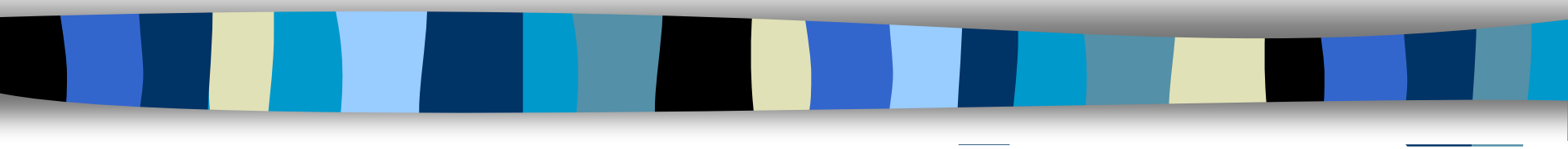
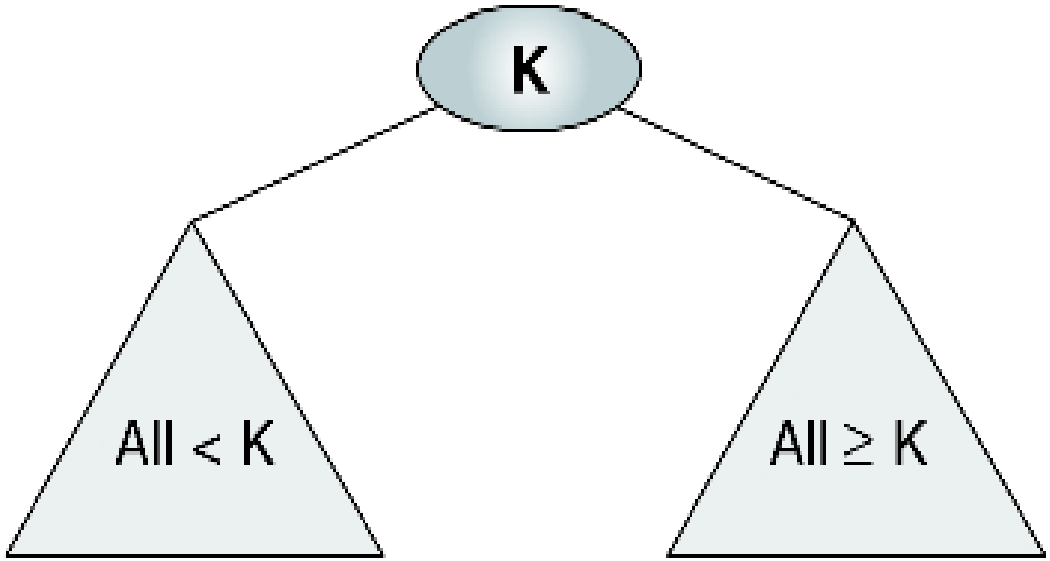
Algoritma dan Struktur Data



AVL TREE

Review BST

Apakah Binary Search Tree itu ?

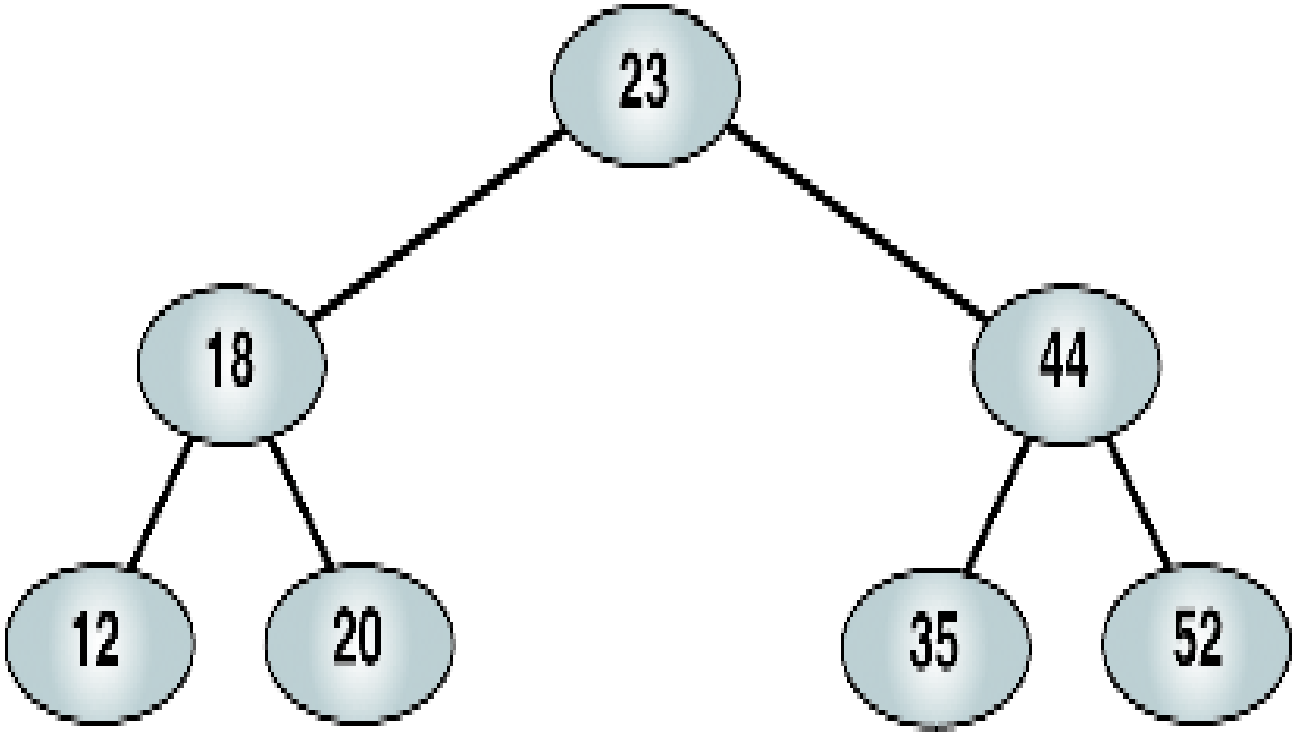


- Pemakaian tree structure dalam proses pencarian (search)
- Sifat Binary Tree:

  Pada sebuah node x,

  – elemen yang berada di LEFT sub-tree selalu lebih KECIL daripada x
  – elemen yang berada di RIGHT sub-tree selalu lebih BESAR Atau SAMA DENGAN daripada x
- Binary Search Tree: proses pencarian (SEARCHING) berbasis binary tree

Example of a binary search tree

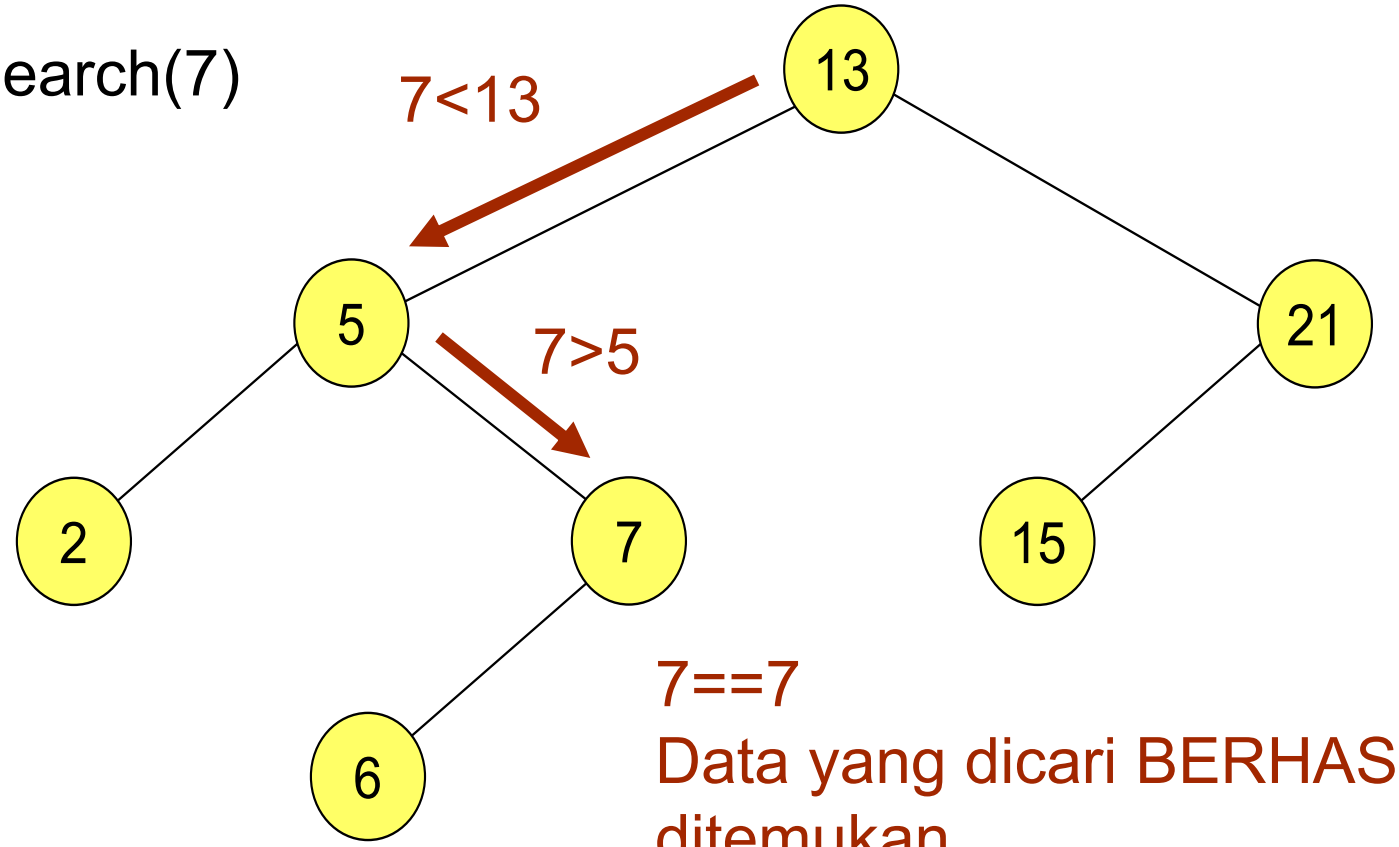


LATIHAN BST

- Coba simulasikan penambahan pada sebuah BST dengan urutan penambahan:
  - 14, 16, 10, 8, 12, 4, 9, 1
  - 8, 12, 4, 9, 1, 14, 16, 10

Proses Pencarian (contoh SUKSES)

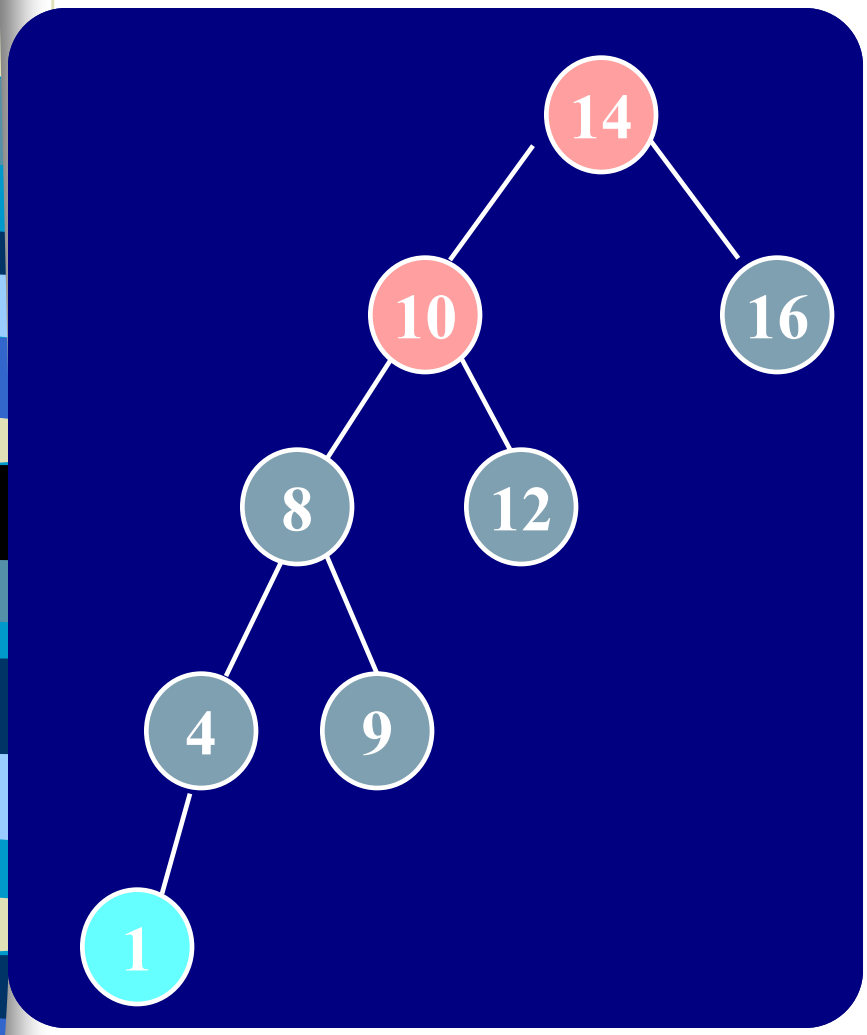
search(7)

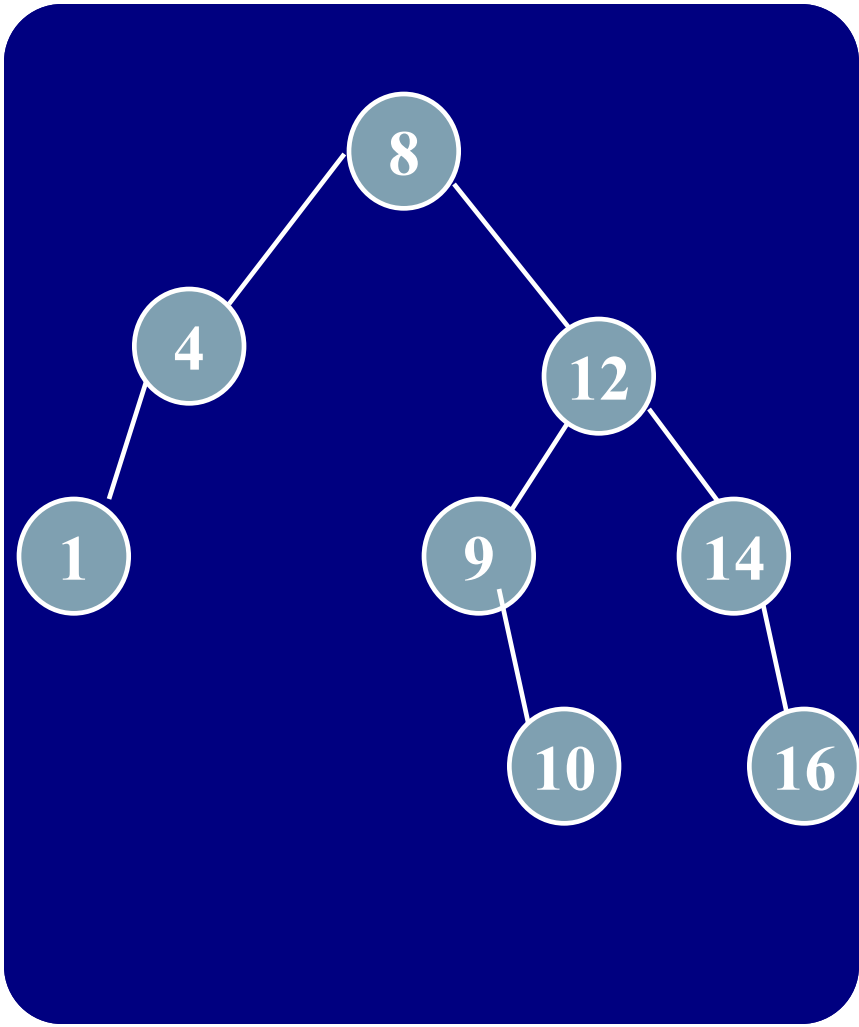


7==7
Data yang dicari BERHASIL ditemukan

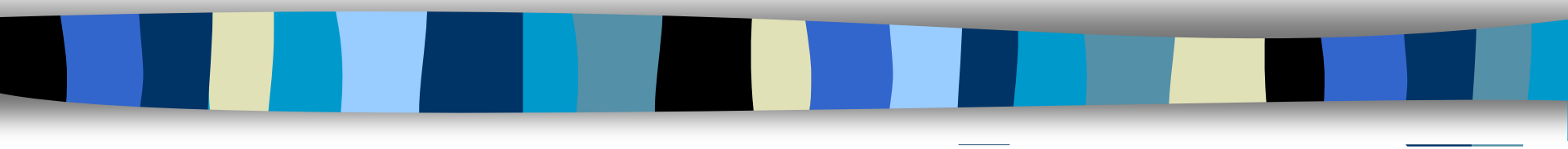
13→5 →7

LATIHAN BST





Algoritma dan Struktur Data



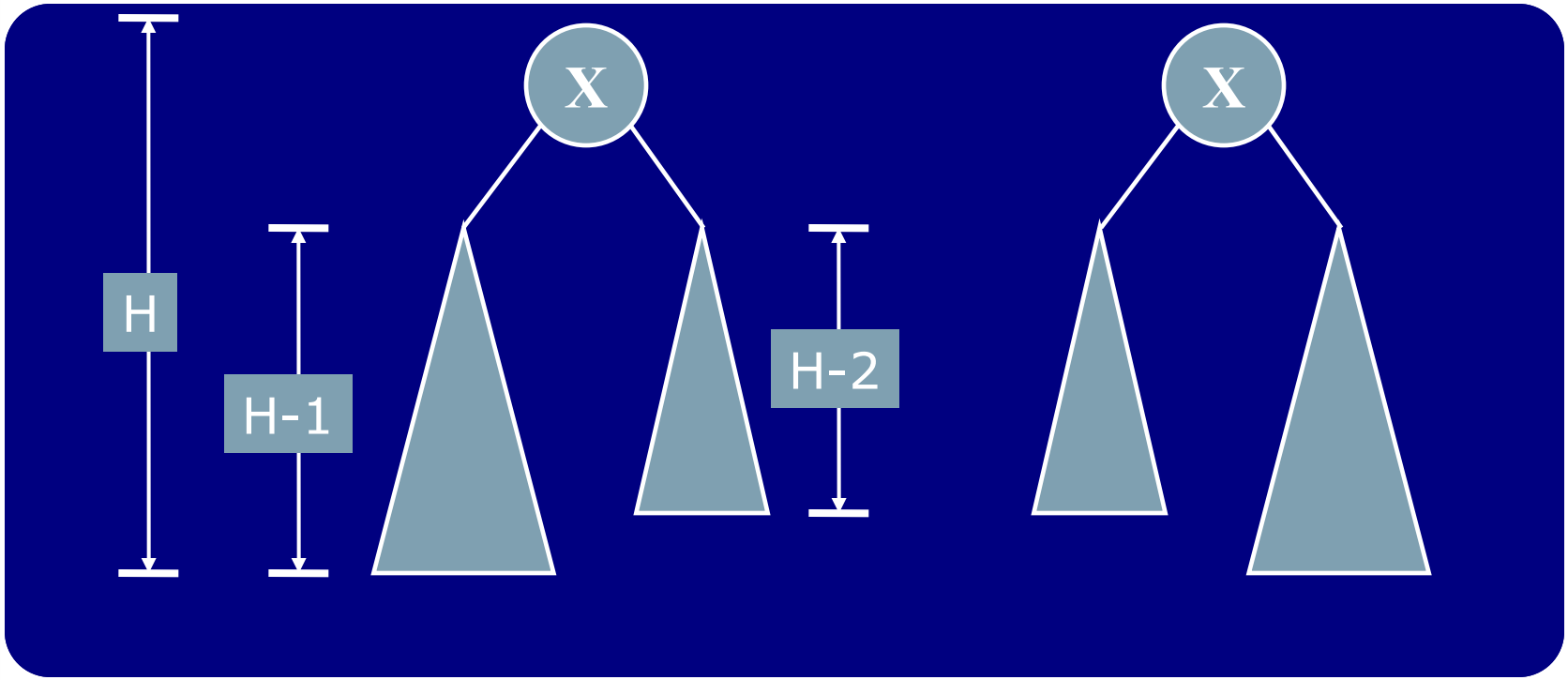
AVL TREE

Tujuan

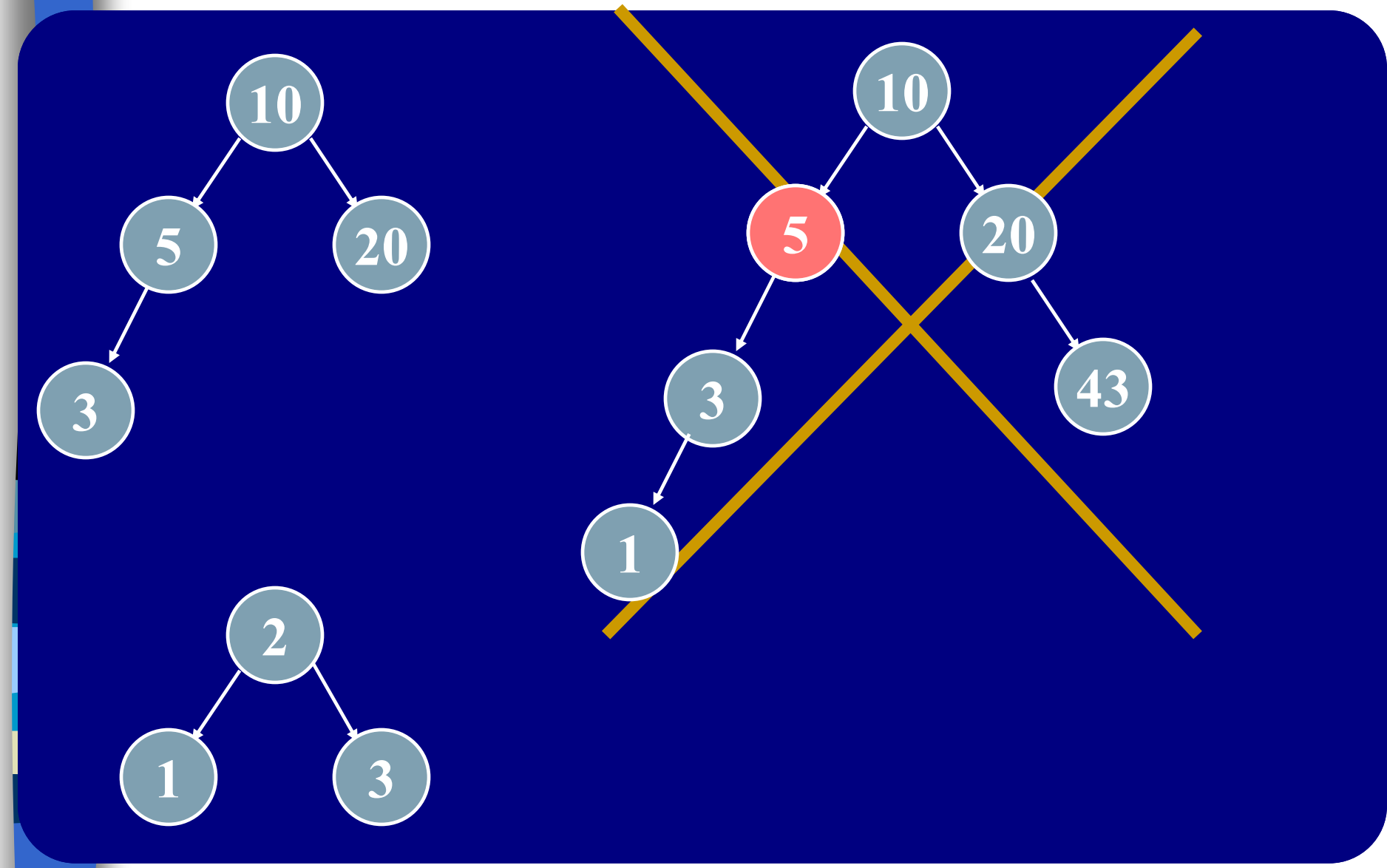
- Memahami variant dari Binary Search Tree yang balanced
- Binary Search Tree yang tidak *balance* dapat membuat seluruh operasi memiliki kompleksitas running time O(n) pada kondisi worst case.

AVL Trees

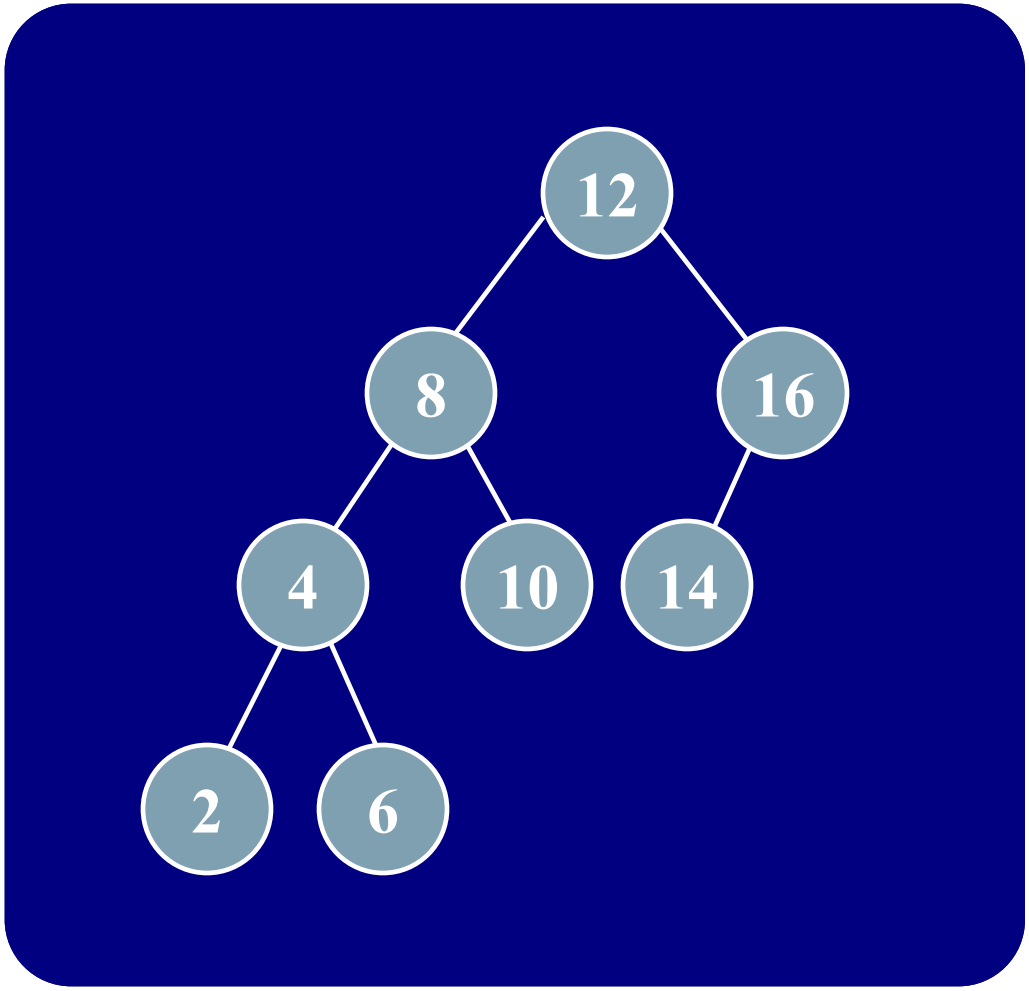
- Untuk setiap node dalam tree, ketinggian subtree di anak kiri dan subtree di anak kanan hanya berbeda maksimum 1.



AVL Trees

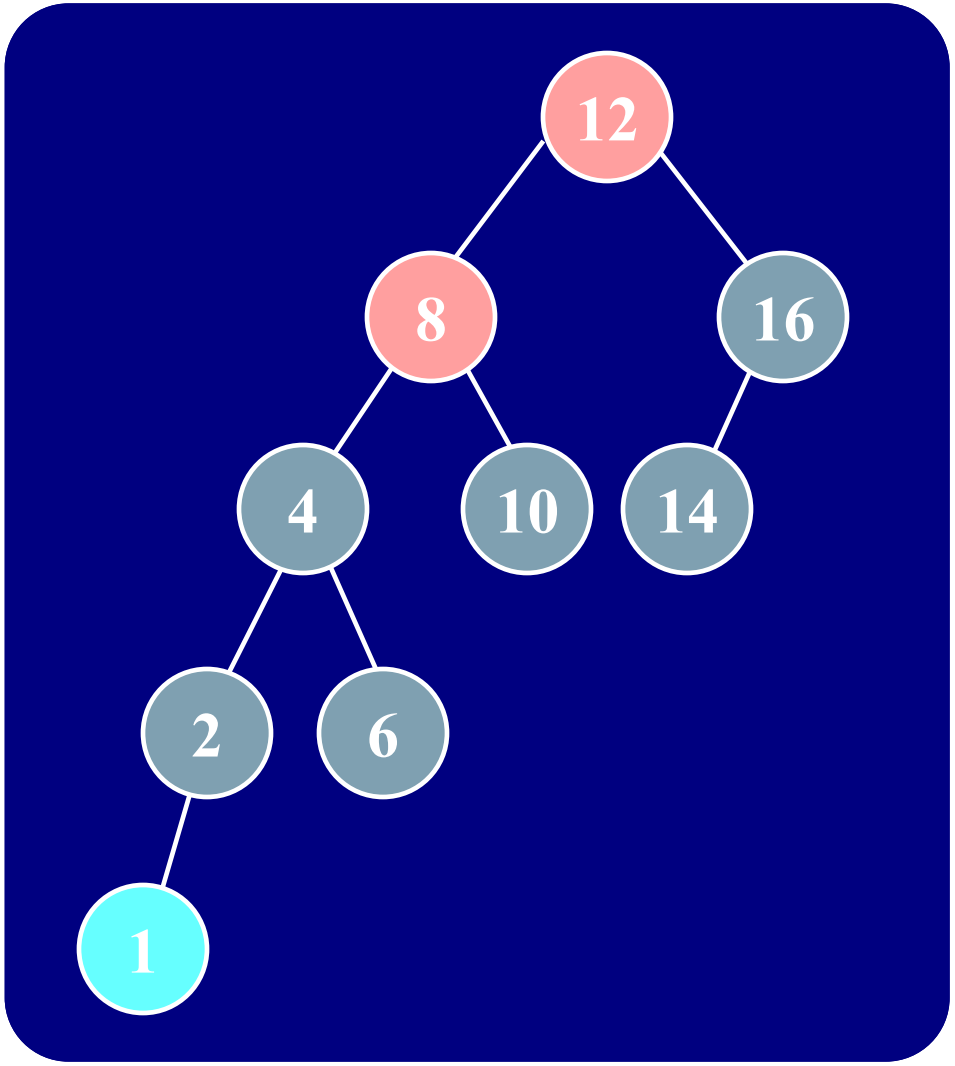


AVL Trees



Insertion pada AVL Tree

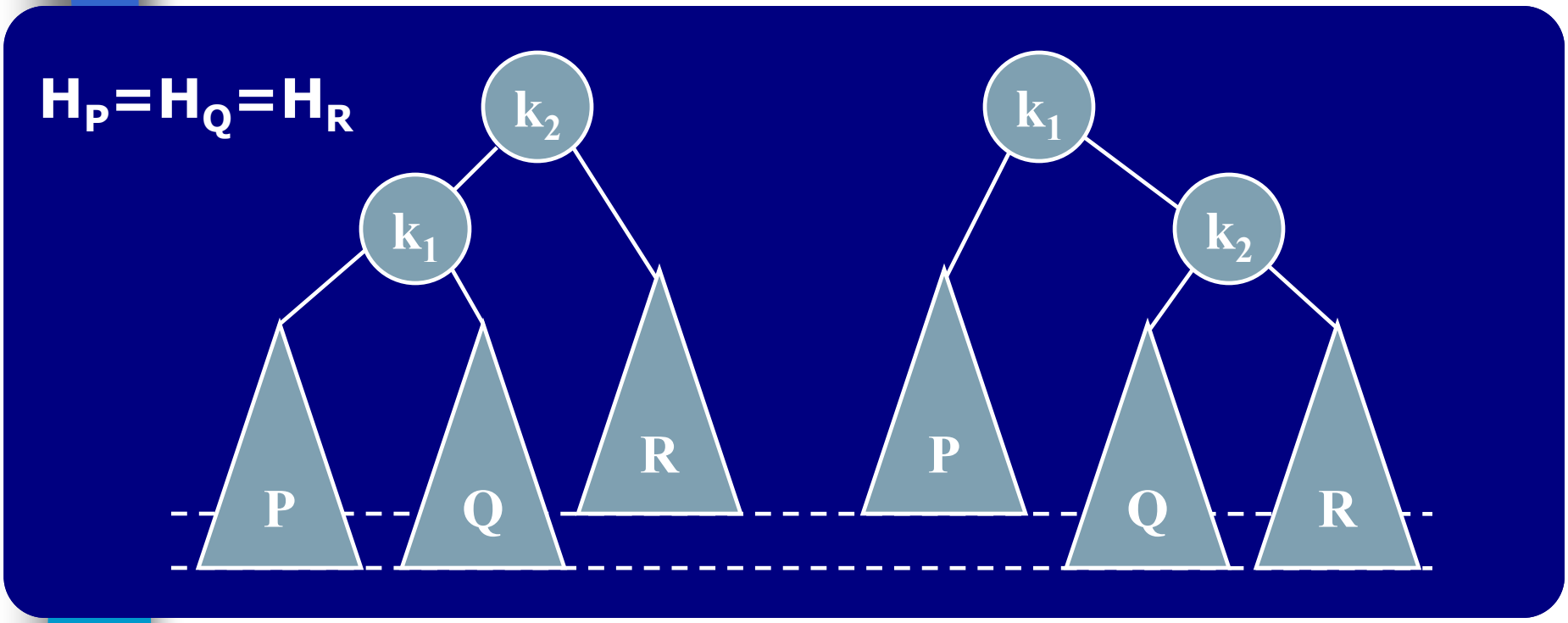
- Setelah *insert* 1



Insertion pada AVL Tree

- Untuk menjamin kondisi *balance* pada AVL tree, setelah penambahan sebuah node. jalur dari node baru tersebut hingga root di simpan dan di periksa kondisi *balance* pada tiap node-nya.
- Jika setelah penambahan, kondisi *balance* tidak terpenuhi pada node tertentu, maka lakukan salah satu rotasi berikut:
  - *Single rotation*
  - *Double rotation*

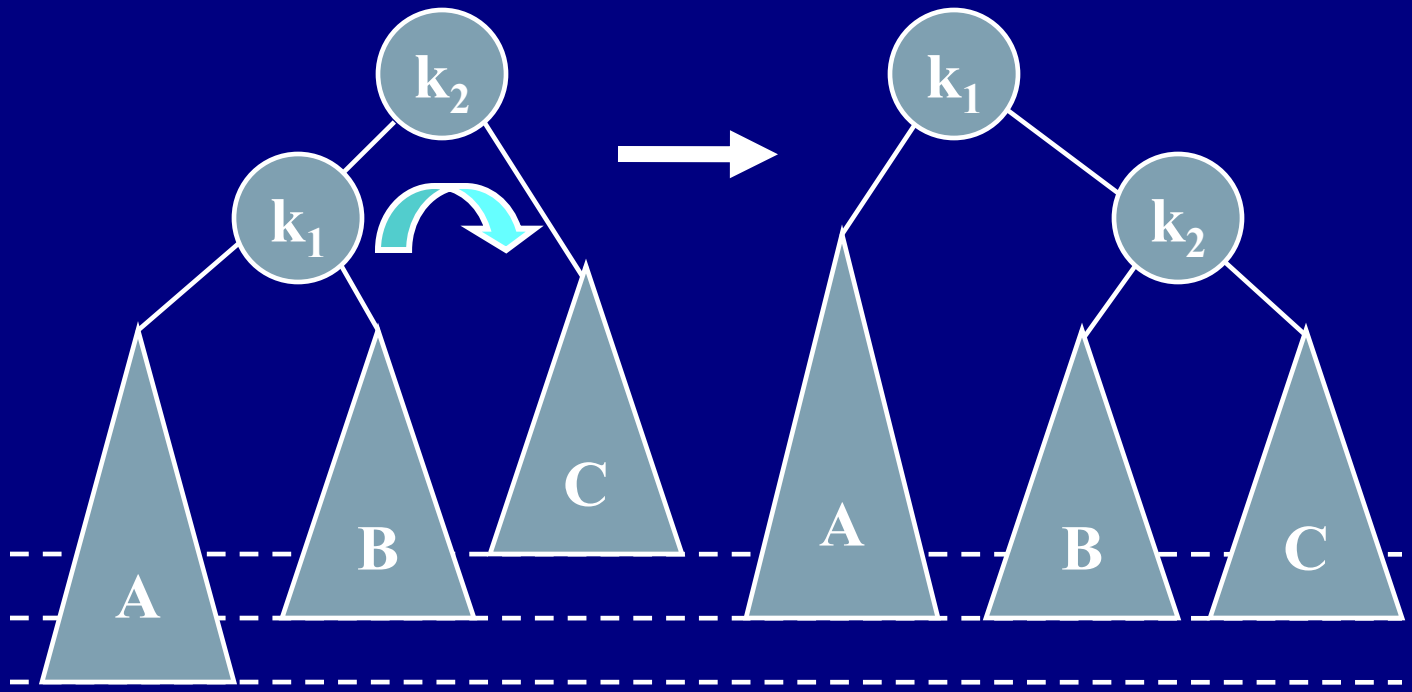
Kondisi tidak *balance*



- Sebuah penambahan pada subtree:
  - P (outside) - case 1
  - Q (inside) - case 2
- Sebuah penambahan pada subtree:
  - Q (inside) - case 3
  - R (outside) - case 4

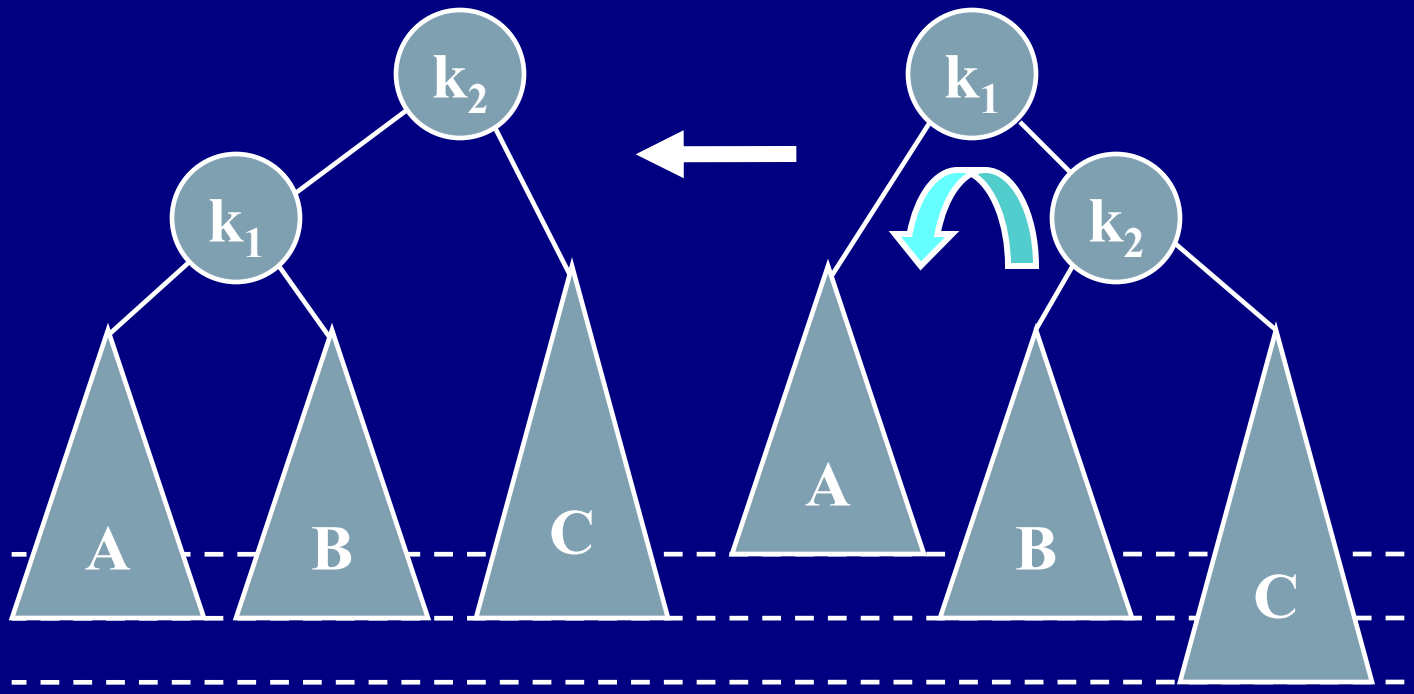
Single Rotation (case 1)

$H_A = H_B + 1$
$H_B = H_C$



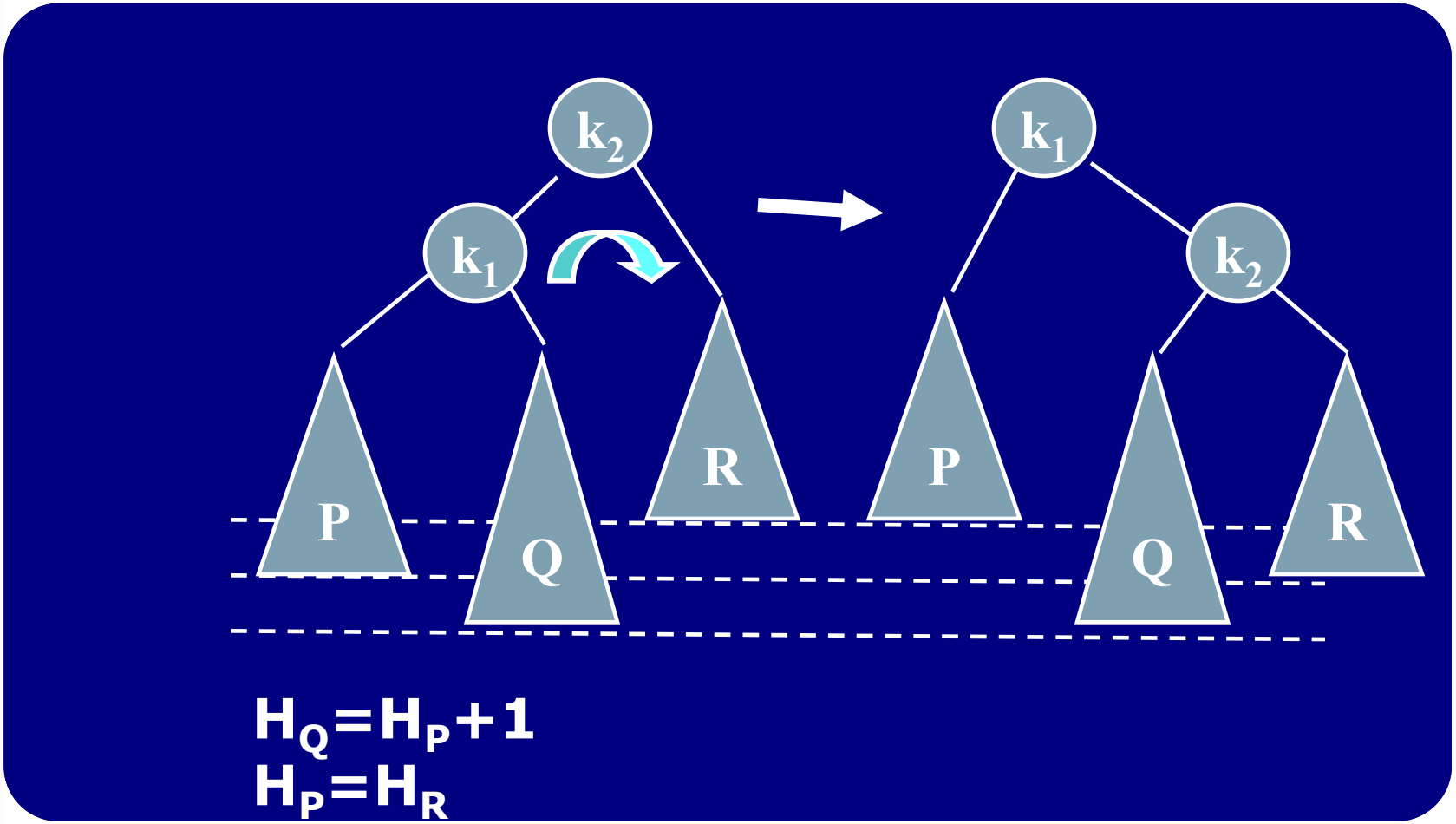
Single Rotation (case 4)

$H_A = H_B$
$H_C = H_B + 1$

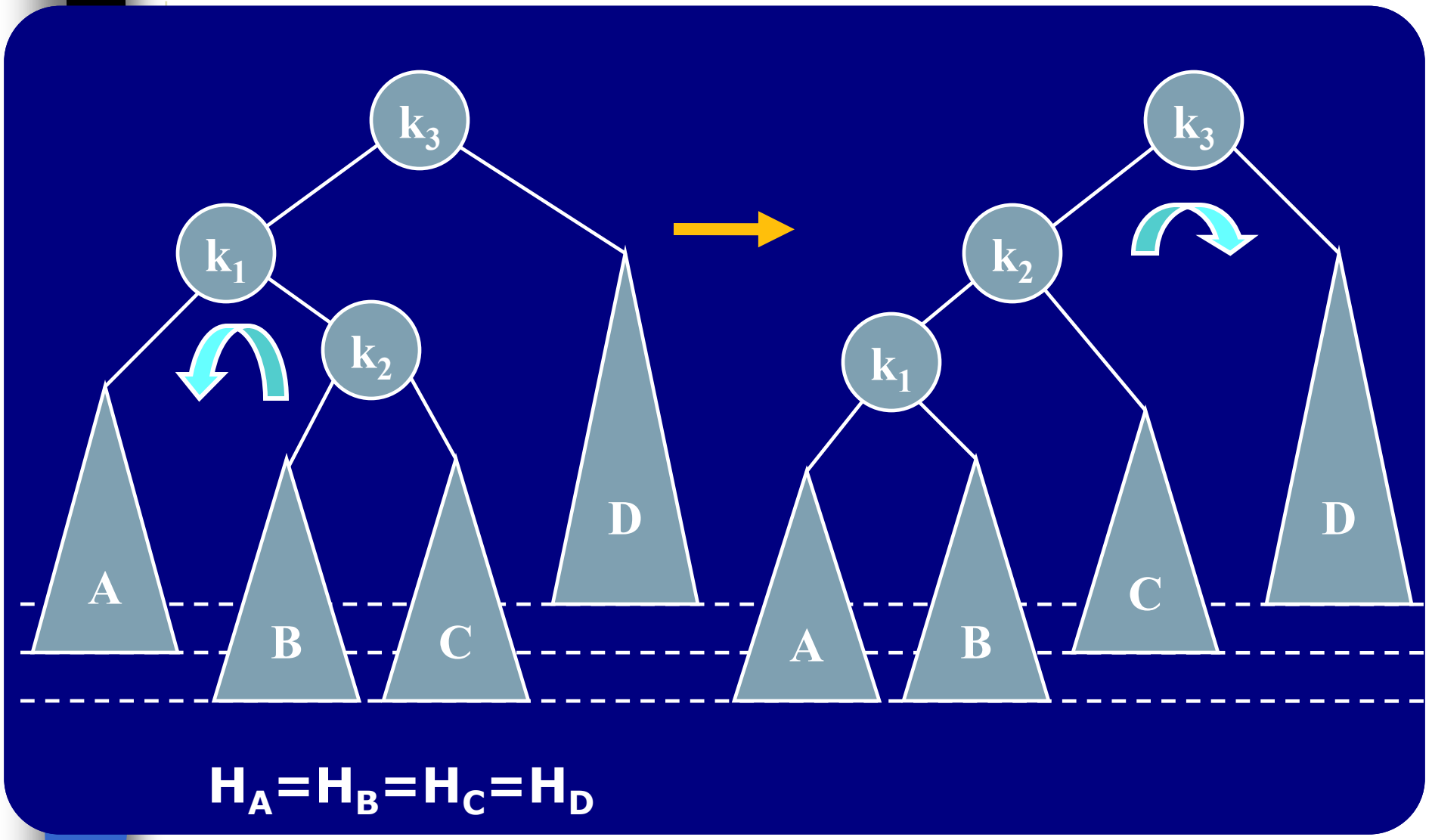


Keterbatasan Single Rotation

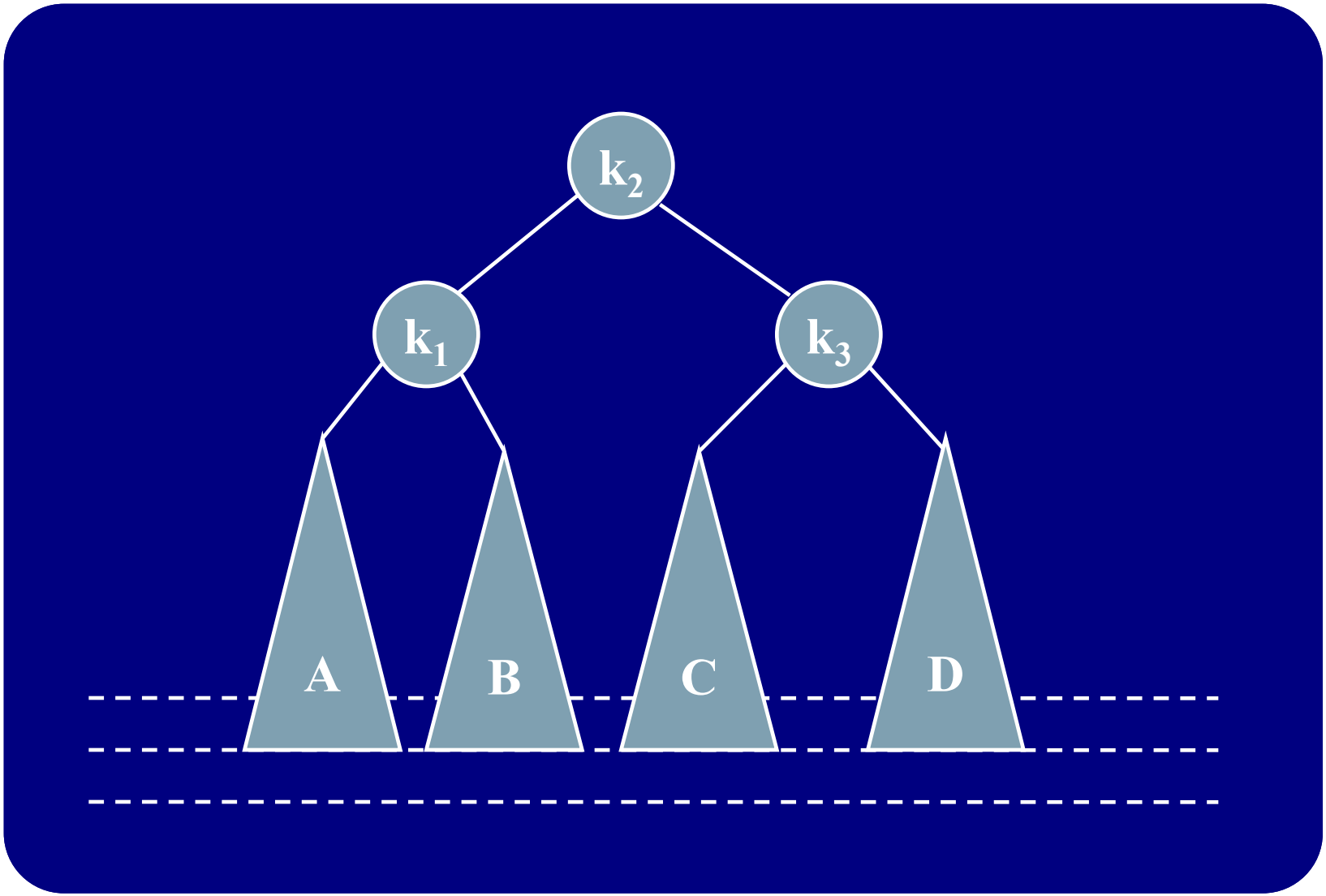
- Single rotation tidak bisa digunakan untuk kasus 2 dan 3 (*inside case*)



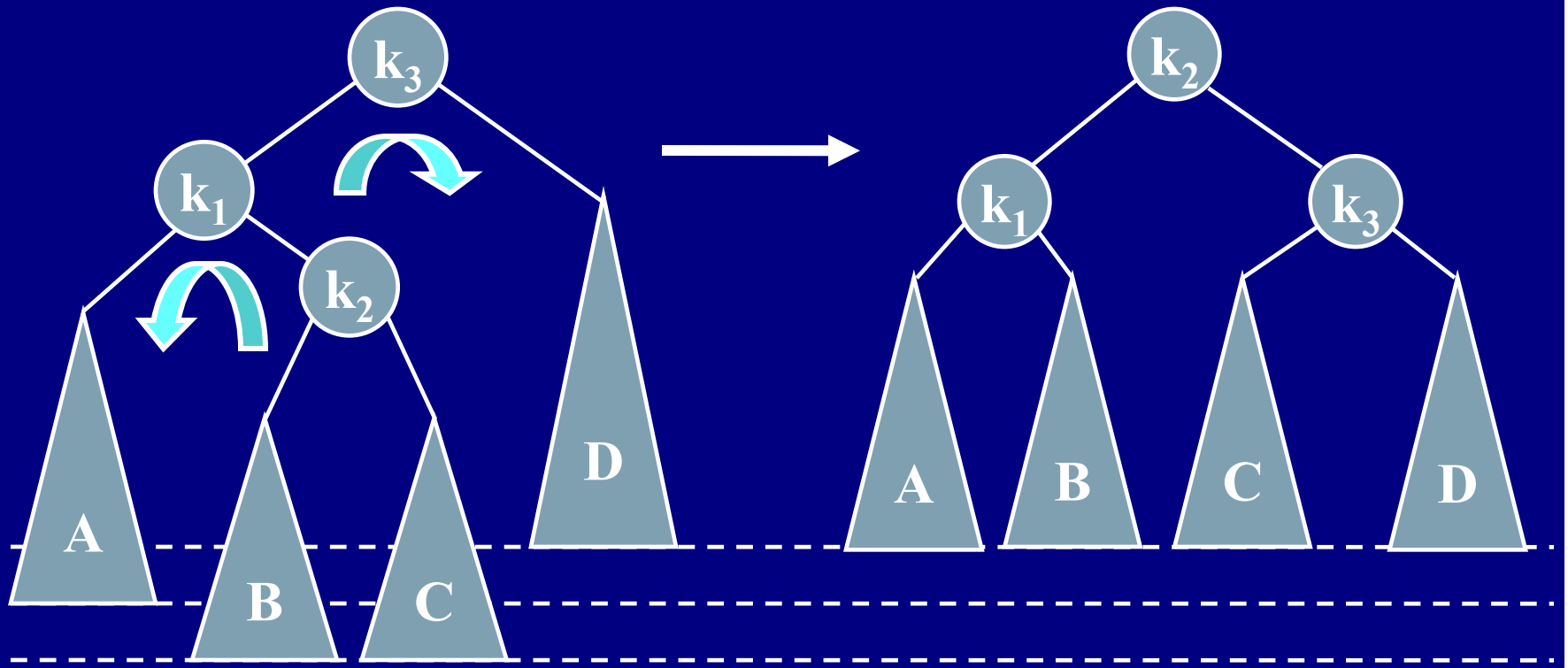
Double Rotation: Langkah



Double Rotation: Langkah

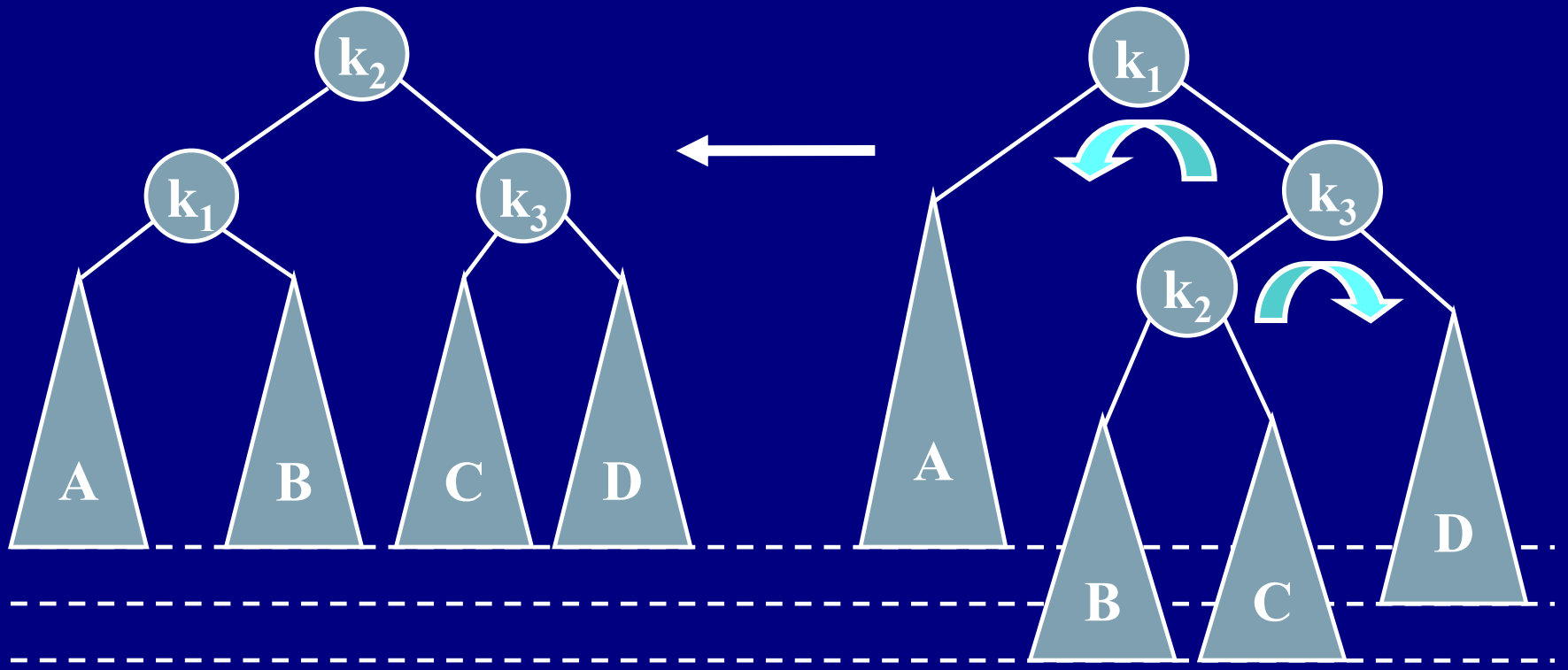


Double Rotation



$H_A = H_B = H_C = H_D$

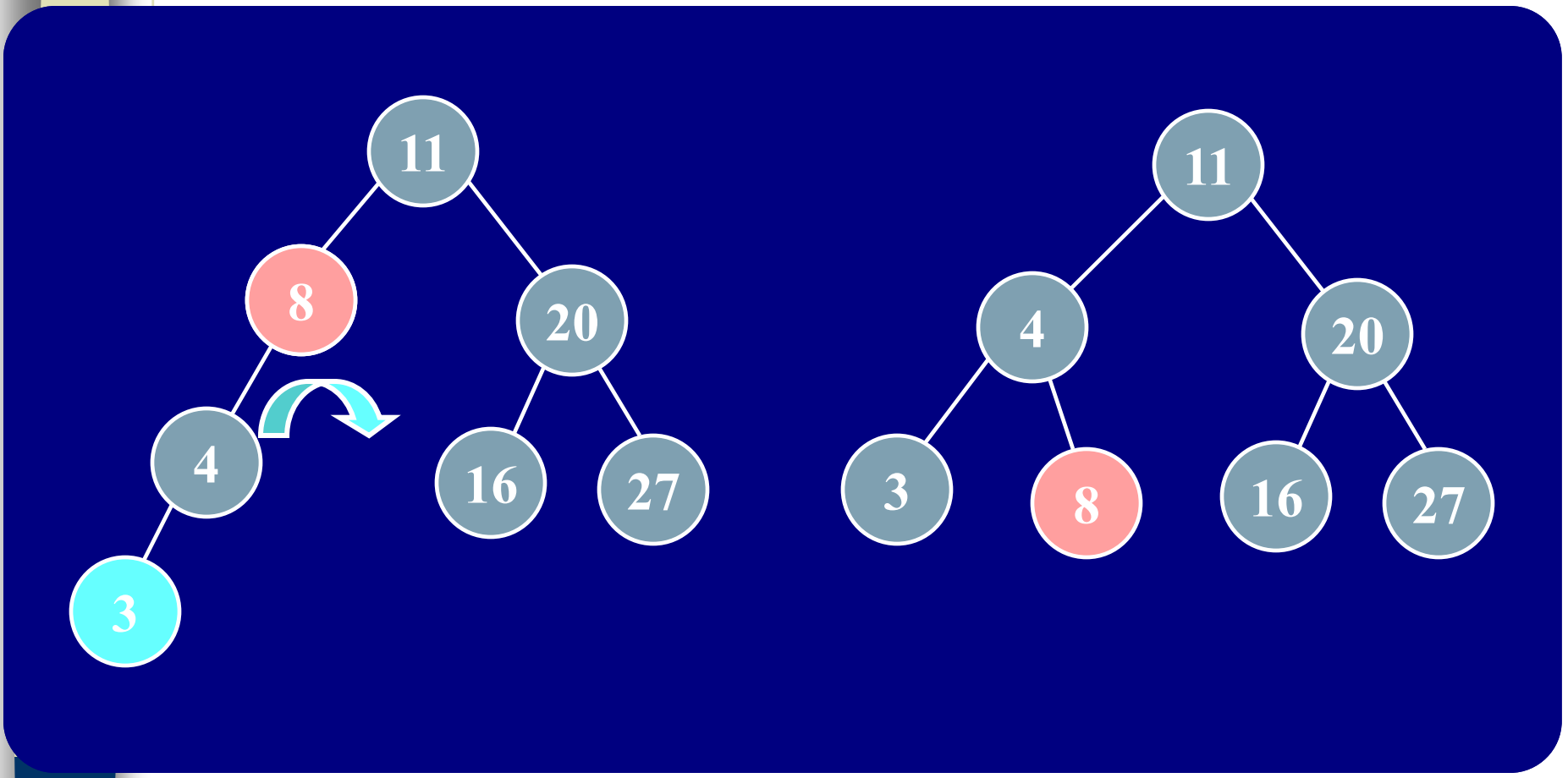
Double Rotation



$H_A = H_B = H_C = H_D$

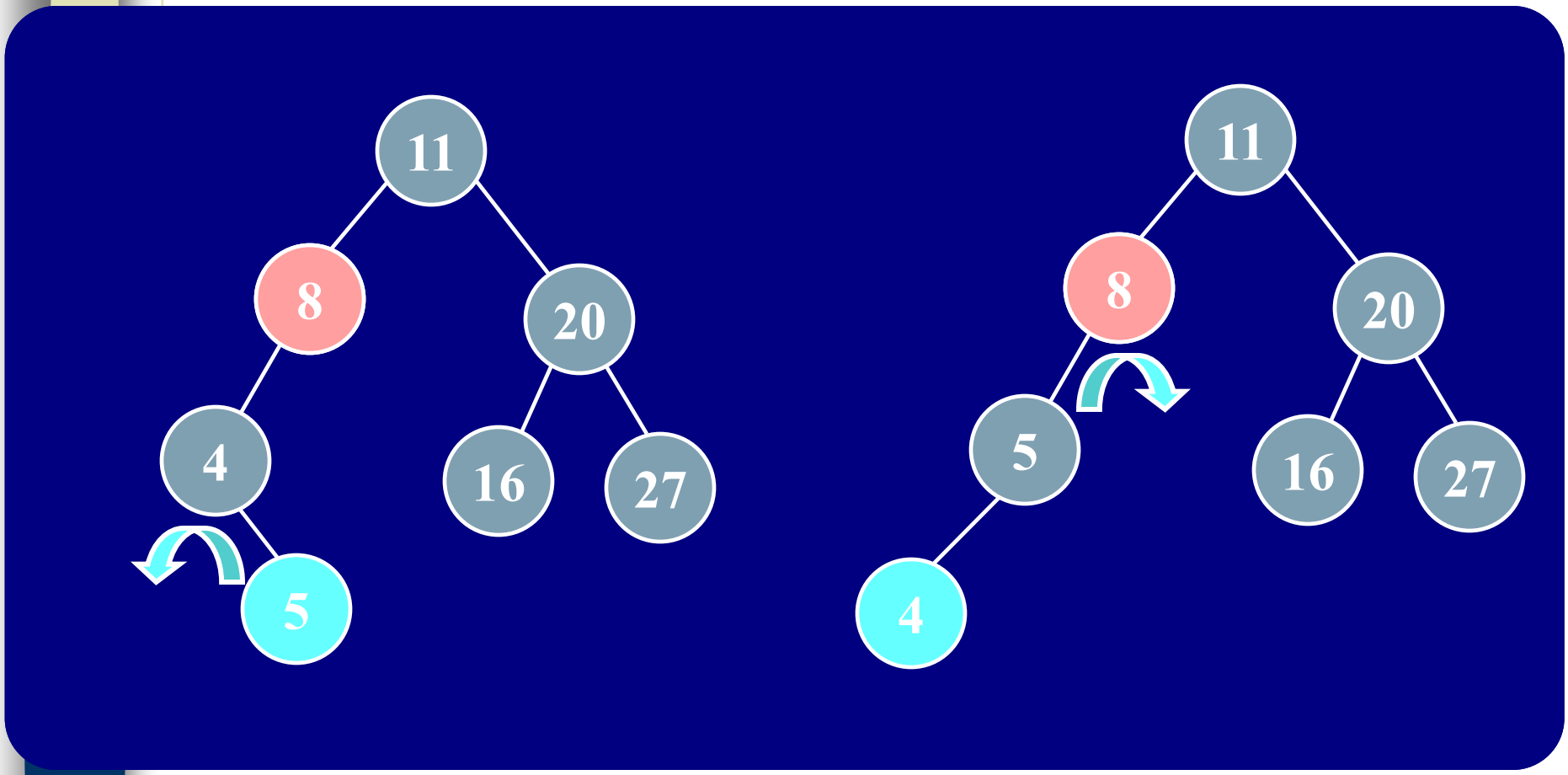
Contoh

- penambahan 3 pada AVL tree



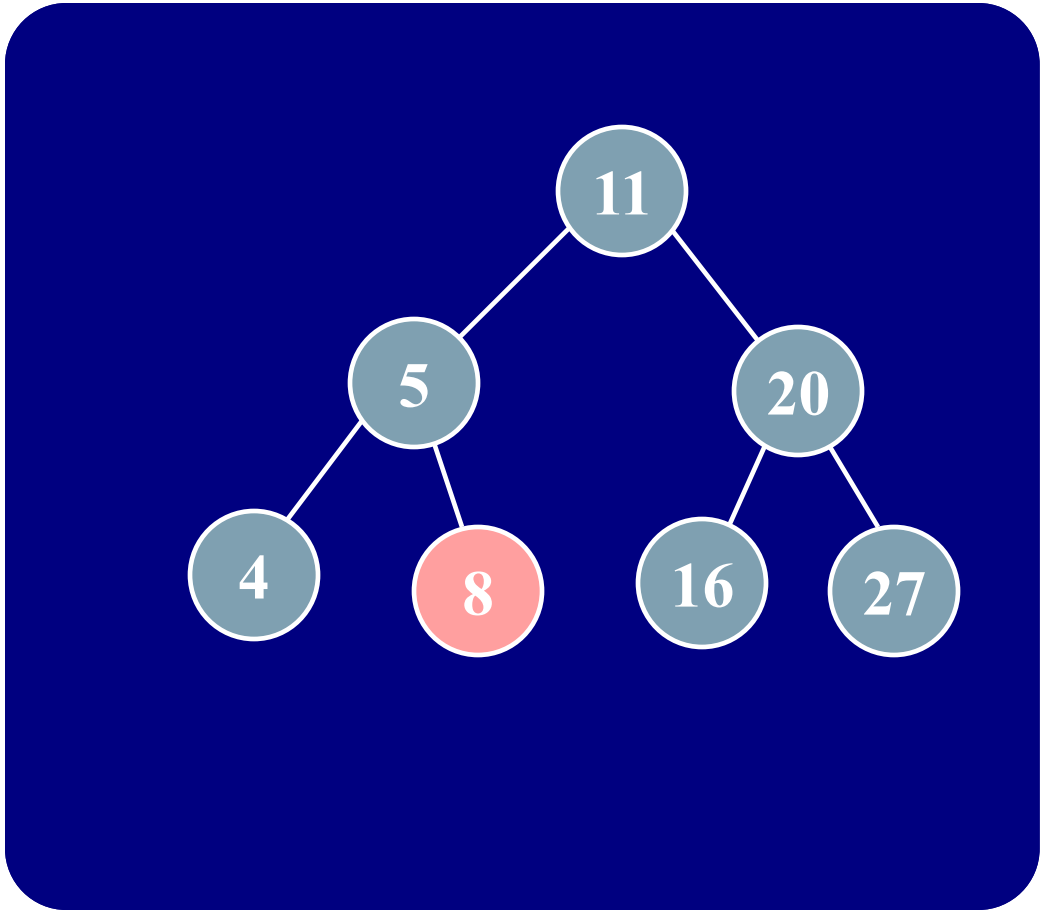
Contoh

- penambahan 5 pada AVL tree



Contoh

- Rotasi ke 2



AVL Trees: Latihan

- Coba simulasikan penambahan pada sebuah AVL dengan urutan penambahan:
  - 10, 85, 15, 70, 20, 60, 30, 50, 65, 80, 90, 40, 5, 55

http://www.strille.net/works/media_technology_projects/avl-tree_2001/

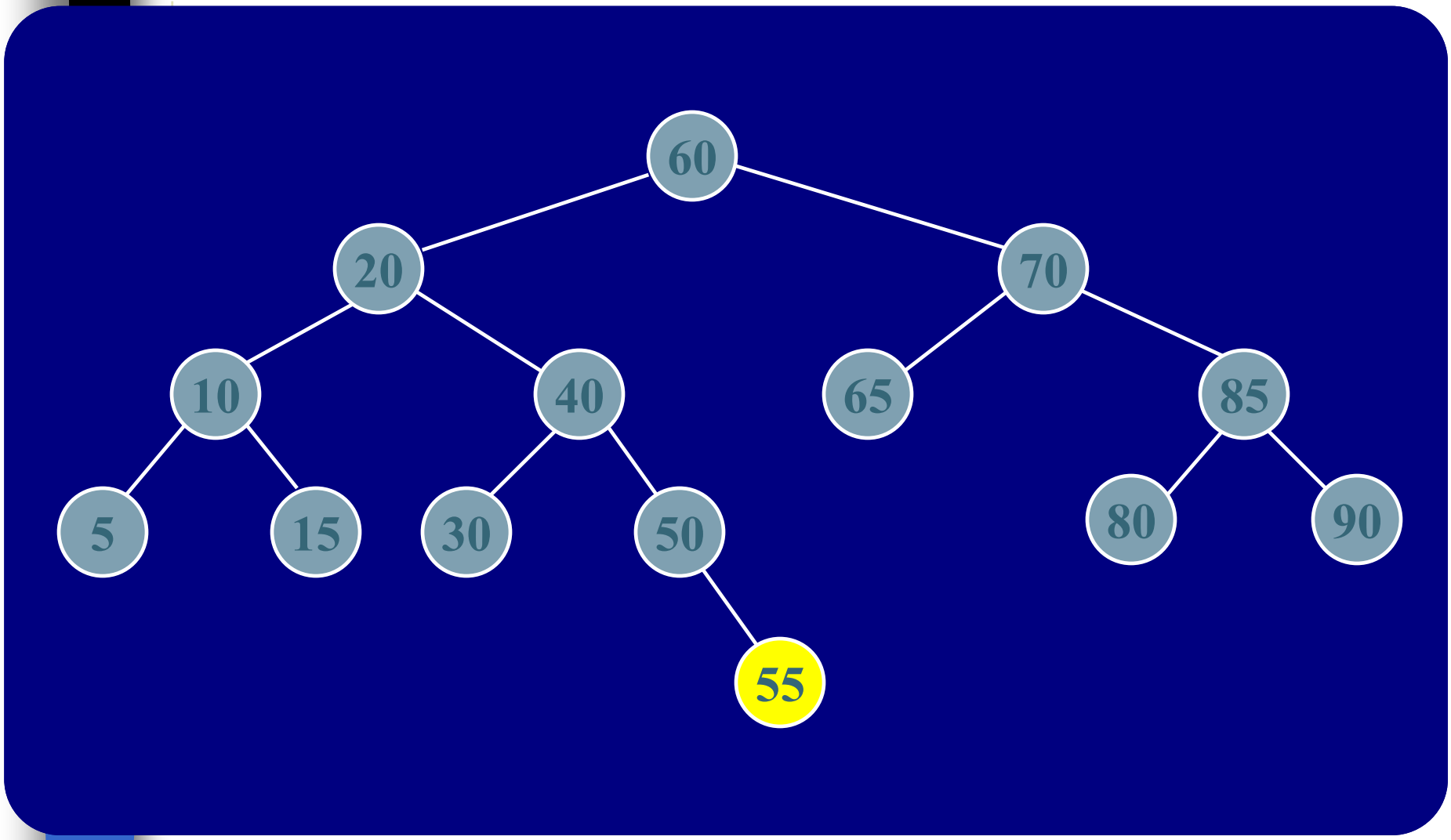
Operasi: Remove pada AVL Tree

1. Menghapus node pada AVL Tree sama dengan menghapus binary search tree procedure dengan perbedaan pada penanganan kondisi tidak *balance*.
2. Penanganan kondisi tidak balance pada operasi menghapus node AVL tree, serupa dengan pada operasi penambahan. Mulai dari node yang diproses (dihapus) periksa seluruh node pada jalur yang menuju root (termasuk root) untuk menentukan node tidak balance yang pertama
3. Terapkan ***single*** atau ***double rotation*** untuk menyeimbangkan *tree*.
4. Bila Tree masih belum balance, ulangi lagi dari langkah 2.

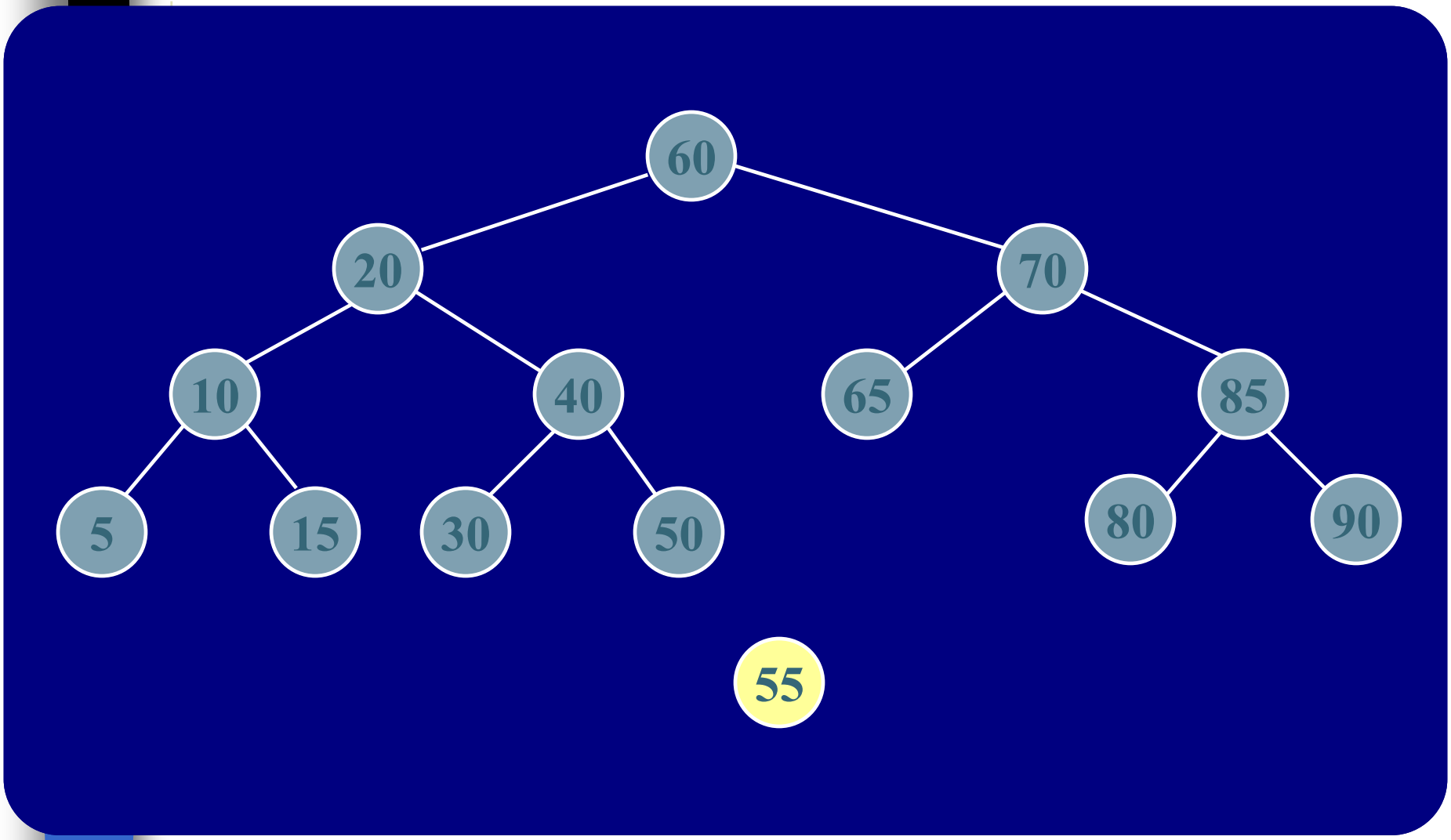
Menghapus node X pada AVL Trees

- Deletion:
  - Kasus 1: jika X adalah leaf, delete X
  - Kasus 2: jika X punya 1 child, X digantikan oleh child tsb.
  - Kasus 3: jika X punya 2 child, ganti X secara rekursif dengan predecessor-nya secara inorder
- Rebalancing

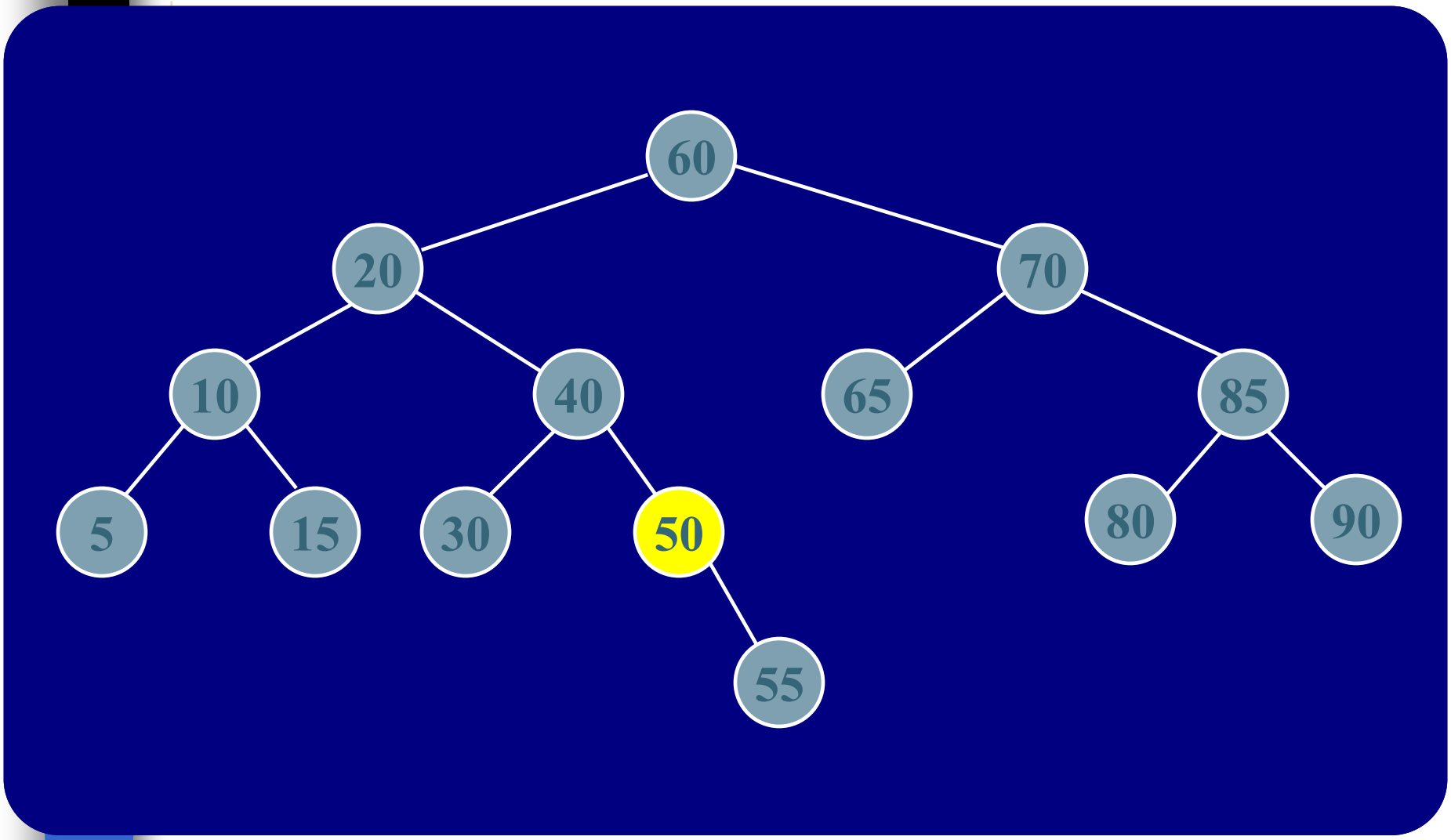
Delete 55 (Kasus 1)



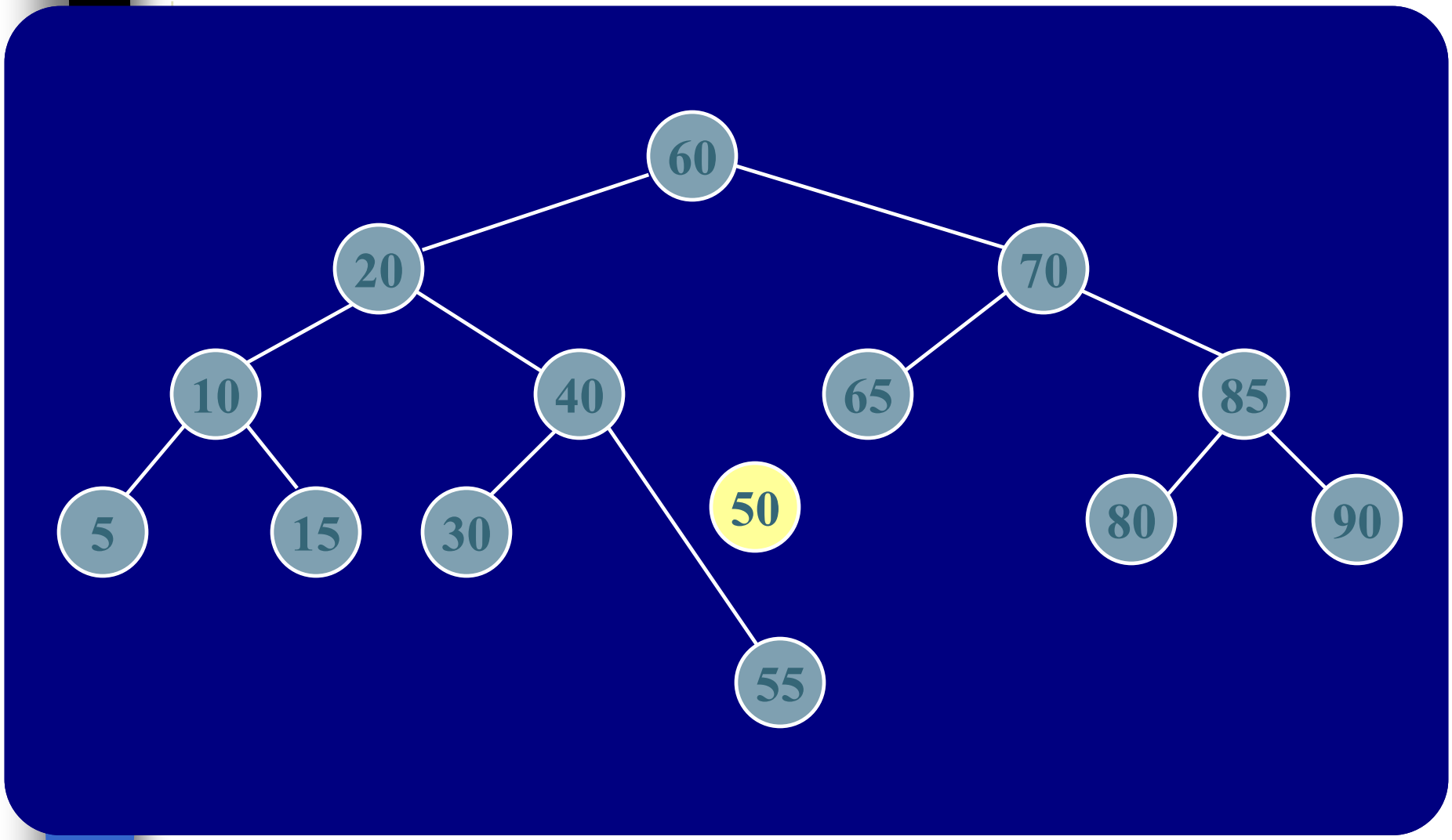
Delete 55 (Kasus 1)



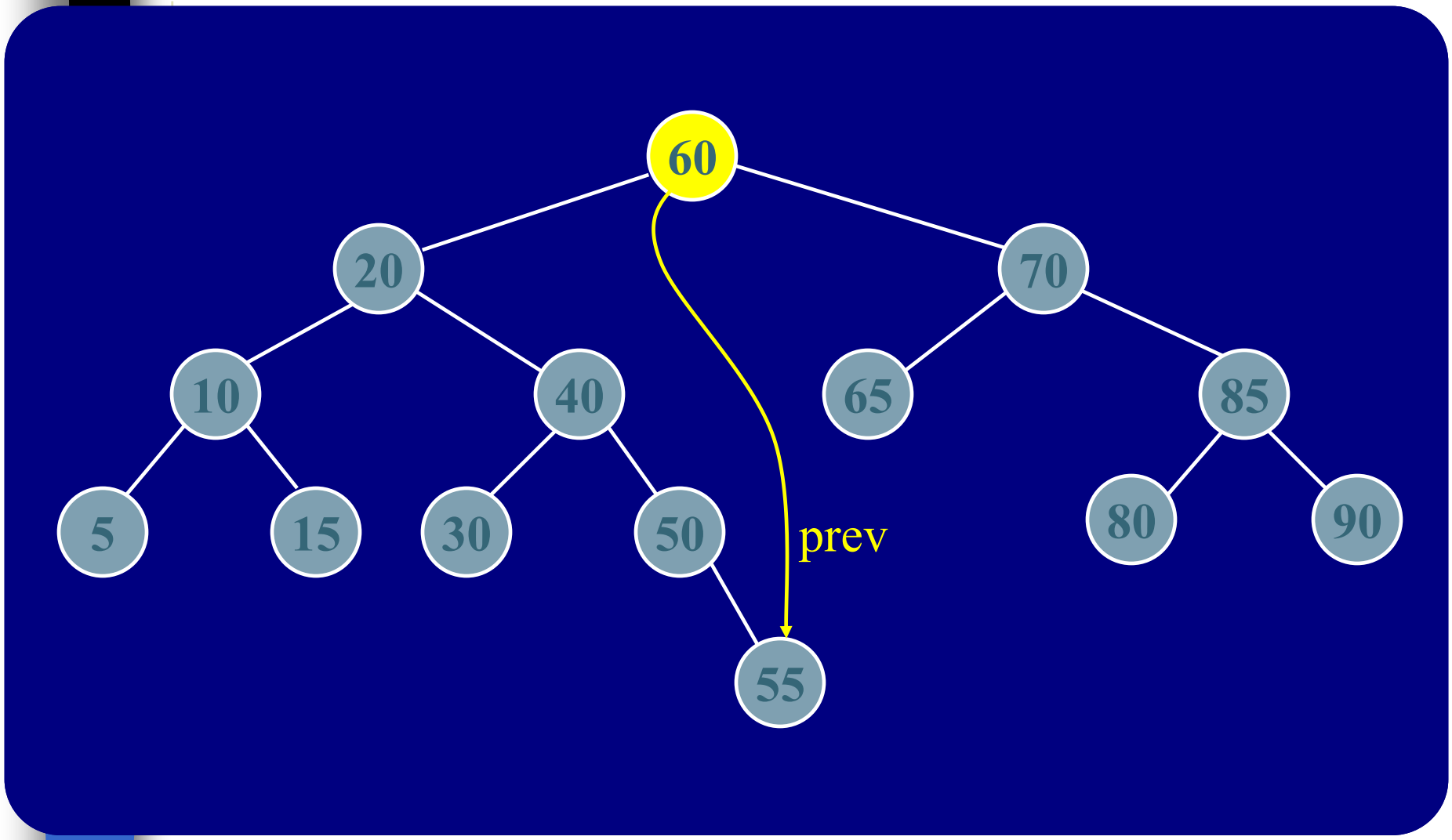
Delete 50 (Kasus 2)



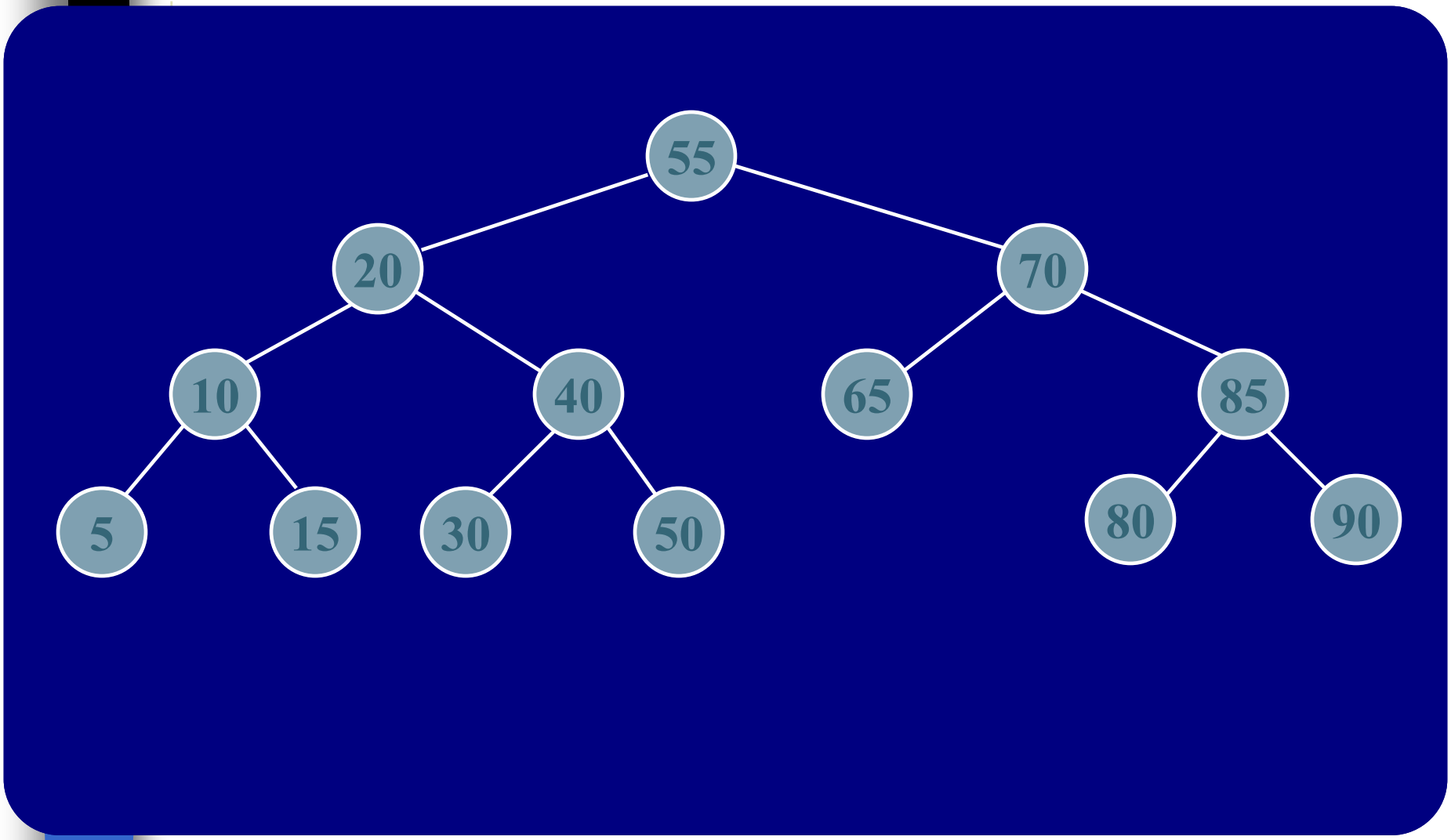
Delete 50 (Kasus 2)



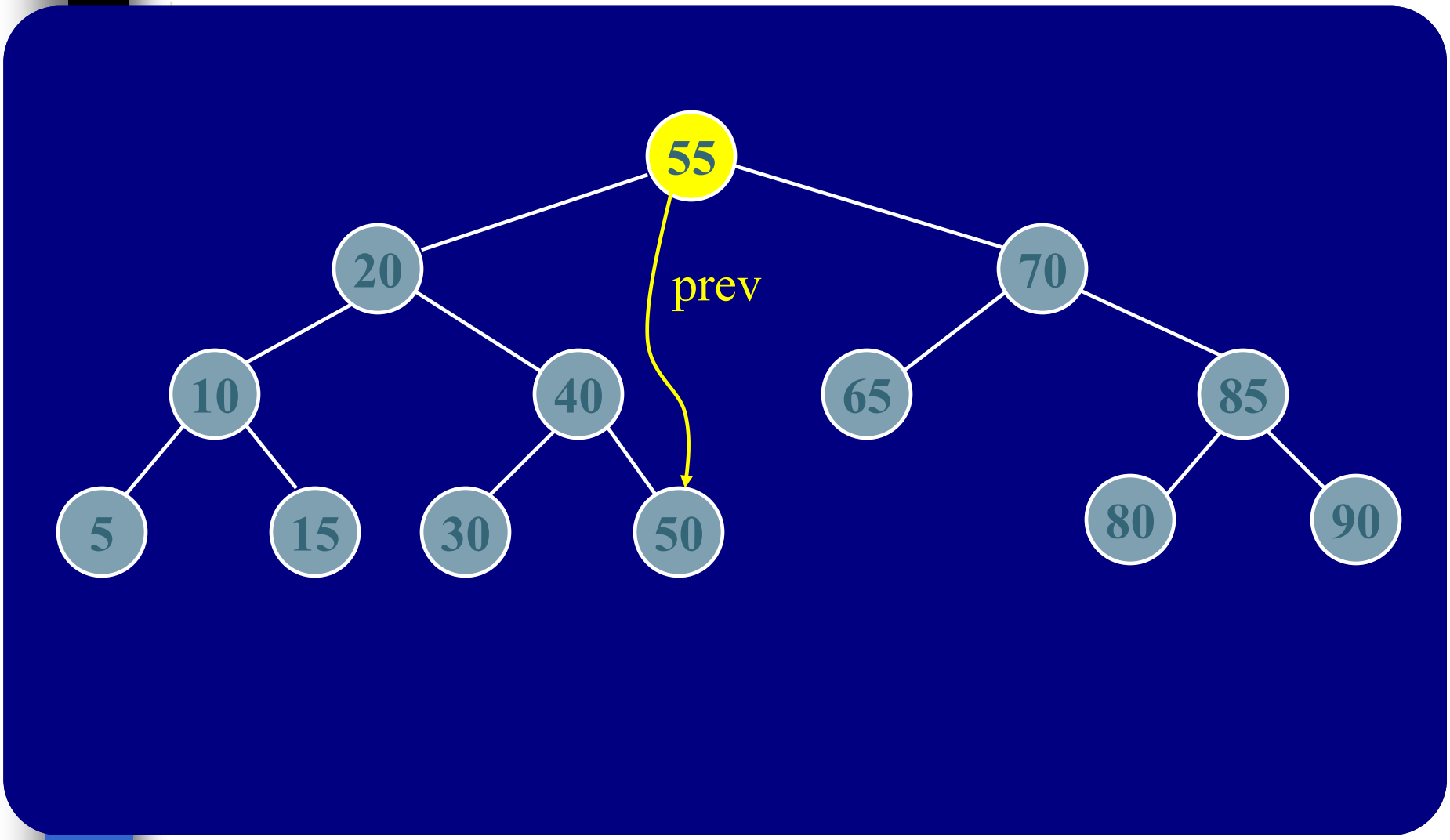
Delete 60 (Kasus 3)



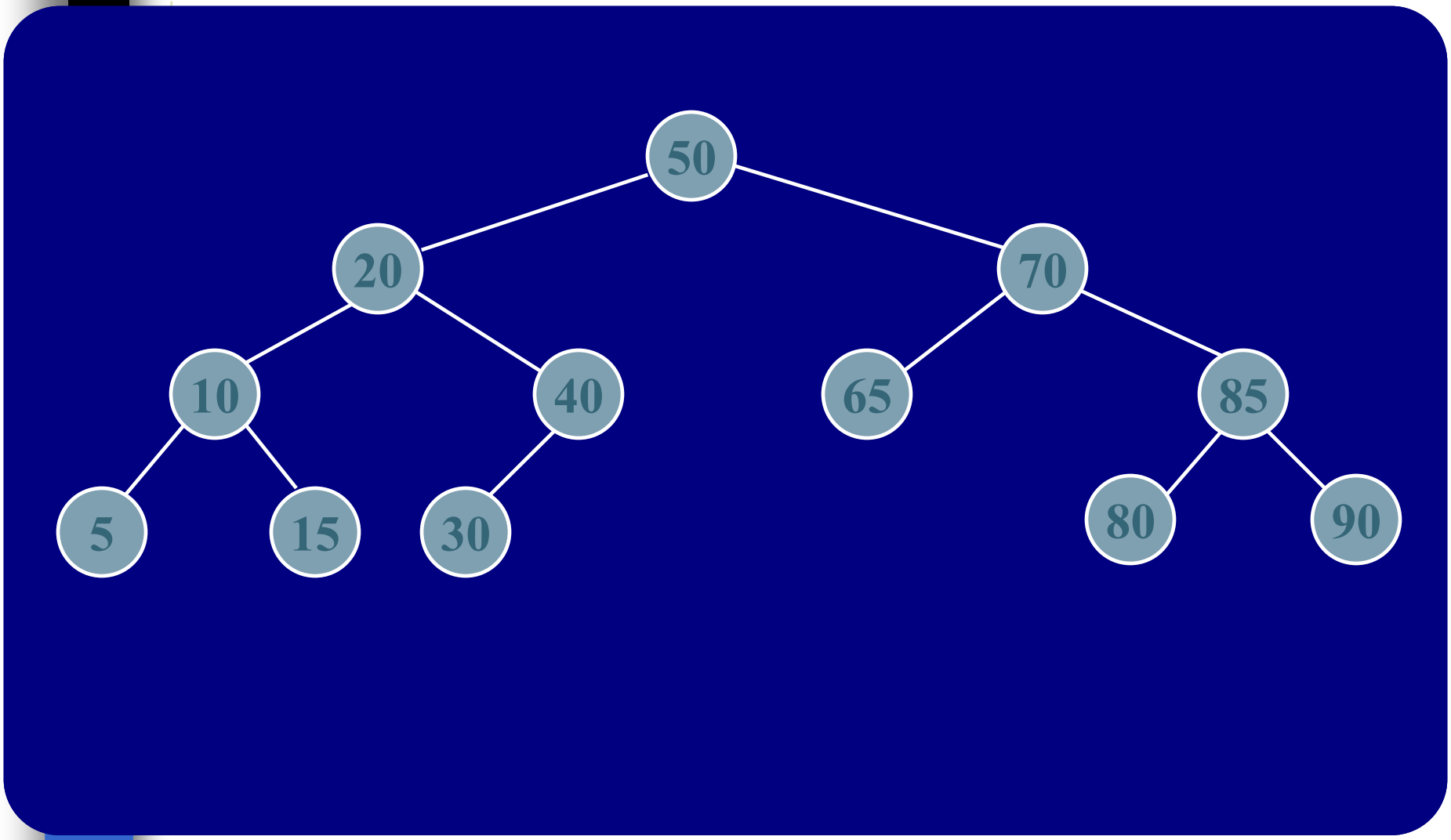
Delete 60 (Kasus 3)



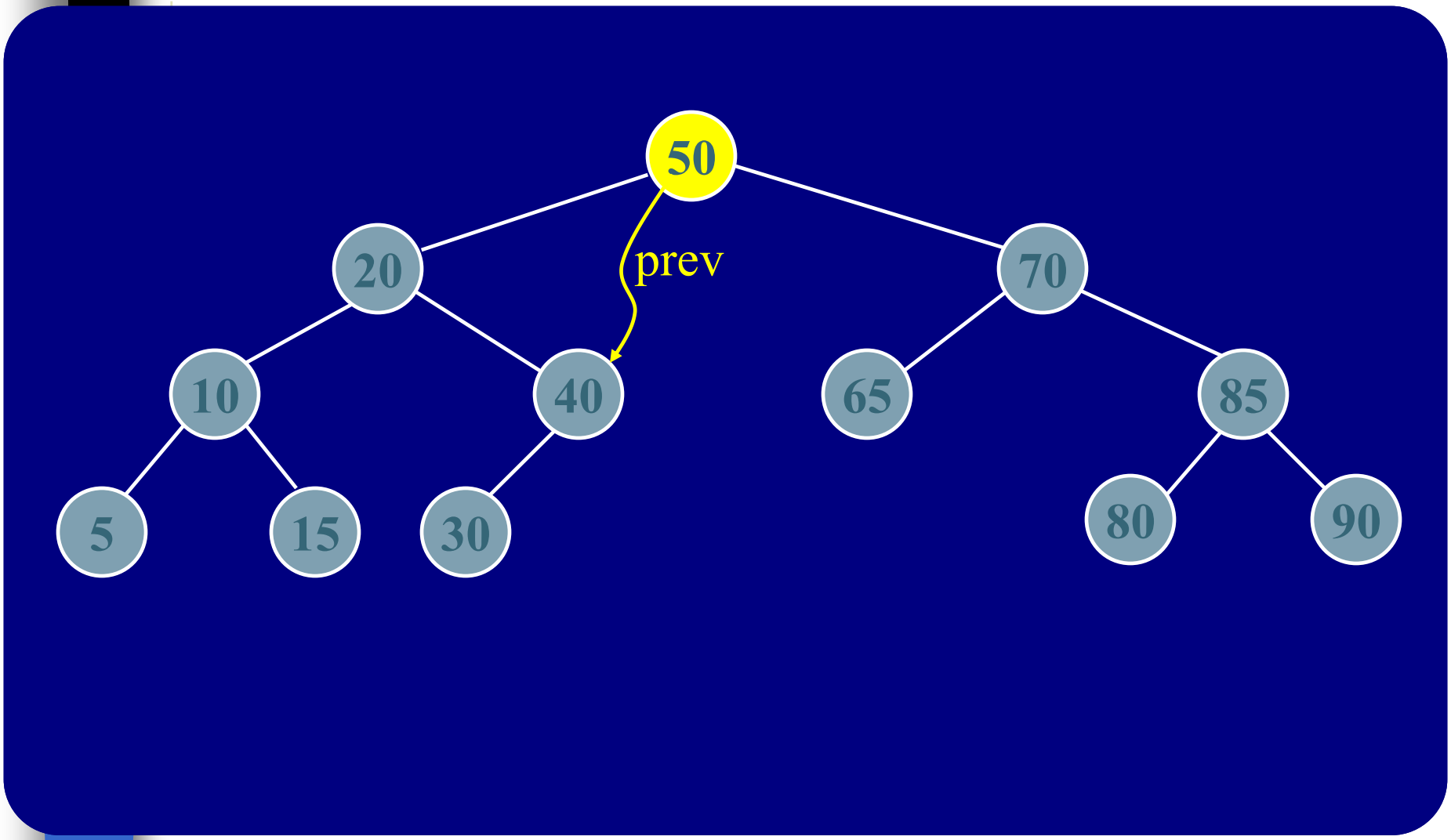
Delete 55 (Kasus 3)



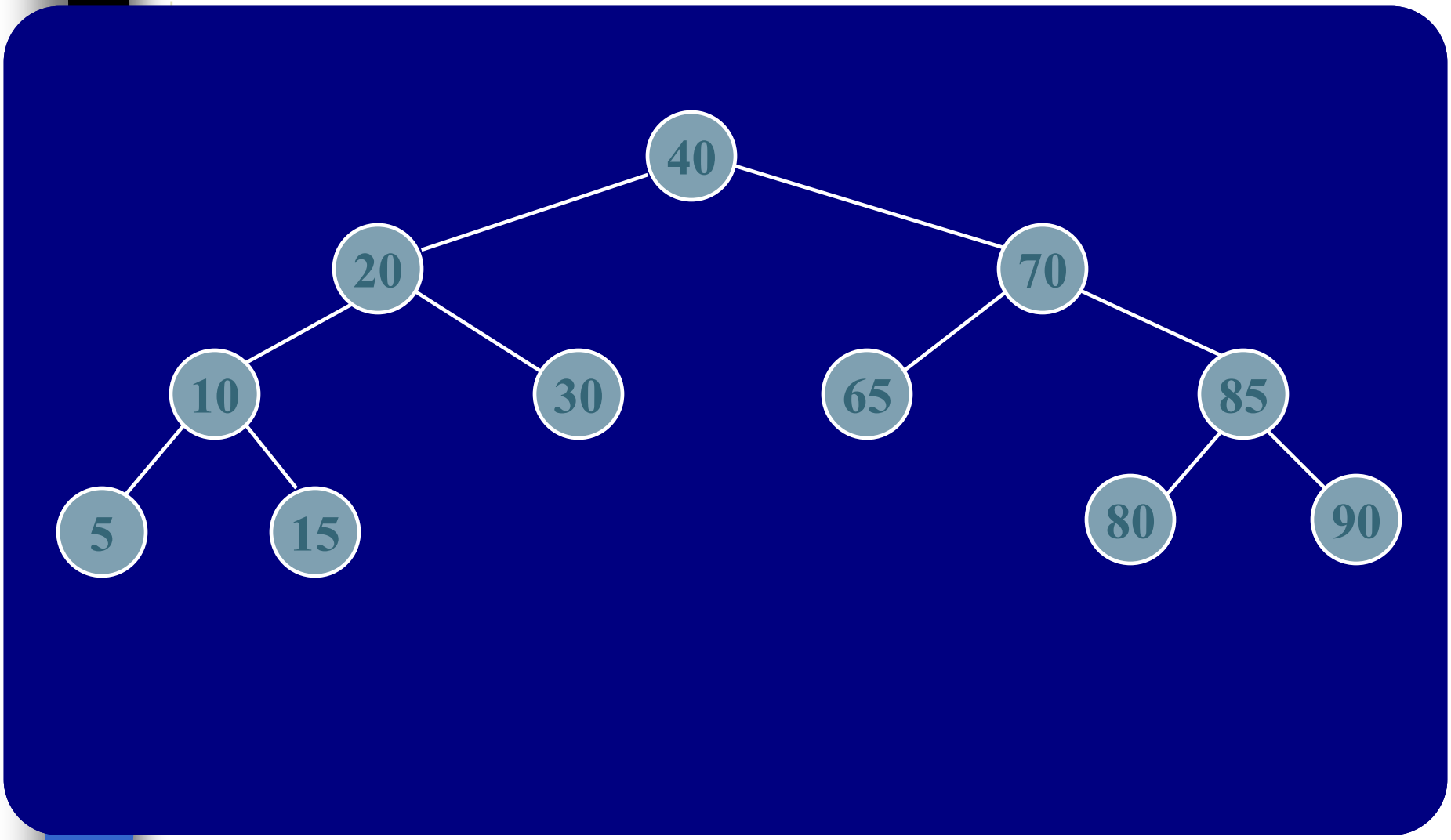
Delete 55 (Kasus 3)



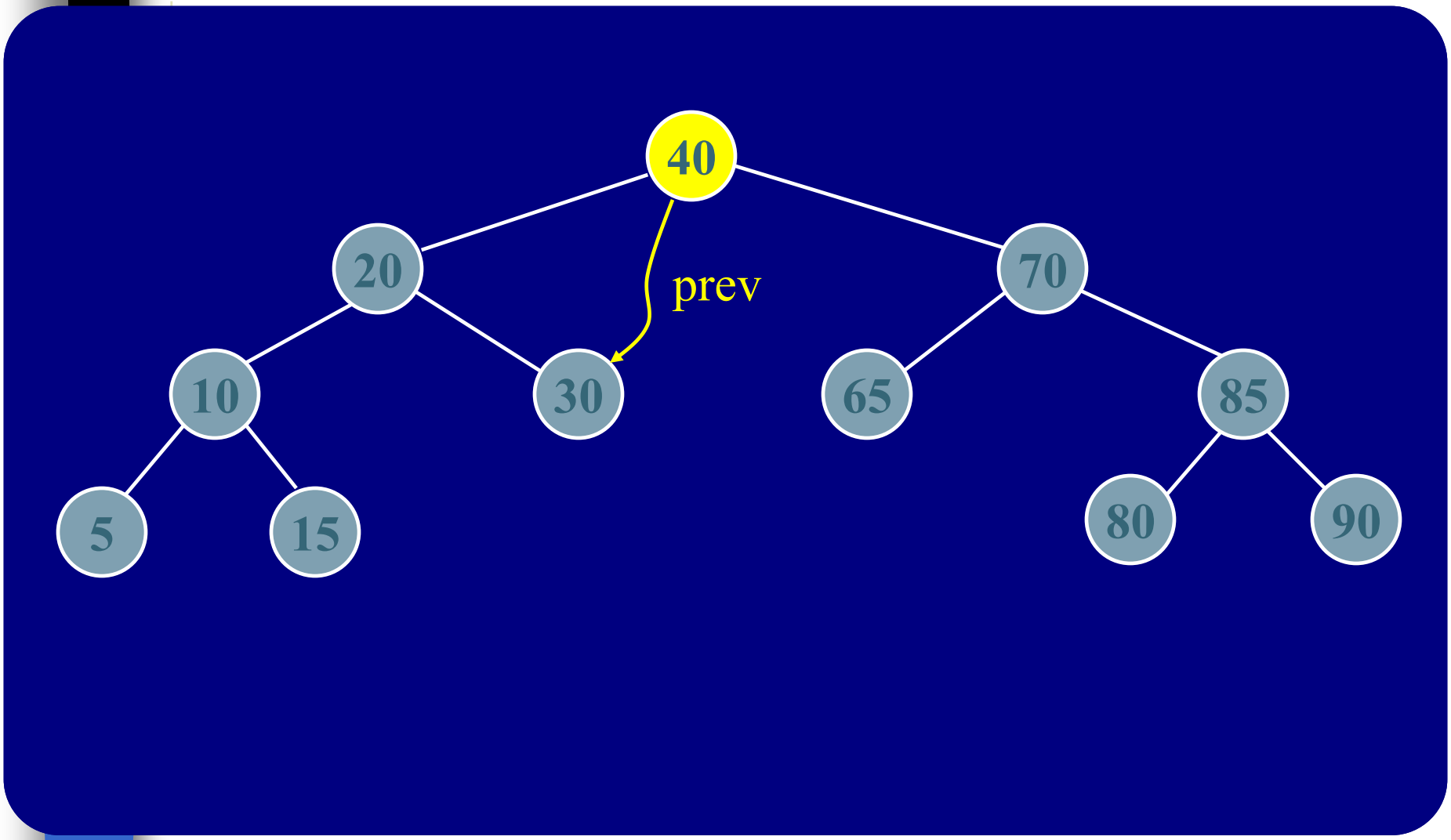
Delete 50 (Kasus 3)



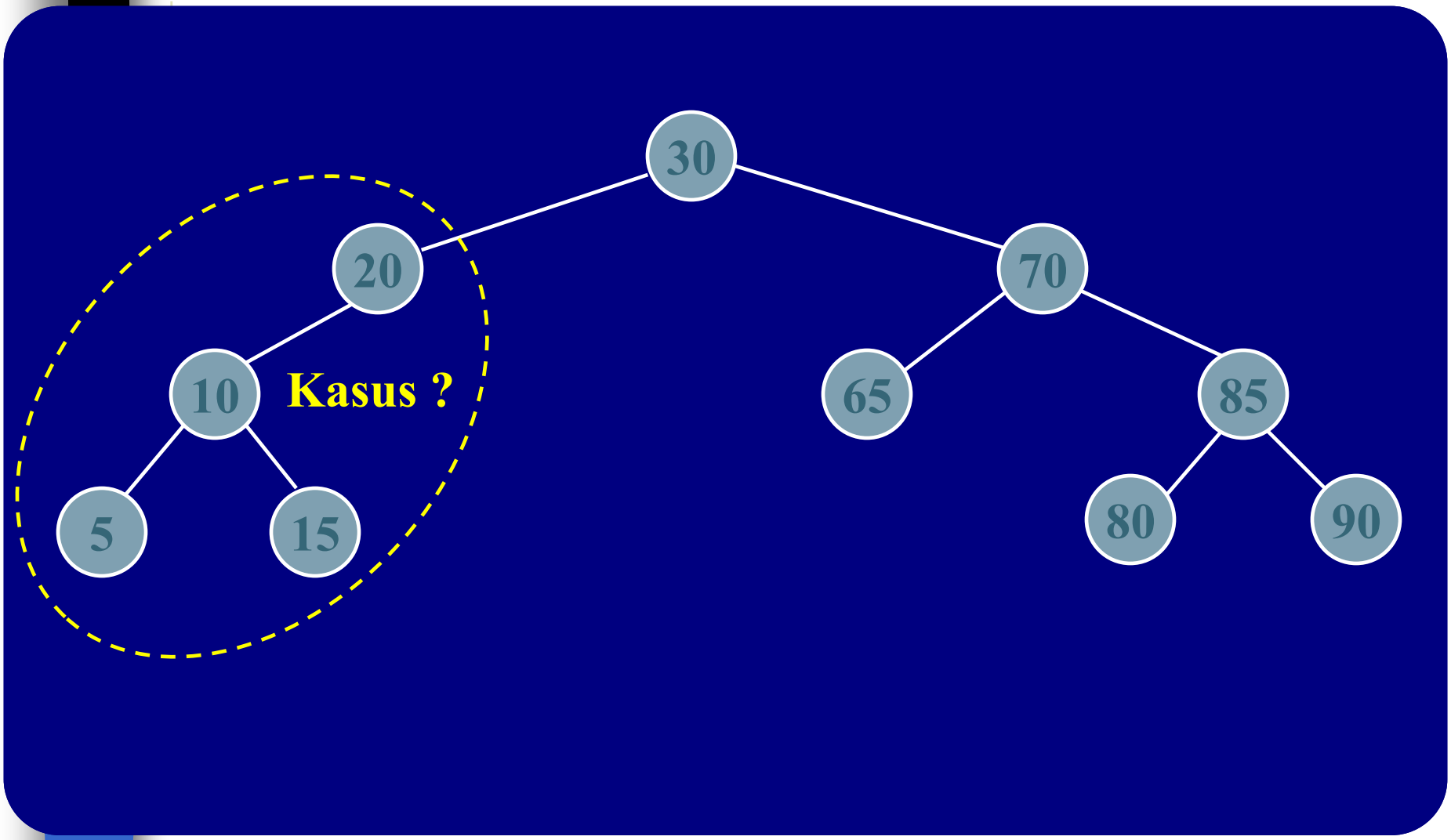
Delete 50 (Kasus 3)



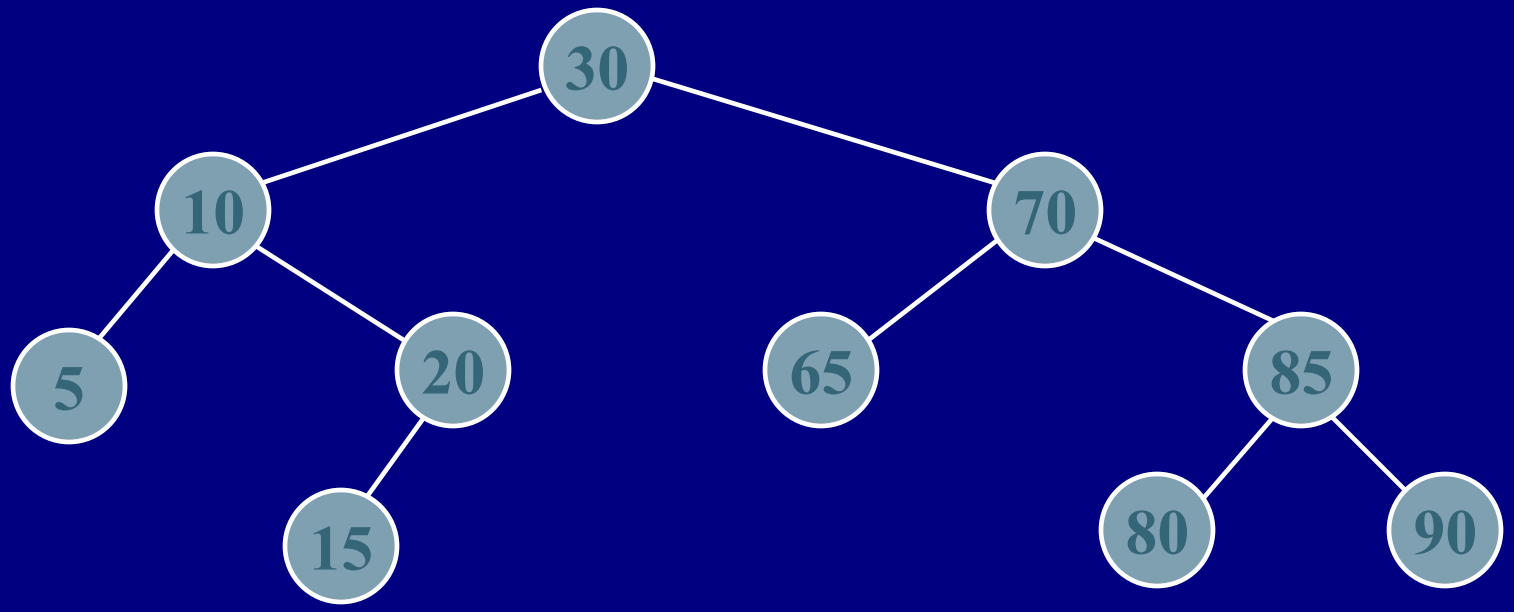
Delete 40 (Kasus 3)



Delete 40 : Rebalancing



Delete 40: setelah *rebalancing*



Single rotation is preferred!