

Typografie a publikování – 5. projekt

Hashovacia tabuľka

Jakub Bartko

`xbartk07@stud.fit.vutbr.cz`

Vysoké učení technické v Brně
Fakulta informačních technologií

7. januára 2022

Hashovacia tabuľka je jednou zo základných abstraktných dátových štruktúr

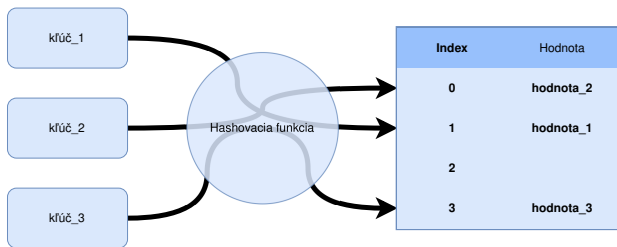
- asociatívne pole využívajúce **hashovaciú funkciu** na mapovanie kľúčov na hodnoty
- **hashovacia funkcia** pre zadaný kľúč vypočíta *hash*, ktorý určuje konkrétny *index*, na ktorom sa hľadaná hodnota nachádza

Prečo práve hashovacia tabuľka, oproti obyčajnému poľu alebo jednosmerne viazanému zoznamu?

- zvýšenie efektivity najpoužívanějších operácií
- lineárne prehľadávanie prvkov je v hashovacej tabuľke poslednou možnosťou, no v zozname samozrejmosťou

Ďalšie vlastnosti

- Priemerná zložitosť hashovacej tabuľky je $O(1)$, najhoršia zložitosť je $O(n)$ (degraduje na lineárny zoznam),
- veľkosť vyberajte tak, aby *zaplnenie nepresiahlo 70%*,
- voľba prvočísla ďalej znižuje výskyt kolízií indexov.



Obr. 1: Princíp hashovacej tabuľky

Výber hashovacej funkcie

Typografie
a publikování –
5. projekt

Jakub Bartko

Motivácia
a základy

Vlastnosti

Hashovacia
funkcia

Synonymá

Operácie

Search

Insert

Delete

Zdroje

- Využitím hashovania získavame rýchlosť vyhľadávania na úkor využívaného dátového priestoru.
- Najbežnejšie metódy hashovania: (*podľa zložitosti*)
 - 1 bitové operácie,
 - 2 násobenie,
 - 3 delenie.
- Okrem komplexnosti sa zohľadňuje najmä *rovnomerná distribúcia odlišných kľúčov* medzi indexy.
- Príkladmi efektívnych hashovacích funkcií sú *MurmurHash*¹ alebo *Fowler–Noll–Vo*²

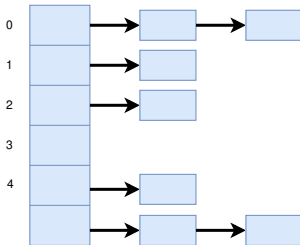
¹ <https://en.wikipedia.org/wiki/MurmurHash>

² <https://w.wiki/XKZ>

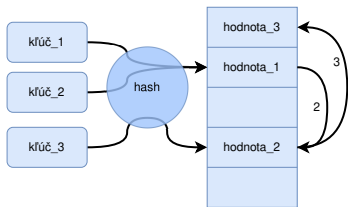
Riešenie synonym

Čo s rôznymi kľúčmi, ktoré sa namapujú na rovnaký index (**synonymá**)?

- **Explicitné** zreťazenie – napr. linerárny zoznam na každom indexe
- **Implicitné** zreťazenie – cieľový index získaný z pôvodného indexu



(a) Explicitné zreťazenie s lineárnym zoznamom



(b) Implicitné zreťazenie s pevným krokom 2

Princíp: Na príslušnom indexe prejdí zoznam prvkov a vráť ten s hľadaným kľúčom

Algoritmus 0: Vyhľadávanie prvku podľa kľúča `KEY`

```
index = hashFunc(KEY);
bucket = hashTable[index];
item = bucket.start;
while (item is not bucket.end) do
    if (item.key is KEY) then
        return item;
    else
        item = item.next;
    end
end
return NOT_FOUND;
```

Insert

Typografie
a publikování –
5. projekt

Jakub Bartko

Motivácia
a základy

Vlastnosti

Hashovacia
funkcia

Synonymá

Operácie

Search

Insert

Delete

Zdroje

Princíp: Na príslušnom indexe nájsi voľnú pozíciu a vlož na ňu nový prvok

Algoritmus 1: Vkladanie prvku s kľúčom `KEY`

```
index = hashFunc(new_item.KEY);
```

```
bucket = hashTable[index];
```

```
item = bucket.start;
```

```
while (item is not bucket.end) do
```

```
    if item is empty then
```

```
        item = new_item;
```

```
        return;
```

```
    else
```

```
        item = item.next;
```

```
    end
```

```
end
```

Delete

Typografie
a publikování –
5. projekt

Jakub Bartko

Motivácia
a základy

Vlastnosti

Hashovacia
funkcia

Synonymá

Operácie

Search

Insert

Delete

Zdroje

Princíp: Na príslušnom indexe nájsi hľadaný prvok a odstráň ho zo zoznamu

Algoritmus 2: Vymazanie prvku s kľúčom `KEY`

```
index = hashFunc(new_item.KEY);
bucket = hashTable[index];
item = bucket.start;
while (item is not bucket.end) do
    if (item.key is KEY) then
        delete item;
        link item.previous and item.next;
        return;
    else
        item = item.next;
    end
end
return NOT_FOUND;
```

Použité zdroje

Typografie
a publikování –
5. projekt

Jakub Bartko

Motivácia
a základy

Vlastnosti

Hashovacia
funkcia

Synonymá

Operácie

Search

Insert

Delete

Zdroje



Predmet IAL – 6. prednáška "Vyhledávací tabulky", FIT VUT, 2020



Which hashing algorithm is best for uniqueness and speed?
<https://softwareengineering.stackexchange.com/questions/49550/which-hashing-algorithm-is-best-for-uniqueness-and-speed>