

Keresés hasító táblázatokban. Láncolt listás és közvetlen címzéses hasító technika.

Hasító (hash) táblázatok

Hasítótáblák

Adott egy nagyméretű univerzum amelyhez a kulcsuk által azonosított elemeket tartoznak. Ezek közül szeretnénk elemeket tárolni egy m méretű T tömbben úgy, hogy az elemek a kulcs alapján hatékonyan megtalálhatóak legyenek. A továbbiakban feltesszük, hogy az elemek maguk a kulcsok, de a valódi alkalmazásokban a kulcsok csak azonosításra szolgálnak, és az elemek további adatokat tartalmaznak.

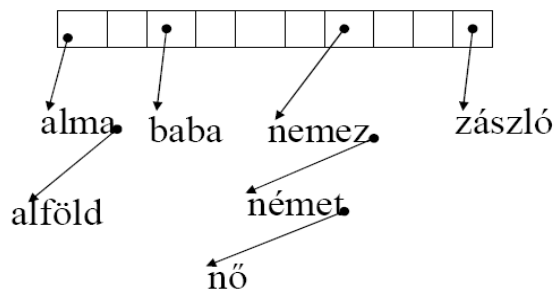
Választunk egy $h : U \rightarrow \{0, \dots, m-1\}$ hasító függvényt, amely minden a halmazelemre megadja azt a tömbindexet, ahol az elemet tárolni szeretnénk $T[h(a)] := a$.

Abban az esetben van probléma, ha két különböző elemet ugyanott szeretnénk tárolni azaz, ha $a = b$ és $h(a) = h(b)$. Az ilyen esetekben ütközésről beszélünk. Az ütközések feloldására két elterjedt módszer a láncolás és a nyílt címzés.

Ütközésfeloldás láncolással

Ebben az esetben az ütközést úgy oldjuk fel, hogy ugyanarra a helyre rakjuk az elemeket. Tehát a T tömb elemei láncok, minden láncelem egy adatból és egy csat mutatóból áll, amely a következő láncelemre mutat.

Ütközésfeloldás láncolással



- Alapötlet: az ütköző (ugyanazon címre leképeződő) elemeket láncolt listába összefoglaljuk

- Beszúr(x)

hash= $h(\text{kulcs}(x))$

if $T(\text{hash}) \neq \text{NIL}$ then $T(\text{hash}).\text{Beszúr}(x)$

else $T(\text{hash}) = \text{LáncLista}(x)$

- Keres(x)

hash= $h(\text{kulcs}(x))$

if $T(\text{hash}) = \text{NIL}$ then return NIL

else return $T(\text{hash}).\text{Keres}(x)$

- Töröl(x)

hash= $h(\text{kulcs}(x))$

if $T(\text{hash}) \neq \text{NIL}$ then $T(\text{hash}).\text{Töröl}$

if $T(\text{hash}).\text{Méret} = 0$ then $T(\text{hash}) = \text{NIL}$

Láncolásos hasító technika elemzése

- Legrosszabb eset: egyetlen láncolt lista
- Kitöltöttségi arány ($a = n/m$), ahol n a tárolt elemek összes száma, m a hash-tábla hossza
- Tétel: láncolt hasító technika és egyenletes hash függvény esetén, az átlagos keresési idő: $\Theta(1 + a)$.
- Biz: az átlagos keresési idő az átlagos lista végigjárásával arányos (sikeres és sikertelen keresés)

- Következtetés: 1. Fix hash-tábla esetén, sok elemre lineáris $\Theta(n)$ 2. Ha a hash-tábla a tárolt elemekkel arányosan nő, ill. kevés a tárolt elem, akkor $\Theta(1)$

Hasító függvények

- Mikor jó a hasító függvény? Ha a láncok kb. egyenletesen nőnek \leq azaz a $T(y)$ index egyenletes
- Mivel $T(y) = T(h(U(x)))$, függ az $U(x)$ kulcseloszlástól
- Milyen az $U(x)$ eloszlása?
 - fordítóprogramok: egymáshoz közeli szimbólumok
- H eloszlása független legyen az adatok esetleges szabályszerűségeitől
- pl. a kulcs maradéka egy prímszámmra nézve
- Feltételezzük, hogy a kulcs egész (ha nem az, akkor leképezhető)

Hasító függvények

- Osztásos módszer: $h(k) = k \bmod m$, ahol m a hash tábla mérete
- Szorzásos módszer $h(k) = m * (k * A \bmod 1)$, ahol A egy állandó, $\bmod 1$: törtrészképzés
Pl. tfh: k befér egy gépi szóba, és $m = 2^p$. Ekkor $k * (A * m)$ egy gépi szorzás \rightarrow 2 szó, ez $\bmod m$ az alsó p bit levágását jelenti. MFC (Donald Knuth)
- Minden rögzített hasító függvényhez létezik „rossz” adat (csak 1 lánc jön létre, $\Theta(n)$ elérés) \rightarrow a tényleges függvényt adott függvényosztályból véletlenszerűen (és adatfüggetlenül) választjuk ki. „Univerzális hasítási technika”

Nyílt címzés

- Nyílt címzésű a hash tábla, ha az adatok NEM láncolt listában, hanem benn a táblában vannak tárolva. Ütközés esetén újabb és újabb pozíciókat próbálunk ki, amíg csak üresre nem bukkanunk
 - \rightarrow a tábla betelhet
 - \rightarrow nincs mutató, nincs láncolt lista
 - \rightarrow a kipróbálandó rések címét a hash függvény (a kipróbálási számtól is függően) adja meg
 - \rightarrow adott kulcshoz a hash függvény a T címtér egy permutációját adja meg (vagyis mindent kipróbál)

Ütközésfeloldás nyílt címzéssel

(más: Nyílt címzés esetén, egy helyre csak egy elemet teszünk, és az ütközést úgy oldjuk fel, hogy nem csak egy hasítófüggvényünk van, hanem minden elemhez tartozik egy próbasorozat. Az elemet a próbasorozatból az első üres helyre tesszük.)

A nyílt címzéses hash technika elemzése

- Kitéltöttségi arány ($a = n/m$), ahol n a tárolt elemek összes száma, m a hash-tábla hossza $\rightarrow a \leq 1$
- Tfh. A hasítás egyenletes, vagyis egy $(h(k,0), h(k,1), \dots, h(k,m-1))$ kipróbálási sorozat egyenletesen állítja elő a $(0,1, \dots, m)$ sorozat permutációit.
- Legyen p_i annak valószínűsége, hogy pontosan i próba talál foglalt rést. ($0 \leq i \leq n-1$), $i > n$ -re $p_i = 0$
- A próbák számának várhatóértéke tehát: $1 + \sum_{i=0}^{\infty} i * p_i$
- Legyen q_i annak valószínűsége, hogy legfeljebb i próba talál foglalt rést. (aztán jön a szabad)
- $\sum_{i=0}^{\infty} i * p_i = \sum_{i=0}^{\infty} q_i$ (annak a valószínűsége, hogy pontosan $0;1; \dots$ próba talál rést, ugyanannyi, mint a valószínűsége, hogy legfeljebb $0;1; \dots$ próba talál rést)
- Mennyi a q_i -k értéke?
- $q_1 = n/m$ annak a valószínűsége, hogy az első próba foglalt elemet talál
- $q_2 = n/m * (n-1)/(m-1)$
- $q_i = n/m * (n-1)/(m-1) * \dots * (n-i+1)/(m-i+1) \leq a^i$
- A próbák számának várhatóértéke: $1 + \sum_{i=0}^{\infty} i * p_i = 1 + \sum_{i=0}^{\infty} q_i \leq 1 + a + a^2 + a^3 + \dots = 1/(1-a)$

- 1. Elem biztos, 2. Elem a, 3. Elem a2...
- Ha félig van kitöltve, akkor próbaszám=2 lépés
- Ha 90%-ig van kitöltve, akkor próbaszám=10 lépés
- Kitöltöttségi arány ($a=n/m$), ahol n a tárolt elemek összes száma, m a hash-tábla hossza $\rightarrow a \leq 1$
- Ilyenkor a sikeres keresés várható próbaszáma: $1/a * \ln(1/(1 - a))$
- Bizonyítás nélkül... (az előzőhöz hasonló)
 - \rightarrow 50% kitöltöttségre: 1,387
 - \rightarrow 90% kitöltöttségre: 2.559
 - \rightarrow a nyílt címzéses hash technika a keresés optimalizálását a beszúrás lassulásán keresztül éri el...