Универзитет “Св. Кирил и Методиј” во Скопје
Факултет за електротехника и информациски технологии

ПРОГРАМИРАЊЕ И АЛГОРИТМИ

Сортирање и пребарување

- Програмски јазик C –

ЗОШТО СОРТИРАЊЕ?

- Сортирањето го користат многу програми:
  - Facebook ги сортира пријателите по азбучен редослед
  - Facebook ви ги дава најновите статуси
  - Excel ги сортира колоните според определена вредност
  - Youtube ги сортира видеата според популарност
- Дадено: листа од предмети
  - броеви, букви, имиња,...
- Предметите може да се сортираат:
  - во азбучен редослед
  - од најголем до најмал
  - од најмал до најголем
  - од најтемни до најсветли,...

АЛГОРИТМИ ЗА СОРТИРАЊЕ

- Множество од чекори кои компјутерот ги извршува со цел да ги подреди елементите во правилен (сортиран) редослед

- Постојат многу алгоритми за сортирање
  - Но... сите имаат различна ефикасност и брзина
  - Тука нема да се анализираат овие параметри, но истите имаат голема важност при изборот на алгоритмот во пракса

- Ќе се разгледуваат следниве алгоритми за сортирање:
  - Selection Sort
  - Bubble Sort
  - Insertion Sort
  - Simple Sort (на аудиториски вежби)

СОРТИРАЊЕ: SELECTION SORT

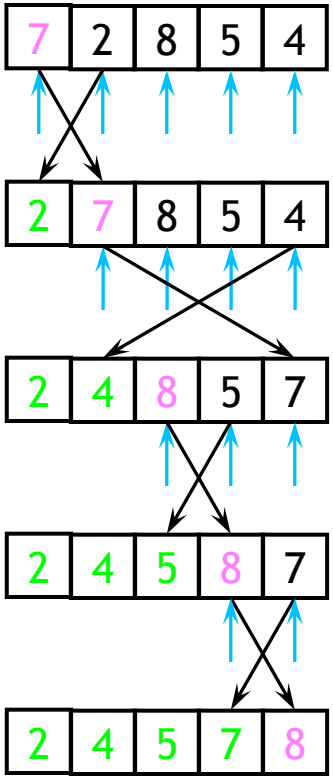
- Дадена е низа од n елементи:
  - Се пребаруваат елементите од 0 до n-1 и се избира најмалиот
    - Се заменува со елементот на позиција 0
  - Се пребаруваат елементите од 1 до n-1 и се избира најмалиот
    - Се заменува со елементот на позиција 1
  - Се пребаруваат елементите од 2 до n-1 и се избира најмалиот
    - Се заменува со елементот на позиција 2

    .

    .

    .
  - Се продолжува се дури не остане ниту еден елемент за проверка

СОРТИРАЊЕ: SELECTION SORT



СОРТИРАЊЕ: SELECTION SORT

```
void selectionSort(int a[], int n){
    int i,j,pom,min;
    for(i=0;i<n-1;i++){
        min=i;
        for(j=i+1;j<n;j++)
            if(a[min]>a[j])
                min=j; //го ажурираме само
                       //индексот

        //потоа ги заменуваме i-тиот елемент
        //со елементот со индекс min
        pom=a[i];
        a[i]=a[min];
        a[min]=pom;
    }
}
```

ПРИМЕР 1 (КОМБИНИРАНО СОРТИРАЊЕ)

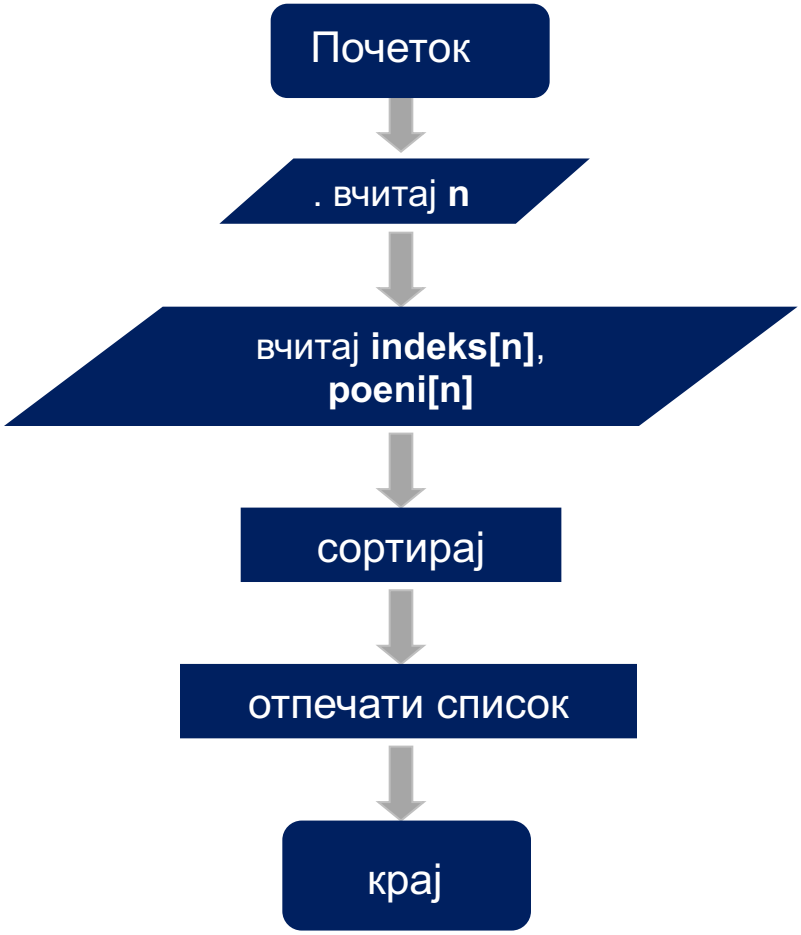
- **Проблем:** Даден е список на студенти од една генерација со нивните броеви на индекс и поени од испит. Да се напише програма која ќе ги подреди студентите според бројот на освоени поени по опаѓачки редослед.

- **Анализа на проблемот:**
  - Студентите се од една генерација, па доволно е да се чува само првиот дел од бројот на индексот (цел број)
  - Поените се цели броеви во опсег 0 до 100
  - Внесувањето на студентите и поените е по случаен редослед
  - Потребно е да се подреди низата со поени според опаѓачки редослед и според тоа подредување да се подредат и броевите на индексите.

ПРИМЕР 1 (КОМБИНИРАНО СОРТИРАЊЕ)

- Што треба да се направи?

1. Вчитај ја должината на низите
2. Вчитај ги елементите на двете низи
3. Сортирај ја низата со поени,
   и на ист начин низата со индекси
4. Отпечати ги подредените вредности

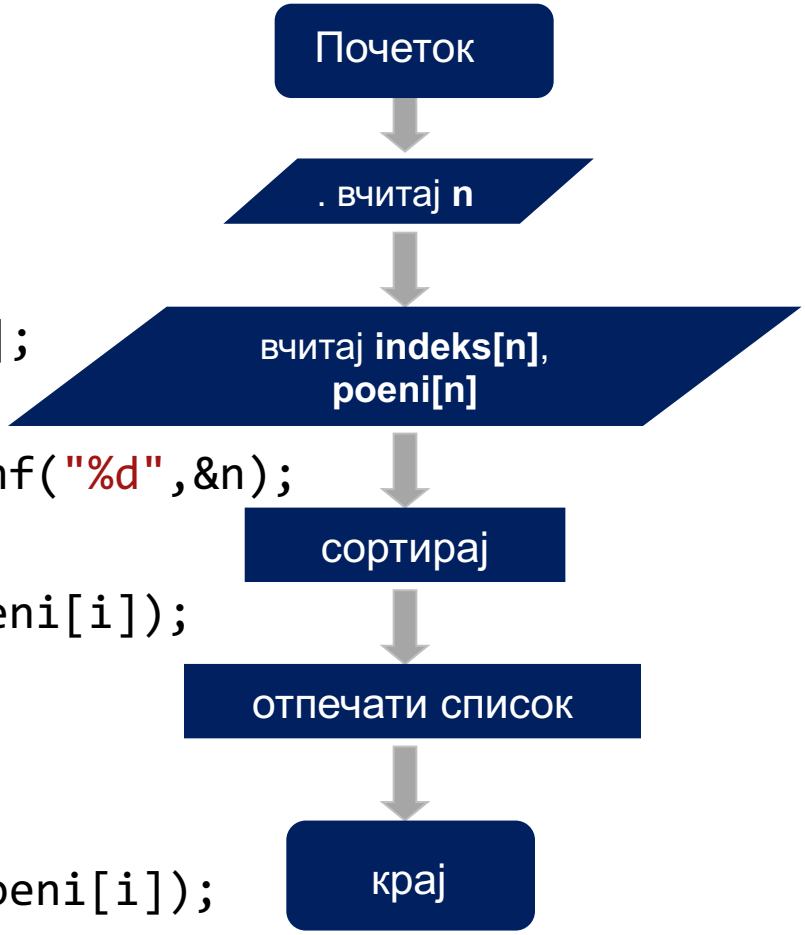


ПРИМЕР 1 (КОМБИНИРАНО СОРТИРАЊЕ)

- **Алгоритам:**
  - Најди го студентот со најмногу поени и замени го во низата со студентот кој е внесен прв (замени ги и поените и бројот на индекс). Со тоа на **почеток** на низата ќе биде студентот со најмногу поени
  - Во низата од вториот студент до крајот најди го студентот со најмногу поени и замени го во низата со студентот кој е внесен втор. Со тоа на **второто место** ќе биде студентот со најмногу поени
  - Повтори ја постапката заклучно со претпоследниот внесен студент.

ПРИМЕР 1 (КОМБИНИРАНО СОРТИРАЊЕ)

```
#include "stdio.h"
#define MaxElem 400
void sortiraj(int *, int *, int);
int main() {
    int indeks[MaxElem],poeni[MaxElem];
    int i,n;
    printf("Broj na studenti: "); scanf("%d",&n);
    for (i=0; i<n; i++)
        scanf("%d%d",&indeks[i], &poeni[i]);
    sortiraj(indeks,poeni,n);
    printf("Spisok:\n");
    for (i=0; i<n; i++)
        printf("%d %d\n",indeks[i],poeni[i]);
    return 0;
}
```



ФУНКЦИИ SORTIRAJ, SWAP И MAXIMUM

```
int maximum(int [], int);
void swap(int *,int *);

void sortiraj(int a[],int b[],int n)
{
    int i,max_pos;

    for (i=0; i<n-1; i++){
        max_pos=maximum(&b[i], n-i);
        swap(&a[i],&a[i+max_pos]);
        swap(&b[i],&b[i+max_pos]);
    }
}
```

```
int maximum(int a[],int n){
    int i, ind=0;

    for (i=1; i<n; i++)
        if (a[i]>a[ind])
            ind=i;

    return ind;
}

void swap(int *a, int *b){
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
```

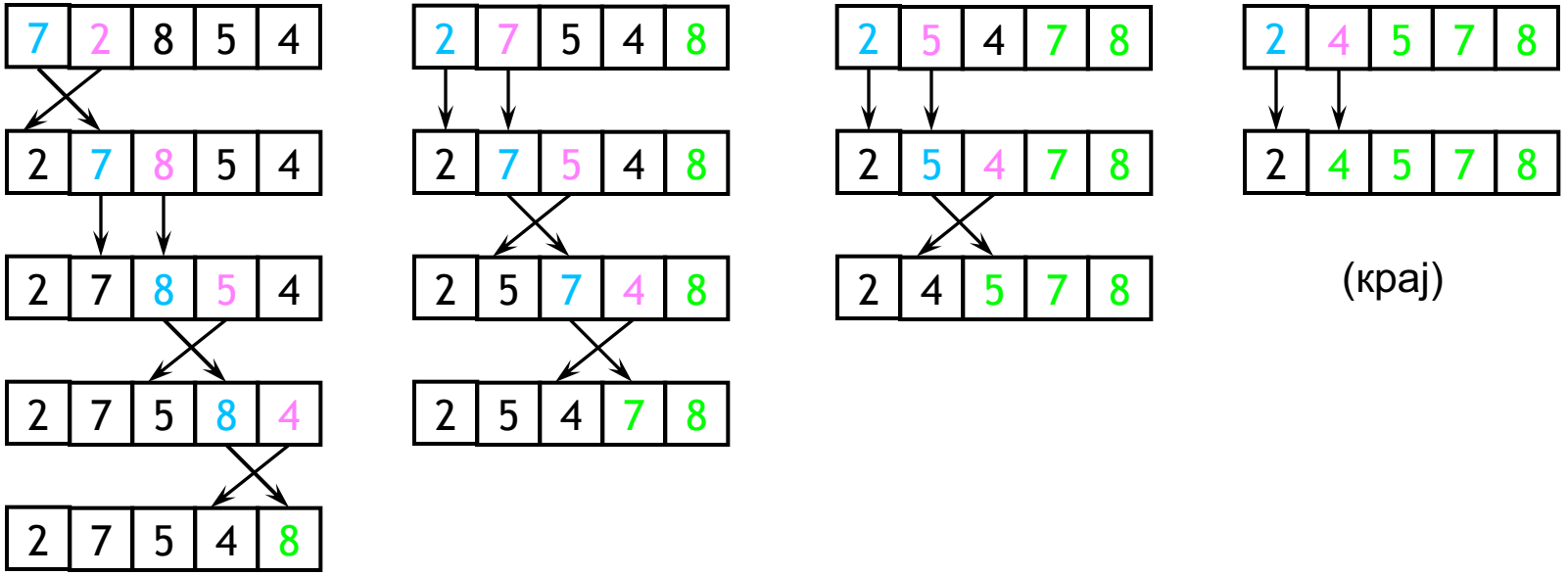
СОРТИРАЊЕ: BUBBLE SORT

- Ако се сортира низа во **растечки** редослед:
- **Итерација 1**: се споредува секој елемент (освен последниот) со неговиот десен сосед
  - ако не се правилно подредени, се заменуваат
  - после итерација 1, на **крај** од низата е **најголемиот елемент**
  - последниот елемент сега е на правилна позиција
- **Итерација 2**: се споредува секој елемент (освен последните два) со неговиот десен сосед
  - ако не се правилно подредени, се заменуваат
  - по итерација 2, **вториот најголемиот** елемент е претпоследен
  - последните два елементи сега се на правилна позиција

...

- Се продолжува со итерирање се додека постојат несортирани елементи од лева страна

СОРТИРАЊЕ: BUBBLE SORT

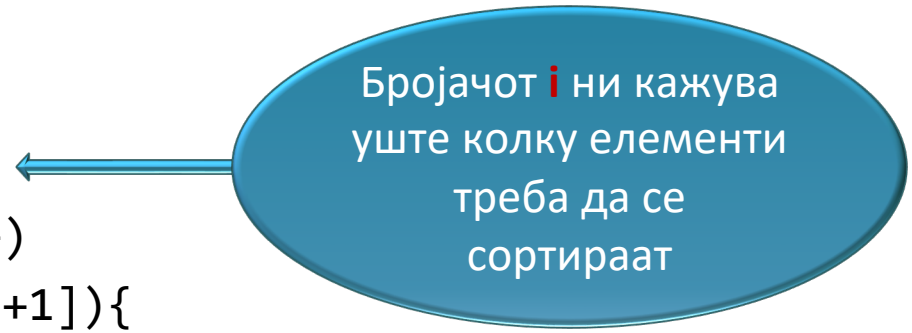


https://www.youtube.com/watch?v=nmhjrI-aW5o

СОРТИРАЊЕ: BUBBLE SORT

```
void bubbleSort(int a[], int n) {
    int i,j,pom;

    for(i=n-1;i>0;i--)
       for(j=0;j<i;j++)
           if(a[j]>a[j+1]){
              pom=a[j];
              a[j]=a[j+1];
              a[j+1]=pom;
           }
}
```



СОРТИРАЊЕ: INSERTION SORT

- Поедноставено објаснување:
  - Избери прва карта – таа е сортирана
  - Избери втора карта – спореди ја со првата карта и стави ја на соодветното место
  - Избери трета карта – спореди ја со првите две карти и вметни ја на соодветното место
  - ...
  - Избери n-та карта – спореди ја со n-1 карти и вметни ја на соодветното место
- Се сортира **левата** страна од низата (подниза) се додека не биде сортирана цела низа.
- Се вметнуваат нови (несортирани) елементи од десната подниза на точно место во левата (сортирана) подниза

СОРТИРАЊЕ: INSERTION SORT

- Ако се сортира низа во **растечки** редослед:

(1) се избира еден елемент

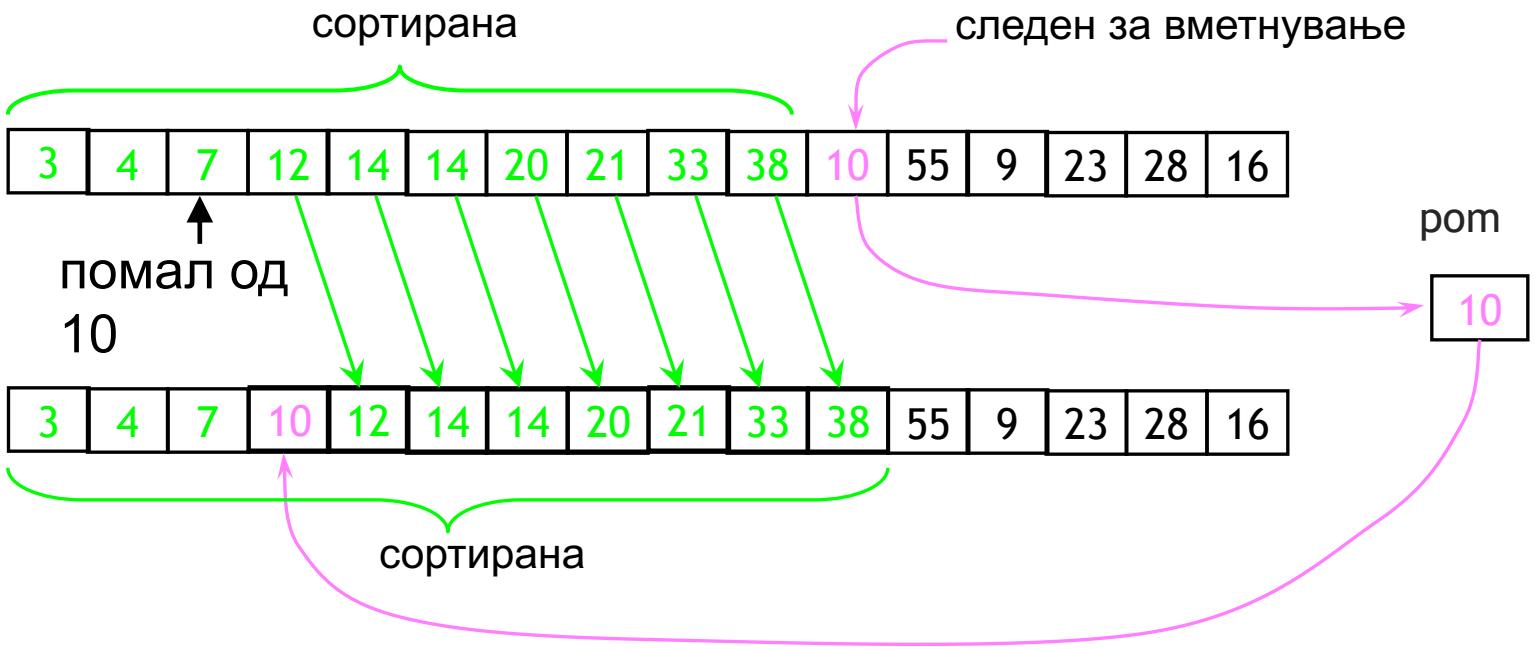
(2) сите елементи **лево** од него се споредуваат со **избраниот** елемент

(3) се додека елементите лево од избраниот се поголеми од него, се **поместуваат** за едно место во десно

(4) кога се наоѓа елемент кој не е повеќе поголем од избраниот, на празното место се вметнува избраниот елемент

(5) се продолжува на ист начин со елемент кој е десно од оној кој првично е избран , се додека не остане само последниот елемент од низата

СОРТИРАЊЕ: INSERTION SORT (ЕДНА ИТЕРАЦИЈА)



https://www.youtube.com/watch?v=OGzPmgsI-pQ

СОРТИРАЊЕ: INSERTION SORT

```
void insertionSort( int a[], int n)
{
    int i,j,pom;
    for(i=1;i<n;i++){
        pom=a[i];
        for(j=i; j>0 && a[j-1]>pom; j--)
            a[j]=a[j-1];
        a[j]=pom;
    }
}
```

Надворешна јамка:
кој елемент од несортираната низа треба да се вметне во сортираната

Внатрешна јамка:
ги изминува сортираните елементи (почнувајќи од последниот), сите елементи поголеми од избраниот ги поместува за едно место во лево

Избраниот елемент од надворешната јамка го сместува на точно место

ПРЕБАРУВАЊЕ
ЛИНЕАРНО ПРЕБАРУВАЊЕ

- Се пребарува поле за да се пронајде зададена ***клуч вредност***
- Циклус кој што ги изминува сите елементи почнувајќи од првиот, се додека не се најде бараниот елемент
- Линеарно пребарување
  - Едноставно
  - Спореди го **секој елемент од полето** со зададената клуч вредност (вредноста што се бара)
  - Корисно за мали и несортирани полиња

БИНАРНО ПРЕБАРУВАЊЕ

- Ја определува позицијата (индексот) на дадена вредност (клуч) во подредена низа од броеви (примерот е за опаѓачки редослед):
  - **Во секој чекор:** се споредува **клучот** со **средниот елемент од низата**
  - Ако **се исти**, тогаш елементот е пронајден. Индексот на средниот елемент е бараниот индекс.
  - Доколку **не се исти**, тогаш
    - ако средниот елемент е **помал** од бараната вредност, барањето се продолжува во **горниот (десниот)** полуинтервал (половина од полето)
    - ако средниот елемент е **поголем** од бараната вредност, барањето се продолжува во **долниот (левиот)** полуинтервал (половина од полето)
  - Се повторува се додека:
    - не се пронајде вредноста; или
    - полуинтервалот не е празен интервал. Тогаш бараната вредност **не се содржи во низата**.

БИНАРНО ПРЕБАРУВАЊЕ

- Кај било кое пребарување комплексноста (бројот на чекори е многу значаен параметар)
- Бинарното пребарување е многу брз алгоритам. Најмногу **n чекори** ако **2^n > бројот на елементи** во полето
  - За поле од 30 елементи потребни се најмногу 5 чекори
    - $2^5 > 30$ => најмногу 5 чекори

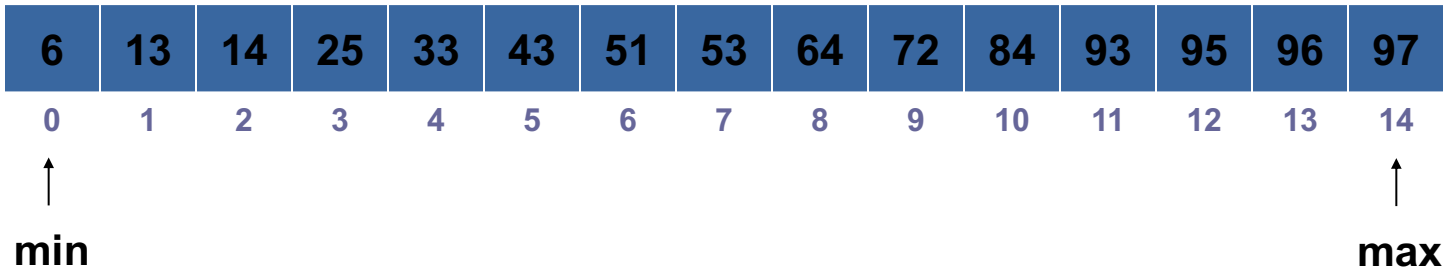
БИНАРНО ПРЕБАРУВАЊЕ

```
int binary_search(int a[], int key, int min, int max) {
    int srd;

    while (max >= min) {/*prebaruvaj se dodeka [min,max] ne e
                          prazen*/
       srd = (min+max)/2; /* presmetaj sredina */
       if (a[srd]<key) /*sredniot element pomal od baraniot*/
          min = srd+1; /* noviot min e nad sredniot */
       else if (a[srd]>key )
          max = srd - 1; /* noviot max e pod sredniot */
       else /* key e pronajden vo elem. so indeks srd */
          return srd;
    }
    return -1; /* key ne e pronajden */
}
```

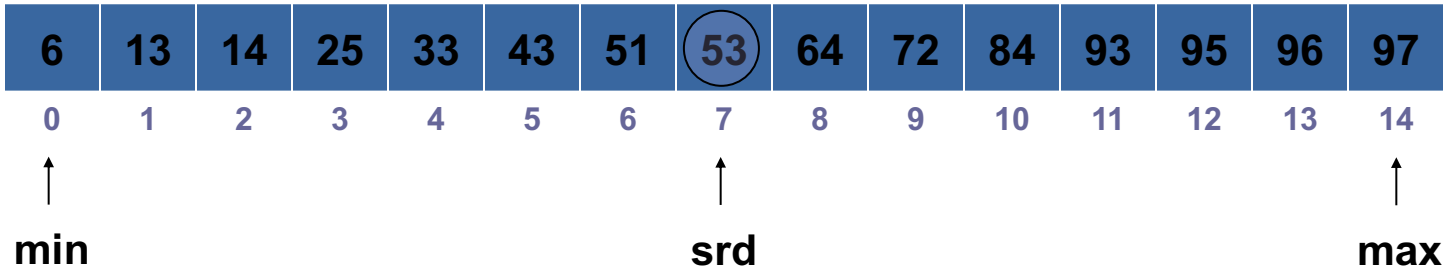
БИНАРНО ПРЕБАРУВАЊЕ

- Пр. Бинарно пребарување за елементот 33



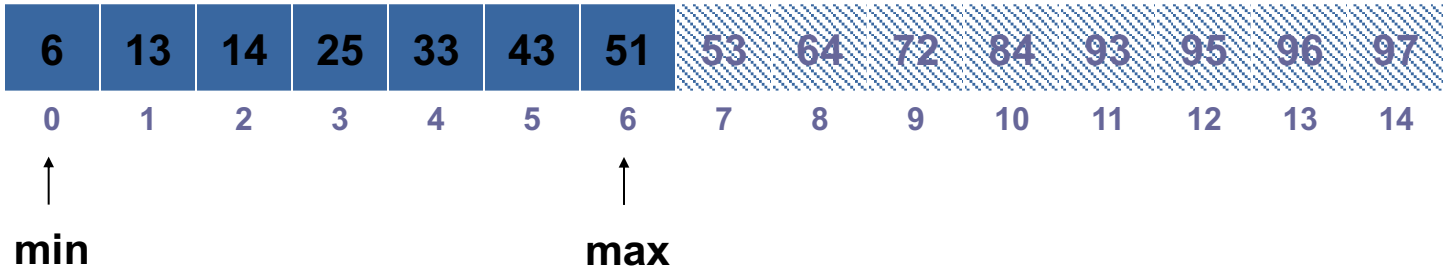
БИНАРНО ПРЕБАРУВАЊЕ

- Пр. Бинарно пребарување за елементот 33



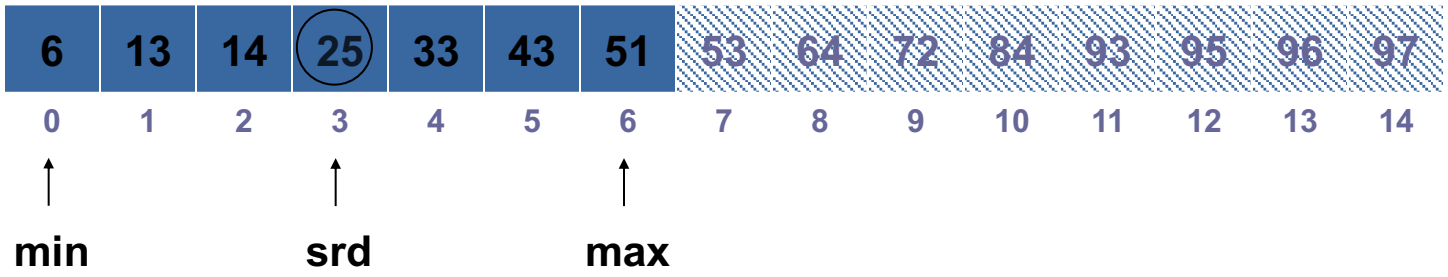
БИНАРНО ПРЕБАРУВАЊЕ

- Пр. Бинарно пребарување за елементот 33



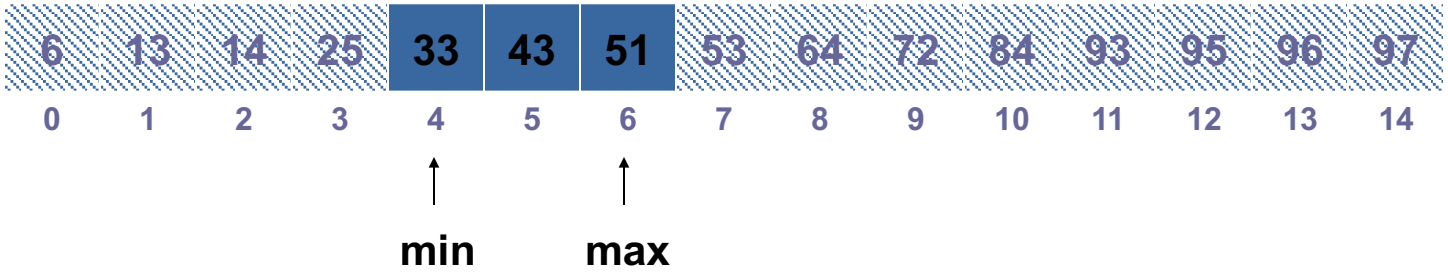
БИНАРНО ПРЕБАРУВАЊЕ

- Пр. Бинарно пребарување за елементот 33



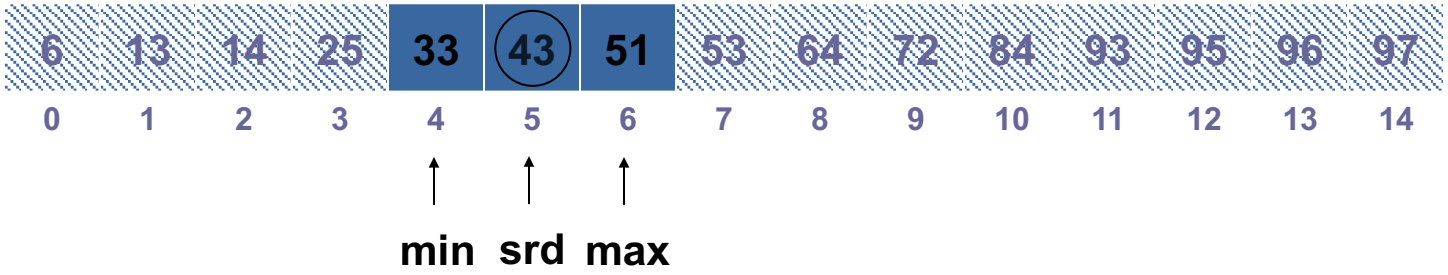
БИНАРНО ПРЕБАРУВАЊЕ

- Пр. Бинарно пребарување за елементот 33



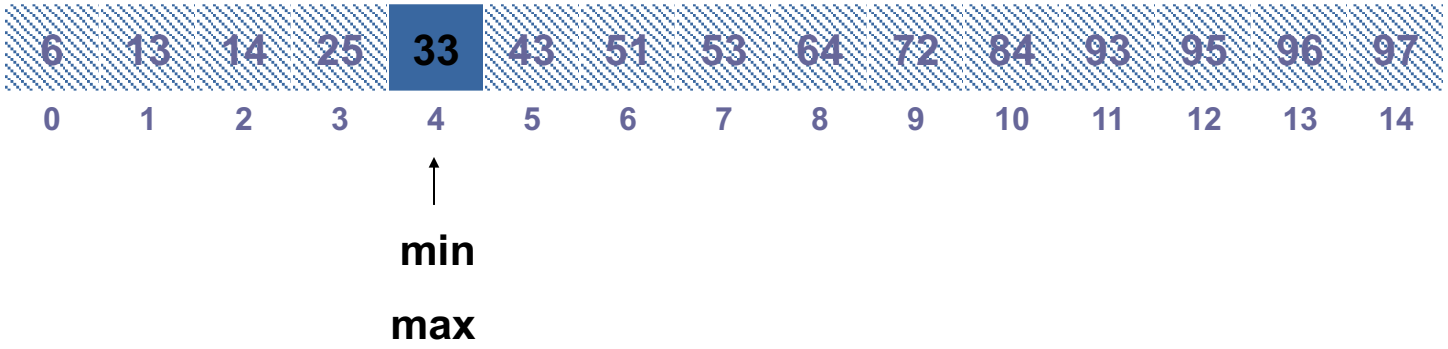
БИНАРНО ПРЕБАРУВАЊЕ

- Пр. Бинарно пребарување за елементот 33



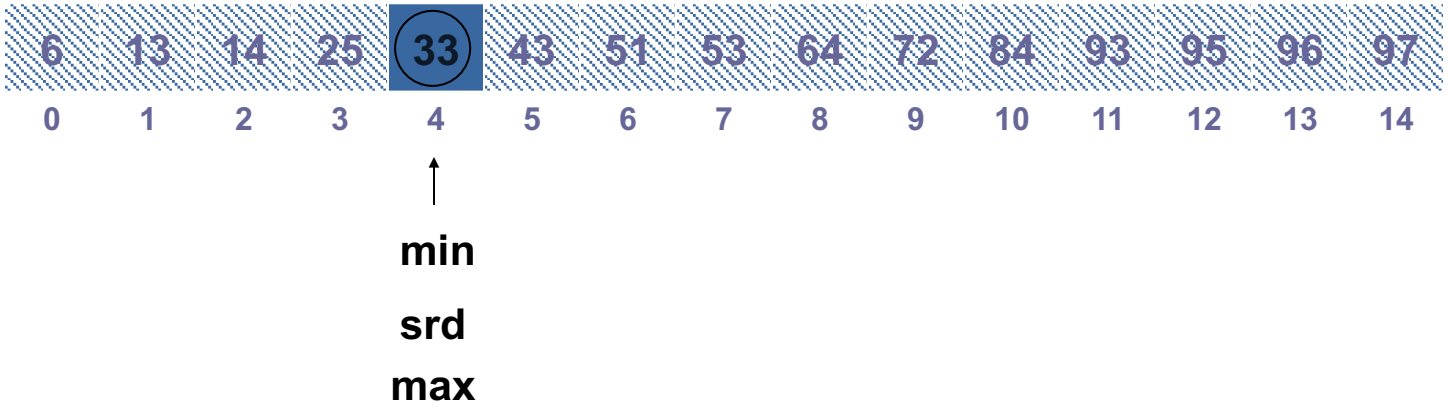
БИНАРНО ПРЕБАРУВАЊЕ

- Пр. Бинарно пребарување за елементот 33



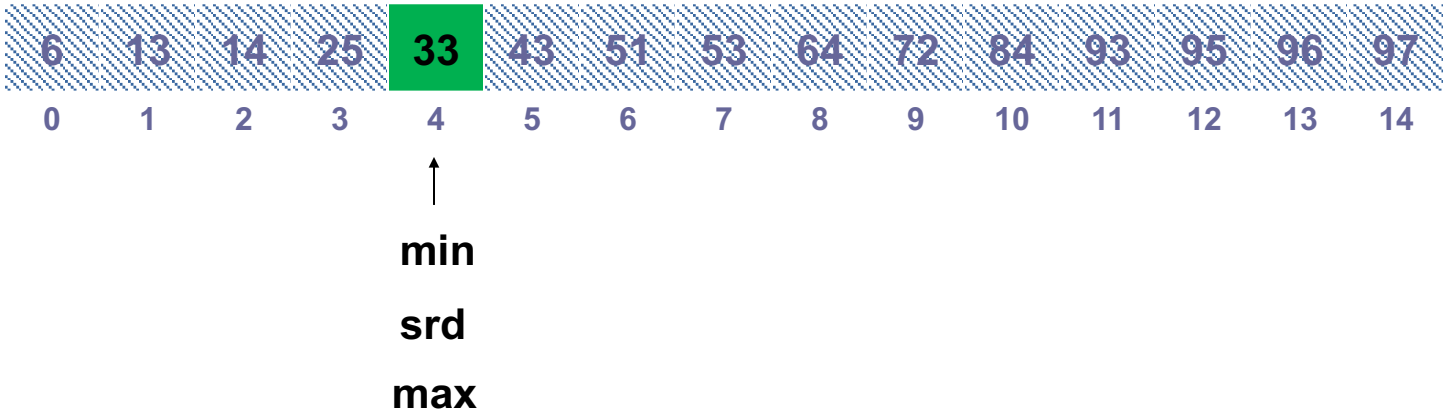
БИНАРНО ПРЕБАРУВАЊЕ

- Пр. Бинарно пребарување за елементот 33



БИНАРНО ПРЕБАРУВАЊЕ

- Пр. Бинарно пребарување за елементот 33



ПРИМЕРИ ЗА БИНАРНО ПРЕБАРУВАЊЕ

```
Enter a number between 0 and 28: 25
Subscripts:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
---------------------------------------------------------------
  0   2   4   6   8  10  12  14* 16  18  20  22  24  26  28
                                 16  18  20  22* 24  26  28
                                                 24  26* 28
                                                 24*

25 not found

Enter a number between 0 and 28: 8
Subscripts:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
---------------------------------------------------------------
  0   2   4   6   8  10  12  14* 16  18  20  22  24  26  28
  0   2   4   6*  8  10  12
                  8  10* 12
                  8*
8 found in array element 4
```

ПРИМЕРИ ЗА БИНАРНО ПРЕБАРУВАЊЕ

```
Enter a number between 0 and 28: 6
Subscripts:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
---------------------------------------------------------------
  0   2   4   6   8  10  12  14* 16  18  20  22  24  26  28
  0   2   4   6*  8  10  12

6 found in array element 3
```

КРАЈ