Insertion into a B-tree.

(We assume search keys are unique, i.e.: key attribute(s)).

Insert Algorithm: \( \text{Insert } x \).

1. Find the leaf node \( L \) where search key \( x \) belongs.
2. If leaf node \( L \) has an empty slot:
   a. Shift keys to make room for \( x \)
   b. Insert \( x \) and the record address of \( x \)

\[ \text{DONE.} \]

3. If leaf node \( L \) is full:
   See next page.
(3) If leaf node \( L \) is full:

- If \( L \) must have \( n \) keys and \( n+1 \) pointers, including \((x, \text{pointers to } x)\), we have:

  \[
  k_1, k_2, \ldots, x, \ldots, k_n
  \]

  We have: \( n+1 \) keys.

  \( n+2 \) pointers.

- Split the node in halff:

  (See next page)

  \[
  m = \left\lceil \frac{n}{2} \right\rceil
  \]

  \[
  \begin{cases} 
  n=3, m=2 \\
  n=4, m=2 \\
  n=5, m=3 \\
  n=6, m=3.
  \end{cases}
  \]
B) **split** the node in half:

\[ L \]

\[ \begin{array}{c}
L_1 = L \\
\end{array} \]

\[ \begin{array}{c}
L_2 \\
\end{array} \]

\[ \begin{array}{c}
k_1 \ldots \ldots k_m \\
\end{array} \]

\[ \begin{array}{c}
A \ \ B \ \ C \\
\end{array} \]

\[ \begin{array}{c}
k_{m+1} \ldots \ldots k_n \\
\end{array} \]

\[ \begin{array}{c}
E \ \ F \\
\end{array} \]

Create new node \( \text{split HERE} \).

Move to start + copy keys + pointers.

Make the IML!!!

You will gain one extra pointer!!

C) **Insert** \((k_{m+1}, L_2)\) into the parent node of \(L\).

(The parent node can split '!!!

→ cascade to leaf '!!!)
Insert case 1: leaf node has space

Insert (10, ptr(10))

\[ 13 \]

\[ 7 \]

\[ 23 \ 31 \ 43 \]

\[ 2 \ 3 \ 5 \ 7 \ 11 \ 13 \ 17 \ 29 \ 23 \ 29 \ 31 \ 37 \ 41 \ 43 \ 47 \]

\[ \downarrow 10 \leq 7 \]

\[ \downarrow 10 < 13 \]

\[ \uparrow \]

\[ \text{insert (10, ptr(10)) here.} \]

\[ \text{Result:} \]

\[ 13 \]

\[ 7 \]

\[ 10 \]

\[ 11 \]

\[ \text{use board!} \]

\[ \text{ptr(10)} \]
Insert case 2: cascade level

Insert(4)

L

2 3 5

7 11

A B C

Insert(4) here.

13 17 29

23 29

31 37 41

43 47

4 < 13

4 < 7

13 17 29

23 29

31 37 41

43 47

(1) Initial state:

L

1 2 3 4 5

A B C

Create new node + copy keys + pointers:

L' 4 5 1

New Link!!!

(2) After split:

L

1 2 3

A B

(3) Insert (4, L') in parent at L.

before:

L

2 3

17

4 5

L'

after:

L

4 17

L'

4 5

1 2 3

17

4 5

10/8/2014 11:35 AM
Result:
Insert Case 3: Cascade Insert Multiple Levels

\[
\text{Insert } (40, \text{ptr}(40))
\]

1. Initial state:

![Initial state diagram](image)

2. After split:

![After split diagram](image)

3. Insert \((40, L')\) in parent of \(L\).

![Insertion diagram](image)
4) Initial state after inserting \((40, L')\):

```
I   L   L'   X
| 23 | 31 | 40 | 43 |

I  L  L'  X
| 31 | 37 | 40 | 41 |
    ↓  ↓  ↓  ↓
    A  B  C  D
```

Split internal node is different - (different procedure)

Extract and split:

```
I
| 23 | 31 | 40 | 43 |

I
| 23 | 31 |
```

After split:

```
I
| 23 | 31 |

I'
| 43 |
```

\((40, L')\) is inserted in parent of I."
Worked out example

(Show with board!?)
After insert, before split:
Propagate: \((40, L')\) to parent

After split:
After insert, before split: 

(40, L') inserted
Split internal node:

1. 13
2. 23 3 40 43
3. 23 29
4. 7 11
5. 2 3 5

40 + pointer to right child

(3 way split)
After insert:

DONE !!!