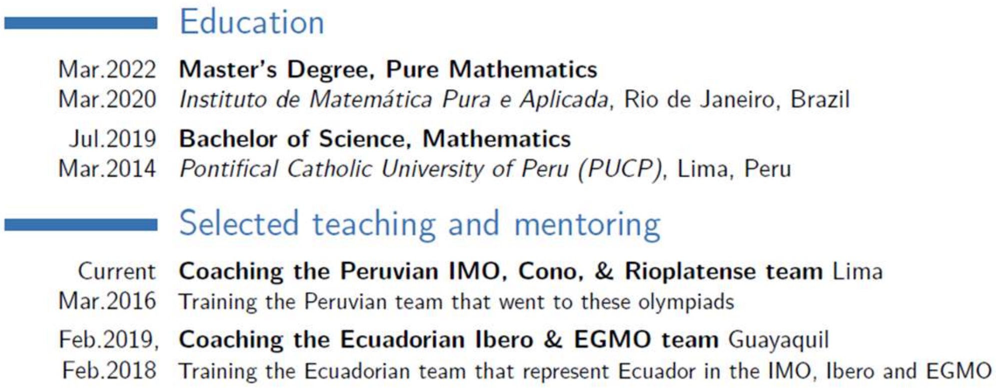
UTEC

UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA

An Introduction to Segment Tree

Angel Gerardo Napa Bernuy



Education

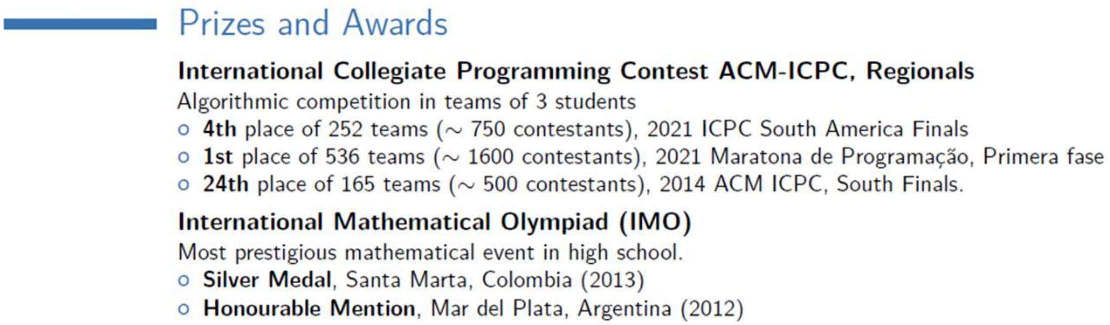
Mar.2022 – Mar.2020 **Master's Degree, Pure Mathematics**
Instituto de Matemática Pura e Aplicada, Rio de Janeiro, Brazil

Jul.2019 – Mar.2014 **Bachelor of Science, Mathematics**
Pontifical Catholic University of Peru (PUCP), Lima, Peru

Selected teaching and mentoring

Current – Mar.2016 **Coaching the Peruvian IMO, Cono, & Rioplatense team** Lima
Training the Peruvian team that went to these olympiads

Feb.2019, Feb.2018 **Coaching the Ecuadorian Ibero & EGMO team** Guayaquil
Training the Ecuadorian team that represent Ecuador in the IMO, Ibero and EGMO



Prizes and Awards

International Collegiate Programming Contest ACM-ICPC, Regionals
Algorithmic competition in teams of 3 students

- **4th** place of 252 teams (~ 750 contestants), 2021 ICPC South America Finals
- **1st** place of 536 teams (~ 1600 contestants), 2021 Maratona de Programação, Primera fase
- **24th** place of 165 teams (~ 500 contestants), 2014 ACM ICPC, South Finals.

International Mathematical Olympiad (IMO)
Most prestigious mathematical event in high school.

- **Silver Medal**, Santa Marta, Colombia (2013)
- **Honourable Mention**, Mar del Plata, Argentina (2012)

0	1	2	3	4	5
2	1	5	8	0	4

- A list of n integers
- K actions to do:

1. Sum an interval [L, R]

   Sum[1,4] = 1 + 5 + 8 + 0 = 14

2. Update an element of a[n]

0	1	2	3	4	5
2	1	3	8	0	4

0	1	2	3	4	5
2	1	5	8	0	4

1. Sum an interval [L, R]

2. Update an element of a[n]

Method 1:

0	1	2	3	4	5
2	1	5	8	0	4

1. Sum an interval [L, R]

2. Update an element of a[n]

Method 1:

0	1	2	3	4	5
2	1	5	8	0	4

1. Sum an interval [L, R]

2. Update an element of a[n]

Method 1:

0	1	2	3	4	5
2	1	5	8	0	4

Method 1:

1. Sum an interval [L, R]

O(n)

2. Update an element of a[n]

O(1)

0	1	2	3	4	5
2	1	3	8	0	4

0	1	2	3	4	5
2	1	5	8	0	4

0	1	2	3	4	5
2	3	8	16	16	20

1. Sum an interval [L, R]

2. Update an element of a[n]

Method 2:

0	1	2	3	4	5
2	1	5	8	0	4

0	1	2	3	4	5
2	3	8	16	16	20

1. Sum an interval [L, R]

2. Update an element of a[n]

Method 2:

0	1	2	3	4	5
2	1	3	8	0	4

0	1	2	3	4	5
2	1	5	8	0	4

Method 2:

0	1	2	3	4	5
2	3	8	16	16	20

1. Sum an interval [L, R]

O(1)

2. Update an element of a[n]

O(n)

0	1	2	3	4	5
2	3	6	14	14	18

0	1	2	3	4	5
2	1	5	8	0	4

	Method 1	Method 2
Complexity of K tasks:	O(nK)	O(nK)

0	1	2	3	4	5
2	1	5	8	0	4

Segment Tree

1. Sum an interval [L, R]

   $O(\log_2 n)$

2. Update an element of a[n]

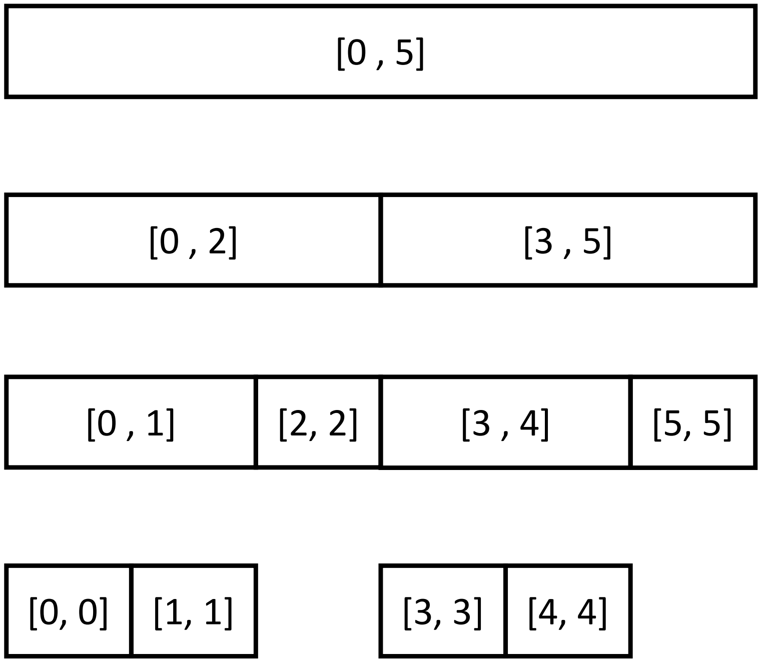
   $O(\log_2 n)$

0	1	2	3	4	5
2	1	5	8	0	4

Complexity of K tasks:

Segtree

$O(K\log_2 n)$



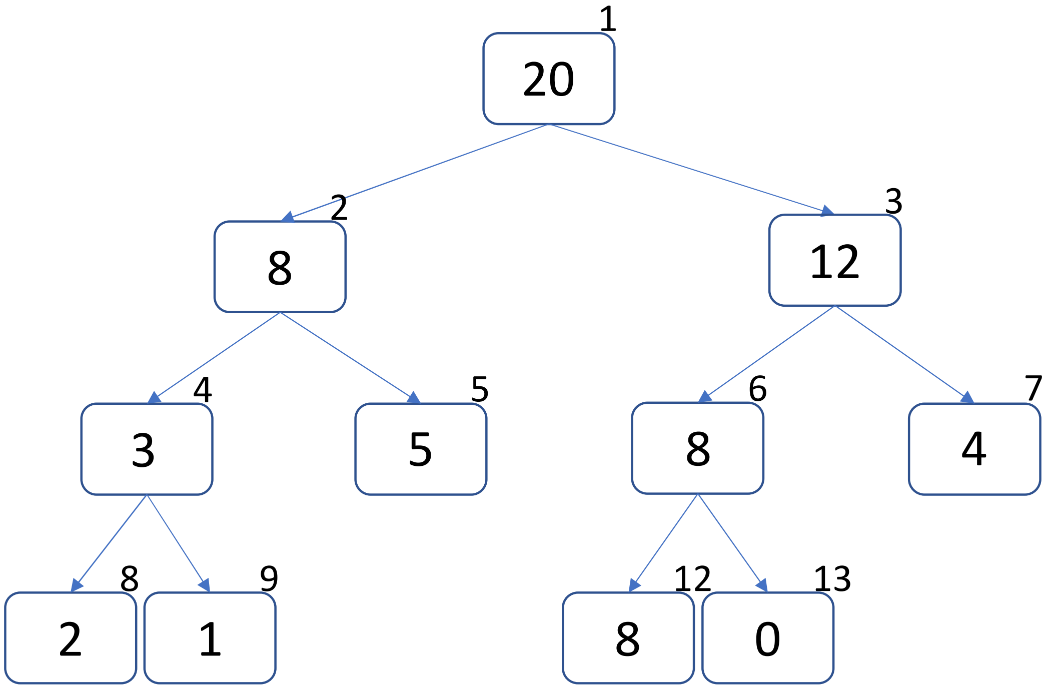
[0 , 5]

[0 , 2] [3 , 5]

[0 , 1] [2, 2] [3 , 4] [5, 5]

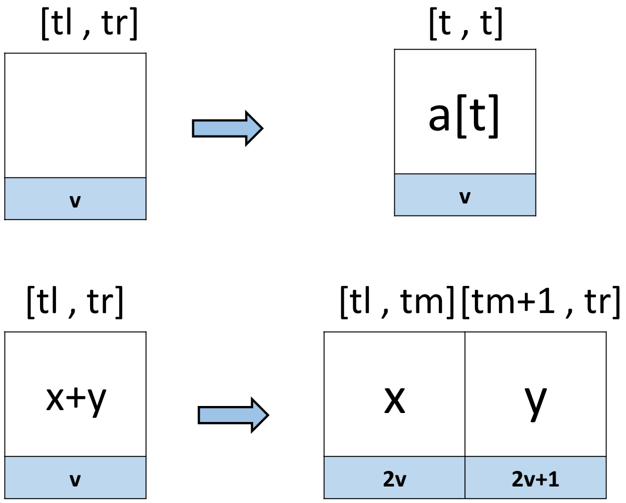
[0, 0] [1, 1] [3, 3] [4, 4]

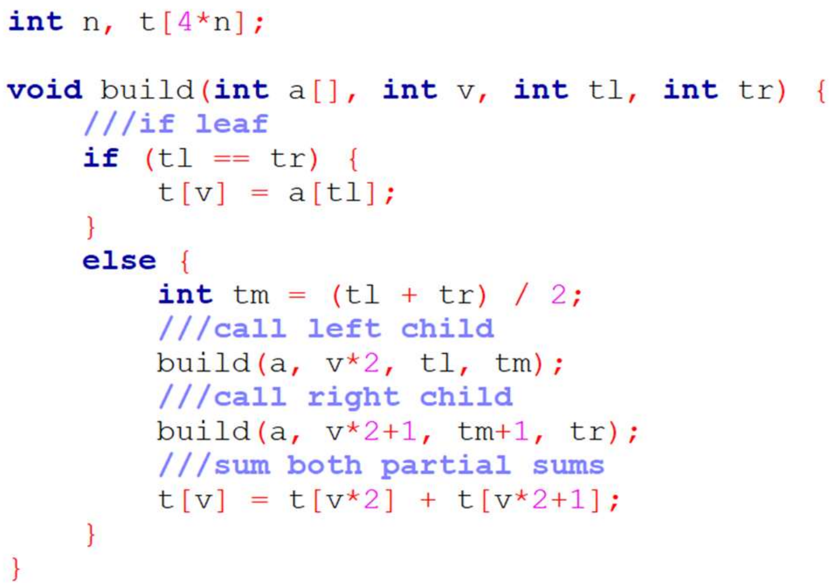
2	1	5	8	0	4
0	1	2	3	4	5



2	1	5	8	0	4
0	1	2	3	4	5

Implementation





```
int n, t[4*n];

void build(int a[], int v, int tl, int tr) {
    ///if leaf
    if (tl == tr) {
        t[v] = a[tl];
    }
    else {
        int tm = (tl + tr) / 2;
        ///call left child
        build(a, v*2, tl, tm);
        ///call right child
        build(a, v*2+1, tm+1, tr);
        ///sum both partial sums
        t[v] = t[v*2] + t[v*2+1];
    }
}
```

Implementation

[0 , 5]

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

[0 , 5]	[0 , 2]	[3 , 5]												
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

2	1	5	8	0	4
0	1	2	3	4	5

Implementation

[0 , 5]	[0 , 2]	[3 , 5]	[0 , 1]	[2 , 2]	[3 , 4]	[5 , 5]								
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

[0 , 5]	[0 , 2]	[3 , 5]	[0 , 1]	[2 , 2]	[3 , 4]	[5 , 5]	[0 , 0]	[1 , 1]			[3 , 3]	[4 , 4]		
				a[2]		a[5]								
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

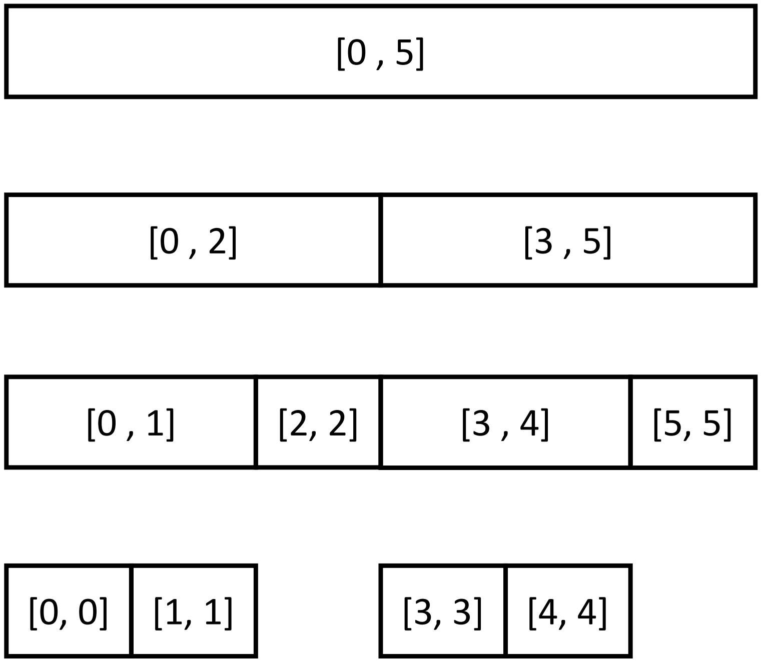
2	1	5	8	0	4
0	1	2	3	4	5

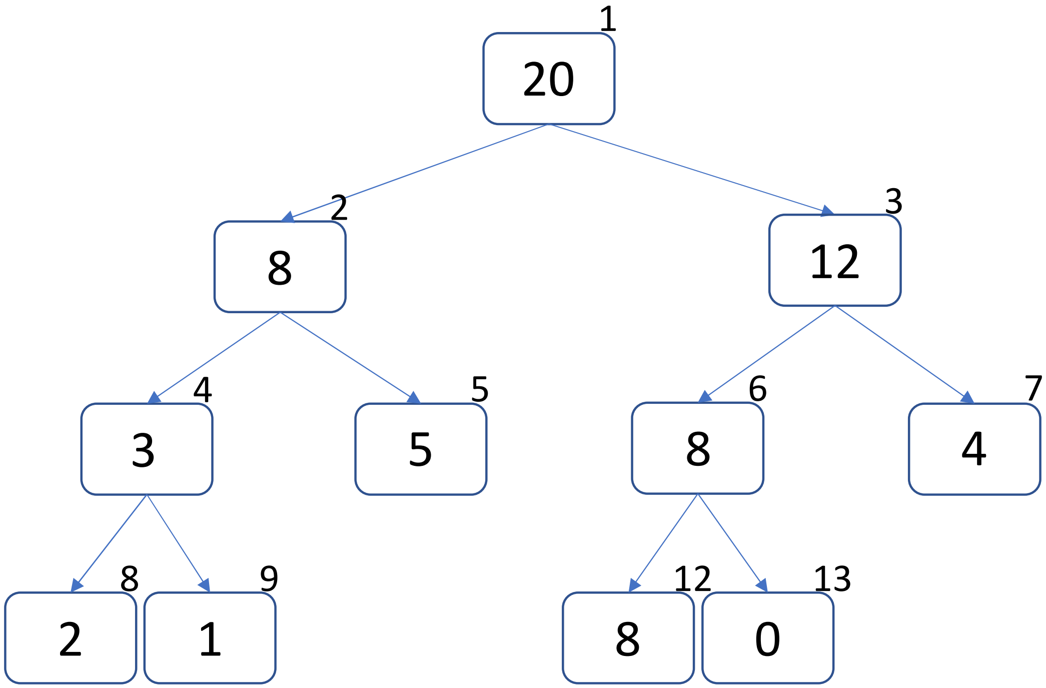
Implementation

2	1	5	8	0	4
0	1	2	3	4	5

[0 , 5]	[0 , 2]	[3 , 5]	[0 , 1]	[2 , 2]	[3 , 4]	[5 , 5]	[0 , 0]	[1 , 1]			[3 , 3]	[4 , 4]		
				5		4	a[0]	a[1]			a[3]	a[4]		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

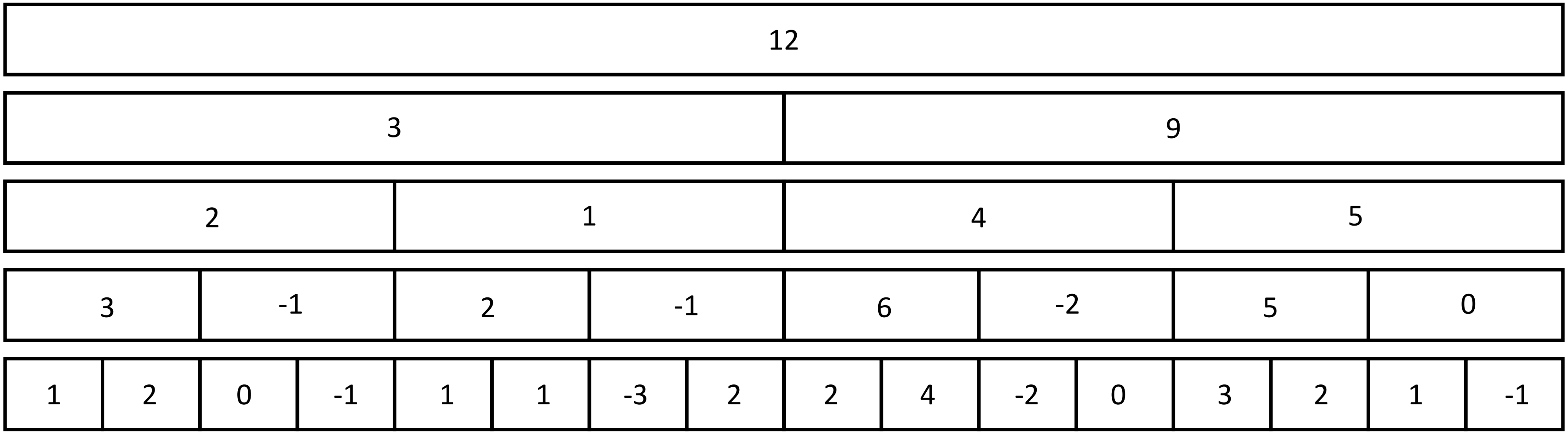
[0 , 5]	[0 , 2]	[3 , 5]	[0 , 1]	[2 , 2]	[3 , 4]	[5 , 5]	[0 , 0]	[1 , 1]			[3 , 3]	[4 , 4]		
20	8	12	3	5	8	4	2	1			8	0		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15





2	1	5	8	0	4
0	1	2	3	4	5

2	1	5	8	0	4
0	1	2	3	4	5



12

3 9

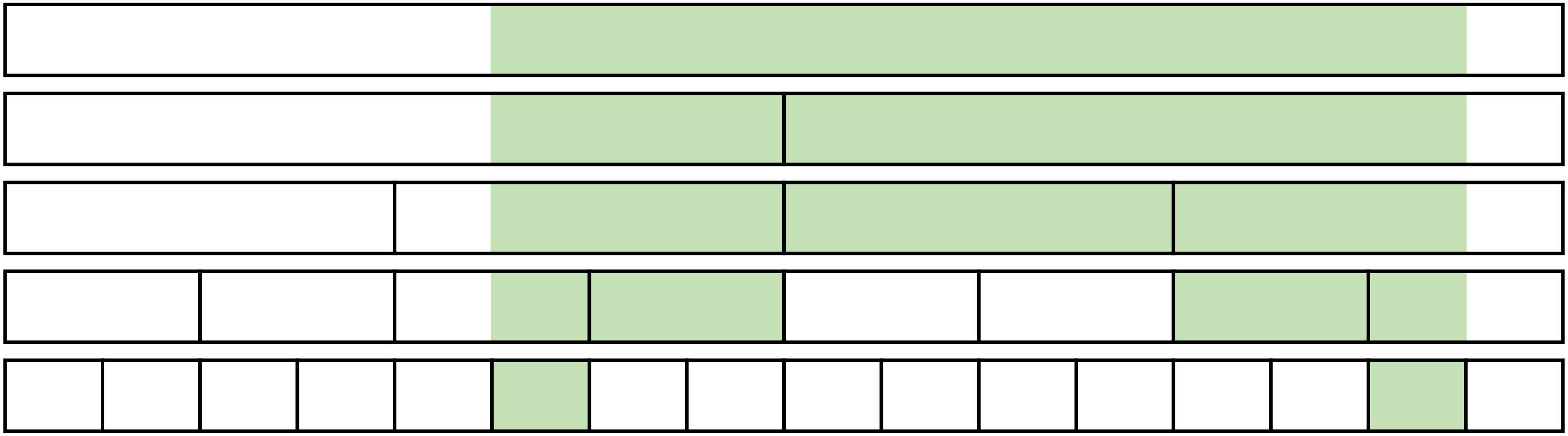
2 1 4 5

3 -1 2 -1 6 -2 5 0

1 2 0 -1 1 1 -3 2 2 4 -2 0 3 2 1 -1

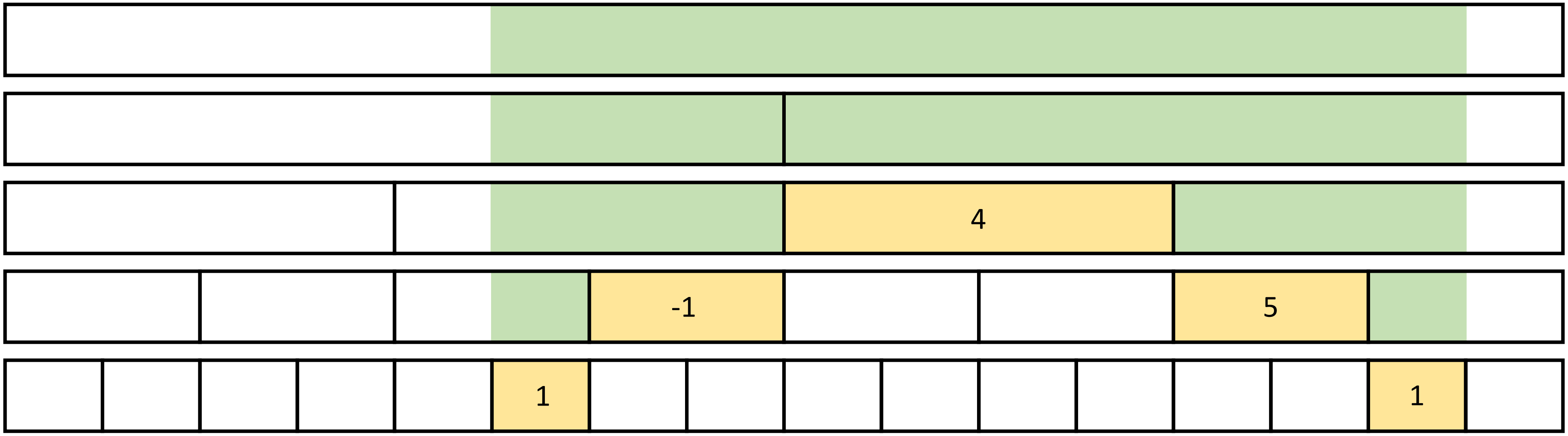
1	2	0	-1	1	1	-3	2	2	4	-2	0	3	2	1	-1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

1	2	0	-1	1	1	-3	2	2	4	-2	0	3	2	1	-1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

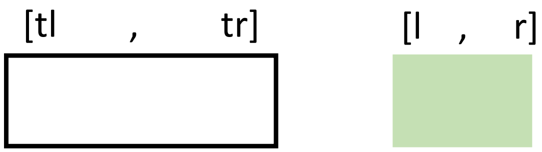


1	2	0	-1	1	1	-3	2	2	4	-2	0	3	2	1	-1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

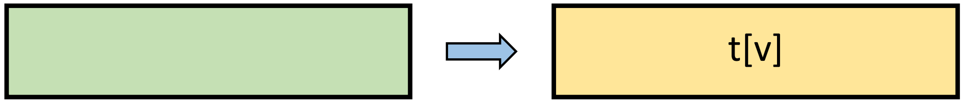
1	2	0	-1	1	1	-3	2	2	4	-2	0	3	2	1	-1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



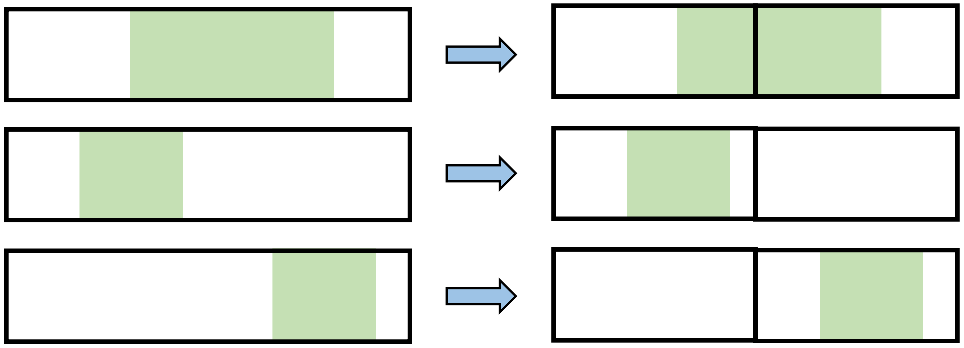
1	2	0	-1	1	1	-3	2	2	4	-2	0	3	2	1	-1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



Case 1:



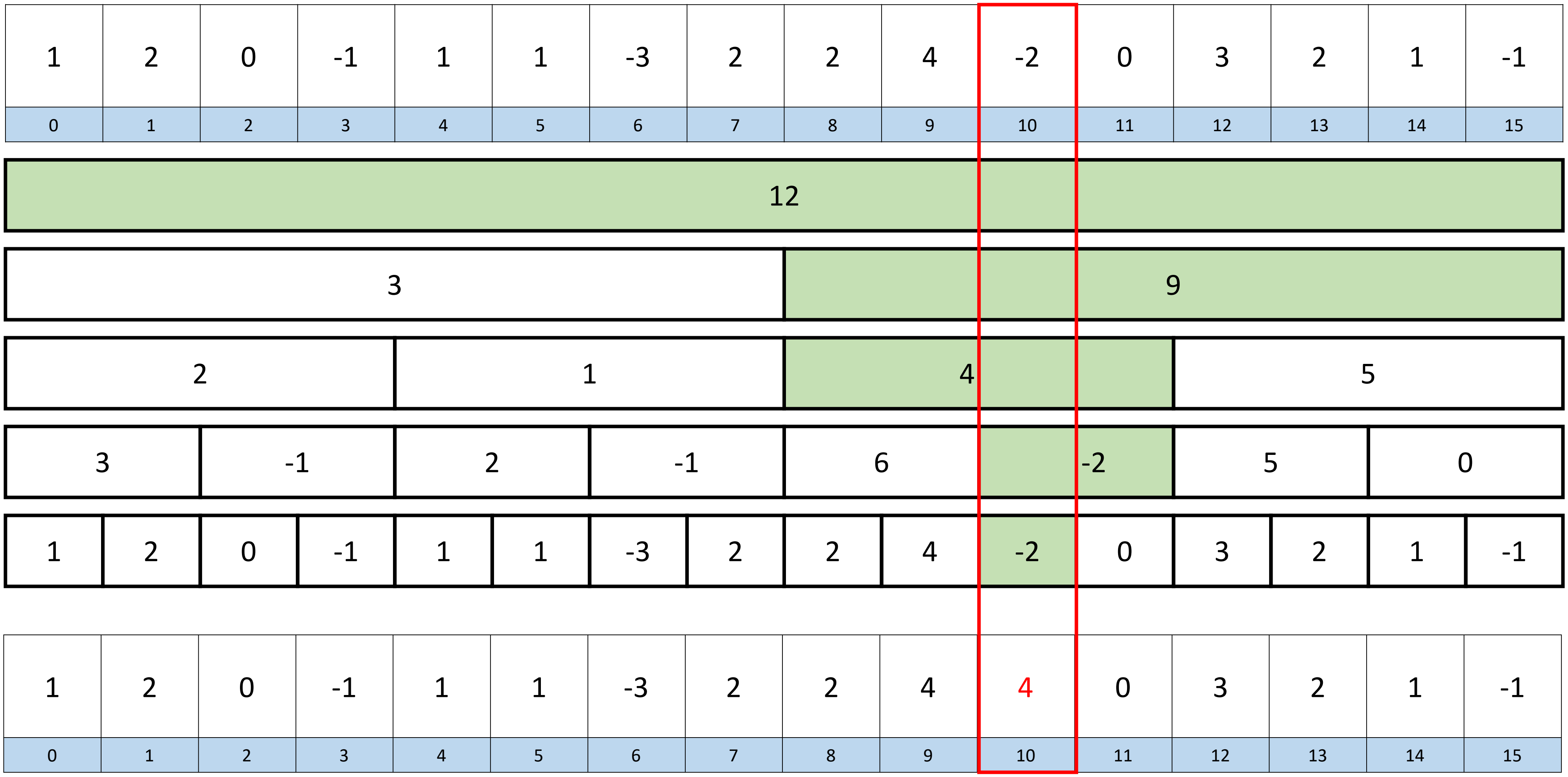
Case 2:



```
int sum(int v, int tl, int tr, int l, int r) {
    ///if nonexistent interval sum = 0
    if (l > r)
        return 0;
    /// if interval in segtree sum = value
    if (l == tl && r == tr) {
        return t[v];
    }
    /// else return sum of both children values
    int tm = (tl + tr) / 2;
    return
      sum(v*2  , tl  , tm,      l      , min(r, tm))
    + sum(v*2+1, tm+1, tr, max(l, tm+1),     r      );
}
```

<table>
<tr><td colspan="16">12</td></tr>
<tr><td colspan="8">3</td><td colspan="8">9</td></tr>
<tr><td colspan="4">2</td><td colspan="4">1</td><td colspan="4">4</td><td colspan="4">5</td></tr>
<tr><td colspan="2">3</td><td colspan="2">-1</td><td colspan="2">2</td><td colspan="2">-1</td><td colspan="2">6</td><td colspan="2">-2</td><td colspan="2">5</td><td colspan="2">0</td></tr>
<tr><td>1</td><td>2</td><td>0</td><td>-1</td><td>1</td><td>1</td><td>-3</td><td>2</td><td>2</td><td>4</td><td>-2</td><td>0</td><td>3</td><td>2</td><td>1</td><td>-1</td></tr>
</table>

1	2	0	-1	1	1	-3	2	2	4	-2	0	3	2	1	-1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



1	2	0	-1	1	1	-3	2	2	4	-2	0	3	2	1	-1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

12

3 9

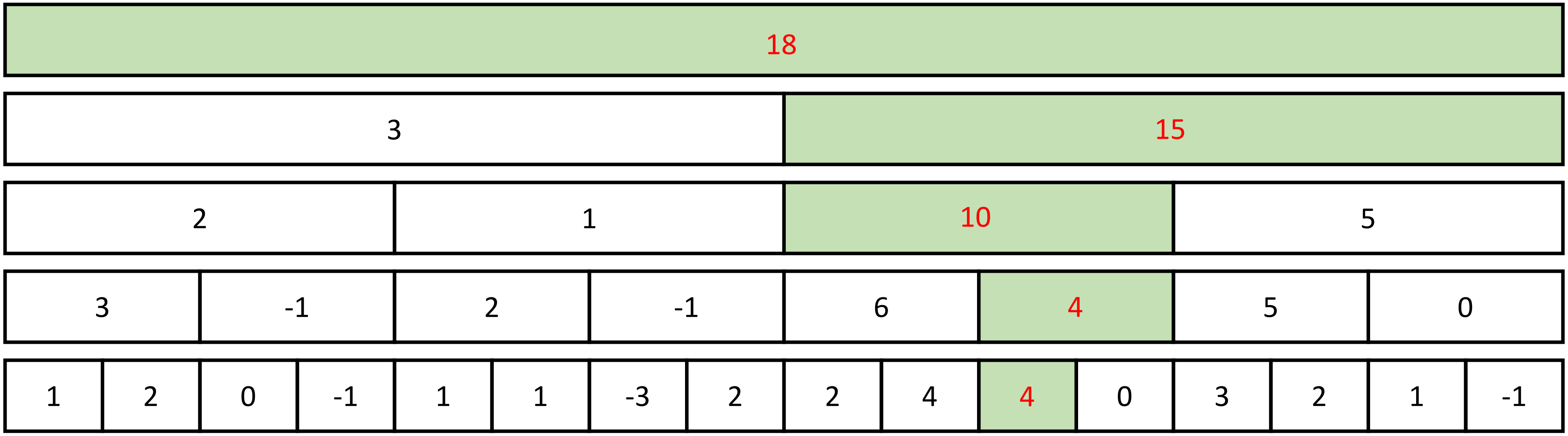
2 1 4 5

3 -1 2 -1 6 -2 5 0

1 2 0 -1 1 1 -3 2 2 4 -2 0 3 2 1 -1

1	2	0	-1	1	1	-3	2	2	4	4	0	3	2	1	-1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

1	2	0	-1	1	1	-3	2	2	4	-2	0	3	2	1	-1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



18

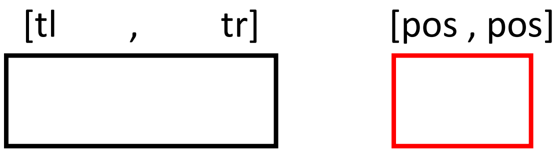
3 15

2 1 10 5

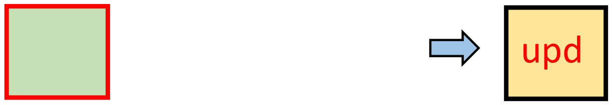
3 -1 2 -1 6 4 5 0

1 2 0 -1 1 1 -3 2 2 4 4 0 3 2 1 -1

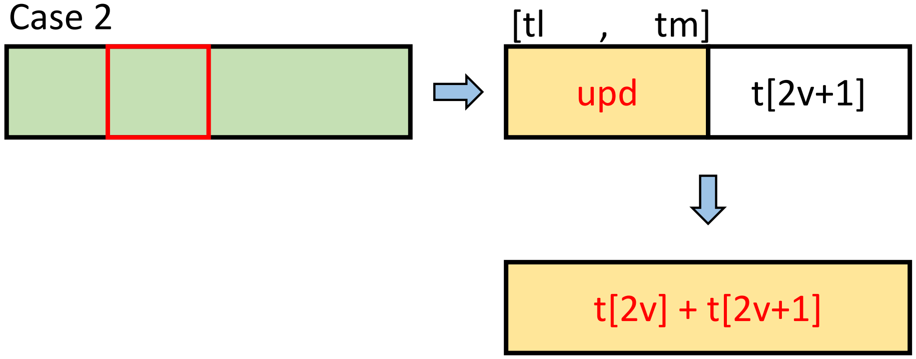
1	2	0	-1	1	1	-3	2	2	4	4	0	3	2	1	-1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

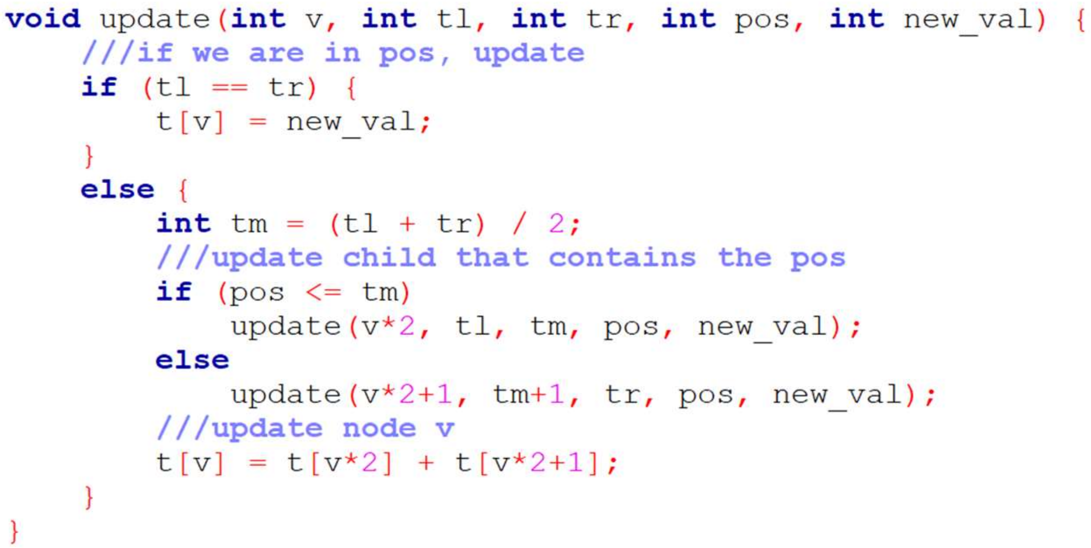


Case 1



Case 2





```
void update(int v, int tl, int tr, int pos, int new_val) {
    ///if we are in pos, update
    if (tl == tr) {
        t[v] = new_val;
    }
    else {
        int tm = (tl + tr) / 2;
        ///update child that contains the pos
        if (pos <= tm)
            update(v*2, tl, tm, pos, new_val);
        else
            update(v*2+1, tm+1, tr, pos, new_val);
        ///update node v
        t[v] = t[v*2] + t[v*2+1];
    }
}
```

¡Gracias!