Heimadæmi 08| TÖL203G

Donn Eunice Bartido Patambag | 2507943479 | deb5@hi

[Línuleg Könnun]

Heimadæmi (skila í Gradescope)

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

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

Gerið 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).

Lausn 1.

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

Þetta er það sem við vitum

C - þarf að fara á undan F annars hefði F farið í C þar sem F er með hakkagildið 6.
Ef C fer á undan F þá er þetta röðuninn útfrá vísbending

0	1	2	3	4	5	6
F	**D**	**E**				**C**

Fyrst kemur C - því C þarf að vera á undan F
Svo F - Því F þarf að koma á eftir C og á undan D
Svo D - Því D þarf að koma á eftir F
Svo E - því E þarf að koma á eftir D en á undan A.

Allt þetta er útfrá hakkagildinu sem að við fengum fyrir dæmið.

Útfrá þessu eru 3x lyklar sem gætu mögulega verið síðustu lyklar í röðinni.

A, B og G

Hér eru mögulegar innsetingar útfrá því

Ef A - er síðasti lykill inn

Fyrst G svo B svo síðasti lykill A

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

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

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

Eða

Fyrst B svo G og síðasti lykill A

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

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

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

Ef G er síðasti lykill inn

Fyrst B svo A og síðasti lykill G

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

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

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

Eða

Fyrst A svo B og síðasti lykill G

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

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

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

Ef B er síðasti lykill

Fyrst A svo G og síðasti lykill B

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

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

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

Eða

Fyrst G svo A og síðasti lykill B

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

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

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

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 hólf $h(k) + 1^2$, síðan í hólf $h(k) + 2^2$, 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 $h(k) = (3k + 5)$ **mod** 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.

0	1	2	3	4	5	6	7	8	9	10
53	**89**			**40**	**77**	**78**		**67**	**34**	**31**

Við vitum að

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 hólf $h(k) + 1^2$, síðan í hólf $h(k) + 2^2$, svo $h(k) + 3^2$, o.s.frv.

31: h(31) = (3 * 31 + 5) = 98
9 mod 11 = 10
Þá fer 31 í sæti 10.
67: h(67) = (3 * 31 + 5) = 206
206 mod 11 = 8
Þá fer 67 í sæti 8

53: h(53) = (3 * 53 + 5) = 164
164 mod 11 = 10
Þar sem sæti 10 er ekki laust hoppar 53 í næsta sæti sem er 0

34: h(34) = (3 * 34 + 5) = 107
107 mod 11 = 8
Þar sem sæti 8 er ekki laust hoppar 34 í næsta sæti sem er 9

89: h(89) = (3 * 89 + 5) = 272
272 mod 11 = 8
Þar sem sæti 8 er ekki laust hoppar 89 í næsta sæti sem er 9 en það sem það er ekki laust heldur hoppar 89 í annað sæti ($h(k) + 2^2$- ss hoppar um 4 sæti). 89 fer í sæti 1

40: h(40) = (3 * 40 + 5) = 125
125 mod 11 = 4
40 fer í sæti 4 (það er laust)

78: h(78) = (3 * 78 + 5) = 239
239 mod 11 = 8
Þar sem sæti 8 er ekki laust hoppar 89 í næsta sæti sem er 9 en það sem það er ekki laust heldur hoppar 78 í annað sæti ($h(k) + 3^2$, hoppar um 9 sæti). 78 fer í sæti 6

77: h(77) = (3 * 77 + 5) = 236
236 mod 11 = 5

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;
}
```

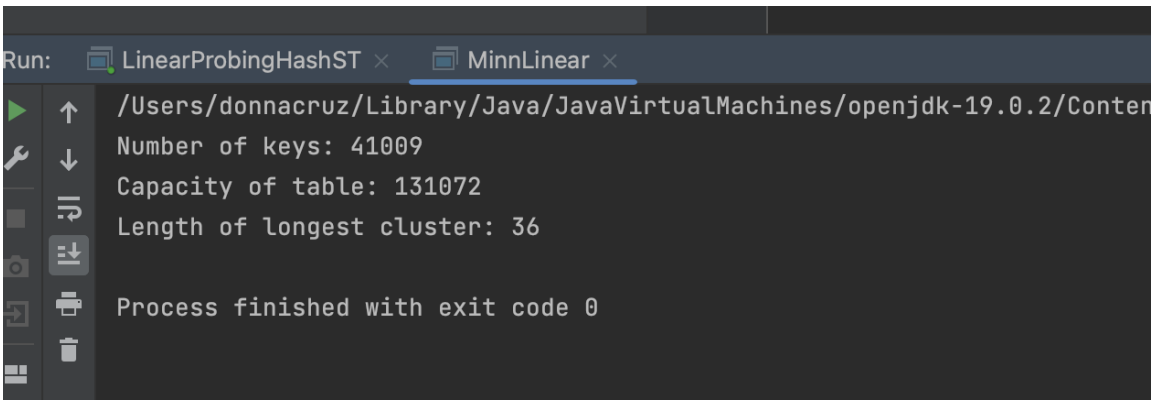
- **longestCluster()**

```
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



4. [Táknatöflur]

4. [*Táknatöflur*] Skrifið forrit til að bera saman hraða á tvíleitartrjám (BST), rauð-svart trjám (RedBlackBST), hakkatöflum með sjálfstæðri keðjun (SeparateChainingHashST) og hakkatöflum 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: 0.158
  - RedBlackBST Insertion Time: 0.201
  - SeparateChainingHashST Insertion Time: 0118
  - LinearProbingHashST Insertion Time: 0.038

Lesa orðin í bókinni "A Tale of Two Cities" inn í fylki (tæplega 140k orð) og leitið að öllum orðum hennar í gagnagrindunum 10 sinnum.

- Niðurstöður
  - BST Search Time: 0.34
  - RedBlackBST Search Time: 0.33
  - SeparateChainingHashST Search Time: 0.126
  - linearProbingHashST Search Time: 0.066

5.[Tákntöflur]

Lausn 5.

Forritið

```java
import java.util.List; import java.util.ArrayList;
2 usages  Donna Cruze *
public class daemi5 {
    no usages  Donna Cruze *
    public int hashCode(String string) {
        int hash = 0;
        for (int i = 0; i < string.length(); i ++)
            hash = (hash * 31) + string.charAt(i); return hash;
    }
    1 usage  new *
    private List<String> generateStringsInput(int n) {
        String[] values = {"Aa", "BB"}; List<String> strings = new ArrayList<>();
        generateStrings(n, currentIndex: 0, strings, currentString: "", values); return strings;
    }
    2 usages  Donna Cruze *
    private void generateStrings(int n, int currentIndex, List<String> strings, String currentString, String[] values) {
        if (currentIndex == n) { strings.add(currentString); return;
        }
        for (String value : values) {
            String newValue = currentString + value; int newIndex = currentIndex + 1;
            generateStrings(n, newIndex, strings, newValue, values); }
    }
    no usages  Donna Cruze *
    public static void main(String[] args) {
        daemi5 hashAttack = new daemi5();
        List<String> hashAttackInput = hashAttack.generateStringsInput( n: 3);
        for (String string : hashAttackInput) { System.out.println(string); System.out.println(string.hashCode());
        } }
}
```

Output á forrit;

```
/Users/donnacruz/Library/Java/JavaVirtualMachines/openjdk-19.0.2/Contents/
AaAaAa
1952508096
AaAaBB
1952508096
AaBBAa
1952508096
AaBBBB
1952508096
BBAaAa
1952508096
BBAaBB
1952508096
BBBBAa
1952508096
BBBBBB
1952508096

Process finished with exit code 0
```

In [ ]: