Algoritmus

Jakub Macháček

ČVUT FEL, STM, 1.ročník, machaj22@fel.cvut.cz

Abstrakt: Zaměření článku na historii a původ výrazu algoritmus, přiblížení jeho vývoje. Seznámení se s druhy algoritmů a jejich vlastností. Vysvětlení principu složitosti algoritmů. Jednoduchý příklad algoritmu.

Klíčová slova: Algoritmus, složitost, vlastnosti algoritmů, druhy algoritmů

Etymologie

Jako původ slova algoritmus se v dnešní době uvádí práce a především jméno perského matematika a astronoma Abú Abd Alláh Muhammad Ibn Músá al-Chórezmí Abú Dža'fara, který svojí prací ovlivnil řadu svých následovníků, především pak evropských středověkých matematiků jako například Leibnitze. Od jeho části jména al-Chorézmí je odvozeno slovo algoritmus, přičemž vzniklo postupným zkomolováním a přebíráním slova jednotlivými jazyky, latiny, řečtiny, a postupem času i dalších. Al-Chorézmí stojí i za jiným dnes velmi často skloňovaným slovem, a to algebrou, a to když ve svém díle Hisáb al-džabr wa-l-muqábala položil základy algebry jako samostatné matematické disciplíny. Zkomolením slova al-džabr vzniklo slovo algebra.

Algoritmus

Algoritmus je přesný návod či postup, kterým lze vyřešit daný typ úlohy. Můžeme se na něj dívat jako na soupis jednotlivých pokynů, instrukcí a souborů akcí, které nás zavedou k námi určenému, předem danému cíli. Pojem algoritmu se nejčastěji objevuje při 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. 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 chutného pokrmu.

Vlastnosti algoritmů

Základní vlastnosti algoritmů si popíšeme v následující tabulce, kde jsou zobrazeny i s jejich krátkým popisem. V praxi jsou proto předmětem zájmu hlavně takové algoritmy, které jsou v nějakém smyslu kvalitní. Takové algoritmy splňují různá kritéria, měřená např. počtem kroků potřebných pro běh algoritmu, jednoduchost, efektivitu či eleganci. 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.

Tab.1 Vlastnosti algoritmů 1

Konečnost (finitnost)	Každý algoritmus musí skončit v konečném počtu kroků. Tento počet kroků může být libovolně velký (podle rozsahu a hodnot vstupních údajů), ale pro každý jednotlivý vstup musí být konečný.
Obecnost (hromadnost, univerzálnost)	Algoritmus neřeší jeden konkrétní problém (např. „jak spočítat 3×7“), ale obecnou třídu obdobných problémů (např. „jak spočítat součin dvou celých čísel“), má širokou množinu možných vstupů.
Determinovanost	Každý krok algoritmu musí být *jednoznačně* a *přesně* definován; v každé situaci musí být naprosto zřejmé, co a jak se má provést, jak má provádění algoritmu pokračovat, takže pro stejné vstupy dostaneme pokaždé stejné výsledky.
Výstup (resultativnost)	Algoritmus má alespoň jeden *výstup*, veličinu, která je v požadovaném vztahu k zadaným vstupům, a tím tvoří odpověď na problém, který algoritmus řeší.

Složitost algoritmů

Vzhledem ke konečnosti algoritmů je nutné uvést ještě jeden výraz, a tím je algoritmická složitost. To, že program skončí v konečném počtu kroků nám nezaručuje to, že výsledek dostaneme ve smysluplném časovém horizontu. Protože se může stát, že náš algoritmus, přestože povede k řešení problému, bude trvat tak dlouho, že již nebudeme schopni ho využít, ba co hůř, pro jeho výpočet či provedení nebudeme mít ani dostatečné prostředky. Proto musíme zajistit, aby program trval jen nejnutnější dobu. Pro vyčíslení výpočetní složitosti algoritmů v závislosti na velikosti vstupních dat se používá asymptotický zápis závislosti výpočetního času na rozsahu úlohy (typicky na počtu vstupních údajů). Například *O(log N)* znamená, že počet kroků algoritmu závisí logaritmicky na velikosti vstupních dat. Pokud u takového algoritmu zdvojnásobíme rozsah vstupních údajů, doba výpočtu se zvýší o jednu jednotku času, pokud bude vstupních dat čtyřikrát více, doba

výpočtu se prodlouží o dvě jednotky času, a tak dále. To je třeba případ nalezení jednoho prvku o určité hodnotě v seznamu prvků seřazeném podle hodnoty (např. nalezení jména v telefonním seznamu). Asymptotická složitost algoritmů jako taková je téma na samostatný článek, mohu doporučit [2].

Druhy algoritmů

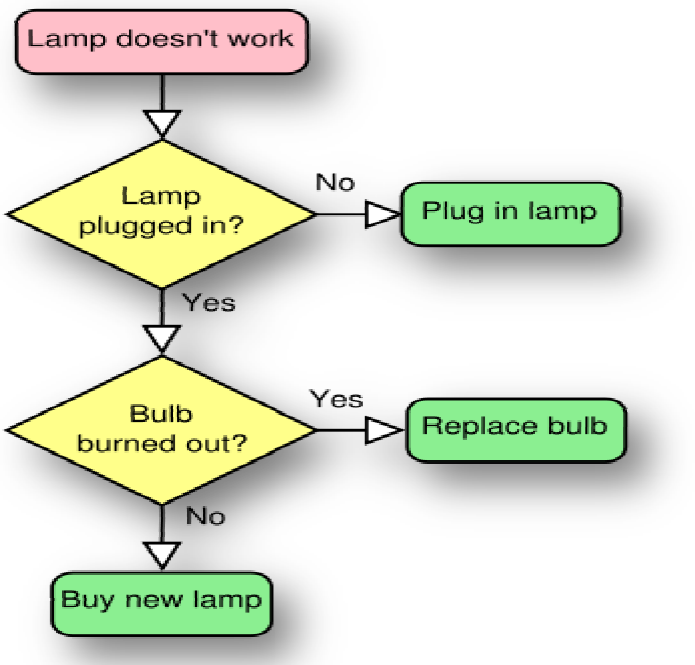
Algoritmy můžeme klasifikovat různými způsoby. Mezi důležité druhy algoritmů patří:

- Rekurzivní algoritmy, které využívají (volají) samy sebe.
- Hladové algoritmy se k řešení propracovávají po jednotlivých rozhodnutích, která, jakmile jsou jednou učiněna, už nejsou dále revidována.
- 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.
- 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.

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

Příklad

Na obr. 1 vidíme jednoduchý příklad algoritmu, postupu, jakým vyřešit problém s nesvítící žárovkou. Základem je, že lampa nesvítí. Co se s touto situací dá udělat? Musíme zkontrolovat, zda-li je lampa zapojená v zásuvce. Pokud ano, v tomto problém není. Pokud ne, zapojíme lampu a problém je vyřešen. Pokud je tedy lampa zapojená v zásuvce, musíme hledat problém jinde. Jako dalším logickým krokem se jeví zkontrolovat funkčnost žárovky. Pokud je prasklá, vyměníme jí, a problém je opět vyřešen. Pokud je v pořádku, pak nám asi nezbývá než koupit novou lampu. Tento příklad je ryze ilustrativní, sami víte, že problém by mohl být vyřešen spoustou dalších věcí, například odborným zásahem do vypínače lampy apod. ale jako příklad je, myslím si, dostatečný.



Obr. 1 Popis jednoduchého algoritmu 1, převzato z [3]

Závěr

Závěrem bych chtěl podotknout, že tento článek neměl za úkol seznámit čtenáře s řešením jednotlivých algoritmů, neboť na toto téma bylo vydáno spousta daleko odbornějších publikací, avšak jeho smyslem bylo dle autora seznámení se se základními pojmy v oblasti algoritmů a pochopení jeho důležitosti, stejně tak jako nástin jeho možných využití, či případné základní problematiky s ním spojené.

Použité zdroje

[1] Wikipedia heslo algoritmus, stav ke dni 2.4.2009 http://cs.wikipedia.org/wiki/Algoritmus

[2] Algoritmy.net, stav ke dni 2.4.2009 strana http://www.algoritmy.net/index.html?str=68&desc=Asymptotick%C3%A1+slo%C5%BEitost

[3] Wikipedia.com heslo algorithm, stav ke dni 2.4.2009 http://en.wikipedia.org/wiki/Algorithm

[4] Stránky věnované matematikovi Chvarízmímu http://cs.wikipedia.org/wiki/Al-Khwarizmi