Cvičení Programování I

Cvičící: **Pavel Surynek, KTIML**
surynek@ktiml.mff.cuni.cz
http://ktiml.mff.cuni.cz/~surynek
Semestr: **Zima 2007/2008**
Kroužek: **Matematika/53**
Rozvrh: **Středa 12:20-13:50 (učebna K7)**

Stručné poznámky ke cvičení z 24.10.2007

1. Organizační záležitosti. Za domácí úkol byly úlohy **potravní řetězec**, **podivný sud** a **zase sirky**.

2. Konečnost algoritmu. Konečnost algoritmů se ověřuje tak, že jednotlivé stavy algoritmu ohodnotíme - například přirozenými nebo nezápornými reálnými čísly a snažíme se dokázat, že toto ohodnocení v průběhu algoritmu klesá.

a) Dokažte, že Euklidův algoritmus na nalezení největšího společného dělitele dvou čísel je konečný.

b) Uvažujme dva algoritmy, jejichž stavy jsou ohodnoceny reálnými čísly. Oba algoritmy končí při ohodnocení 0. Pro ohodnocení stavů $h : S \rightarrow R$, kde S je množina stavů, prvního algoritmu splatí, že

$\exists \delta \in R, \delta > 0$, že $\forall i \;\; h(stav(i)) \geq h(stav(i+1)) + \delta$, kde $stav(i)$ je stav algoritmu v kroku i. Pro druhý algoritmus platí, že

$\forall i \; \exists \delta \in R, \delta > 0$, že $h(stav(i)) \geq h(stav(i+1)) + \delta$, kde $stav(i)$ je stav algoritmu v kroku i. Vyšetřete konečnost obou algoritmů.

Řešení. a) Vsuvka: Jak pracuje Euklidův algoritmus? Mějme čísla n a m, hledáme $NSD(m,n)$. Platí, že $NSD(m,n) = NSD(m,n-m)$, přičemž $n > m$. Dále platí, že $NSD(m,m) = m$. Algoritmus pracuje tak, že v každém kroku algoritmu hledáme $NSD(m,n)$. Když jsou n a m různá, od většího z n a m odečteme menší n a , jinak už máme $NSD(m,n)$ (tj. když jsou n a m stejná).

Stavem algoritmu jsou čísla n a m. Nabízí se tedy ohodnocení $h((m,n)) = m + n$. Přirozená čísla mají nejmenší prvek, ohodnocení se v každém kroku zmenší aspoň o 1, algoritmus tedy skončí.

b) První algoritmus jistě skončí. Nechť je na začátku běhu algoritmu ohodnocení počátečního stavu $h(stav(0)) = K$. Jistě platí, že nejvýše po $\frac{K}{\delta}$ krocích algoritmus skončí. Ohodnocení se snižuje „stejnoměrně“ (viz. matematická analýza - stejnoměrná konvergence, spojitost atd.). Druhý algoritmus obecně nemusí skončit. Nechť například ohodnocení počátečního stavu je $h(stav(0)) = 2$ a nechť $\delta_i = \frac{1}{2^i}$ (δ je v každém kroku jiné, proto ten index). V takovém případě nemůžeme dokázat, že ohodnocení někdy klesne na 0, protože i po nekonečně mnoha krocích klesne ohodnocení aspoň o 1 ($\sum_{i=1}^{\infty} \delta_i = \sum_{i=1}^{\infty} \frac{1}{2^i} = 1$), což nemusí stačit.

3. Jak dlouho může běžet program. Je možné pro libovolné $t > 0$ napsat program, který běží více než t vteřin a pak zastaví. Předpokládáme přitom, že počítač, na kterém program běží, splňuje běžná fyzikální omezení, tj. například provedení elementární operace trvá určitou dobu, velikost paměti je

konečná atd. Na druhou stranu si situaci trochu idealizujeme - předpokládáme, že počítač může bez poruchy fungovat libovolně dlouho.

Řešení. Jestliže má počítač konečnou paměť, řekněme k bitů, pak existuje 2^k různých stavů, ve kterých se počítač (program) může nacházet. Elementární operace převádí počítač (program) z jednoho stavu do stavu následujícího. Jestliže se stane, že počítač (program) přejde do některého ze stavů, ve kterých již byl, pak můžeme říct, že program je zacyklený a poběží věčně. Nechť provedení elementární operace trvá t_o vteřin. Když program běží déle než $t_o 2^k$, pak se jistě do některého ze stavů dostal podruhé, je tedy zacyklen a poběží už navěky. Požadujeme-li tedy, aby program běžel déle než $t_o 2^k$ vteřin a pak zastavil, není tomuto požadavku možno vyhovět.

4. Sedlový bod. Je dána čtvercová matice nad celými čísly. Navrhněte postup jak co nejrychleji v této matici nalézt sedlový bod. Sedlový bod je místo v matici, které obsahuje největší hodnotu ve svém řádku a nejmenší hodnotu ve svém sloupci nebo nejmenší hodnotu ve svém řádku a největší hodnotu ve svém sloupci. Snažte se minimalizovat počet kroků, přičemž za krok považujeme každé podívání se na políčko matice. Nápověda: předvýpočet.

Řešení. Neefektivně bychom mohli postupovat tak, že pro každý prvek matice ověříme zda je sedlovým bodem. K ověření, zda dané místo matice obsahuje sedlový bod lze provést prohlédnutím všech prvků v odpovídajícím řádku respektive sloupci. Toto prohlédnutí spotřebuje zhruba $2N$ kroků, když pracujeme s maticí typu $N \times N$. Hledání sedlových bodů v celé matici tedy spotřebuje $2N^3$, protože musíme provést uvedené ověření pro každé místo v matici, kterých je N^2.

Efektivněji můžeme postupovat tak, že pro každý řádek resp. sloupec nalezneme místo (místa), která obsahují největší resp. nejmenší prvek. Nalezení nejmenšího prvku v řádku spotřebuje N kroků, totéž pro největší prvek v řádku a podobně i pro sloupce. Na každý řádek tedy připadá $2N$ kroků a na každý sloupec připadá také $2N$ kroků. Celkem $2N^2$ kroků. Pro každé místo v matici pak ověříme, zda odpovídá místu největšího prvku v daném řádku a nejmenšímu prvku v daném sloupci nebo naopak tj. místu nejmenšího prvku v daném řádku a největšímu prvku v daném sloupci. Toto ověření spotřebuje zhruba 4 kroky. Celkem tedy máme $2N^2 + 4N^2 = 6N^2$. To je lepší než $2N^3$.

5. Vyhledávání s setříděné posloupnosti. Je dána setříděná posloupnost N přirozených čísel (například: 5, 20, 27, 35, 60, 66, 91). Navrhněte algoritmus, který pro dané přirozené číslo x (například: 26) a zmiňovanou posloupnost a rozhodne, zda se zadané přirozené číslo nachází v zadané posloupnosti (v uvedeném příkladě bude odpověď ne). Algoritmus může provádět operace porovnání a v každém kroku se „může podívat“ na jedno číslo na libovolné pozici v posloupnosti. Snažte se minimalizovat počet kroků algoritmu (porovnání, podívání se, ... se považuje za krok).

Řešení. Úkol je nalézt zadané číslo v setříděné posloupnosti mezi pozicí 1 a pozicí N (včetně). Můžeme se na situaci dívat obecněji: úkolem je nalézt zadané číslo v setříděné posloupnosti mezi pozicí d a pozicí h (včetně), kde $d \le h$. Když $d = h$, ověříme číslo na pozici d a odpovíme (na toto spotřebujeme jeden krok). Když $d < h$, podíváme se na číslo na pozici $\lceil (d+h)/2 \rceil$. Když je rovno hledanému x, odpovídáme ano, když je menší než x, řešíme úlohu stejného typu pro hledání mezi pozicemi $\lceil (d+h)/2 \rceil + 1$ a h, a nakonec když je větší než x, řešíme úlohu stejného typu pro hledání mezi pozicemi d a $\lceil (d+h)/2 \rceil - 1$ (spotřebujeme jeden krok a kroky na vyřešení menší úlohy stejného typu).

Počet kroků na vyřešení úlohy velikosti N označíme jako $T(N)$, kde velikost úlohy je počet prvků mezi pozicemi, mezi kterými hledáme. Platí rovnosti $T(1) = 1$ a $T(N) = T(N/2) + 1$. Z těchto rovností lze odvodit, že $T(N) = \lceil \log_2 N \rceil$. Nahlédneme indukcí. Pro malá N ověříme. Pak, když $T(n) = \lceil \log_2 n \rceil$ pro $(\forall n)(n < N)$, odvodíme $T(N) = T(N/2) + 1 = \lceil \log_2 (N/2) \rceil + 1 =$

$= \lceil \log_2 N - \log_2 2 \rceil + 1 = \lceil \log_2 N - 1 \rceil + 1 = \lceil \log_2 N \rceil - 1 + 1 = \lceil \log_2 N \rceil$. Indukční předpoklad se použije pro $T(N/2)$, neboť $N/2 < N$.

6. Pigeonhole principle (každý holub má svou přihrádku). Booleovská formule v konjunktivně disjunktivním tvaru je konjunkce disjunkcí. Například $(x_1 \vee x_3 \vee \neg x_5) \wedge (x_2 \vee \neg x_3 \vee x_6) \wedge (\neg x_2 \vee x_3)$, kde x_i jsou Booleovské proměnné, tj. proměnné, které mohou nabývat hodnot *pravda* či *nepravda*. Pro formule tohoto tvaru se často používá zjednodušený zápis kdy pouze vypíšeme indexy proměnných v jednotlivých disjunkcích s případným znaménkem mínus, pokud jde o negaci proměnné. Formule $(x_1 \vee x_3 \vee \neg x_5) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee x_3)$ by se tedy dala napsat jako:

1 3 -5
2 -3 4
-2 3

Máme-li formuli, zajímavou otázkou je, zda pro ni existuje ohodnocení proměnných takové, že celkově je tato formule pravdivá. Formule $(x_1 \vee x_3 \vee \neg x_5) \wedge (x_2 \vee \neg x_3 \vee x_6) \wedge (\neg x_2 \vee x_3)$ má pravdivé ohodnocení $x_1 = pravda$, $x_2 = pravda$, $x_3 = pravda$ a ostatní proměnné libovolně. Pokuste se zjistit, zda existuje pravdivé ohodnocení pro následující formuli. Navrhněte algoritmus, který by provedl práci za Vás.

-1 -7
-1 -13
-1 -19
-1 -25
-1 -31
-1 -37
-7 -13
-7 -19
-7 -25
-7 -31
-7 -37
-13 -19
-13 -25
-13 -31
-13 -37
-19 -25
-19 -31
-19 -37
-25 -31
-25 -37
-31 -37
-2 -8
-2 -14
-2 -20
-2 -26
-2 -32
-2 -38
-8 -14
-8 -20
-8 -26
-8 -32
-8 -38
-14 -20
-14 -26
-14 -32
-14 -38
-20 -26
-20 -32
-20 -38
-26 -32
-26 -38
-32 -38
-3 -9
-3 -15
-3 -21
-3 -27
-3 -33
-3 -39

-9 -15
-9 -21
-9 -27
-9 -33
-9 -39
-15 -21
-15 -27
-15 -33
-15 -39
-21 -27
-21 -33
-21 -39
-27 -33
-27 -39
-33 -39
-4 -10
-4 -16
-4 -22
-4 -28
-4 -34
-4 -40
-10 -16
-10 -22
-10 -28
-10 -34
-10 -40
-16 -22
-16 -28
-16 -34
-16 -40
-22 -28
-22 -34
-22 -40
-28 -34
-28 -40
-34 -40
-5 -11
-5 -17
-5 -23
-5 -29
-5 -35
-5 -41
-11 -17
-11 -23
-11 -29
-11 -35
-11 -41
-17 -23
-17 -29
-17 -35
-17 -41
-23 -29
-23 -35
-23 -41
-29 -35
-29 -41
-35 -41
-6 -12
-6 -18
-6 -24
-6 -30
-6 -36
-6 -42
-12 -18
-12 -24
-12 -30
-12 -36
-12 -42
-18 -24
-18 -30
-18 -36
-18 -42
-24 -30
-24 -36
-24 -42
-30 -36
-30 -42
-36 -42
6 5 4 3 2 1

12	11	10	9	8	7
18	17	16	15	14	13
24	23	22	21	20	19
30	29	28	27	26	25
36	35	34	33	32	31
42	41	40	39	38	37

Řešení. Lze si všimnout, že výrok x_1 lze interpretovat jako výrok „holub číslo 1 v přihrádce 1“. Výrok x_2 lze interpretovat jako výrok „holub číslo 1 v přihrádce 2“, atd.. Výrok x_7 lze interpretovat jako „holub číslo 2 v přihrádce 1“. Výrok x_8 lze interpretovat jako „holub číslo 2 v přihrádce 2“, a tak dále pro celkem 7 holubů a 6 přihrádek. Disjunkce se dvěma členy interpretujeme obdobně: čili první disjunkce se dvěma členy by se interpretovala neplatí „holub číslo 1 v přihrádce 1“ nebo neplatí „holub číslo 2 v přihrádce 1“, jinak řečeno nesmí nastat situace, že „holub číslo 1 a zároveň holub číslo 2 v přihrádce 1“. Prohlédneme-li si v tomto smyslu všechny dvoučlenné disjunkce můžeme říci, že žádní dva holubi nesmí být v žádné přihrádce společně (v jedné přihrádce tedy nejvýše jeden holub).

Disjunkce na konci obsahující šest členů říkají, že každý holub má být v některé z přihrádek. Celá formule tedy požaduje: dejte všechny holuby do přihrádek a přitom v každé přihrádce smí být nejvýše jeden holub. Holubů je 7, ale přihrádek jen 6. Zadaná formule tedy nemá žádné ohodnocení svých proměnných, které by ji splňovalo.