DMA 2021 ugeseddel 10

Rasmus Pagh

Litteratur

- CLRS 11.0, 11.1, 11.2, 11.3 (dog ikke 11.3.3), 11.4 (dog ikke Analysis of open-address hashing). Vi bruger i denne uge notation defineret i CLRS sektion 3.2, sektionerne "Floors and Ceilings" samt "Modular arithmetic" (side 54)
- CLRS kapitel 17.0, 17.1, 17.2
- Note: Cuckoo hashing for undergraduates

Mål for ugen

- Kendskab til hashtabeller (chaining, linear probing, cuckoo hashing) og deres egenskaber
- Introduktion til amortiseret analyse

Plan for ugen

- Mandag: Hashtabeller med hægtede lister
- Tirsdag: Open addressing, cuckoo hashing
- Fredag: Introduktion til amortiseret analyse

Opgaver

For opgaver markeret med † kan du finde svar via Absalon, men det er vigtigt at du forsøger at løse opgaven, evt. med hjælp fra din instruktor, inden du kigger dér. **NB! Opgaver markeret med "*" er svære, "**" er meget svære, og "***" har du formentlig ikke en chance for at løse.**

Mandag

1. CLRS 11.1-1 †
2. CLRS 11.1-2. † Antag at hvert element i tabellen indeholder et w-bit heltal, dvs. hvert element kommer fra mængden $\{0, \ldots, 2^w - 1\}$. Vi vil gerne gemme vores bitvektor i en tabel T af størrelse $\lceil m/w \rceil$. Din løsning skal gøre brug af *bit-operationer* for at tilgå og manipulere indgangene i T. Et overblik over bitoperationer i F# er her.
3. CLRS 11.2-2
4. CLRS 11.3-1 †
5. CLRS 11.3-4

6. Lad K være en sekvens af heltals-nøgler gemt i en hashtabel A ved brug af hægtet hashing. Givet A, hvordan kan man finde det maksimale element i K? Hvad er den asymptotiske køretid som funktion af antal nøgler n og størrelsen af hashtabellen m?

7. **Multimængder. *** Det er muligt at gemme den samme nøgle flere gange (med forskellig satellit-data i hver kopi) i en hashtabel med hægtede lister. Det gør mængden af nøgler til en *multimængde* med n nøgler $k_1, \ldots, k_n$ der ikke nødvendigvis er forskellige. Udvid CHAINED-HASH-SEARCH(T, k) til at returnere *alle* $t(k)$ satellit-data, der er gemt sammen med nøglen k. Vis at den forventede søgetid kan begrænses som $O(1 + \alpha + t(k))$, hvor $\alpha = n/m$ som i CLRS kapitel 11.

8. CLRS 11.1-4 **

Tirsdag

1. **Håndkørsel af linear probing.** Indsæt nøglesekvensen 2, 32, 43, 16, 77, 51, 1, 17, 42, 111 i tabel af størrelse 17 vha. linear probing med hashfunktionen $h(k) = k \bmod 17$.

2. Antag at vi laver sletning i lineær probering uden at markere indgange som slettet (vi fjerner bare elementet hvis vi finder det). Giv en sekvens af ordbogsoperationer, der viser at dette ikke virker korrekt.

3. CLRS 11.4-2

4. **Sletning i linear probing.** † Slet 111 og 51 fra tabellen produceret i opgaven ovenfor, og bekræft at søgning efter de resterende nøgler stadig virker.

5. **Håndkørsel af cuckoo hashing.** † Indsæt nøglesekvensen 1, 17, 7, 5, 20, 24, 15 i tabel af størrelse 13 ved brug af cuckoo hashing med hashfunktionerne $h_1(k) = k \bmod 13$ og $h_2(k) = (2k + 1) \bmod 13$.

6. **Simple tabulation hashing. ** En streng x af længde ℓ kan betragtes som en tabel af heltal $x_1, \ldots, x_k \in \{0, \ldots, 255\}$ (hvor tegn nummer i er repræsenteret ved tallet x_i (i for eksempel ASCII formatet). Simple tabulation hashing definerer en hashfunktion på sådanne strenge ved ℓ arrays, $T_1, \ldots, T_\ell$, der hver indeholder 256 tal valgt uniformt tilfældigt fra $\{0, \ldots, m-1\}$:

$$h(x) = (T_1[x_1] + T_2[x_2] + \cdots + T_k[x_\ell]) \bmod m$$

   Argumentér for at for to forskellige strenge x og y af længde ℓ er sandsynligheden for at $h(x) = h(y)$ lig med $1/m$.[1] Udvid funktionen til at håndtere strenge af længde *højst* ℓ.

7. **Cuckoo hashing fejlsandsynlighed**. *** Vis at sandsynligheden for at cuckoo hashing med n nøgler fejler (så nye hashfunktioner er nødvendige) er $O(1/n)$.

Fredag

1. CLRS 17.1-1

2. CLRS 17.1-2

3. CLRS 17.1-3 †

4. CLRS 17.2-2 †

5. CLRS 17.2-3 † *

[1] KU-professor Mikkel Thorup og Mihai Pătraşcu fra MIT viste i 2012 at simple tabulation hashing har meget stærke egenskaber, og det har siden ledt til en række af effektive hashfunktioner med stærke teoretiske garantier.

6. **Mere amortisering. *** Forestil dig at vi har en datastruktur hvor den i'te operation tager $f(i)$ tid (vi vil definere $f(i)$ herunder). Lad os udføre n operationer, som altså totalt koster $f(1)+f(2)+\ldots+f(n)$. For hver af følgende definitioner af $f(i)$, hvad er den amortiserede køretid af en operation?

   (a) $f(i) = i^2$ hvis i er en potens af 2 og $f(i) = 1$ ellers.

   (b) $f(i) = i$ hvis i er ulige og $f(i) = 1$ ellers.