhráli" co nejvíce.

Úlohy

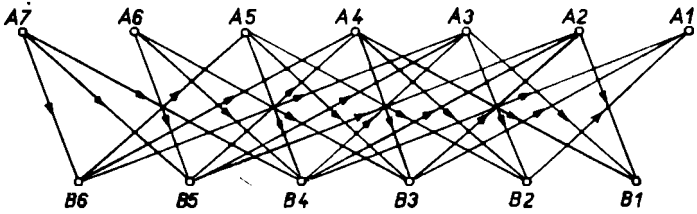
1. Na stole leží patnáct zápalek. Hrají dva hráči, kteří střídavě odebírají jednu, dvě nebo tři zápalky. Prohrává hráč, na němž zbude poslední zápalka. Určete strategii, která jednoho z hráčů povede s jistotou k vítězství.

2. Nakreslete graf této hry. Pro jednoduchost počítejte se sedmi zápalkami.

3. Zkoumejte tuto hru pro jiné počty zápalek než patnáct.

4. Hráči střídavě kladou na obdélníkový stůl vždy po jedné minci; všechny mince jsou stejné. Mince musí ležet celou svou plochou na desce stolu, nesmí ani částečně překrývat jinou minci. Prohrává hráč, který už nemá kam položit minci. Určete vyhrávající strategii.

5. Hra z úlohy 4 je obměněna tak, že přesně uprostřed stolu stojí váza s květinami, která má kruhové dno. Vázu není dovoleno se stolu sejmout.



Obr. VII.6

Řešení úloh

1. Hráč, který začíná, může vždy vyhrát. Dosáhne toho tak, že poprvé vezme dvě zápalky a potom vždy $4 - n$ zápalek, kde n je počet zápalek, které odebral druhý hráč v předešlém tahu.

2. Graf je na obr. VII.6. Každý uzel je označen symbolem hráče (A nebo B) a počtem zápalek, které zbývají v dané pozici.

3. Pro jakýkoliv počet zápalek, který při dělení čtyřmi dává zbytek tři, je postup stejný jako u patnácti zápalek. Je-li zbytek při dělení počtu zápalek čtyřmi roven dvěma, bere první hráč poprvé jednu zápalku, je-li roven nule, bere tři zápalky. Dále postupuje stejně jako v předchozím případě. Je-li tento zbytek roven jedné, existuje vyhrávající strategie pro druhého hráče (který nezačíná). Tento hráč opět bere $4 - n$ zápalek, kde n je počet zápalek, které odebral první hráč v předchozím tahu.

4. První hráč poprvé položí minci přesně doprostřed stolu. Potom klade minci vždy souměrně podle středu stolu k minci, kterou položil druhý hráč v předešlém tahu. Takto první hráč zaručeně vyhraje.

5. V tomto případě má vyhrávající strategii druhý hráč. Opět klade vždy minci souměrně podle středu stolu k minci, kterou položil první hráč v předešlém tahu.