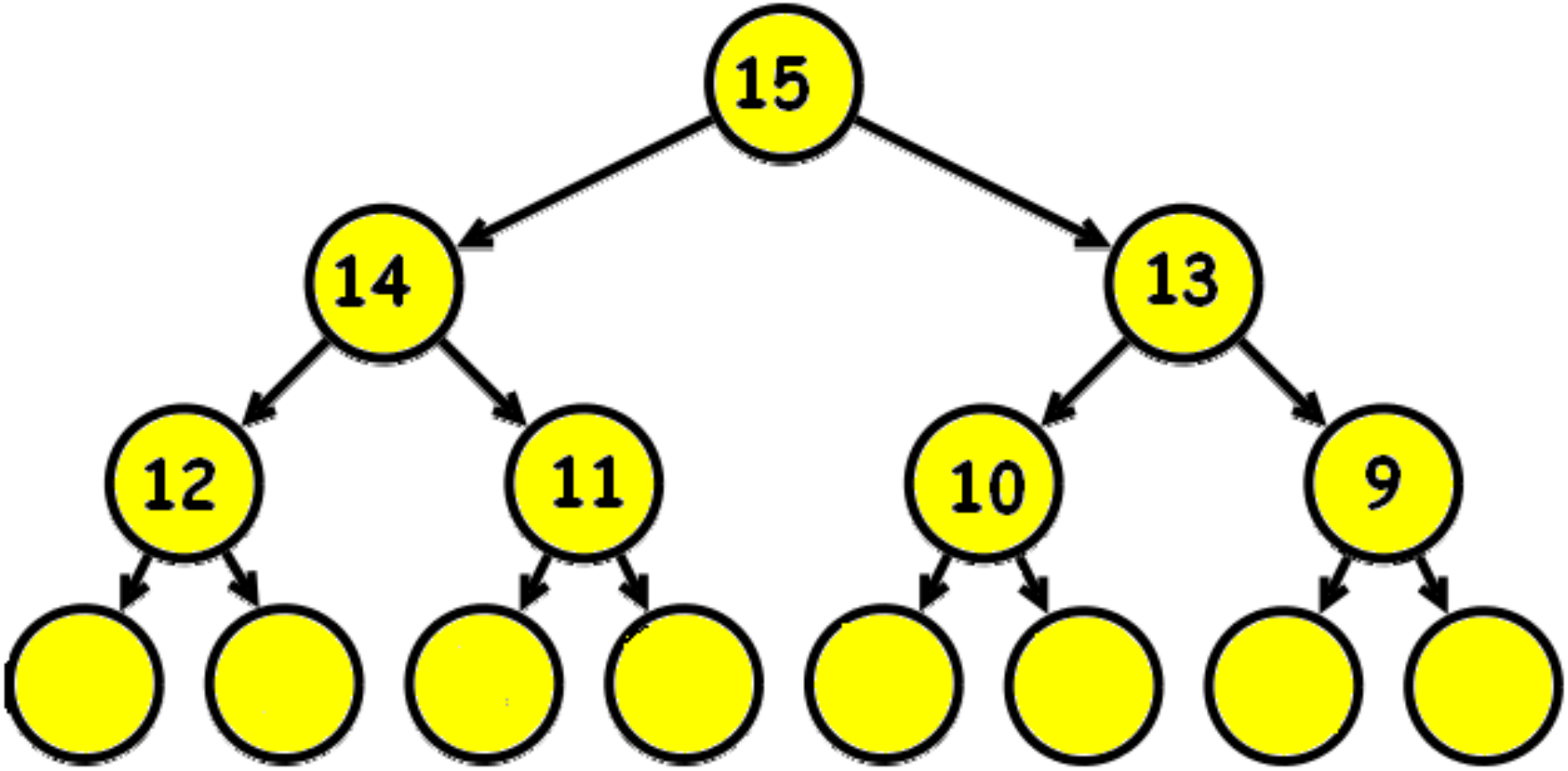
If you have a max-heap where the values in the heap are distinct, and you want to find the smallest value in the heap:

1. Where can it be in the heap?

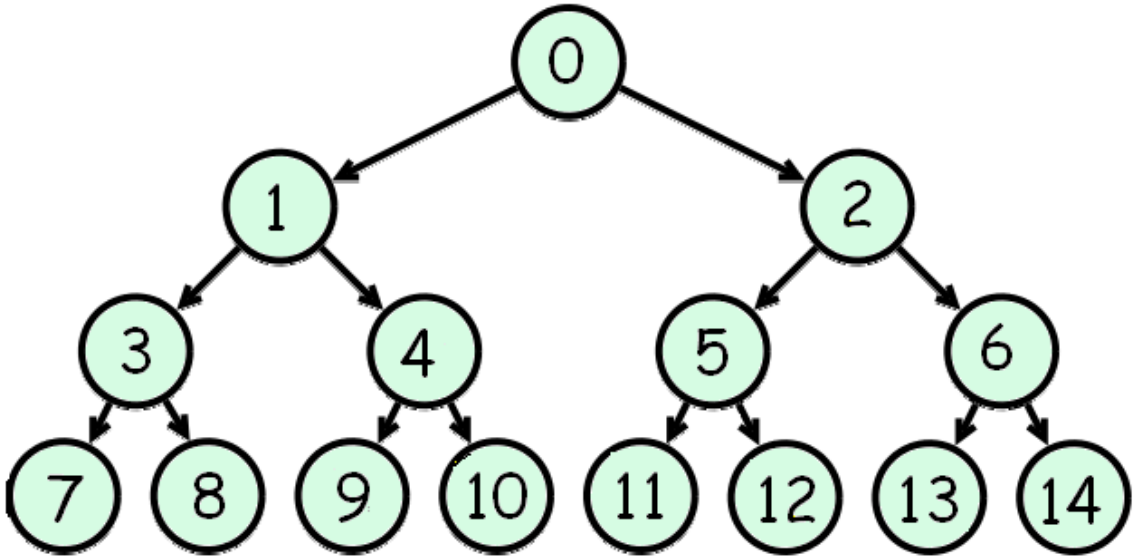
2. How much time does it take to find it?

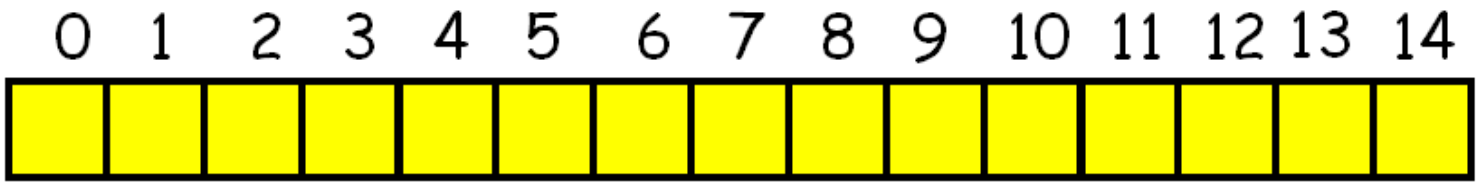
3. What would you do if you wanted to delete it?

Where could you put values 1-8?



Array positions of heap elements:





Drop deadline: Tuesday Oct. 31

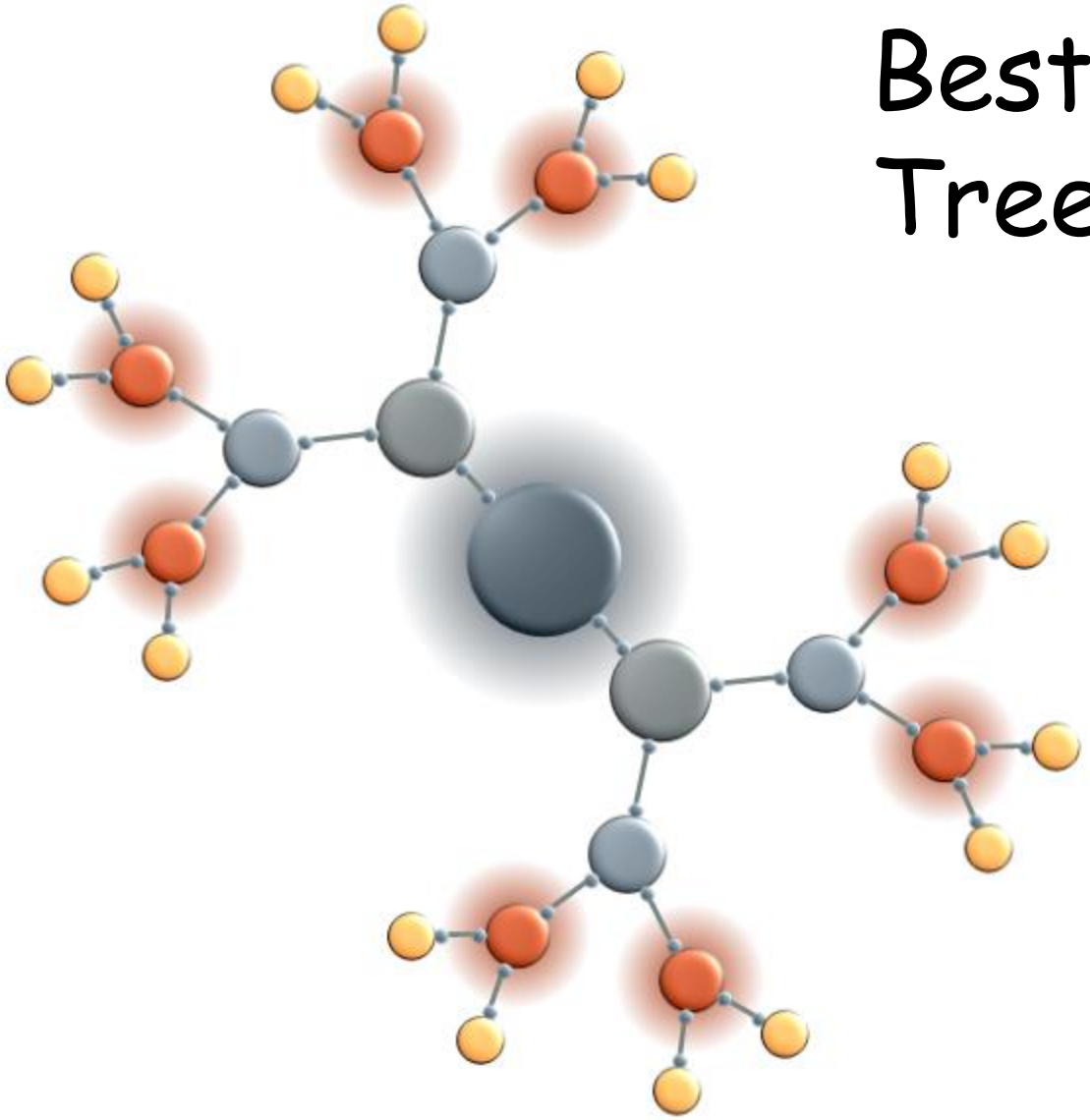
Recall that you need a passing assignment average to write the final exam.

You can miss at most 6 classes.

Do not sign the attendance sheet if you do not plan to attend the class.

You should be doing your programming assignments by yourself: not by copying the file of another student and changing some variable names.

Best Case Binary Tree Sort



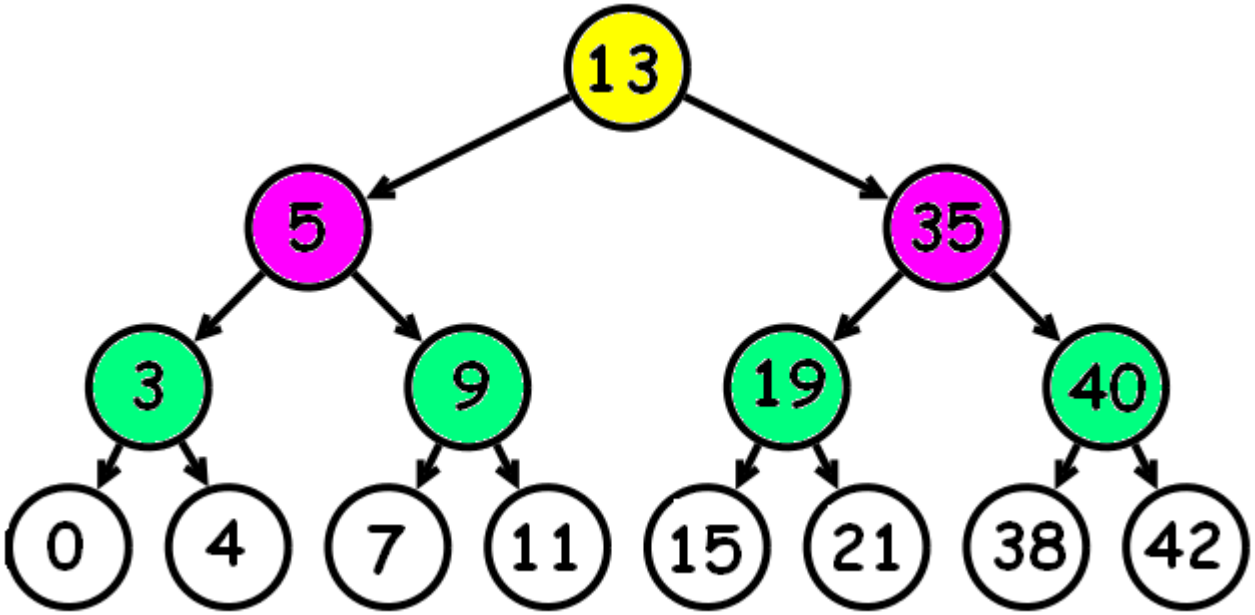
http://www.nevron.com/Gallery.DiagramFor.NET.Symmetrical.aspx

Binary search tree:

If a node has key value x then

-all keys in the left subtree have a data value which is less than or equal to x, and

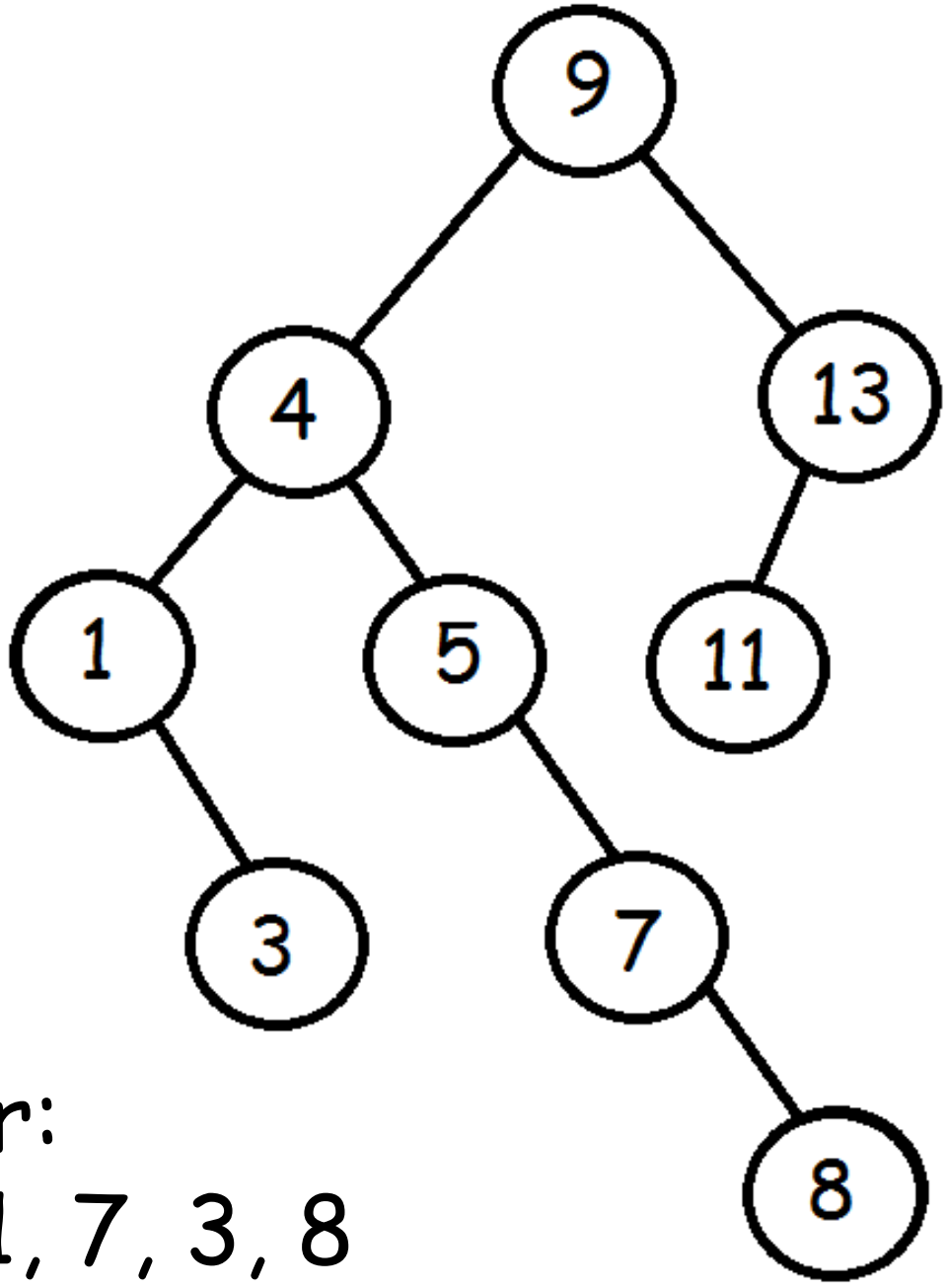
- all keys in the right subtree have a data value which is greater than x.



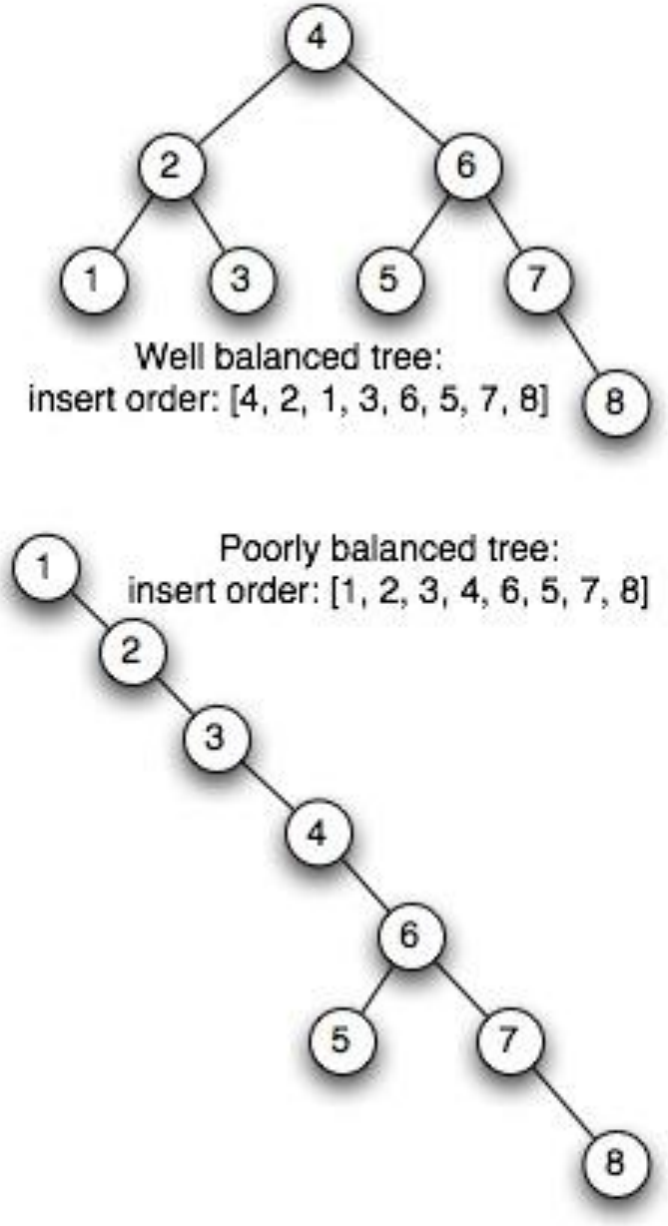
Insert into a Binary Search Tree:

9, 13, 11, 4, 5, 1, 7, 3, 8

If items are inserted one by one, the resulting tree can have arbitrary shape:

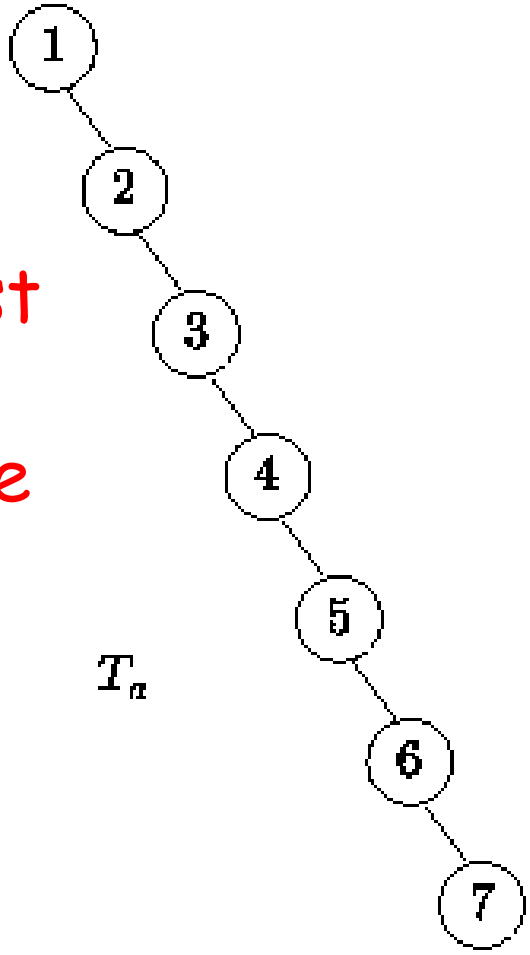


Insertion order:
9, 13, 11, 4, 5, 1, 7, 3, 8



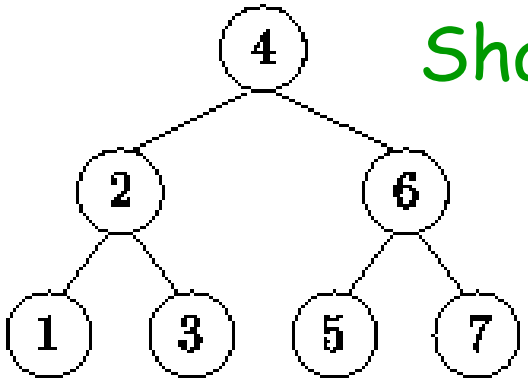
http://scienceblogs.com/goodmath/upload/2007/01/unbalanced-trees.jpg

Worst Case Shape



T_a

Best Case Shape



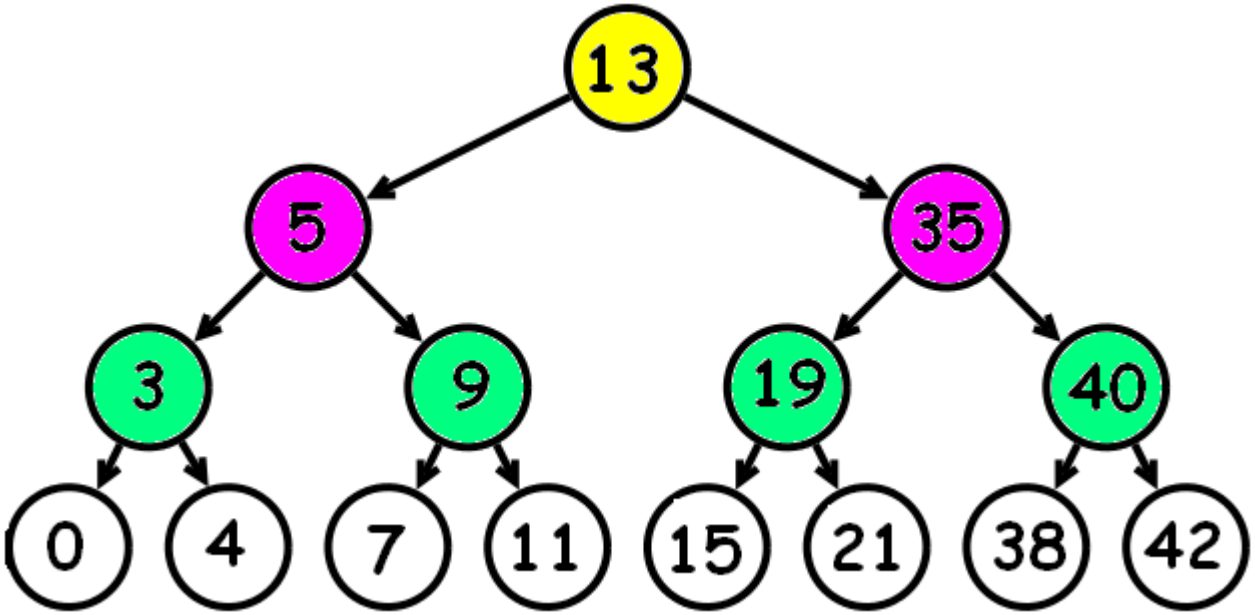
T_b

http://www.brpreiss.com/books/opus5/html/page304.html

Inductive Definition of a Binary Search tree:
A binary search tree is built up of nodes each of which has a key value, a left child and a right child (where the children are pointers to other nodes).
[Basis] A tree with 0 nodes is a binary search tree with root NULL.
[Inductive step] If T_1 and T_2 are binary search trees with roots r_1 and r_2 respectively, and r is a node with key value k, and further, all nodes in T_1 have key value $\leq$ k, and all nodes in T_2 have key value > k, then setting the left child of r to point to r_1 and the right child of r to point to r_2 creates a new binary search tree T with root r.

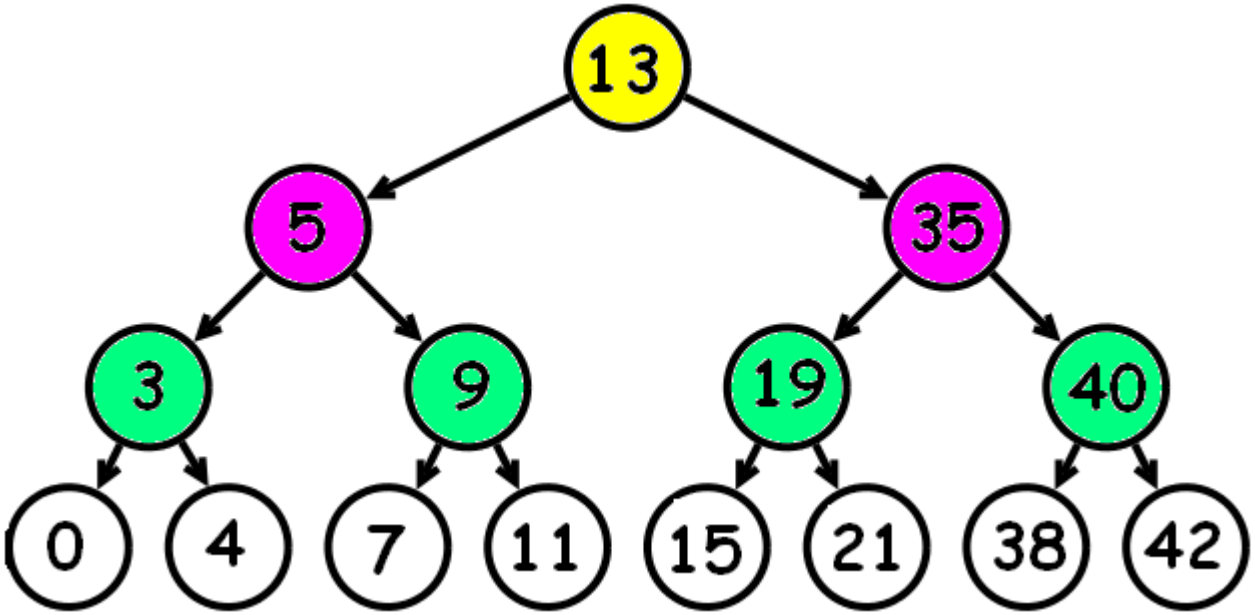
T_1 and T_2 are said to be respectively the *left* and *right subtrees* of the tree T rooted at r.

How much time does it take in the best case to build a complete binary tree of height h?



Recipe: "best case" order for the input:

1. Sort the input in an array.
2. Visualize the array as we do to create the underlying binary search tree for a binary search.
3. Write down keys using a level-order traversal.

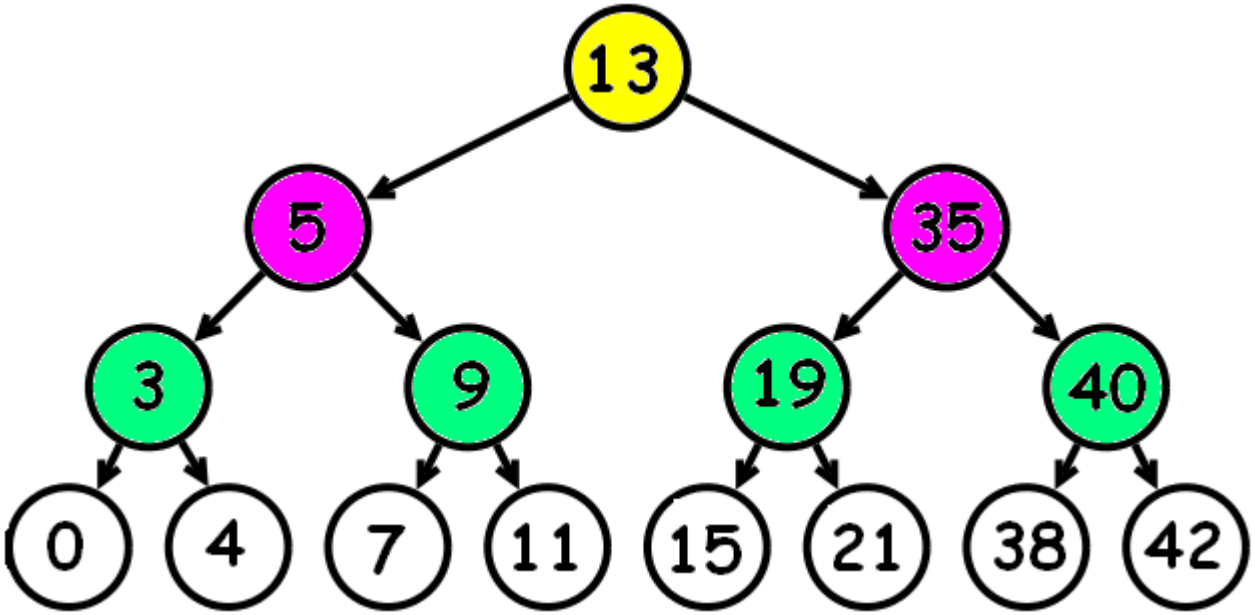


How much work is done to create the binary search tree when the input is ordered like this?

$S(n) = 1*1 + 2*2 + 4*3 + 8*4 + \ldots + 2^h*(h+1)$

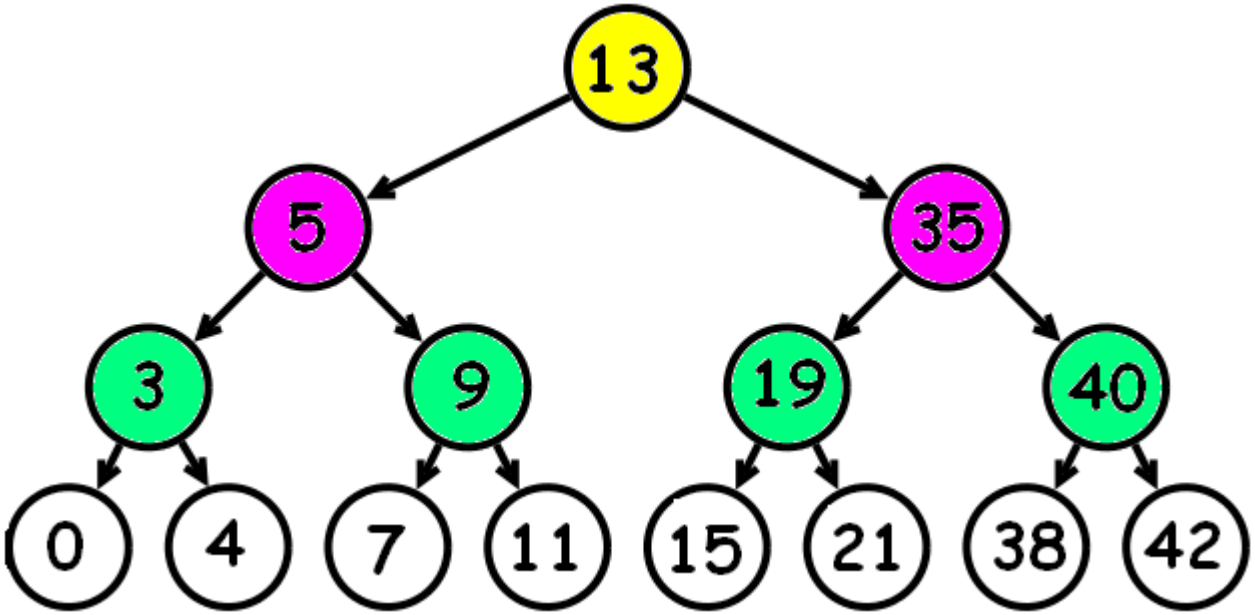
where $n = 1 + 2 + 4 + \ldots + 2^h = 2^{h+1} - 1$.

Last class we proved that $S(n) \in \Theta(n \log_2(n))$



Our lower bounding argument:

The key concept: Because for approximately $\frac{1}{2}$ of the insertions the time is in $\Omega(f(n))$, the total time is in $\Omega(n^* f(n))$.



For a binary search tree:

What is a worst case example for the insertion order for some keys for a given value n?

How much time does it take to build a binary search tree in the worst case?