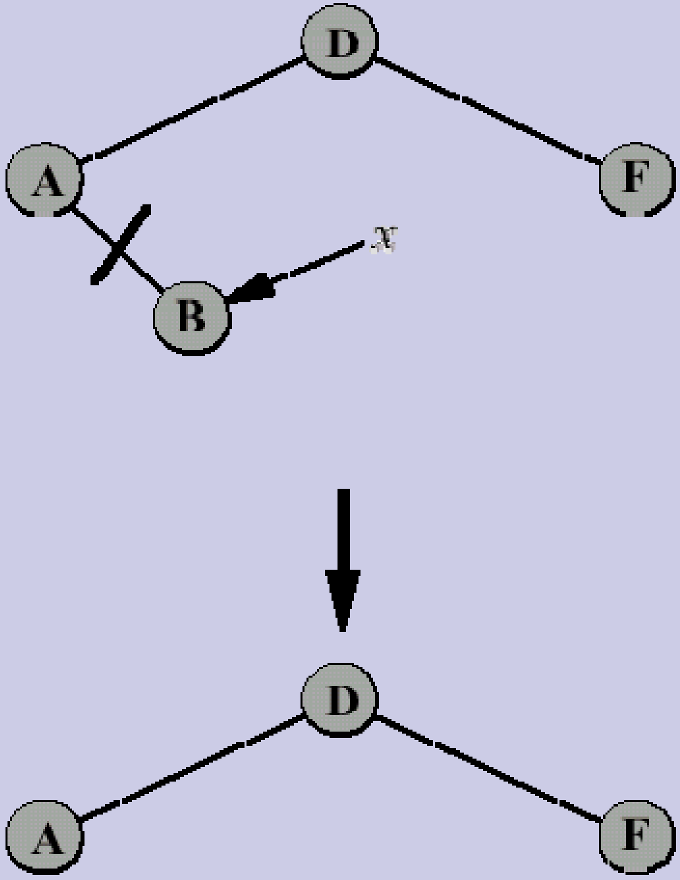
Deletion

- Delete node *x* from a tree *T*
- We can distinguish three cases
  - *x* has no children
  - *x* has one child
  - *x* has two children

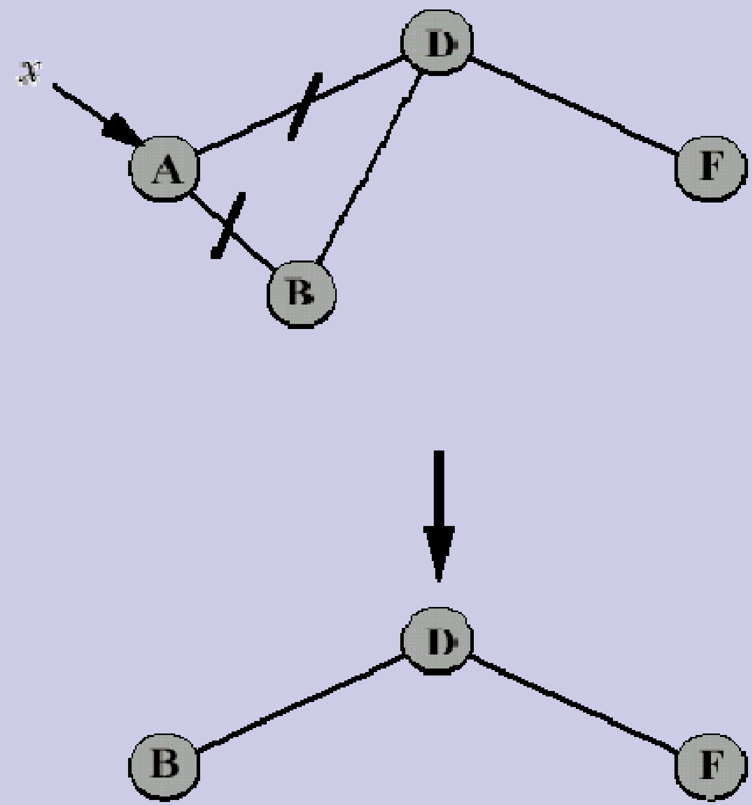
Deletion Case 1

- If *x* has no children – just remove *x*



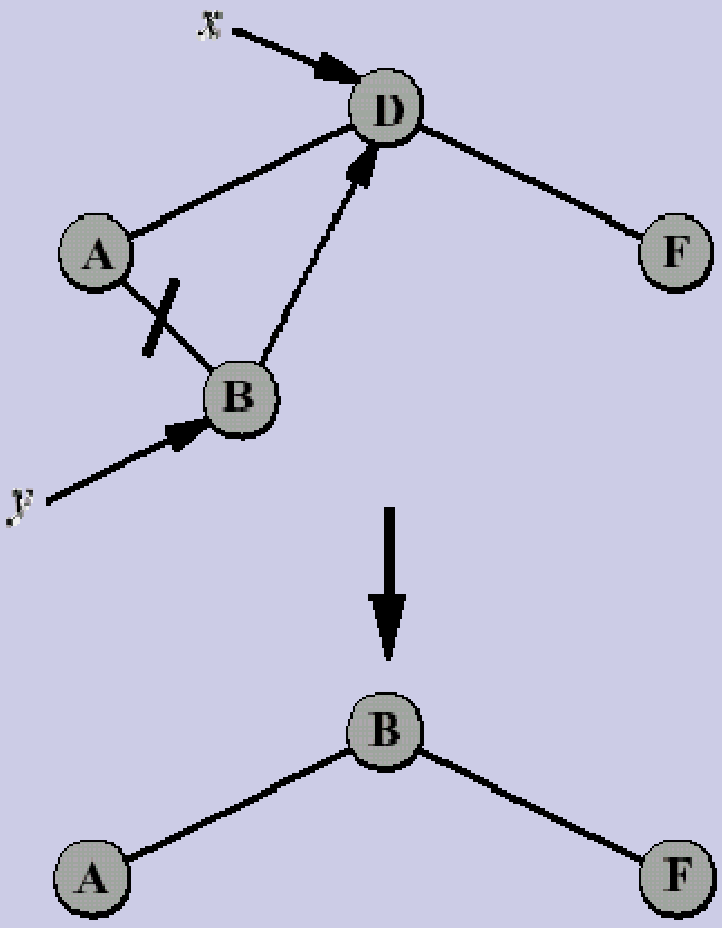
Deletion Case 2

- If *x* has exactly one child, then to delete *x*, simply make p[x] point to that child



Deletion Case 3

- If *x* has two children, then to delete it we have to
  - find its successor (or predecessor) *y*
  - remove y (note that y has at most one child – why?)
  - replace x with y

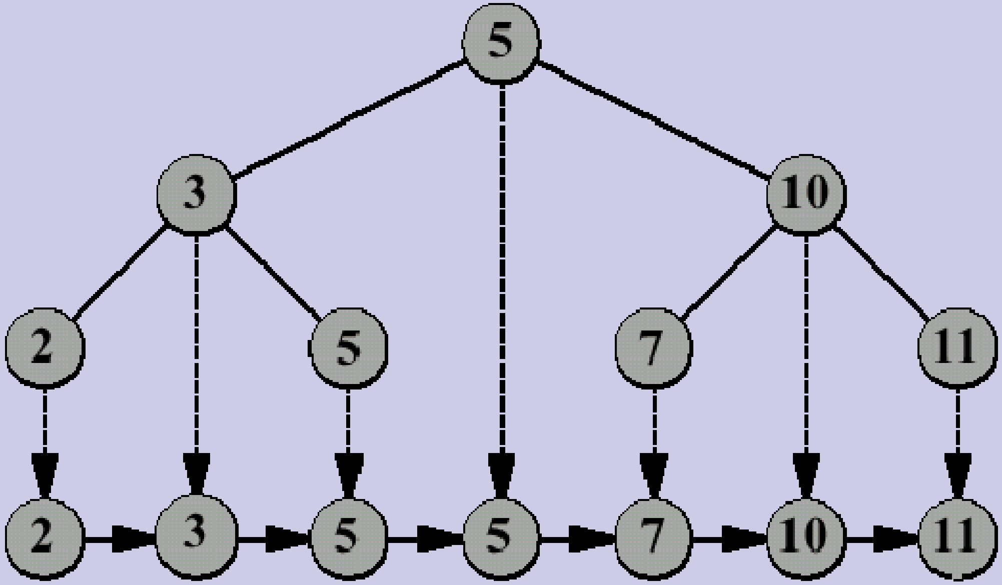


Delete Pseudocode

```
TreeDelete(T,z)
if left[z] = NIL or right[z] = NIL
  then y ← z
  else y ← TreeSuccessor(z)
if left[y] ≠ NIL
  then x ← left[y]
  else x ← right[y]
if x ≠ NIL
  then p[x] ← p[y]
if p[y] = NIL
  then root[T] ← x
  else if y = left[p[y]]
    then left[p[y]] ← x
    else right[p[y]] ← x
if y ≠ z
  then key[z] ← key[y] //copy all fileds of y
return y
```

In order traversal of a BST

- ITW can be thought of as a projection of the BST nodes onto a one dimensional interval



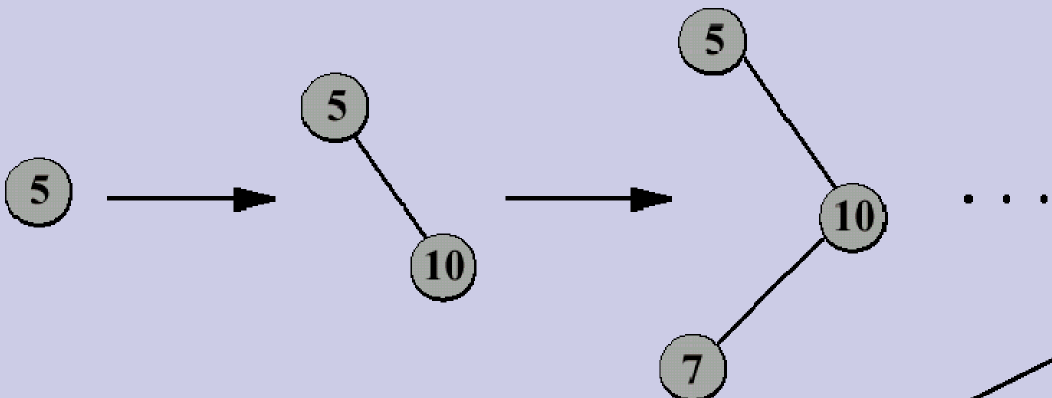
BST Sorting

- Use TreeInsert and InorderTreeWalk to sort a list of *n* elements, *A*

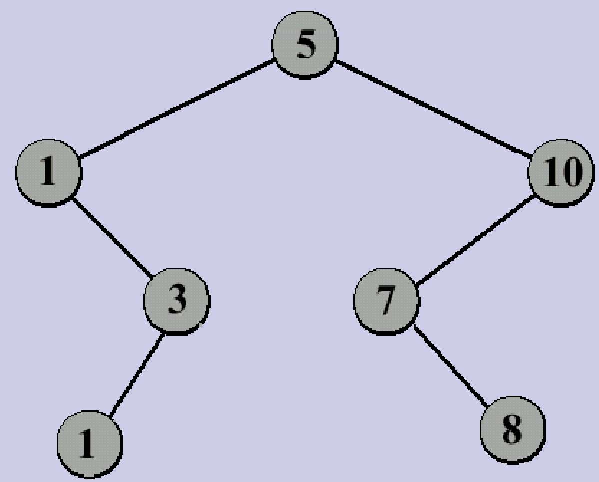
```
TreeSort(A)
root[T] ← NIL
for i ← 1 to n
  TreeInsert(T,A[i])
InorderTreeWalk(root[T])
```

BST Sorting (2)

- Sort the following numbers
  5 10 7 1 3 1 8
- Build a binary search tree



- Call InorderTreeWalk



1 1 3 5 7 8 10

In what order should we insert?

- We want to sort numbers {1,2,…,n}
- Total time taken to insert these numbers equals the sum of the level numbers of the nodes.
- Thus if numbers were inserted in ascending order we would get a tree of height n-1 in which there is one node at each level.
- So total time for insertion in this case is $1+2+3+\ldots+n-1 = O(n^2)$.

Inserting a random permutation

- Suppose we take a random permutation of the keys and inserted them in this order.
- The total time required for insertion is now a random variable.
- We want to compute the expected value of this r.v.
- Recall that the expected value of a r.v. is the average value it takes over a large number of trials.

Expected insertion time for a random permutation

- We will compute the average time taken to insert keys in the order specified by the n! permutations.
- In other words, for each of the n! permutations we will compute the time taken to insert keys in that order and then compute the average.
- Let T(n) denote this quantity.

Inserting a random permutation (2)

- Of the n! permutations, there are (n-1)! permutations in which the first element is i.
- The tree formed in these instances has i as the root. The left subtree has keys 1..(i-1) and the right subtree has keys (i+1)..n
- Consider the ordering of keys 1..(i-1) in the (n-1)! permutations. All (i-1)! permutations appear and each occurs (n-1)!/(i-1)! times.

Inserting a random permuation(3)

- Recall that if we only had keys 1..(i-1) then average time taken to insert them is T(i-1).
- The average is taken over all permutations of 1..(i-1).
- hence total time to insert all (i-1)! permutations is (i-1)!T(i-1).

Inserting a random permutation(4)

- When inserting keys 1..(i-1) into the left subtree, each key has to be compared with the root.
- This leads to an additional unit cost for each key.
- So total time to insert all (i-1)! permutations is (i-1)!(T(i-1)+(i-1)).
- Since each permutation appears (n-1)!/(i-1)! times, total time to insert keys 1..(i-1) is (n-1)!(T(i-1)+(i-1))

Inserting a random permutation(5)

- Time to insert keys 1..(i-1) is (n-1)!(T(i-1)+(i-1))
- Similarly, time to insert keys (i+1)..n is (n-1)!(T(n-i)+(n-i))
- Total time to insert all n keys in permutations where the first key is i is (n-1)! (T(i-1)+T(n-i)+ n-1)
- Total time to insert all n keys in all n! permutations is
  $(n-1)! \sum_{i=1}^{n} (T(i-1)+T(n-i)+ n-1)$.

Building the recurrence

Average time to insert n keys is

$$\begin{aligned} T(n) &= \frac{1}{n}\sum_{i=1}^{n}(T(i-1)+T(n-i)+n-1) \\ &= \frac{2}{n}\sum_{i=0}^{n-1}T(i)+n-1 \end{aligned}$$

Note that T(0)=0

- We are expressing the value of function T() at point n in terms of the value of T() at points 0..n-1. This is called a recurrence relation.

Solving the recurrence

Since

$$T(n-1) = \frac{2}{n-1}\sum_{i=0}^{n-2} T(i) + n - 2$$

we have

$$\frac{2}{n}\sum_{i=0}^{n-2} T(i) = \frac{n-1}{n}(T(n-1) - n + 2)$$

Substituting in the expression for T(n) we get

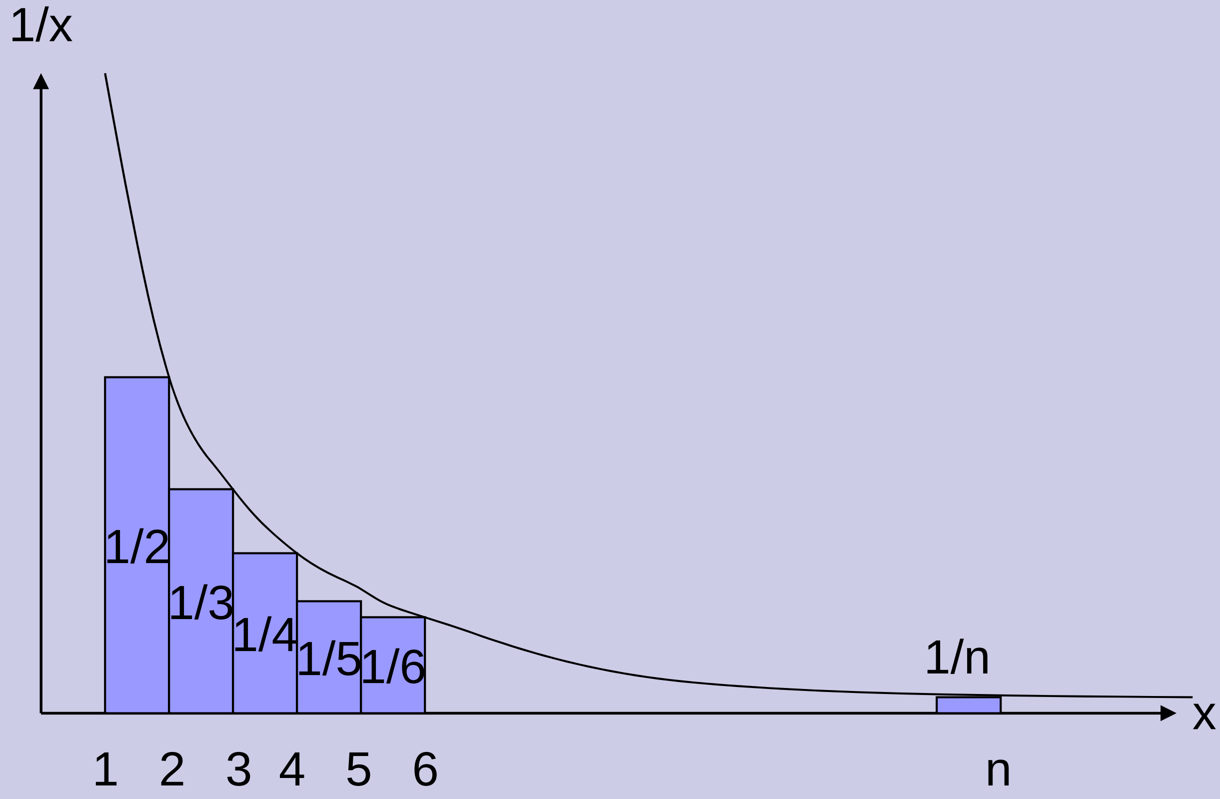
$$\begin{aligned} T(n) &= \frac{n-1}{n}(T(n-1) - n + 2) + \frac{2}{n}T(n-1) + n - 1 \\ &= \frac{n+1}{n}T(n-1) + \frac{2(n-1)}{n} \end{aligned}$$

Solving the recurrence(2)

$$
\begin{aligned}
T(n) &= \frac{n+1}{n}T(n-1) + 2(n-1)/n \\
&\le (1+1/n)T(n-1) + 2 \\
&\le \frac{n+1}{n}\left(\frac{n}{n-1}T(n-2)+2\right)+2 \\
&= \frac{n+1}{n-1}T(n-2) + \frac{2(n+1)}{n} + 2 \\
&\le \frac{n+1}{n-2}T(n-3) + 2(n+1)\left(\frac{1}{n}+\frac{1}{n-1}\right)+2 \\
&\le \frac{n+1}{n-3}T(n-4) + 2(n+1)\left(\frac{1}{n}+\frac{1}{n-1}+\frac{1}{n-2}\right)+2 \\
&\le (n+1)T(0) + 2(n+1)\left(\frac{1}{n}+\frac{1}{n-1}+\frac{1}{n-2}+\cdots+\frac{1}{2}\right)+2 \\
&= 2(n+1)\left(\frac{1}{n}+\frac{1}{n-1}+\frac{1}{n-2}+\cdots+\frac{1}{2}\right)+2
\end{aligned}
$$

Summing the harmonic series

- What is the sum ½+ 1/3 + ¼+…1/n?



It is at most the area under the curve f(x)=1/x between limits 1 and n

The expected time for insertion

$$\begin{aligned} T(n) &\leq 2(n+1)\int_1^n \frac{dx}{x} + 2 \\ &= 2(n+1)\ln n + 2 \\ &= O(n \log n) \end{aligned}$$

Thus the expected time for inserting a randomly chosen permutation of n keys is O(n log n)

Minimum time to insert n keys

- The time required to insert n keys is minimum when the resulting tree has the smallest possible height.
- A binary tree on n nodes has height at least $\log_2 n$
- To insert the n/2 nodes at level $\log_2 n$ we require at least $(n/2)\log_2 n$ time.
- Hence inserting n keys requires $\Omega(n\log_2 n)$ time.

Summary of Running times

- To insert n keys into a binary search tree which is initially empty requires
- $O(n^2)$ time in the worst case.
- $O(n\log n)$ time in the best case.
- $O(n\log n)$ time in the average case; the average is taken over the n! different orders in which the n keys could be inserted.