ADATSZERKEZETEK ÉS ALGORITMUSOK

Hash táblák

Motiváció

Nagyon sok adatban kell nagyon gyorsan keresni

O(log(n)) nem elég gyors

Háttértárat/memóriát használunk a processzoridő helyett

Konstans idejű

- Beszúrás
- Törlés
- Keresés

Az alapötlet

Az elem helyét *(indexét)* a kulcsából generáljuk

Az index független a többi értéktől, nincs összehasonlítás

Konstans idő alatt generálható

Feladat

Alakítsd át a hasító tábla osztályt, hogy nyílt címzést használjon, lineáris kipróbálással!

...quadratic probe-bal! (https://en.wikipedia.org/wiki/Quadratic_probing)

...dupla hasítással! (https://en.wikipedia.org/wiki/Double_hashing)

Hashtáblák az STL-ben

C++11 óta:

- `std::unordered_set`
- `std::unordered_multiset`
- `std::unordered_map`
- `std::unordered_multimap`
- std::hash

Feladat

Hasonlítsd össze az `std::set`, az `std::unordered_set` és a saját hashtáblánk futási idejét!

Hogyan lehetne gyorsítani?

Amit még nézzetek át

STL-es adatszerkezetek:

- Lesz még szó róluk előadáson. Használjátok nyugodtan őket a zh-ban is, sokszor gyorsabb megírni így, mintha a saját megoldásainkat használnánk.

Futási idők:

- Tudjátok, hogy mi mennyi idő alatt fut le (O szempontból; pl. beszúrás, törlés, rendezés így/úgy/amúgy, iterálás tömbön/fán/hashtáblán), hogy minél jobb megoldást tudjatok adni a feladatokban.
- Gyakran, ha meg van mondva, akkor segítség a futási idő abban, hogy milyen megoldást érdemes választani (lásd pl. a gyakorló feladatokat vagy a tavalyi zh-t).

Gyakorló feladat

Készíts programot, ami egy txt-ben szereplő könyv szavait beolvassa, és azokból gyakoriságot számol. Az írásjeleket vegyétek ki az egyes szavakból, illetve a kis- és nagybetűk között ne legyen különbség. A program beolvasás (*O(n)*) után tudja a következő funkciókat:

- Megadja, hogy hány különböző szó található a könyvben. *O(1)*
- Megadja a leggyakrabban használt szavakat. *O(k)*
- Egy adott szóra rákeresve megadja, hogy az hányszor szerepel. (*O(1)*)

Ahol a hatékonyságnál n az összes szó száma a szövegben, k pedig a különböző szavak száma.

A teszteléshez txt fájlokat pl. innen állíthattok elő:

- http://mek.oszk.hu/00600/00656/html/

Gyakorló feladat

B-fa: bár idén gyakorlaton nem szerepelt, gyakorlásnak, illetve haladóbbaknak mindenképp ajánlott lehet ezeket átnézni, megírni.

Ehhez segítségként elérhető anyagok:

- Korábbi diasor: https://wiki.itk.ppke.hu/twiki/pub/PPKE/AdatAlg201314/09_btree.pdf
- Korábbi órai kód: https://wiki.itk.ppke.hu/twiki/pub/PPKE/AdatAlg201314/09_btree.zip

Egészítsétek ki az órai kódot, nézzétek át, hogy hogyan működik.

Az ezzel és az önálló adatszerkezet-írással szerzett tapasztalat hasznos lehet pl. későbbi állásinterjúk esetén is.

Gyakorló feladat

Készíts programot, ami egy txt-ben szereplő könyv szavait beolvassa, azokból egy hash-t készít, majd egy számlálóval együtt beteszi egy hash táblába. Ha többször beolvassuk ugyanazt a szót, akkor a megfelelő hash tábla elemhez tartozó számlálót növeljük. Kulcsütközés esetén láncolt listát használj, hogy véletlenül se növeld ugyanazt a számlálót két különböző szóra. Ezzel megkapjuk a szövegben előforduló szavak gyakoriságát.

Készíts statisztikát arról, hogy mekkora hash táblával (lehetséges kulcsok száma) érhető el 25%, 10% és 1% körüli kulcsütközés.

Szövegfájl:

http://mek.oszk.hu/01200/01268/01268.htm