CS F364: Design & Analysis of Algorithms

Lecture-kt15: B-Trees (contd..) + Fibonacci Heap

Dr. Kamlesh Tiwari,
Assistant Professor,
Department of Computer Science and Information Systems,
BITS Pilani, Rajasthan-333031 INDIA

Feb 28, 2017 (Campus @ BITS-Pilani Jan-May 2017)

Recap: B-Tree

- **B-Tree** is a generalization of binary search trees that is balanced and has height $h \leq \log_t \frac{n+1}{2}$ for n-keys

Recap: B-Tree

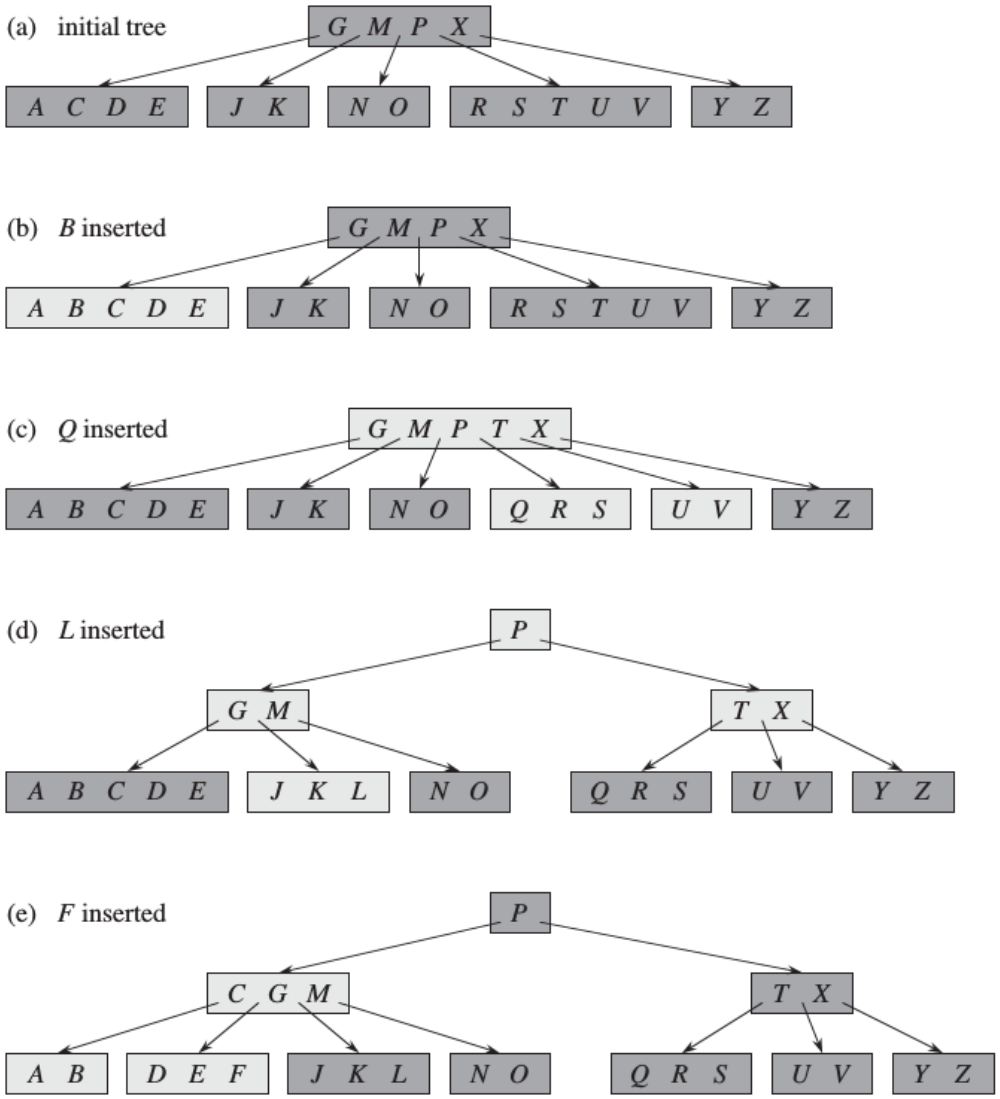
- **B-Tree** is a generalization of binary search trees that is balanced and has height $h \leq \log_t \frac{n+1}{2}$ for n-keys
- For **minimum degree** $t \geq 2$, non-root nodes must have $t-1$ to $2t-1$ keys

Recap: B-Tree

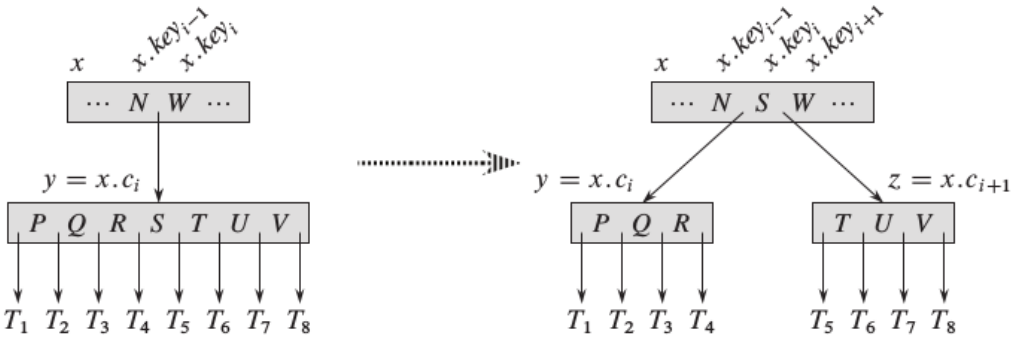
- **B-Tree** is a generalization of binary search trees that is balanced and has height $h \leq \log_t \frac{n+1}{2}$ for n-keys
- For **minimum degree** $t \geq 2$, non-root nodes must have $t-1$ to $2t-1$ keys
- Node x has $x.n$ keys and $x.n+1$ pointers

Recap: B-Tree

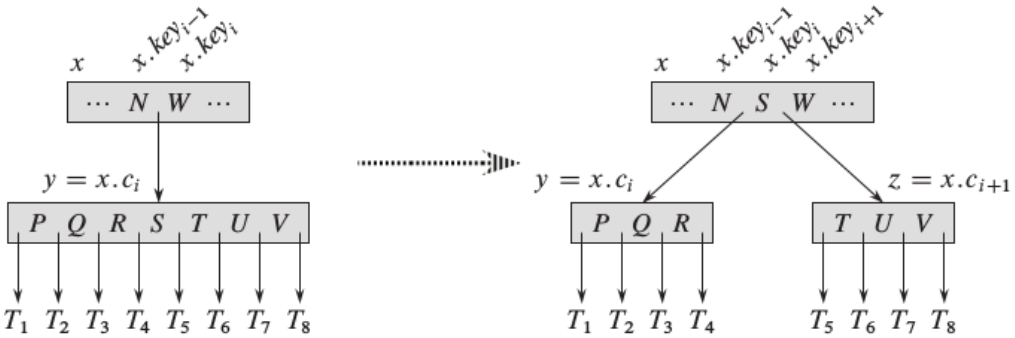
- **B-Tree** is a generalization of binary search trees that is balanced and has height $h \leq \log_t \frac{n+1}{2}$ for n-keys
- For **minimum degree** $t \geq 2$, non-root nodes must have $t-1$ to $2t-1$ keys
- Node x has $x.n$ keys and $x.n+1$ pointers
- **Insertion in B-Tree**



B-Tree Split Child

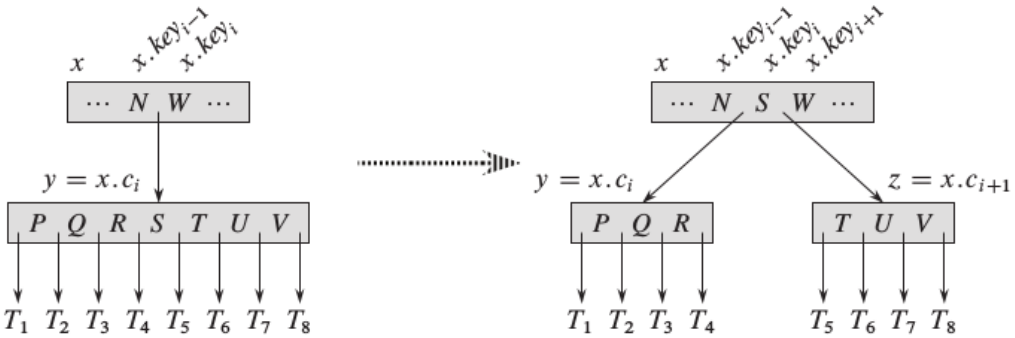


B-Tree Split Child



A Copy $t - 1$ items in new node

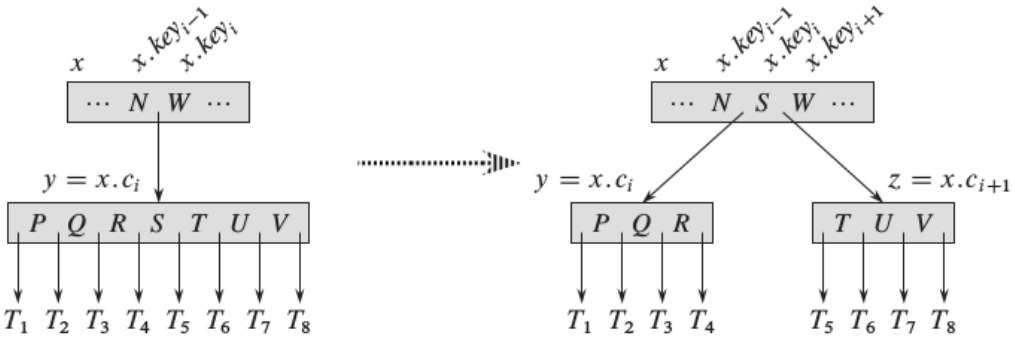
B-Tree Split Child



A Copy $t - 1$ items in new node

B Copy t pointer as well (if needed)

B-Tree Split Child

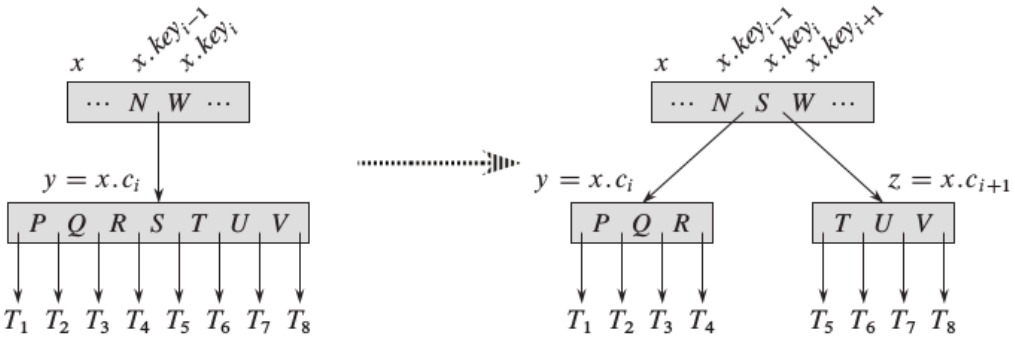


A Copy $t - 1$ items in new node

B Copy t pointer as well (if needed)

C Shift pointers in parent

B-Tree Split Child



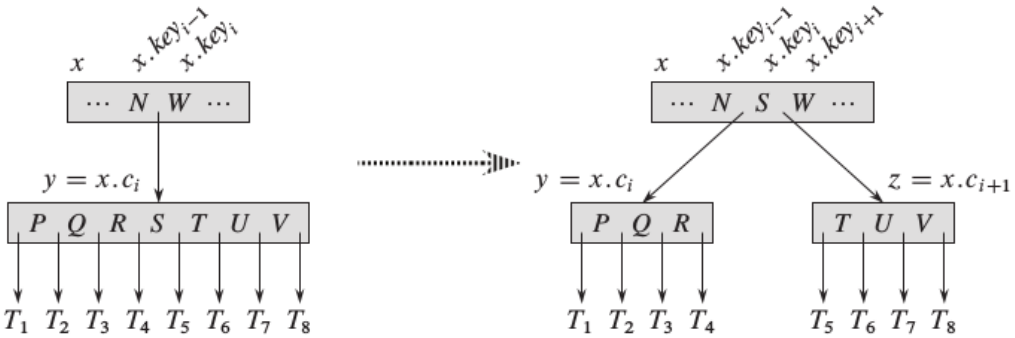
A Copy $t - 1$ items in new node

B Copy t pointer as well (if needed)

C Shift pointers in parent

D Shift Keys in parent

B-Tree Split Child



A Copy $t - 1$ items in new node

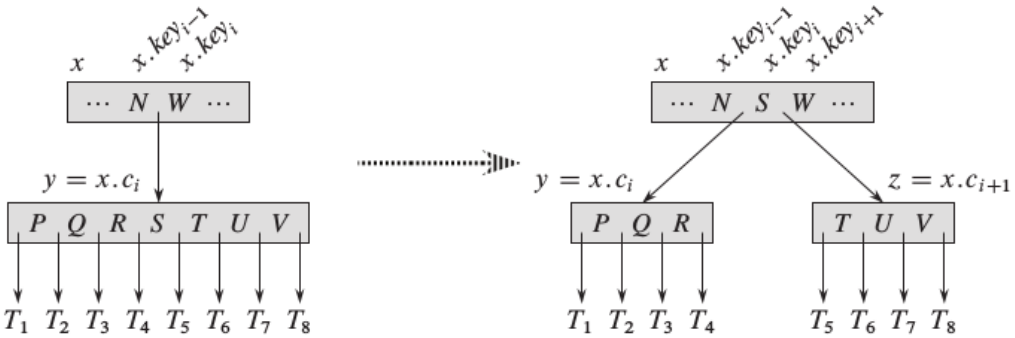
B Copy t pointer as well (if needed)

C Shift pointers in parent

D Shift Keys in parent

E Copy the key in parent

B-Tree Split Child



A Copy $t - 1$ items in new node

B Copy t pointer as well (if needed)

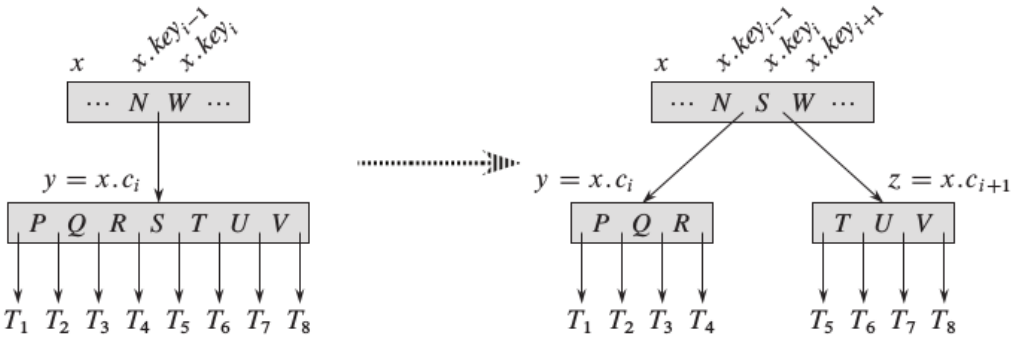
C Shift pointers in parent

D Shift Keys in parent

E Copy the key in parent

F Finalize

B-Tree Split Child



A Copy $t - 1$ items in new node

B Copy t pointer as well (if needed)

C Shift pointers in parent

D Shift Keys in parent

E Copy the key in parent

F Finalize

B-TREE-SPLIT-CHILD(x, i)

```
1  z = ALLOCATE-NODE()
2  y = x.c_i
3  z.leaf = y.leaf
4  z.n = t − 1
5  for j = 1 to t − 1
6        z.key_j = y.key_{j+t}
7  if not y.leaf
8        for j = 1 to t
9              z.c_j = y.c_{j+t}
10 y.n = t − 1
11 for j = x.n + 1 downto i + 1
12       x.c_{j+1} = x.c_j
13 x.c_{i+1} = z
14 for j = x.n downto i
15       x.key_{j+1} = x.key_j
16 x.key_i = y.key_t
17 x.n = x.n + 1
18 DISK-WRITE(y)
19 DISK-WRITE(z)
20 DISK-WRITE(x)
```

B-Tree Insert

- Item is inserted at leaf only

B-Tree Insert

- Item is inserted at leaf only
- Traverse till leaf by splitting any fill node encountered

B-Tree Insert

- Item is inserted at leaf only
- Traverse till leaf by splitting any fill node encountered

```
B-TREE-INSERT(T, k)
 1  r = T.root
 2  if r.n == 2t − 1
 3      s = ALLOCATE-NODE()
 4      T.root = s
 5      s.leaf = FALSE
 6      s.n = 0
 7      s.c_1 = r
 8      B-TREE-SPLIT-CHILD(s, 1)
 9      B-TREE-INSERT-NONFULL(s, k)
10  else B-TREE-INSERT-NONFULL(r, k)
```

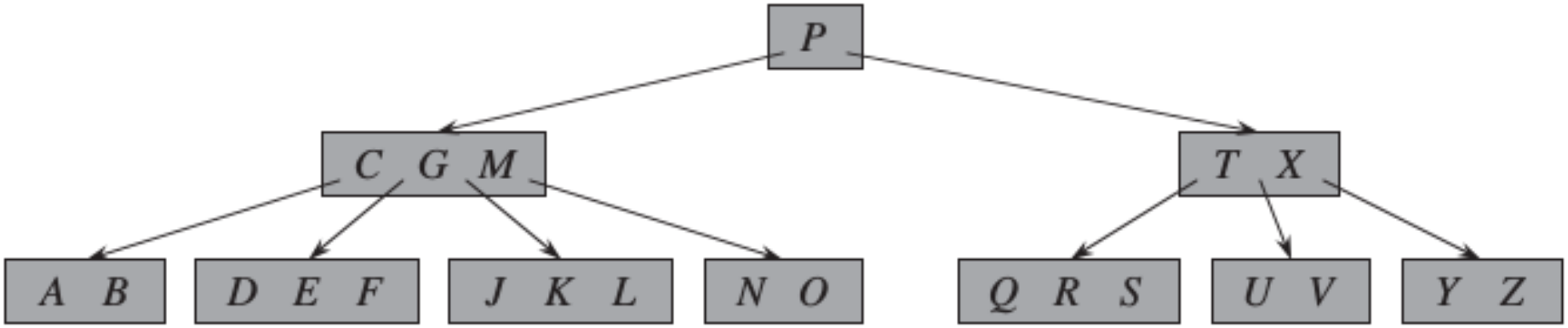
B-Tree Insert

- Item is inserted at leaf only
- Traverse till leaf by splitting any fill node encountered

```
B-TREE-INSERT(T, k)
 1  r = T.root
 2  if r.n == 2t − 1
 3       s = ALLOCATE-NODE()
 4       T.root = s
 5       s.leaf = FALSE
 6       s.n = 0
 7       s.c_1 = r
 8       B-TREE-SPLIT-CHILD(s, 1)
 9       B-TREE-INSERT-NONFULL(s, k)
10  else B-TREE-INSERT-NONFULL(r, k)
```

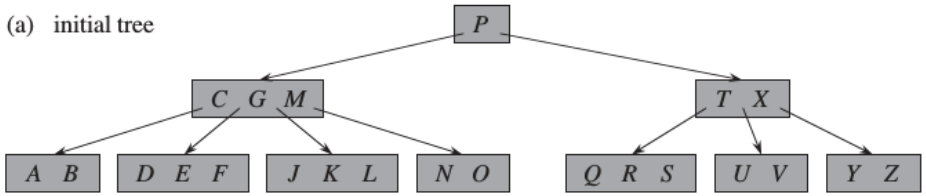
```
B-TREE-INSERT-NONFULL(x, k)
 1  i = x.n
 2  if x.leaf
 3       while i ≥ 1 and k < x.key_i
 4            x.key_{i+1} = x.key_i
 5            i = i − 1
 6       x.key_{i+1} = k
 7       x.n = x.n + 1
 8       DISK-WRITE(x)
 9  else while i ≥ 1 and k < x.key_i
10            i = i − 1
11       i = i + 1
12       DISK-READ(x.c_i)
13       if x.c_i.n == 2t − 1
14            B-TREE-SPLIT-CHILD(x, i)
15            if k > x.key_i
16                 i = i + 1
17       B-TREE-INSERT-NONFULL(x.c_i, k)
```

B-Tree Delete

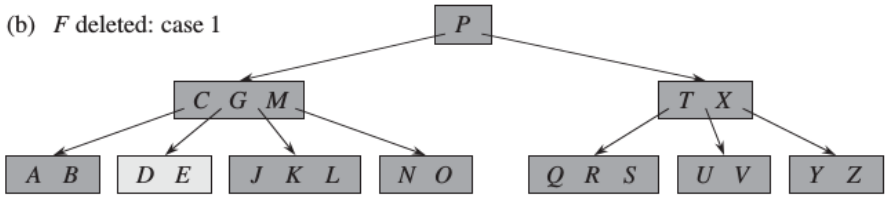


B-Tree Delete

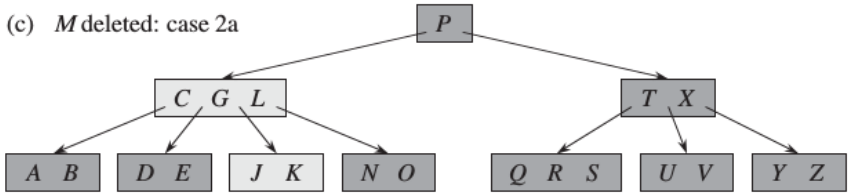
(a) initial tree



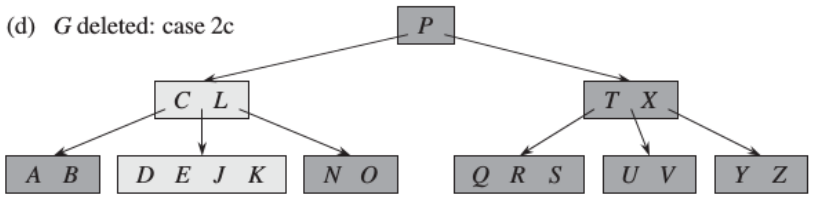
(b) *F* deleted: case 1



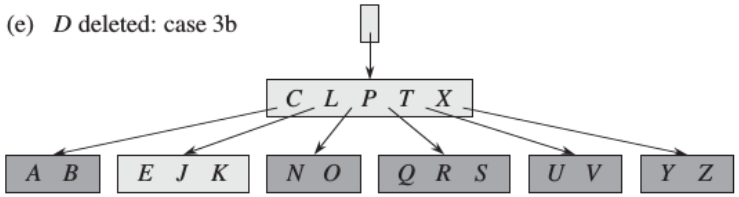
(c) *M* deleted: case 2a



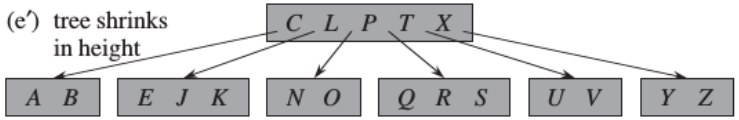
(d) *G* deleted: case 2c



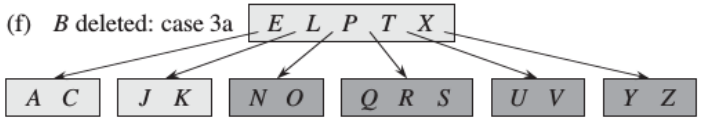
(e) *D* deleted: case 3b



(e′) tree shrinks in height



(f) *B* deleted: case 3a

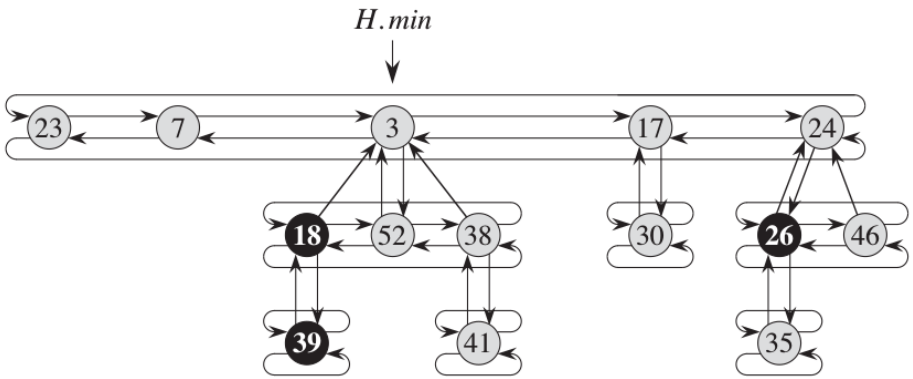


Fibonacci Heaps

- It supports **mergeable heaps** operations

Procedure	Binary Heap (worst case)	Fibonacci Heap (amortized)
MAKE-HEAP	$\Theta(1)$	$\Theta(1)$
INSERT	$\Theta(\log n)$	$\Theta(1)$
MINIMUM	$\Theta(1)$	$\Theta(1)$
EXTRACT-MIN	$\Theta(\log n)$	$\Theta(\log n)$
UNION	$\Theta(n)$	$\Theta(1)$
DECREASE-KEY	$\Theta(\log n)$	$\Theta(1)$
DELETE	$\Theta(\log n)$	$\Theta(\log n)$

- Several Fibonacci-heap operations run in constant amortized time
- Circular, doubly linked lists

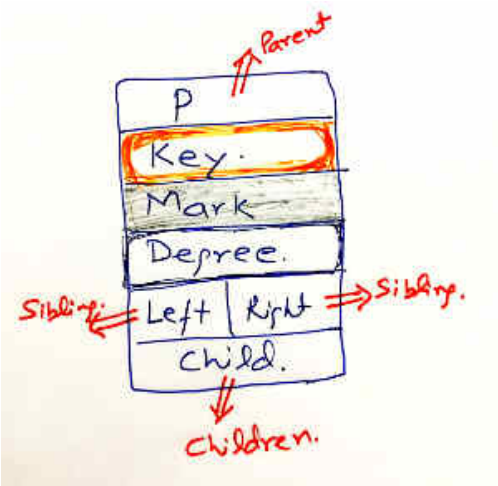


Fibonacci Heap Insertion

- Heap H has two values $n[H]$ and $min[H]$

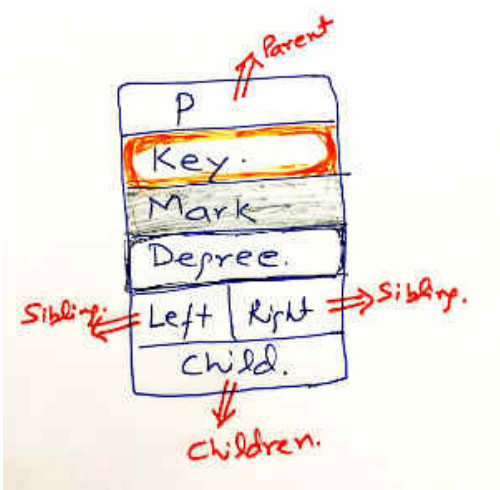
Fibonacci Heap Insertion

- Heap H has two values $n[H]$ and $min[H]$
- Other nodes have



Fibonacci Heap Insertion

- Heap H has two values $n[H]$ and $min[H]$
- Other nodes have

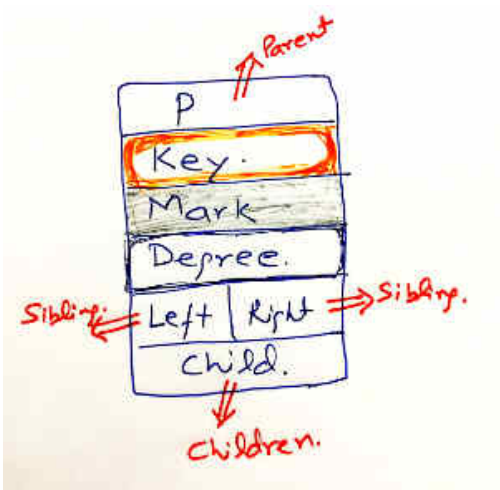


FIB-HEAP-INSERT(H, x)

```
1  x.degree = 0
2  x.p = NIL
3  x.child = NIL
4  x.mark = FALSE
5  if H.min == NIL
6       create a root list for H containing just x
7       H.min = x
8  else insert x into H's root list
9       if x.key < H.min.key
10           H.min = x
11 H.n = H.n + 1
```

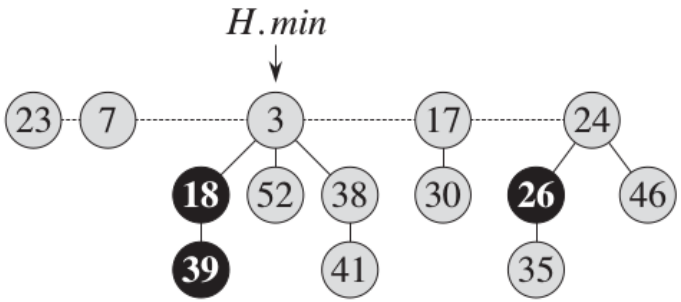
Fibonacci Heap Insertion

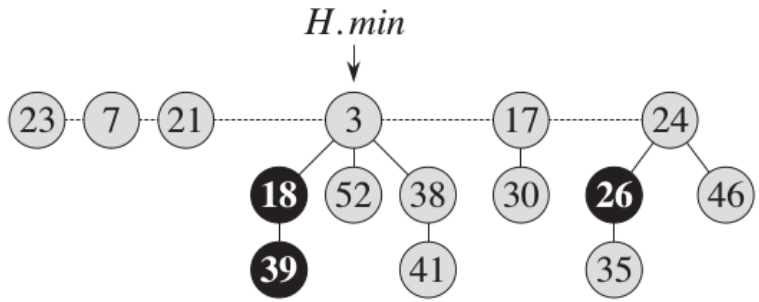
- Heap H has two values $n[H]$ and $min[H]$
- Other nodes have



FIB-HEAP-INSERT(H, x)

```
1   x.degree = 0
2   x.p = NIL
3   x.child = NIL
4   x.mark = FALSE
5   if H.min == NIL
6       create a root list for H containing just x
7       H.min = x
8   else insert x into H's root list
9       if x.key < H.min.key
10          H.min = x
11  H.n = H.n + 1
```





Find minimum is $O(1)$ operation

Thank You!

Thank you very much for your attention!

Queries ?