Computational Thinking

Computational Thinking

* Different from programming
* Fundamental skill of computer science
* Enables software engineering

Computational Thinking

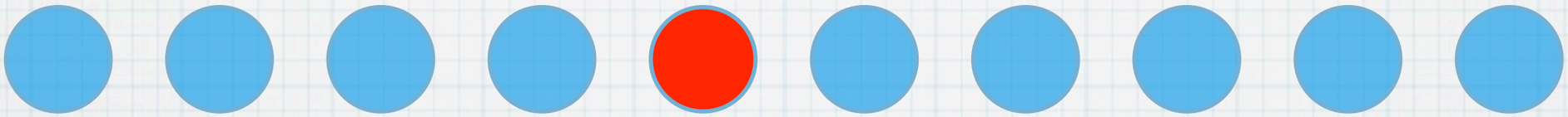
* Problem solving
* Algorithms
* System design

A problem to solve

For now...

Getting the right answer is not as important as finding a good strategy to solve the problem.

Handshake Problem #1



Handshake Problem #1

You are one of 10 people in a room.

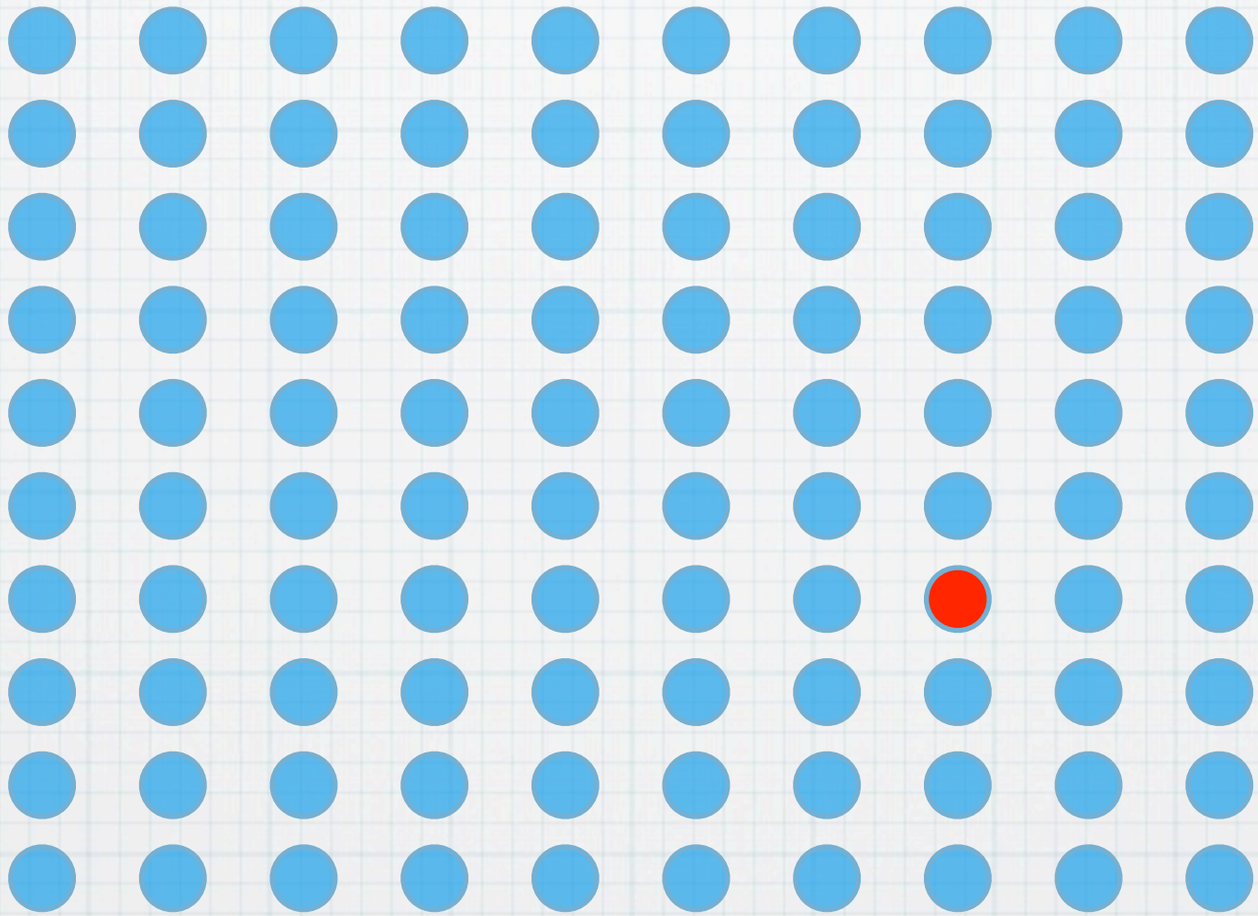
You must shake hands with everyone in the room.

How many hands will you shake?

What if there are 100 people? How many hands?

And what if there are N (> 2) people?

Handshake Problem #1



Handshake Problem #1

N?

Handshake Problem #2

Handshake Problem #2

You are one of 10 people in a room.

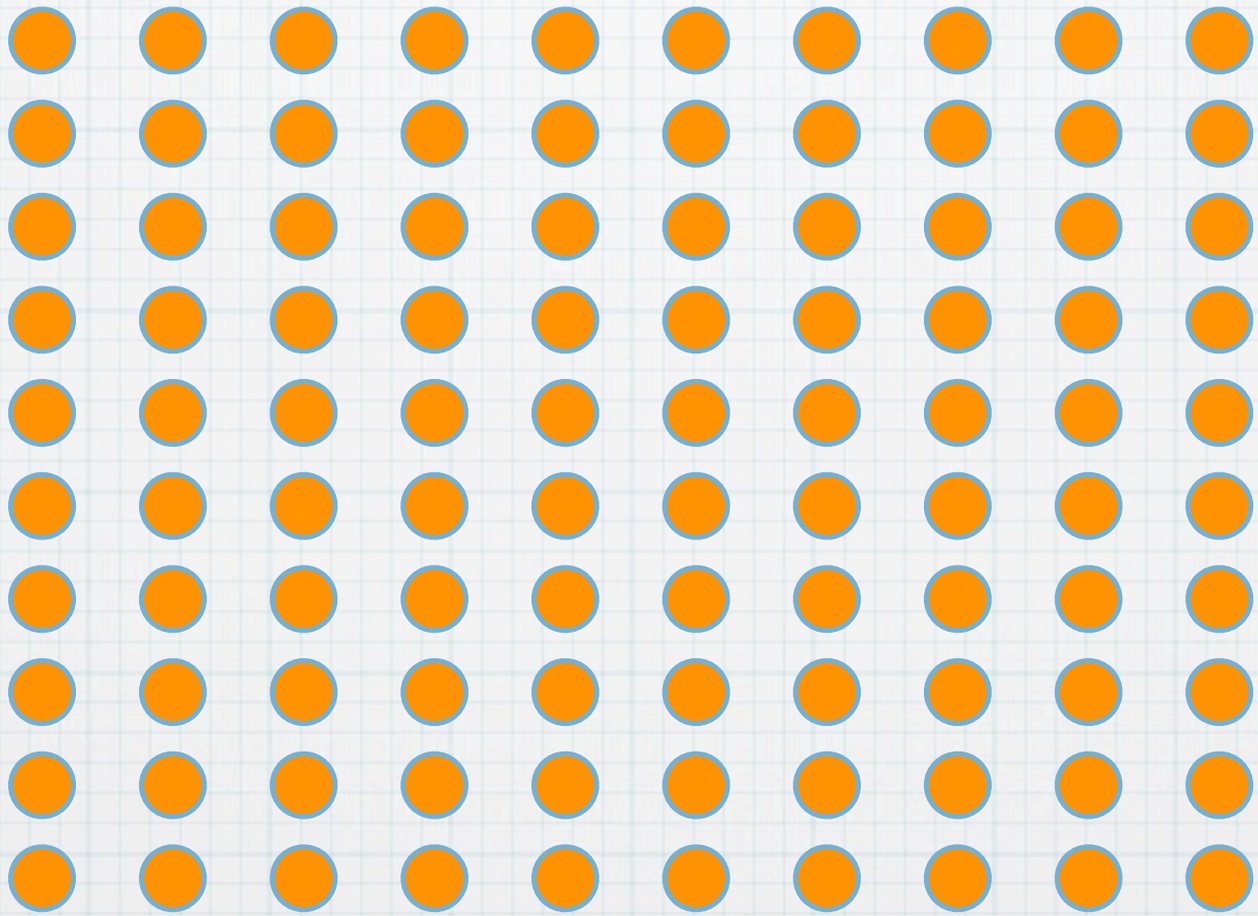
Each person in the room must shake hands one time, and ONLY ONE time, with ALL the other people in the room. (No two people shake hands more than once.)

How many TOTAL handshakes will occur?

What if there are 100 people? How many handshakes?

And what if there are N people?

Handshake Problem #2



Handshake Problem #2

N?

Problem Solving

* Small groups work together
* Find a METHOD to solve the problem
* Write out the steps of your plan
* Execute it and show your work
* Give your plan to another group to try
* Share the method with the class

Possible methods

* Act it out
* Draw diagrams
* Analyze the math
* Write a program (!)

Reflection

* Why is this kind of problem important?
* Where might it come up in the world?
* How could your solution be used in other domains (e.g., carpentry, cooking, fashion) ?
* (extra) What is the solution for N ? Can you prove it mathematically?