NPRG062 Algoritmizace 2/1 Z+Zk 4 kr.

Pavel Töpfer
Katedra softwaru a výuky informatiky MFF UK
MFF Malostranské nám., 4. patro, pracovna 404

pavel.topfer@mff.cuni.cz
https://ksvi.mff.cuni.cz/~topfer

Algoritmizace
- metody řešení úloh na počítači
- algoritmy, datové struktury, programovací techniky

Správnost algoritmu
- konečnost
- parciální korektnost (správné výsledky, když výpočet skončí)

Efektivita algoritmu (složitost algoritmu)
- časová (počet vykonaných operací)
- prostorová (paměť potřebná na uložení pracovních dat)

Učivo

- algoritmus, časová a paměťová složitost
- dělitelnost čísel, Eukleidův algoritmus
- test prvočíselnosti, Eratosthenovo síto
- rozklad čísla na cifry
- aritmetika s vyšší přesností („dlouhá čísla“)
- Hornerovo schéma, poziční číselné soustavy
- algoritmy vyhledávání v poli (sekvenční, binární, zarážka)
- třídění čísel v poli - přímé metody, heapsort, mergesort, quicksort
- složitost problému třídění
- přihrádkové třídění
- reprezentace dat v paměti
- zásobník, fronta, slovník, halda
- spojový seznam

Učivo (pokračování)

- rekurze – princip, příklady, efektivita
- binární a obecný strom – reprezentace, průchod, použití
- binární vyhledávací strom, princip vyvažování
- notace aritmetického výrazu – vyhodnocení, převody
- hešovací tabulka
- prohledávání stavového prostoru do hloubky a do šířky
- metody zrychlení backtrackingu – ořezávání, heuristiky
- programování her, algoritmus minimaxu
- metoda Rozděl a panuj
- dynamické programování
- reprezentace grafu
- prohledávání grafu, základní grafové algoritmy

Studijní zdroje

Prezentace z přednášek
- aktuálně vždy po přednášce v kurzu Moodle
- často rozšířené a doplněné oproti tomu, co bylo přednášce
- ukázkové programy

Kurz v univerzitním systému Moodle
- https://dl1.cuni.cz/course/view.php?id=8186
- přihlášení do systému Moodle: přístupové údaje do CAS (SIS)
- přihlášení do kurzu Algoritmizace heslem:
    kód vašeho cvičení Algoritmizace ze SISu ve tvaru 24aNPRG062x??
- prezentace, videonahrávka a programové ukázky i z druhé paralelky

- informace o zkouškách, ukázky starších zadání úloh

Pavel Töpfer: Algoritmy a programovací techniky
Prometheus Praha 1995, 2. vydání 2007
tištěná v knihovnách, nyní k zakoupení pouze jako e-kniha
https://www.prometheus-eknihy.cz/

Programátorské kuchařky KSP
krátké studijní texty k jednotlivým tématům algoritmizace a programování
http://ksp.mff.cuni.cz/kucharky/

Martin Mareš, Tomáš Valla: Průvodce labyrintem algoritmů
CZ.NIC Praha 2017
text pdf volně ke stažení
http://pruvodce.ucw.cz/
https://knihy.nic.cz/

Zápočet

- uděluje cvičící
- pravidelná aktivní práce na cvičeních
- domácí úkoly (průběžně řešit!)
- případné další požadavky cvičícího (písemky)

Zkouška

- společné požadavky a zkušební termíny pro obě paralelky přednášky
- přihlašování prostřednictvím SISu
- k účasti na zkoušce je nutné ***předchozí získání započtu***
- povinná písemná část a nepovinná ústní část
- podrobnosti a ukázky starších zadání jsou v kurzu Moodle
- informace o průběhu zkoušky včas upřesníme

Algoritmus

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

Vlastnosti:

- konečnost
- hromadnost
- resultativnost
- jednoznačnost
- determinismus

Formální modely algoritmu

rekurzivní funkce (Kurt Gödel, 1934)
Turingův stroj (Alan Turing, 1936)
lambda kalkul (Alonzo Church, 1941)
RAM počítač

Popis a zápis algoritmu

slovní popis v přirozeném jazyce
pseudokód
program (zjednodušené konstrukce programovacího jazyka)

Největší společný dělitel

X, Y – dvě kladná celá čísla $\rightarrow$ určit největší společný dělitel NSD(X, Y)

Algoritmy:

1. NSD(X, Y) = největší z celých čísel od 1 do *min(X*, Y),
které je dělitelem obou čísel X a Y

a) postupně zkoušet čísla od 1 do *min(X*, Y),
průběžně si pamatovat posledního nalezeného společného dělitele

b) postupně zkoušet čísla v pořadí od *min(X*, Y) dolů k 1
až do nalezení prvního společného dělitele

2. Určit prvočíselné rozklady čísel X a Y
– jejich maximální společná část určuje NSD(X, Y)

Např. 396 = $\underline{2.2.3.3}$.11, 324 = $\underline{2.2.3.3}$.3.3

$\rightarrow$ NSD(396, 324) = 2.2.3.3 = 36

3. **Eukleidův algoritmus**
Eukleidés: Základy (řecky Stoicheia, 13 knih), cca 300 př.n.l.

Idea: když X < Y NSD(X, Y) = NSD(X, Y-X)
když X > Y NSD(X, Y) = NSD(X-Y ,Y)
když X = Y NSD(X, Y) = X

Příklad:

NSD(396, 324) =
NSD(72, 324) =
NSD(72, 252) =
NSD(72, 180) =
NSD(72, 108) =
NSD(72, 36) =
NSD(36, 36) = 36

Algoritmus (pro kladná celá čísla X, Y):

dokud X ≠ Y dělej
    od většího z čísel X, Y odečti menší z čísel X, Y

```
while x != y:
    if x > y:
        x -= y
    else:
        y -= x
print(x)
```

Možnost urychlení (pro některé vstupní hodnoty):
místo odčítání použít zbytek po celočíselném dělení

```
while y > 0:
    z = x % y
    x = y
    y = z
print(x)
```

Program funguje i v případě X < Y, jenom vykoná o jednu iteraci více a při první iteraci se hodnoty X, Y navzájem prohodí.

Jiný zápis téhož postupu:

```
while y > 0:
    x, y = y, x % y
print(x)
```

Správnost Eukleidova algoritmu

1. Konečnost

= výpočet pro jakákoliv vstupní data skončí

- na začátku výpočtu i stále v jeho průběhu je X > 0, Y > 0

- v každém kroku výpočtu se hodnota X+Y sníží alespoň o 1

  → nejpozději po X+Y krocích výpočet skončí, je tedy konečný

2. Parciální (částečná) správnost

= když výpočet skončí, vydá správný výsledek

- pro X = Y zjevně platí NSD(X, Y) = X

- ukážeme, že pro X > Y platí NSD(X, Y) = NSD(X-Y, Y)

Důkaz sporem:
Nechť N = NSD(X, Y), tedy N dělí X a zároveň N dělí Y.
Proto také N dělí X-Y a je tedy N společným dělitelem X-Y a Y.
Pokud by neplatilo, že N = NSD(X-Y, Y), musí existovat A > 1 tak, že N.A = NSD(X-Y, Y).
Tedy N.A dělí X-Y i Y, takže N.A dělí i jejich součet, což je X.
Jelikož N.A dělí Y a zároveň N.A dělí X, je N.A společným dělitelem čísel X, Y, což je spor s předpokladem, že N = NSD(X, Y).
Proto skutečně N = NSD(X-Y, Y).

Problém stabilních manželství

N mužů a N žen $\rightarrow$ chceme vytvořit N párů
- existuje $N!$ způsobů, jak lze sestavit páry

Každý muž a každá žena má svůj seznam preferencí – obsahuje všechny ženy resp. muže v pořadí od „nejlepší“ po „nejhorší“.

Hledáme **stabilní** párování = neexistuje dvojice muž – žena, kteří spolu nejsou v páru a přitom se **oba navzájem** preferují před svými současnými partnery (tzn. oba by si ze svého hlediska polepšili, kdyby se dali dohromady a opustili svého současného partnera).

Otázky:
Lze vždy nalézt nějaké stabilní párování?
Je stabilní párování určeno jednoznačně?
Jakým algoritmem a s jakou časovou složitostí lze úlohu vyřešit?

Příklad:
tři muži: 1, 2, 3
tři ženy: A, B, C

preference:	1	A B C	A	2 3 1
	2	B C A	B	3 1 2
	3	C A B	C	1 2 3

Stabilní párování lze vytvořit třemi způsoby (z celkových šesti možných párování):

a) 1A 2B 3C
b) 1B 2C 3A
c) 1C 2A 3B

→ stabilní párování tedy **nemusí být určeno jednoznačně**

Ostatní párování jsou nestabilní, např. 1A 2C 3B
- dvojice 3A je zde zdrojem nestability (muž 3 by chtěl raději ženu A než svoji současnou ženu B, rovněž žena A by chtěla raději muže 3 než svého současného muže 1).

Primitivní řešení hrubou silou (zkoušením všech možností):

- postupně sestavíme všechna možná párování
- pro každé z nich ověříme, zda je stabilní,
  tzn. testujeme každou dvojici muž-žena, která není v párování,
  zda je zdrojem nestability

Časová složitost:
- všech párování je $N!$
- pro každé z nich testujeme *řádově* N^2 dvojic muž-žena
- test jedné takové dvojice vyžaduje provést *řádově* N operací
  (průchod preferenčních seznamů délky N)

→ celkem *řádově* $N^3.N!$ operací

(upřesníme později)

Orientační doba výpočtu na počítači:
- při rychlosti procesoru přibližně 10^9 operací za sekundu

N	*počet operací $N^3.N!$*	*doba výpočtu*
10	$4 . 10^9$	jednotky sekund
12	$8 . 10^{11}$	desítky minut
14	$2 . 10^{14}$	desítky hodin
16	$8 . 10^{16}$	jednotky roků
18	$4 . 10^{19}$	tisíce roků
20	$2 . 10^{22}$	statisíce roků
…		
30	$7 . 10^{36}$	řádově 10^{22} roků

Stabilní párování **vždy existuje** (už víme, že jich může být více).
Dokážeme konstruktivně – popíšeme algoritmus, který ho nalezne.

Algoritmus Gale – Shapley:

- „pánská volenka“
- každý muž postupně nabízí sňatek jednotlivým ženám v pořadí podle svého seznamu
- volná žena nabídku vždy přijme
- zadaná žena nabídku porovná podle svého seznamu s partnerem, kterého zatím má – buď ji přijme, nebo odmítne
- proces se opakuje, dokud nevznikne *N* párů
- z více existujících stabilních párování najde to, které je nejlepší pro muže

Analogicky lze postupovat z druhé strany jako „dámská volenka“.

všechny osoby označit jako volné

dokud existuje volný muž, opakovaně prováděj:

M = libovolný volný muž

Z = první žena v seznamu muže M, které M dosud nenabízel sňatek

jestliže Z je volná,

tak Z přijme nabídku a zasnoubí se s M

jinak jestliže Z preferuje svého současného partnera před M,

tak Z odmítne nabídku a M zůstane volný

jinak

Z opustí svého současného partnera, ten se tím stane volným

Z se zasnoubí s mužem M

nakonec všechny zasnoubené páry uzavřou manželství

Konečnost:
- každá iterace cyklu = jedna nabídka k sňatku
- žádný muž nenabízí sňatek téže ženě dvakrát
  → cyklus se provede nejvýše N^2-krát

Úplnost párování:
- jakmile je žena jednou zadána, už nikdy nebude volná
  (může jedině změnit partnera)
- žena může odmítnout nabídku, jen když je zadána
- pokud by byl některý muž odmítnut i poslední ženou ze svého seznamu preferencí, muselo by v tu chvíli být zadáno všech N žen, což je spor, když aspoň tento muž je volný

→ na konci výpočtu tedy skutečně dostaneme N párů

Stabilita:

- nechť muž M je po skončení výpočtu zasnouben se ženou Z, ale více preferuje ženu Y
- M má tedy Y na svém seznamu před Z
  → nabízel jí manželství, ale byl odmítnut,
  neboť Y dostala (dříve nebo později) pro ni výhodnější nabídku
- tedy Y nepreferuje muže M před svým manželem
  → dvojice M, Y není zdrojem nestability

Časová složitost:

- cyklus se provede nejvýše N^2-krát
- příkazy v těle cyklu lze vykonat v konstantním čase (při použití vhodných datových struktur)

→ celkem řádově N^2 operací
tzn. časová složitost kvadratická vzhledem k N
časová složitost lineární vzhledem k délce vstupu