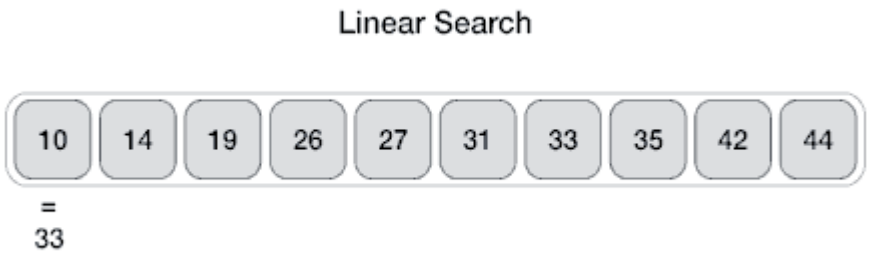
Algoritmi pretrage

Linearna pretraga

- Naziva se još i sekvencijalna pretraga
- Počinje od početka niza
- Sekvencijalno se proveravaju element niza jedan po jedan
- Kraj pretrage
  - Pronađen željeni element - uspešna pretraga, vraćemo indeks prvog pojavljivanja željenog elementa
  - Nije pronađen željeni element - neuspešna pretraga, vraćamo rezultat -1



Implementacija algoritma za linearnu pretragu

```
static int LinSearch(int[] x, int a)
{
    int n = x.Length;

    for (int i = 0; i < n; i++)
    {
        if (x[i] == a)
        {
            return i;
        }
    }
    return -1;
}
```

Radnja	Broj jedinica vremena
Dužina niza	1
Inicijalizacija i = 0	1
Provera uslova i < n	n + 1
Poređenje x[i] == a	n
Inkrementiranje i++	n - 1
Vraćanje vrednosti (return)	1
Ukupno (najgori slučaj)	3n + 3

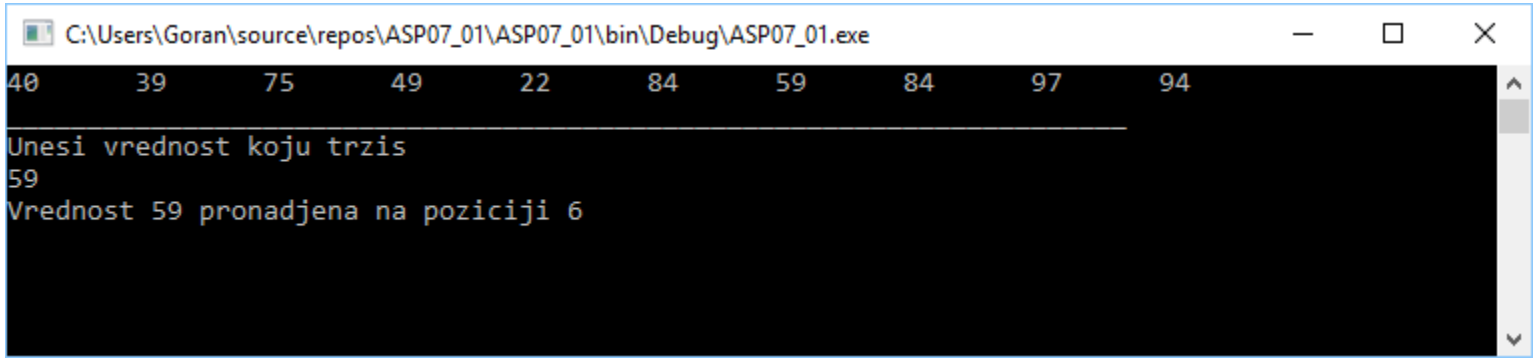
U najgorem slučaju imamo 3n+3 instrukcija

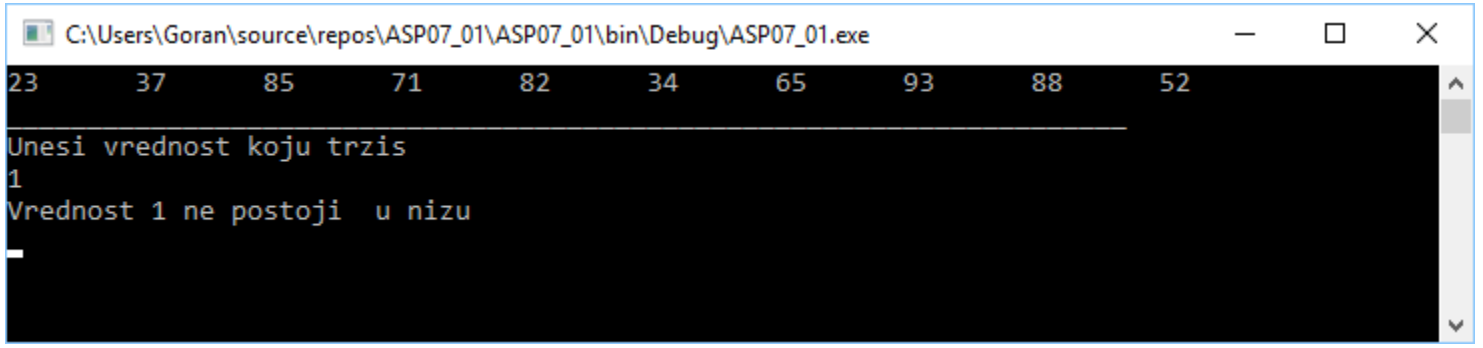
T(n) = 3n+3
O(n) = n

Poziv algoritma za sekvencijalnu pretragu

```
static void Main(string[] args)
{
    int[] x = KreirajNiz(10);
    PisiNiz(x);
    Linija(70);

    Console.WriteLine("Unesi vrednost koju trzis");
    int a = int.Parse(Console.ReadLine());

    int indeks = LinSearch(x, a);

    if (indeks > -1)
    {
        Console.WriteLine($"Vrednost {a} pronadjena na poziciji {indeks}");
    }
    else
    {
        Console.WriteLine($"Vrednost {a} ne postoji  u nizu");
    }

    Console.ReadLine();
}
```

Prikaz rezultata pretrage





Analiza linearne pretrage

- Najbolji slučaj: tražena vrednost se nalazi na početku niza
  - 3 vremenske jedinice
  - Vremenska komplesnost $O(1)$
- Najgori slučaj: tražena vrednost nije u nizu
  - 3*n+3
  - Vremenska kompleksnost je $O(n)$
- Prosečan slučaj: tražena vrednost se nalazi na poziciji i
  - Broj poređenja 3* i + 3, i <n
  - Vremenska kompleksnost je $O(n)$

Linearna pretraga korišćenjem stražara (sentinel)

- Čuvamo poslednji element niza u pomoćnoj promenljivoj
- Traženu vrednost privremeno upisujemo na poslednji indeks niza (postavljamo stražara)
- Petlja while se izvršava dok se ne naiđe na vrednost jednaku traženoj (bilo da je to pravi element ili stražar)
- Nakon završetka petlje vraćamo originalni poslednji element niza
- Ako je pronađeni indeks manji od n−1, element je stvarno pronađen, u suprotnom moguće je da je pronađen stražar
- Izbegava se provera uslova i < n u svakoj iteraciji ima manje instrukcija nego kod klasične pretrage
- Najveća ušteda je kada element ne postoji u nizu, jer sentinel uklanja proveru granice u svakoj iteraciji, pa je ukupno vreme značajno manje

Linearna pretraga korišćenjem stražara - implementacija

```
static int LinSearchSentinel(int[] x, int a)
{
    int n = x.Length;
    int poslednji = x[n - 1];
    x[n - 1] = a;
    int i = 0;
    while (x[i] != a)
    {
       i++;
    }
    x[n - 1] = poslednji;

    if ((i < n - 1) || (a == x[n - 1]))
    {
        return i;
    }
    else
    {
        return -1;
    }
}
```

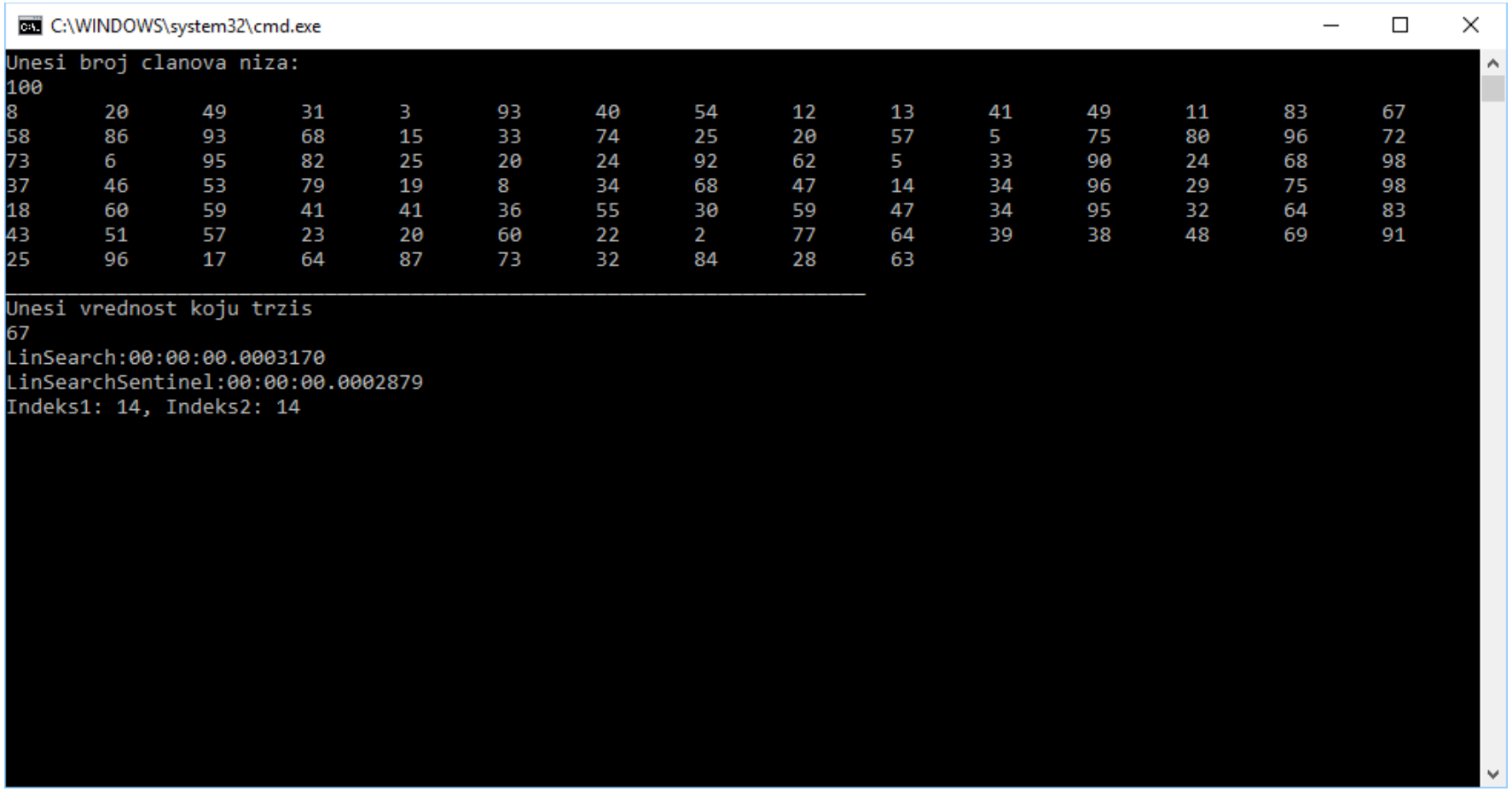
2*n+7

Radnja	Broj jedinica vremena	Komentar
Određivanje dužine niza (int n = x.Length)	1	Jedinica vremena za određivanje dužine niza.
Čuvanje poslednjeg elementa (poslednji = x[n - 1])	1	Jedinica vremena za spremanje poslednjeg elementa.
Postavljanje poslednjeg elementa na traženi element (x[n - 1] = a)	1	Jedinica vremena za postavljanje poslednjeg elementa.
Inicijalizacija indeksa (i = 0)	1	Inicijalizacija promenljive i.
Provera uslova u while (x[i] != a)	n	Petlja se izvršava do kraja niza, uslov proveravamo n puta.
Inkrementiranje (i++)	n	Inkrementacija u svakoj iteraciji petlje, n puta.
Vraćanje poslednjeg elementa na mesto (x[n - 1] = poslednji)	1	Jedinica vremena za vraćanje poslednjeg elementa.
Provera uslova i < n - 1 u if	1	Provera uslova i < n - 1 u if izrazu.
Provera uslova a == x[n - 1] u if	1	Provera uslova a == x[n - 1] u if izrazu.
OR operacija	1	Operacija OR između provere i < n - 1 i a == x[n - 1].
	2n + 7	Ukupno vreme izvršavanja u najgorem slučaju.

Esperimentalno upoređivanje algoritama za linearnu pretragu

```
static void Main(string[] args)
{
    Console.WriteLine("Unesi broj clanova niza: ");
    int n = int.Parse(Console.ReadLine());
    int[] x = KreirajNiz(n);
    PisiNiz(x);
    Linija(70);

    Console.WriteLine("Unesi vrednost koju trzis");
    int a = int.Parse(Console.ReadLine());

    Stopwatch t1 = new Stopwatch();
    t1.Start();
    int indeks1 = LinSearch(x, a);
    t1.Stop();
    TimeSpan vreme1 = t1.Elapsed;
    t1.Reset();


    t1.Start();
    int indeks2 = LinSearchSentinel(x, a);
    t1.Stop();

    TimeSpan vreme2 = t1.Elapsed;

    Console.WriteLine($"LinSearch:{vreme1}");
    Console.WriteLine($"LinSearchSentinel:{vreme2}");

    Console.WriteLine($"Indeks1: {indeks1}, Indeks2: {indeks2}");


    Console.ReadLine();
}
```

Rezultat eksperimenta



Algoritam za linearnu pretragu sortiranog niza

```
static int LinSearchSort(int[] x, int a)
{
    int i =0;
    int n = x.Length;
    for ( i = 0; i < n; i++)
    {
        if (x[i] >=a)
        {
            break;
        }
    }

    if (x[i] == a)
    {
        return i;
    }
    else
    {
        return -1;
    }
}
```

Radnja	Broj instrukcija
Inicijalizacija i = 0	1
Određivanje dužine niza n = x.Length	1
Inicijalizacija u for petlji i = 0	1
Provera uslova i < n	**n + 1**
Provera x[i] >= a	**n**
Inkrementacija i++	**n**
Provera x[i] == a posle petlje	1
return vrednost	1

T(n) = 3n + 6

Linearna pretraga sortiranog niza

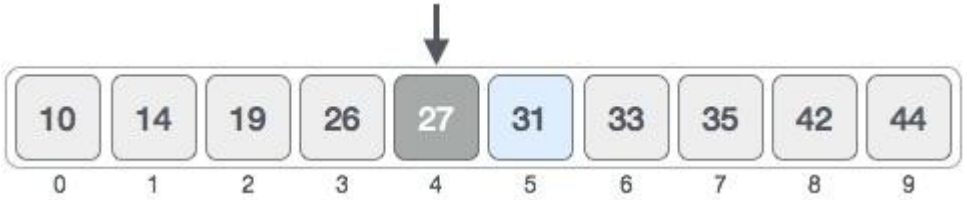
- Poboljšava vreme pretrage kada tražena vrednost ne postoji u nizu
- Zahvaljujući sortiranosti, nije potrebno pretražiti ceo niz kao u nesortiranom slučaju
- Pretraga se prekida čim naiđemo na element veći ili jednak od traženog
- U odnosu na sentinel pretragu (2n + 7), linearna pretraga sortiranog niza ima veći broj instrukcija u najgorem slučaju (3n + 6), ali je u proseku brža jer se pretraga prekida čim naiđemo na element ≥ od traženog

Binarna pretraga

traži se broj a=31

Niz sa kojim radimo mora biti sortiran.

donja =0;
gornja =9
sredina = (donja + gornja)/2 = (9+0)/2 = 4 - zaokružujemo na manju vrednost



x[4] =27 < a

ako je a == x[sredina], pronađena vrednost, kraj pretrage
ako je a>x[sredina] , pretražuje se desni podniz
ako je a<x[sredina], pretražuje se levi podniz

Binarna pretraga

traži se broj a=31

sredina =4; // stara vrednost
donja =sredina +1 =5; // kada se pretražuje desni podniz
gornja =9
sredina = (donja + gornja)/2 = (5+9)/2 = 7

x[7] =35 > a

Binarna pretraga

traži se broj a=31

sredina =7; // stara vrednost
donja = 5;
gornja =sredina -1 =6 // kada se pretražuje levi podniz
sredina = (donja + gornja)/2 = (5+6)/2 = 5

x[5] =31 = a

Implementacija algoritma

```
static int BinSearch(int[] x, int a)
{
    int n = x.Length;
    int gornja = n - 1;
    int donja = 0;
    int sredina;

    while (donja <= gornja)
    {
        sredina = (donja + gornja) / 2;

        if (a == x[sredina])
        {
            return sredina;
        }
        else if (a < x[sredina])
        {
            // pretrazujem levi podniz
            gornja = sredina - 1;
        }
        else
        {
            // pretrazujem desni podniz
            donja = sredina + 1;
        }
    }
    return -1;
}
```

Analiza algoritma binarne pretrage

- Na početku 1. iteracije, dužina oblasti pretrage iznosi n
- Na početku 2. iteracije, dužina oblasti pretrage iznosi približno n/2
- Na početku 3. iteracije, dužina oblasti pretrage iznosi približno n/4
- Na početku poslednje m-te iteracije, dužina oblasti pretrage iznosi približno $n / 2^{m-1}$

Analiza algoritma binarne pretrage -2

$n / 2^{m-1} = 1$

$m - 1 = \log_2 n$

$m = \log_2 n + 1$

$T(n) \approx m = \log_2 n$

za n =1000 broj poređenja nije veći od 10

Vremenska složenost algoritma binarne pretrage je **O(log n)**

Pitanje 1

Vremenska složenost algoritma linearnog pretraživanja je:

a. O(n)
b. O(log n)
c. $O(n^2)$

Odgovor: a

Pitanje 2

Vremenska kompleksnost algoritma binarnog pretraživanja je:

a. O(n)
b. O(log n)
c. $O(n^2)$

Odgovor: b

Pitanje 3

Algoritam binarnog pretraživanja:

a. zahteva da podaci prethodno budu sortirani
b. ne zahteva da podaci prethodno budu sortirani
c. radi i sa sortiranim i nesortiranim podacima

Odgovor: a

Pitanje 4

Koja verzija linearne pretrage ima najmanji broj instrukcija u najgorem slučaju?

a. Klasična linearna(sekvencijalna) pretraga
b. Linearna pretraga sa stražarom
c. Linearna pretraga sortiranog niza

Odgovor: b

Pitanje 5

Šta je glavna prednost sentinel pretrage?

a. Nije potrebno poređenje vrednosti
b. Izbegava se provera granice u svakoj iteraciji
c. Može da prekine pretragu ranije jer je niz sortiran

Odgovor: b

Pitanje 6

Koja pretraga najranije prekida rad u prosečnom slučaju kada traženi element ne postoji u nizu?

a. Klasična linearna pretraga
b. Linearna pretraga sortiranog niza
c. Linearna pretraga sa stražarom

Odgovor: b

Pitanje 7

U kom slučaju linearna pretraga sa stražarom ostvaruje najveću uštedu u odnosu na običnu linearnu pretragu?

a. Kada je traženi element na početku niza
b. Kada je niz sortirani rastuće
c. Kada se traženi element ne nalazi u nizu

Odgovor: c