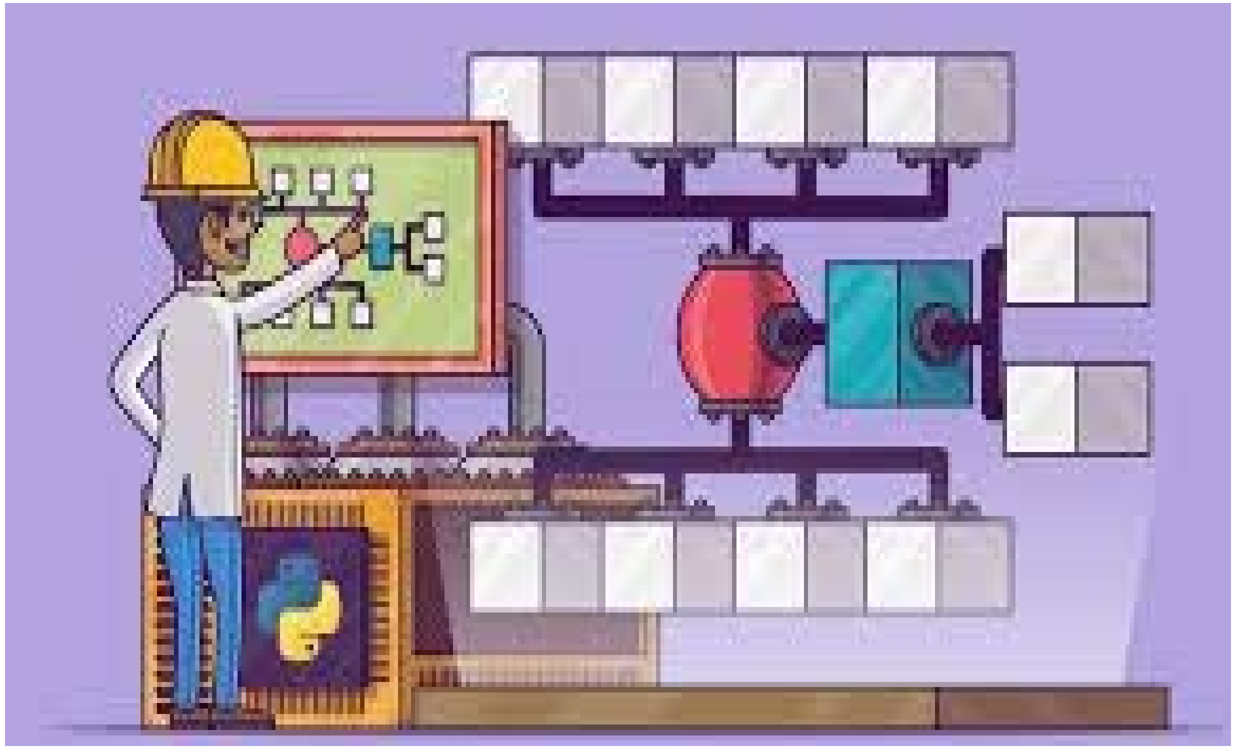
ساختمان داده ها



مدرس:
سمانه حسینی سمنانی

دانشگاه صنعتی اصفهان- دانشکده برق و کامپیوتر

درخت ها

- مفاهیم اولیه
- پیمایش درخت
- درخت دودویی معادل
- پیاده سازی درخت
- درخت جستجوی دودویی
- درخت عبارت
- Heap tree (هرم بیشینه)

درخت ها

- Red-black tree
- AVL tree
- B-Trees

Deleting a key from a B-tree

- Had to ensure that a node didn't get too big due to insertion
- Ensure that a node doesn't get too small during deletion
- Except that the root is allowed to have fewer than the minimum number t 1 of keys

Deleting a key from a B-tree

1. If the key k is in node x and x is a leaf, delete the key k from x.
2. If the key k is in node x and x is an internal node, do the following:

a. If the child y that precedes k in node x has at least t keys, then find the predecessor k' of k in the subtree rooted at y. Recursively delete k', and replace k by k' in x. (We can find k' and delete it in a single downward pass.)

Deleting a key from a B-tree

b. If y has fewer than t keys, then, symmetrically, examine the child z that follows k in node x. If z has at least t keys, then find the successor k' of k in the subtree rooted at z. Recursively delete k', and replace k by k' in x. (We can find k' and delete it in a single downward pass.)

c. Otherwise, if both y and z have only $t - 1$ keys, merge k and all of z into y, so that x loses both k and the pointer to z, and y now contains $2t - 1$ keys. Then free z and recursively delete k from y.

Deleting a key from a B-tree

3. If the key k is not present in internal node x, determine the root $x.c_i$ of the appropriate subtree that must contain k, if k is in the tree at all. If $x.c_i$ has only $t-1$ keys, execute step 3a or 3b as necessary to guarantee that we descend to a node containing at least t keys. Then finish by recursing on the appropriate child of x.

Deleting a key from a B-tree

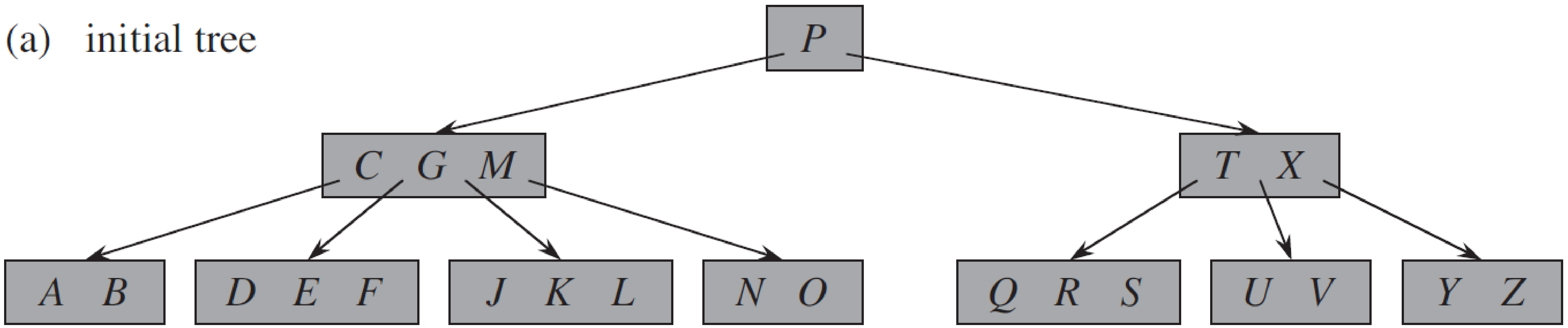
a. If $x.c_i$ has only $t-1$ keys but has an immediate sibling with at least t keys, give $x.c_i$ an extra key by moving a key from x down into $x.c_i$, moving a key from $x.c_i$'s immediate left or right sibling up into x, and moving the appropriate child pointer from the sibling into $x.c_i$.

b. If $x.c_i$ and both of $x.c_i$'s immediate siblings have $t-1$ keys, merge $x.c_i$ with one sibling, which involves moving a key from x down into the new merged node to become the median key for that node.

Deleting a key from a B-tree

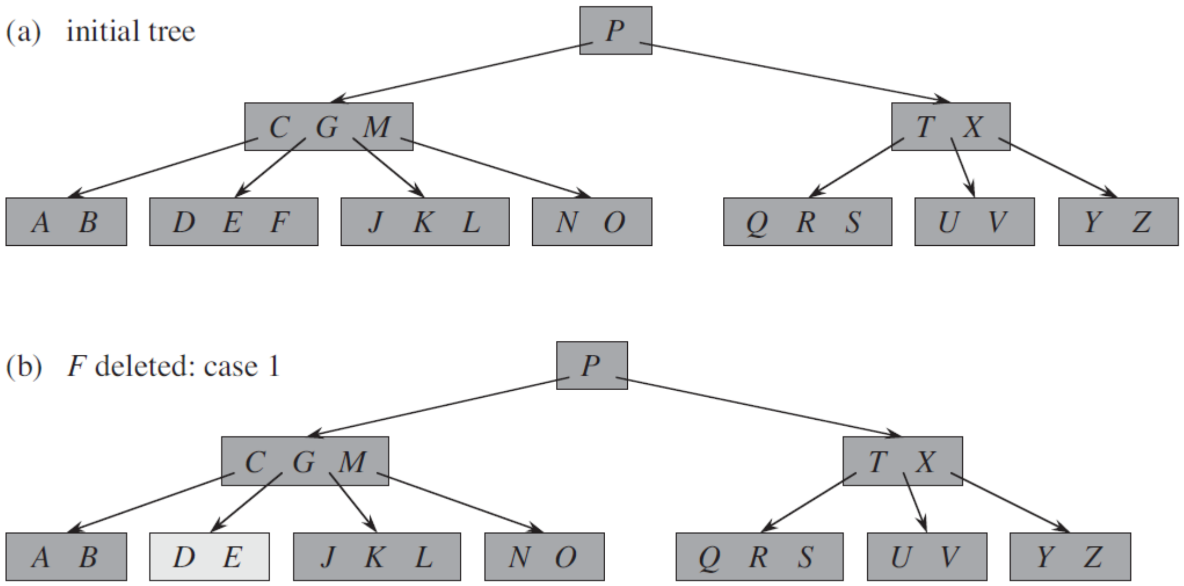
The minimum degree for this B-tree is t = 3, so a node (other than the root) cannot have fewer than 2 keys.

(a) initial tree



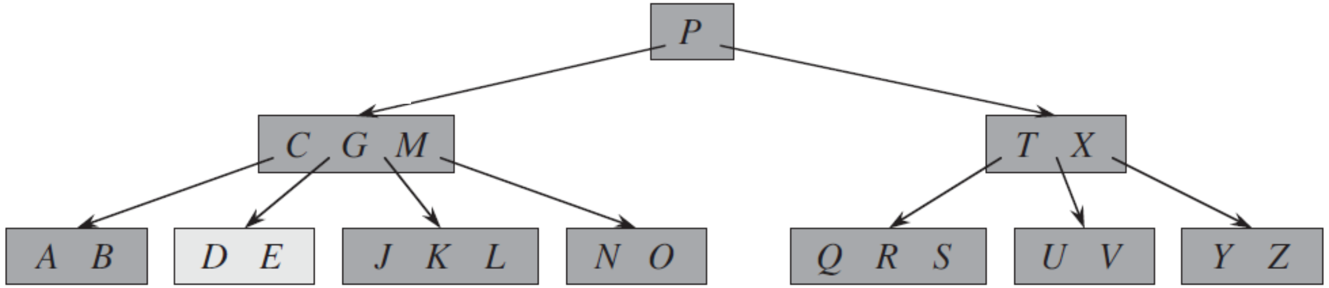
Deleting a key from a B-tree

The minimum degree for this B-tree is $t = 3$, so a node (other than the root) cannot have fewer than 2 keys.

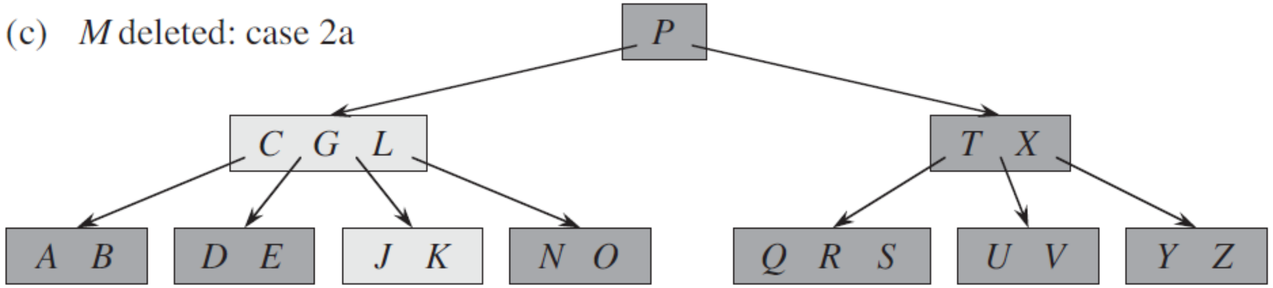


1. If the key k is in node x and x is a leaf, delete the key k from x.

Deleting a key from a B-tree



(c) *M* deleted: case 2a



2. If the key k is in node x and x is an internal node, do the following:

   a. If the child y that precedes k in node x has at least t keys, then find the predecessor k' of k in the subtree rooted at y. Recursively delete k', and replace k by k' in x. (We can find k' and delete it in a single downward pass.)

Deleting a key from a B-tree

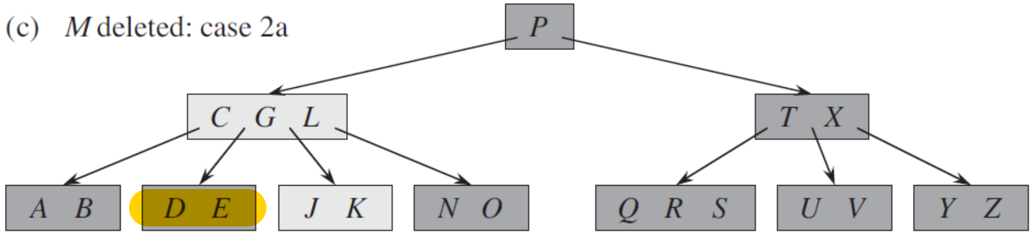
2. If the key k is in node x and x is an internal node, do the following:

a. If the child y that precedes k in node x has at least t keys, then find the predecessor k' of k in the subtree rooted at y. Recursively delete k', and replace k by k' in x. (We can find k' and delete it in a single downward pass.)

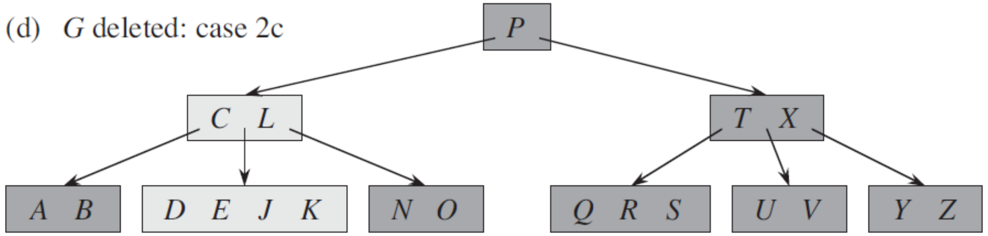
b. If y has fewer than t keys, then, symmetrically, examine the child z that follows k in node x. If z has at least t keys, then find the successor k' of k in the subtree rooted at z. Recursively delete k', and replace k by k' in x. (We can find k' and delete it in a single downward pass.)

c. Otherwise, if both y and z have only $t - 1$ keys, merge k and all of z into y, so that x loses both k and the pointer to z, and y now contains $2t - 1$ keys. Then free z and recursively delete k from y.

(c) *M* deleted: case 2a



(d) *G* deleted: case 2c

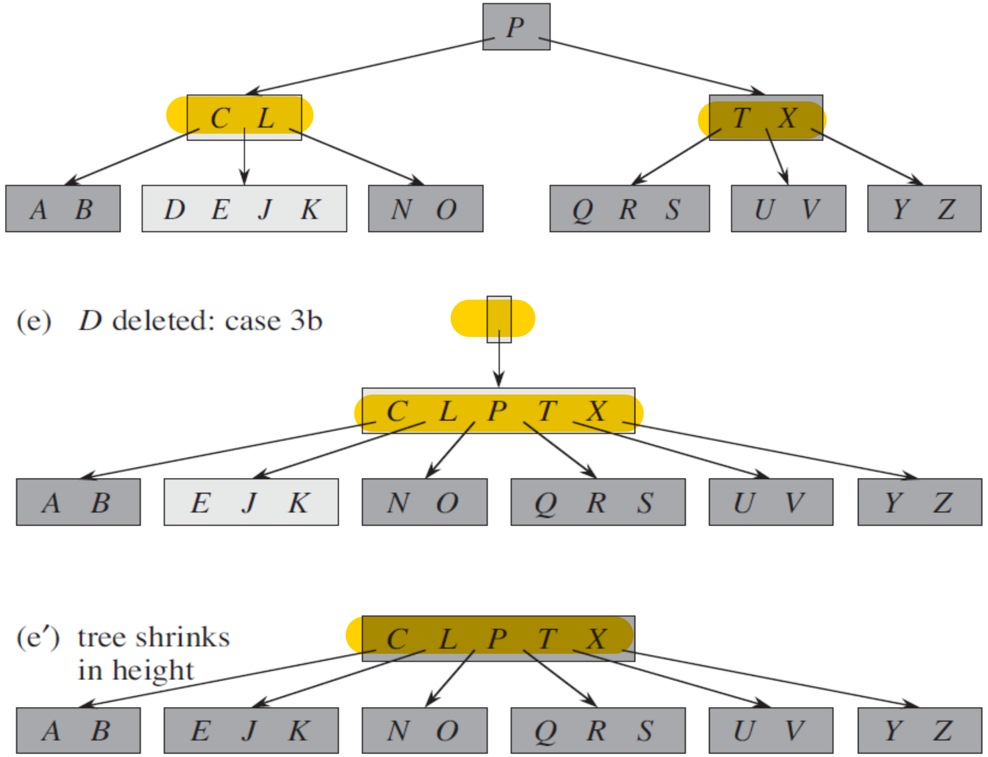


Deleting a key from a B-tree

3. If the key k is not present in internal node x, determine the root $x.c_i$ of the appropriate subtree that must contain k, if k is in the tree at all. If $x.c_i$ has only $t-1$ keys, execute step 3a or 3b as necessary to guarantee that we descend to a node containing at least t keys. Then finish by recursing on the appropriate child of x.

   a. If $x.c_i$ has only $t-1$ keys but has an immediate sibling with at least t keys, give $x.c_i$ an extra key by moving a key from x down into $x.c_i$, moving a key from $x.c_i$'s immediate left or right sibling up into x, and moving the appropriate child pointer from the sibling into $x.c_i$.

   b. If $x.c_i$ and both of $x.c_i$'s immediate siblings have $t-1$ keys, merge $x.c_i$ with one sibling, which involves moving a key from x down into the new merged node to become the median key for that node.



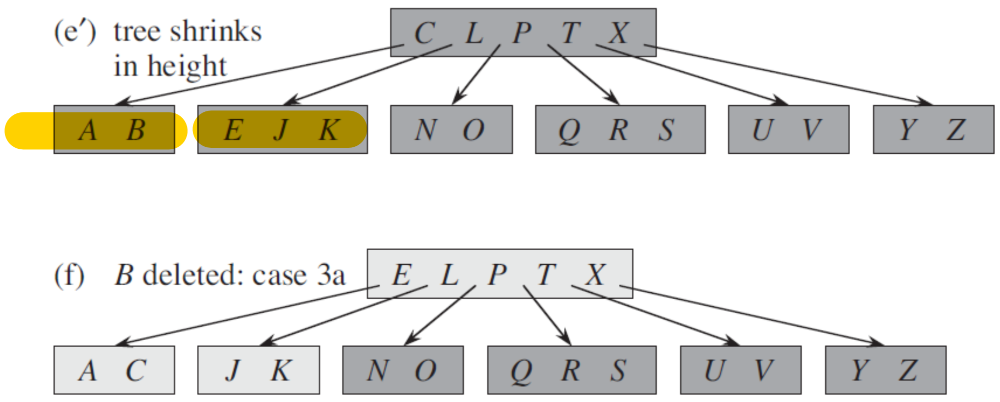
Although this procedure seems complicated, it involves only O(h) disk operations for a B-tree of height h

Deleting a key from a B-tree

3. If the key k is not present in internal node x, determine the root $x.c_i$ of the appropriate subtree that must contain k, if k is in the tree at all. If $x.c_i$ has only $t-1$ keys, execute step 3a or 3b as necessary to guarantee that we descend to a node containing at least t keys. Then finish by recursing on the appropriate child of x.

   a. If $x.c_i$ has only $t-1$ keys but has an immediate sibling with at least t keys, give $x.c_i$ an extra key by moving a key from x down into $x.c_i$, moving a key from $x.c_i$'s immediate left or right sibling up into x, and moving the appropriate child pointer from the sibling into $x.c_i$.

   b. If $x.c_i$ and both of $x.c_i$'s immediate siblings have $t-1$ keys, merge $x.c_i$ with one sibling, which involves moving a key from x down into the new merged node to become the median key for that node.



Although this procedure seems complicated, it involves only O(h) disk operations for a B-tree of height h