Ola Kristoffer Hoff & Oline Amundsen

Algoritmer og datastrukturer
Øving 4 – Hashtabeller

Jeg har samarbeidet med Oline Amundsen. Vi har gjort alt sammen, samarbeidet har vært faglig diskusjon, ikke fordeling av arbeidsmengde.

Deloppgave 1: Hashtabeller med tekstnøkkler

Vi implementere en hashtabell-klasse basert på eksempler i boka og forelesningen. Laget en abstrakt klasse Hashtabell og to klasser (HashtabellRestDiv og HashtabellHeltallMulti) som arvet det meste av implementasjonen fra denne. Vi implementere deretter enkellenka lister basert på bokas eksempler og brukte en liste av dette objektet som datastrukturen vi lagret tekststrengene til hashtabellen i. Klassene HastabellRestDiv og HashtabellHeltallMulti implementere vi deres respektive hash-fuksjoner, da restdivisjons-hash og heltallsmultiplikasjons-hash.

Restdivisjons funksjon

- I denne metoden endte vi opp med å finne siste primtall større enn antall elementer, men som ga maksimalt overhead på 25%, og da potensielt en lastfaktor på 0.8. Så brukte vi dette som størrelse på tabellen. Hvis vi ikke finner et godt primtall bruker vi bare en verdi som gir oss overhead på 25%. Hashtabellen fikk grei spredning selv med ikke-primtallverdier så vi gjorde ikke noen videre optimalisering av denne.
- For å optimalisere den enda mer kunne vi sett på primtallet som kommer rett før innsendt verdi også, da hvis denne ligger ganske nære vi få bedre spredning med denne enn en ikke-primtallverdi (som regel).

Heltallsmultiplikasjons funksjon

- I denne metoden sjekker vi hva toer potensen som ligger rett før og rett etter (evt. på) den innsendte verdien er. Hvis den størst sin lastfaktor blir mindre enn 0.6 bruker vi den minste verdien og har da lastfaktor på 1.2 og mindre. Siden lastfaktoren til den miste verdien alltid er dobbelt så stor som den større sin (siden: $\frac{x}{2^i}$ $\alpha = \frac{x}{2^{i+1}} \rightarrow \alpha = \frac{1}{2}$) vil det fort lønne seg å bruke den større hvis den innsendte verdien ikke ligger veldig nær den minste verdien. Vi testet litt og fant at 0.6 i lastfaktor for den største var en grei grense, da 1.2 og ned til 1 i lastfaktor var bedre enn 0.5 i lastfaktor

Konvertering av streng til heltall

- I denne metoden loop'er vi gjennom alle bokstavene og parser dem til en int, da får UNICODE/ASCII-verdier, og disse blir ganget med $7^{(i \ mod \ 6)}$ hvor «i» er det i'te elementet vi er på (loop conuter'en), vi måtte implementere «mod 6» da vi fikk et problem med at når «i» ble «stor» (8, 9, 10, og oppover) så ville

totalsummen nå grensen til en int og forbli der, noe som resulterte i at alle navnene som ble sendt inn havnet på samme plass, men med en «6» som kontroll tall vil strenger opp i lengden mot hundre tegn få verdier under 32 millioner og grensen til en int er på 2,147,483,647 (ca. 2.1 milliarder), så vi har en god del å gå på.

- Navnene fjerner vi komma- og mellomromtegn fra, da disse vil forekomme i alle og ikke bedra til unikheten til de forskjellige elementene, men heller kanskje motvirke, f. eks. da disse tegnene kommer rett etter hverandre.
- Selve file med navn la vi i «src» mappen.

Vi implementerte søk ved å gjøre akkurat dem samme som i innleggingen, annet en at vi så om elemente vi søkte etter lå der vi ville lagt det inn.

```
Hashtabell med restdivisjon
Hvem leter du etter? (Skriv inn fult navn, fornavn så etternavn)
Skriv '5' for å slutte

Ola Kristoffer Hoff
'Ola Kristoffer Hoff' er i hastabellen

Hvem leter du etter? (Skriv inn fult navn, fornavn så etternavn)
Skriv '5' for å slutte

kake
'kake' er ikke i hastabellen

Hvem leter du etter? (Skriv inn fult navn, fornavn så etternavn)
Skriv '5' for å slutte
```

```
Hashtabell med heltallsmultiplikasjon
Hvem leter du etter? (Skriv inn fult navn, fornavn så etternavn)
Skriv '6' for å slutte

Ola Kristoffer Hoff
'Ola Kristoffer Hoff' er i hastabellen

Hashtabell med heltallsmultiplikasjon
Hvem leter du etter? (Skriv inn fult navn, fornavn så etternavn)
Skriv '6' for å slutte

kake
'kake' er ikke i hastabellen
```

Vi printer ut lastfaktor, antall kollisjoner, kollisjoner per person, og alle kollisjoner under innsetting og søk:

```
Lastfaktor (restDiv): 0.8037383177570093
Kollisjoner restDiv (under innsetting): 40
Kollisjoner per person restDiv (under innsetting): 0.46511627906976744

Lastfaktor (heltallMulti): 0.671875
Kollisjoner heltallMulti (under innsetting): 24
Kollisjoner per person heltallMulti (under innsetting): 0.27906976744186046


Kollisjoner under innsetting i restDiv (kolidert med | innsettings verdi):
[Henrik Tengs,Hafsø | Michal Robert,Panasewicz, Lars-Håvard Holter,Bråten | Eirik,

Kollisjoner under innsetting i heltallMulti (kolidert med | innsettings verdi):
[Linda Katrine,Larsen | Henrik Tengs,Hafsø, Eirik,Steira | Ola Kristoffer,Hoff, To

Kollisjoner under søk i restDiv (kolidert med | søke verdi):
[Morten Stavik,Eggen | Oline Amundsen, Mathilde Kvam,Bugge-Hundere | kake]

Kollisjoner under søk i heltallMulti (kolidert med | søke verdi):
[Eirik,Steira | Ola Kristoffer Hoff, Ingebrigt Kristoffer Thomassen,Hovind | kake]
```

	Restledd divisjon	Heltall multiplikasjon
Lastfaktor	~ 0.8037	~ 0.6719
Totalt antall kollisjoner under innsetting	40	24
Antall kollisjoner per person	~ 0.47	~ 0.28

Som vi ser er kollisjoner per person godt under 3

Fullstendig kollisjonslister ved innsetting fra utskrift: (Vurderte dette til den beste måten å vise alle kollisjonene da det var 64 stykker og alle inneholder to navn, så den ble ganske lang) Kollisjonene fra søk er med i bilde ovenfor, søkte på tre ting i begge («Ola Kristoffer Hoff», «Oline Amundsen» og «kake»), søk som treffer på første forsøk har da naturligvis ikke hatt noen kollisjoner og er da ikke i listen

Kollisjoner under innsetting i restDiv (kolidert med | innsettings verdi):
[Henrik Tengs,Hafsø | Michal Robert,Panasewicz, Lars-Håvard Holter,Bråten | Eirik,Steira, Morten Stavik,Eggen | Oline,Amundsen, Thomas Thien Dinh,Tran | Mahmoud Hasan,Shawish, Morten Stavik,Eggen | Tommy Duc,Luu, Oline,Amundsen | Tommy Duc,Luu, Nicolay,Schiøll-Johansen | Mathias,Myrold, Ingebrigt Kristoffer Thomassen,Hovind | Martin Slind,Hagen, Karl Klykken,Labrador | Magnus Øvre,Sygard, Ola Kristoffer,Hoff | Jenny Farstad,Blindheimsvik, Henrik Tengs,Hafsø | Matilde Volle,Fiborg, Michal Robert,Panasewicz | Matilde Volle,Fiborg, Magnus,Bredeli | Truls Kolstad,Stephensen, Olaf,Rosendahl | Håvard,Tysland, Simon,Jensen | Lea,Grønning, Endré,Hadzalic | Erling Sung,Sletta, Peder Johan,Lindberg | Øyvind,Henriksen, Peder Johan,Lindberg | Thomas,Huru, Øyvind,Henriksen | Thomas,Huru, Rokas,Bliudzius | Torbjørn,Øverås, Stine,Rygh | Torstein,Øvstedal, Simon,Jensen | Chloe Kumari,Hansen,

Lea,Grønning | Chloe Kumari,Hansen, Linn Camilla,Bauer | Ole,Løkken, Lars-Håvard Holter,Bråten | Eivind,Berger-Nilsen, Eirik,Steira | Eivind,Berger-Nilsen, Viljar Svare,Ødelien | Mads,Lundegaard, Viljar Svare,Ødelien | Jens Mjønes,Loe, Mads,Lundegaard | Jens Mjønes,Loe, Erik Kaasbøll,Haugen | Jesper Forrest,Hustad, Hogne Heggdal,Winther | Mattias Agentoft,Eggen, Peder Johan,Lindberg | Olof André,Marklund, Øyvind,Henriksen | Olof André,Marklund, Thomas,Huru | Olof André,Marklund, Hogne Heggdal,Winther | Nora Evensen,Jansrud, Mattias Agentoft,Eggen | Nora Evensen,Jansrud, Jørgen,Selsøyvold | Georg Vilhelm,Seip, Torbjørn,Bakke | Arvid Jr,Kirkbakk, Mai Helene,Grosås | Henrik Latsch,Haugberg, Sergio,Martinez | Jostein Johansen,Aune]

Kollisjoner under innsetting i heltallMulti (kolidert med | innsettings verdi):
[Linda Katrine,Larsen | Henrik Tengs,Hafsø, Eirik,Steira | Ola Kristoffer,Hoff, Torbjørn,Bakke | Tommy Duc,Luu, Mahmoud Hasan,Shawish | Odin,Kvarving, Kenneth,Solvoll | Sindre August,Strøm, Thomas Thien Dinh,Tran | Matilde Volle,Fiborg, Hermann Owren,Elton | Erik Kaasbøll,Haugen, Michal Robert,Panasewicz | Olaf,Rosendahl, Sergio,Martinez | Viljar Svare,Ødelien, Truls Kolstad,Stephensen | Endré,Hadzalic, Martin Slind,Hagen | Håvard,Tysland, Kamilla,Mortensen | Thomas,Huru, Sergio,Martinez | Eivind,Berger-Nilsen, Viljar Svare,Ødelien | Eivind,Berger-Nilsen, Peder Johan,Lindberg | Diderik,Kramer, Øyvind,Henriksen | Mats Erik Tuhus,Olsen, Fredrik Holm,Julsen | Simon Dreyer,Vetter, Stian Fjæran,Mogen | Stian Valbekmo,Selvåg, Ilona,Podliashanyk | Nora Evensen,Jansrud, Sander,Pettersen | Georg Vilhelm,Seip, Torbjørn,Bakke | Scott Rydberg,Sonen, Tommy Duc,Luu | Scott Rydberg,Sonen, Fredrik Holm,Julsen | Jostein Johansen,Aune, Simon Dreyer,Vetter | Jostein Johansen,Aune]

<u>Deloppgave 2: Hashtabeller med heltallsnøkkler og ytelse</u>

1. Vi implementerte en hashtabell som tar heltallsnøkkler. Vi fant at primtallet 13.333.339 gir overhead på ca. 33% med en lastfaktor på ca. 0.75, som gjør ytelsen relativt god. Siden vi velger å sikre at størrelsen på tabellen er et primtall så vil hash funksjonene våre opprettholder relativ primhet til listelengden og probingen alltid vil fungere. Trenger heller da ikke sjekke for relativ primhet. Vi har valgt de to hash funksjonene: $hash1(k) = k \ mod \ 17$ og $hash2(k) = (k \ mod \ 16) + 1$, grunnet at de var brukt i boka til nøyaktig samme formål.
2. Vi laget en metode som returnerte en int-array med 10 millioner tilfeldige elementer, der vi brukte hele spekteret til Random.nextInt()- metoden (fra: - 2.147.483.648 til: 2.147.483.647, hele int-spekteret)

```java
public static int[] getRandomList()
{
    Random random = new Random();
    int[] returListe = new int[10000000];
    for (int i = 0; i < returListe.length; i++)
    {
        int newValue = random.nextInt();
        if (newValue != Integer.MIN_VALUE)
        {
            returListe[i] = newValue;
        }
        else
        {
            i--;
        }
    }
    return returListe;
}
```

I hash-metodene våre, for å kunne ta et negativt tall og returnere kun positive tall (siden de skal brukes som indexer), gjør vi en sjekk på tallet før det returneres og hvis det er negativ vil det bli ganget med -1 og ellers bare returnert, men den minste int-verdien ganget med -1 vil forbli uendret og derfor har vi sikret at akkurat det tallet ikke kommer med i listen vår og gir oss en negativ index-verdi.

3.

```
Tid: 2.860975346sec
Antall kollisjoner: 8487366
Lastfaktor: 0.7499996812501355
```

Figure 1: Enkel kjøring

```
Tid: 2.671412628sec
Antall kollisjoner: 8493397
Lastfaktor: 0.7499996812501355
```

Figure 2: Gjennomsnitt av ti kjøringer

Vi kjørte en del ganger og fikk 8,4 millioner ± noen titusener kollisjoner hver gang, lastfaktoren blir jo alltid den samme, da vi på forhånd vet antall elementer og har satt størrelsen på hashtabellen, så fikk vi en gjennomsnittlig kjøre tid på 2.67 sekunder.

4.

```
Java sin HashMap tid: 3.637290318sec
```

Figure 3: Java HashMap gjennomsnittlig hastighet ved 10 gjennomganger

```
Vår hashtabell tid: 2.633054658sec
Java sin HadhMap tid: 4.116573044sec
```

Figure 4: Enkel kjøring av vår og Java sin kjøring

Vi slo Java! Java lå mellom gjennomsnittlig 3.64 sekunder og som sagt lå vi på 2.67 sekunder, det er jo nesten et helt sekund.