

Segmenttipuut

BOI-leiri 2024

Alkuperäiset diat: Topi Talvitie (2017)

Välikyselysegmenttipuu: summat

4	3	1	5	2	1	2	5
---	---	---	---	---	---	---	---

Välikyselysegmenttipuu: summat

7	6	3	7
4	3	1	5

Välikyselysegmenttipuu: summat

13				10			
7		6		3		7	
4	3	1	5	2	1	2	5

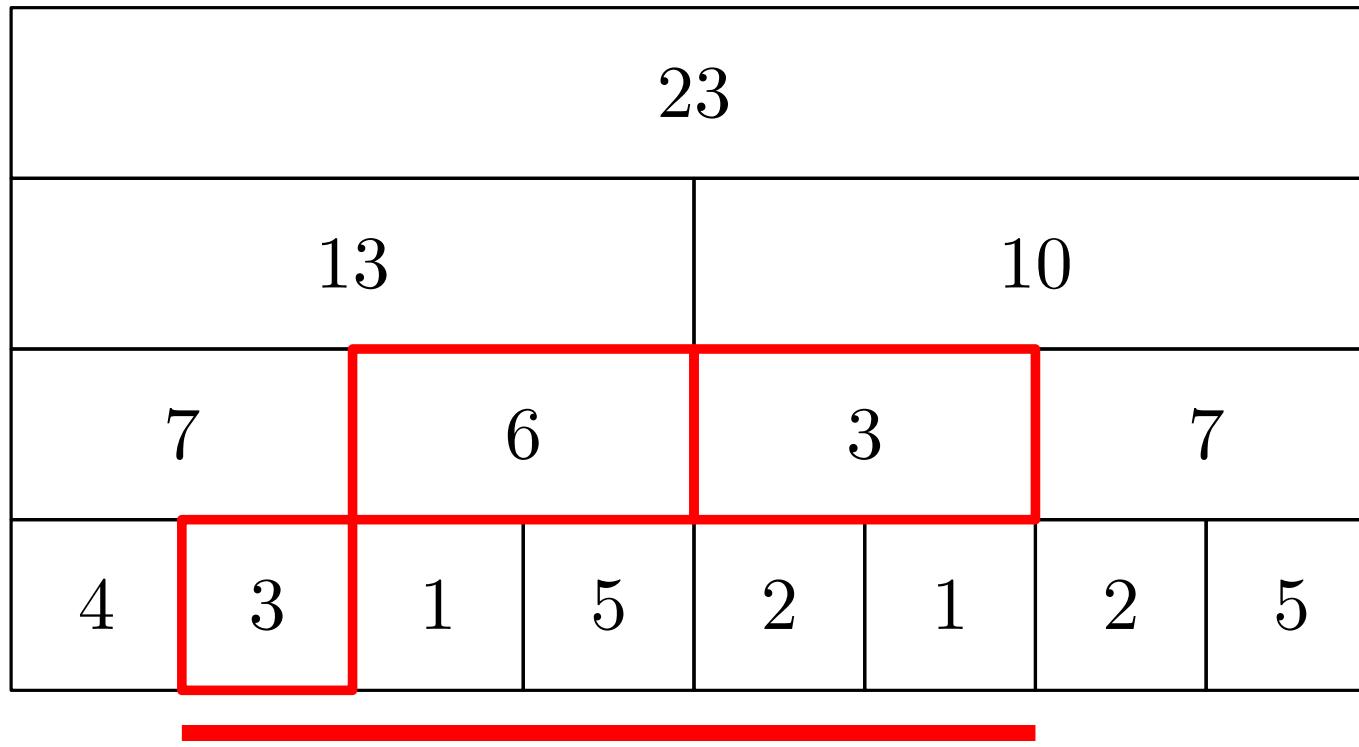
Välikyselysegmenttipuu: summat

23							
13				10			
7		6		3		7	
4	3	1	5	2	1	2	5

Välikyselysegmenttipuu: summat

23							
13				10			
7		6		3		7	
4	3	1	5	2	1	2	5

Välikyselysegmenttipuu: summat



Välikyselysegmenttipuu: summat

23							
13				10			
7		6		3		7	
4	3	1	5	2	1	2	5

Välikyselysegmenttipuu: summat

								10
7				3			7	
4	3	1	0	2	1	2	5	

Välikyselysegmenttipuu: summat

				10			
7		1		3		7	
4	3	1	0	2	1	2	5

Välikyselysegmenttipuu: summat

8				10			
7		1		3		7	
4	3	1	0	2	1	2	5

Välikyselysegmenttipuu: summat

18							
8				10			
7		1		3		7	
4	3	1	0	2	1	2	5

Välikyselysegmenttipuu: maksimit

5							
4				5			
4		1		2		5	
4	3	1	0	2	1	2	5

Välikyselysegmenttipuu: kertolaskut

0							
0				20			
12		0		2		10	
4	3	1	0	2	1	2	5

Alhaalta ylös -toteutus

23							
13				10			
7		6		3		7	
4	3	1	5	2	1	2	5

Alhaalta ylös -toteutus

```
void change(int i, int k) {  
    i += N;  
    T[i] = k;  
    for (i /= 2; i >= 1; i /= 2) {  
        T[i] = T[2 * i] + T[2 * i + 1];  
    }  
}
```

1	23															
2	13										3	10				
4	7			5	6			6	3			7	7			
$N \rightarrow$	8	9	10	11	12	13	14	15	4	3	1	5	2	1	2	5

Alhaalta ylös -toteutus

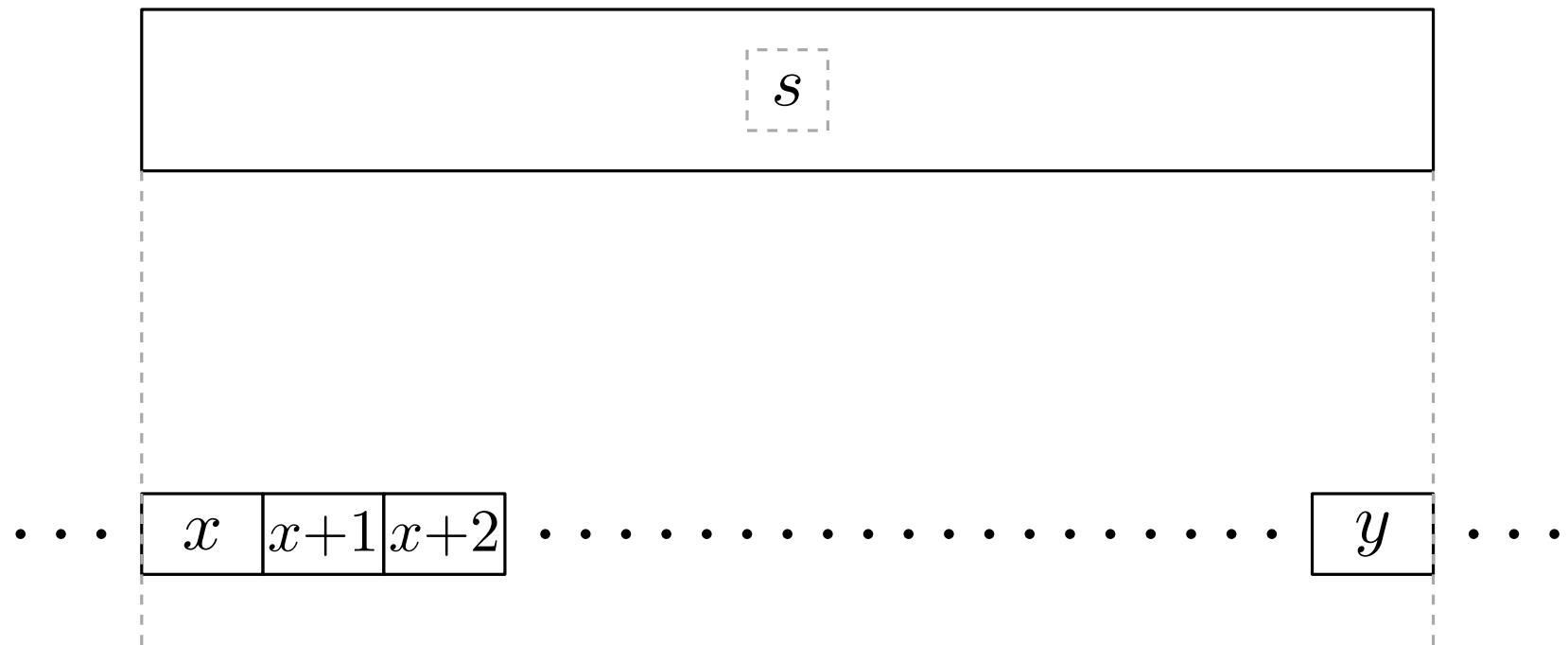
```
int query(int l, int r) {  
    l += N; r += N;  
    int s = 0;  
    while (l <= r) {  
        if (l%2 == 1) s += T[l++];  
        if (r%2 == 0) s += T[r--];  
        l /= 2; b /= 2;  
    }  
}
```

The diagram shows a 4x10 grid of numbers. The grid is divided into four quadrants by dashed lines. The top-left quadrant contains the number 1. The top-right quadrant contains the number 23. The middle-left quadrant contains the number 2. The middle-right quadrant contains the number 10. The bottom-left quadrant contains the number 4. The bottom-right quadrant contains the number 7. The bottom row of the grid is labeled with the letter N followed by an arrow pointing to the first cell of the bottom row.

1									
2									10
4	7		5	6		6	3		7
N →	8	9	10	11	12	13	14	15	
	4	3	1	5	2	1	2	5	

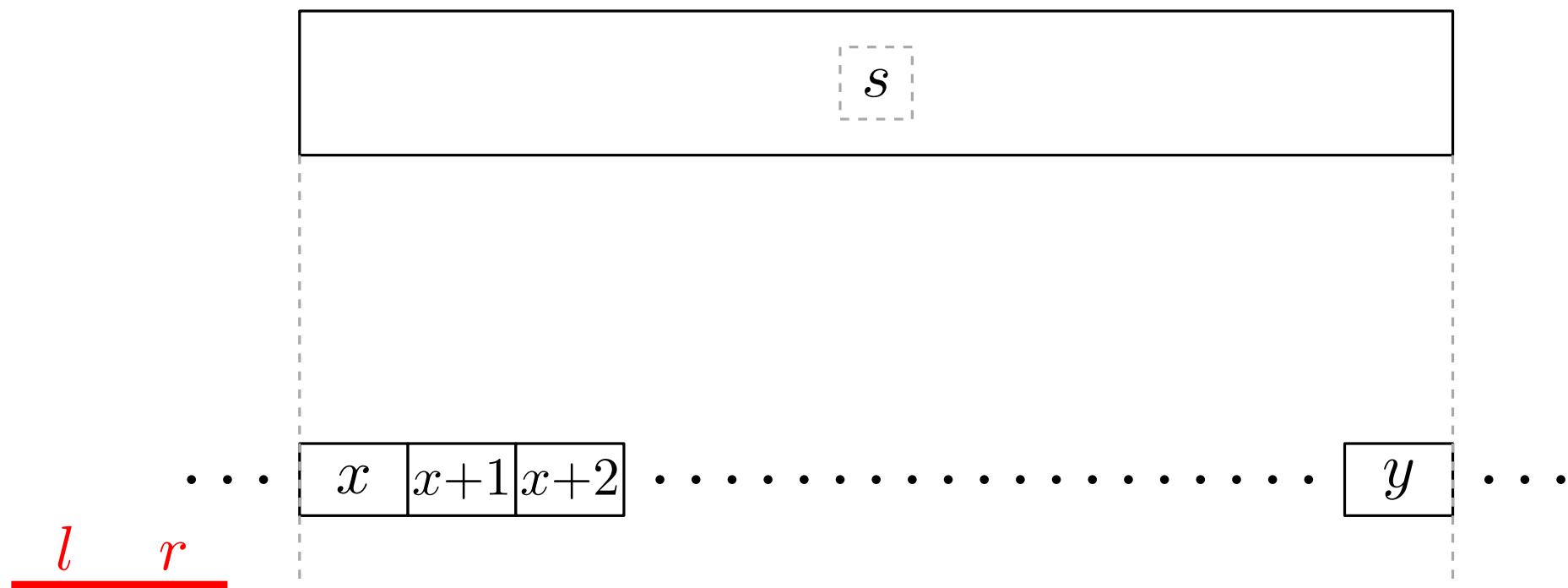
Ylhäältä alas -toteutus

```
int query(int l, int r, int s=1, int x=0, int y=N-1) {  
    if (y < l || r < x) return 0;  
    if (l <= x && y <= r) return T[s];  
    int m = (x + y) / 2;  
    return query(l, r, 2 * s, x, m) +  
        query(l, r, 2 * s + 1, m + 1, y);  
}
```



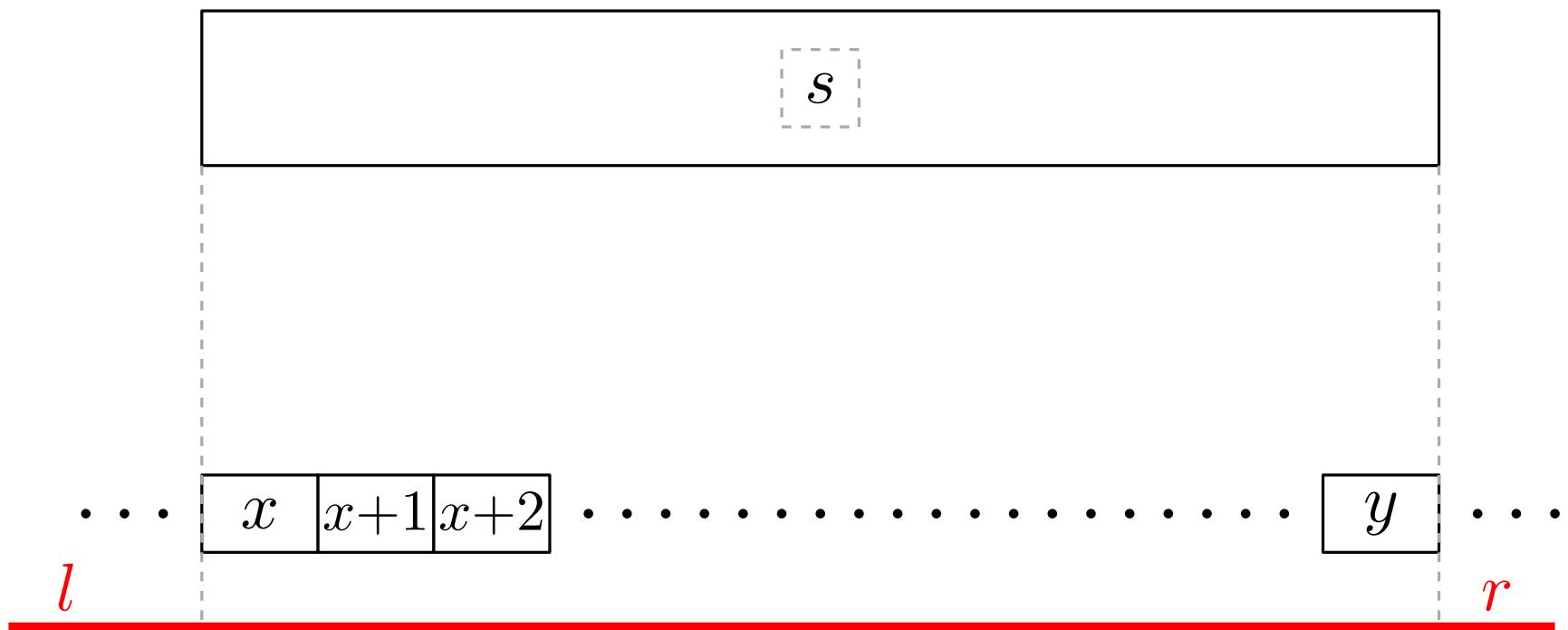
Ylhäältä alas -toteutus

```
int query(int l, int r, int s=1, int x=0, int y=N-1) {  
    if (y < l || r < x) return 0;  
    if (l <= x && y <= r) return T[s];  
    int m = (x + y) / 2;  
    return query(l, r, 2 * s, x, m) +  
        query(l, r, 2 * s + 1, m + 1, y);  
}
```



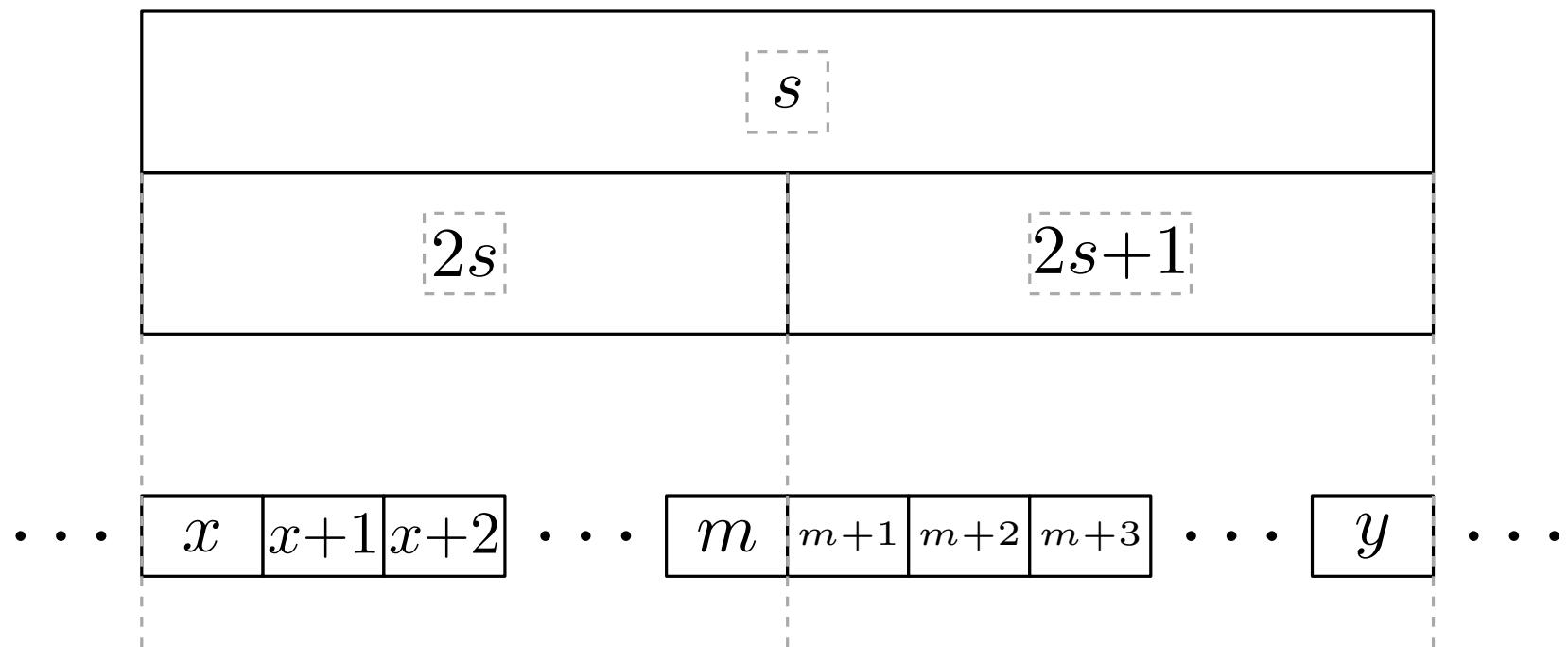
Ylhäältä alas -toteutus

```
int query(int l, int r, int s=1, int x=0, int y=N-1) {  
    if (y < l || r < x) return 0;  
    if (l <= x && y <= r) return T[s];  
    int m = (x + y) / 2;  
    return query(l, r, 2 * s, x, m) +  
        query(l, r, 2 * s + 1, m + 1, y);  
}
```



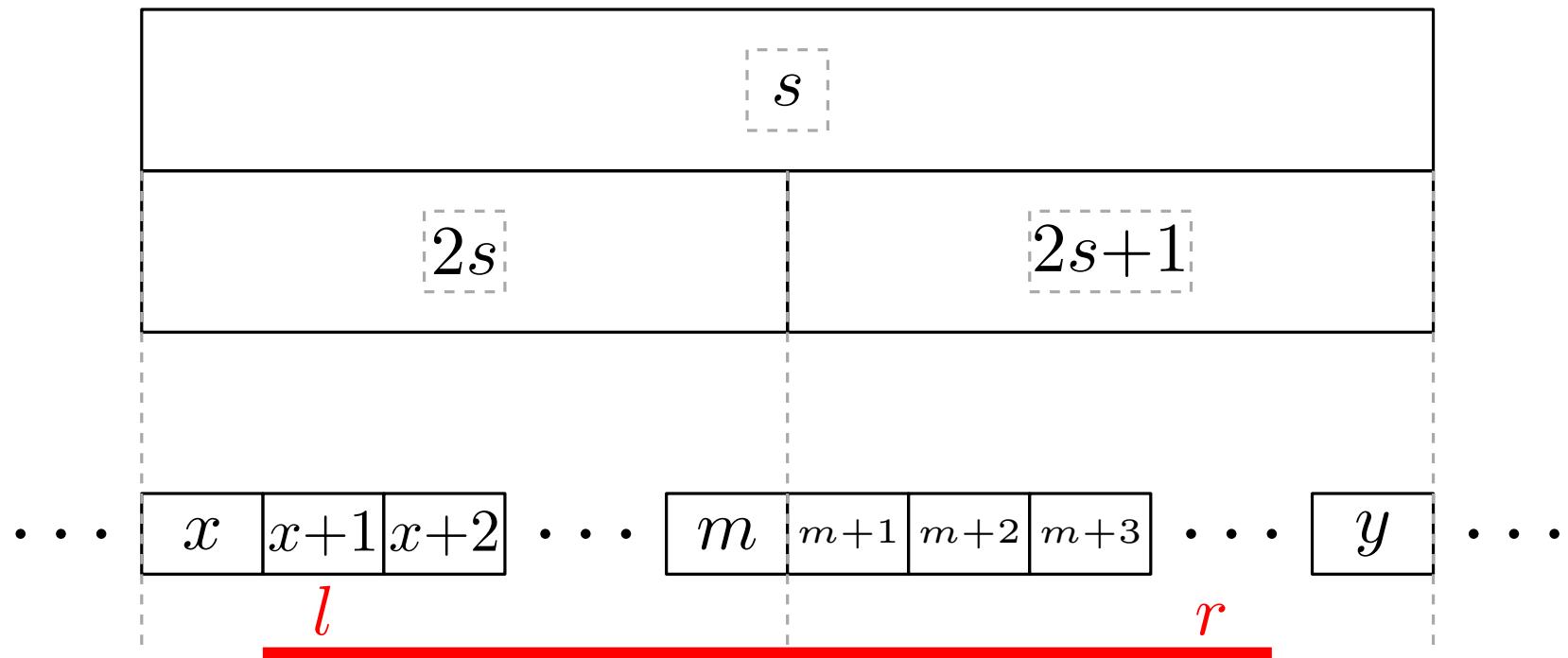
Ylhäältä alas -toteutus

```
int query(int l, int r, int s=1, int x=0, int y=N-1) {  
    if (y < l || r < x) return 0;  
    if (l <= x && y <= r) return T[s];  
    int m = (x + y) / 2;  
    return query(l, r, 2 * s, x, m) +  
        query(l, r, 2 * s + 1, m + 1, y);  
}
```



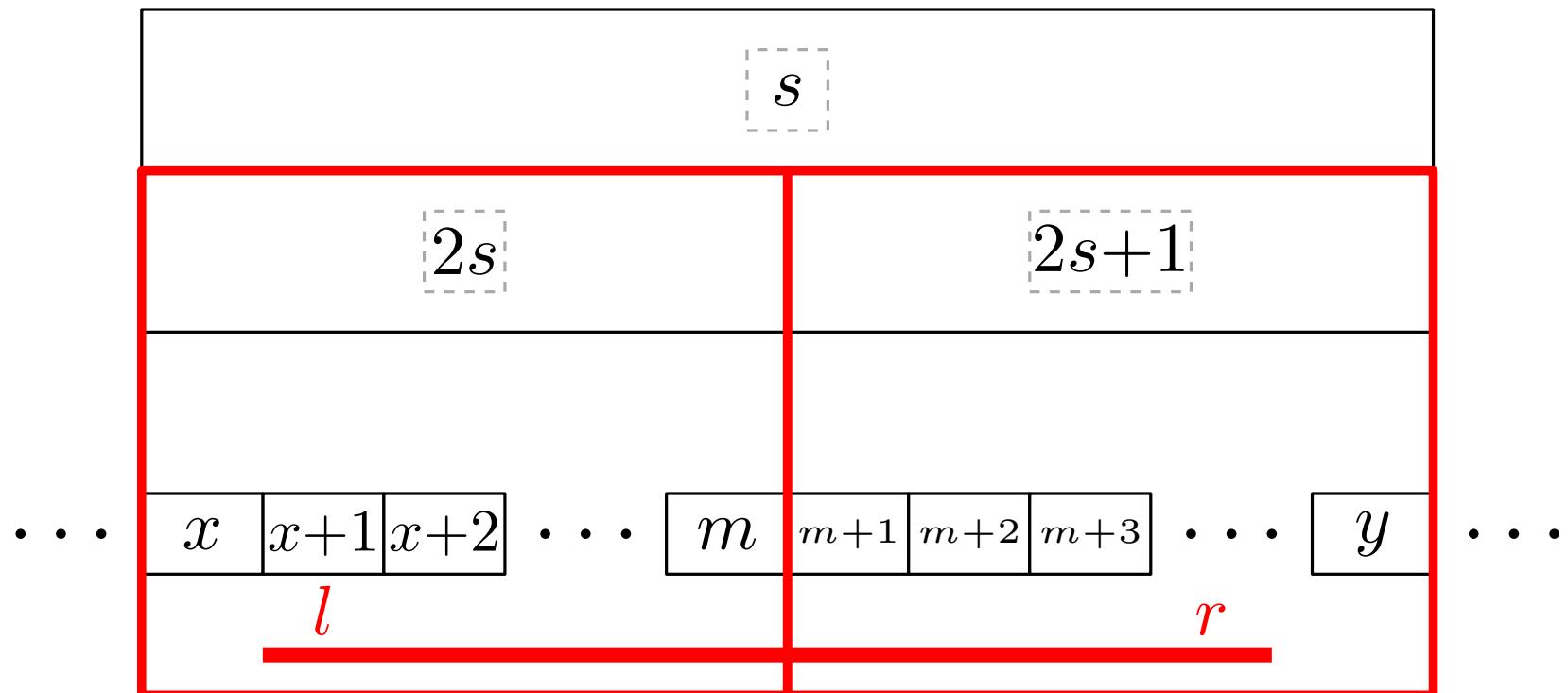
Ylhäältä alas -toteutus

```
int query(int l, int r, int s=1, int x=0, int y=N-1) {  
    if (y < l || r < x) return 0;  
    if (l <= x && y <= r) return T[s];  
    int m = (x + y) / 2;  
    return query(l, r, 2 * s, x, m) +  
        query(l, r, 2 * s + 1, m + 1, y);  
}
```



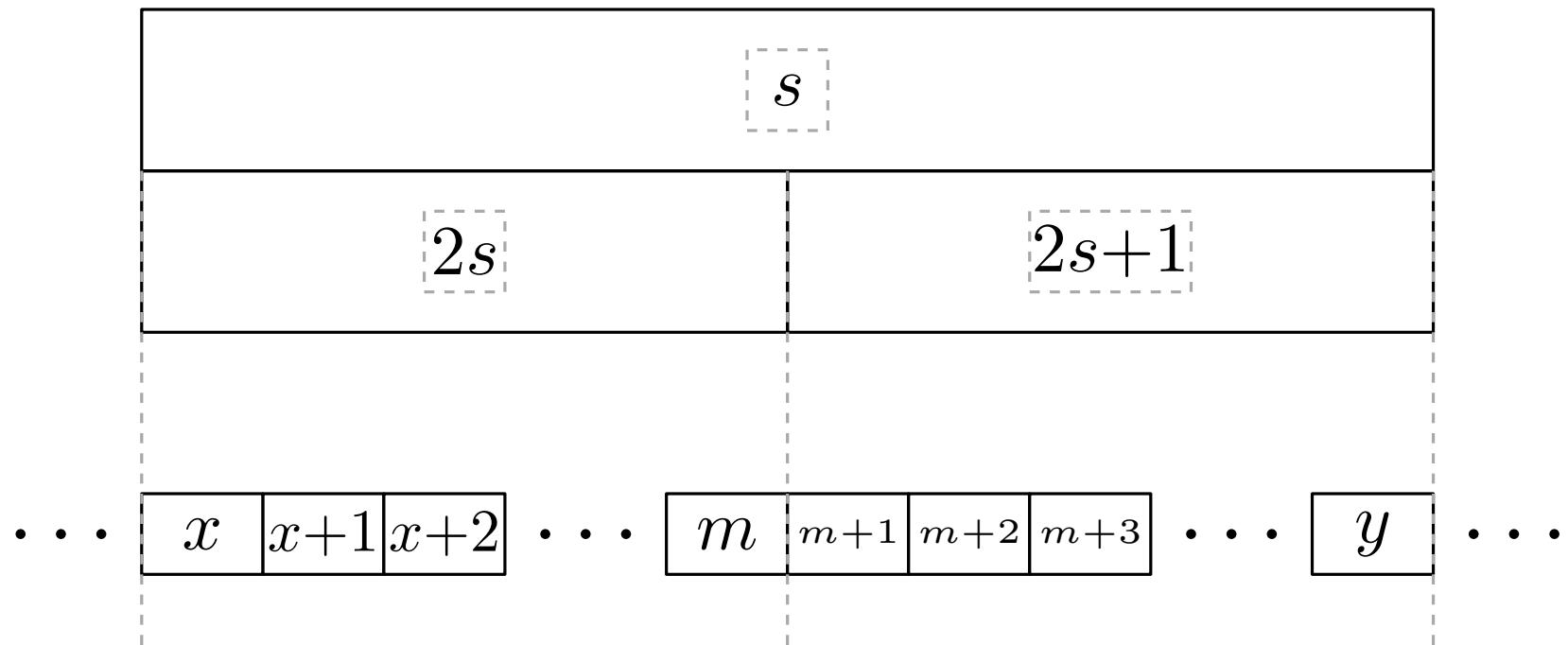
Ylhäältä alas -toteutus

```
int query(int l, int r, int s=1, int x=0, int y=N-1) {  
    if (y < l || r < x) return 0;  
    if (l <= x && y <= r) return T[s];  
    int m = (x + y) / 2;  
    return query(l, r, 2 * s, x, m) +  
        query(l, r, 2 * s + 1, m + 1, y);  
}
```



Ylhäältä alas -toteutus

```
int query(int l, int r, int s=1, int x=0, int y=N-1) {  
    if (y < l || r < x) return 0;  
    if (l <= x && y <= r) return T[s];  
    int m = (x + y) / 2;  
    return query(l, r, 2 * s, x, m) +  
        query(l, r, 2 * s + 1, m + 1, y);  
}
```



Laiska segmenttipuu

- $T[s]$: välin summa, kuten yleensä
- $D[s]$: kaikkiin solmua s vastaaviin alkioihin on lisättävä $D[s]$

```
void push(int s, int len) {
    apply(2 * s, len / 2, D[s]);
    apply(2 * s + 1, len / 2, D[s]);
    D[s] = 0;
}

void apply(int s, int len, int k) {
    T[s] += k * len;
    D[s] += k;
}

void pull(int s) {
    T[s] = T[2 * s] + T[2 * s + 1];
}
```

Laiska segmenttipuu

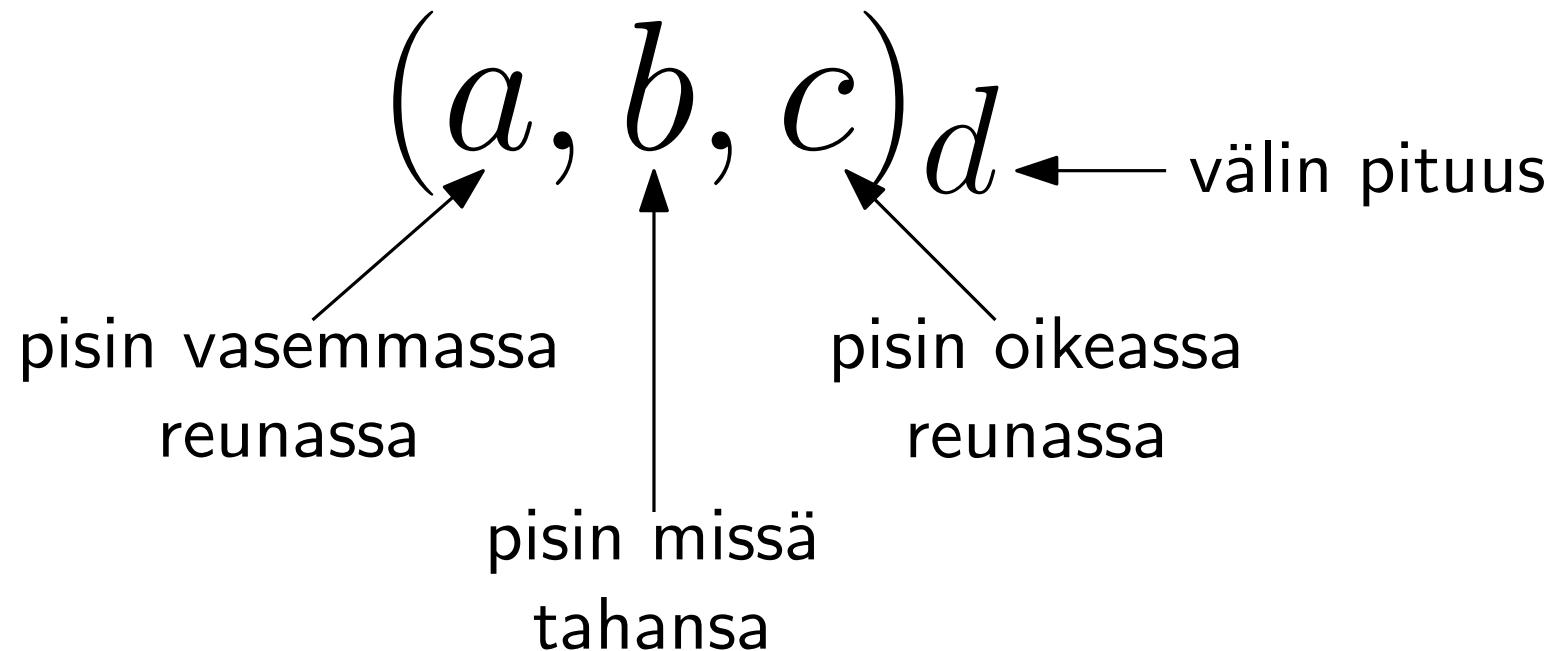
- $T[s]$: välin summa, kuten yleensä
- $D[s]$: kaikkiin solmua s vastaaviin alkioihin on lisättävä $D[s]$

```
int query(int l, int r, int s=1, int x=0, int y=N-1) {  
    if (y < l || r < x) return 0;  
    if (l <= x && y <= r) return T[s];  
    push(s, y - x + 1);  
    int m = (x + y) / 2;  
    return query(l, r, 2 * s, x, m) +  
        query(l, r, 2 * s + 1, m + 1, y);  
}  
  
void change(int l, int r, int k, int s=1, int x=0, int y=N-1) {  
    if (y < l || r < x) return;  
    if (l <= x && y <= r) return apply(s, y - x + 1, k);  
    push(s, y - x + 1);  
    int m = (x + y) / 2;  
    change(l, r, k, 2 * s, x, m);  
    change(l, r, k, 2 * s + 1, m + 1, y);  
    pull(s);  
}
```

Välikyselysegmenttipuu: yleinen operaatio

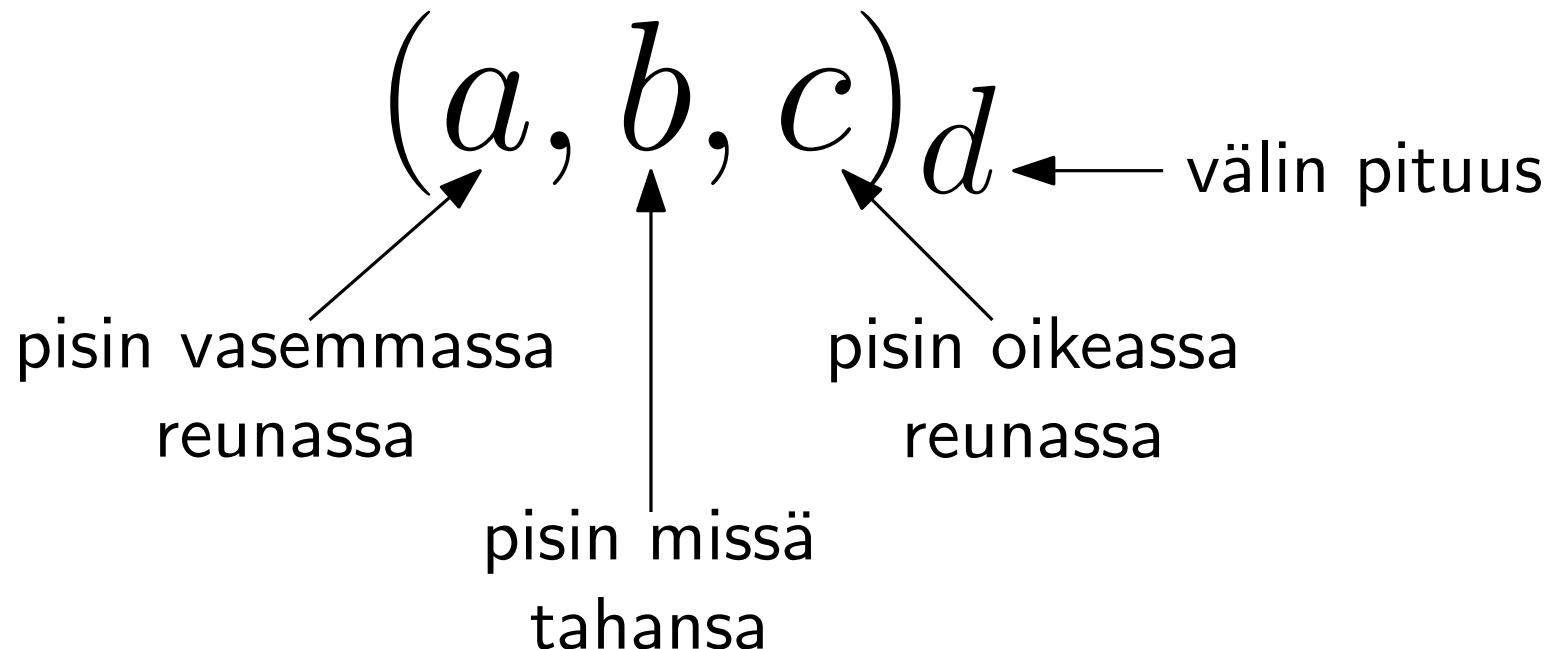
$a \oplus b \oplus c \oplus d \oplus e \oplus f \oplus g \oplus h$							
$a \oplus b \oplus c \oplus d$				$e \oplus f \oplus g \oplus h$			
$a \oplus b$		$c \oplus d$		$e \oplus f$		$g \oplus h$	
a	b	c	d	e	f	g	h

Välikyselysegmenttipuu: pisin musta ketju



$(0, 1, 1)_2$	$(0, 0, 0)_2$	$(2, 2, 2)_2$	$(0, 1, 1)_2$	$(2, 2, 2)_2$	$(1, 1, 0)_2$	$(0, 1, 1)_2$	$(2, 2, 2)_2$

Välikyselysegmenttipuu: pisin musta ketju



$(0, 1, 0)_4$	$(2, 2, 1)_4$	$(3, 3, 0)_4$	$(0, 3, 3)_4$
$(0, 1, 1)_2$	$(0, 0, 0)_2$	$(2, 2, 2)_2$	$(0, 1, 1)_2$

Välikyselysegmenttipuu: pisin musta ketju

$(0, 4, 3)_{16}$							
$(0, 2, 1)_8$				$(3, 3, 3)_8$			
$(0, 1, 0)_4$		$(2, 2, 1)_4$		$(3, 3, 0)_4$		$(0, 3, 3)_4$	
$(0, 1, 1)_2$	$(0, 0, 0)_2$	$(2, 2, 2)_2$	$(0, 1, 1)_2$	$(2, 2, 2)_2$	$(1, 1, 0)_2$	$(0, 1, 1)_2$	$(2, 2, 2)_2$

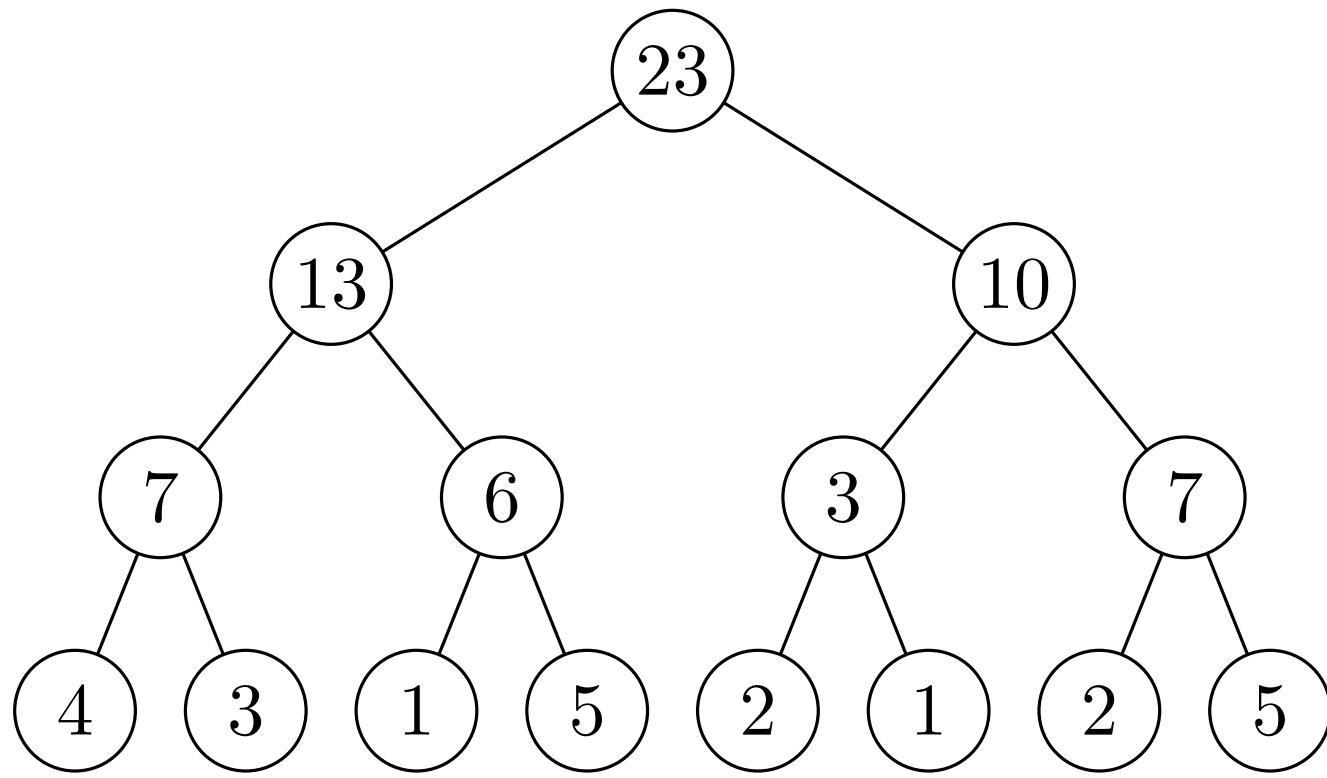
Harvat segmenttipuut

- Tallennetaan s :n vasemman lapsen indeksi $L[s]$ ja oikean lapsen indeksi $R[s]$.
- Jos indeksi on 0, solmua ei vielä ole.

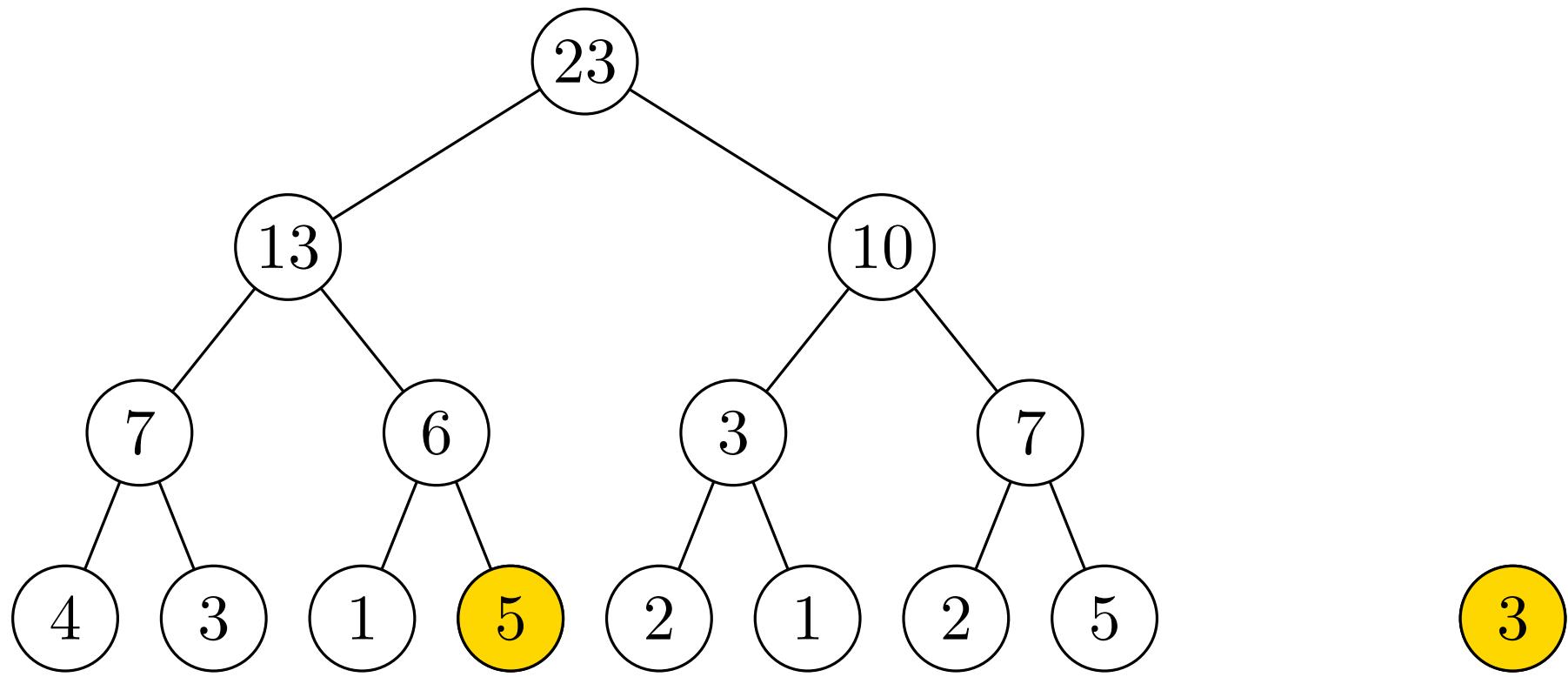
```
int idx = 1;
int left(int s) {
    if(!L[s]) {
        L[s] = idx++;
    }
    return L[s];
}
int right(int s) {
    if(!R[s]) {
        R[s] = idx++;
    }
    return R[s];
}
```

- Korvaa $2*s \rightarrow \text{left}(s)$ ja $2*s+1 \rightarrow \text{right}(s)$

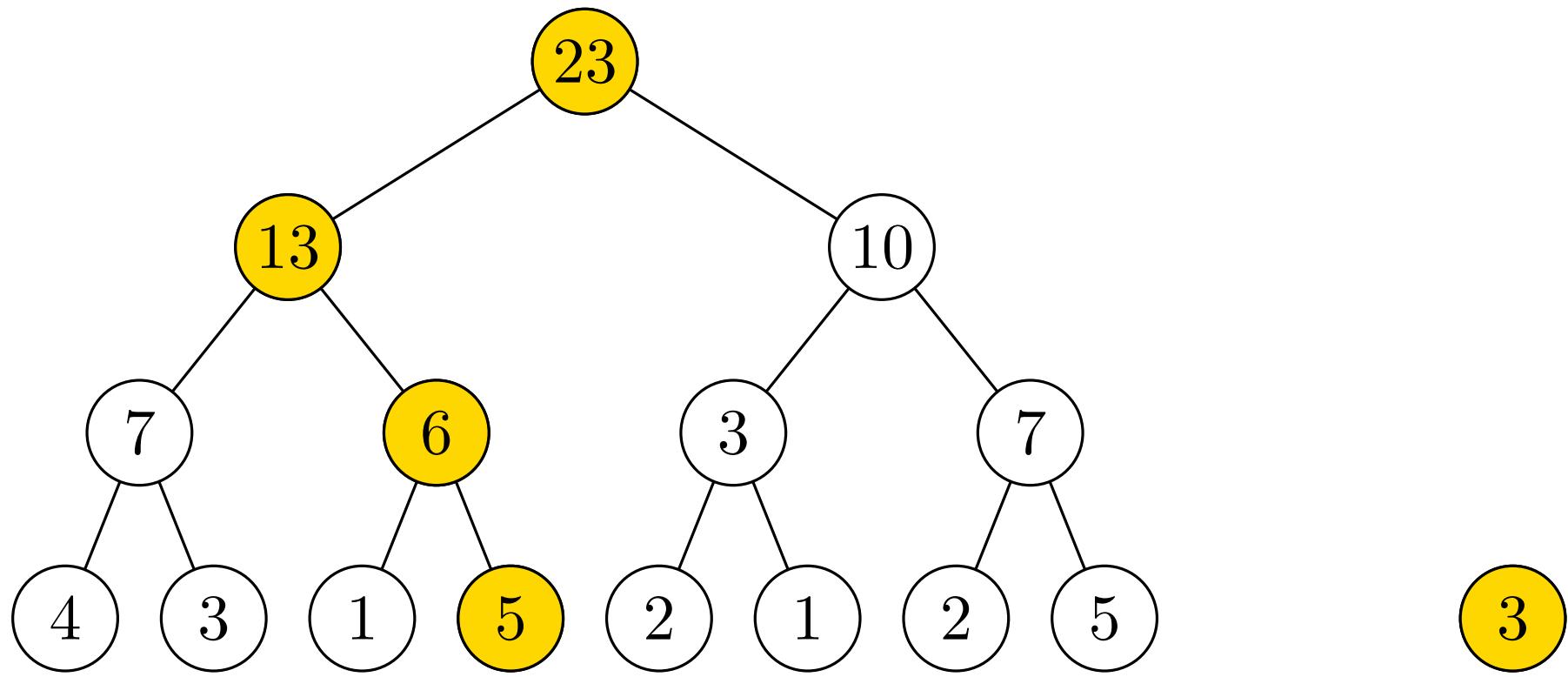
Persistentit segmenttipuut



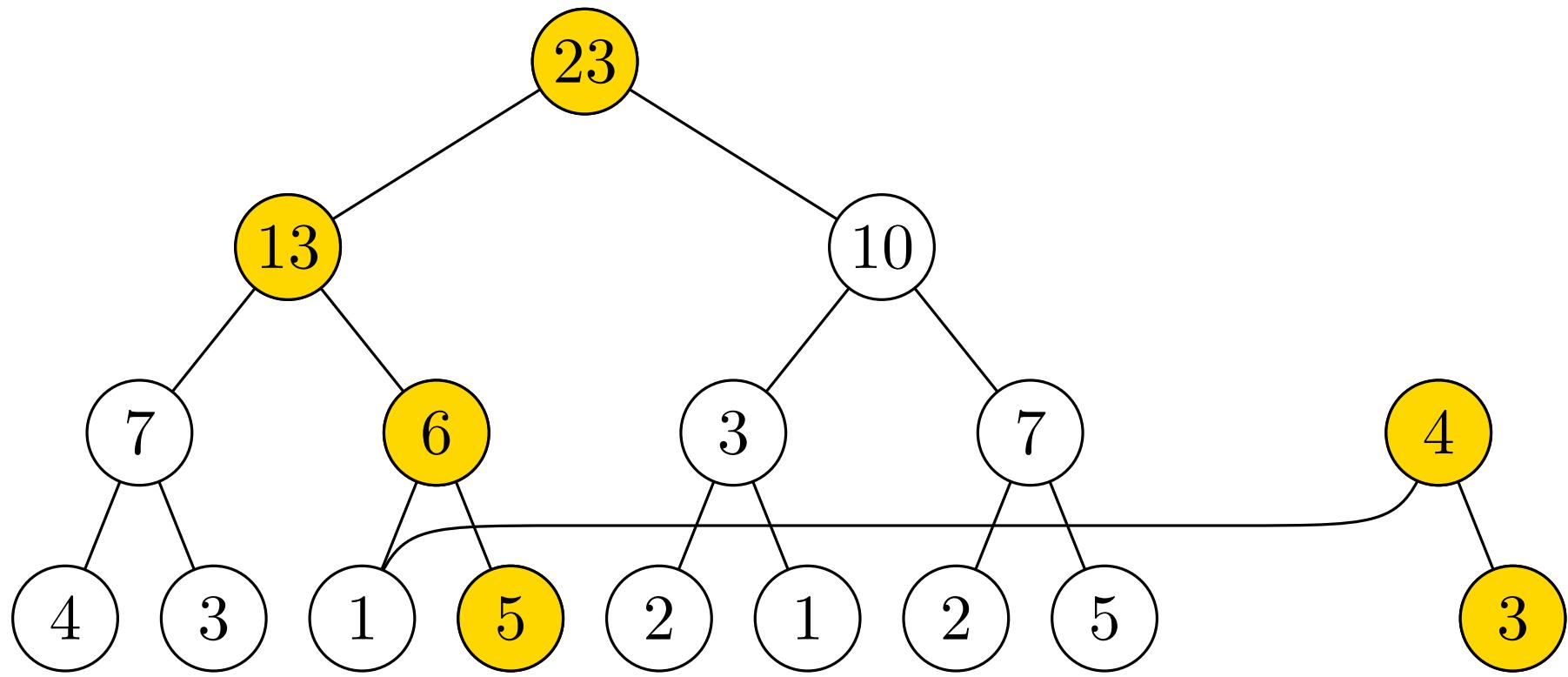
Persistentit segmenttipuut



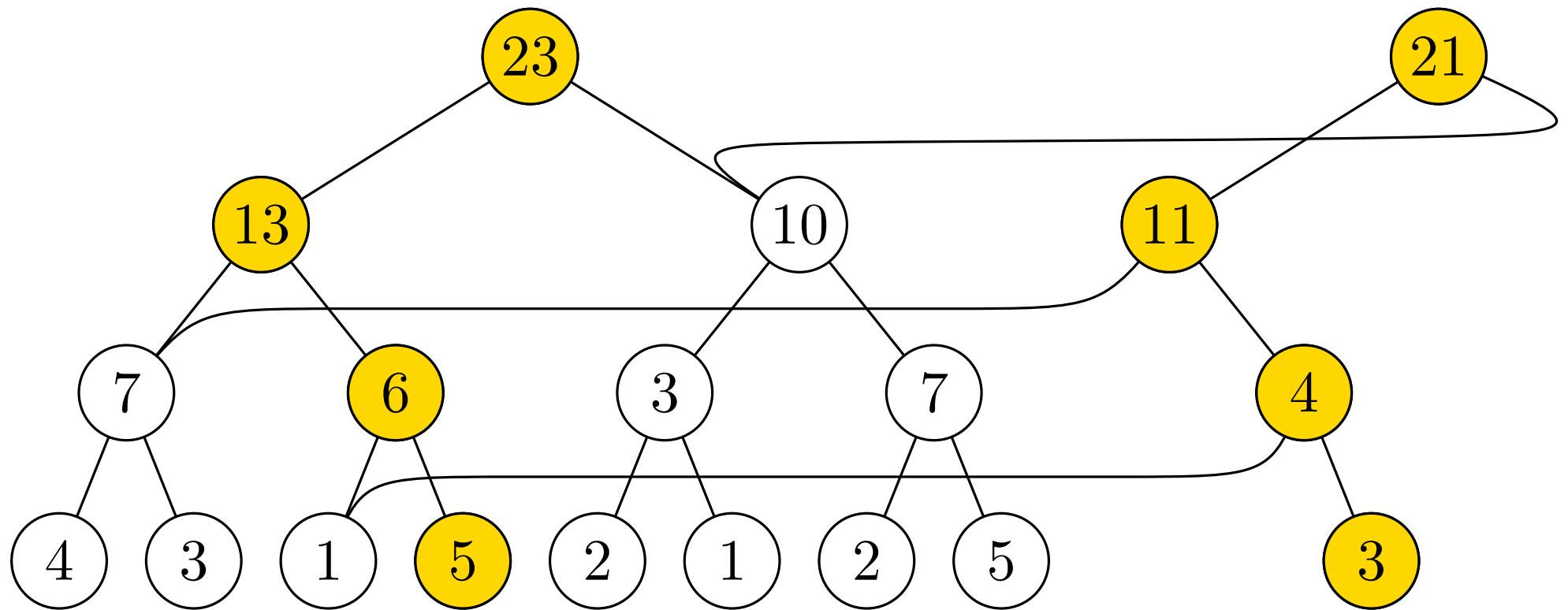
Persistentit segmenttipuut



Persistentit segmenttipuut



Persistentit segmenttipuut



Persistentit segmenttipuut

