Tema NR 4: Hashtabeller

Annika Svedin `ansv9785`
Felix Törnqvist `fetr0498`

10 februari 2017

Hashfunktion

En hashtabell består av nycklar och nycklarnas värde. Om hashtabellen är arraybaserad är värdet elementet och nyckeln elementets index. .... En hashfunktion konverterar ett värde till ett index i arrayen.

Exempel:

Kollisionshantering

...

Det finns olika modeller för att lösa problemet med kollisioner vid insättning. Ett sätt är att inkrementera index tills en ledig plats hittats. Om fyra nycklar mappar över samma index x blir placeringen av dessa $x, x+1, x+2$ och $x+4$. Det första

av de fyra som sätts in sätts in på x, nr två sätts in på indexet efter x o.s.v. Söker man nyckeln som ligger på $x + 4$ så börjar man kolla på x, följt av $x + 1, x + 2$ och $x + 3$. Nyckel fyra finns på $x + 3$ och kan returneras. Skulle nyckeln vi söker inte finnas på $x0 - x4$, avbryts sökningen då $x + 5$ visar sig vara tom. Sökningen blir linjär och därav benåmningen linear probing.

Börjar hashtabellen/listan bli fylld innebär det långa kluster som behöver genomsökas, vilket tar tid. Av den anledningen skapar man dubbelt så många platser jämfört med antal nycklar. Då får man en större spridning av elementen och kan spendera kortare tid på sökning.

Den här metoden fungerar bra till en viss gräns. Har man otur med balansen mellan nycklarn [...]

I värsta fall behöver man söka sig igenom alla nycklar i klustret innan man får hittar det man vill. Samtidigt är det många index i arrayen som gapar tomma.

Ett andra sätt

För att komma undan problemet med långa kluster och öka spridningen kan man undvika ansamlingarna genom att ta längre skutt för varje ny nyckel på samma index. Istället för $x + 0$, $x + 1$, $x + 2$, $x + 3$ lägger man till tvåpotenserna av ökningen av x; $x + 0^2$, $x + 1^2$, $x + 2^2$, $x + 3^2$ osv.

Avståndet mellan elementen ökar (mer än)kvadratiskt för varje insättning, därav benämningen quadratic probing. Alla nycklar som är bundna till ett visst index x kommer finnas inom samma intervall i förhållande till x.

Ett tredje sätt

Ett tredje sätt att angripa problemet med kollisioner är att, som i föregående exempel låta index x öka med ett visst antal steg, men istället för att det ska bli samma antal steg mellan nycklarna i för alla index låter man istället antalet steg bestämmas av värdet på x. Man låter x genomgå en ny hashning, och låter resultatet bli storleken på intervallen mellan nycklarna under x. Metoden kallas double hashing.

Ett fjärde sätt

Men man kanske vill komma bort från den dubbla hashningen eller den allt mer komplexa strukturen i en växande samling. Då är en idé att låta varje index i arrayen vara en länkad lista. Genomsökningen kommer i värsta fall ta lika lång tid som i det första exemplet, med linear probing(ELLER hur...?), men man slipper göra en dubbelt så stor lista som antalet nycklar. Att låta nycklarna bestå av länkade listor kallas separate chaining. Mer om detta?