3

$= n = k$

N no of keys.

COL 362 & COL 632

Indexing

14 Mar 2023

$\max \log_{\lceil \frac{k}{2} \rceil}(N)$

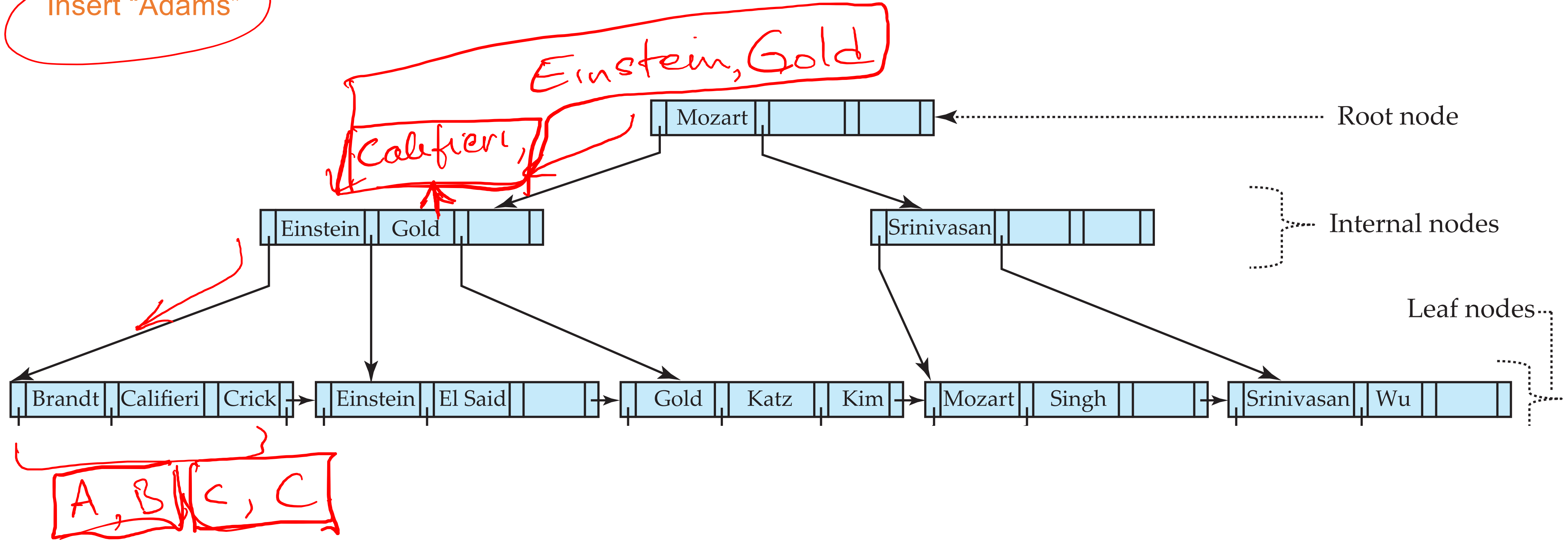
$\min \log_{k}(N)$

B+-Tree Insertion

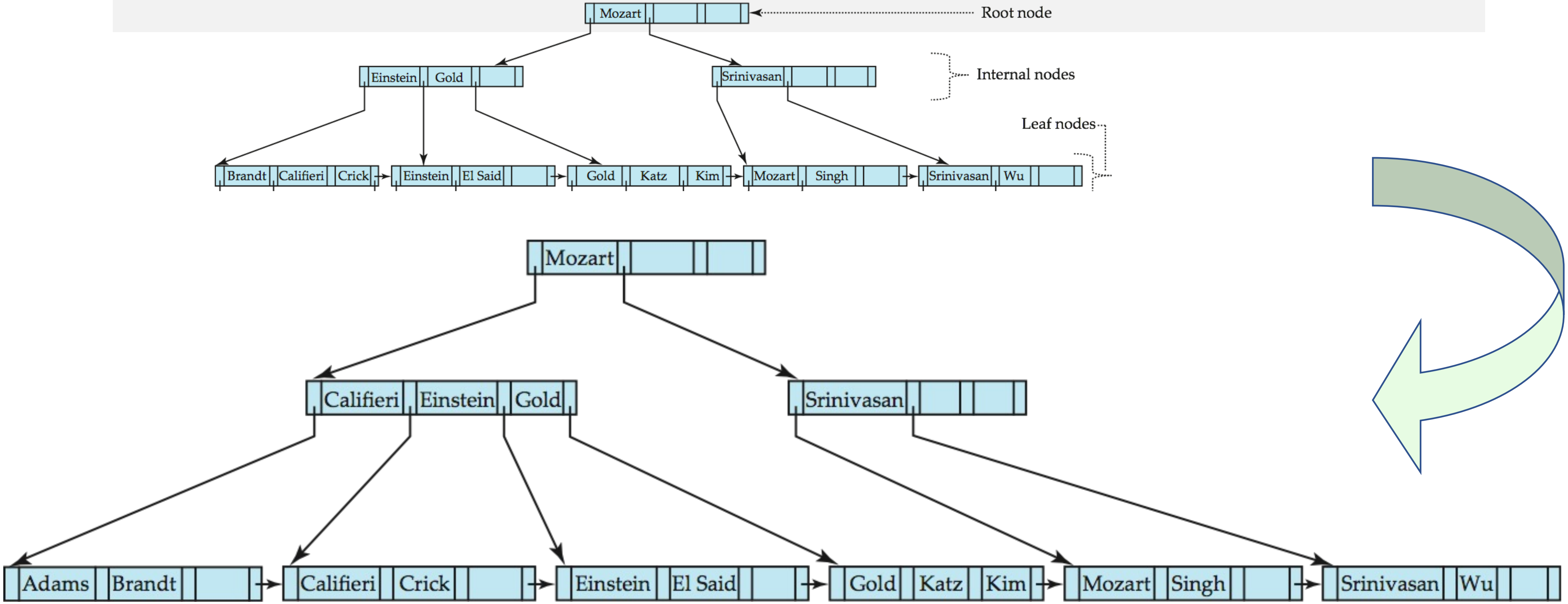
k = 3

~~varchar(4096)~~
varchar(20)

Insert "Adams"



B^+-Tree Insertion

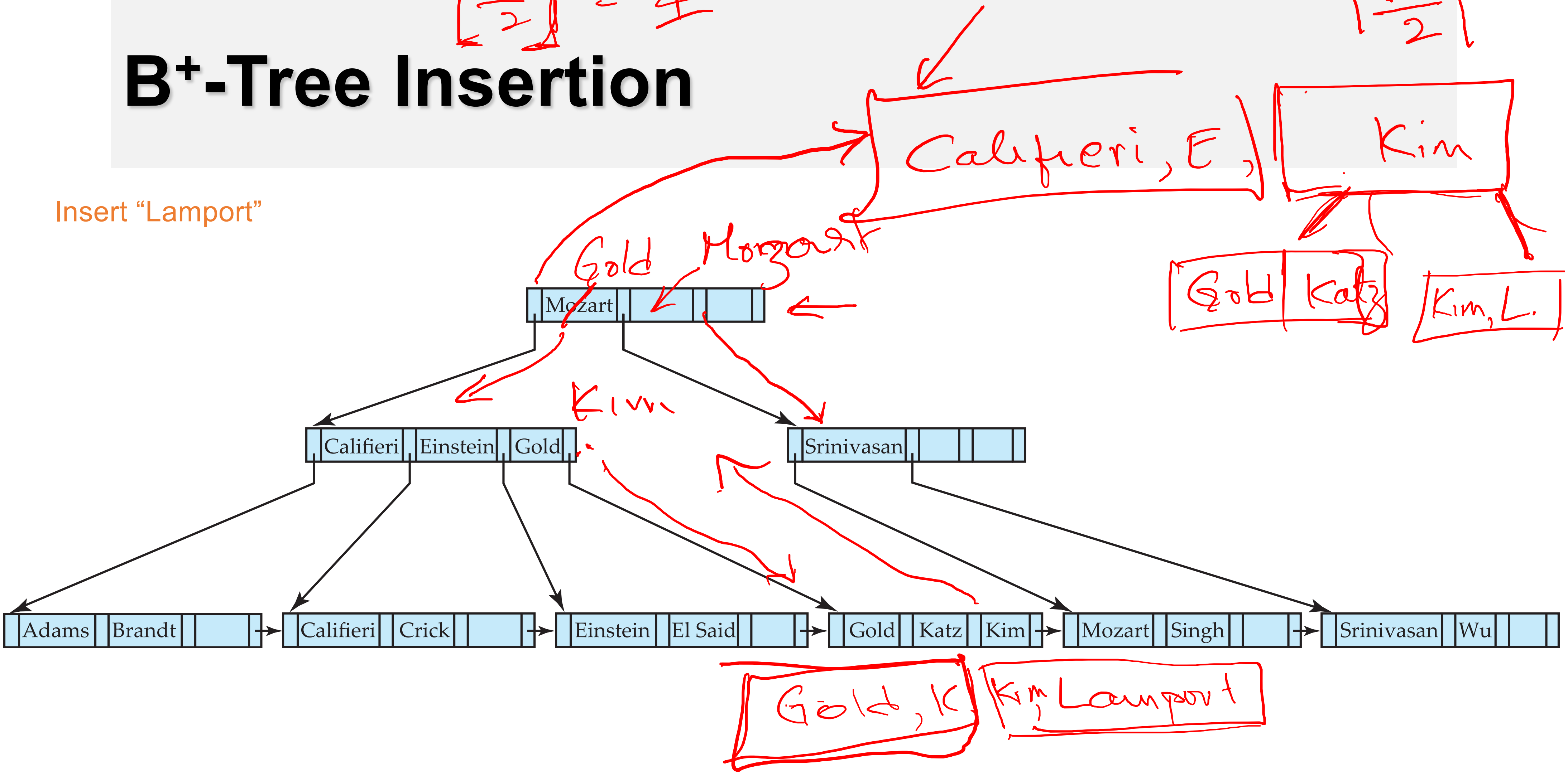


B^+-Tree before and after insertion of “Adams”

$\left\lceil \frac{3}{2} \right\rceil = 2$ $\left\lceil \frac{K}{2} \right\rceil$

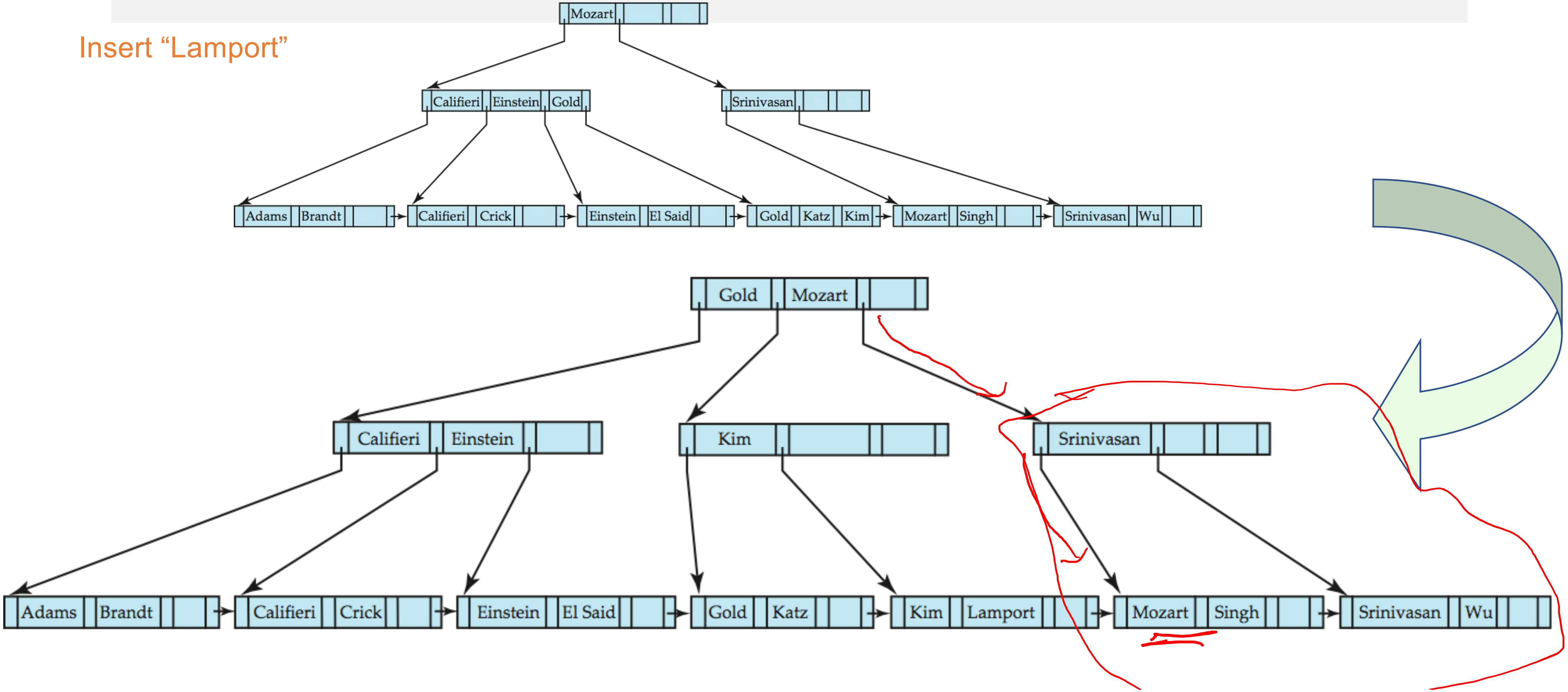
B⁺-Tree Insertion

Insert “Lamport”



B+-Tree Insertion

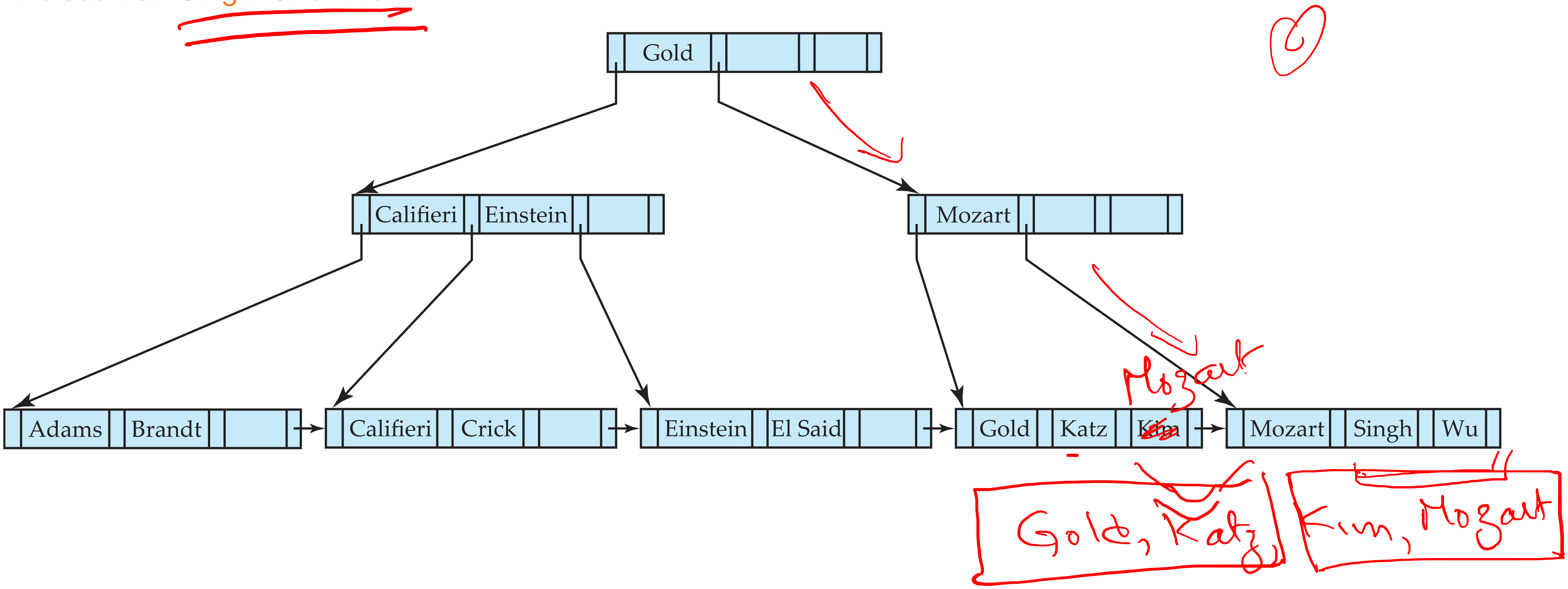
Insert “Lamport”



B+-trees – Deletion

B+-Tree Deletion (redistribute keys)

Deletion of ”Singh” and “Wu”



Gold

Califieri | Einstein

Mozart

Adams | Brandt

Califieri | Crick

Einstein | El Said

Gold | Katz | ~~Kim~~

Mozart | Singh | Wu

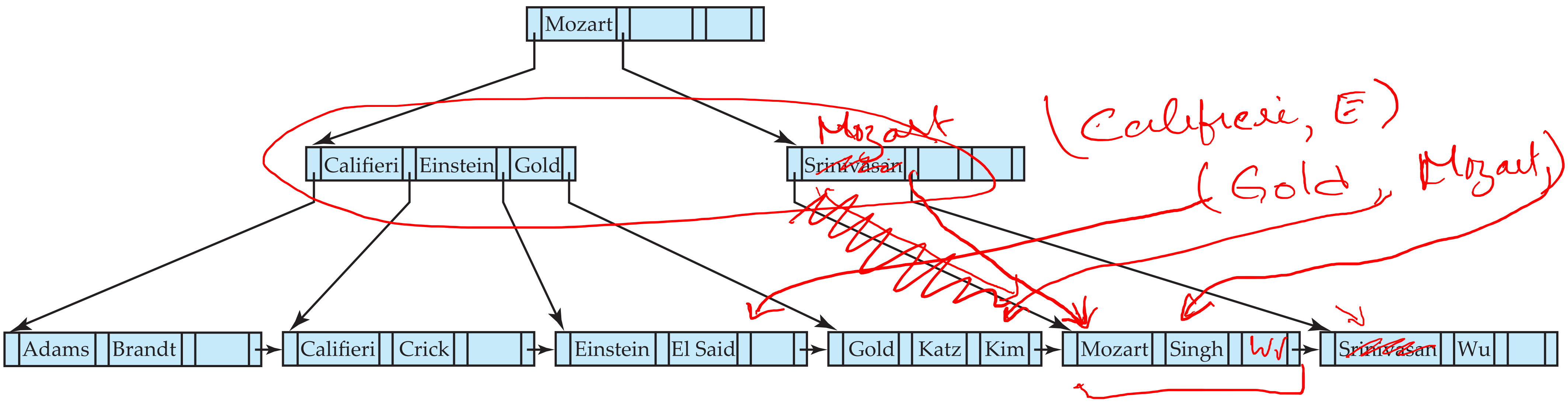
Mozart

Gold, Katz

Kim, Mozart

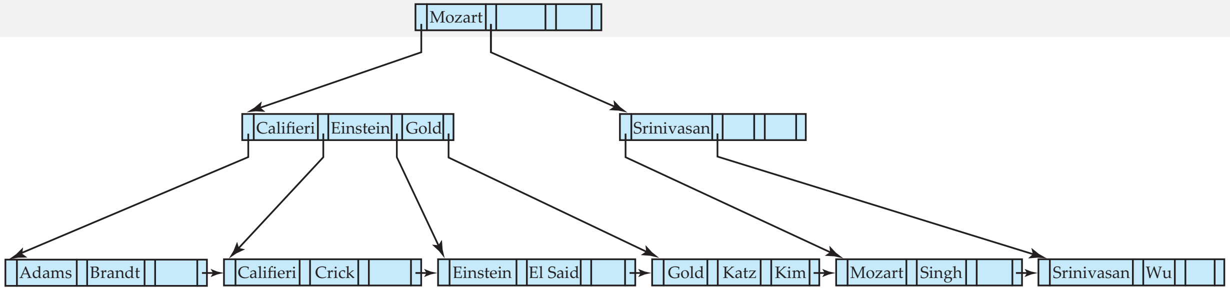
B$^+$-Tree Deletion (merge siblings)

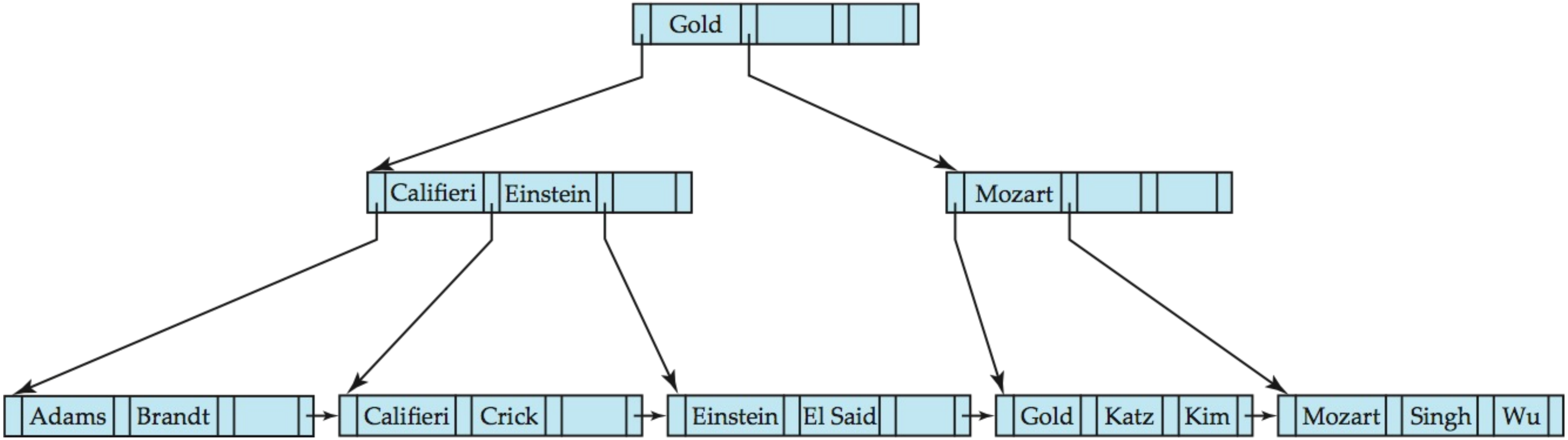
Delete "Srinivasan"



B$^+$-Tree Deletion (merge siblings)

Delete “Srinivasan”





100

Cost of Insertion and Deletion

- Insertion and Deletion
  - Worst-case no. of I/O operations needed : $\log_{\lceil \frac{n}{2} \rceil}(N)$
- Proportional to the height of the tree => efficient ✓
- In practice, no. of disk operations is much fewer
- Most of non-leaf nodes are already in buffer !
- With a fan-out of 100 (or sometimes even more) most inserts / deletions do not require splitting / merging

Self-study

- **B+Tree** – Handling of non-unique keys, Index file organization, bulk loading, and indexing of variable length strings (prefix compression)

- **Hash-indexes** – static hashing techniques, closed hashing

- Create Index SQL command
- Bitmap index

- (We will **not** focus on LSM, Spatial - Temporal indexesetc.)