Tutorial Week 6

Nishttha Sharma

sharmn99@mcmaster.ca

AGENDA

- Views
- Indexing
- Discussion of Assignment 1

Views

- A view provides a mechanism to hide certain data from the view of certain users
  - Virtual: not stored in the database; just a query for constructing the relation
  - Materialized: actually constructed and stored

- How to Declare?
  - `CREATE [MATERIALIZED] VIEW <name> AS <query>;`
  - A query to specify the view contents
  - Default is virtual

Why use Views?

- Easy to access (read) data without query statements repeatedly.
  - Simplify the query statement for users (ex. Provide joined table, ...)

- Provide limited information to specific users (security)
  - Users cannot know the existence of information
  - Ex.) If an officer (tax, crime, ...) want to check that some specific credit card was used in a specific area last month or not
    - Original DB has all the information about users
    - Make a VIEW for credit card number, area (province, city, ...), time (date)

Views - Example

- Frequents (drinker, bar)

Drinker	Bar
Luke	Grit & Grain
John	Twisted Tap
Sally	Twisted Tap
Lucy	Twist & Sip

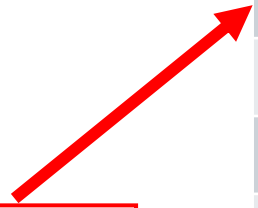
- Sells (bar, beer, price)

Bar	Beer	Price
Grit & Grain	Bud Light	$4.50
Grit & Grain	Coors	$5.00
Twisted Tap	Guinness	$6.50
Twisted Tap	Bud Light	$5.00
Twisted Tap	Coors	$5.50
Twist & Sip	Guinness	$6.00

Views - Example

Create CanDrink (drinker, beer) as a view

```
CREATE VIEW CanDrink AS
    SELECT drinker, beer
    FROM Frequents, Sells
    WHERE Frequents.bar = Sells.bar;
```



CanDrink

Drinker	Beer
Luke	Bud Light
Luke	Coors
John	Guinness
John	Bud Light
John	Coors
Sally	Guinness
Sally	Bud Light
Sally	Coors
Lucy	Guinness

Views - Example

Create CanDrink (drinker, beer) as a view

Drinker	Beer
Luke	Bud Light
Luke	Coors
John	Guinness
John	Bud Light
John	Coors
Sally	Guinness
Sally	Bud Light
Sally	Coors
Lucy	Guinness

A query on the view -

```
SELECT beer
FROM CanDrink
WHERE drinker = 'Sally';
```

Same query without the view -

```
SELECT beer
FROM (SELECT drinker, beer
        FROM Frequents, Sells
        WHERE Frequents.bar = Sells.bar)
WHERE drinker = 'Sally';
```

Indexing

- Disk-based structures linked to tables or views.
- Efficient retrieval of records.
  - Fast search: without index, system needs full table scan.
  - Works well with MIN, MAX, ORDER BY.
- Requires additional space and operations (insert, delete, update).
- When to use?
  - Large tables.
  - Infrequent insert/delete/update; mostly read.
  - Less duplicated data.

Indexing Types

Clustered

- Created when both these conditions are satisfied:
  - Data should be sorted.
  - There should be a key value (i.e., it cannot have repeated values).
- A table can have at most one clustered index.
- If you set the primary key, it automatically acts as the ‘clustered index.’
- It offers faster retrieval but may slow down insert and update operations.

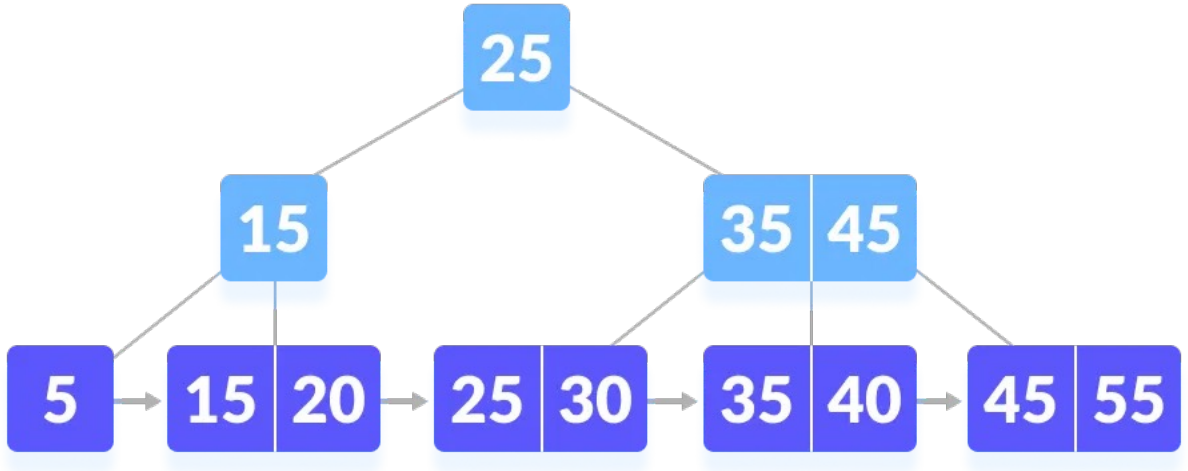
Indexing Types

Non-Clustered

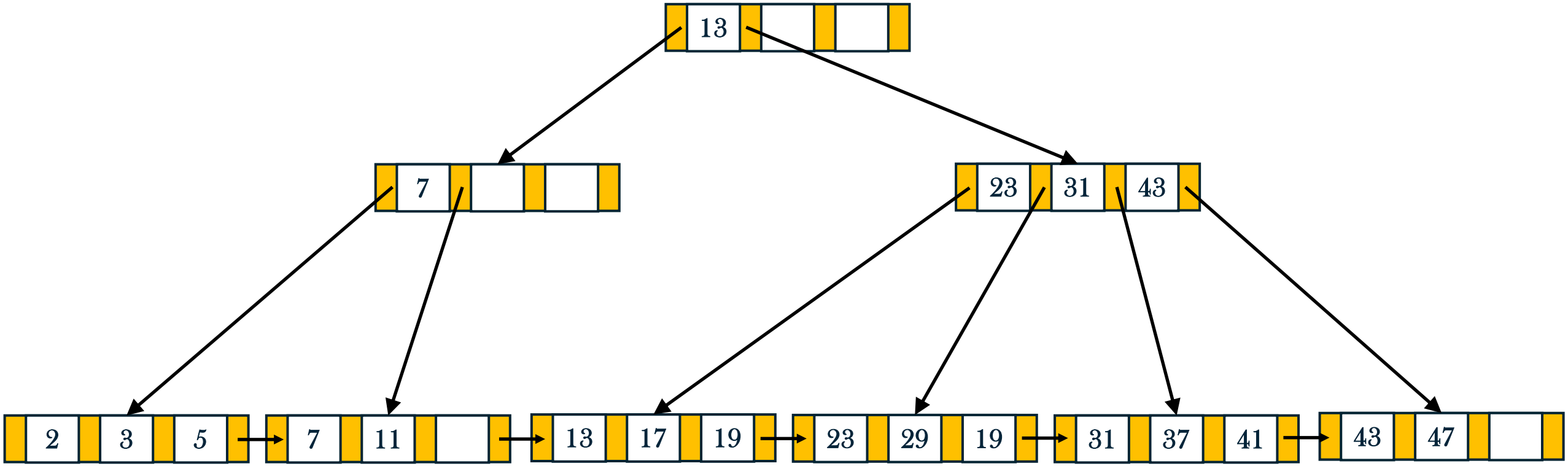
- Multiple non-clustered indexes are allowed per table.
- The non-clustered index stores both the value and a pointer to the actual row that holds the data.
- They offer flexibility but may result in slower retrieval compared to clustered indexes.

B+ Trees

- Self-balancing tree.
- Each node at least 50% full.
- Actual data is saved in leaf nodes.
- Leaf nodes are connected via a linked list.
- Better performance for “<, >” compared to Hash table.

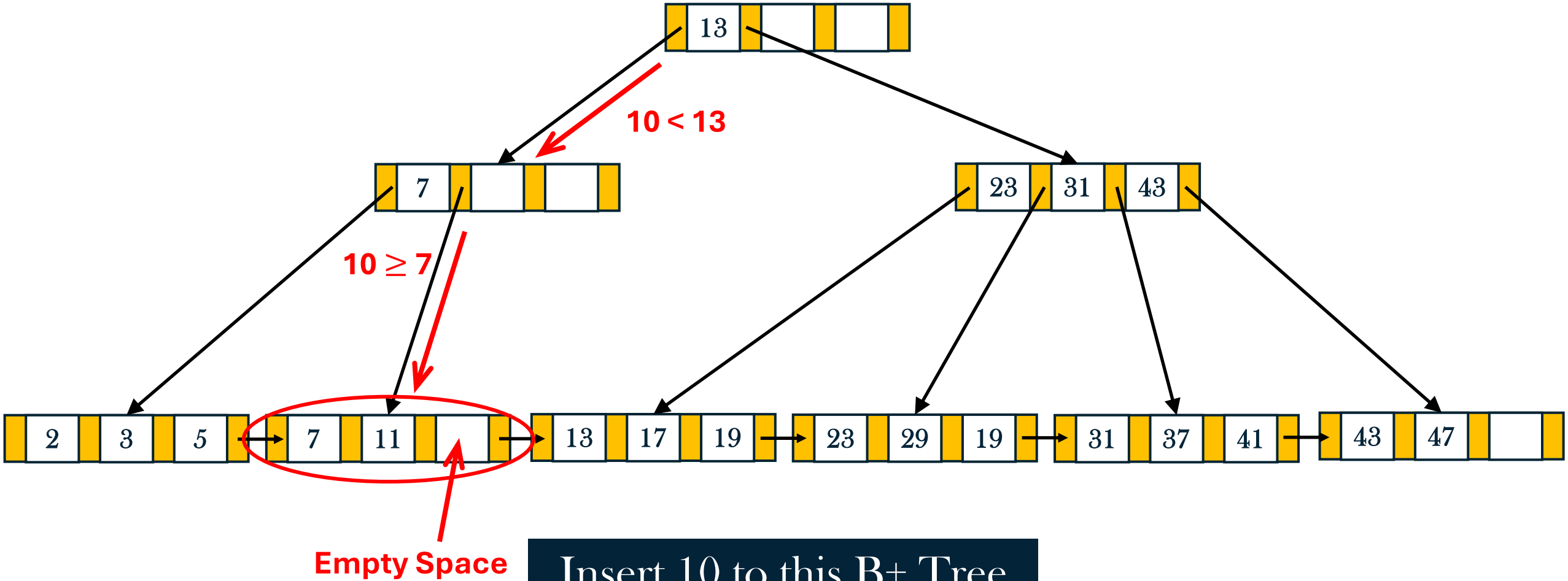


B+ Trees – Insertion

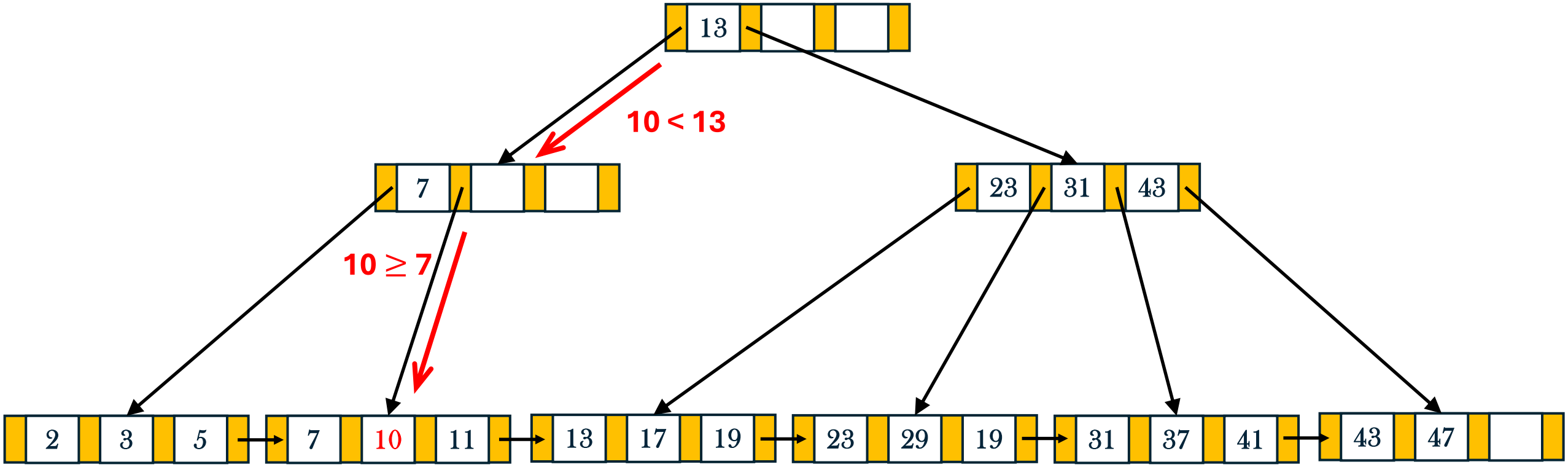


Insert 10 to this B+ Tree

B+ Trees – Insertion

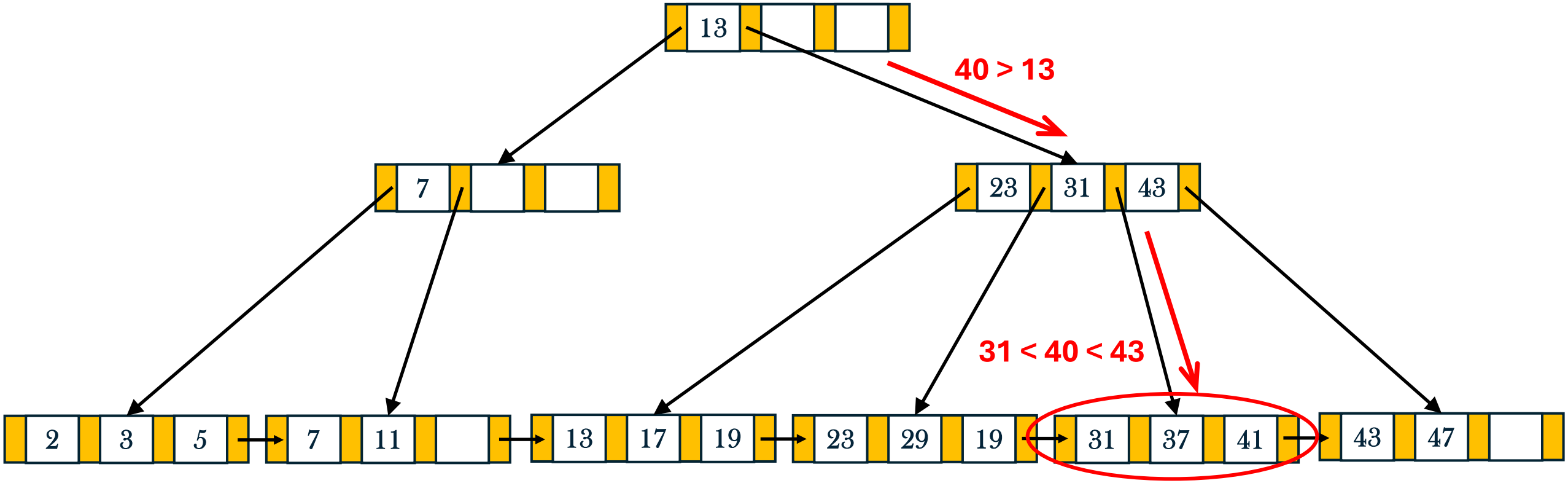


B+ Trees – Insertion



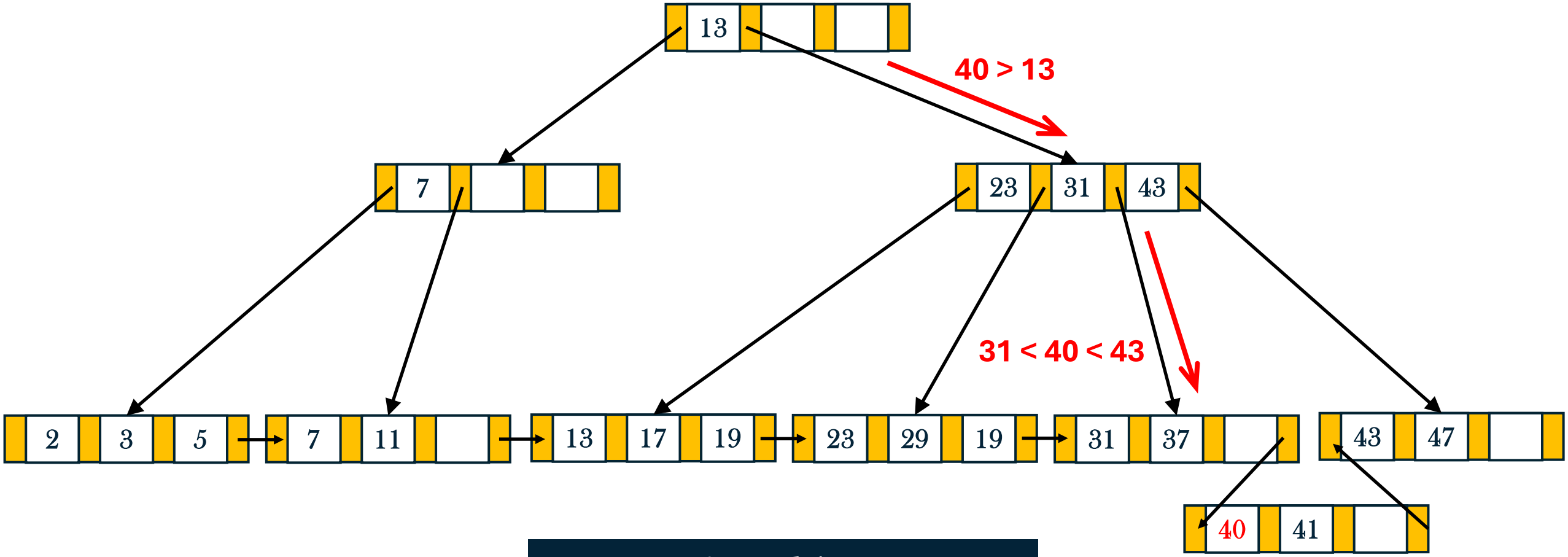
Insert 10 to this B+ Tree

B+ Trees – Insertion



Insert 40 to this B+ Tree

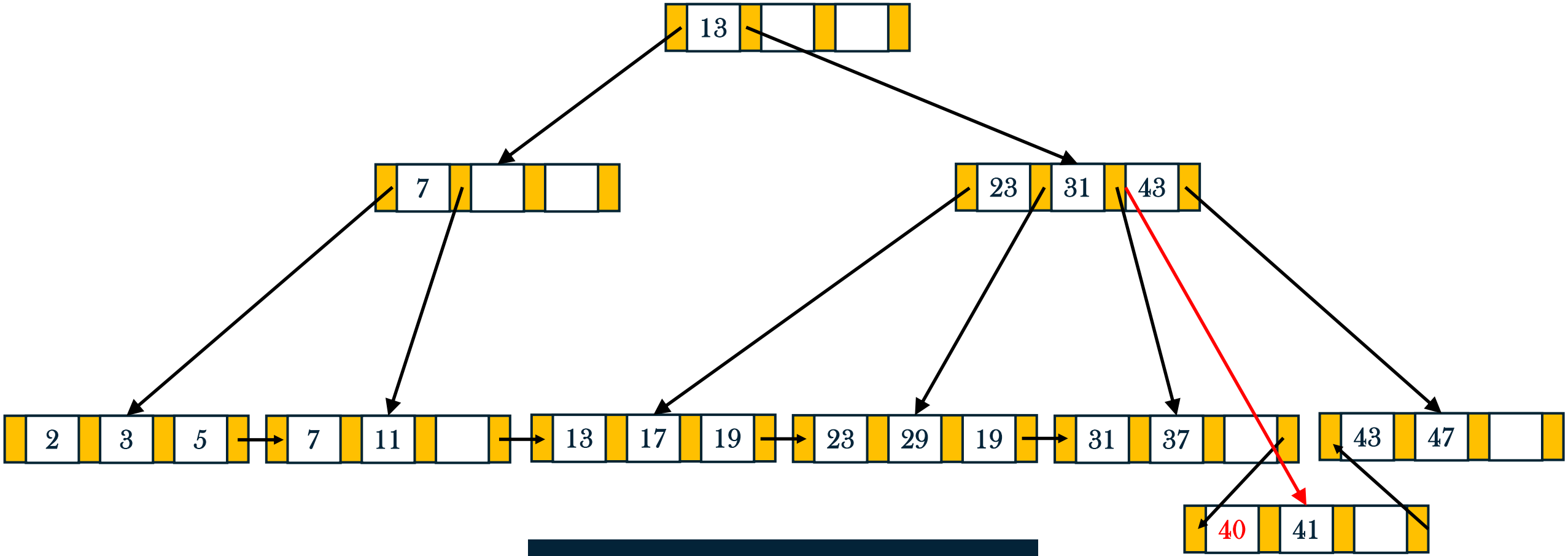
B+ Trees – Insertion



Insert 40 to this B+ Tree

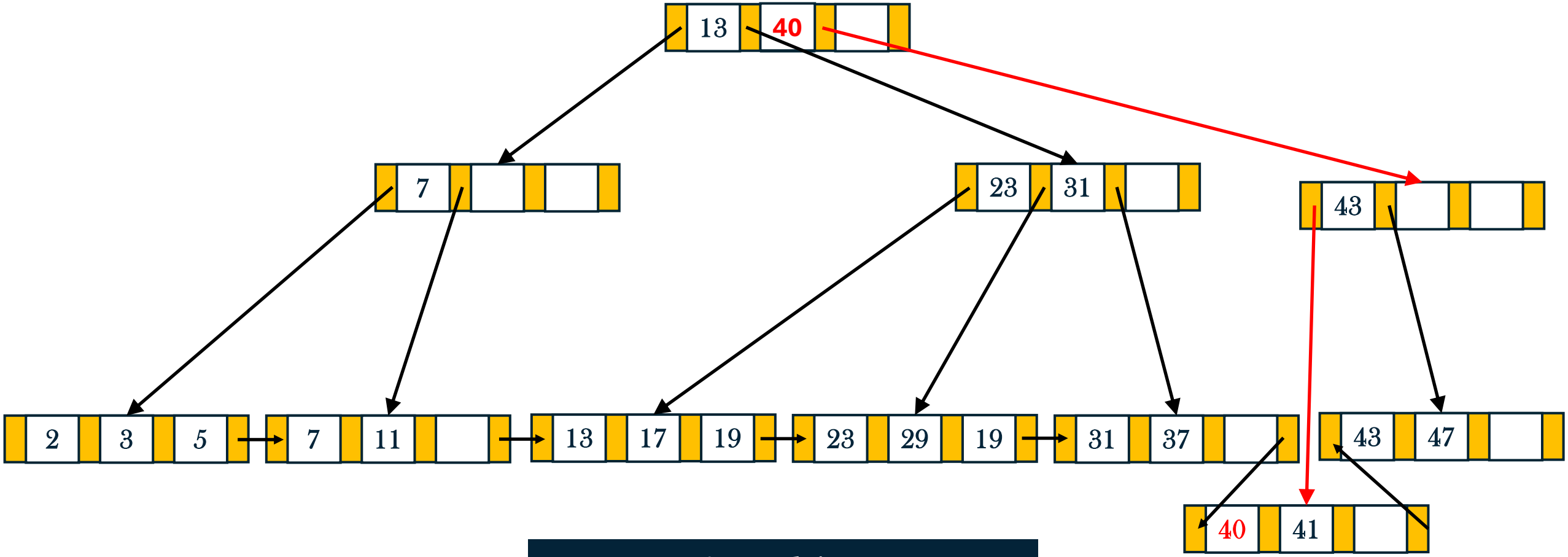
NEW NODE – needs to be linked to the tree

B+ Trees – Insertion



Insert 40 to this B+ Tree

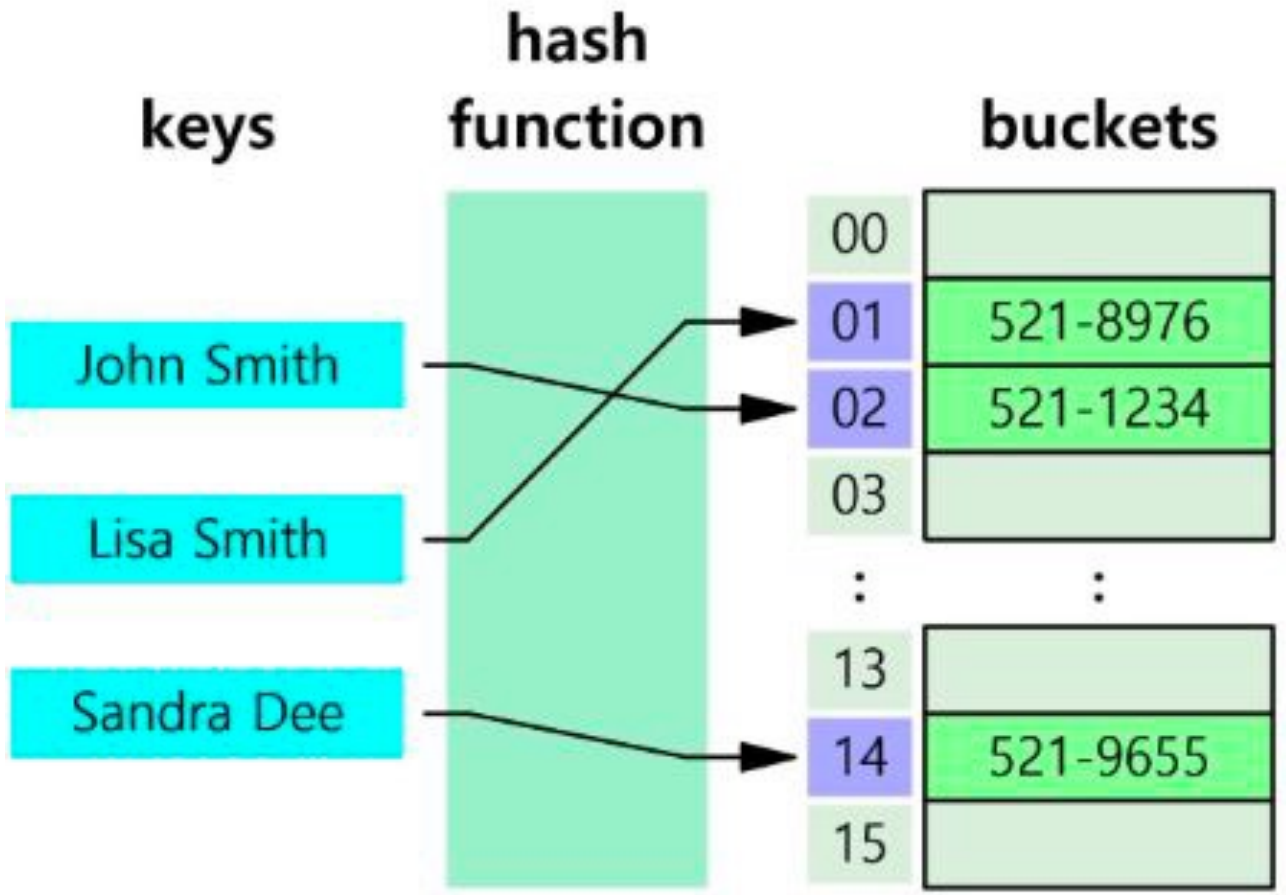
B+ Trees – Insertion



Insert 40 to this B+ Tree

Hashing

- Good for equality searches.
- Hash function *h* returns value (address).
- (Key, Value) pairs.
- Can have collisions.
- Performs best when the data is discrete and random.



For Assignment 2

- How to save the result of query?
- Your queries are in a file queries.sql

```
$ db2 -tnf queries.sql > queries.results
```

- Copy queries.results to your local system

```
$ scp sharmn99@se3db3.cas.mcmaster.ca:/u40/sharmn99/queries.results
/Users/nishttha/Desktop
```

```
Last login: Tue Oct 17 09:11:25 on ttys000
(base) nishttha@Nishtthas-MacBook-Air ~ % cd Desktop
(base) nishttha@Nishtthas-MacBook-Air Desktop % scp queries.sql sharmn99@se3db3.
cas.mcmaster.ca:/u40/sharmn99/
sharmn99@se3db3.cas.mcmaster.ca's password:
queries.sql                                   100%  728    26.3KB/s   00:00
(base) nishttha@Nishtthas-MacBook-Air Desktop % ssh sharmn99@se3db3.cas.mcmaster
.ca
sharmn99@se3db3.cas.mcmaster.ca's password:
Last login: Tue Oct 17 09:11:54 2023 from 172.18.194.190
[sharmn99@se3db3 ~] ls
createTables.ddl  dropscript.sql  insert_script.ddl  queries.sql
[sharmn99@se3db3 ~] db2 -tnf queries.sql>queries.results
[sharmn99@se3db3 ~] ls
createTables.ddl  insert_script.ddl  queries.sql
dropscript.sql    queries.results
[sharmn99@se3db3 ~]
```

```
Connection to se3db3.cas.mcmaster.ca closed.
(base) nishttha@Nishtthas-MacBook-Air Desktop % scp sharmn99@se3db3.cas.mcmaster
.ca:/u40/sharmn99/queries.results /Users/nishttha/Desktop
sharmn99@se3db3.cas.mcmaster.ca's password:
queries.results                               100% 3892    73.8KB/s   00:00
(base) nishttha@Nishtthas-MacBook-Air Desktop % ls
Screenshot 2023-10-17 at 9.13.29 AM.png
queries.results
queries.sql
(base) nishttha@Nishtthas-MacBook-Air Desktop %
```