

KSI 2013/2014

Úloha 5-2: Kolekce

Jan Horáček

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

13. dubna 2014

1 Řešení

1.1 Plyšové logo

Pro tuto úlohu bych zvolil uložení řešitelů do seznamu iterovatelného přes celočíselný klíč. Konkrétně do pole. Každý prvek pole bude jeden řešitel (v C lze realizovat např. polem struktur).

Následující popis účelnosti a efektivnosti tohoto řešení předpokládá, že máme k dispozici počet účastníku - velikost pole, což obvykle máme. Označme tento počet jako *cnt*.

Náhodného účastníka vybereme pomocí vygenerování náhodného čísla v intervalu klíčů. V případě slušného céčkovského pole indexovaného od nuly je tedy náhodný index typicky $i = \text{rand()} \% \text{cnt}$, tedy v intervalu validních indexů $\langle 0, \text{cnt} - 1 \rangle$. K datům tohoto účastníka se poté dostaneme jednoduše tak, že číslo *i* použijeme jako index našeho pole.

Toto řešení je vhodné, protože stačí jednoduše vygenerovat náhodné číslo, velmi rychle přistoupit k položce pole (vyplývá z principů pole) a vrátit příslušného řešitele. Časová náročnost tohoto řešení je vzhledem k počtu účastníků konstantní! V případě pole alokovaného přesně na počet účastníků je tato struktura také paměťově efektivní. Neukládáme žádné ukazatele na další prvky, jako by tomu bylo v případě dynamických datových struktur.

1.2 Propagační materiály

Navrhuji využít multislovník, kde klíčem bude název školy. Z multislovníku dostaneme všechny klíče bez větších problémů a jelikož jsou těmito klíči unikátní názvy škol, stačí už jen zjistit adresy a poslat jedny propagační materiály na každý klíč.

Tuto datovou strukturu volím, protože dokáže velmi rychle vrátit množinu klíčů. Tento problém lze prakticky realizovat například množinou (polem, dynamickým seznamem, ...) škol, kde každá škola bude ukazovat na větší množství řešitelů. Pro vyřešení problému propagačních materiálů stačí jen vrátit množinu škol, což je teoreticky operace s konstantní časovou složitostí (stačí např. vrátit ukazatel na pole škol). Základním principem této struktury je, že mapování řešitelů na klíče (školy) provádí při každém přidání řešitele a výsledek, který požadujeme, si tak prakticky "skládá" během přidávání řešitelů.

1.3 Výtisky první sady

K získání seznamu škol bych využil naprosto totožnou strukturu, jako v předchozí kapitole. Pole klíčů (klidně iterovatelné celočíselným indexem) si ale musí pamatovat, kolik řešitelů se v dané škole nachází. Pro názornost doporučuji pročíst příklad 1.3 v programovacím pseudo-jazyce C.

```
typedef struct {
    int id, points;
    string surname, lastname;
} solver;

typedef struct {
    string name;
    int solver_cnt;
    solver[] solvers;
} school;

struct {
    int cnt;
    school[] schools;
} schools_db;

int send_letters()
{
    for (int i in schools_db.schools)
        send_letter(schools_db.schools[i].name,
                    schools_db.schools[i].solver_cnt);
}
```

Využití této struktury je, podobně jako v předchozí kapitole, motivováno lineární časovou složitostí $O(n)$, kde n je počet škol.

1.4 Plyšové logo KSI

Pro řešení tohoto problému bych zvolil naprosto totožné řešení, jako v úloze 1, s tím rozdílem, že se nebude vracet jméno řešitele, ale jeho škola. Škola je uložena ve struktuře každého jednotlivého řešitele.

Toto řešení splňuje vstupní požadavek prioritizovaného randomu, protože školy jsou v poli řešitelů zastoupeny nerovnoměrně. Je tedy pravděpodobnější, že se vybere škola, kde je více řešitelů, než škola s méně řešiteli. Celý problém jsme tak převedli na problém vybrání náhodného řešitele.

1.5 Maskot

Pro řešení tohoto problému bych využil uspořádanou množinu křestních jmen seřazenou dle abecedy, protože nám umožňuje přesně to, co potřebujeme po datové struktuře v této úloze.