

SCALABLE DATA MANAGEMENT SYSTEMS

LAB 0 ANNOUNCEMENT

29. OCT. 2021



TECHNISCHE
UNIVERSITÄT
DARMSTADT

AGENDA

- **Lab 0 content**
- Submission Process & GitLab Hints
- Recap: LRU-K and B-trees
- Paper exercise

THE DATA MANAGEMENT-DB (DMDB)

- A simple database implementation in JAVA
- Goal: Is to help you understand how databases are implemented
- We will implement concepts that are taught in the lecture

LAB 0 – OVERVIEW

Deadline: 12.11.2020, 09:50am (i.e., before the presentation of the next lab/solution of the last lab)

Gives **2 bonus points**

Part 1: Datatype serialization

Part 2: Data persistence (disk manager)

Both parts do not need any specific database knowledge!

Reminder: all labs need to be solved *alone*, therefore working in groups is not permitted.

LAB 0 – PART 1

DMDB stores data as bytes on disk.

As explained in the lecture, in order to increase efficiency, data is written/read in fixed chunks of bytes (= pages).

To create those pages we must serialize the data (items) that we want to store. To read existing data from the pages we must de-serialize:

- **Serialize:** Convert values of a datatype to bytes
- **Deserialize:** Convert bytes to values of a datatype

Example: 1-Byte TINYINT

71 (TINYINT) \Leftrightarrow 0100 0111 (binary)

LAB 0 – PART 1

Complete the implementation of the JAVA Classes for the data types SQLInteger (fixed length of 4 byte), SQLBigInteger (fixed length of 8 byte) and SQLVarchar (variable length character), implement the methods:

- `void deserialize(byte[] data)`: Deserialize the value of the data type from the provided byte array.
- `byte[] serialize()`: Serialize the value of the data type into a byte array.

Requirement: Use Bit-Operations for SQLInteger/SQLBigInteger and pay attention to masking in Java.

You may use `String.getBytes()` when implementing SQLVarchar!

LAB 0 – PART 2

The DiskManager is responsible to persist pages to disk. DMDB stores all data in a single file called `testdb.sdms` (see `AbstractDiskManager.DB_FILENAME`).

Complete the class `DiskManager` by implementing the required methods:

- `void close()`: Close File descriptor and make sure everything in consistent state
- `AbstractPage readPage(Integer pageId)`: Read a page from Disk and return it
- `void writePage(AbstractPage page)`: Write a given page to disk

HINTS

You define both the serialization and deserialization of data types – the concrete implementation is your choice.

Page de/serialization is already provided.

You can use standard java.io classes to interact with the disk/filesystem – no external dependencies are required.

GENERAL REQUIREMENTS & HINTS

- Please document your code!
- You should use the basic **Unit-Tests** in the package `de.tuda.dmdb.test` to validate your implementation.
- Have a look at the tests to also learn how the interfaces are used and how different components relate to each other.
- It is recommended to implement further tests to cover all scenarios/edge cases – the initially provided test cases don't do that.
- **The description of advanced test cases (i.e., more information about what the tests do) is found under `de.tuda.dmdb.*.advanced`.**

GENERAL REQUIREMENTS & HINTS

- Have a look at the Abstract and Base classes to see what functions can be reused and to get some inspiration.
- **Feel free to add additional helper methods in the classes** to structure your code and make it modular.
- **In general, you do not need to add additional classes/files!** This either means we have forgotten something (=> pls. come and talk to us to fix it for everybody) or you might need to have another look at the code base
- Use meaningful standard JAVA exception classes to throw exceptions (e.g., `IllegalArgumentException`, `RuntimeException`)

GENERAL REQUIREMENTS & HINTS

Reference environment:

- Java 11 (we use openjdk11)
- Junit 5
- Gradle 6.7

PROJECT CODE STRUCTURE

```
src/main/java
├── de.tuda.dmdb.buffer
├── de.tuda.dmdb.buffer.exercise
├── de.tuda.dmdb.catalog
├── de.tuda.dmdb.catalog.objects
├── de.tuda.dmdb.storage
├── de.tuda.dmdb.storage.exercise
├── de.tuda.dmdb.storage.types
└── src/test/java
    ├── de.tuda.dmdb
    ├── de.tuda.dmdb.buffer
    └── de.tuda.dmdb.storage
JRE System Library [JavaSE-11]
Project and External Dependencies
bin
libs
└── exerciseCommonBase.jar
src
├── build.gradle
└── User.txt
```

The `.exercise` packages contain the classes that you have to complete. Those classes extend superclasses from other packages!

Here, you will find the basic tests. Later, we will add `.advanced` packages that only contain the specification.

We use this folder to provide you with dependencies that you need to build the project. (Think: classes/concepts that we implemented for you)

You have to update this file with your Matriculation number

AGENDA

- Lab 0 content
- **Submission Process & GitLab Hints**

SDMS GITLAB INSTANCE

Gitlab is used to both, publish labs (by us) and submit solutions (by you)

Gitlab URL: <https://gitlab.dm.informatik.tu-darmstadt.de/>

Please register by completing the questionnaire on Moodle!

If you already registered on Moodle for the lab, you should have gotten an email with login on the email you have registered in your Moodle account.

If you didn't register yet, we will occasionally create new accounts.

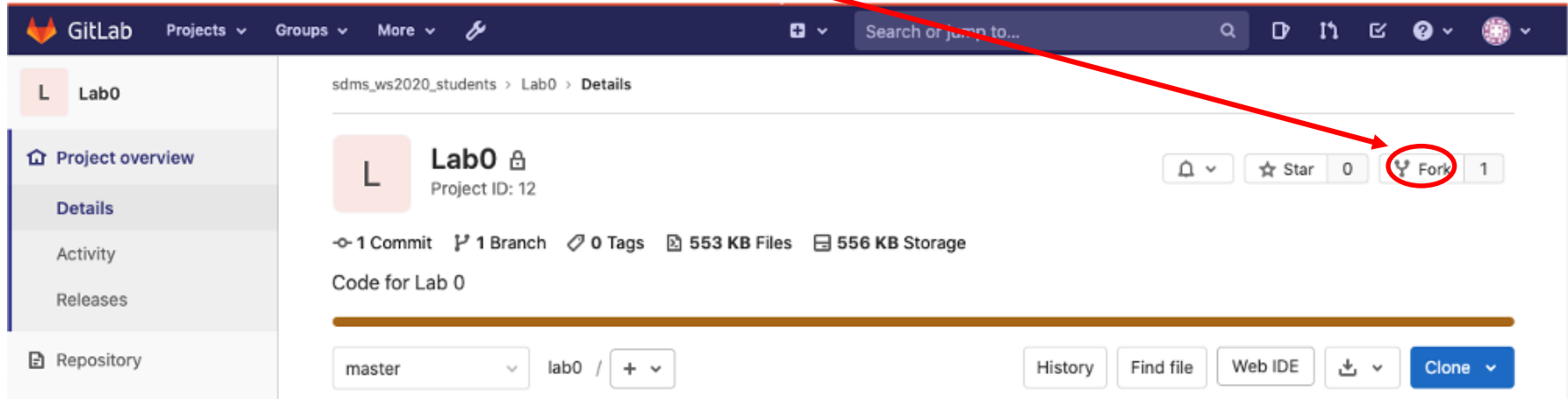
EXERCISE SUBMISSION PROCESS

1. Fork project from `sdms_ws2021_students` group in GitLab (this will create a private copy that no other student will see)
2. Clone your own repository to your local development environment.
3. Add your Matriculation number to the `User.txt` file
4. Implement and test code (locally)
5. (Optional) Pull changes from the upstream repository
6. Each push to master branch triggers tests evaluation (you can push multiple times, until the deadline)

FORK PROJECT FROM SDMS_WS2021_STUDENTS

We release the stencil code for the Project under the sdms_ws2021_students group (Groups -> Your Groups)

This Project needs to be forked in order to complete the task successfully!



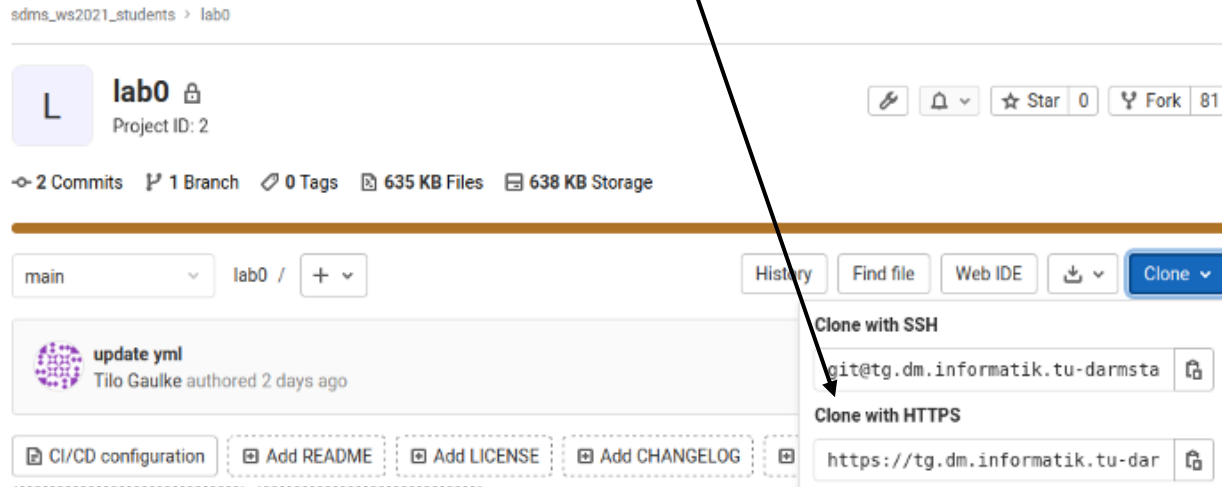
The screenshot shows the GitLab web interface for a project named 'Lab0' under the 'sdms_ws2020_students' group. The left sidebar contains navigation links: 'Project overview', 'Details' (selected), 'Activity', 'Releases', and 'Repository'. The main content area displays the project details, including the project name 'Lab0', its ID '12', and statistics: '1 Commit', '1 Branch', '0 Tags', '553 KB Files', and '556 KB Storage'. At the bottom of the main content area, there are buttons for 'History', 'Find file', 'Web IDE', and 'Clone'. A red arrow points from the text 'This Project needs to be forked' to the 'Fork' button, which is circled in red.

PULL CODE FROM SERVER

You can use your favorite Git client. We use the git command line:

```
git clone https://tg.dm.informatik.tu-darmstadt.de/NAME/lab0.git
```

You can copy the link from here



The screenshot shows a web interface for a Git repository named 'lab0'. The repository is owned by 'sdms_ws2021_students' and has a Project ID of 2. It features 2 commits, 1 branch, 0 tags, 635 KB of files, and 638 KB of storage. A 'Clone' button is visible, which has a dropdown menu open. The dropdown menu shows two options: 'Clone with SSH' with the link 'git@tg.dm.informatik.tu-darmsta' and 'Clone with HTTPS' with the link 'https://tg.dm.informatik.tu-dar'. An arrow points from the text 'You can copy the link from here' to the SSH link in the dropdown menu.

IMPLEMENT AND TEST LOCALLY I

Update the file `User.txt` with your matriculation number, the line should look like: `MatriculationNumber=1234567`

You can use your favorite IDE to work on the code.

We use Gradle to build and test the project on the evaluation server. The Gradle binary `gradlew(.bat)` is already included in the project.

The `build.gradle` file in the project should help you to import the project into the IDE of your choice!

IMPLEMENT AND TEST LOCALLY II

You can run the test locally using the `test` task. E.g., in the command line you can run:

```
./gradlew test
```

(OPTIONAL) PULL CHANGES FROM UPSTREAM

For the next labs we might publish the advanced tests specification later... this should not cause any merge conflicts

Sometimes we also need to do fixes in the code base...this might cause merge conflicts in the worst case (we try to avoid this)

To get the latest code changes, you have 2 options:

- 1) Backup your implementation, delete your forked repository, and fork the project again (you will lose commit history).
- 2) Use the approach from the next slide

(OPTIONAL) PULL CHANGES FROM UPSTREAM

Follow the instructions from here to sync your repo:

<https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/syncing-a-fork>:

```
git remote add upstream https://tg.dm.informatik.tu-  
darmstadt.de/sdms_ws2021_students/labx.git  
git fetch upstream  
git checkout master  
git merge upstream/master
```

PUSH YOUR CHANGES TO THE SERVER

You can use your favorite Git client. We use the git command line.
Make sure to stage and commit your local changes and the push them to the server, e.g.:

```
git add .  
git commit -m "My commit contains the following changes"  
git push
```

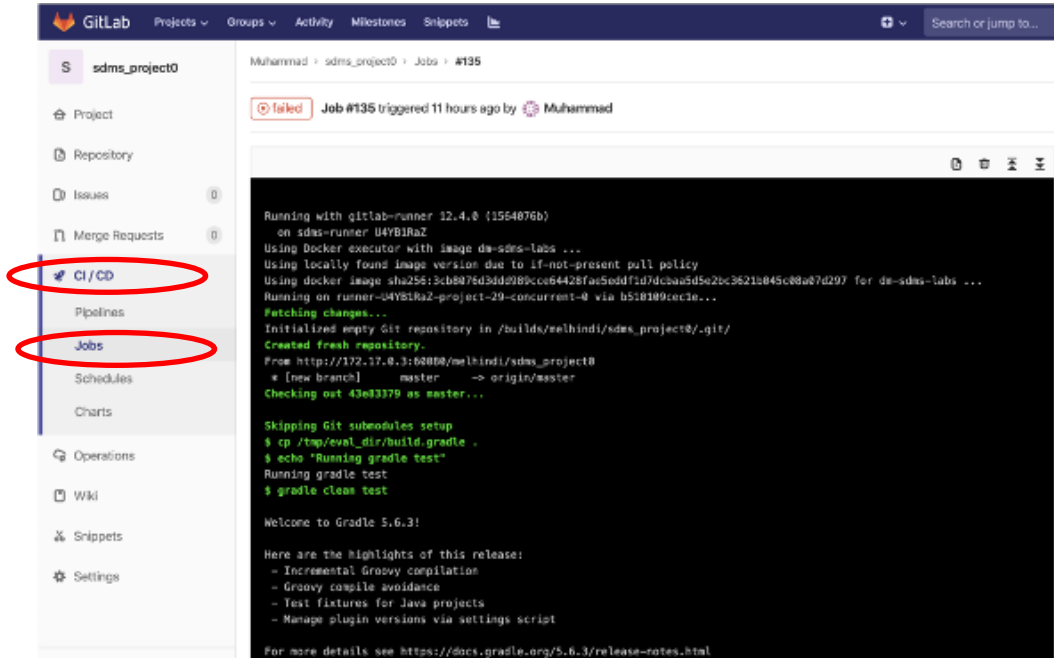
Once you push your changes to the master branch on the server the evaluation will start.

If you create a different branch, e.g., dev, and push it to the server, the evaluation will not be triggered for commits to that branch.

Only commits/pushes to the master branch are considered a submission.

PUSH YOUR CHANGES TO THE SERVER

You can see the result of your evaluation in the CI/CD section:



The screenshot displays the GitLab web interface for a project named 'sdms_project0'. The left sidebar contains navigation links: Project, Repository, Issues (0), Merge Requests (0), CI/CD (highlighted with a red circle), Pipelines, Jobs (highlighted with a red circle), Schedules, Charts, Operations, Wiki, Snippets, and Settings. The main content area shows the 'Jobs' section for 'sdms_project0', displaying 'Job #135 triggered 11 hours ago by Muhammad'. The job status is 'failed'. Below the job header is a terminal window showing the execution logs of the CI job. The logs indicate that the job is running on a GitLab Runner, using a Docker executor, and executing a Gradle build script. The build process includes initializing a Git repository, creating a new branch, and running a Gradle clean test. The logs also mention the Gradle version (5.6.3) and provide a link to the release notes.

```
Running with gitlab-runner 12.4.0 (1564076b)
on sdms-runner U4YB1RaZ
Using Docker executor with image da-sdms-labs ...
Using locally found image version due to if-not-present pull policy
Using docker image sha256:3cb8076d3dd2899ccc64428fae5edd7d7dcbaw5d5e2bc3621b045c88a7d207 for da-sdms-labs ...
Running on runner-U4YB1RaZ-project-29-concurrent-0 via b518109ce1e...
Patching changes...
Initialized empty Git repository in /builds/mehindi/sdms_project0/.git/
Created fresh repository.
From http://172.17.0.3:60808/mehindi/sdms_project0
 * [new branch]      master    -> origin/master
Checking out 43e03379 as master...

Skipping Git submodules setup
$ cp /tmp/eval_dir/build.gradle .
$ echo "Running gradle test"
Running gradle test
$ gradle clean test

Welcome to Gradle 5.6.3!

Here are the highlights of this release:
- Incremental Groovy compilation
- Groovy compile avoidance
- Test fixtures for Java projects
- Manage plugin versions via settings script

For more details see https://docs.gradle.org/5.6.3/release-notes.html
```

RECOMMENDED READINGS AS REFRESHER

Git:

<https://docs.gitlab.com/ee/gitlab-basics/start-using-git.html>

<https://rogerdudler.github.io/git-guide/>

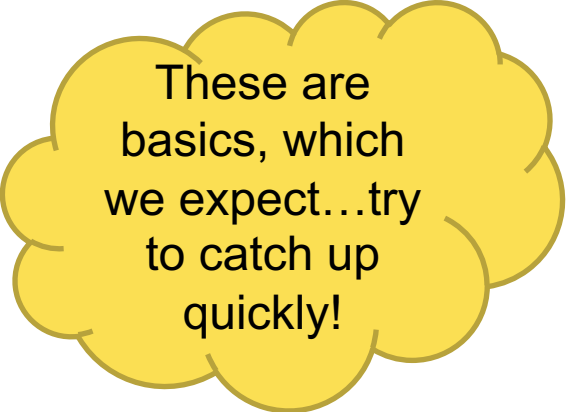
<https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf>

Java:

<https://www.codecademy.com/learn/learn-java>

<https://codegym.cc/>

<https://www.w3schools.com/java/>



These are basics, which we expect...try to catch up quickly!

AGENDA

- Lab 0 content
- Submission Process & GitLab Hints
- **Recap: LRU-K and B-trees**

EVICTON STRATEGIES

If buffer is full when a new page is requested via a PIN operation, **then a victim frame needs to be selected (called page eviction)**

Goal is to **evict pages that are not likely to be used in near future**

Typical eviction strategies

- **LRU:** evicts the page whose access the furthest in the past
- **LRU-k:** evicts the page whose K-th most recent access is furthest in the past. For example, LRU-1 is simply LRU
- **Clock / Second chance:** Low-overhead approximation of LRU

LEAST-RECENTLY-USED (LRU)

Main idea: keep least-recently-used pages in buffer

Implementation alternatives:

#1: Keep a timestamp of last access per page

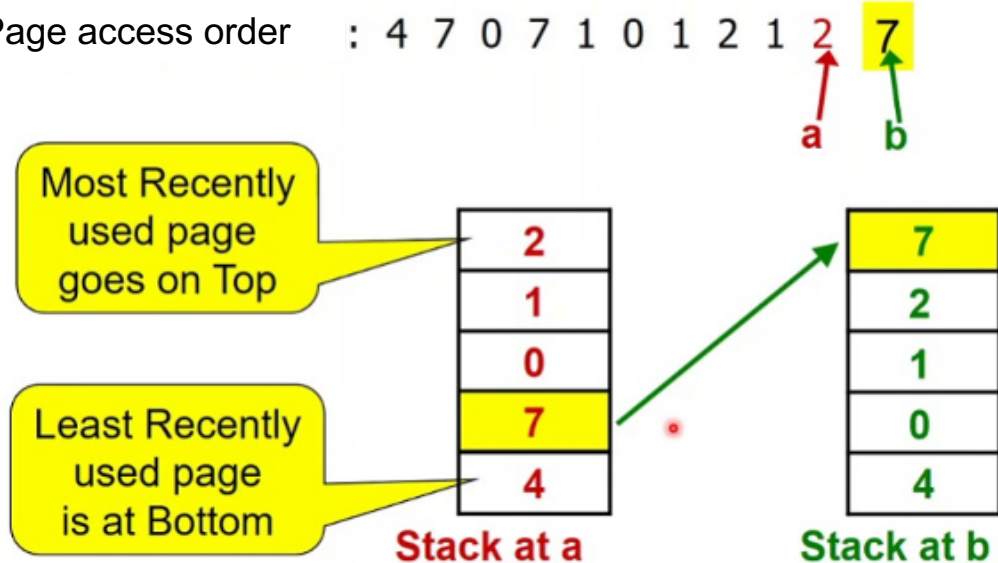
#2: Keep lists (e.g., stack) of page Ids ordered by access time

Both variants can cause **high overhead**

Implementation variant #2:

Keep ordered list of page numbers

Page access order : 4 7 0 7 1 0 1 2 1 2 7



LRU-K

Main idea: determine which page to evict based on the last K accesses.
Page to evict = page in buffer with largest *backward K -distance*.

Benefit of LRU-K vs LRU: can differentiate between pages frequently accessed (e.g., the root-node page in an index) or infrequently (like data pages).

Example with $K=2$:

Accesses:	1	3	0	2	3	2	3	0	1	2
Times:	0	1	2	3	4	5	6	7	8	9
K-distance:	x	x	x	x	3	2	2	5	8	4

B-tree Family

B-tree is a **tree-based index** for individual attributes (i.e., it is a one-dimensional index) based on the idea of a binary tree

B-tree is **most common index structure** in DBMSs

Many **different variants**

- Basic B-tree
- B+ tree
- B^{link} tree
- ...

B-tree: Search

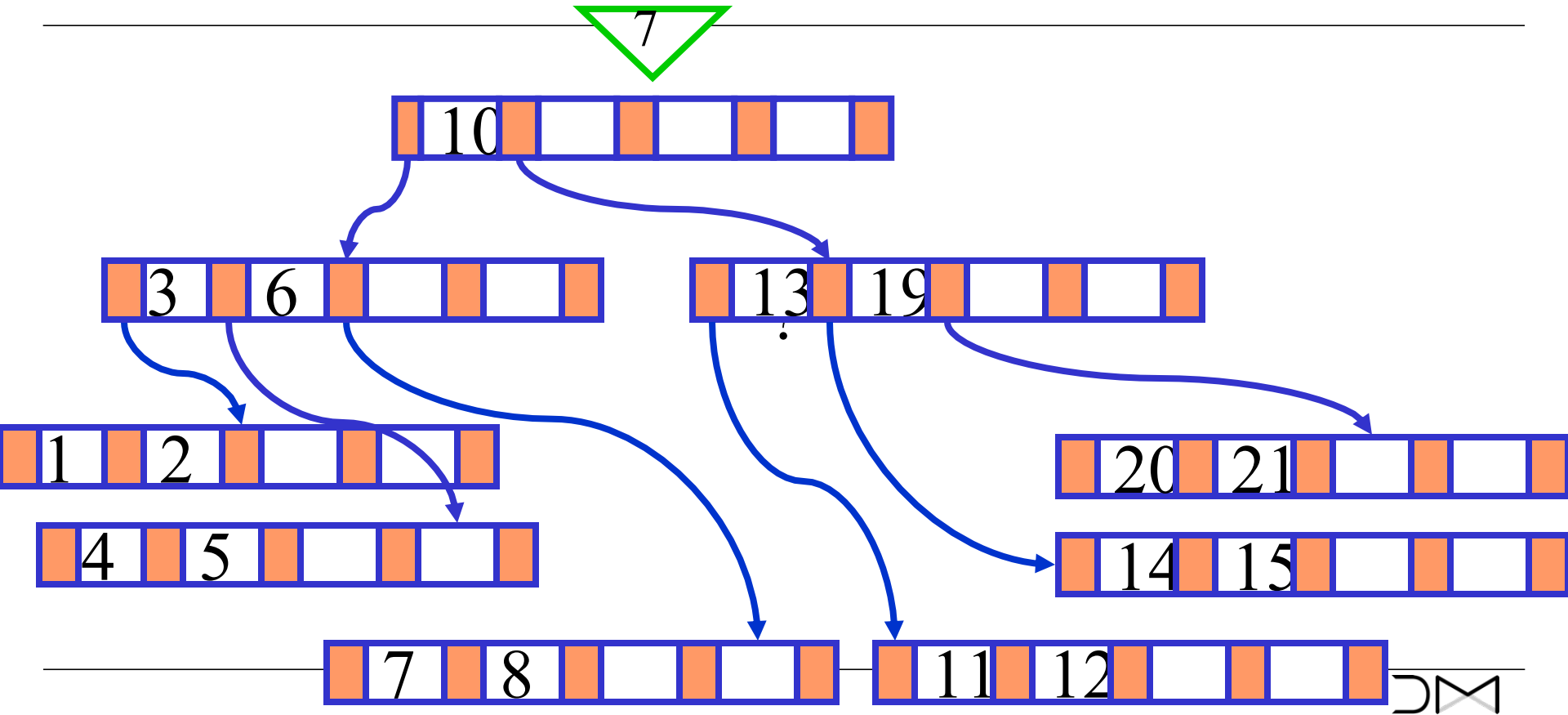
Key lookup: e.g. $k=6$

- Search tree from root to leaves
- Sequential or binary search within each node / leaf
- If searched key K_i is found, then follow pointer P_i to data

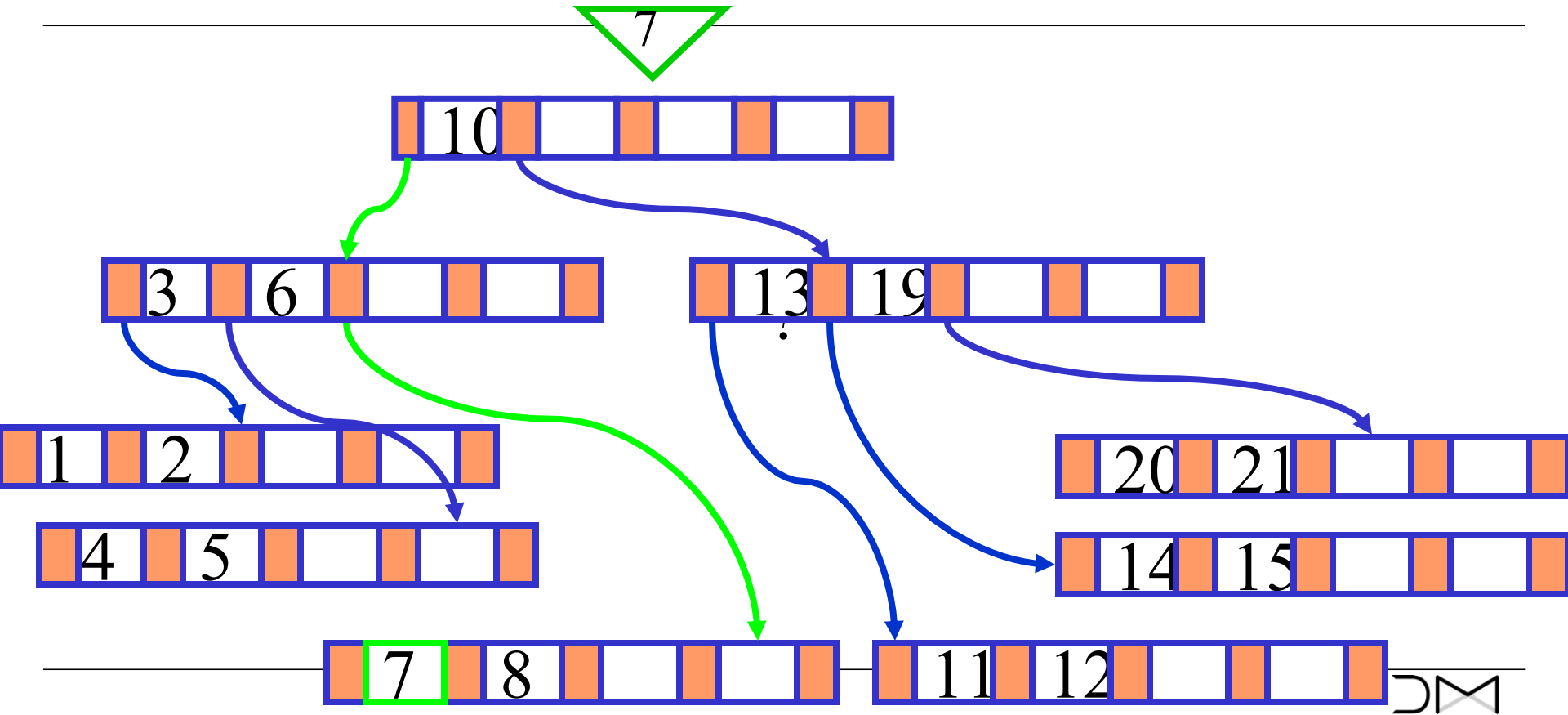
Range search: e.g. $5 \leq k \leq 10$

- Search lower bound (e.g., 5)
- Traverse tree using an in-order search until upper bound (e.g., 10)

B-tree: Key Lookup



B-tree: Key Lookup



B-tree: Insertion

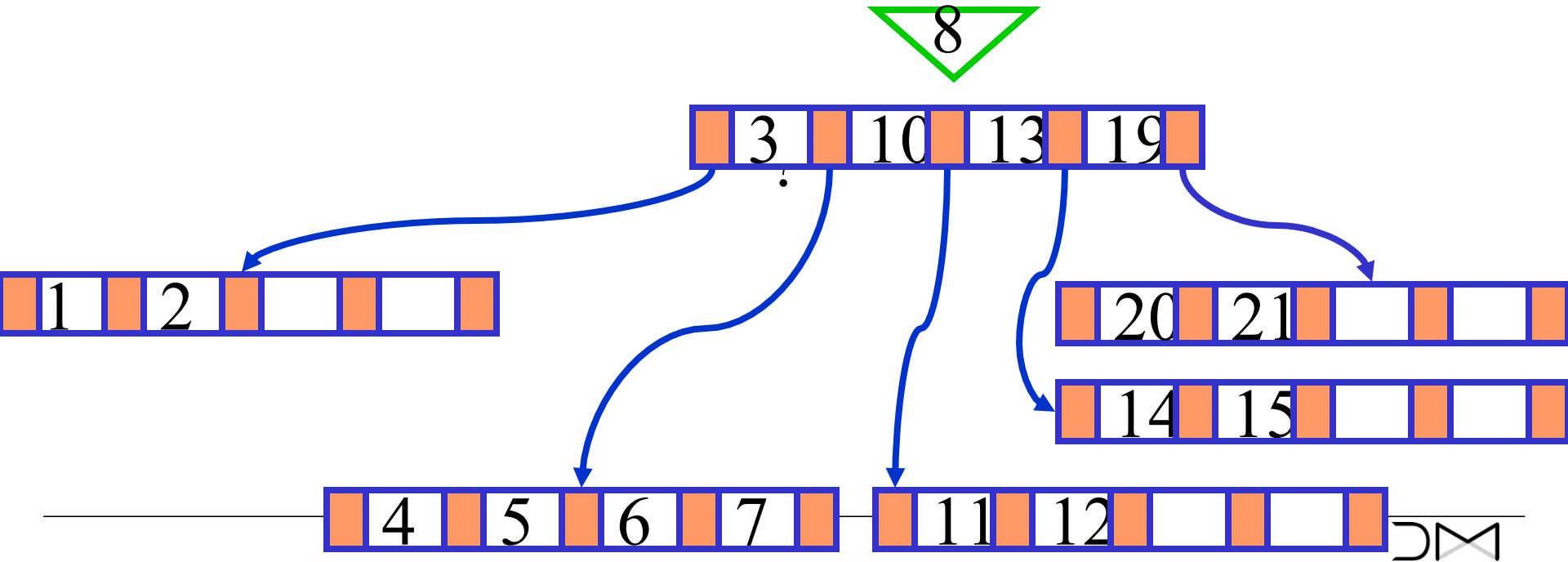
Search position in leaf node to **insert new key / RID into leaf node in “sort order”**

If fill grade of leaf $> 2M$ (overflow), then split leaf and pull middle entry to the parent node (inner node)

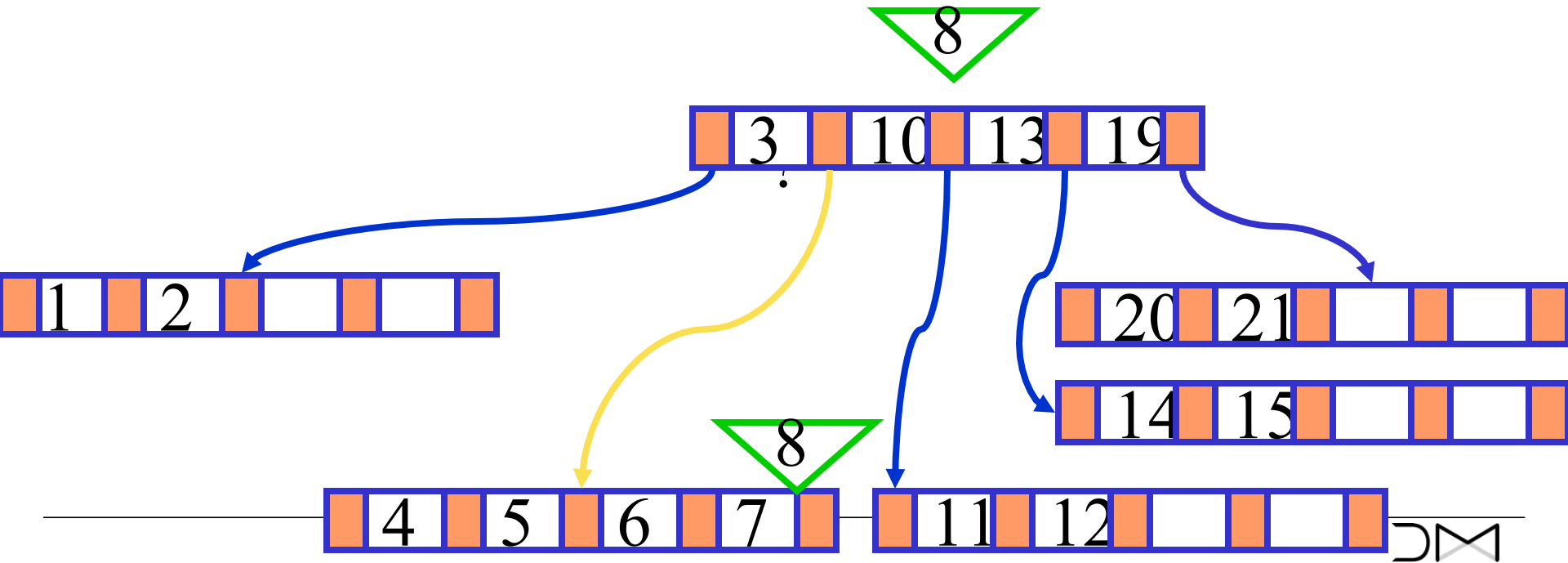
If fill grade of parent $> 2M$, then split parent node and pull middle entry to parent node -> cnt. recursively

If split node is root node, then add new root node

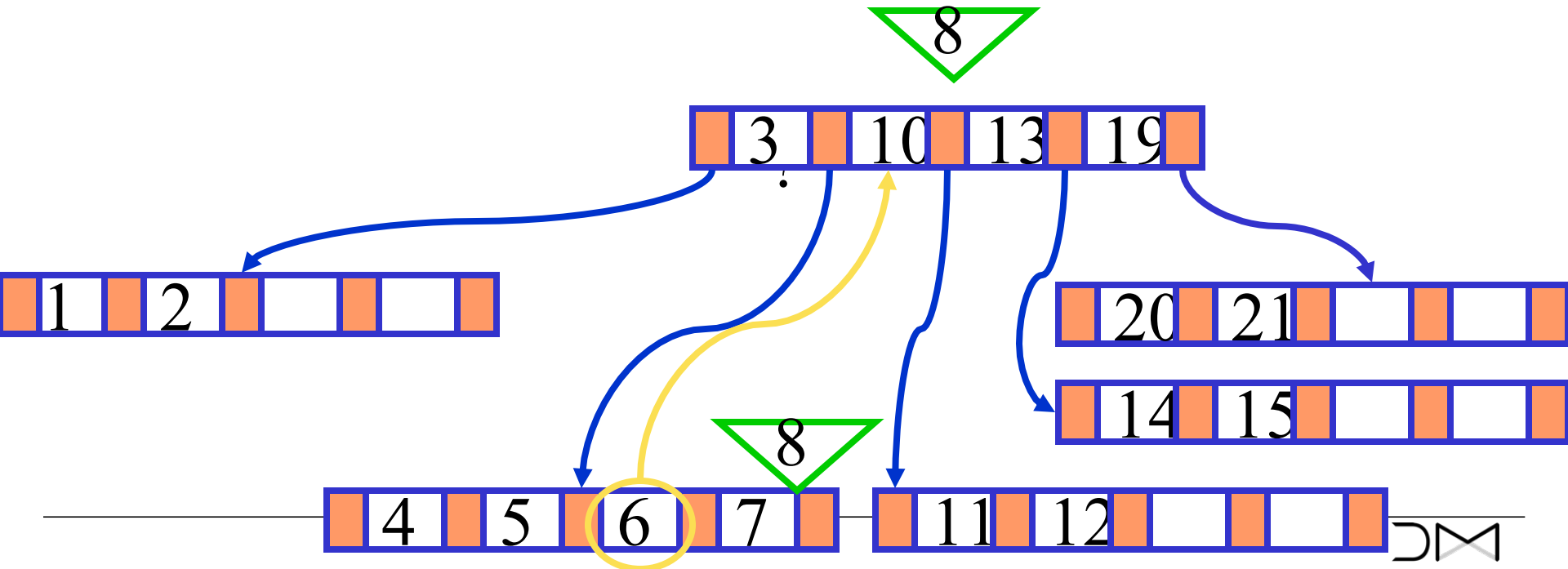
B-tree: Insertion



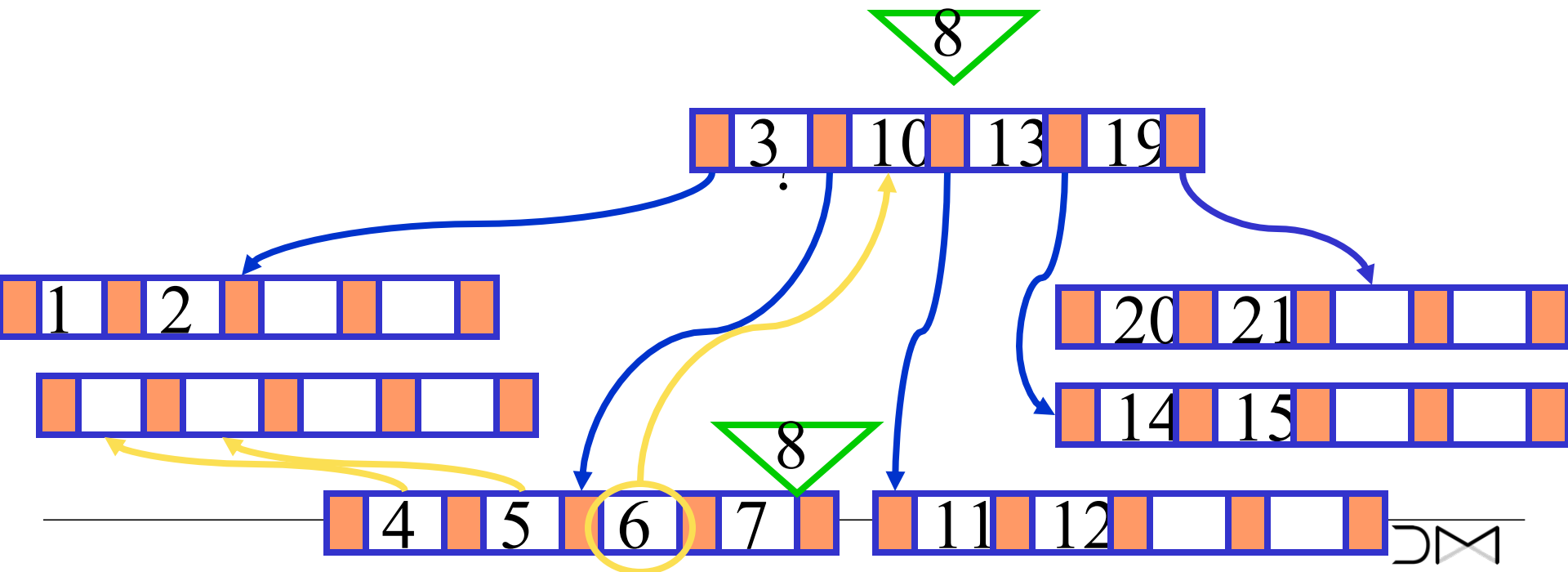
B-tree: Insertion



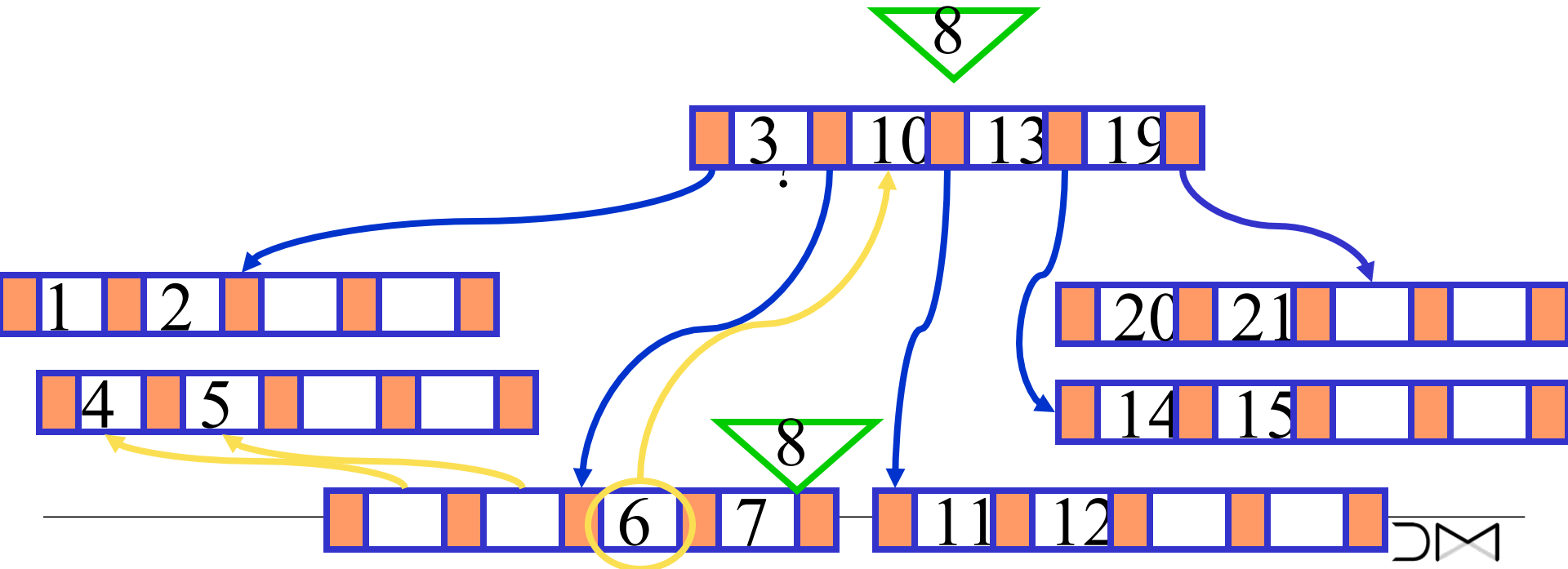
B-tree: Insertion



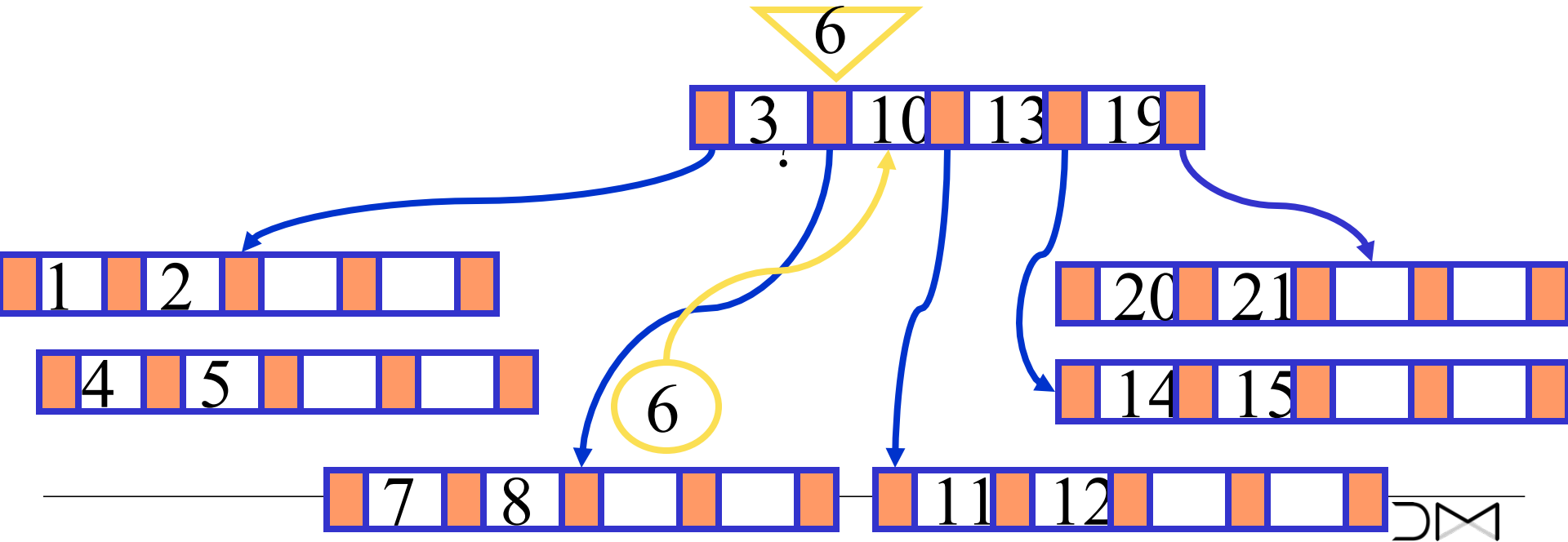
B-tree: Insertion



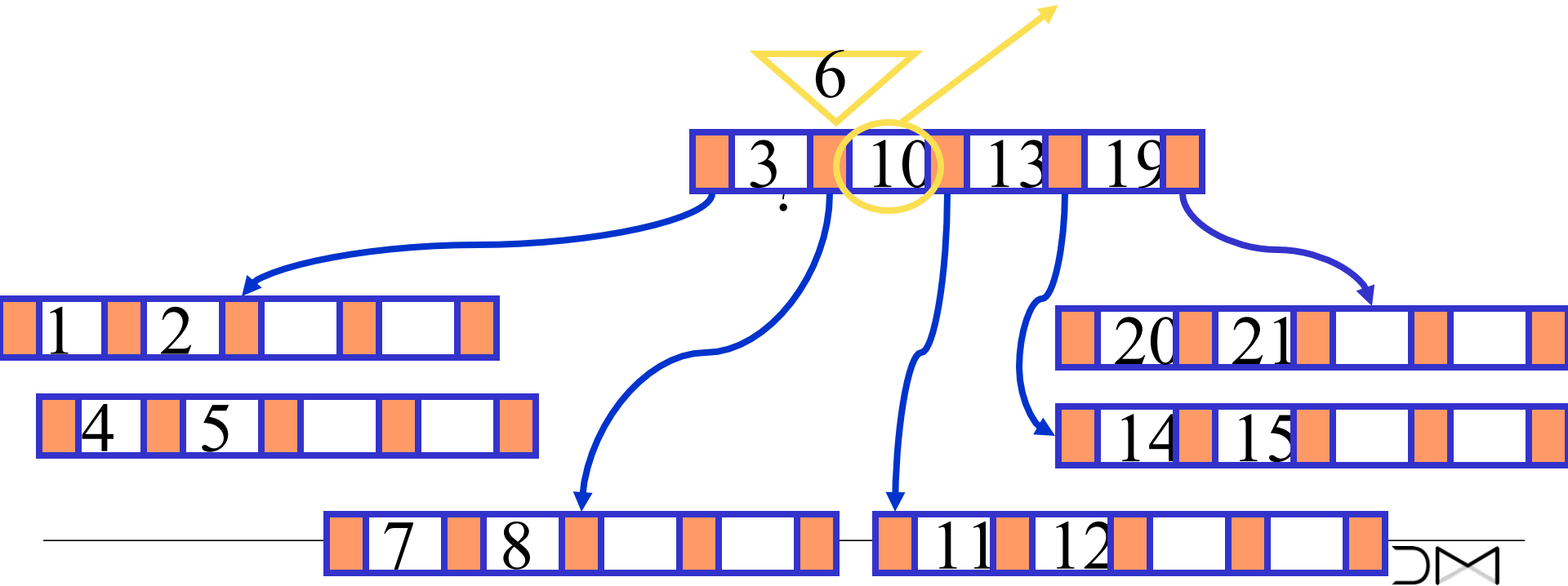
B-tree: Insertion



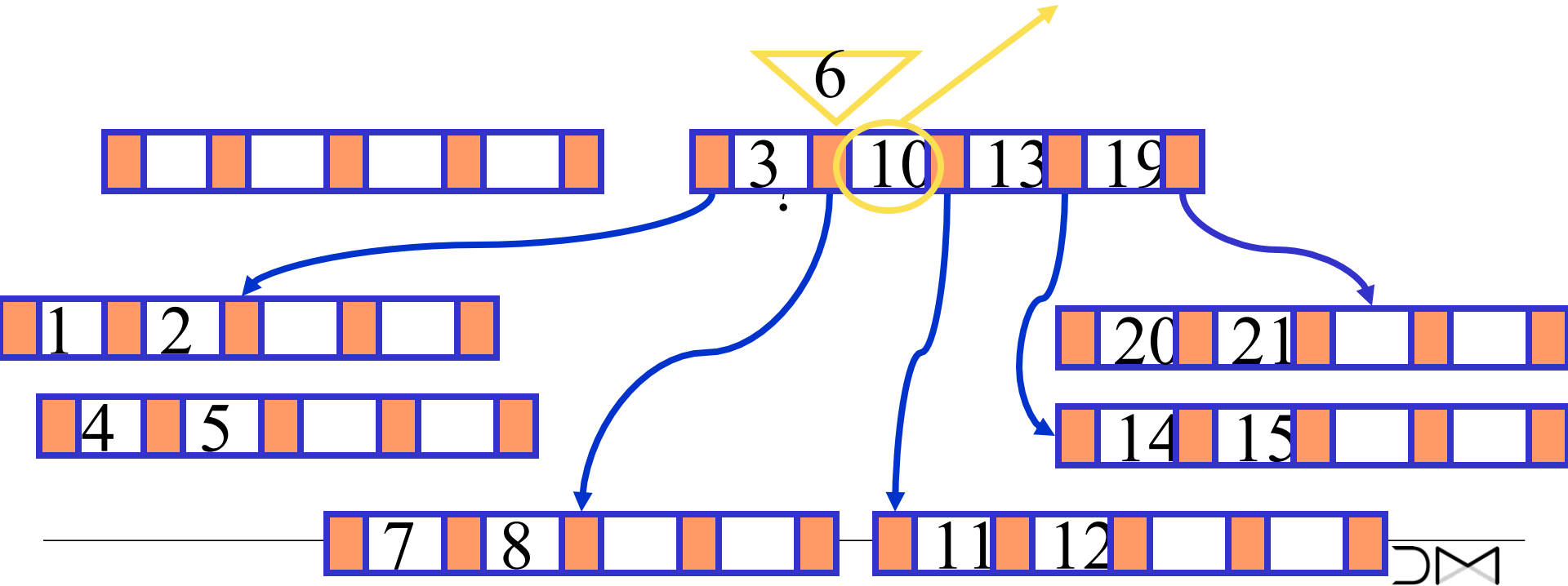
B-tree: Insertion



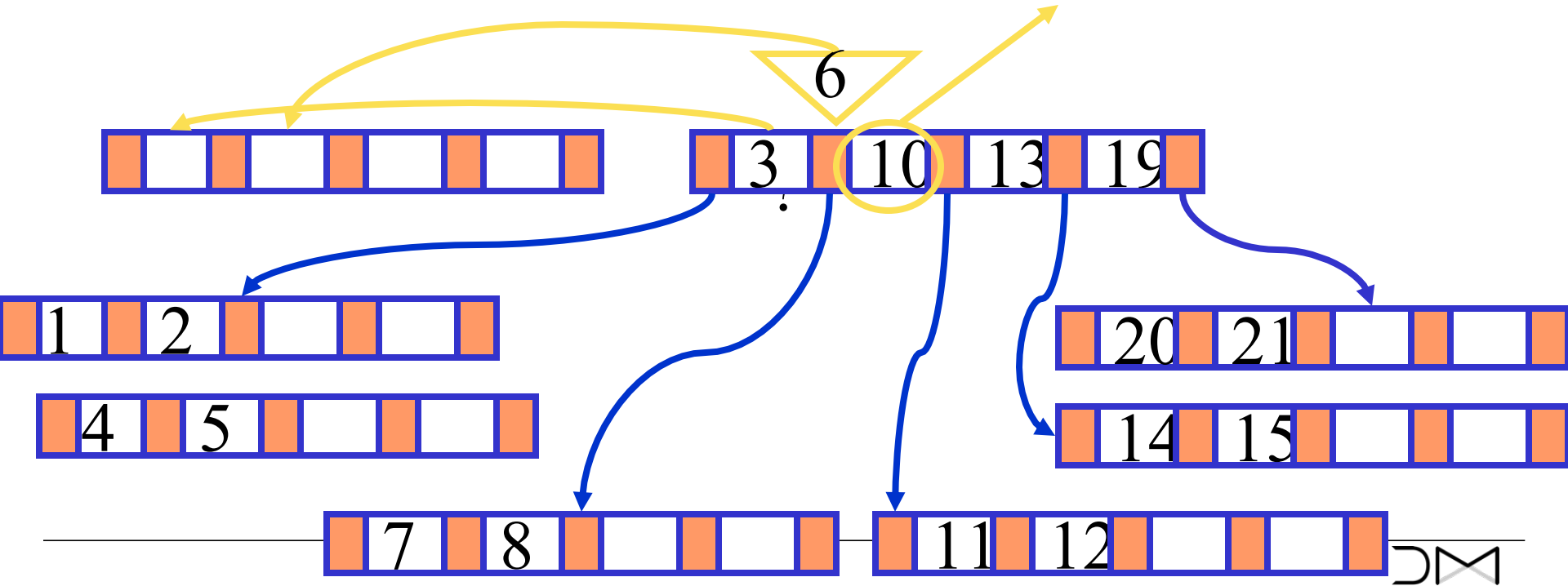
B-tree: Insertion



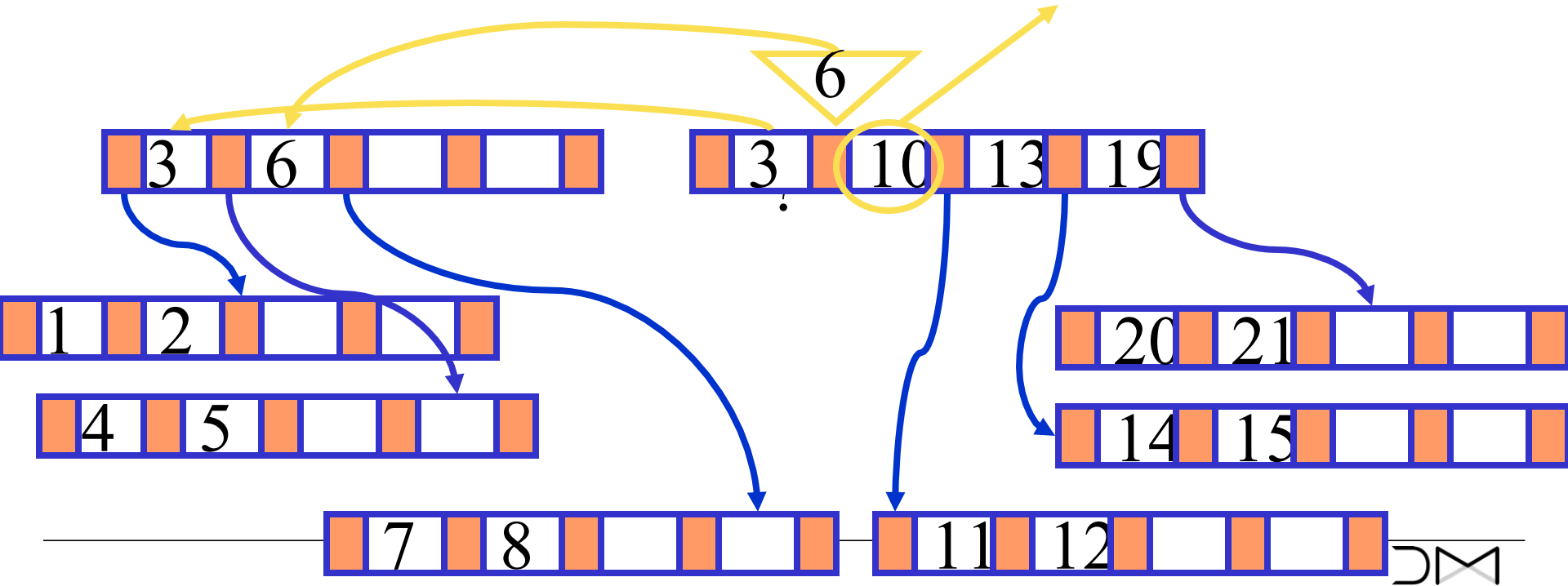
B-tree: Insertion



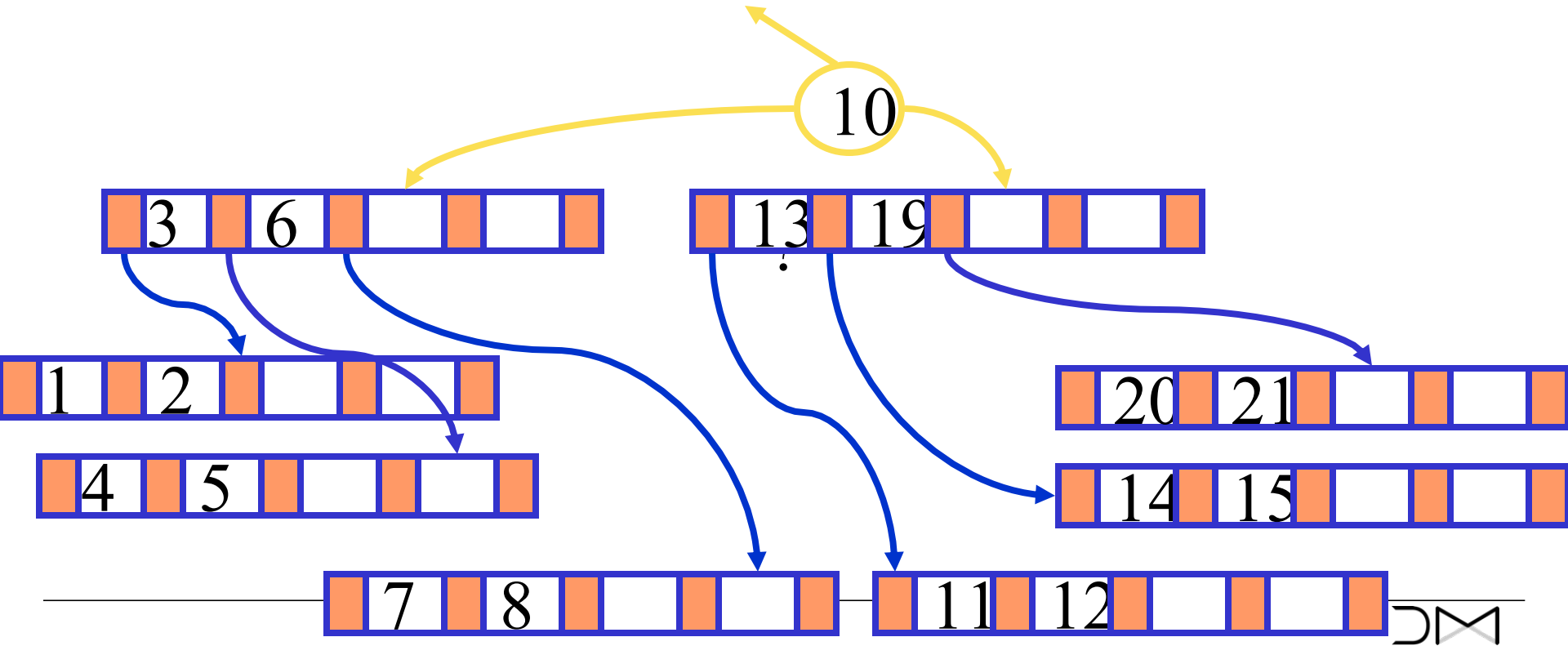
B-tree: Insertion



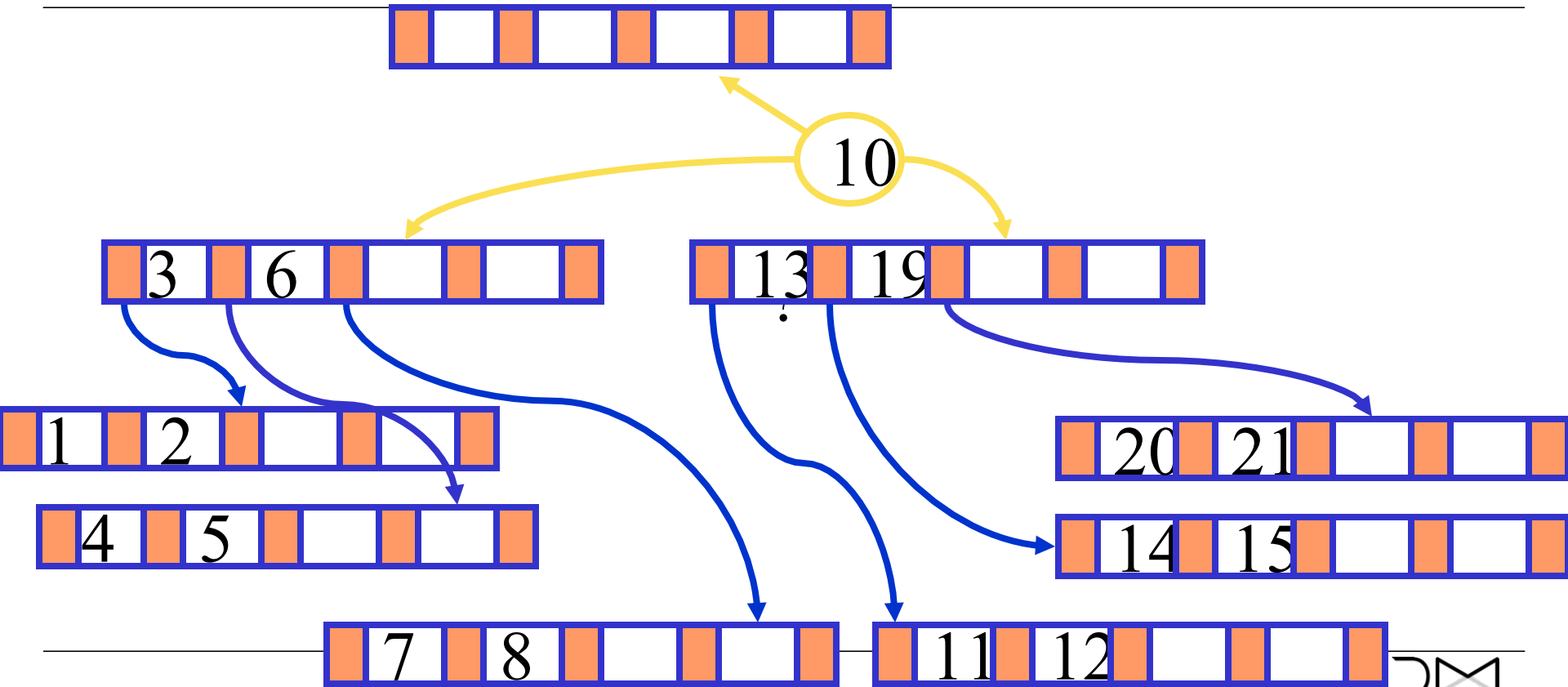
B-tree: Insertion



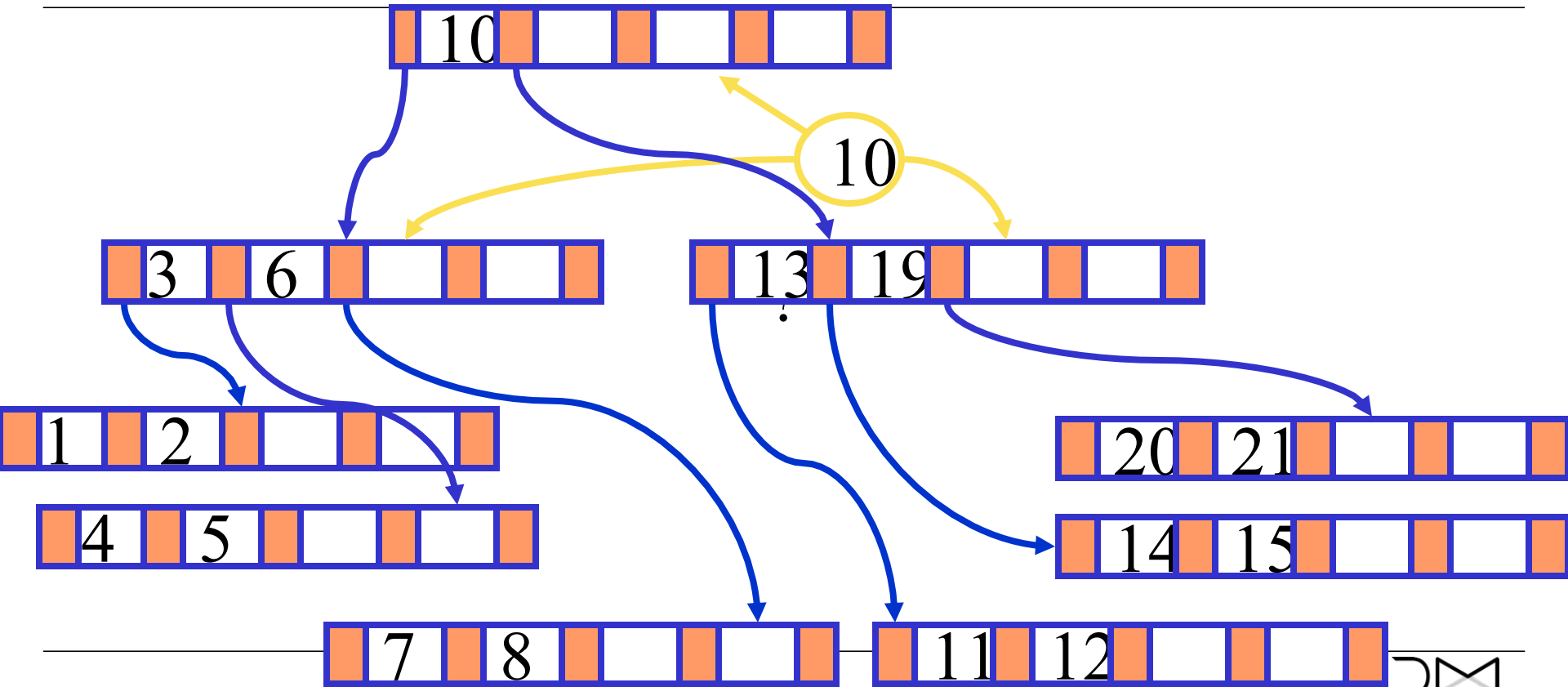
B-tree: Insertion



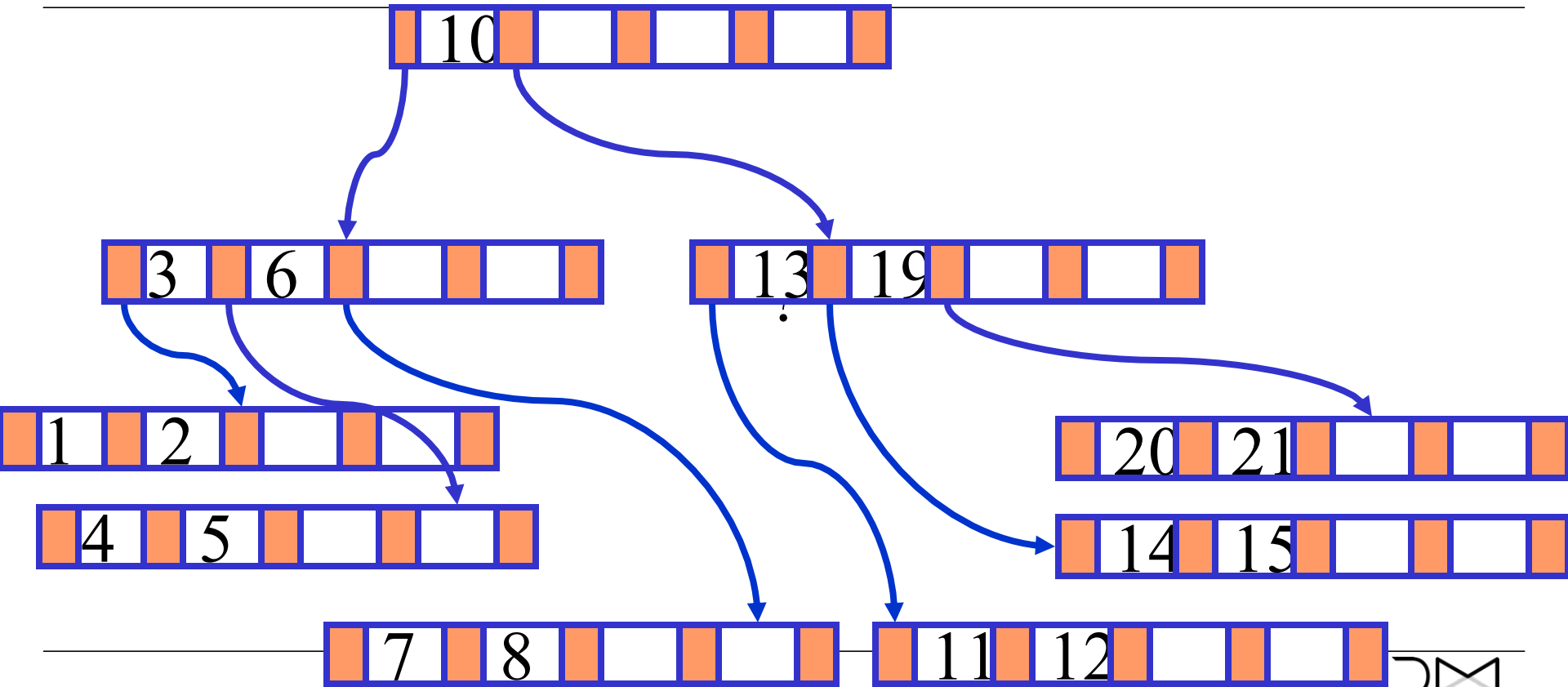
B-tree: Insertion



B-tree: Insertion



B-tree: Insertion



B-tree: Deletion

Search key to delete

If key to delete is in inner node:

- delete successor (next larger key) in leaf node and
- replace key in inner node with key in leaf

Otherwise, delete key from leaf node directly

If fill grade of leaf $< M$, then reorganize leaves (by moving or merging keys – **prefer moving over merging**)

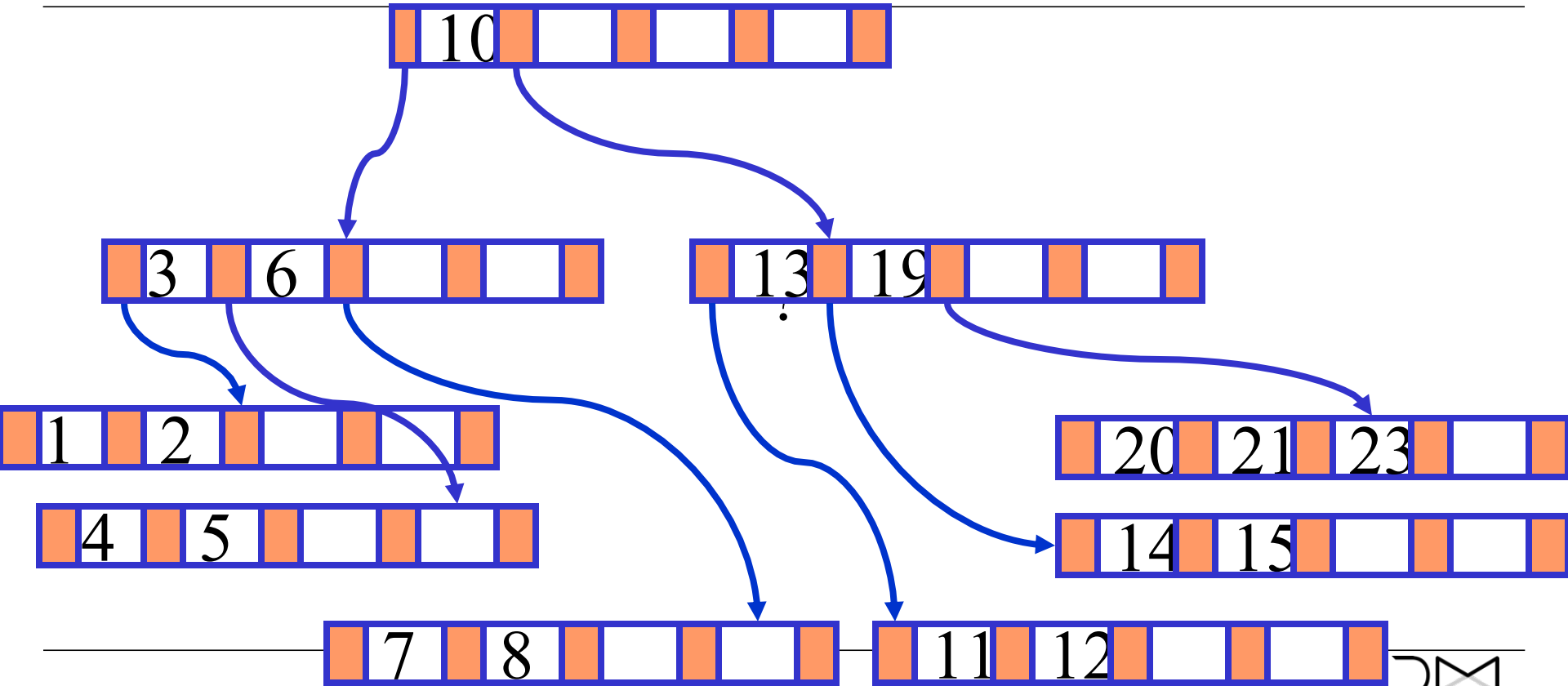
B-tree: Reorg due to Deletions

If possible, then move keys from left or right siblings such that all nodes have fill grade $\geq M$ (rotation)

Otherwise merge with left or right sibling

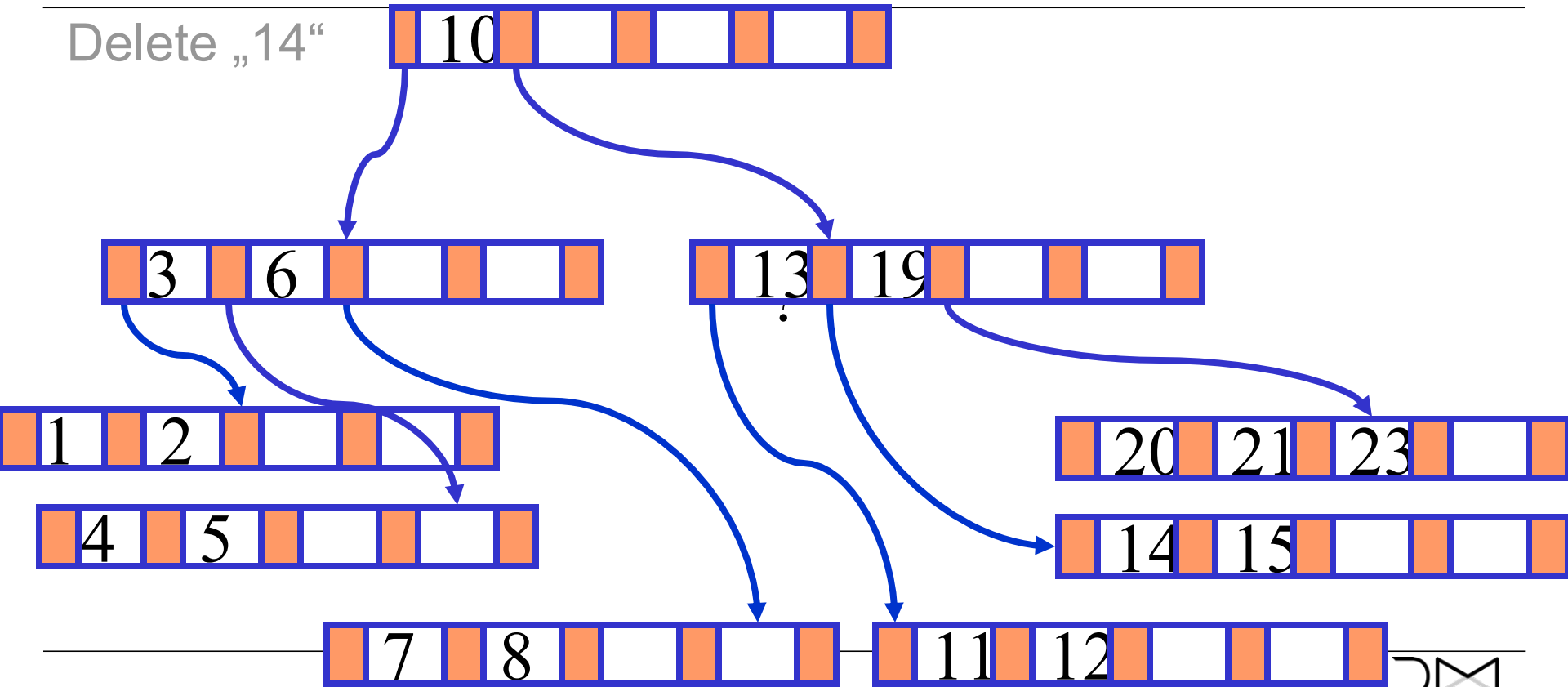
- Key from parent needs to be pulled into merged node
- Parent might need to be reorganized since fill grade of parent $< M$

B-tree: Deletion



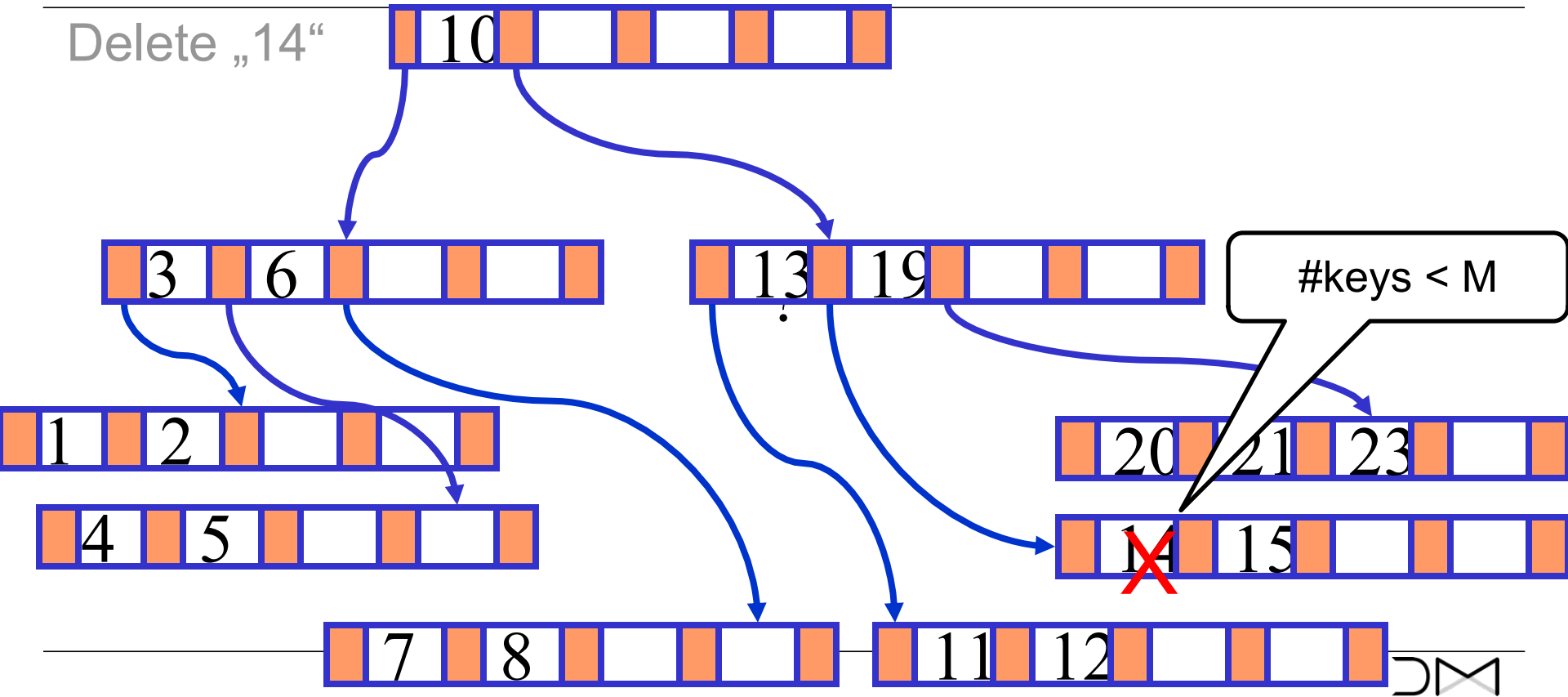
B-tree: Deletion

Delete „14“



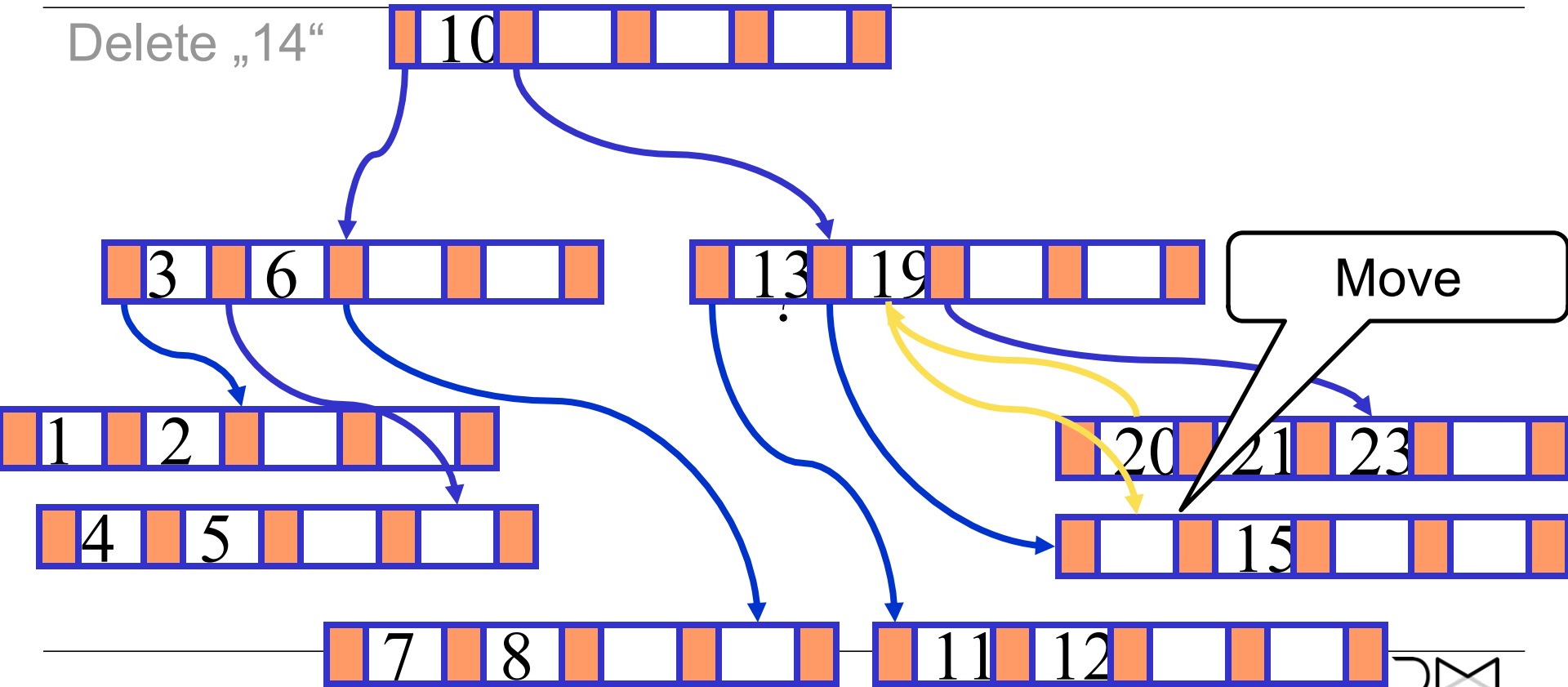
B-tree: Deletion

Delete „14“



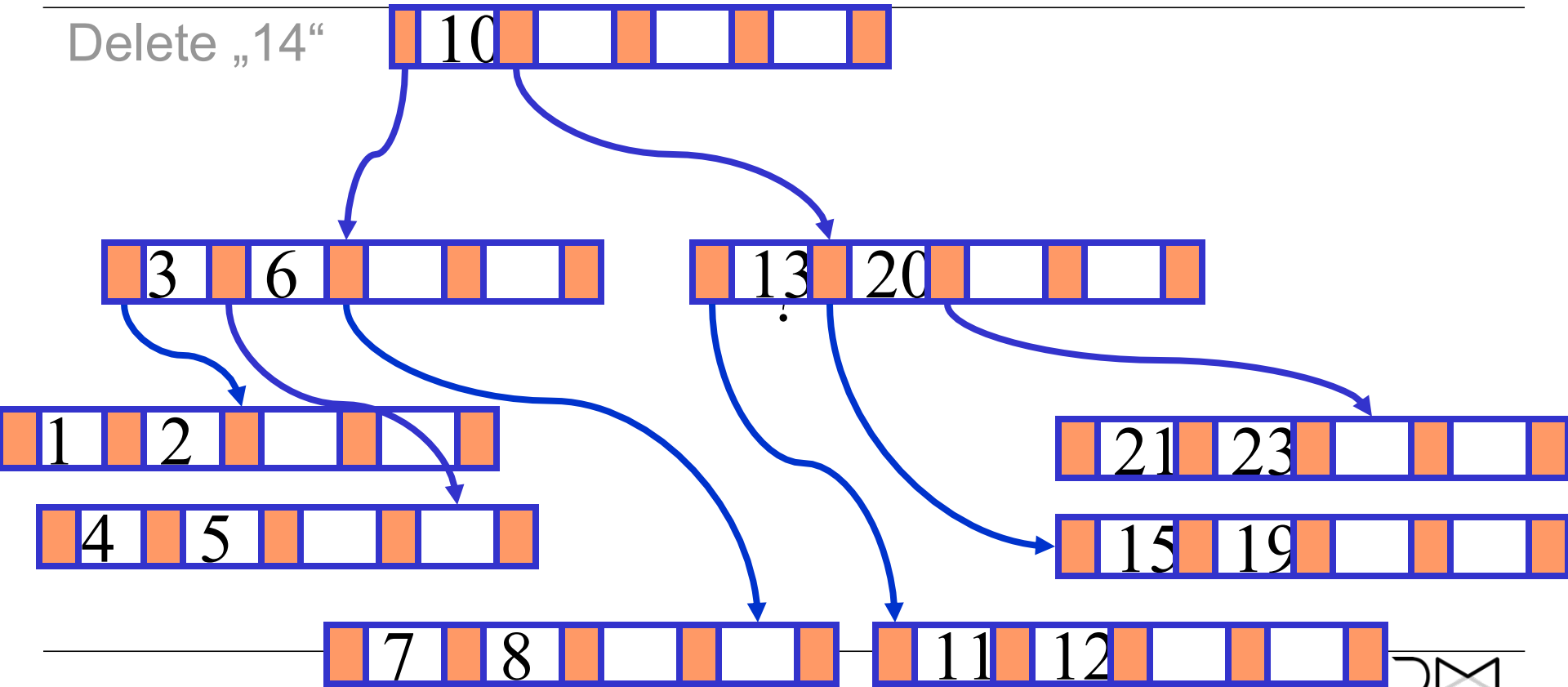
B-tree: Deletion

Delete „14“



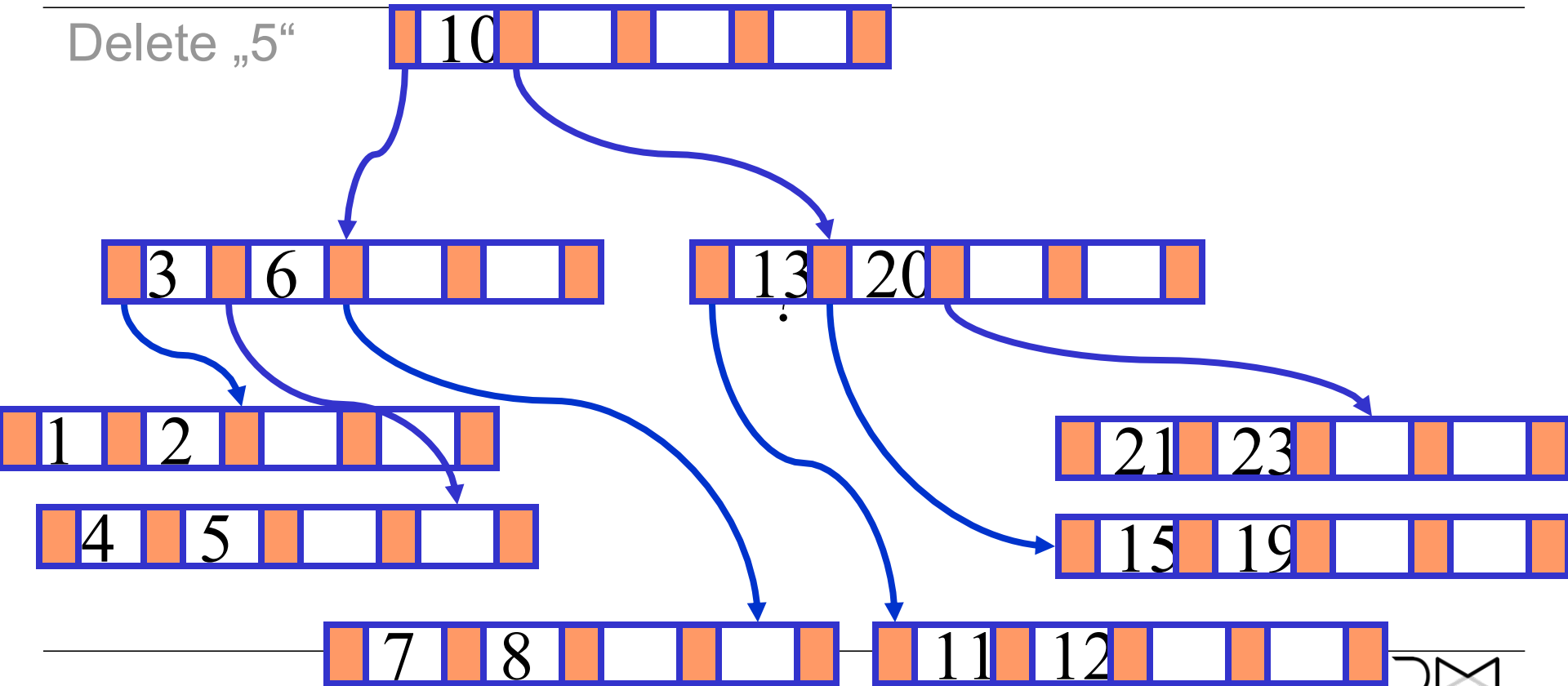
B-tree: Deletion

Delete „14“



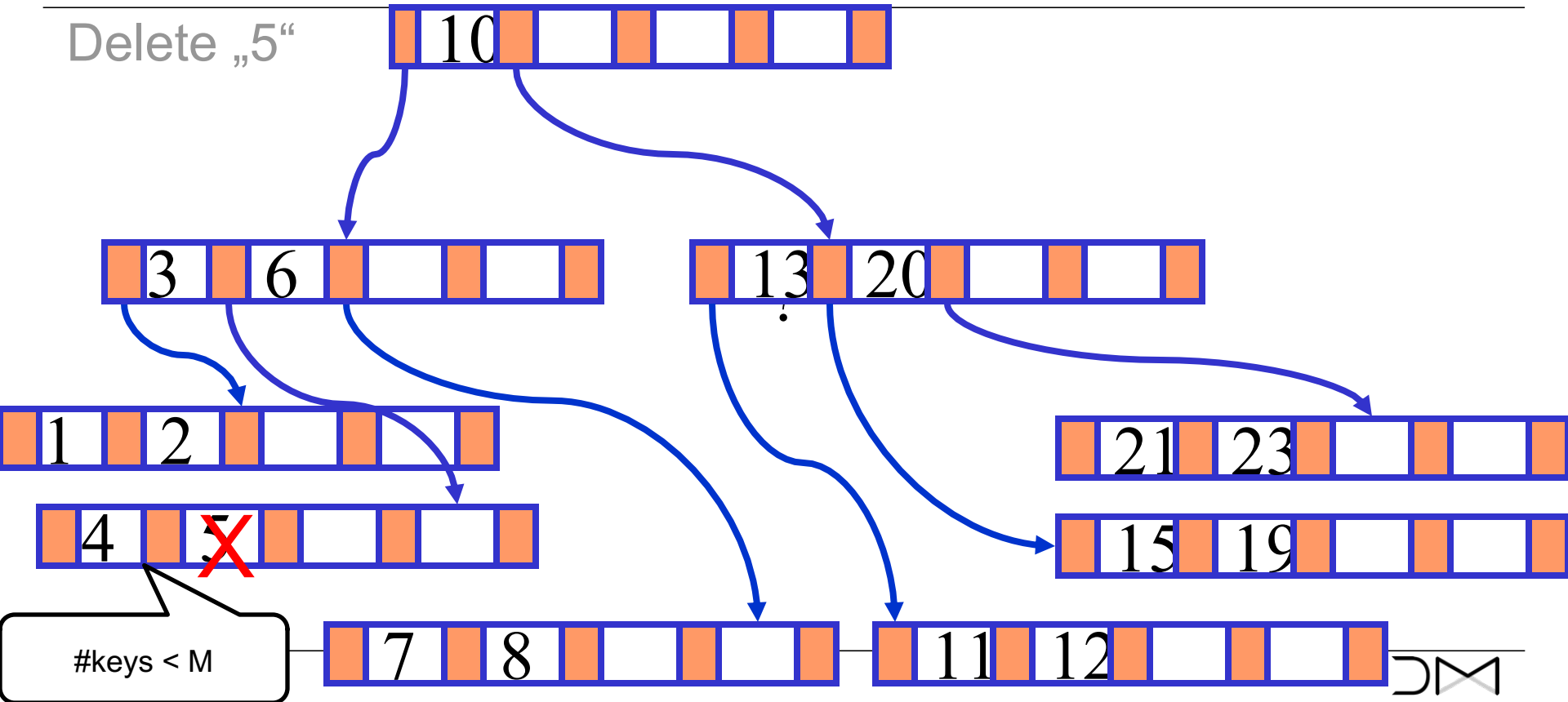
B-tree: Deletion

Delete „5“



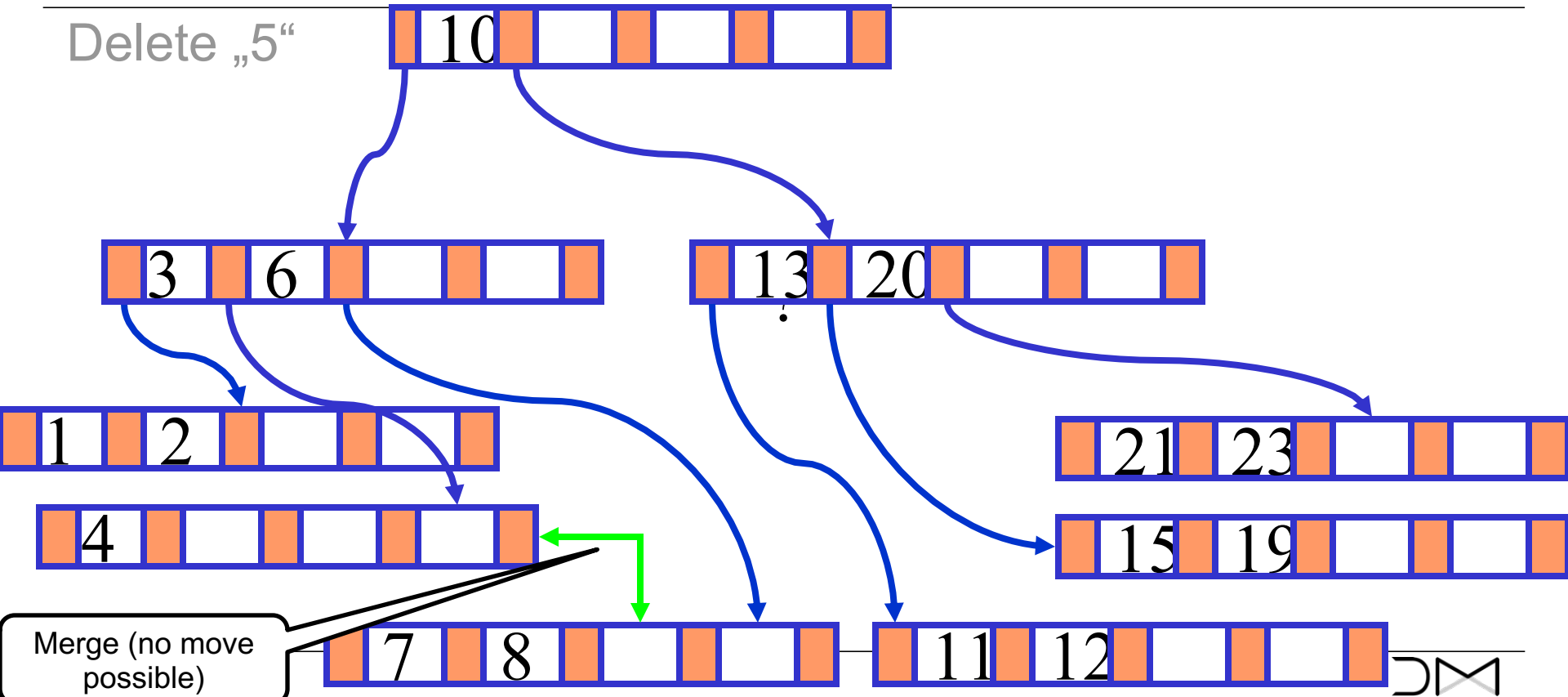
B-tree: Deletion

Delete „5“

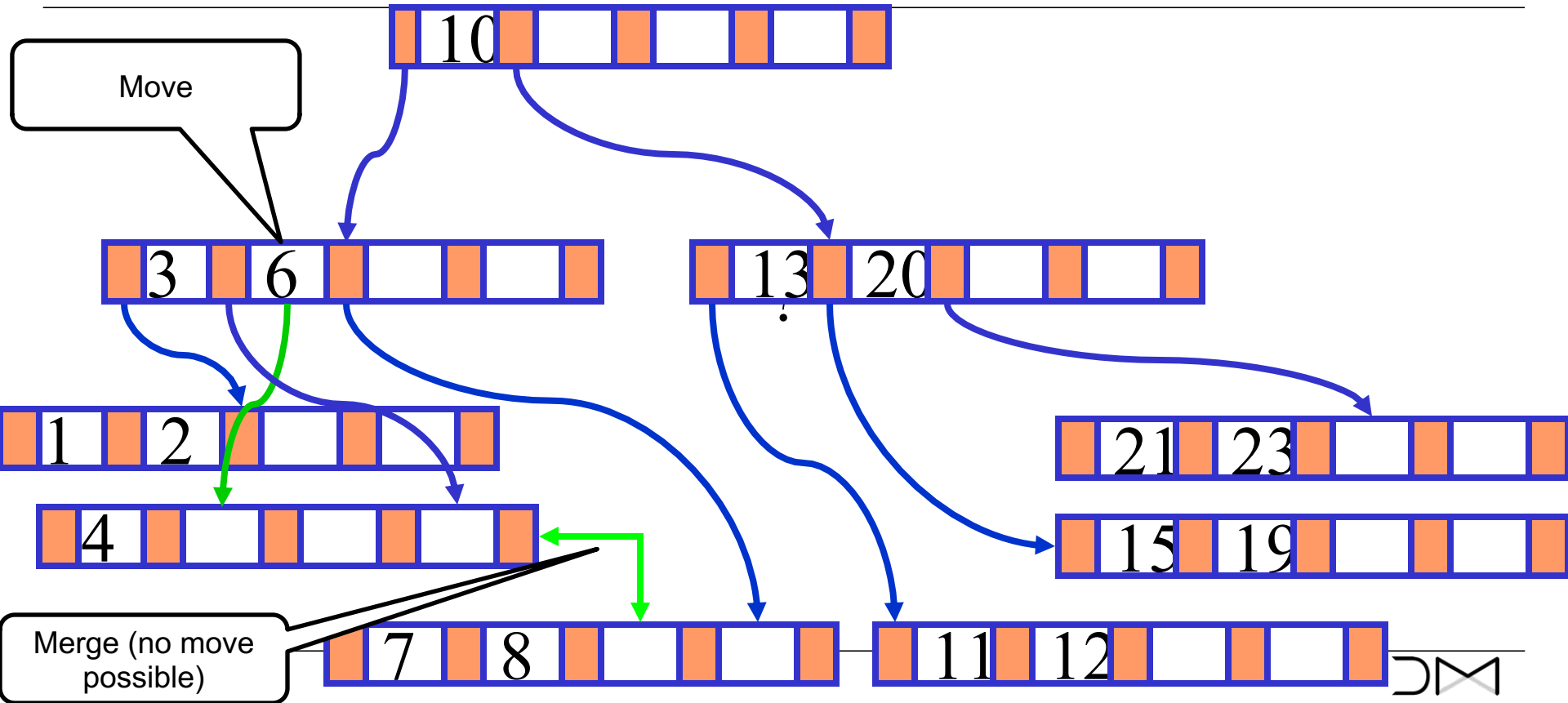


B-tree: Deletion

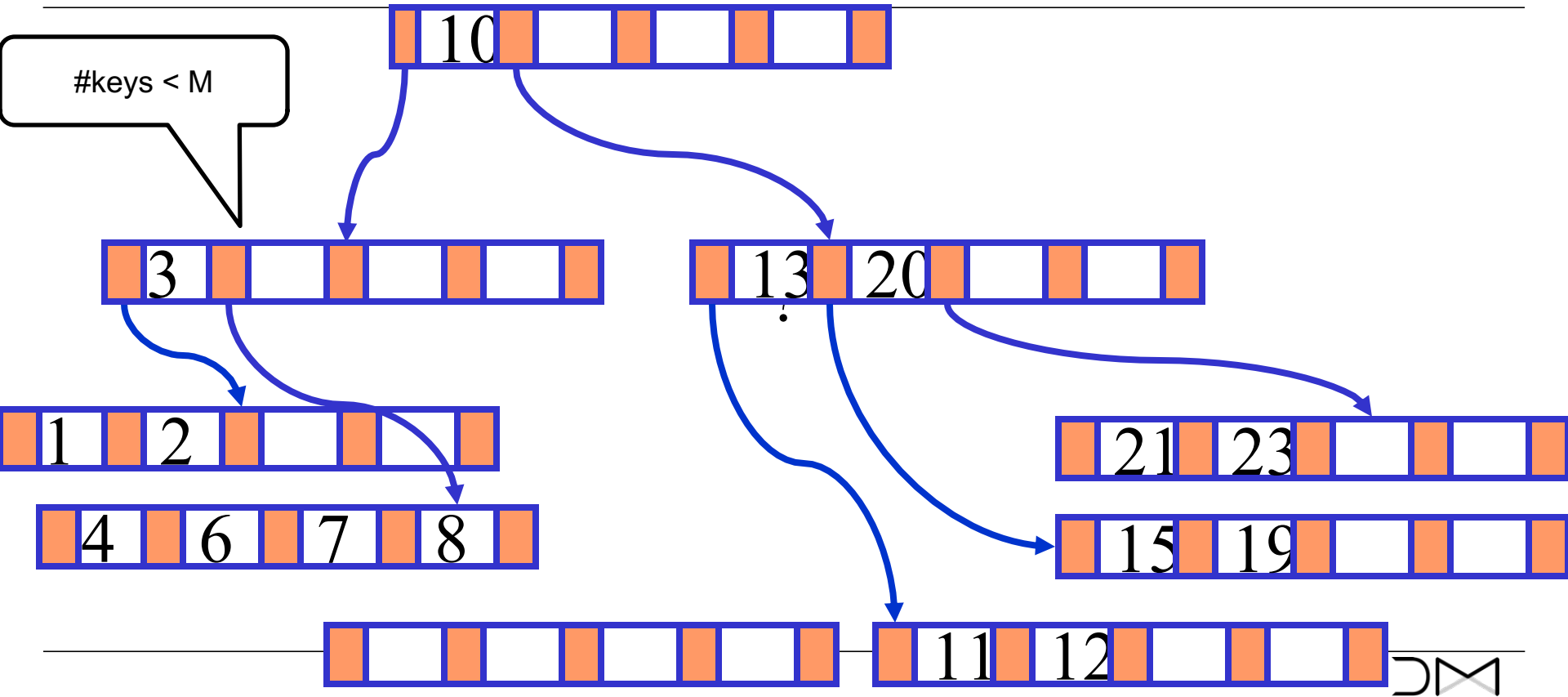
Delete „5“



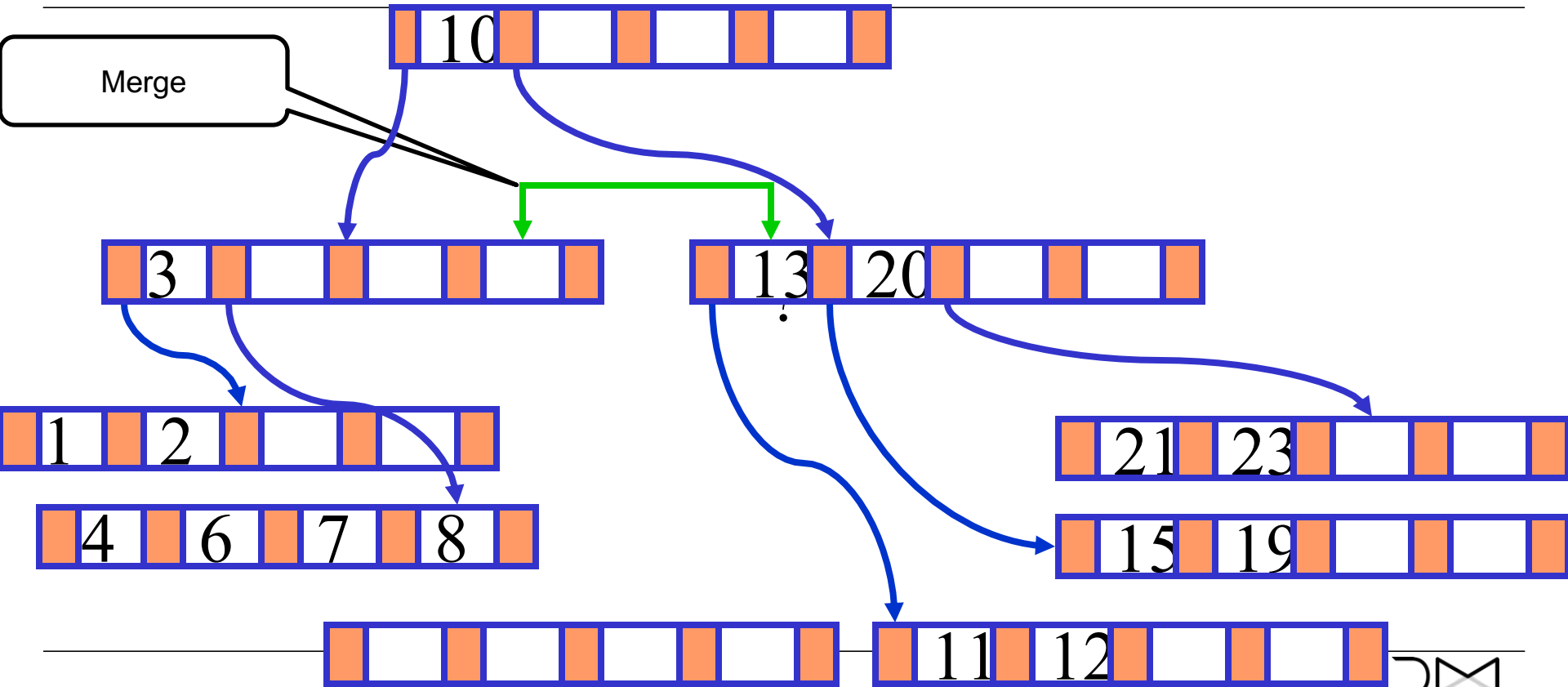
B-tree: Deletion



B-tree: Deletion



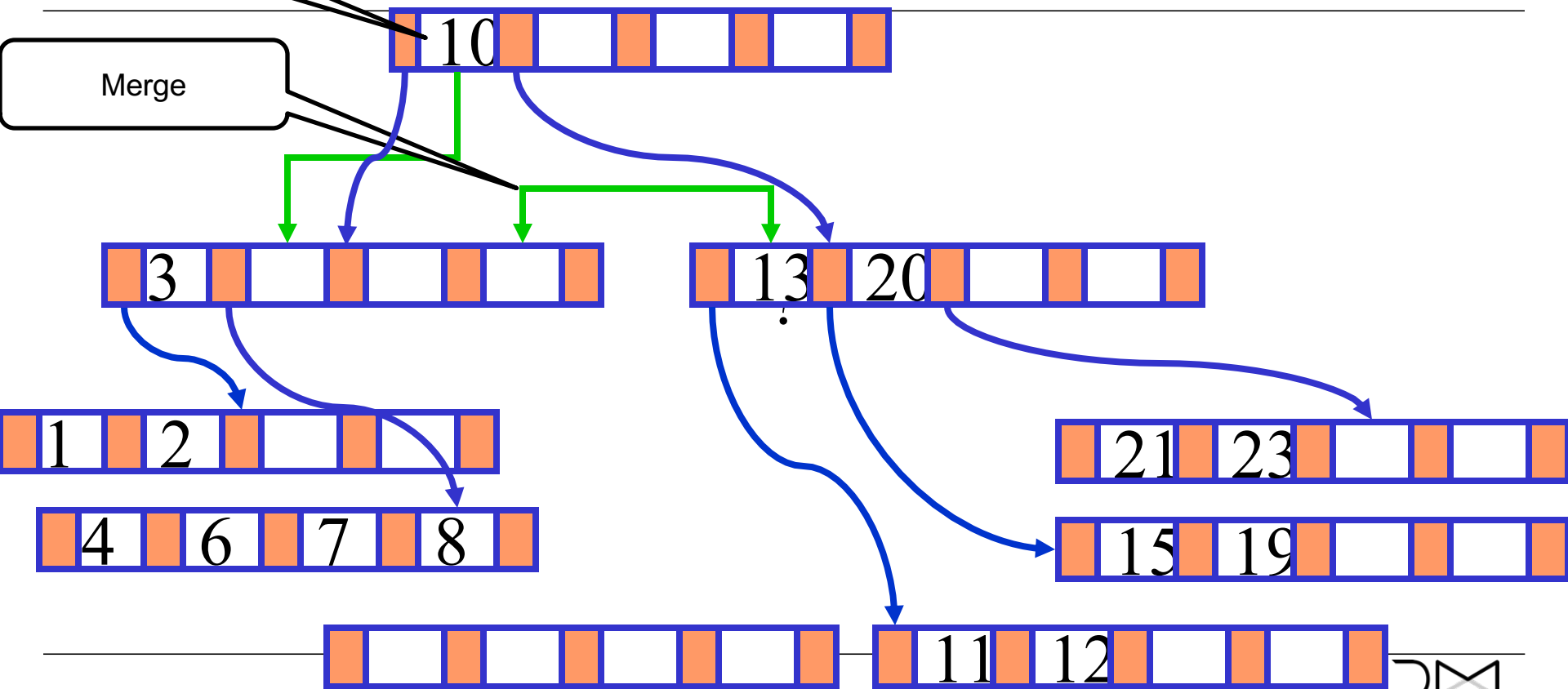
B-tree: Deletion



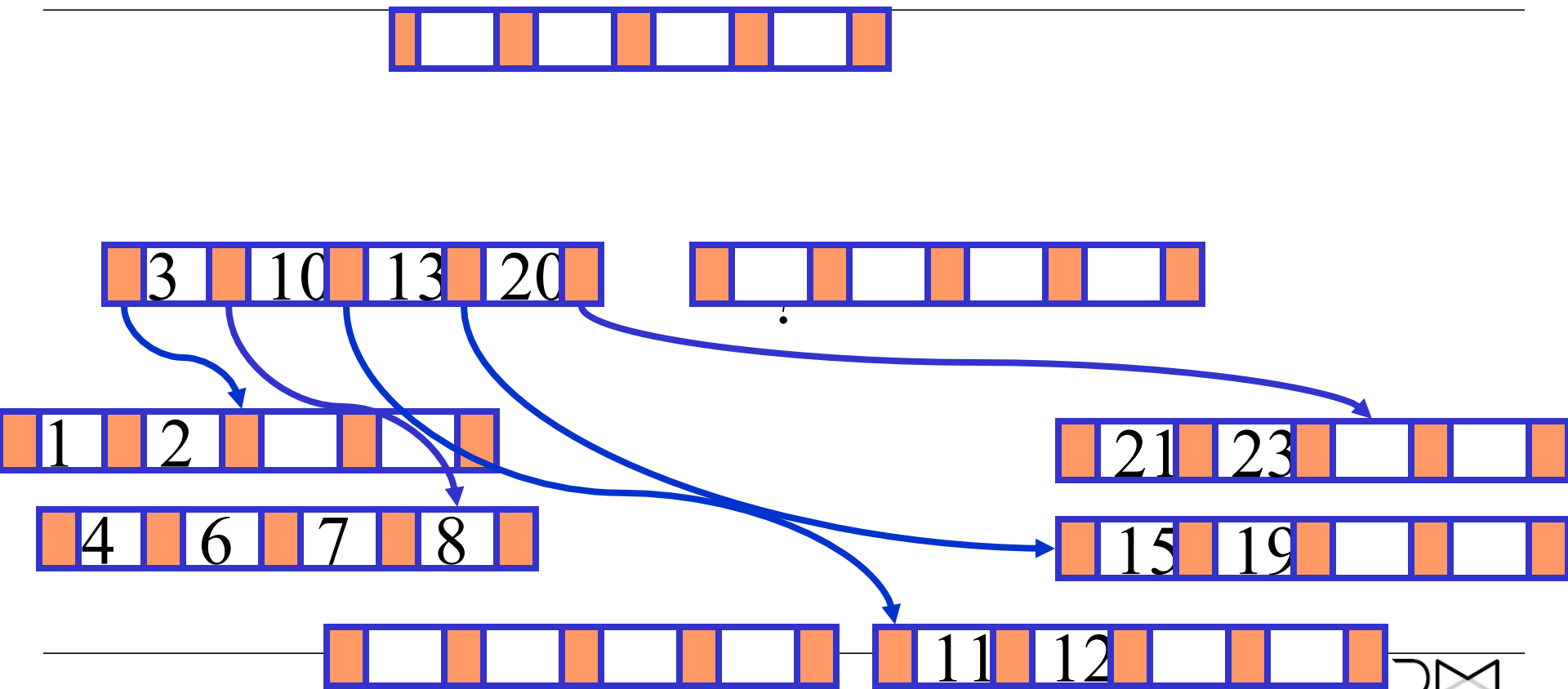
Move

Deletion

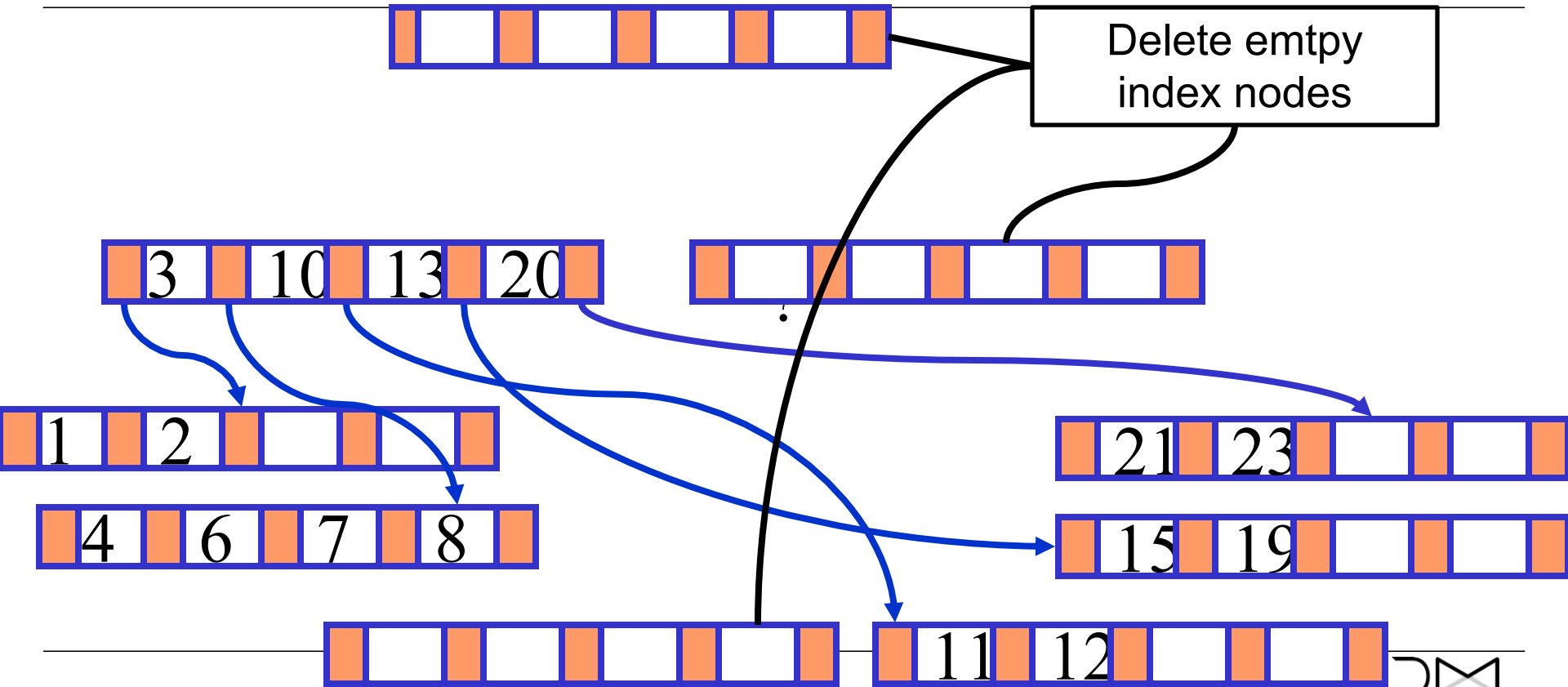
Merge



B-tree: Deletion



B-tree: Deletion



B-tree: Deletion

