Pojem algoritmus

doc. Mgr. Jiří Dvorský, Ph.D.

Katedra informatiky
Fakulta elektrotechniky a informatiky
VŠB – TU Ostrava

Prezentace ke dni 12. února 2020



Osnova přednášky

Pojem algoritmus

- Algoritmický problém
- Pojem algoritmu
- Vlastnosti algoritmu
- Prostředky pro zápis algoritmů
- Algoritmus – formálnější přístup
- Shrnutí

Algoritmický problém versus algoritmus

- vstupní data
- vstupní podmínka – co vše musí splňovat vstupní data
- výstupní data
- výstupní podmínka – co vše musí splňovat výstupní data

Příklad

Algoritmický problém – „Vyvrtání díry do betonového panelu“

- vstupní data – betonový panel
- vstupní podmínka – specifikace panelu
- výstupní data – díra v panelu
- výstupní podmínka – průměr a hloubka vyvrtané díry.

Algoritmický problém versus algoritmus (pokrač.)

- Algoritmický problém reprezentuje zadání úlohy a algoritmus představuje způsob, jak tento problém řešit.
- Zřejmě může existovat více algoritmů řešících tentýž algoritmický problém.
- **Algoritmus** budeme chápat jako postup, jak vstupní data transformovat na výstupní.
- K našim účelům bude stačit tato intuitivní definice algoritmu

Příklad

Možné algoritmy řešící problém „Vyvrtání díry do betonového panelu“:

1. dloubání kávovou lžičkou,
2. ruční vrtačka
3. příklepová vrtačka
4. vrtací kladivo s SDS upínáním. . .

Algoritmický problém versus algoritmus (pokrač.)

Poznámky

- Sestavit správný algoritmus řešení problému nemusí být vždy snadnou činností.
- Logická chyba v algoritmu může vést k nesprávným výsledkům.
- Pro řešení jedné a té samé úlohy může existovat více různých algoritmů, tyto algoritmy se můžou lišit v množství spotřebovaného času a paměťového prostoru.
- Algoritmus popisuje **konstruktivní** řešení problému – řešení problému je pomocí algoritmu **konstruováno** ze vstupních dat! Stejný postup používá například geometrie.

Nekonstruktivní řešení problému aneb Na tohle není algoritmus!

Příklad

Naším úkolem je najít dvě iracionální čísla x a y taková aby platilo, že x^y je číslo racionální.

Řešení

Zvolíme $x = \sqrt{2}$ a $y = \sqrt{2}$. Je-li x^y číslo racionální jsme hotovi, není-li x^y číslo racionální, zvolíme například $x = \sqrt{2}^{\sqrt{2}}$ a $y = \sqrt{2}$. Potom dostaneme

$$x^y = \left(\sqrt{2}\right)^{\sqrt{2}^{\sqrt{2}}} = \left(\sqrt{2}\right)^{\sqrt{2}\sqrt{2}} = \left(\sqrt{2}\right)^2 = 2$$

Je zřejmé, že řešením je buď dvojice čísel $x = \sqrt{2}$ a $y = \sqrt{2}$ nebo dvojice čísel $x = \sqrt{2}^{\sqrt{2}}$ a $y = \sqrt{2}$, přičemž nejsme z uvedeného řešení schopni říci která dvojice je vlastně řešením našeho problému.

Nekonstruktivní řešení problému aneb Na tohle není algoritmus! (pokrač.)

V čem spočívá problém?

1. není jasné zda je $\sqrt{2}^{\sqrt{2}}$ číslo iracionální nebo ne,
2. není jasné jak volit kandidáty na řešení – iracionální čísla jsou nespočetná.

Spočetnost množiny čísel

- O množině řekneme, že je spočetná pokud ji lze zobrazit na množinu přirozených čísel.
- Přirozená čísla lze „probírat, vyjmenovat, jedno po druhém“.
- Stejně to lze provést s čísly celými a racionálními (zlomky).
- Nelze to provést s čísly iracionálními, reálnými a komplexními.
- Po 1 následuje nutně 2, po $2,5$ následuje co? $2,51$ nebo $2,501$ nebo $2,5001$? Jaké číslo následuje po $\sqrt{2}$?

O původu slova algoritmus

- Slovo „algoritmus“ je odvozeno od perského matematika jménem *Abú Abd Alláh Muhammad Ibn Músá al-Chórezmí Abú Dža'far* česky
  Otec Abdulláha, Mohamed, syn Mojžíšův, pocházející z města Chwarizm.
- Město Chwarizm, někdy též Chórézm, najdete v dnešním Uzbekistánu pod jménem Chiva.
- Jeho mateřským jazykem byla perština, ale knihy psal arabsky.
- Jméno Al-Chórezmí bylo ve středověku latinizované na Al-Gorizmí, později na Algorismí.
- Zhruba v roce 825 napsal knihu Kitáb al-džám'a wa-l-tafríq bil-hisáb al-hindi, kterou v roce 1145 přeložil Robert z Chesteru do latiny jako Algorithmi de numero indorum.

O původu slova algoritmus (pokrač.)

- V této knize popsal indický poziční číselný systém, výpočty s ním a číslo 0.
- Slovo algorismus, odvozeno od jeho jména, původně znamenalo „provádění aritmetiky pomocí arabských číslic“; abacisté počítali pomocí abaku, algoristé pomocí algorismů.
- Zavedl algebru jako samostatnou matematickou disciplínu, slovo algebra pochází z arabského al-džabr.
- Dále zavedl pojem neznámá a označovat ji tradičně písmenem x.

Algoritmus

Algoritmus je předpis, který se skládá z kroků a který zabezpečí, že na základě vstupních dat jsou poskytnuta požadovaná data výstupní. Navíc každý algoritmus musí mít následující vlastnosti:

- konečnost,
- hromadnost,
- jednoznačnost,
- opakovatelnost,
- rezultativnost,
- elementárnost.

Vlastnosti algoritmu

Konečnost

- Požadovaný výsledek musí být poskytnut v „rozumném“ čase.
- Pokud by výpočet trval na nejrychlejším počítači např. jeden milion let, těžko bychom mohli hovořit o algoritmu řešení, nemluvě o výpočtu, který by neskončil vůbec.
- Za rozumný lze považovat čas, kdy nám výsledek výpočtu k něčemu bude.

Hromadnost

- Vstupní data nejsou v popisu algoritmu reprezentována konkrétními hodnotami, ale spíše množinami, ze kterých lze data vybrat (např. při třídění přirozených čísel bude vstup konečnou podmnožinou množiny všech přirozených čísel).
- Při popisu algoritmu v programovacím jazyce se to projeví tím, že vstupy do algoritmu jsou označeny symbolickými jmény.

Vlastnosti algoritmu (pokrač.)

Jednoznačnost

- Každý předpis je složen z kroků, které na sebe navazují.
- Každý krok můžeme charakterizovat jako přechod z jednoho stavu algoritmu do jiného, přičemž každý stav je určen zpracovávanými daty.
- Tím, jak data v jednotlivých stavech algoritmu vypadají, musí být jednoznačně určeno, který krok následuje.
- Například v řešení trojúhelníka může nastat situace, kdy vychází na základě vstupních dat jedno nebo dvě řešení. Situace je tedy nejednoznačná, řešení musí být jednoznačné, tzn. v předpisu se s touto možností musí počítat a musí v něm být návod, jak ji řešit.

Vlastnosti algoritmu (pokrač.)

Opakovatelnost

- Při použití stejných vstupních údajů musí algoritmus dospět vždy k témuž výsledku.
- Algoritmus si tedy můžeme představit jako mlýnek na maso – pokud do něj vložíme stejné suroviny a zatočíme klikou, dostaneme vždy stejný výsledek jako při předchozím mletí.

Rezultativnost

- Algoritmus musí vydat aspoň jeden výstup, který je správný tj. vyhovuje výstupní podmínce.

Elementárnost

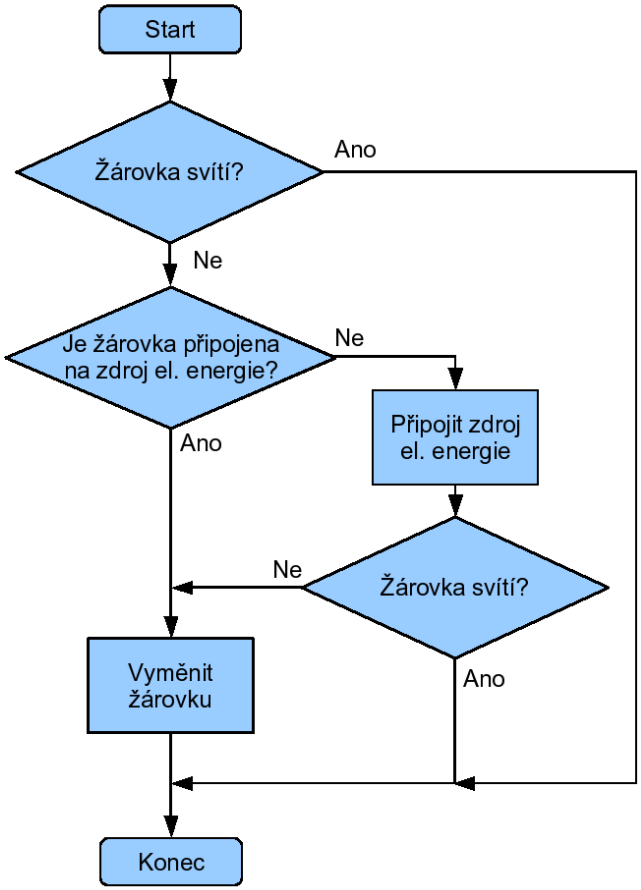
- Algoritmus se skládá z konečného počtu jednoduchých (elementárních) kroků.

Prostředky pro zápis algoritmů

Pro zápis algoritmů můžeme využít:

- přirozený jazyk – slovní popis například receptu, „Vraž do toho kopu vajec a dvě libry másla. . . “
- grafický zápis – vývojový diagram, strukturogram,
- pseudokód nebo
- programovací jazyk.

Vývojový diagram



Vývojový diagram [2]

- grafické znázornění jednotlivých kroků algoritmu,
- obrazce různého tvaru, navzájem propojené pomocí šipek,
- obrazce reprezentují jednotlivé kroky:
  - obdélník s popisem – dílčí krok zpracování,
  - kosočtverec – větvení postupu v závislosti na splnění podmínky,
  - obdélník se zaoblenými rohy – počátek nebo ukončení zpracování,
- šipky určují směr zpracování
- Frank Gilbreth, 1921, dokumentace procesů v průmyslu.

Pseudokód

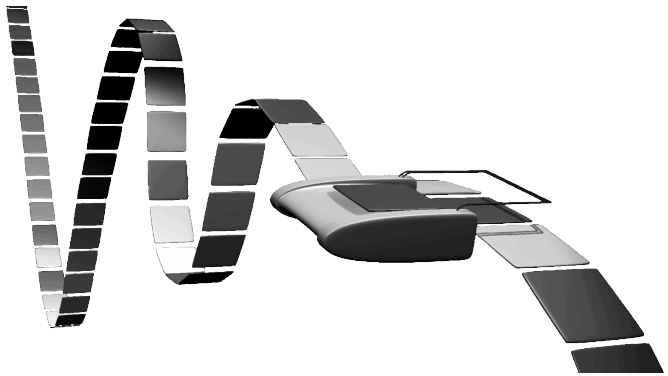
```
1 while i < n do
2 |   a while loop including a repeat–until loop;
3 |   repeat
4 |   |   if x ∈ A then
5 |   |   |   we do that;
6 |   |   end
7 |   until this end condition;
8 end
```

- kompaktní a neformální způsob zápisu počítačového algoritmu,
- používá strukturní konvence programovacích jazyků,
- nezahrnuje detailní syntaxi specifické pro konkrétní programovací jazyk,
- pro srozumitelnost je částečně doplněn popisy podrobností v přirozeném jazyce nebo kompaktně vyjádřeným matematickým zápisem.

Turingův stroj

Turingův stroj

- teoretický model počítače popsaný matematikem Alanem Turingem,
- procesorová jednotka, tvořená konečným automatem,
- program ve tvaru pravidel přechodové funkce,
- a pravostranně nekonečné pásky pro zápis mezivýsledků.



Turingův stroj [1]

Neformálně řečeno je Turingův stroj primitivní počítač s co nejjednoduššími instrukcemi a jednou pamětí, páskou.

Turingův stroj (pokrač.)

Výpočet na Turignově stroji:

1. na pásku zapíšeme vstupní posloupnost symbolů w
2. Turingův stroj začne zpracovávat vstup a na pásce budeme očekávat výstupní symboly.
3. Turingův stroj ukončí výpočet ve třech možných stavech:
   - 3.1 vstup w odpovídá vstupním podmínkám, vstup byl přijat a na pásce je výsledek, **přijato**,
   - 3.2 vstup w neodpovídá vstupním podmínkám, vstup nebyl přijat, **nepřijato**
   - 3.3 Turingův stroj se **zacyklí** a nikdy se nezastaví, **nelze rozhodnout**

Turingův stroj tedy **neumí řešit všechny problémy**.

Church-Turingova teze

Church-Turingova teze

Ke každému algoritmu existuje ekvivalentní Turingův stroj.

- Jinak řečeno, vše co lze vypočítat Turingovým strojem (počítačem), lze vypočítat i algoritmem a naopak.
- Naše počítače jsou tedy stejně „schopné“ jako kterékoliv jiné počítače, které by bylo možné sestrojit „na druhém konci vesmíru“.
- Nebereme ale v úvahu rychlost počítače a velikost paměti, což v praxi hraje významnou roli.
- Kvůli neformální definici pojmu algoritmus nemůže být tato teze nikdy dokázána.
- Lze ji ale vyvrátit, podaří-li se sestrojit algoritmus, který bude umět řešit problémy, které Turingův stroj řešit neumí.

Church-Turingova teze (pokrač.)

- Jelikož každý počítačový program lze přeložit do jazyka Turingova stroje a obvykle i naopak, lze tezi ekvivalentně formulovat pro kterýkoli běžně používaný programovací jazyk, který je turingovsky úplný tj. ekvivalentní Turingovu stroji.
- Z toho plyne, že všechny programovací jazyky mají stejnou vyjadřovací sílu, lze v nich naprogramovat stejné algoritmy.
- Otázkou zůstává, proč existuje takové množství programovacích jazyků? Důvod: důvod je praktický, programovací jazyky jsou uzpůsobeny pro jistý okruh problémů. V jednom jazyce se lépe programuje tato úloha, v jiném jazyce jiná.

Shrnutí

- Algoritmický problém
- Algoritmus
  - konečnost,
  - hromadnost,
  - jednoznačnost,
  - opakovatelnost,
  - rezultativnost,
  - elementárnost.
- Turingův stroj
- Church-Turingova teze

Literatura

[1] Turingův stroj. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-09-26]. Dostupné z: https://cs.wikipedia.org/wiki/Turing%C5%AFv_stroj#/media/File:Maquina.png

[2] Vývojový diagram. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-09-25]. Dostupné z: https://commons.wikimedia.org/w/index.php?curid=9024401

Děkuji za pozornost