Solution to 8 (18.2-6)
If the search within a node is implement by binary search, then the time taken by the **while** loop of line 2-3 in the search algorithm presented on page 442 is $O(\lg t)$. Then the total CPU time is $O(t \cdot h) = O(\lg t \cdot \log_t n) = O(\lg n)$. □

Solution to 9 (18-2)

1. At each node of the tree, keep a field $height[x]$, such that $height[x] = 0$ if x is a leave, otherwise $hight[x]$ = the length of a path from x to a leave. With a few changes, the field is maintain within the procedure of splitting or merging. Therefore, the asymptotic running time of any operation is the same.

2. If $h' > h''$, then insert k into the rightmost node N' at height $h'' + 1$ of T'. If N' is full, extract its largest key k', fill k into its space, and recursively insert k' into the parent of N'. Finally, attach the tree T'' to the right of k. If $h' = h''$, create a new root with k, and attach T' and T'' to the left and right of k. If $h' < h''$, insert k into the leftmost node at height $h' + 1$ of T''. The procedure is the similar to the first case. Since the procedure has at most $|h' - h''| + 1$ recursions, each takes $O(1)$ time. The total time for joining is $O(|h' - h''| + 1)$.

3. Suppose the path p has length l, then $P = \{c_{a_1}[x_1], \ldots, c_{a_l}[x_l], k\}$, where $a_i \in \{1, 2, 3, 4\}$. For $i = 1, \ldots, l$, do:

   (a) if $c_i = 1$, then $k_i = NIL$ and $T'_i = \emptyset$;

   (b) if $c_i = 2$, then $k_i = key_1[x_i]$ and $T'_i = c_1[x_i]$;

   (c) if $c_i = 3$, then $k_i = key_2[x_i]$ and T'_i = a tree formed by a root containing $key_1[x_i]$ and two subtrees $c_1[x_i]$ and $c_2[x_i]$;

   (d) if $c_i = 4$, then $k_i = key_3[x_i]$ and T'_i = a tree formed by a root containing $key_1[x_i]$ and $key_2[x_i]$, and three subtrees $c_1[x_i]$, $c_2[x_i]$, and $c_3[x_i]$.

   Finally consider the node x' containing k. If $k = key_1[x']$, then $T'_{l+1} = c_1[x']$; If $k = key_2[x']$, then T'_{l+1} = a tree formed by a root containing $key_1[x']$ and two subtrees $c_1[x']$ and $c_2[x']$; If $k = key_3[x']$, then T'_{l+1} = a tree formed by a root containing $key_1[x']$ and $key_2[x']$, and three subtrees $c_1[x']$, $c_2[x']$ and $c_3[x']$. After tress $T_1, \ldots, T'_{l+1}$ and keys $k'_1, \ldots, k'_l$ are generated, remove all $T'_i = \emptyset$ and $k_i = NIL$. The result is a set of trees $\{T'_0, \ldots, T'_m\}$ and a set of keys $\{k'_1, \ldots, k'_m\}$. By the properties of a 2-3-4 tree, these are the desired sets. For $i = 1, \ldots, m$, the height of T'_{i-1} is greater or equal to the height of T'_i. The procedure to break S'' into sets of trees and keys is symmetric.

4. To split a tree, first find a path from the root to the key $k = key[x]$. Then break the set S' into sets of trees $\{T'_0, \ldots, T'_m\}$ and keys $k'_1, \ldots, k'_l$ as described above. Next for i from m to 1, do $join(T'_i, k_i, T'_{i-1})$. The result is a 2-3-4 tree containing the set S'. Do the same for S'' and create another tree for S''. Finally, return S' and S''. To analyze the running time, observe that the time to join two trees of hight h' and h'' is $O(1 + |h' - h''|)$ by (b). Then the total time to join all the trees in S' is

$$\sum_{i=1}^{m}(1 + |h(T'_{i-1}) - h(T'_i)|) = m + \sum_{i=1}^{m}[h(T'_{i-1}) - h(T'_i] = m + h(T'_0) - h(T'_m) \in O(\lg n),$$

   since $m, h(T'_0) \leq h(T) \in O(\lg n)$, $h(T'_m) \geq 0$. Since the time for finding the path, breaking S' and S'' is also bounded by $O(\lg n)$, the running time for splitting is $O(\lg n)$.

□

Note: The solutions given here are terse, and in some cases, incomplete. Your answers should be complete and have more details. But you will lose points if they are too long or too complex.