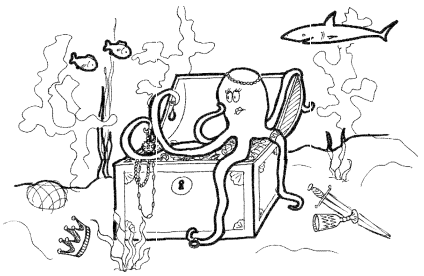
INFORMATIKA

Složitost algoritmů

ARNOŠT VEČERKA

Přírodovědecká fakulta UP, Olomouc



Algoritmus je přesný návod nebo popis dostatečně elementárních operací, jak vyřešit daný *typ úlohy*. Od algoritmu tedy požadujeme jeho *hromadnost*, tj. vstupní data nejsou reprezentována konkrétními hodnotami, ale údaji se symbolickými jmény, za něž lze dosazovat různá konkrétní data. Dále se požaduje *determinovanost* a *rezultativnost*, vždy musí být jednoznačně určeno, který krok algoritmu následuje, stejná vstupní data musí vést ke stejnému výsledku, přičemž některé vlastnosti výsledku dovedeme dopředu formulovat. A na závěr uveďme *konečnost* – po konečném počtu kroků se provádění instrukcí zastaví, přičemž požadovaný výsledek musí být poskytnut v rozumném čase. Za rozumný čas lze považovat čas, kdy nám výsledek ještě k něčemu bude.

V běžném životě je takovým příkladem algoritmu recept na přípravu jídla obsažený v kuchařské knize, neboť jsou v něm popsány jednotlivé kroky, jak dané jídlo připravit a samozřejmě je popsaný postup takový, že jsou splněny všechny uvedené vlastnosti algoritmu. Vedle vlastního popisu postupu je významným rysem algoritmu jeho složitost. Když vezmeme příklad receptu na přípravu jídla, bude záležet na tom, pro kolik osob jídlo budeme připravovat. Čím více těchto osob bude, tím budeme potřebovat např. jednak *větší nádoby* a *lepší vybavení kuchyně* a jednak nám *bude*

i příprava jídla trvat déle. Budeme muset třeba oškrábat více brambor nebo připravit více plátků masa.

Obdobně v algoritmech, které používáme pro výpočty nebo pro práci s daty (třídění, vyhledávání, grafové algoritmy a další) zpravidla platí, že čím více údajů budeme zpracovávat, tím budeme potřebovat větší rozsah paměti pro jejich uložení a rovněž čas výpočtu bývá delší. Závislost velikosti potřebné paměti na počtu zpracovávaných údajů nazýváme *paměťovou složitostí algoritmu* a závislost doby výpočtu na počtu zpracovávaných údajů nazýváme *časovou složitostí algoritmu.* U algoritmů pro nás bývá více důležitá zpravidla jejich časová složitost. Paměťová složitost má přirozeně také svůj význam. Nicméně při současných velikostech pamětí počítačů nebývá u většiny algoritmů omezujícím faktorem a proto se jí při běžném použití algoritmů výrazněji nezabýváme (jinak je tomu při hromadném zpracování dat).

Vyjadřovat časovou složitost přímo v čase, který udává dobu výpočtu, by bylo velmi obtížné a problematické. Například vezměme zcela jednoduchou úlohu, kdy máme vypočítat součet n čísel. Zřejmě bychom postupovali tak, že bychom vzali první číslo a k němu postupně přičítali druhé, třetí až n-té číslo. Celkově bychom udělali $n-1$ operací sčítání. Pokud bychom to dělali ručně, pak by celkový čas byl $n-1$ násobkem času, který potřebujeme pro jedno sčítání (pokud nepřihlížíme k tomu, že by sčítání bylo stále obtížnější). Jestliže to ale budeme provádět pomocí programu, bude určení času potřebného pro přičtení jednoho čísla obtížné. Program totiž většinou píšeme v nějakém programovacím jazyce a jak dlouho v něm bude trvat přičtení jednoho čísla při vlastním výpočtu, přímo nevíme. Přesněji řečeno, je to značně obtížné určit, neboť program je před výpočtem nejdříve přeložen do jazyka procesoru, což znamená, že bychom museli časovou složitost počítat ze strojových instrukcí přeloženého programu. Náš program pro sečtení n čísel bude patrně obsahovat cyklus, což reprezentuje několik strojových instrukcí, které zvyšují hodnotu proměnné cyklu a které dále zajišťují, aby se cyklus po sečtení všech čísel ukončil. Sčítané hodnoty budeme mít patrně uloženy v poli, což znamená, že v programu budou strojové instrukce pro výpočet adresy prvku pole, jehož hodnotu přičítáme. Je zřejmé, že určení délky výpočtu ze strojových instrukcí by bylo poměrně komplikované a to i u tak jednoduchého algoritmu, jako je sečtení n čísel. Navíc doba výpočtu bude také záviset na tom, jak výkonný počítač pro výpočet použijeme. Abychom tyto nepříjemné aspekty eliminovali, počítáme časovou složitost nikoliv v časových

jednotkách, ale v počtech základních operací, na kterých je příslušný algoritmus založen. Stanovení základních operací je dáno povahou algoritmu. Například u algoritmů třídění, kde srovnáváme prvky a děláme jejich přesuny, je základní operací srovnání dvou prvků a další základní operací je přesun prvku. V našem příkladu součtu čísel je základní operací sčítání. Časová složitost tohoto algoritmu je pak vyjádřena lineární funkcí $n-1$, neboť právě tolik sčítání algoritmus provede.

Obecně jsou časové složitosti vyjádřeny matematickými funkcemi, jejichž argumentem je zpravidla n počet údajů, které zpracováváme (zůstaňme jen u tohoto případu). Velmi často tyto funkce jsou přímo polynomy, z ostatních funkcí se poměrně často vyskytuje logaritmická funkce. Například funkce časové složitosti mohou být

$$\frac{n^2}{4} - n + 3 \qquad 3n \log_2(n-1) \qquad \text{atd.}$$

Časová složitost nám sice neudává přímo čas výpočtu, ale dá se z ní odhadnout, jak dlouho řádově bude výpočet pro daný počet údajů trvat na běžném počítači (vteřiny, minuty, hodiny atd.), resp. jak se výpočet prodlouží, zvětší-li se n. To nám stačí k tomu, abychom posoudili, zda daný algoritmus je pro nás použitelný. Dále je časová složitost důležitá v případě, kdy máme pro danou úlohu více algoritmů a máme se rozhodnout, který z nich použít. Často máme na jedné straně algoritmy, které jsou jednoduché, ale mají větší časovou složitost, a na druhé straně máme algoritmy s příznivější časovou složitosti, které jsou ale komplikovanější, čímž jejich implementace, tj. naprogramování je obtížnější. V tomto případě nám pro menší počty údajů stačí jednodušší algoritmus a naopak máme-li hodně údajů, musíme zvolit sice pracnější, ale na druhé straně rychlejší algoritmus.

Věnujme se nyní vlastnímu vyjádření časové složitosti. Uvažujme následující tři funkce:

$$\frac{n^2}{4} - n + 3 \quad \frac{n^2}{4} \quad n^2$$

Nechť první z těchto funkcí je časovou složitostí $T(n)$ nějakého konkrétního algoritmu. Druhou jsme dostali zanedbáním jejího lineárního a absolutního členu a v třetí jsme navíc odstranili koeficient u kvadratického členu. Všechny tři funkce jsou kvadratickým polynomem. Přitom ta poslední je

z nich nejjednodušší. Vzhledem k tomu, že u všech jde o polynom stejného řádu, mají všechny obdobný charakter růstu funkční hodnoty při rostoucím počtu údajů, tj. při zvyšující se hodnotě argumentu n. Už jsme uvedli, že časovou složitost používáme nikoliv k přesnému určení doby výpočtu, ale jen k základnímu odhadu závislosti času zpracování na (např. počtu vstupů) n. K tomu nám místo první funkce, která sice přesně vyjadřuje složitost algoritmu, ale je poměrně komplikovaná, zcela vyhoví mnohem jednodušší poslední funkce n^2. Pro tento účel se používá operátor *Theta*, který reprezentuje všechny funkce, jež mají stejný charakter složitosti, jako má funkce v něm uvedená. Tj. místo abychom psali, že algoritmus má časovou složitost

$$\frac{n^2}{4} - n + 3,$$

napíšeme jen, že jeho časová složitost $T(n)$ je

$$\Theta(n^2).$$

Skutečnost, že dvě funkce časové složitosti mají obdobný charakter, můžeme matematicky vyjádřit tak, že existují čísla $0 < a \leq b < +\infty$ taková, že pro všechna n od určitého n_0 počínaje platí

$$0 < a \leq \frac{T(n)}{f(n)} \leq b < +\infty,$$

jinak

$$0 < \lim_{n \to +\infty} \sup \frac{T(n)}{f(n)} < +\infty,$$

což v mnoha případech prakticky znamená, že podíl $\frac{T(n)}{f(n)}$ konverguje ke konečné nenulové hodnotě pro rostoucí argument n. Je-li tedy časová složitost algoritmu dána funkcí $T(n)$, pak můžeme říci, že časová složitost algoritmu je $\Theta(f(n))$.

Přejděme nyní k vlastnímu časovému odhadu. Čas potřebný pro výpočet jedné základní operace algoritmu bude záviset na tom, jak je tato operace komplikovaná. V přeloženém programu operace reprezentuje určitý počet strojových instrukcí a čím je operace komplikovanější, tím více strojových instrukcí v přeloženém programu zahrnuje a tím také delší bude

doba jejího výpočtu. Uvážíme-li, že taktovací kmitočet dnešních procesorů je v oblasti GHz, můžeme předpokládat, že se provede 10 milionů základních operací algoritmu za vteřinu (zopakujme, že zde mluvíme o operacích algoritmu, ne o strojových instrukcích procesoru). Kdybychom pro jednoduchost předpokládali, že procesor provede miliardu strojových instrukcí za vteřinu, znamenalo by to, že na provedení jedné základní operace algoritmu máme k dispozici 100 strojových instrukcí. Tento odhad je dostatečně konzervativní (opatrný) na to, aby reálně odpovídal všem běžným algoritmům. Přitom u mnoha algoritmů se dá očekávat, že počet provedených základních operací za vteřinu bude vyšší, než uvádí náš odhad. Následující tabulka tak ukazuje orientační horní odhady potřebného času pro různé časové složitosti a různé počty údajů. Časy jsou pro přehlednost zaokrouhleny (tab. 1).

Tab. 1

Složitost	Počty údajů					
	10	100	1 000	10 000	100 000	1 000 000
$\log_2(n)$	0.3 µs	0.7 µs	1 µs	1.3 µs	1.6 µs	2 µs
n	1 µs	10 µs	100 µs	1 ms	10 ms	100 ms
$n \log_2(n)$	3 µs	67 µs	1 ms	13 ms	166 ms	2 s
n^2	10 µs	1 ms	100 ms	10 s	17 min	28 hod.
2^n	102 µs	$4 \cdot 10^{15}$ roků	$3 \cdot 10^{286}$ roků			

Algoritmy, kterým odpovídají první čtyři časové složitosti v tabulce, řadíme do třídy algoritmů s polynomickou časovou složitostí. Třída těchto algoritmů zahrnuje všechny algoritmy, jejich časová složitost je přímo vyjádřena polynomem anebo pro jejich funkci časové složitosti existuje polynom, který ji shora ohraničuje. Například pro funkci $\log_2(n)$ je tímto polynomem lineární polynom, neboť platí

$$\log_2(n) \leq n \quad \text{pro} \quad n = 1, 2, \ldots$$

(přitom základ logaritmu není podstatný, protože hodnoty logaritmů o různých základech se liší jen multiplikativní konstantou). Algoritmy s polynomickou časovou složitostí označujeme jako algoritmy pracující v přijatelném čase, i když už u kvadratického polynomu je pro větší počty údajů čas potřebný pro výpočet dosti citelný.

Poslední složitost v tabulce je vyjádřena exponenciální funkcí (opět nezáleží na tom, jaký je základ, pokud je větší než jedna). Je zřejmé, že hodnota této funkce roste tak drasticky, že i pro poměrně malé počty údajů (např. stovky) je potřebný čas tak velký, že je nemožné takovými algoritmy provést výpočet. Čtenář zde může namítnout, že jsme tento závěr učinili na základě našeho odhadu časů, který je značně opatrný, tedy horní odhad. Nicméně je zřejmé, že i při méně opatrném odhadu bychom došli ke stejnému závěru.

Algoritmy, které mají exponenciální složitost, patří do třídy algoritmů s nepolynomiální časovou složitostí. Je to třída algoritmů, u kterých funkci jejich časové složitosti nelze shora ohraničit žádným polynomem. Plyne to ze známé limity

$$\lim_{n \to +\infty} \frac{a^x}{x^k} = +\infty$$

platné pro $a > 1$, a libovolné přirozené číslo k. I když algoritmy této třídy lze obecně považovat za nerealizovatelné v přijatelném čase, neplatí to v některých zvláštních případech, kdy jsou v úlohách jen nepříliš rozsáhlá data (např. desítky). U takových úloh lze algoritmy s exponenciální časovou složitostí použít (a používají se), i když se nejprve raději vždy snažíme o „lepší“ algoritmus, tj. s polynomickou časovou složitostí.

Existuje řada praktických úloh, jejich řešení vede k algoritmům této třídy. Například sem patří některé grafové úlohy. Nemožnost tyto praktické úlohy řešit algoritmy této třídy se v praxi řeší sestavením algoritmů s přijatelnou (polynomickou) časovou složitostí, které danou úlohu neřeší přesně, ale jen přibližně, čímž zpravidla dávají o něco horší výsledky, než by dal přesný algoritmus s nepolynomickou časovou složitostí. Tyto algoritmy, které nazýváme heuristické, bývají často založeny na nějaké intuitivní myšlence. Jak horší výsledek dávají, to se v mnoha případech nedá přímo odvodit. Zjišťuje se to statisticky.

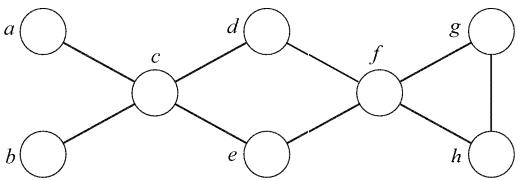
Příkladem úlohy z této třídy je nalezení minimální nezávislé dominující množiny uzlů grafu. Ta je podmnožinou uzlů grafu s vlastnostmi:

- Žádné dva její uzly nejsou sousední (nejsou spojeny hranou).
- Každý uzel grafu, který do ní nepatří, je sousední s některým uzlem v ní.
- Je minimální ve smyslu, že v grafu neexistuje podmnožina uzlů, která by splňovala předchozí dvě podmínky a obsahovala přitom méně uzlů.

Algoritmus, který tuto úlohu řeší, sestavuje jednotlivé podmnožiny uzlů grafu a mezi nimi hledá podmnožinu, která má vlastnosti nezávislé domi-

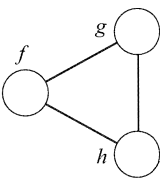
nující množiny a obsahuje přitom nejmenší počet uzlů. I když promyšleným postupem lze při tomto hledání řadu podmnožin uzlů, které zjevně nesplňují podmínky nezávislé dominující množiny, vynechat, přesto podstatnou část podmnožin uzlů grafu musí algoritmus projít. A uvážíme-li, že graf s n uzly má 2^n podmnožin uzlů, plyne z toho, že časová složitost algoritmu je nepolynomiální, neboť počet podmnožin exponenciálně roste s počtem uzlů.

Pro větší grafy se dá použít jen heuristický algoritmus. Ten minimální nezávislou dominující množinu sestavuje postupným přidáváním uzlů do ní. Po přidání každého uzlu do množiny následně z grafu odebere tento uzel a všechny jeho sousedy. Uvážíme-li, že čím větší je stupeň přidávaného uzlu, tím více uzlů se následně odebere, je zřejmé, že v každém kroku budeme vybírat uzel, který má ze všech stávajících uzlů největší stupeň. Je-li takových uzlů více, vybereme libovolný z nich. Jako příklad vezměme graf na obr. 1.



Obr. 1

V něm největší stupeň mají uzly c a f. Zvolíme třeba uzel c. Ten dáme do dominující množiny a odstraníme ho spolu s jeho sousedy z grafu (obr. 2).



Obr. 2

Nyní zbývají tři uzly, které mají stejný stupeň. Zvolíme z nich třeba uzel f. Přidáme ho do množiny a odstraníme tento uzel a jeho sousedy

z grafu. Po odstranění dostáváme prázdný graf, čímž je algoritmus ukončen. Výsledkem je nezávislá dominující množina se dvěma uzly c a f. V tomto případě heuristický algoritmus viditelně poskytl přesný výsledek.

U algoritmu sčítání n čísel, který jsme uvedli na začátku, počet provedených operací nezávisí na hodnotách čísel, jejichž součet počítáme. Vezměme nyní příklad algoritmu, kde tomu tak není. Máme dánu posloupnost prvků (například čísel)

$$a_1, a_2 \ldots, a_n$$

a máme dánu hodnotu x. Naším úkolem je najít v posloupnosti takový prvek a_i, jehož hodnota je rovna x. Budeme procházet posloupnost od začátku, postupně jednotlivé prvky v ní srovnávat se zadanou hodnotou x, až prvek s touto hodnotou najdeme anebo se dostaneme na konec posloupnosti a zjistíme, že takový prvek v posloupnosti není. Základní operací v tomto algoritmu je srovnání. Počet provedených srovnání bude v rozmezí od 1 do n. Jedno srovnání nám postačí, když hledaná hodnota bude hned na začátku posloupnosti. Naopak n srovnání budeme potřebovat v případech, kdy hledaná hodnota bude v posloupnosti až poslední anebo hledaná hodnota v ní vůbec nebude. U takových algoritmů můžeme uvažovat tři časové složitost: časovou složitost v nejhorším případě, časovou složitost v nejpříznivějším případě a časovou složitost v průměrném případě. U našeho algoritmu tyto časové složitosti budou

Tab. 2

nejhorší	n
nejpříznivější	1
průměrná	$\frac{n+1}{2}$

Průměrnou složitost zde vypočítáme jako průměr z počtu srovnání, kdy hledaný prvek se vyskytuje na prvním, druhém až n-tém místě:

$$\frac{1+2+\cdots+n}{n} = \frac{n(n+1)}{2n} = \frac{n+1}{2}.$$

Je zřejmé, že v průměrném a nejhorším případě je časová složitost vyjádřená lineární funkcí. Můžeme ji tedy napsat $\Theta(n)$.

Uveďme ještě komplexnější příklad; bude to odvození časové složitosti známého algoritmu bublinkového třídění, který je nejjednodušším třídícím algoritmem. Třídění probíhá postupným procházením tříděné posloupnosti a srovnáváním vždy dvou prvků, které bezprostředně následují za sebou. Pokud jsou v nesprávném pořadí (prvek s větší hodnotou je před prvkem s menší hodnotou), provede se jejich výměna. Základní operací je zde srovnání, další prováděnou operací je výměna. Výměn je ale nejvýše tolik, kolik je srovnání, čímž nám postačí uvažovat jen počet provedených srovnání. Při prvním průchodu procházíme celou tříděnou posloupnost, v níž máme celkem $n-1$ dvojic sousedních prvků, čímž provedeme $n-1$ srovnání. Na konci průchodu se největší prvek dostane na konec posloupnosti a je na svém cílovém místě. V dalším průchodu posloupností nám stačí už projít jen prvních $n-1$ prvků, což reprezentuje $n-2$ srovnání. Tímto průchodem se opět dostane největší prvek z procházené části na své cílové místo. Ve třetím průchodu už budeme procházet jen $n-2$ prvků, což reprezentuje $n-3$ srovnání. Když budeme tímto způsobem pokračovat, dostaneme se k poslednímu průchodu, ve kterém máme už jen dva prvky a provedeme 1 srovnání. Sečtením počtů srovnání ve všech průchodech dostaneme

$$(n-1)+(n-2)+(n-3)+\ldots+1=\frac{n(n-1)}{2}.$$

Výsledkem je kvadratická funkce. Nahlédneme-li do naší tabulky, zjistíme, že bublinkové třídění nám vyhovuje pro třídění orientačně tisíců údajů; záleží ovšem na tom, o jakou jde aplikaci a jaký je hardware. Poznamenejme zde, že bublinkové třídění jsme zde zvolili jako příklad pro jeho snadný popis, i když je obecně známo, že z jednoduchých metod třídění, které mají kvadratickou složitost, je nejpomalejší.

Pro třídění větších počtů údajů je účelné použít třídící algoritmus, který má lepší časovou složitost, než je kvadratická. Takovým je třídící algoritmus s názvem *Quicksort*, jehož typická časová složitost je $\Theta(n\ln(n))$. Není to přitom jediný třídící algoritmus s touto složitostí. Časovou složitost $\Theta(n\ln(n))$ má i třídění *haldou*. Třídění haldou je ale komplikovanější, čímž jeho implementace je pracnější a navíc čas třídění bývá přibližně dvakrát vyšší než čas třídění metodou *Quicksort*.

Na závěr doplňme, že kromě popsaného operátoru Θ, který vyjadřuje tzv. asymptotickou složitost, jsou zavedeny ještě operátory Ω a O, které označují dolní a horní asymptotickou hranici složitosti. Když se vrátíme k našemu příkladu, kde jsme odvodili, že časová složitost nalezení čísla

v posloupnosti je v rozmezí od 1 do n, můžeme pomocí těchto operátorů napsat, že časová složitosti je $\Omega(1)$ a $\mathrm{O}(n)$.

Význam použití notace však bohužel není v literatuře jednotný. Uveďme si jeho základní definici, která říká, že časová složitost $T(n)$ daného algoritmu je $\mathrm{O}(f(n))$, právě když existují kladné konstanty c a n_0 takové, že

$$\forall n \geq n_0 \quad \text{platí} \quad T(n) \leq c \cdot f(n),$$

nebo jinak, když

$$\lim_{n \to +\infty} \sup \frac{T(n)}{f(n)} < +\infty.$$

Ale pravděpodobně se s operátorem O spíše setkáme ve stejném významu asymptotické složitosti, jako má operátor Θ.

(Autorkou úvodní ilustrace je *Mgr. Jaroslava Čermáková* ze Suchých Lazců.)

Informační a komunikační technologie ve skupinové výuce

DITA MARYŠKOVÁ

Slovanské gymnázium, Tř. Jiřího z Poděbrad, Olomouc

Vývoj a využívání informační a komunikační technologie (IKT) ve světě i u nás rychle postupuje. Základní součástí IKT jsou počítače, dnes již snad neexistuje odvětví, kde by se počítače v nějaké podobě nevyskytovaly; pronikly také i do základních škol. Jedna věc je však mít počítače ve škole k dispozici, jinou věcí je jejich využití. Na mnohých školách se setkáváme s žáky, kteří mají znalosti lepší než vyučující, ale na druhou stranu i se žáky, pro které jsou počítače velkou neznámou. Naším úkolem je žáky s touto technikou seznámit a naučit je ji účelně využívat. Přitom podle mého názoru nestačí žáky učit pracovat s počítači pouze v hodinách informatiky a výpočetní techniky. Žáci musí vidět aktivní zapojení IKT do výuky a pro získávání informací i v dalších školních předmětech.

Při výuce matematiky jsou počítače běžně používány zejména dvěma způsoby:

a) *Počítač jako demonstrační nástroj.* Učitel si připraví část hodiny na počítači a žákům například zobrazí zadání slovní úlohy na plátně (přes datový projektor; v této funkci pracuje počítač jen jako meotar). Sem patří i využití prezentačního programu PowerPoint, např. lze ukázat některé složitější obrázky, přehledné tabulky, obrázky z historie apod., aniž by ztrácel čas kolováním těchto materiálů. Již toto využití má pro hodiny matematiky obrovský význam. Učitel sice stráví více času přípravou hodiny, ale v hodině samotné má volnější ruce a může se tak více věnovat žákům. Již jenom projití třídy a kontrola, že si žáci do sešitu zaznamenávají správné informace, je obrovským plusem pro žáky. Většinou právě u slabších žáků je často problém v chybném opisování a následně neporozumění dané problematice.

Na vyšší úrovni je využití k ukázkám konstrukcí v Cabri geometrii, neboť díky možnosti pohybu může učitel ukazovat měnící se situace při rozdílném zadání úlohy. Vrcholem technických možností současnosti je pak používání (a správné využívání) interaktivních tabulí v hodinách matematiky.

Navíc jakékoli obohacení hodiny způsobí zvětšení zájmu žáků, který je podporován jejich pocitem, že si hrají a ne že se učí, a tím se zvyšuje i efektivita výuky.

b) *Žák pracuje s počítačem.* Většinou sedí jeden žák u jednoho počítače a plní podle vybraného programu určitý úkol. Po zjištění správného výsledku pak pokračuje na další úkol dokud nesplní všechna zadání. K dispozici jsou výuková CD, případně některé matematické programy přímo na internetu. Některé školy si podobné úkoly zpracovávají samy zásluhou aktivních kolegů.

Kromě kladných efektů z využití techniky jednou z uvedených možností, vidím i negativní stránky věci.

Při využití počítače pouze učitelem zůstává i nadále zachována frontální výuka. Učitel přinese poznatky žákům „na stříbrném podnose“ a oni to často považují jen za „nudu“ převlečenou do počítače. Stále mají v hodině sedět a poslouchat, a ke slovu se dostanou jen velmi málo.

Když sedí jeden žák u jednoho počítače, „komunikuje“ pouze se strojem. Hlasité mluvení se opět nepřipouští. Matematické komunikační kompetence žáků se nejenom nerozvíjejí, ale naopak ještě více upadají. Dostat

dnes z žáků osmého ročníku základní školy celou srozumitelnou větu týkající se matematického učiva, je často nadlidský výkon. (Přitom o přestávkách se mnohým z nich pusa nezastaví.) V jejich slovním vyjadřování nevolí správné matematické (ani češtinářské) termíny a jsou na štíru i s logikou vyjádření.

Jedním z cílů moderního českého školství je rozvíjet také mluvený projev. A právě z důvodu rozvoje komunikačních dovedností a zatraktivnění výuky matematiky jsem zkusila v hodinách matematiky zapojit počítače do skupinové výuky. Zopakujme si:

Skupinové vyučování je organizační forma výuky, v níž žáci pracují ve skupinách vytvořených podle různých kritérií, např. obtížnosti úkolu, charakteru činnosti, výkonu nebo učebního tempa žáků. Vytváření skupin je uplatňováno především při problémové metodě ve výuce a při týmovém řešení učebních úloh [8].

Skupinová práce je hodnocena jako prostředek, který podporuje mezi-osobní vztahy. Jako prostředek řešení problémů vztahujících se k práci. Má však význam i v rodinném a manželském životě. Rodina sama je v ohrožení, odpovědnost se přesouvá na školu [9].

Skupinová výuka je organizační forma práce známá již ve starém Řecku. Historie ukazuje její větší či menší způsoby zapojení v celém svém průběhu. V některých obdobích je více uplatňována v USA, přibližně od začátku 20. století se dostává více ke slovu i v Evropě. Její podoby jsou různé. Někteří učitelé využívají skupinovou výuku výjimečně, jiní ji používají častěji. Všichni se však shodují na tom, že zapojením jiných forem výuky práci v hodině dynamizují a žáky motivují k větší aktivitě.

Nejčastěji se ve školních podmínkách setkáváme se skupinami vytvořenými náhodně – spojením žáků ze dvou lavic. Snad si ze školních let vzpomeneme na větu „otočte se k sobě“. Podle odborníků se doporučuje velikost skupiny mezi 3 až 5 členy. Ve větších skupinách přestává být aktuální komunikace a především ochabuje spolupráce mezi jednotlivými členy. Více vznikají menší skupiny uvnitř jedné větší a to už není v souladu s pravidly skupinové formy práce. Menší skupina je naopak dvojice nebo práce jednotlivce, ale to už se dostáváme do jiné formy výuky. Dále se doporučuje nejvíce 8 skupin v jedné třídě. To je totiž ještě počet, který stihne vyučující v průběhu práce obejít a zkontrolovat. Spojením obou požadavků dohromady je pochopitelné, že za teoreticky optimální velikost třídy považujeme 32 žáků, přičemž třídu rozdělíme na 8 skupin po 4 žácích (srovnej [10]).

Spojení obou metod jsem volila z několika důvodů:

a) *Rozvíjení komunikačních dovedností.* V době, kdy u jednoho počítače sedí a pracuje na stejném úkolu více žáků, je potřeba si práci rozdělit. Všichni nemohou současně sedět u klávesnice a každý psát své poznatky. Proto si musí dohodnout dělbu práce někteří členové skupiny pracují „na papíře", jiní u počítače a další člen třeba radí a pomáhá odstranit chyby. Přitom se navzájem radí. V případě nutnosti může některý z členů navštívit externí informační zdroj a poznatky vyhledat např. v knihách a učebnicích. Žáci se musí dohodnout, kdo bude mít jaký úkol. Musí umět zformulovat vlastní myšlenky a také odpovědi na dodatkové otázky. Když se někdo na něco zeptá a kolega mu nedokáže odpovědět, je nucen hledat jinou formulaci. Je to proces neustálého učení se – a nejen toho matematického.

b) *Rozvíjení sociálních kompetencí.* Práce ve skupině obnáší také další projevy společenského chování. Každý člen by měl umět ustoupit za svého názoru, umět přehodnotit vlastní myšlenky a dokázat dát za pravdu kolegovi. Žáci se učí reagovat na nápady spolužáků, učí se hledat argumenty i protiargumenty. V době, kdy mnohé podniky zakládají svou prestiž na práci týmů je pro žáky velkým přínosem, pokud se alespoň některým dovednostem dokáží naučit již v době školního vzdělávání.

c) *Je potřeba méně počítačů.* Pokud používáme pro každého žáka samostatný počítač, potom potřebujeme pro jednu gymnaziální třídu asi 32 počítačů. Jestliže však žáky poskládáme do skupin, stačí nám pouze 8 počítačů. Optimální tedy je tedy mít třídu vybavenou osmi počítači (připravenými např. v zadní části učebny). Lze samozřejmě využít již hotové počítačové učebny, kde je dostatek počítačů pro všechny studenty jedné třídy. Častým problémem však zůstává křížení hodin výpočetní techniky s matematikou, a tím nemožnost se v době matematiky do učebny dostat. Menší počet počítačů ve třídě je přínosem asi i po ekonomické stránce.

Pojednejme tedy nyní o skupinové výuce v hodinách matematiky. Poprvé práce začíná rozdělením žáků do pracovních skupin. Skupiny jsou trvalé, většinou alespoň po dobu pololetí. Ke spravedlivému rozdělení žáků do skupin lze zvolit například sociometrickou techniku (viz [1], [2]). V dalším kroku se musíme rozhodnout, v jaké fázi učiva techniku použijeme. Opět závisí pouze na konkrétním člověku – učiteli, jaký způsob využije.

Skupinovou výuku můžeme zařadit na začátku kapitoly jako *motivační faktor*. Lze ji využít *pro vlastní objevování nových poznatků* (To se využívá zejména v USA, kde je metoda tvořivého učení na mnohem vyšší úrovni než u nás – bohužel to ovšem přináší mnohem méně poznatků než mají naši žáci.) Osobně nejraději využívám skupinového vyučování ve spojení s IKT pro *upevnění a procvičení učiva*. V některých případech i „za odměnu" z dobře vykonané práce.

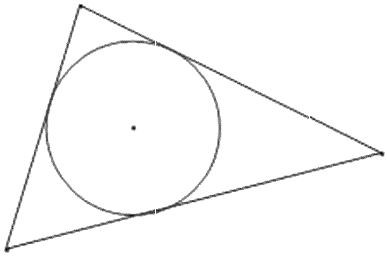
V každém případě je nutné při skupinovém vyučování počítat s poměrně dlouhým časovým úsekem pro práci skupin. Často je to otázka celé vyučovací hodiny, nebo ještě lépe spojením dvou hodin do jednoho celku. Každá hodina by měla obsahovat *úvod* (zadání práce, pravidla pro její úspěšné splnění), *vlastní práci, prezentaci a hodnocení práce*. Nejkratší částí je úvod, většinou nám na vysvětlení stačí asi 5 minut. Nejdelší je prezentace výsledků jednotlivých skupin. Zde potřebujeme alespoň 20 minut, často se to však protáhne. V žádném případě nelze vynechat hodnocení práce a jejích výsledků, protože očekávané hodnocení bývá často motivací chuti do práce. Je nutné ještě podotknout, že zpočátku, než si žáci na skupinové práce zvyknou, se při skupinové práci stihne jen málo; žáci si většinou stihnou pouze rozdělit úkoly. Postupem doby se časy zkracují, ale podmínkou jsou trvalé skupiny.

Příklady provedených skupinových prací

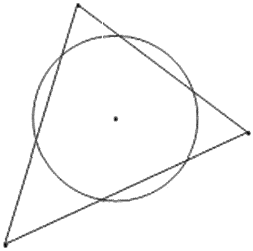
Příklad 1. Kružnice trojúhelníku vepsané a opsané.

Je to práce z kategorie „vlastní objevování nových poznatků". Již v šestém ročníku se probírají základní vlastnosti trojúhelníku. Kružnice trojúhelníku vepsané a opsané je jednou z učebních látek, kde se mi studovaná technika osvědčila. Žáci již znali své skupiny i popis trojúhelníku. Uměli sestrojit osy úseček, osy úhlů, těžnice i výšky v trojúhelníku. Před hodinou dostali za úkol se připravit na práci ve skupině (většinou to znamená jiný zasedací pořádek, často doplněný o přesun lavic).

Na začátku hodiny dostali žáci úkol: 4 skupiny měly hledat střed a poloměr kružnice trojúhelníku vepsané, zbývající skupiny hledaly střed a poloměr kružnice opsané. Kromě toho jsem využila počítač jako motivující nástroj. Na promítacím plátně jsem ukázala prostřednictvím Cabri geometrie správně sestrojenou kružnici vepsanou a též případ nesprávné polohy kružnice (pochopitelně tajemství konstrukce bylo skryté) – viz obr. 1a, 1b.



Obr. 1a Kružnice vepsaná – správně



Obr. 1b Kružnice vepsaná – špatně

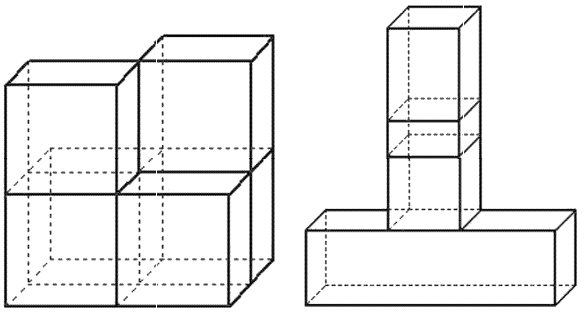
Žáci ve skupinách si úkol rozdělili. Někteří hledali správnou konstrukci a rýsovali na papír, jiní používali počítač s Cabri geometrií a někteří slabší žáci hledali pomoc v učebnici. Nakonec se všem podařilo úkol splnit. Problém nastal pouze v případě spuštění kolmic na příslušné strany a zjištění tak poloměru kružnice vepsané. Podobně postupovaly skupiny hledající konstrukci kružnice opsané.

Příklad 2. Povrchy a objemy těles

Šestý i sedmý ročník je věnován hranolům. Při probrání látky a vypsání základních vzorců pro povrch a objem jednotlivých těles jsem zařadila dvě skupinové práce. Bohužel jsou obě časově náročné a jedna vyučovací hodina je nedostačující. V některých případech bylo možné nechat připravit nebo dokončit úlohy doma.

Pro první práci si žáci měli donést z domu slepené (hranoly nebo) kvádry (libovolné, ale rozměry si v každé skupině předem dohodli). V hodině pak z vlastních kvádrů vytvořili jedno společné těleso (o cíli práce byli informováni před výrobou svých kvádrů). Jejich úkolem bylo narýsovat obraz vytvořeného tělesa ve volném rovnoběžném promítání (ručně nebo na počítači) a vypočítat jeho povrch a objem. Příklady výsledný obrázků viz na obr. 2.

Druhým úkolem bylo vytvořit tabulku, ve které jsou využity vzorce pro povrch a objem těles (obr. 3). Po splnění úkolu stačilo do buněk excelu s rozměry těles napsat čísla a „program“ sám vypočítal příslušnou hodnotu povrchu a objemu tělesa. Zde šlo více o procvičení počítačů, ale i o procvičení příslušných vzorců. Samotnou mě ovšem překvapilo, že největším problémem je právě znalost vzorců.



Obr. 2 Tělesa

Tělesa:								
	rozměry v cm						povrch v cm2	objem v cm3
krychle	a =	4,0					96,00	64,000
kvádr	a =	3,5	b =	4,0	c =	5,5	110,50	77,000
prav.trojboký hranol	a =	2,0	v =	4,2			31,35	7,274
prav.čtyřboký hranol	a =	12,3	v =	19,6			1266,90	2965,284
válec	r =	3,0	v =	5,4			158,34	152,681
koule	r =	2,8					98,52	91,952

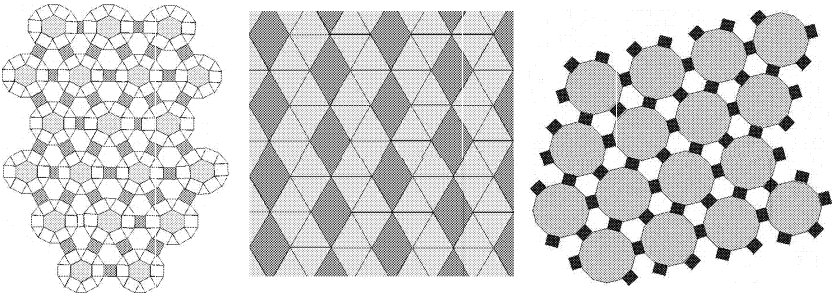
Obr. 3 Vzorce v Excelu

Příklad 3. Rovinné útvary – parketáž

Jiným neméně zajímavým úkolem bylo vytvořit vzory parket. Když jsme se setkali na začátku školního roku, bylo potřeba si připomenout vlastnosti rovinných obrazců. Kromě tradičního psaní a memorování vlastností, připomenutí vztahů mezi stranami, úhly, úhlopříčkami, zopakování si obvodů a obsahů obrazců, jsem zvolila historický prvek. Na mnoha hradech a zámcích v České republice se nacházejí na podlaze nejrůznější vzory parket. Základní vlastností je pravidlo, že parkety beze zbytku pokrývají celou podlahovou plochu. Navíc musí zůstat v rovině (v žádné místnosti nejsou „boule“).

Opět se našli jednotlivci, kteří raději než s pravítkem a kružítkem, pracovali pomocí počítače. Některé výsledky viz na obr. 4 (v originále jsou barevné). Pro skupiny brzy hotové jsem měla připravený doplňující úkol:

Kolik dlaždic od každého typu by museli zajistit, kdyby příslušným vzorem chtěli vyzdobit místnost obdélníkového tvaru s rozměry 20 m a 30 m. Tyto výpočty už všechny skupiny prováděly pouze jednoduchým výpočtem „na papíře“.



Obr. 4 Parkety

Příklad 4. Matematické domino

Bezesporu nejoblíbenější součástí matematické motivace se stalo matematické domino. Vychází z dětské hry, kdy místo bílých teček na hrací kostce máme v levém poli zadání příkladu na výpočet obsahu zde znázorněného obrazce a v pravém poli jiné kostky jeho řešení. Domino lze při troše času a chuti vytvořit téměř na jakoukoli učební látku. Často jej využíváme jako „odměnu“ za aktivitu a motivaci k další práci. Na druhou stranu si žáci jinou formou procvičí učivo, které za jiných okolností považují za obtížné pro zapamatování nebo nudné. Kromě toho při této hře si paměť žáků uchová mnohem více poznatků. Alespoň jeden příklad za všechny jsou dominové karty na obr. 5, vytvořené pomocí počítače.

$$1 - 3 - 6 - 5 - 12 - 8 - 2 - 4 - 10 - 7 - 9 - 11 - 1$$

Největší odměnou pro mě bylo, když mi žáci donesli svá vlastní domina. Přesvědčila jsem se, že „kdo si hraje, nezlobí“ a navíc se ještě mnohému naučí. Poprvé jsem se s dominem setkala v [3], některé další poznatky viz [4].

Spolu s rozvojem IKT škola stále hledá možnosti jejich využití. Spojení IKT se skupinovou výukou je jen jedna z mnoha cest.

a	ΔABC	polopřímka BA
b	$A \in p$	bod A náleží kružnici k
c	$A \in k$	trojúhelník ABC
d	AB	úsečka AB má stejnou velikost jako úsečka CD
e	$\|AB\| = 4$ cm	přímky p a q jsou rovnoběžné
f	$\leftarrow AB$	přímky p a q jsou různoběžné
g	$\leftrightarrow AB$	bod A náleží přímce p
h	$A \in k \cap p$	úsečka AB má délku 4 cm
i	$S = A - B$	bod A náleží rovině
j	$\|AB\| = \|CD\|$	přímka AB
k	$k(S;r)$	bod S je středem úsečky AB
l	$p \parallel q$	úsečka AB
m	$p \times q$	kružnice k se středem S a poloměrem r
n	$A \in \rho$	bod A leží na průniku přímky a kružnice

Obr. 5 Matematické domino

Literatura

[1] *Chráska, M.*: Základy výzkumu v pedagogice. VUP Olomouc, 1998.

[2] *Maryšková, D.*: Rozdělení žáků do skupin. In: Moderní vyučování, 2007, roč. XII, č. 2, s. 15 – 17.

[3] *Kruteckij, V. A.*: Psichologija matematičeskich sposobnostej školnikov. Moskva, Prosveščenije, 1968.

[4] *Maryšková, D.*: Matematické domino. In: Moderní vyučování, 2007, roč. XII, č. 8, zelené stránky 10 – 16.

[5] *Kuřina, F.*: Umění vidět v matematice. SPN, Praha 1989.

[6] *Mareš, J.* – *Křivohlavý, J.*: Sociální a pedagogická komunikace ve škole. SPN, Praha 1989.

[7] *Petty, G.*: Moderní vyučování. Portál, Praha 1996.

[8] *Průcha, J.* – *Walterová, E.* – *Mareš, J.*: Pedagogický slovník. Portál, Praha 2003.

[9] *Kasíková, H.*: Kooperativní učení, kooperativní škola. Portál, Praha 1997.

[10] *Mechlová, E.* – *Horák, F.*: Skupinové vyučování na základní a střední škole. SPN, Praha 1986.