Chapter 9 B—trees

- Fast access to all members of the tree
- Linear time for insertion and deletion

Basic Properties of a B—tree of order *m*

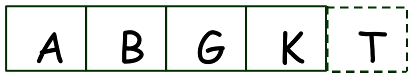
- Every page (node) has a maximum of *m* descendents
- Every page, except the root and the leaves, has a least $\lceil m/2 \rceil$ descendants
- The root has at least two descendants (unless it is a leaf)
- All leaves appear on the same level
- The leaf level forms a complete, ordered index of the associated data file

  Note: this last item means you can link the leaves and use them to get an ordered list records in the file.

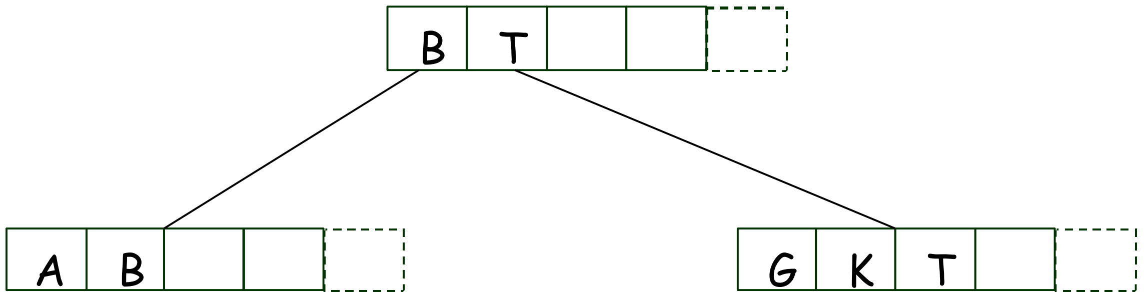
So a B—tree of order 200 has a minimum of 100 keys and a maximum of 200 keys per node, except for the root node which must have minimum of two keys

EXAMPLE: B-tree of order 4

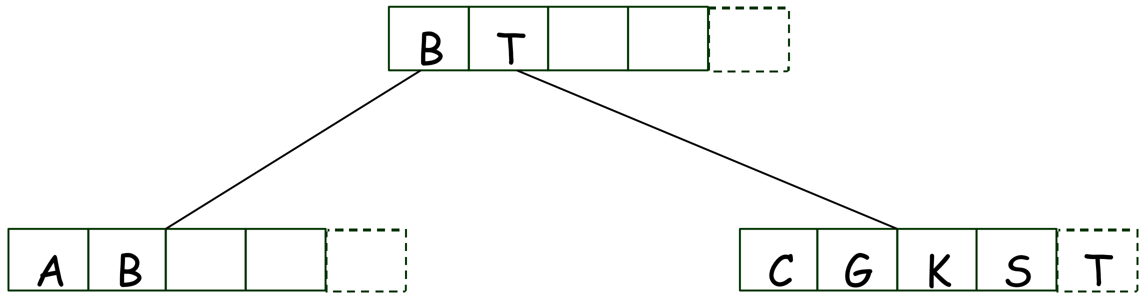
INPUT: A B T G K C S F E D Q P Y O Z L M I N R X W H J U V



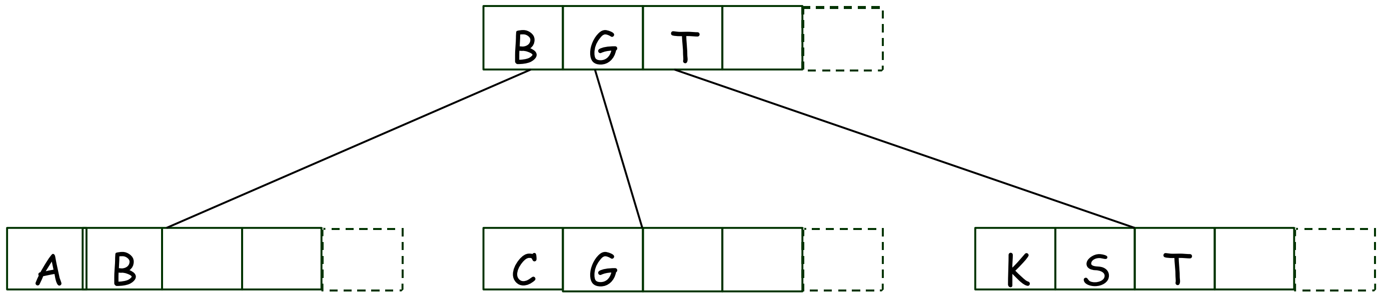
At this point we recognize that the node is overfull so we split the original node.



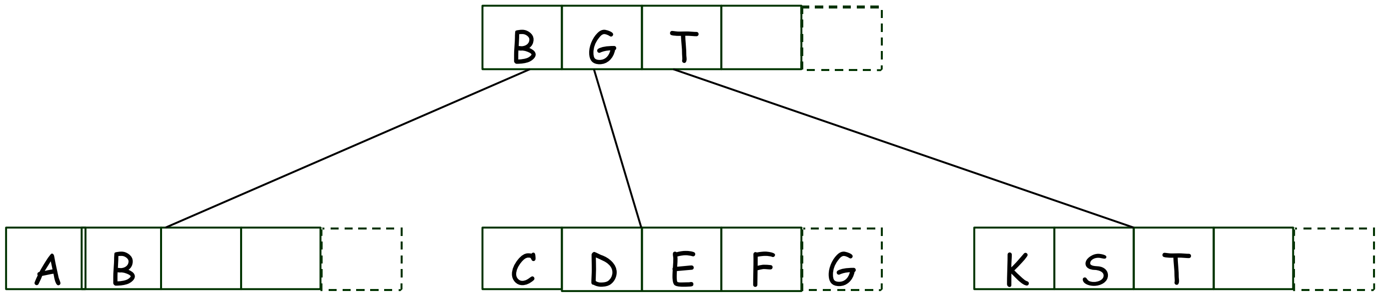
Remaining input: C S F E D Q P Y O Z L M I N R X W H J U V



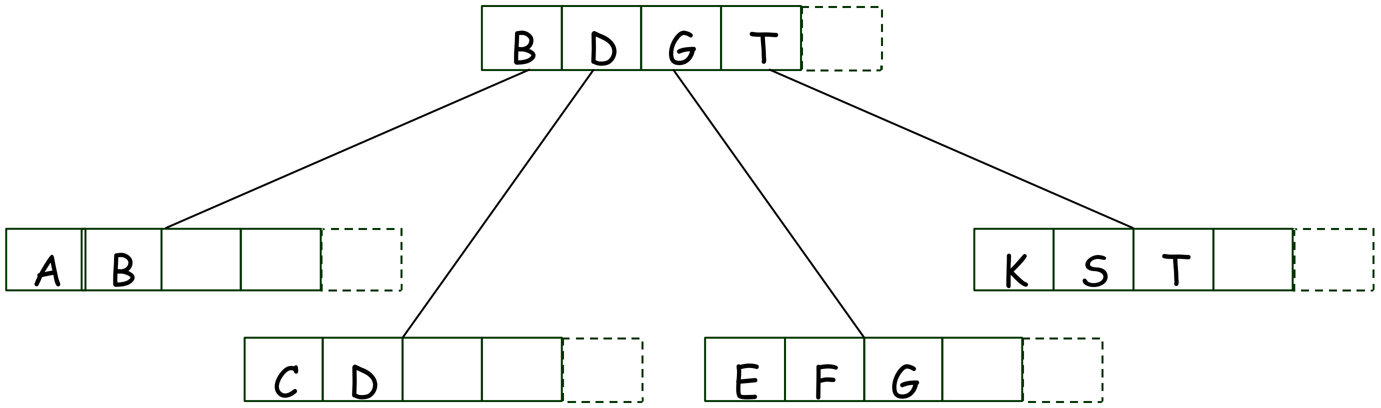
At this point we need to split the node with C G K S and T.



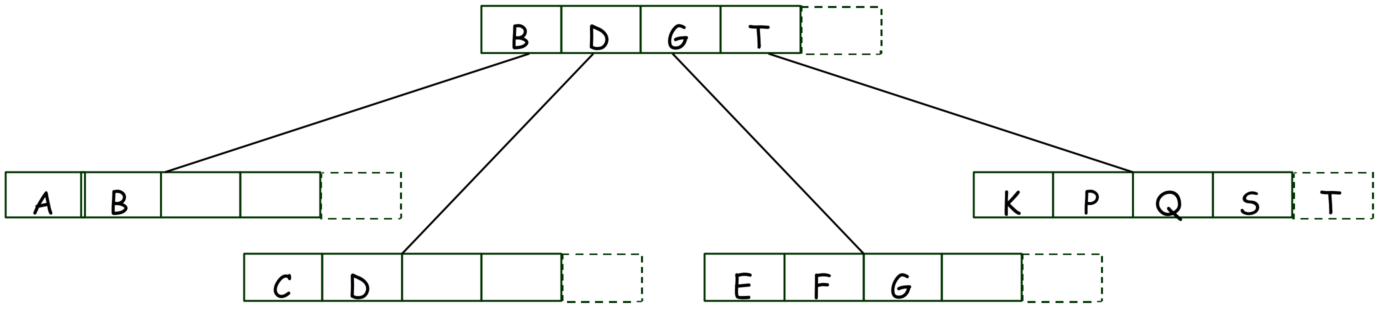
Remaining Input: F E D Q P Y O Z L M I N R X W H J U V



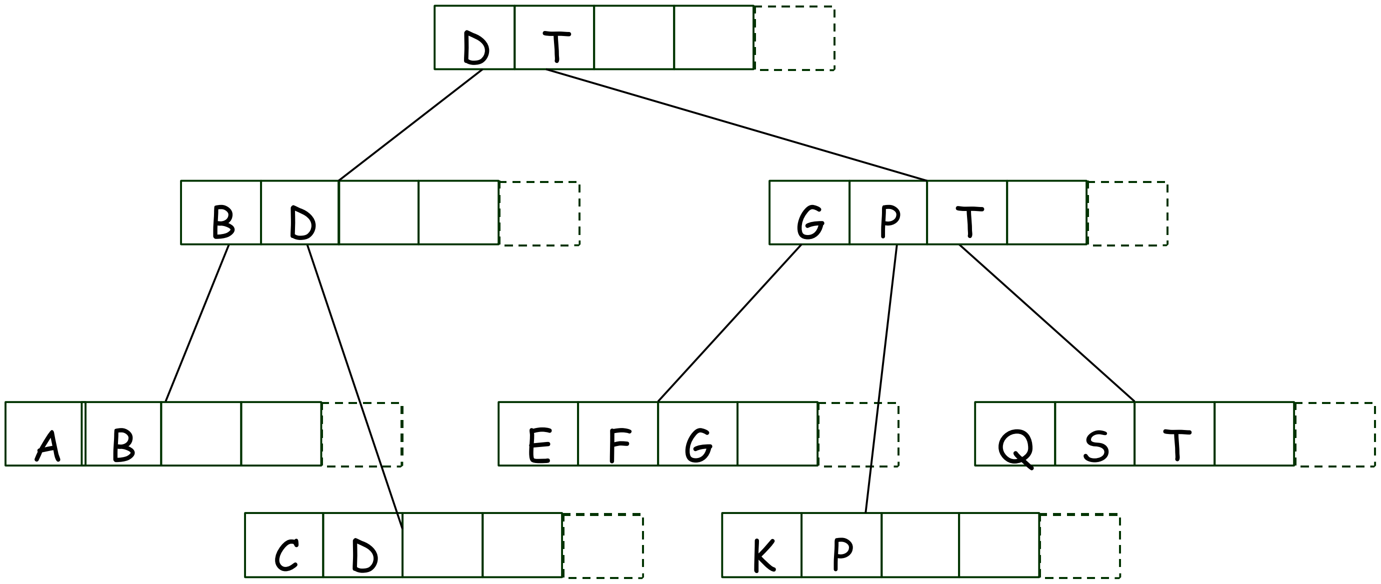
Now we have to split the node with C D E F G



Remaining Input: Q P Y O Z L M I N R X W H J U V

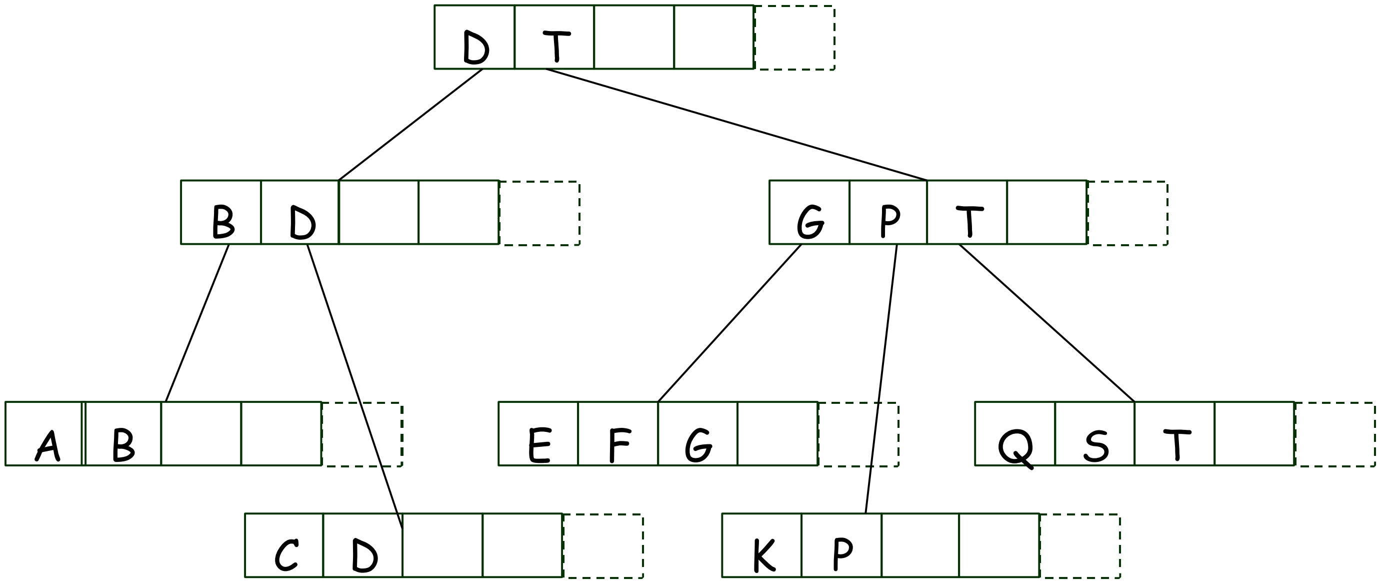


Now we need to split K P Q S T and the root node



Lets finish

Remaining Input: Y O Z L M I N R X W H J U V



- What happens when we add something greater than the largest element already in the tree?
- Notice that we actually have an extra space in the nodes to make splitting more elegant
- Methods
    - Search
    - Insert: split
    - Delete: merge
- Time Complexity

Searching

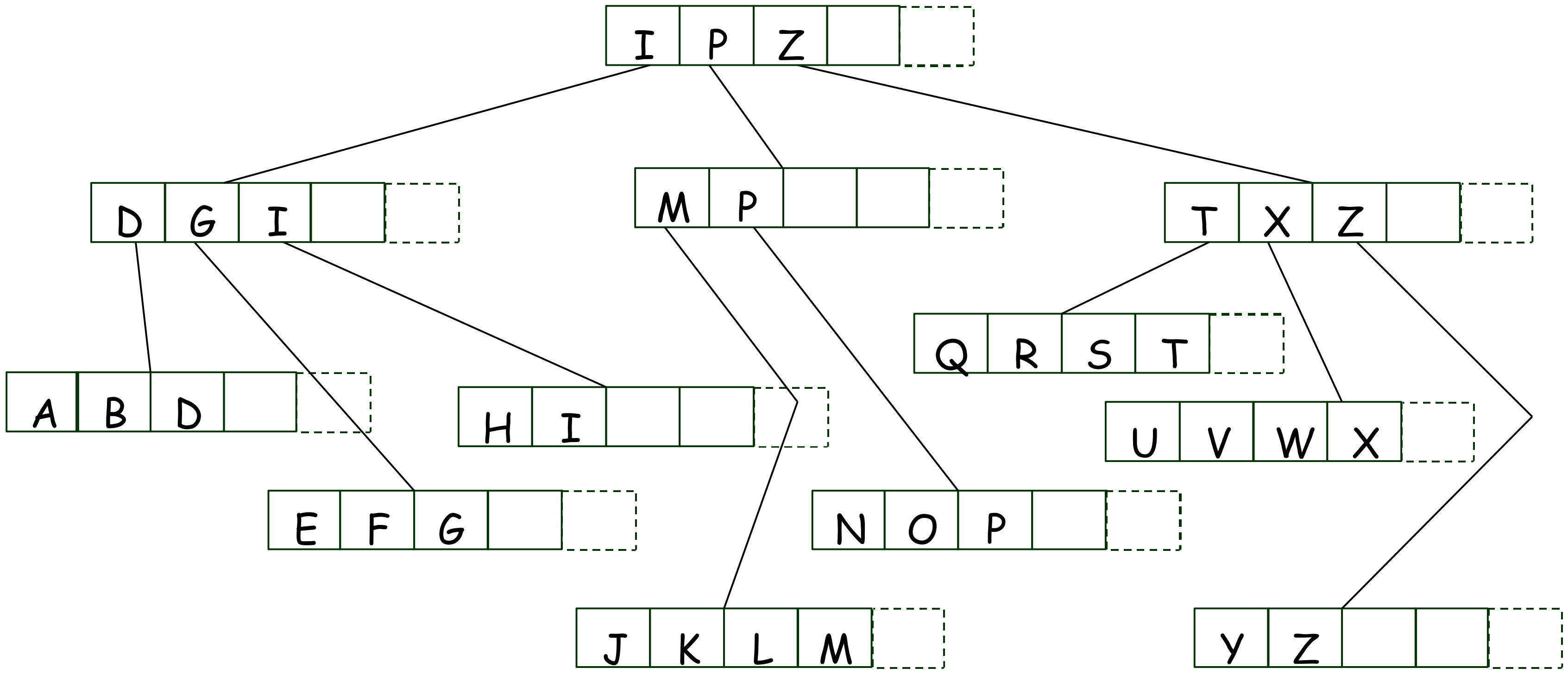
- Look for the entry in the node with key >= the key we are searching for
- Follow the link to the node.
    - The node will either be the record we are looking for or
    - Another interior node

Insertion

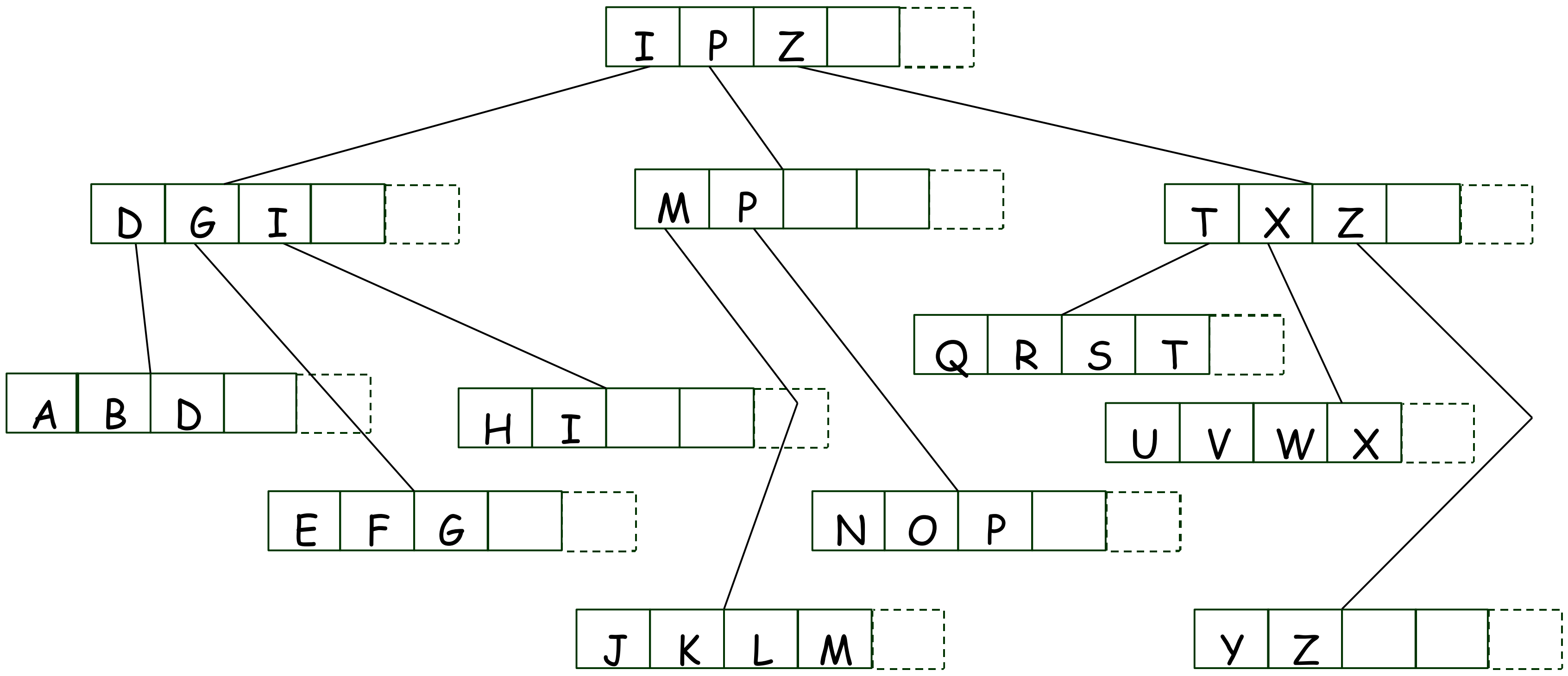
- Search to the leaf level for the correct space for the new node
- Insert, check for overflow, split on the way back up
- Create a new root node if necessary
    - note: checking for overflow is easier with the additional spot in the records
    - note: record 0 always points to the root node
    - note: special case for keys larger than the largest so far

Deletion

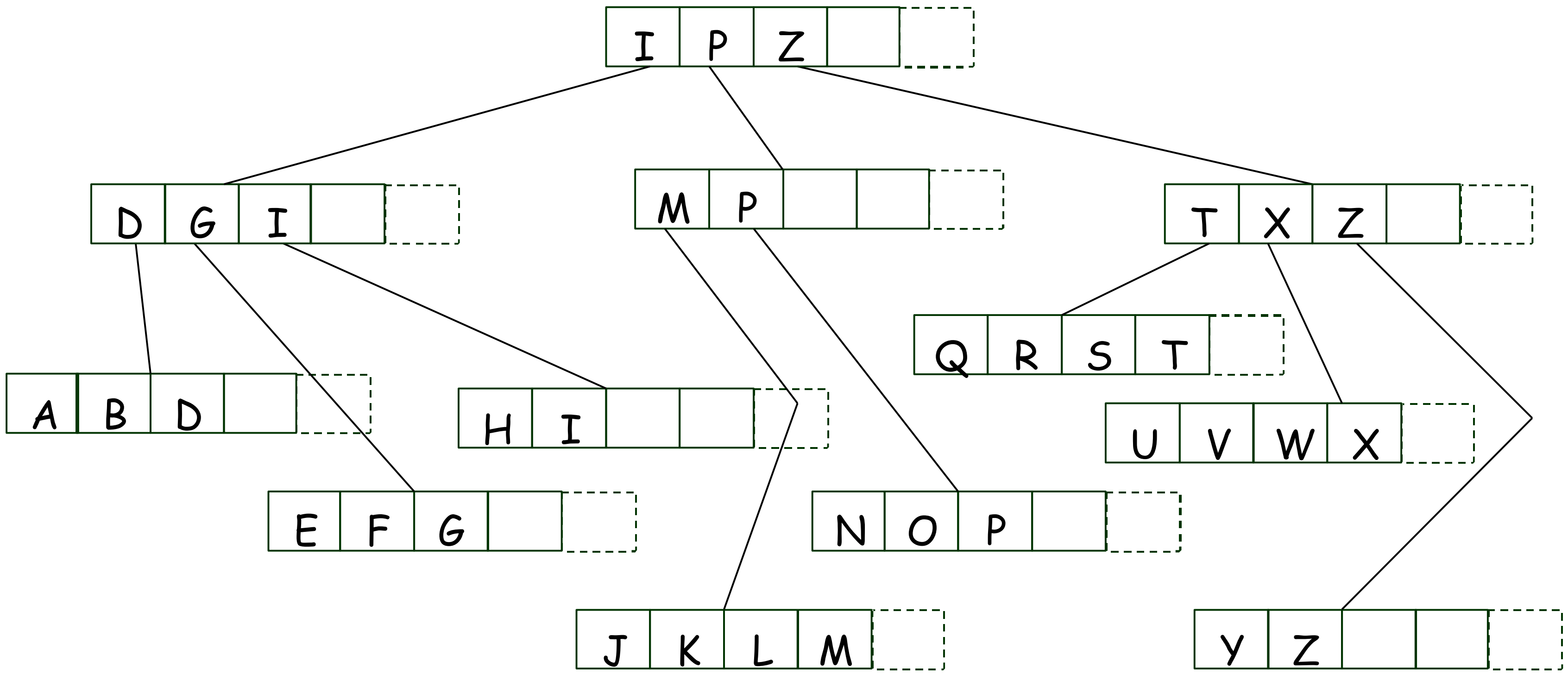
- Deletion requires that we maintain the B—tree specifications
  - Every page except for the root and the leaves has at least $\lceil m/2 \rceil$ descendants
  - A page contains at least $\lceil m/2 \rceil$ keys and no more than m keys



What happens when we delete P?



What happens when we delete H?



Rules for deleting key *k* from a node *n* in a B—tree

- If *n* has more than the minimum number of keys and the *k* is not the largest in *n,* simply delete *k* from *n.*
- If *n* has more than the minimum number of keys and the *k* if the largest in *n,* delete *k* and modify the higher level indexes to reflect the new largest key in *n.*
- If *n* has exactly the minimum number of keys and one of the siblings of *n* has few enough keys, merge *n* with its sibling and delete a key from the parent node.
- If *n* has exactly the minimum number of keys and one of the siblings of *n* has extra keys, redistribute by moving some keys from a sibling to *n,* and modify the higher level indexes to reflect the new largest keys in the affected nodes

The implementation of delete gets to decide which rule to use if more than one applies.

Redistribution and Insertion

Insertion can also use redistribution.

Look at the earlier tree

With redistribution the space utilization goes from approximately 66% to 85%

Worst Case Search Depth

Look at the following table of the number of descendents

Level	Minimum number of descendent s
1 (root)	2
2	$2 \times \lceil m/2 \rceil$
3	$2 \times \lceil m/2 \rceil \times \lceil m/2 \rceil$
4	$2 \times \lceil m/2 \rceil^3$
...	...
d	$2 \times \lceil m/2 \rceil^{d-1}$

If the tree has N keys it will have N keys in its leaves $N \geq 2 \times \left| \frac{m}{2} \right|^{d-1}$

Solving for d gives $d \leq 1 + \log_{\lceil m/2 \rceil}(N/2)$

So for a tree of order 512 that contains 1,000,000 keys we get

$d \leq 1 + \log_{\lceil 256 \rceil}(500{,}000)$ so $d \leq 3.37$

The tree will have three levels