

Vistula, IT Faculty, 2014

Algorithms and Complexity

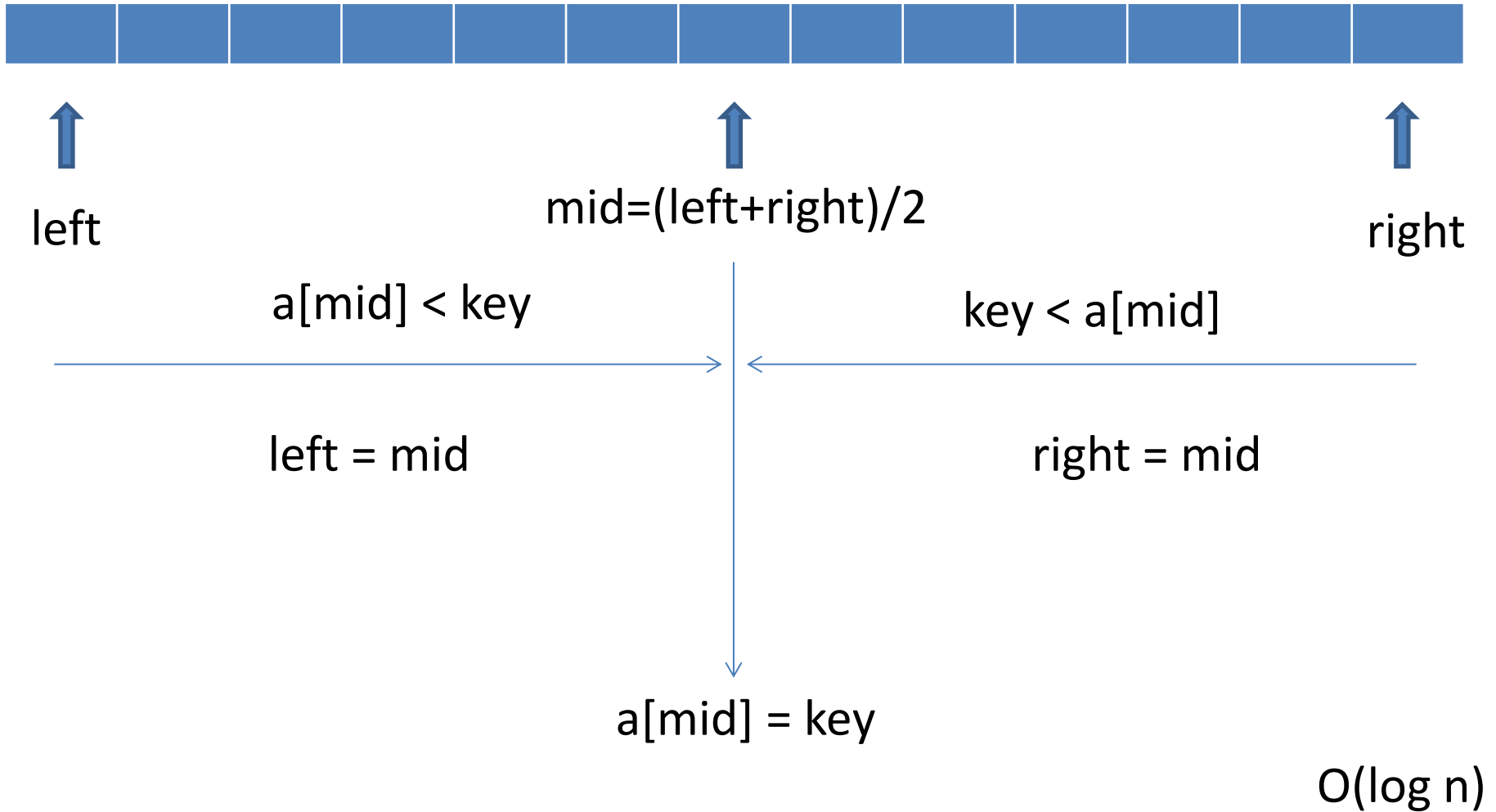
Dmitry A. Zaitsev

<http://daze.ho.ua>

Lecture 10:

Search algorithms

Dichotomy in a sorted table



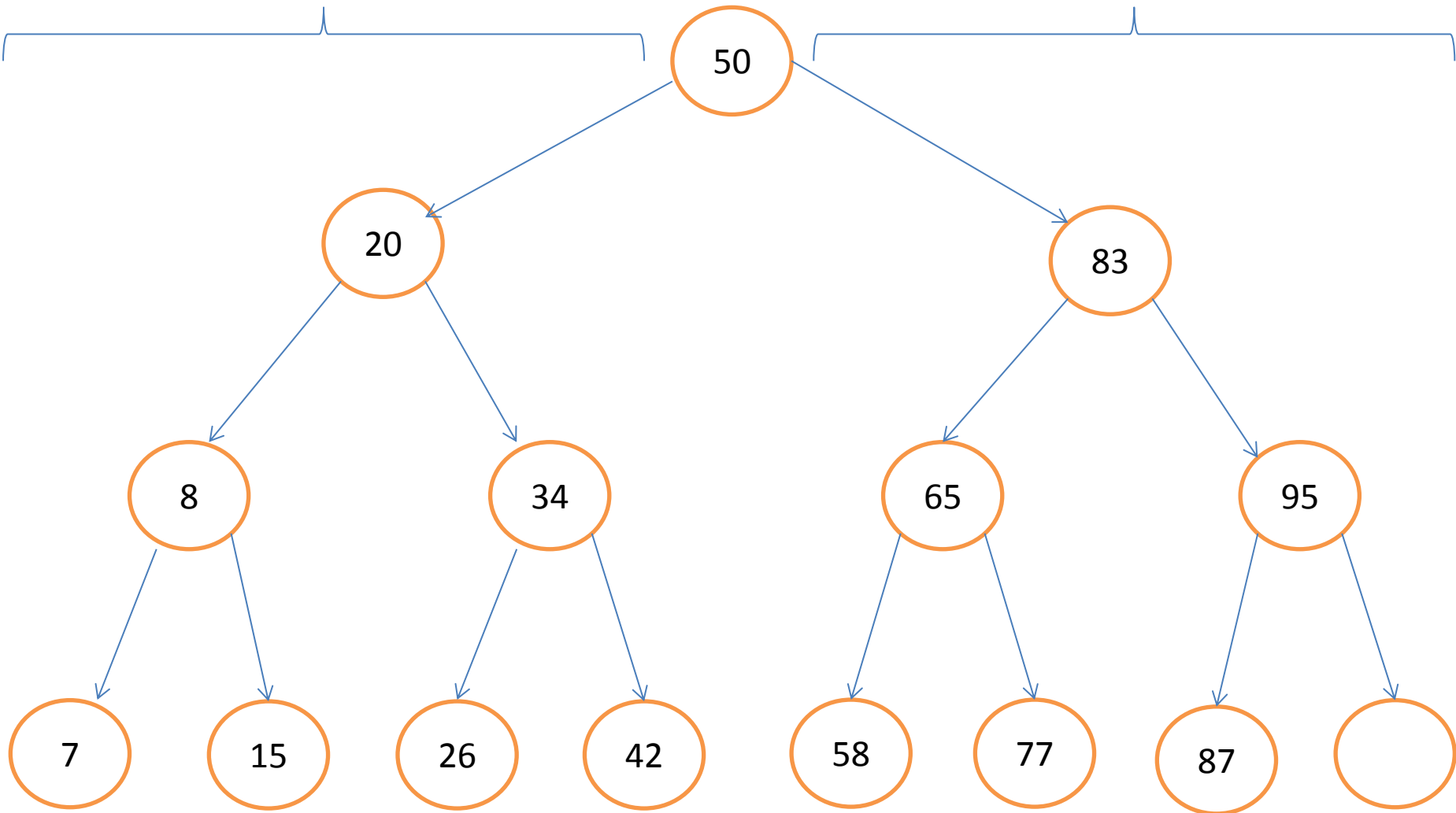
Dichotomy program

```
found=0;
left=0;
right=n-1;
while(left<right)
{
    mid=(left+right)/2;
    c=strcmp(argv[1], table[mid].surname);
    if(c==0){found=1;break;}
    else if(c<0)
        right=mid;
    else
        left=mid;
}
```

Binary trees

key < 50

key > 50

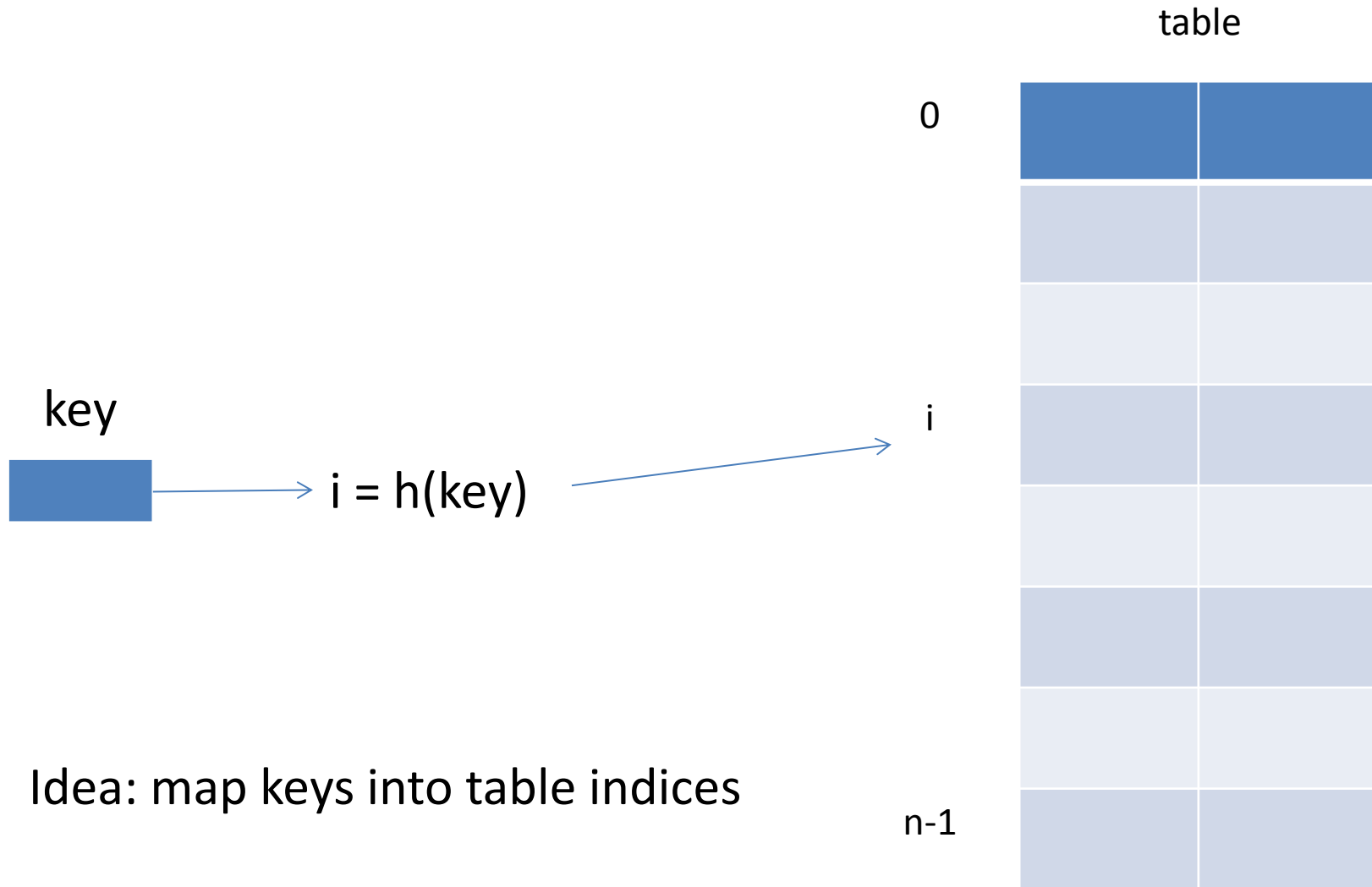


$O(\log n)$

Binary tree search

```
struct btree * bsearc( struct btree * bt, int key)
{
    if(bt==NULL) return(NULL);
    if(bt->key == key) return(bt);
    else if(bt->key < key) return(bsearch(bt->left));
        else return(bsearch(bt->right));
}
```

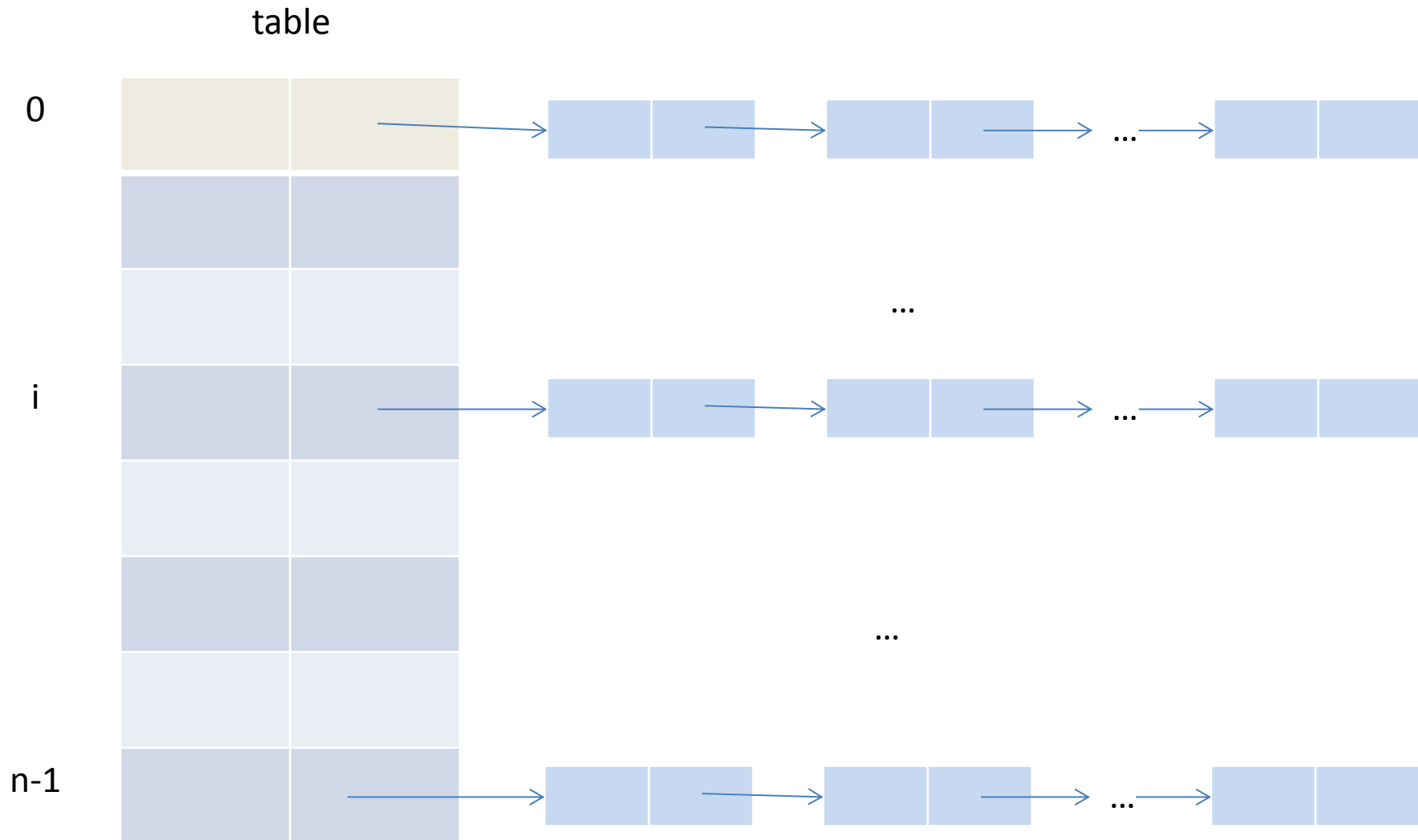
Hash tables



Idea: map keys into table indices

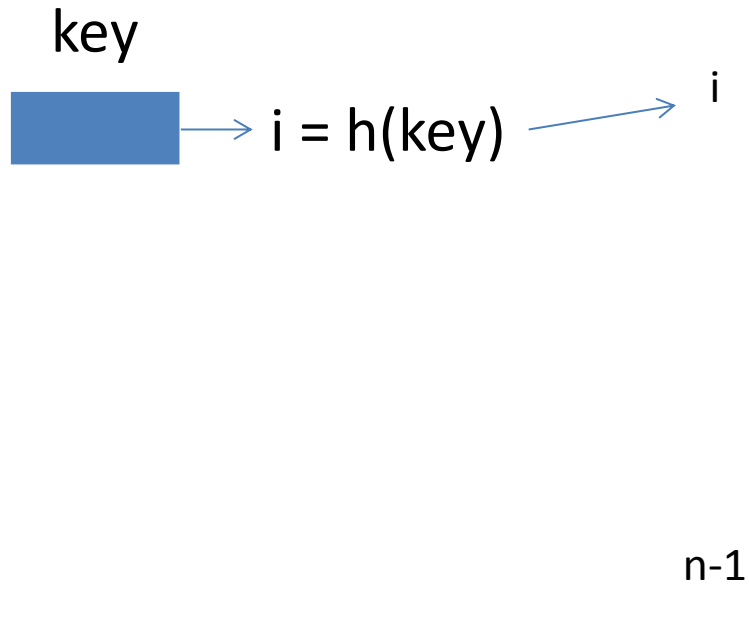
$O(1)$

Overflow chains (lists)



Open hashing

n and d are
mutually prime
numbers:
 $\text{gcd}(n,d)=1$



table

if(table[i].key != key)

$i = (i+d) \% n$

etc

Hash functions

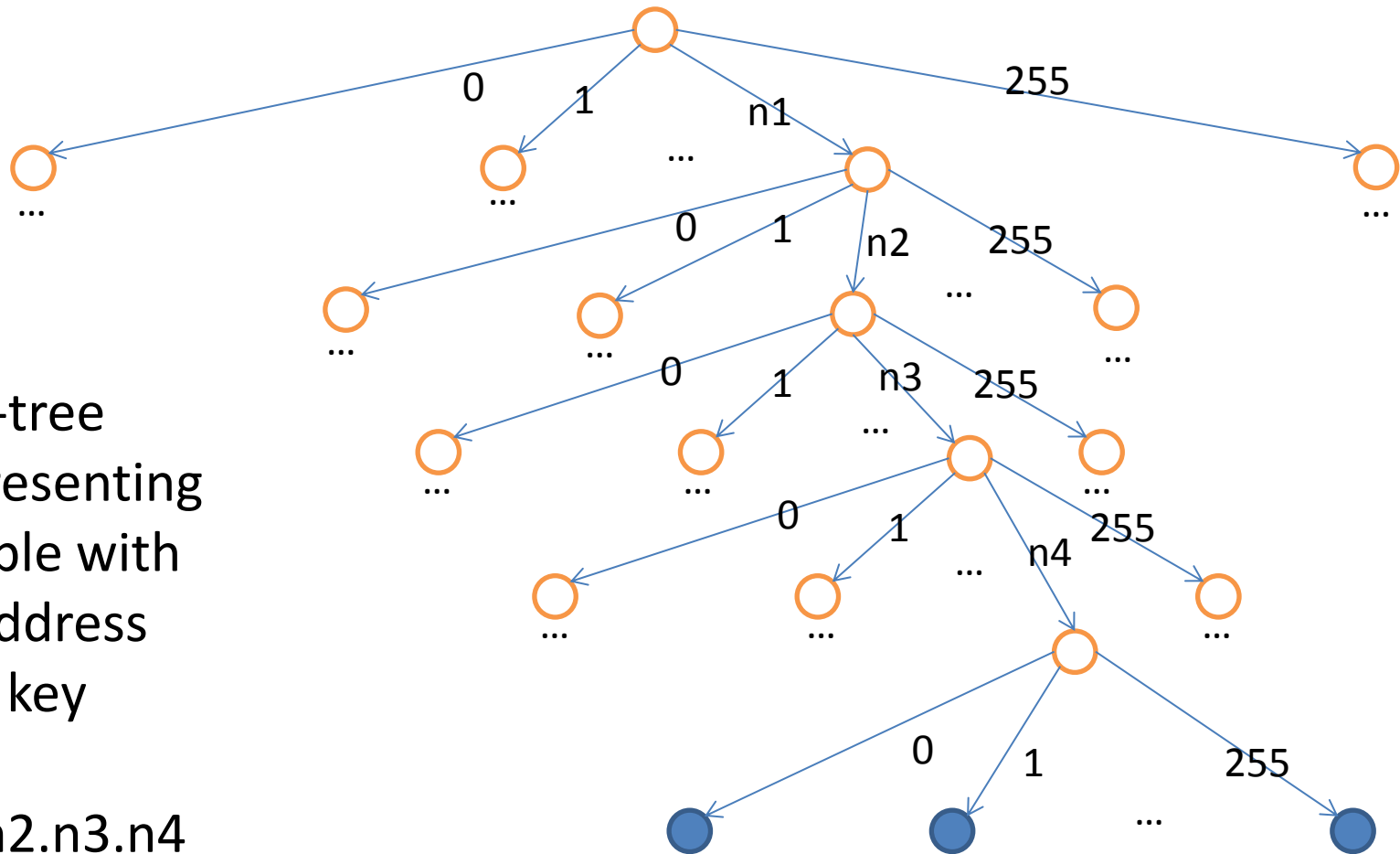
- Simple
- Efficiently computed
- Provide good scattering
- Example $h(s) = (\sum_{i=0}^{i < n} s_i) \bmod n$

Numerical trees

256-tree
representing
a table with
IP-address
as a key
as a key

$n1.n2.n3.n4$

$O(k)$



Records of the table