TEACHING EFFICIENCY OF ALGORITHMS ON CZECH GRAMMAR SCHOOLS

Daniel LESSNER

Abstract: The contribution introduces partial results of experimental computer science education on Czech grammar schools (general secondary education, students are 15–19 years old). Informatics on our grammar schools aims on user skills and misses the scientific aspects, while computer science seems to be promising regarding main educational goals on grammar schools, mainly the key competence to solve problems. Our research is based on the idea of teaching computer science on par with traditional science subjects, such as physics or biology. We have developed an experimental course program and tested it on a group of grammar school students in Prague. In this contribution we describe the Efficiency chapter and its educational results. If the educational goals are set appropriately, even this abstract and demanding topic can be included into grammar school curriculum.

Key words: computer science, algorithm complexity, experimental teaching.

VÝUKA EFEKTIVITY ALGORITMŮ NA GYMNÁZIÍCH

Resumé: Příspěvek uvádí dílčí výsledky experimentální výuky informatiky na gymnáziích. Výuka informatiky na gymnáziích se zaměřuje především na uživatelské dovednosti a pomíjí vědní stránku informatiky. Přitom z hlediska cílů gymnaziálního vzdělávání se informatika jeví jako slibná oblast, především v souvislosti s klíčovou kompetencí k řešení problémů. Náš výzkum vychází z představy výuky informatiky jako vědecké disciplíny vedle ostatních tradičních vědních předmětů, jako jsou fyzika nebo biologie. Sestavili jsme odpovídající program výuky a otestovali ho na skupině žáků jednoho pražského gymnázia. V tomto příspěvku uvádíme podrobnější popis modulu zaměřeného na problematiku efektivity a výsledky jeho výuky. Ukazuje se, že při vhodném nastavení vzdělávacích cílů je možné i toto poměrně abstraktní a náročné téma do gymnaziální výuky úspěšně zařadit.

Klíčová slova: teoretická informatika, složitost algoritmů, experimentální výuka.

1 Úvod

Pojetí výuky související s počítači v Rámcovém vzdělávacím programu (RVP) pro gymnázia (1) ve vzdělávacím oboru *Informatika a informační a komunikační technologie* nešťastně směšuje uživatelské dovednosti s informatikou jako vědním oborem (2). Další nepříjemností, zvláště s ohledem na neukotvenost oboru, jsou nejasné či zavádějící formulace v popisu vzdělávací oblasti. Z hlediska tohoto příspěvku je poměrně palčivé například uvedené pojetí informatiky jako vědního oboru (RVP, str. 63): „*V rámci oblasti Informatika a ICT se žák seznámí se základy informatiky jako vědního oboru, který studuje výpočetní a informační procesy z hlediska používaného hardwaru i softwaru,*“ Další zvláštnosti vyniknou zejména při srovnání s anglickou verzí RVP (3).

Naznačené problémy se pochopitelně promítají i do výuky (tím spíš, že oblast není historicky ukotvena).

Východiskem našeho výzkumu je idea gymnaziální informatiky jako předmětu rovnocenného s již tradičními přírodními vědami, nikoliv s tělesnou výchovou. Smyslem její výuky je tedy seznámení se základními pojmy, typickými problémy, metodami jejich řešení, fundamentálními výsledky a v neposlední řadě s jejich dopady mimo obor.

Pohled ze zahraničí ukáže, že máme kde se inspirovat. Od konce devadesátých let funguje výuka informatiky na Izraelských středních školách (4, 5). Naopak čerstvý vývoj můžeme sledovat na Novém Zélandu (6) a ve Spojeném Království (7). Komplexní osnovy pro primární a sekundární všeobecné vzdělávání vypracoval tým ACM (8). Poměrně aktuální přehled o situaci v světě podávají reference v článku (9).

Cílem našeho výzkumu je zodpovědět otázky, které z uvedených idejí plynou, především tedy: Je to možné? A dále: Může to být k něčemu dobré? Abychom na tyto otázky mohli odpovědět, sestavili jsme program výuky úvodu

do informatiky v rozsahu dvou vyučovacích hodin týdně po dobu jednoho školního roku. „Počítače“ se při takové výuce neučí o nic více (ale ani o nic méně) než při výuce ostatních předmětů. Žáci se pochopitelně dozvědí leccos o tom, jak počítače fungují — podobně jako se dozvědí ve fyzice, jak funguje jaderná elektrárna nebo spalovací motor.

V první části příspěvku blíže představíme uvedené pojetí a uvedeme některé důsledky pro sestavování pilotního programu pro takový předmět. Už při volbě cílů se odlišíme od univerzitního pojetí oboru, ale částečně i od pojetí ostatních gymnaziálních předmětů. Chybějící tradici informatiky na středoškolské úrovni zde vnímáme jako výhodu. Jednotlivá probíraná témata krátce popíšeme.

Ve druhé části příspěvku blíže představíme výuku efektivitu algoritmů, která na gymnáziu představuje složitost – klasickou partii teoretické informatiky. Gymnazista by si patrně měl být vědom jejích hlavních výsledků. Při uzpůsobení učiva podmínkám gymnázií ale musí dojít k zásadním úpravám.

Motivací pro dané téma je potřeba porovnávat algoritmy mezi sebou a předvídat jejich použitelnost. Asymptotická složitost sama o sobě ale nemůže být hlavním nástrojem. Žákům chybí potřebný matematický aparát, navíc v gymnaziálním, tedy všeobecném kontextu jsou důležité i další souvislosti. V běžném životě záleží i na konstantách, při porovnávání algoritmů pak bereme v úvahu i mnoho dalších kritérií kromě spotřebovaného času. Dále, převoditelnost úlohy je příliš abstraktní koncept na to, aby bylo možné zavést pojem NP-úplnosti. Na druhou stranu, žák by měl vědět o existenci problémů, které jsou ve větším měřítku neřešitelné. V příspěvku popíšeme posuny ve výukových cílech a z nich vyplývající rozhodnutí o rozsahu učiva.

V závěru příspěvku uvedeme a zhodnotíme poznatky a zkušenosti s danou kapitolou získané při experimentální výuce informatiky podle vyvinutého plánu v minulém školním roce.

2 Program výuky

Smyslem celé naší práce je prozkoumat možnost a případné přínosy výuky informatiky na gymnáziích a poskytnout tak podklady pro případné budoucí diskuse o tomto tématu. Za tímto účelem jsme sestavili příslušný výukový program k otestování a zjištění potřebných výsledků. V této části program popíšeme a poskytneme tak kontext pro jádro příspěvku,

kterým je výuka efektivitu. Podrobnosti o plánu výuky a prvních výsledcích byly publikovány v (10).

Vzdělávací cíle naší výuky vycházejí z cílů gymnaziálního vzdělávání uvedených v RVP a z představy informatiky v postavení odpovídajícím ostatním přírodním vědám. Výuka má tedy jednak vést k rozvoji klíčových kompetencí (v našem případě nejvýrazněji kompetence k řešení problémů a komunikační) a jednak seznámit žáky s informatikou jako vědou.

Vazba na vzdělávací cíle českých gymnázií a komplexnost zpracování jsou zásadními důvody pro vytvoření vlastní koncepce. Podpůrné materiály a programy sice k dispozici jsou, ale zpravidla se zaměřují na nadané studenty a na určité dílčí téma, aniž by poskytl ucelený přehled o oboru.

S oblastí ICT se naše výuka prolíná stejně, jako by tomu mělo být v ostatních předmětech. Technika je zapojena jako přirozená součást světa: elektronická komunikace je považována za běžnou dovednost, stejně jako zpracování většího množství dat pomocí tabulkového procesoru. ICT používáme, kdykoliv je to v dané situaci účelné. Vzdělávacích cílů přímo souvisejících s ICT je ovšem poměrně málo (práce s editorem grafů, práce s nějakým programovacím prostředím apod.), což vyplývá z východisek naší práce.

Důležitou otázkou je zahrnutí programování. Je to obvyklá součást výuky volitelné „pokročilé“ informatiky, leckdy je obojí ztotožňováno. Programování jako takové (vývoj softwaru, popř. zápis algoritmů) ale na gymnáziu těžko obstojí v roli vzdělávacího cíle. Tímto cílem může být kultivace myšlení, rozvoj schopnosti systematicky analyzovat a řešit problémy, nikoliv znalost konkrétního programovacího jazyka. Na základě dostupné literatury, např. (11), a vlastní zkušenosti jsme usoudili, že výuka programování se vzhledem k našim hlavním cílům (rozvoj klíčových kompetencí a úvod do informatiky) nevyplatí. Ovládnout programování na dostatečné úrovni k tomu, aby bylo nápomocné pro dosahování těchto cílů, by zabralo neúměrně mnoho času. S programováním se žáci během výuky setkají, neklademe si ale za cíl je programování naučit.

Program výuky jsme samozřejmě sestavili proto, abychom ho mohli vyzkoušet v praxi. K tomu bylo třeba přihlídnout při návrhu jeho struktury. Rozhodli jsme se optimisticky předpokládat jeden školní rok výuky a dvě vyučovací hodiny týdně. Učivo jsme rozdělili

do tematických bloků, každý má trvat přibližně měsíc. Doba trvání se přizpůsobuje jednak organizaci školního roku, jednak doplňováním nutných předpokladů z jiných předmětů, především z matematiky. Žáci obvykle rychle zapomínají logiku a logaritmy, a naopak příliš pozdě probírají kombinatoriku a pravděpodobnost. I proto plánujeme modulů pouze osm¹.

Každý modul je výslovně zaměřen na určité zásadní téma informatiky. V pozadí se přitom samozřejmě objevují témata další. Zejména koncepty související s algoritmy a s efektivitou jsou postupně rozvíjeny v předcházejících modulech tak, aby žáci mohli pracovat s dostatečným počtem známých příkladů. Při výběru zaměření modulů jsme pracovali se základními idejemi podle (12) a s navazujícími koncepty v (13) a (14). Pro středoškolskou informatiku jsme vybrali čtyři ústřední pojmy: informace, problém, algoritmus a efektivita. Každému z nich je věnován samostatný modul. V ostatních modulech jsou dále rozvíjeny tyto i další pojmy (např. grafy, modelování, rekurse).

Nyní stručně popíšeme jednotlivé moduly a uvedeme jejich hlavní souvislosti s modulem Efektivita. Zde je na místě připomenout, že se jedná o výuku určenou všem studentům gymnázia, tedy nikoliv jen zájemcům, navštěvujícím např. výběrový seminář. Úroveň požadovaných znalostí a dovedností je tedy nutně nižší, než na těchto seminářích, na druhou stranu je nepoměrně vyšší, než současný standard. Pro nadané nebo motivované studenty se pochopitelně nabízí nepřeborné množství rozšiřující látky a zájmových aktivit.

Prvním je modul **Informace**. Žáci mají porozumět samotnému pojmu. K tomu slouží dva přístupy, o kterých se následně ukáže, že jsou skutečně jen jiným pohledem na tentýž jev. Prvním přístupem je chápání informace jako datové zprávy spolu s její interpretací. Druhým přístupem je informace jako úbytek možností² (15). Na tomto přístupu postavíme také měření množství informace. Na hře o hádání myšleného zvířete a potom myšleného čísla si žáci ujasní, které výroky a které otázky přinášejí nejvíce informace. Kromě toho se seznámí s konceptem

rozhodovacích stromů a jejich efektivitou v nejlepším, nejhorším a průměrném případě (a tím i s faktem, že spotřeba zdrojů je ovlivněna i konkrétním obsahem vstupu, nejen jeho velikostí). Na základě toho potom porovnávají a přibližně optimalizují stromy pro daný rozhodovací problém. Další podrobnosti a dílčí výsledky jsou publikovány v (16).

Následuje modul **Grafy** (17). Žáci sice s grafy a schématy v životě i ve škole poměrně běžně pracují, nejsou si ale vědomi žádné jednotící teorie v pozadí, ani výsledků, které tato teorie může poskytnout. V první řadě si tedy žáci na ukázkových úlohách uvědomí, že se mnohdy hodí využít přehledného nákresu situace, neboli modelování problému grafem. Dohodneme terminologii, abychom si rozuměli. Dalším krokem je ukázat, že graf, přestože se jeví jako od podstaty vizuální struktura, lze úspěšně reprezentovat znakově, např. pomocí matice sousednosti. Taková notace je jednak přístupná strojovému zpracování, jednak spolehlivější v případě velkých grafů.

Většina času je věnována typickému grafovému problému, kreslení jednotáček (eulerovských grafů). Motivací je optimální obchůzka strážného, resp. vhodný návrh cestíček v parku. Aspoň někteří žáci úspěšně odvodí příslušné věty, abychom následně zformulovali postup ověření nakreslitelnosti grafu jedním tahem. Tento postup je žákům předložen naprogramovaný v jazyce Python, žáci se seznámí se základy jazyka, prostředím, a zkusí si několik jednoduchých modifikací. Při formulaci postupu i při práci s programem si všimají, že si lze ušetřit práci (např. po zjištění třetího vrcholu s lichým stupněm lze rovnou vydat odpověď a skončit). Na konci modulu se žáci seznámí s pojmem isomorfismu.

Modul **Problémy** se zabývá obecnými strategiemi řešení problémů. Žáci mnohé z nich podvědomě tuší a intuitivně používají. V tomto modulu mají příležitost tyto strategie výslovně zformulovat. Pro život jsou důležité zásady „Proč je vůbec potřeba to řešit?“ a podobné, většina z nich ale směřuje především k precizní a systematické práci v souladu s (18): „Dá se problém rozdělit na menší části?“. Žáci si brzy uvědomí důležitost precizní formulace výchozího a cílového stavu a identifikace možností přechodů do dalších stavů. To vede přímo na koncept stavového prostoru a jeho systematického procházení, pokud možno tak, abychom co nejdříve došli k řešení. Žáci řeší ukázkové problémy a všimají si, jak a kdy

1 Je jich pro jistotu připraveno víc, ale nezdá se být reálné je stihnout.

2 Jde o speciální případ úbytku entropie, kdy je každá alternativa možná stejně, rozdělení pravděpodobnosti je tedy rovnoměrné. Díky tomu se lze bez pojmu pravděpodobnosti obejít.

aplikují jednotlivé heuristiky. Řešení konkrétních problémů je v tomto smyslu předstupněm tvorby algoritmů, při níž se uplatní velmi podobné zásady.

Čtvrtým modulem v pořadí je **Algoritmus**. Žáci už mají zkušenosti s pracovními postupy definovanými tak přesně, že je lze provádět takřka bezmyšlenkovitě. V tomto modulu definujeme vlastnosti, které musí algoritmický postup mít, a naučíme žáky základní možnosti jejich ověřování. V tomto ohledu jsou nejzákladnější determinismus a konečnost. Cílem je umět rozpoznat algoritmické postupy, resp. identifikovat a pokud možno odstranit místa, která algoritmickosti brání. Kromě toho by žáci měli rozumět důsledkům plynoucím z jednotlivých vlastností a umět posoudit výhody a nevýhody algoritmického přístupu. Na konci modulu se žáci seznámí s problémy, které algoritmické řešení z principu nemají. Ať už proto, že nemají řešení žádné, protože nemáme dostatek dat nebo dobré zadání (světový mír), protože nemáme dost času (porovnání velikosti dvou iracionálních čísel, tedy čísel s nekonečným desetinným rozvojem), nebo z hlubších důvodů, jako je odkaz sama na sebe (problém zastavení).

Smyslem zavedení pojmu algoritmu na gymnáziu je kromě jeho významu v informatice také význam především v pracovním životě při zadávání i provádění úkolů. Jisté úrovně algoritmickosti by měly dosahovat i např. zákony nebo lékařská diagnostika (19). I z tohoto důvodu věnujeme pozornost schopnosti přesně popsat algoritmus v přirozeném jazyce. Spolu s žáky odvozujeme potřebné zásady a doporučení, které mají vést jednak ke kvalitnějším popisům, jednak k usnadnění porozumění zdrojovým kódům programů. Obtížnost precizního vyjadřování v přirozeném jazyce žákům navíc pomáhá ocenit jazyky formální.

Následuje modul **Efektivita**, jehož podrobnější popis uvádíme níže zvlášť. Po efektivitě zopakujeme dosavadní znalosti a dovednosti v modulu **Topologické řazení**. Motivací je plánování vzájemně závislých pracovních úkonů. Argumentace při návrhu efektivního hledání potřebného uspořádání je obdobou argumentace u úlohy kreslení jednotáček. Přitom lze najít různé efektivní algoritmy a využít tak poznatky z předchozího modulu. Kromě samotného řazení řešíme také celkovou délku trvání projektu při možnosti paralelní práce metodou kritické cesty.

Modul **Evoluční algoritmy** ukazuje žákům pokročilou partii informatiky a zároveň vrhá nové světlo na dosud probranou látku: Nealgoritmický postup je užitečný, počítač produkuje něco, co jsme mu přímo neřekli, úloha s exponenciální časovou složitostí se zdá být řešitelná. Žáci pracují s předem připravenou kostrou programu pro hledání hamiltonovských kružnic (řešení problému obchodního cestujícího) a experimentují s parametry modelu. Řeší základní otázky (nejen) evolučních algoritmů: konvergenci, zachování nejlepších jedinců do příští generace, rovnováhu explorační a exploatační. Tento modul je zároveň přípravou pro modul poslední.

Modul **Humanitní souvislosti** staví na nabytých zkušenostech a prostřednictvím Turingova testu (20) stimuluje žáky k úvahám o důsledcích výsledků informatiky na život společnosti. Žáci neřeší informatické problémy tak jako v ostatních modulech, výuka se podobá spíše výuce filozofie. Žáci se seznamují s argumenty, formulují vlastní, diskutují a utvářejí své postoje. Pracují s materiály, které jim přibližují současnou špičku vývoje umělé inteligence a zároveň její pronikání do našeho každodenního života. Smyslem není jen intelektuální zábava a lepší porozumění limitům výpočetních procesů a charakteristice lidskosti, ale také uvážení volby kariéry.

3 Modul Efektivita

S alespoň hrubým přehledem o ostatních modulech můžeme přistoupit k popisu samotného modulu Efektivita. Tento modul vychází ze složitosti jako klasické oblasti informatiky, ovšem na úrovni, která je pochopitelná a prakticky použitelná pro studenty gymnázií. To znamená, že nelze jít do hloubky obvyklé na vysokoškolské úrovni (nebo na úrovni talentovaných studentů středních škol), a zároveň je třeba zahrnout i jiné pohledy, než ve složitosti převládající důraz na asymptotický čas. Souvisí to i s tím, že programování není ve středu zájmu naší výuky. Proto při nastavování cílů a při jejich ověřování nemá smysl přesně přebírat přístup z článku (21), nicméně v principu postupujeme obdobně.

Východiskem modulu je potřeba porovnávat algoritmy mezi sebou. Žáci už vědí, že pro jeden tentýž problém mohou najít více různých algoritmů, nabízí se tedy otázka, jak se mezi nimi rozhodnout. Dále, jestliže určíme nějaká měřítka kvality, je možno se ptát, jak jednotlivé algoritmy vylepšit.

Žáci si nejprve musí uvědomit, že podobně jako je správnost algoritmu spjatá se zadáním, je efektivita spjatá s kritériem, resp. s prostředkem či zdrojem, kterým se snažíme šetřit. Zároveň se s těmito zdroji seznámí. Už zde je patrné rozšíření oproti klasicky pojímané složitosti. Nezajímáme se jen o nejhorší časovou a prostorovou složitost jednotlivých algoritmů. Zajímá nás i čas průměrný, zajímá nás čas s ohledem na multiplikativní konstanty, zajímají nás ale i prostředky strávené vývojem a implementací optimálního postupu, cena použitých nástrojů atp.

Z klasické teorie složitosti si žáci mají odnést především koncept základního kroku a jeho opakování jako určujícího faktoru pro odhadování spotřebovaného času. S tím souvisí poznání, že význam efektivity stoupá s velikostí vstupů. Žáci se naučí hrubě odhadovat počet základních operací (např. podle počtu vnořených cyklů) a porovnávat výsledné výrazy mezi sebou, resp. rozlišovat logaritmické, jednotlivé polynomiální, a vyšší časové složitosti. Nezavádíme O-notaci, pracujeme s matematickou intuicí žáků a s jejich zkušeností z příslušných experimentů.

Nenašli jsme způsob, jak přijatelně zahrnout problematiku NP-úplných úloh, ale také jsme pro ni nenašli praktické využití. Podstatná je existence úloh, pro které neznáme rychlejší, než exponenciální algoritmy – což ty úlohy činí prakticky neřešitelnými (což má poněkud paradoxně i pozitivní důsledky, např. pro kryptografii). V této souvislosti se žáci seznamují s konceptem složitosti problému a jednoduchými ukázkovými odhady (např. hledání maxima seznamu).

Poslední skupina vzdělávacích cílů ohledně efektivity se váže k optimalizaci. Umíme-li říci, jak je co efektivní, mohli bychom to také umět učinit efektivnějším. Žáci poznají základní optimalizační heuristiky, jako např. „použijvej znovu, co už máš“ (jak mezivýsledky výpočtů, tak existující algoritmy), „předpřiprav si vstupy“, „skonči, jakmile máš, co podle zadání potřebuješ“, „měj po ruce, co je často třeba“, „plně využívej paralelní zdroje“.

Na začátku studenti vypracovávají sadu základních úloh. Zjišťujeme tím úroveň jejich znalostí jednak pro přizpůsobení výuky a výběr následujících úloh, jednak pro následné posouzení zlepšení. Jako příklad uveďme úkol porovnat efektivitu dvou způsobů výměny obsahu celočíselných proměnných a a b . Z odpovědí žáků (vč. zvolených kritérií

efektivit) usuzujeme na jejich pojetí efektivit a na místa, na která se máme zaměřit ve výuce.

Výměna obsahu proměnných prvním způsobem:

$$c = a$$

$$a = b$$

$$b = c$$

Výměna obsahu proměnných druhým způsobem:

$$a = a + b$$

$$b = a - b$$

$$a = a - b$$

Jako úvodní motivační příklad slouží algoritmus půlení intervalů, díky němuž si žáci mohou uvědomit, že zabývat se efektivitou může skutečně vést k nezanedbatelným úsporám. Samotný koncept půlení je jim znám už od prvního modulu, zde se k němu ale vrací a uvědomují si nové souvislosti a kvantitativní vztahy. Zároveň si uvědomují jeho obecnou použitelnost a mají být schopni myšlenku použít i v jiných kontextech, než hledání čísla v seřazeném seznamu. Ty, které nepřesvědčí výpočty, přesvědčí malá soutěž např. v hledání ve slovníku.

Žáci řeší cvičení vztahující se k jednotlivým výukovým cílům, podle své volby samostatně či skupinově. Kromě toho řeší rozsáhlejší skupinové úkoly. Jako příklad uvedeme soutěž, kdy mají co nejrychleji porovnat dvě skupiny záznamů. Jedna je vytištěná ve formě abecedně seřazeného seznamu, druhá je ve formě jednotlivých promíchaných lístků. Úkolem je najít lístky, které jsou navíc, hodnotí se počet položek přečtených ze seznamu. Optimalizace postupu spočívá v seřazení lístků podle abecedy, takže seznam je třeba projít nanejvýš jednou. Objevuje se zde idea atomické operace, zásada promyšlené přípravy zpracovávaných dat, a je také možnost zažít časový rozdíl různě efektivních přístupů. Konečným výstupem aktivity je žáky zformulovaný popis co nejeefektivnějšího postupu.

Další větší úlohy řeší žáci už bez simulace (pokud se pro ni sami nerozhodnou). Jde jak o klasické úlohy z programování, jako je hledání prvočísel, hádanky a rébusy vedoucí proti hladovému přístupu (např. smažení tří topinek na pánvi, kam se vejdou jen dvě, nebo známá rodinka s různě rychlými členy procházející tunelem po dvojicích), a úlohy související

s praktickou situací (co nejrychlejší nákup v samoobsluze podle seznamu, co nejrychlejší donáška poštovních dopisů). Zde stojí za zmínku ve vědách někdy opomíjená postojová stránka klíčových kompetencí. V informatice učíme mj. oceňovat elegantní řešení a právě efektivitu. Projevit by se to mělo např. tím, že se žáci raději předem zamyslí, případně odvedou nějakou práci „navíc“, aby si následně mnohem víc práce ušetřili. Lenost je v informatice pozitivní vlastnost.

Dále žáci prozkoumají rychlost růstu hodnot výrazů, které reprezentují počet operací algoritmů pro dané velikosti vstupů, a zformulují a ověří základní hypotézy. Zjistí tedy, že výraz s kvadratickým členem nakonec vždy porazí výraz s nejvýše lineárním členem atp., a zároveň uvidí, že pro malé vstupy situace není tak jednoznačná. Výrazy s exponenciálním členem nemá pro větší vstupy ani smysl porovnávat graficky, což už mnohým žákům dostatečně napoví, že takové algoritmy v praxi nelze použít. Protože se nelze spoléhat na porozumění chování logaritmické (a exponenciální) funkce, pracujeme v jejím případě s přiblížením „jeden krok navíc umožní zpracovat dvakrát větší vstup“, což dostatečně přibližuje obrovskou úsporu.

Studenti jsou na konci každého modulu hodnoceni. V případě efektivity vypracovávají sadu úloh, které ukazují, nakolik si osvojili vytyčené cíle. Na vypracování mají týden, práci zahajují už v hodině, abychom na místě vyřešili případné dotazy. Práci odevzdávají na papíře nebo elektronickou poštou. Výsledná známka se počítá na základě celkového bodového zisku za jednotlivé úlohy. Po odevzdání studenti diskutují své výsledky a přístupy a seznamují se se správnými odpověďmi i nejčastějšími chybami.

Prověrka obsahuje bonusové úlohy, které nepřispívají k požadovanému bodovému základu. To poněkud snižuje obtížnost, resp. zlepšuje výsledné známky. Látka je sama o sobě poměrně obtížná, v prověrce je navíc minimum typových úloh. Je třeba uplatňovat známé principy, ale v nových situacích. To je pro studenty poměrně nezvyklý a náročný požadavek. Ne vždy se jim povede použít správnou myšlenku na správný problém. Nechceme se ale vzdát hodnocení této dovednosti, proto jsme jako cestu zvolili přidat úlohy a zvýšit tak šanci studentů na úspěch.

Jako příklad úlohy uvedeme úlohu, ve které mají žáci porovnat dvě funkce napsané v Pythonu řešící stejnou úlohu a popsat, v čem je která lepší a proč.

Obě funkce dostanou jako parametr dva vzestupně seřazené seznamy unikátních čísel A a B a vrátí seznam s čísly obsaženými v obou vstupních seznamech zároveň³. Zdrojový kód v prověrce záměrně neobsahuje vysvětlující komentáře.

```
def prunik_prvni(A, B) :
    vystup = []
    i, j = 0, 0
    while i<len(A) and j<len(B) :
        if A[i] == B[j] :
            vystup.append(A[i])
            i += 1
            j += 1
        elif A[i] < B[j] :
            i += 1
        else :
            j += 1
    return vystup

def prunik_druhy(A, B) :
    vystup = []
    for prvek in A :
        for prvek2 in B :
            if prvek == prvek2 :
                vystup.append(prvek)
    return vystup
```

Žáci si mohou všimnout například faktu, že první funkce vyžaduje jediný průchod oběma seznamy, kdežto druhá dosahuje kvadratické časové složitosti. Druhá funkce na druhou stranu nepotřebuje mít seznamy seřazené a snáze se implementuje (navíc s menším rizikem chyby),

Jinou úlohou ze závěrečné prověrky je hledání rozbité antény:

Televizní anténa na střeše panelového domu přestala správně fungovat. Potřebuje spravit, ale nevíš, která to je, na střeše jsou i antény sousedů. Postupně odpojuješ anténu za anténou a chodíš dolů dívat, jestli jsi odpojil svou, nebo ne. Jsi na to sám.

1. Vymysli a přesně popiš postup odpojování tak, aby ses nachodil (televize-střecha) výrazně méně.

2. Přibližně odhadni závislost množství chození a počtu antén v příznivém, průměrném a nepříznivém případě u tvého postupu i u naivního postupu (ze zadání).

³ Zkušenosti ukazují, že na porozumění množinové terminologii se nelze spoléhat

Žáci využívají dříve získanou dovednost přesně popsat pracovní postup. Co se týče efektivity, očekáváme použití principu půlení. Bonusová úloha vedla k nalezení dalších kritérií, která bychom mohli chtít plnit, například různé realizovaný ohled na sousedy (minimalizace počtu či délky odpojení antén).

Prověrka obsahovala 6 úloh, z nichž některé obsahovaly dílčí a bonusové podúlohy. Body tedy bylo možno získat celkem za 11 položek, z nichž 4 byly bonusové. Za základní úlohy bylo možno získat 48 bodů, za bonusové dalších 20 bodů.

4 Průběh a výsledky výuky

Získat ke spolupráci školy a jejich studenty se ukázalo jako poměrně obtížné. Bylo třeba najít školy s vedením, které by bylo myšlenke našeho výzkumu dostatečně nakloněno, nebo se o dění ve výuce dostatečně nezajímalo. Nakonec jsme získali možnost vyučovat povinně volitelný seminář na jednom pražském gymnáziu. Podmínky výuky tedy neodpovídaly našemu záměru a výsledky tedy nevypovídají o možnosti vyučovat informatiku na gymnáziích obecně. Přesto ale, s přihlédnutím k charakteristice účastníků semináře, jsou získané výsledky užitečné⁴.

Na seminář docházeli společně čtyři studenti maturitního ročníku a osm studentů o ročník níže. Maturanti ale pracovali většinu času zvlášť se svým učitelem a připravovali se na zkoušku, proto jsme je do výzkumu nemohli zahrnout. Další komentář se tedy týká ostatních osmi.

Motivace k účasti na semináři většiny z nich spočívala ve volbě nejmenšího zla z předložené nabídky povinně volitelných seminářů. Někteří žáci měli různě bohaté zkušenosti s programováním (PHP, Pascal, C#...), většina z nich ale neměla žádné předchozí znalosti. Žádný ze studentů neměl k matematice pozitivní postoj, dva studenti měli postoj negativní. Tyto informace vyplynuly z osobních rozhovorů během úvodních hodin.

V různých modulech jsme museli doplňovat látku z jiných předmětů (zejm. z matematiky), a program výuky jsme přizpůsobovali původnímu učebnímu plánu dané školy. Proto jsme se k modulu Efektivita dostali až v květnu. Následující modul (opakování na úloze topologického řazení) jsme přeskočili a ve zbytku

času stihli zkrátce probrat evoluční algoritmy a Turingův test.

Výsledky prověrky byly překvapivě pozitivní. Úspěšnost vyjádříme jako poměr bodů získaných a získatelných (bez bonusů) v procentech zaokrouhlených na jednotky. Úspěšnost se pohybovala mezi 50 a 100 procenty. Medián činil 82 %, průměr 79 %. Bodový zisk přitom nebyl bonusovými úlohami ovlivněn tak výrazně, jak jsme předpokládali. Kdybychom je vůbec nezapočetli, klesla by nejvyšší dosažená úspěšnost na 92 %, medián na 74 % a průměr na 71 %.

Okomentujeme nyní podrobněji výsledky dvou úloh uvedených v předchozí části.

V úloze o průniku seznamů všichni žáci až na jednoho správně poznali, že první způsob, byť s delším kódem, je při velkých vstupech časově úspornější. Průměrná úspěšnost dosáhla 80 %. Většina žáků uvedla i další aspekty porovnání obou programů.

V úloze s hledáním antény získali všichni žáci mezi 80 a 100 % bodů. Přitom principu půlení využili všichni, stržené body byly za formulační nedostatky a funkční opominutí (nedostatečný popis půlení při lichém počtu antén apod.).

Doplňující úlohu, ve které měli žáci vyčíslit nebo aspoň odhadnout, nakolik je jejich postup lepší než metoda pokus-omyl, nikdo nezvládl úplně. Každý ovšem dostal alespoň třetinu bodů, což značí, že se odhodlal k nějakému odhadu a ten nebyl zcela nesmyslný. Průměrná úspěšnost doplňující úlohy byla 56 %.

Během výuky modulu i v prověrce jsme sledovali, jestli se objeví nesprávné prekoncepty pozorované v článku (22):

- kratší zdrojový kód znamená kratší dobu běhu,
- méně proměnných znamená kratší dobu běhu,
- programy se stejnými příkazy v různém pořadí jsou stejně efektivní,
- programy řešící stejnou úlohu jsou stejně efektivní.

Narazili jsme na ně jen v několika náznacích, žáci se pokaždé sami opravili. Přičítáme to obecnějšímu pojetí výuky (výuka v odkazovaném článku cílí výrazněji na programování) a především faktu, že jsme o nesprávných prekonceptech věděli předem. Lze tedy předpokládat, že jsme je odstranili během výuky mimovolně.

⁴ V tomto školním roce už máme víc štěstí a vyučujeme jednu kompletní kvintu na jiné škole.

Na základě vyhodnocení prověrky a poznámek během výuky usuzujeme, že výuka základů efektivit algoritmů na gymnáziu je možná a má smysl. Žákům sice chybí užitečný matematický aparát (často i ten, který by podle učebních plánů chybět neměl), jsou ovšem schopni si osvojit znalosti a dovednosti uzpůsobené odpovídající úrovni.

5 Závěr

V příspěvku jsme představili a popsali jedno z možných pojetí výuky vědní informatiky na gymnáziích. Uvedli jsme základní východiska, cíle výuky a stručný obsah jednotlivých tematických modulů. Dále jsme se soustředili na modul věnovaný efektivitě algoritmů. Ten vychází z algoritmické složitosti, přizpůsobené úrovni gymnaziálních studentů a rozšířené o některé souvislosti. Důraz neklademe na asymptotickou složitost, ale na porovnávání efektivit pracovních postupů z různých úhlů pohledu.

Na základě průběhu a výsledků experimentální výuky jsme ukázali, že téma není tak nedosažitelné, jak se na první pohled může zdát.

Navazující práce spočívá v úpravě programu výuky na základě poznatků získaných z prvního běhu v loňském roce a v širším vyzkoušení v reálné výuce. Výsledkem bude ověřený program výuky publikovaný pro učitelskou veřejnost. Kromě toho bychom během pilotáže rádi zkoumali přínos výuky informatiky pro rozvoj klíčových kompetencí.

5 Literatura

- [1] Rámcový vzdělávací program pro gymnázia (online). Praha: Výzkumný ústav pedagogický v Praze, 2007 [cit. 9. prosince 2012]. ISBN 978-80-87000-11-3. URL : http://www.vuppraha.cz/wp-content/uploads/2009/12/RVPG-2007-07_final.pdf
- [2] NEUMAJER, Ondřej. Proč a jak inovovat pojetí ICT v rámcových vzdělávacích programech. *Metodický portál: Články* (online). 2009, [cit. 12. December 2012]. URL : <http://clanky.rvp.cz/clanek/o/z/2989/PROC-A-JAK-INOVOVAT-POJETI-ICT-V-RAMCOVYCH-VZDELAVACICH-PROGRAMECH.html>
- [3] Framework Education Programme for Secondary General Education (Grammar Schools). Praha: Výzkumný ústav pedagogický v Praze, 2007.
- [4] HABIBALLA, Hashim, FOJTÍK, Rostislav, VOLNÁ, Eva a TELNAROVÁ, Zdenka. Výuka informatiky na středních školách v České republice. In: *ISKI 2007*. Nitra: s.n., 2007, s. 50–57.
- [5] GAL-EZER, Judith a HAREL, David. Curriculum and Course Syllabi for a High-School Program in Computer Science. *Computer Science Education*. 1999, roč. 9, s. 114–147.
- [6] BELL, Tim, ANDREAE, Peter a LAMBERT, Lynn. Computer science in New Zealand high schools. In: *Proceedings of the Twelfth Australasian Conference on Computing Education*. S.l.: s.n., 2010, s. 15–22.
- [7] CRICK, Tom a SENTANCE, Sue. Computing at school: stimulating computing education in the UK. In: *Proceedings of the 11th Koli Calling International Conference on Computing Education Research* (online). New York, NY, USA: ACM, 2011, s. 122–123. ISBN 978-1-4503-1052-9. URL : <http://doi.acm.org/10.1145/2094131.2094158>.
- [8] TUCKER, Allen, DEEK, Fadi, JONES, Jill, MCCOWAN, Dennis, STEPHENSON, Chris a VERNON, Anita. A Model Curriculum for K-12 Computer Science: Final Report of the ACM K-12 Task Force Curriculum Committee. Second Edition. New York: Computer Science Teachers Association, 2003
- [9] HUBWIESER, Peter a kol. Computer science/informatics in secondary education. Proceedings of the 16th annual conference reports on Innovation and technology in computer science education - working group reports - ITiCSE-WGR '11 (online). 2011, s. 19–38. URL : <http://dl.acm.org/citation.cfm?doid=2078856.2078859>.
- [10] LESSNER, Daniel. Introducing Computer Science into Czech Grammar Schools: First Results. In: *EDULEARN12 Proceedings*. IATED, Barcelona, 2012, s. 246–255.
- [11] BELL, Tim, CURZON, Paul, CUTTS, Quintin, DAGIENE, Valentina a HABERMAN, Bruria. Overcoming Obstacles to CS Education by Using Non-programming Outreach Programmes. In: Ivan KALAŠ and Roland T. MITTERMEIR, eds. *Informatics in Schools. Contributing to 21st Century Education (ISSEP 2011)* (online). S.l.: Springer Berlin Heidelberg, 2011, s. 71–81. [cit. 6. December 2012]. ISBN 978-3-642-24721-7. URL : <http://www.springerlink.com/index/45026521606Q8037.pdf>.

- [12] BRUNER, Jerome S. The process of education. S.l.: Harvard University Press, 1977.
- [13] SCHWILL, Andreas. Fundamental Ideas: Rethinking Computer Science Education. *Learning & Leading with Technology*. 1997, roč. 25, č. 1, s. 28–31.
- [14] PASTERNAK, Arno a VAHRENHOLD, Jan. Design and evaluation of a braided teaching course in sixth grade computer science education. In: *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (online). New York, NY, USA: ACM, 2012, s. 45–50. ISBN 978-1-4503-1098-7. URL : <http://doi.acm.org/10.1145/2157136.2157154>.
- [15] HARTLEY, Ralph V. L. Transmission of information. *Bell System techn. Journal*. 1928, roč. 7, s. 535–563.
- [16] LESSNER, Daniel. Information Theory on Czech Grammar Schools: First Findings. In: KNOBELSDORF, Maria, ROMEIKE, Ralf, eds. *Pre-proceedings of the 7th Workshop in Primary and Secondary Computing Education (WiPSCE)*. Hamburg, 2012, s. 139–142.
- [17] MATOUŠEK, Jiří a NEŠETŘIL, Jaroslav. Kapitoly z diskrétní matematiky. Praha: Karolinum, 2009. ISBN 9788024617404.
- [18] POLYA, George. How to solve it: A new aspect of mathematical method. 2. S.l.: Princeton University Press, 1957.
- [19] LESSNER, Daniel. Proč se vlastně na gymnáziu učit o algoritmech? In: *Počítač ve škole 2012: sborník příspěvků*. Nové Město na Moravě, 2012, s. 5–7.
- [20] TURING, Alan M. Computing machinery and intelligence. *Mind*. 1950, roč. 59, č. 236, s. 433–460.
- [21] GAL-EZER, Judith a ZUR, Ela. The Concept of “Algorithm Efficiency” in the High School CS curriculum. In: *Proceedings of the 32nd ASEE/IEEE Frontiers in Education Conference* (online). Boston, 2002, s. 2–7. [cit. 9. prosince 2012]. ISBN 0780374444. URL : <http://xlmjfte.epinnovations.com/fie2002/papers/1145.pdf>.
- [22] GAL-EZER, Judith a ZUR, Ela. The efficiency of algorithms—misconceptions. *Computers & Education* (online). Duben 2004, roč. 42, č. 3, s. 215–226. [cit. 9. prosince 2012]. URL : <http://linkinghub.elsevier.com/retrieve/pii/S0360131503000848>.
- Mgr. Daniel Lessner**
Kabinet software a výuky informatiky
Matematicko-fyzikální fakulta UK
Malostranské nám. 25
118 00, Praha 1, ČR
E-mail: lessner@ksvi.mff.cuni.cz
Www: ksvi.mff.cuni.cz/~lessner