Lecture 15: Indexes

Friday, May 7, 2010

Outline

- Index structures (14.1, 14.2)
- B-trees (14.3)

Note: in old edition this is Chapter 13 instead of 14

File Types

The **data file** can be one of:

- Heap file:
  - Set of records, partitioned into blocks
  - Unsorted
- Sequential file:
  - Sorted according to some attribute(s) called *key*

Note: “key” here means something else than “primary key”

Index

- A (possibly separate) file, that allows fast access to records in the data file
- The index contains (key, value) pairs:
  - The key = an attribute value
  - The value = one of:
    - pointer to the record *secondary index*
    - or the record itself *primary index*

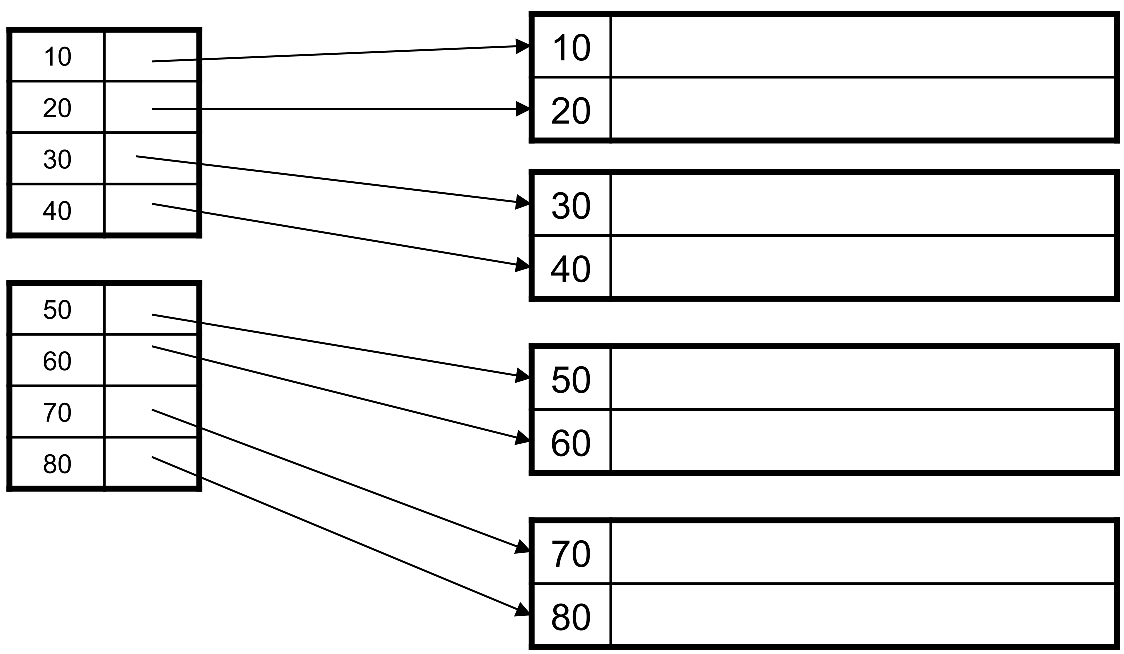
Note: “key” (aka “search key”) again means something else

Index Classification

- Clustered/unclustered
  - Clustered = data file is ordered by the index’ search key
  - Unclustered = othewise
- Primary/secondary:
  - Meaning 1: same as clustered/unclustured
  - Meaning 2:
    - Primary = is over attributes part of the primary
    - Secondary = cannot reorder data
- Organization: B+ tree or Hash table

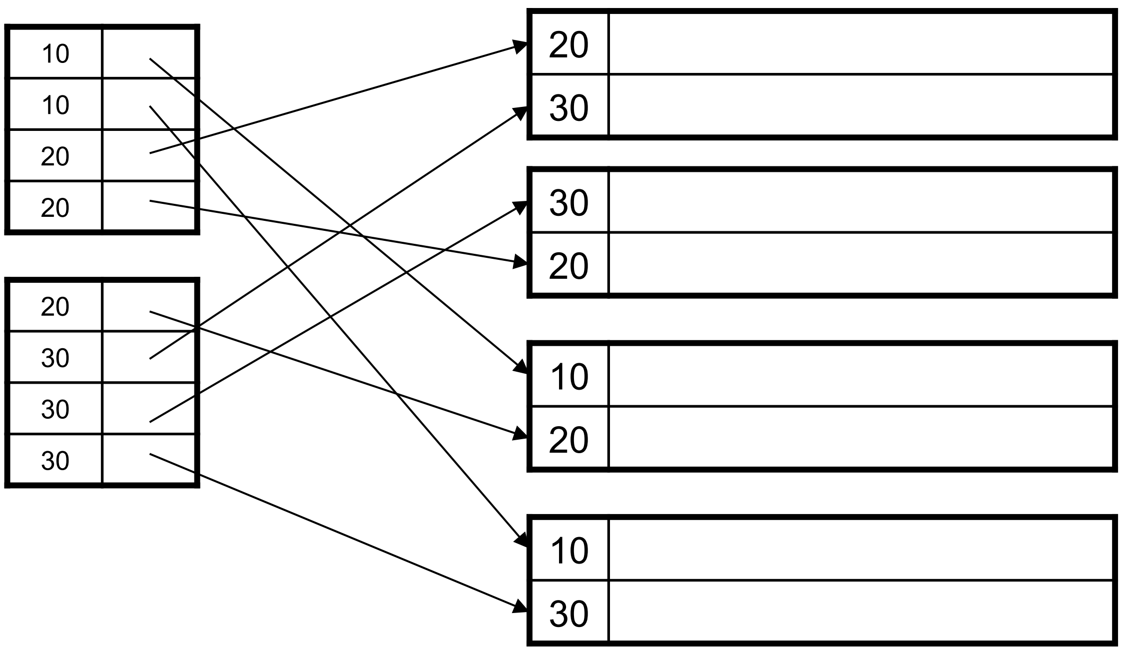
Clustered Index

- File is sorted on the index attribute
- Only one per table

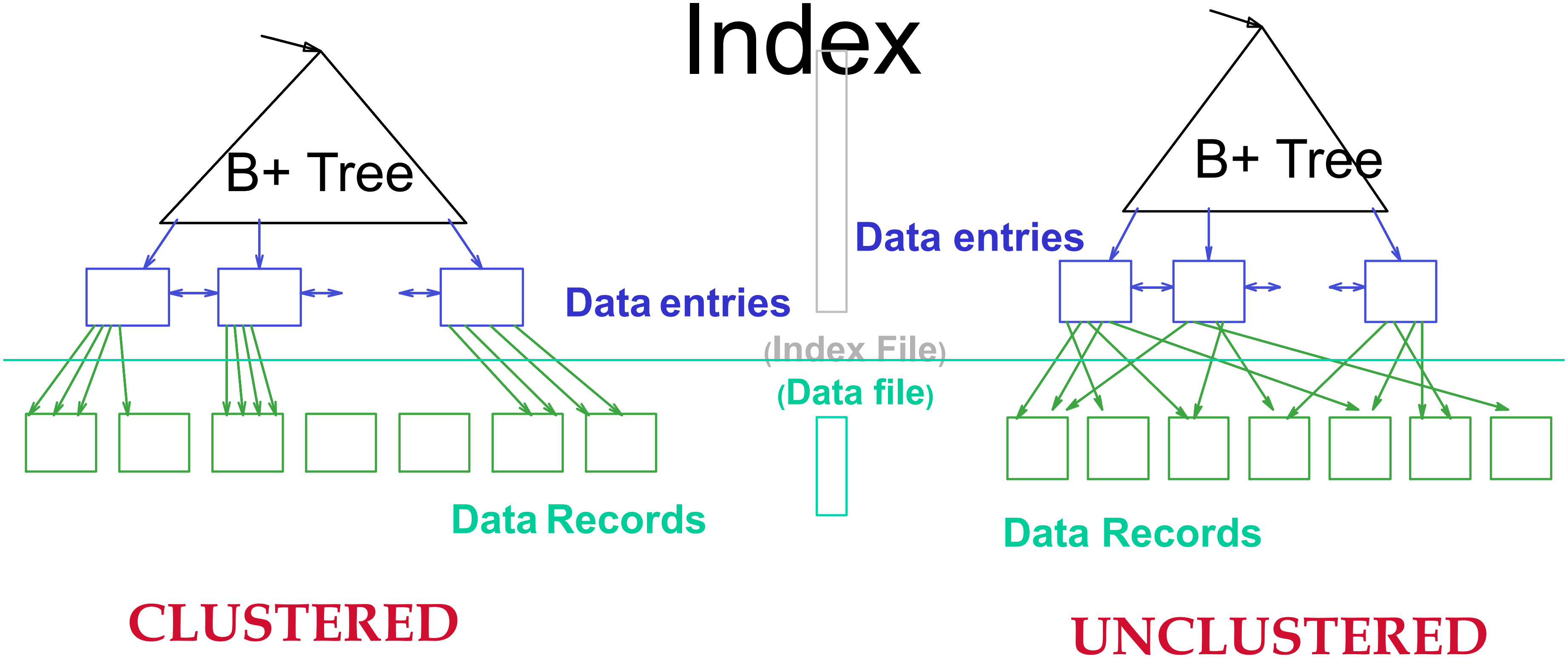


Unclustered Index

- Several per table



Clustered vs. Unclustered Index

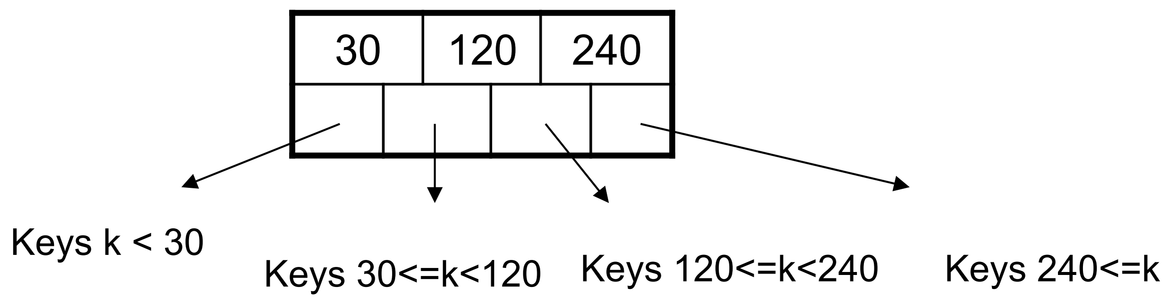


B+ Trees

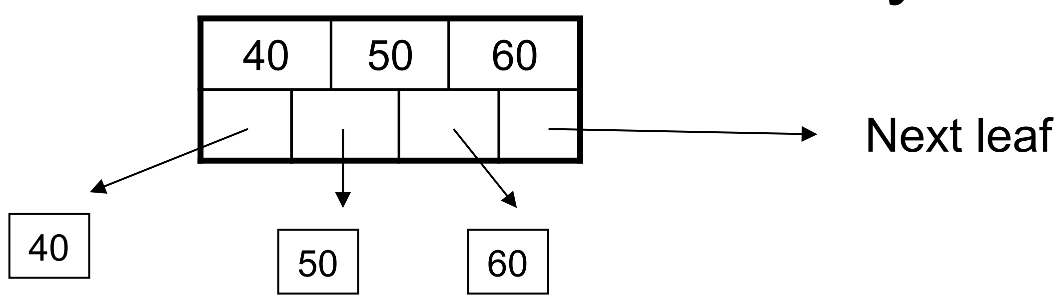
- Search trees
- Idea in B   Trees:
  - make 1 node = 1 block
- Idea in B+ Trees:
  - Make leaves into a linked list (range queries are easier)

B+ Trees Basics

- Parameter d = the *degree*
- Each node has >= d and <= 2d keys (except root)

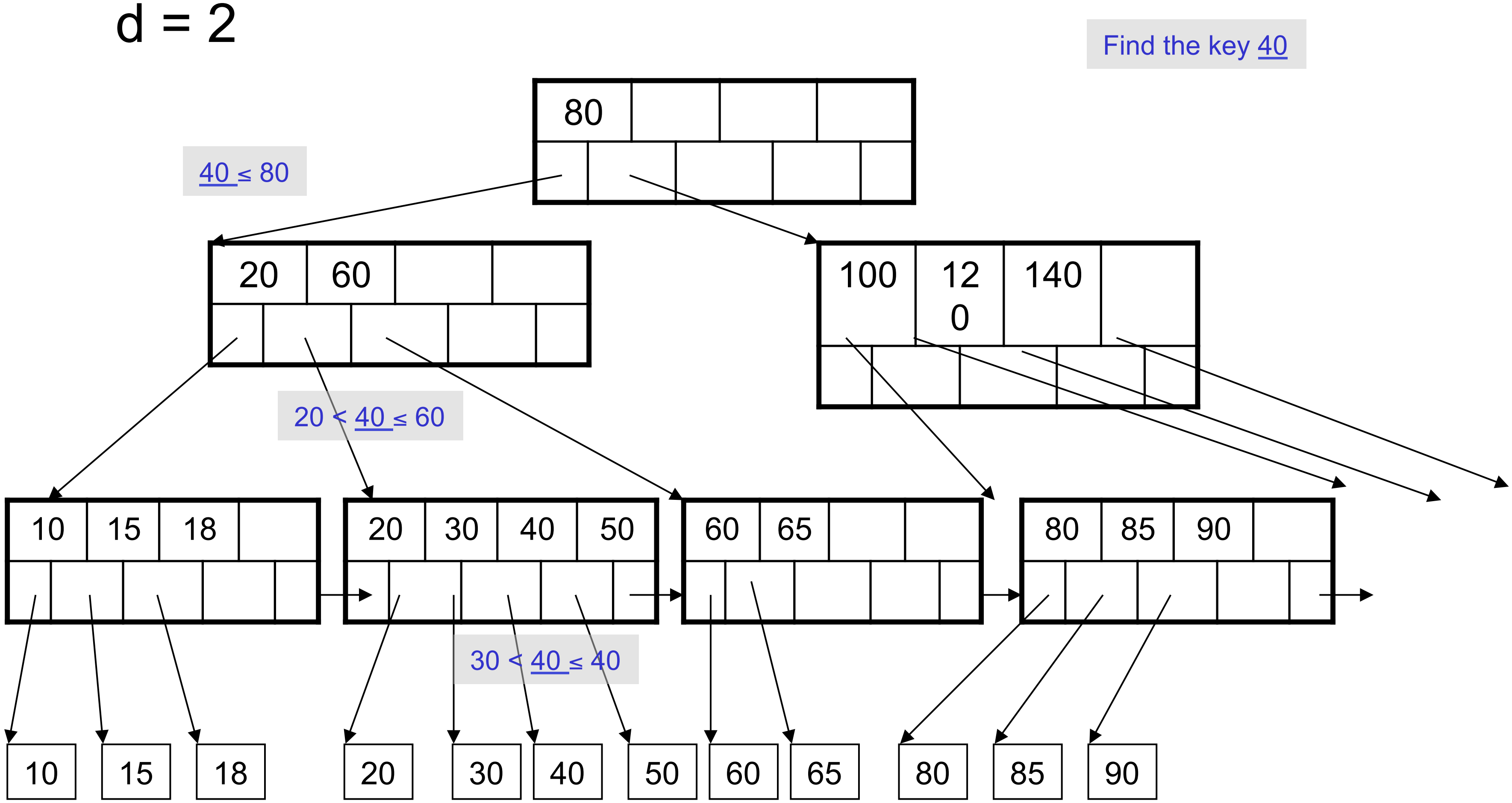


- Each leaf has >=d and <= 2d keys:



B+ Tree Example

d = 2



Using a B+ Tree

Index on People(age)

- Exact key values:
  - Start at the root
  - Proceed down, to the leaf

```
Select name
From People
Where age = 25
```

- Range queries:
  - As above
  - Then sequential traversal

```
Select name
From People
Where 20 <= age
  and  age <= 30
```

Which queries can use this index ?

Index on People(name, zipcode)

```
Select *
From People
Where name = 'Smith'
  and zipcode = 12345
```

```
Select *
From People
Where name = 'Smith'
```

```
Select *
From People
Where zipcode = 12345
```

B+ Tree Design

- How large d ?
- Example:
  - Key size = 4 bytes
  - Pointer size = 8 bytes
  - Block size = 4096 byes
- 2d x 4 + (2d+1) x 8 <= 4096
- d = 170

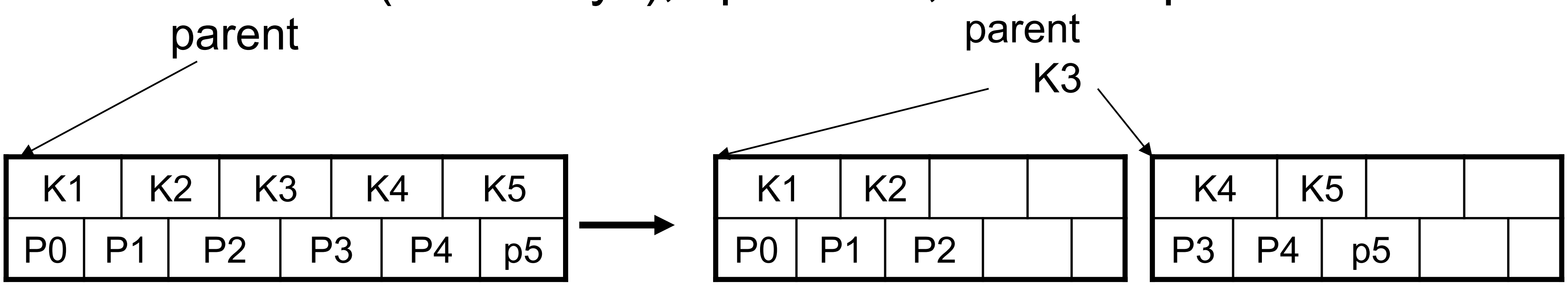
B+ Trees in Practice

- Typical order: 100. Typical fill-factor: 67%.
  - average fanout = 133
- Typical capacities:
  - Height 4: 133^4 = 312,900,700 records
  - Height 3: 133^3 = 2,352,637 records
- Can often hold top levels in buffer pool:
  - Level 1 = 1 page = 8 Kbytes
  - Level 2 = 133 pages = 1 Mbyte
  - Level 3 = 17,689 pages = 133 MBytes

Insertion in a B+ Tree

Insert (K, P)

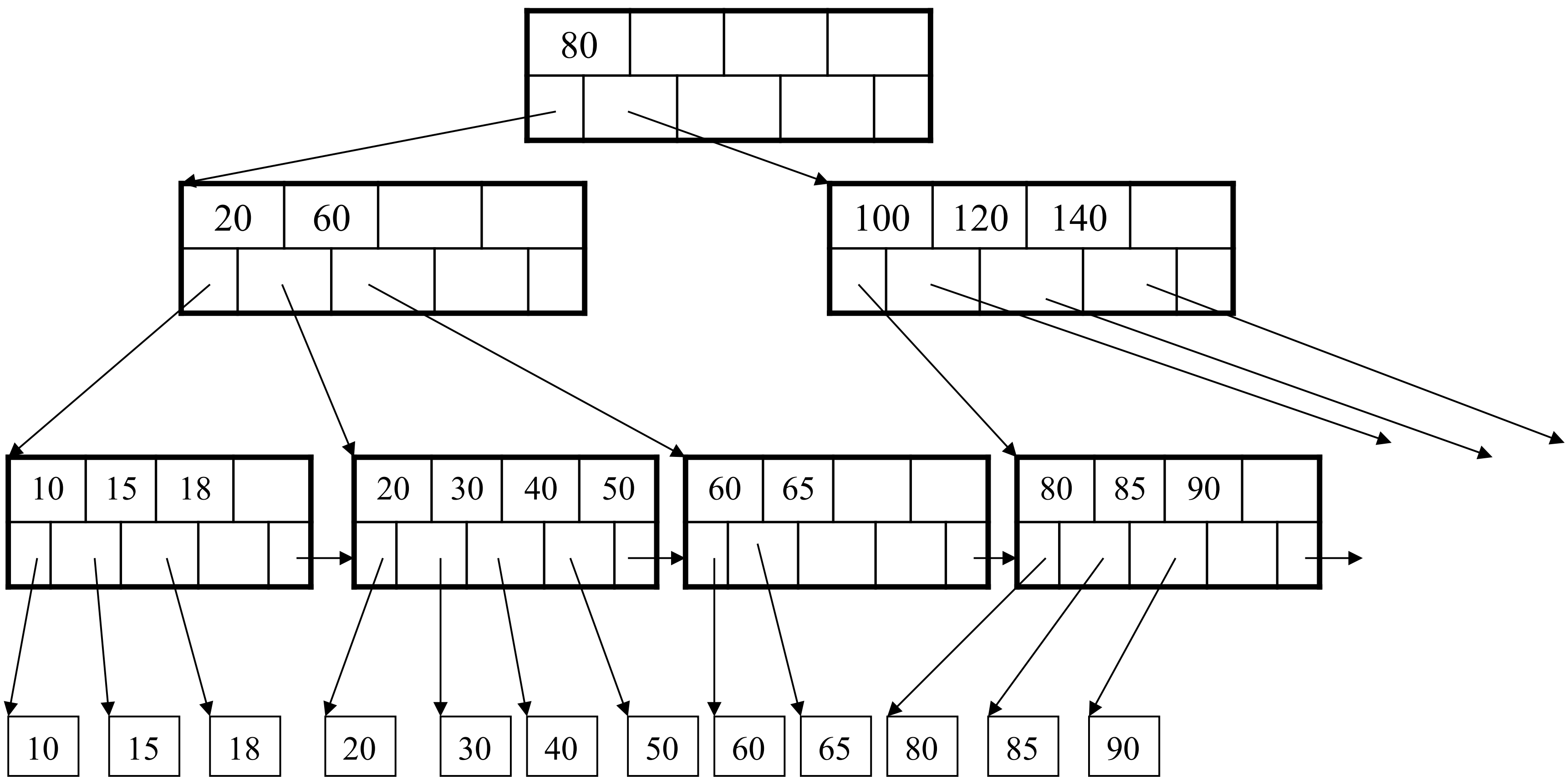
- Find leaf where K belongs, insert
- If no overflow (2d keys or less), halt
- If overflow (2d+1 keys), split node, insert in parent:



- If leaf, keep K3 too in right node
- When root splits, new root has 1 key only

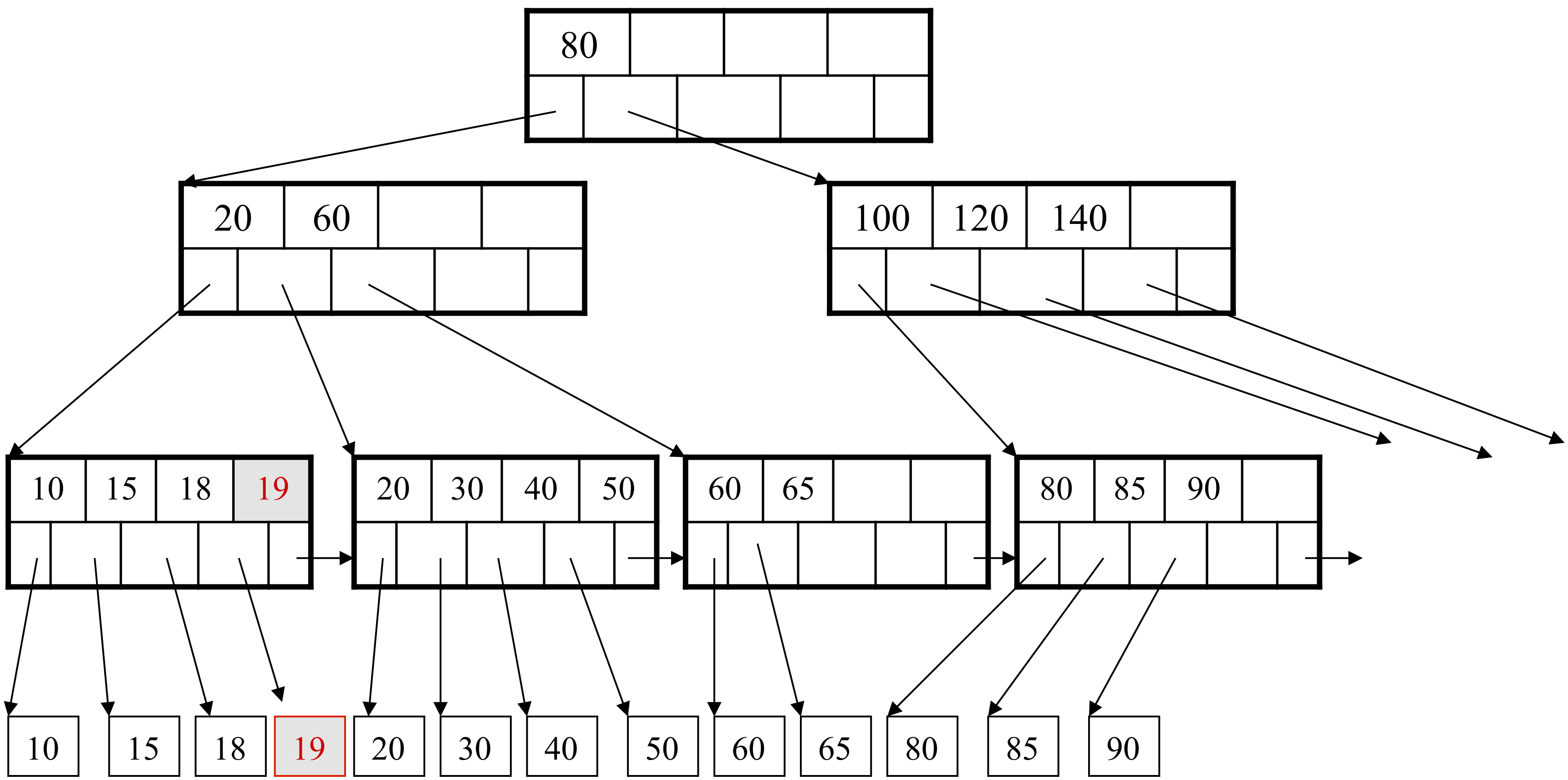
Insertion in a B+ Tree

Insert K=19



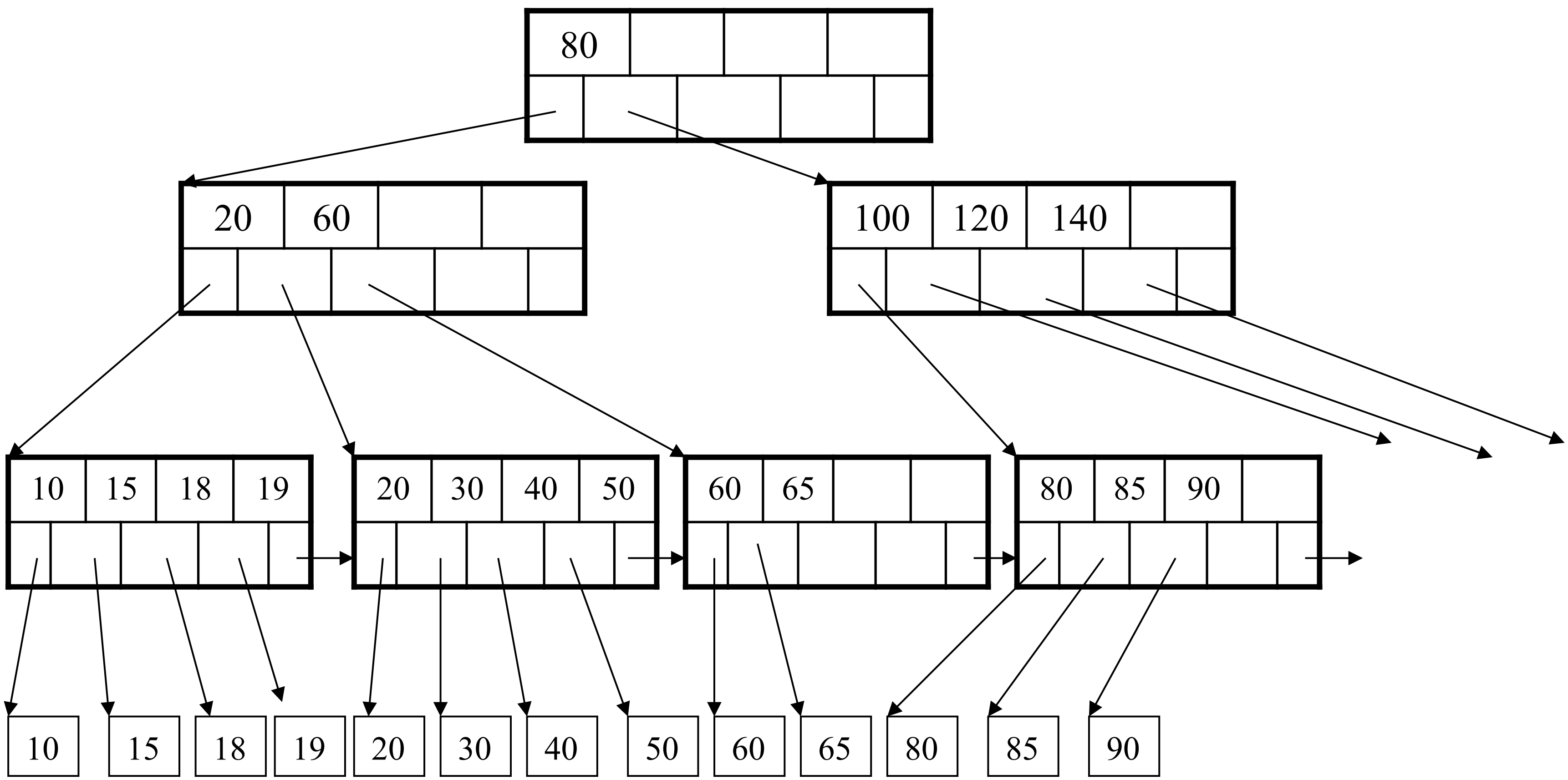
Insertion in a B+ Tree

After insertion



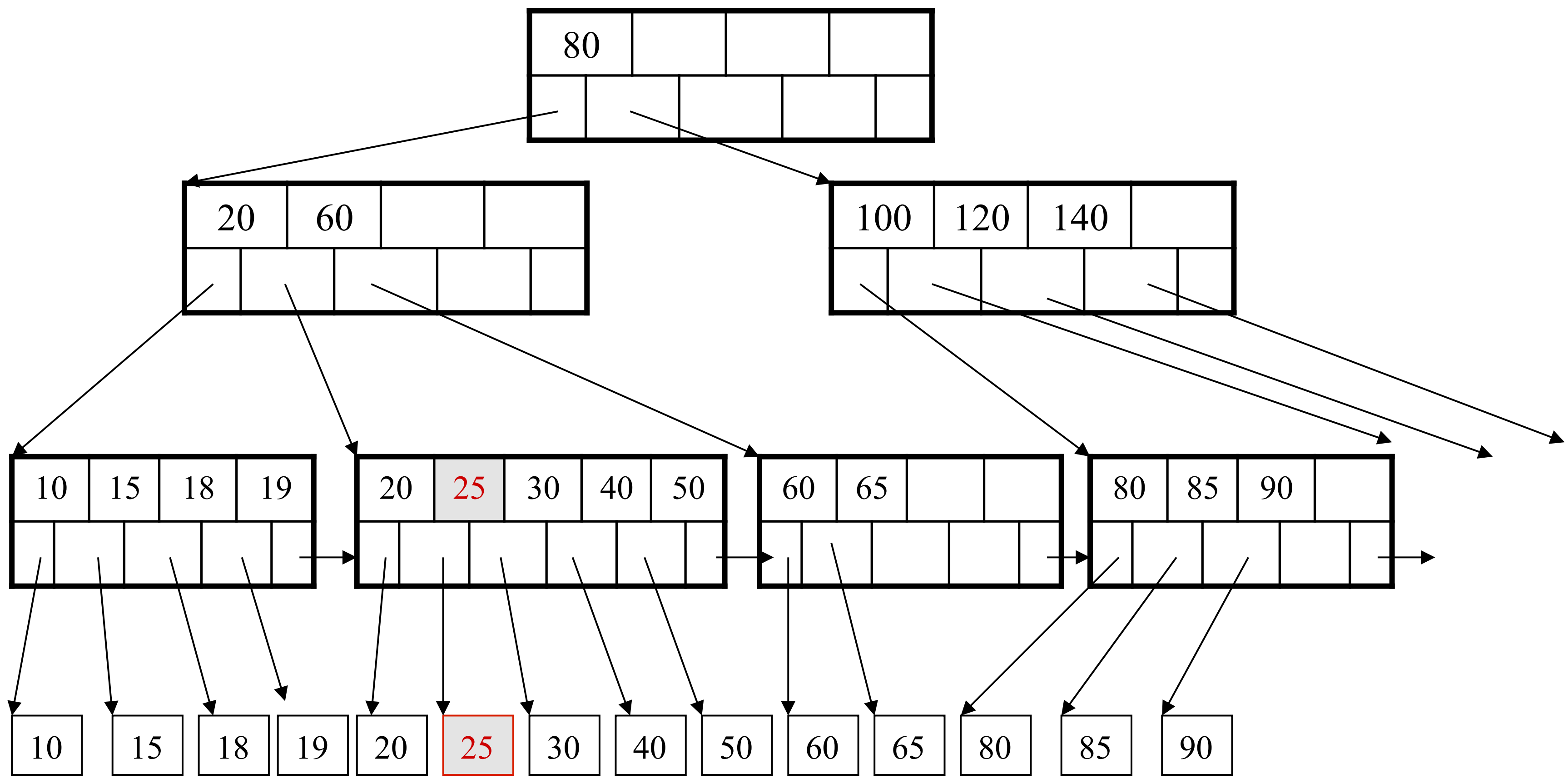
Insertion in a B+ Tree

Now insert 25



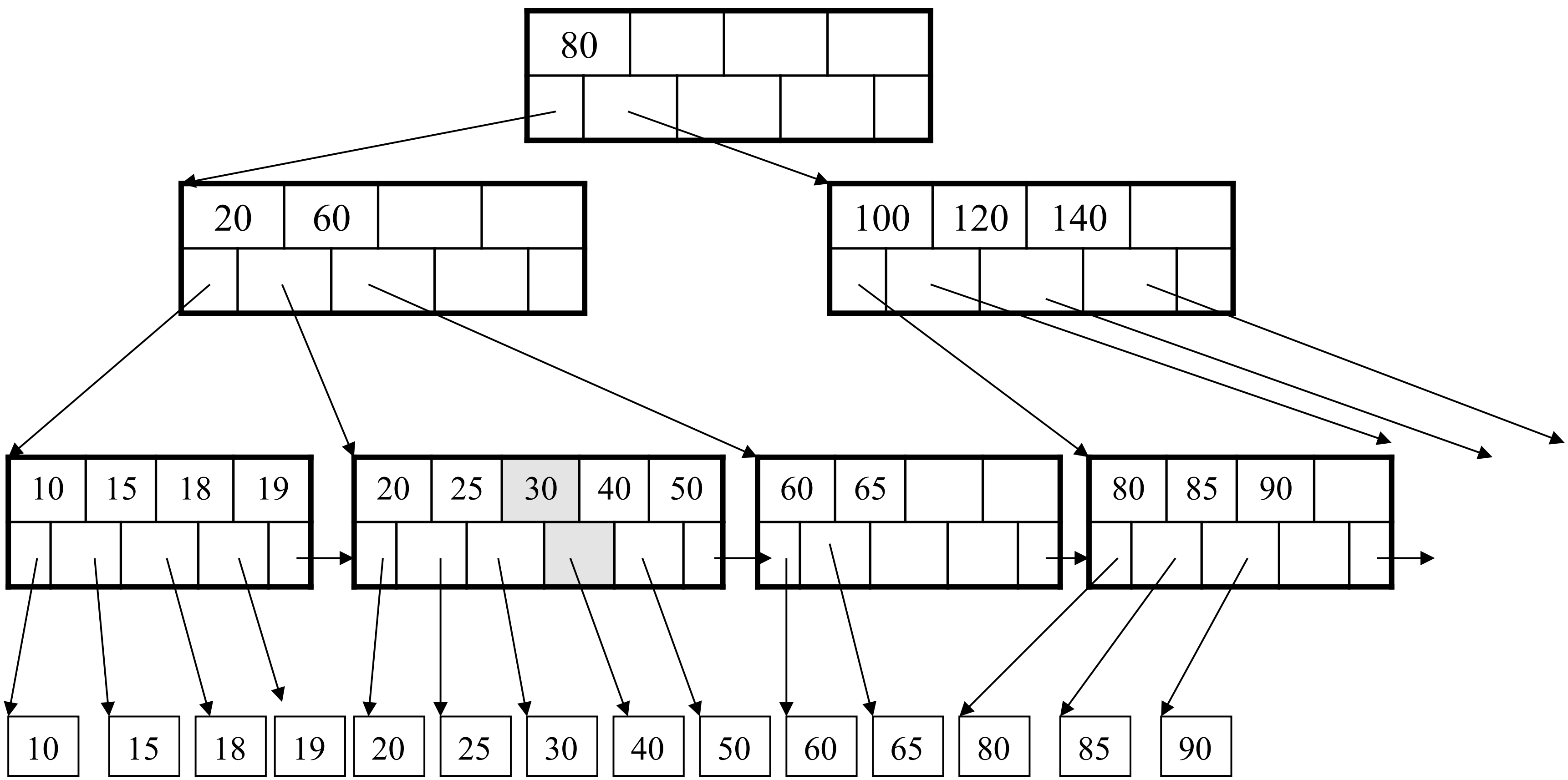
Insertion in a B+ Tree

After insertion



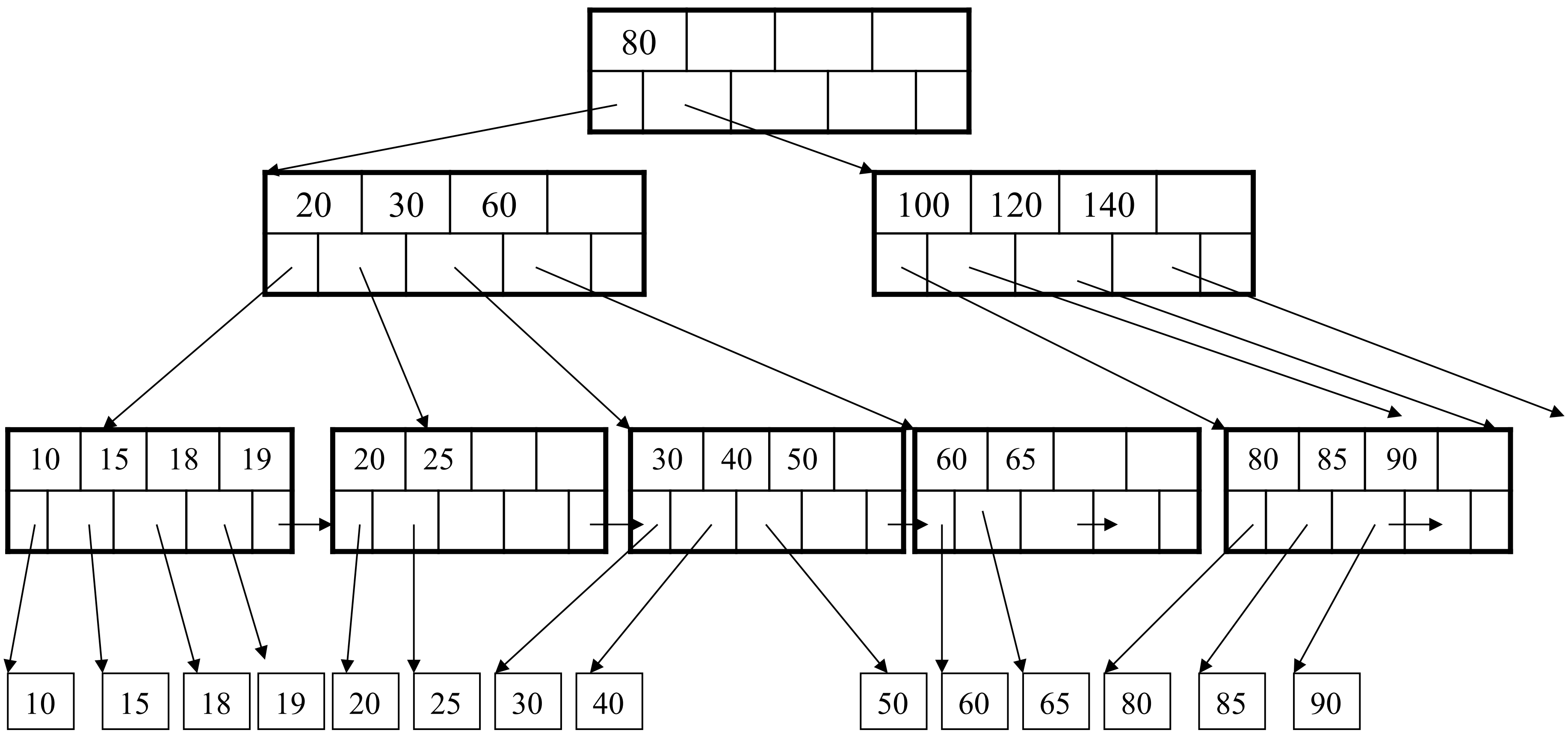
Insertion in a B+ Tree

But now have to split !



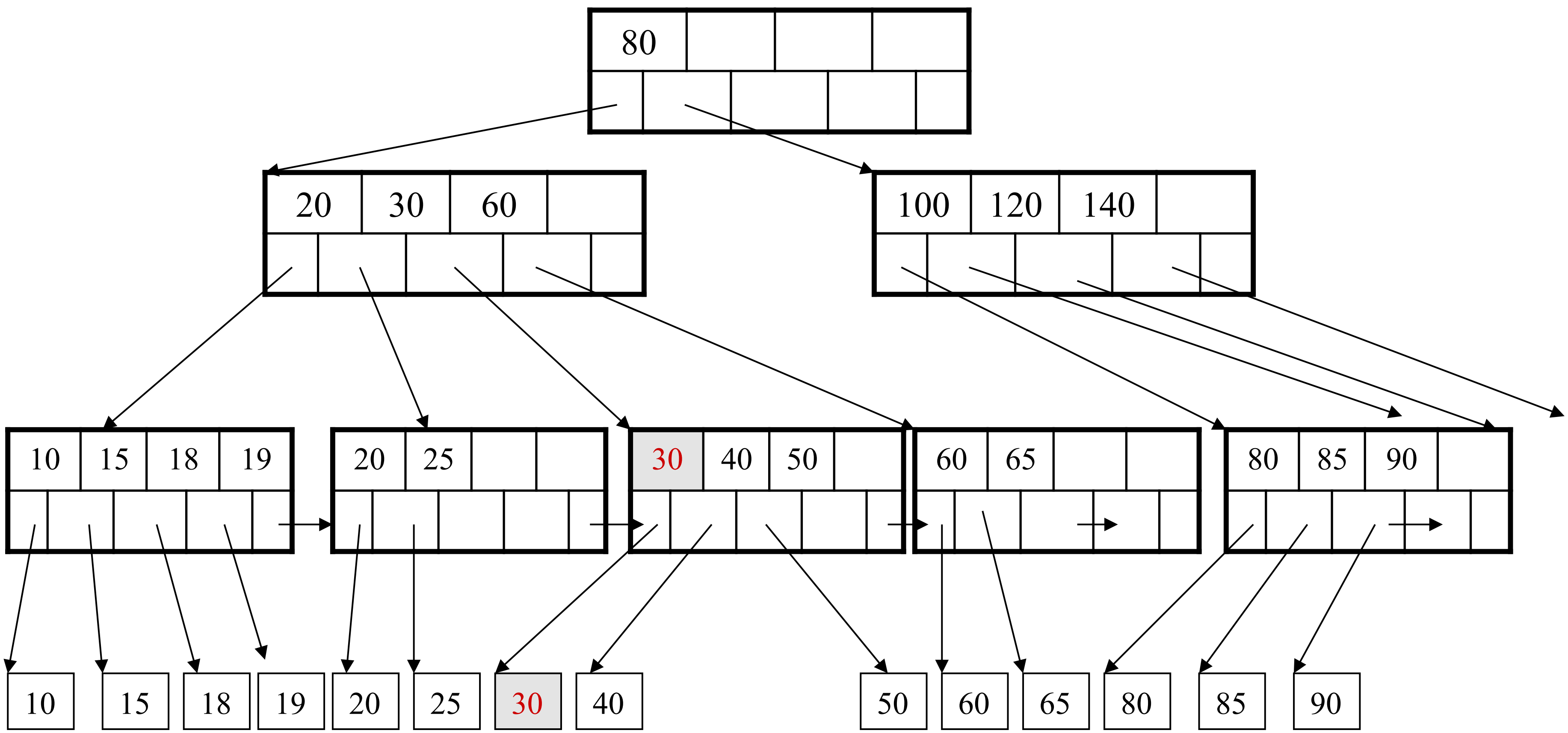
Insertion in a B+ Tree

After the split



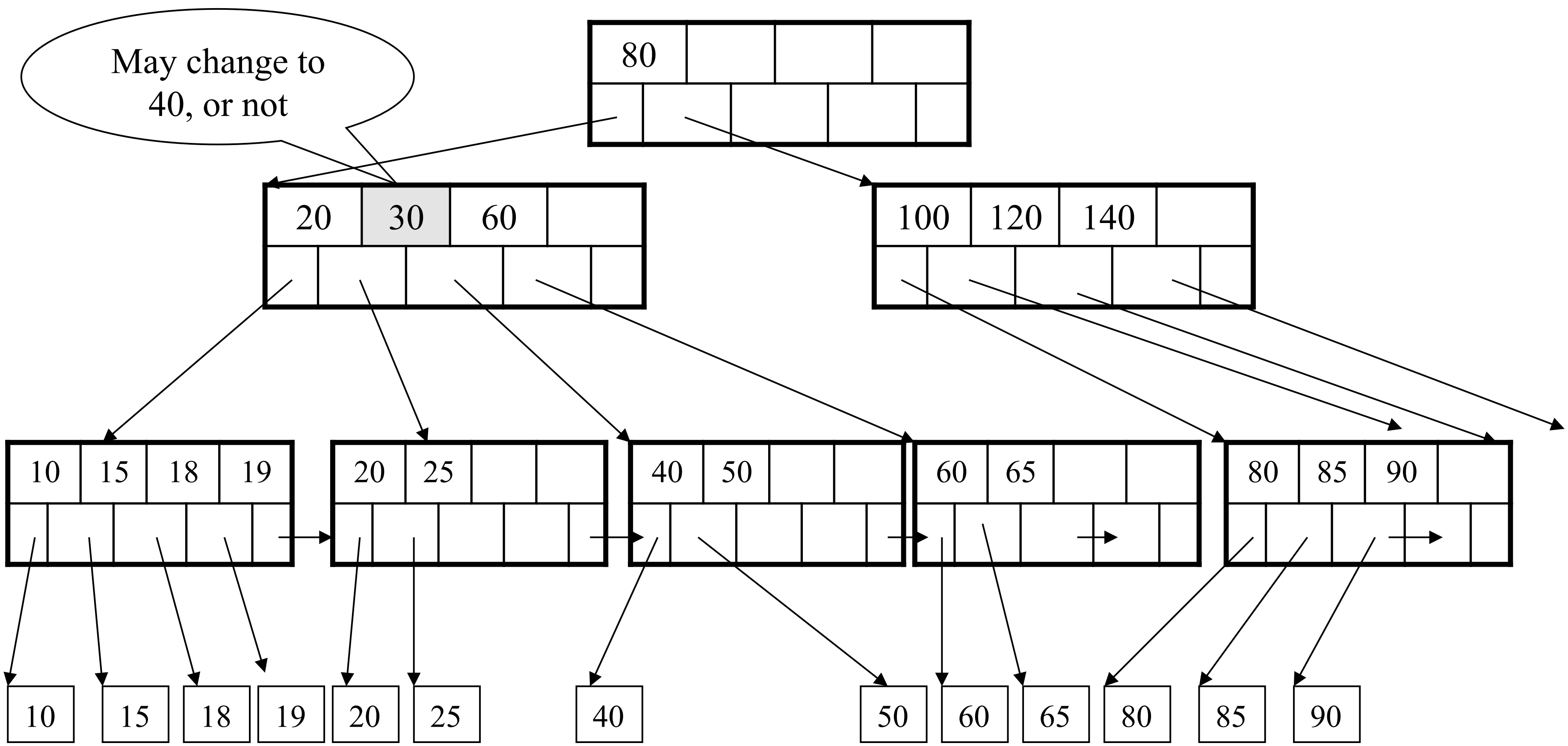
Deletion from a B+ Tree

Delete 30



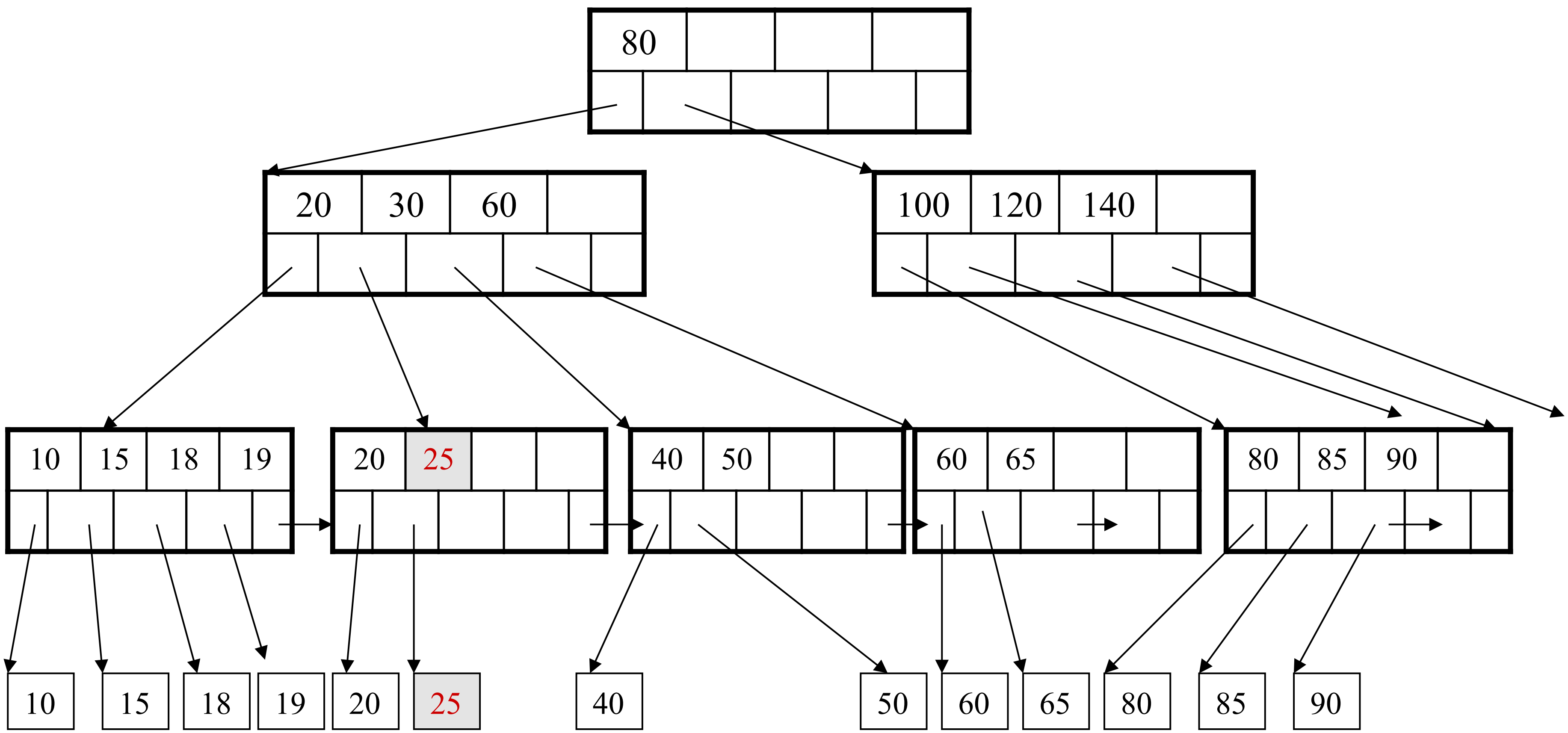
Deletion from a B+ Tree

After deleting 30



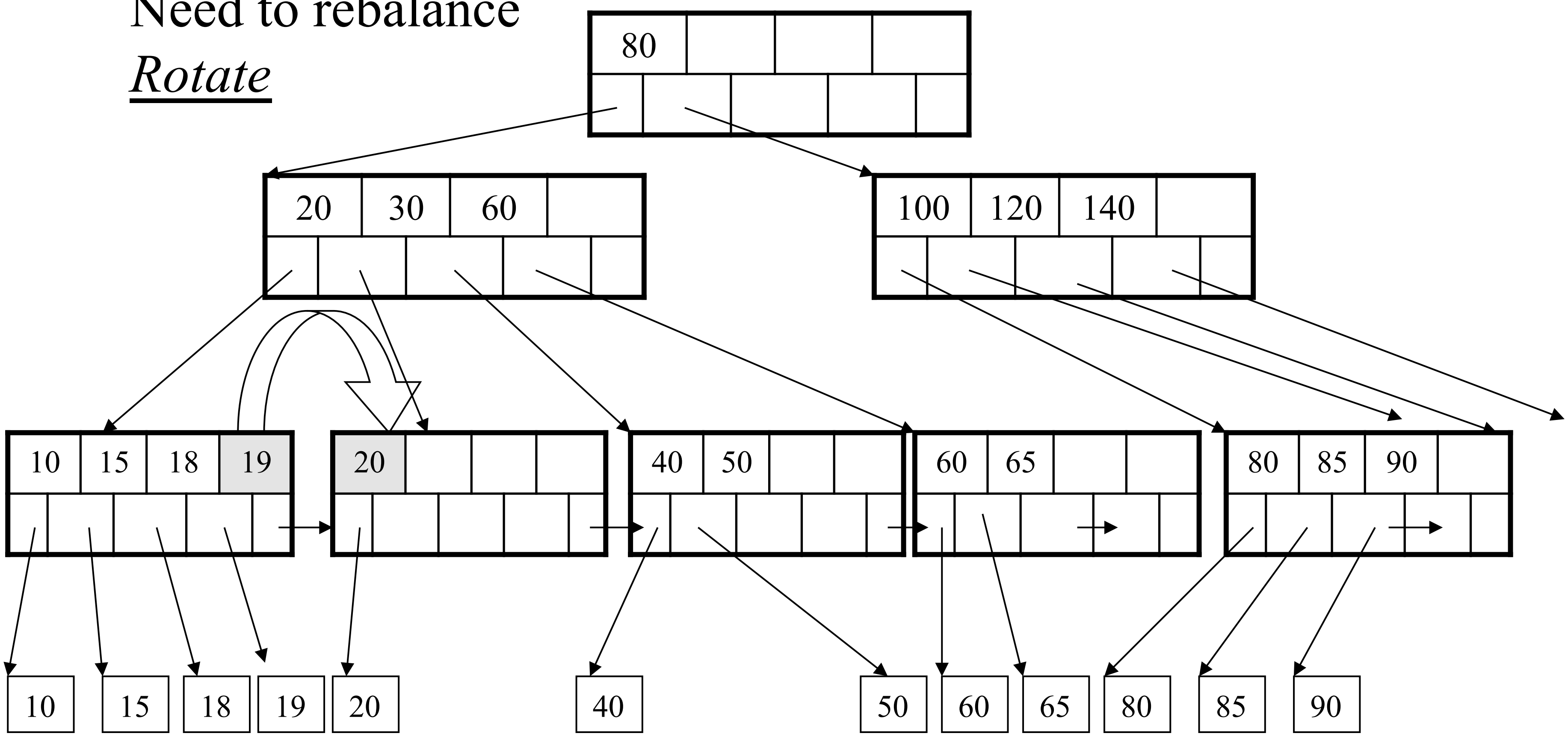
Deletion from a B+ Tree

Now delete 25



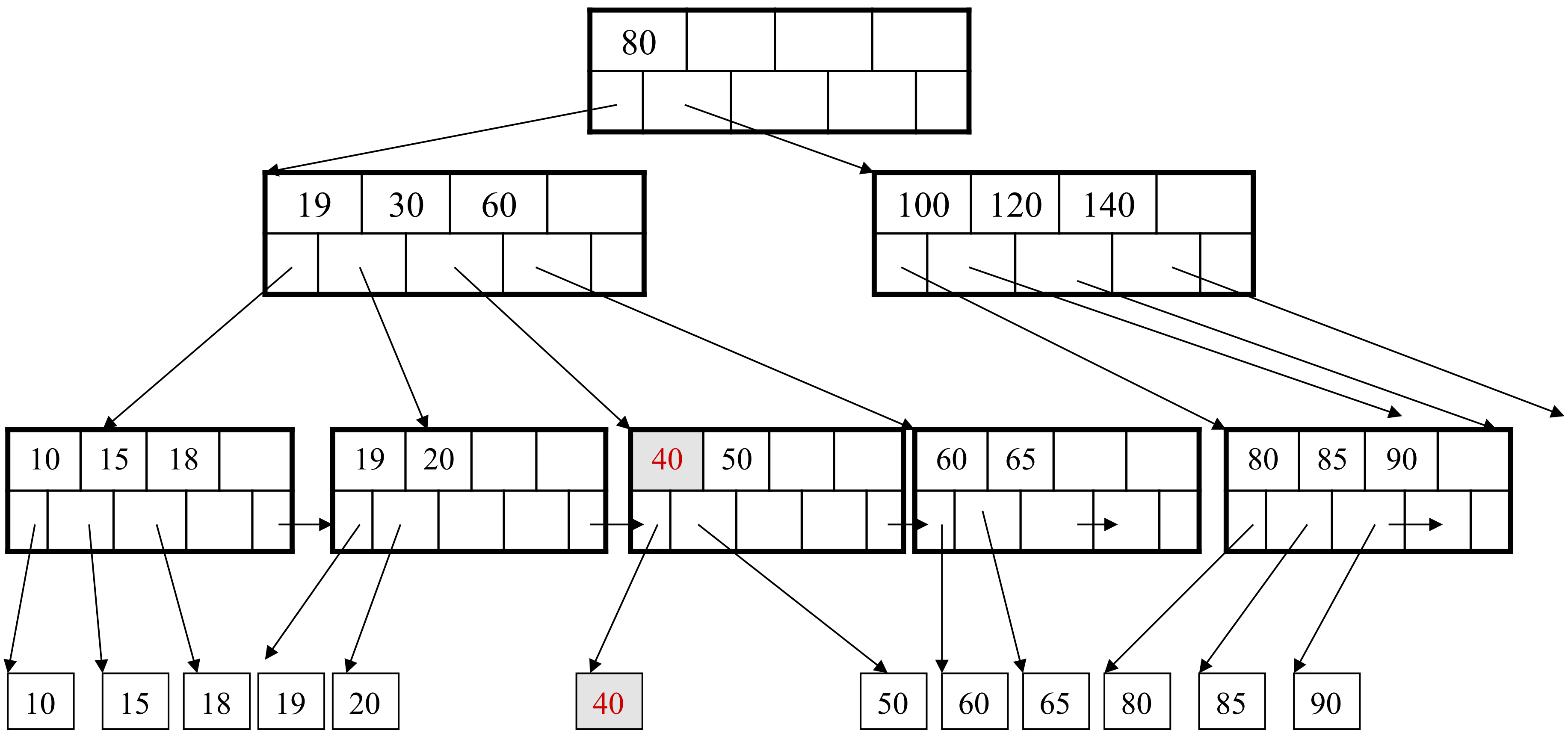
Deletion from a B+ Tree

After deleting 25
Need to rebalance
<u>*Rotate*</u>



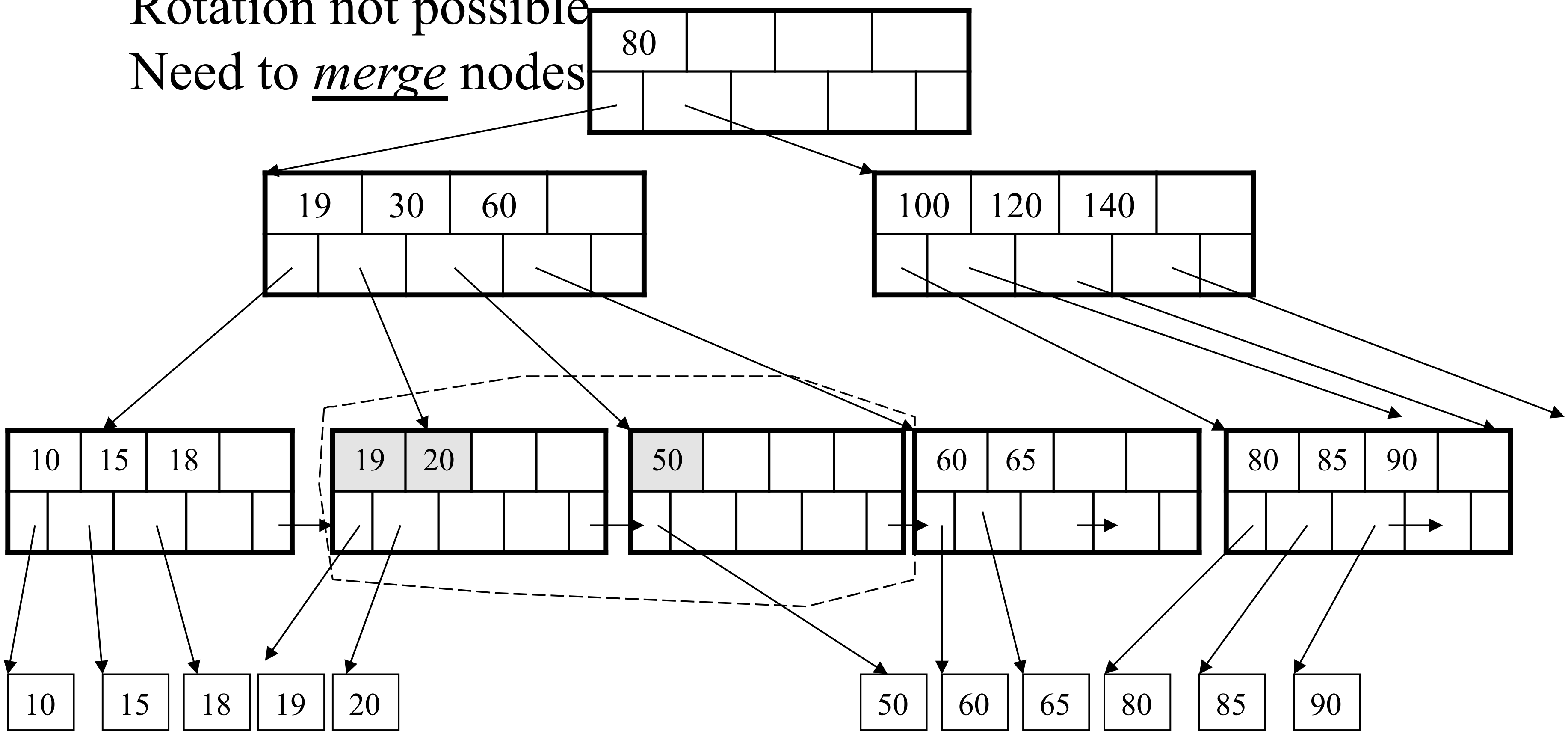
Deletion from a B+ Tree

Now delete 40



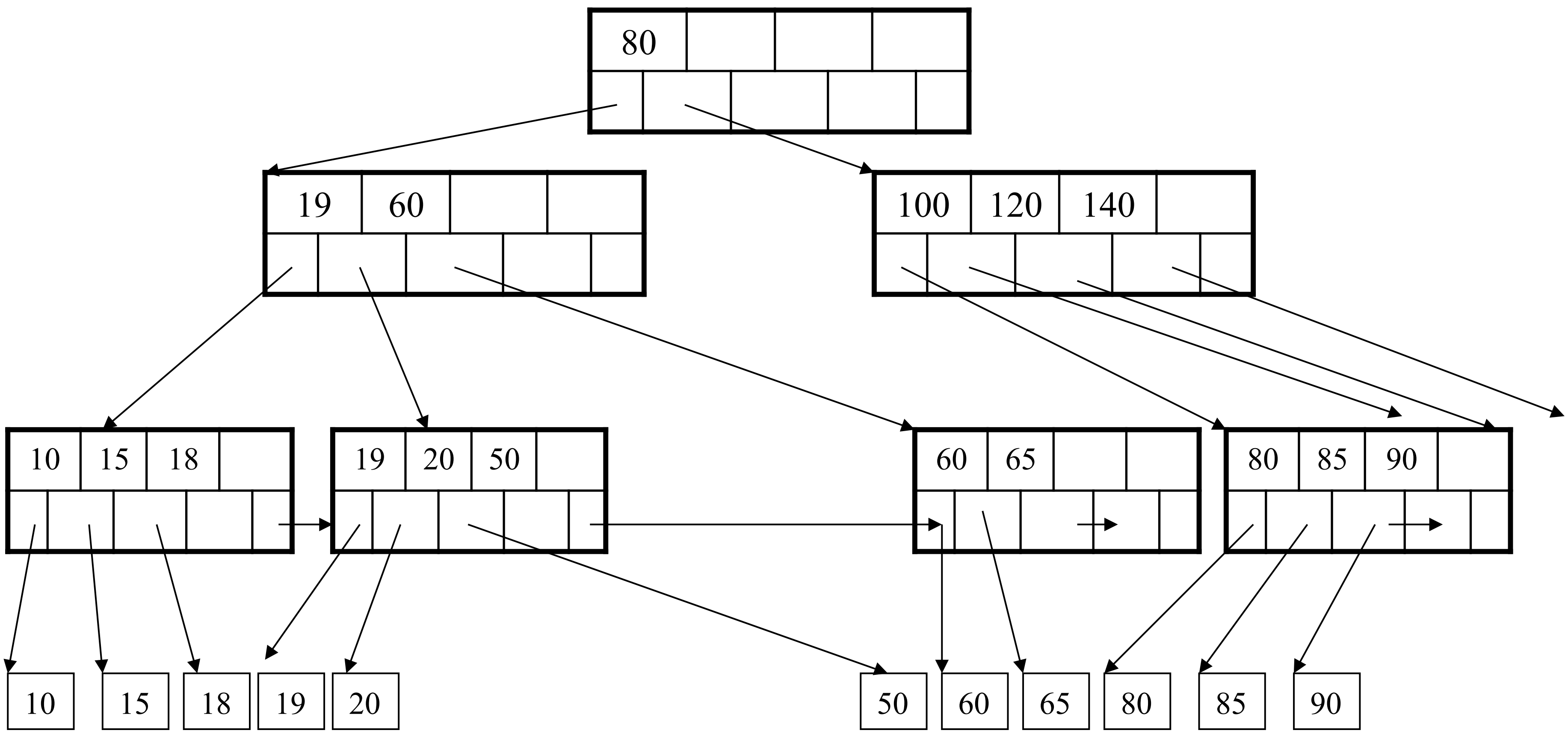
Deletion from a B+ Tree

After deleting 40
Rotation not possible
Need to *merge* nodes



Deletion from a B+ Tree

Final tree



Summary on B+ Trees

- Default index structure on most DBMS
- Very effective at answering ‘point’ queries:
  productName = ‘gizmo’
- Effective for range queries:
  50 < price AND price < 100
- Less effective for multirange:
  50 < price < 100 AND 2 < quant < 20

Indexes in Postgres

```
CREATE TABLE V(M int, N varchar(20), P int);
```

```
CREATE INDEX V1_N ON V(N)
```

```
CREATE INDEX V2 ON V(P, M)
```

```
CREATE INDEX VVV ON V(M, N)
```

```
CLUSTER V USING V2
```

Makes V2 clustered

The Index Selection Problem

- Given a database schema (tables, attributes)
- Given a “query workload”:
  - Workload = a set of (query, frequency) pairs
  - The queries may be both SELECT and updates
  - Frequency = either a count, or a percentage
- Select a set of indexes that optimizes the workload

In general this is a very hard problem

Index Selection Problem 1

V(M, N, P);

Your workload is this

100000 queries:

```
SELECT *
FROM V
WHERE N=?
```

100 queries:

```
SELECT *
FROM V
WHERE P=?
```

What indexes ?

Index Selection Problem 1

V(M, N, P);

Your workload is this

100000 queries:

```
SELECT *
FROM V
WHERE N=?
```

100 queries:

```
SELECT *
FROM V
WHERE P=?
```

A: V(N) and V(P) (hash tables or B-trees)

Index Selection Problem 2

```
V(M, N, P);
```

Your workload is this

100000 queries:

```
SELECT *
FROM V
WHERE N>? and N<?
```

100 queries:

```
SELECT *
FROM V
WHERE P=?
```

100000 queries:

```
INSERT INTO V
VALUES (?, ?, ?)
```

What indexes ?

Index Selection Problem 2

V(M, N, P);

Your workload is this

100000 queries:

```
SELECT *
FROM V
WHERE N>? and N<?
```

100 queries:

```
SELECT *
FROM V
WHERE P=?
```

100000 queries:

```
INSERT INTO V
VALUES (?, ?, ?)
```

A: definitely V(N) (must B-tree); unsure about V(P)

Index Selection Problem 3

```
V(M, N, P);
```

Your workload is this

100000 queries:

```
SELECT *
FROM V
WHERE N=?
```

1000000 queries:

```
SELECT *
FROM V
WHERE N=? and P>?
```

100000 queries:

```
INSERT INTO V
VALUES (?, ?, ?)
```

What indexes ?

Index Selection Problem 3

```
V(M, N, P);
```

Your workload is this

100000 queries:

```
SELECT *
FROM V
WHERE N=?
```

1000000 queries:

```
SELECT *
FROM V
WHERE N=? and P>?
```

100000 queries:

```
INSERT INTO V
VALUES (?, ?, ?)
```

A: V(N, P)

Index Selection Problem 4

```
V(M, N, P);
```

Your workload is this

1000 queries:

```
SELECT *
FROM V
WHERE N>? and N<?
```

100000 queries:

```
SELECT *
FROM V
WHERE P>? and P<?
```

What indexes ?

Index Selection Problem 4

```
V(M, N, P);
```

Your workload is this

1000 queries:

```
SELECT *
FROM V
WHERE N>? and N<?
```

100000 queries:

```
SELECT *
FROM V
WHERE P>? and P<?
```

A: V(N) secondary, V(P) primary index

The Index Selection Problem

- SQL Server:
  - Automatically, through the *AutoAdmin* project
  - Much acclaimed successful research project from mid 90’s, similar ideas adopted by the other major vendors
- Postgres:
  - You will do it manually, part of project 3