Přednáška 1

Organizace Kurzu

www stránky: www.uamt.feec.vutbr.cz/~richter/vyuka/BPCALD

Úvod – terminologie, motivace

Jednoduché algoritmy

Úvod – terminologie, motivace

Terminologie se může v různých zdrojích různit. Proto je potřebné přistupovat při studiu z různých zdrojů s opatrností.

Tento kurz si dává za cíl seznámit studenty s nejběžnějšími algoritmy.

Jsou prezentovány jednodušší formy algoritmů – složitější, optimalizované formy jsou většinou nad časové možnosti tohoto kurzu (z hlediska srozumitelného vysvětlení a pochopení, i když se jedná o „jednoduchý“ vzorec nebo návod).

Pomocí seznámení s algoritmy a jejich realizací na cvičeních by mělo dojít k zažití programovacích postupů a technik.

Vybrané algoritmy jsou uvedeny k realizaci na cvičeních. Slouží také k procvičení programování v jazyce C. Proto jsou v tomto textu také uvedeny některé vlastnosti jazyka C, které pokládám za užitečné znát. Zároveň je text nutno chápat v kontextu jazyka C (pokud například tvrdím, že není datový typ množina, potom to znamená, že nepatří mezi základní datové typy v C, ale v jiném jazyce být může, stejně tak jako je možné ho v C naprogramovat).

Jako zdroj pro studium je možné používat uvedená hesla na wikipedii.

Algoritmy

je návod, postup řešení

Slouží ke zpracování informace

- rychlost
- spolehlivost

Algoritmy fungují obecně pro různá data, možnost opakovat postupy, a tím urychlit vývoj

Abstrakce
- pracuje se s daty, jejichž původní reprezentace není podstatná (hrušky, jablka, rohlíky, šrouby …)
- umožňuje jednotný přístup bez ohledu na způsob pořízení a původní význam dat
- abstrakce od určitého stupně – výběr dat pro jednotné/standardní zpracování je závislý na problému, možných vstupech a požadovaných výstupech

Algoritmy

Požadavky na algoritmy

- konečnost
- obecnost, univerzálnost – počítá s proměnnými vstupy (tj. není pouze jednorázově použitelný)
- výstup – má konkrétní výsledek, cíl, postup
- elementárnost, determinovanost – skládá se z konečného počtu definovaných operací

Vede k:
- zobecnění, univerzálnosti – zachování pouze podstatného
- znovupoužitelnosti
- snadnější hledání a odstranění chyb

Postup návrhu

- Rozdělováním algoritmu na jednodušší - shora dolů
- Spojováním dílčích řešení – zdola nahoru

Volba reprezentace dat

komplexní číslo – složkový tvar (x,y), vektorový tvar (Amp, fáze)

- ovlivňuje složitost řešení (počet a složitost funkcí)
- výhody a nevýhody – sčítání, násobení, (od)mocnění … vhodnost jiných reprezentací
- nelze určit obecně, lze zvolit pro část řešení
- Pozice v 2D prostoru obecně – dvě reálné souřadnice (lépe ve struktuře)
- 2D struktura typu mřížka (šachovnice, skladové pozice, dlaždičky) – celočíselná reprezentace

Typy algoritmů

- rekurze
- pravděpodobnost – využívají náhody pro řešení (náhodný výběr – vstupů, postupů, směrů řešení), každé řešení může být jiné, při několika řešeních je možné získat střední řešení, odchylky,
- genetické – hledají řešení pomocí mutací a křížení
- heuristické – hledají přibližné řešení, algoritmy využívající heuristiku (postup kdy není znám algoritmus nebo přesná metoda). Staví na odhadu, zkušenosti (např. metoda pokus omyl, nepřesnost řešení je kompenzována za rychlostí)

způsob řešení

- (brutální) hrubá síla (brute force) – zkusit všechny možnosti, všechny kombinace a vyberou se ty vhodné (například hledání hesel); jednoduchý na vymyšlení a implementaci; pomalé řešení; vždy najde řešení, zrychlení nalezení řešení, postup výběru parametrů pro rychlejší eliminace (ne)vhodných
- znalostní řešení – využívá znalosti problému a jeho pravidel/vlastností pro optimalizace – vyloučení nebo přijetí kandidátů (lze vyloučit naráz celé množiny a ne pouze jedno řešení), neprochází celý vstupní prostor
- pravděpodobnostní a genetické – hledají řešení pomocí náhodně generovaných podmnožin a jejich úprav

Jednoduché algoritmy

Rozklad na prvočísla.
Nejmenší společný násobek.
Největší společný dělitel
Výpočet odmocniny
Vlk, koza, zelí;
misionáři a kanibalové

Společní dělitelé, Nejmenší společný násobek a největší společný dělitel

Největší společný dělitel (greatest common divisor)

- největší číslo, které bezezbytku dělí oba parametry dělení
- krácení zlomků
- dává smysl pro celočíselné parametry
- např. 1 je dělitelná pouze sama sebou (její uvažování jako dělitele nedává smysl)
  čísla 12 a 21 mají společného dělitele 3 (jediný dělitel)
  čísla 12 a 24 mají společného dělitele 12 (ale také 2, 3, 4, 6)
  čísla 23 a 529 mají jediného společného dělitele 23 (529 = 23*23)
-

Hrubá síla: zkusit vše od začátku do konce

Vylepšení: maximum je polovina (?) nebo odmocnina (?)

Je lepší začít od největšího nebo od nejmenšího čísla?

Euklidův algoritmus

- využívá vlastnosti, že po odečtení menšího dělitele od většího zůstane největší společný dělitel stejný (představu dává grafické řešení – násobení je plocha obdélníku).
  `NSD(u,v) = NSD(v, u – qv)` – q je celočíselný násobek, u,v jsou vstupní celá čísla
- Algoritmus je obecný, dá se použít např. i pro polynomy. Definice algoritmu:
  `dělitel = dělenec * podíl  + zbytek;`
  pro další krok `dělenec -> dělitel; zbytek -> dělenec`
  ukončení pro `dělitel = 0`
  dělitel je „větší“ (ve smyslu vhodné normy daného typu) než dělenec.
  Pozn.: algoritmus bude (např. pro celá čísla, polynomy) fungovat i bez splnění poslední podmínky (u>v). Řešení však bude delší. (u=0.v + u; a v následujícím kroku u<->v)
- Násobení je obdélník. Společný násobek dává čtvercový rastr tohoto obdélníku. Pokud odečteme kratší stranu (která je v rastru) od delší, společný násobitel se nezmění (rastr zůstane)
- ```
  vstup je u,v
  dokud v není nula
      r = zbytek po celočíselném dělení u / v
      u = v
      v = r
  výsledek je v u
  ```
  ```

Nejmenší společný násobek (least common multiple)

- převedení zlomku na společného jmenovatele C = a. jm1 = b . jm2
- např. 5 a 10 = 10 (a=2, b = 1)
  6 a 4 = 12 (a = 2 , b = 3)
- Pomocí největšího společného dělitele
  NSN(a,b) = a . b / NSD(a,b)

Rozklad na prvočísla (dále jen rozklad)

- hrubá síla – dělení všemi čísly; vylepšení – dělení prvočísly
- patří k náročným řešením zvláště pro násobení velkých prvočísel (využití u kryptografie), smysl má pro čísla, jež jsou násobky menších čísel

```
vstupem je číslo u
v = 2
dokud je u různé od jedné
    pokud je u dělitelné bezezbytku v,
            vlož v do seznamu dělitelů
            u = u / v
    jinak
            zvyš v o jedna
v seznamu jsou všichni dělitelé (někteří mohou být i vícekrát)
```

Rozklad na prvočísla algoritmus

- rozložená čísla lze použít ke stanovení NSD a NSN
  NSD – všechna čísla, která jsou společná oběma rozkladům
  NSN – všechna čísla, která jsou v rozkladech (společná pouze jednou)
- 5 a 10 mají rozklady 5 a 2;5 ; NSD je tedy 5; NSN 2.5= 10
  6 a 4 mají rozklady 2;3 a 2;2; NSD je tedy 2; NSN 2 . 2 . 3 = 12
- algoritmus NSD

```
vstupem jsou dva seznamy u a v seřazené podle velikosti
nsd = 1
nastav se na počátek seznamů
opakuj, dokud jsou v obou seznamech čísla
      pokud jsou čísla v obou seznamech stejná
            nsd = nsd * číslo v seznamu
            posuň se v obou seznamech na další číslo
      jinak
            posuň se v seznamu s menší hodnotou na další číslo
v nsd je výsledek
```

Využití rozkladu na prvočísla pro výpočet NSD a NSN

- rozložená čísla lze použít ke stanovení NSD a NSN
  NSD – použije všechna čísla, která jsou společná oběma rozkladům
  NSN – použije všechna čísla, která jsou v rozkladech (společná pouze jednou)
- 5 a 10 mají rozklady 5 a 2;5 ; NSD je tedy 5; NSN 2.5= 10
  6 a 4 mají rozklady 2;3 a 2;2; NSD je tedy 2; NSN 2 . 2 . 3 = 12

algoritmus NSN

```
vstupem jsou dva seznamy u a v seřazené podle velikosti
nsn = 1
nastav se na počátek seznamů
opakuj, dokud jsou v obou seznamech čísla
      pokud jsou čísla v obou seznamech stejná
            nsn = nsn * číslo v seznamu
            posuň se v obou seznamech na další číslo
      jinak
            nsn = nsn * číslo ze seznamu s menší hodnotou
            posuň se v seznamu s menší hodnotou na další číslo
se seznamem, ve kterém zbyly čísla, opakuj
       nsn = nsn * číslo ze seznamu
       posuň se v seznamu na další pozici
v nsn je výsledek
```

Příklad řešení pro čísla 630 a 2772

Euklidův algoritmus - NSD

r	u	v
	2772	630
252	630	252
126	252	126
0	**126**	0

nejmenší společný násobek = číslo . číslo / NSD = 2772 . 630 / 126 = 13860

co	číslo								Poznámka
Rozklad	630		2	3	3	5	7		
Rozklad	2772	2	2	3	3		7	11	stejná čísla pod sebe
NSD = „průnik“	126		2	3	3		7		společná čísla z rozkladů
NSN = „sjednocení“	13860	2	2	3	3	5	7	11	všechna čísla z rozkladů

vlk, koza, zelí

Má se převézt přes řeku po jednom. Nesmí však zůstat na jedné straně nebezpečná dvojice.
Nepovolené kombinace [v k], [k z]
algoritmus
Opakuj do vyřešení (všichni budou vpravo)
vezmi jednoho/dalšího zleva (ne toho, který se právě vrátil zprava)
pokud zůstala nepovolená kombinace, vrať se k předchozímu bodu
vezmi jednoho/dalšího zprava (přednostně nikoho; ne toho, který se právě přijel zleva)
pokud zůstala nepovolená kombinace, vrať se k předchozímu bodu.

Pro tuto úlohu bude fungovat. Obecně by mělo být přidáno:
pokud vyčerpáš všechny kombinace pro převoz a nenajdeš řešení, vrať se do stavu před předchozím převozem.
Obecně by byla nutná kontrola, zda jsme v podobném stavu již nebyli - zacyklení

Misionáři a lidožrouti

Misionáři jsou rovnocenní. Lidožrouti jsou rovnocenní. Tj. nezáleží u nich na pořadí.

Na břehu jsou tři lidožrouti a tři misionáři. Mají se dostat na druhý břeh.

Loďka má dvě místa (jede tedy jeden nebo dva. Posádka může být smíšená, nebo stejnorodá)

řešení např. stromem (v každém uzlu pět možnosti převozu)

Malé množství kombinací a nutnost pamatovat si již dosažených stavů (eliminace cyklů) – můžu si pamatovat stav ML (například 32 – tři misionáři a dva lidožrouti). Stav je vázán i na polohu lodi (proto musím např. 320 pro loď vlevo a 321 pro loď vpravo, anebo dvě nezávislé tabulky minulých stavů).

Výpočet odmocniny

- pomocí iterativní metody – n-tá odmocnina z A. Zvolí se počáteční odhad x a iteruje se tak dlouho, dokud se nedosáhne požadované přesnosti:

$$x_{k+1} = \frac{1}{n}\left((n-1)x_k + \frac{A}{x_k^{n-1}}\right)$$

- druhá odmocnina pomocí násobení
  odmocnina daného čísla x, přesnost p

```
rad = 1
y = 0
zvyšuj rad v násobcích 10, dokud nenajdeš rad * rad > x
opakuj dokud abs(y * y – x) > p
   sniž rad na rad / 10
   Opakuj y = y + rad dokud nenajdeš y * y > x
   y = y – rad
```

pomocí y = x / y x je vstup; y odmocnina

```
- vezmi nějakou hodnotu y a vypočti x/y. Pokud je výsledek
dostatečně blízký y skonči
- odhad mezí řešení z nejvyššího řádu. Rozložíme na  xx * 10^(2*n)
a pro xx určíme nejbližší a nejvyšší celou mocninu
- vezmi nový odhad hodnoty y (např. (y+x/y)/2)
```

Newtonova metoda

- iterativní metoda využívající geometrické interpretace a derivací

$f(x) = x^2 - a = 0$ Pokud rovnici vyřešíme, a se rovná odmocnině z x

$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$ algoritmus končí pro f(x) = 0, nebo dostatečně blízko

$x_{i+1} = x_i - \frac{x_i^2 - a}{2x_i}$

Fast inverse square root - výpočet $\frac{1}{\sqrt{x}}$ pomocí znalosti architektury výpočetní jednotky

původně pro normalizaci vektorů v grafice

vychází ze znalosti architektury a bitů v typu float 32 (pro 64 bitů 0x5fe6eb50c7b537a9)

principem je převod na int (log2(x)) a následný výpočet na log2($\frac{1}{\sqrt{x}}$)) a převod zpět

```
float Q_rsqrt( float number )
// zdroj wikipedia – fast inverse square root
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;

    x2 = number * 0.5F;
    y  = number;
    i  = * ( long * ) &y; // evil floating point bit level hacking
    i  = 0x5f3759df - ( i >> 1 );               // what the fxxk?
    y  = * ( float * ) &i;
    y  = y * ( threehalfs - ( x2 * y * y ) );   // 1st iteration
// doladění Newtonovým algoritmem
//  y  = y * ( threehalfs - ( x2 * y * y ) );
// 2nd iteration, this can be removed

    return y;}
```

Výpočet Pi

- různé metody – geometrické, matematické, náhodné (statistické),

- Experimentálně: zvolit vhodný průměr, vytvořit kruh/kružnici a změřit délku obvodu
- Matematicky: geometrické metody výpočet obvodu opsaného/vepsaného mnohoúhelníku – poměr délky kružnice k průměru Pi.d = O – geometrické určení – vypočítat obvod pomocí trojúhelníků
- nekonečný rozvoj. Různé metody lišící se dobou vzniku a kvalitou – rychlost konvergence (jak rychle se blíží konečnému řešení), dosažitelná přesnost v omezeném čase, většinou algoritmy, které nejsou jednoduše pochopitelné/popsatelné

Pomocí náhodných jevů

- obsah – srovnání obsahu kruhu a čtverce – do obou umisťujeme body a srovnáváme počet bodů v kružnici a čtverci – počet bodů je úměrný ploše. umisťování bodů je možné náhodně (monte carlo) nebo v rastru
- hod jehlou do čarového rastru

Porovnání obsahů pro rastr

- rastr – výhoda – snadno generovatelné, univerzální.
- Zjednodušení pomocí symetrie – 8 stejných sekcí
  středový bod patří do … sekcí
  úhlopříčka patří do … sekcí
  osa strany patří do … sekcí
- Počítat celé bloky – hranicí je oblouk kružnice