KSI 2012
Úloha 2-4: Farmář Gusta

Jan Horáček
Gymnázium, Brno, Vídeňská 47; jan.horacek@seznam.cz

18. listopadu 2012

1 Úvod

Jak co nejefektivněji provést průnik n množin? To je zadání úlohy přepsané do jazyka informatiků.

U této úlohy mi hodně napovědělo téma této sady KSI - třídění. A když jsem si uvědomil, že bude zapotřebí něco třídit a že třídit umíme docela rychle, zjistil jsem, že algoritmus může být opravdu velmi rychlý.

2 Zdrojový kód

K tomuto algoritmu není přiložen žádný spustitelný kód. Níže bude slovně popsán kompletní algoritmus, kterým lze daný problém efektivně vyřešit.

3 Popis řešení

Toto řešení úlohy je založeno na seřazení jmen druhů na daných zemědělských polích a jejich porovnávání.

Následuje kompetní slovní popis algoritmu.

3.1 Vstupní data

Vstupními daty nechť jsou data uložená v datových strukturách, které program vyižívá. Jelikož úkolem je pouze vytvořit algoitmus, jejich načítání zde nebudu žádným způsobem řešit.

Nechť těmito strukturami je pole polí obsahující řetězce. Nazvěme si jej *input*. Jedná se tedy o seznam druhů pro každé zemědělské pole. Každý druh je přitom definován unikátním řetězcem.

Máme tedy n textových možin.

3.2 Parsing vstupních dat

Pro efektivní vyřešeni problému je nezbytné, aby vstupními daty nebyly řetězce, ale čísla. Tento problém řeší následující část algoritmu.

Jednou metodou je říci Gustovi, aby měl každý druh své unikátní číslo a nikoliv unikátní název, anebo, pokud chce Gusta zadávat názvy, musíme v programu převést každý název na unikátní číslo.

K převední řetězce na čísla můžeme využít několik metod, například:

1. **Znak jako char a řetězec jako pole charů**

   Velmi rychlá (lineární) metoda, která vyjadřuje každý znak (char) jeho číselnou hodnotou v tabulce ASCII a toto číslo ukládá do paměti. Každý retězec má tedy v paměti jeho ekvivalent v podobě obrovského čísla, na nějž ani ty největší inty nestačí. Pro konkrétní implemetaci by bylo zapotřebí využít speciálních knihoven.

2. **Hash řetězce**

   Hashovací funkce by musela opět vracet pouze čísla a samořejmě by se jednalo opět o obrovská čísla. Jedinou výhodou této metody je snadná impelementace v podobě zavolání funkce, kterou už možná někdo někdy napsal.

Výsledkem této funkce bude tedy pole označené jako *int_input*. Bude se opět jednat o pole polí, nýbrž né typu string, ale typu int.

Máme tedy n číselných možin.

3.3 Řazení polí: Quicksort

Vezměme tedy každé zemědělské pole a seřaďmě na něm položky (druhy rostlin) podle velikosti jejich číselných reprezentací, které jsme získali v předchozím kroku. A to nejlépe co nejryhleji, tedy quicksortem.

Celý princip quicksortu je datailně popsaný na wikipedii a já budu psát spíše o tom, jak ho modifikovat pro tuto úlohu.

Problémy algoritmu:

1. **Volba pivotu**

   Na wikipedii se detailně píše o tom, jak je pivot důležitý zejména pro časovou náročnost algoritmu.

   Já osobně bych zvolil pivota jako mediána 3 hodnot na pseudonáhodných indexech pole. Tato cesta mi přijde relativně rychlá, jednoduchá a hlavně dostaneme relevantní výsledek. Výběr mediánu ze 3 bych provedl nějakým řadícím algoritmem: zde postačí obyčejný bubble sort, ale pokud by někdo stál o zmenšení časové náročnosti, může samozřejmě implementovat libovolný řadící algoritmus.

2. **Nestabilita algoritmu**

   Pokud budeme tento algoritmus pouštět na slušných zařízeních, které odpovídají 21. století, nemělo by dojít k přeplnění zásobníku. Samozřejmě, pokud do programu vložíme opravdu velmi vysoký počet rostlin (například větší, něž int), algoritmus na tom bude špatně a asi pravděpodobně skončí na "Stack overflow". V takovém případě bych doporučoval upravit Quicksort na stabilnější, ale více prostorově náročnou verzi.

Máme tedy n vzestupně seřazených možin.

3.4 Zjištění stejných druhů

Zaveďmě pole i jako pole indexů, přičemž je toto pole vélké n prvků a na začátku jsou ve všech prvcích nuly.

Porovnejme tedy čísla nacházející se na indexech i ve všech daných n polích co do velikosti. Tedy $int\_input[0][i[0]]$, $int\_input[1][i[1]]$, $int\_input[2][i[2]]$, ..., $int\_input[n-1][i[n-1]]$

Pokud jsou všechna čísla shodná, našli jsme druh nacházející se na všech n zemědělských polích. Pokud chceme uživateli dát vědět o tom, který druh se vyskytuje na všech polích, bylo by dobré dané číslo $int\_input[0][i[0]]$ právě nyní vypsat na obrazovku (na indexu v poli i v tomto případě nazáleží). Po vypsáni inkrementujeme všechna i o 1.

Pokud jsou čísla neshodná, najdeme nejmenší z nich a provedeme $i[nejmensizempole]++$.

Celý postup opakujeme, dokud je ve všech množinách (zemědělských polích) alespoň 1 prvek.

Problém: Každého asi napadne, že pokud pan Gusta do programu zadával druhy jako řetězce, vypisovaná čísla jsou pro něj nesrozumitelná. Zde bych doporučil sjednotit pole *int_input* a *input* do jednoho pole struktur, přičemž každá struktura bude obsahovat druh slovně vyjádřený i číselně vyjádřený. Počítalo by se samozřejmě s čísly, nýbrž pokud bychom našli stejné prvky možin, vypíšeme jejich textovou reprezentaci.

4 Závěr

Při průměrných vstupních hodnotách lze říci, že časová náročnost quicksortu je $O(NlogN)$, ba i lepší, protože jsme pivota zvolili relativně sofistikovaně a přesně. Průměrná časová náročnost řazení všech polí je pak $O(n*(NlogN))$, přičemž n je počet zemědělských polí a N průměrný počet rostlin na všech polích. Časová náročnost hledání stejných prvků v seřazených množinách je lineární. Časová náročnost nahrazení řetězců čísly je lineární.

Reference

[1] Wikipedia *Quick sort*
http://cs.wikipedia.org/wiki/Quicksort