Øving 5

Oppgave 1

Programmet printer ut alle kollisjoner, og dermed lastefaktoren til hashtabellen.

Programmet tester også at nøklene (navnene) kan brukes til å hente ut riktig verdi. «Test»-feltet indikerer om testen gikk bra eller ikke. Testen feiler hvis en av nøklene returnerer feil verdi.

Utskrift fra kjøring

Vegard Johnsen =/= Jens S ├ªther Jordal
Erik Turmo Nords ├ªther =/= Ramtin Forouzandehjoo Samavat
Magnus Gjerlaugsen =/= Henrik Werner Lerv ├Ñg
Birthe Emilie Christiansen =/= Martin Sannes Hvistendahl
Jostein Johansen Aune =/= Elias Ward Heimdal
Heine M ├ªrde Brakstad =/= Trygve J ├©rgensen
Anders Emil Bergan =/= Kristians Janis Matrevics
Tini Tran =/= Hanne-Sofie Marie Scisly S ├©reide
Hallvard Torsvik Bamrud =/= Kristoffer Thorsdal Fredriksen
Oda Lib ├ªk =/= Markus Evald Dalbakk
Eric Bieszczad-Stie =/= Henrik Gulbrandsen Nilsen
Per Henrik Bergene Holm =/= Erik Turmo Nords ├ªther
Markus Hysing J ├©ssund =/= Martin Clementz
Magnus Rindal =/= Gia Hy Nguyen
Henrik Dybdahl Berg =/= Helene Nordby
Agnethe Kval-Engstad =/= Olav Sie Rotv ├ªr
Anders H ├©vik =/= Ari Maman
Julia Vik Rem ├©y =/= Agnethe Kval-Engstad
Miroja Sivachandran =/= Emil Skogheim
Markus ├ÿyen Lund =/= Kristoffer Longva Eriksen
Yasin Ali Marouga =/= Svein K ├Ñre S ├©restad
Zahid Andr ├® Kristiansen =/= Sondre Adrian Oksvik
Nicolai Forsberg Sommerfelt =/= Emil Johnsen
Jacob Forsdahl Iqbal =/= Karen Johannesen
Sander Kvenild =/= Emil Slettbakk
Vahideh Rezaei =/= Oscar Stentun Stadskleiv

Test:	success
Load factor:	0.527344
Number of collisions:	26
Number of elements:	135
Collisions per element:	0.192593

Oppgave 2

Dette er utskriften fra en kjøring av programmet. Prosentskalaen tar utgangspunkt i 10 000 019 elementer (primtall).

	Duration	Load_factor	Collisions	Collisions_per_element
Linear probing - 100% of test data	27.90645	1	4 999 447	0.499944
Linear probing - 99% of test data	1.27506	0.99	4 899 885	0.494937
Linear probing - 90% of test data	0.748504	0.9	4 049 086	0.449898
Linear probing - 80% of test data	0.595244	0.8	3 198 878	0.399859
Linear probing - 50% of test data	0.283536	0.5	1 249 627	0.249925
Double hashing - 100% of test data	3.80349	1	5000017	0.500001
Double hashing - 99% of test data	1.631548	0.99	4900475	0.494997
Double hashing - 90% of test data	1.177708	0.9	4049743	0.449971
Double hashing - 80% of test data	0.875055	0.8	3200380	0.400047
Double hashing - 50% of test data	0.415925	0.5	1250679	0.250135

Forsøk med binæroperatoren &

Ut av nysgjerrighet prøvde jeg å bruke en hashfunksjon som benyttet en bit-maske til å holde resultatet innenfor grensene til hashtabellen. Alternativet var å bruke modulo til å gjøre det samme. Jeg ville se om jeg kunne øke hastigheten.

Problemet er at man er nødt til å tvinge hashtabellen til å ha en 2-komplement-størrelse for at dette skal fungere. Det gjør at man får et veldig stort avvik mellom anmodet størrelse og faktisk størrelse. Jeg ble lurt av dette, og trodde at løsningen var raskere helt til jeg forstod at det var økningen i tabellens størrelse som ga forbedringen. Da jeg fylte hashtabellen 100%, viste det seg at hashfunksjonen ikke gir mye forbedring i tid likevel.

Det viste seg også å være en løsning som var inkompatibel med dobbel hashing. Problemet var at nøkler endte opp med en h2-verdi som skapte en uendelig løkke. Synderen var tabellstørrelsens 2-komplement-størrelse. Det er mulig å omgå problemet ved å minke h2-verdien hver gang man ender opp på den originale posisjonen, men det vil ytterligere sinke programmet. Dermed er det mer hensiktsmessig å sette tabellstørrelsen til neste nærmeste primtall slik at man ikke ender opp med dette problemet til å begynne med.