

# Heimadæmi 08| TÖL203G

Donn Eunice Bartido Patambag | 2507943479 | deb5@hi

## [Línuleg Könnun]

### Heimadæmi (skila í Gradescope)

1. [Línuleg könnun] Segjum að lyklarnir A til G séu settir inn í tóma hakkatöflu með línulegri könnun. Við vitum ekki röðina á innsetningunum, en lokastaða hakkstöflunnar er þessi:

0	1	2	3	4	5	6
F	D	E	A	G	B	C

Gerðu ráð fyrir því að hakkataflan hafi 7 sæti og stærð hennar breytist ekki. Fyrir **hvern lykil** segið hvort hann gæti hafa verið **síðasti lykillinn** sem var settur inn í hakkatöfluna. Ef það er mögulegt sýnið þá innsetningaröð á lyklunum sem veldur því, en annars rökstyðjið að það sé ekki mögulegt.

*Vísbending:* Til dæmis getur D ekki verið síðasti innsetti lykillinn, því þá hefðu A eða E lent í sæti 1, en ekki D (því D er með hakkagildið 0).

Lykill	Hakkagildi
A	1
B	5
C	6
D	0
E	1
F	6
G	4

### Lausn 1.

- F** getur verið síðasti lykillinn sem var settur inn. Hann er í síðasta sætinu og hefur hakkagildi sem er stærri en hakkagildi öllum öðrum lyklum sem eru ekki í töflunni.
- D, E og A** geta ekki verið síðasti lykillinn sem var settur inn. Sæti 1, sem er staðsett á undan síðasta sætinu, hefur hakkagildið 0 og því geta einungis lyklar með hakkagildið 0 verið settir í það sætið. D, E og A hafa öll hærri hakkagildi.
- G, B og C** geta verið síðasti lykillinn sem var settur inn ef annaðhvort B eða C var sett inn í sæti 1. Sæti

## 2. [Línuleg könnun]

2. [Línuleg könnun] [Annars stigs könnun](#) (*quadratic probing*) í hakkatöflum virkar þannig að í stað þess að kíkja alltaf í hólfið strax á eftir (þ.e.  $h(k) + 1$ ), þá kíkjum við fyrst í hólfið  $h(k) + 1^2$ , síðan í hólfið  $h(k) + 2^2$ , svo  $h(k) + 3^2$ , o.s.frv. Þetta brýtur aðeins upp klösunina (*clustering*) sem getur orðið mjög slæm í línulegri könnun. Segjum að við höfum 11-staka hakkatöflu ( $M = 11$ ) og hakkafallið er  $h(k) = (3k + 5) \bmod 11$ . Sýnið hvernig lyklarnir 31, 67, 53, 34, 89, 40, 78, 77 hakkast inn í 11-staka töfluna. Sýnið útreikning á hverju gildi fyrir sig, í hvaða sæti það lendir og lokatöfluna.

### Lausn 2.

útreikningar á lyklum;

- **31:**  $h(31) = (3 * 31 + 5) \bmod 11 = 10$  Setja í sæti **10** það er laust
- **67:**  $h(67) = (3 * 67 + 5) \bmod 11 = 8$  setja í sæti **8** það er laust
- **53:**  $h(53) = (3 * 53 + 5) \bmod 11 = 10$  ætti að vera í sæti 10 en það er ekki laust þannig það fer í sæti **0**
- **34:**  $h(34) = (3 * 34 + 5) \bmod 11 = 8$  ætti að vera í sæti **8** en það er ekki laust þannig það fer í sæti **9**
- **89:**  $h(89) = (3 * 89 + 5) \bmod 11 = 8$  ætti að vera í sæti **8** en það er ekki laust og ekki **9, 10, 0** þannig það fer í sæti **1**
- **40:**  $h(40) = (3 * 40 + 5) \bmod 11 = 4$  setja í sæti **4** og það er laust
- **78:**  $h(78) = (3 * 78 + 5) \bmod 11 = 8$  ekki laust og ekki heldur **9, 10, 0, 1** fer í sæti **2**
- **77:**  $h(77) = (3 * 77 + 5) \bmod 11 = 5$  setja í sæti 5 það er laust.

### Lokataflan

0	1	2	3	4	5	6	7	8	9	10
<b>53</b>	<b>1</b>	<b>78</b>		<b>40</b>	<b>77</b>			<b>67</b>	<b>34</b>	<b>31</b>

## 3.[Línuleg könnun]

3. [Línuleg könnun] Bætið aðferðinni `longestCluster()` við [LinearProbingHashST](#). Aðferðin rennir í gegnum fylkið `keys` í og finnur lengd á lengstu röð ekki-null staka í fylkinu og skilar þeirri tölu. Bætið líka við aðferðinni `size()`, sem skilar fjölda lykla í töflunni (þ.e.  $N$ ) og `capacity()` sem skilar stærð fylkisins (þ.e.  $M$ ).
- Lesið svo öll orðin í bókinni „War and Peace“ ([war+peace.txt](#)) inn í hakkatöfluna og prentið út fjölda lykla, stærð töflunnar og lengd lengsta klasa (*cluster*). Skilið kóða fyrir nýju aðferðirnar og skjáskoti af keyrslu.

## Kóða fyrir nýju aðferðir

- `capacity()`

```
1 usage
public int capacity() {
    return m;
}
```

- `longestCluster()`

```
1 usage
public int longestCluster() {
    int maxCluster = 0;
    int currCluster = 0;
    for (int i = 0; i < m; i++) {
        if (keys[i] != null) {
            currCluster++;
        } else {
            maxCluster = Math.max(maxCluster, currCluster);
            currCluster = 0;
        }
    }
    return maxCluster;
}
```

- `size()`

```
*/
public int size() { return n; }
```

## Skjáskót af keyrslu

```
Run: LinearProbingHashST x MinnLinear x
/Users/donnacruz/Library/Java/JavaVirtualMachines/openjdk-19.0.2/Content
Number of keys: 41009
Capacity of table: 131072
Length of longest cluster: 36
Process finished with exit code 0
```

## 4. [Táknatöflur]

4. [Táknatöflur] Skriðið forrit til að bera saman hraða á tvíleitartrjám ([BST](#)), rauð-svart trjám ([RedBlackBST](#)), hakatöflum með sjálfstæðri keðjun ([SeparateChainingHashST](#)) og hakatöflum með línulegri könnun ([LinearProbingHashST](#)). Forritið ykkar á að lesa inn orðin í bókinni „War and Peace“ ([war+peace.txt](#)) inn í fylki (það eru tæplega 565 þús. orð í skránni) og setja orðin í þessar fjórar gagnagrindur og tímamæla innsetningarnar. Lesið svo orðin í bókinni „A Tale of Two Cities“ ([tale.txt](#)) inn í fylki (tæplega 140 þús. orð) og leitið að öllum orðum hennar í gagnagrindunum 10 sinnum.

Skilið niðurstöðum tímamælinganna á innsetningu orðanna í „War and Peace“ í þessar fjórar gagnagrindur og tímamælingunum á leitinni að orðunum í „A Tale of Two Cities“ í gagnagrindunum fjórum. Ekki þarf að skila forritskóðanum.

## Lausn 4.

Tímamælinganna á innsetningu orðanna í „War and Peace“ í þessar fjórar gagnagrindur

- **Insertion**
  - BST Insertion Time: 308 ms
  - RedBlackBST Insertion Time: 302 ms
  - SeparateChainingHashST Insertion Time: 181 ms
  - LinearProbingHashST Insertion Time: 74 ms
- **Search**
  - BST Search Time: 29 ms
  - RedBlackBST Search Time: 40 ms
  - SeparateChainingHashST Search Time: 45 ms
  - LinearProbingHashST Search Time: 33 ms

Tímamælinganna á innsetningu orðanna í „Tale of Two Cities“ í þessar fjórar gagnagrindur

- **Insertion**
  - BST Insertion Time: 91 ms
  - RedBlackBST Insertion Time: 114 ms
  - SeparateChainingHashST Insertion Time: 67 ms
  - LinearProbingHashST Insertion Time: 34 ms
- **Search**
  - BST Search Time: 6 ms
  - RedBlackBST Search Time: 11 ms
  - SeparateChainingHashST Search Time: 9 ms
  - LinearProbingHashST Search Time: 13 ms

## 5.[Tákntöflur]

### Lausn 5.

Forritið

```
import java.util.ArrayList; import java.util.List;
```

```
public class daemi5 {
```

```
    public static List<String> generateStrings(int n) {  
        List<String> firstSet = generateStrings(n, 'a');  
        List<String> secondSet = generateStrings(n, 'b');  
  
        // Sameina báða listana af strengjum  
        firstSet.addAll(secondSet);  
  
        return firstSet;  
    }
```

```
private static List<String> generateStrings(int n, char base)
{
    // Base case
    if (n == 0) {
        List<String> list = new ArrayList<>();
        list.add("");
        return list;
    }

    // Endurkvæmni
    List<String> subList = generateStrings(n - 1, base);
    List<String> list = new ArrayList<>();

    // Búa til 2N strengi út frá subList með öfugum
    stafasetti
    for (String s : subList) {
        String reversed = new
StringBuilder(s).reverse().toString();
        list.add(base + s + base);
        list.add(base + reversed + base);
    }

    return list;
}

public static void main(String[] args) {
    int n = 4;
    List<String> strings = generateStrings(n);

    System.out.println("Hash gildi: " +
strings.get(0).hashCode());
    for (String s : strings) {
        System.out.println(s);
    }
}
```

**Keyrsla á forrit**

LinearProbingHashST x daemi5 x

↑

↓

↶

↷

📄

🗑

/Users/donnacruz/Library/Java/JavaVirtualMachines/openjdk

Hash gildi: 312017024

aaaaaaaa

aaaaaaaa

aaaaaaaa

aaaaaaaa

aaaaaaaa

aaaaaaaa

aaaaaaaa

aaaaaaaa

aaaaaaaa

aaaaaaaa

In [ ]: