Hashing, del I

How to get away with running your code on wimpy hardware

Diskret Matematik og ~~Algoritmer~~
Københavns Universitet, efterår 2021

Rasmus Pagh

Hvad sker der?

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

Motiverende case

- Data: En liste af n Facebook venskaber

  `[(Rasmus,Anna),(Rasmus,Thore),(Thore,Tony),(Anna,Thore),…]`

- Spørgsmål: Hvor mange venner har hver person i listen?
- Kan løses med værktøjer I har set:
  - Dobbelt `for`-løkke
  - Hægtede lister
  - Binære søgetræer
  - Sortering
- Kan vi gøre det bedre?

Ordbøger (dictionaries)

- Ordbog: Datastruktur der gemmer en mængde af *nøgler* (fx strenge eller heltal)
- Til hver nøgle kan knyttes en *værdi* (= *satellit-data*)
- Operationer: `Search`, `Insert`, `Delete`

- Mulige løsninger: Array (sorteret), hægtet liste, binært søgetræ

Direkte addressering

- Hvis nøglerne er heltal i $U = \{0, \ldots, m-1\}$ kan vi bruge *direkte addressering* til en tabel af længde m

Hash

Også hash

Hashing

- **Idé**: Transformér en nøgle x til et heltal $h(x) \in \{0, \ldots, m-1\}$, og lav direkte addressering med $h(x)$.

Øvelse (5 min)

- Antal at nøglerne er fødselsdatoer, `yyyy-mm-dd`, på nulevende personer.
  - Hvad er en god hashfunktion på denne mængde?
  - Hvor lille kan m gøres hvis vi helt skal undgå kollisioner?

Tilfældige hashfunktioner

- Tilfældig (random): Valgt ud fra en sandsynlighedsfordeling ($\neq$ arbitrær)
- Ønsker at sandsynligheden for kollisioner, $h(x) = h(y)$, skal være lav for $x \neq y$:

$$\Pr[h(x) = h(y)] \leq 1/m \quad \text{(simple uniform hashing)}$$

- **Eksempler på hashfunktioner:**
  - For heltal x: $h(x) = ((ax + b) \mod p) \mod m$
    hvor p er et stort primtal og a, b er valgt uniformt tilfældigt i $\{0,1,\ldots,p-1\}$
  - For streng $x_1x_2\ldots x_n$: $h(x_1x_2\ldots x_n) = \left(\left(\sum_{i=1}^{n} a_i x_i\right) \mod p\right) \mod m$

Problem: Hash-kollisioner

Fra uge 4: Hægtede lister

I hukommelsen:

List-Search(L, k): Søg fra begyndelsen efter nøgleværdi k.

```
List-Search(L, k)
  x = L.head
  while x ≠ NIL and x.key ≠ k
    x = x.next
  return x
```

$\Theta(n)$ tid.

Hashing med hægtede lister

Ordbøger i F#

uge 10 - hashing og amortiseret analyse — mono /Library/Frameworks/Mo...

```
[> open System.Collections.Generic                                    ]
[- let dict = new Dictionary<string, int>()                           ]
[- dict.Add("Thore",1)                                                ]
[- dict.Add("Rasmus",2);;                                             ]
 val dict : Dictionary<string,int> = dict [("Thore", 1); ("Rasmus", 2)]
 val it : unit = ()

[> printfn "%i" dict.["Thore"];;                                      ]
 1
 val it : unit = ()

[> printfn "%b" (dict.ContainsKey("Rasmus"));;                        ]
 true
 val it : unit = ()

[> dict.Remove("Rasmus")                                              ]
[- printfn "%b" (dict.ContainsKey("Rasmus"));;                        ]
 false
 val it : unit = ()

 >
```

HashSet

- Hashtabel med nøgler, ingen værdier.
- Anvendelse: Fællesmængde

HashSet.fsx — uge 10 - hashing og amortiseret analyse

```
open System.Collections.Generic

// Two approaches to set intersection
let hash_intersect (a:list<'t>) (b:list<'t>) =
    let s = new HashSet<'t>(a)
    b |> List.filter (fun x -> s.Contains(x))

let simple_intersect (a:list<'t>) (b:list<'t>) =
    a |> List.filter (fun x -> List.exists (fun y -> y = x) b)

```

Analyse af hashing med hægtede lister

- Analyse af `Search` for hashtabel T med nøglemængde K:
  - Tid til at evaluere hashfunktionen og hente referencen $T[h(x)]$: $O(1)$
  - Gennemløbe hægtet liste: $O(X)$ hvor $X = |\{k_i \in K \mid h(k_i) = h(k)\}|$
  - $X = \sum_{k_i \in K} X_{k_i}$, hvor $X_{k_i} = 1$ hvis $h(k_i) = h(k)$ og $X_{k_i} = 0$ ellers
  - Forventet tid (uden store-O):

$$1 + E[X] = \left(1 + \sum_{k_i \in K} E[X_{k_i}]\right) \leq 2 + \sum_{k_i \in K} 1/m = 2 + n/m$$

Opsummering

- Hashtabeller er en tids- og pladseffektiv løsning til ordbogsproblemet
- Vigtigt at hashfunktionen “ligner” en tilfældig funktion på nøglemængden, ellers kan tiden per operation blive $O(n)$
- Flere detaljer findes i CLRS sektion 11.0 - 11.3.2 [ekstra: 11.3.3]
- I morgen: Mere hashing!

- **Husk**: Nye lokaler til øvelsestimer i blok 4