

# СЕРМЕНТНО СТАБЛО

КАКО ОДРЕДИТИ ЗБИР ЕЛЕМЕНАТА ПОДНИЗА ДАТОГ НИЗА?

	0	1	2	3	4	5	6	7
niz	1	3	<del>2</del>	-5	7	5	8	10
prefix 0	1	4	<del>6</del>	<del>1</del>	<del>8</del>	<del>13</del>	<del>21</del>	<del>31</del>

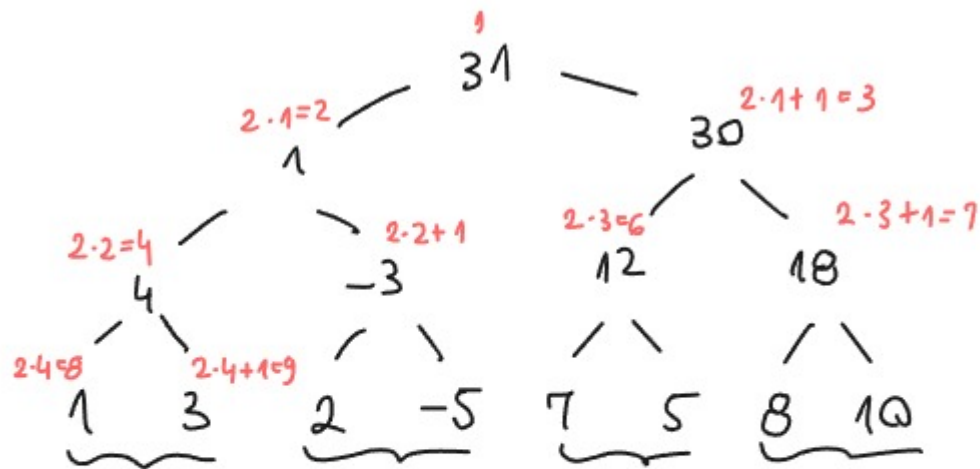
ПРЕФИКСНЕ  
СУМЕ?

$$\begin{array}{c} \uparrow \\ 2 \\ \sum_{i=0}^{2} \text{niz}[i] \end{array}$$

$$\begin{array}{c} \uparrow \\ 5 \\ \sum_{i=0}^{5} \text{niz}[i] \end{array}$$

$$\Rightarrow \text{prefix}[5] - \text{prefix}[2]$$

! ПРОБЛЕМ НАСТАЈЕ КАДА МЕНЈАМО ЕЛЕМЕНТЕ НИЗА



У СЕГМЕНТНОМ СТАБЛУ ЗА  
 СВАКИ ЧВОР  $k$  ВАЖИ ДА МУ  
 ЛЕВО ДЕТЕ ИМА ИНДЕКС  $2k$ ,  
 А ДЕСНО  $2k+1$

seg tree :      1      2      3      4      5      6      7      8      9      10      11      12      13      14      15

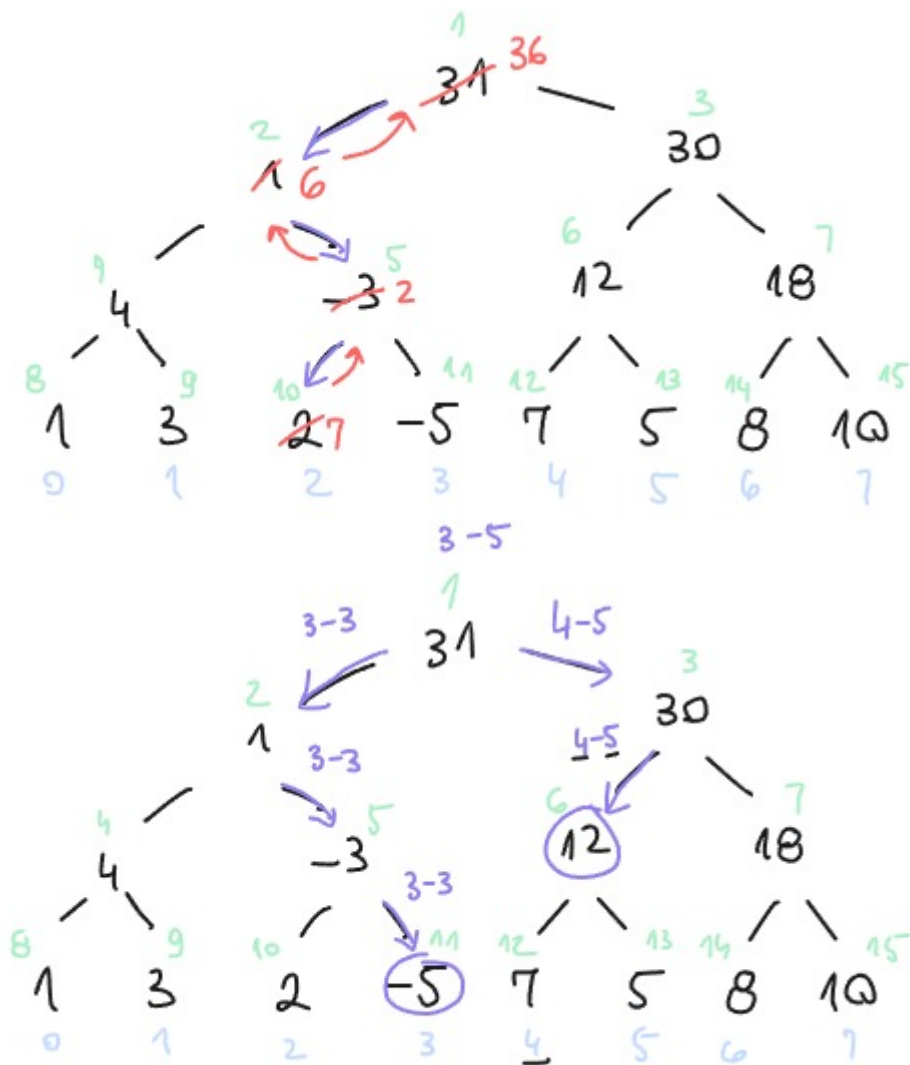
3 1    1    30    4    -3    12    18    1    3    2    -5    7    5    8    10

МЕНЬАМО ЕЛЕМЕНТ НА ПОЗИЦИЈИ 2  
У 7

СЛОЖНОСТ МЕНЬАЊА ЕЛЕМЕНТА:  
 $O(\log n)$

РАЧУНАМО ЗБИР ЕЛЕМЕНАТА  
ОД 3. ДО 5. ЕЛЕМЕНТА

$rez += -5$   
 $rez += 12$   
 $\rightarrow rez = 7$   
 $O(\log n)$



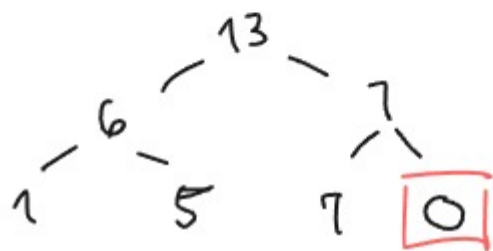
## КУЦАЊЕ

I НАЧИН (НЕ ПРЕТЕРАНО ЛЕПО) ИТЕРАТИВНО

→ НИЈЕ СТРАШНО АКО ЈЕ ДУЖИНА НИЗА  $2^k$

→ ПОСТАВИМО ДАТИ НИЗ НА КРАЈ СЕГМЕНТНОГ

→ РЕДОМ РАЧУНАМО ОСТАЛЕ ЕЛЕМЕНТЕ:  $segtree[k] = segtree[2k] + segtree[2k+1]$



← ДОПУЊУЈЕМО ДО  $2^k$   
НЕУТРАЛНИМ ЕЛЕМЕНТОМ

II НАЧИН РЕКУРЗИВНО

↓ ЧВОР У СТАБЛУ У КОМ СЕ НАЛАЗИМО

void build(int v, int tl, int tr)

{

if (tl == tr) { segtree[v] = a[tl]; return; } // АКО СМО У ЛИСТУ, УПИСУЈЕМО ГА

int mid = (tl + tr) / 2;

build(2v, tl, mid); // ГРАДИМО ЛЕВО ПОДСТАБЛО

build(2v+1, mid+1, tr); // ГРАДИМО ДЕСНО ПОДСТАБЛО

segtree[v] = segtree[2v] + segtree[2v+1]; // РАЧУНАМО ВРЕДНОСТ У V

}

ПРИМЕР РЕКУРЗИВНЕ ФЈЕ:

```
int fibonaci(int n)
{
    if (n == 1) return 1;
    if (n == 2) return 1;
    return fibonaci(n-1) + fibonaci(n-2);
}
```

a: <sup>0</sup>1 <sup>1</sup>-2 <sup>2</sup>7 <sup>3</sup>3 <sup>4</sup>-4

npbv no3ub: build(1, 0, n-1)  $\rightarrow$  segtree[1] = 5  
4

L  $\rightarrow$  mid = 2

build(2, 0, 2)  $\rightarrow$  segtree[2] = 6

$\xrightarrow{\text{mid}=1}$  build(4, 0, 1)  $\rightarrow$  segtree[4] = -1

$\xrightarrow{\text{mid}=0}$  build(8, 0, 0)  $\rightarrow$  segtree[8] = a[0] = 1

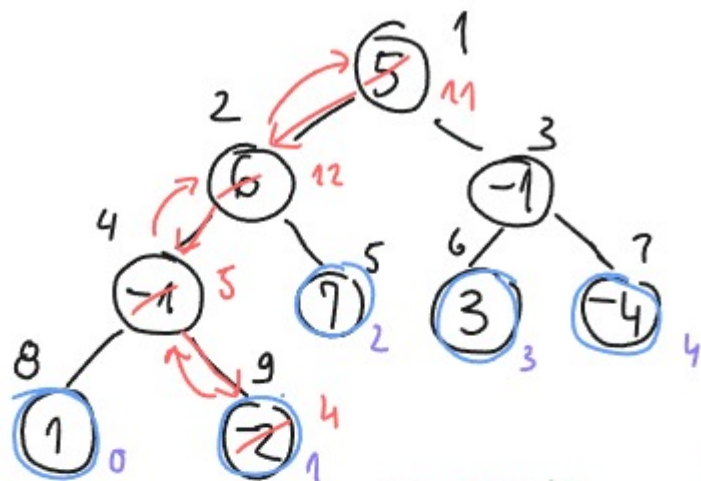
$\xrightarrow{\quad}$  build(9, 1, 1)  $\rightarrow$  segtree[9] = a[1] = -2

$\rightarrow$  build(5, 2, 2)  $\rightarrow$  segtree[5] = a[2] = 7

L  $\rightarrow$  build(3, 3, 4)  $\rightarrow$  segtree[3] = -1

$\xrightarrow{\text{mid}=3}$  build(6, 3, 3)  $\rightarrow$  segtree[6] = a[3] = 3

$\xrightarrow{\quad}$  build(7, 4, 4)  $\rightarrow$  segtree[7] = a[4] = -4



ФУНКЦИЈЕ:

```
void update (int v, int tl, int tr, int pos, int val)
```

ИСТО КАО РАНЈИЈЕ

ПОЗИЦИЈА ЕЛЕМЕНТА  
КОЈИ МЕНЈАМО

ВРЕДНОСТ КОЈУ МУ  
ДОДЕЉУЈЕМО

```
{
  if (tl == tr) { segtree[v] = val; return; }
```

// АКО СМО ДОШЛИ ДО ТРАЖЕНО ЕЛЕМЕНТА  
МЕНЈАМО ГА

```
  int mid = (tl + tr) / 2;
```

```
  if (pos <= mid) update (2v, tl, mid, pos, val); // ГЛЕДАМО ИДЕМО ЛИ У ЛЕВО
```

```
  else update (2v + 1, mid + 1, tr, pos, val); // ИЛИ ДЕСНО ПОДСТАБЛО
```

```
  segtree[v] = segtree[2v] + segtree[2v + 1]; // МЕНЈАМО ЧВОР
```

```
}
```

ИСТО КАО РАНЈИЈЕ

ГРАНИЦЕ ИНТЕРВАЛА ЗА  
КОЈИ РАЧУНАМО СУМУ

```
int sum (int v, int tl, int tr, int l, int r)
```

```
{
```

```
if (l > r) return 0; // ПРОВЕРЯВАМО ДА ЛИ НАМ ЈЕ ОВАЈ СЕГМЕН ВАЖАН
```

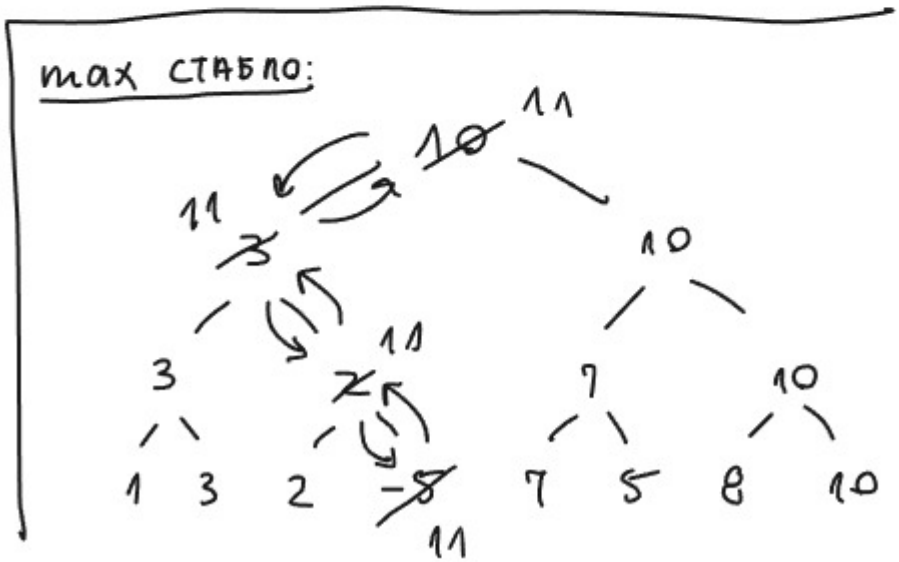
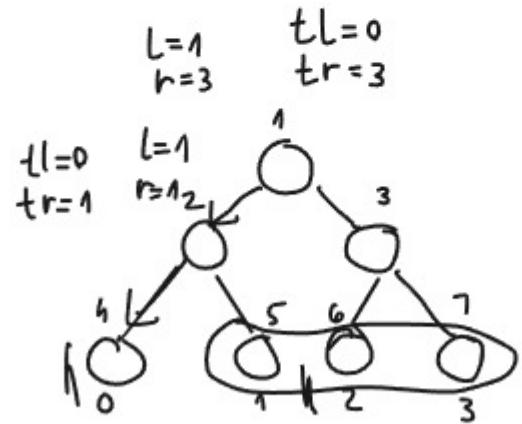
```
int mid = (tl + tr) / 2;
```

```
if (tl == l && tr == r) return segtree[v]; // ДА ЛИ ЧВОР ОБУХВАТА ЦЕО СЕГМ.
```

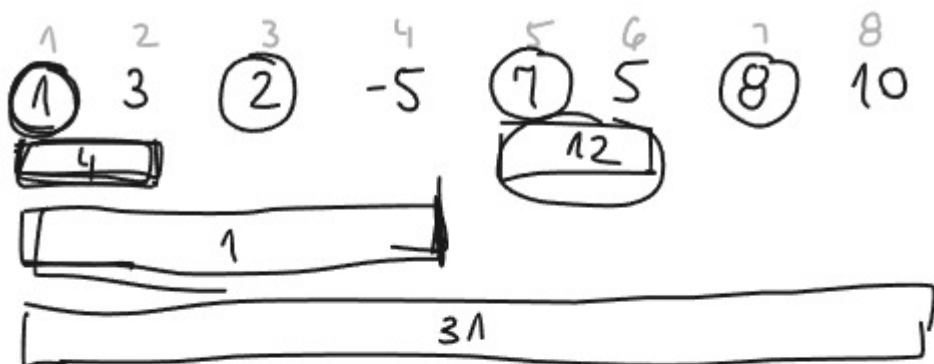
```
return sum (2v, tl, mid, l, min(r, mid)) // РАЧУНАМО ДЕЛОВЕ СЕГМЕНТА
```

```
+ sum (2v+1, mid+1, tr, max(m+1, l), r); // У ЛЕВОМ И ДЕСНОМ ПОДСТАВУ
```

```
}
```



# ФЕНВИКОВО СТАБЛО



1 4 2 1 7 12 8 31

```
void update (int pos, int val, int n)
{
```

```
  while (pos <= n)
```

```
  { fen[pos] += val;
```

```
    pos += ((-pos) & pos); ←
```

}  
4  
ВРЕДНОСТ НАЈВЕЋЕГ  
СТЕПЕНА 2 КОЈИ ДЕЛИ POS

```
int sum (int pos)    ← РАЧУНА СУМУ ОД 1. ЕЛЕМЕНТА  
{                   ДО ПОЗИЦИЈЕ POS  
  int rez=0;  
  while (pos>0)  
  {  
    rez += fen[pos];  
    pos -= ((-pos) & pos);  
  }  
  return rez;  
}
```