# Pertemuan 14
# B-Tree

# Learning Outcomes

**Pada akhir pertemuan ini, diharapkan mahasiswa akan mampu :**

- **menerangkan operasi pada B - Tree**
- **membuat representasi B-Tree**

# Outline Materi

- Karakteristik
- Pencarian elemen
- Insert
- Delete

# Karakteristik

**B-TREE adalah TREE yang setiap nodenya bisa berisi lebih dari pada satu elemen.**

**Jumlah elemen dalam 1 node tergantung kepada order (*d*) dari B-Tree tersebut.**

**Jumlah minimum elemen dalam setiap node (kecuali ROOT) adalah *d*, dan jumlah maksimumnya adalah 2*d*, dimana *d* adalah order. Untuk ROOT, jumlah minimum elemen 1, dan jumlah maksimum adalah 2*d*.**

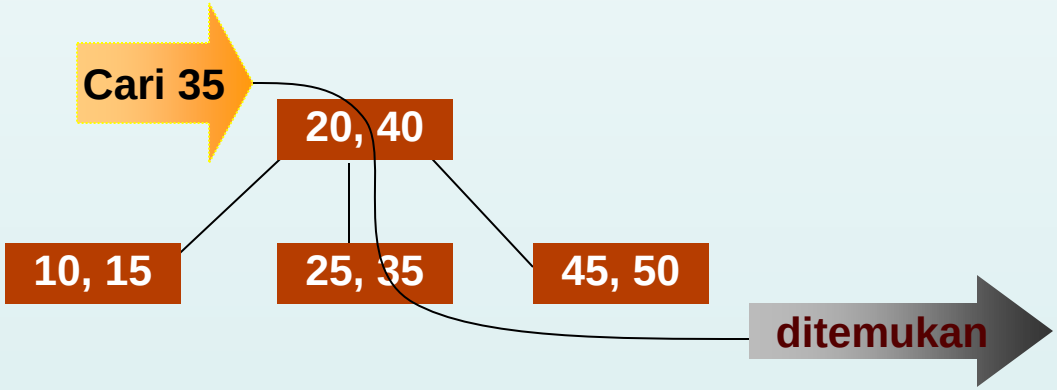
**Jumlah minimum children dari suatu node dalam B-TREE adalah 0, dan jumlah maksimumnya adalah jumlah elemen + 1.**

# Pencarian Elemen

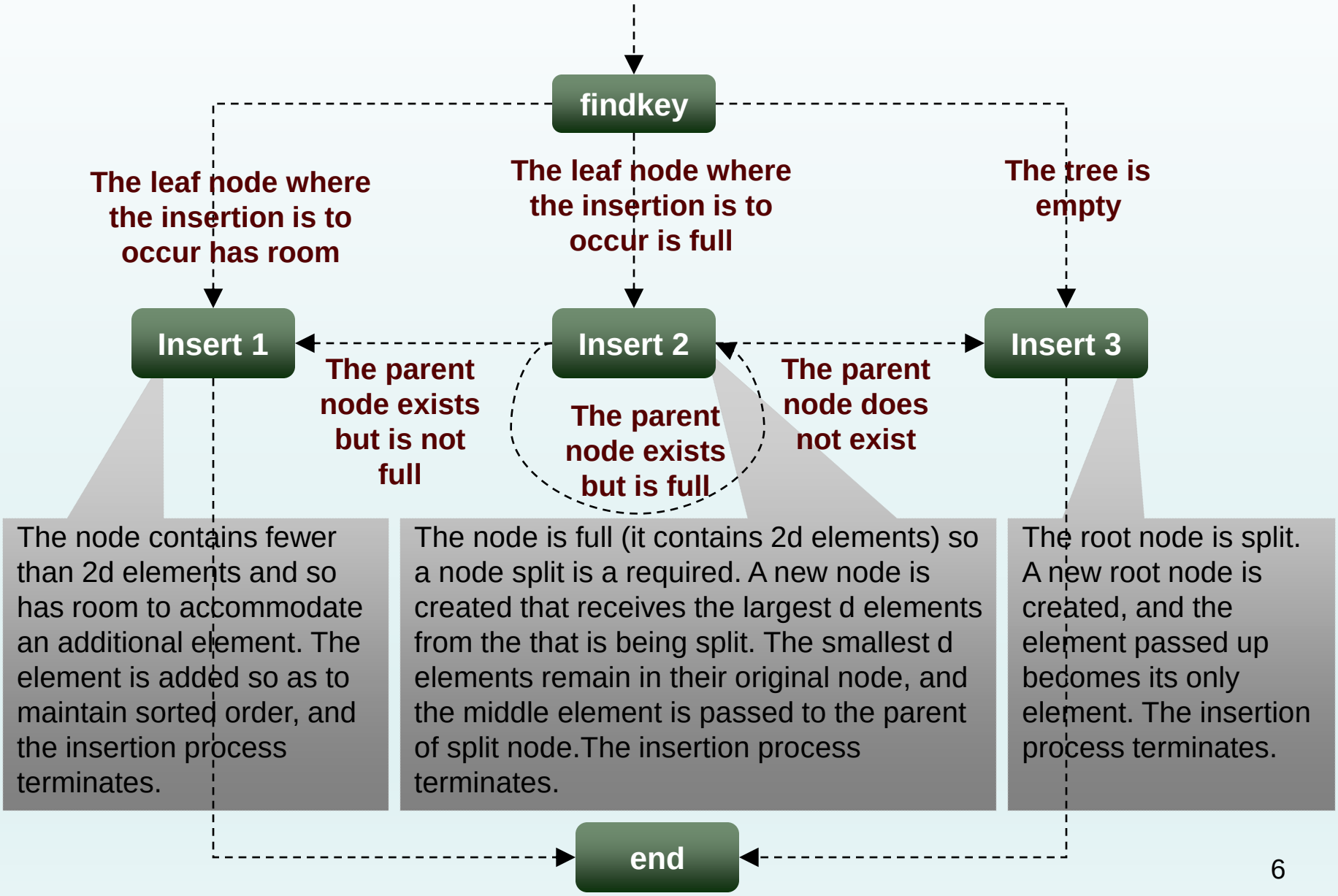
**Pencarian elemen pada B-tree hampir sama dengan proses pencarian elemen pada BST.**

**Proses pencarian hanya membutuhkan single path dari ROOT hingga LEAF.**

**Contoh :**



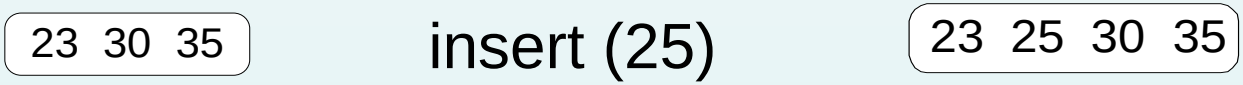
# Operasi Insert



# Contoh Operasi Insert

**Jika node belum penuh (jumlah elemen < 2d), maka elemen bisa langsung diinsert.**

**Contoh :**



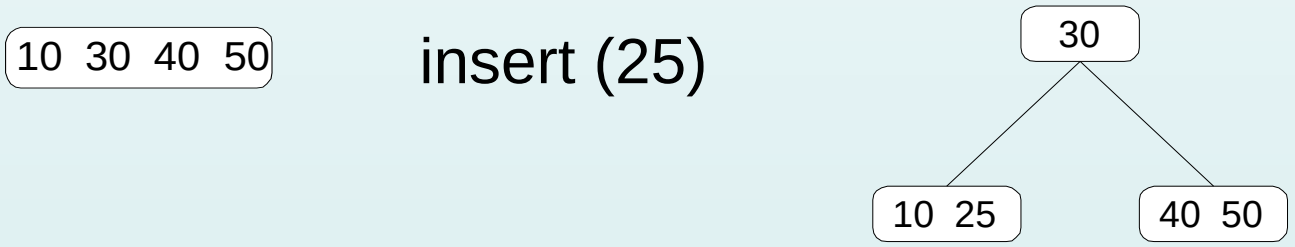
# Contoh Operasi Insert(2)

**Jika node sudah penuh, maka perlu dilakukan Node Split.**

**Langkah Node Split:**

- **Split node menjadi 2.**
- **Letakkan d elemen terkecil di node kiri.**
- **Letakkan d elemen terbesar di node kanan.**
- **Letakkan elemen tengah ke node parentnya.**

**Contoh :**



# Operasi Delete

- Sebagaimana proses insert, proses delete dimulai dari leaf node.
- Jika elemen yang dihapus tidak ada di dalam leaf node, akan diganti dengan :
  - Elemen terkecil node paling kiri dari subtree kanan, atau
  - Elemen terbesar node paling kanan dari subtree kiri
- Pergantian elemen ini memerlukan perpindahan elemen dari leaf node.

# Mekanisme Delete

**Target node adalah node dimana terdapat elemen yang akan dihapus.**

- **Delete 1 :**

  **Target node memiliki lebih dari *d* elemen. Setelah penghapusan elemen proses langsung selesai.**

- **Delete 2 :**

  **Target node memiliki tepat *d* elemen, sehingga penghapusan elemen akan menyebabkan *underflow*. Jika ada sibling yang memiliki lebih dari d elemen, maka dilakukan peminjaman elemen dari sibling tsb dengan melibatkan parent.**

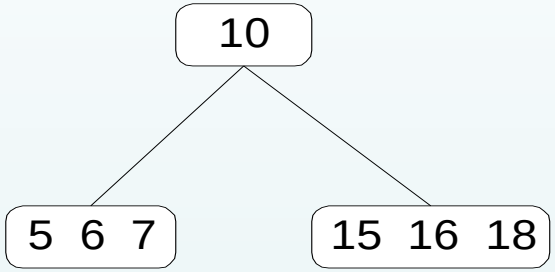
  **Jika tidak, maka target node bergabung dengan salah satunya untuk membentuk node baru yang mengandung 2d elemen. Salah satu elemennya tsb berasal dari parent.**

  **Jika parent mengandung lebih dari *d* elemen maka proses selesai, selain itu parent menjadi target node dan proses berlanjut.**

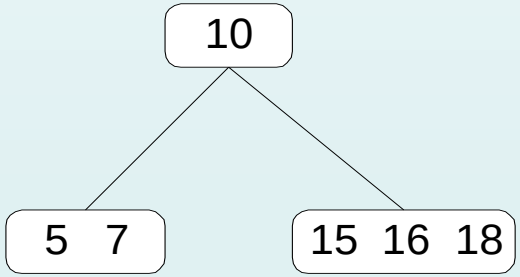
- **Delete 3 :**

  **Target node adalah root. Selama masih tersisa minimal 1 elemen, tinggi dari tree tidak berubah. Jika lelemn terakhir dihapus, maka anak yang tersisa (child node)**

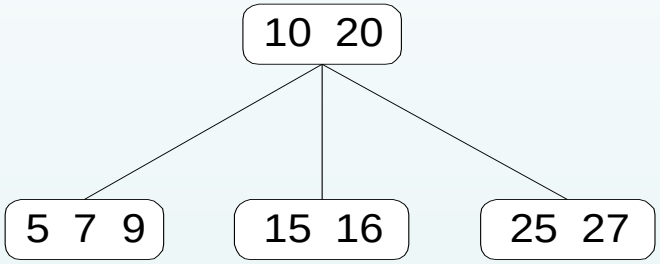
# Contoh Operasi Delete



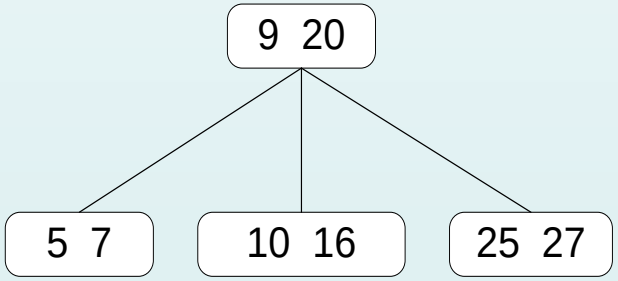

delete (6)



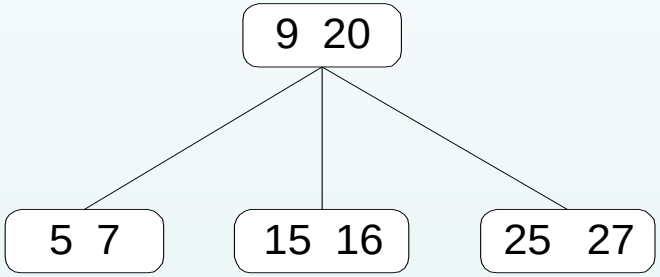
# Contoh Operasi Delete(2)



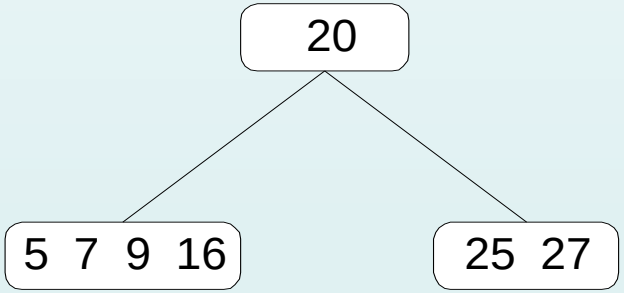

delete(15)



# Contoh Operasi Delete(3)




delete(15)



# Latihan

**Jika + berarti Insert, maka buatlah B - Tree Orde 2 dari operasi-operasi berikut ini :**

**+ 76 + 35 + 90 + 50 + 25**

**+ 150 + 100 + 80 + 40 + 50**

**+ 125 + 60 + 75 + 140 + 55**

**+ 140 + 20 + 120 + 66 + 82**

# Latihan

**Jika + berarti Insert, maka buatlah B - Tree Orde 2 dari operasi-operasi berikut ini :**

| | | | | |
|---|---|---|---|---|
| + 76 | + 35 | + 90 | + 50 | + 25 |
| + 70 | + 30 | + 80 | + 40 | + 57 |
| + 15 | + 60 | + 75 | + 45 | + 55 |
| + 87 | + 29 | + 65 | + 86 | + 72 |

# Latihan

**Jika + berarti Insert,**

**dan – berarti Delete, maka**

**buatlah B - Tree Orde 2 dari**

**Operasi - operasi berikut ini :**

**+ 76 + 35 + 90 + 50 + 25**

**+ 150 + 100 + 80 + 40 - 50**

**+ 125 + 60 - 76 + 140 + 55**

**+ 140 - 25 + 120 - 60 - 80**

# Selesai