KSI 2014/2015

Úloha 2-1: Vánoční akce

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

14. prosince 2014

1 Úvod

Pro efektivní řešení Karlíkova problému se v podstatě můžeme vydat dvěma cestami:

1. **hash mapa**, která nám umožní program implementovat v lineární časové složitosti, ale zato v nejhorším případě s paměťovou složitostí až $O(n)$ a
2. **řadící algoritmus**, nímž docílíme seřazení vstupní posloupnosti ve větším čase $O(n*log(n))$, ale zato není potřeba alokovat žádnou pomocnou paměť (tedy paměťová složitost je konstantní). Ze seřazené posloupnosti pak dokážeme najít unikátní čísla prostým lineárním průchodem a porovnáváním sousedních hodnot.

Při rozhodování se mezi těmito dvěma cestami jsem strávil velké množství času googlením, které mj. ukázalo, že existují i "Extremely cool"řešení, která umí problém řešit v čase $O(n)$ a paměti $O(1)$ [1]. Naneštěstí, tato řešení mnou nebyla zcela pochopena a tak jsem se k jejich implementaci neuchýlil.

Díky skutečnosti, že

1. paměťová složitost hash mapy je snesitelná,
2. paměť je v dnešní době levnější, než čas procesoru a
3. tato sada je zaměřená na datové struktury

jsem zvolil možnost 1), tedy řešit problém pomocí hash mapy.

2 Popis algoritmu

Algoritmus využívá struktury *hash mapa*, v případě Pythonu slovník, neboli *dictionary*. Tato struktura umožňuje namapovat libovolný klič na libovolnou hodnotu. V mém případě bude klíčem vždy číslo s Karlíkova seznamu a hodnotou bude číslo 1 právě když se položka v Karlíkové seznamu vykytuje lichý počet krát. Pokud se číslo v Karlíkově seznamu vyskytuje sudý počet krát, ve slovníku se nenachází. Z toho plyne, že po provedení algoritmu stačí vrátit první prvek ze slovníku, což také program dělá.

Plnění slovníku tak, aby byly splněny výše zmíněné charakteristiky pak probíhá následovně:

[1] např. zde

```
hashmap = prazdny_slovnik
for cislo in Karlikuv_seznam:
        if cislo in hashmap:
                smaz hashmap[cislo]
        else:
                hashmap[cislo] = 1
```

3 Závěr

Z vlastností struktury `dicionary` plyne časová složitost algoritmu [2] : všechny operace, které nad slovníkem provádíme, mají konstantní časovou složitost (přidání prvku, zjištění přítomnosti prvku, smazání prvku, níže pak zjištění prvního prvku). Asymptotická časová složitost algoritmu je tedy vůči počtu čísel Karlíkova seznamu lineární ($O(n)$).

Paměťová složitost je definována velikostí slovníku, která je v nejhorším případě (tedy v tom, kdy jsou všechny čísla v seznamu právě jedenkrát) lineární ($O(n)$). Reálná paměťová složitost algoritmu je menší, protože program vhodně využívá toho, že položky, které jsou v Karlíkově seznamu sudý počet krát, si nemusí pamatovat (ze seznamu je tedy maže, viz pseudokód výše).

[2] https://wiki.python.org/moin/TimeComplexity