2-14 B-Trees

Hengfeng Wei

hfwei@nju.edu.cn

June 02, 2020

Organization and Maintenance of Large Ordered Indexes

R. BAYER and E. MCCREIGHT

Received September 29, 1971

Summary. Organization and maintenance of an index for a dynamic random access file is considered. It is assumed that the index must be kept on some pseudo random access backup store like a disc or a drum. The index organization described allows retrieval, insertion, and deletion of keys in time proportional to $\log_k I$ where I is the size of the index and k is a device dependent natural number such that the performance of the scheme becomes near optimal. Storage utilization is at least 50% but generally much higher. The pages of the index are organized in a special data-structure, so-called B-trees. The scheme is analyzed, performance bounds are obtained, and a near optimal k is computed. Experiments have been performed with indexes up to 100000 keys. An index of size 15000 (100000) can be maintained with an average of 9 (at least 4) transactions per second on an IBM 360/44 with a 2311 disc.

Organization and Maintenance of Large Ordered Indexes

R. BAYER and E. MCCREIGHT

Received September 29, 1971

Summary. Organization and maintenance of an index for a dynamic random access file is considered. It is assumed that the index must be kept on some pseudo random access backup store like a disc or a drum. The index organization described allows retrieval, insertion, and deletion of keys in time proportional to $\log_k I$ where I is the size of the index and k is a device dependent natural number such that the performance of the scheme becomes near optimal. Storage utilization is at least 50% but generally much higher. The pages of the index are organized in a special data-structure, so-called B-trees. The scheme is analyzed, performance bounds are obtained, and a near optimal k is computed. Experiments have been performed with indexes up to 100000 keys. An index of size 15000 (100000) can be maintained with an average of 9 (at least 4) transactions per second on an IBM 360/44 with a 2311 disc.

"Bayer and McCreight introduced B-trees in 1972;

they did not explain their choice of name."

Organization and Maintenance of Large Ordered Indexes

R. Bayer and E. McCreight

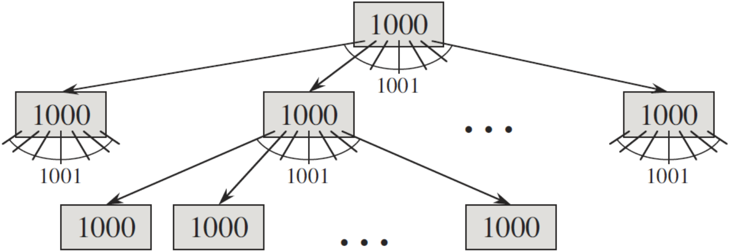
Received September 29, 1971

Summary. Organization and maintenance of an index for a dynamic random access file is considered. It is assumed that the index must be kept on some pseudo random access backup store like a disc or a drum. The index organization described allows retrieval, insertion, and deletion of keys in time proportional to $\log_k I$ where I is the size of the index and k is a device dependent natural number such that the performance of the scheme becomes near optimal. Storage utilization is at least 50% but generally much higher. The pages of the index are organized in a special data-structure, so-called B-trees. The scheme is analyzed, performance bounds are obtained, and a near optimal k is computed. Experiments have been performed with indexes up to 100000 keys. An index of size 15000 (100000) can be maintained with an average of 9 (at least 4) transactions per second on an IBM 360/44 with a 2311 disc.

ID	Name	Gender	Age	...

"Bayer and McCreight introduced B-trees in 1972;

they did not explain their choice of name."



2-way *vs.* multi-way

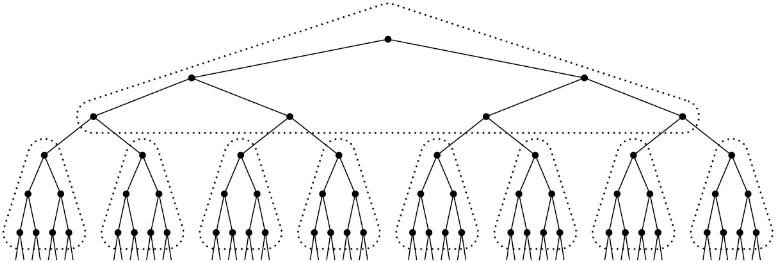
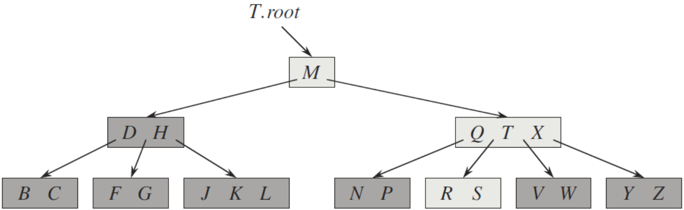


Fig. 29. A large binary search tree can be divided into "pages."

indexes (keys) *vs.* pages

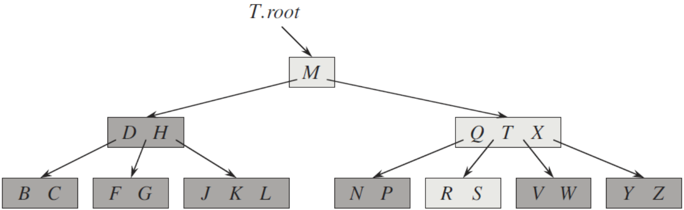
Minimum (TC 18.2-3)

Explain how to find the minimum key stored in a B-tree.



Minimum (TC 18.2-3)

Explain how to find the minimum key stored in a B-tree.



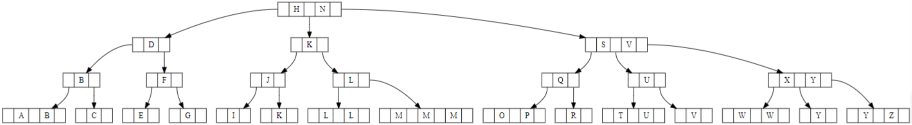
the leftmost key in the leftmost node

Predecessor (TC 18.2-3)

Explain how to find the predecessor of a given key (x, i) stored in a B-tree.

Predecessor (TC 18.2-3)

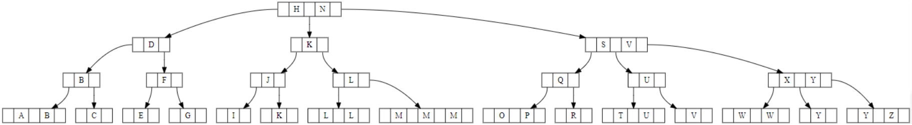
Explain how to find the predecessor of a given key (x, i) stored in a B-tree.



B-tree with 37 characters.

Predecessor (TC 18.2-3)

Explain how to find the predecessor of a given key (x, i) stored in a B-tree.

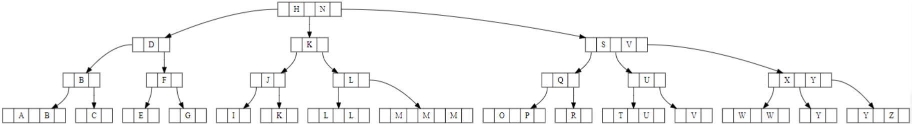


B-tree with 37 characters.

$$x.leaf = 0$$

Predecessor (TC 18.2-3)

Explain how to find the predecessor of a given key (x, i) stored in a B-tree.



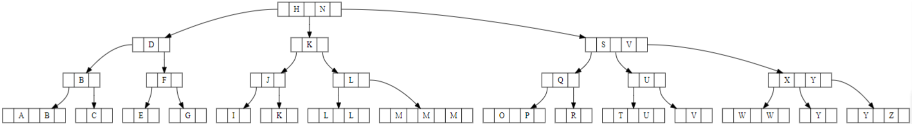
B-tree with 37 characters.

$$x.leaf = 0$$

H N S V

Predecessor (TC 18.2-3)

Explain how to find the predecessor of a given key (x, i) stored in a B-tree.



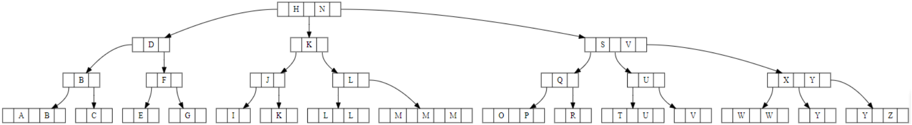
B-tree with 37 characters.

$x.leaf = 0$

$H \quad N \quad S \quad V$

find the rightmost key in $x.c_i$

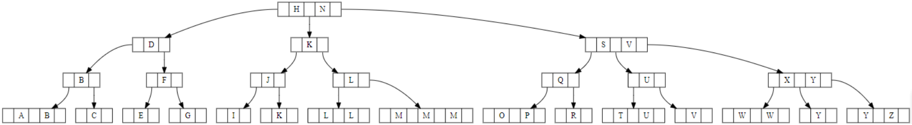
$$(x, i)$$



$$x.leaf = 1$$

$P \quad U \quad T \quad O \quad A$

$$(x, i)$$



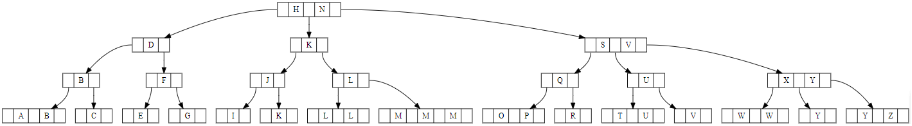
$$\color{red}{x.leaf = 1}$$

$P \quad U \quad T \quad O \quad A$

$$i \geq 2 \implies (x, i-1)$$

$i = 1 \implies$ find (y, j) such that x is the leftmost key in $y.c_{j+1}$

$$(x, i)$$



$$\color{red}{x.leaf = 1}$$

P U T O A

$$i \geq 2 \implies (x, i-1)$$

$i = 1 \implies$ find (y, j) such that x is the leftmost key in $y.c_{j+1}$

A is the only key which has no predecessor.

```
procedure B-TREE-PREDECESSOR(T, x, i)        ▷ x.key_i in B-tree T
    if x.leaf = 0 then
        return the rightmost key in x.c_i
```

```
procedure B-TREE-PREDECESSOR(T, x, i)        ▷ x.key_i in B-tree T
    if x.leaf = 0 then
        return the rightmost key in x.c_i
    else if i ≥ 2 then                       ▷ x.leaf = 1
        return (x, i − 1)
```

procedure B-TREE-PREDECESSOR(T, x, i) ▷ $x.key_i$ in B-tree T

  if $x.leaf = 0$ **then**

    return the rightmost key in $x.c_i$

  else if $i \geq 2$ **then** ▷ $x.leaf = 1$

    return $(x, i-1)$

  else ▷ $x.leaf = 1 \land i = 1$

    $y \leftarrow x.p$

    while $y \neq T.root \land x = y.c_1$ **do** ▷ exit: $y = T.root \lor x \neq y.c_1$

      $x \leftarrow y$

      $y \leftarrow y.p$

```
procedure B-TREE-PREDECESSOR(T, x, i)              ▷ x.key_i in B-tree T
    if x.leaf = 0 then
        return the rightmost key in x.c_i
    else if i ≥ 2 then                             ▷ x.leaf = 1
        return (x, i − 1)
    else                                           ▷ x.leaf = 1 ∧ i = 1
        y ← x.p
        while y ≠ T.root ∧ x = y.c_1 do            ▷ exit: y = T.root ∨ x ≠ y.c_1
            x ← y
            y ← y.p
        if x = y.c_1 then                          ▷ y = T.root ∧ x = y.c_1
            return "no predecessor"
```

```
procedure B-TREE-PREDECESSOR(T, x, i)          ▷ x.key_i in B-tree T
    if x.leaf = 0 then
        return the rightmost key in x.c_i
    else if i ≥ 2 then                          ▷ x.leaf = 1
        return (x, i − 1)
    else                                        ▷ x.leaf = 1 ∧ i = 1
        y ← x.p
        while y ≠ T.root ∧ x = y.c_1 do         ▷ exit: y = T.root ∨ x ≠ y.c_1
            x ← y
            y ← y.p
        if x = y.c_1 then                       ▷ y = T.root ∧ x = y.c_1
            return "no predecessor"
        else                                    ▷ x ≠ y.c_1
            j ← 2
            while y.c_j ≠ x do
                j ← j + 1
            return (y, j − 1)                   ▷ x = y.c_j
```

Insertion (TC 18.2-4 *)

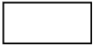
Suppose that we insert the keys $\{1, 2, \ldots, n\}$ in increasing order into an empty B-tree with minimum degree 2.

How many nodes, denoted X_n, does the final B-tree have?

Insertion (TC 18.2-4 ⋆)

Suppose that we insert the keys $\{1, 2, \ldots, n\}$ in increasing order into an empty B-tree with minimum degree 2.

How many nodes, denoted X_n, does the final B-tree have?



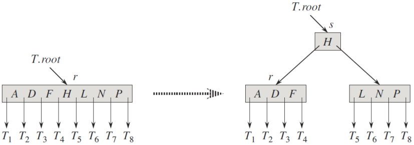
$X_0 = 1$

By Yangjing Dong (June 2018)

https://maxmute.com/TC18.2-4.html

Only SPLIT can create new nodes.

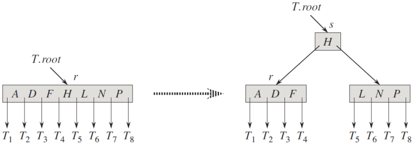
Only SPLIT can create new nodes.

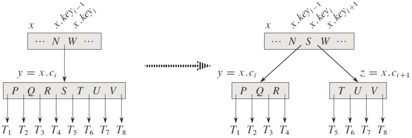


root SPLIT

+2

Only SPLIT can create new nodes.





root SPLIT

$+2$

non-root SPLIT

$+1$

(I) Which nodes will SPLIT? S

(II) When does each node $s \in S$ SPLIT? $T_s = \langle s_1, s_2, \ldots \rangle$

(III) How does it SPLIT, as a root or a non-root? $T_s = s_R \uplus s_{NR}$

(I) Which nodes will SPLIT? S

(II) When does each node $s \in S$ SPLIT? $T_s = \langle s_1, s_2, \dots \rangle$

(III) How does it SPLIT, as a root or a non-root? $T_s = s_R \uplus s_{NR}$

$$X_n = 1 + \sum_{s \in S} \Big(2\, |s_R| + |s_{NR}| \Big)$$

(I) Which nodes will SPLIT? S

(II) When does each node $s \in S$ SPLIT? $T_s = \langle s_1, s_2, \ldots \rangle$

(III) How does it SPLIT, as a root or a non-root? $T_s = s_R \uplus s_{NR}$

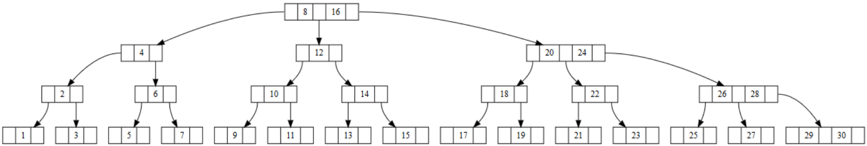
$$X_n = 1 + \sum_{s \in S} \Big(2\,|s_R| + |s_{NR}|\Big)$$

$$X_n = 1 + \sum_{s \in S} \Big(2\,|s_R| + |s_{NR}| \Big)$$

(I) Which nodes will SPLIT?

$$X_n = 1 + \sum_{s \in S} \left(2\,|s_R| + |s_{NR}| \right)$$

(I) Which nodes will SPLIT?



$$1, 2, \ldots, 30$$

$$S = \left\{ \text{the nodes in the rightmost chain} \right\}$$

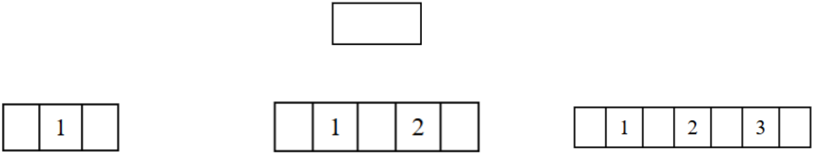
(II) When does each node $s \in S$ SPLIT?

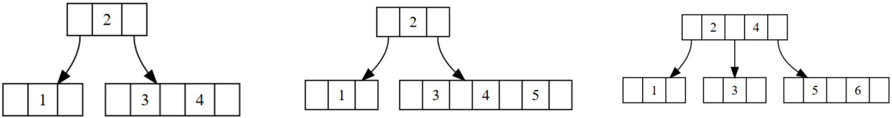
$$T_s = \langle s_1, s_2, \ldots \rangle$$

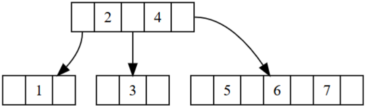
(II) When does each node $s \in S$ SPLIT?

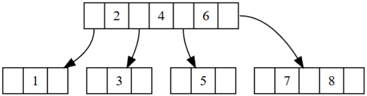
$$T_s = \langle s_1, s_2, \ldots \rangle$$

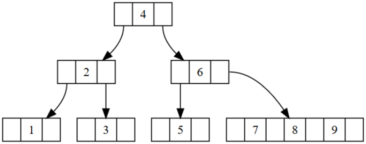
Let's focus the rightmost node first, denoted A.

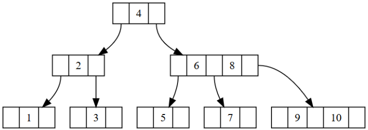

1
1 2
1 2 3

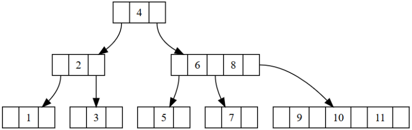

2
1
3
4
2
1
3
4
5
2
4
1
3
5
6

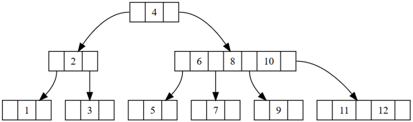

2
4
1
3
5
6
7

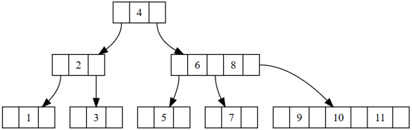

2
4
6
1
3
5
7
8

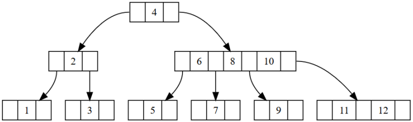

4
2
6
1
3
5
7
8
9


4
2
6
8
1
3
5
7
9
10


4
2
6
8
1
3
5
7
9
10
11


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





A SPLIT : 4, 6, 8, 10, 12, ...

(II) When does each node SPLIT?

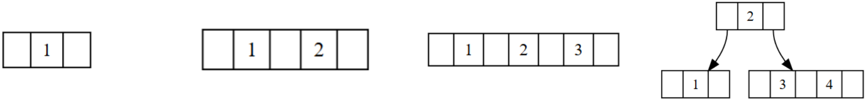
(II) When does each node SPLIT?

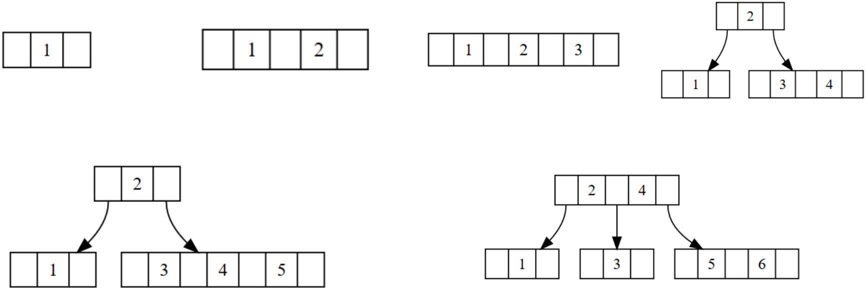
Let's consider the parent of A, denoted $B \triangleq p(A)$.

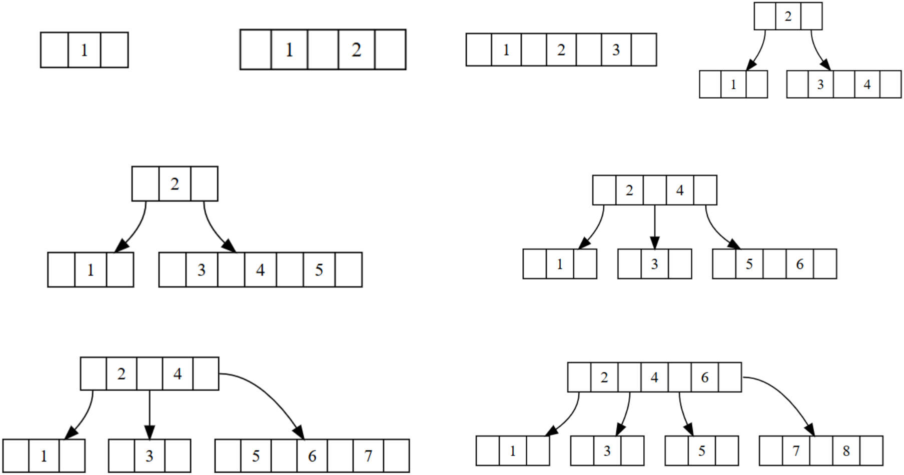
(II) When does each node SPLIT?

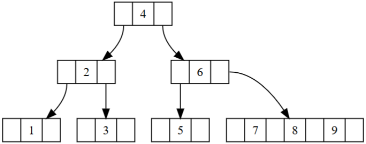
Let's consider the parent of A, denoted $B \triangleq p(A)$.

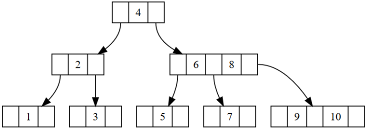
Every time A splits, B obtains a new key.

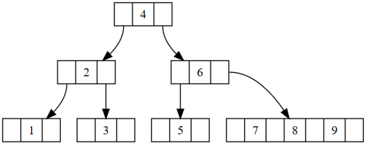

1
1 2
1 2 3
2
1
3 4

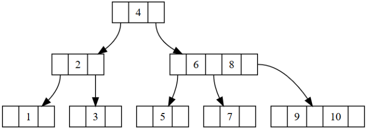

1
1 2
1 2 3
2
1
3 4
2
1
3 4 5
2 4
1
3
5 6

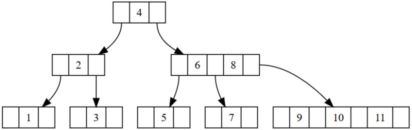

1
1
2
1
2
3
2
1
3
4
2
1
3
4
5
2
4
1
3
5
6
2
4
1
3
5
6
7
2
4
6
1
3
5
7
8

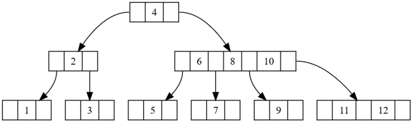

4
2
6
1
3
5
7
8
9

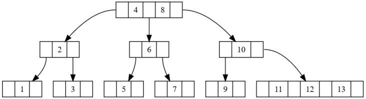

4
2
6
8
1
3
5
7
9
10

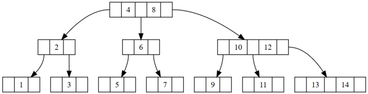

4
2
6
1
3
5
7
8
9

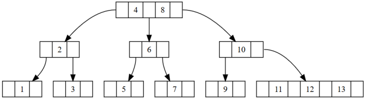

4
2
6
8
1
3
5
7
9
10

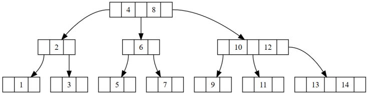

4
2
6
8
1
3
5
7
9
10
11

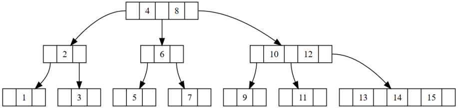

4
2
6
8
10
1
3
5
7
9
11
12

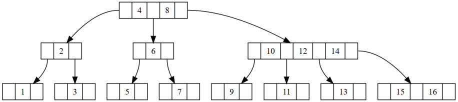

4 8
2
6
10
1
3
5
7
9
11 12 13

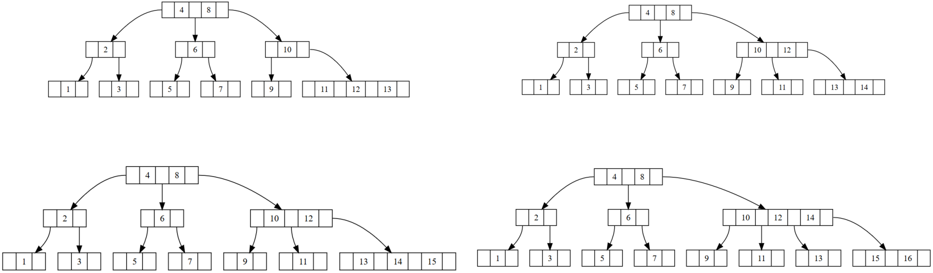

4 8
2
6
10 12
1
3
5
7
9
11
13 14


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


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


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


4 8
2
6
10 12 14
1
3
5
7
9
11
13
15 16



A SPLIT : 4, 6, 8, 10, 12, ...

B SPLIT : 9, 13, 17, 21, 25, ...

(II) When does each node SPLIT?

(II) When does each node SPLIT?

Let's consider the parent of B, denoted $C = p(B)$.

$$A \text{ SPLIT} : 4, \quad 6, \quad 8, \quad 10, \quad 12, \quad \ldots$$

$$B \text{ SPLIT} : 9, \quad 13, \quad 17, \quad 21, \quad 25, \quad \ldots$$

$$C \text{ SPLIT} : 18, \quad 26, \quad 34, \quad 42, \quad 50, \quad \ldots$$

$$A \text{ SPLIT}: 4, \quad 6, \quad 8, \quad 10, \quad 12, \quad \ldots$$

$$B \text{ SPLIT}: 9, \quad 13, \quad 17, \quad 21, \quad 25, \quad \ldots$$

$$C \text{ SPLIT}: 18, \quad 26, \quad 34, \quad 42, \quad 50, \quad \ldots$$

$$A : 1 \qquad B : 2 \qquad C : 3$$

T_i : the first time point the i-th node splits

$$A \text{ SPLIT}: 4, \quad 6, \quad 8, \quad 10, \quad 12, \quad \dots$$

$$B \text{ SPLIT}: 9, \quad 13, \quad 17, \quad 21, \quad 25, \quad \dots$$

$$C \text{ SPLIT}: 18, \quad 26, \quad 34, \quad 42, \quad 50, \quad \dots$$

$$A: 1 \qquad B: 2 \qquad C: 3$$

$T_i:$ the first time point the i-th node splits

$$T_1 = 4$$

$$A \text{ SPLIT}: 4, \quad 6, \quad 8, \quad 10, \quad 12, \quad \ldots$$

$$B \text{ SPLIT}: 9, \quad 13, \quad 17, \quad 21, \quad 25, \quad \ldots$$

$$C \text{ SPLIT}: 18, \quad 26, \quad 34, \quad 42, \quad 50, \quad \ldots$$

$$A:1 \qquad B:2 \qquad C:3$$

T_i : the first time point the i-th node splits

$$T_1 = 4$$

$$T_i = \underbrace{T_{i-1}}_{\text{its right child first split}} + \underbrace{2 \times 2^{i-1}}_{\text{its right child split twice more}} + \underbrace{1}_{\text{insert one more}}$$

$$A \text{ SPLIT} : 4, \quad 6, \quad 8, \quad 10, \quad 12, \quad \ldots$$
$$B \text{ SPLIT} : 9, \quad 13, \quad 17, \quad 21, \quad 25, \quad \ldots$$
$$C \text{ SPLIT} : 18, \quad 26, \quad 34, \quad 42, \quad 50, \quad \ldots$$

$$A : 1 \qquad B : 2 \qquad C : 3$$

$T_i :$ the first time point the i-th node splits

$$T_1 = 4$$

$$T_i = \underbrace{T_{i-1}}_{\text{its right child first split}} + \underbrace{2 \times 2^{i-1}}_{\text{its right child split twice more}} + \underbrace{1}_{\text{insert one more}}$$

$$T_i = 2^{i+1} + i - 1$$

$$X_n = 1 + \sum_{s \in S} \Big(2\, |s_R| + |s_{NR}| \Big)$$

$$(T_s = s_R \uplus s_{NR})$$

(III) How does it SPLIT, as a root or a non-root?

$$X_n = 1 + \sum_{s \in S} \Big(2\,|s_R| + |s_{NR}| \Big)$$

$$(T_s = s_R \uplus s_{NR})$$

(III) How does it SPLIT, as a root or a non-root?

$$s_R = \{s_1\} \qquad s_{NR} = \{s_2, s_3, \dots\}$$

$$|s_R| = 1 \qquad |s_{NR}| = |T_s| - 1$$

$$X_n = 1 + \sum_{s \in S} \Big(2\ |s_R| + |s_{NR}| \Big)$$

$$(T_s = s_R \uplus s_{NR})$$

(III) How does it SPLIT, as a root or a non-root?

$$s_R = \{s_1\} \qquad s_{NR} = \{s_2, s_3, \dots\}$$

$$|s_R| = 1 \qquad |s_{NR}| = |T_s| - 1$$

$$X_n = 1 + \sum_{s \in S} \Big(2 + |T_s| - 1 \Big) = 1 + \sum_{s \in S} \Big(|T_s| + 1 \Big)$$

$$X_n = 1 + \sum_{s \in S} \Big(|T_s| + 1 \Big) \qquad T_i = 2^{i+1} + i - 1$$

$$X_n = 1 + \sum_{s \in S} \Big(|T_s| + 1 \Big) \qquad T_i = 2^{i+1} + i - 1$$

$$X_n = 1 + \sum_{s \in S} \Big(|T_s| + 1 \Big) \qquad T_i = 2^{i+1} + i - 1$$

$$X_n = 1 + \sum_{i=1}^{\infty} [T_i \le n] \bigg(\bigg(\left\lfloor \frac{n - T_i}{2^i} \right\rfloor + 1 \bigg) + 1 \bigg)$$

$$X_n = 1 + \sum_{s \in S} \Big(|T_s| + 1 \Big) \qquad T_i = 2^{i+1} + i - 1$$

$$\begin{aligned} X_n &= 1 + \sum_{i=1}^{\infty} [T_i \le n] \left(\left(\left\lfloor \frac{n - T_i}{2^i} \right\rfloor + 1 \right) + 1 \right) \\ &= 1 + \sum_{i=1}^{\infty} [T_i \le n] \left(\left\lfloor \frac{n - T_i}{2^i} \right\rfloor + 2 \right) \end{aligned}$$

$$X_n = 1 + \sum_{s \in S} \Big(|T_s| + 1 \Big) \qquad T_i = 2^{i+1} + i - 1$$

$$\begin{aligned}
X_n &= 1 + \sum_{i=1}^{\infty} [T_i \le n] \left(\left(\left\lfloor \frac{n - T_i}{2^i} \right\rfloor + 1 \right) + 1 \right) \\
&= 1 + \sum_{i=1}^{\infty} [T_i \le n] \left(\left\lfloor \frac{n - T_i}{2^i} \right\rfloor + 2 \right) \\
&= 1 + \sum_{i=1}^{\infty} [T_i \le n] \left(\left\lfloor \frac{n - 2^{i+1} - i + 1}{2^i} \right\rfloor \right)
\end{aligned}$$

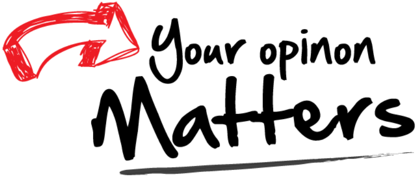
$$X_n = 1 + \sum_{s \in S} \Big(|T_s| + 1 \Big) \qquad T_i = 2^{i+1} + i - 1$$

$$\begin{aligned} X_n &= 1 + \sum_{i=1}^{\infty} [T_i \le n] \left(\left(\left\lfloor \frac{n - T_i}{2^i} \right\rfloor + 1 \right) + 1 \right) \\ &= 1 + \sum_{i=1}^{\infty} [T_i \le n] \left(\left\lfloor \frac{n - T_i}{2^i} \right\rfloor + 2 \right) \\ &= 1 + \sum_{i=1}^{\infty} [T_i \le n] \left(\left\lfloor \frac{n - 2^{i+1} - i + 1}{2^i} \right\rfloor \right) \\ &= 1 + \sum_{\substack{i \ge 1 \\ 2^{i+1} + i - 1 \le n}} \left\lfloor \frac{n - i + 1}{2^i} \right\rfloor \end{aligned}$$

$$X_n = 1 + \sum_{s \in S} \Big(|T_s| + 1 \Big) \qquad T_i = 2^{i+1} + i - 1$$

$$\begin{aligned}
X_n &= 1 + \sum_{i=1}^{\infty} [T_i \le n] \left(\left(\left\lfloor \frac{n - T_i}{2^i} \right\rfloor + 1 \right) + 1 \right) \\
&= 1 + \sum_{i=1}^{\infty} [T_i \le n] \left(\left\lfloor \frac{n - T_i}{2^i} \right\rfloor + 2 \right) \\
&= 1 + \sum_{i=1}^{\infty} [T_i \le n] \left(\left\lfloor \frac{n - 2^{i+1} - i + 1}{2^i} \right\rfloor \right) \\
&= 1 + \sum_{\substack{i \ge 1 \\ 2^{i+1} + i - 1 \le n}} \left\lfloor \frac{n - i + 1}{2^i} \right\rfloor \\
&= 1 + \sum_{\substack{i \ge 0 \\ 2^{i+2} + i \le n}} \left\lfloor \frac{n - i}{2^{i+1}} \right\rfloor
\end{aligned}$$


Thank
You!



Office 302

Mailbox: H016

hfwei@nju.edu.cn