**Serie 6**

Theoretische Aufgaben

**1. Hashtabelle**

Hashtabelle Grösse m = 512

Hashfunktion: h(k) := ⎣m(kA mod 1)⎦ mit A := ( - 1)/2

a) Plätze der Schlüssel 2022, 2023, 2024, 2025

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| m: | 512 |  |  |  |
| A: | 0.618 |  |  |  |
| **k** | **kA** | **kA mod 1** | **m(kA mod 1)** | **⎣m(kA mod 1)⎦** |
| 2022 | 1249.665 | 0.665 | 340.339 | 340 |
| 2023 | 1250.283 | 0.283 | 144.773 | 144 |
| 2024 | 1250.901 | 0.901 | 461.206 | 461 |
| 2025 | 1251.519 | 0.519 | 265.640 | 265 |

b) Schlüsser der Kollision mit 2021 auslöst und eine PLZ von Bern ist

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| m: | 512 |  |  |  |
| A: | 0.618 |  |  |  |
| **k** | **kA** | **kA mod 1** | **m(kA mod 1)** | **⎣m(kA mod 1)⎦** |
| 2021 | 1249.047 | 0.047 | 23.906 | 23 |
| 3001 | 1854.720 | 0.720 | 368.640 | 368 |
| 3003 | 1855.956 | 0.956 | 489.507 | 489 |
| 3004 | 1856.574 | 0.574 | 293.940 | 293 |
| 3005 | 1857.192 | 0.192 | 98.374 | 98 |
| 3006 | 1857.810 | 0.810 | 414.807 | 414 |
| 3007 | 1858.428 | 0.428 | 219.241 | 219 |
| 3008 | 1859.046 | 0.046 | 23.674 | 23 |

**3. Einfügen von Schlüsseln in Hashtabelle**

Hashtabelle Grösse m = 11

Hashfunktion: **h’(k) := k mod m**

Ein Bild, das Text, Kreuzworträtsel enthält.

Automatisch generierte Beschreibunga) lineares Sondieren

Ein Bild, das Text, Kreuzworträtsel enthält.

Automatisch generierte Beschreibungb) quadratsiches Sondieren mit **c1 = 1** und **c2 = 3**

Ein Bild, das Text, Kreuzworträtsel enthält.

Automatisch generierte Beschreibungc) doppeltes Hashing mit **h2(k) = 1 + (k mod(m-1))**

**2. Laufzeiten Optimierung: Sortieren**

Suchen:

Die Erfolgreiche Suche bleibt gleich mit oder ohne Sortierung, sie endet beim gesuchten Element (Θ(1 + α), mit α = n/m (Anzahl Elemente/Grösse Hashtabelle). Die Erfolglose Suche hingegen läuft ohne Sortierung bis ans Ende einer Kette, mit Sortierung muss sie das jedoch z.T. nicht und hat so eine kürzere Laufzeit.

Einfügen:

Ohne Sortierung: Element wird am Kopf der verkettet Liste eingefügt, Worst-Case Laufzeit Ο(1) (unter Annahme dass Schlüssel nicht schon existiert, sonst davor eine erfolglose Suche).

Mit Sortierung: Es kommt eine (erfolgreiche) Suche nach der richtigen Position dazu, was eine Laufzeit von Θ(1 + α) hat.

Löschen:

Mit oder ohne Sortieren gleich. Ist die Liste doppelt verkettet, dann ist keine Suche nötig und die Laufzeit beträgt Ο(1). Ansonsten muss davor eine erfolgreiche Suche durchgeführt werde (Θ(1 + α)). Bei einfach verketteten Listen hat also Löschen und Suchen die gleiche asymptotische Laufzeit.

**4. Kollisionsauflösung**