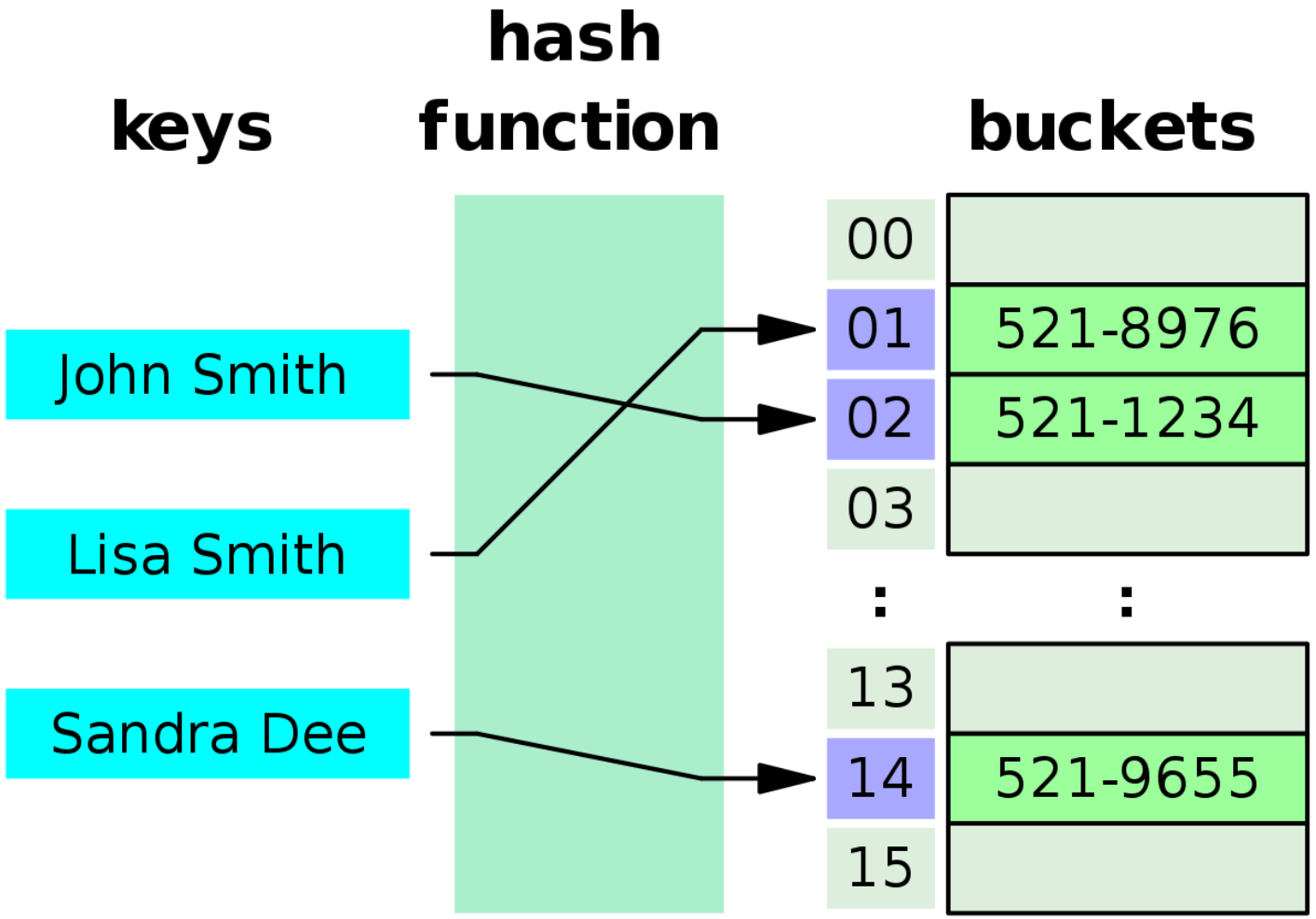
Хештаблици

Търсене с хеширане



Търсене с хеширане

- Търсене без сравнение;
- Елементът се намира по адрес, а не по стойност;
- Използва се хеш функция;
- При няколко елемента с един адрес има колизия;

Хеш функция

Изисквания:

- Бързо изчисляване;
- Минимум колизии;
- Равномерно разпределение на стойностите;
- Да определя малък размер на хеш таблицата.

Пр: Хеш функция при ключ число – деление с остатък

- Ако ключът е число и максималния размер на таблицата е n:

$$h(k) = k \,\%\, n$$

- Не е добре n да е число степен на двойката.
  Пр: $n = 2^4 = 16$ и $k = 173$ (10101101), $173\%16 = 13$ (1101)

- Най-често за n се избира просто число.

Пр: Хеш функция при ключ число – мултипликативно [Knuth-3/1968]

- Осигурява добро разпределение на ключовете;
- Ако ключът е число и максималният размер на таблицата е n се избира константа $0 < a < 1$:

$$h(k) = \lfloor n(ka - \lfloor ka \rfloor) \rfloor$$

Пр: Хеш функция при ключ текст - 1

$$h(k) = \left(\sum_{i=0}^{n-1} k[i] \right) \% n$$

```
static int HashFunction(string[] array, string s)
{
    var total = 0;
    var c = s.ToCharArray();

    for (int k = 0; k <= c.Length; k++)
        total += (int)c[k];

    return total % array.Length;
}
```

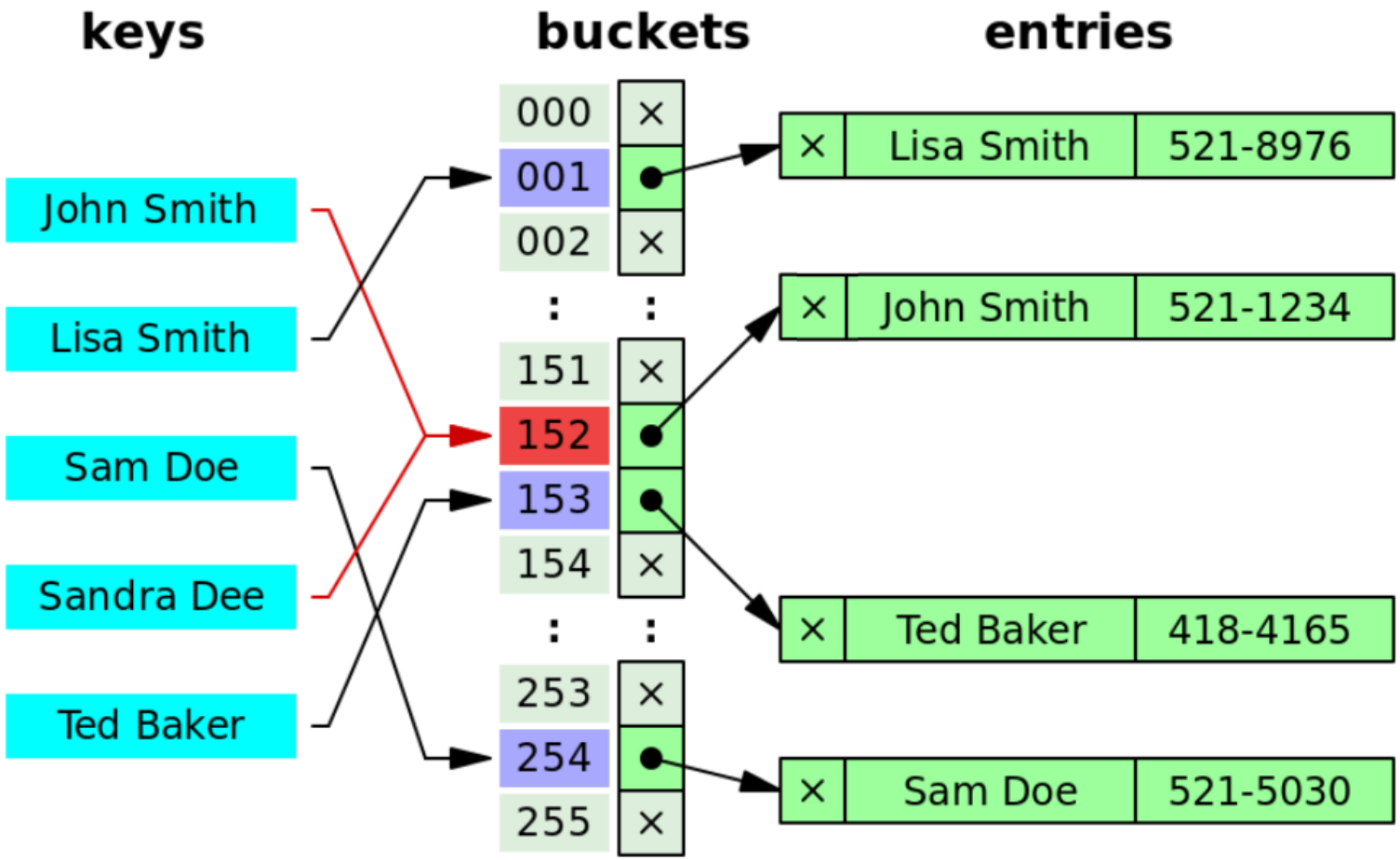
Пр: Хеш функция при ключ текст - 2

$$h(k) = \left(\sum_{i=0}^{n-1} k[i]S \% n \right)$$

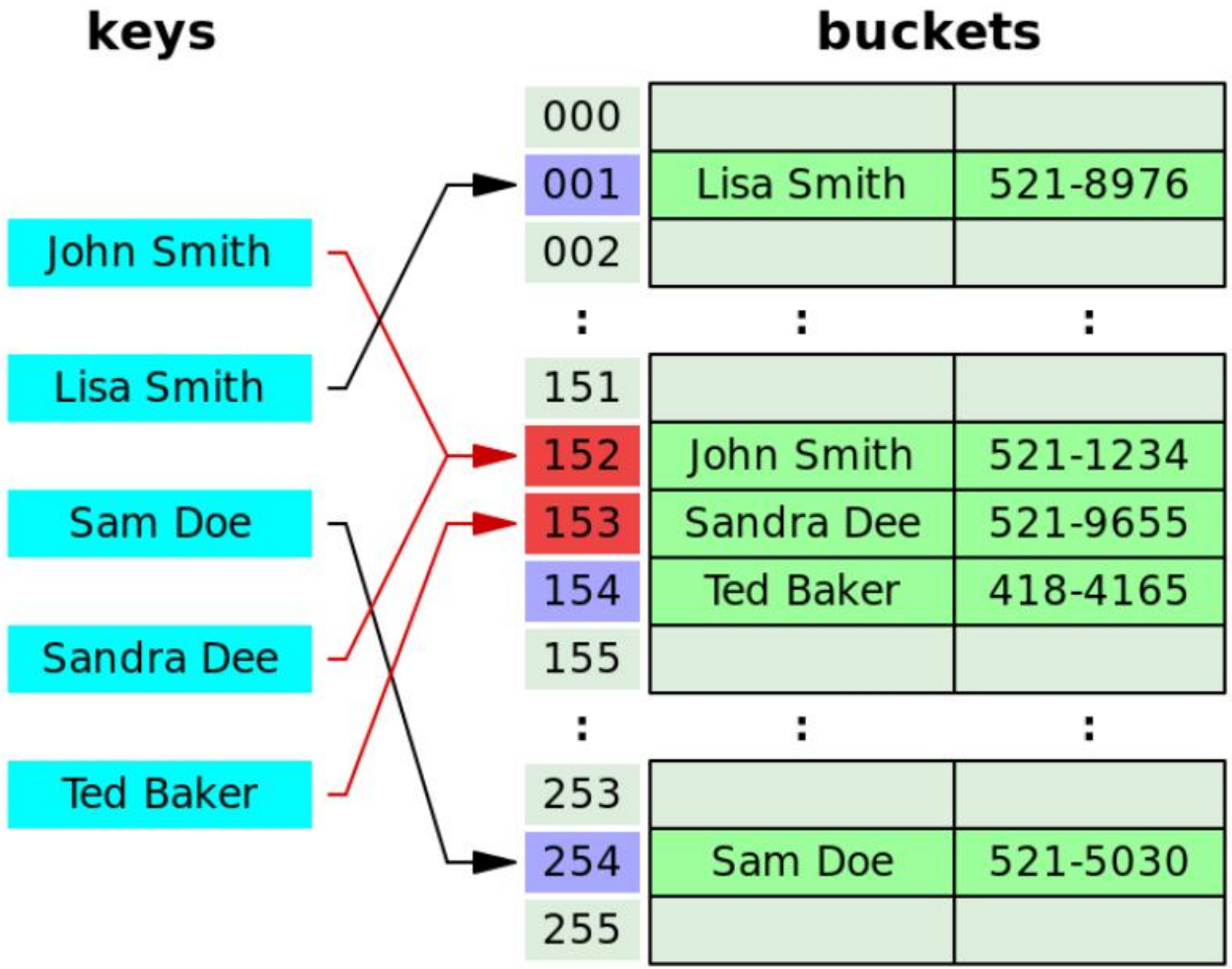
```
static int HashFunction2(string[] array, string t)
{
    var s = 32;
    var key = 0;
    var c = t.ToCharArray();

    for (int k = 0; k <= c.Length; k++)
        key = (key * s + (int)c[k]) % array.Length;

    return key;
}
```

- Осигурява по-добро разпределение на ключовете;
- Недостатък е големия брой деления. Компромисен вариант е да се взимат само част от буквите.

Колизии


keys
buckets
entries
John Smith
Lisa Smith
Sam Doe
Sandra Dee
Ted Baker
000
001
002
151
152
153
154
253
254
255
Lisa Smith
521-8976
John Smith
521-1234
Ted Baker
418-4165
Sam Doe
521-5030

Затворено хеширане



Отворено хеширане