

Z Wikipedie, otevřené encyklopedie

Algoritmus je přesný návod či postup, kterým lze vyřešit daný typ úlohy.

Pojem algoritmus se nejčastěji objevuje při [programování](#), kdy se jím myslí teoretický princip řešení problému ([kontrast s programem](#)). Všechny využití algoritmu může být využito v jakémkoli jiném vědeckém odvětví. Jako jistý druh algoritmu se může chápout i např. kuchařský recept. Zpravidla však na algoritmy klademe určitá omezení.

Vlastnosti algoritmu

V užším smyslu se souběhem algoritmus označují takové postupy, které splňují některé silnější požadavky:

Elementárnost

Algoritmus se skládá z konečného počtu jednoduchých (elementárních) kroků.

Konečnost (finitnost)

Algoritmus musí končit v konečném počtu kroků.

Algoritmus musí pro každou možnost vstupních údajů poskytovat jednoznačnou a jednu výstupní hodnotu (výsledek).

Algoritmus musí pro každý jednotlivý vstup poskytovat jednoznačný výstup. Postupy, které tuto podmínku nesplňují, se nazývají *výpočetní metody*.

Speciálním příkladem nekonečné výpočetní metody je *reaktivní proces*, který průběžně reaguje s okolním prostředím. Někteří autoři však mezi algoritmy zahrnují i takovéto postupy.

Obecnost (hromadnost, masovost, univerzálnost)

Algoritmus neřeší jeden konkrétní problém (např. „jak spočítat 3×7 “), ale celou třídu obdobných problémů (např. „jak spočítat součin dvou celých čísel“), má širokou množinu možných vstupů.

Determinovanost

Algoritmus je determinovaný, pokud za stejnými počínáněmi (pro stejný vstup) poskytuje stejný výstup. Tato vlastnost je požadována u velké většiny úloh; existují však úlohy, kdy je naopak vyžadována náhodnost (např. vklad simulace [vrhu kostkou](#), [míchání karet](#), generování [hesel](#) a [kryptografických klíčů](#)); na standardních počítačích je dosažení náhodnosti významně obtížné. Pravděpodobnostní algoritmy v sobě mají zahrnutu náhodu a nemusí být determinované.

Determinismus

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. Protože přirozené jazyky neposkytují naprostou přesnost a jednoznačnost vyjadřování, byly pro zápis algoritmů navrženy programovací jazyky, ve kterých má každý příkaz jasně definovaný význam. Vyjádření výpočetní metody v programovacím jazyce se nazývá program. Některé algoritmy jsou sice determinované, ale nejsou deterministické (například řadící algoritmus rychlé řazení s náhodnou volbou pivota).

Vídeos

Problematika efektivity algoritmů, tzn. neboť není, jak z množiny všech možných algoritmů konkrétní problém vybrat ten vhodný, se zabývají odvětví informatiky nazývané algoritrická analýza a teorie složitosti.

Metody návrhu

Algoritmus na počítači je vždy řešení pro danou úlohu

- **Shora dolů** - postup řešení nekládáme na "zadnou řadu" operací, až dospívame k elementárním krokům.
 - **Zdola nahoru** - z elementárních kroků vytváříme prostředky, které následně umožní zadávat různý počínací proces.
 - **Kombinace obojí** - obvyklý postup shora dolů doplňujeme "členěním kruhem" zdele nahoru tím, že se napříkazd použijí knihovny funkci, využit programovací jazyk nebo systém pro vytváření programů (**CASE**).

ພະນັກງານມະນຸຍາກົມ ພົມປະເທດລາວ

Při návrhu algoritmu se uplatňuje zásada výplňového přístupu. Znamená to, že abstraktní podklad je využit k vytvoření konkrétního objektu. K používání je potřeba povolení. Význam algoritmu je tedy:

Rozděl a panuj

Související informace naleznete také v článku [Rozděl a panuj \(algoritmus\)](#).

Klasický případ aplikace postupu odshora dolů. 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.

Zpracovává se množina V složená z n údajů. Tato množina se rozdělí na k dílčích množinách, které se zpracovají násobně. Získané dílčí výsledky se pak spojí a odvodí se z nich řešení pro celou množinu V .

Klasickým případem je binární vyhledávání nebo řadící algoritmus rychlé řazení.

Hladový algoritmus

Související informace naleznete také v článku [Hladový algoritmus](#).

Velice působivý přístup k řešení určité řady optimalizačních úloh.

Zpracovává se množina V složená z n údajů. Úkolem je najít podmnožinu W množiny V , která vyhovuje určitým podmínkám a přitom optimalizuje předepsanou účelovou funkci. Jakákoliv množina W , vyhovující daným podmínkám, se nazývá přípustné řešení. Přípustné řešení, pro které nabývá účelová funkce optimální hodnoty, se nazývá optimální řešení.

Hladový algoritmus se skládá z kroků, které budou procházet jednotlivé prvky množiny V a v každém kroku bude provádět určitou optimalizační proceduru.

Proces je využívá v pořadí určeném jistou výběrovou procedurou. Výběrová procedura bude založená na nějaké optimalizační míře – funkci, která může být odvozena od účelové funkce. V každém kroku ale musíme dostat přípustné řešení. Jakmile je vytvořeno takové řešení, ještě je musíme nově optimalizovat. Proces je tedy hned po vytvoření řešení opakovat.

Dynamický programování

Společně s informacemi z předešlých kroků v dynamickém programování

Dynamické programování se používá v případech kdy lze optimální řešení složit z řešení jednodušších problémů. Protože se požadavky na řešení jednodušších podproblémů mohou mnohokrát opakovat, je nutné zvolit správné pořadí jejich řešení a výsledky si zapamatovat pro opakované použití.

Opírá se o *princip optimality*: Optimální posloupnost rozhodnutí má tu vlastnost, že ať je počáteční stav a rozhodnutí jakékoliv, musí být všechna následující rozhodnutí optimální vzhledem k výsledkům rozhodnutí prvního.

Typickým příkladem využití dynamického programování jsou grafové úlohy a jejich příslušné grafové algoritmy.

Použití hrubé síly

! Uněkterých úloh nezbývá než postupně probírat všechna možná řešení – tak zvaná metoda hrubé síly – vygenerují se všechny možné posloupnosti a pak se všechna řešení hodnotí.

Backtracking

stavového prostoru

prohledávání do hloubky

vektor

n-tici

...

složky. Pokud jsou vyčerpány všechny hodnoty i-té složky, vrátí se metoda zpět o jeden krok a zkouší další možnou hodnotu x_{i-1} .

Dříkladom je třeba **problém osmi dam** řešit pomocí **čínského koně** celou **šachovnicí**.

šachové

výpočetní složitosti algoritmů asymptotický

uioni (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 případ nalezení jednoho prvku o určité hodnotě v seznamu prvků seřazeném podle hodnoty (např. **nalezení jména v telefonním seznamu**).

Druhy algoritmů

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

- **Rekurzivní algoritmy**, které využívají (volají) samy sebe.
- **Pravděpodobnostní algoritmus** (někdy též *probabilistické*) provádí některá rozhodnutí náhodně či **pseudonáhodně**.

- V případě, že máme k dispozici více počítačů, můžeme úlohu mezi ně rozdělit, což nám umožní ji vyřešit rychleji; tomuto cíli se věnují parallelní algoritmy.
V parallelních algoritmech je úloha rozdělena na části, které mohou být zpracovány současně. V genetickém programování se často používají aplykace přímo na úvodžatry (např. programy), které mají mnoho výpočty (ještě výše uvedené primární).
 - Asynchronní algoritmy se začlenují do sítě počítačů (gridu), kde jednotlivé výpočty jsou vydávány v akci, když čekají na zadání (např. čekají na poslání dat) nebo využívají výpočetního prostoru (mimo počítač zadavatele). Když je výpočet dokončen, je výsledek vydán zadavateli.

These factors support the need to increase the value of the economy; supported by the increased production of manufactured

Ways of seeing in poetry

- **Indirect consequences**
 - **Highly supportive**
 - **Significant in Canada**
 - **Different significances**
 - **Reinforcement of family structure**

Environ Biol Fish

However, the most significant factor in the development of the modern state was the French Revolution (1789-1799), triggered by the Abolition of the Slave Trade (1807) and the French Revolution (1789). The revolution in France led to the French Empire (1804-1815), which was followed by the Spanish Empire (1808-1814) and the British Empire (1815-1816). The French Empire was established by Napoleon Bonaparte, who became the Emperor of France in 1804. The Spanish Empire was established by King Ferdinand VII in 1808. The British Empire was established by King George III in 1815. The French Empire was overthrown in 1815 by the Prussian Empire (1815-1848), which was followed by the Spanish Empire (1815-1848) and the British Empire (1815-1848). The Spanish Empire was overthrown in 1848 by the French Empire (1848-1852), which was followed by the British Empire (1848-1852). The British Empire was overthrown in 1852 by the Spanish Empire (1852-1868), which was followed by the French Empire (1868-1875). The Spanish Empire was overthrown in 1875 by the British Empire (1875-1886), which was followed by the French Empire (1886-1898). The British Empire was overthrown in 1898 by the Spanish Empire (1898-1908), which was followed by the French Empire (1908-1918). The Spanish Empire was overthrown in 1918 by the British Empire (1918-1928), which was followed by the French Empire (1928-1938). The British Empire was overthrown in 1938 by the Spanish Empire (1938-1948), which was followed by the French Empire (1948-1958). The Spanish Empire was overthrown in 1958 by the British Empire (1958-1968), which was followed by the French Empire (1968-1978). The British Empire was overthrown in 1978 by the Spanish Empire (1978-1988), which was followed by the French Empire (1988-1998). The Spanish Empire was overthrown in 1998 by the British Empire (1998-2008), which was followed by the French Empire (2008-2018). The British Empire was overthrown in 2018 by the Spanish Empire (2018-2028), which was followed by the French Empire (2028-2038).

Přistupem času se kvůli neznalosti původu slova jeho podoba měnila, záměnou arabského kořene s kořenem řeckého slova ἀριθμός (arithmos) se z algoritmu stal algoritmus. (Později byly v některých jazycích i včetně češtiny označení měnily, v každání se vrátilo.) Toto slovo se používalo jako označení různých matematických postupů, např. v 18. století označoval itinský termín *algorithmus infitesimalis* „metodu výpočtů s využitím nečné malých veličin, vynalezenou německým matematikem Leibnizem“. Slovo *algoritmus* v dnešním významu se používá až zhruba od 20. století.

Historie: Vývoj pojmu „algoritmus“

Starověké Řecko

Algoritmy byly použity ve starověkém Řecku. Například Eratosthenovo síto Eukleidův algoritmus.

Původ

Algoritmus pochází z 9. století a je odvozeno z příjmení perského matematika Al-Chorezmí. Slovo původně odkazovalo na pravidla provádění aritmetických operací s arabskými číslicemi, ale vyvinulo se prostřednictvím arabských matematikov a jména na „algoritmus“ v 18. století a zahrnuje všechny určité postupy pro řešení problémů nebo plnění úkolů.

Diskrétní a rozeznatelné symboly

Značky: K průříčání stáří, pytlů s uhlím a peněz ve starověku se používaly akumulační kameny, značky vyškrábané na holých nebo záznamových lítivých symbolů v jílu. Značky jsou obvykle v jedničkové soustavě, která se používá při kódování informací pro Turingovy stroje v teorii automatů.

Mechanická zařízení s diskrétními stavami

Hodiny: Podle Boltera je vynález mechanických hodin jedním z klíčových vynálezů. Zejména pak jejich setrvačná část - Lihýř. Přesný automat vedl od žitě k mechanickému automatu (začátek 13. století) a nakonec k mechanickým strojům - diferenční a analytický stroj (Charles Babbage a Ada Lovelace) v polovině 19. století. Lovelace je považována za první vytvoření

Analýtický stroj je považován za první Turingův kompletní počítač. Charles Babbage je někdy nazýván jako historicky první programátor.

Logické stroje 1870 – [Jevonsovo](#) logické počítadlo a logický stroj: Technický průlom využívající sítě sítí, které byly přešlechovány podobě podobné tomu, co je nyní známo jako [Karnaughova mapa](#). Jevons (1880) využívá první jednoduché počítadlo ze dřeva vybavené kolíky tak, aby jakákoli jeho třída kombinací šla vyzvednout mechanicky. Tento stroj je představen členům královské společnosti v roce 1870.

[Tkalcovský stav](#), [děrné štítky](#), [telegrafie](#) a [telefonie](#) – elektromechanické relé: Bell a Newell (1971) označují, že tkalcovský stav (1801), předchůdce děrných štítků (1887) a telefonní spínací technologie vedly k vývoji prvních počítačů. V polovině 19. století telegraf, předchůdce telefonu, byl v provozu po celém světě. V roce 1910 se objevil [dálnopis](#), který využíval mezinárodní telegrafní abecedu.

[Telefonní síť elektromechanických relé](#) – George Stibitz (1937) pracoval v Bellových laboratořích a dokončil kalkulátor, který je schopen pracovat s komplexními čísly.

Matematika v průběhu 19. století až do poloviny 20. století

Symboly a pravidla: V rychlém sledu za sebou matematika George Boole, [Gottlob Frege](#) a [Giuseppe Peano](#) redukovala aritmetiku do sekvence symbolů, se kterými se manipulovalo pomocí daných pravidel. Peanova *The principles of arithmetic, presented by a new method* (1888) byl první pokus o axiomatizování matematiky v symbolický jazyk.

Heijenoort dává Fregemu (1879) tuto slávu: Fregovo dílo je možná nejdůležitější práce, která kdy bylo v logice napsána. Tato práce byla dále zjednodušena a umocněna [Alfredem North Whiteheadem](#) a [Bertrandem Russellem](#) v jejich Principia Mathematica (1910-1913).

Paradoxy: Ve stejné době se objevila řada znepokojivých elementů v literatuře, zejména [Burali-Fortiho paradox](#) (1897), [Russellův paradox](#) (1902-1903) a Richardův paradox. Výsledné úvahy vedly k [Gödelovým větám o neúplnosti](#).

Efektivní výčíslitelnost: Ve snaze vyřešit [Entscheidungsproblem](#) přesně definovaným [Hilbertem](#) v roce 1928 museli matematici nejprve definovat, co se rozumí pod pojmem „efektivní metoda“ nebo „efektivní výpočet“. V rychlém sledu se objevili [Alonzo Church](#), [Stephen Cole Kleene](#) a J. B. Rosser, kteří jsou známí především díky [lambda kalkulu](#). Church pak společně s [Turingem](#) ukázal, že lambda kalkul (a další výpočetní modely) má výpočetní sílu [Turingova stroje](#), což otevřelo cestu k [Churchově–Turingově tezi](#).

Právní ustanovení

Algoritmy nejsou obvykle patentovány. Samotná manipulace s abstraktními pojmy, čísla, či dokonce signály není v USA (dle USPTO 2006) považována za

[Gottschalk v. Benson](#)

[Diamond v. Diehr](#)

[zpětné vazby](#)

[softwaru](#)

[kompresi dat.](#)

[Ottův slovník naučný](#)

[Donald Ervin Knuth *The Art of Computer Programming*](#)

[ISBN 0-201-48541-9](#)

[Handbook of Algorithms and Data Structures.](#)

[Dictionary of Algorithms and Data Structures](#)

[United States Patent and Trademark Office – \(2006\), 2106.03.](#)

[**>Mathematical Algorithms: 2100 Patentability](#), Manual of Patent Examining Procedure (MPEP). Latest revision August 2006

Externí odkazy

- Obrázky, zvuky či videa k tématu [algoritmus](#) na Wikimedia Commons

- Téma [Algoritmus](#) ve Wikicitátech
- Slovníkové heslo [algoritmus](#) ve Wikislovníku
- [První algoritmus napsala žena. Programovala dávno před prvním počítačem](#)

Citováno z „<https://cs.wikipedia.org/w/index.php?title=Algoritmus&oldid=24926977>“

Kategorie:

- [Algoritmy](#)
- [Programování](#)

SKRYTE KATEGORIE:

- [Monitoring:Články s identifikátorem NKC](#)
- [Monitoring:Články s identifikátorem BNE](#)
- [Monitoring:Články s identifikátorem BNF](#)
- [Monitoring:Články s identifikátorem GND](#)
- [Monitoring:Články s identifikátorem LCCN](#)
- [Monitoring:Články s identifikátorem LNR](#)
- [Monitoring:Články s identifikátorem NDL](#)
- [Monitoring:Články s identifikátorem NII](#)
- [Monitoring:1000 nejdůležitějších článků/střední](#)

<input type="text" value="Hledání"/>	
<input type="button" value="Hledat"/>	
Speciální:Hledání	
<input type="button" value="Hledat"/>	
Algoritmus	
<input type="button" value="Hledat"/>	

136 jazyků

[Přidat téma](#)