MINISTERUL EDUCAȚIEI ȘI CERCETĂRII
ROMÂNIA
UNIVERSITATEA DE MEDICINĂ,
FARMACIE, ȘTIINȚE ȘI TEHNOLOGIE
„GEORGE EMIL PALADE"
DIN TÂRGU MUREȘ

FACULTATEA DE INGINERIE ȘI TEHNOLOGIA INFORMAȚIEI

Algoritmi fundamentali

Curs 8

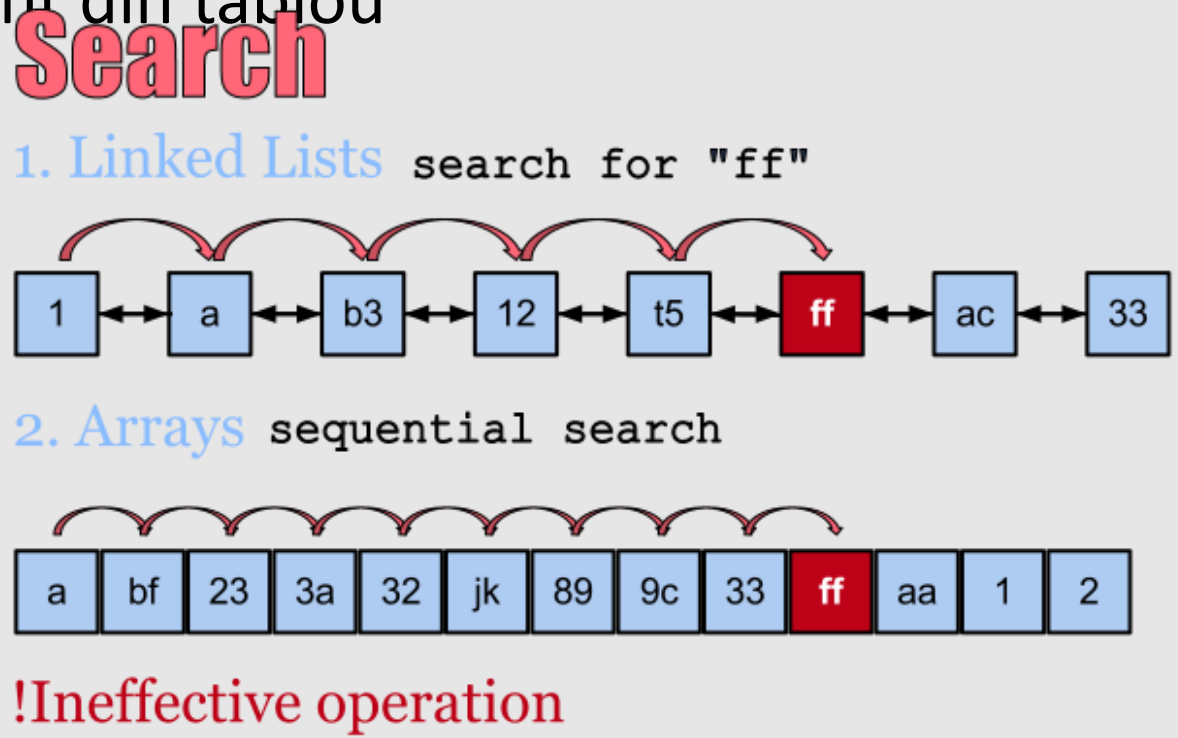
Algoritmi de cautare

Dr. ing. Kiss Istvan

istvan.kiss@umfst.ro

Cuprins

1. Cautare secventiala
2. Cautare binara
3. Cautare prin interpolare
4. Cautare prin Hashing
5. Stergere element din tablou
6. Probleme



1. Cautare secventiala

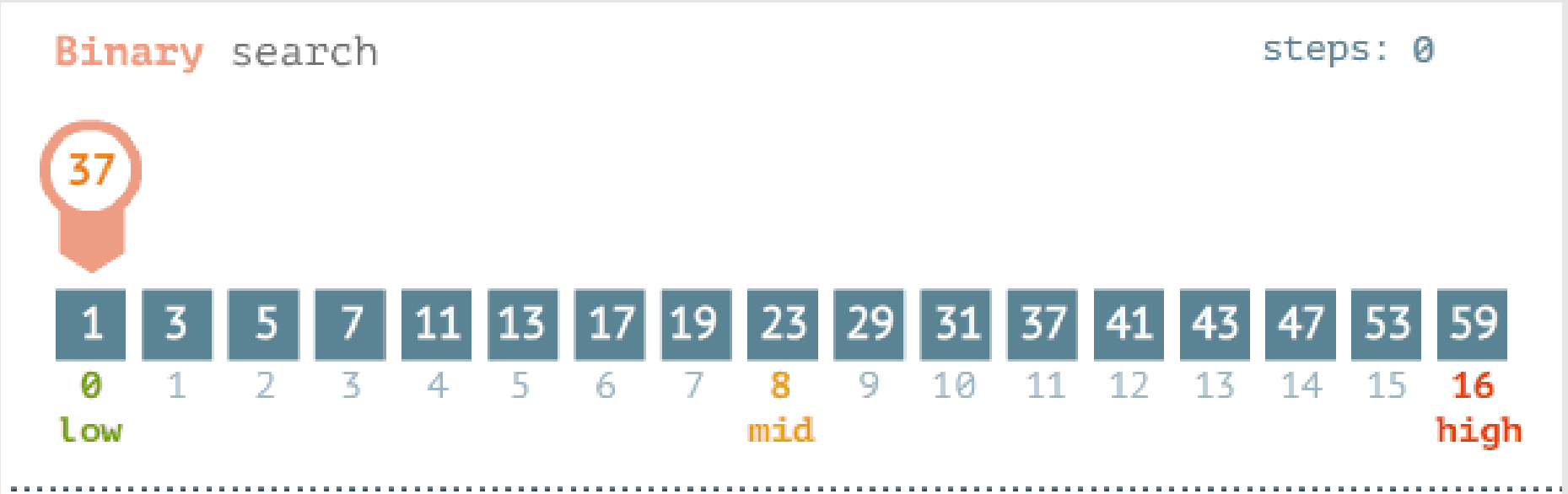
- Verifica aparitia unui element intr-un sir de elemente de acelasi tip.
- Se parcurge sirul si se verifica egalitatea fiecarui element cu elementul cautat.
- Subalgoritmul “cautare” poate furniza la iesire o variabila booleana, care indica succesul in cautare si/sau pozitia elementului gasit.

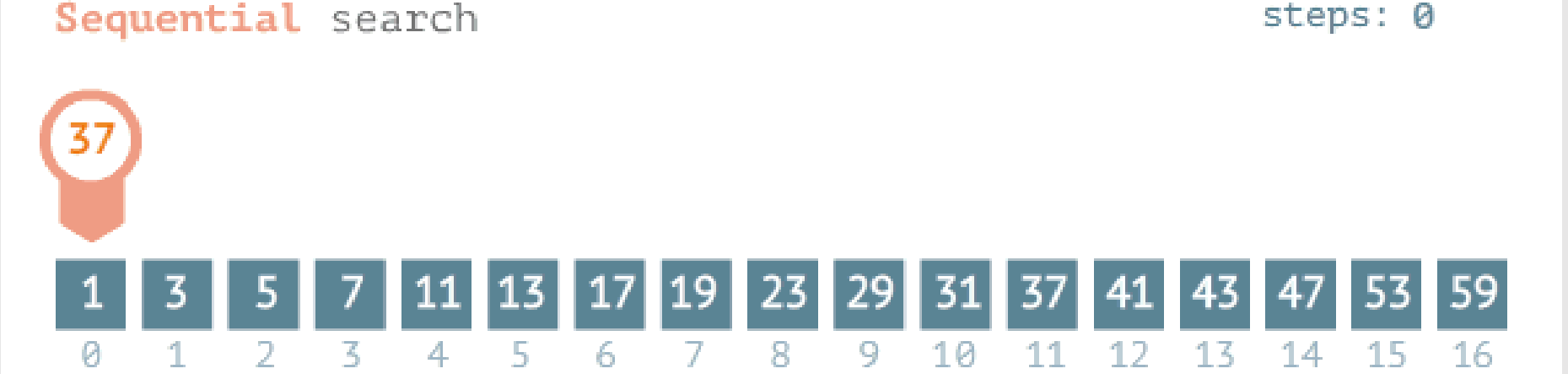
1. Cautare secventiala

- **Implementare:**

```
subalgoritm cautare_secv(vector, n, cheie_cautata):
    bool gasit:=false;
    pentru i:=0, n-1 exec.
        daca (vector[i]==cheie_cautata) atunci
          afiseaza i;
          gasit:=true;
          returneza i;
        sf.daca
    sf.pentru
    daca gasit==false atunci
        returneaza -1;
    sf.daca
sf.subalg.
```

1. Cautare secventiala





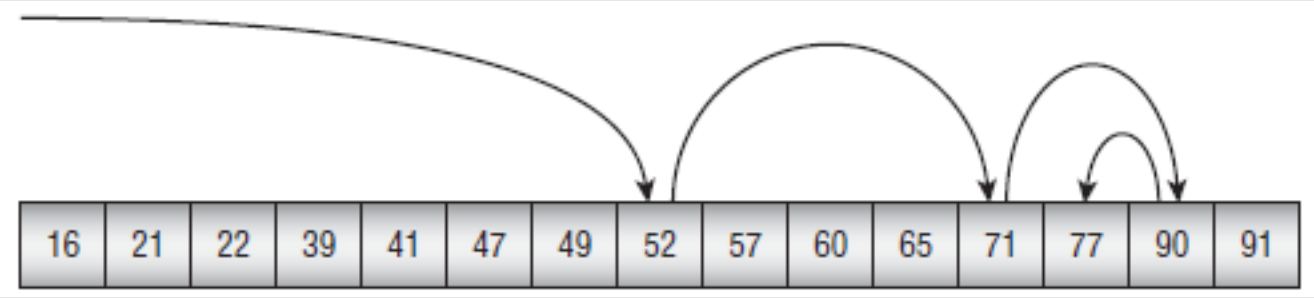
www.penjee.com

1. Cautare secventiala

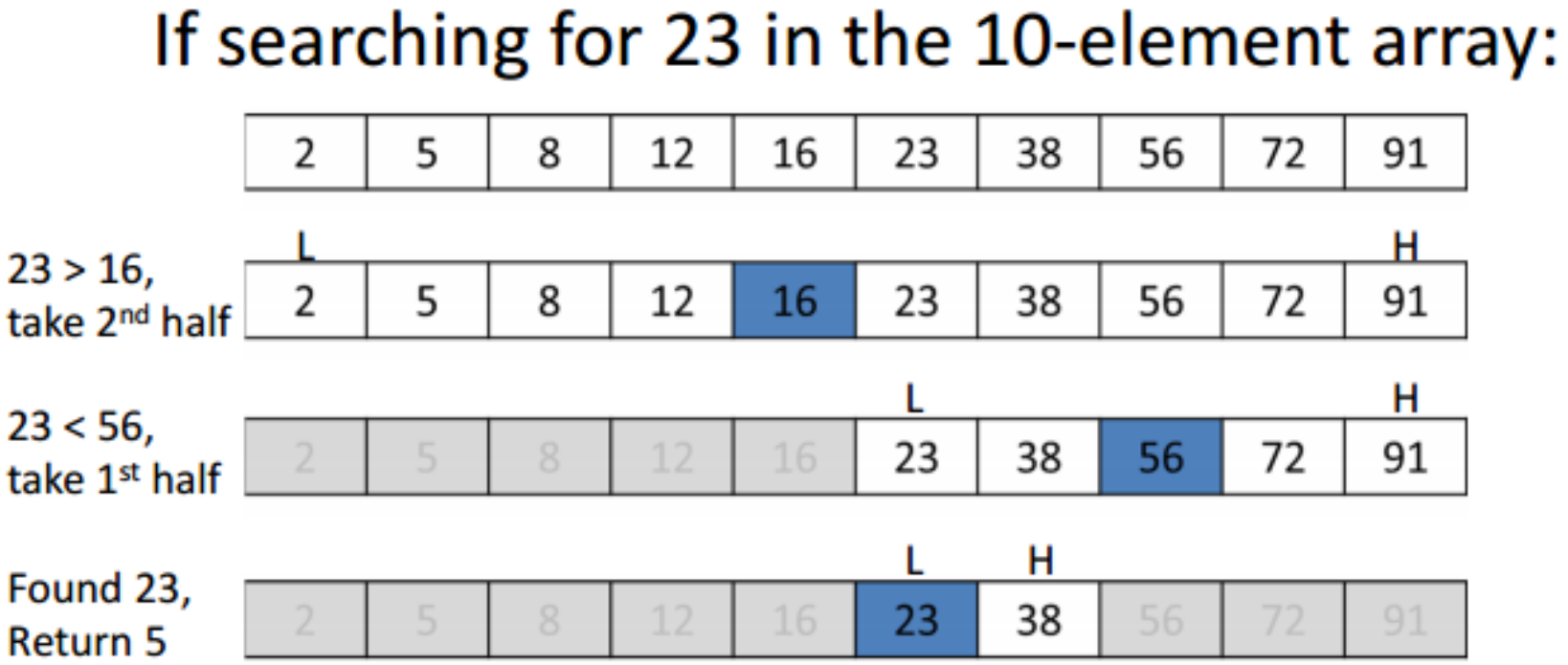
- **Complexitatea** algoritmului cautare secventiala:
  - se cauta primul element O(1)
  - elemental cautat nici nu exista O(n)
  - repetitii in mediu: (n+1)/2
  - caz general O(n)

2. Cautare binara

- Se bazeaza pe Divide et Impera si este o metoda mai eficienta
- Conditii preliminare: elemente in ordine
- Daca cheia cautata este mai mica decat elementul din mijloc atunci se continua cautarea in jumatatea inferioara, in cazul contrar in jumatatea superioara.
- Cautarea se repeta in mod recursiv.
- Daca cheia cautata este tocmai la mijloc, atunci se opreste cautarea.
- Are complexitate O(logn)



2. Cautare binara



2. Cautare binara

- **Implementare:**

```
subalgoritm cautareBinara(vector, inf, sup, cheie):
        intreg mijl;
        cattimp inf<=sup exec.
                mijl:=(inf+sup)/2;
                daca vector[mijl]==cheie atunci
                        returneaza mijl;
                altfel daca vector[mijl]<cheie atunci
                        inf:=mijl+1;
                altfel
                        sup:=mijl-1;
                sf.daca
        sf.cat
        returneaza -1;
sf.subalg.
```

2. Cautare binara

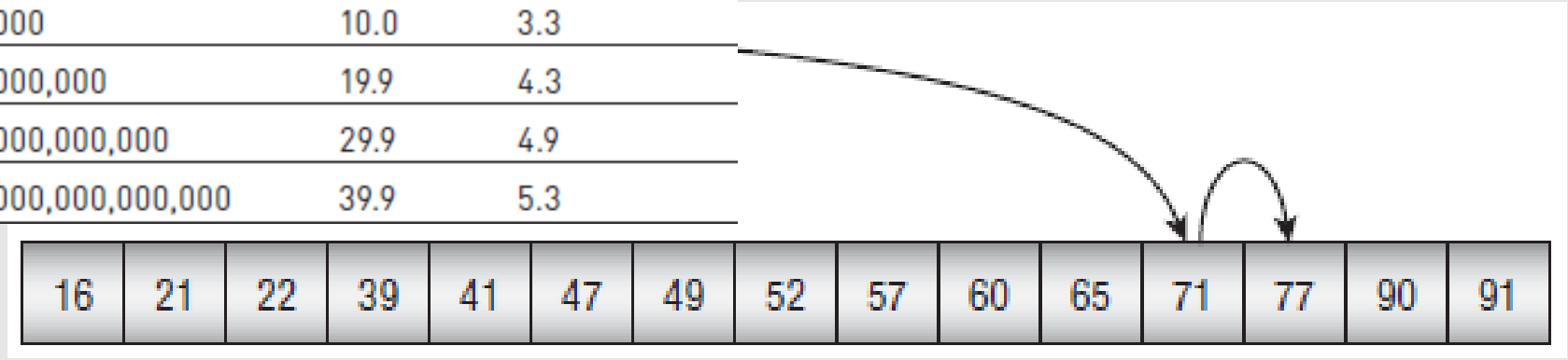
- **Implementare recursiva:**

procedure *binary search*$(i, j, x$: i, j, x integers, $1 \leq i \leq j \leq n)$
$m := \lfloor (i + j)/2 \rfloor$
if $x = a_m$ **then**
    return m
else if $(x < a_m$ and $i < m)$ **then**
    return *binary search*$(i, m - 1, x)$
else if $(x > a_m$ and $j > m)$ **then**
    return *binary search*$(m + 1, j, x)$
else return 0
{output is location of x in $a_1, a_2, \ldots, a_n$ if it appears; otherwise it is 0}

3. Algoritm de cautare prin interpolare

- Spre deosebire de cautare binara, cautarea prin interpolare (Interpolation Search) incearca sa aproximeze pozitia elementului cautat.
- Conditii preliminare:
  - elemente in ordine
  - conteaza distributia elementelor in tablou
  - Daca elementele sunt uniform distribuite atunci algoritmul evidentiaza ordinul de complexitate O(log(logn)), altfel O(n)

N	LOG_2 N	$LOG_2(LOG_2$ N)
1,000	10.0	3.3
1,000,000	19.9	4.3
1,000,000,000	29.9	4.9
1,000,000,000,000	39.9	5.3



3. Algoritm de cautare prin interpolare

- **Implementare**:

```
Functie InterpolationSearch(Data values[], Data target)
        Integer min := 0
        Integer max := values.Length - 1
        While (min <= max)
                // Find the dividing item.
                mid := min + (max - min) *
                        (target - values[min]) / (values[max] - values[min])
                If (values[mid] == target) Then Return mid
                <Set min or max to search the left or right half.>
        End While
        Return -1
End InterpolationSearch
```

3. Algoritm de cautare prin interpolare

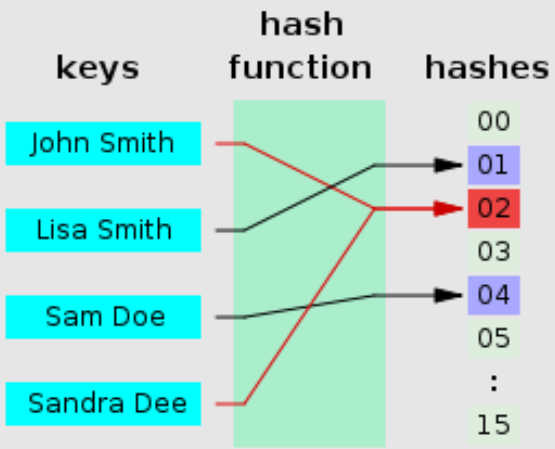
- **Exemplu**
- Daca values[min] este 100 si values[max] este 200, iar target este 125 atunci:
  - (125-100)/(200-100)=25/100=0.25

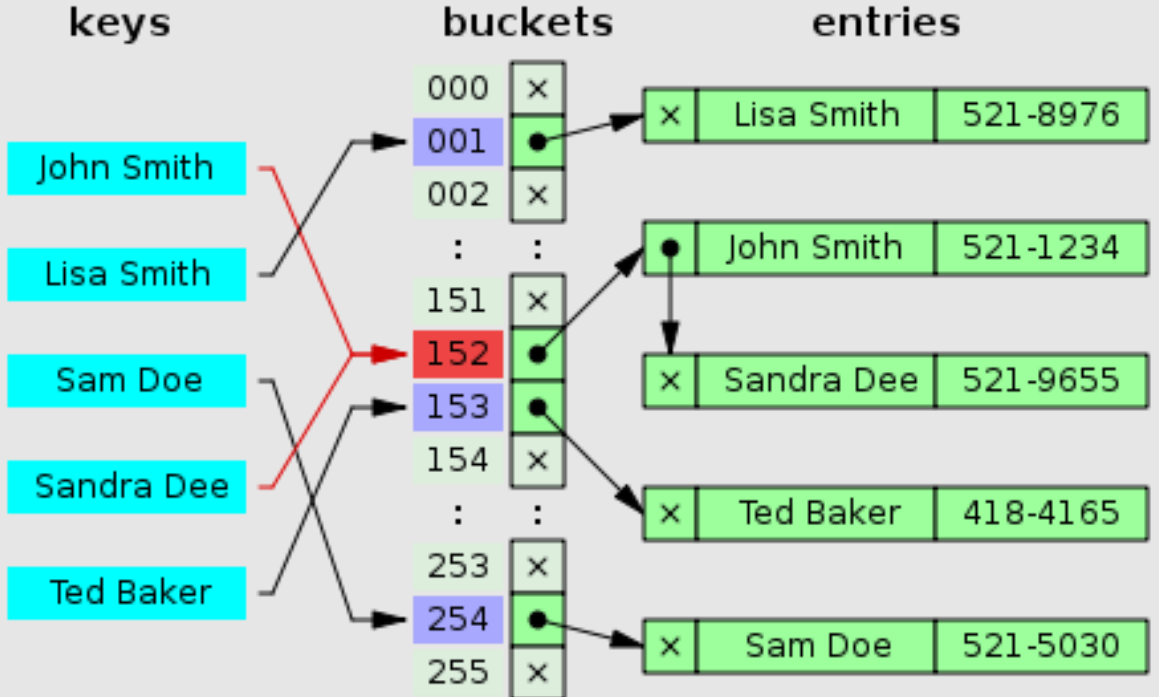
3. Algoritm de cautare prin interpolare

```
template <typename T>
int interpolation_search(T arr[], int size, T key)
{
    int low = 0;
    int high = size - 1;
    int mid;

    while ((arr[high] != arr[low]) && (key >= arr[low]) && (key <= arr[high])) {
        mid = low + ((key - arr[low]) * (high - low) / (arr[high] - arr[low]));

        if (arr[mid] < key)
            low = mid + 1;
        else if (key < arr[mid])
            high = mid - 1;
        else
            return mid;
    }

    if (key == arr[low])
        return low ;
    else
        return -1;
}
```

4. Cautare cu ajutorul tabelor Hash

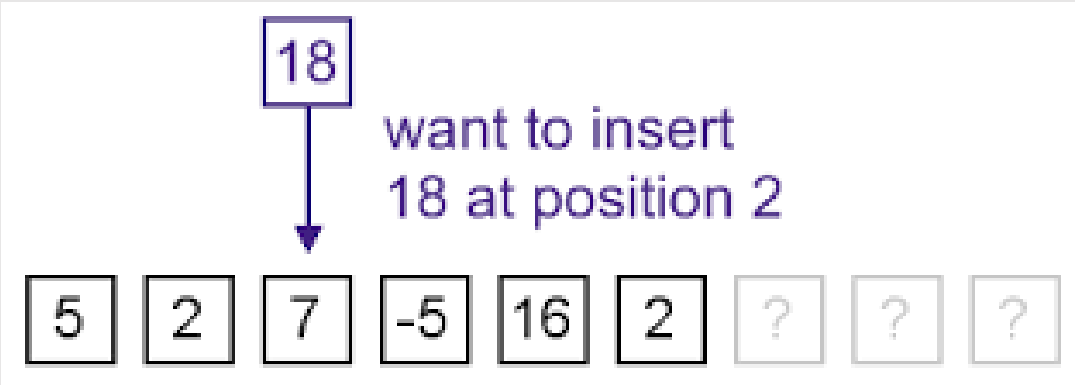
- Cu ajutorul unei functii Hash se creeaza un tabel asociat de dimensiune fixa.
- Inaintea cautarii se determina pozitia aproximativa pentru cheia cautata:
  - index=f(cheia, dim_sir) **sau**
  - hash = hashfunc(cheia); index = hash % dim_sir;

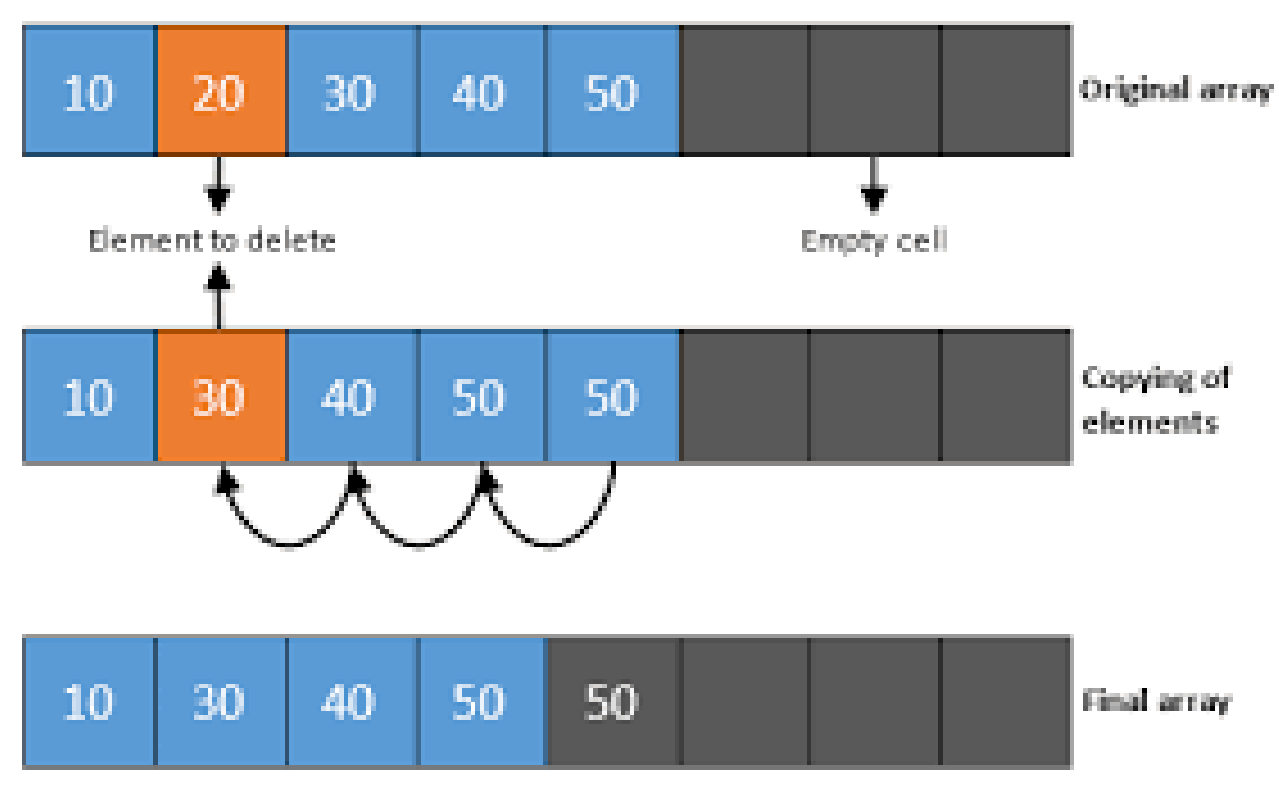




5. Stergere element din tablou

- Operatii:
  - stergere
  - adaugare (inserare)





6. Probleme algoritmi cautare

- **Vezi documentul lab_problem.pdf**