

EADŠ - cvičenie 4

13. októbra 2022

Slovník (dict) - opakovanie

Abstraktná dátová štruktúra s operáciami:

- ▶ `insert(key, value)`
- ▶ `search(key)`
- ▶ `delete(key)`

Rôzne implementácie:

- ▶ neutriedené pole
- ▶ utriedené pole
- ▶ linked list
- ▶ hash tabuľka
- ▶ binárny vyhľadávací strom

hashovanie - opakovanie

Idea: chceli by sme vedieť akýkoľvek¹ objekt dať do poľa tak, aby sme ho vedeli rýchlo nájsť.

Budeme mať funkciu, ktorá priradí každému objektu prirodzené číslo $0, \dots, m - 1 = \mathbf{klúč}$.

Potrebujeme, aby funkcia rovnakým objektom priradila rovnaký klúč (nemôže byť náhodná).

hashovanie, kolízie - opakovanie

Čo ak nastane kolízia ($h(u_1) = h(u_2)$, $u_1 \neq u_2$)?

- ▶ hashovanie s uzavretou adresáciou (kýbliky / spájané zoznamy)
- ▶ hashovanie s otvorenou adresáciou (skúšame iné indexy)
 - ▶ lineárne
 - ▶ kvadratické
 - ▶ dvojité

Hashovanie - 1

Máme pole čísel ($A = [5, 3, 14, 7, 2, 3]$). Chceme vedieť, či sa v ňom nachádza podpostupnosť (súvislá), ktorej súčet je k .

Hashovanie - 2

Máme pole čísel ($A = [5, 4, 14, 7, 2, 3]$). Chceme vedieť, aká je najväčšia podmnožina čísel (ktorá je tvorená po sebe idúcimi číslami) sa v ňom nachádza.
(Pre toto pole je to $\{5, 4, 2, 3\}$).

Hashovanie - 3

Ako zahashovať string?

Hashovanie - 3

Ako zahashovať string?

$$h(s) = s[0] \cdot p^0 + s[1] \cdot p^1 + s[2] \cdot p^2 + \dots$$

Hashovanie - 4

Máme číslo v desiatkovej sústave. Nech hashovacia funkcia je:
 $h(x) = x \% 9$. Aké to má problémy?

Hashovanie - 4

Máme číslo v desiatkovej sústave. Nech hashovacia funkcia je:

$h(x) = x \% 9$. Aké to má problémy?

$$1 \% 9 = 10 \% 9 = 100 \% 9 = 1000 \% 9 = \dots = 1$$

$$\begin{aligned} h(412) &= 412 \% 9 = ((4 * 100) \% 9 + (1 * 10) \% 9 + (2 * 1) \% 9) \% 9 = \\ &= ((4 * 1) \% 9 + (1 * 1) \% 9 + (2 * 1) \% 9) \% 9 = \\ &= (4 + 1 + 2) \% 9 \end{aligned}$$

Rovnako aj

$$\begin{aligned} h(241) &= 241 \% 9 = ((2 * 100) \% 9 + (4 * 10) \% 9 + (1 * 1) \% 9) \% 9 = \\ &= ((2 * 1) \% 9 + (4 * 1) \% 9 + (1 * 1) \% 9) \% 9 = \\ &= (2 + 4 + 1) \% 9 \end{aligned}$$

Hashovanie - 5

Hľadanie výskytu krátkeho stringu (abab) dĺžky m v dlhom stringu (babaabab) dĺžky n .

Hashovanie - 6

Máme 2 reťazce, chceme zistiť, či existuje také zrotovanie prvého, že dostaneme druhý.

Používanie poľa bez inicializácie

V jazyku C si vieme naalokovať veľké pole bez toho, aby sme vynulovali pamäť. Dá sa takéto pole používať ako hashtabuľka bez toho, aby sme museli pole najprv vynulovať?

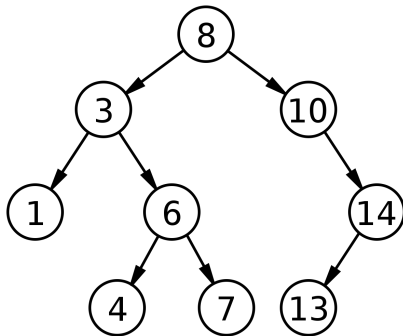
Hashovanie - 7

Používanie poľa bez inicializácie

```
def insert(key):  
    A[n]=key  
    B[key]=n  
    n++  
  
def isMember(key):  
    if B[key]>n or B[key]<0  
        return false  
    else if A[B[key]]==key  
        return true  
    else  
        return false
```

binárny vyhľadávací strom (BST) - opakovanie

- ▶ stromová dátová štruktúra
- ▶ *nie nutne* binárny strom
- ▶ ľavý syn je *menší* ako rodič
- ▶ pravý syn je *väčší* ako rodič



binárny vyhľadávací strom (BST) - opakovanie

Binárny vyhľadávací strom má dobré zložitosti iba ak je vyvážený.
Algoritmy na vyvažovanie:

- ▶ AVL
- ▶ scapegoat
- ▶ red-black tree
- ▶ treap
- ▶ ...

BST - 1

Máme BST.

- ▶ Ako nájsť najmenší väčší prvok k prvku x ?
- ▶ Ako nájsť najväčší menší prvok k prvku x ?

BST - 2

V BST máme bežne operácie:

- ▶ insert
- ▶ find
- ▶ delete

Ako urobiť operáciu `lower_than(k)`?

BST - 3

Máme pole $([5, 3, 1, 4, 7, 8])$. Chceme pre všetky i povedať, koľko prvkov na intervale $[: i]$ je menších ako i .

Praktické okienko: Python

```
s = set()  
d = dict()
```

- ▶ sú implementované ako *hashset/hashmap*
- ▶ operácie v insert, delete v $O(1)$

Praktické okienko: Python

Python (na rozdiel od C++) nemá (v základných knižniciach)
`sorted set (BST)` :(

Praktické okienko: Python - hashovanie

Ak chcete niečo vložiť ako key do dict, alebo to chcete vložiť do set, tak to musí byť hashovateľné, a musí to byť immutable.

```
# set
```

```
s = set()
s.add( (1,2) )    # tuple je ok
s.add(1)          # int je ok
s.add(7.312)     # float je ok
s.add("abcd")    # string je ok

s.add(['a', 1, 1.5]) # list nie je ok
s.add( set() )      # set nie je ok
s.add( dict() )     # dict nie je ok
```

Praktické okienko: Python - hashovanie

Ak chcete niečo vložiť ako key do dict, alebo to chcete vložiť do set, tak to musí byť hashovateľné (musí to byť immutable).

```
# dict
```

```
d = dict()
d[(1,2,3)] = ['a', 1, 1.5] # tuple je ok
d[1] = 23                 # int je ok
d[7.312] = 'ahoj'        # float je ok
d["abcd"] = set()        # string je ok

d[['a', 1, 1.5]] = 1     # list nie je ok
d[set()] = 23           # set nie je ok
d[dict()] = 'abc'       # dict nie je ok
```

Praktické okienko: Python - hashovanie

Ak chcete vložiť hashovať vlastnú triedu, musíte implementovať vlastnú metódu `__hash__(self)`. Ak už implementujete vlastnú metódu `__hash__(self)`, implementujte aj metódu implementovať vlastnú metódu `__eq__(self, other)`.

```
class Clovek:
    def __init__(self, vyska, meno):
        self.vyska = vyska
        self.meno = meno
    def __eq__(self, other):
        return self.vyska == other.vyska and self.meno
        == other.meno
    def __hash__(self):
        return hash((self.vyska, self.meno))

s = set()
jozef = Clovek(18, "Jozef")
s.add(jozef)
hash(jozef) # 1994389404
```