Hashing

Dictionaries

Ordered dictionary: Datastruktur som understøtter operationerne:

- Search(key)
- Insert(key)
- Delete(key)
- Predecessor(key)/Succesor(key)

Balancerede binære søgetræer (f.eks. rød-sorte træer) understøtter alle disse (samt mange flere, f.eks. ved at tilføje ekstra information i knuderne) i $O(\log n)$ tid.

Hvis kun de tre første operationer skal understøttes, kaldes det en *unordered dictionary*.

Sådanne kan implementeres med en teknik kaldet hashing, på en måde som giver operationer i forventet tid $O(1)$.

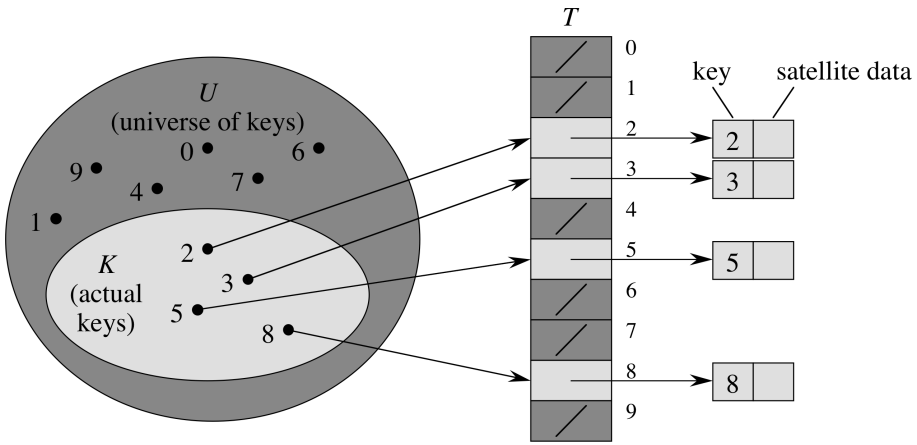
I DM507: datastrukturen, men ikke analysen.

Idé

Vi antager keys er heltal (for andre datatyper må man tildele dem en heltalsværdi, jvf. hashCode() i java) op til en max-grænse.

Universe $U = \{0, 1, 2, \ldots |U| - 1\}$

Grundideen er at se på keys som indekser i et array:



Problem: Pladsforbrug. Ofte $2^{32} = |U| >> |K| = n$.

Hash-funktioner

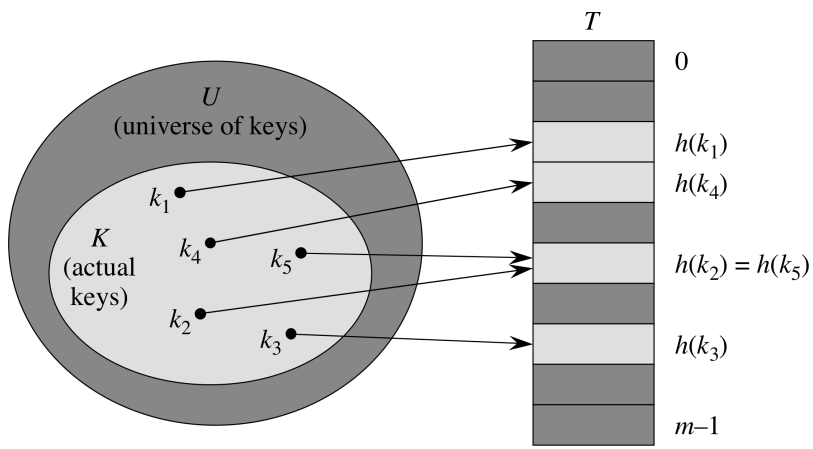
Hash-funktion:

$h : U \to \{0, 1, 2, \ldots m - 1\}$.

Her er m den ønskede tabel størrelse. Ofte vælges $m = O(n)$.

F.eks.:

$$h(k) = k \mod m$$



Hash-funktioner

Endnu bedre:

$$h(k) = ((a \cdot k + b) \mod p) \mod m,$$

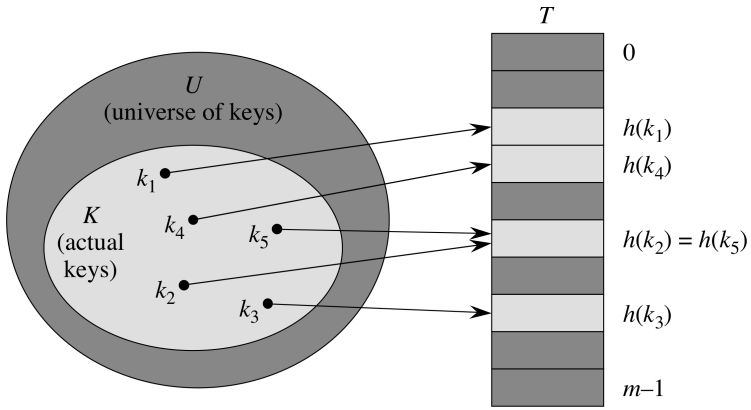
hvor a, b er faste, men tilfældigt valgte tal, og p er et primtal større end $|U|$.

Analyse heraf senere i studiet: kvalitet kan garanteres i en bestemt forstand (kaldet universal hashing).

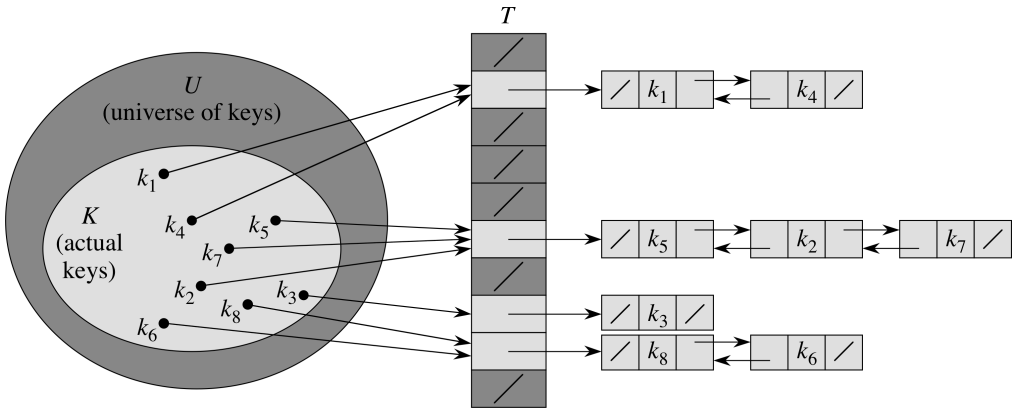
I DM507-bog (inden for DM507 pensum): flere forslag til hash-funktioner, baseret på "erfaring", men alle uden teoretisk garanti for kvalitet.

Kollisioner

To keys hash'er til samme array index:

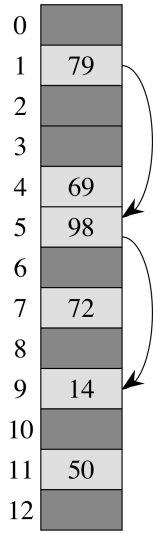


Een simpel løsning: Chaining (lænkede lister).



Open addressing

En anden løsning: Forsøg at finde tom slot.



$$h(k,i) = (h'(k) + i) \mod m$$

$$h(k,i) = (h'(k) + c_1 \cdot i + c_2 \cdot i^2) \mod m$$

$$h(k,i) = (h'(k) + i \cdot h''(k)) \mod m$$

Insert: $i = 0, 1, 2, \ldots$ forsøges til en empty slot findes.

Search: $i = 0, 1, 2, \ldots$ forsøges til element eller empty slot findes.

Sletninger: besværligt.

- Det er nødvendigt at $n \leq m$ (da alle n elementer ligger i tabellen).
- $\{h(k,0), h(k,1), h(k,2), \ldots, h(k,m-1)\}$ bør være $\{0, 1, 2, \ldots, m-1\}$ for alle k (så hele tabellen gennemsøges).