

# Programovanie (1) v C/C++ 2024/25

## Cvičenia 9, príklad 5, bonus

### Bloom

Hašovacie funkcie sa používajú v rôznych dátových štruktúrach. V tejto úlohe naprogramujete štruktúru nazývanú Bloomov filter, ktorá udržiava informácie o množine kladných celých čísel. Prvky sú do množiny pridávané funkciou `add` a funkciou `contains` testujeme, či sa v množine daný prvok nachádza. Na rozdiel od implementácií množiny z prednášky, ktoré vždy dávali správnu odpoveď, Bloomov filter sa môže aj pomýliť. Oproti bežnej hašovacej tabuľke potrebuje ale menej pamäte, takže sa používa v situáciách, kde pracujeme s veľkými množstvami dát a občasná chyba nám nevádi alebo ju vieme ďalšími kontrolami neskôr odhaliť.

Bloomov filter má tabuľku boolovských hodnôt veľkosti  $m$ , ktorá je na začiatku vyplnená hodnotou `false`. Okrem toho má  $k$  rôznych hašovacích funkcií. My použijeme funkcie tvaru  $(a * x) \% p \% m$ , kde  $x$  je spracovávaný prvok,  $m$  je veľkosť tabuľky,  $p$  je prvočíslo a hodnota  $a$  je pre každú z  $k$  hašovacích funkcií iná.

Operácia `add` sa vykonáva tak, že vstupná hodnota  $x$  sa postupne zahašuje všetkými  $k$  funkciami, čím dostaneme  $k$  indexov do tabuľky a na všetky tieto miesta sa zapíše hodnota `true` (na niektorých z nich sa už táto hodnota mohla nachádzať). Operácia `contains` sa vykonáva tak, že vstupná hodnota sa taktiež zahašuje všetkými funkciami a ak sa na všetkých príslušných miestach tabuľky nachádzajú hodnoty `true`, funkcia vráti `true`, inak vráti `false`. Ak sme hodnotu  $x$  vložili pomocou `add`, `contains x` teda bude vracať `true`. Ak sme však  $x$  nevložili do množiny, môže sa stať, že príslušné políčka boli zmenené na `true` pri vkladaní iných prvkov a teda že `contains x` vráti nesprávnu hodnotu `true`.

Vašou úlohou je do **priloženej kostry** naprogramovať funkcie `add` a `contains`, ktoré sa budú správať podľa popisu vyššie. V kostre je už hotové načítanie a výpis. Všetky údaje súvisiace s Bloomovým filtrom sú uložené v štruktúre `table`, ich popis nájdete v komentároch. Hotová je aj hašovacia funkcia `hash`, ktorú vo vašich funkciách použite. **Nemeňte už hotové časti programu.**

Na prvom riadku vstupu sú za slovom `hash` hodnoty  $m$ ,  $k$ ,  $p$  a postupnosť  $a[0], \dots, a[k-1]$  určujúca koeficienty jednotlivých hašovacích funkcií. Potom kostra načítava príkazy `insert`, `contains` a `print`, pričom vypisuje vykonané príkazy aj ich prípadné výsledky. Funkcia `print` vypíše obsah hašovacej tabuľky ako postupnosť písmen T (`true`) a F (`false`). Na poslednom riadku vstupu je príkaz `end`.

#### Príklad vstupu:

```
hash 6 2 23 16 9
print
add 2
print
contains 2
contains 4
contains 16
add 4
print
contains 4
end
```

#### Príklad výstupu:

```
print: FFFFFFFF
add 2
print: TFFTFF
contains 2: yes
contains 4: no
contains 16: yes
add 4
print: TTFTFF
contains 4: yes
end
```

Pri vkladaní čísla 2 sa spočítajú hodnoty  $2 * 16 \% 23 \% 6$ , čo je 3 a  $2 * 9 \% 23 \% 6$ , čo je 0. Na týchto pozíciách v tabuľke sa teda nastaví hodnota na `true`. Pri `contains 2` sa opäť pozrieme na tieto pozície a nakoľko sú na nich hodnoty `true`, výsledok bude tiež `true`.

Pri `contains 4` sú hodnoty hašovacích funkcií 0 a 1. Iba na prvej z týchto pozícií je hodnota `true`, výsledok teda bude `false`. Pri `contains 16` však opäť spočítame indexy 3 a 0, dostávame teda odpoveď `true`, ktorá je z hľadiska množiny nesprávna, podľa algoritmu ju však váš program má takto vypísať. Napokon pri pridávaní 4 sa nastavia na `true` pozície 0 a 1, pričom ale na pozícii 0 už bolo `true` aj predtým.