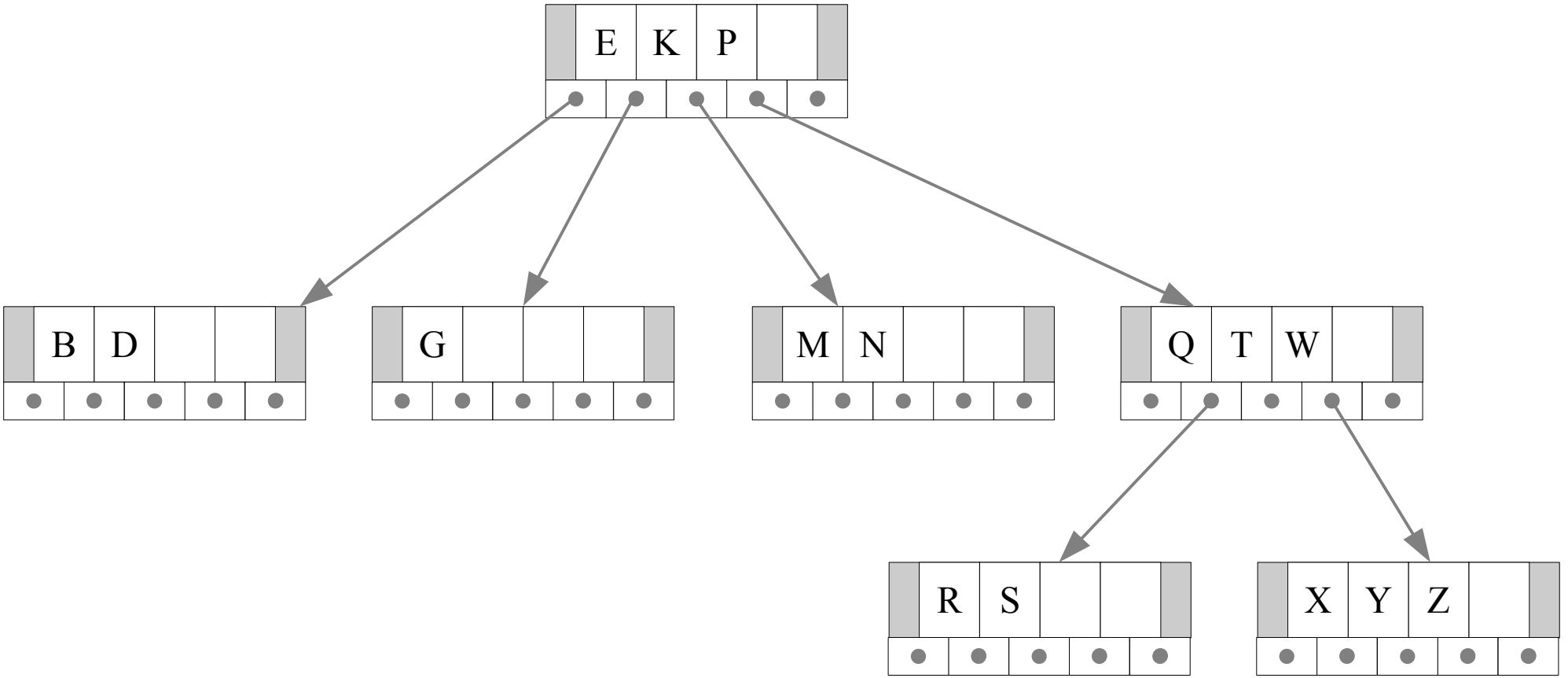
Example multiway tree of order 5
(This is not a B-Tree, BTW)



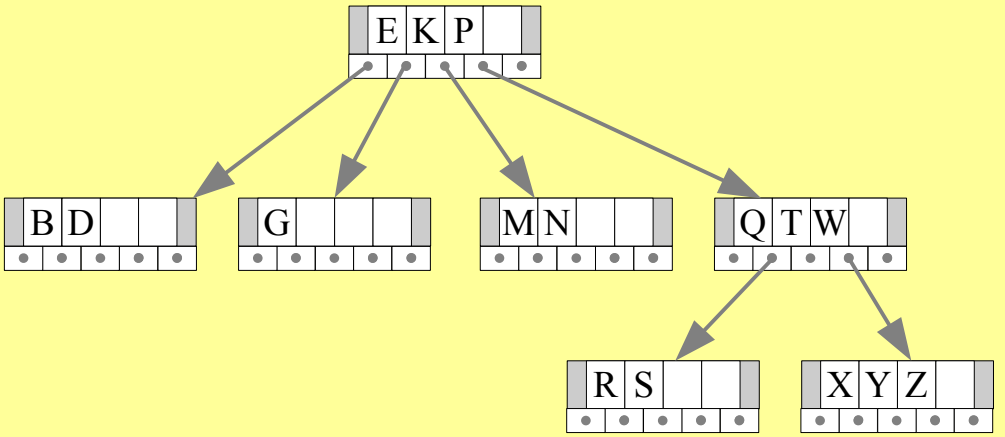
B-Tree Invariants

- Affix the order - *m*
- All nodes, except the top, have *ceil(m/2)-1* keys.
- All external nodes are at the same level.

B-Tree Invariants

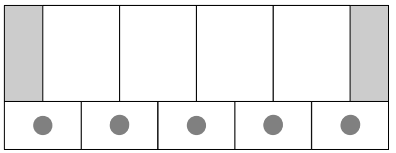
- Affix the order - *m*
- All nodes, except the top, have *ceil(m/2)-1* keys.
- All external nodes are at the same level.

The tree from the first page is not a B-Tree.



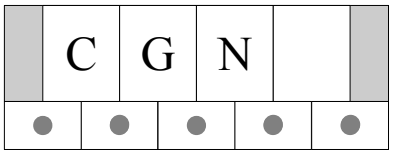
To Construct a B-Tree

- Fill in the first node's keys, until it's full
- Example: Insert C, G and N into a tree of order 5:



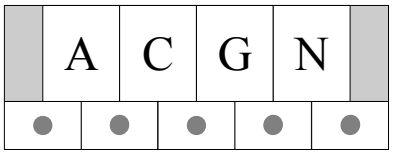
To Construct a B-Tree

- Fill in the first node's keys, until it's full
- Example: Insert C, G and N into a tree of order 5:
- Next, insert A:



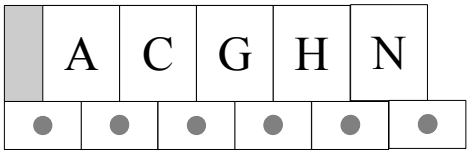
To Construct a B-Tree

- Fill in the first node's keys, until it's full
- Example: Insert C, G and N into a tree of order 5:
- Next, insert A:
- Suppose we now insert H.



To Construct a B-Tree

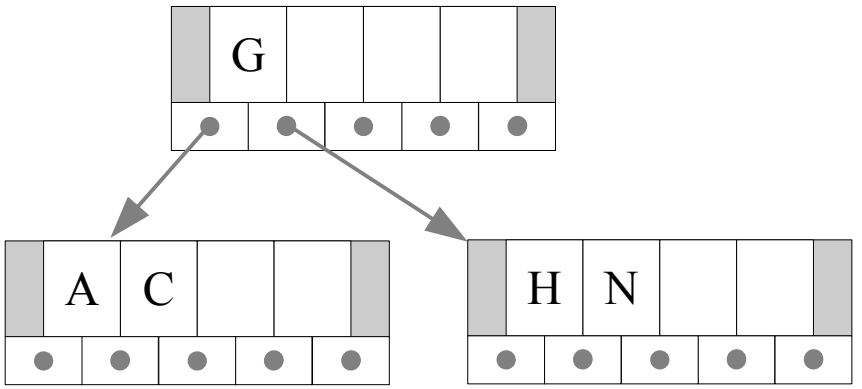
- Fill in the first node's keys, until it's full
- Example: Insert C, G and N into a tree of order 5:
- Next, insert A:
- Suppose we now insert H.



It's too big now,
so we “split” it into
three nodes:

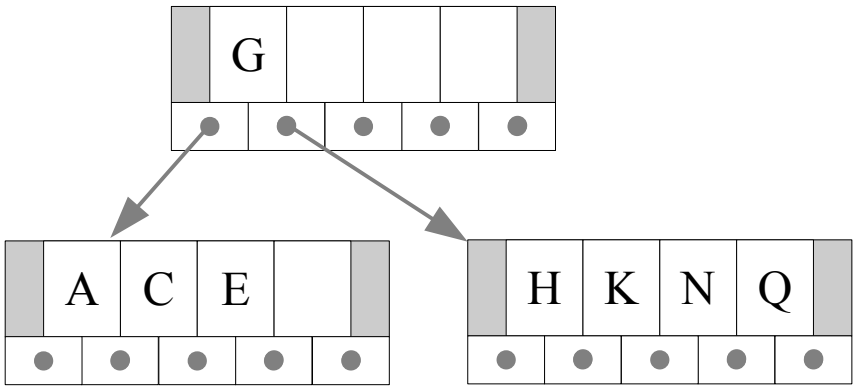
To Construct a B-Tree

- Next, we'll insert E, K and Q – these all go into leaf nodes.



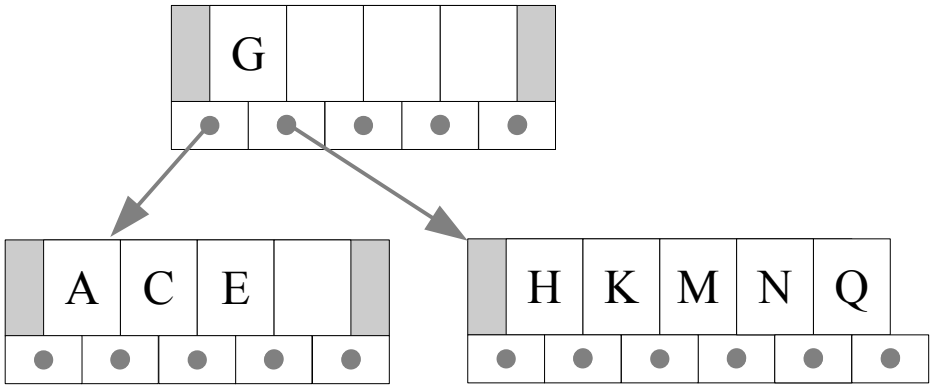
To Construct a B-Tree

- Next, we'll insert E, K and Q – these all go into leaf nodes.
- Next, insert M – this requires a split:



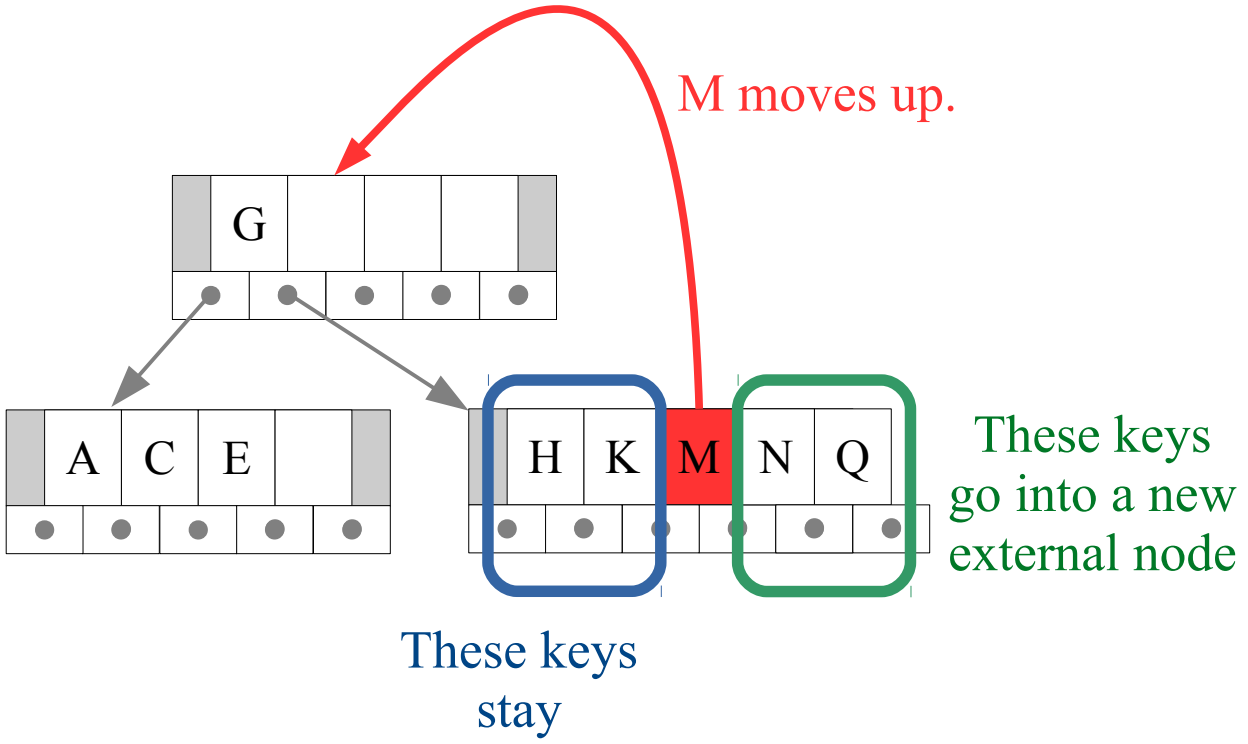
To Construct a B-Tree

- Next, we'll insert E, K and Q – these all go into leaf nodes.
- Next, insert M – this requires a split:



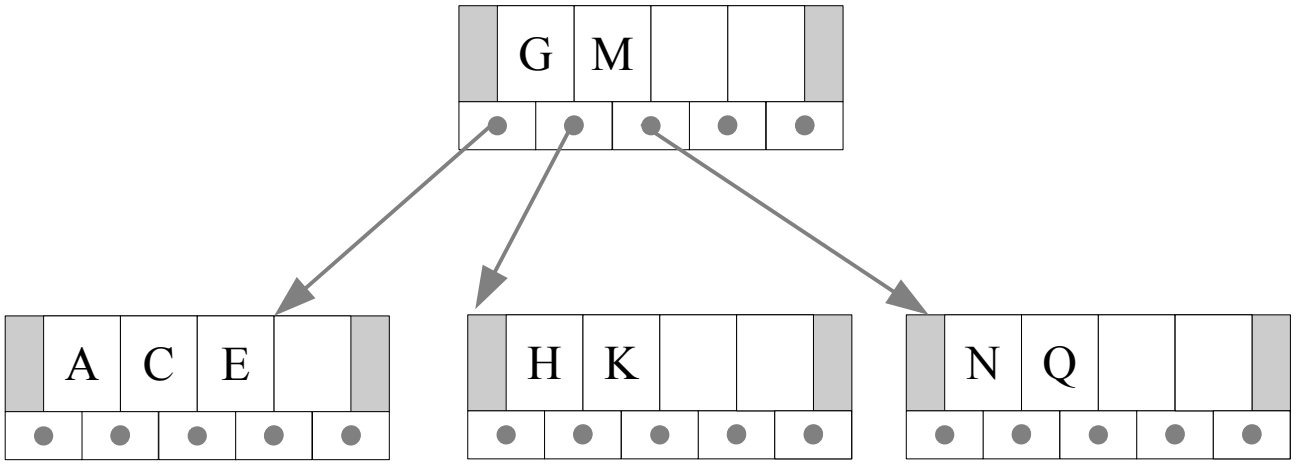
To Construct a B-Tree

- Next, we'll insert E, K and Q – these all go into leaf nodes.
- Next, insert M – this requires a split:



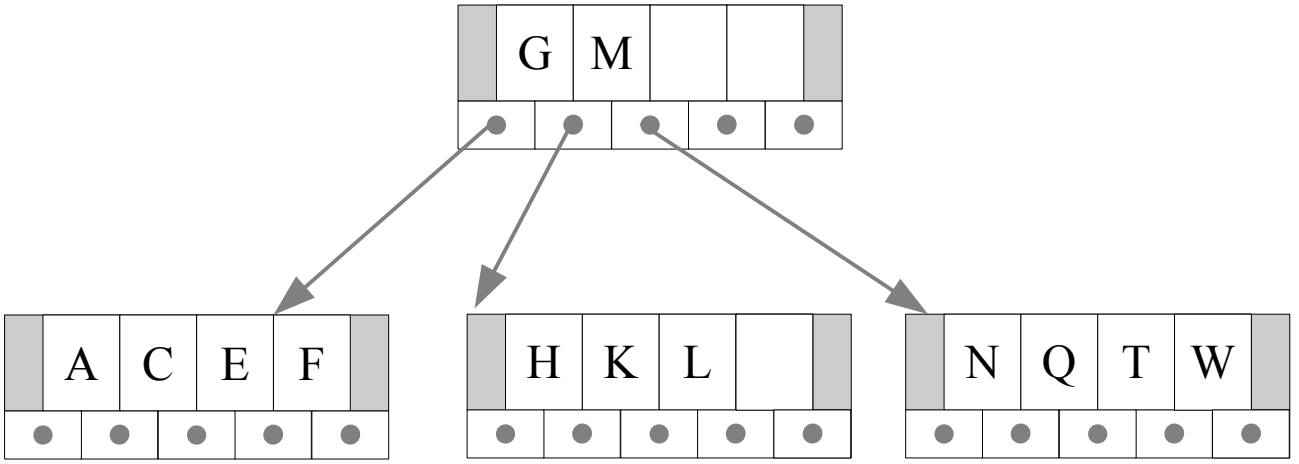
To Construct a B-Tree

- Next, we'll insert E, K and Q – these all go into leaf nodes.
- Next, insert M – this requires a split.
- Insert F, W, L, T



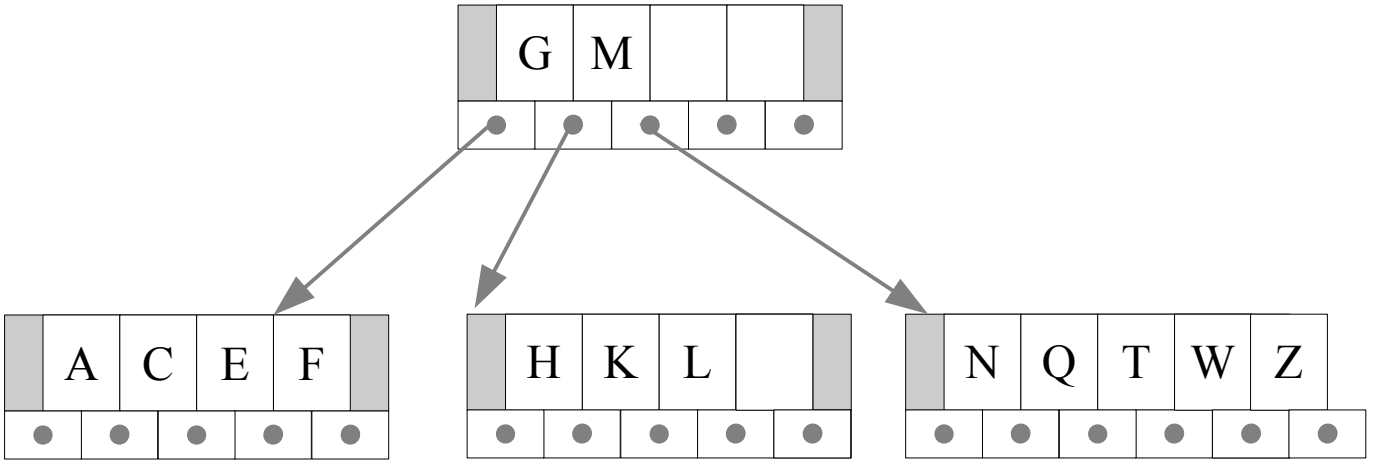
To Construct a B-Tree

- Next, we'll insert E, K and Q – these all go into leaf nodes.
- Next, insert M – this requires a split.
- Insert F, W, L, T
- Insert Z: Another split



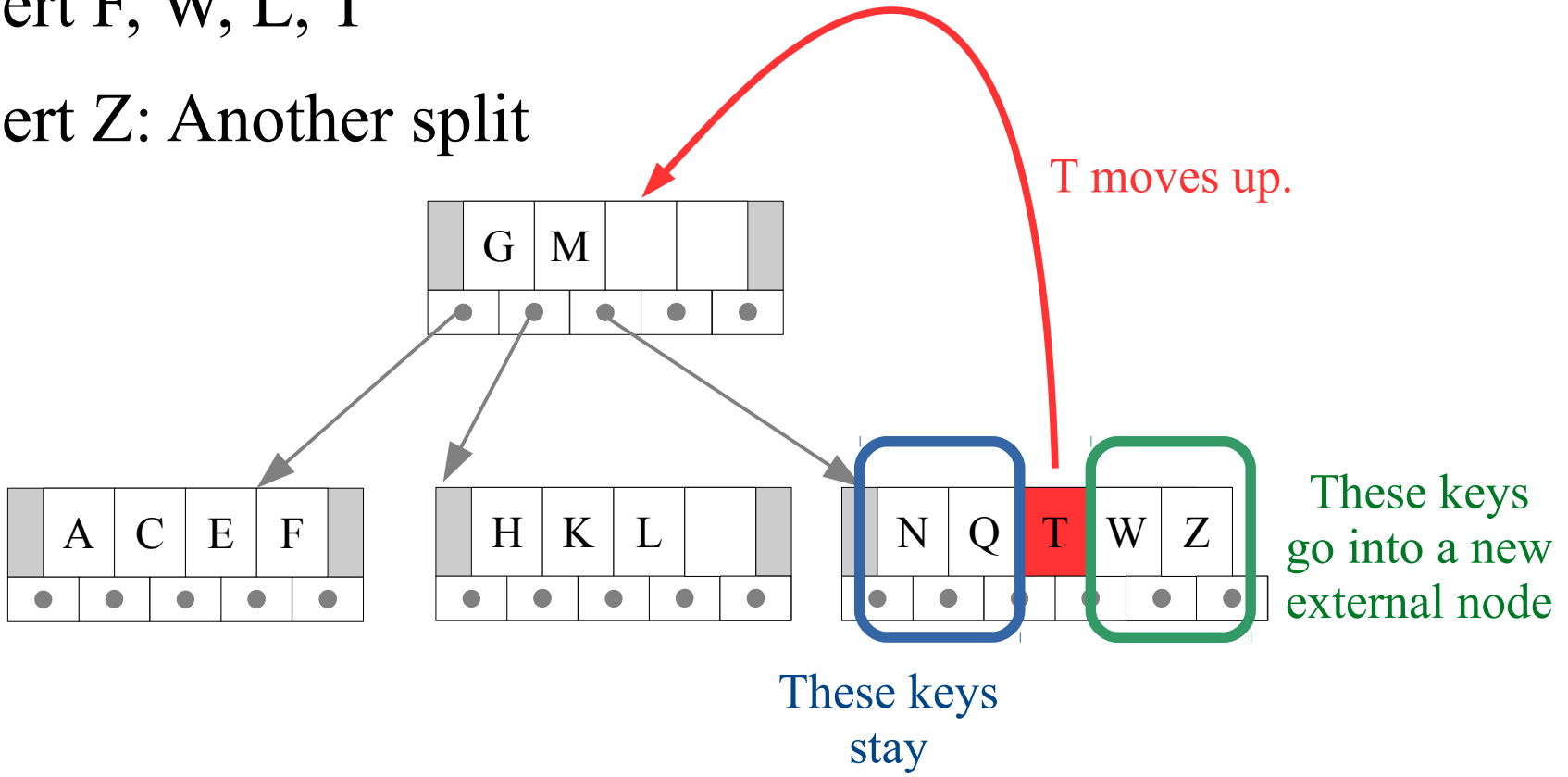
To Construct a B-Tree

- Next, we'll insert E, K and Q – these all go into leaf nodes.
- Next, insert M – this requires a split.
- Insert F, W, L, T
- Insert Z: Another split



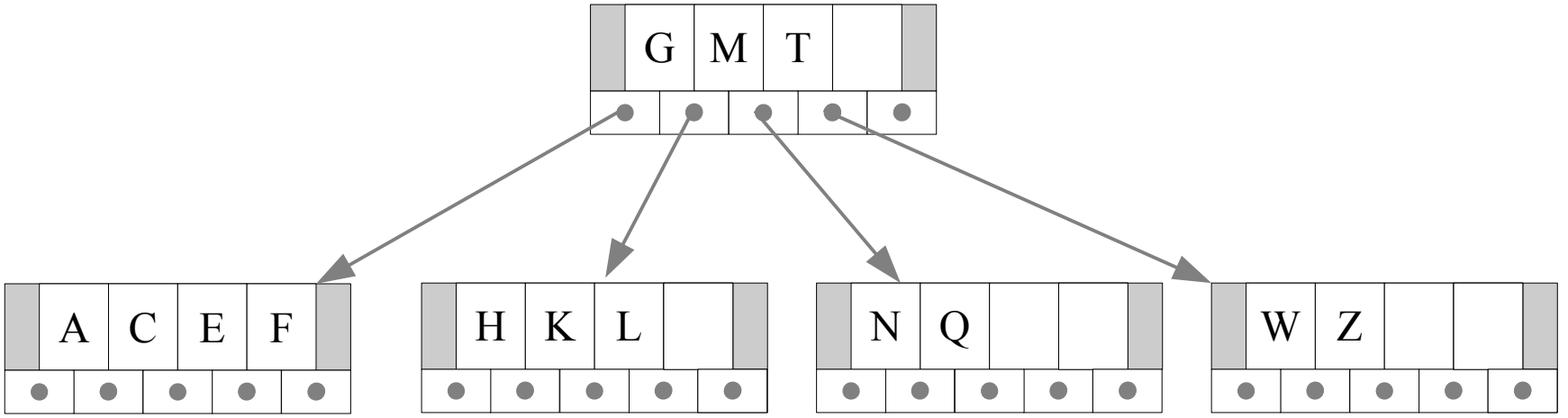
To Construct a B-Tree

- Next, we'll insert E, K and Q – these all go into leaf nodes.
- Next, insert M – this requires a split.
- Insert F, W, L, T
- Insert Z: Another split



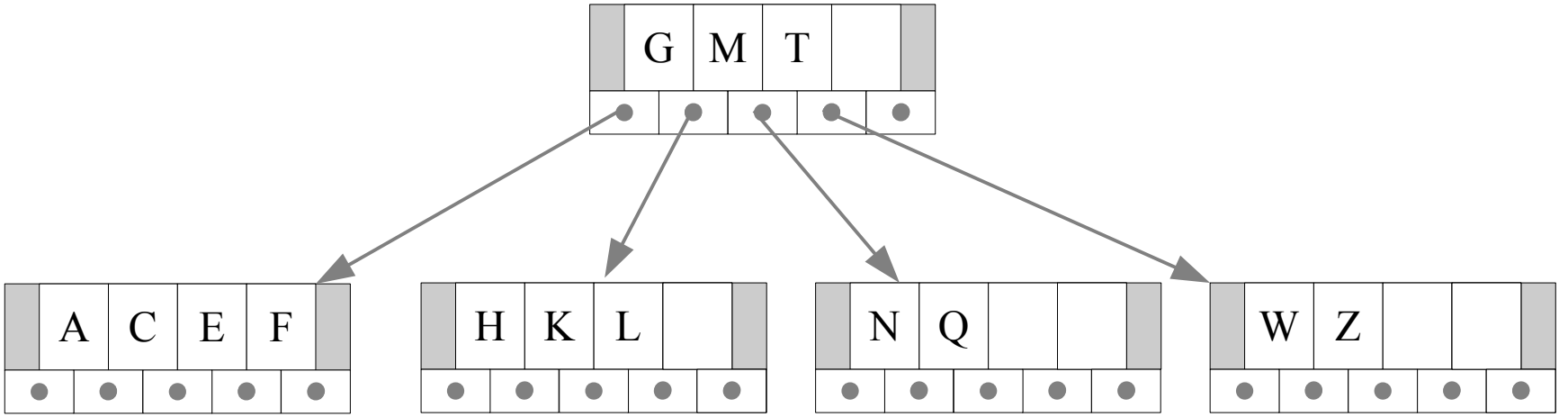
To Construct a B-Tree

- Next, we'll insert E, K and Q – these all go into leaf nodes.
- Next, insert M – this requires a split.
- Insert F, W, L, T
- Insert Z: Another split



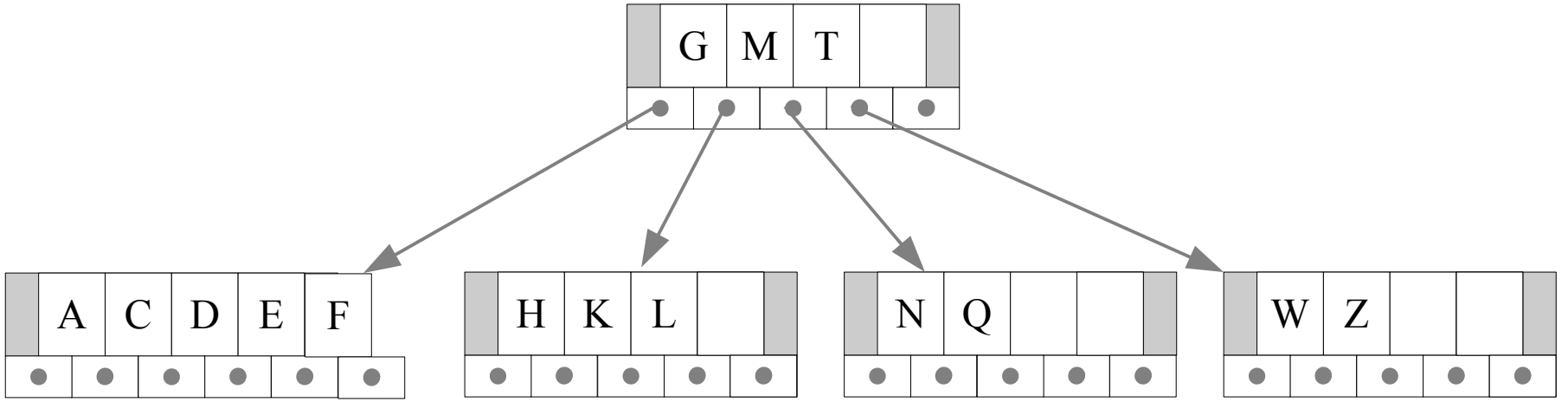
To Construct a B-Tree

- Insert D – another split:.



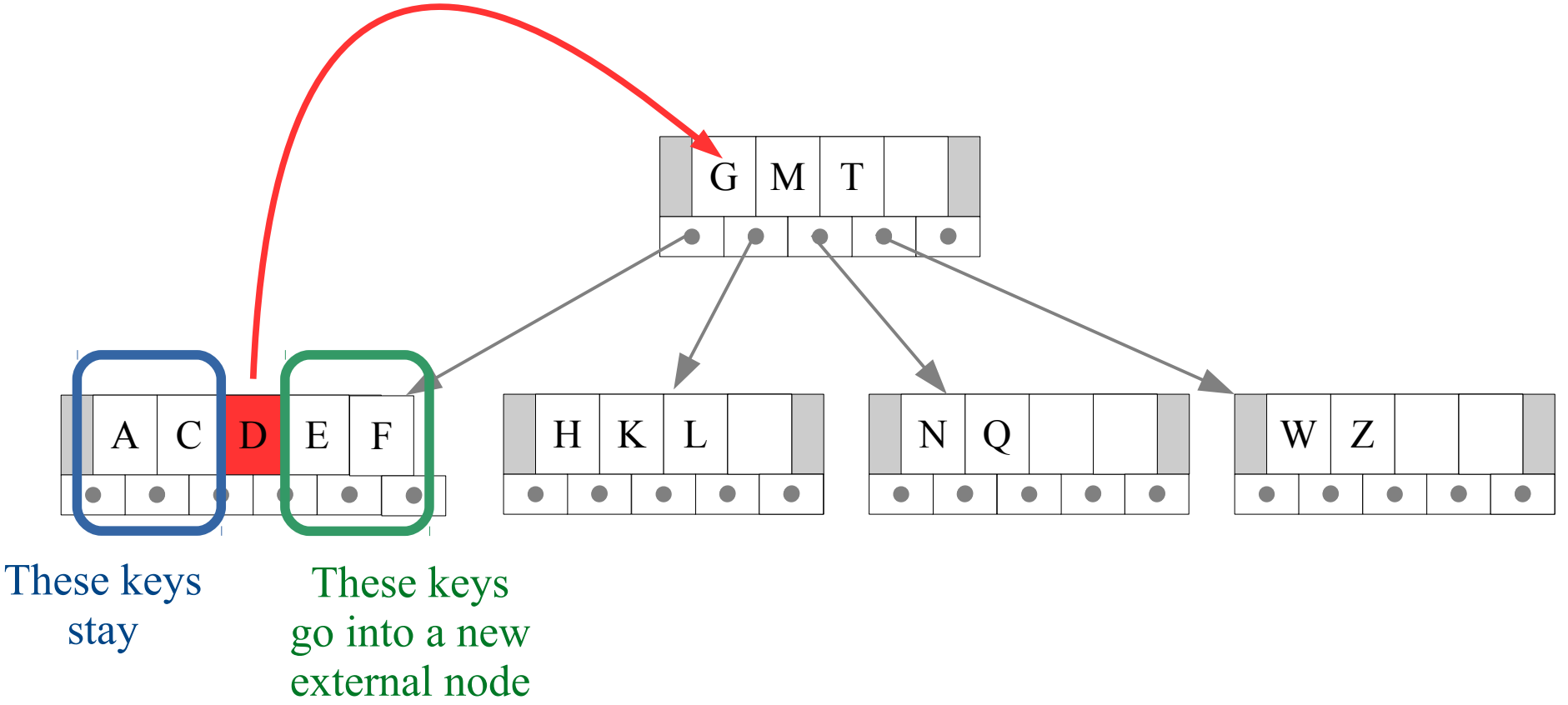
To Construct a B-Tree

- Insert D – another split:



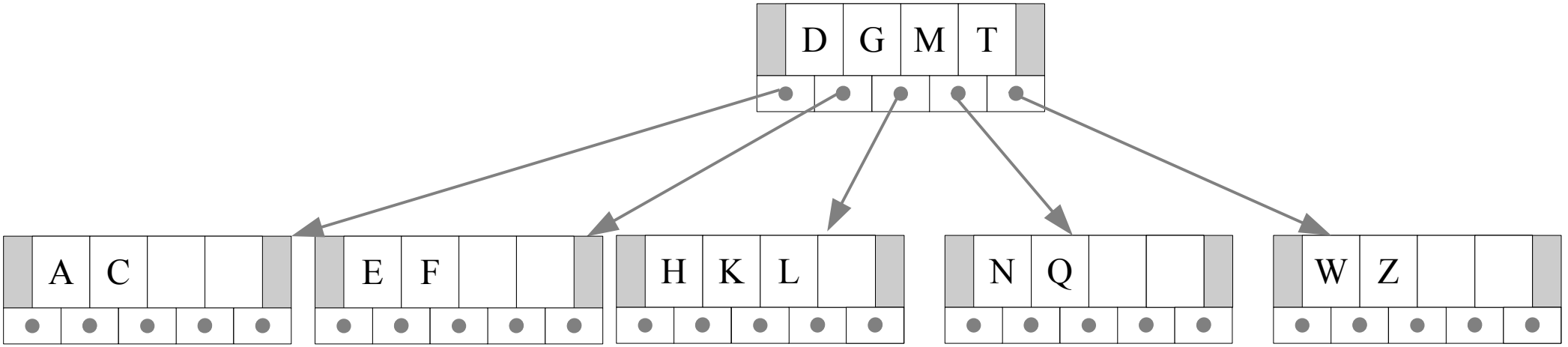
To Construct a B-Tree

- Insert D – another split:



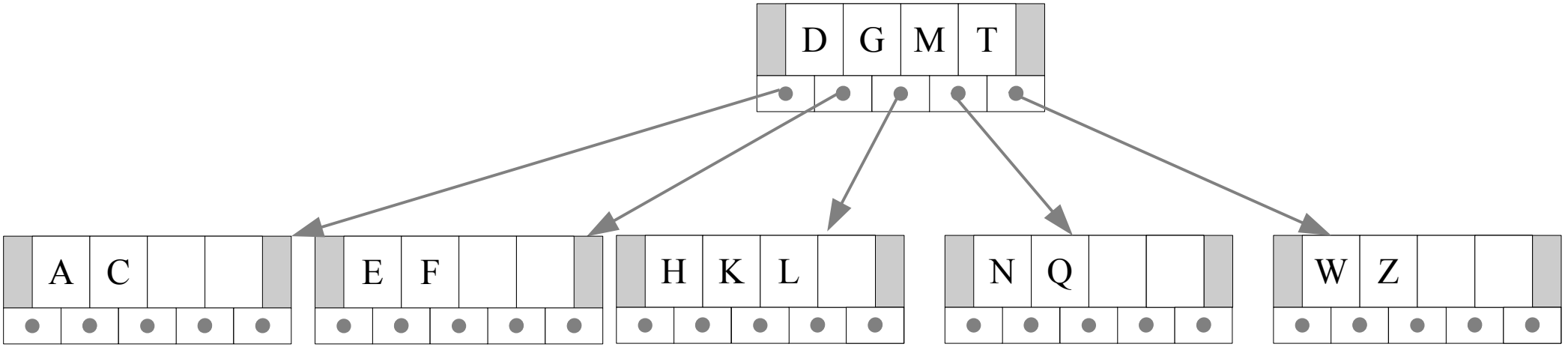
To Construct a B-Tree

- Insert D – another split:



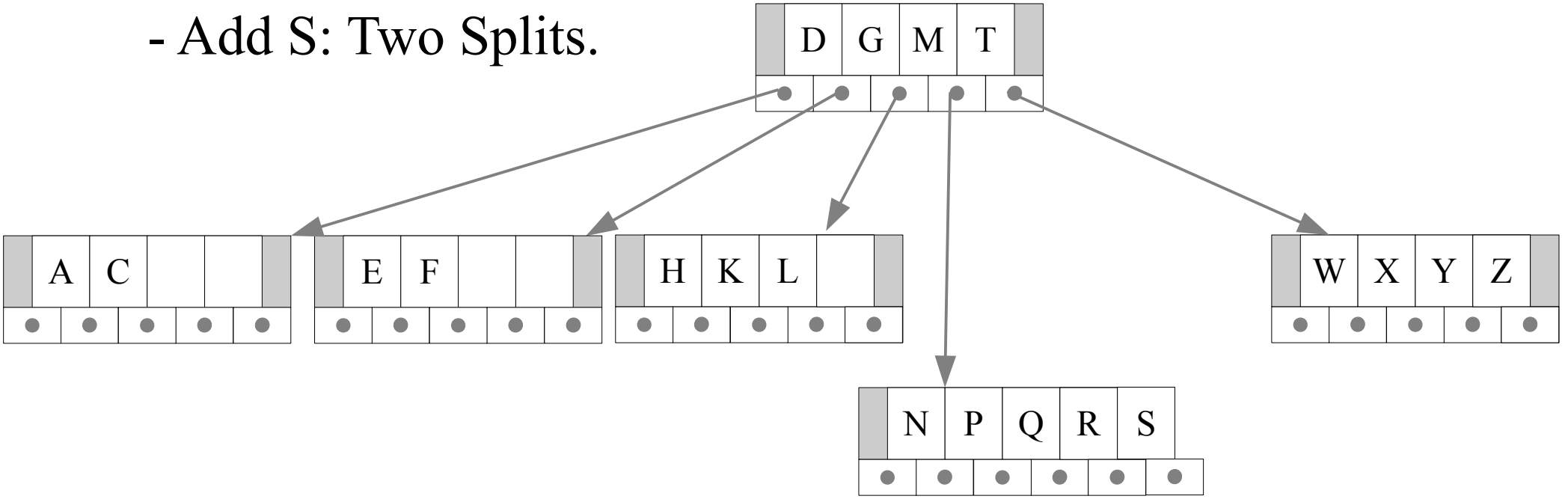
To Construct a B-Tree

- Insert D – another split:
- Add P, R, X, Y



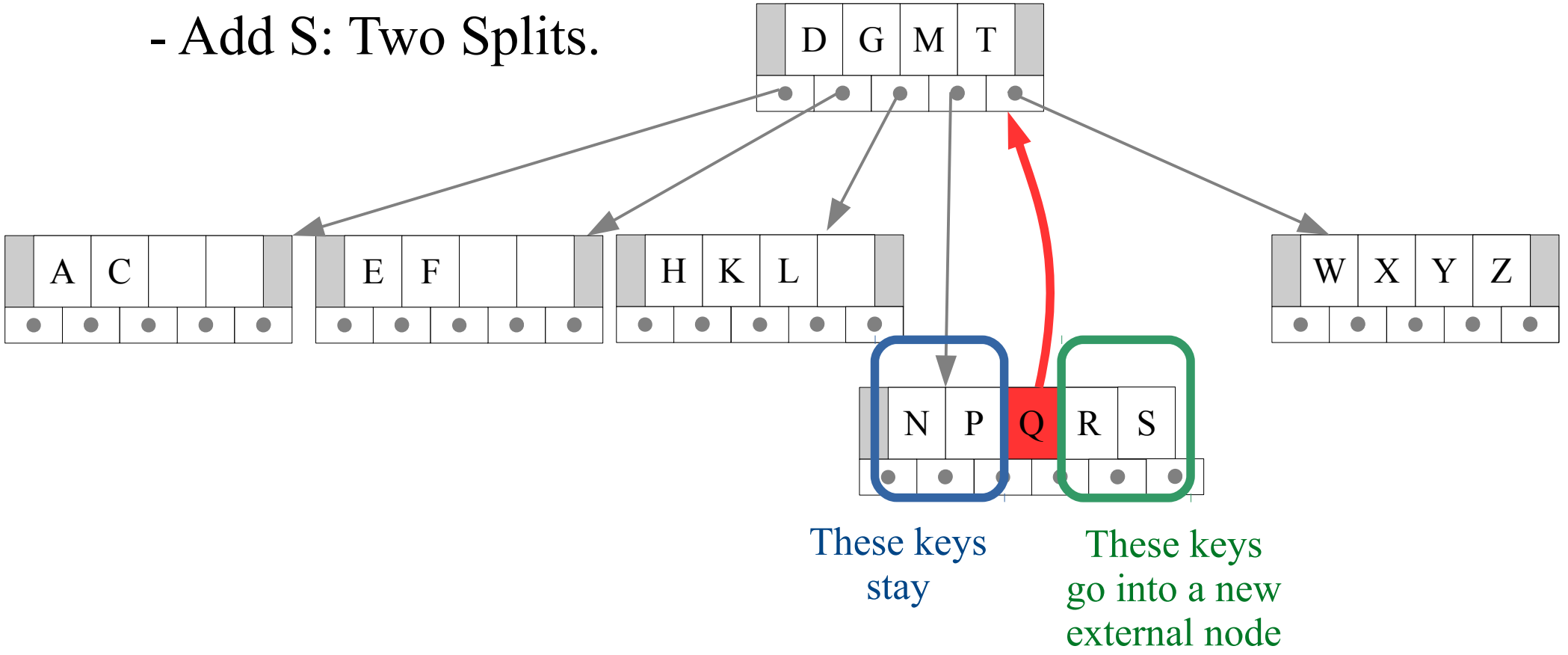
To Construct a B-Tree

- Insert D – another split:
- Add P, R, X, Y
- Add S: Two Splits.



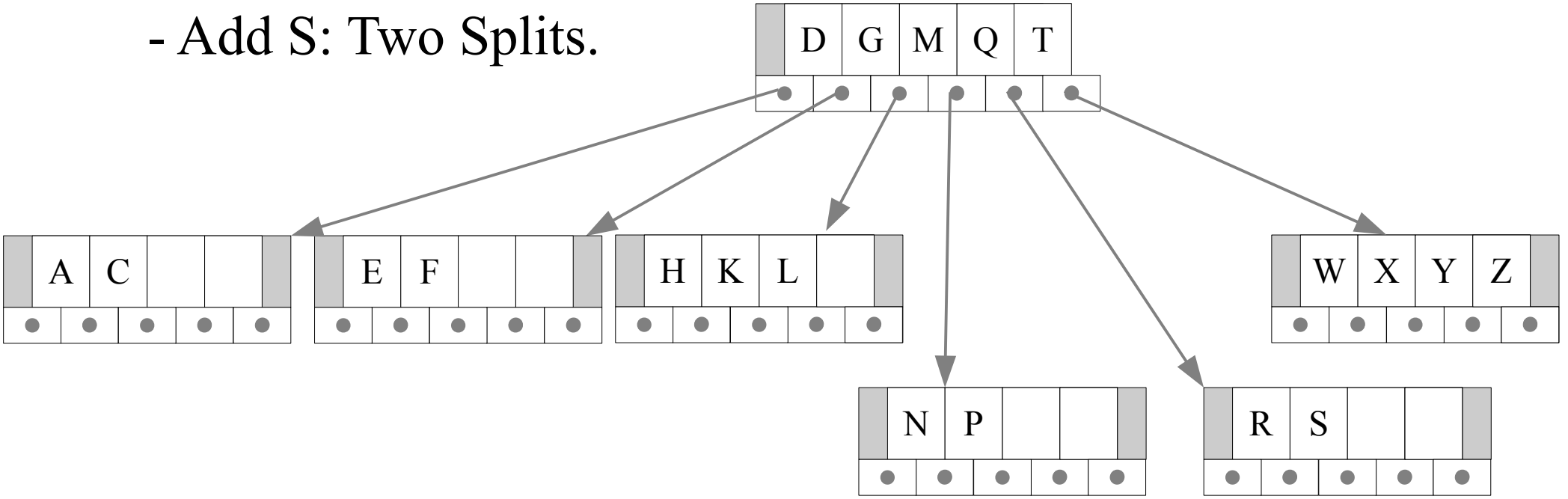
To Construct a B-Tree

- Insert D – another split:
- Add P, R, X, Y
- Add S: Two Splits.



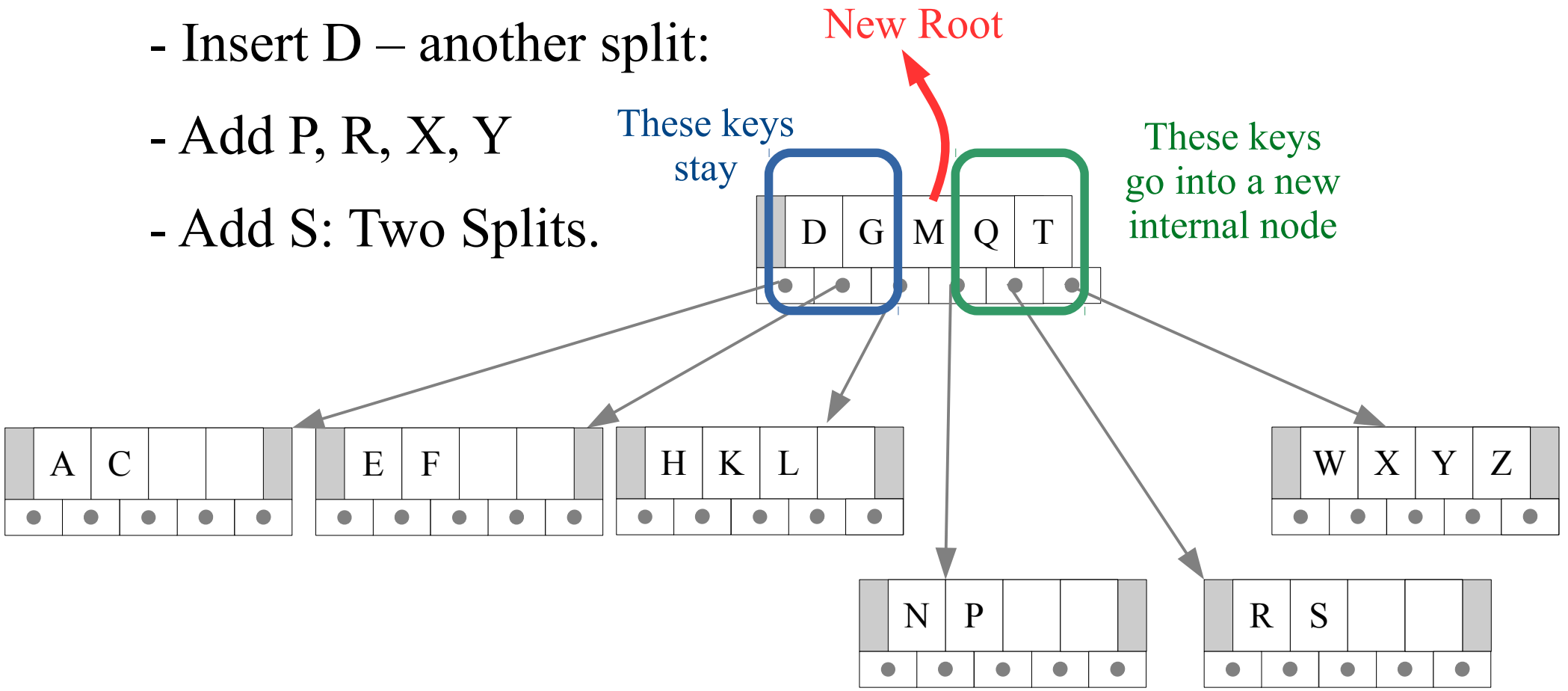
To Construct a B-Tree

- Insert D – another split:
- Add P, R, X, Y
- Add S: Two Splits.



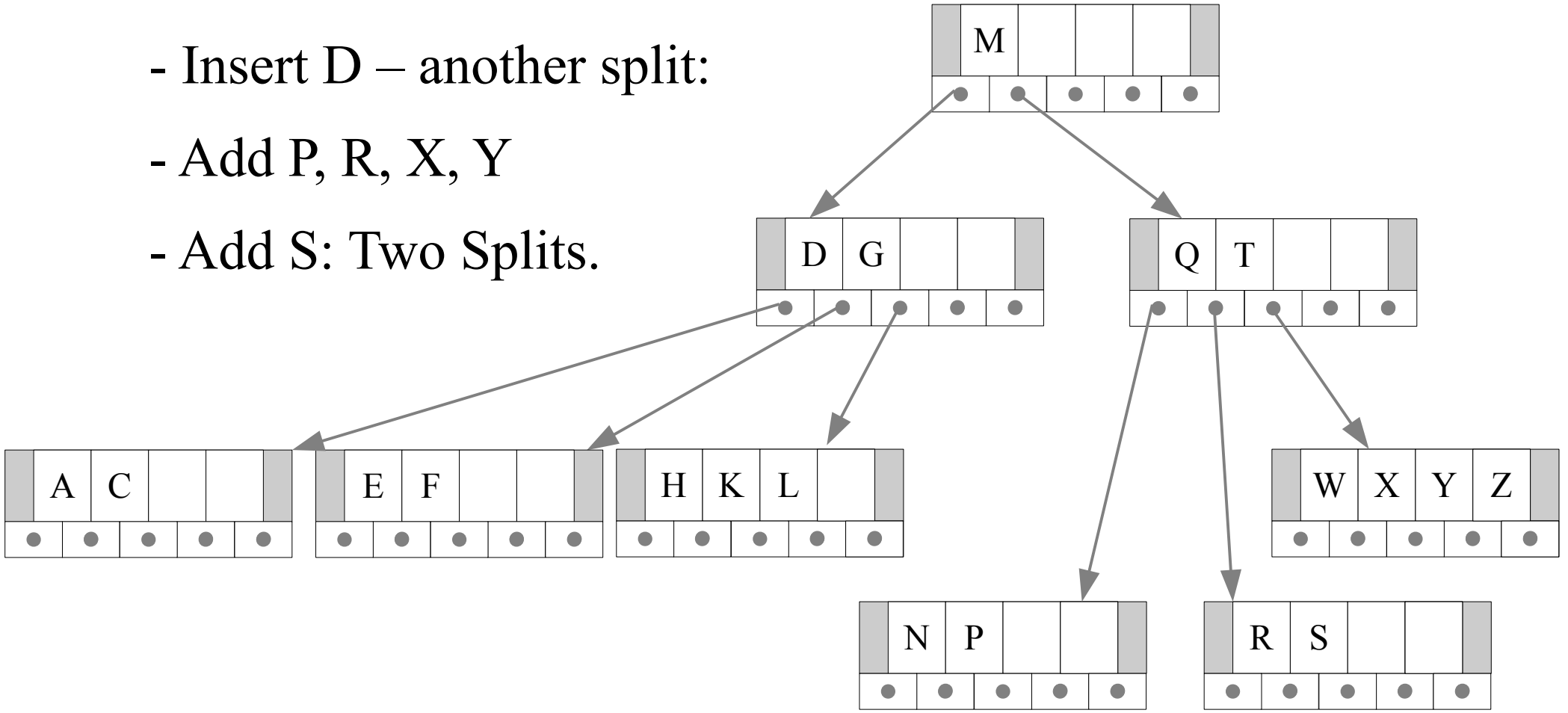
To Construct a B-Tree

- Insert D – another split:
- Add P, R, X, Y
- Add S: Two Splits.



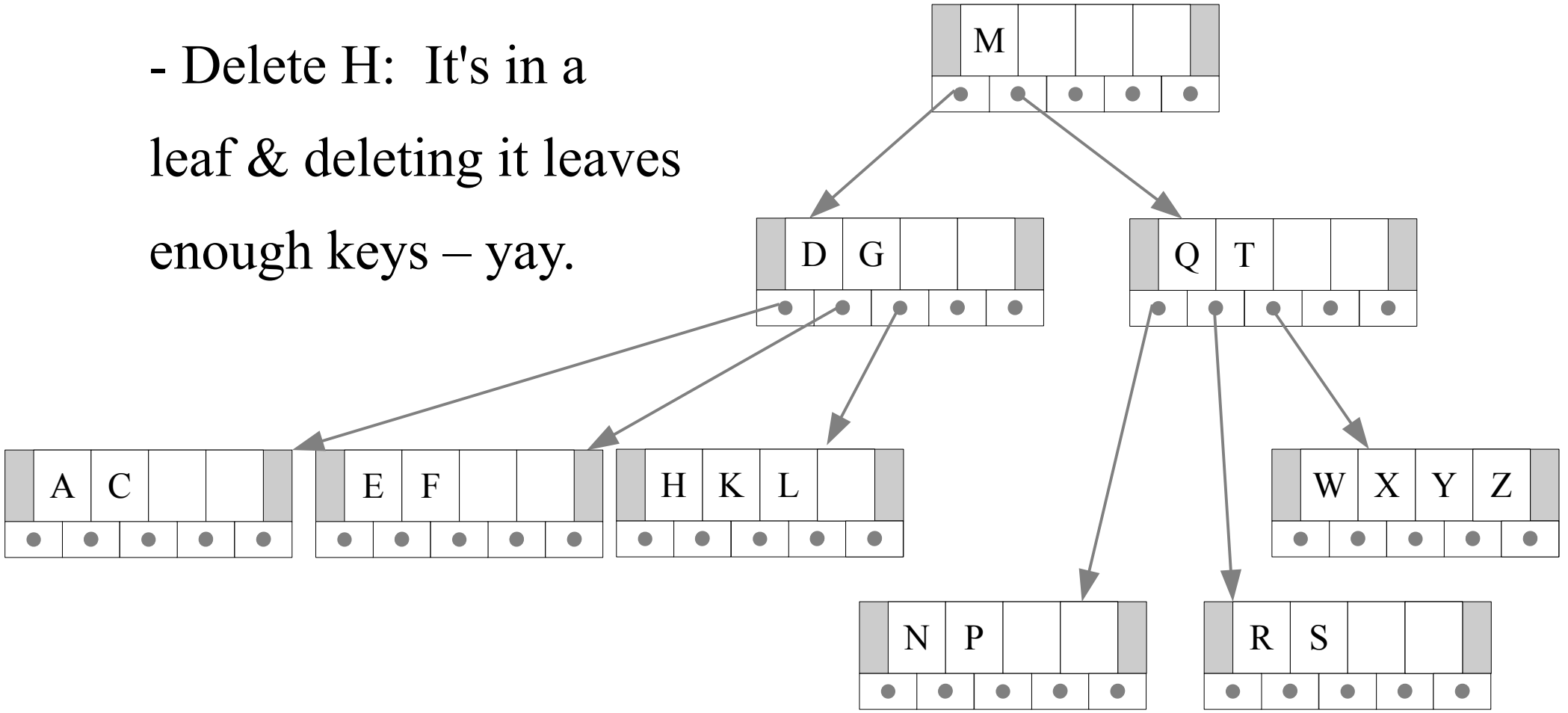
To Construct a B-Tree

- Insert D – another split:
- Add P, R, X, Y
- Add S: Two Splits.



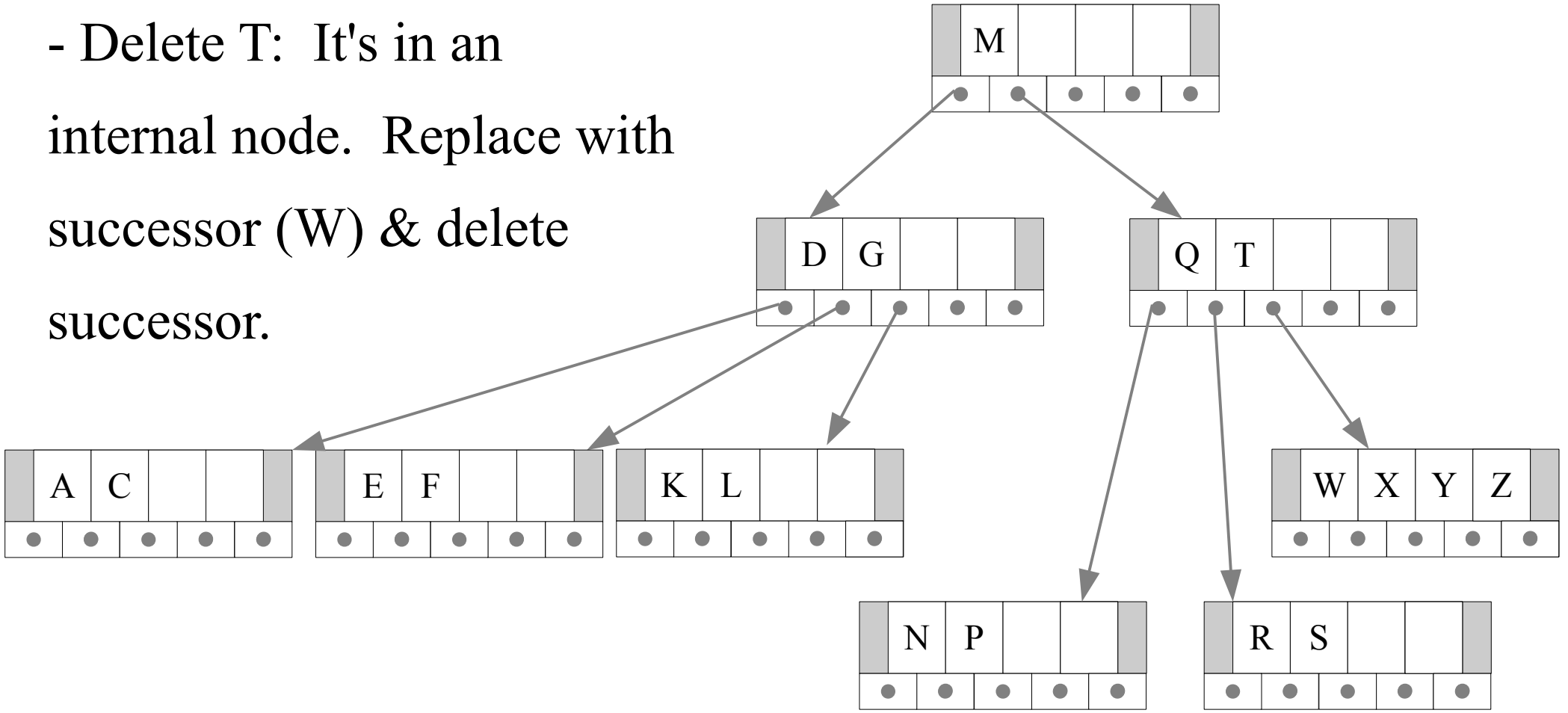
Onto Deletion

- Delete H: It's in a leaf & deleting it leaves enough keys – yay.



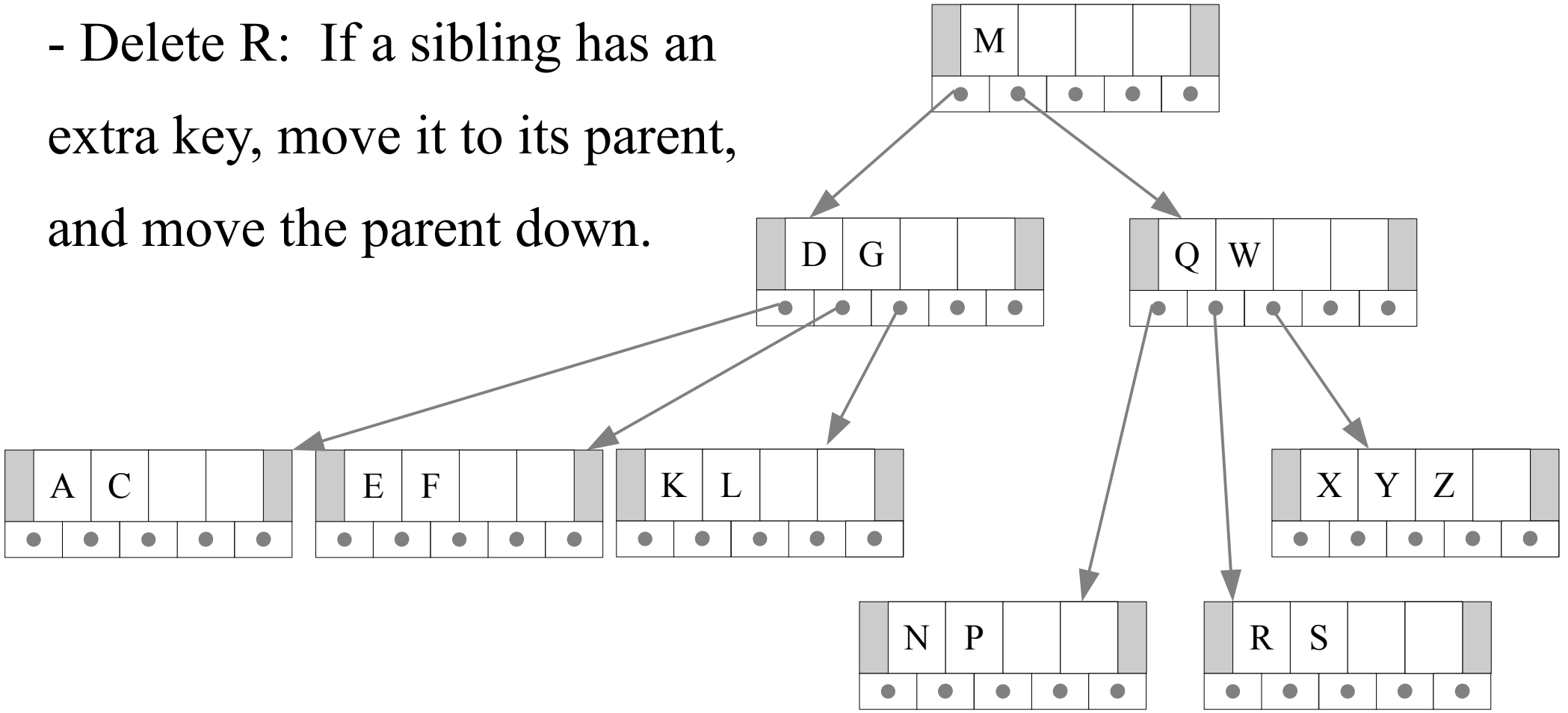
Onto Deletion

- Delete T: It's in an internal node. Replace with successor (W) & delete successor.



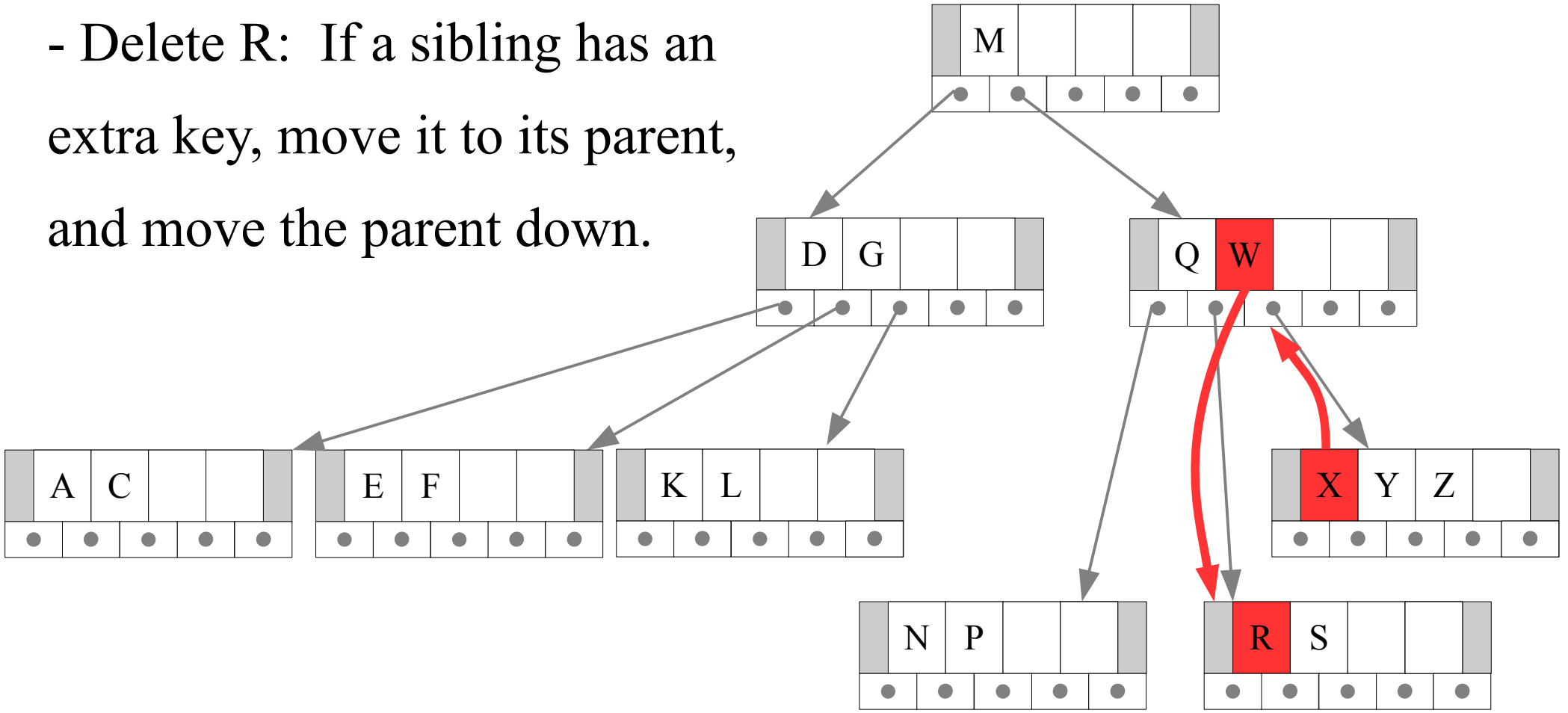
Onto Deletion

- Delete R: If a sibling has an extra key, move it to its parent, and move the parent down.



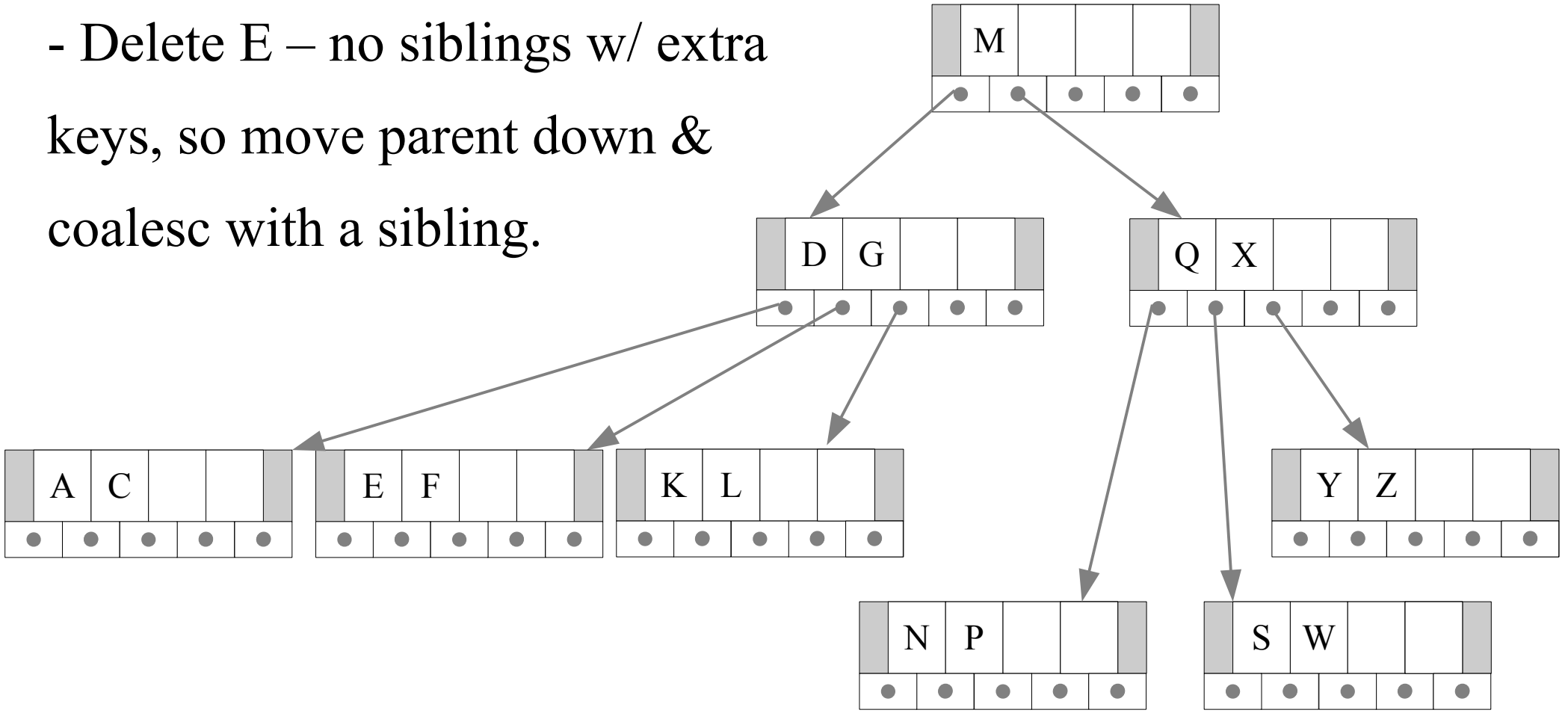
Onto Deletion

- Delete R: If a sibling has an extra key, move it to its parent, and move the parent down.



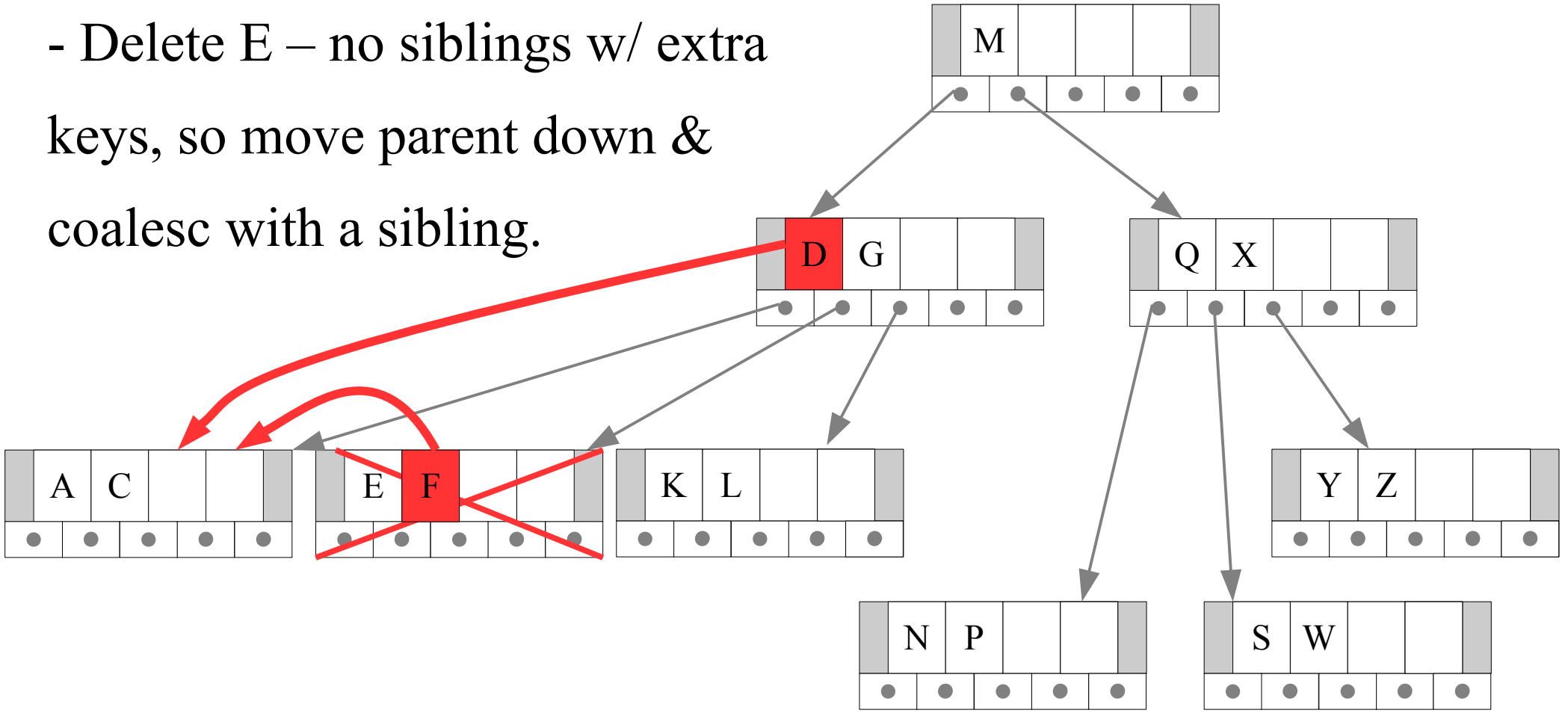
Onto Deletion

- Delete E – no siblings w/ extra keys, so move parent down & coalesc with a sibling.



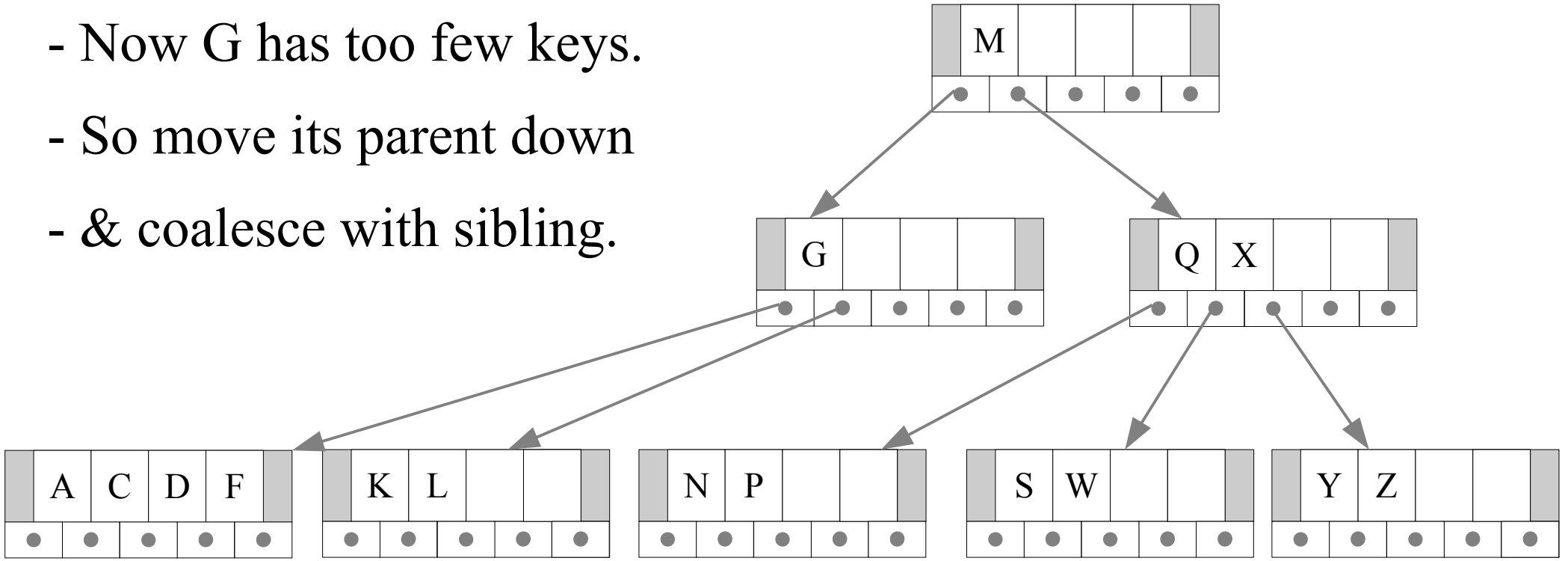
Onto Deletion

- Delete E – no siblings w/ extra keys, so move parent down & coalesc with a sibling.



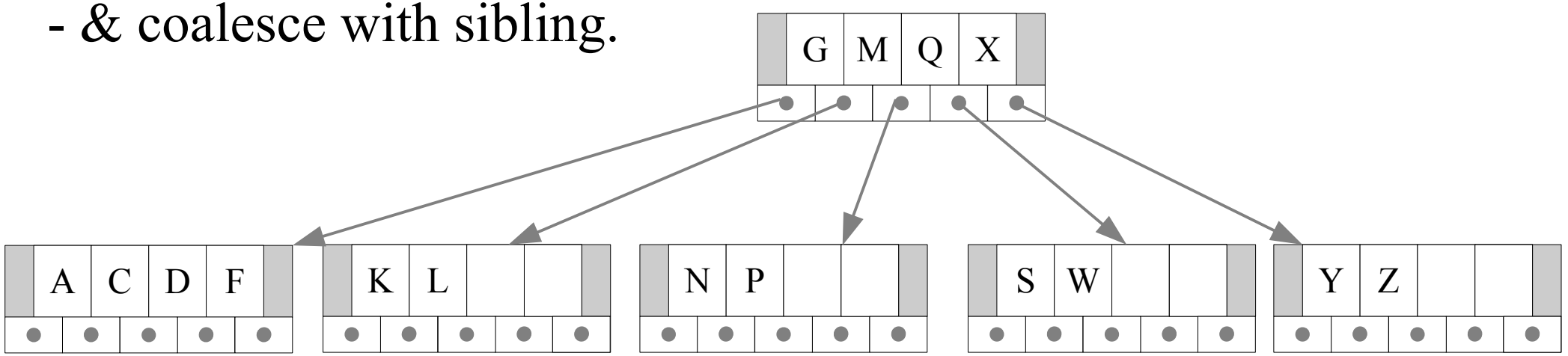
Onto Deletion

- Now G has too few keys.
- So move its parent down
- & coalesce with sibling.



Onto Deletion

- Now G has too few keys.
- So move its parent down
- & coalesce with sibling.



B+ Tree (Backward from stvincent's explanation)

- External nodes have pointers too.
- These are to data associated w/ key.
- Internal key's data in predecessor's rightmost rightmost empty val.