Avanserte segment-trær

Lazy propagnation, multidimensjonale, persistent og dynamiske

---

Birk Tjelmeland

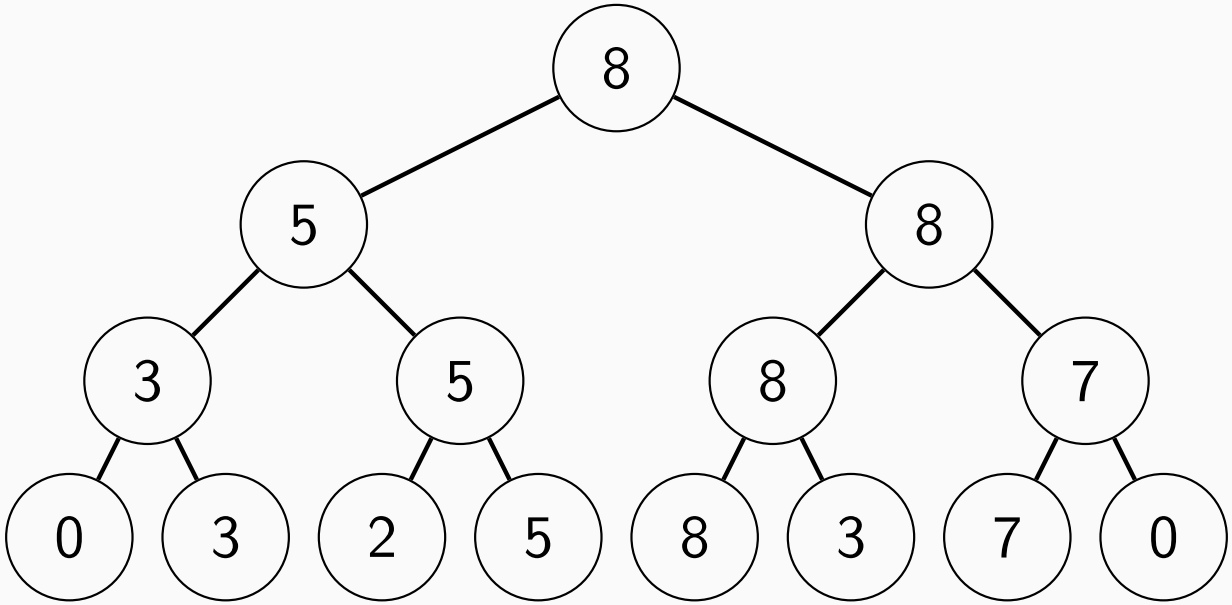
17. september 2018

Introdukjson til segment-trær

- $O(\log n)$ punkt-oppdateringer og intervall-spøringer.
- $O(n)$ konstruksjon.
- Kan brukes i mange ulike situasjoner f.eks: RSQ, minste element, største element, osv.
- Kan i teorien brukes med alle monoider (ting som har et identitets-element og en assosiativ operasjon)

Introdukjson til segment-trær

Et eksempel max-segment-tre.



0	3	2	5	8	3	7	0

Introdukjson til segment-trær

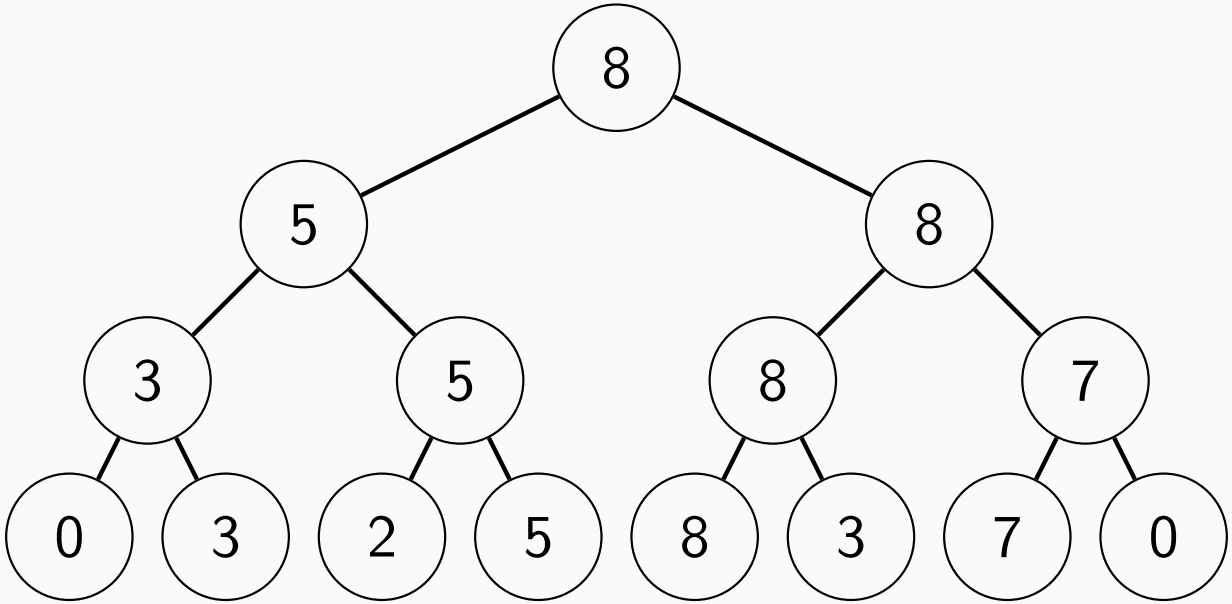
Implentasjon av et enkelt generisk segment-tre.

```
template<class T, class O>
class SegmentTree {
    int N, offset;
    vector<T> v;
public:
    SegmentTree(vector<T> initial) : N(1) {
        while (N < initial.size()+2) N*=2;
        offset = N+1; N*=2;
        v = vector<T>(N, O::e);
        for (int i = 0; i < initial.size(); i++)
            v[i+offset] = initial[i];

        for (int i = offset-1; i > 0; i--)
            v[i] = O{}(v[i*2], v[i*2+1]);
    }
};
```

Introdukjson til segment-trær

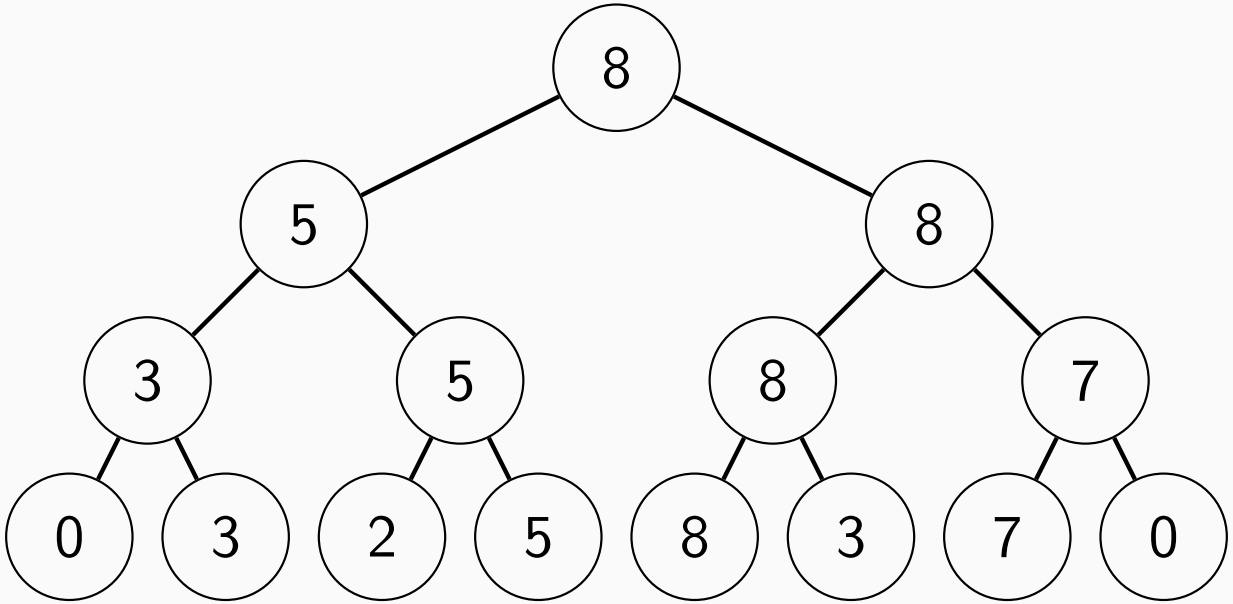
En eksempel spøring på intervalet 1-4.



0	3	2	5	8	3	7	0

Introdukjson til segment-trær

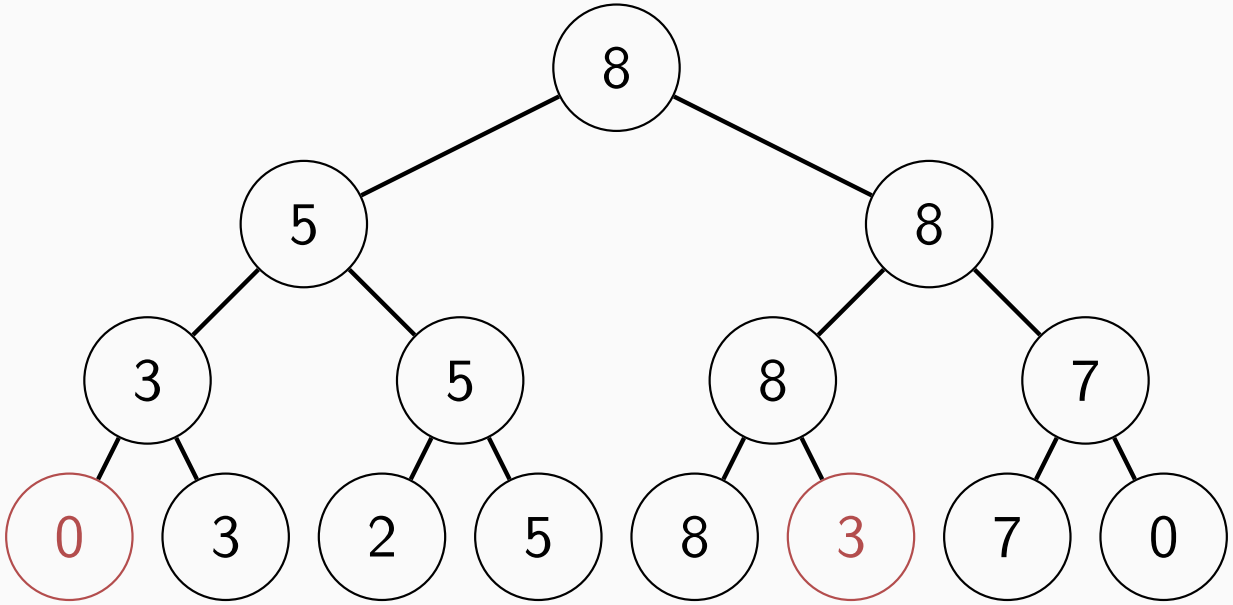
En eksempel spøring på intervalet 1-4.



0	3	2	5	8	3	7	0

Introdukjson til segment-trær

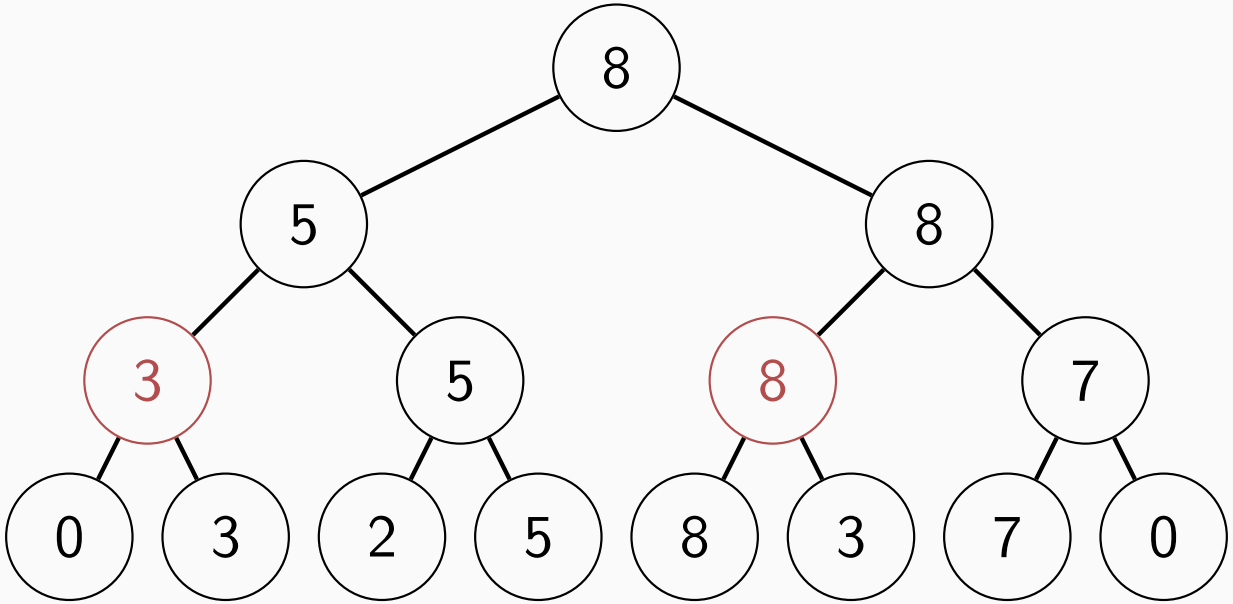
En eksempel spøring på intervalet 1-4.



0	3	2	5	8	3	7	0

Introdukjson til segment-trær

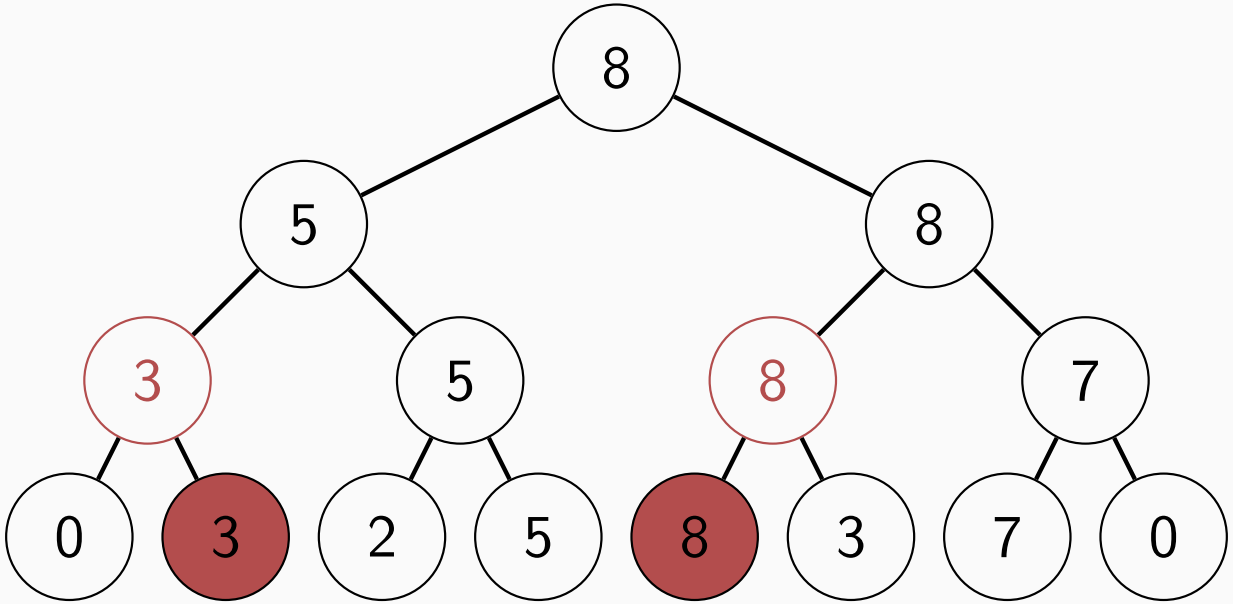
En eksempel spøring på intervalet 1-4.



0	3	2	5	8	3	7	0

Introdukjson til segment-trær

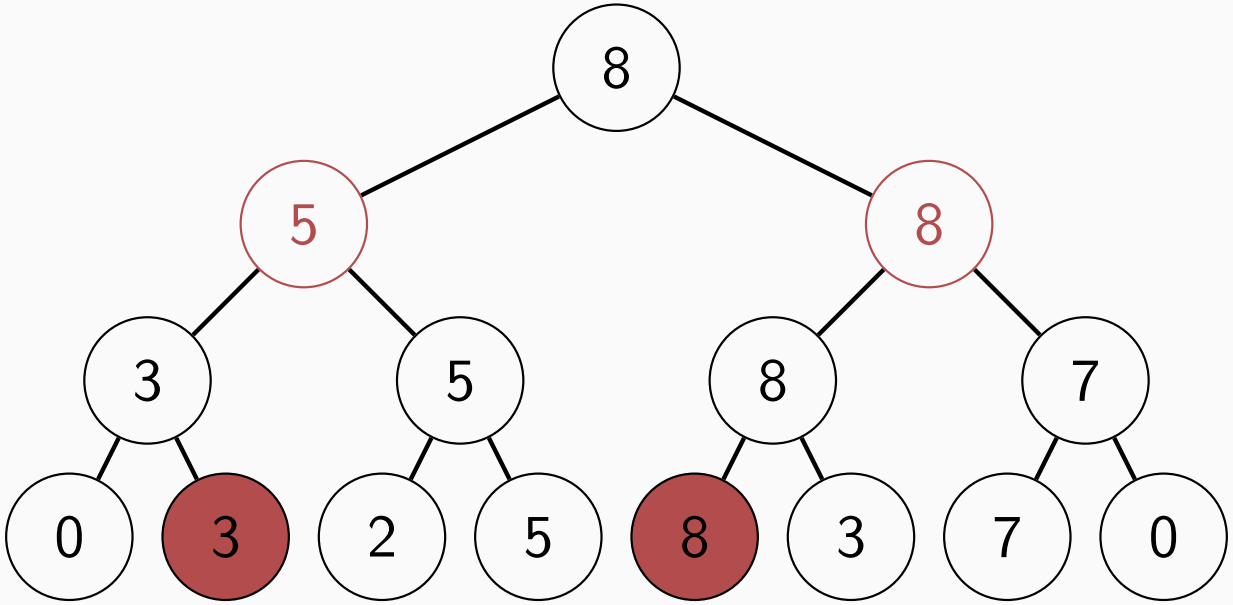
En eksempel spøring på intervalet 1-4.



0	3	2	5	8	3	7	0

Introdukjson til segment-trær

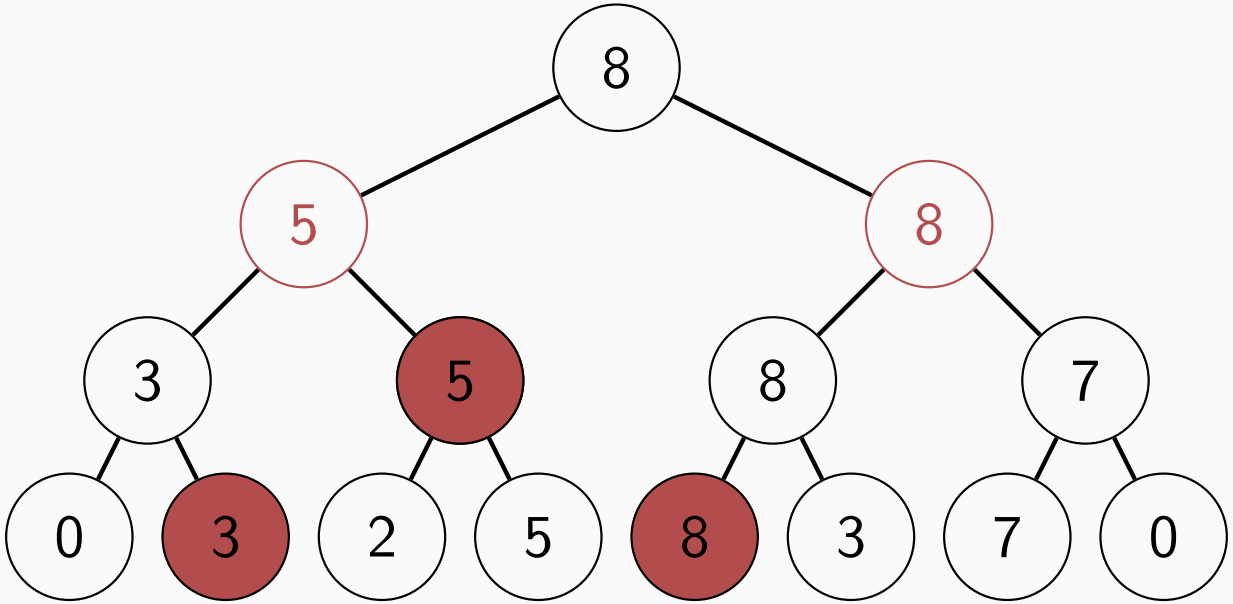
En eksempel spøring på intervalet 1-4.



0	3	2	5	8	3	7	0

Introdukjson til segment-trær

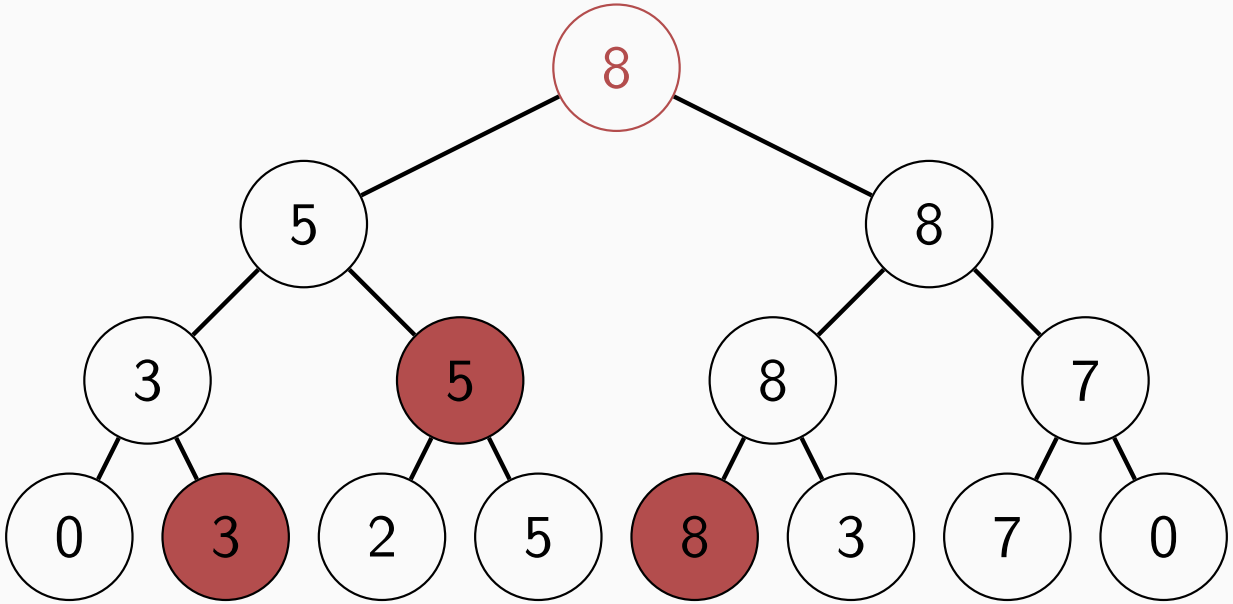
En eksempel spøring på intervalet 1-4.



0	3	2	5	8	3	7	0

Introdukjson til segment-trær

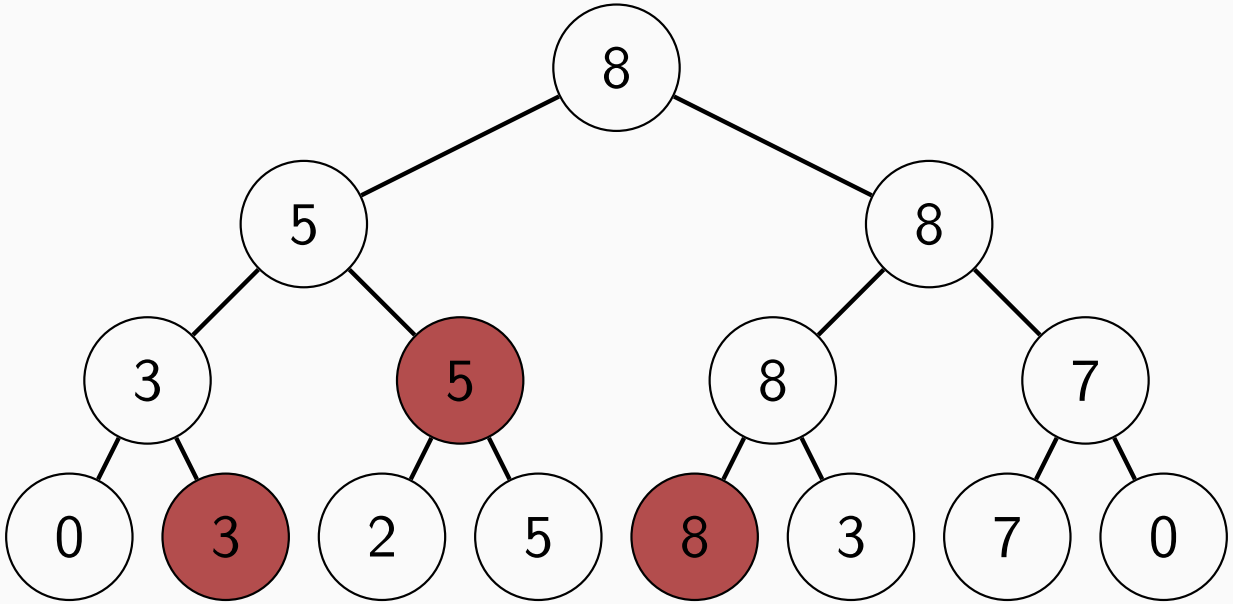
En eksempel spøring på intervalet 1-4.



0	3	2	5	8	3	7	0

Introdukjson til segment-trær

En eksempel spøring på intervalet 1-4.



0	3	2	5	8	3	7	0

Svaret blir $\max(3, 5, 8) = 8$

Introdukjson til segment-trær

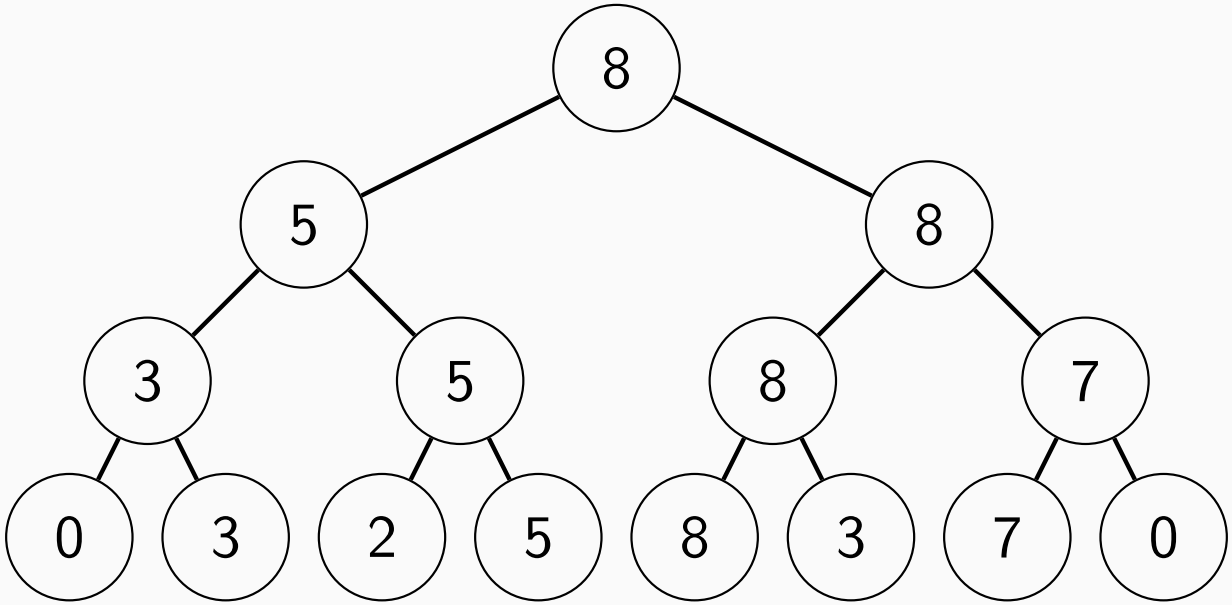
Implentasjon av spøring av et segment-tre.

```
T get(int l, int r) {
    l += offset - 1;
    r += offset + 1;
    T lv = O::e;
    T rv = O::e;
    while (l / 2 != r / 2) {
        if (l % 2 == 0) lv = O{}(lv, v[l+1]);
        if (r % 2 == 1) rv = O{}(v[r-1], rv);
        l /= 2;
        r /= 2;
    }

    return O{}(lv, rv);
}
```

Introdukjson til segment-trær

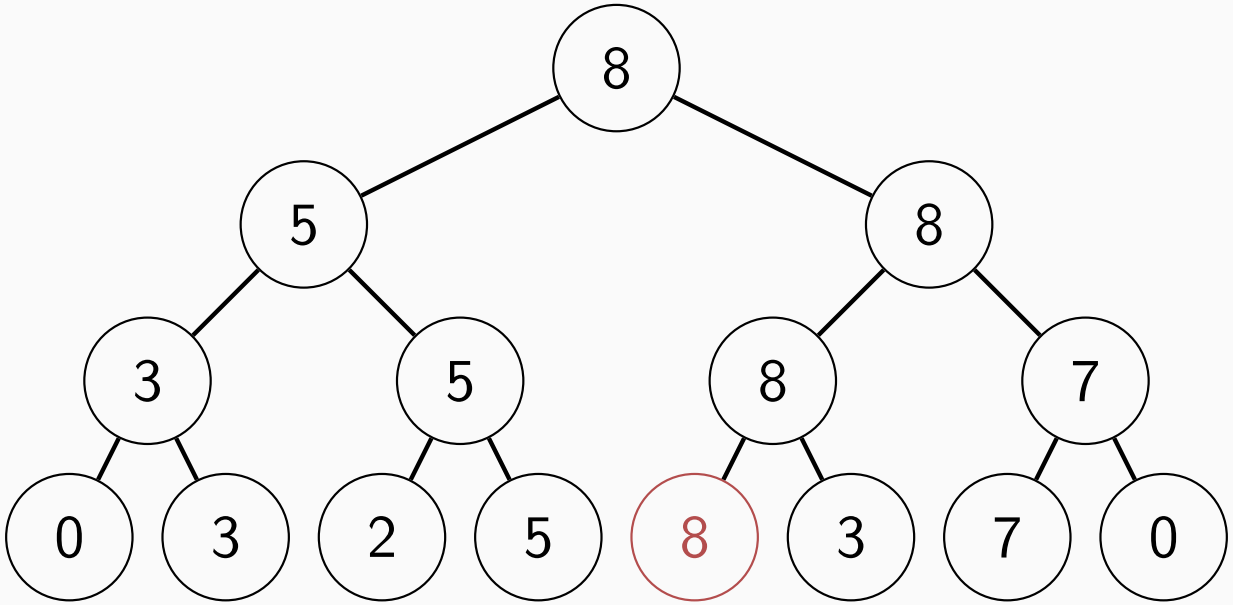
En eksempel oppdatering på indeks 4 fra 8 til 2.



0	3	2	5	8	3	7	0

Introdukjson til segment-trær

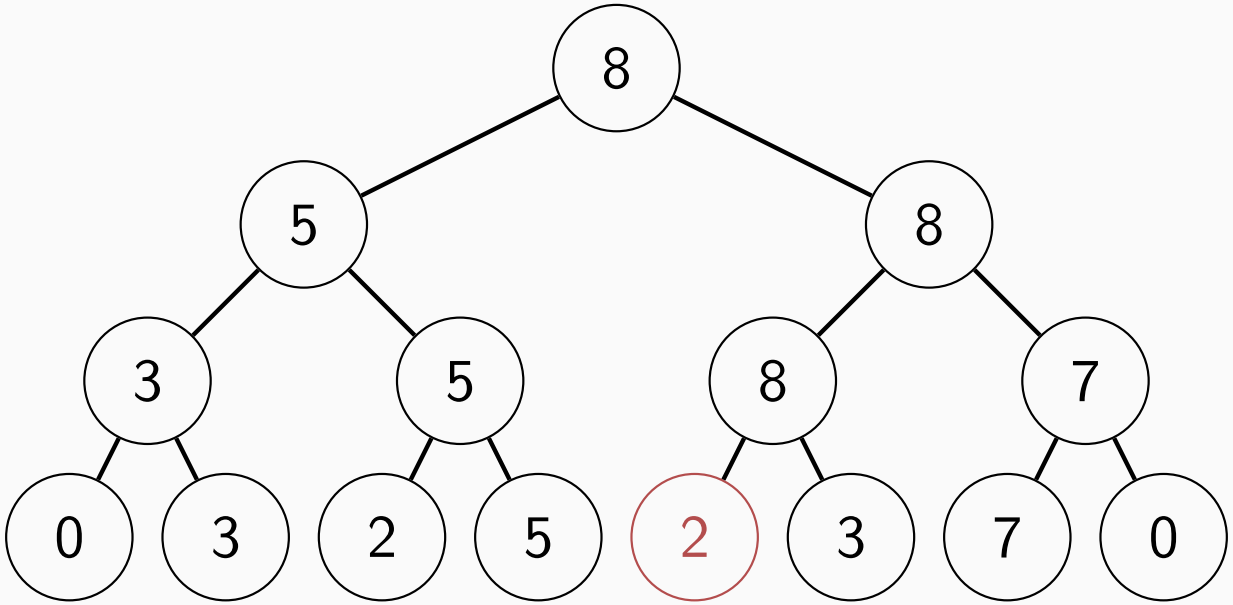
En eksempel oppdatering på indeks 4 fra 8 til 2.



0	3	2	5	8	3	7	0

Introdukjson til segment-trær

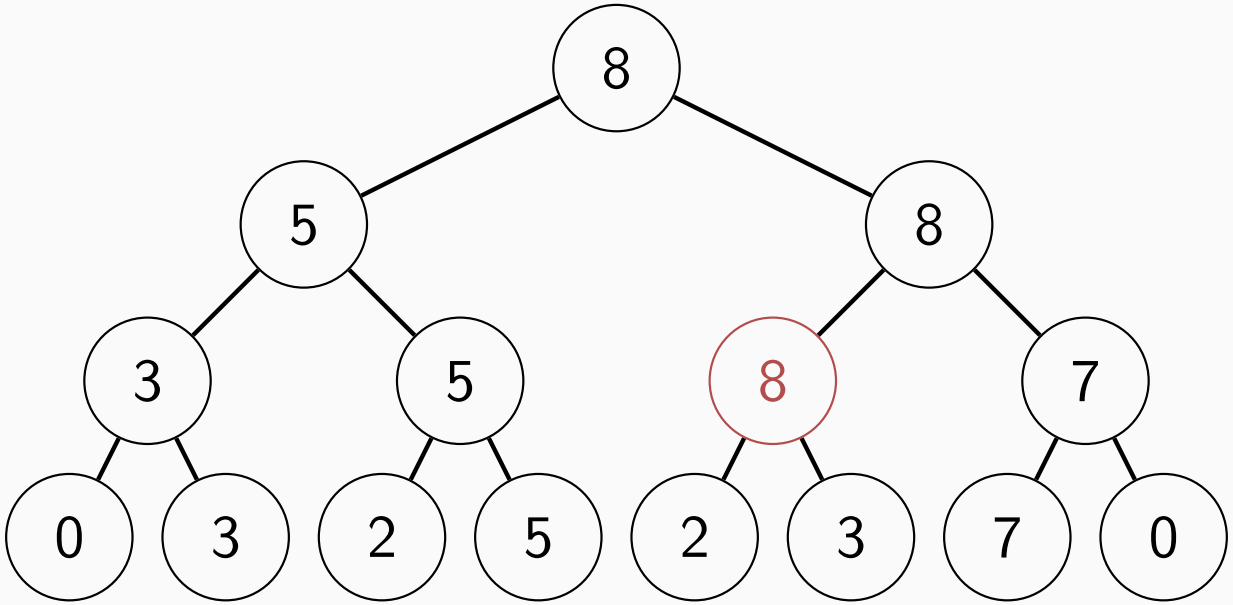
En eksempel oppdatering på indeks 4 fra 8 til 2.



0	3	2	5	2	3	7	0

Introdukjson til segment-trær

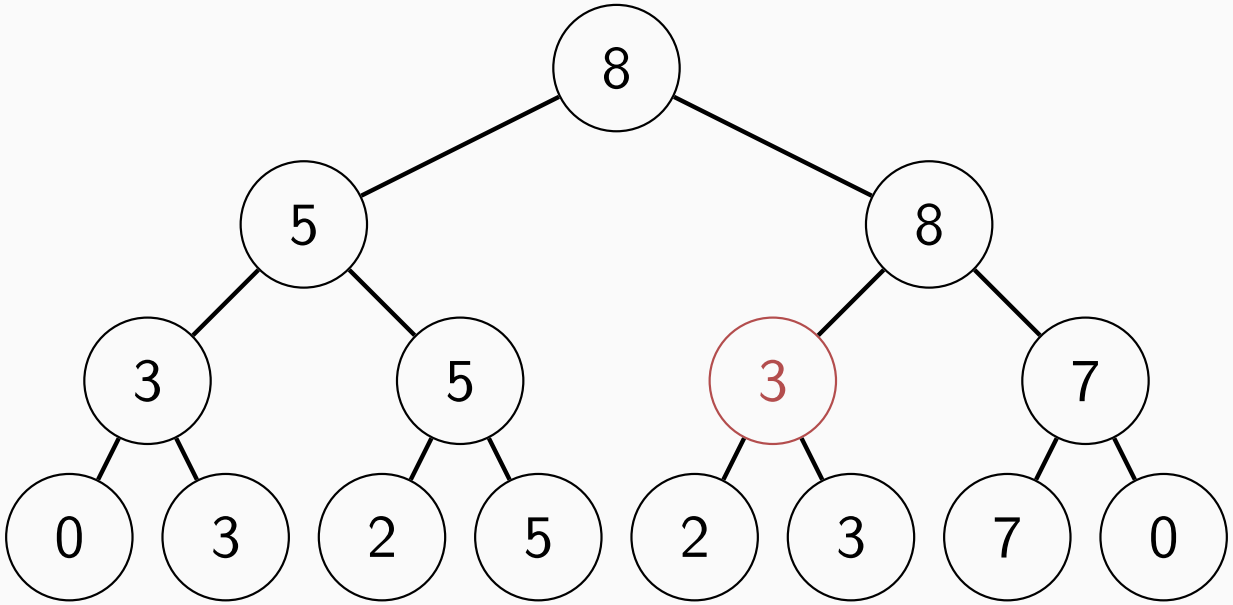
En eksempel oppdatering på indeks 4 fra 8 til 2.



0	3	2	5	2	3	7	0

Introdukjson til segment-trær

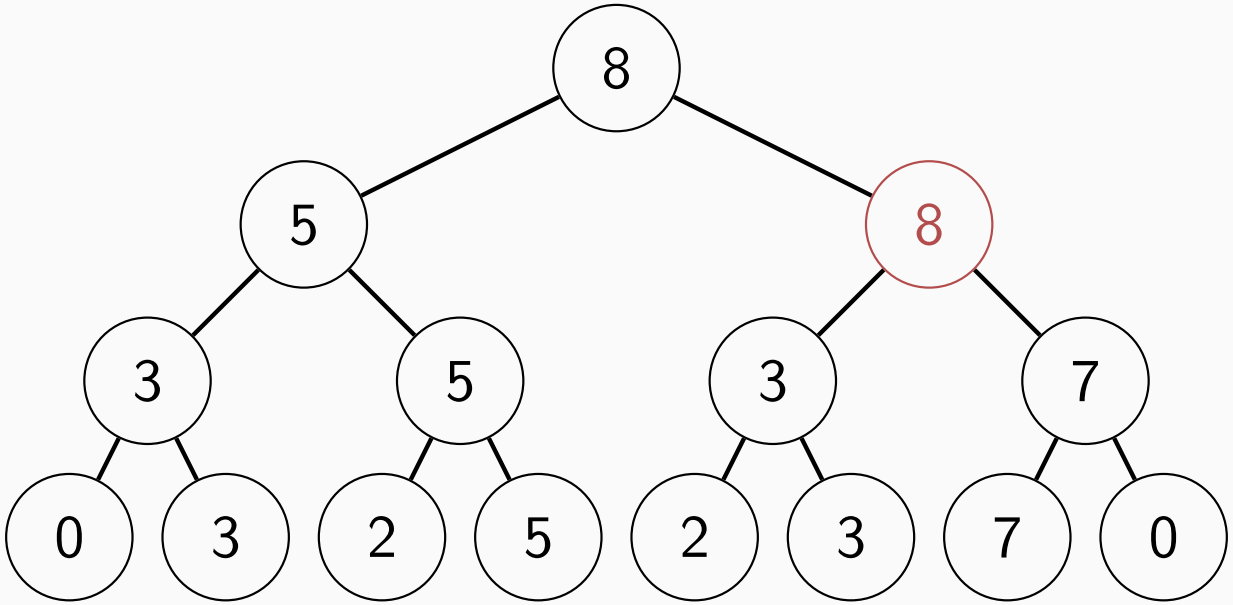
En eksempel oppdatering på indeks 4 fra 8 til 2.



0	3	2	5	2	3	7	0

Introdukjson til segment-trær

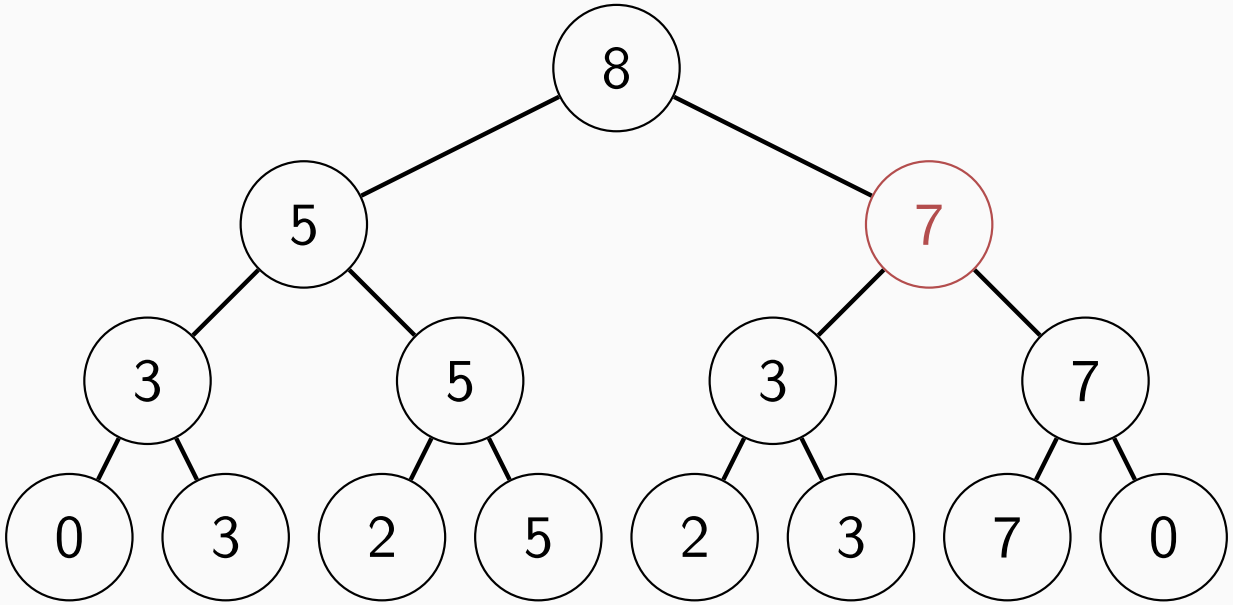
En eksempel oppdatering på indeks 4 fra 8 til 2.



0	3	2	5	2	3	7	0

Introdukjson til segment-trær

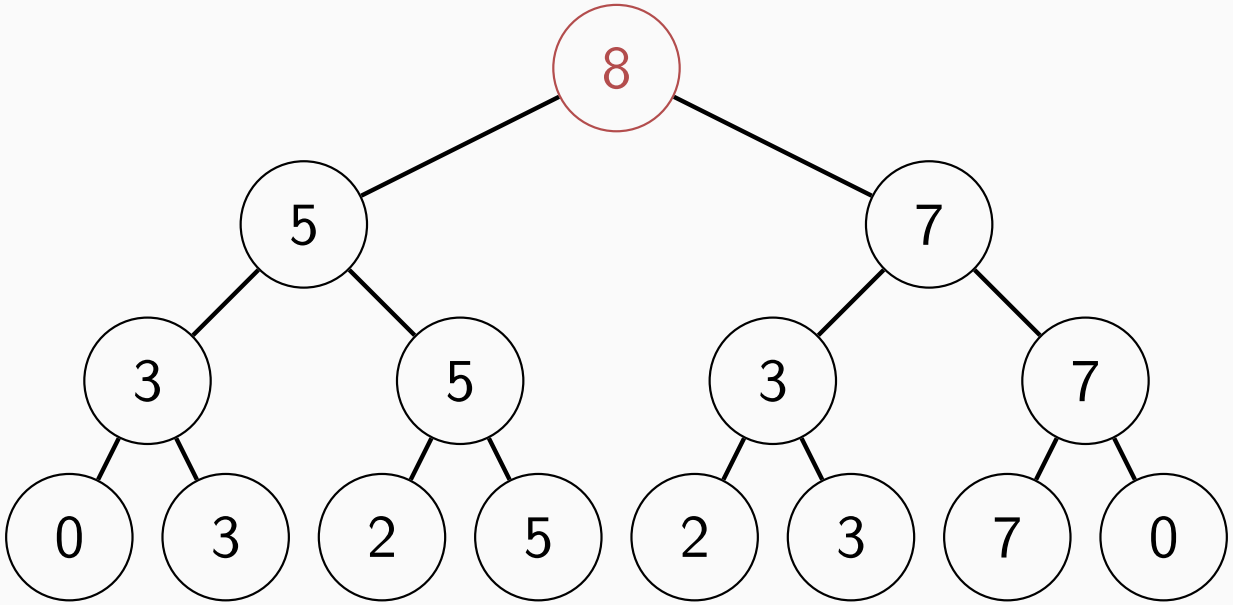
En eksempel oppdatering på indeks 4 fra 8 til 2.



0	3	2	5	2	3	7	0

Introdukjson til segment-trær

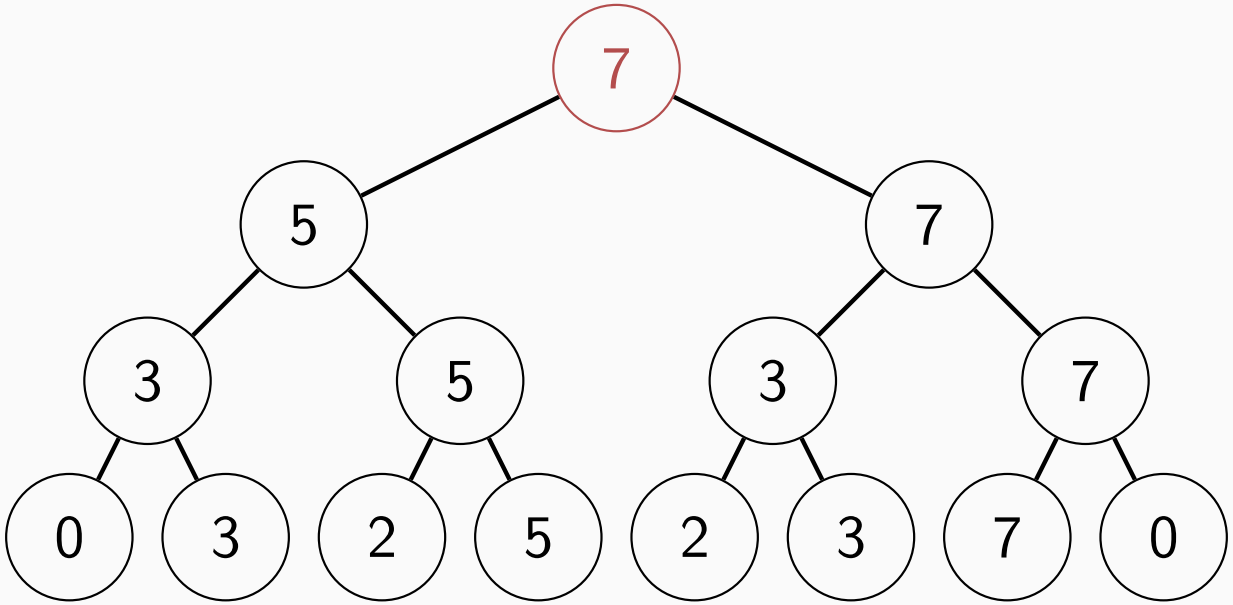
En eksempel oppdatering på indeks 4 fra 8 til 2.



0	3	2	5	2	3	7	0

Introdukjson til segment-trær

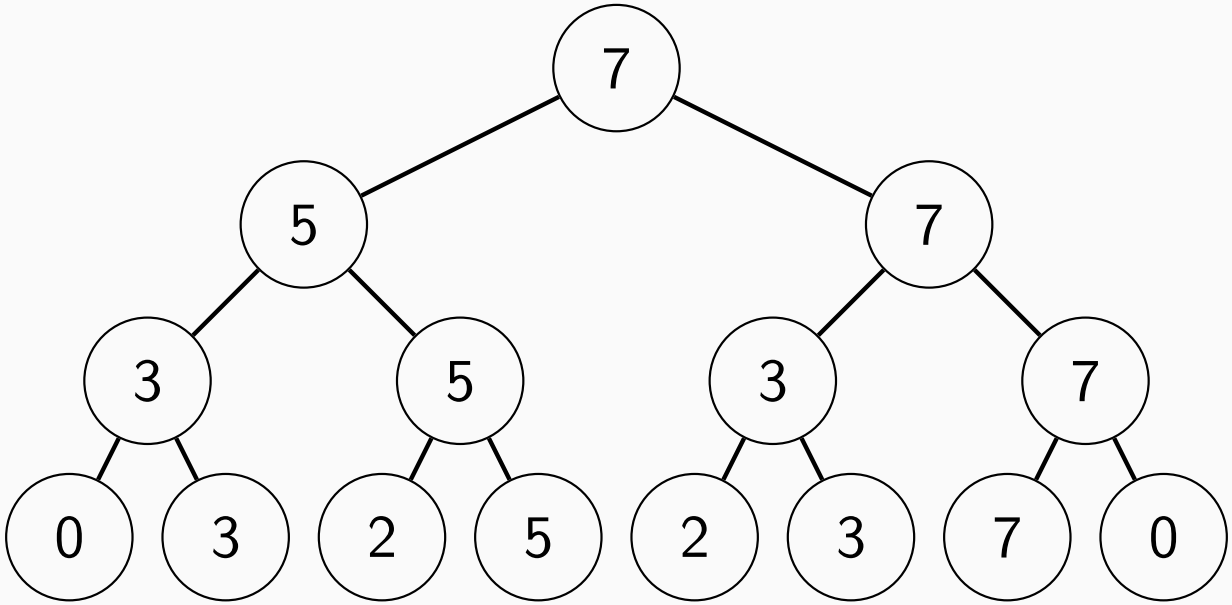
En eksempel oppdatering på indeks 4 fra 8 til 2.



0	3	2	5	2	3	7	0

Introdukjson til segment-trær

En eksempel oppdatering på indeks 4 fra 8 til 2.



0	3	2	5	2	3	7	0

Introdukjson til segment-trær

Implentasjon av oppdatering av et segment-tre.

```
void update(int i, T val) {
    v[i] = val;
    i /= 2;
    while (i > 0) {
        v[i] = O{}(v[i*2], v[i*2+1]);
        i /= 2;
    }
}
```

Introdukjson til segment-trær

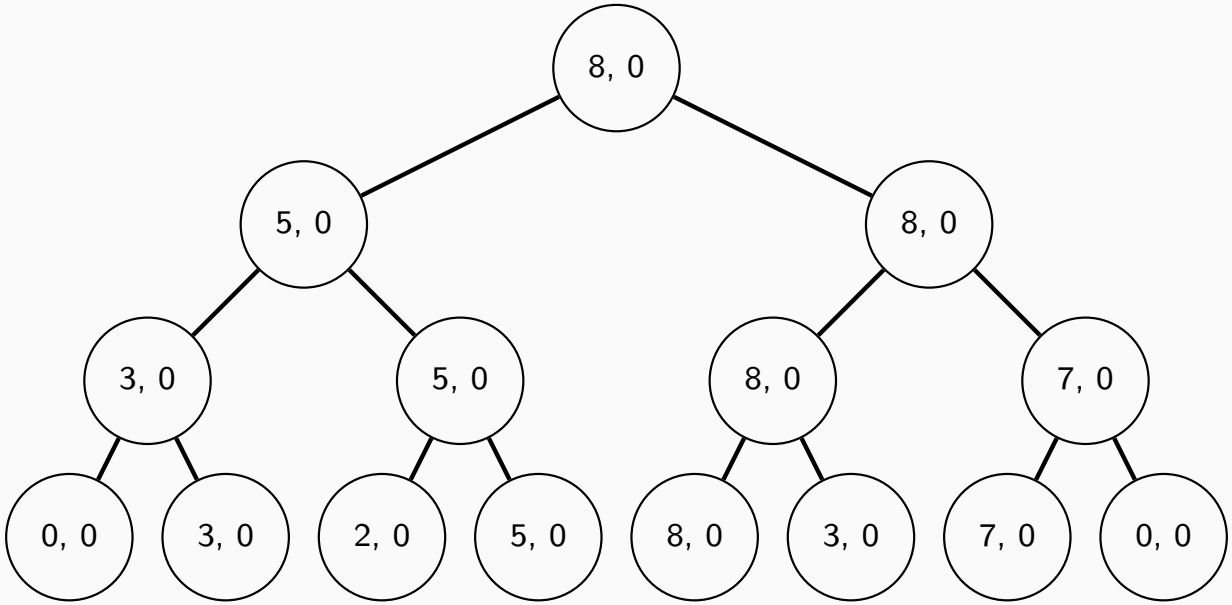
- An annen variant er $O(\log n)$ punkt-spøringer og intervall oppdateringer.
- Fungerer med nesten samme prinsipet, bare at `update` og `get` er byttet om på.

Lazy propagnation

- Lazy propagnation tillater $O(\log n)$ intervall-spøringer og intervall-oppdateringer.
- Men det kan være mer krevende å få til rett og det er større begrensninger i hva slags type elementer som kan være med.
- Et eksempel er `max` spøringer og + oppdateringer.
- I motsetning til vanlige segment-trær må alle operasjoner kjøres top-ned for at de skal fungere.
- Trikset er å alltid sende lazy veriden til en node nedover når du besøker den.

Lazy propagnation

Et eksempel max-pluss-segment-tre.



0	3	2	5	8	3	7	0

Lazy propagnation

Implentasjon av et generisk lazy-propagnation segment-tre.

```
template<class T, class L, class O>
class LazySegmentTree {
    int N, offset;
    vector<T> vals; vector<L> lazy;
    vector<int> lis, ris;
public:
    LazySegmentTree(vector<T> initial) : N(1) {
        while (N < initial.size()+2) N*=2;
        offset = N+1; N*=2;
        vals = vector<T>(N, O::e);
        lazy = vector<L>(N);
        ris = lis = vector<int>(N);
        for (int i = 0; i < initial.size(); i++) {
            vals[i+offset] = initial[i];
            lis[i+offset] = ris[i+offset] = i;
        }
        for (int i = offset-1; i > 0; i--)  {
            vals[i] = O{}(vals[i*2], vals[i*2+1]);
            lis[i] = min(lis[i*2], lis[i*2+1]);
            ris[i] = max(ris[i*2], max[i*2+1]);
        }
    }
};
```

Lazy propagnation

Implentasjon av et generisk lazy-propagnation segment-tre.

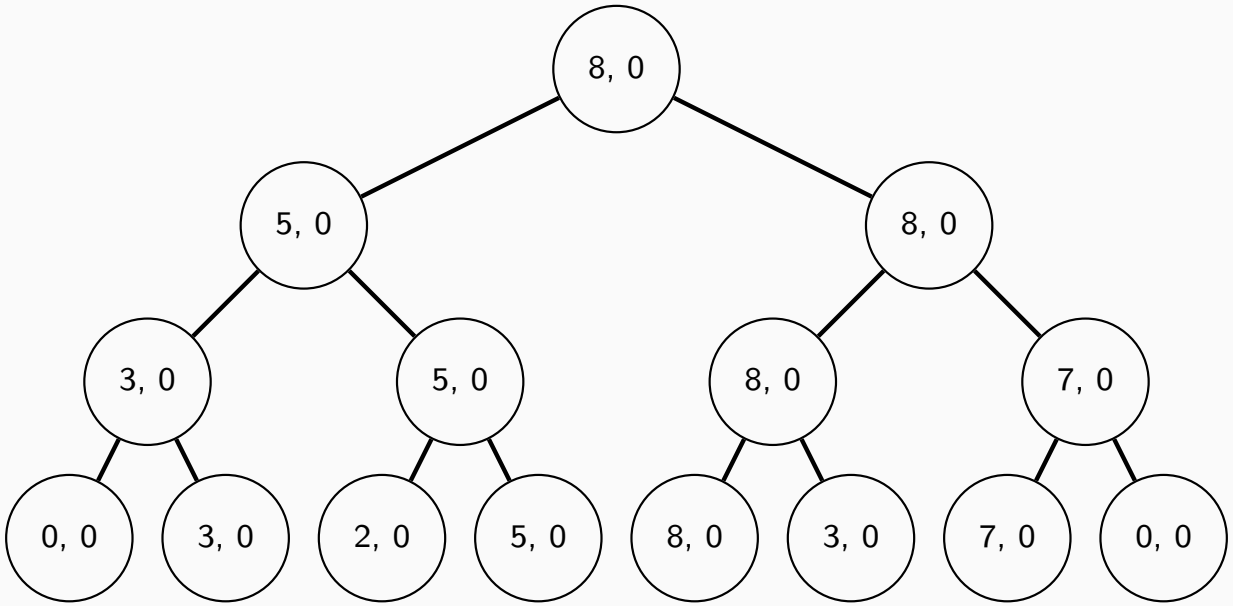
```
void propagnate_lazy(int i) {
    vals[i] = lazy[i].apply(vals[i]);
    if (i*2 <= N) lazy[i*2].append(lazy[i]);
    if (i*2+1 <= N) lazy[i*2+1].append(lazy[i]);
    lazy[i] = L();
}

T get_value(int i) {
    propagnate_lazy(i);
    return vals[i];
}
```

Lazy propagnation

Et eksempel på oppdatering av et lazy-propagnation segment-tre.

Legge til 3 til alle tallene fra index 1 til 5.

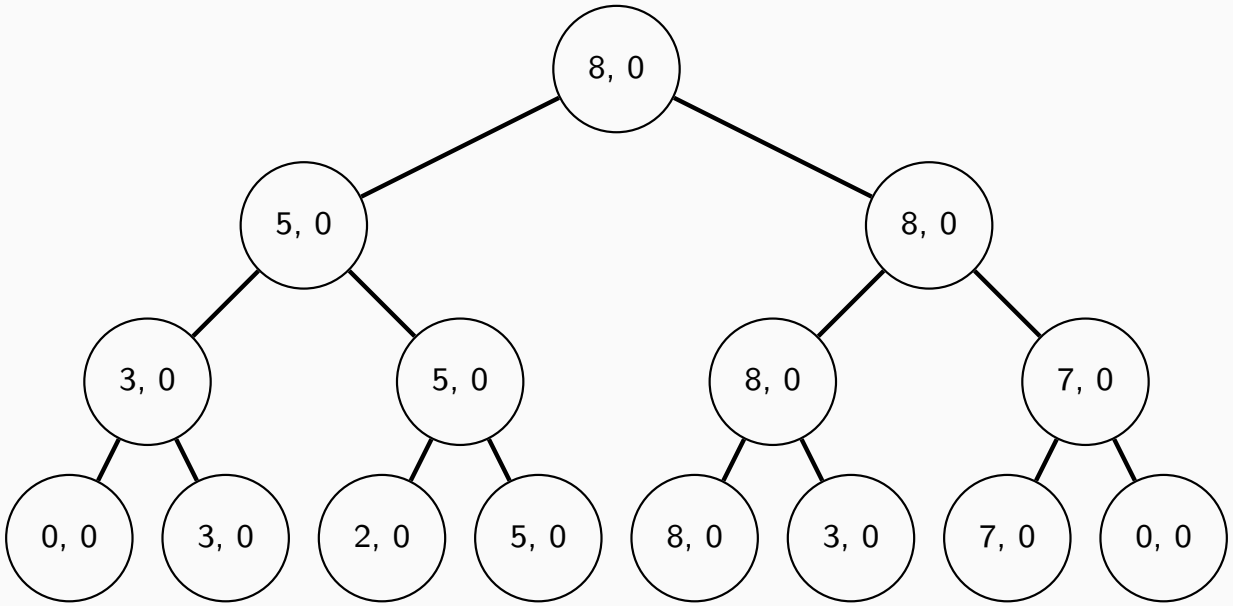


0	3	2	5	8	3	7	0

Lazy propagnation

Et eksempel på oppdatering av et lazy-propagnation segment-tre.

Legge til 3 til alle tallene fra index 1 til 5.

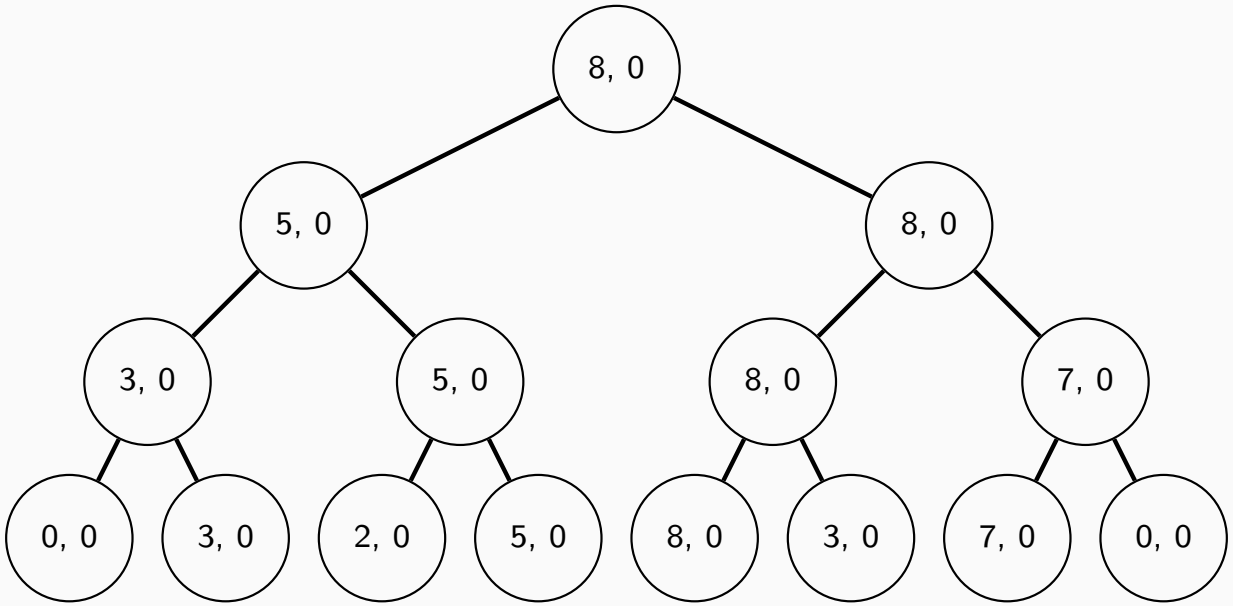


0	3	2	5	8	3	7	0

Lazy propagnation

Et eksempel på oppdatering av et lazy-propagnation segment-tre.

Legge til 3 til alle tallene fra index 1 til 5.

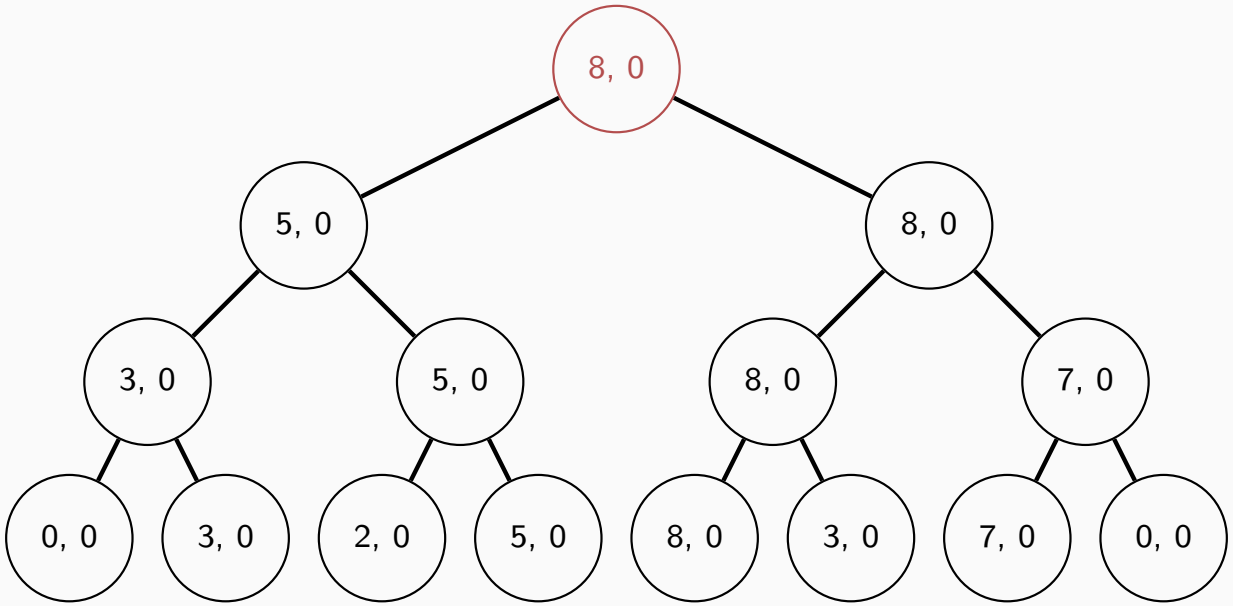


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på oppdatering av et lazy-propagnation segment-tre.

Legge til 3 til alle tallene fra index 1 til 5.

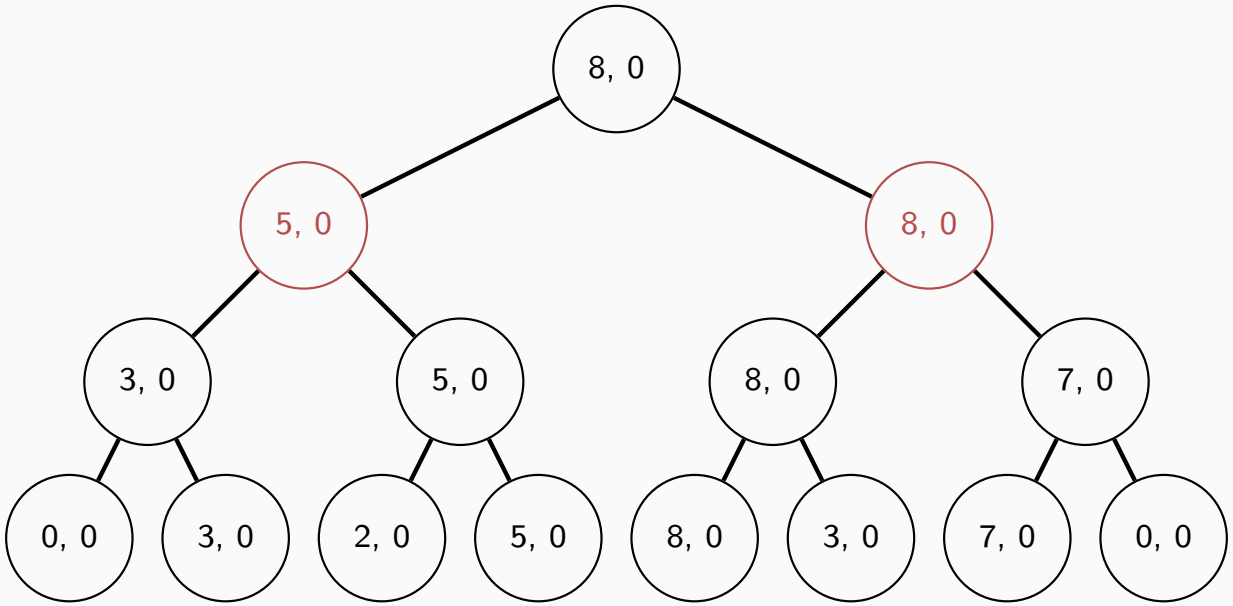


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på oppdatering av et lazy-propagnation segment-tre.

Legge til 3 til alle tallene fra index 1 til 5.

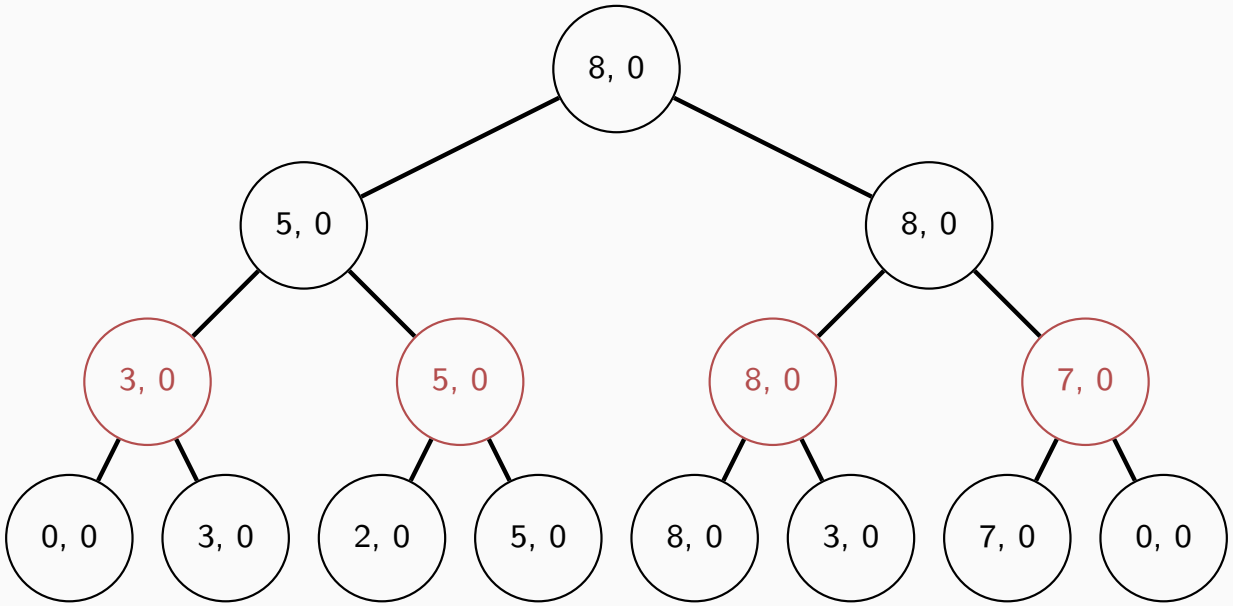


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på oppdatering av et lazy-propagnation segment-tre.

Legge til 3 til alle tallene fra index 1 til 5.

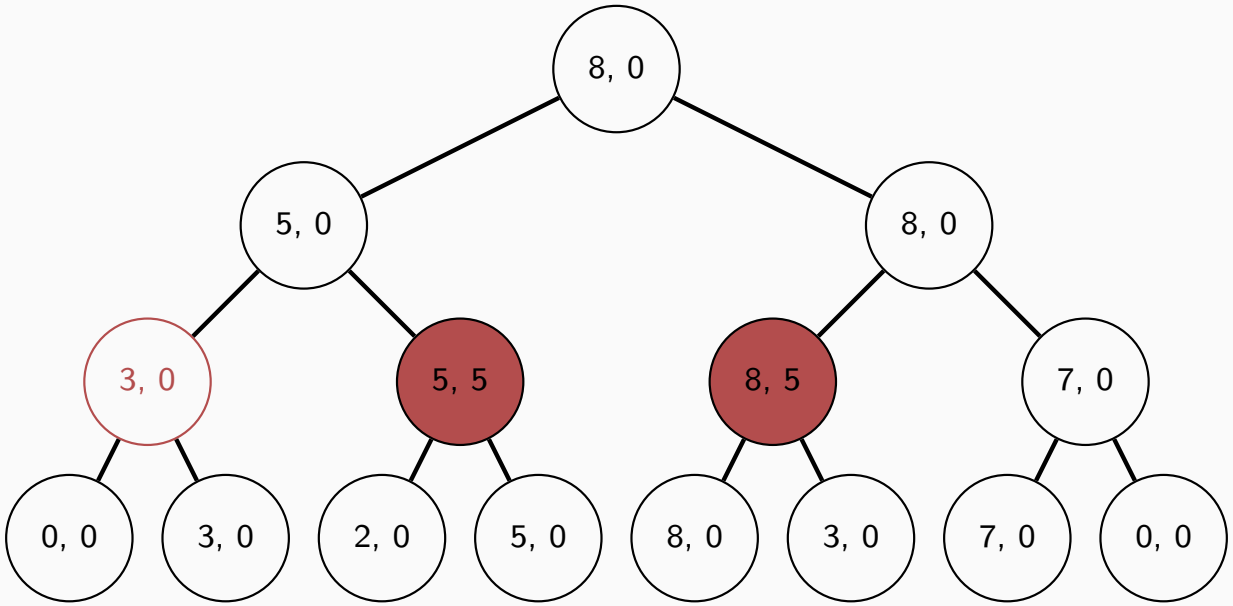


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på oppdatering av et lazy-propagnation segment-tre.

Legge til 3 til alle tallene fra index 1 til 5.

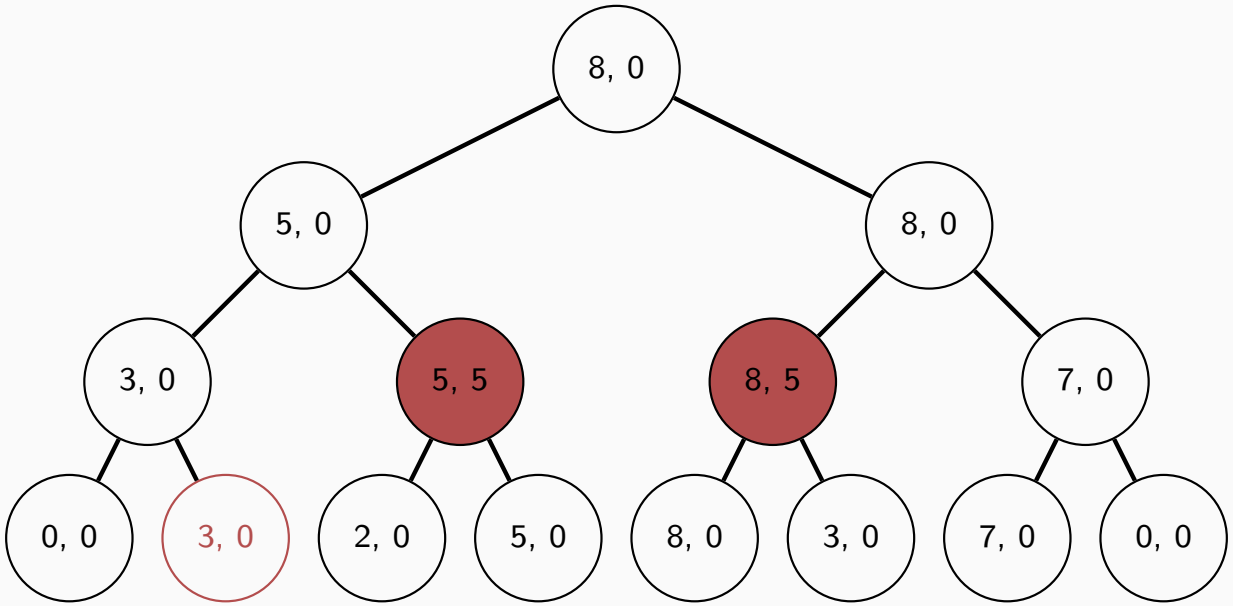


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på oppdatering av et lazy-propagnation segment-tre.

Legge til 3 til alle tallene fra index 1 til 5.

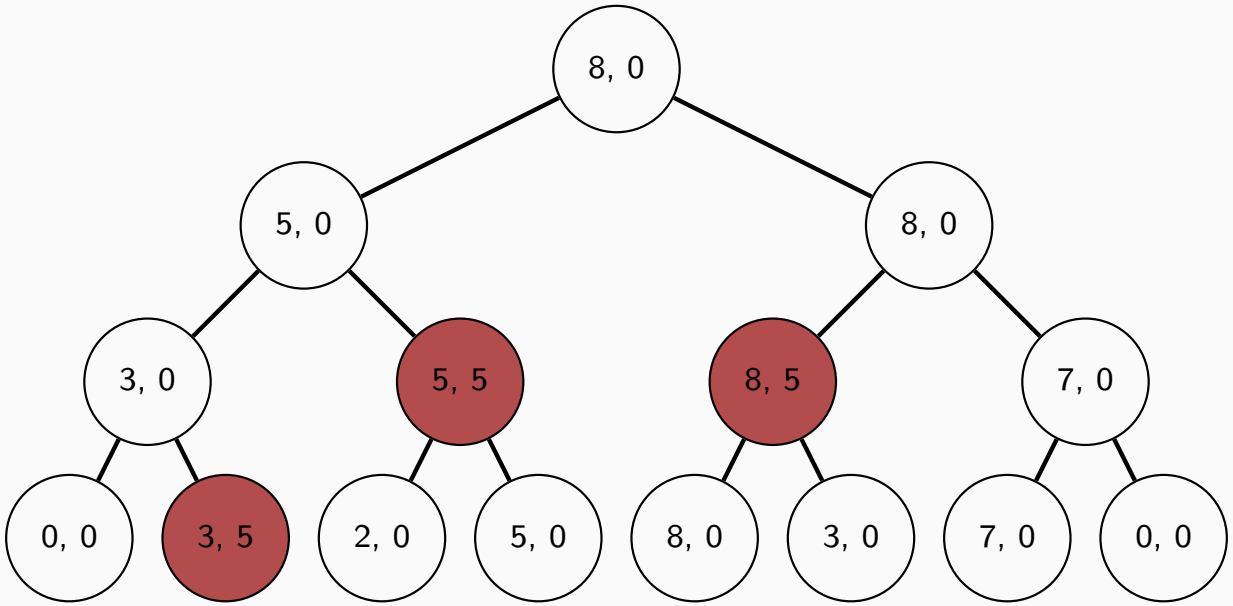


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på oppdatering av et lazy-propagnation segment-tre.

Legge til 3 til alle tallene fra index 1 til 5.

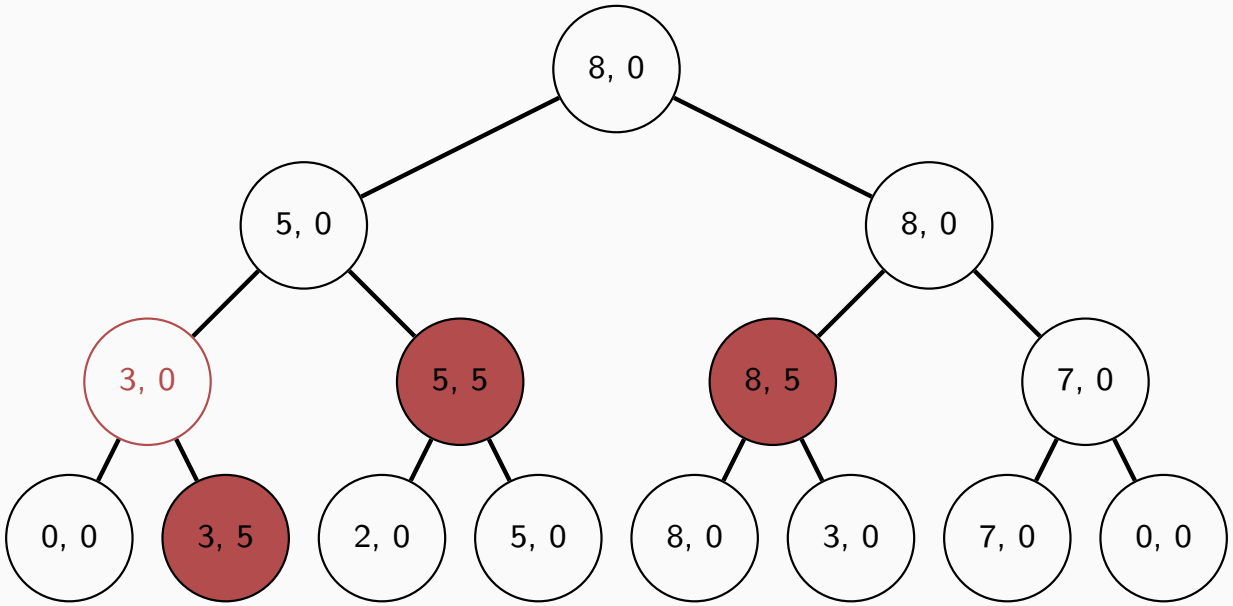


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på oppdatering av et lazy-propagnation segment-tre.

Legge til 3 til alle tallene fra index 1 til 5.

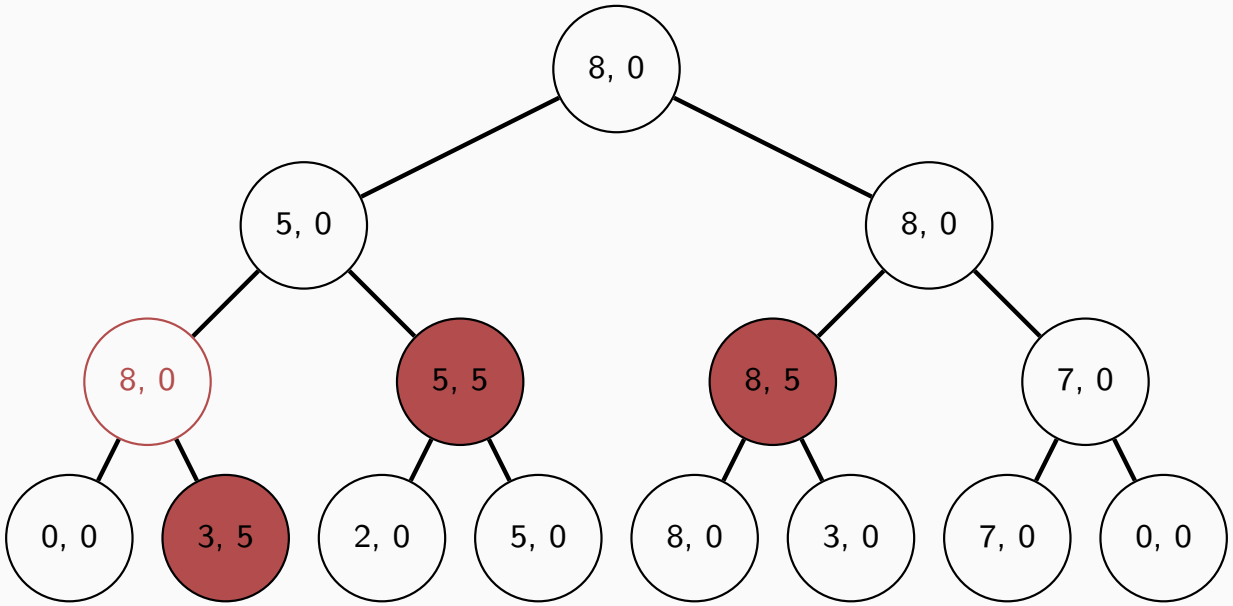


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på oppdatering av et lazy-propagnation segment-tre.

Legge til 3 til alle tallene fra index 1 til 5.

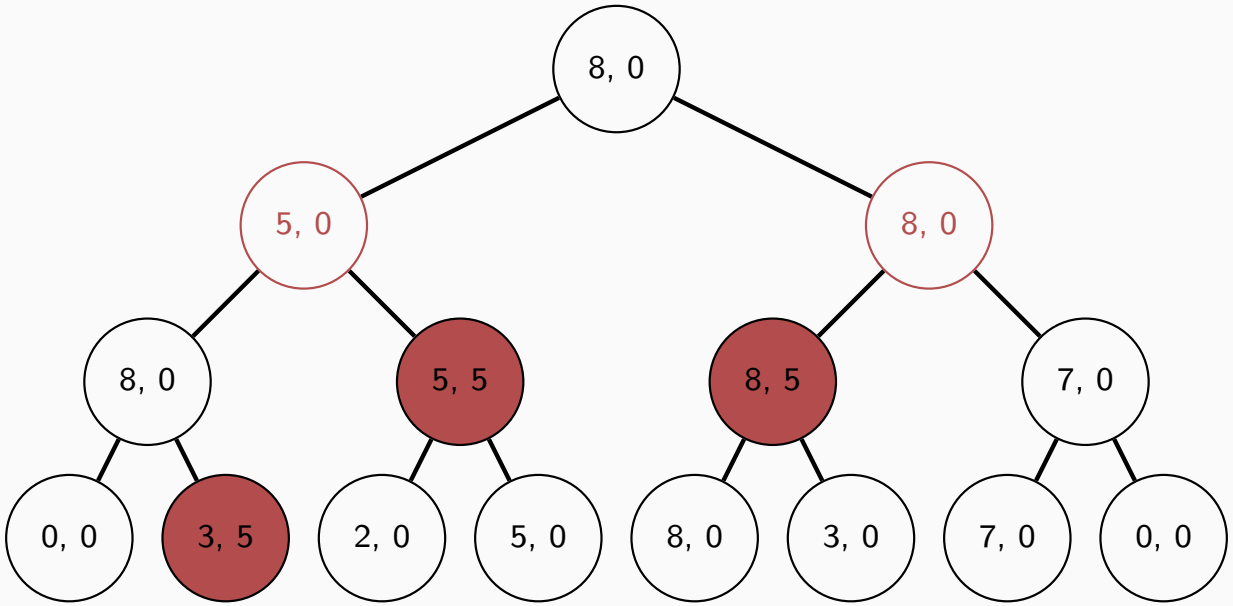


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på oppdatering av et lazy-propagnation segment-tre.

Legge til 3 til alle tallene fra index 1 til 5.

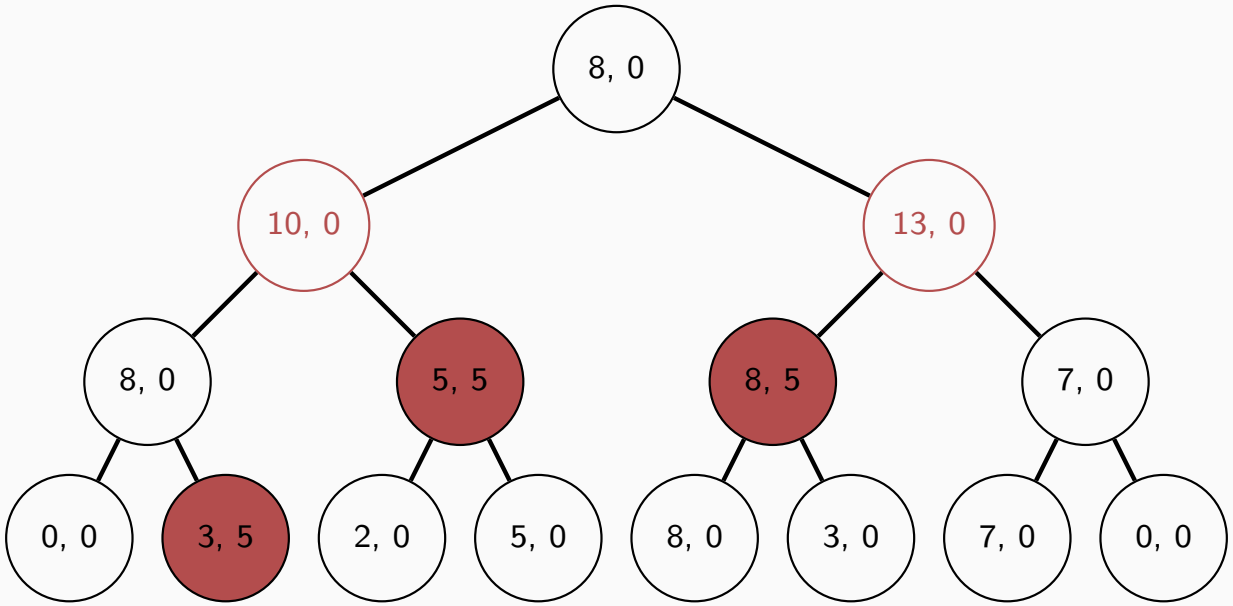


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på oppdatering av et lazy-propagnation segment-tre.

Legge til 3 til alle tallene fra index 1 til 5.

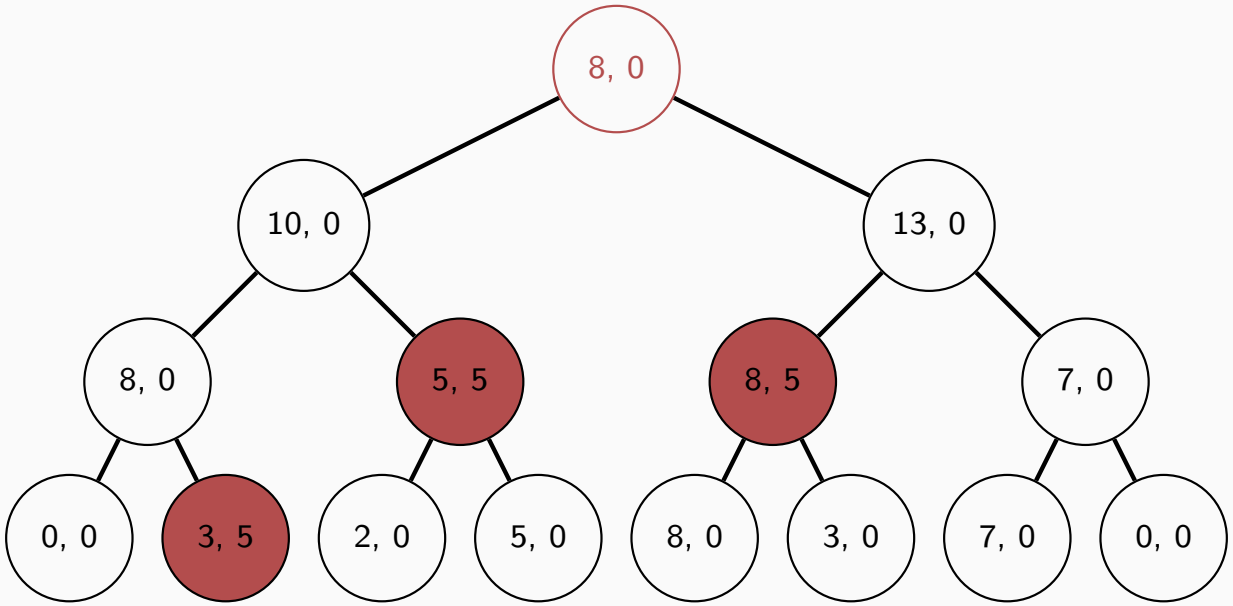


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på oppdatering av et lazy-propagnation segment-tre.

Legge til 3 til alle tallene fra index 1 til 5.

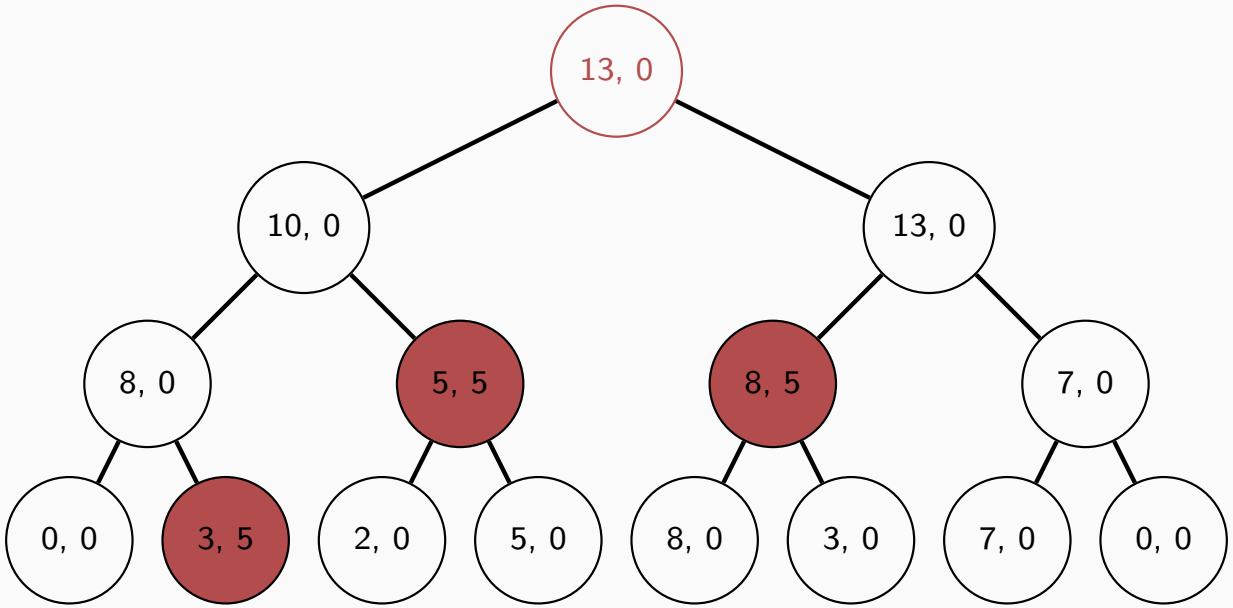


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på oppdatering av et lazy-propagnation segment-tre.

Legge til 3 til alle tallene fra index 1 til 5.

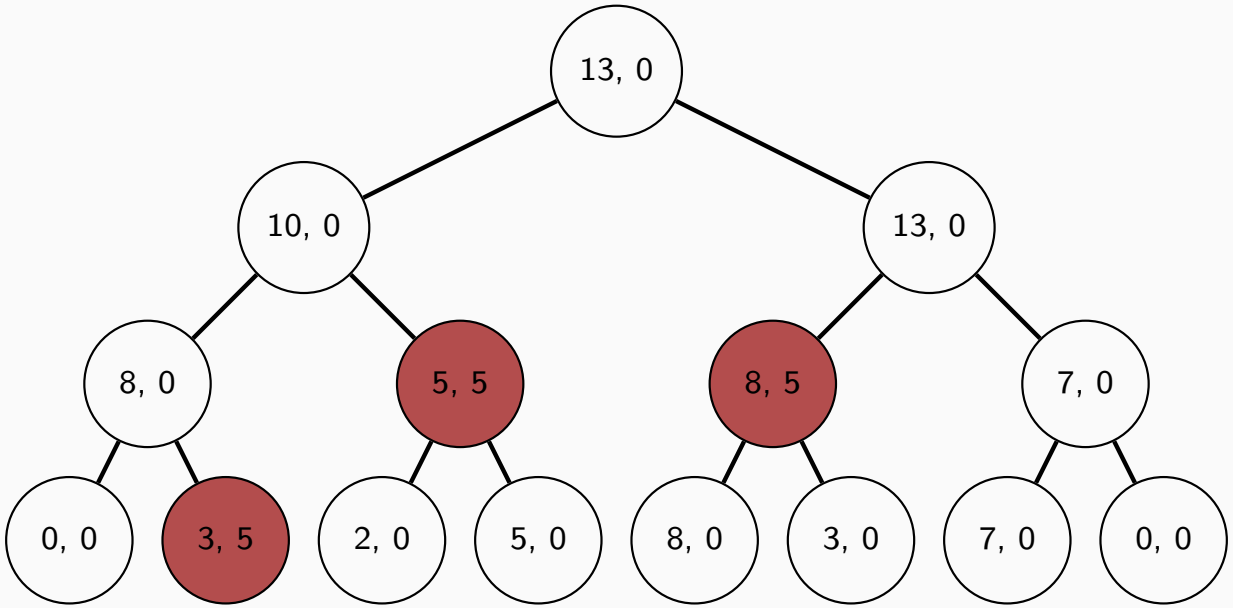


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på oppdatering av et lazy-propagnation segment-tre.

Legge til 3 til alle tallene fra index 1 til 5.



0	8	7	10	13	8	7	0

Lazy propagnation

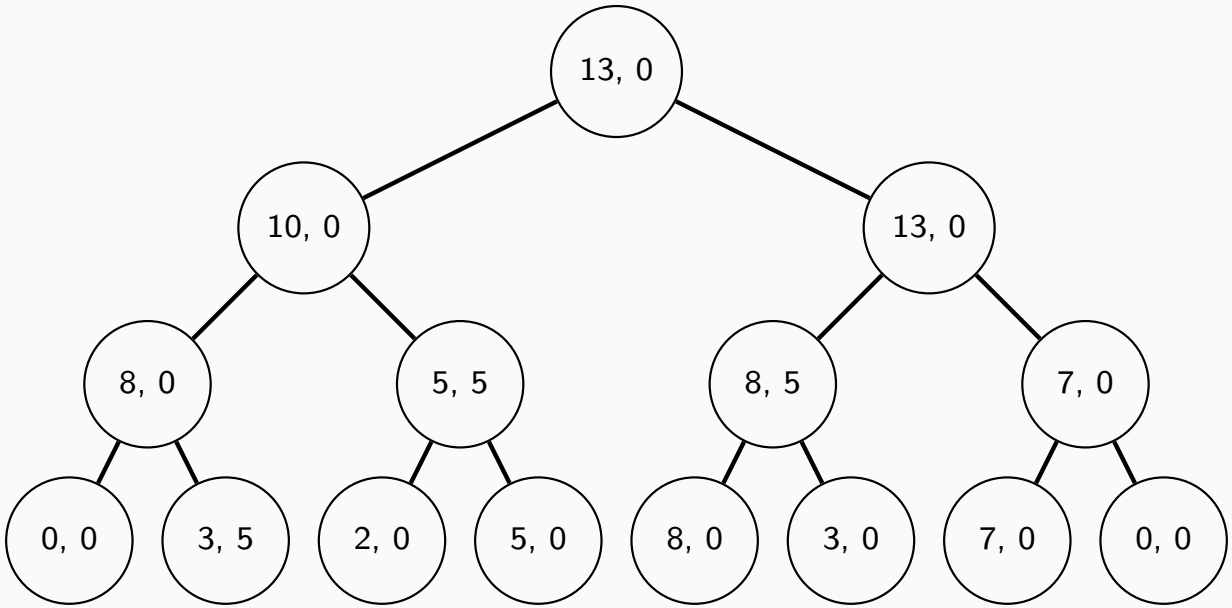
Implmentasjon av lazy-propagnation oppdatering.

```
void update(int l, int r, L u) {
    update(l, r, u, 1);
}
void update(int l, int r, L u, int i) {
    if (i >= N || lis[i] > r || ris[i] < l) return;
    propagnate_lazy(i);
    if (lis[i] <= l && ris[i] >= r) {
        lazy[i].append(u);
        return;
    }
    update(l, r, u, i*2);
    update(l, r, u, i*2+1);
    vals[i] = O{}(get_value(i*2), get_value(i*2+1));
}
```

Lazy propagnation

Et eksempel på spøringer av et lazy-propagnation segment-tre.

Spørre om max fra indeks 3 til 6.

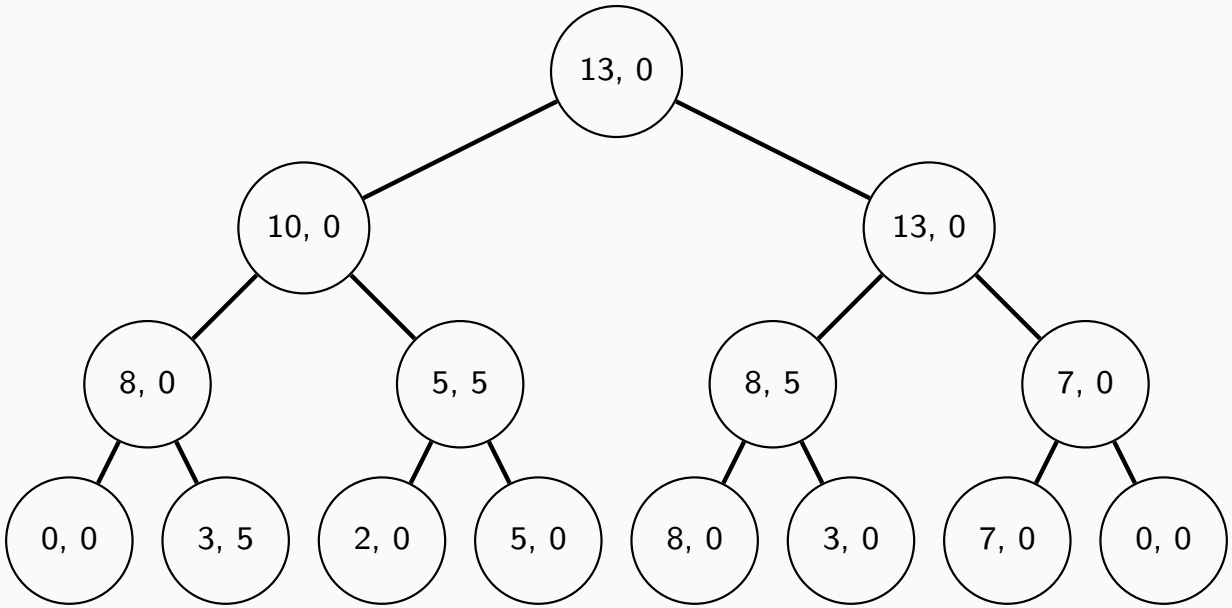


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på spøringer av et lazy-propagnation segment-tre.

Spørre om max fra indeks 3 til 6.

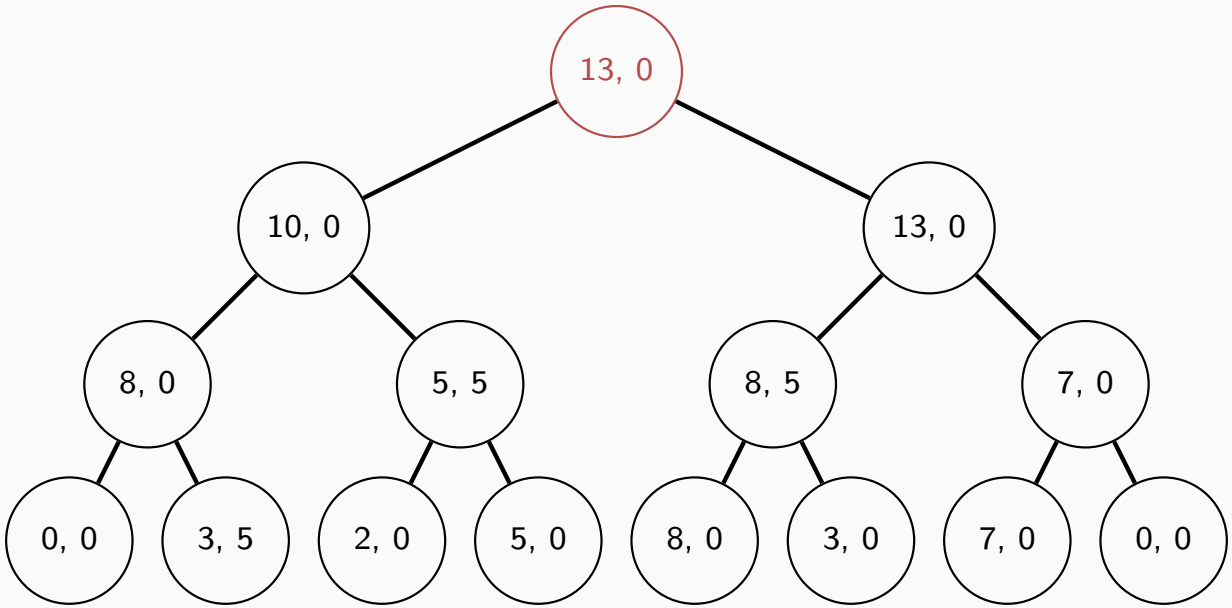


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på spøringer av et lazy-propagnation segment-tre.

Spørre om max fra indeks 3 til 6.

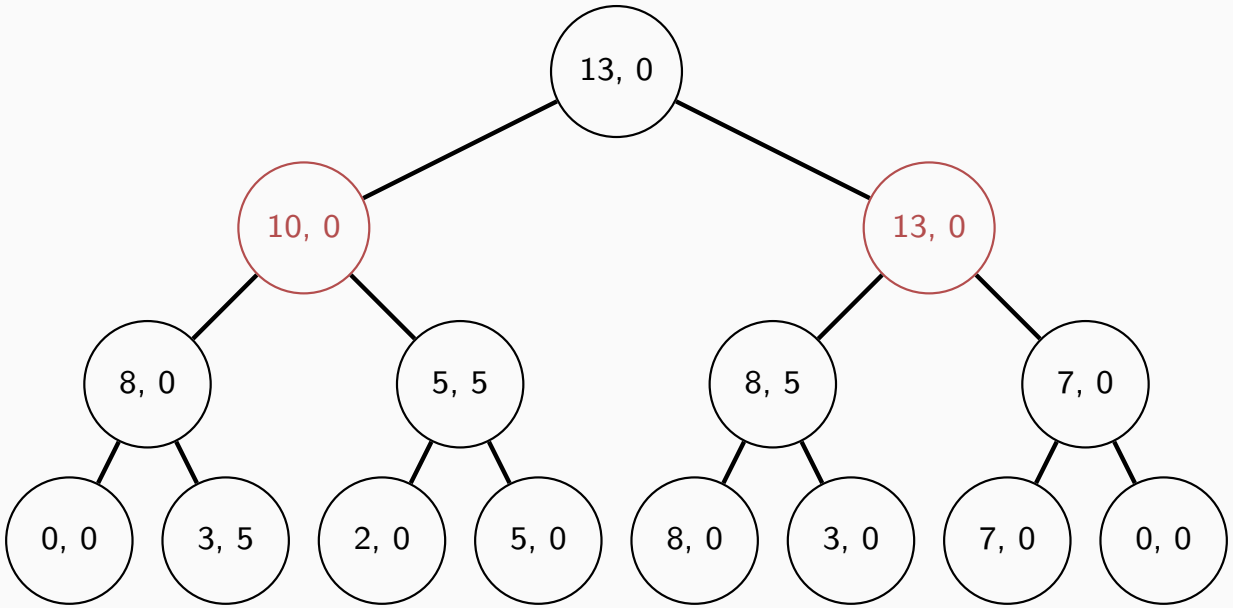


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på spøringer av et lazy-propagnation segment-tre.

Spørre om max fra indeks 3 til 6.

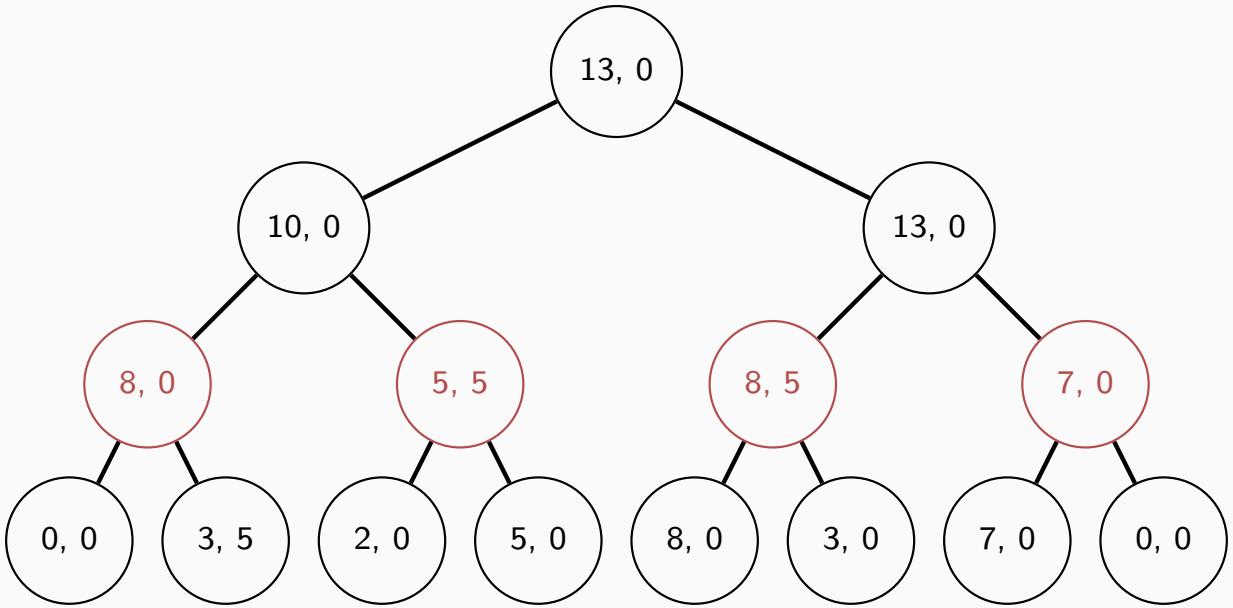


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på spøringer av et lazy-propagnation segment-tre.

Spørre om max fra indeks 3 til 6.

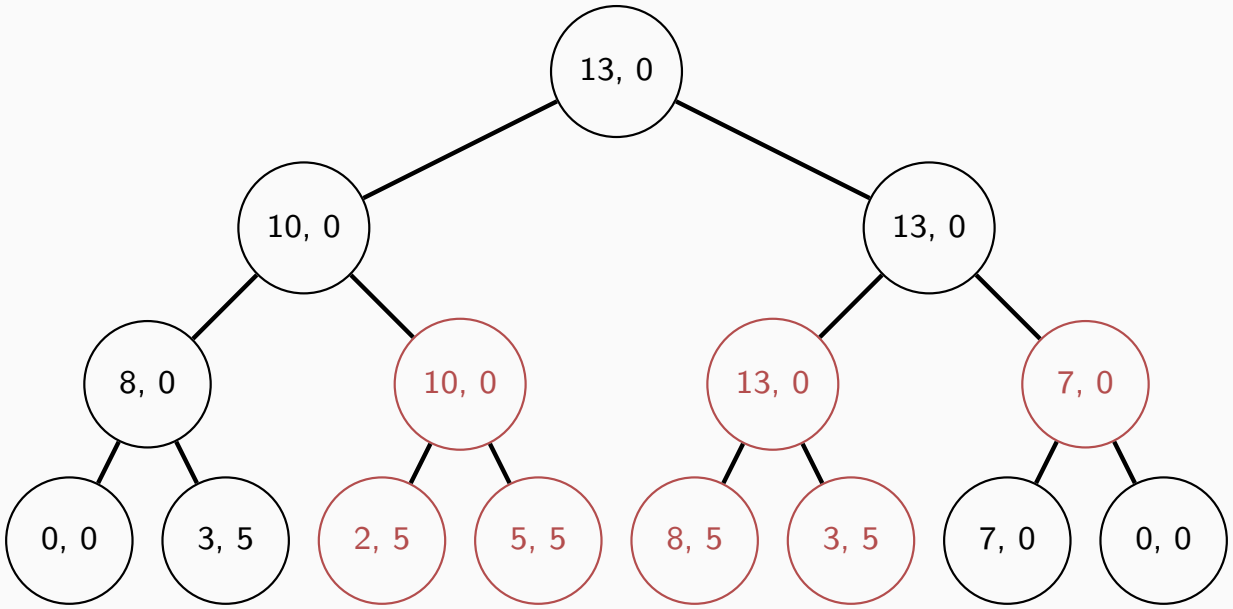


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på spøringer av et lazy-propagnation segment-tre.

Spørre om max fra indeks 3 til 6.

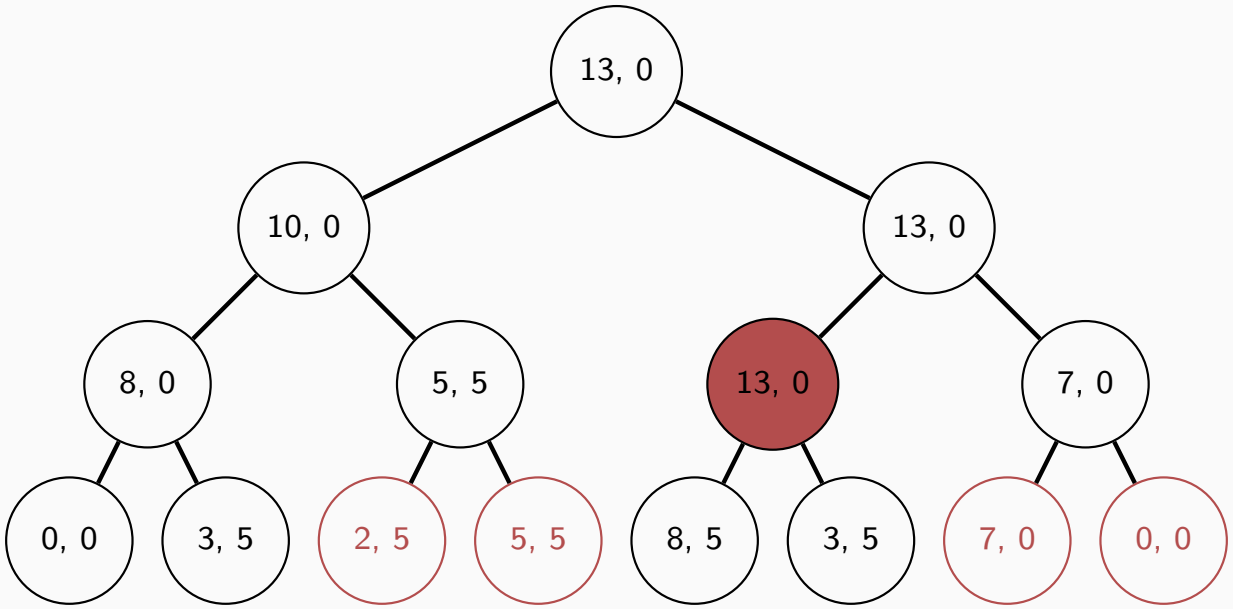


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på spøringer av et lazy-propagnation segment-tre.

Spørre om max fra indeks 3 til 6.

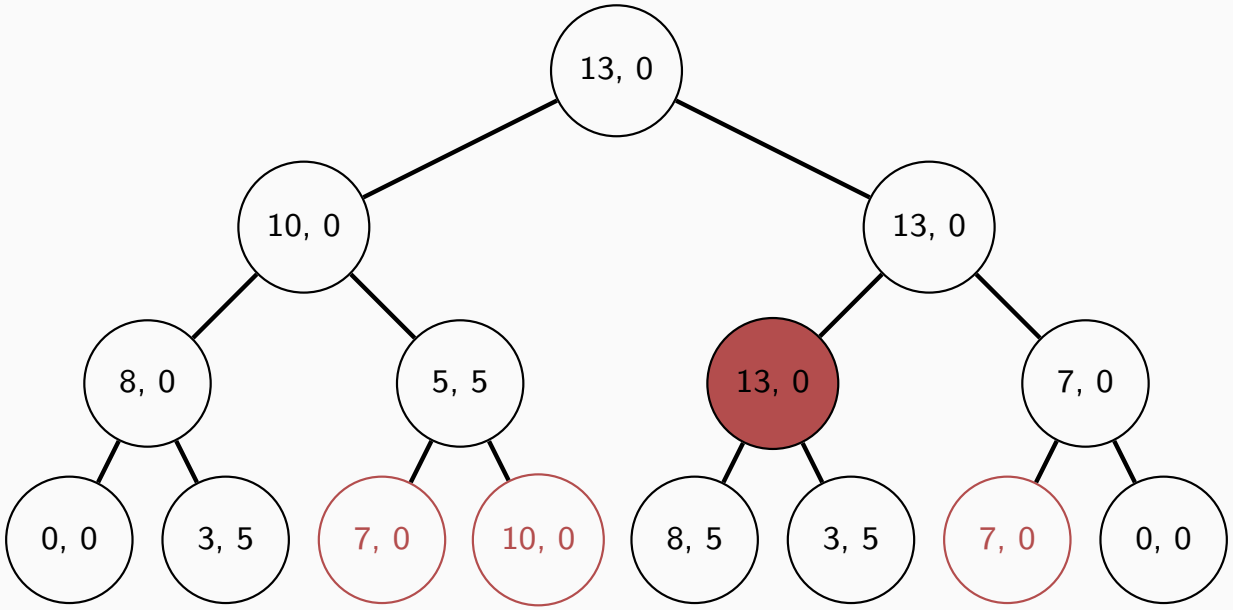


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på spøringer av et lazy-propagnation segment-tre.

Spørre om max fra indeks 3 til 6.

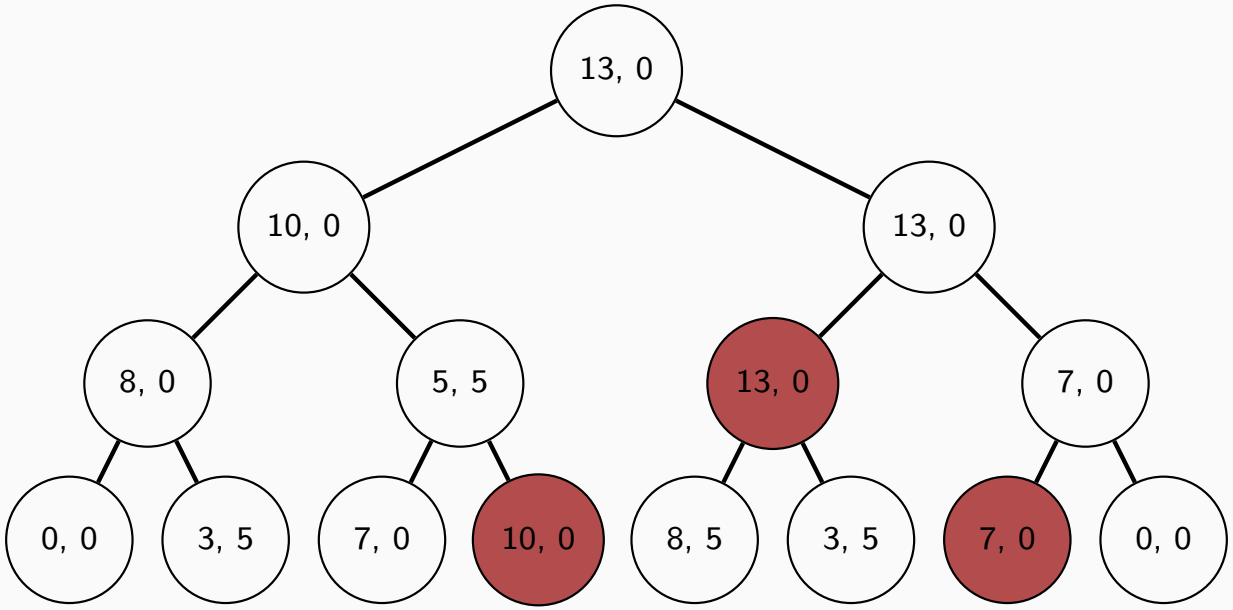


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på spøringer av et lazy-propagnation segment-tre.

Spørre om max fra indeks 3 til 6.

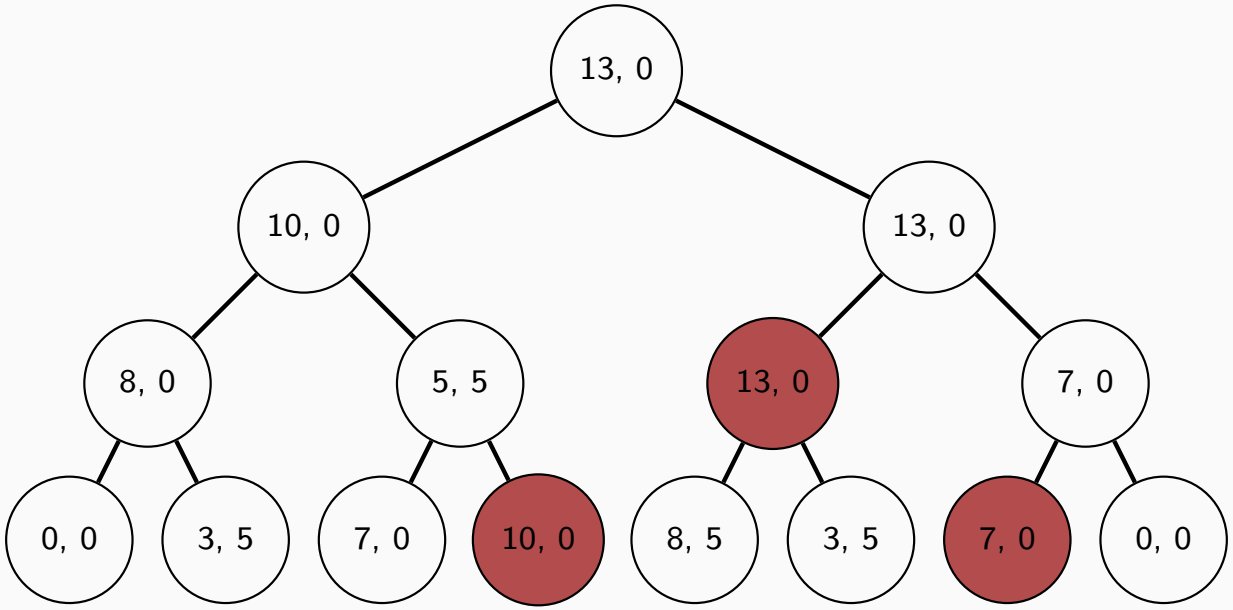


0	8	7	10	13	8	7	0

Lazy propagnation

Et eksempel på spøringer av et lazy-propagnation segment-tre.

Spørre om max fra indeks 3 til 6.



0	8	7	10	13	8	7	0

Svaret er $\max(10, 13, 7) = 13$

Lazy propagnation

Implmentasjon av lazy-propagnation spøring.

```
T get(int l, int r) {
    return get(l, r, 1);
}
T get(int l, int r, int i) {
    if (i >= N || lis[i] > r || ris[i] < l) return O::e;
    propagnate_lazy(i);
    if (lis[i] <= l && ris[i] >= r) {
        return vals[i];
    }

    return O{}(get(l, r, i*2), get(l, r, i*2+1));
}
```

Lazy propagnation

En lazy propagnation oppgave: Bubble Sort 2 fra JOI 2018[1]

Bubble sort is an algorithm to sort a sequence. Let's say we are going to sort a sequence $A_0, A_1, \ldots, A_{N-1}$ of length N in non-decreasing order. Bubble sort swaps two adjacent numbers when they are not in the correct order. Swaps are done by repeatedly passing through the sequence. Precisely speaking, in a pass, we swap A_i and A_{i+1} if $A_i > A_{i+1}$, for $i = 0, 1, \ldots, N-2$ in this order. It is known that any sequence can be sorted in non-decreasing order by some passes. For a sequence A, we define the **number of passes by bubble sort** as the number of passes needed to sort A using the above algorithm.

JOI-kun has a sequence A of length N. He is going to process Q queries of modifying values of A. To be specific, in the $(j+1)$-th query $(0 \leq j \leq Q-1)$, the value of A_{X_j} is changed into V_j.

JOI-kun wants to know the number of passes by bubble sort for the sequence after processing each query.

---

[1] Har ikke funnet en online judge med oppgaver herfra

Lazy propagnation

Begrensninger:

- $1 \leq N \leq 500\,000$
- $1 \leq Q \leq 500\,000$
- $1 \leq A_i \leq 1\,000\,000\,000$
- $1 \leq V_i \leq 1\,000\,000\,000$

Lazy propagnation

Løsning:

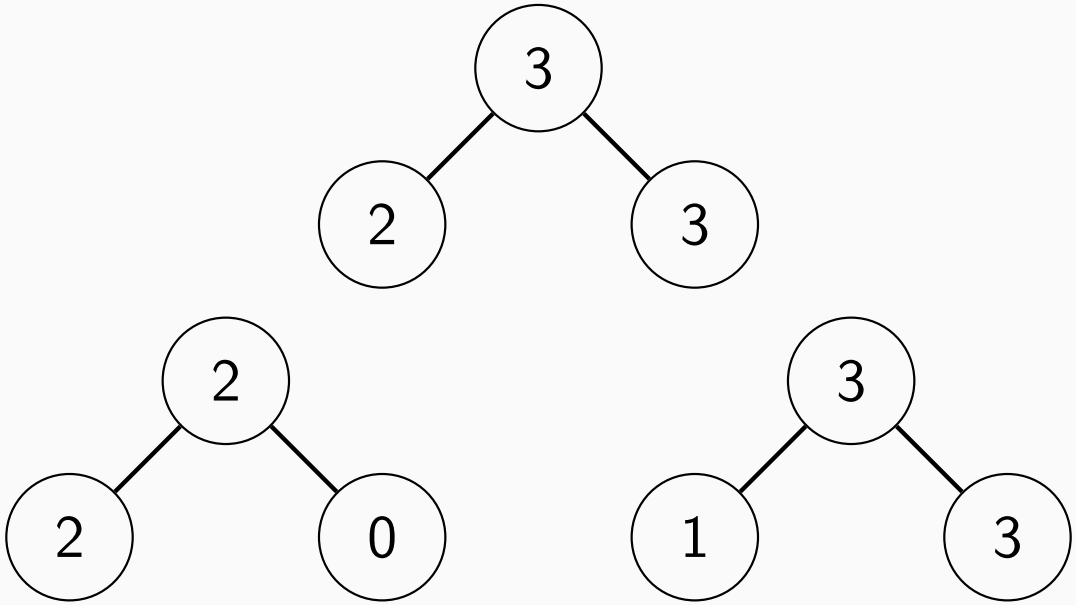
- Antall passes er lik den største differansen mellom den oprinlige poisisjonen og den sorterte posisjonen til hvert tall
- Ved å sortere alle tallene og putte alle differansene inn i et max-tre
- For hver spøring kan man flytte tallet i spøringen og øke differansen til alle tallene til høyre for posisjonen og senke differansen til alle tallene til venstre for posisjonen

Multidimensjonale segment-trær

- Multidimensjonale segment-trær kan gjøre spøringer og oppdateringer i flere dimensjoner
- Eks: største element innenfor et rektangel i et grid
- Skal se på 2d segment-trær, men kan generaliseres til flere dimensjoner
- Multidimensjonale segment-trær er **ikke** det samme som quad/oct-trær da disse har $O(n)$ verste falls kompleksitet, mens multidimensjonale segment-trær har $O(\log^d n)$ verste falls kompleksitet
- Implmenteres ved å ha et segment-tre av segment-trær
- Operasjone i et multidimensjonalt segment-tre burde være kommutativ i tillegg til assosiativ.

Multidimensjonale segment-trær

Et eksempel max-2d-segment-tre.



0	3
2	1

Multidimensjonale segment-trær

Implentasjon av et generisk 2d-segment-tre.

```
template<class T, class O>
class SegmentTree2D {
    int N, offset;
    vector<SegmentTree<T, O>> vals;
public:
    SegmentTree2D(vector<vector<T>> initial) : N(1) {
        while (N < initial.size()+2) N*=2;
        offset = N+1; N*=2;
        vals = vector<T>(N, SegmentTree<T, O>({}));
        for (int i = 0; i < initial.size(); i++)
            vals[i+offset] = SegmentTree<T,O>(initial[i]);

        for (int i = offset-1; i > 0; i--)
            vals[i] = SegmentTree::combine(vals[i*2], vals[i*2+1]);
    }
};
```

Multidimensjonale segment-trær

Implentasjon av et generisk 2d-segment-tre.

```
static SegmentTree<T, O> SegmentTree::combine(
        SegmentTree<T, O> l, SegmentTree<T, O> r) {
    N = l.N;
    offset = l.offset;
    vals = l.vals;
    for (int i = 0; i < N; i++)
        vals[i] = O{}(vals[i], r.vals[i]);
}
```

Multidimensjonale segment-trær

Oppdatering av et 2d-segment-tre.

```
void update(int i, int j, T val) {
    i += offset;
    vals[i].update(j, val);
    i /= 2;
    while (i > 0) {
        vals[i].update(j,
            O{}(vals[i*2].get(j,j), vals[i*2+1].get(j,j)));
        i /= 2;
    }
}
```

Multidimensjonale segment-trær

Spørring av et 2d-segment-tre.

```
T get(int l1, int r1, int l2, int r2) {
    l1 += offset-1;
    r1 += offset+1;
    T res = O::e;
    while (l1/2 != r1/2) {
        if (l1 % 2 == 0) res = O{}(res, vals[l1+1].get(l2, r2));
        if (r1 % 2 == 1) res = O{}(res, vals[r1-1].get(l2, r2));
        l1 /= 2;
        r1 /= 2;
    }

    return res;
}
```

Multidimensjonale segment-trær

Kommer tilbake til oppgave

En alternativ representasjon

- De andre segment-tre variantene trenger en alternativ representasjon.
- Istedenfor å bruke arrays med indeksering så må vi bruke noder med pekere til andre noder
- Disse andre variantene tar å dynamisk legger til og fjerner noder

```
template<class T, class O>
struct SegmentTree {
    T value;
    int li;
    int ri;
    shared_ptr<SegmentTree<T, O>> ln;
    shared_ptr<SegmentTree<T, O>> rn;
};
```

En alternativ representasjon

```
SegmentTree(T initial, int l, int r) {
    li = l;
    ri = r;
    value = initial;
}

SegmentTree(vector<T> &initial, int l, int r) {
    li = l;
    ri = r;
    if (l == r) {
        value = initial[l];
        return;
    }

    ln = make_shared<SegmentTree<T, O>>(initial, l, (l+r)/2);
    rn = make_shared<SegmentTree<T, O>>(initial, (l+r)/2+1, r);

    value = O{}(ln->value, rn->value);
}

SegmentTree(vector<T> &initial) : SegmentTree(initial, 0, initial.size()-1) {}
```

Persistent segment-trær

- Persistent segment-trær lagrer alle tidligere trær slik at man kan gjøre spøringer på eldre versjoner av datastrukturen
- Siden en oppdatering på et segment-tre bare påvirker $O(\log n)$ noder bruker et persistent segment-tre ikke mer enn $O(n + q \log n)$ plass

Persistent segment-trær

Implmentasjon av persistent segment-tre

```
template<class T, class O>
class PersistentSegmentTree {
    vector<SegmentTree<T, O>> trees;
public:
    PersistentSegmentTree(vector<T> initial) {
        trees.push_back(SegmentTree<T, O>(initial));
    }
};
```

Oppdatering av persistent segment-tre

```
void update(int i, T val) {
    trees.push_back(create(i, val, trees[trees.size()-1]));
}

SegmentTree<T, O> create(int i, T val, SegmentTree<T, O> &curr) {
    SegmentTree<T, O> tree = SegmentTree<T, O>(val, curr.l, curr.r);
    if (curr.l == curr.r) {
        tree.value = val;
        return tree;
    }

    tree.ln = curr.ln;
    tree.rn = curr.rn;

    if (i <= (curr.l+curr.r)/2)
        tree.ln = make_shared<SegmentTree<T,O>>(create(i, val, *curr.ln));
    else
        tree.rn = make_shared<SegmentTree<T,O>>(create(i, val, *curr.rn));

    tree.value = O{}(tree.ln->value, tree.rn->value);

    return tree;
}
```

Persistent segment-trær

Spørring av persistent segment-tre

```
T get(int k, int l, int r) {
    return trees[i].get(l, r);
}

T SegmentTree::get(int l, int r) {
    if (ri < l || li > r)
        return O::e;

    if (li >= l && ri <= r)
        return value;

    return O{}(ln->get(l, r), rn->get(l, r));
}
```

Persistent segment-tre oppgave: K-th Number[2]

You are working for Macrohard company in data structures department. After failing your previous task about key insertion you were asked to write a new data structure that would be able to return quickly k-th order statistics in the array segment.

That is, given an array $a[1 \ldots n]$ of different integer numbers, your program must answer a series of questions $Q(i, j, k)$ in the form: "What would be the k-th number in $a[i \ldots j]$ segment, if this segment was sorted?"

For example, consider the array $a = (1, 5, 2, 6, 3, 7, 4)$. Let the question be $Q(2, 5, 3)$. The segment $a[2 \ldots 5]$ is $(5, 2, 6, 3)$. If we sort this segment, we get $(2, 3, 5, 6)$, the third number is 5, and therefore the answer to the question is 5.

[2] Oppgaven finnes på: `https://www.spoj.com/problems/MKTHNUM/`

Persistent segment-trær

Begrensninger:

- $1 \leq n \leq 100\,000$
- $1 \leq m \leq 5000$
- $0 \leq a[i] \leq 10^9$

Persistent segment-trær

Løsning:

- Et persistent segment tree over hvor mange det finnes av hver tall
- Første iterasjon er ingen av elementene i listen
- Andre iterasjon er alle elementene i listen opp til det første
- N-te iterasjon er alle elementene i listen opp til det n-te
- En spørring blir et binær-søk på differansen av det i-te og j-te treet
- Indeks komprimering
- Rå pekere

Dynamiske segment-trær

- Dynamiske segment-trær er tomme til å begynne med og man legger til noder etter hvert som man trenger dem
- Passer til situasjoner hvor man har et segment-tre over et enormt område og hvor man ikke vet spørringene på forhånd

Dynamiske segment-trær

Implmentasjon av dynamiske segment-tre

```
template<class T, class O>
struct DynamicSegmentTree {
    T value;
    int li;
    int ri;
    bool leaf;
    shared_ptr<DynamicSegmentTree<T, O>> ln;
    shared_ptr<DynamicSegmentTree<T, O>> rn;

    DynamicSegmentTree(int l, int r) {
        value = O::e;
        li = l;
        ri = r;
        leaf = true;
    }
};
```

Dynamiske segment-trær

Oppdatering av dynamiske segment-tre

```
void update(int i, T val) {
    if (i < li || i > ri) return;
    if (li == ri) {
        value = val;
        return;
    }
    if (leaf) {
        ln = make_shared<DynamicSegmentTree<T, O>>(li, (li+ri)/2);
        rn = make_shared<DynamicSegmentTree<T, O>>((li+ri)/2+1, ri);
        leaf = false;
    }

    ln->update(i, val);
    rn->update(i, val);

    value = O{}(ln->value, rn->value);
}
```

Dynamiske segment-trær

Spørring av dynamiske segment-tre

```
T get(int l, int r, T val) {
    if (r < li || l > ri) return O::e;

    if(l >= li && r <= ri) return value;

    if (leaf) return value;

    return O{}(ln->value, rn->value);
}
```

Kombinasjoner av segment-trær

- De fleste teknikene beskrevet så langt kan kombineres til enda kraftigere segment-trær
- Her er noen oppgaver hvor det er nødvendig

IOI 2013: Game[3]

Bazza and Shazza are playing a game. The board is a grid of cells, with R rows numbered $0, \ldots, R-1$, and C columns, numbered $0, \ldots, C-1$. We let (P, Q) denote the cell in row P and column Q. Each cell contains a non-negative integer, and at the beginning of the game all of these integers are zero.

The game proceeds as follows. At any time, Bazza may either:

- update a cell (P, Q), by assigning the integer that it contains
- ask Shazza to calculate the greatest common divisor (GCD) of all integers within a rectangular block of cells, with opposite corners (P, Q) and (U, V) inclusive.

Bazza will take $NU + NQ$ actions (updating cells NU times and asking questions NQ times) before he gets bored and goes outside to play cricket.

Your task is to work out the correct answers.

[3]Oppgaven kan finnes på `http://contest.yandex.com/ioi`

IOI 2013: Game

Begrensninger:

- $1 \leq R, R \leq 10^9$
- $0 \leq K \leq 10^{18}$
- $N_U \leq 22\,000$
- $N_Q \leq 250\,000$

IOI 2013: Game - Løsning

- 2D dynamisk segment-tre
- Path compression
- Bare generere de nodene du må

IOI 2013: Game - Løsning

```
long long gcd(long long a, long long b) {
    if (b == 0) return a;
    return gcd(b, a % b);
}

struct GCD {
    static constexpr long long e = 0;
    long long operator() (long long a, long long b) {
        return gcd(a, b);
    }
};

DynamicSegmentTree2D<long long, GCD> tree(0, 0, 0, 0);

void init(int R, int C) {
    tree = DynamicSegmentTree2D<long long, GCD>(0, R-1, 0, C-1);
}

void update(int P, int Q, long long K) {
    tree.update(P, Q, K);
}

long long calculate(int P, int Q, int U, int V) {
    return tree.get(P, U, Q, V);
}
```