Welcome!

CS 106B

Programming Abstractions
Fall 2016
Stanford University
Computer Science Department

(any Freshmen?)

CS106B Instructors

Chris Piech

Chris Gregg

Chris Piech

- Career:
- Childhood through elementary: Nairobi, Kenya
- High School: Kuala Lumpur, Malaysia
- Stanford University BS and MS degree (George Forsyth Award)
- Stanford University Ph.D. in the AI Lab (Neural Networks to understand how students learn)
- Lecturer in Computer Science
- Research lab on AI for Social Good
- Personal website: http://stanford.edu/~cpiech

Chris Piech

Chris Piech at Freshmen Scavenger Hunt, 2006

Chris Gregg

- *New to Stanford!*
- 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!
- Personal website: http://ecosimulation.com/chrisgregg

CS106B Staff

Head TA: Anton Apostolatos

Section Leaders

CS106B

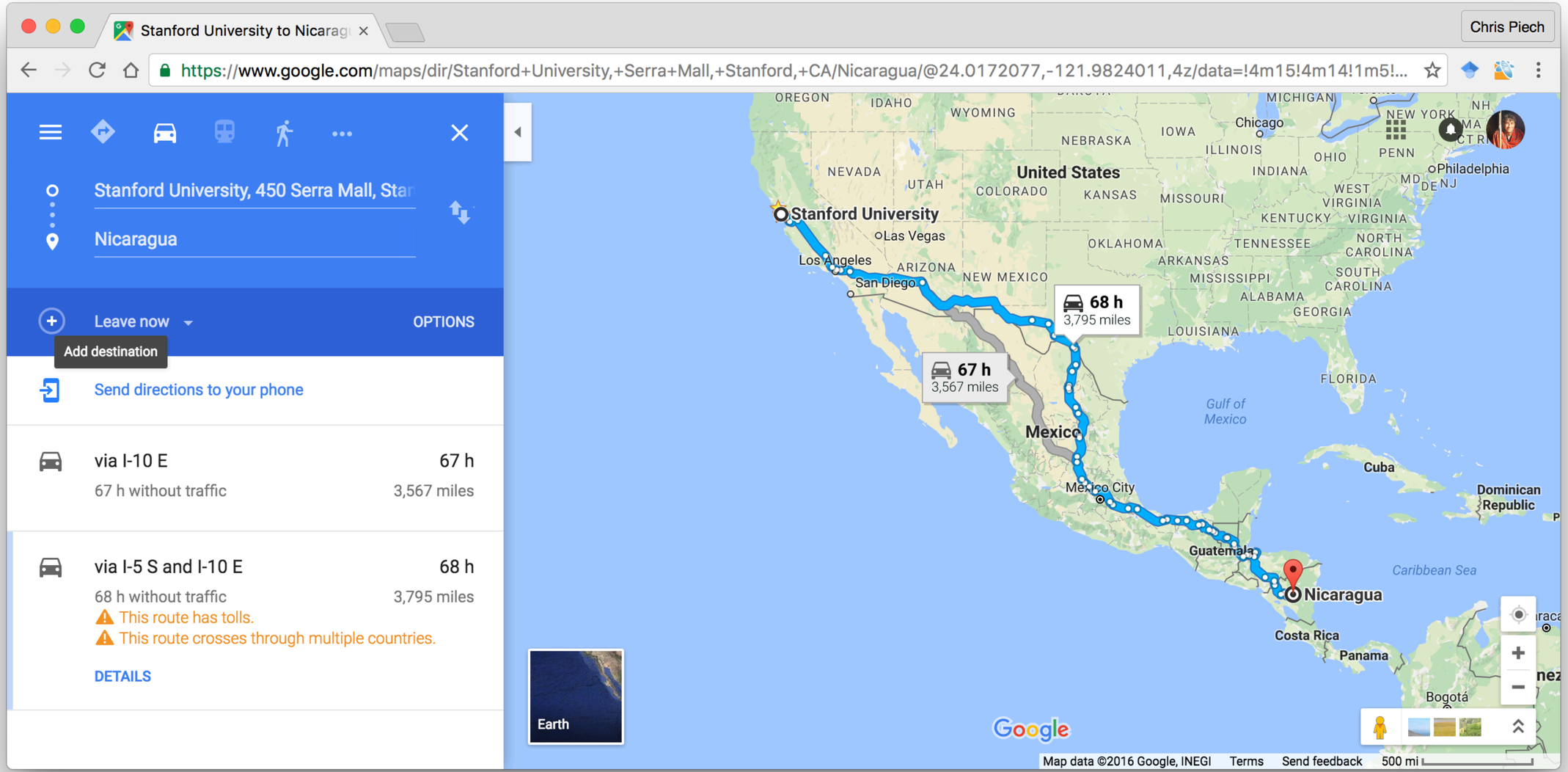
CS106B: Learn core ideas in how to model and solve complex problems with computers

Any sufficiently advanced technology is indistinguishable from magic

- Arthur Clark

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

Instantaneous Directions



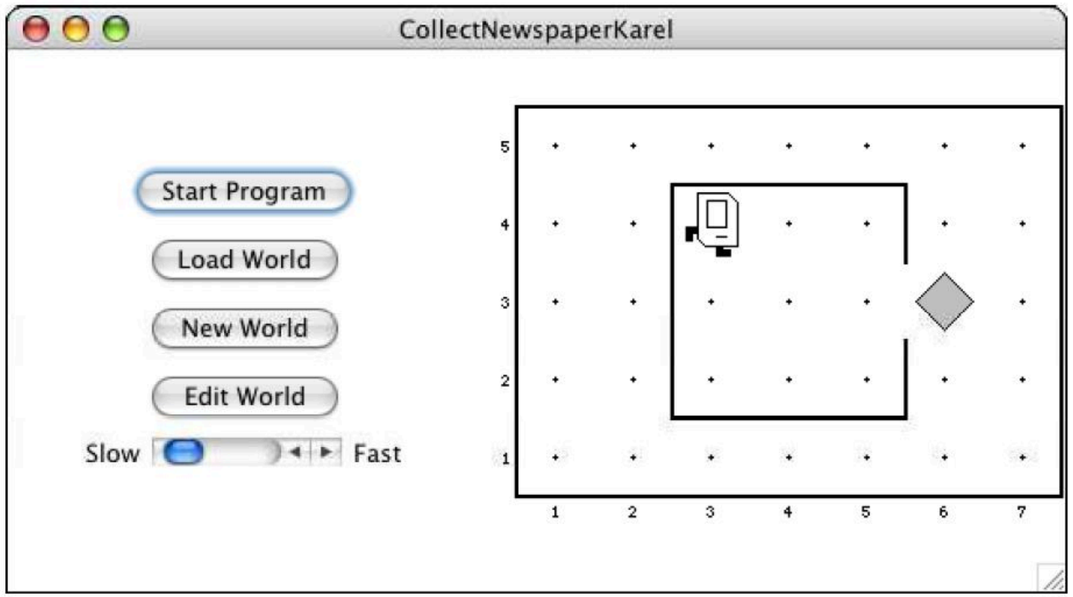
Speech Synthesis

Solution to Counterfeit Medicine

Bright Simons

How does Stanford get you there?

CS106A



In CS106A is a first course in programming, software development

Professional Sand Castle Building

There is more to learn…

Full disclosure, CS106B is necessary but not sufficient to make a self driving car ☺

Goals for CS106B

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 CS106B

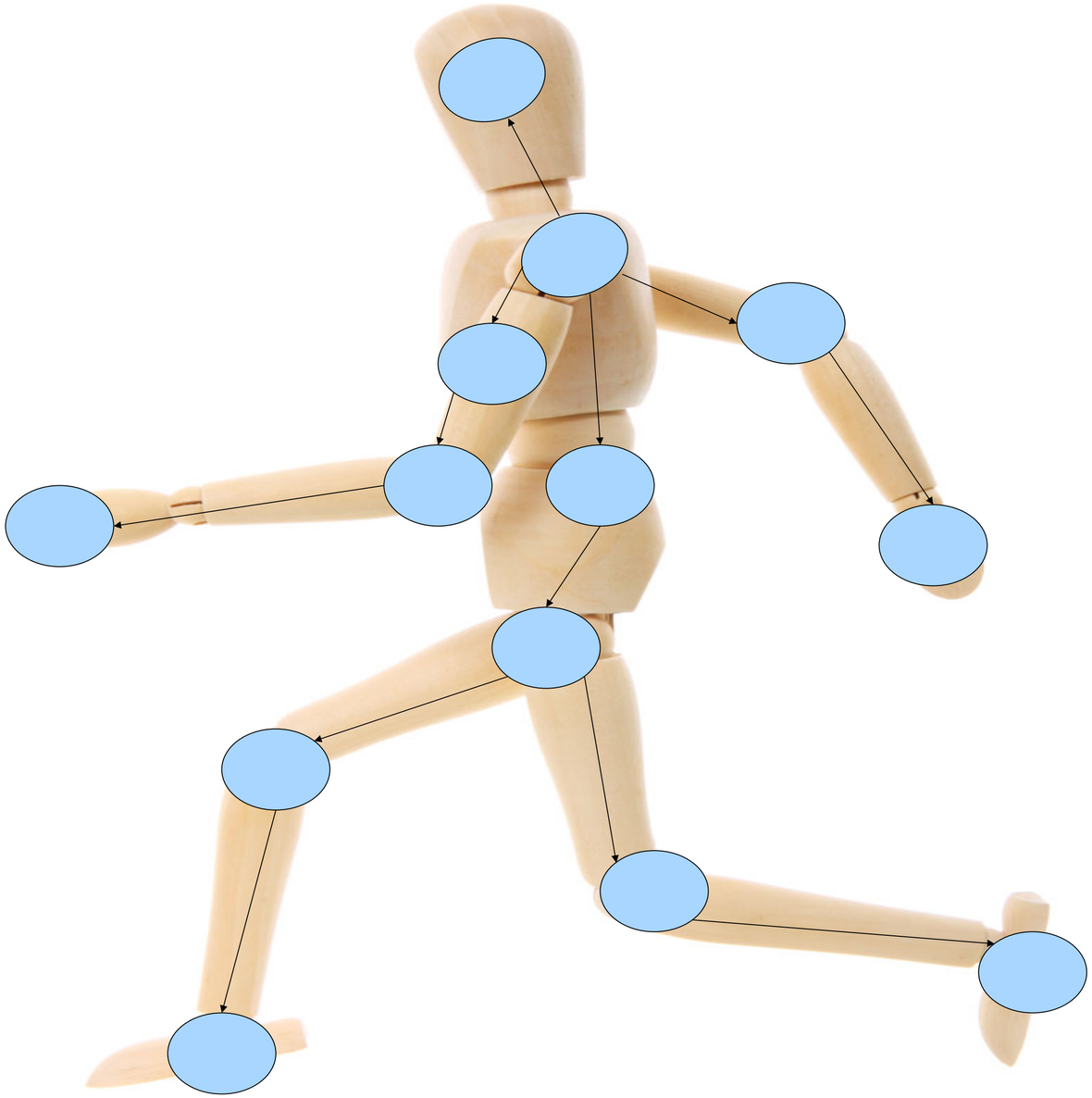
Learn core ideas in how to model and solve complex problems with computers.

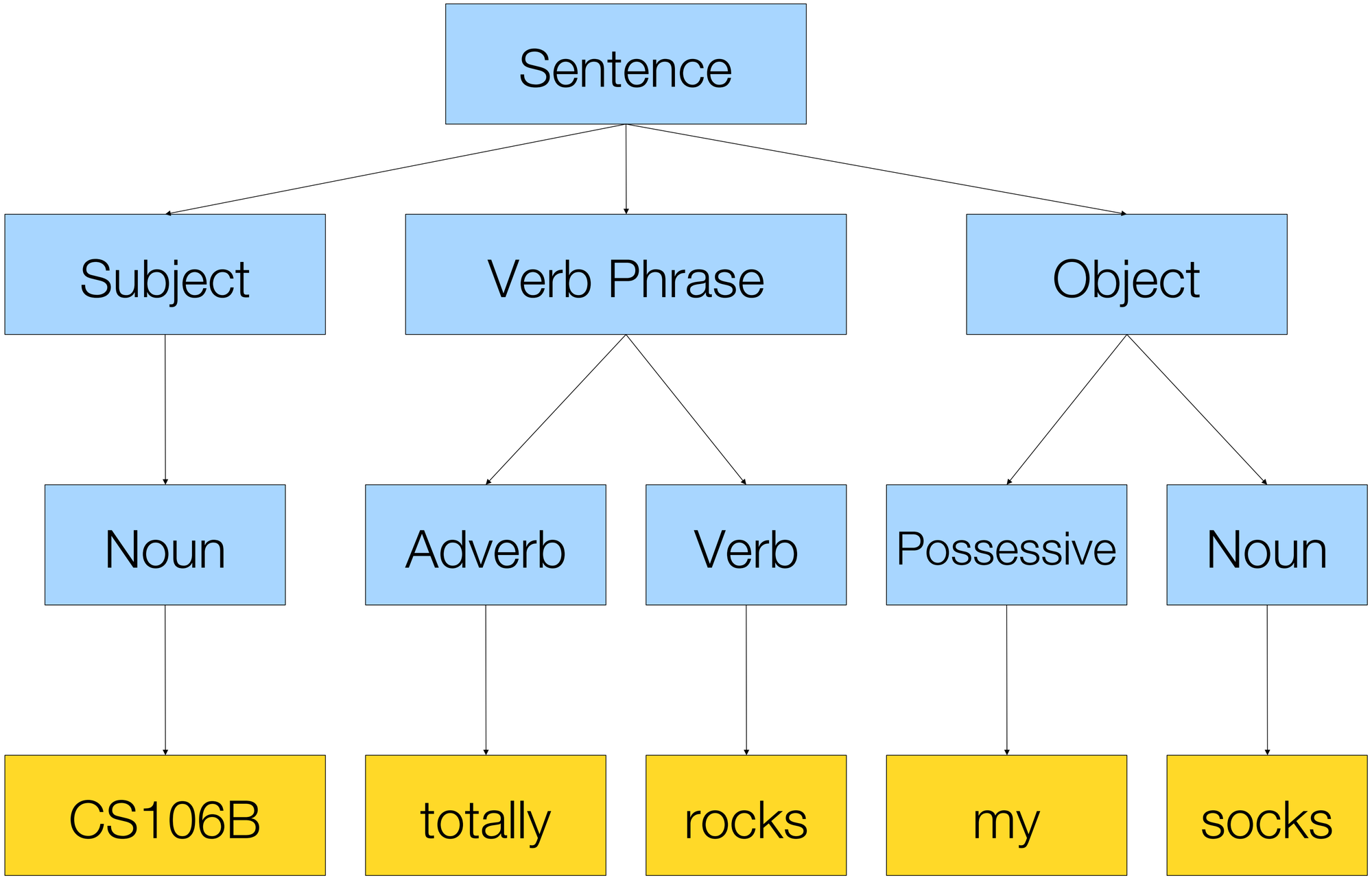
To that end:

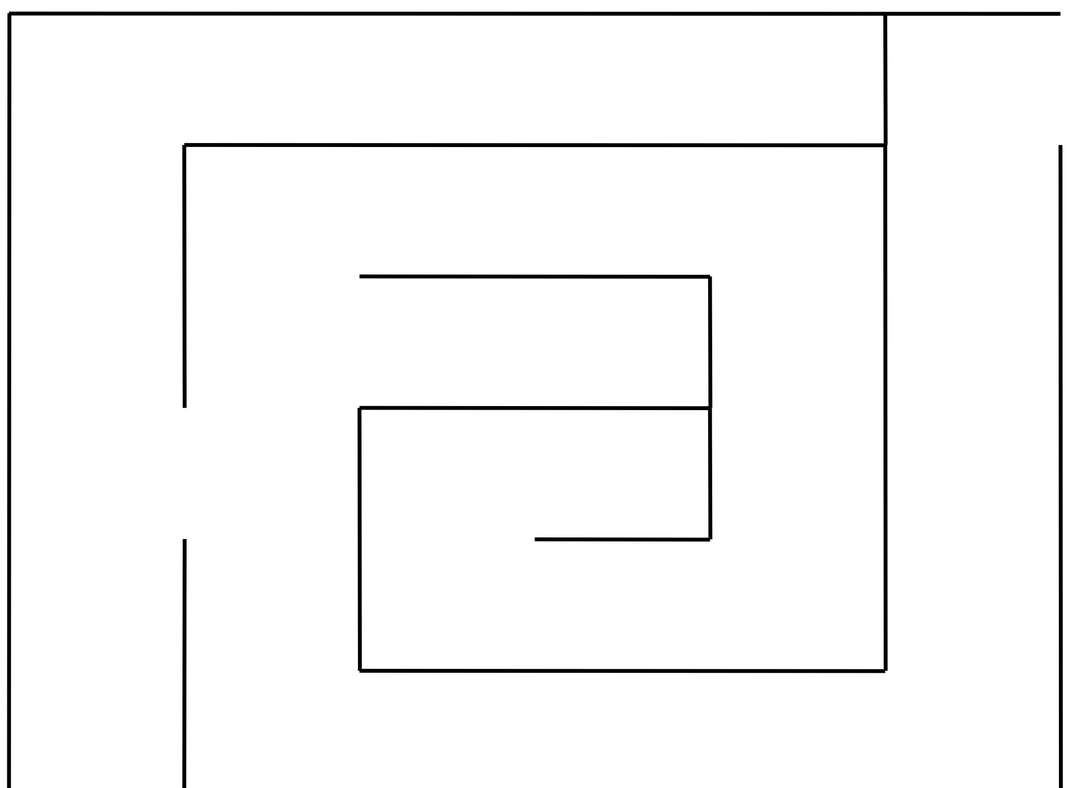
Explore common abstractions

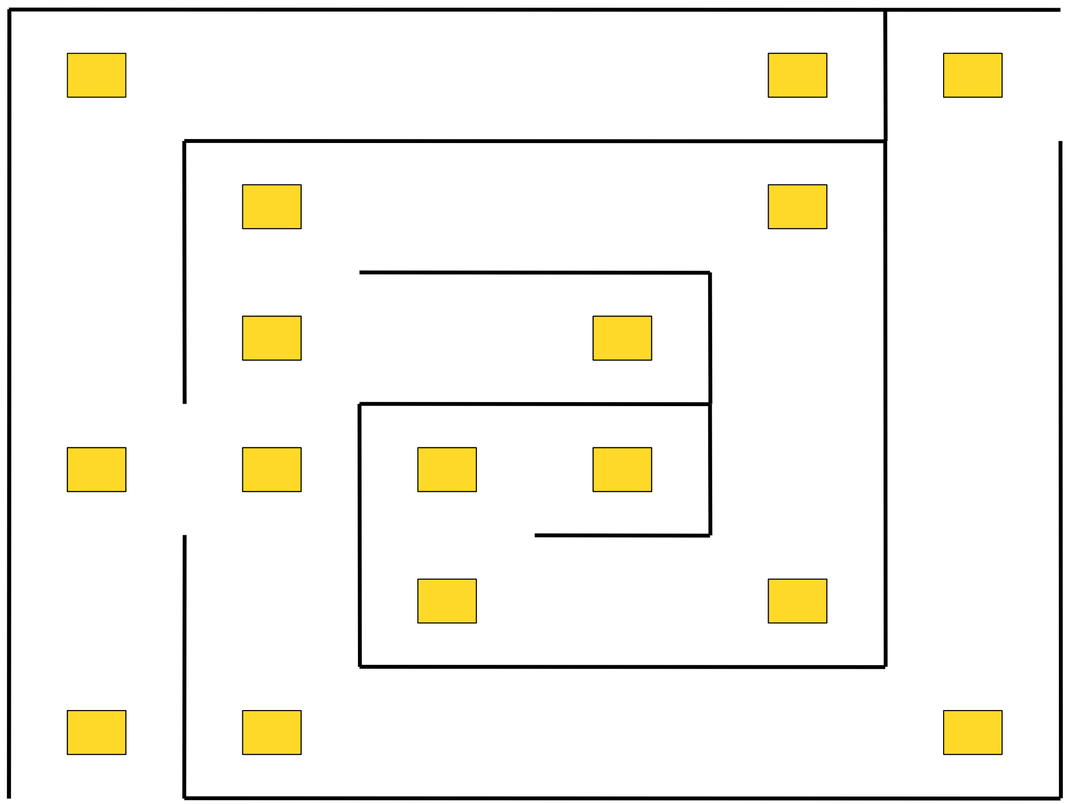
Harness the power of recursion

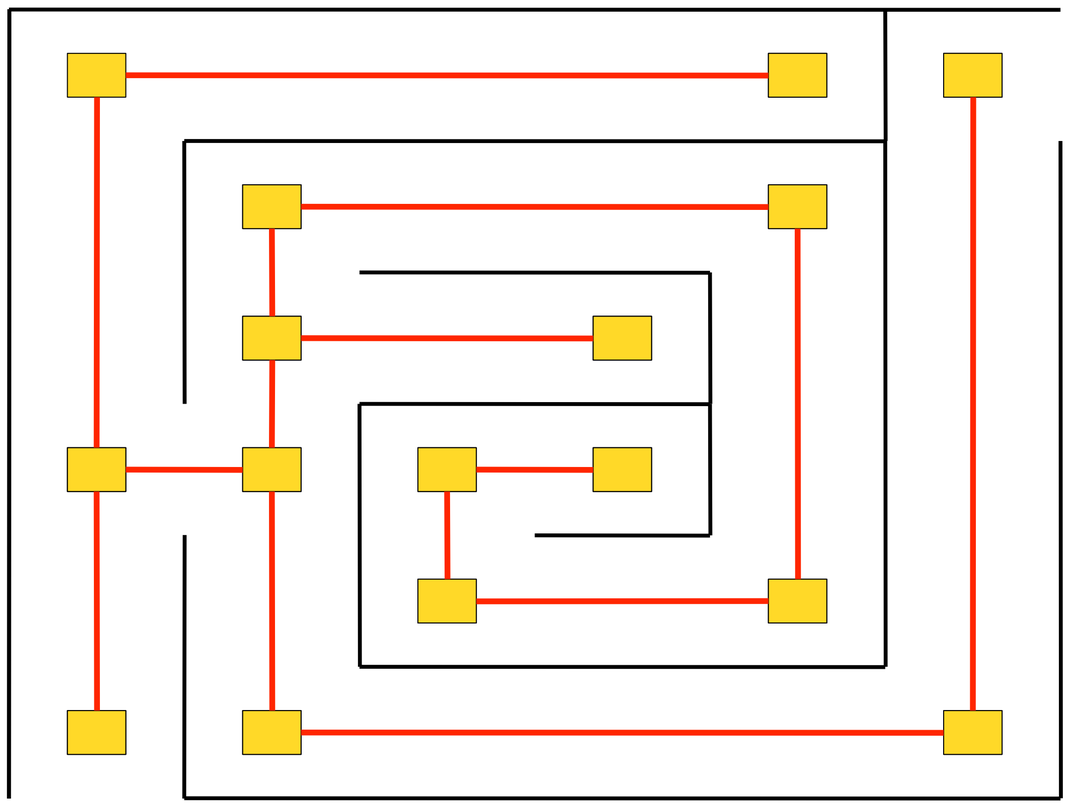
Learn and analyze efficient algorithms

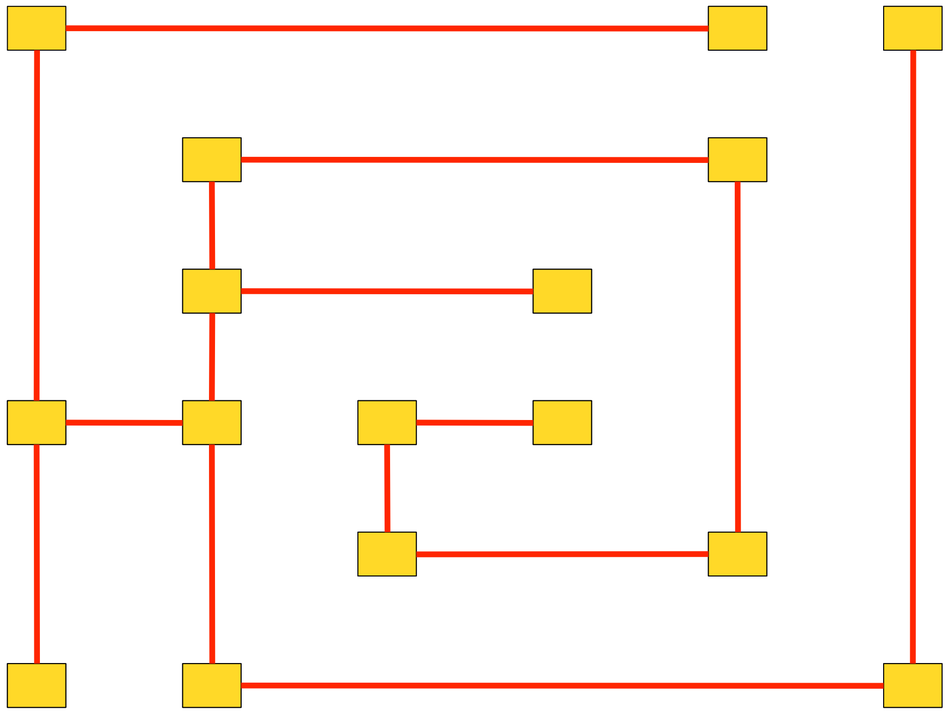


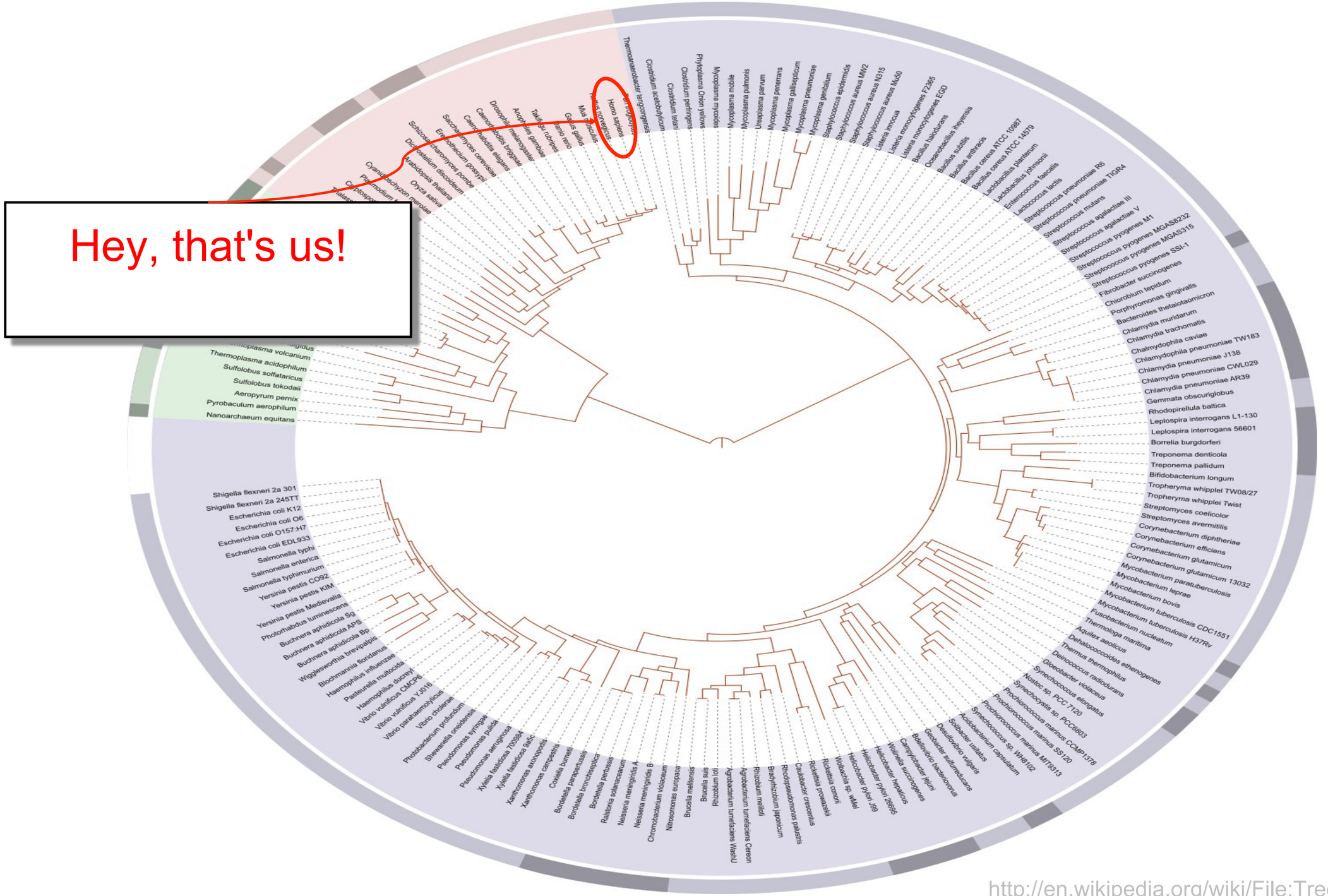

Sentence
Subject
Verb Phrase
Object
Noun
Adverb
Verb
Possessive
Noun
CS106B
totally
rocks
my
socks



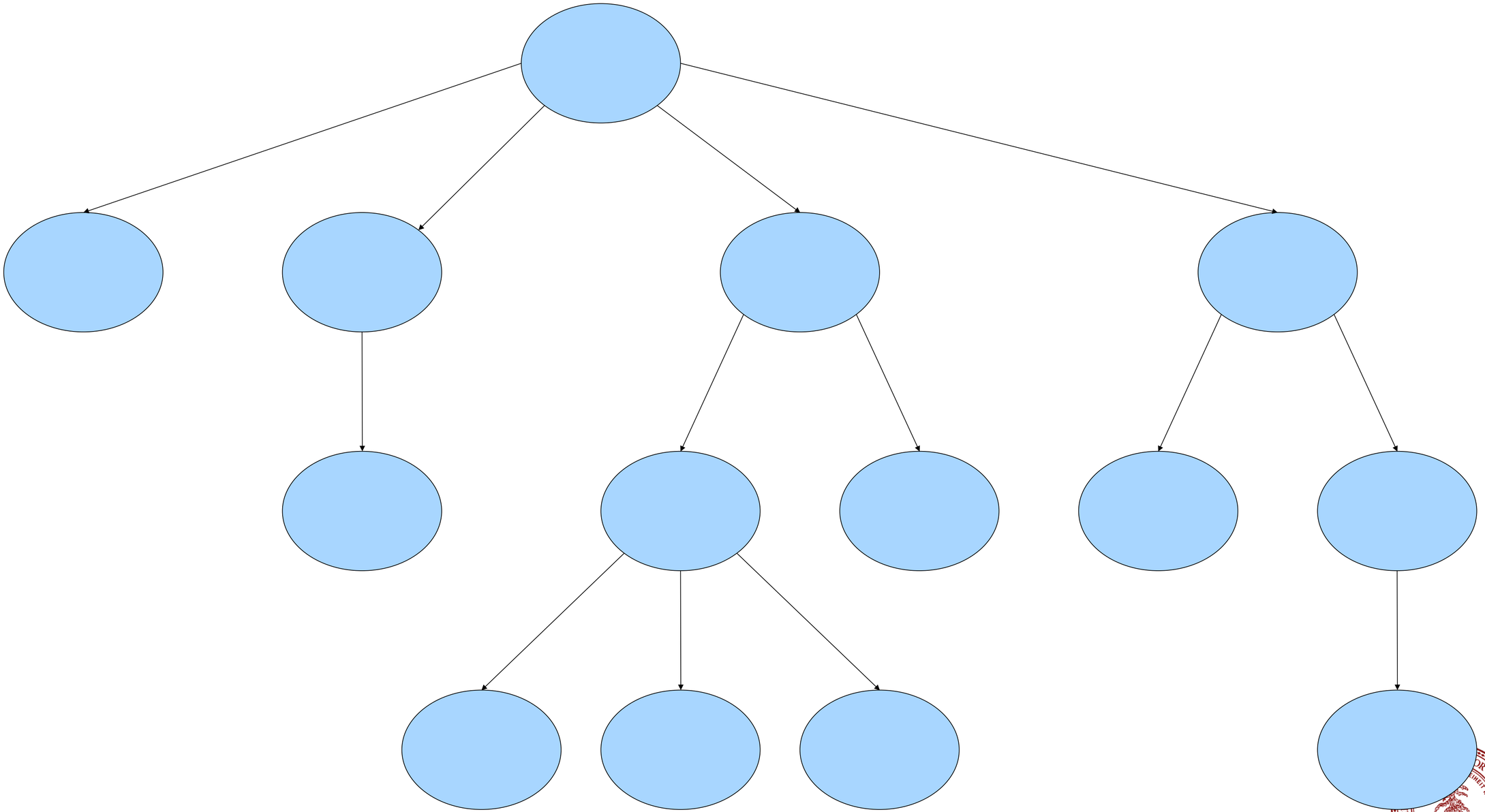








These Problems Use Same Abstraction



Building a vocabulary of **abstractions**
makes it possible to represent and solve a wider class of problems.

Goals for CS106B

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 CS106B

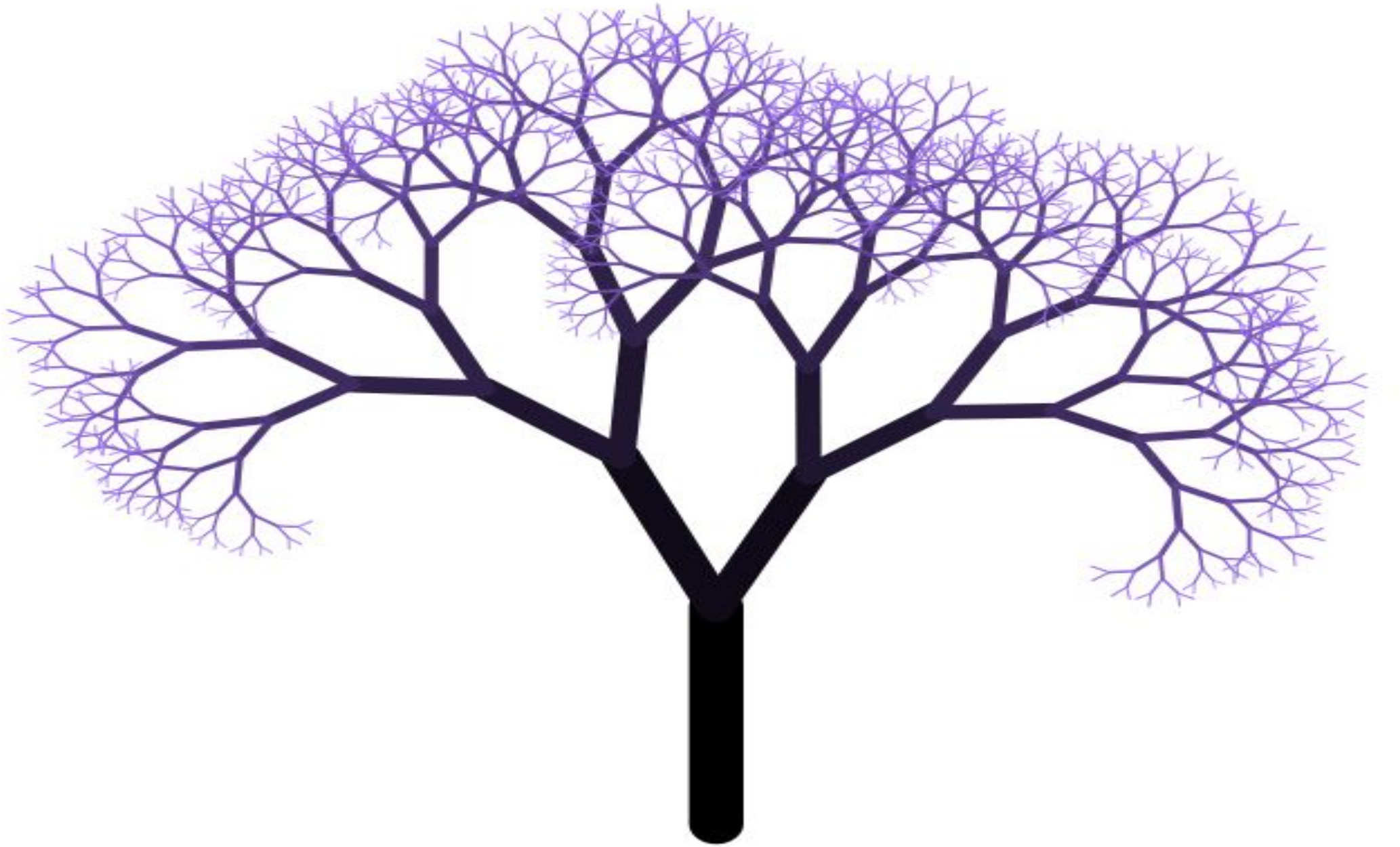
Learn core ideas in how to model and solve complex problems with computers.

To that end:

Explore common abstractions

Harness the power of recursion

Learn and analyze efficient algorithms



Goals for CS106B

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 CS106B

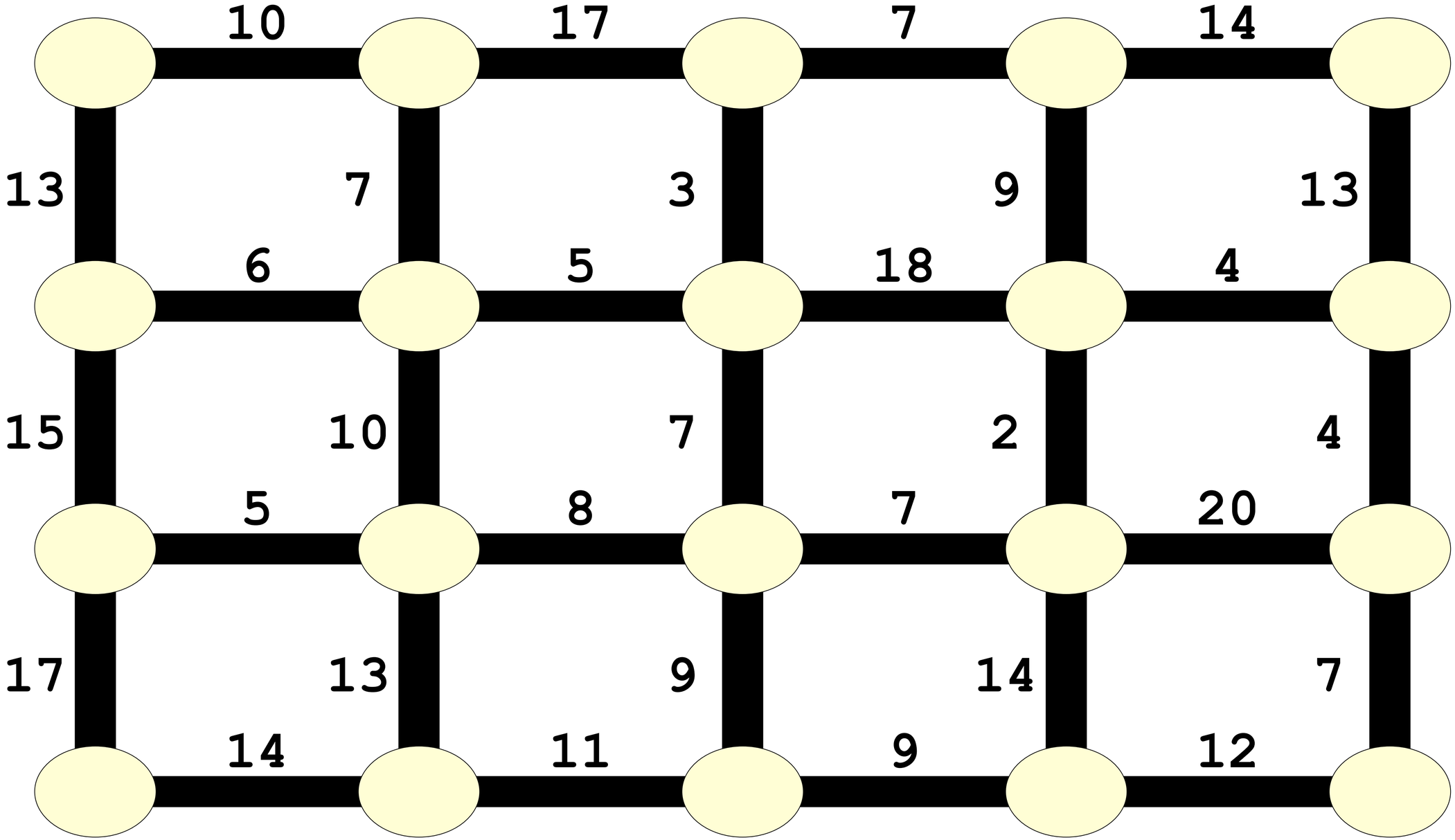
Learn core ideas in how to model and solve complex problems with computers.

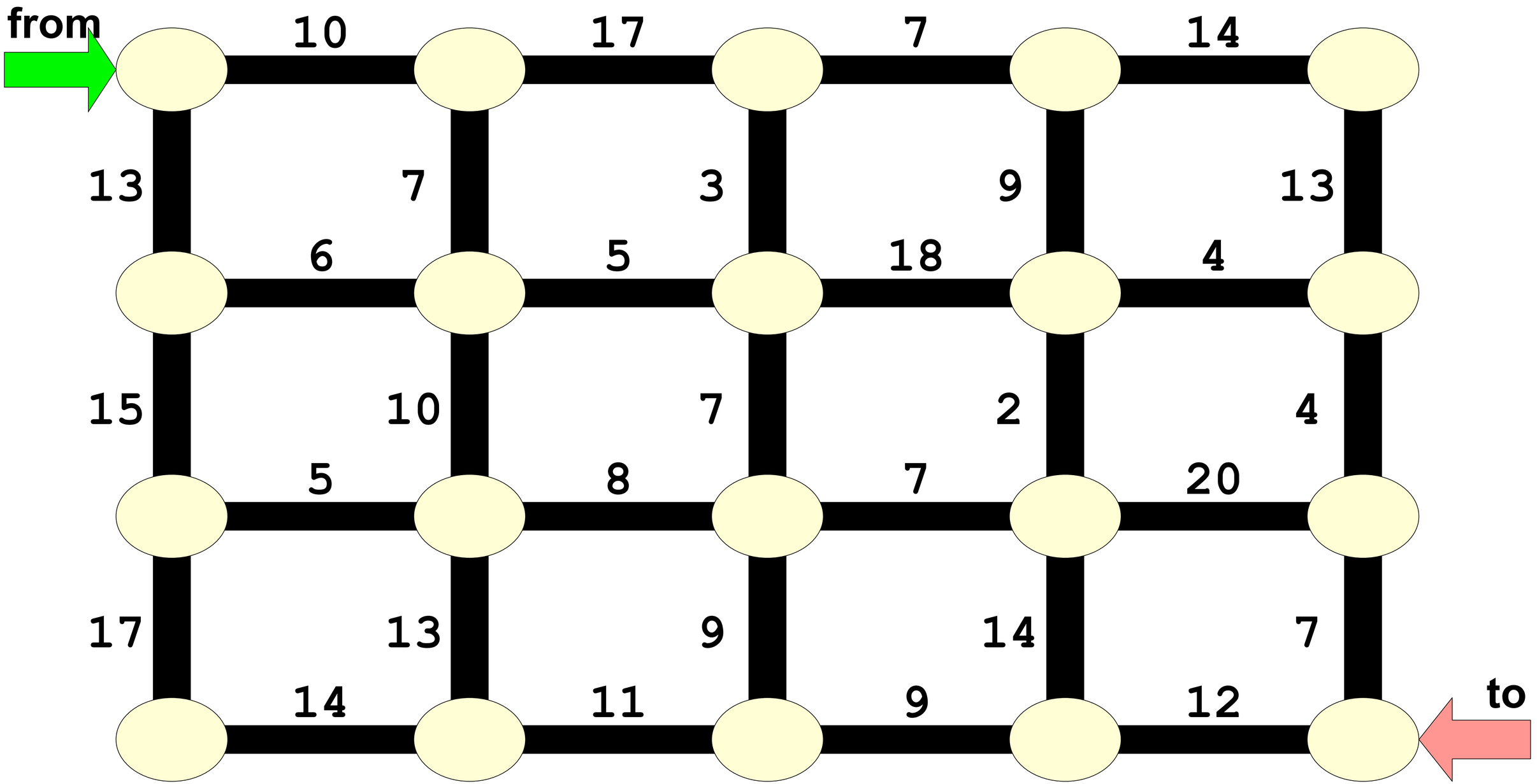
To that end:

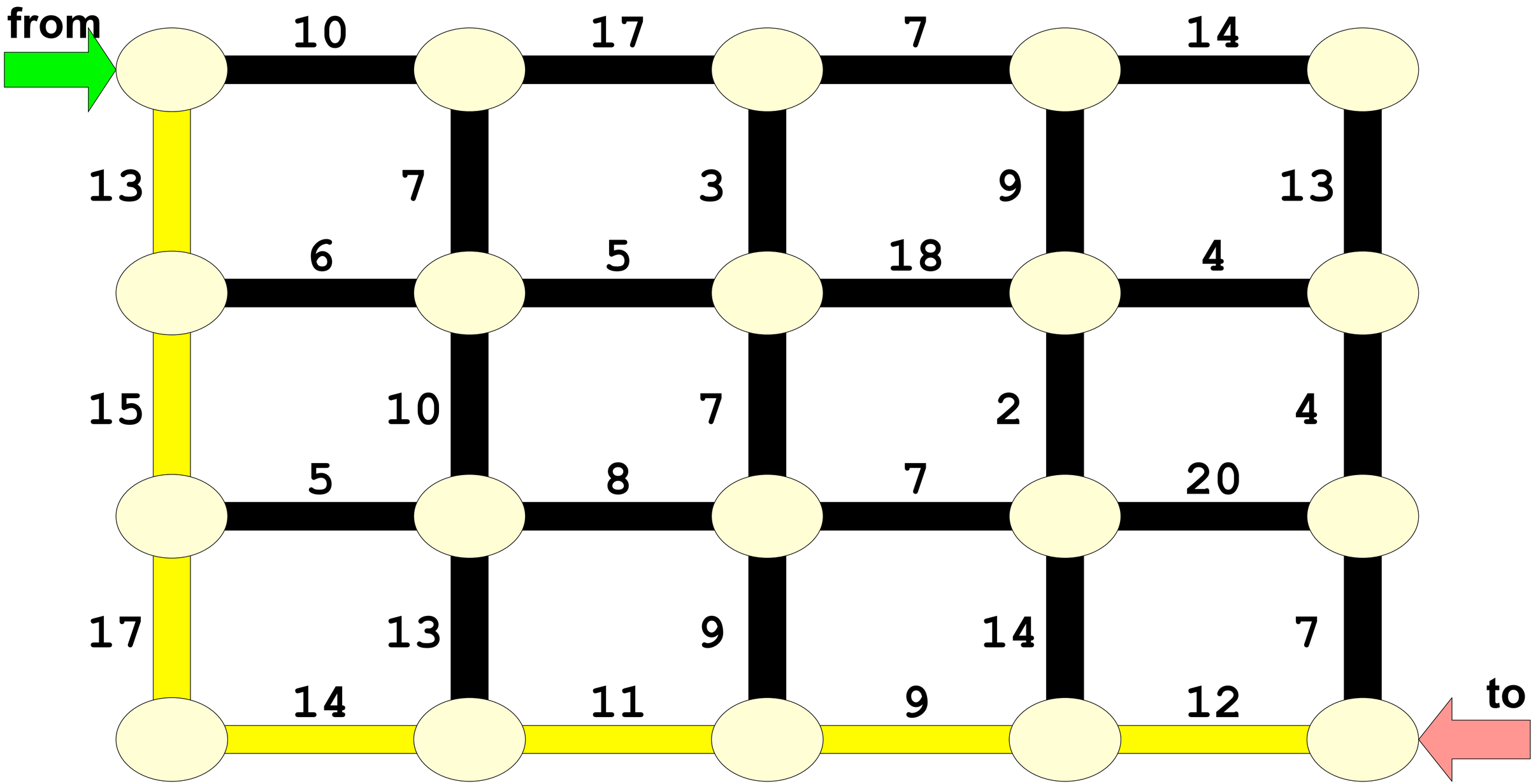
Explore common abstractions

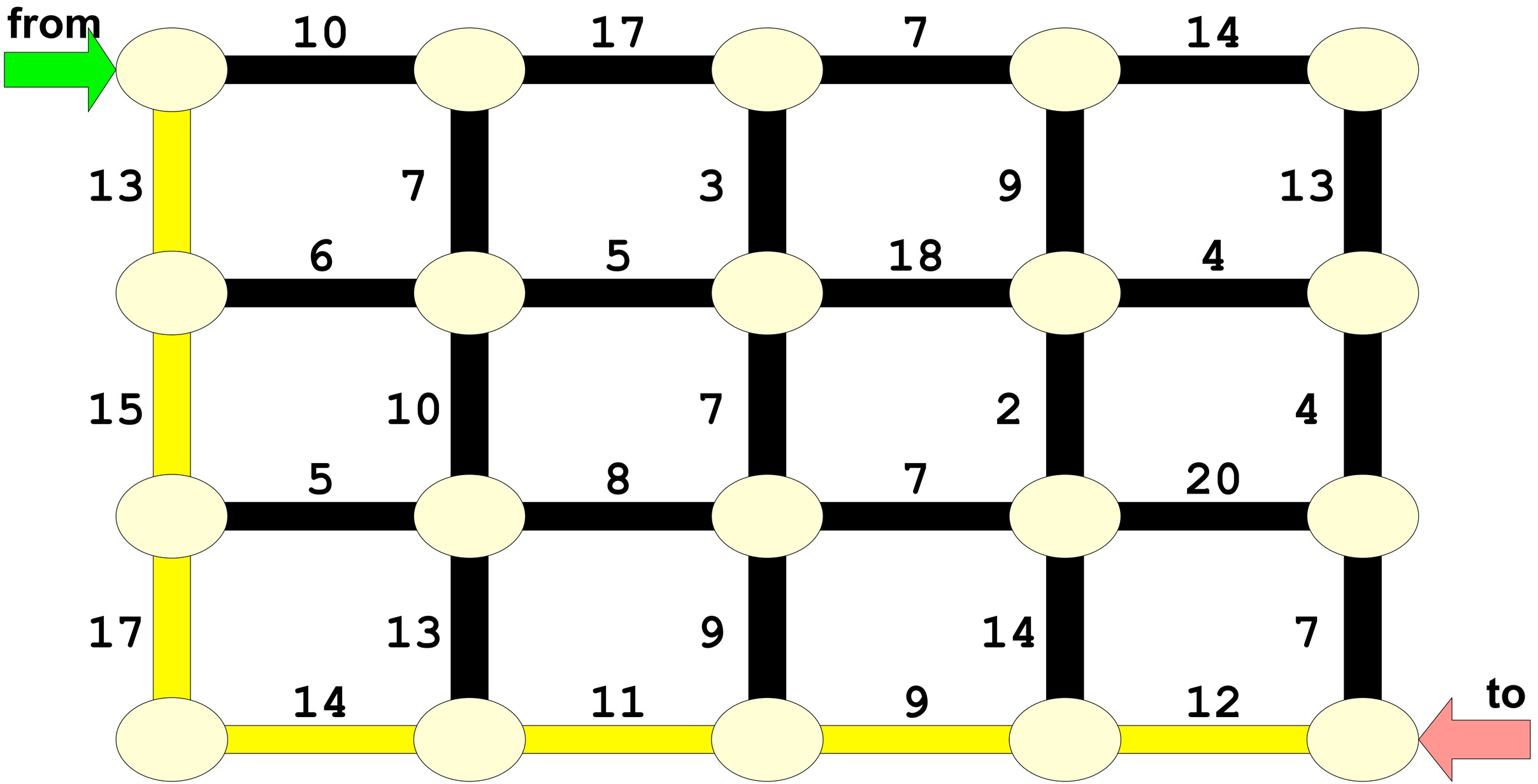
Harness the power of recursion

Learn and analyze efficient algorithms

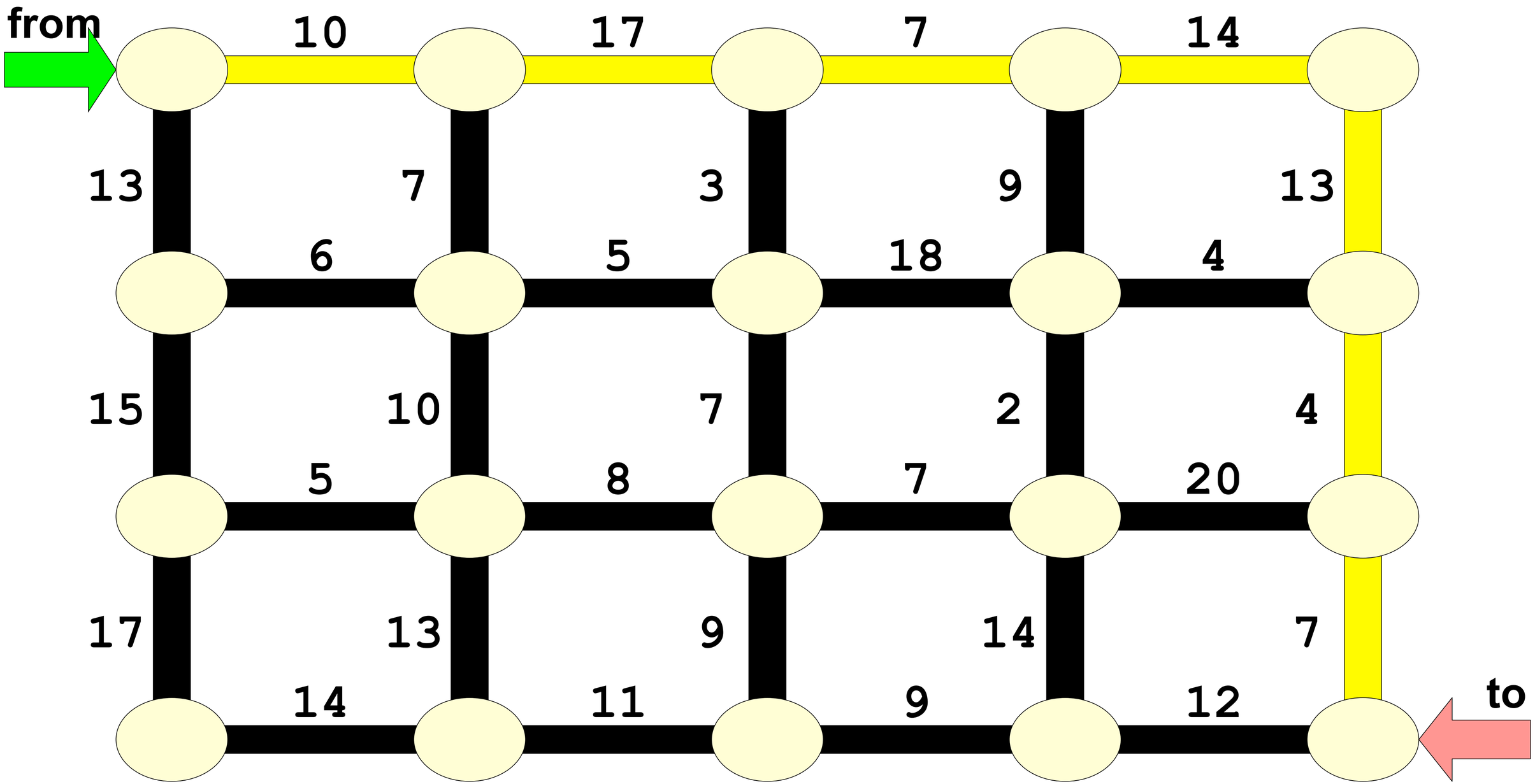

10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12

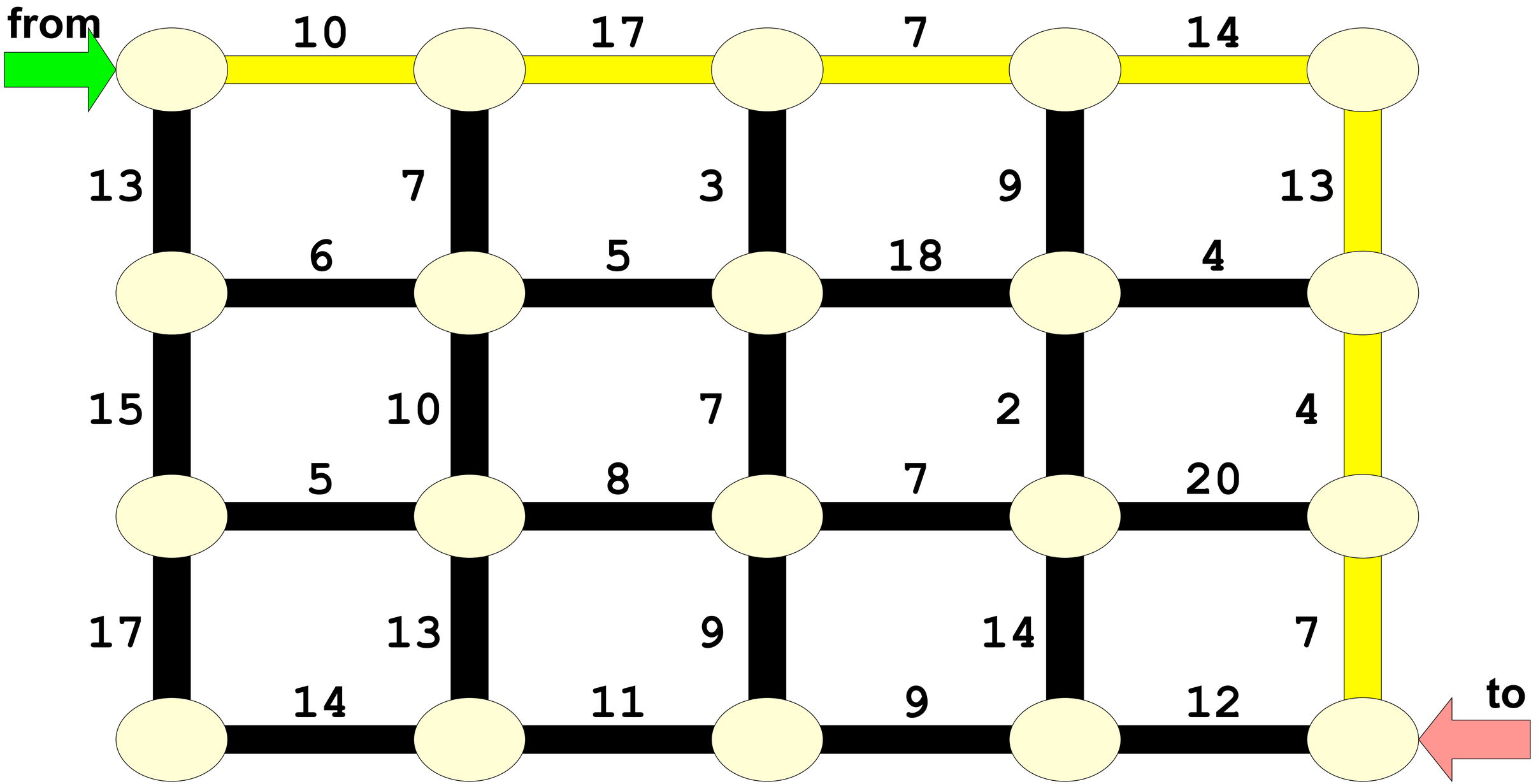

from
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to


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

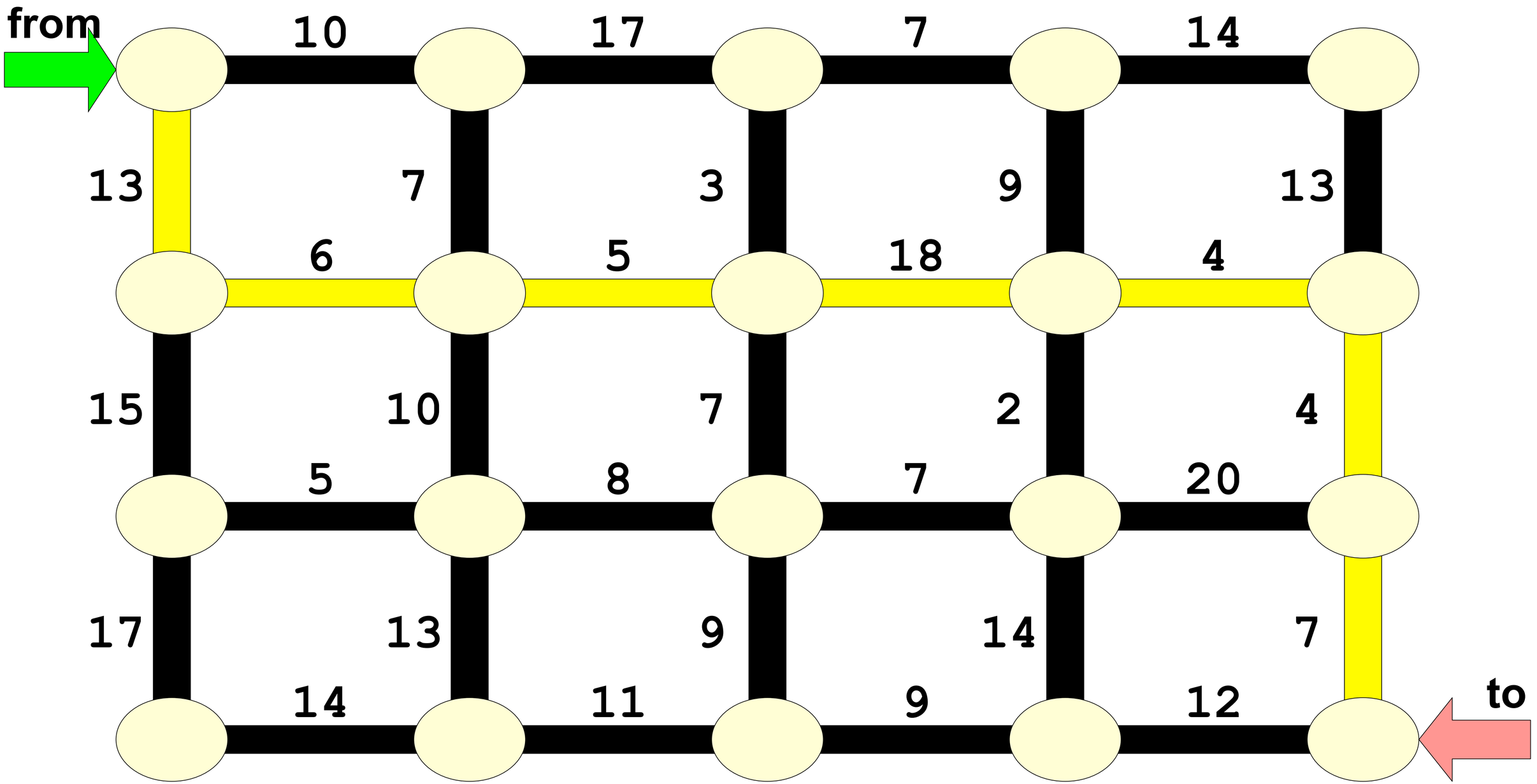


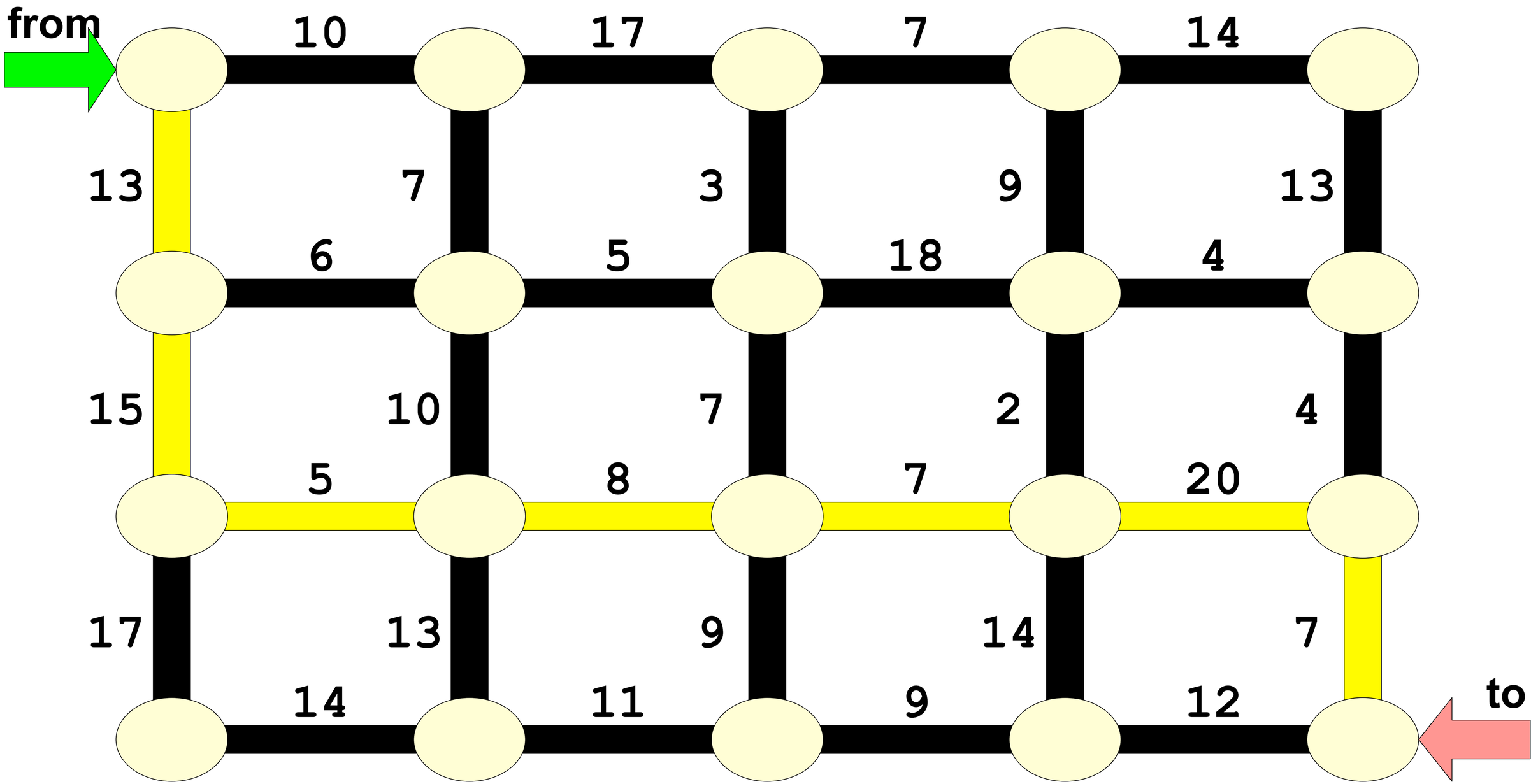
Travel Time: 13 + 15 + 17 + 14 + 11 + 9 + 12 = **91**

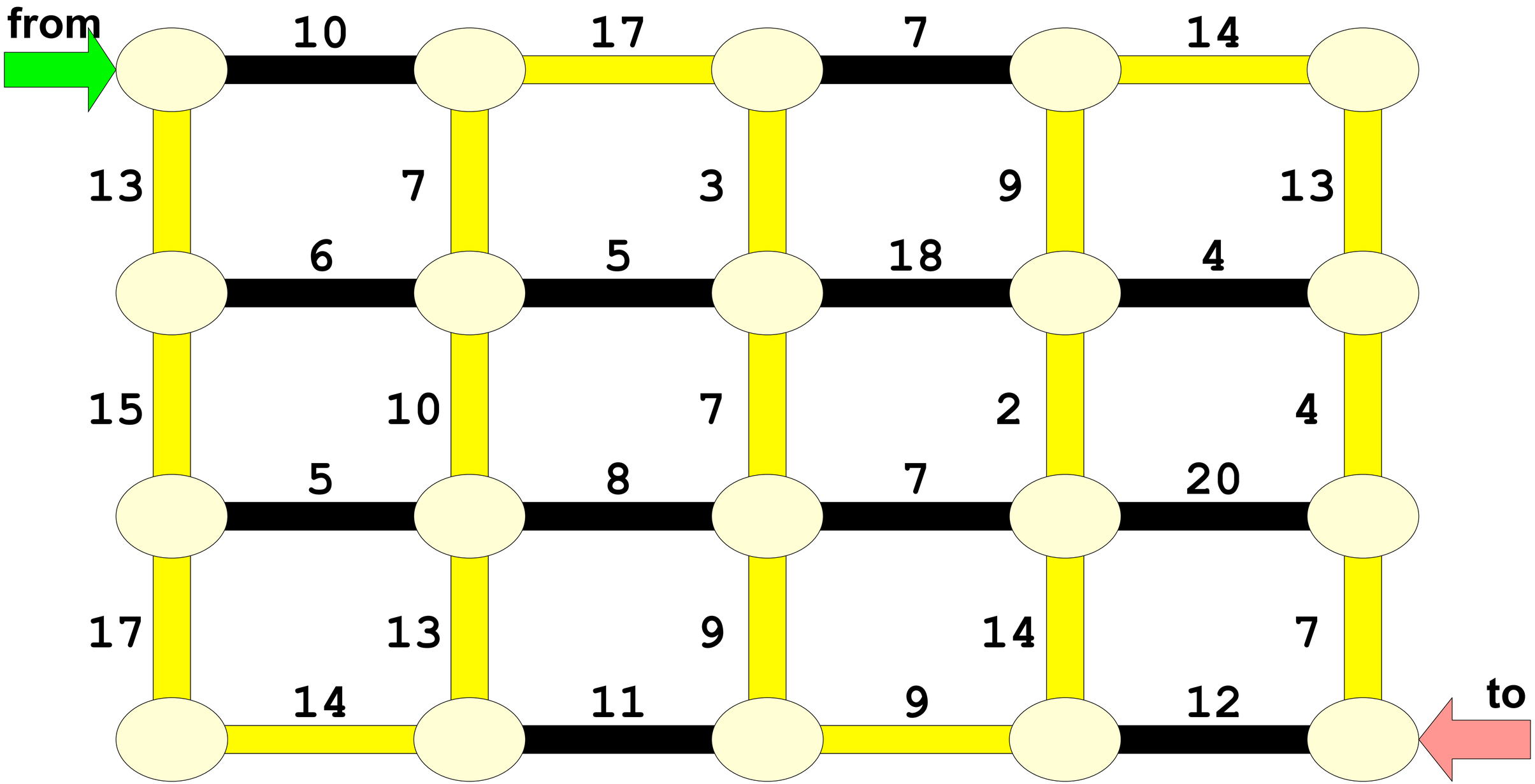

from
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to

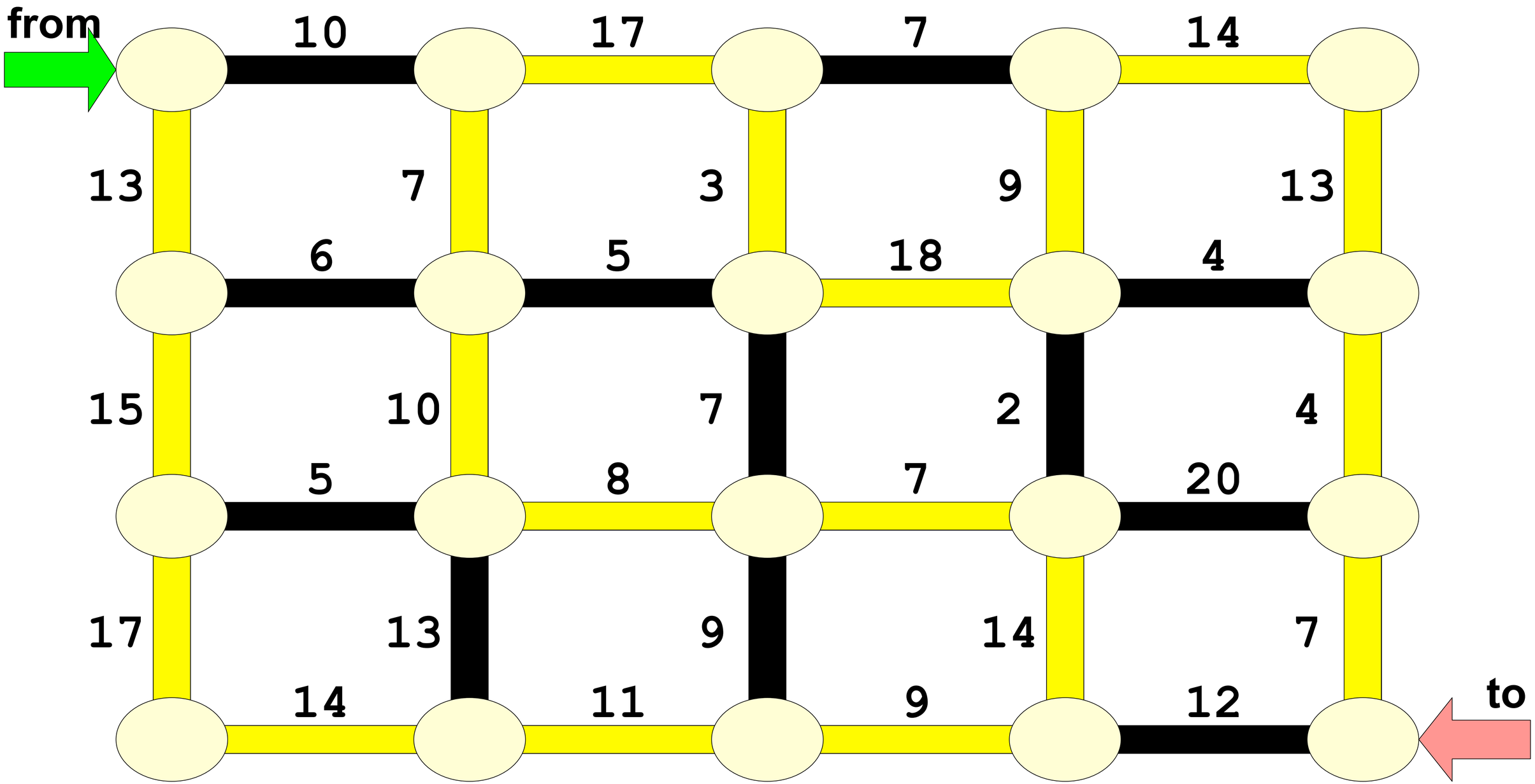


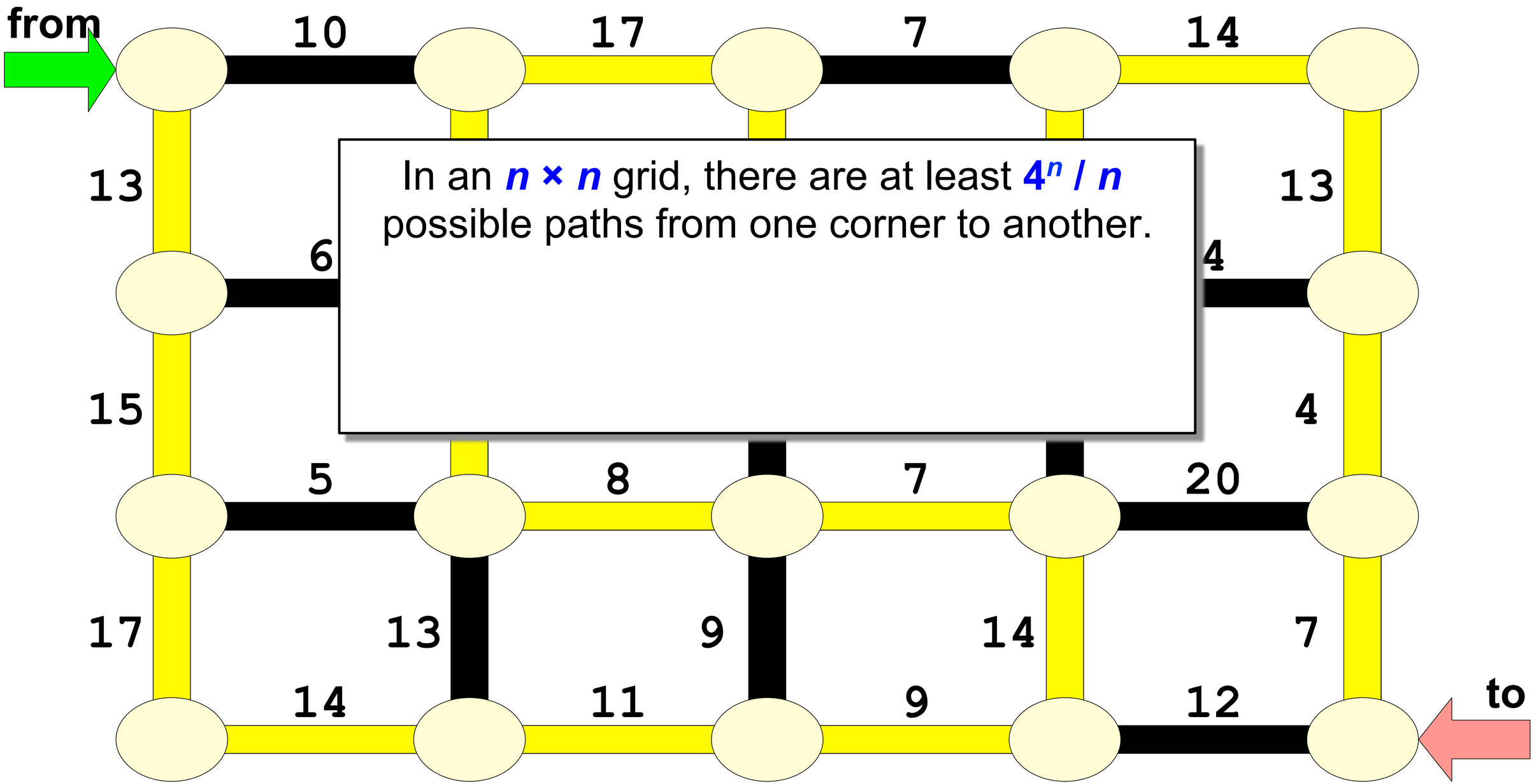
Travel Time: 10 + 17 + 7 + 14 + 13 + 4 + 7 = **72**


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

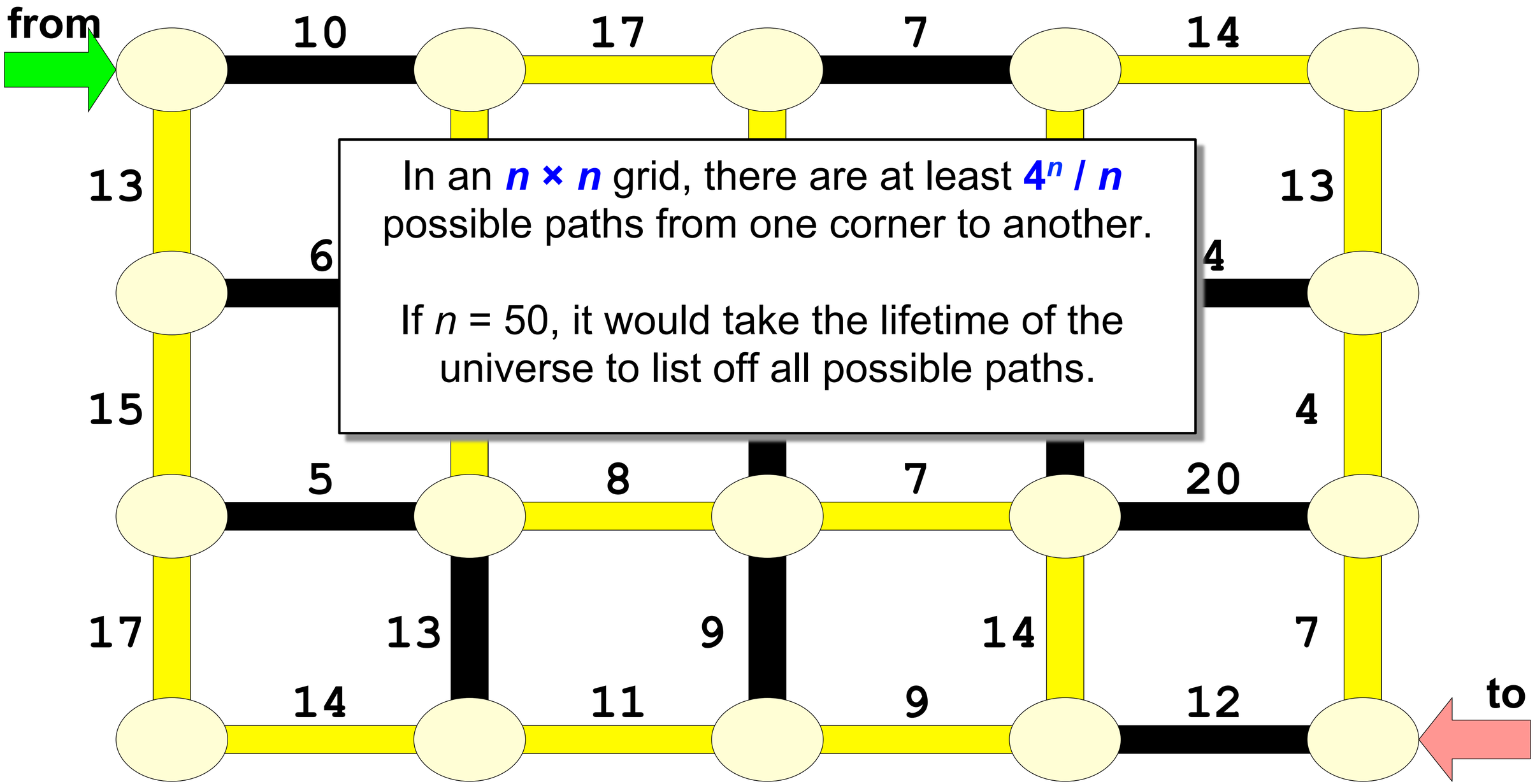

from
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to


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.

from

10 17 7 14

13 13

6 4

15 4

5 20

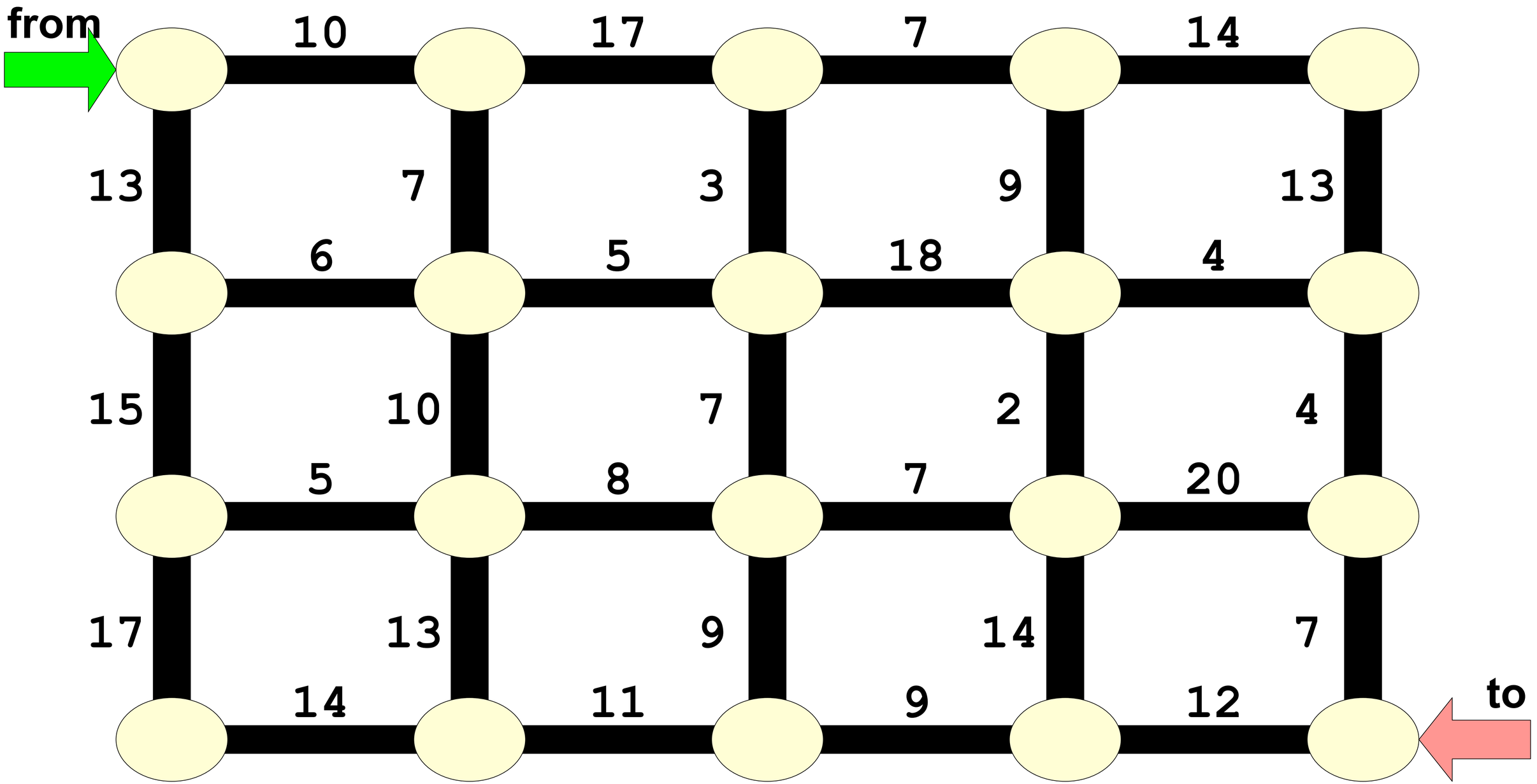
17 13 4 7

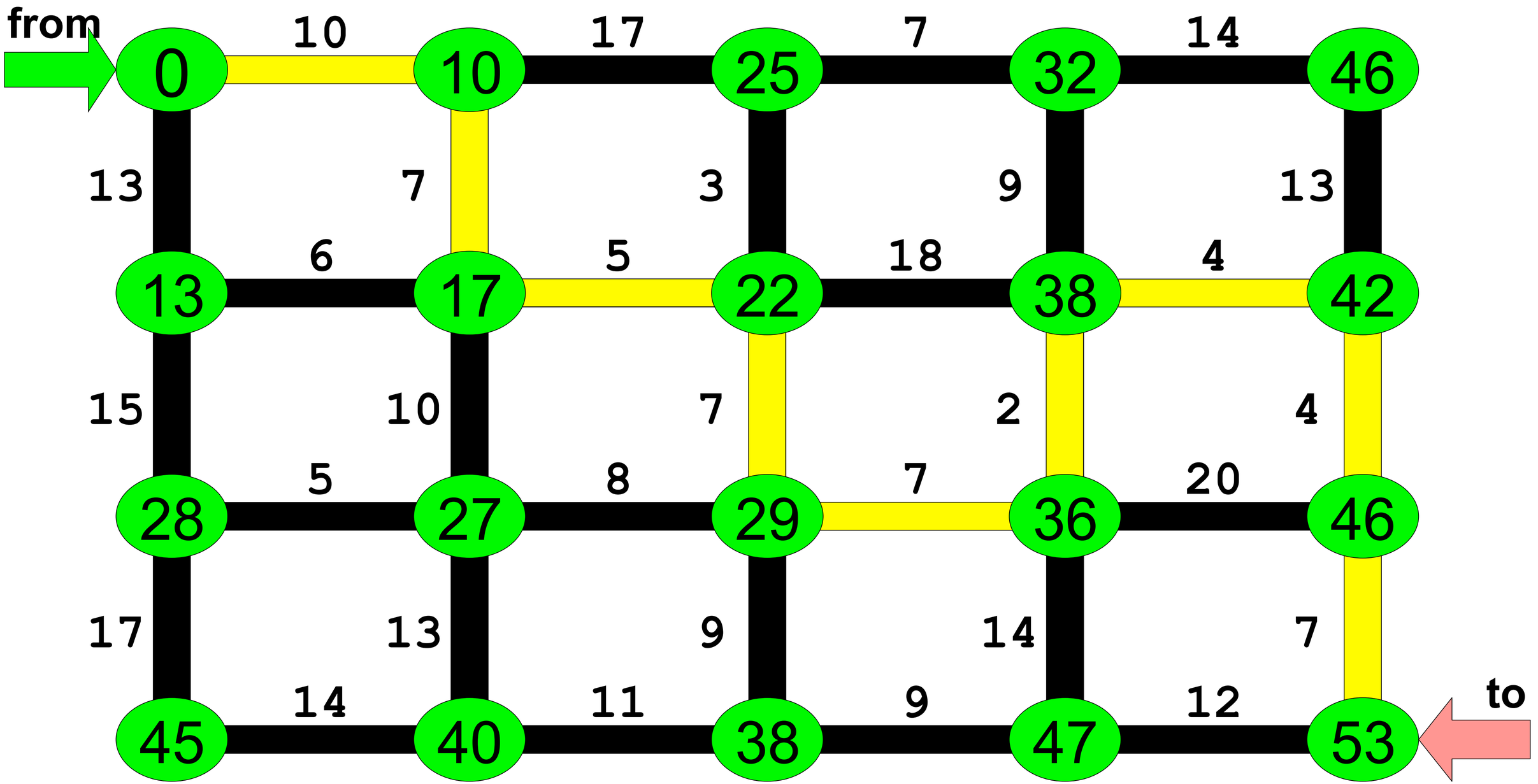
14 12

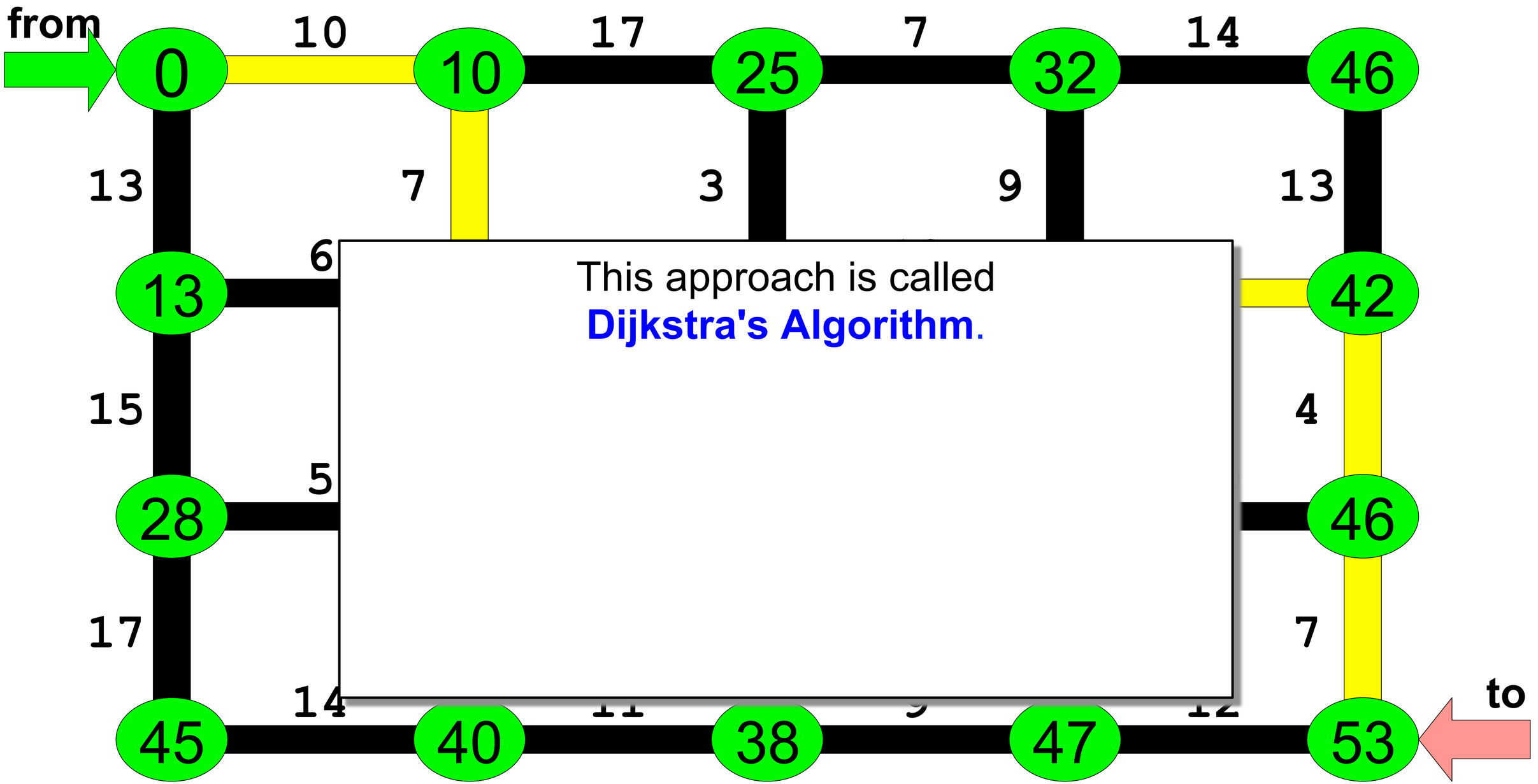
to

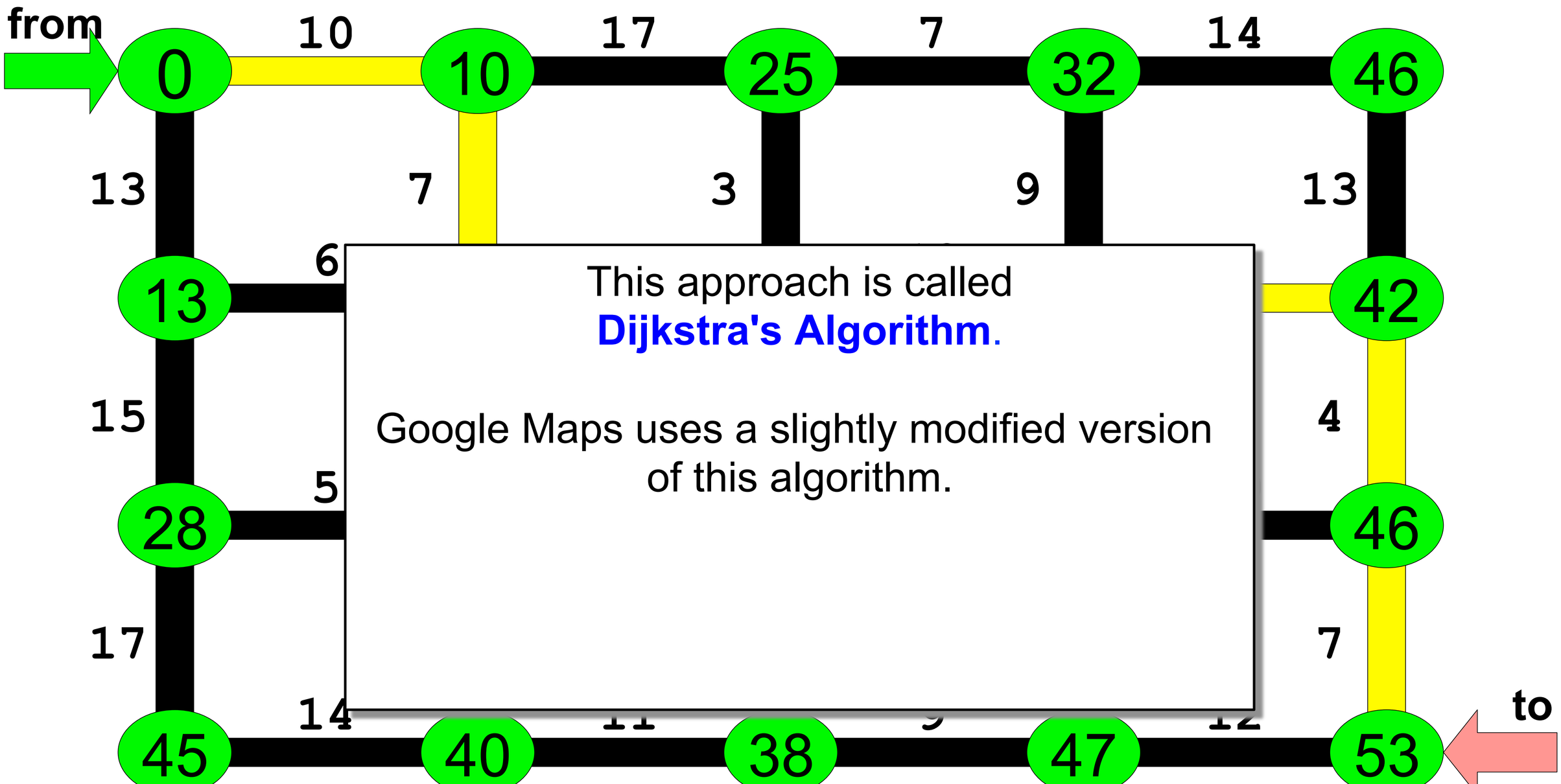
In an $n \times n$ grid, there are at least $4^n / n$ possible paths from one corner to another.

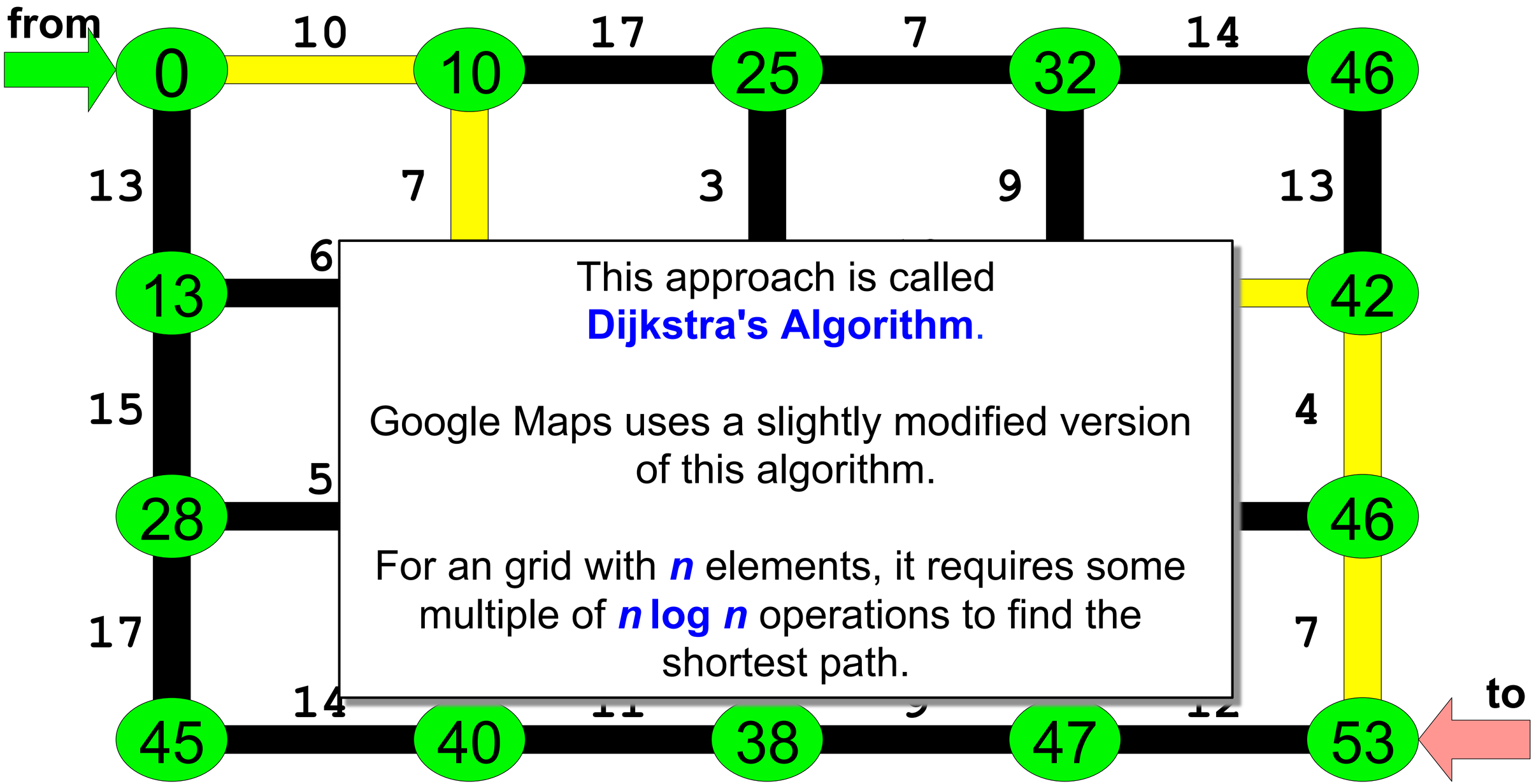
If $n = 50$, it would take the lifetime of the universe to list off all possible paths.


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
0
10
17
7
14
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


from
0
10
17
7
14
10
25
32
46
13
7
3
9
13
6
13
42
This approach is called
Dijkstra's Algorithm.
15
4
5
28
46
17
7
14
45
40
38
47
53
to


from
10
17
7
14
0
10
25
32
46
13
7
3
9
13
6
13
42
This approach is called
Dijkstra's Algorithm.
Google Maps uses a slightly modified version
of this algorithm.
15
4
5
28
46
17
7
14
to
45
40
38
47
53


from
0
10
17
7
14
46
10
25
32
13
7
3
9
13
6
13
42
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.
15
4
5
28
46
17
7
14
to
45
40
38
47
53

High expectations
that you will learn a lot

Where we are going

Course Information

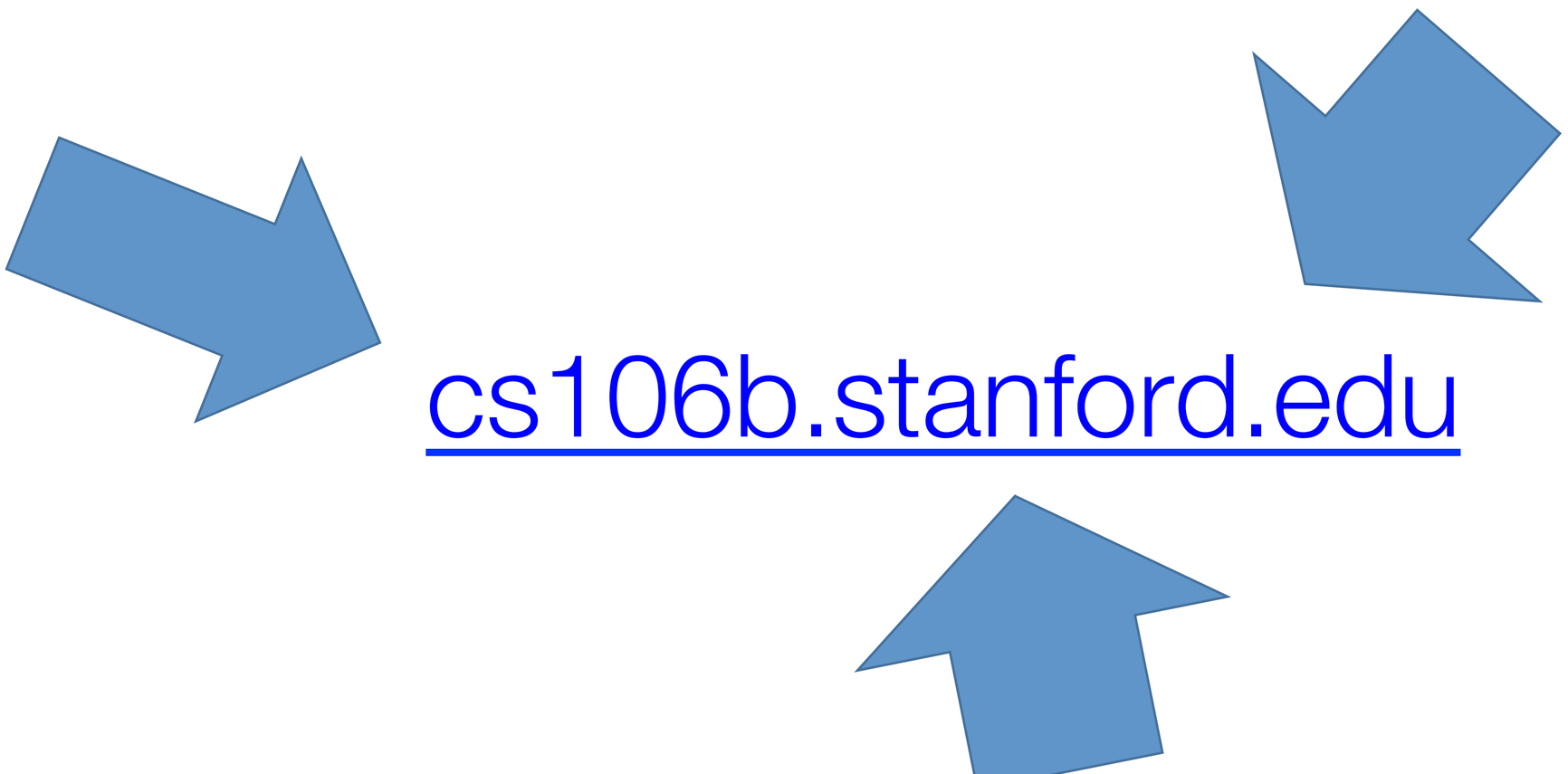
Write our first program

Where we are going

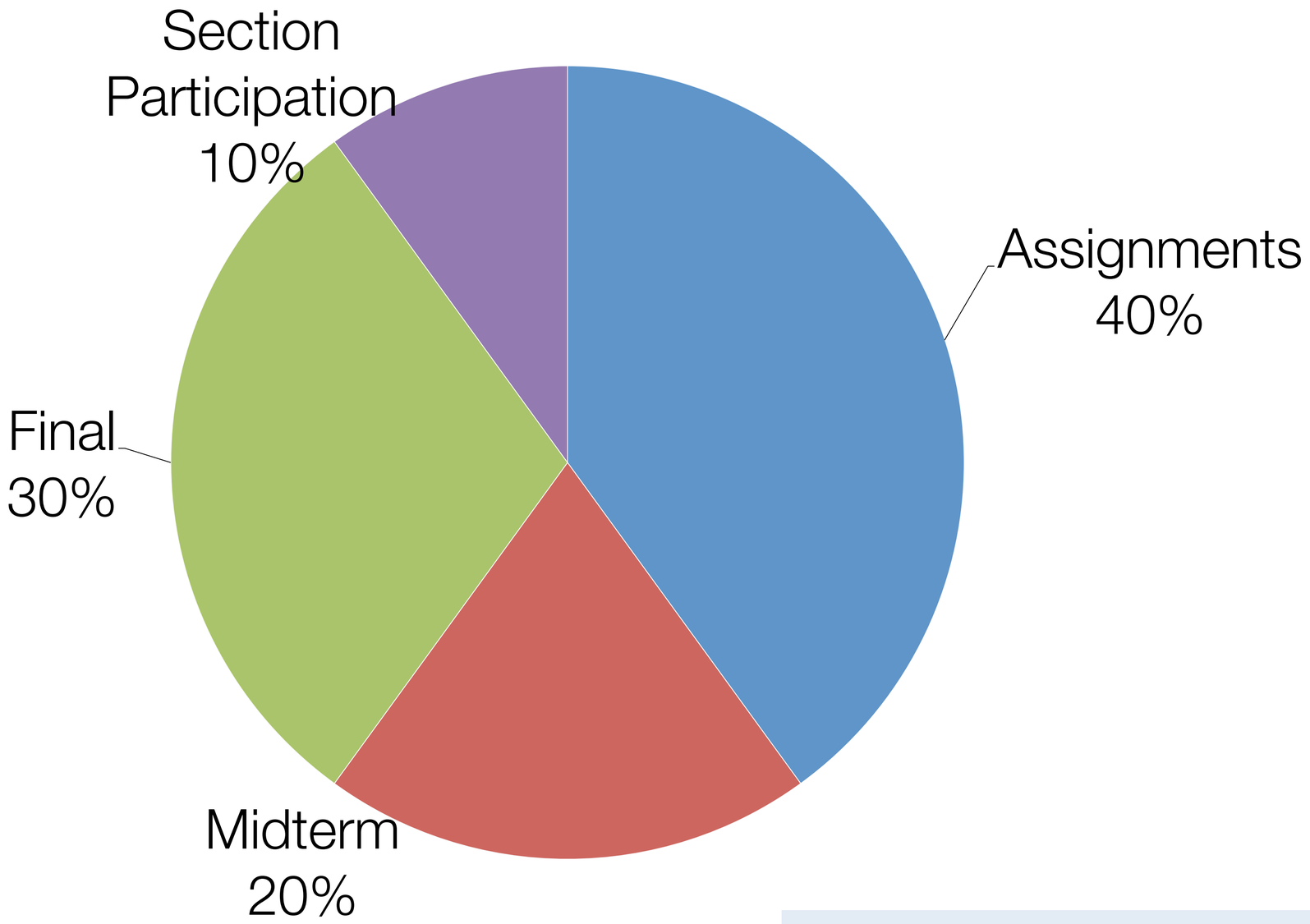
Course Information

Write our first program

Most Important Logistic Point



Components of CS106B



Final: Monday, Dec 12th

Assignments in CS106B

- Due at 12:00P.M.
- Three free “late days”
- Extensions approved by Chris, Chris or Anton.
- Graded by your section leader
- Opportunities for pair programming.
- 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

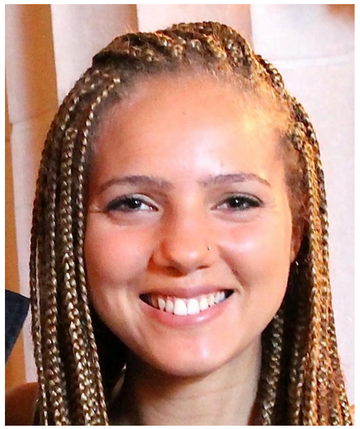
piazza

Review Piazza

2

Go to the LaIR / OH

3

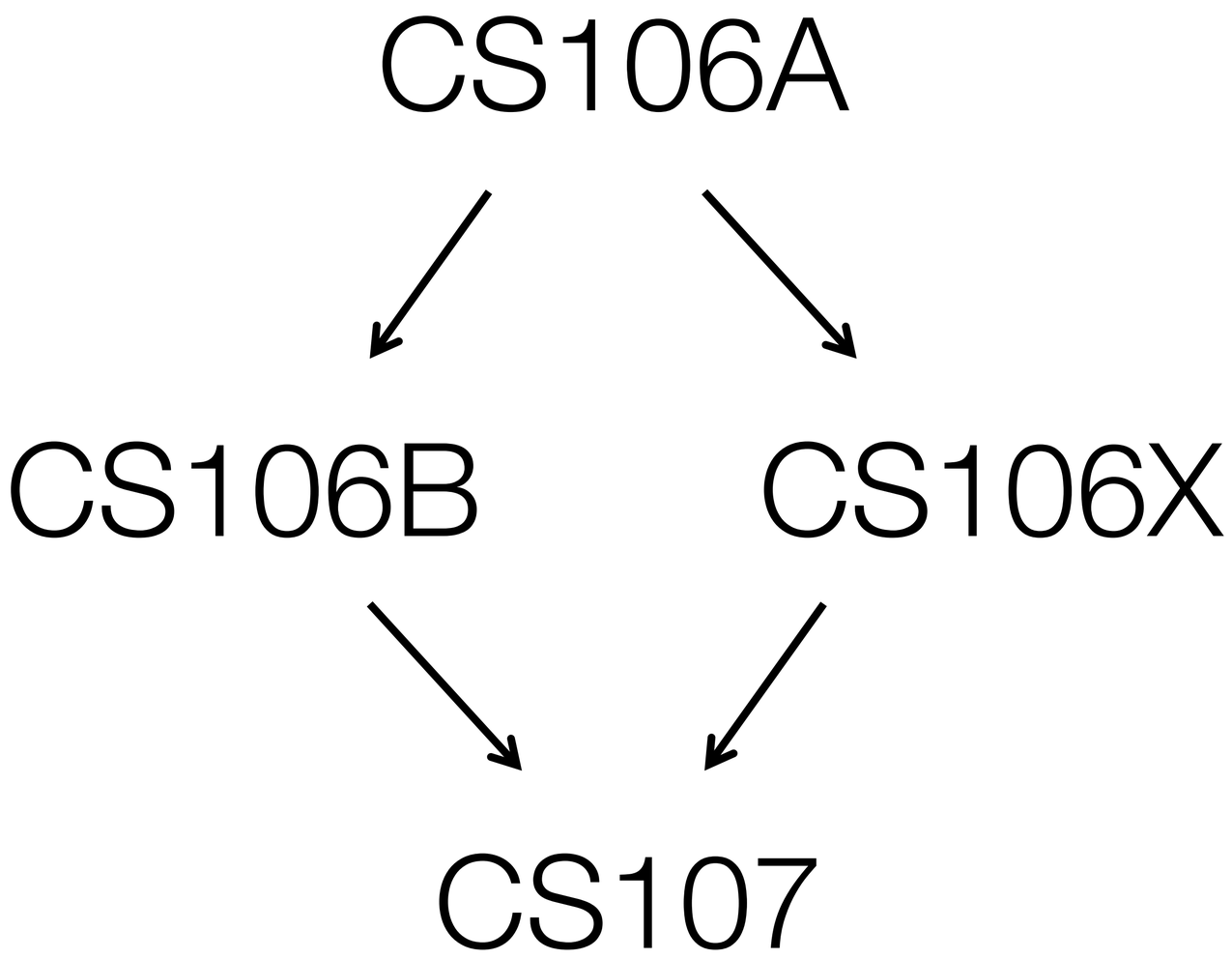


Contact your Section Leader

4

Email Anton or the Chrises

Is CS106B The Right Class?



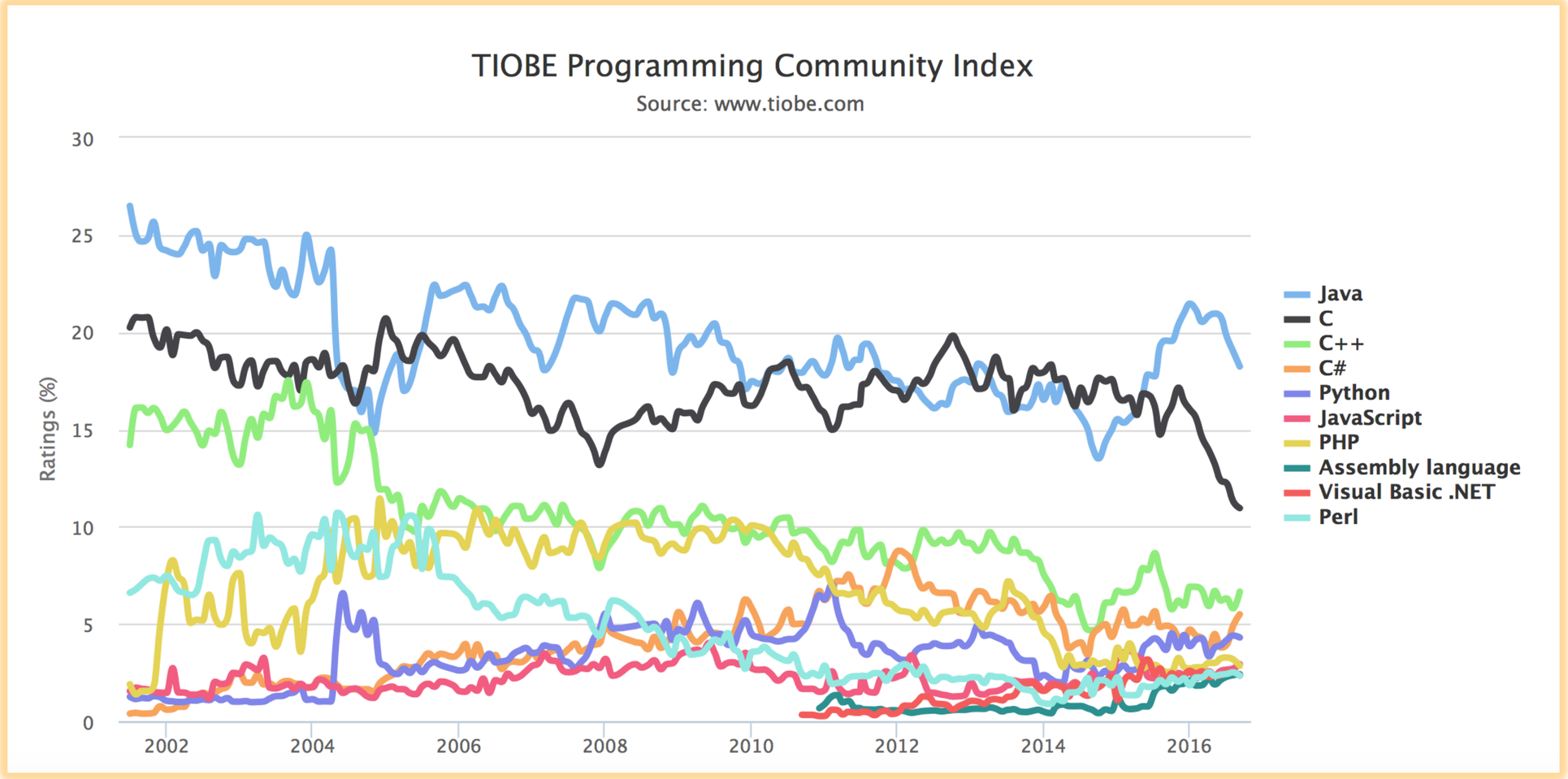
CS106L
CS106S

One last detail…

C++

C++

Although there are [hundreds of computer languages](), in CS 106B 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!)



What is the most used language in programming?

Profanity!

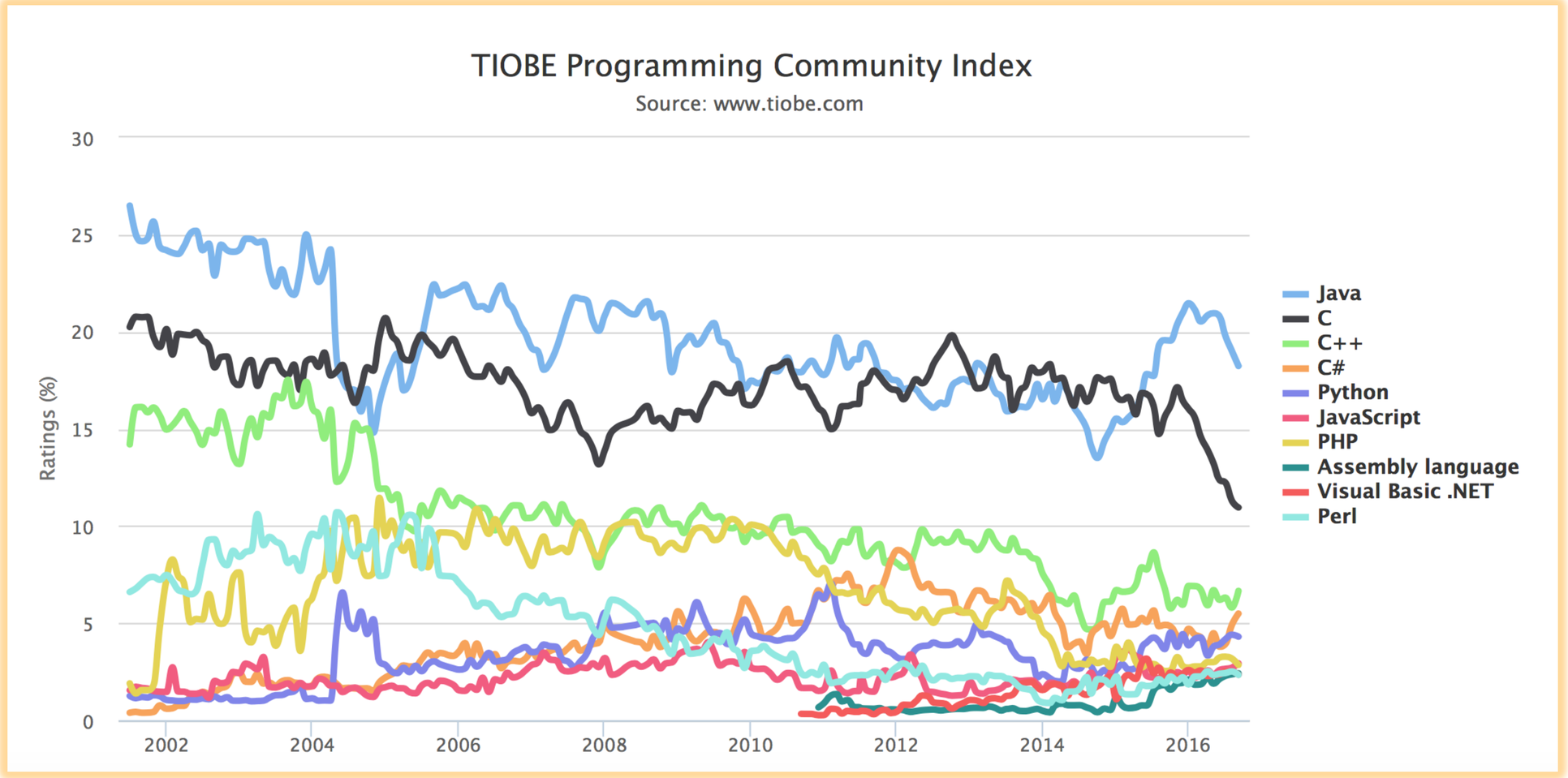
CS 106/107 Languages

The 106/107 languages:

```
106A : Java (1995)
106B : 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).



Where we are going

Course Information

Write our first program

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 CS …

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/cs106b/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!"

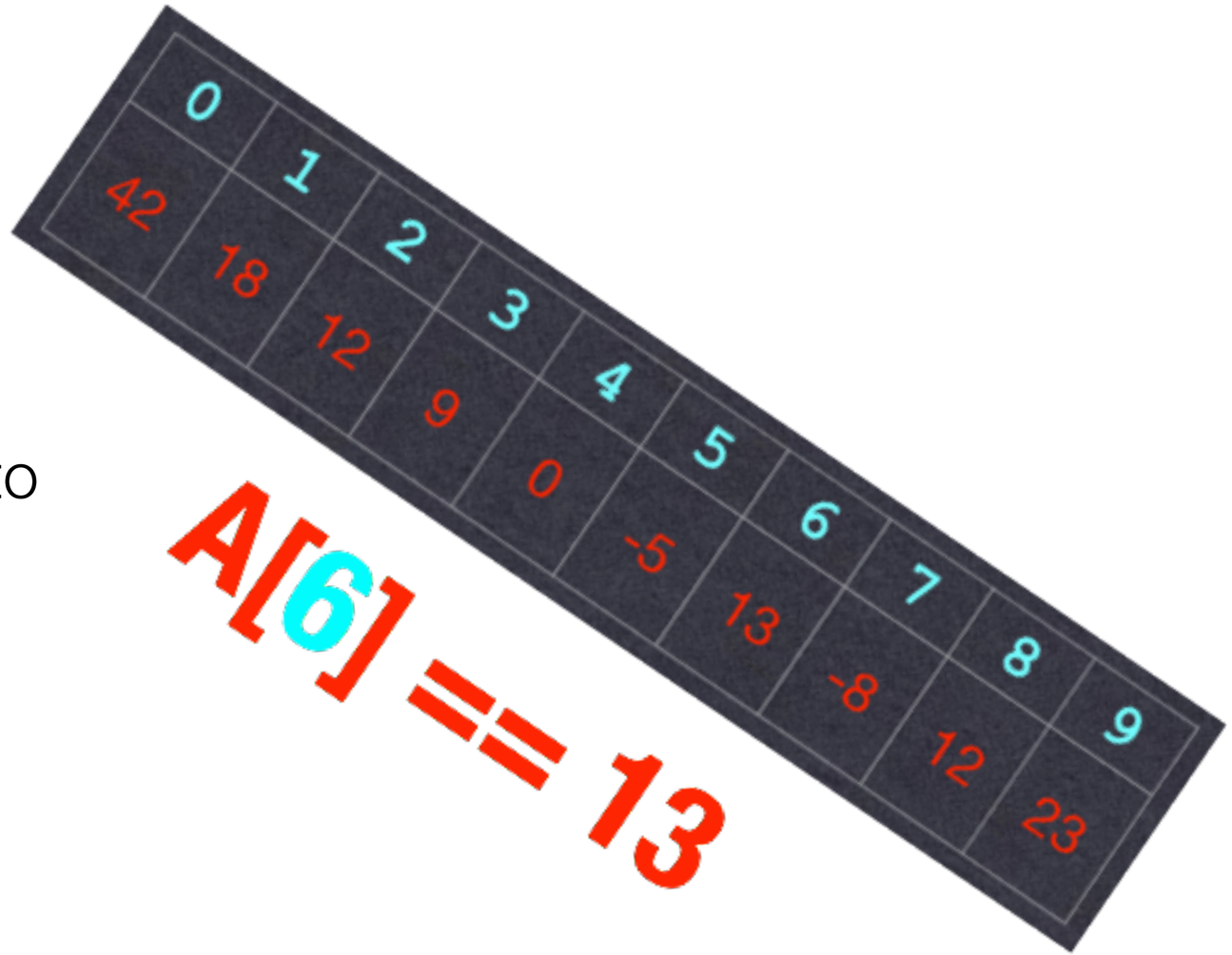
Your *Second* C++ Program!

Because this is 106B, 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.



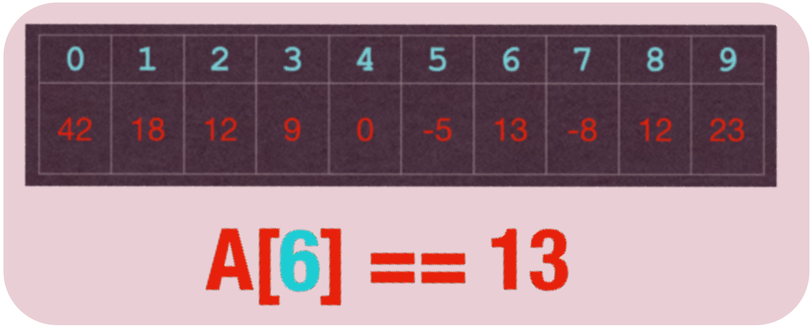
Your *Second* C++ Program!

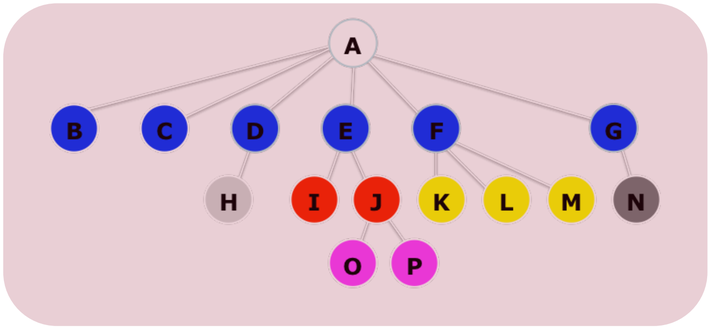
```
// Populate a Vector

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

using namespace std;

const int NUM_ELEMENTS = 100000;
```

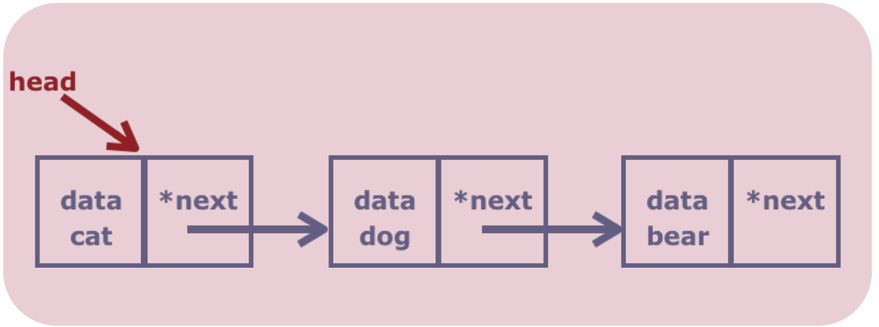
(continued!)

```
// 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







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**

Time and Place

Who are you?

- African Studies
- Applied Physics
- Aeronautics & Astro.
- Bioengineering
- Biology
- Business Administration
- Chemical Engineering
- Chemistry
- Classics
- Civil and Environmental Engineering
- Computational and Mathematical Engineering
- Computer Science
- Creative Writing
- East Asian Studies
- Economics
- Education
- Electrical Engineering
- Energy Resource Engineering
- English
- Financial Mathematics
- Film and Media Studies
- French
- History
- International Relations
- Japanese
- Law
- Materials Science and Engineering
- Mathematical and Computational Sciences
- Mathematics
- Mechanical Engineering
- Medicine
- Management Science and Engineering
- Modern Language
- Music
- Neuroscience
- Physics
- Political Science
- Psychology
- Science, Technology, and Society
- Statistics
- Symbolic Systems
- Undeclared!

Everyone is Welcome