Algoritmizace

Algoritmy a jejich efektivita

Osnova

- Co je to algoritmus?
- Jak budeme algoritmy popisovat?
- Jak budeme ověřovat jejich správnost?
- Jak změřit efektivitu algoritmu?
- Asymptotická složitost

Algoritmus

أبو عبد الله محمد بن موسى الخوارزمي ابو جعفر

Abú Abd Alláh Muhammad Ibn Músá al-Chórezmí

Perský učenec, cca 780 - 850

Algoritmus

أبو عبد الله محمد بن موسى الخوارزمي ابو جعفر

Abú Abd Alláh Muhammad Ibn Músá al-Chórezmí

Perský matematik & astronom, cca 780 - 850

- systém arabských číslic
- základy algebry
- řešení lineárních
  & kvadratických rovnic

Co je to algoritmus?

Intuitivní pojem

Popis takového řešení problému,

které lze realizovat na počítači.

Co je to algoritmus?

Konečná posloupnost
elementárních příkazů,
jejichž provádění umožňuje
pro každá přípustná vstupní data
mechanickým způsobem
získat po konečném počtu kroků
příslušná výstupní data.

J. Drózd, R. Kryl, Začínáme s programováním, Grada, Praha 1992.

Vlastnosti algoritmu

Konečnost

Hromadnost (obecnost, univerzálnost)

Resultativnost (výstup)

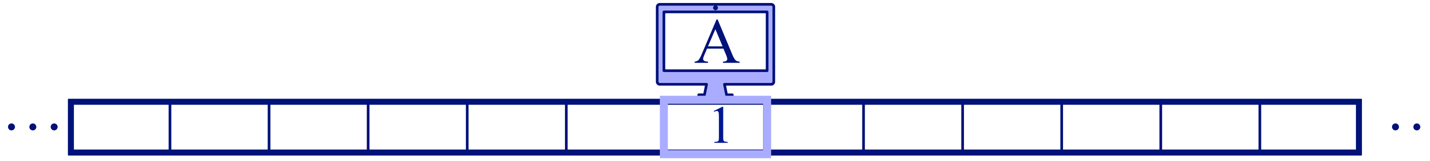
Jednoznačnost

Determinismus

Formální modely algoritmu

Turingův stroj (Alan Turing, 1936)

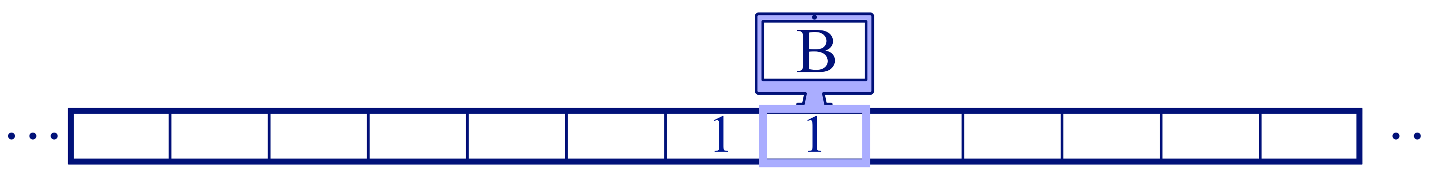
symbol na pásce	stav A	stav B
□	1, B, →	1, A, ←
1	1, B, ←	1, HALT, →



Formální modely algoritmu

Turingův stroj (Alan Turing, 1936)

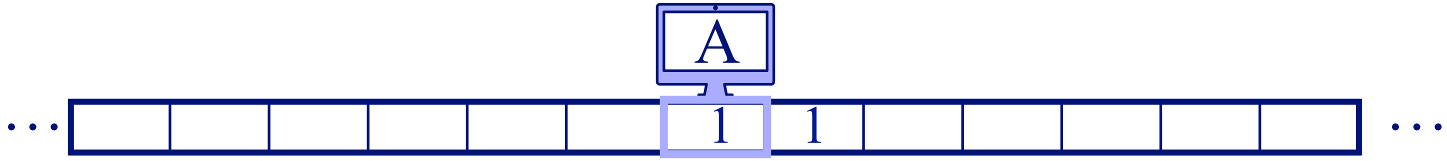
symbol na pásce	stav A	stav B
□	1, B, →	1, A, ←
1	1, B, ←	1, HALT, →



Formální modely algoritmu

Turingův stroj (Alan Turing, 1936)

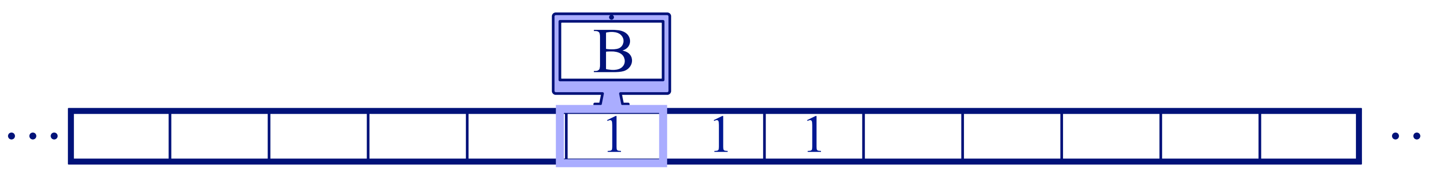
symbol na pásce	stav A	stav B
□	1, B, →	1, A, ←
1	1, B, ←	1, HALT, →



Formální modely algoritmu

Turingův stroj (Alan Turing, 1936)

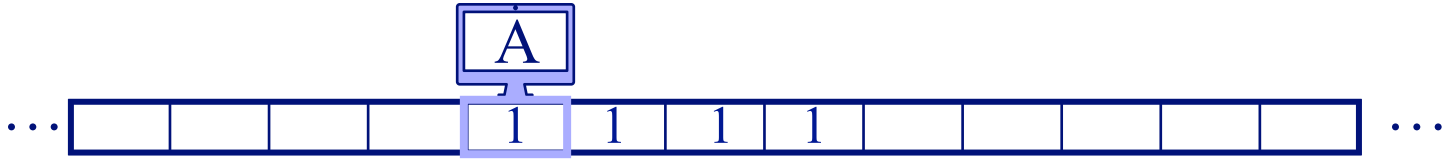
symbol na pásce	stav A	stav B
□	1, B, →	1, A, ←
1	1, B, ←	1, HALT, →



Formální modely algoritmu

Turingův stroj (Alan Turing, 1936)

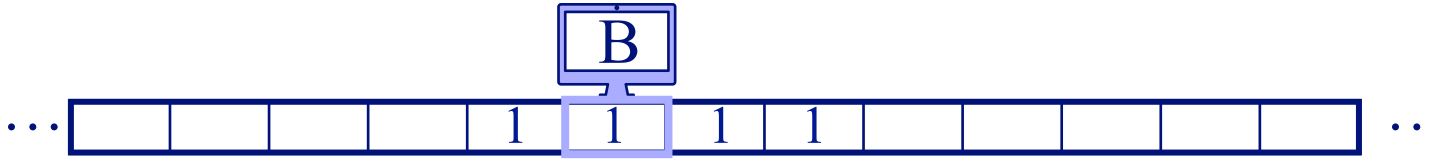
symbol na pásce	stav A	stav B
□	1, B, →	1, A, ←
1	1, B, ←	1, HALT, →



Formální modely algoritmu

Turingův stroj (Alan Turing, 1936)

symbol na pásce	stav A	stav B
□	1, B, →	1, A, ←
1	1, B, ←	1, HALT, →



Formální modely algoritmu

Turingův stroj (Alan Turing, 1936)

Busy Beaver
(T. Radó, 1962)

symbol na pásce	stav A	stav B
□	1, B, →	1, A, ←
1	1, B, ←	1, HALT, →

n	Σ
1	1
2	4
3	6
4	13
5	?

Formální modely algoritmu

Turingův stroj (Alan Turing, 1936)

- Churchova teze

RAM počítač

Rekurzivní funkce (Kurt Gödel, 1934)

Lambda kalkul (Alonzo Church, 1941)

Jak budeme algoritmy popisovat?

Zápis v pseudokódu

- použití přirozeného jazyka
- řídící struktury vypůjčené z jazyka Python

Nebudeme se zabývat problémy softwarového inženýrství jako

- modularita
- objektový přístup
- ošetření chyb apod.

Problém ✎

Jsou dány rovnoramenné váhy a n kuliček.
Navrhněte algoritmus, který najde

1. nejtěžší kuličku na co nejmenší počet vážení
2. nejtěžší i nejlehčí kuličku s použitím nejvýše $3\lfloor n/2 \rfloor$ vážení
3. druhou nejtěžší kuličku s použitím nejvýše n-$2+\lceil \log_2 n \rceil$ vážení.

Ověření správnosti algoritmu

= ověření konečnosti + částečné správnosti

Konečnost

- pro každá přípustná vstupní data obdržíme v konečném čase nějaký výstup

Částečná správnost (parciální korektnost)

- když výpočet nad přípustnými vstupními daty skončí
- pak na výstupu obdržíme správný výsledek

Algoritmus je správný = částečně správný
+ konečný

Porovnávání efektivity algoritmů

Dvě míry

- čas
- prostor (paměť)

Jak změřit časovou / prostorovou náročnost výpočtu?

- délka výpočtu počet kroků
- prostorová náročnost výpočtu rozsah použité pracovní paměti

Co je to krok výpočtu ?

Krok výpočtu

- elementární operace
- kterou lze provést v konstantním čase

Příklady

- provedení logického testu
- aritmetické operace
- přiřazení

Co je to složitost algoritmu?

Délka (prostorové nároky) výpočtu závisí na

- velikosti vstupních dat
- konkrétní hodnotě vstupních dat

Přirozené zjednodušení

- složitost algoritmu bude funkcí velikosti vstupu

Problém

- pro dané n může existovat více přípustných vstupů o této velikosti !

Přístupy k analýze složitosti

Nejhorší případ

maximální délka výpočtu nad vstupem délky n

Nejlepší případ

minimální délka výpočtu nad vstupem délky n

Průměrný případ

součet délek výpočtů nad všemi vstupy délky n
/ počet vstupů délky n

Pravděpodobnostní analýza algoritmů

Příklad

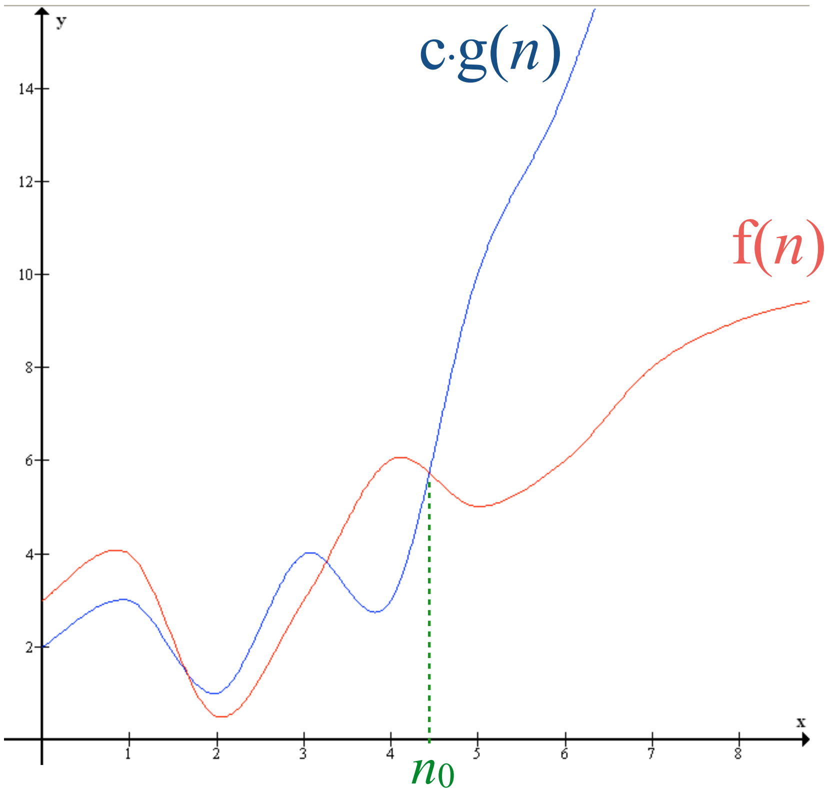
Navrhněte algoritmus, který setřídí n zadaných kuliček $a_1, \ldots, a_n$ od nejlehčí po nejtěžší.

```
for j in range(n-1):
    for i in range(1,n-j):
        if a[i] těžší než a[i+1]:
            vyměň a[i] ↔ a[i+1]
```

Asymptotická notace

Funkce $f(n) = O(g(n))$, pokud $\exists$ $c > 0$ a $n_0 \in \mathbf{N}$ tak,

že $0 \leq f(n) \leq c \cdot g(n)$ pro každé $n \geq n_0$.



Asymptotická notace

Funkce $\mathrm{f}(n) = \Omega(\mathrm{g}(n))$, pokud $\exists$ $\mathrm{c} > 0$ a $n_0 \in \mathbf{N}$ tak,

že $0 \leq \mathrm{c} \cdot \mathrm{g}(n) \leq \mathrm{f}(n)$ pro každé $n \geq n_0$.

Funkce $\mathrm{f}(n) = \Theta(\mathrm{g}(n))$, pokud

$\mathrm{f}(n) = \mathrm{O}(\mathrm{g}(n))$ a $\mathrm{f}(n) = \Omega(\mathrm{g}(n))$.

Spektrum časové složitosti

$\Theta(1)$ (např. je číslo liché / sudé?)

$\Theta(\log n)$ (binární vyhledávání)

$\Theta(n)$ (nalezení minima / maxima)

$\Theta(n \log n)$ (HeapSort, MergeSort)

$\Theta(n^2)$ (BubbleSort, InsertSort)

$\Theta(n^3)$ (násobení matic dle definice)

…

pracují v polynomiálně omezeném čase

$\Theta(2^n)$

$\Theta(n!)$

…

pracují v exponenciálním čase

---

algoritmicky nerozhodnutelné

Problém ✎

Dokažte nebo vyvraťte:

Pro každdou dvojici funkcí f,g: **N**→**R** platí

1. Pokud $f(n)=O(g(n))$, pak $g(n)=O(f(n))$
2. Pokud $f(n)=O(g(n))$, pak $2^{f(n)}=O(2^{g(n)})$
3. pokud $f(n)=O(g(n))$, pak $g(n)=\Omega(f(n))$
4. $f(n)=O(f(n)^2)$