24. Hashování- základní terminologie, princip, asociativní a adresní vyhledávání, hašovací funkce + kolize

Asociativní vyhledávání

- Hledáme porovnáváním klíčů
- Asociativní pole jako BVS
- Když klíč prvku = hledaný klíč => nalezeno
- Složitost logaritmická O(log n)

Adresní vyhledávání

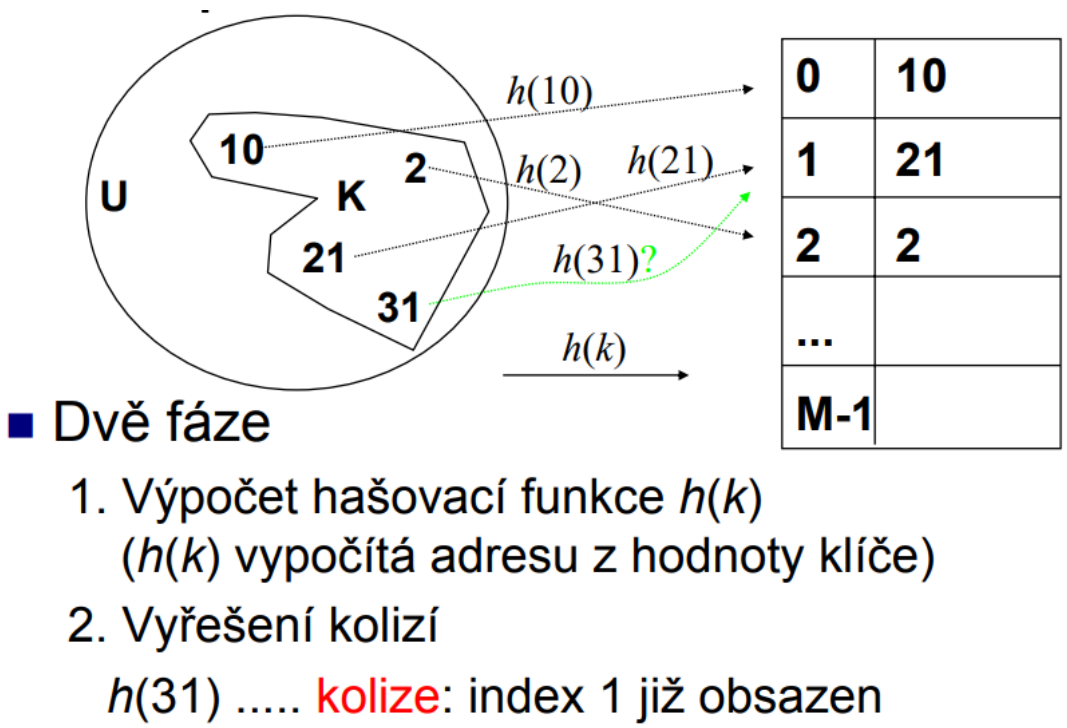
- Přímé:
  - Hledaný klíč je přímo indexem, adresou v paměti
  - Počet klíčů určuje velikost indexu – náročné na paměť
  - Složitost elementární O(1)
- Hašováním:
  - Adresu v paměti vypočteme z hledaného klíče
  - Průměrná složitost je opět O(1)

Hašování

- Je kompromis mezi rychlostí a spotřebou paměti
- Pokud máme nekonečno času – sekvenční vyhledávání
- Pokud máme nekonečno paměti – přímý přístup (indexování klíčem)
- Málo času i paměti:
  - Hašování
  - Velikost hašovací tabulky reguluje čas vyhledávání

- Hašování vhodné pro |K| << |U|
- K množina použitých klíčů

- U universum klíčů



- Dvě fáze
  1. Výpočet hašovací funkce *h*(*k*)
     (*h*(*k*) vypočítá adresu z hodnoty klíče)
  2. Vyřešení kolizí

     h(31) ..... kolize: index 1 již obsazen

- Definice: hašovací funkce h(k) je zobrazením z množiny klíčů K do množiny adres A = <Amin, Amax>
- Množiny A,K mají přibližně stejný počet prvků
- Kolizí nazýváme stav kdy pro dva různé klíče k1 != k2 platí, že h(k1) = h(k2)

Hašovací funkce h(k)

- Je silně závislá na vlastnostech klíčů a jejich reprezentaci v paměti
- Ideální funkce:
  - Výpočetně co nejjednodušší (rychlá)
  - Aproximuje náhodnou funkci
  - Využívá rovnoměrně adresní prostor
  - Generuje minimum kolizí