Algoritmus

Martin Lukeš
ČVUT FEL, STM, 1.ročník
lukesma4@fel.cvut.cz

Abstract: Tento článek by měl čtenáři přiblížit klíčová slova algoritmus a algoritmická složitost. Dalším z úkolů, které si článek vytyčil je ukázat základní vlastnosti algoritmů, nejdůležitější druhy a předvést jednoduchý příklad.

1. Něco málo z historie

Slovu algoritmus propůjčil své jméno perský matematik a astronom Abu Ja'far Muhammad ibn Musa al-Khwarizmi, mezi jehož díla patří i *Al-Khwarizmi on the Hindu Art of Reckoning* (latinsky *Algoritmi de numero Indorum*) pojednávající o arabských číslicích. Toto dílo je zodpovědné za vznik slova algoritmus. Nejvýznamější Al-Khwarizmiho dílo *Hisab al-jabr w'al-muqabala* položilo základy algebře, jak ji známe dnes a dalo podnět k vzniku samotného slova algebra. Jak již bylo řečeno Al-Khwarizmi nebyl jenom matematik, ale také astronom, takže se nemůžeme divit, že byl podle něj pojmenován měsíční kráter - Crater Al-Khwarizmi.

2. Algoritmus

Algoritmus je přesný návod či postup, kterým lze vyřešit daný problém. Můžeme se na něj dívat jako na soupis jednotlivých pokynů, instrukcí a souborů akcí, které nás zavedou k námi určenému cíli. Pojem algoritmu se nejčastěji objevuje v oblasti programování, kdy se jím myslí teoretický princip řešení problému (oproti přesnému zápisu v konkrétním programovacím jazyce). Obecně se ale algoritmus může objevit v jakémkoli jiném vědeckém odvětví a vlastně i v odvětvích nevědeckých - jako jistý druh algoritmu se může chápat i např. kuchyňský recept, neboť ten splňuje výše popsané požadavky. Je to soupis instrukcí, jež nás dovedou k cíli, kterým je uvaření daného pokrmu.

3. Druhy algoritmů

Algoritmy můžeme klasifikovat různými způsoby. Mezi nejdůležitější druhy algoritmů patří:

- Rekurzivní algoritmy, které využívají (volají) samy sebe. Příkladem tohoto typu algoritmu je algoritmus pro výpočet faktoriálu.
- Hladové algoritmy se k řešení propracovávají po jednotlivých rozhodnutích, která, jakmile jsou jednou učiněna, už nejsou dále revidována. Mezi zástupce těchto algoritmů patří Jarníkův a Borůvkův algoritmus sloužící k nalezení minimální kostry grafu.
- Algoritmy typu rozděl a panuj dělí problém na menší podproblémy, na něž se rekurzivně aplikují (až po triviální podproblémy, které lze vyřešit přímo), po čemž se dílčí řešení vhodným způsobem sloučí.
- Algoritmy dynamického programování pracují tak, že postupně řeší části problému od nejjednodušších po složitější s tím, že využívají výsledky již vyřešených jednodušších podproblémů. Mnoho úloh se řeší převedením na grafovou úlohu a aplikací příslušného

grafového algoritmu.

- Pravděpodobnostní (někdy též probabilistické) provádějí některá rozhodnutí náhodně či pseudonáhodně. Zástupcem tohoto algoritmu je metoda Monte Carlo, jejímž použitím můžeme zjistit přibližnou hodnotu čísla π.
- Genetické algoritmy pracují na základě napodobování biologických evolučních procesů, postupným „pěstováním“ nejlepších řešení pomocí mutací a křížení. V genetickém programování se tento postup aplikuje přímo na algoritmy (resp. Programy), které jsou zde chápány jako možná řešení daného problému.

Přičemž jeden algoritmus může patřit zároveň do více skupin, například může být zároveň rekurzivní a typu rozděl a panuj.

Problematikou efektivity algoritmů, tzn. metodami, jak z několika známých algoritmů řešících konkrétní problém vybrat ten nejlepší, se zabývají odvětví informatiky nazývané algoritmická analýza a teorie složitosti.

4. Vlastnosti algoritmů

Základními vlastnostmi algoritmů jsou konečnost (finitnost), obecnost (hromadnost, univerzálnost), determinovanost (jednoznačnost), resultativnost neboli výsledné řešení, které nám algoritmus poskytne. Algoritmy jsou plné protikladů – jsou jednoduché a zároveň složité. Je jednoduché je pochopit, ale většinu kvalitních algoritmů nemůže vymyslet valná většina z nás za celý život. Ty nejjednodušší a „nejhloupější“ algoritmy vymyslí každý. To je také jedním z důvodů, proč byla vymyšlena algoritmická složitost. Nicméně, ať už vymyslíme jakkoliv špatný algoritmus, je možné vymyslet algoritmus, který je ještě horší (slova RNDr. Rudolfa Kryla – přednáška předmětu Programování MFF 2006).

Tabulka 1. Vlastnosti algoritmů

Vlastnost	Popis vlastnosti
konečnost (finitnost)	Algoritmus má konečný počet kroků pro daná vstupní data, pro jiná data se může tento počet lišit, ale musí být **konečný.** Nekonečný algoritmus by vyvracel slova RNDr. Kryla, že musí existovat horší algoritmus a také by odporoval základní vlastnosti dojít k řešení.
obecnost (hromadnost, univerzálnost)	Algoritmus musí mít měnitelná vstupní data, měl by řešit obecný problém, algoritmus k výpočtu příkladu „1+1“ není účelný.
determinovanost (jednoznačnost)	Algoritmus musí mít jednoznačně danou posloupnost navazujících kroků. Pro stejná vstupní data by nám algoritmus měl dávat stejné výsledky.
resultativnost	Algoritmus má alespoň jeden výstup – 1. veličinu, která je v požadovaném vztahu k zadaným vstupům, a tím tvoří odpověď na problém, který algoritmus řeší. Další možností je, že algoritmus řešení nenajde, ale i tato informace se dá považovat za výsledek.

5. Složitost algoritmů

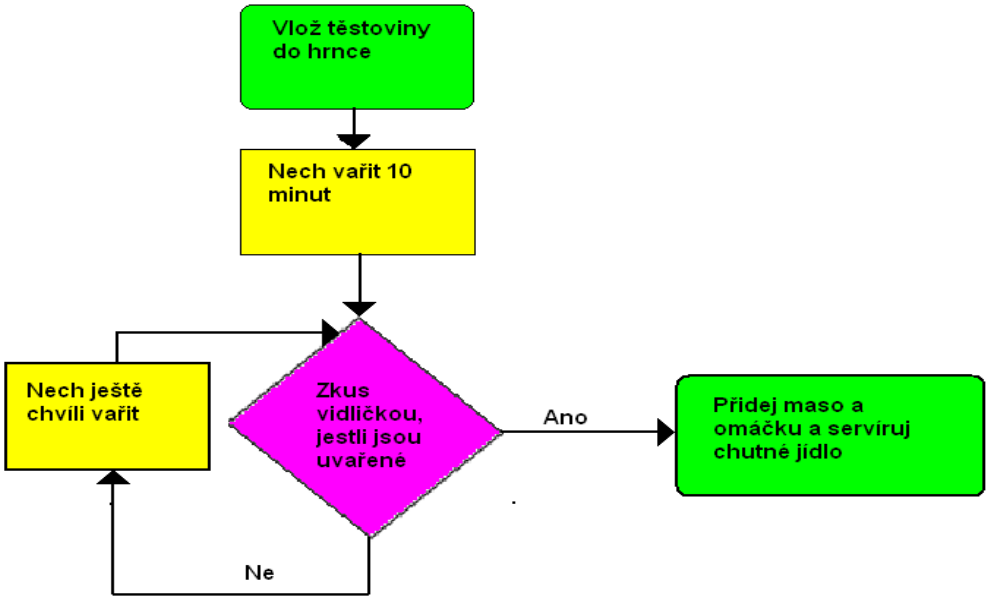
K porovnávání efektivity jednotlivých algoritmů nám pomáhá algoritmická složitost. Konečnost - jedna z vlastností algoritmů nám zaručuje, že algoritmus nám skončí, ale nezaručuje nám, že se tohoto konce dožijeme. Algoritmická složitost počítá kroky a můžeme podle ní třídí algoritmy podle efektivity. Závisí na velikosti vstupních dat, proto ji můžeme popsat jako funkci T(n), kde číslo n udává velikost vstupních dat. Například $T(n) = an + b$, kde a, b jsou nějaké konstanty, je zápis lineární časové složitosti (složitost roste lineárně s rostoucí velikostí vstupů).

Obvykle je pro odhad složitosti důležitý pouze typ funkční závislosti a nikoliv přesné hodnoty konstant - ty se zanedbávají. Při značné velikosti vstupních dat (to je situace, která nás vzhledem k složitosti algoritmu zajímá, při malých vstupech je složitost všech algoritmů poměrně nízká a vyrovnaná) můžeme konstanty a (aditivní konstanta) a b (multiplikativní konstanta) zanedbat, protože vzhledem k velikosti n jsou nepatrné, a výsledná složitost T(n) tak závisí především na velikosti n - velikosti vstupních dat algoritmu. Tuto skutečnost značíme $T = O(n)$, čímž vyjadřujeme asymptotické chování funkce T. O(n) nám značí asymptotickou složitost algoritmu. Nejefektivnější algoritmy mají logaritmickou asymptotickou složitost (např. log (n)), naopak nejméně efektivní algoritmy mají složitost exponenciální (např 2^n nebo vyšší n!).

Za efektivní algoritmy považujeme algoritmy s asymptotickou složitostí polynomiální (např. N^{127}). Provádění exponenciálních algoritmů či dokonce algoritmů o složitosti $T(n) = n!$ může už jen při malém navýšení velikosti vstupních dat trvat i mnoho tisíců a miliónů let. Čím je algoritmus složitější, tím menší je vliv navýšení rychlosti provádění jednotlivých kroků algoritmu při velké velikosti vstupu. Ani polynomiální algoritmy s velkým stupněm polynomu nebo s velkými vstupními daty nejsou příliš rychlé.

6. Příklad

Máme tu příklad, kdy máme uvařit k hlavnímu jídlu těstoviny a náš pomocník zatím připraví maso a omáčku. K znázornění používáme diagramy. Šipky nám určují směr, kterým se budeme ubírat, zelená políčka konec a počátek algoritmu, žlutá políčka jednotlivé kroky, které se musejí provést, růžový kosodélník nám zobrazuje rozhodovací pole, kdy zjišťujeme, zdali jsou těstoviny již vařené. Předpokladem k realizaci tohoto algoritmu je dostatek surovin a pomůcek.



Obr 1. Algoritmus vaření těstovin

7. Shrnutí

Je nutné podotknout, že tento článek plně neobsahuje dostupné informace o tématu Algoritmus, ale to si ani nekladl za cíl. Většina čtenářů pravděpodobně umí uvařit těstoviny a tak může tento algoritmus považovat za zbytečný. Nicméně je nutné říci, že právě tento algoritmus ukazuje sílu algoritmu v jeho plné šíři – noví a noví kuchtíci si ho musí osvojit, je jednoduchý, má různá vstupní data (kuchtíky, různé druhy těstovin, hrnců). Vždy vede k cíli. Algoritmus je konečný a má jednoznačně dané navazující kroky.

Použité zdroje:

[1] http://www.gap-system.org/~history/Biographies/Al-Khwarizmi.html

[2] http://www.beranr.webzdarma.cz/algoritmy/algoritmy.html

[3] http://www.gybon.cz/~urban/pascal/ucebnice/algoritm.htm