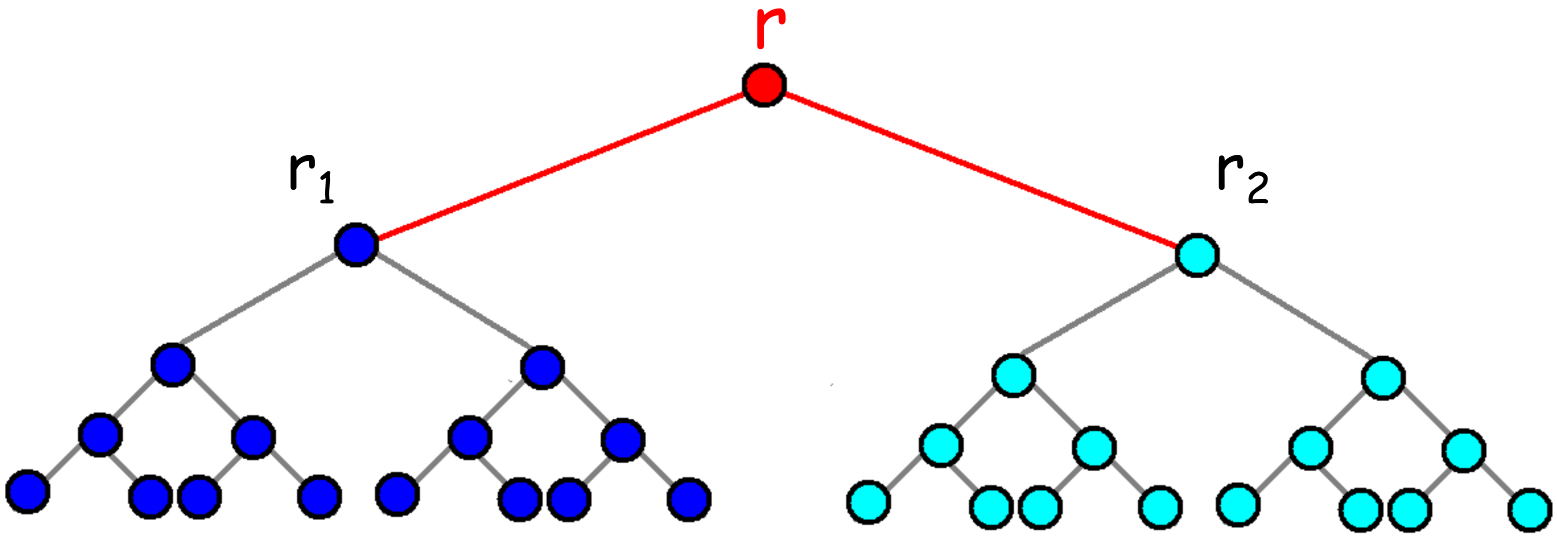
A complete binary tree of height 4:



Problem of the day:

Prove by induction that a complete binary tree of height h has $2^{h+1} - 1$ nodes.

Use the induction definition of a complete binary tree presented last class.

To check your answer for question 5 on assignment 1B:

Add to top of readRear:

```
int count=0;
```

Inside the loop:

```
current= start;
while (current.next != null)
{
      current= current.next; // Statement to count.
      count++;
      // Checkpoint 2.
}
```

At the end:

```
System.out.println("Executed " + count + " times on list of size " + n);
```

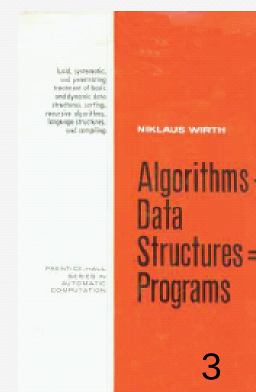
Slide by: Kevin Wayne
Princeton University

Why study algorithms?

To become a proficient programmer.

> *“ I will, in fact, claim that the difference between a bad programmer and a good one is whether he considers his code or his data structures more important. Bad programmers worry about the code. Good programmers worry about data structures and their relationships. ”*
>
> *— Linus Torvalds (creator of Linux)*

> *“ Algorithms + Data Structures = Programs. ”* *— Niklaus Wirth*



Divide and Conquer

1. Divide the problem into two or more subproblems.
2. Solve the subproblems.
3. Marry the solutions.

This is one of the most common problem solving tactics and leads naturally to recursive algorithms.

The floor of a number x denoted by $\lfloor x \rfloor$ is equal to the largest integer k that is ≤ x.

The ceiling of a number x denoted by $\lceil x \rceil$ is equal to the smallest integer k that is ≥ x.

How can you design a divide and conquer algorithms for reversing a list?

Assignment: the divide step splits a list of size n into two lists,
one with the first $\left\lfloor \frac{n}{2} \right\rfloor$ items from the list and
the second one with the next $\left\lceil \frac{n}{2} \right\rceil$ items.

Start with a base case: if the list has size 1 then it is already reversed so return.

For the recursive step: don't try to think deeper than one level. Just assume when you call your routine recursively that its behaviour matches the specifications for the routine in designing the top level.

Recursive call structure for reverse: