Hash Functions → Sunt folosite pentru căutare

$f_h : \mathbb{N}^m \to \mathbb{N}^{32}$, $m \in \mathbb{N}^*$ (Exemplu)

f_1 → ieșire de 3 caractere

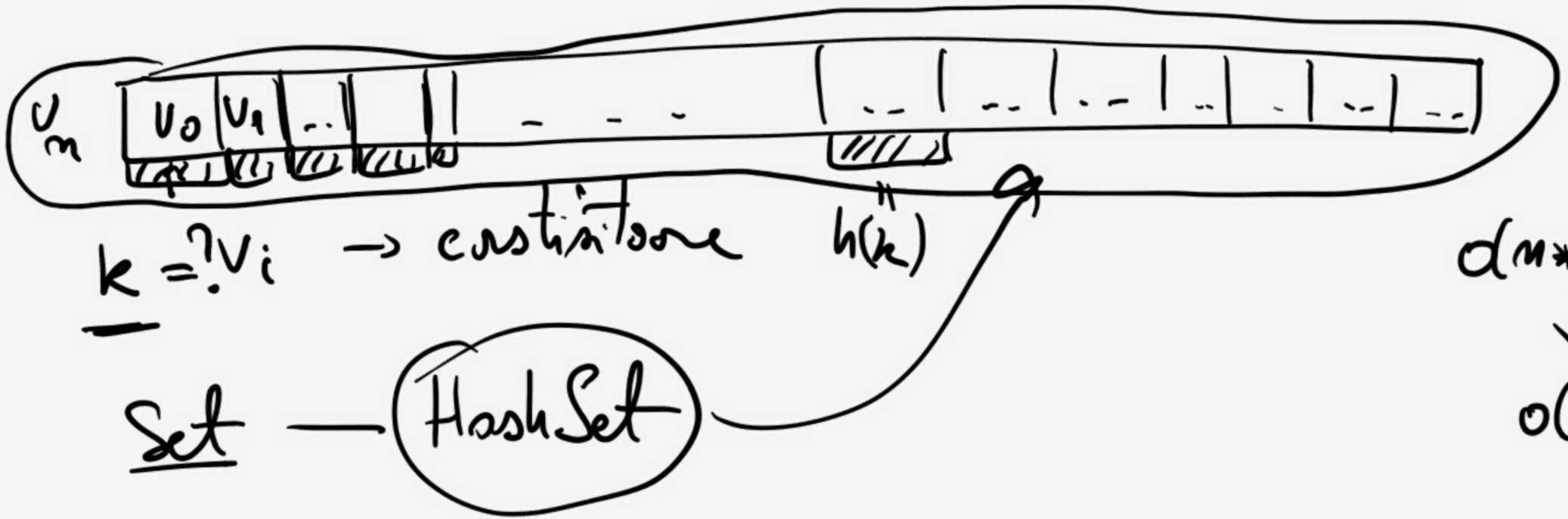
$f_1("abc") = "abc"$

$f_1("abcd") = "abc"$

$f_1("ab") = "abb"$

$a = b \Rightarrow f_h(a) = f_h(b)$

Funcții hash criptografice < MD5 (not secure)
SHA



$a \neq b$, $h(a) = h(b)$ → coliziune de hash

Ex. 1 Funcție hash pe șiruri de caractere, rezultatul funcției pe 32 biți.

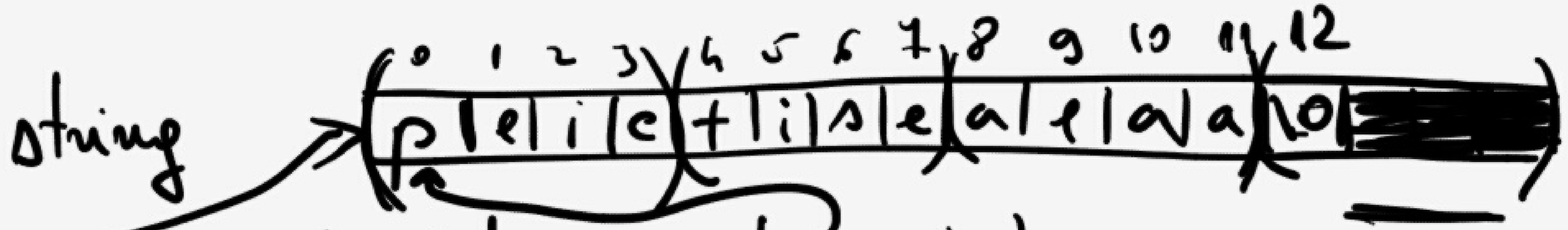
"abcdef..." → int →

↓ nr. pe 32

```
int hash(char *string){
    int word=0; int sum=0;
    int i;
    for(i=0; i<strlen(string); i++){
→       word = (word << 8) | string[i];
        if(i%4==0){sum += word; word=0;}
    }
    return sum+word;
}
```

```
11001010 << 2
00101000

                 'a'
0x00000012 << 8
0x00001200
0x00000034 = 'b'
0x00001234
```




```
int hash (char * string) {
    int * sequence = (int*) string;
    int i;
    int sum=0;
    for(i=0; strlen(string)/4; i++) {
        sum += sequence[i];
    }
    return sum;
}
```

$f_h(x) = 200;$

$x_1 = x_2$

✓

(ALEXANDR)

$a = b \longrightarrow \underline{h}(a) = h(b)$

$c_1 = a_1 + ib_1$

$c_2 = a_2 + ib_2$

$c_1 = c_2$?

def = a.l. $c_1 = c_2 \Leftrightarrow |c_1| = |c_2|$

$|c_1| = |c_2| \Rightarrow$

$\Rightarrow h(c_1) = h(c_2)$

Map → strutture associative

key → valore

Set $((k_0, v), (k_1, v) \dots (k_n, v))$,

a.l. $\forall\, k_i \neq k_j$

$\Rightarrow$ Map

Map → V find(k)
→ bool exists(k)
→ unsigned count()
→ void put(K, V)

```
struct Pair {
    K key;
    V value;
};
```

V_n = | Pair | Pair | ... | | Pair |

- find $O(n)$
- exists $O(n)$
- count $O(1)$
- put $O(n+n) = O(n)$

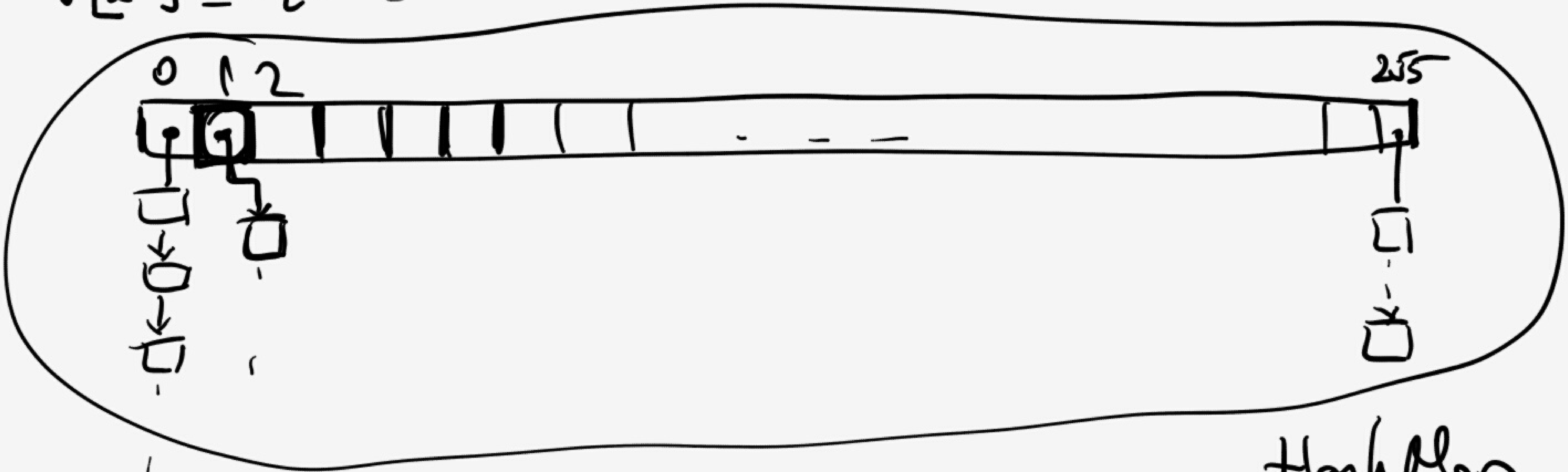
HashMap

HashMap → struct Pair {
K key;
V value;
};

Def. $f_h(k)$ → a.î. nr. val. posibile de hash să fie relativ mic

$f(h)$ cu rezultat pe 8 biți

$V[2^8] \equiv V[256]$



HashMap

Dacă pentru o aplicație se definește o funcție hash
care să nu rezulte în coliziuni, atunci accesul în HashMap
se face în $O(1)$.