CS 106X
Lecture 1: Welcome!

Monday, January 9, 2017

Programming Abstractions (Accelerated)
Winter 2017
Stanford University
Computer Science Department

Lecturer: Chris Gregg

Today's Topics

- Instructor Introductions
- What is CS 106X?
  - Goals for the Course
  - Components of CS 106X
  - Assignments, Grading scale, Due dates, Late days, Sections, Getting Help
  - Is CS 106X the right class?
- C++
  - Why C++?
  - QT Creator
  - Our first program
  - Our second program
  - The importance of Data Structures
- Assignment 0

Chris Gregg

- Career:
- Johns Hopkins University Bachelor's of Science in Electrical and Computer Engineering
- Seven years active duty, U.S. Navy (14+ years reserves)
- Harvard University, Master's of Education
- Seven years teaching high school physics (Brookline, MA and Santa Cruz, CA)
- University of Virginia, Ph.D. in Computer Engineering
- Three years teaching computer science at Tufts University
- Stanford! (arrived, Fall 2016)
- Personal website: http://ecosimulation.com/chrisgregg

CS106X Staff

Head TA: Aaron Broder

Section Leaders

What is CS 106X?

CS106X: Learn core ideas in how to model and solve complex problems with computers

CS106X: Learn core ideas in how to model and solve complex problems with computers

accelerated!

accelerated

- Why "accelerated"?
  - This class goes more deeply into topics than 106B, and will cover some different topics.
  - Some of the assignments are similar to 106B, but there will be more parts to the assignments, and they will generally be more difficult.
  - You are a self-selected group — we expect a lot from you, but we also expect that you like to be challenged!
  - You may already know some of the things we cover in class — great! Use the opportunity to *go deeper* into the knowledge.

- This is a relatively small class, so take advantage of that! Get to know Chris and Aaron, and your Section Leader!

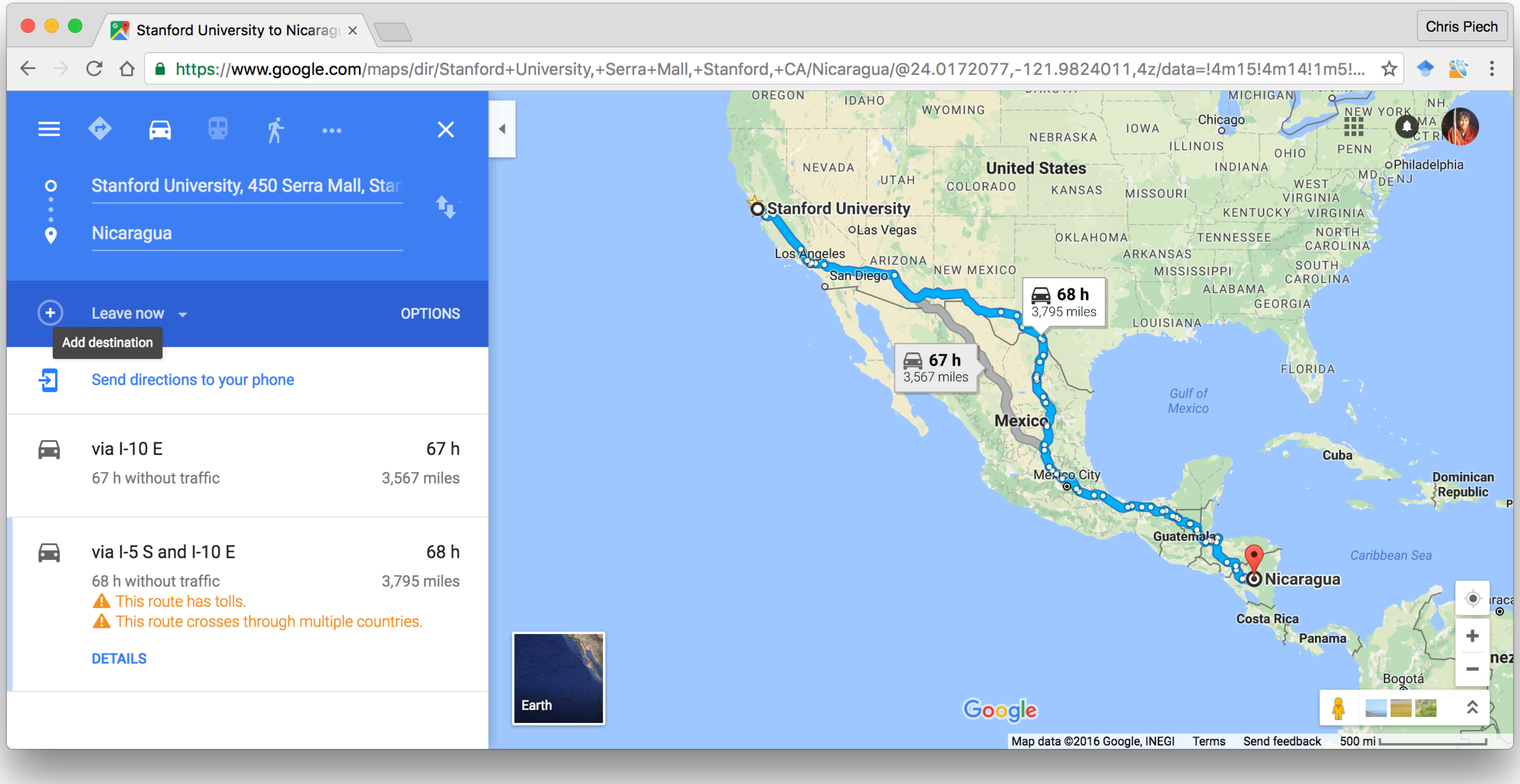
- You will be proud when you've completed this class.

CS106X: Learn core ideas in how to model and solve complex problems with computers

Complex Problems: Self Driving Cars

Stanford's Stanley Self Driving Car, DARPA Grand Challenge, 2006

Complex Problems: Instantaneous Directions



Stanford University to Nicarag ×

Chris Piech

https://www.google.com/maps/dir/Stanford+University,+Serra+Mall,+Stanford,+CA/Nicaragua/@24.0172077,-121.9824011,4z/data=!4m15!4m14!1m5!...

Stanford University, 450 Serra Mall, Star

Nicaragua

Leave now ▾ OPTIONS

Add destination

Send directions to your phone

via I-10 E	67 h
67 h without traffic	3,567 miles

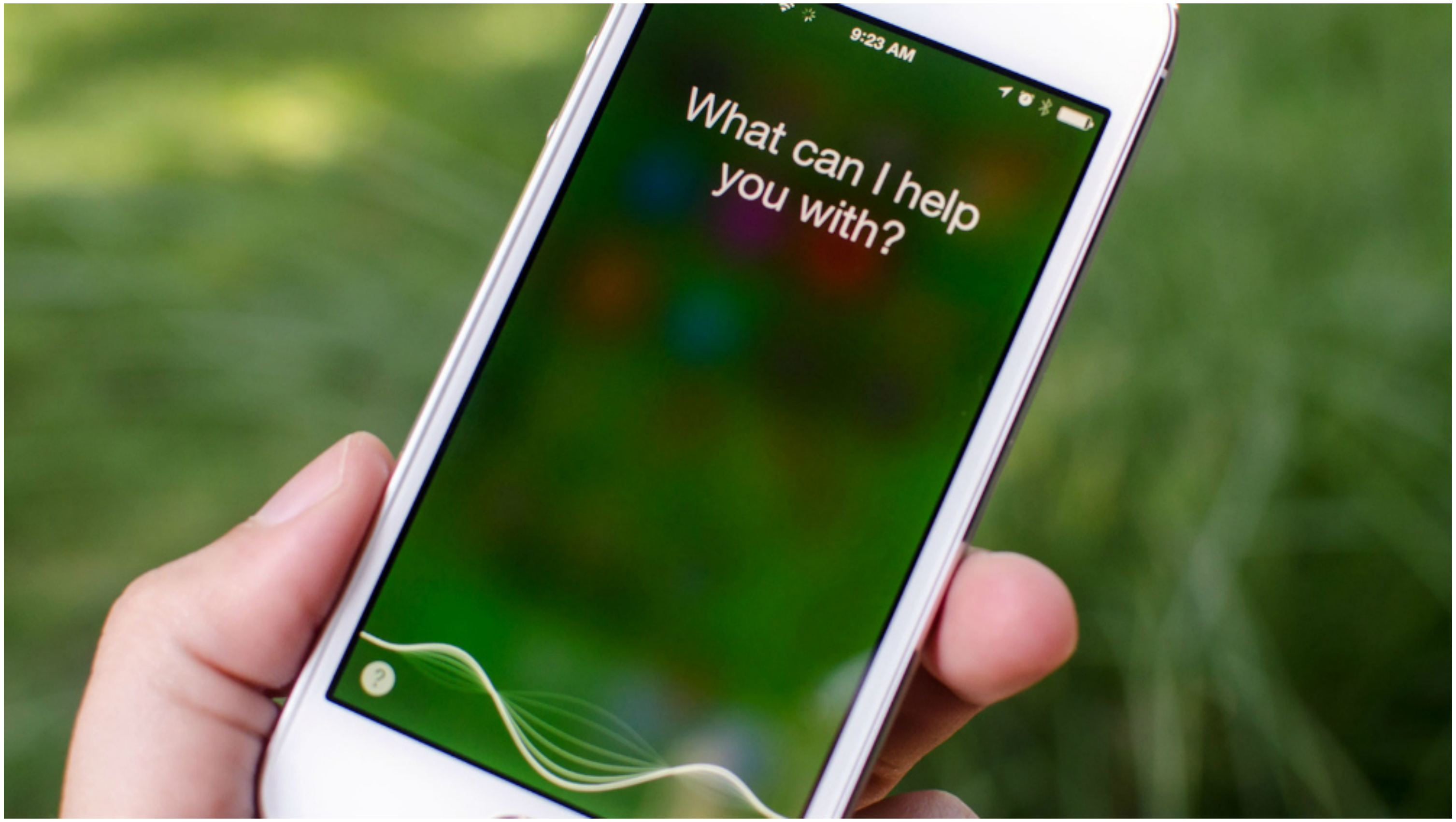
via I-5 S and I-10 E	68 h
68 h without traffic	3,795 miles

⚠ This route has tolls.

⚠ This route crosses through multiple countries.

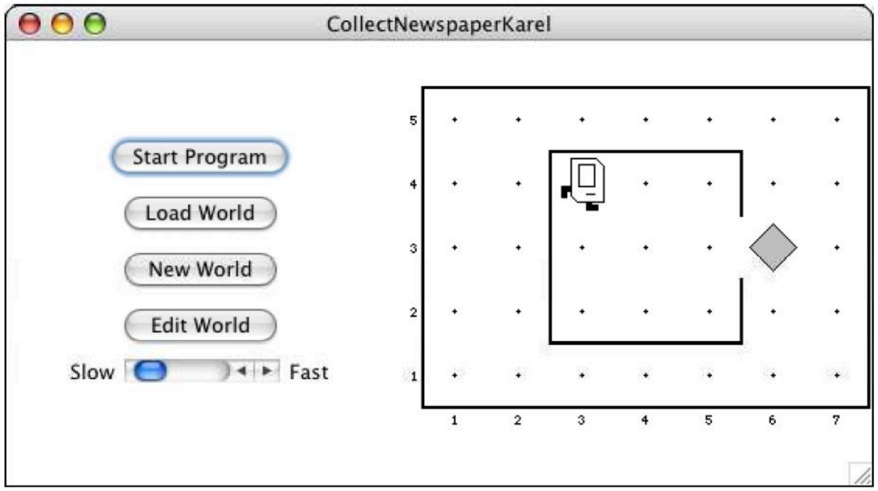
DETAILS

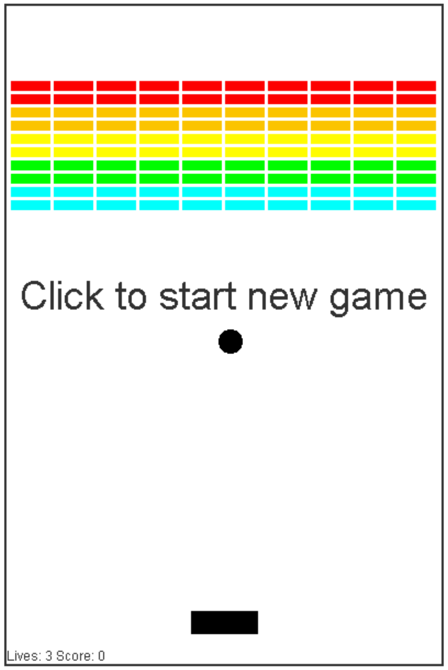
Complex Problems: Speech Recognition



How does Stanford get you there?

CS106A


CollectNewspaperKarel


Click to start new game

Lives: 3 Score: 0

In CS106A is a first course in programming, software development

There is more to learn…

Full disclosure, CS106X is necessary but not sufficient to make a self driving car ☺️

Goals for CS 106X

Learn core ideas in how to model and solve complex problems with computers.

To that end:

Explore common abstractions

Harness the power of recursion

Learn and analyze efficient algorithms

Goals for CS 106X

Learn core ideas in how to model and solve complex problems with computers.

To that end:

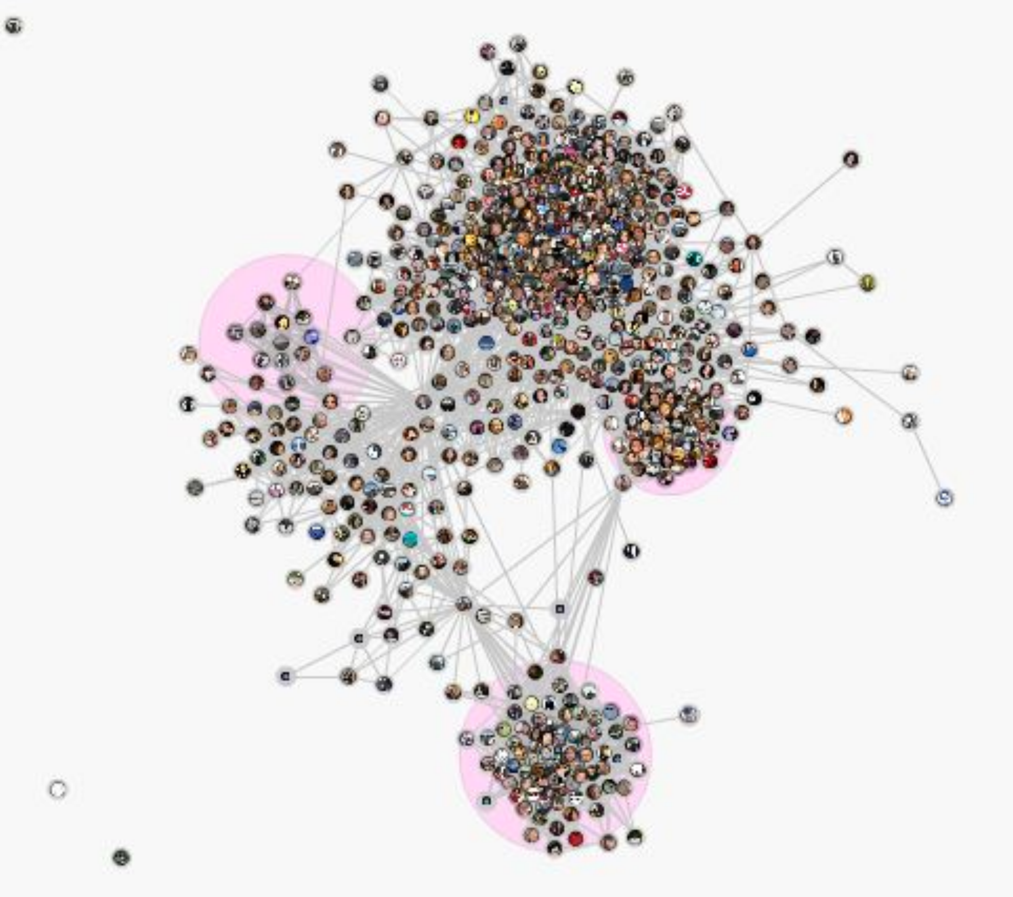
Explore common abstractions

Harness the power of recursion

Learn and analyze efficient algorithms

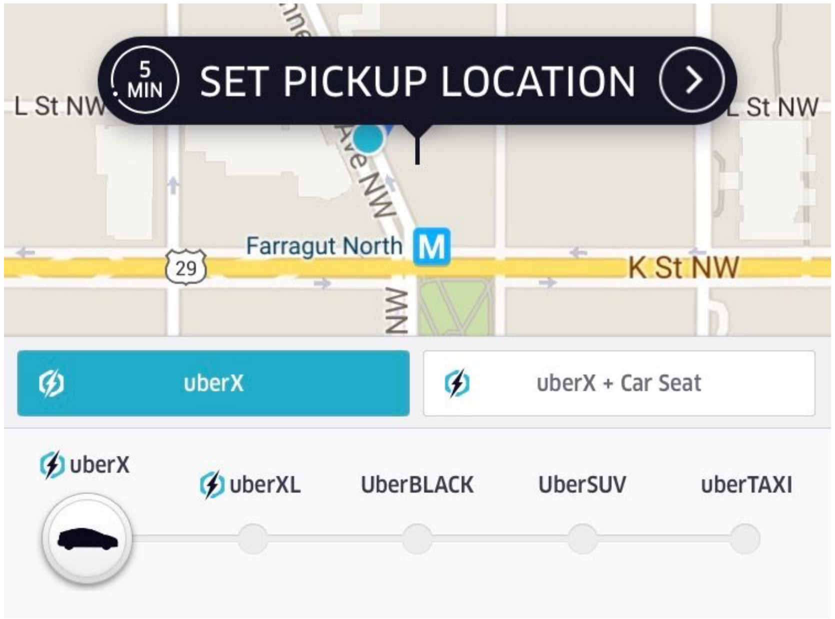
Common Abstractions

- What is the average friend distance between two random Facebook users?



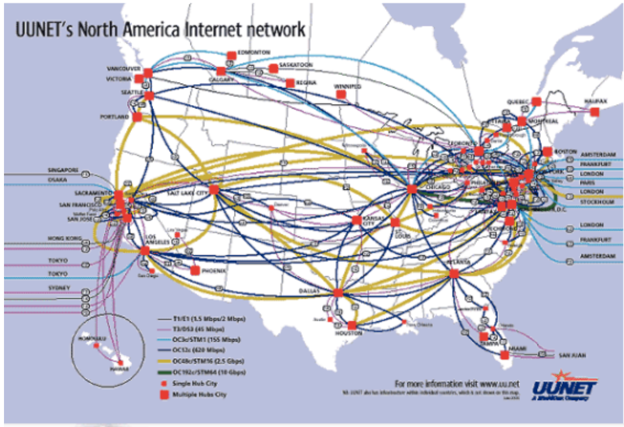
Common Abstractions

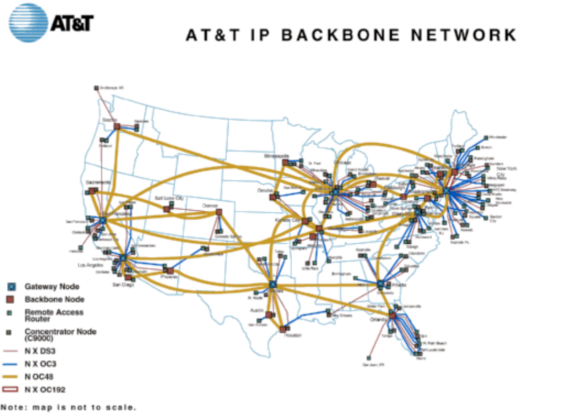
- How should Uber direct drivers in San Francisco at 5pm on a Tuesday, when there are *x* number of people who want a ride, and *y* number of drivers?

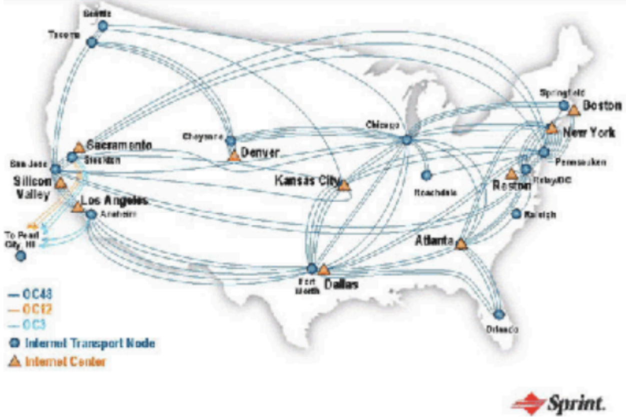


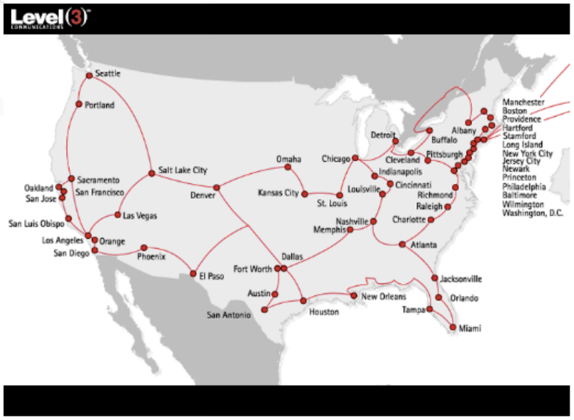
Common Abstractions

- How does email get from Dallas, Texas to Miami, Florida?









Common Abstractions

- What is the average friend distance between two random Facebook users?
- How should Uber direct drivers in San Francisco at 5pm on a Tuesday, when there are *x* number of people who want a ride, and *y* number of drivers?
- How does email get from Topeka, Kansas to Anchorage Alaska?

- These are all solved with the same abstraction! (using a "graph," which we will learn about near the end of the course)
- By learning common abstractions, we can use those abstractions to solve many problems.
- See the course website to see the list of topics we will cover.

Goals for CS 106X

Learn core ideas in how to model and solve complex problems with computers.

To that end:

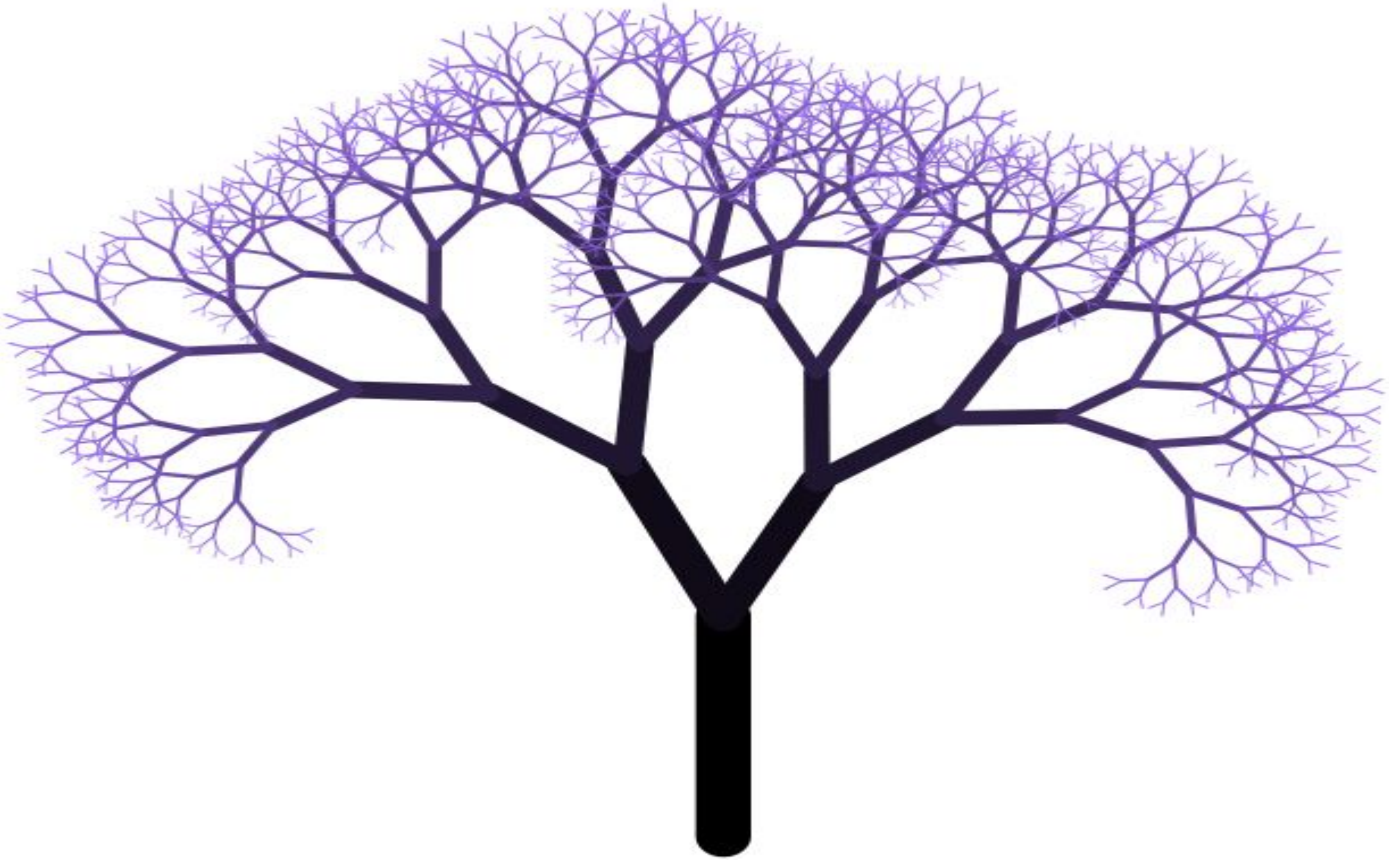
Explore common abstractions

Harness the power of recursion

Learn and analyze efficient algorithms

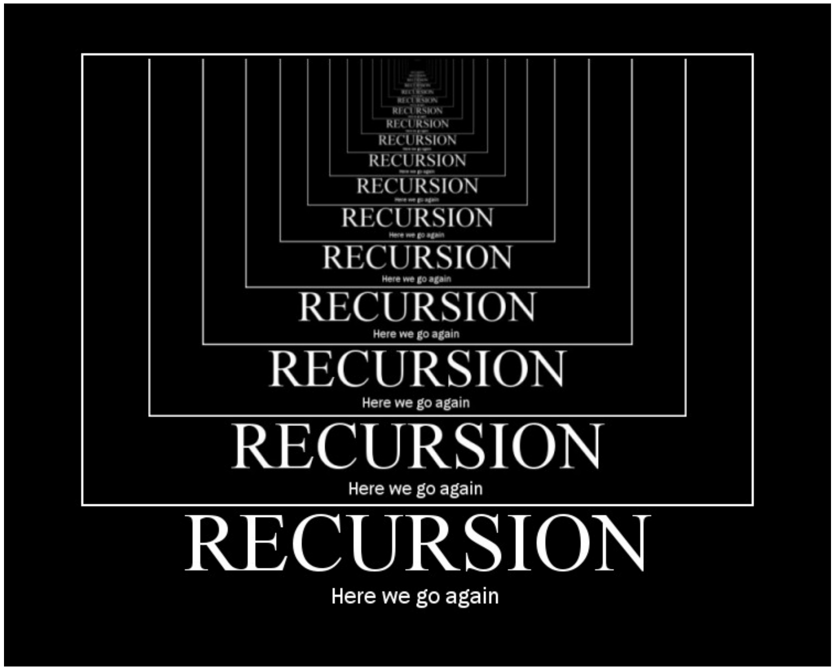
Recursion

In order to understand recursion, you must understand recursion.



Recursion

Recursion is a powerful tool that we will learn — once you start "thinking recursively", you will be able to solve many problems that would be extremely hard to solve without it.



Goals for CS 106X

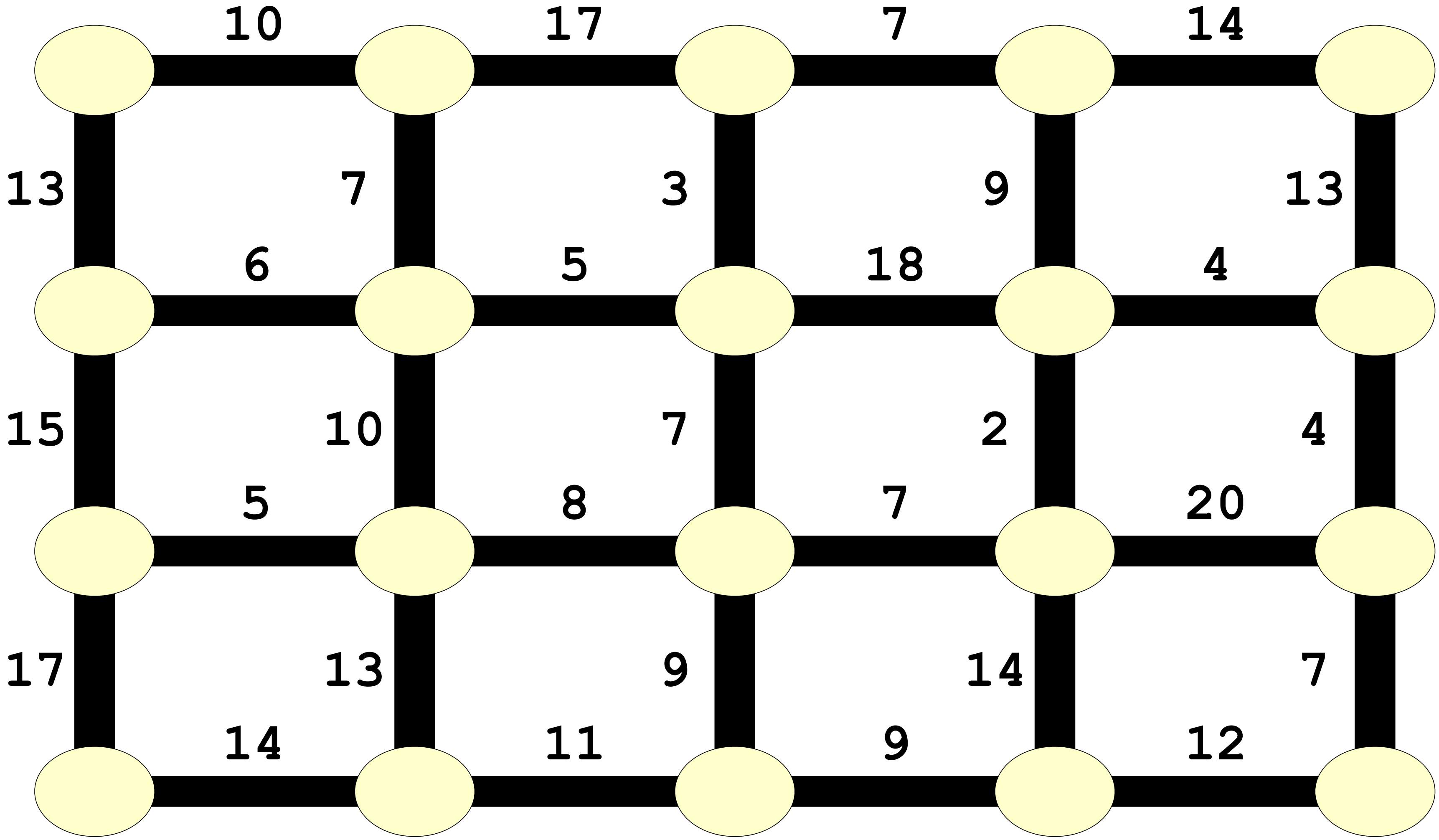
Learn core ideas in how to model and solve complex problems with computers.

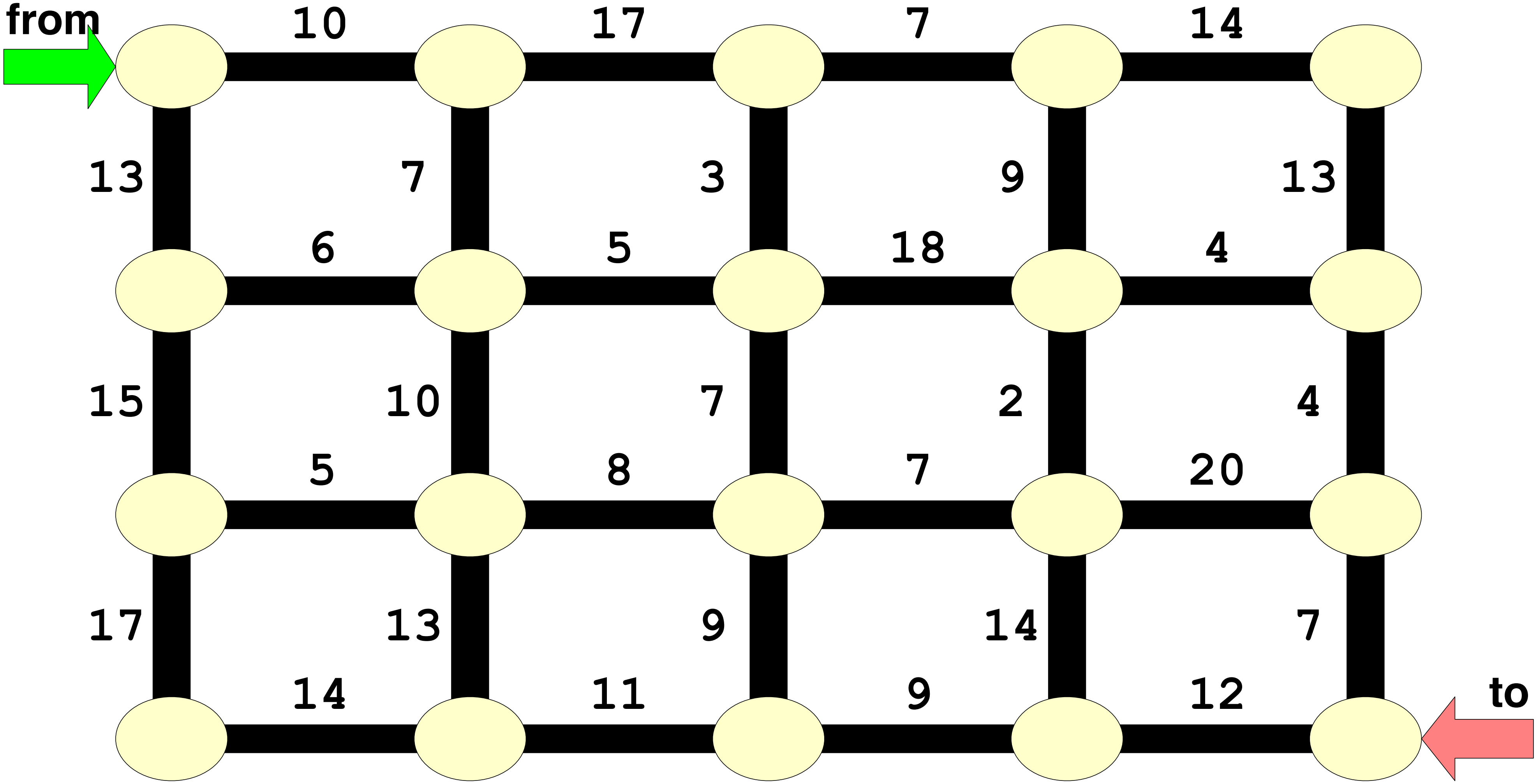
To that end:

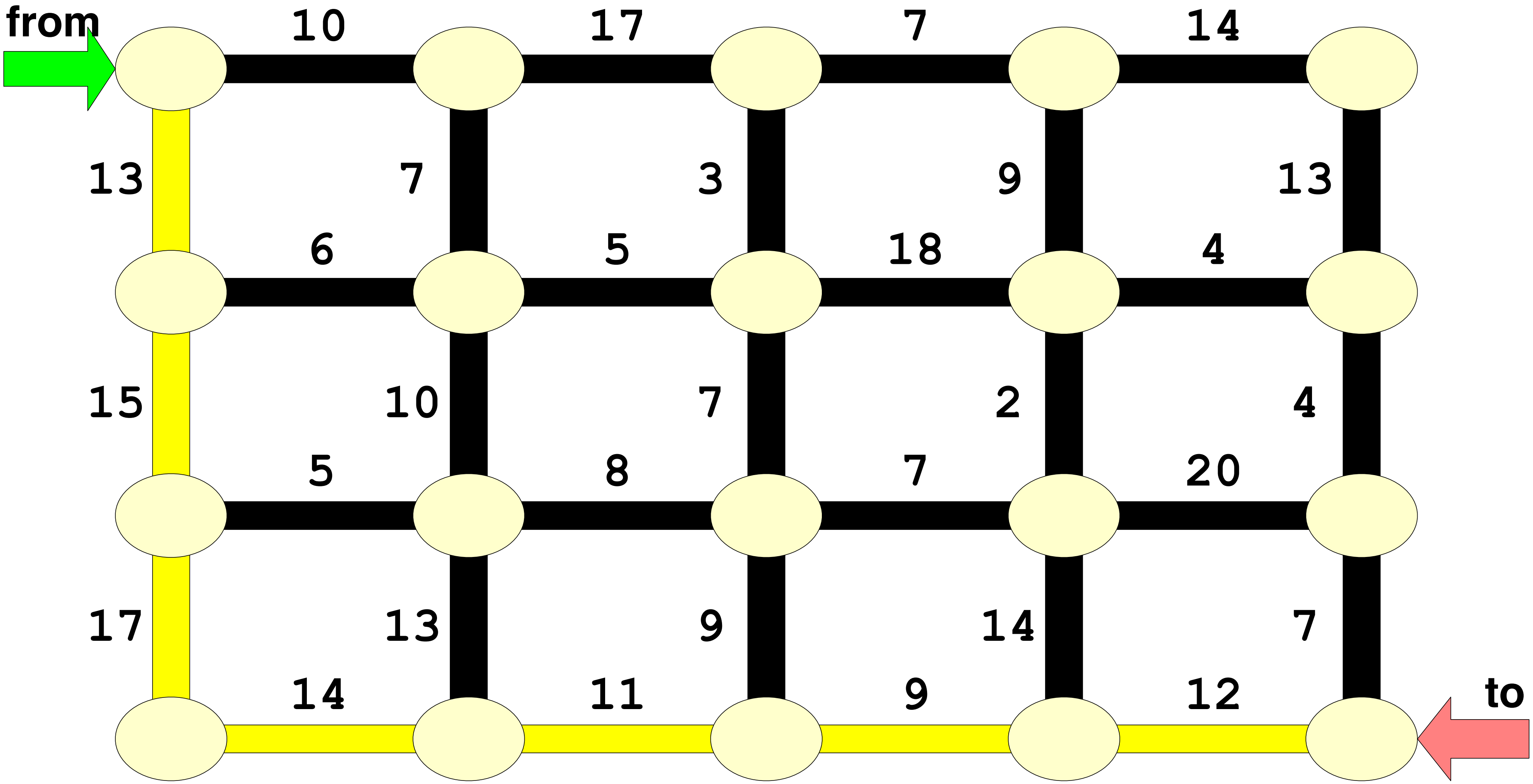
Explore common abstractions

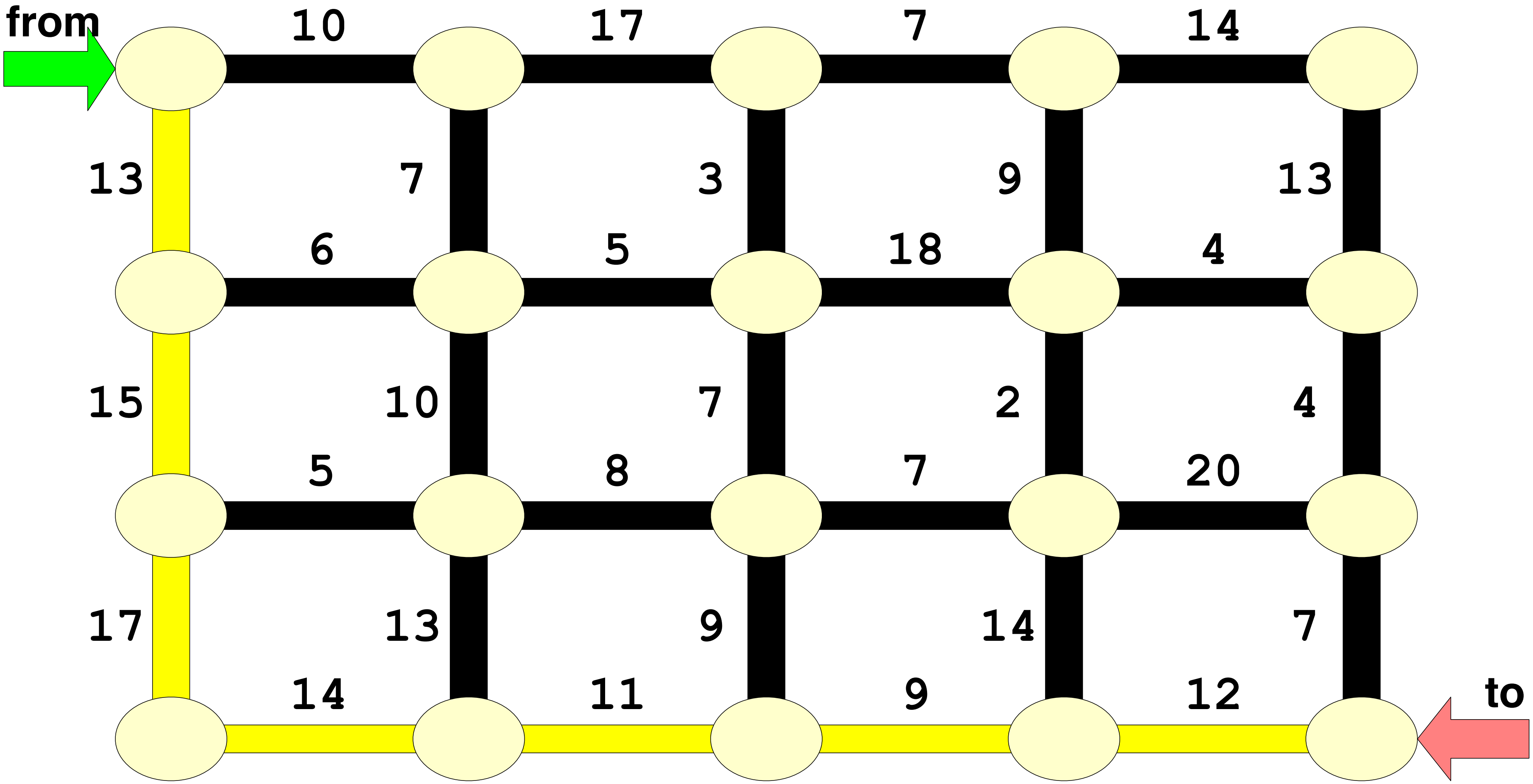
Harness the power of recursion

Learn and analyze efficient algorithms

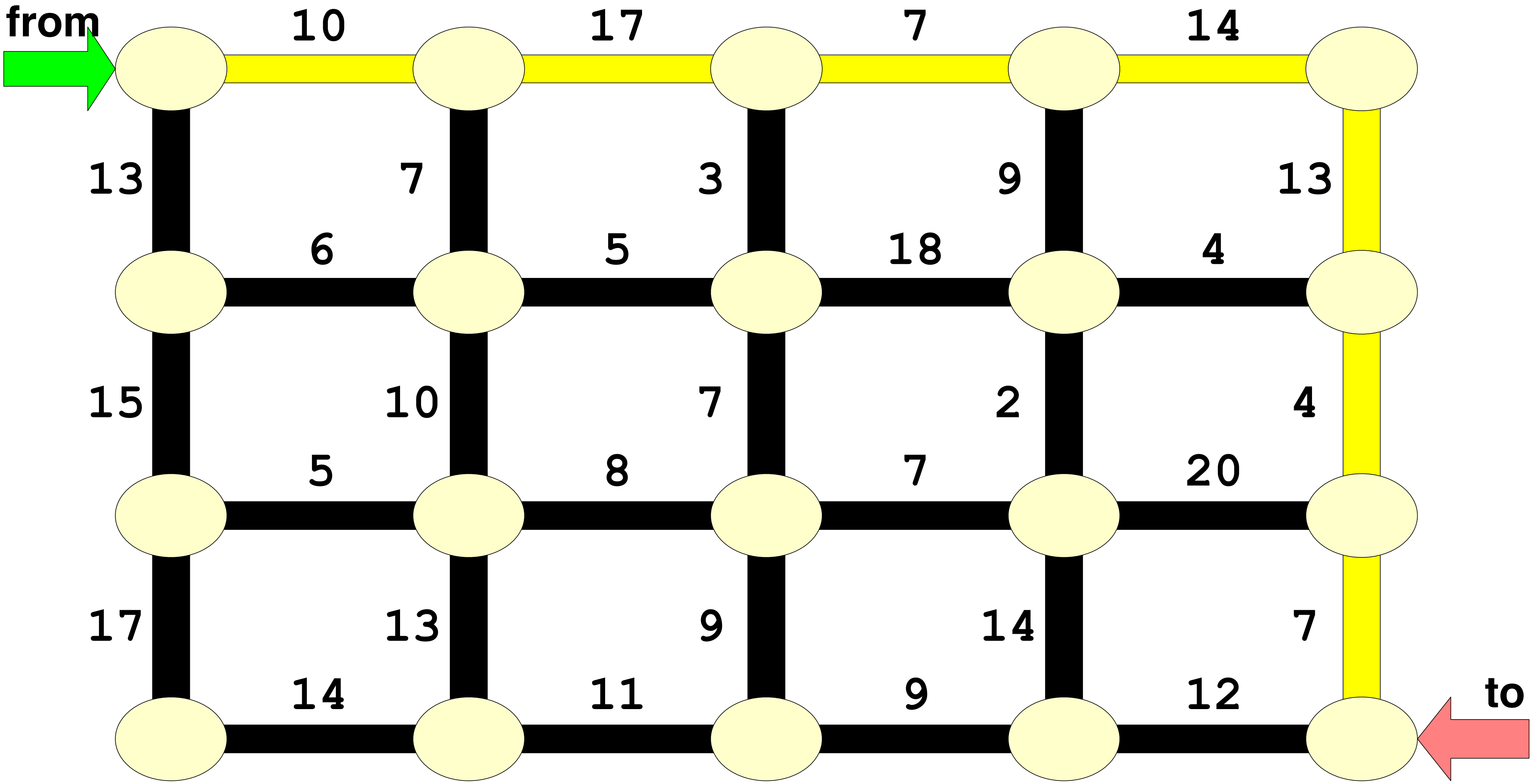

10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12


from
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to

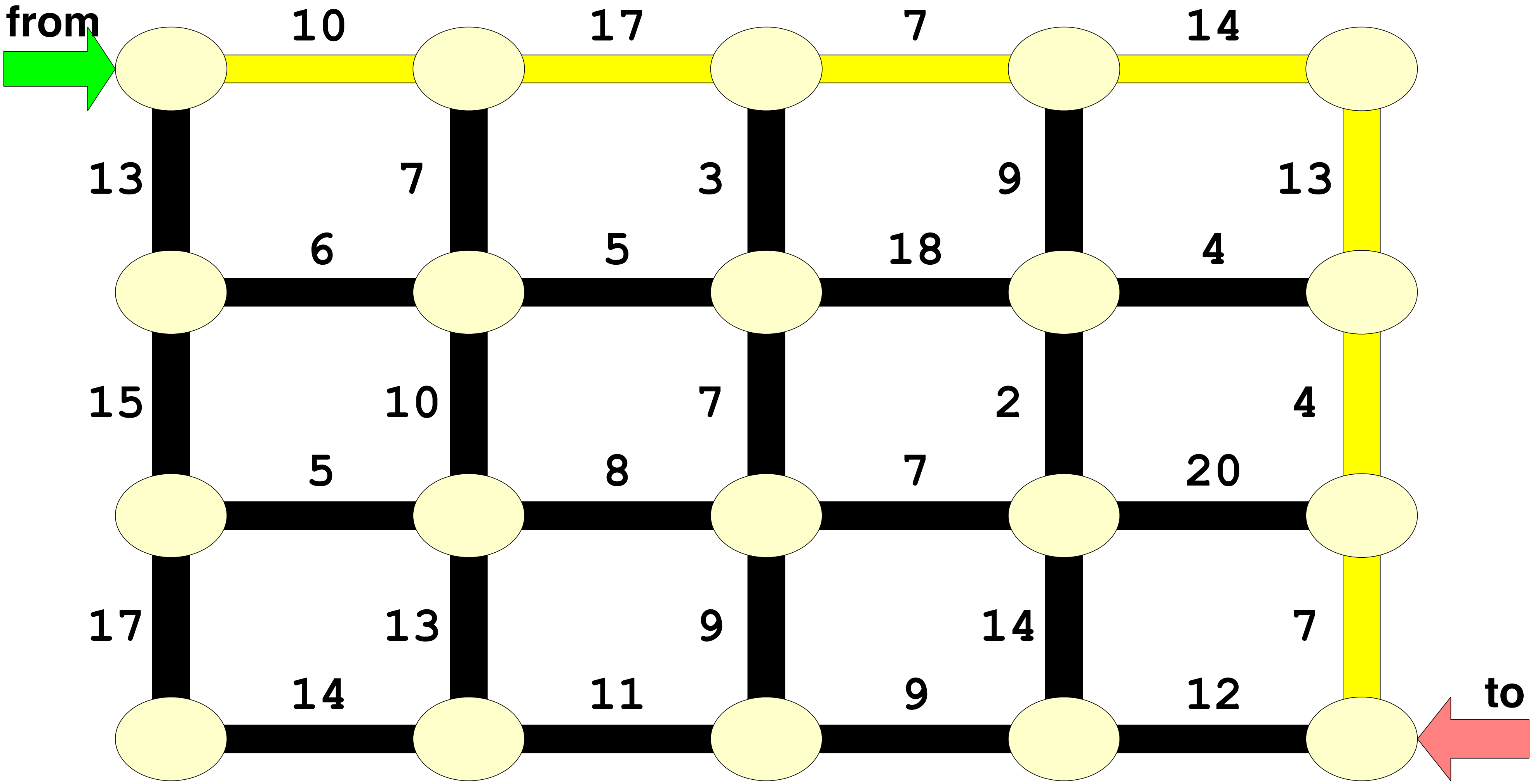

from
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to



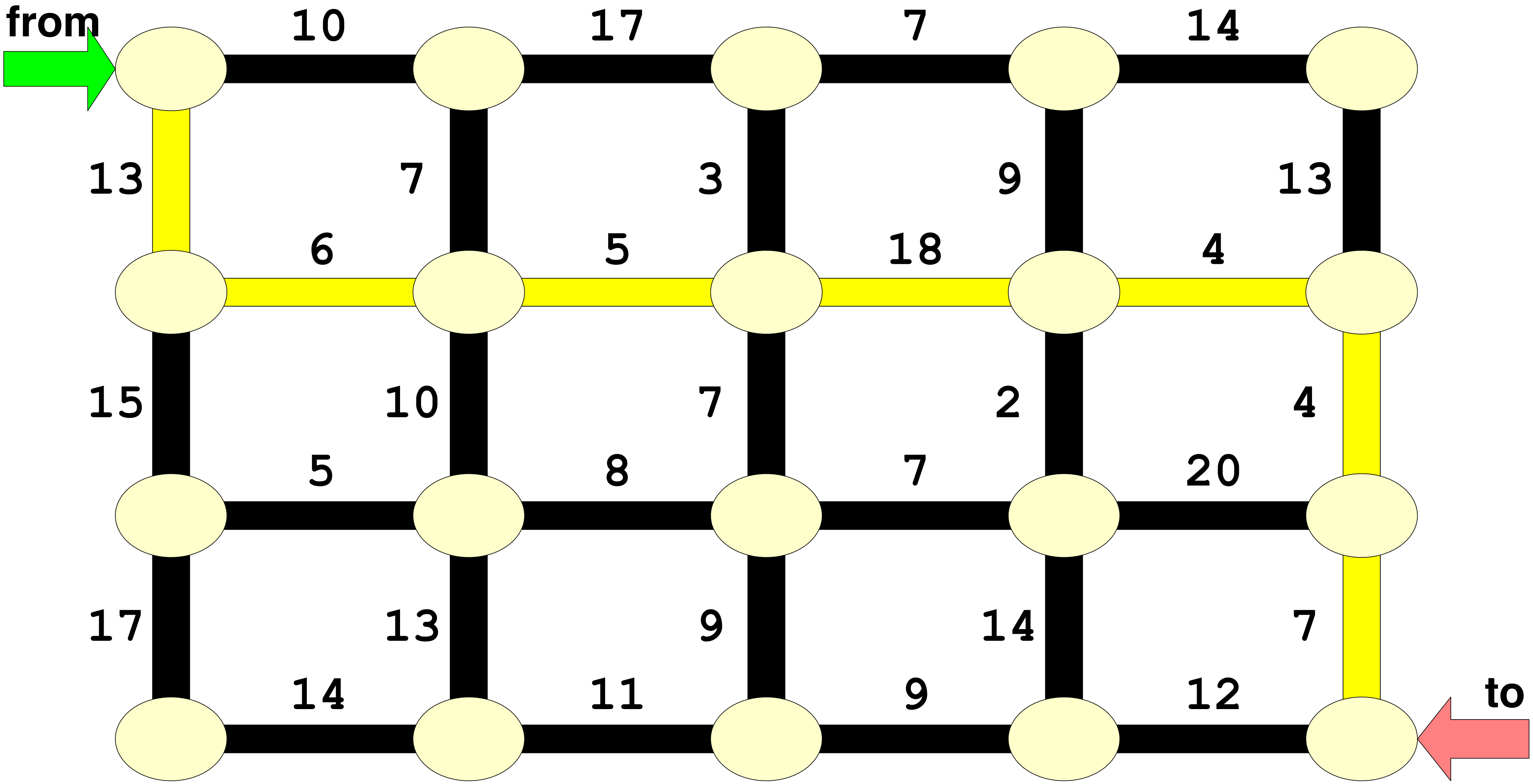
Travel Time: 13 + 15 + 17 + 14 + 11 + 9 + 12 = **91**

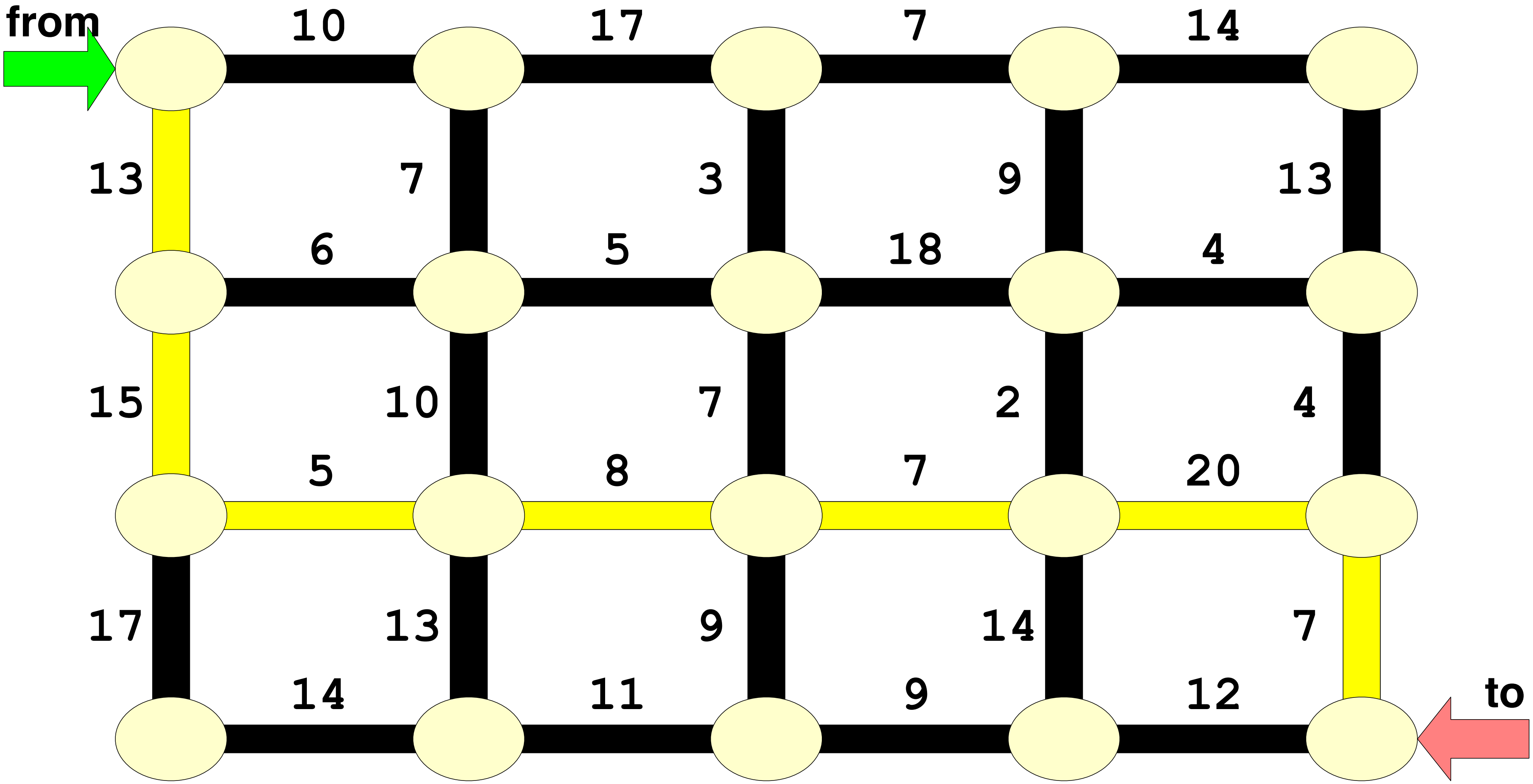


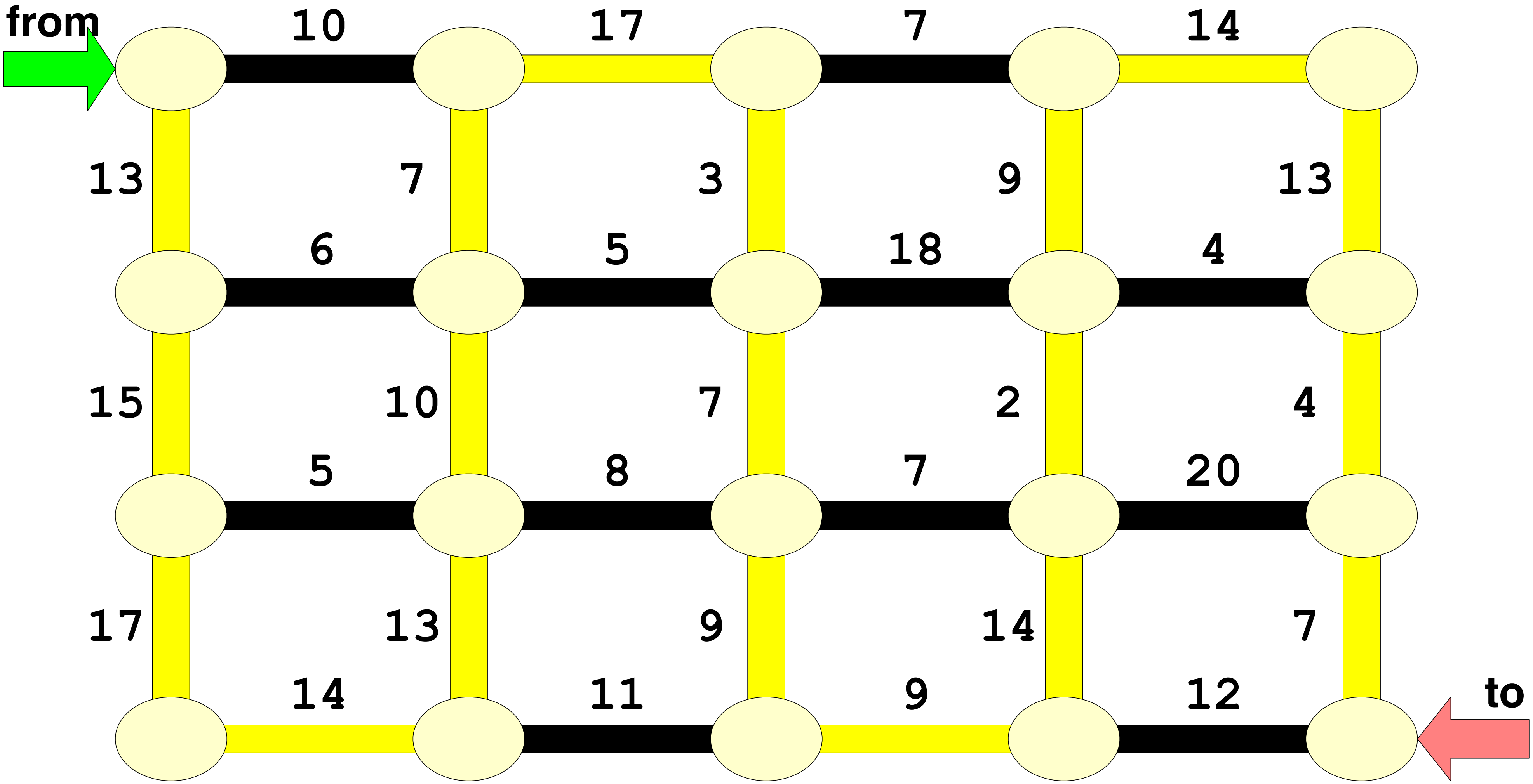
from
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to

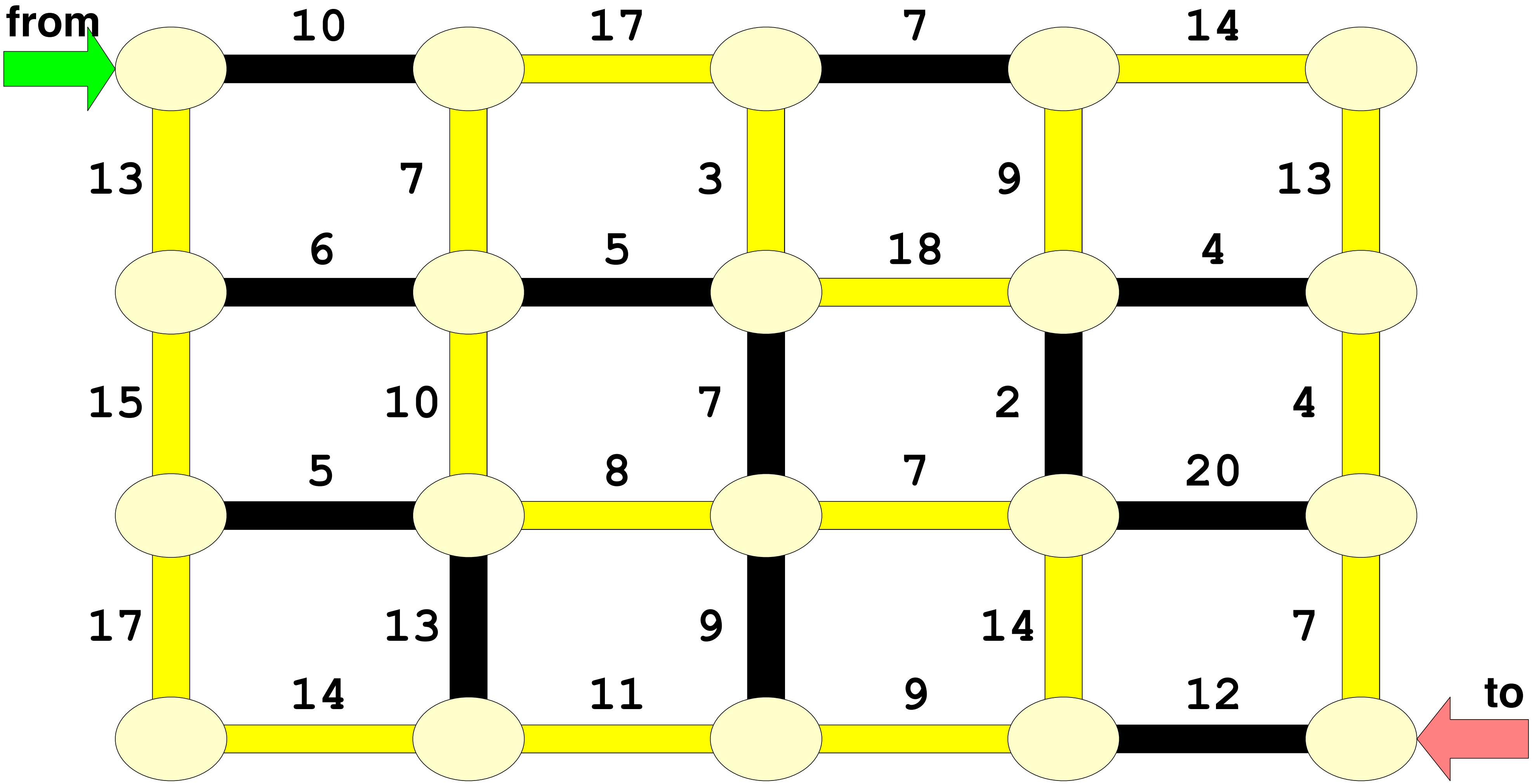


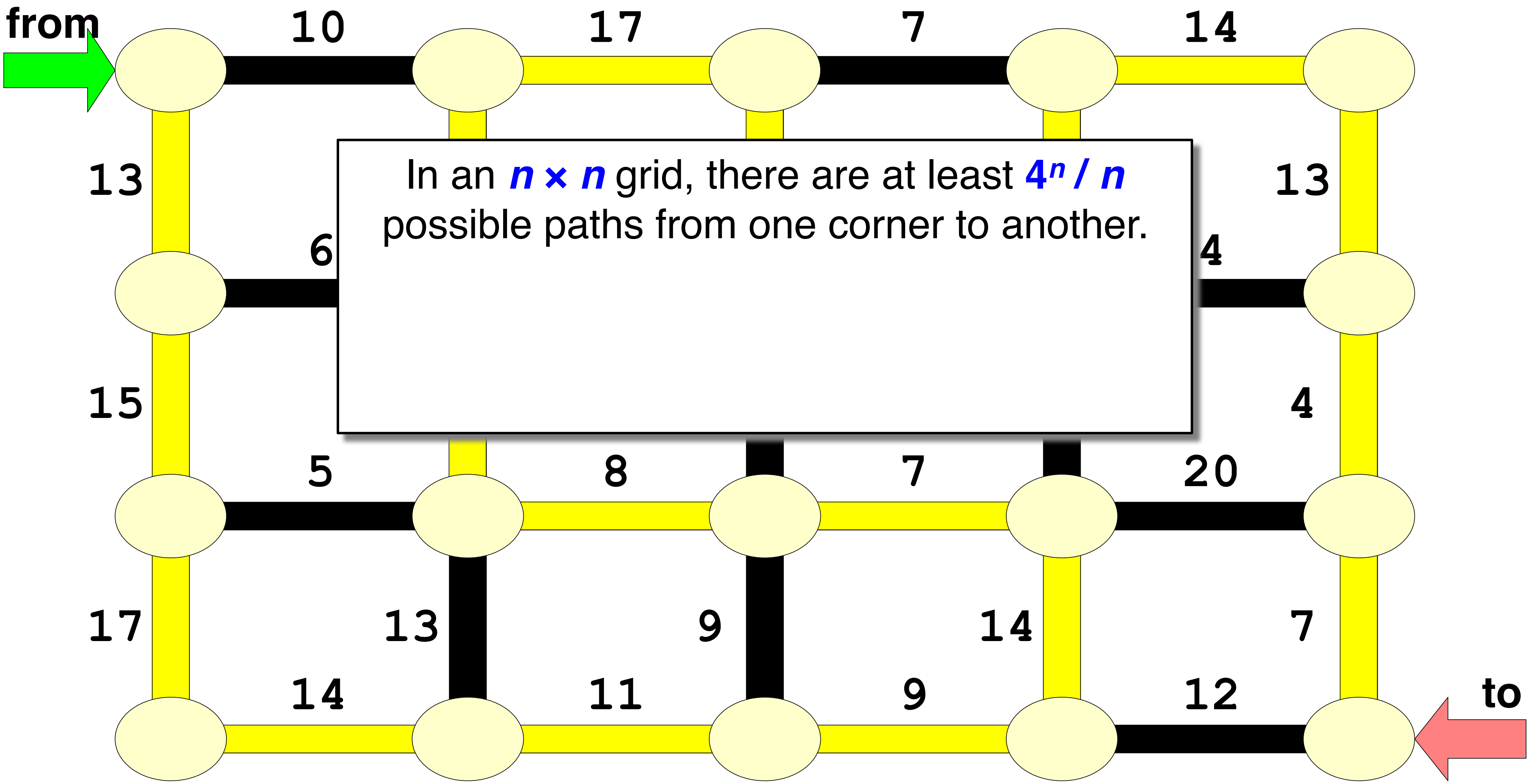
Travel Time: 10 + 17 + 7 + 14 + 13 + 4 + 7 = **72**


from
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to

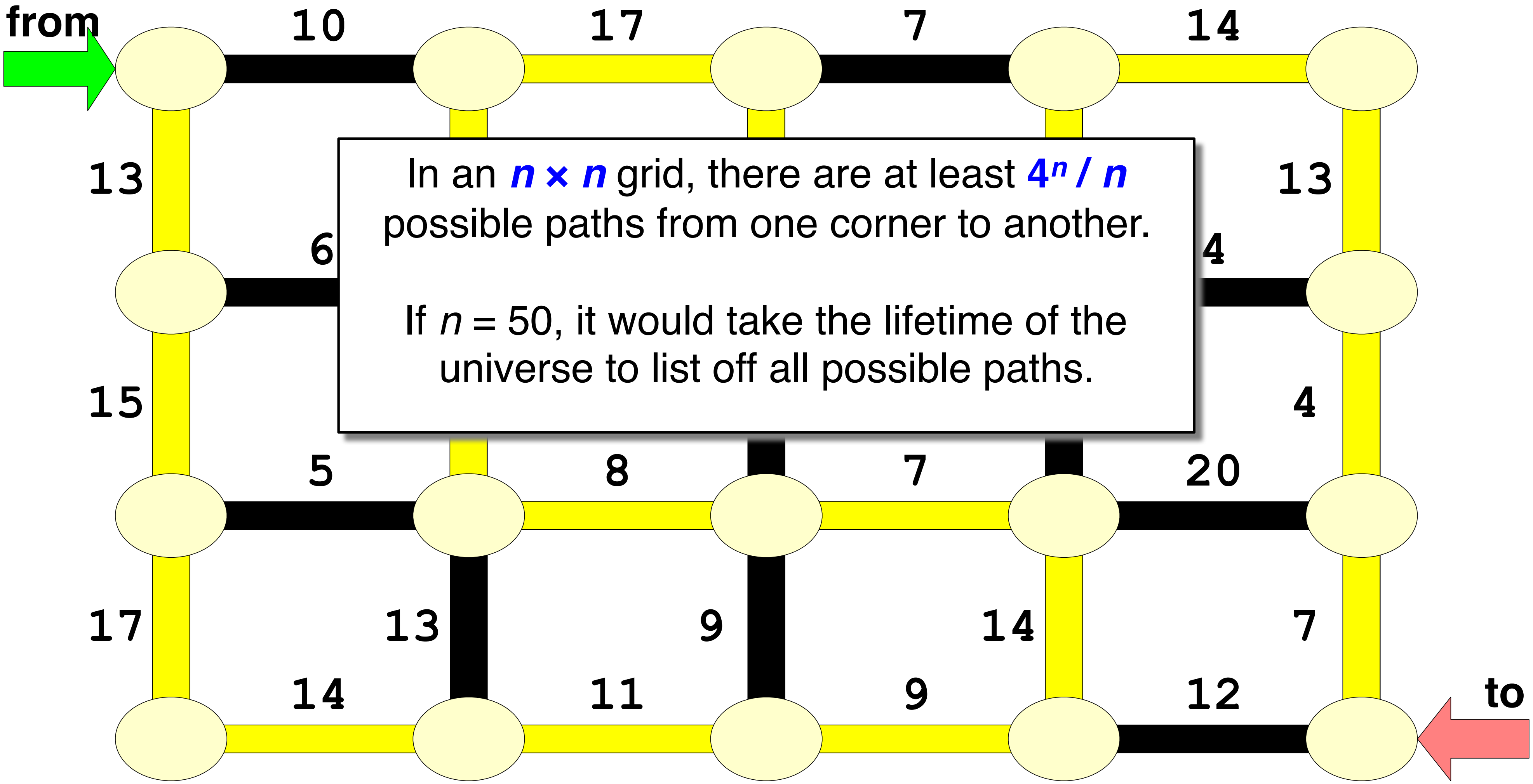

from
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to


from
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to


from
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to

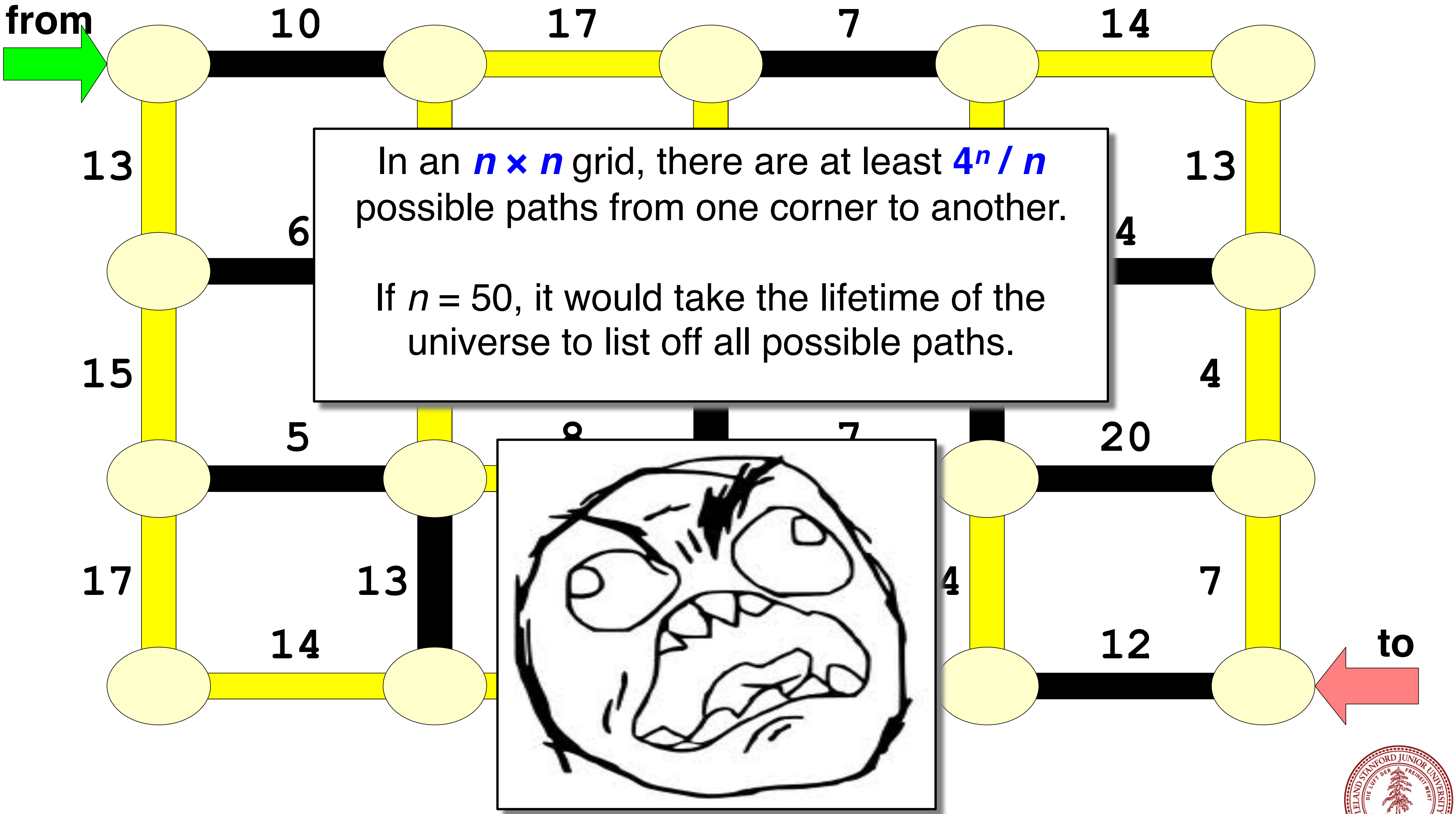


In an $n \times n$ grid, there are at least $4^n / n$ possible paths from one corner to another.



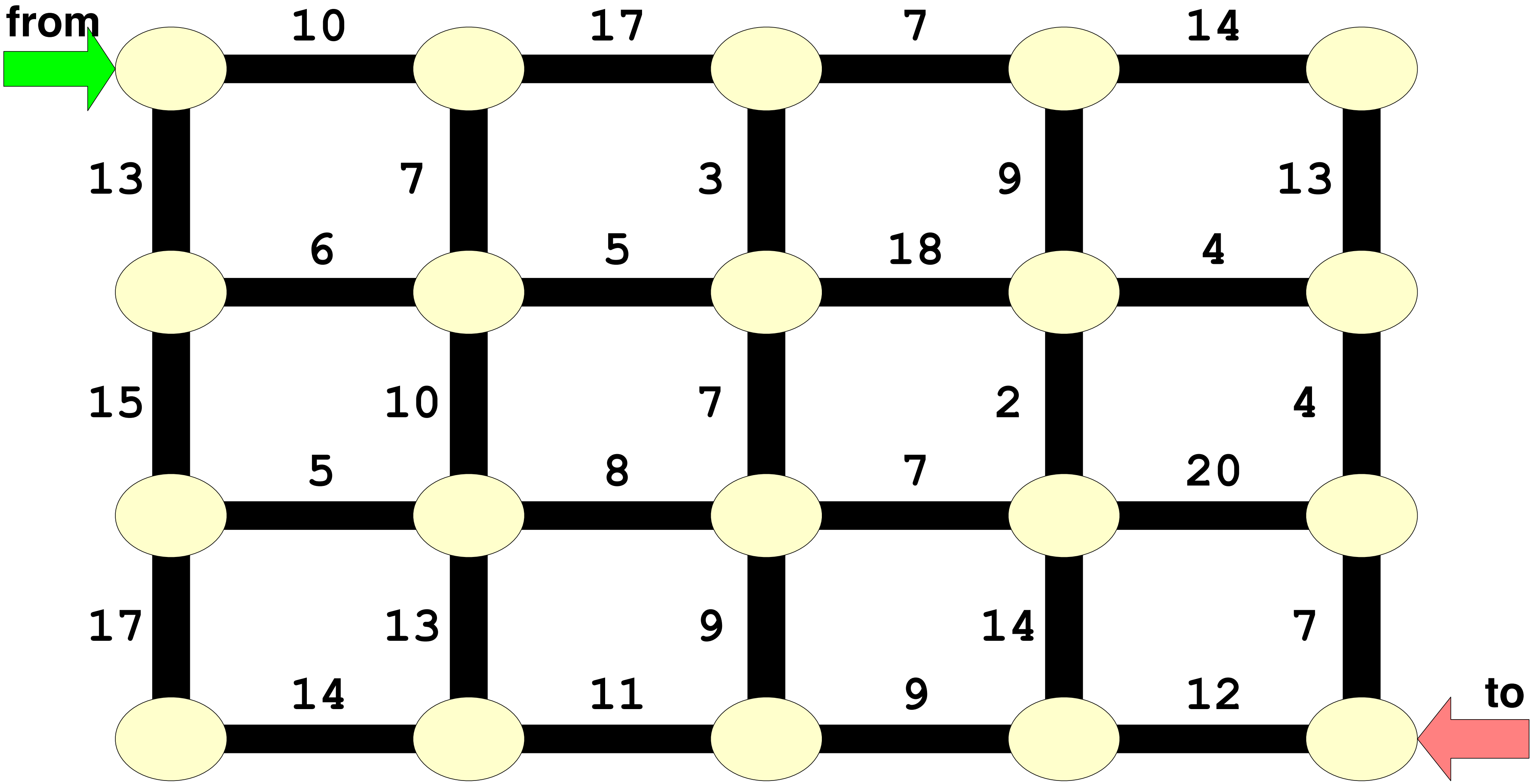
In an $n \times n$ grid, there are at least $4^n / n$ possible paths from one corner to another.

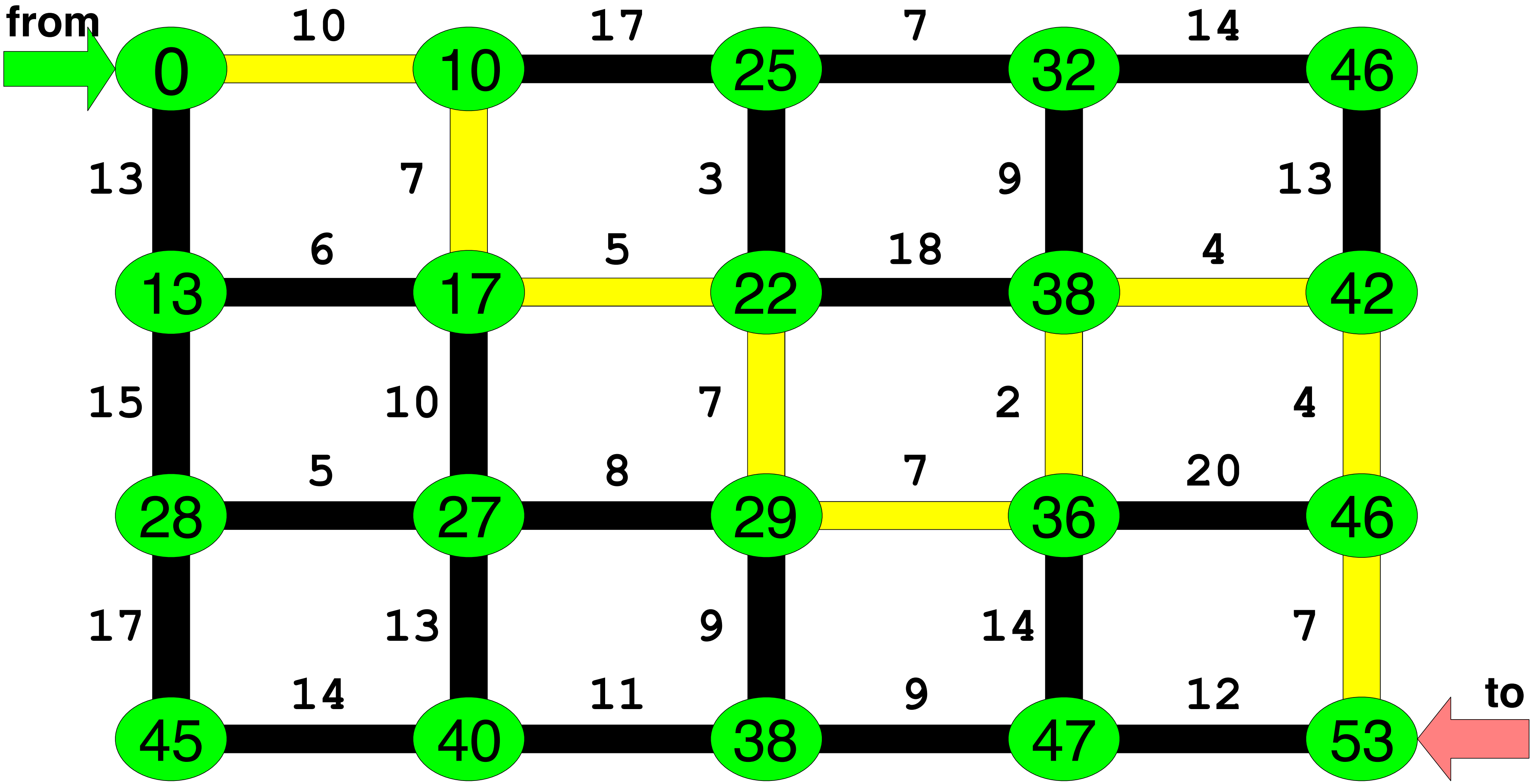
If $n = 50$, it would take the lifetime of the universe to list off all possible paths.

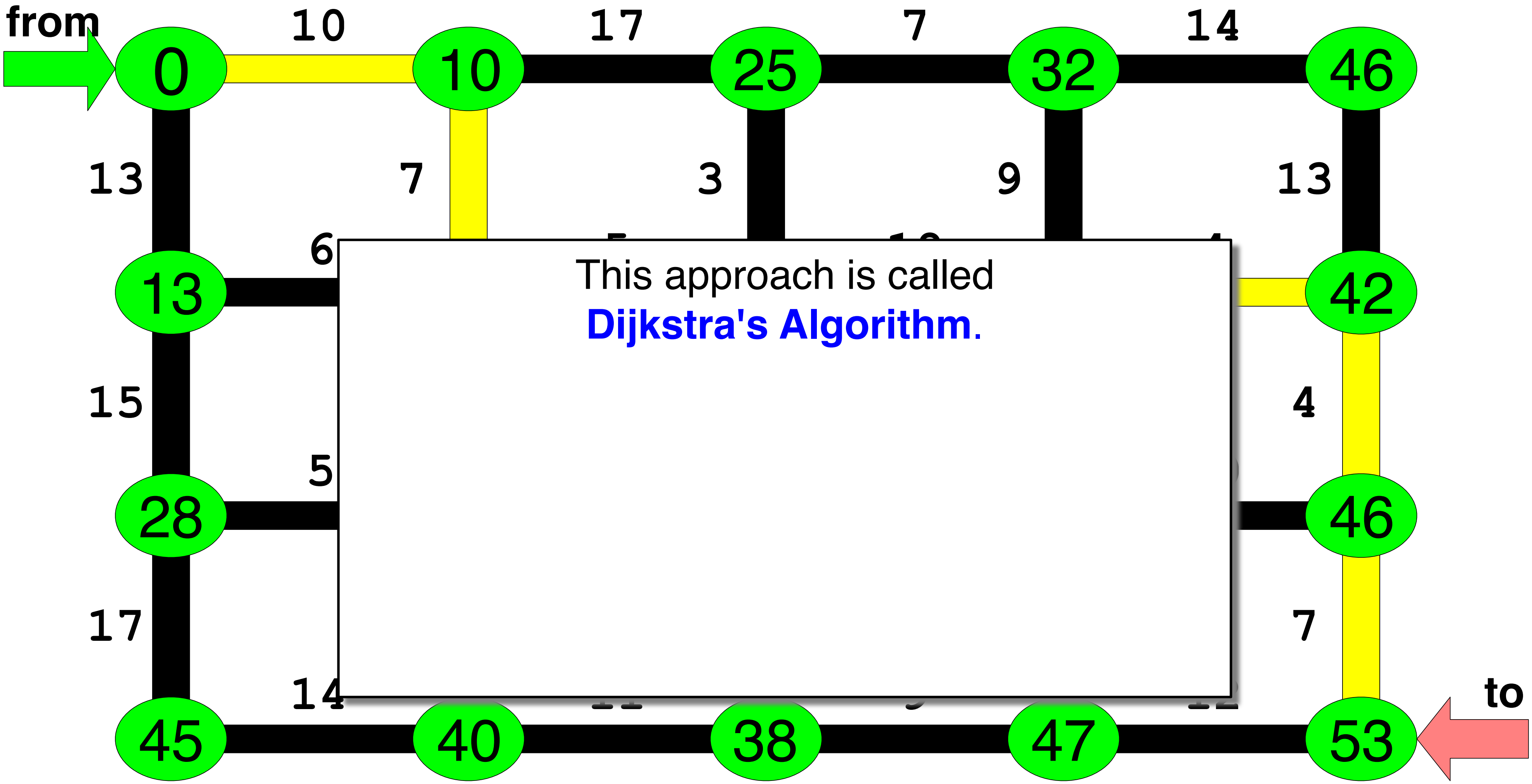


In an ***$n \times n$*** grid, there are at least ***$4^n / n$*** possible paths from one corner to another.

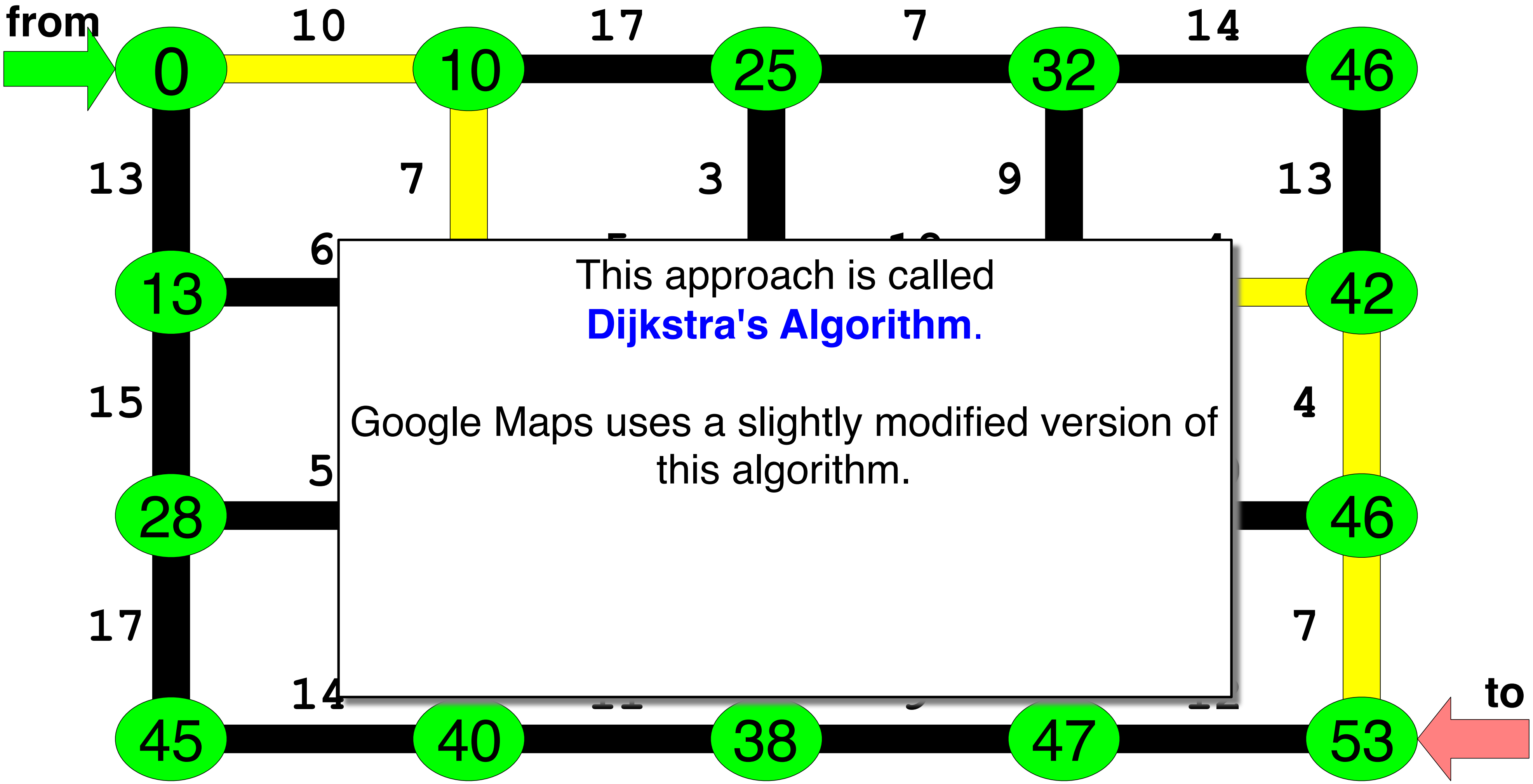
If $n = 50$, it would take the lifetime of the universe to list off all possible paths.


from
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to


from
10
17
7
14
0
10
25
32
46
13
7
3
9
13
6
5
18
4
13
17
22
38
42
15
10
7
2
4
5
8
7
20
28
27
29
36
46
17
13
9
14
7
14
11
9
12
to
45
40
38
47
53



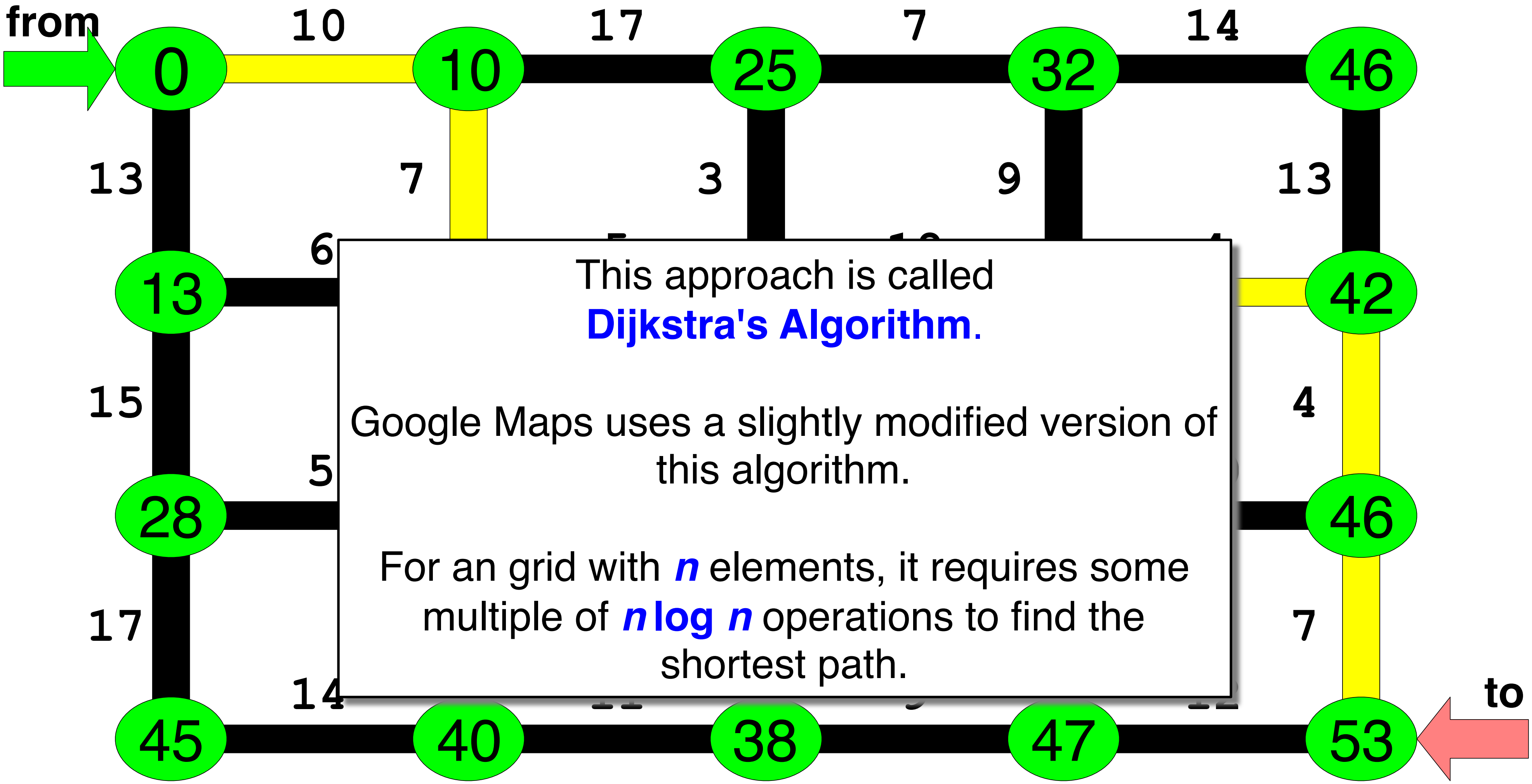
This approach is called
Dijkstra's Algorithm.



This approach is called

Dijkstra's Algorithm.

Google Maps uses a slightly modified version of this algorithm.



This approach is called
Dijkstra's Algorithm.

Google Maps uses a slightly modified version of this algorithm.

For an grid with n elements, it requires some multiple of $n \log n$ operations to find the shortest path.

Course Information

Most important!

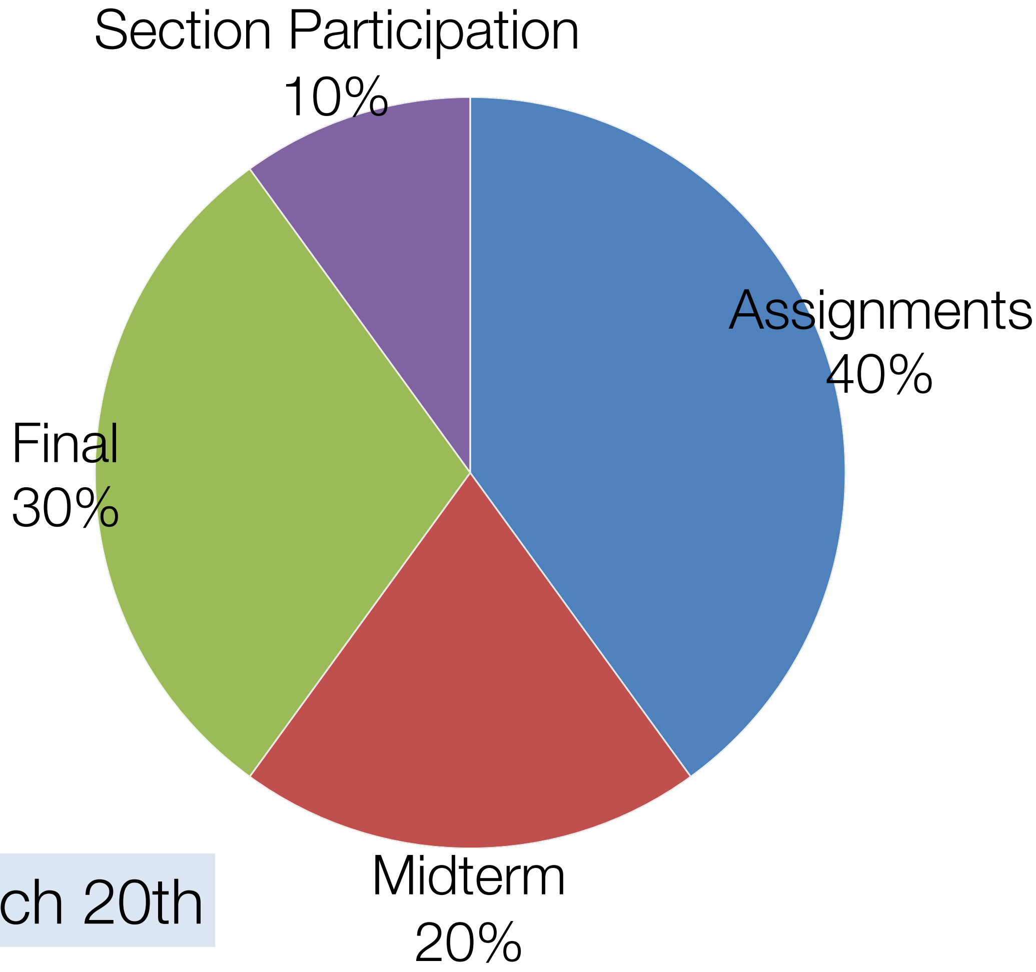
https://cs106x.stanford.edu

Course Information

2nd Most important!

https://piazza.com/stanford/winter2017/cs106x/home

Components of CS106X



Final: Monday, March 20th

Assignments in CS106X

- Due at 12:00P.M.

- Three free “late days”

- Extensions approved by Chris or Aaron.

- Graded by your section leader

- Interactive, one-on-one grading session.

- Graded on Style and Functionality.

Grading Scale

Functionality and style grades for the assignments use the following scale:

++	A submission so good it “makes you weep.”
+	Exceeds requirements.
✓+	Satisfies all requirements of the assignment.
✓	Meets most requirements, but with some problems.
✓−	Has more serious problems.
−	Is even worse than that.
−−	Better than nothing.

Sections

- Weekly 50-min section led by awesome section leaders (the backbone of the class!)
- Signups begin Thursday at 5:00pm
- Signups close Sunday at 5:00pm

You need to ask questions if you are confused

You are here only to learn. Your intelligence is unquestioned.

Getting Help

1

Review Piazza

2

Go to the LaIR / OH

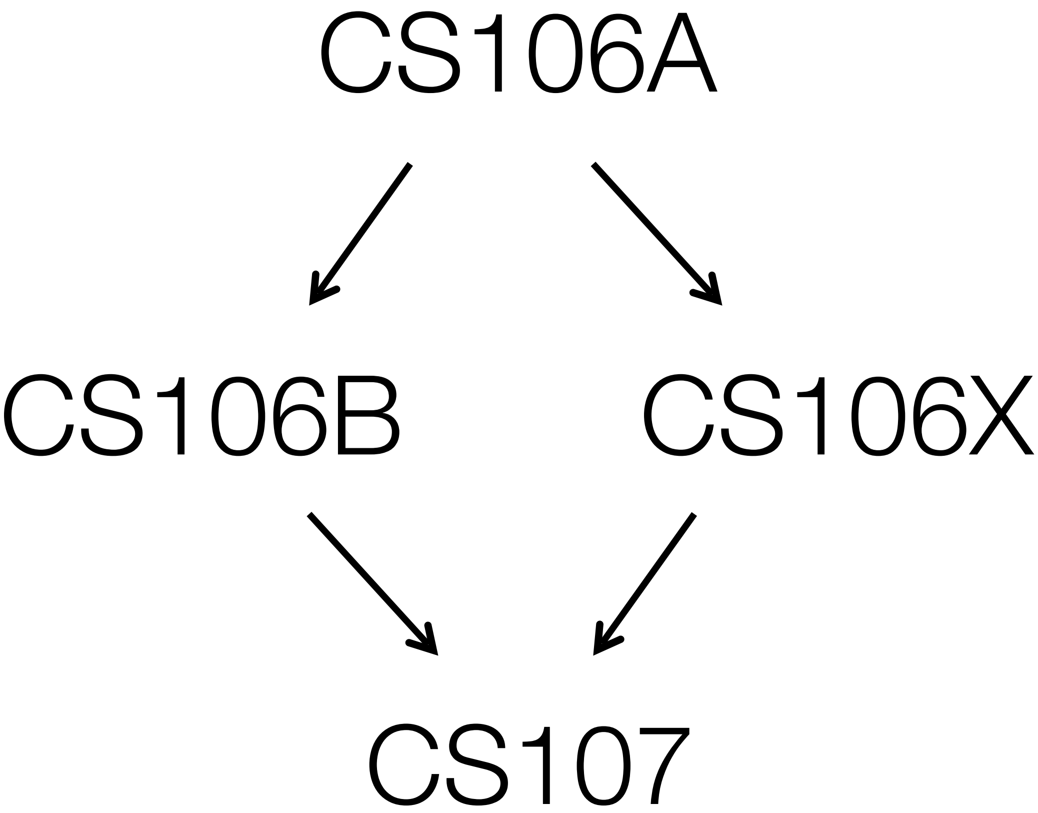
3

Contact your Section Leader

4

Email Chris or Aaron

Is CS106X The Right Class?



CS106A

CS106B

CS106X

CS107

CS106L

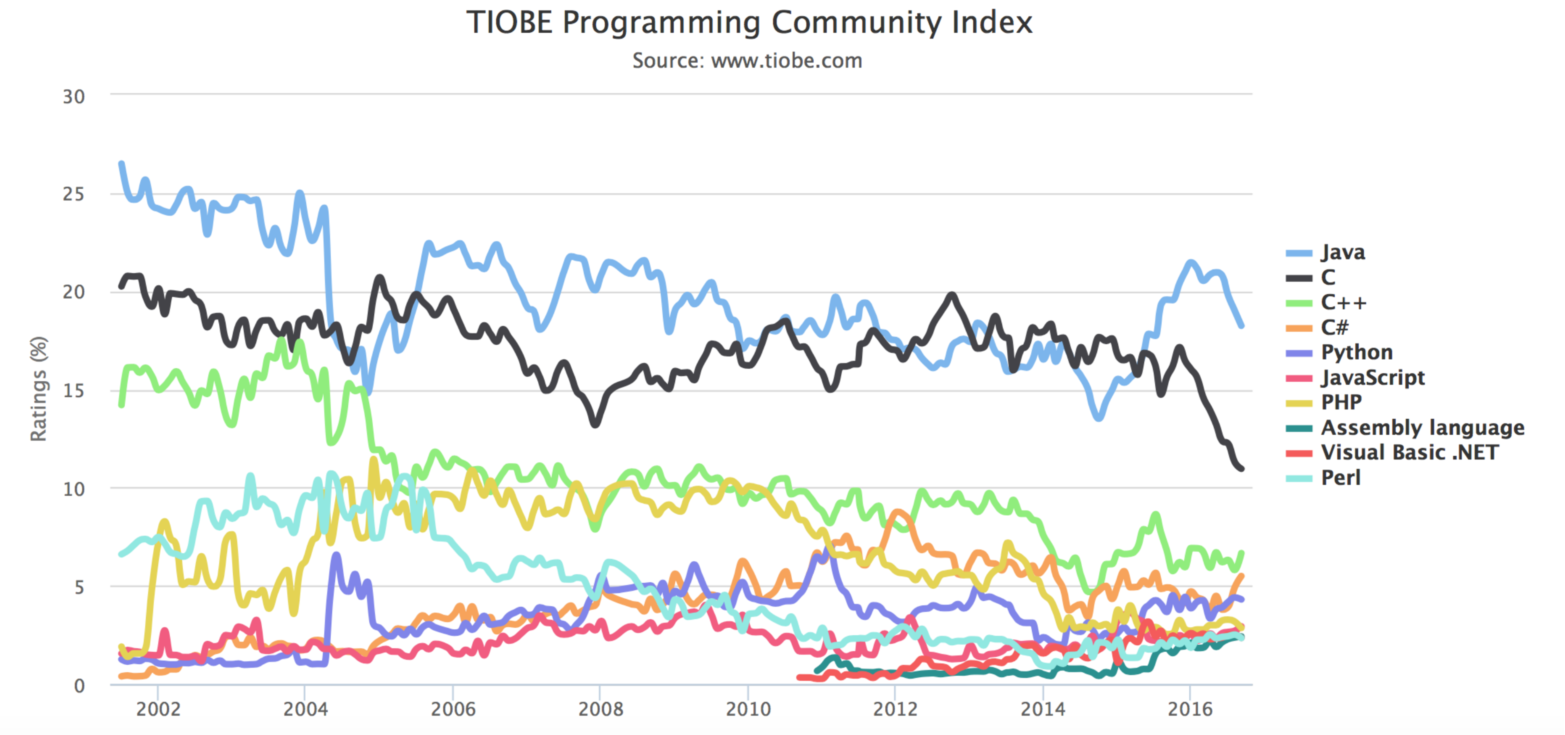
CS106S

One last detail…

C++

C++

Although there are hundreds of computer languages, in CS 106X we will be using the C++ language, which is not the easiest language to learn, but it is powerful and popular (and will help you get an internship!)


TIOBE Programming Community Index

Source: www.tiobe.com

What is the most used language in programming?

Profanity!

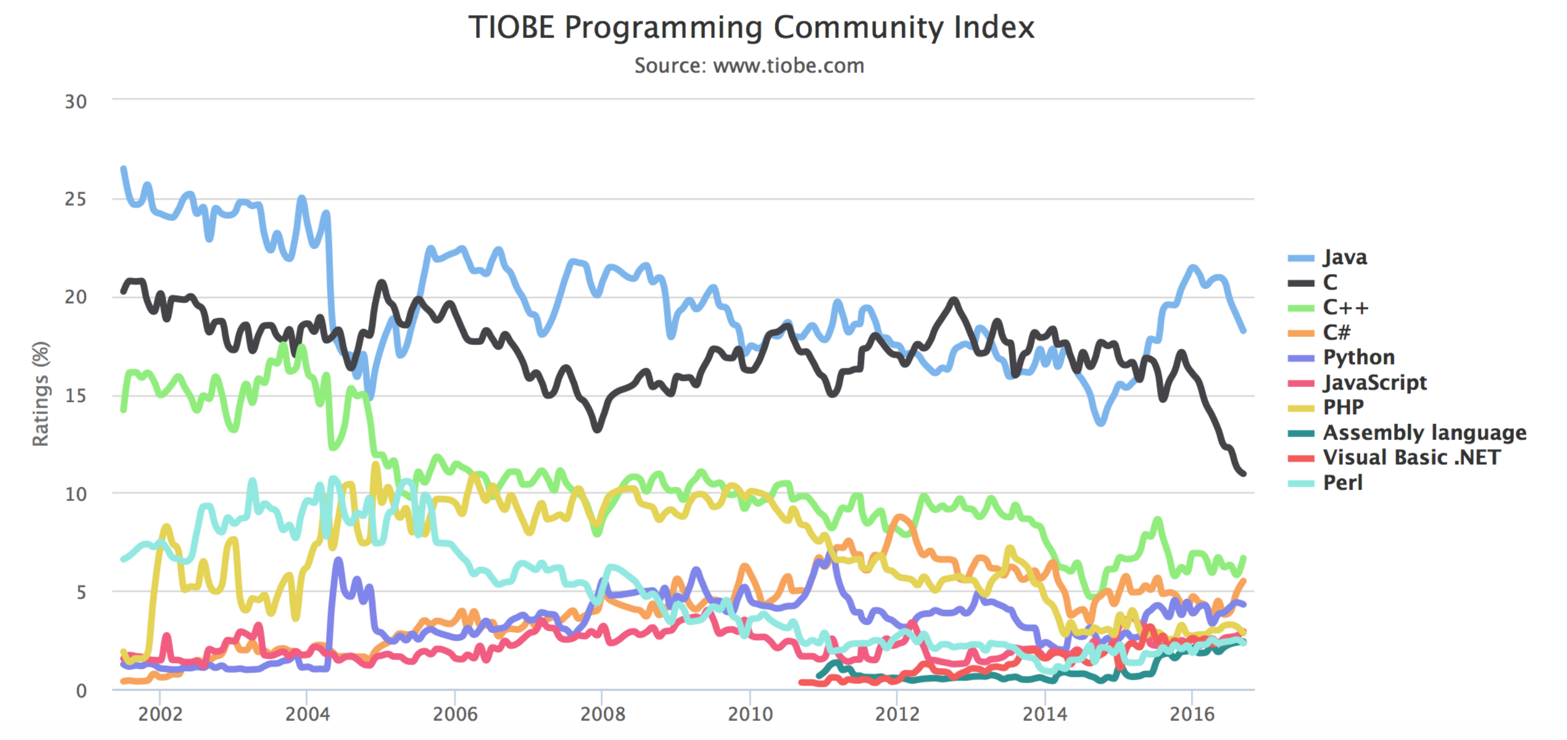
CS 106/107 Languages

The 106/107 languages:

```
106A : Java (1995)
106X : C++  (1983)
107  : C    (1972!)
```

All three languages have their syntax based on C (the good news).

All three are different enough that it does take time to learn them (the not-as-good news).



TIOBE Programming Community Index

Source: www.tiobe.com

Your First C++ Program!

As you'll find out, learning a new language when you already know a language is not really that hard, especially for "imperative" languages like Java, C++, and C (and Javascript, Python, and Ruby, etc.)

Non-imperative languages —"functional" languages — (LISP, Haskell, ML, etc.) take a completely different mentality to learn, and you'll get to those in later CS classes, like Programming Languages.

Let's write our "Hello, World!" program in C++.

Your First C++ Program!

Steps:

1. Install QT Creator (see Assignment 0!)
2. Download the example "simple-project": http://web.stanford.edu/class/cs106x/qtcreator/simple-project.zip
3. Rename the .pro file **`hello-world.pro`**
4. Open the src folder, delete **`hello.h`** and rename **`hello.cpp`** to **`hello-world.cpp`**
5. Open **`hello-world.pro`**
6. Click "Configure Project"
7. Open Sources->src->**`hello-world.cpp`**
8. Delete everything!
9. Now we're ready to code…

Your First C++ Program!

```
// Our first C++ program!

// headers:
#include <iostream>
#include "console.h" // Stanford library

using namespace std;

// main
int main()
{
    cout << "Hello, World!" << endl;
    return 0;
}
```

To compile: Select Build->Build Project "hello-world" (or ⌘-B or Alt-B)

To run in "Debug" mode: Select Debug->Start Debugging->Start Debugging (or ⌘-Y or Alt-Y)

You should see a console window pop up that says, "Hello, World!"

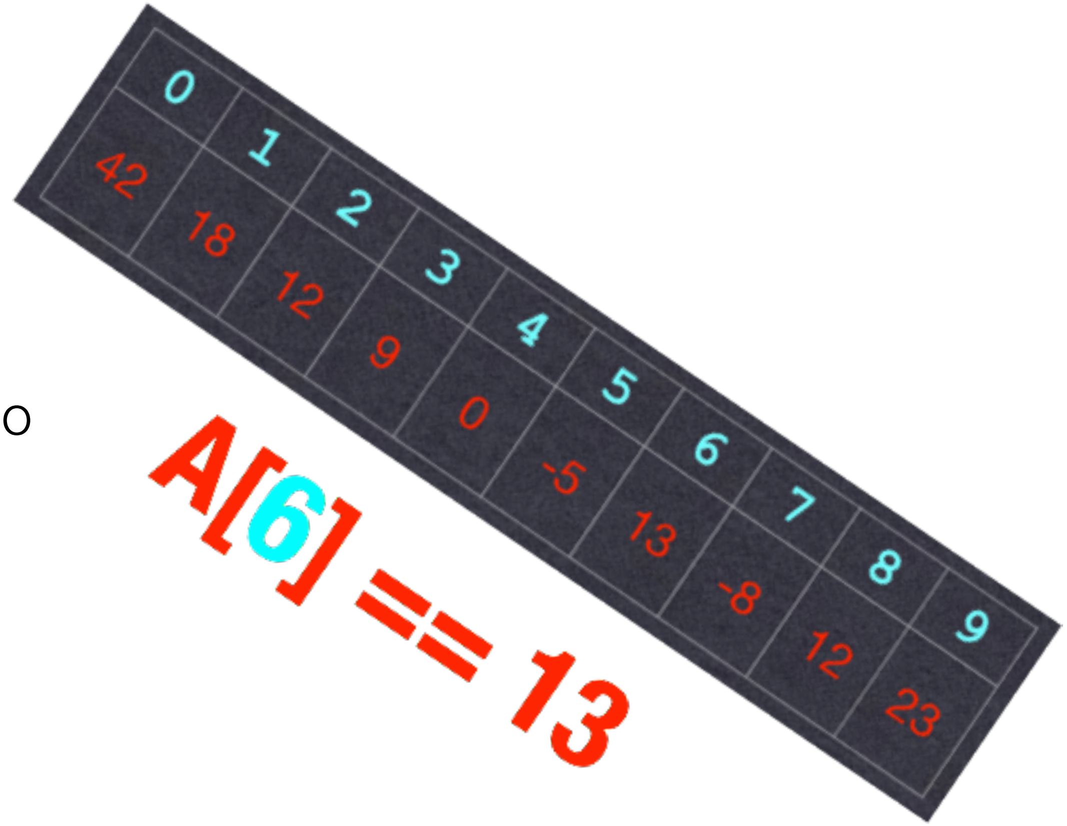
Your *Second* C++ Program!

Because this is 106X, let's write a more advanced program, one that creates a list, and populates the list with 100,000 even integers from 0 to 198,998.

You'll see that this looks strikingly familiar to Java, with a few C++ differences.

The list object we will use is called a "Vector," which is very similar to a Java ArrayList.

For time reasons, we'll just write it in the same hello-world.cpp file.



0	1	2	3	4	5	6	7	8	9
42	18	12	9	0	-5	13	-8	12	23

A[6] == 13

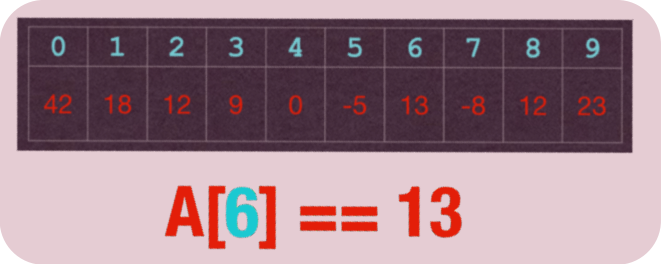
Your *Second* C++ Program!

```
// Populate a Vector

// headers:
#include <iostream>
#include "console.h" // Stanford library
#include "vector.h" // Stanford library

using namespace std;

const int NUM_ELEMENTS = 100000;
```

(continued!)

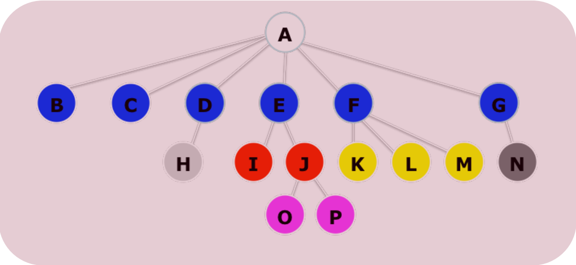
```
// main
int main()
{
    Vector<int> myList;
    cout << "Populating a Vector with
even integers less than "
         << (NUM_ELEMENTS * 2) << endl;

    for (int i=0; i < NUM_ELEMENTS; i++){
        myList.add(i*2);
    }

    for (int i : myList) {
        cout << i << endl;
    }
    return 0;
}
```

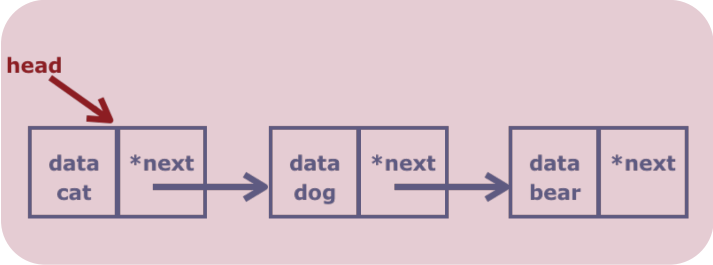
The Importance of Data Structures



0	1	2	3	4	5	6	7	8	9
42	18	12	9	0	-5	13	-8	12	23

A[6] == 13





Why Data Structures are Important

One reason we care about data structures is, quite simply, time. Let’s say we have a program that does the following (and times the results):

- Creates four “list-like” containers for data.
- Adds 100,000 elements to each container – specifically, the even integers between 0 and 198,998 (sound familiar?).
- Searches for 100,000 elements (all integers 0-100,000)
- Attempts to delete 100,000 elements (integers from 0-100,000)

What are the results?

The Importance of Data Structures

Results:

Structure	Overall(s)
Unsorted Vector	15.057
Linked List	92.202
Hash Table	0.145
Binary Tree	0.164
Sorted Vector	1.563

A factor of 103x

A factor of 636x!

Processor: 2.8GHz Intel Core i7 (Macbook Pro)
Compiler: clang++

Overall, the Hash Table "won" — but (as we shall see!) while this is generally a *great* data structure, there are trade-offs to using it.

Note: In general, for this test, we used optimized library data structures (from the "standard template library") where appropriate. The Stanford libraries are not optimized.

Full Results:

Structure	Overall(s)	Insert(s)	Search(s)	Delete(s)
Unsorted Vector	15.057	0.007	10.307	4.740
Linked List	92.202	0.025	46.436	45.729
Hash Table	0.145	0.135	0.002	0.008
Binary Tree	0.164	0.133	0.010	0.0208
Sorted Vector	1.563	0.024	0.006	1.534

Why are there such discrepancies??

Bottom line:

- **Some structures carry more *information* simply because of their design.**
- **Manipulating structures takes time**

Logistics

- Signing up for section: you must put your available times by Sunday January 15th at 5pm (opens Thursday at 5pm).
  - Go to cs198.stanford.edu to sign up.
- Qt Creator installation help: Thursday at 8pm, in Tressider (eating area). Please attempt to install Qt Creator before you arrive (see the course website for details).
  - Remember, Assignment 0 is due Friday at Noon