LINFO 1121

DATA STRUCTURES AND ALGORITHMS

UCLouvain

TP4: Dictionnaires

Question 4.1.1 Implementation of a map

Any ideas?

Unordered list, ordered list, BST, array-based mapping, ...

Question 4.1.2 a + b = x

Given a set S of size n, a number x: find if there exists a pair of element a,b from S such that a + b = x

We have seen that we can solve this problem by first sorting the elements.

This method is in O(n log(n)).

Can we do better ?

Question 4.1.2 a + b = x

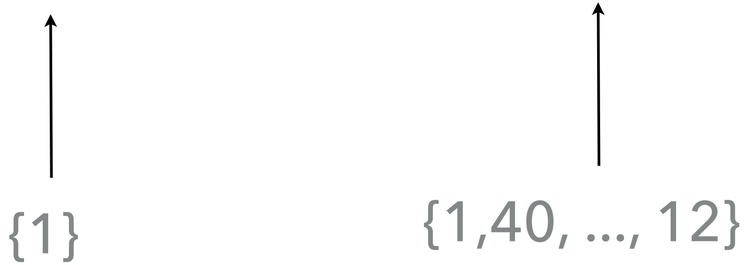
Given a set S of size n, a number x: find if there exists a pair of element a,b from S such that a + b = x

We can notice that if such pair exist, then we have a = x - b

For each element v of S, we must find x - v

x = 36

1, 40, 18, 34, 23, 55, 12



Question 4.1.2 a + b = x

Given a set S of size n, a number x: find if there exists a pair of element a,b from S

such that a + b = x

We can notice that if such pair exist, then we have a = x - b

For each element v of S, we must find x - v

x = 46

1, 40, 18, 34, 23, 55, 12

X = {1, 40, 18, 34, 23, 55, 12}

Is 46 - 1 = 45 in the set X ?

Is 46 - 34 = 12 in the set X ?

O(n + n) time complexity ! Downsides ?

Question 4.1.3 Modulos

$$(a + b) \,\%\, M \qquad ((a \,\%\, M) + b) \,\%\, M$$

Prouvez que ces deux valeurs sont égales.

$$a = (c \cdot M) + (a \,\%\, M) \qquad \text{où } c = \lfloor \frac{a}{M} \rfloor$$

$$\begin{aligned}(a + b) \,\%\, M &= ((c \cdot M) + (a \,\%\, M) + b) \,\%\, M \\ &= ((a \,\%\, M) + b) \,\%\, M\end{aligned}$$

Question 4.1.3 Java hash for String

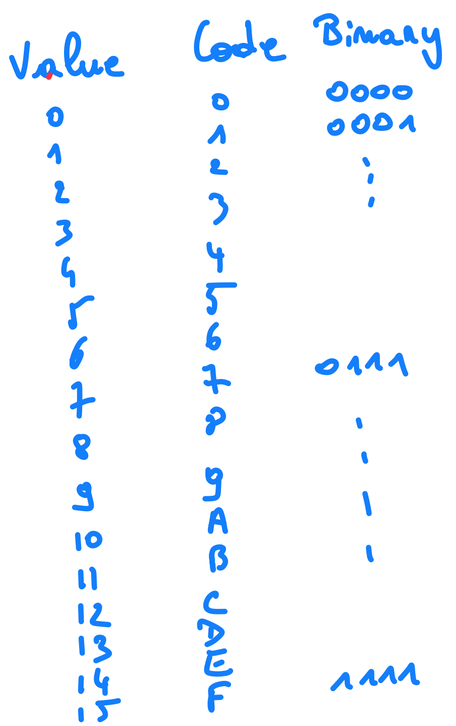
Java hash formula for String: $\sum_{i=1}^{n} 31^{n-i} s_i$

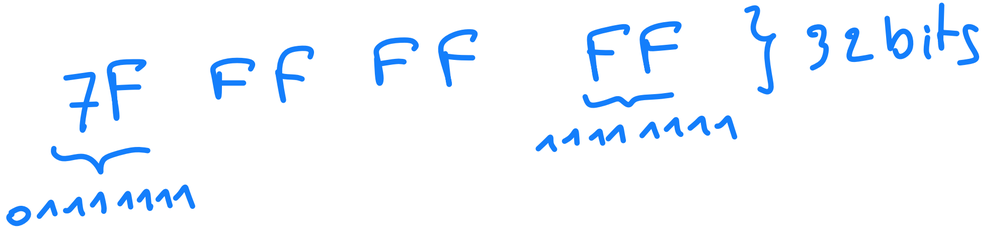
Complexity is ~ n

But, Java uses a cache. So computing the hash N times requires only n + N operations

Question 4.1.4 Générer une clé a partir d'un hash

Private int hash(Key x) { return (x.hashCode() & 0x7fffffff) % M);





Question subsidière: x & 0x7FFFFFFF = abs(x) ?

X Bits	Signed value (Two's complement)	X & F	Signed value (Two's complement) of X&F
0000	0	000	0
0001	1	0001	1
0010	2	0010	2
0011	3	0011	3
1100	-4	0100	4
1101	-3	0101	5
1110	-2	0110	6
1111	-1	0111	7

Question 4.1.5 types de tables de hachages, map et SETS

Nom	Type	Methode	Notes
HashMap	Dictionnaire	Chaining (+ Tree)	
HashTable	Dictionnaire	Chaining	Threadsafe
ConcurrentHashTable	Dictionnaire	Chaining	Threadsafe
HashSet	Set	Chaining + Tree	
ConcurrentHashSet	Set	Chaining	Threadsafe
LinkedHashMap	Dictionnaire	Chaining	Maintien l'ordre d'insertion
IdentityHashMap	Dictionnaire	Chaining	Vérifie les clés par références
WeakHashMap	Dictionnaire	Chaining	Weak keys
TreeSet	Set	Tree	
TreeMap	Dictionnaire	Tree	

Question 4.1.6 COLLISIONS

Hash	0	1	2	3	4	5	6	7	8	9	10	11
Value			14		28	-3		-1	8			

What happens when 19 is added to the set ? (inserted at index 8)

Question 4.1.6 COLLISIONS

Original

Hash	0	1	2	3	4	5	6	7	8	9	10	11
Value			14		28	-3		-1	8			

Separate chaining

Hash	0	1	2	3	4	5	6	7	8	9	10	11
Value			14		28	-3		-1/19	8			

Linear probing

Hash	0	1	2	3	4	5	6	7	8	9	10	11
Value			14		28	-3		-1	8	19		

Question 4.1.6 COLLISIONS

Separate chaining

Hash	0	1	2	3	4	5	6	7	8	9	10	11
Value			14		28	-3		-1/19	8			

Linear probing

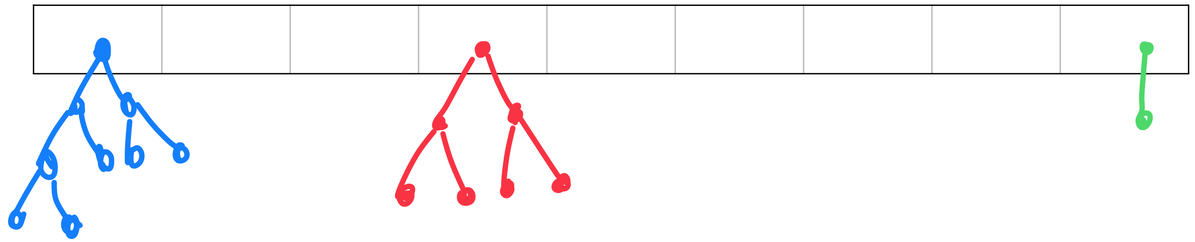
Hash	0	1	2	3	4	5	6	7	8	9	10	11
Value			14		28	-3		-1	8	19		

Influence on run time ?

In which case use one or the other ?

Linear Probing vs Separate Chaining

HashMap use separate chaining and a tree in case of large number of collisions. This prevent the O(n) get/put induced with large lists.



Linear probing can not do that, a bad hash function will cause big cluster and you must pay the O(n) get/put.

Question 4.1.7 LOAD FACTOR

load factor = number of entry in the hashmap / capacité

For separate chaining, this is also the the mean number of entry per list

In linear probing, this is the proportion of filled entry in the array

Question 4.1.9 Linear probing avec marqueur

Hash	0	1	2	3	4	5	6	7	8	9	10	11
Value			14		28	-3		-1	8			

19 goes into bucket 8, but with linear probing it is placed in the next available spot

Hash	0	1	2	3	4	5	6	7	8	9	10	11
Value			14		28	-3		-1	8	19		

Question 4.1.8 Linear probing avec marqueur

Hash	0	1	2	3	4	5	6	7	8	9	10	11
Value			14		28	-3		-1	8	19		

If 8 is deleted, classical linear probing gives us this

Hash	0	1	2	3	4	5	6	7	8	9	10	11
Value			14		28	-3		-1	19			

Question 4.1.8 Linear probing avec marqueur

Hash	0	1	2	3	4	5	6	7	8	9	10	11
Value			14		28	-3		-1	8	19		

The modified linear probing gives this result

Hash	0	1	2	3	4	5	6	7	8	9	10	11
Value			14		28	-3		-1		19		

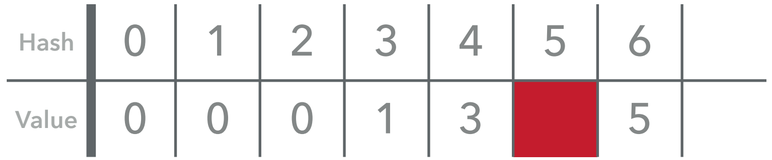
Better ? Worse? Pros/cons ?

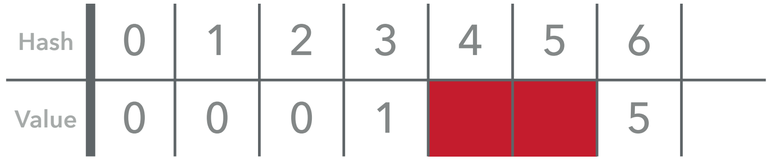
Question 4.1.8 Linear probing avec marqueur

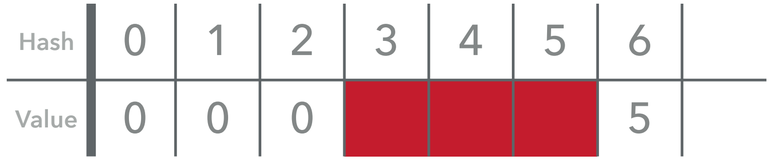
Two problems:

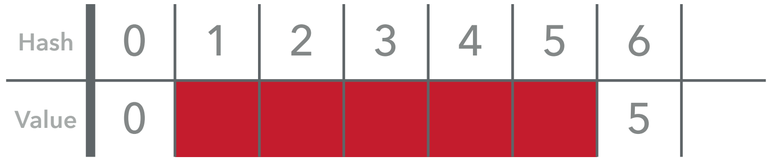
- Nothing removes the markers, gets and puts are slower

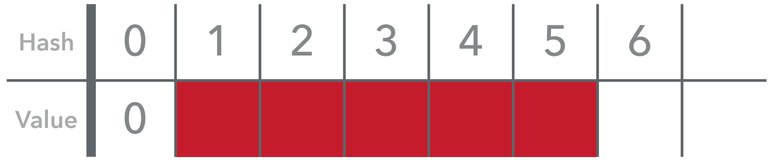
- As a general rules, one does more get/puts than deletes

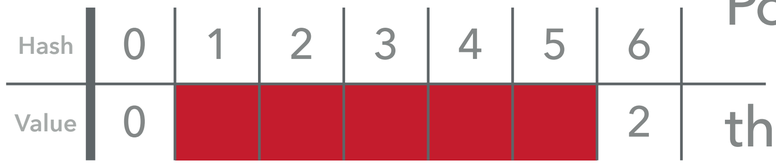












Possible to do "smarts" put, or to re-order the arrays, but the methods become complex

Hash on Integer ?

- What is the hash of int 42 ?
- Binary representation of 42 = 101010
- 42 =

$$1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$$

↳ because base 2

A string is in base 26: "LINFO"

L → 12, I → 9, N → 14, F → 6, O → 15

$$12 \cdot 26^4 + 9 \cdot 26^3 + 14 \cdot 26^2 + 6 \cdot 26^1 + 15 \cdot 26^0$$

ComputinG the hash of A string of size M

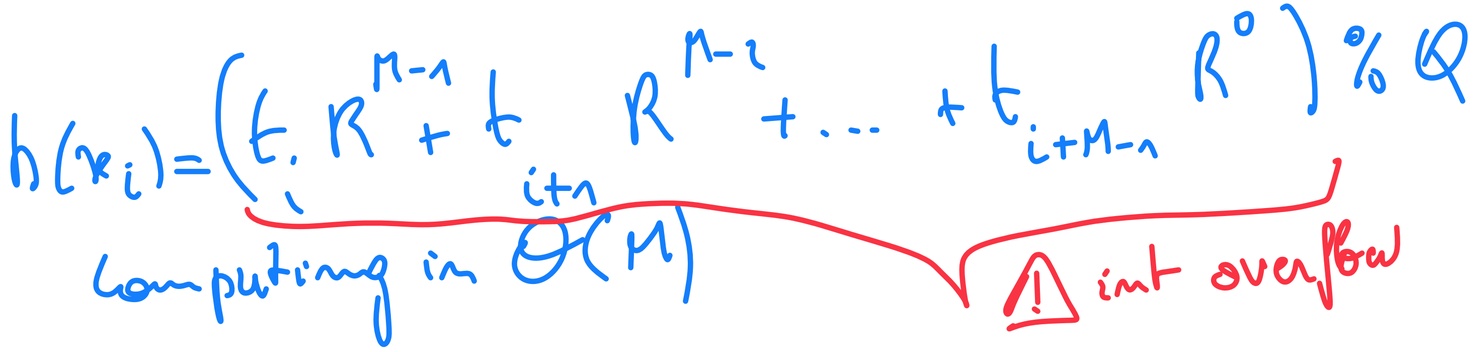
- Time complexity ? t[I,…,i+m-1]

$$\left(t_i R^{m-1} + t_{i+1} R^{m-2} + \cdots + t_{i+m-1} R^{0} \right) \% Q$$

Alphabet Size

ASCII 256 (8 bits)

HorNer Rule: ComputinG




```
private long hash(String key, int M)
{  // Compute hash for key[0..M-1].
   long h = 0;
   for (int j = 0; j < M; j++)
      h = (R * h + key.charAt(j)) % Q;
   return h;
}
```

$(A+B)\ \%\ Q = (A\ \%\ Q + B\ \%\ Q)\ \%\ Q$

$(A \cdot B)\ \%\ Q = ((A\ \%\ Q) \cdot (B\ \%\ Q))\ \%\ Q$

Incremental Hash on String

$$s = t_0 t_1 \ldots t_n$$

$$x_i = t_i R^{M-1} + t_{i+1} R^{M-2} + \ldots + t_{i+M-1} R^0$$

$$\begin{aligned} x_{i+1} &= t_{i+1} R^{M-1} + \ldots + t_{i+M-1} R^1 + t_{i+M} R^0 \\ &= x_i R - t_i R^M + t_{i+M} = (x_i - t_i R^{M-1}) R + t_{i+M} \end{aligned}$$

But ! $h(x_i) = x_i \mod Q$

$$h(x_i + 1) = (((h(x_i) + Q - t_i R^{M-1} \,\%\, Q) \,\%\, Q) R + t_{i+M}) \,\%\, Q$$