We are exploring the algorithm design technique known as **Divide and Conquer**. We'll see various algorithms that use this technique.

Running time analysis of such algorithms naturally involves *recurrences*, since we may state the running time in terms of the running time on smaller inputs. To help us determine the running time of such algorithms, let's think more about solving recurrences!

1 Recurrences

1.1 Recursion Tree Method (CLRS §4.4)

Example $T(n) = 2T\left(\frac{n}{2}\right) + cn^2$, $T(1) = c$

Example $T(n) = 2T\left(\frac{n}{2}\right) + c$, $T(1) = c$

1.2 The substitution method (CLRS §4.3)

Example

$$T(n) = 4T\left(\frac{n}{2}\right) + n,\ T(1) = 1$$

1.2.1 Careful with asymptotic notation and bogus proofs!

1.2.2 Ceilings and floors

Example

$$T(n) = T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 1,\ T(1) = 1$$

1.3 Master Theorem (CLRS §4.5)

Let $a \geq 1$ and $b > 1$ be constants, $f(n)$, $T(n)$ a function defined on natural numbers by the recurrence:

$$T(n) = aT(n/b) + f(n)$$

We interpret n/b to mean either $\lceil n/b \rceil$ or $\lfloor n/b \rfloor$. Then, $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.

Example Use the Master theorem to solve the following:

1. $T(n) = 9T\left(\lceil \frac{n}{3} \rceil\right) + n$
2. $T(n) = T\left(\lceil \frac{2n}{3} \rceil\right) + 1$
3. $T(n) = 2T\left(\lceil \frac{n}{2} \rceil\right) + n^2$

What is an example of a recurrence that does not fit the form of the Master theorem?

2 Quicksort (CLRS §7.1, 7.2)

Idea:

Example

```
1 k=PARTITION
2 QUICKSORT
3 QUICKSORT
```

2.1 Partition

```
1 pivot =
2 i =
3 for j = 1 to n-1
4      if A[j]
5
6           i =
7
8 RETURN i
```

2.2 Correctness of Partition (and Quicksort)

Loop invariant:

- Initialization:

- Maintenance:

- Termination:

2.3 Running Time of Partition and Quicksort

3 Integer Multiplication (Karatsuba) (KT §5.5)

Input:

Goal:

3.1 Elementary School Algorithm

- Time complexity of grade school addition:

- Time complexity of grade school multiplication:

3.2 Algorithm Using Divide & Conquer

First attempt:

Total runtime:

You try! Describe a procedure that given four integers a, b, c, d, outputs the three numbers ab, cd and $ad+bc$ and uses only **three** multiplications (four would be obvious). You are free to use as many additions and subtractions as you wish.
(Hint: Consider the product $(a+c)(b+d)$.)

Second attempt:

4 Closest pair of points (KT §5.4)

Problem: given n points $p_1, p_2, \ldots, p_n \in \mathbb{R}^2$ in the plane, find a pair with the smallest Euclidean distance between them.

- Brute force:

- If all points line on a line:

For simplicity, we will assume that no two points have the same x coordinate.

4.0.1 Divide and Conquer Algorithm

1. Divide

2. Conquer:

3. Combine:

Observation: Only need to consider pairs

The combine step: Compute

1. Take points

2. Sort these points by

3. For each point, compute its distance to

4. Output closest pair found.

Claim.

Proof.

4.1 Correctness

4.2 Running time