Welcome to CS106B!

- Three Handouts
- Today:
  - Course Overview
  - Where are We Going?
  - Introduction to C++

Who's Here Today?

- African Studies
- Applied Physics
- 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!

Course Staff

Instructor: Keith Schwarz (htiek@cs.stanford.edu)

Head TA: Dawson Zhou (zhoud@stanford.edu)

The CS106B Section Leaders
The CS106B Course Helpers

Course Website

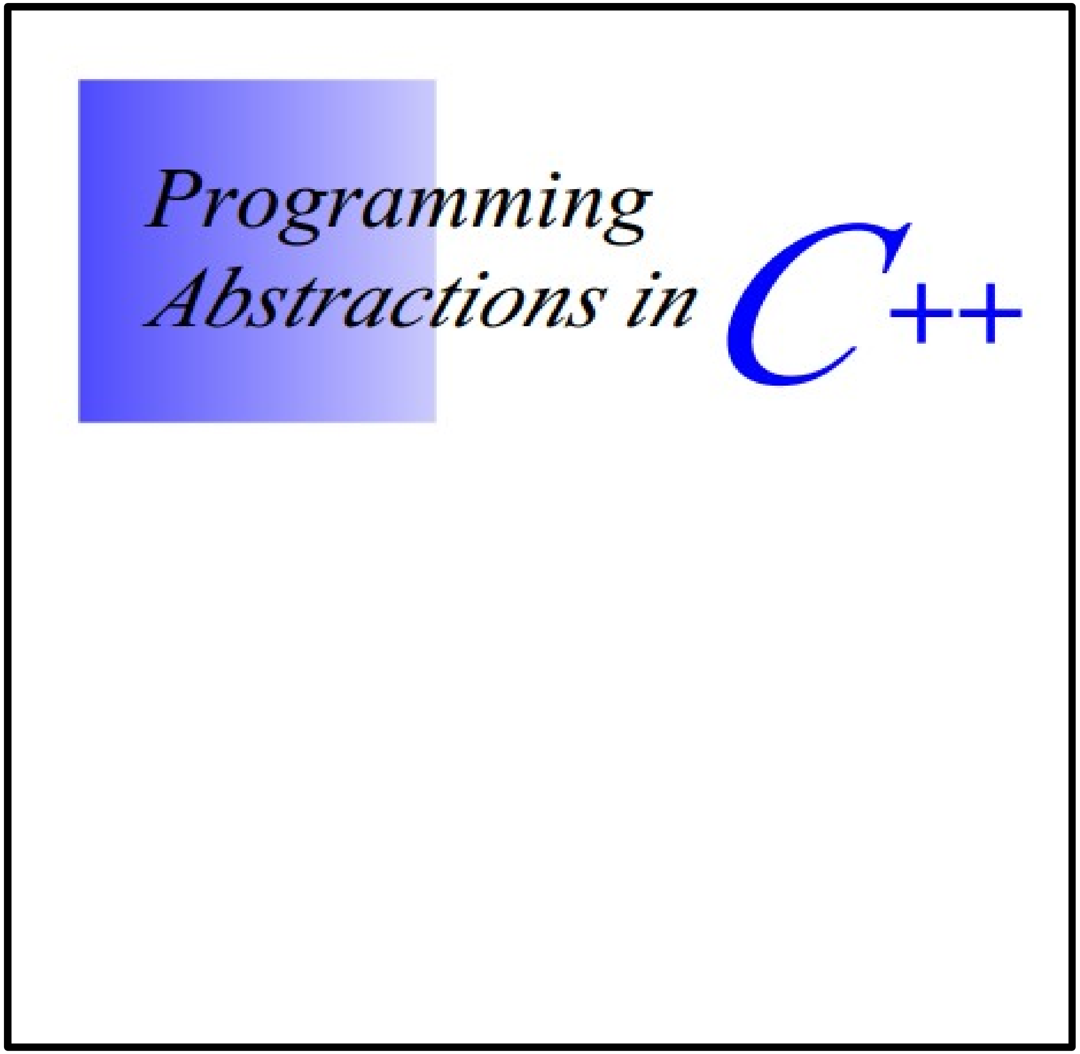
http://cs106b.stanford.edu

Prerequisites

CS106A

(or equivalent)

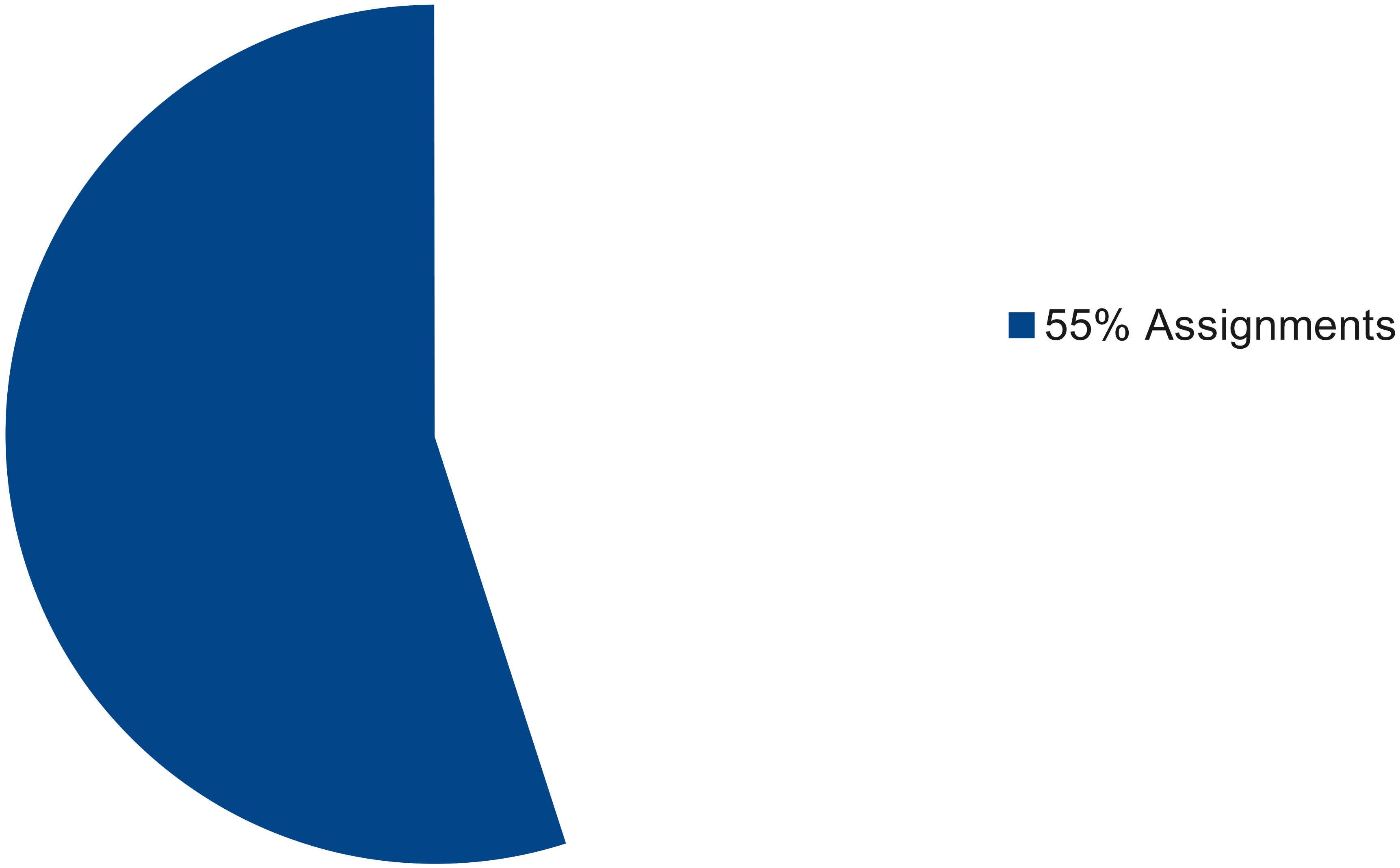
Required Reading



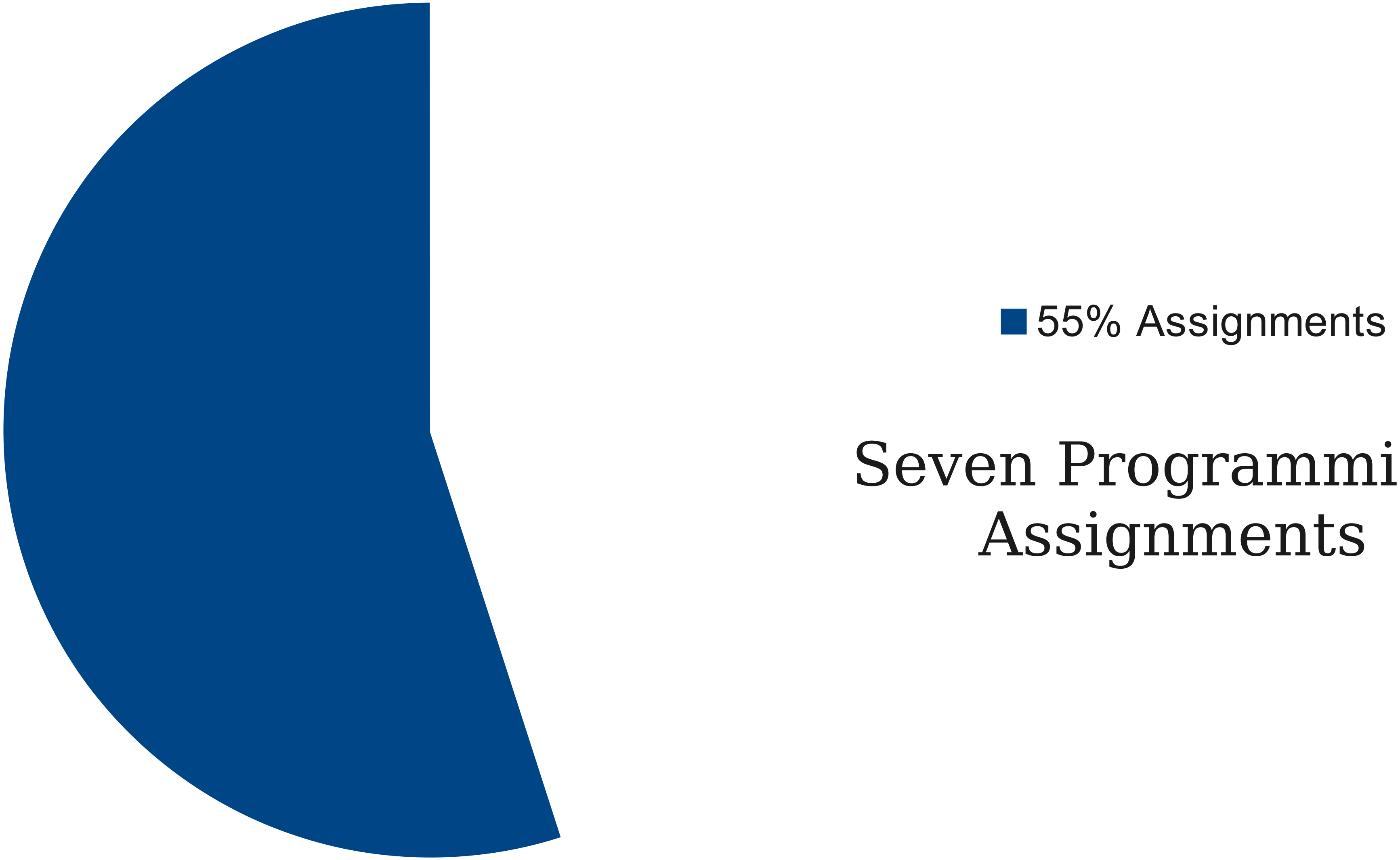
Programming Abstractions in C++

Grading Policies

Grading Policies

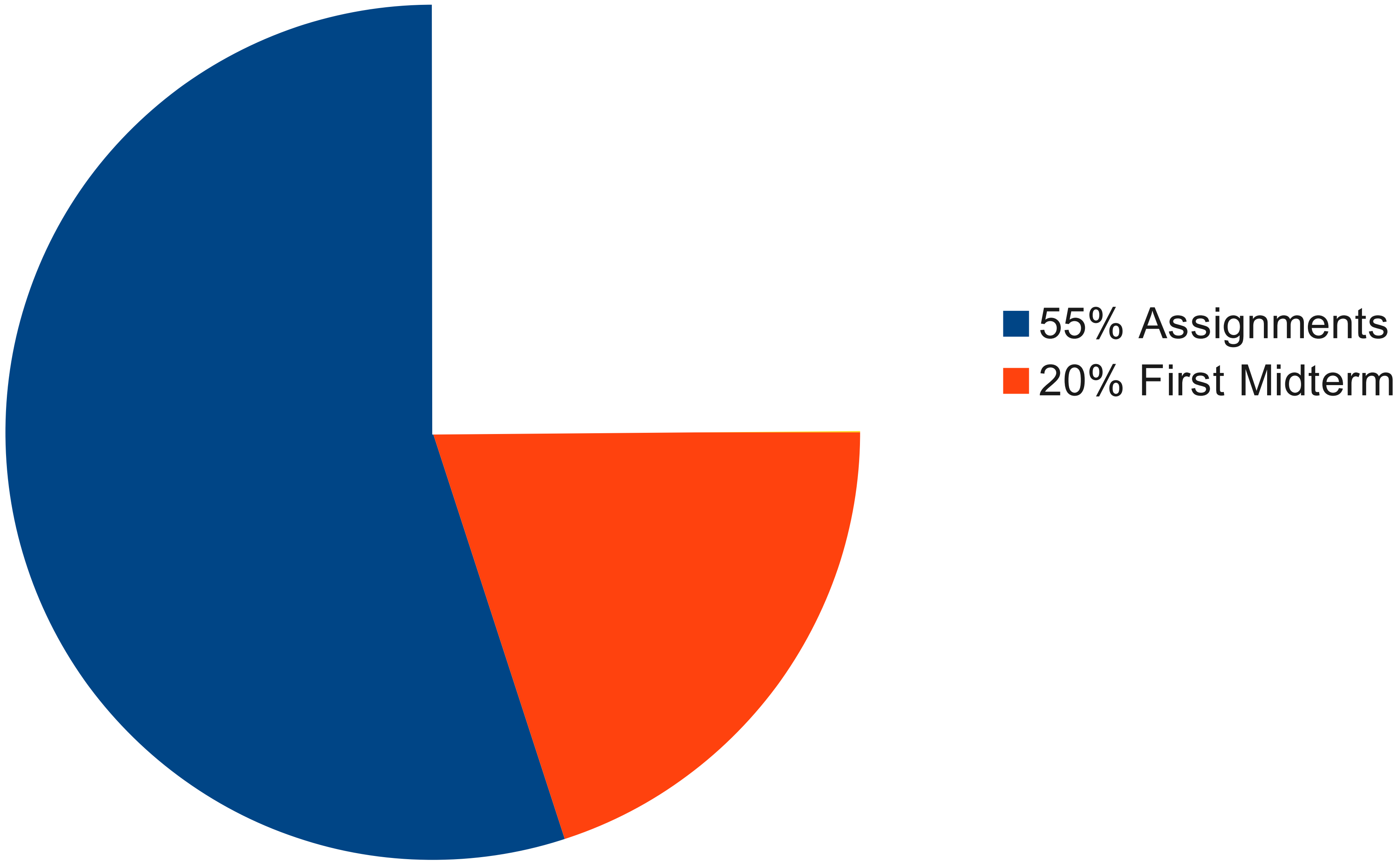


Grading Policies

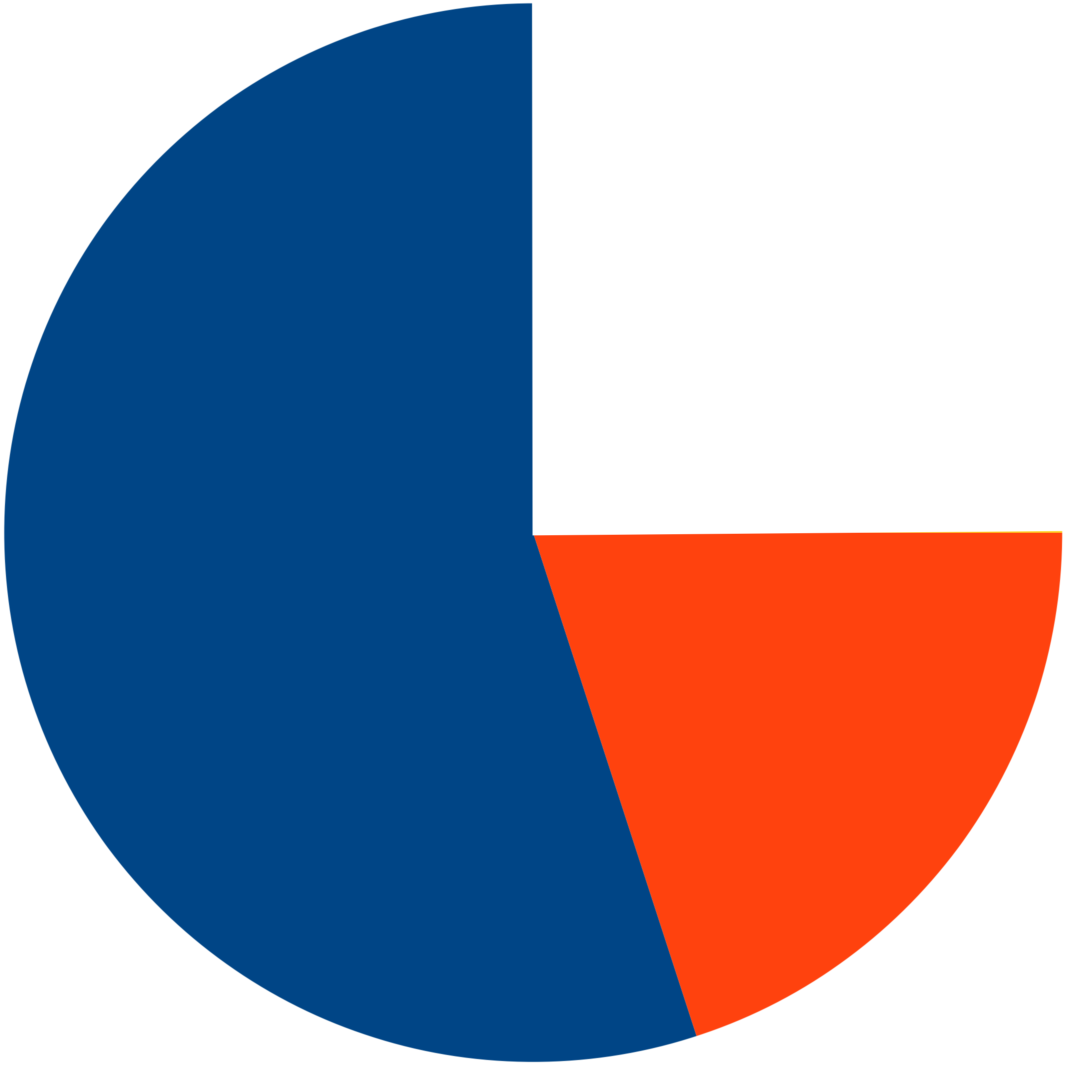


Seven Programming Assignments

Grading Policies



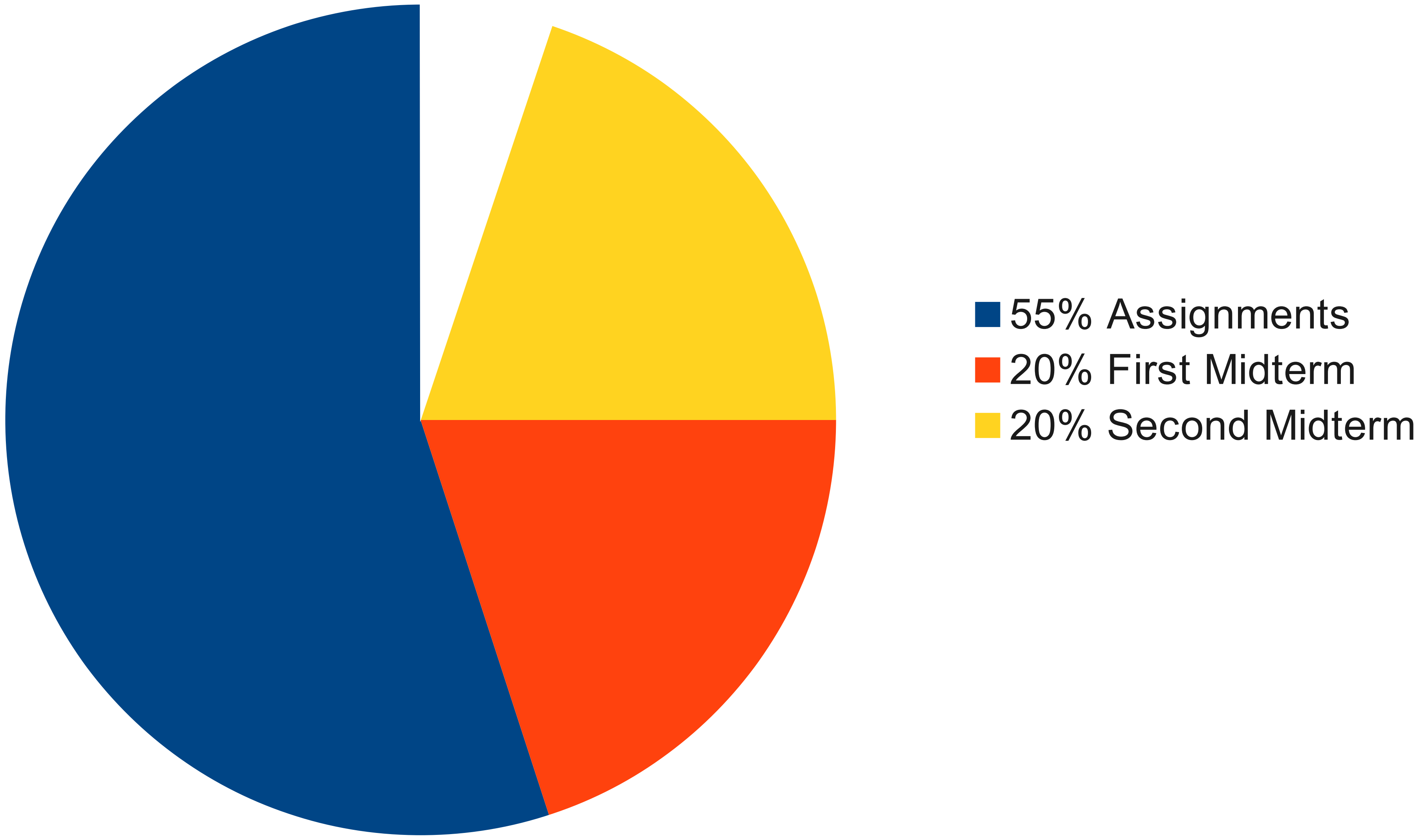
Grading Policies



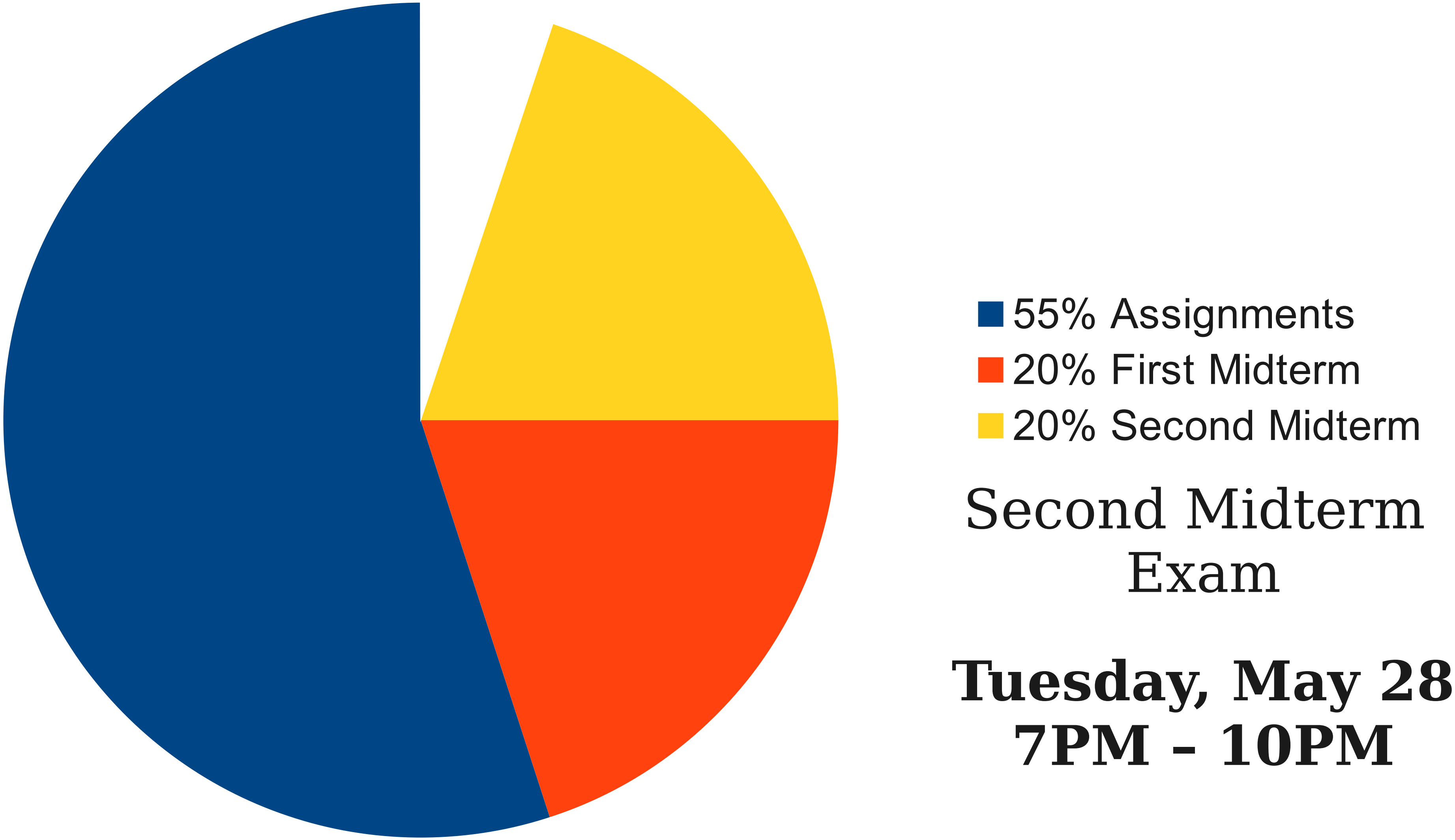
First Midterm Exam

Tuesday, May 7
7PM – 10PM

Grading Policies



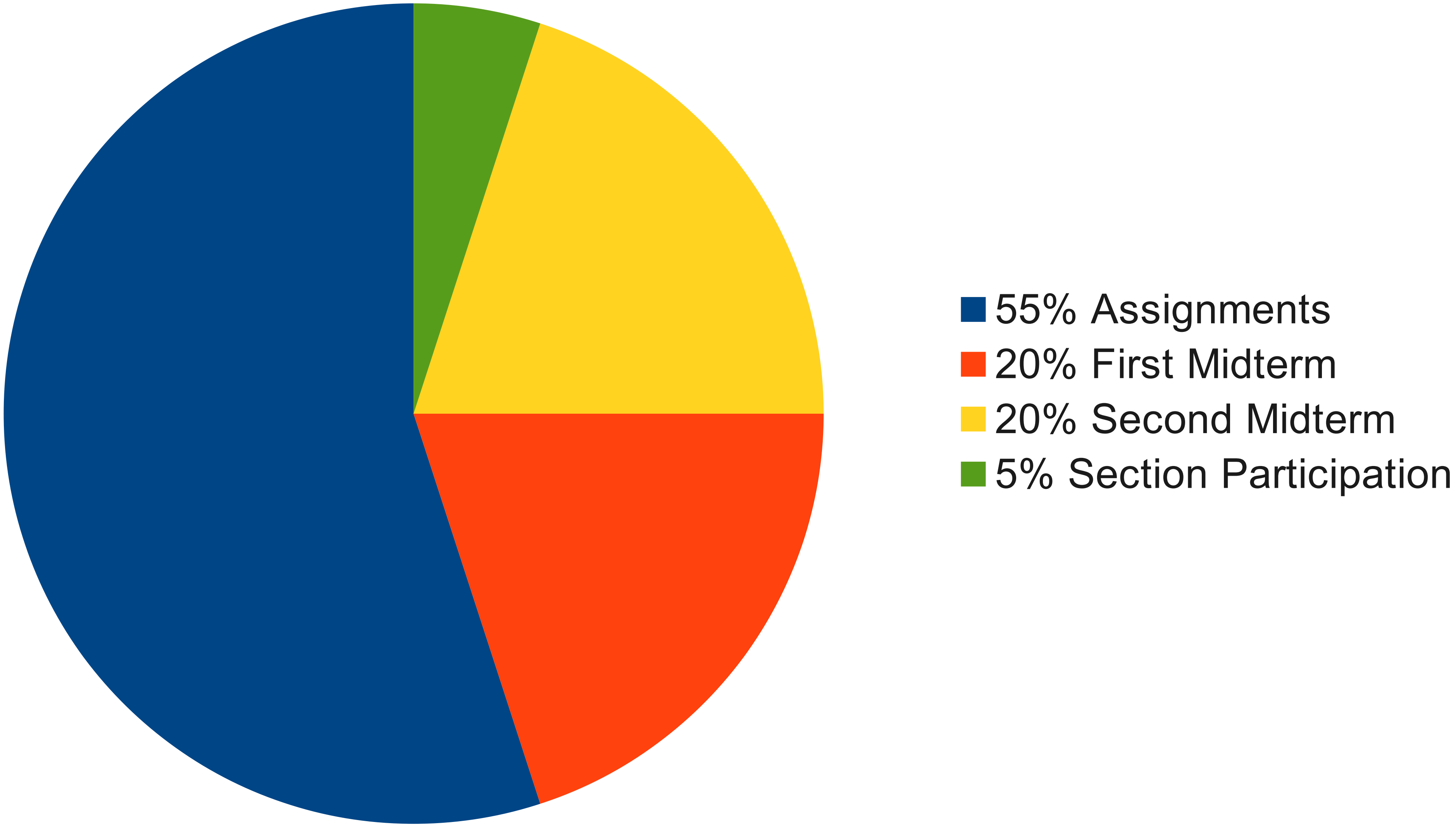
Grading Policies



Second Midterm Exam

Tuesday, May 28
7PM – 10PM

Grading Policies



Discussion Sections

- Weekly discussion sections.
- Section attendance is **required** in CS106B.
- Sign up between Thursday, April 4 at 5:00PM and Sunday, April 7 at 5:00PM at

http://cs198.stanford.edu/section

- You don't need to (and shouldn't!) sign up for a section on Axess; everything is handled through the above link.

How Many Units?

How Many Units?

```
int numUnits(bool isGrad, bool wantsFewerUnits) {



}
```

How Many Units?

```
int numUnits(bool isGrad, bool wantsFewerUnits) {
    if (!isGrad) return 5;







}
```

How Many Units?

```
int numUnits(bool isGrad, bool wantsFewerUnits) {
    if (!isGrad) return 5;
    if (!wantsFewerUnits) return 5;





}
```

How Many Units?

```
int numUnits(bool isGrad, bool wantsFewerUnits) {
    if (!isGrad) return 5;
    if (!wantsFewerUnits) return 5;

    if (reallyBusy()) {
        return 3;
    }



}
```

How Many Units?

```
int numUnits(bool isGrad, bool wantsFewerUnits) {
    if (!isGrad) return 5;
    if (!wantsFewerUnits) return 5;
    if (reallyBusy()) {
        return 3;
    } else {
        return 4;
    }
}
```

Getting Help

Getting Help

- LaIR Hours!
  - Sunday – Thursday, 6PM – Midnight
  - Starts next week.
- Dawson's Office Hours in Gates 160
  - Monday/Wednesday 11AM – Noon
  - Tuesday/Thursday 1PM – 2PM
- Keith's Office Hours in Gates 178
  - Tuesday / Thursday, 2PM – 4PM

What's Next in Computer Science?

Goals for this Course

- **Learn how to model and solve complex problems with computers.**
- To that end:
  - Explore common abstractions for representing problems.
  - Harness recursion and understand how to think about problems recursively.
  - Quantitatively analyze different approaches for solving problems.

Goals for this Course

Learn how to model and solve complex problems with computers.

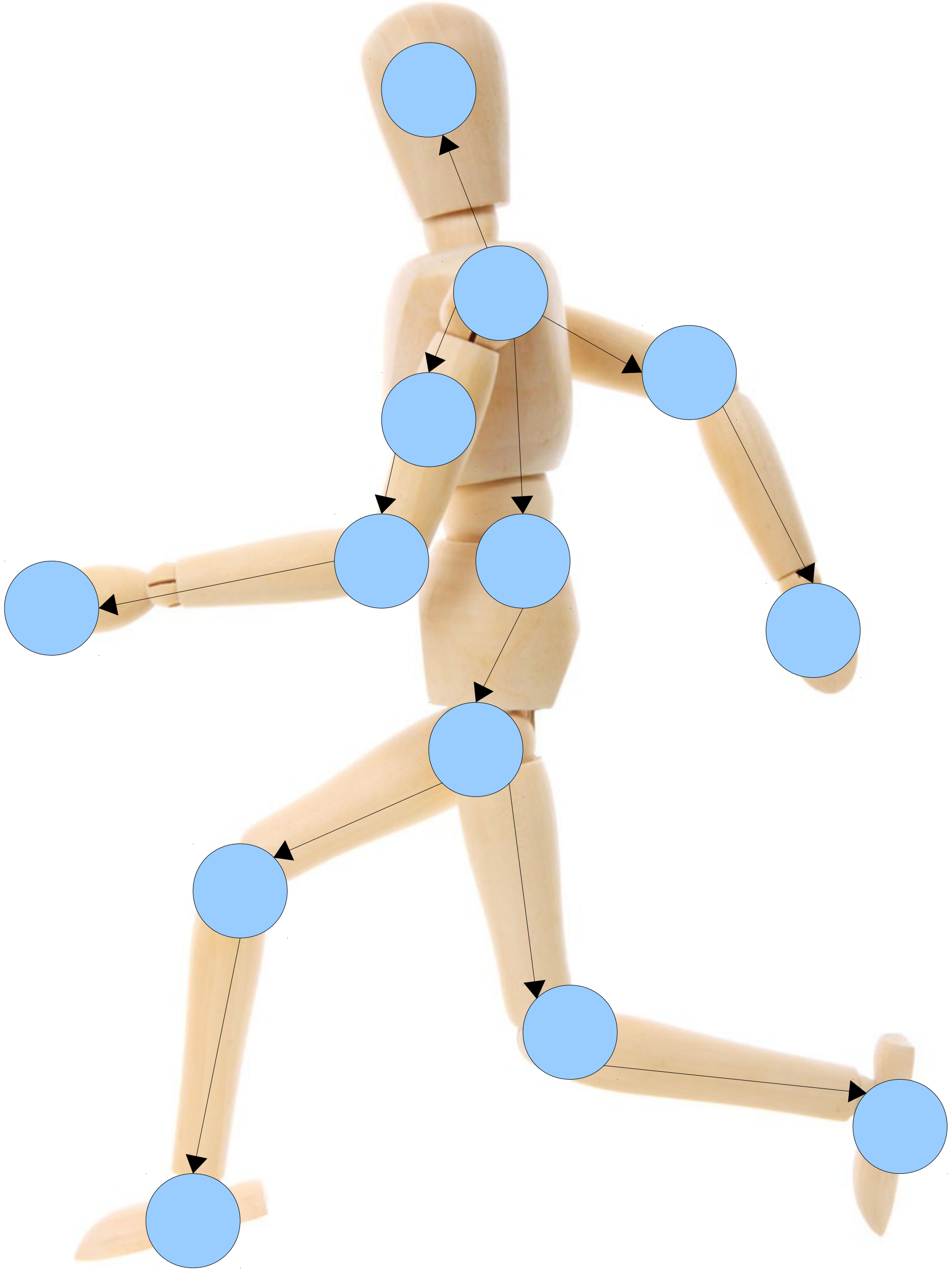
To that end:

- Explore common abstractions for representing problems.

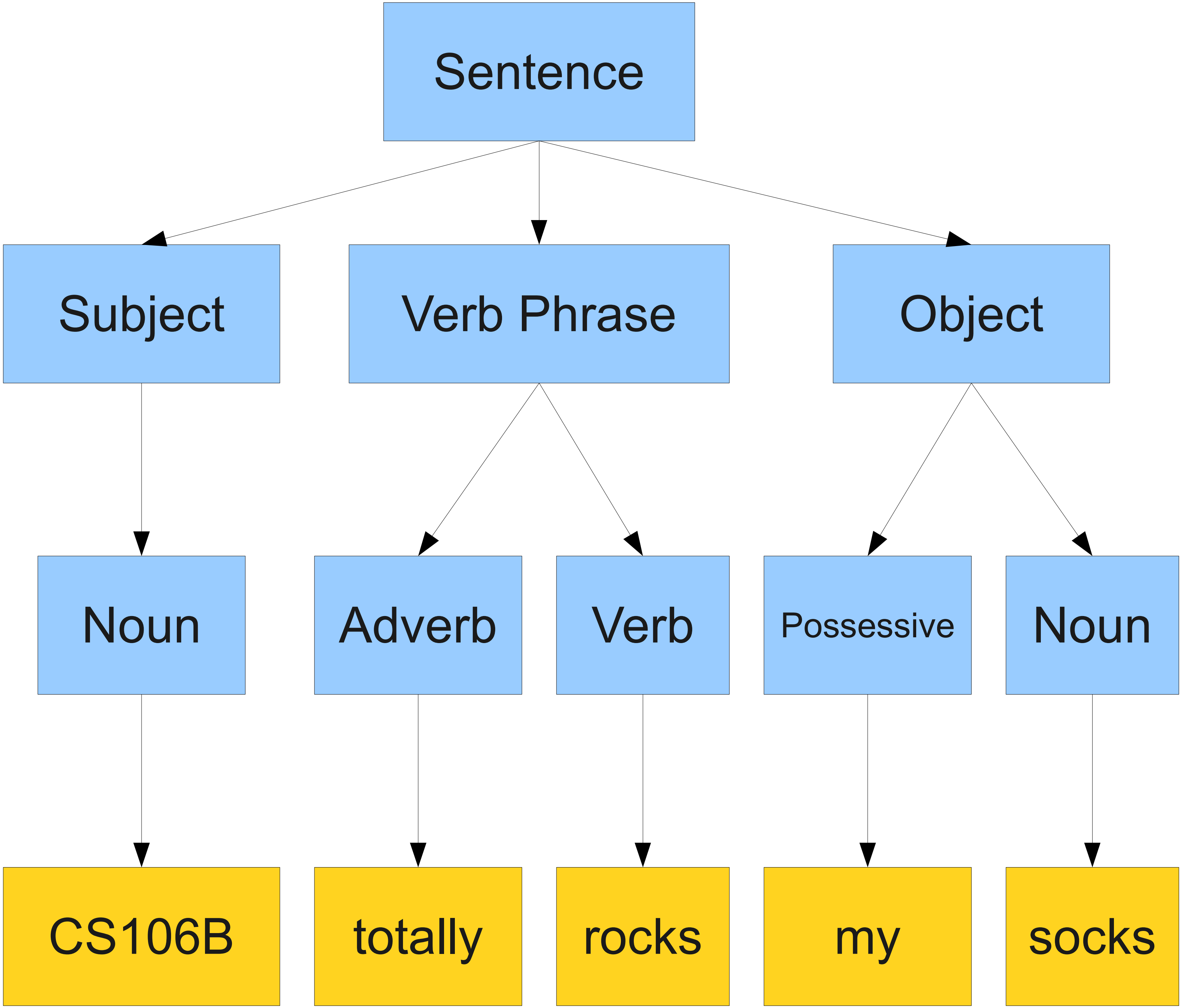
Harness recursion and understand how to think about problems recursively.

Quantitatively analyze different approaches for solving problems.

http://www.publicdomainpictures.net/pictures/10000/velka/1-1265899974oKJ9.jpg



http://www.publicdomainpictures.net/pictures/10000/velka/1-1265899974oKJ9.jpg



Sentence

Subject

Verb Phrase

Object

Noun

Adverb

Verb

Possessive

Noun

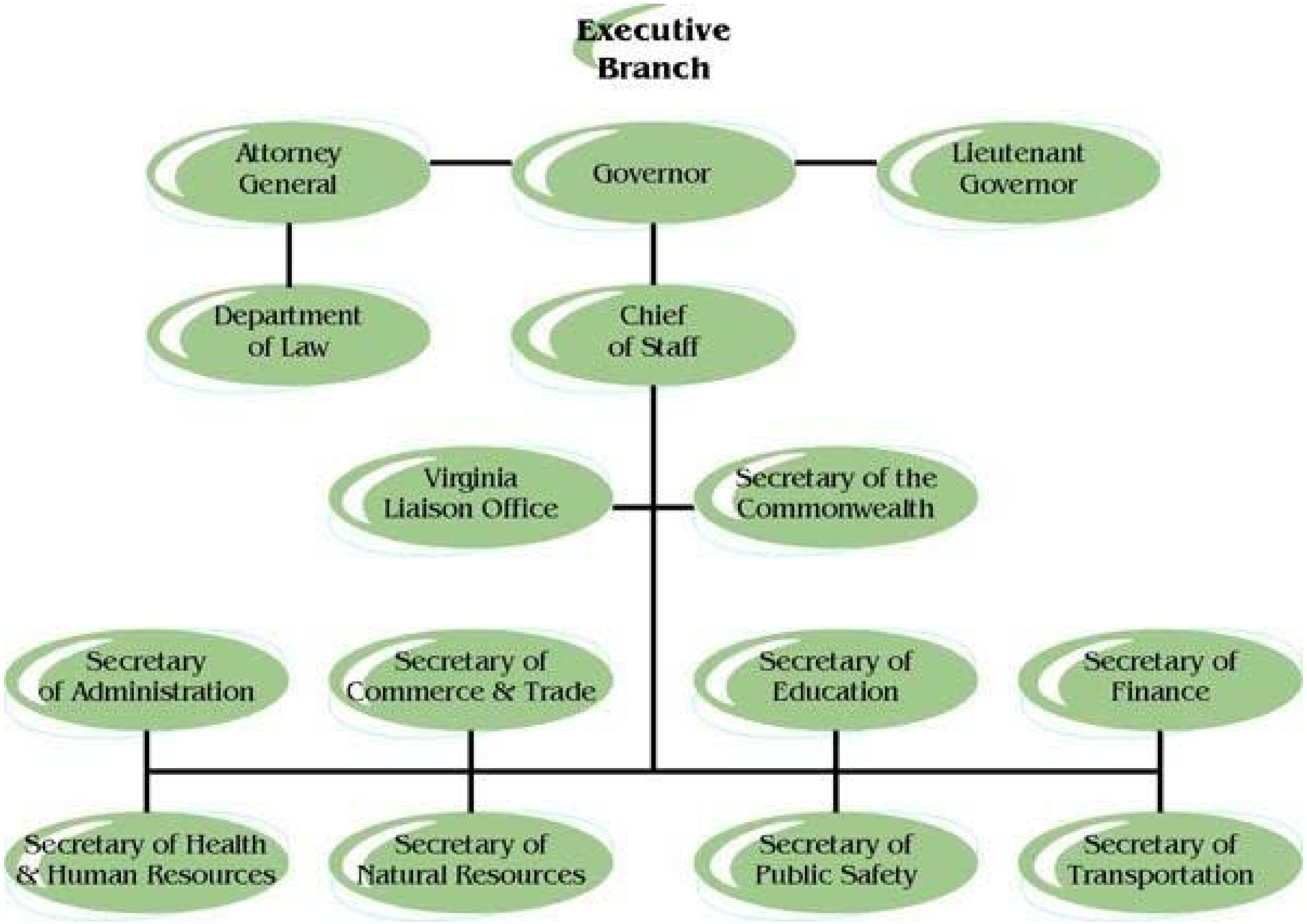
CS106B

totally

rocks

my

socks



Executive Branch

Attorney General

Governor

Lieutenant Governor

Department of Law

Chief of Staff

Virginia Liaison Office

Secretary of the Commonwealth

Secretary of Administration

Secretary of Commerce & Trade

Secretary of Education

Secretary of Finance

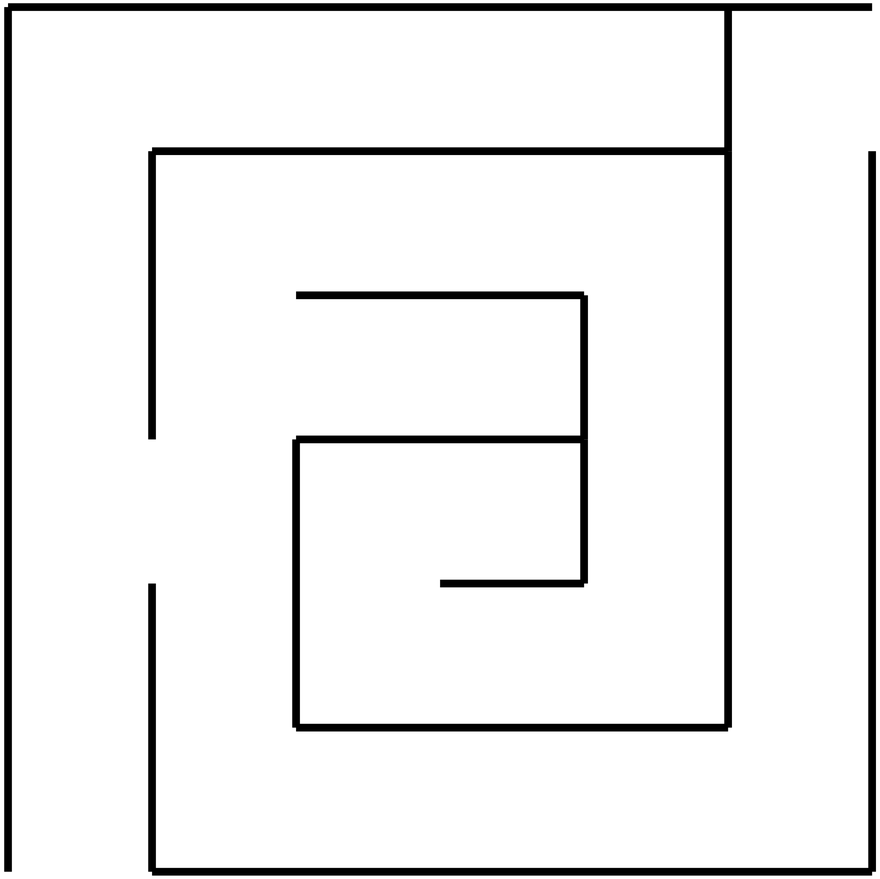
Secretary of Health & Human Resources

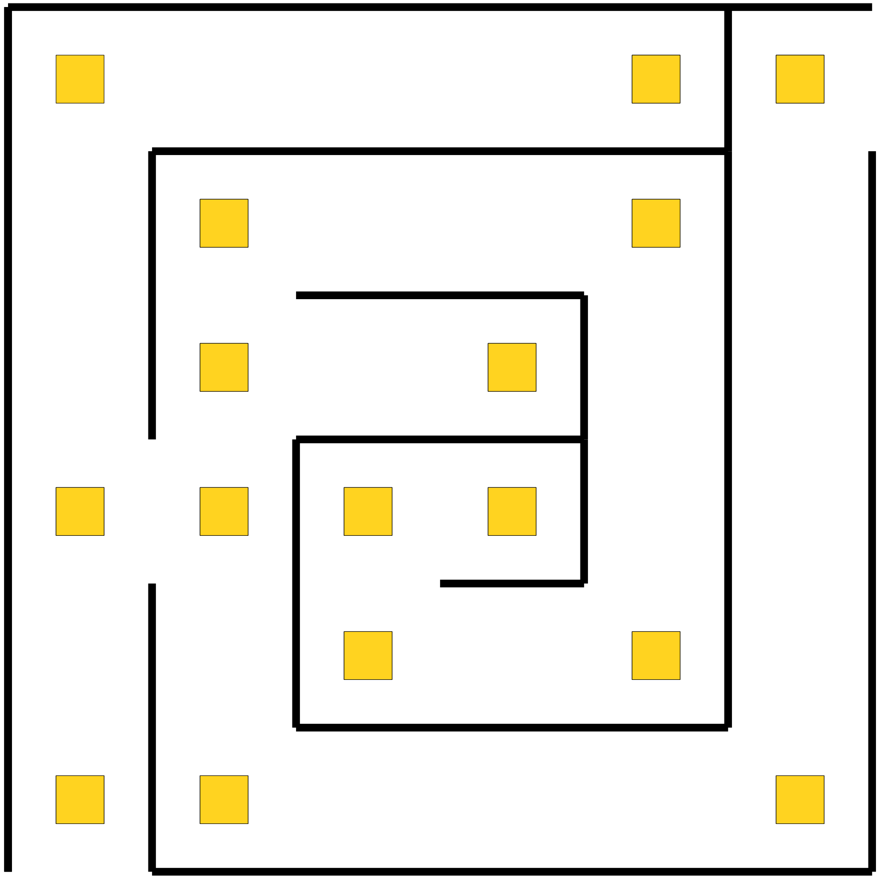
Secretary of Natural Resources

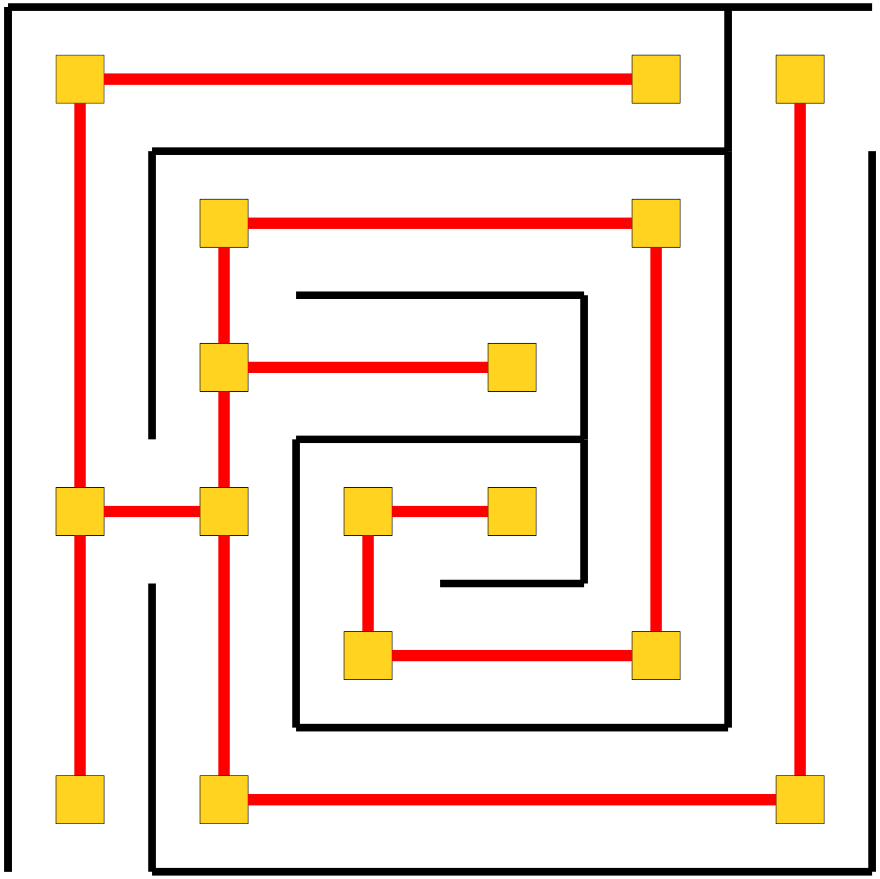
Secretary of Public Safety

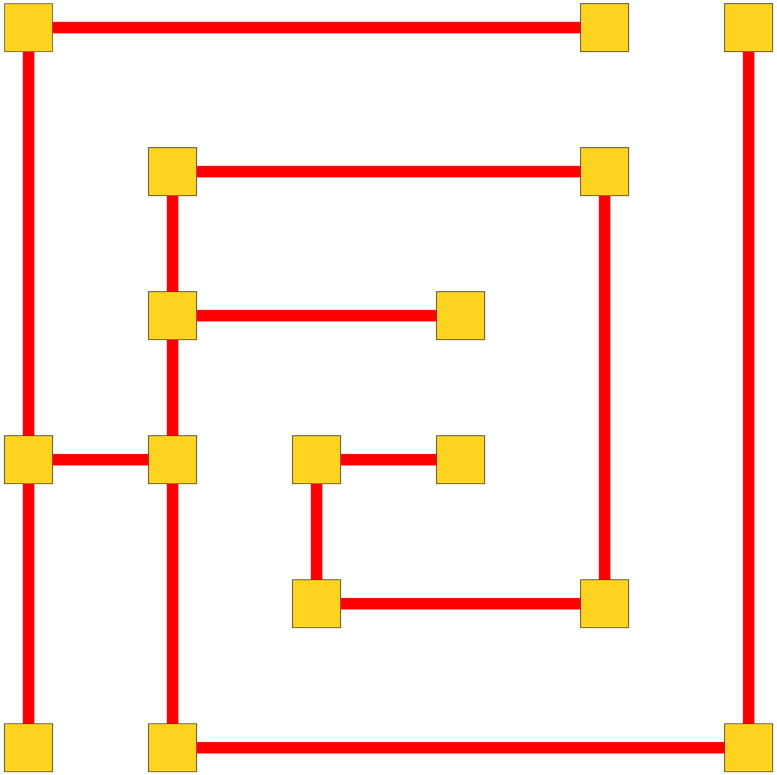
Secretary of Transportation

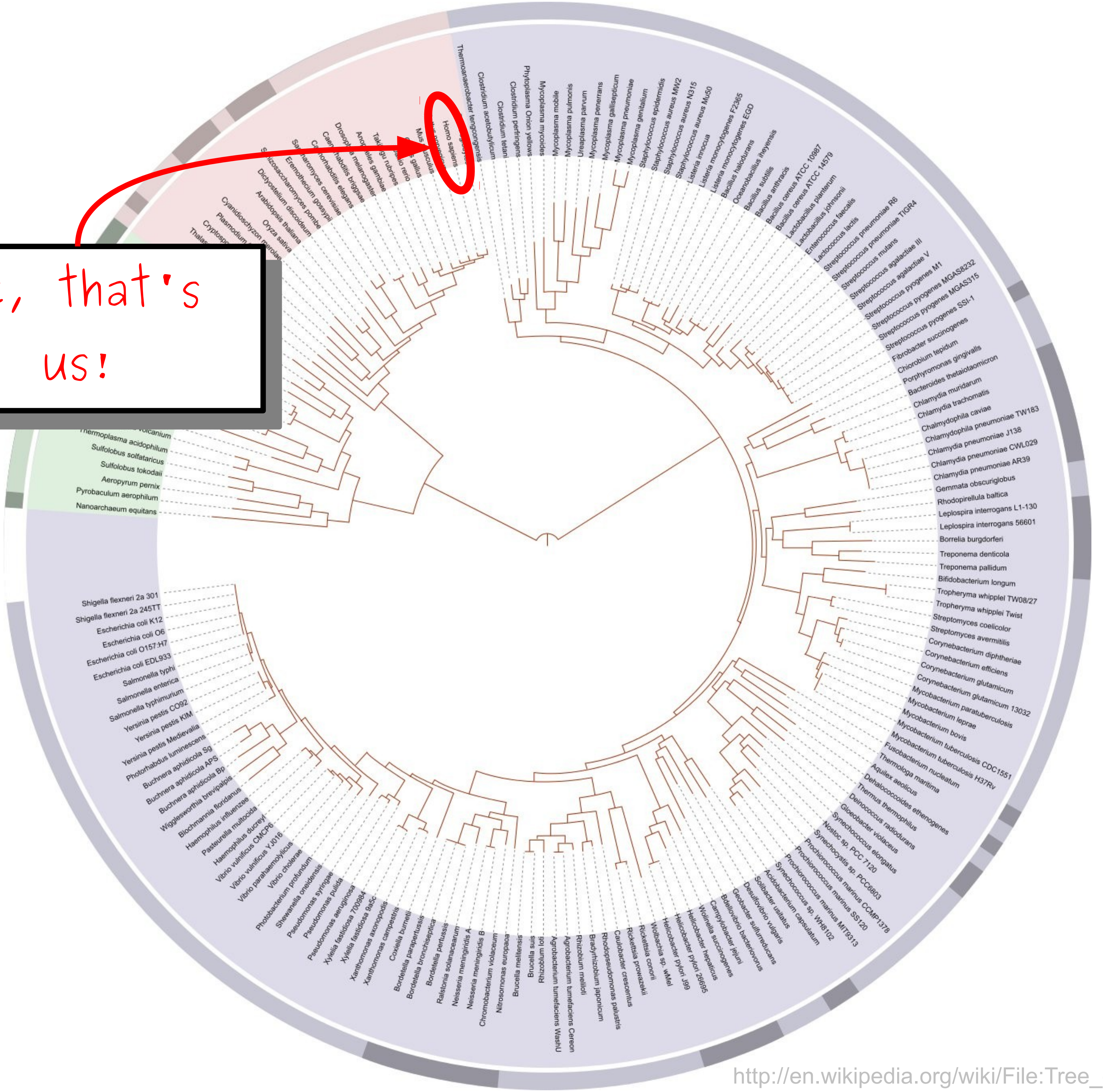
http://www.virginia.gov/images/wholeexecbranch2.jpg





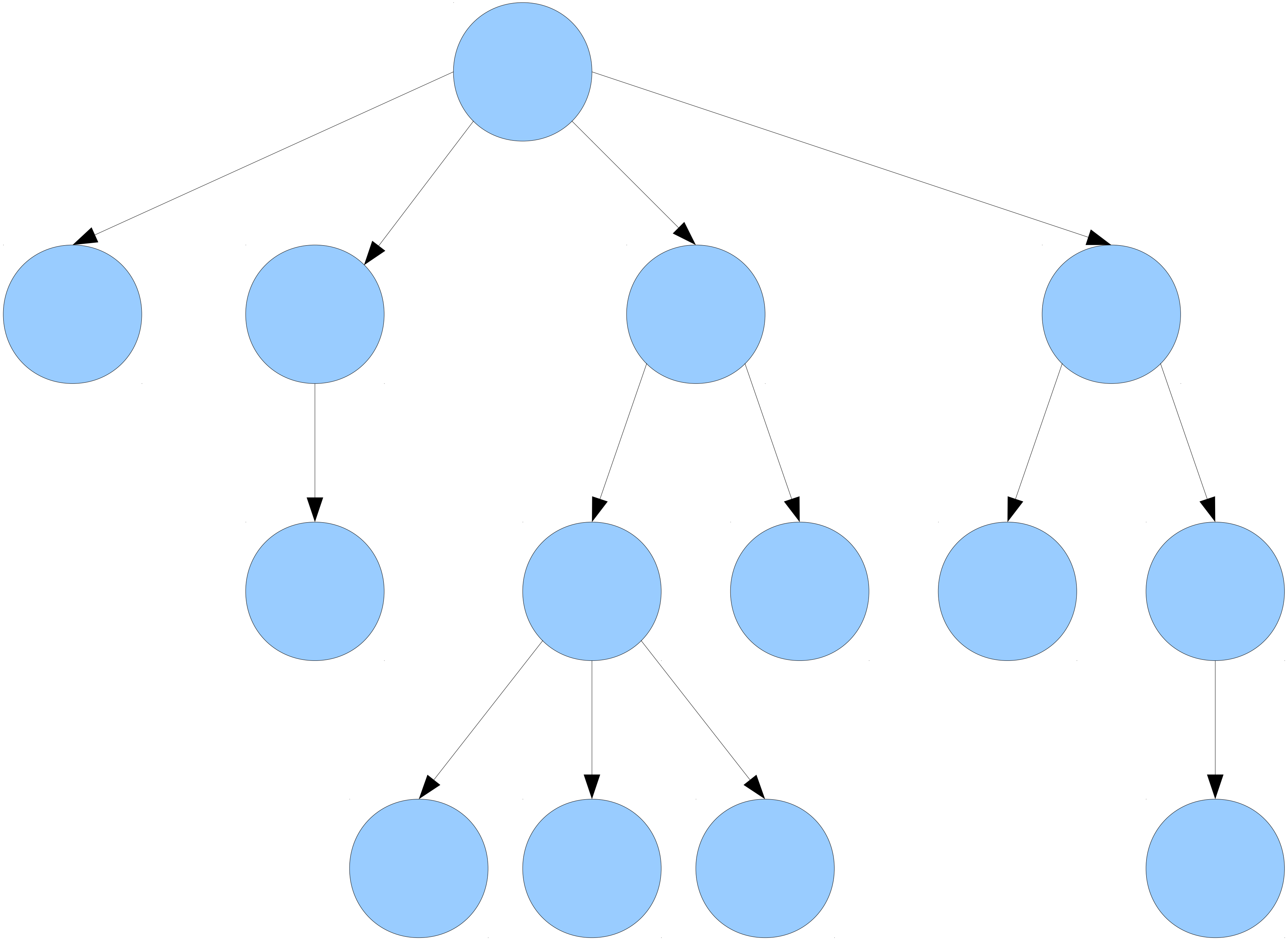






Hey, that's us!

http://en.wikipedia.org/wiki/File:Tree_of_life_SVG.svg



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

Goals for this Course

- **Learn how to model and solve complex problems with computers.**
- To that end:
  - Explore common abstractions for representing problems.
  - Harness recursion and understand how to think about problems recursively.
  - Quantitatively analyze different approaches for solving problems.

Goals for this Course

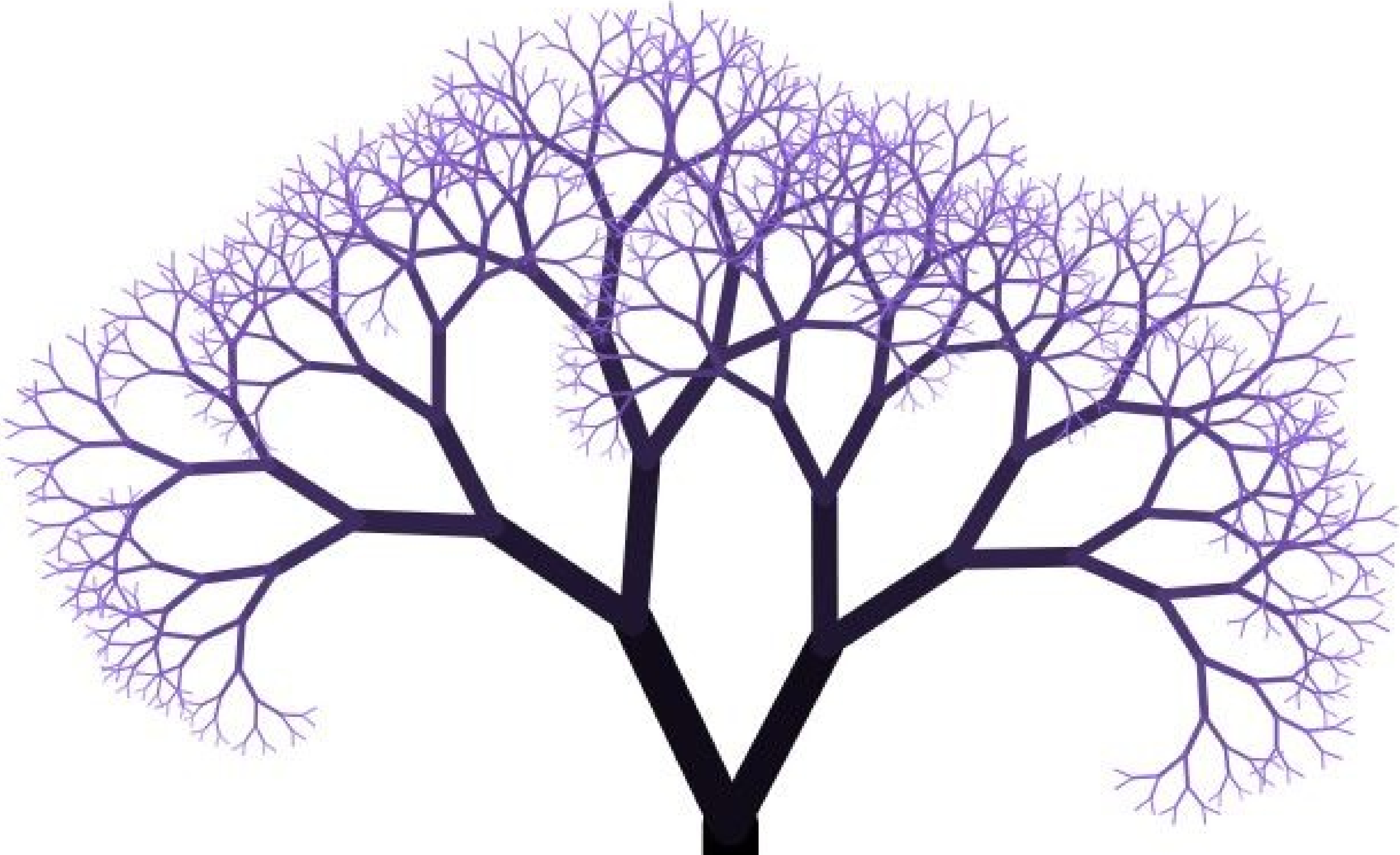
Learn how to model and solve complex problems with computers.

To that end:

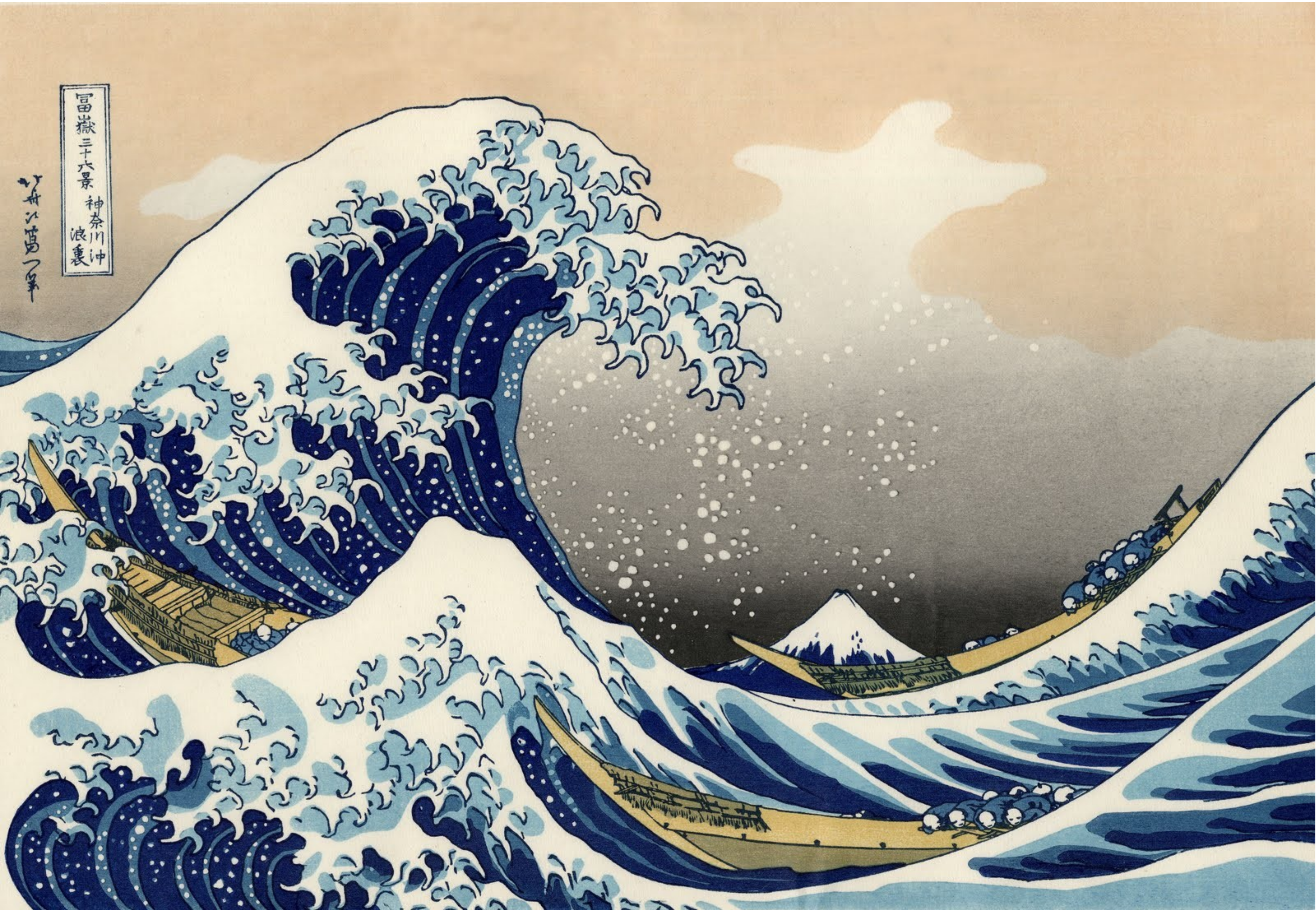
Explore common abstractions for representing problems.

- Harness recursion and understand how to think about problems recursively.

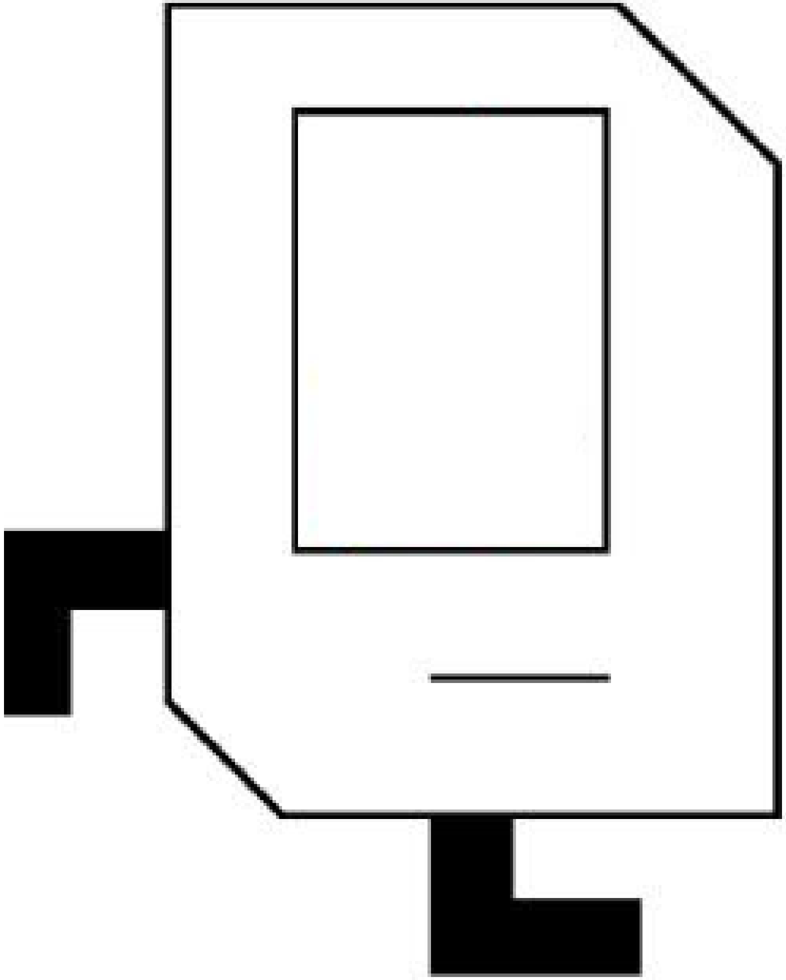
Quantitatively analyze different approaches for solving problems.

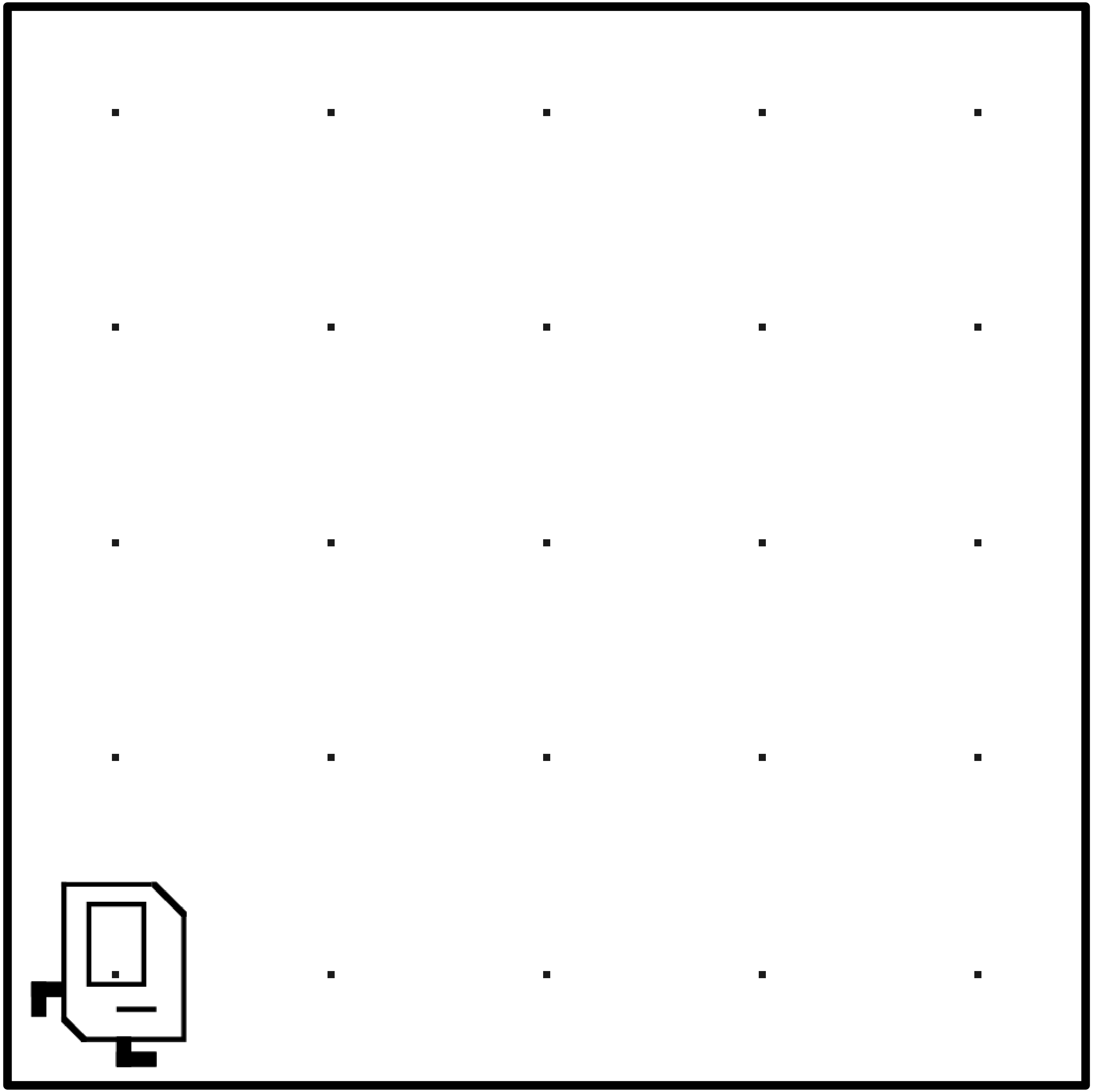


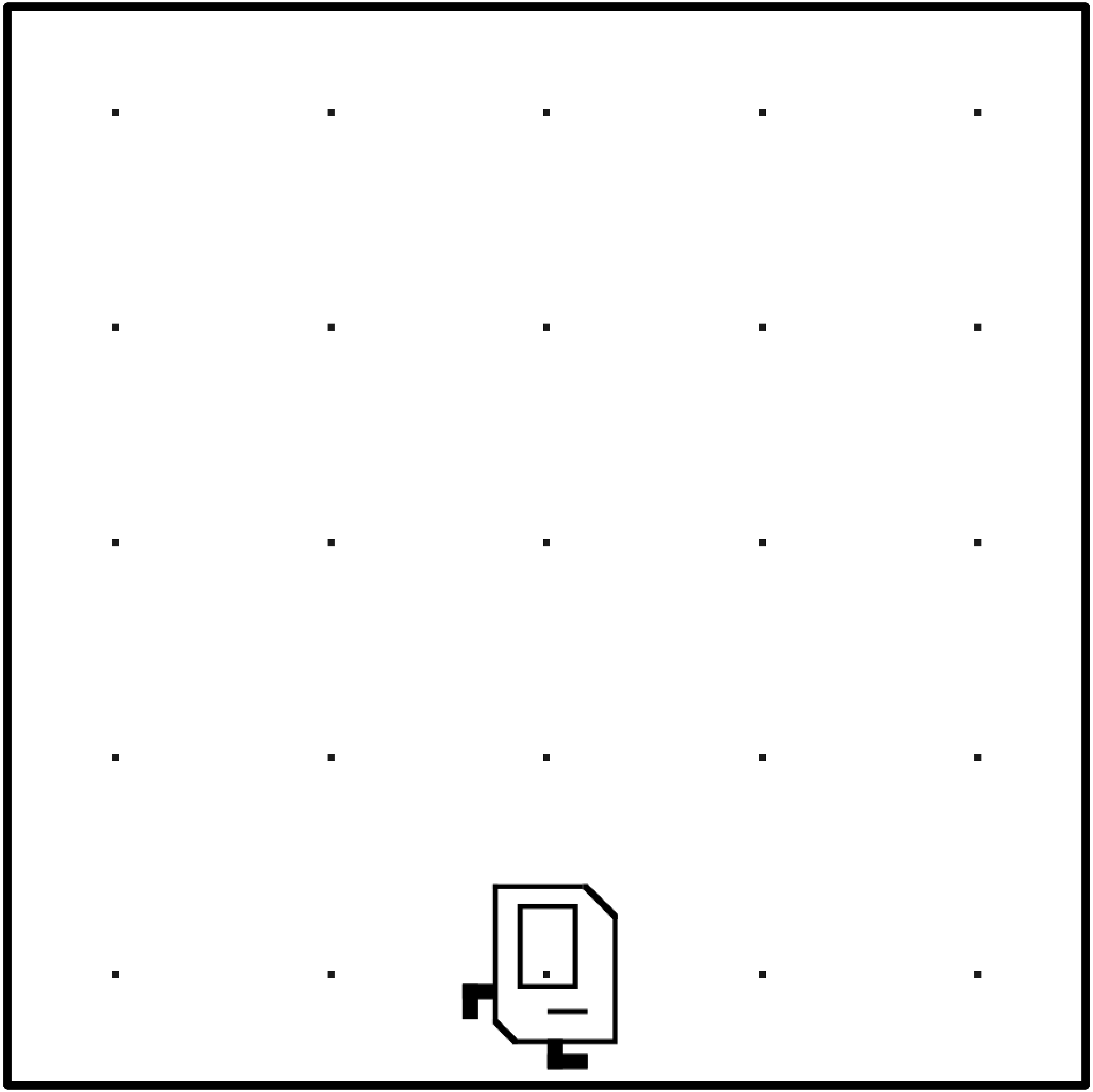
http://www.marketoracle.co.uk/images/2010/Oct/fractal-tree2.jpg



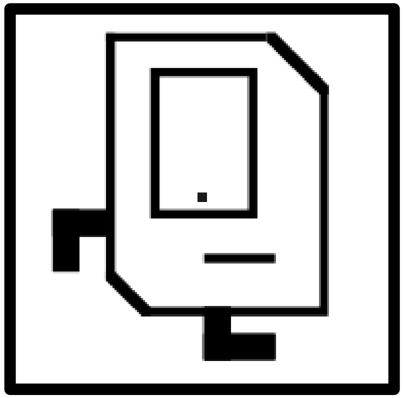
http://1.bp.blogspot.com/-Hwf-U9hRwEI/TX480ONoo6I/AAAAAAAAAcM/fyBi0I5HsUI/s1600/the_great_wave_off_kanagawa.jpg



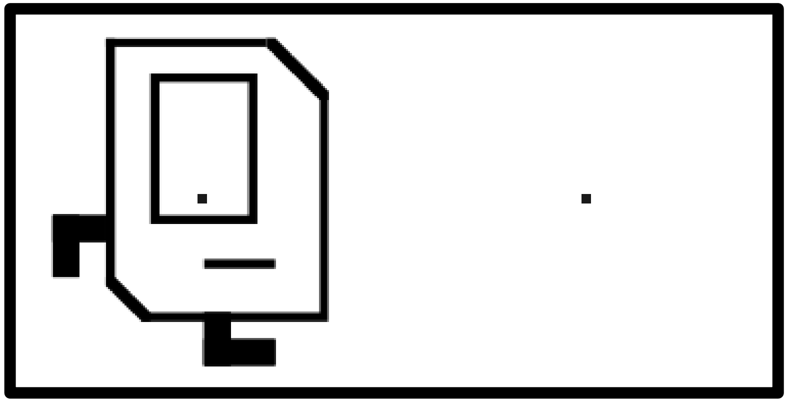


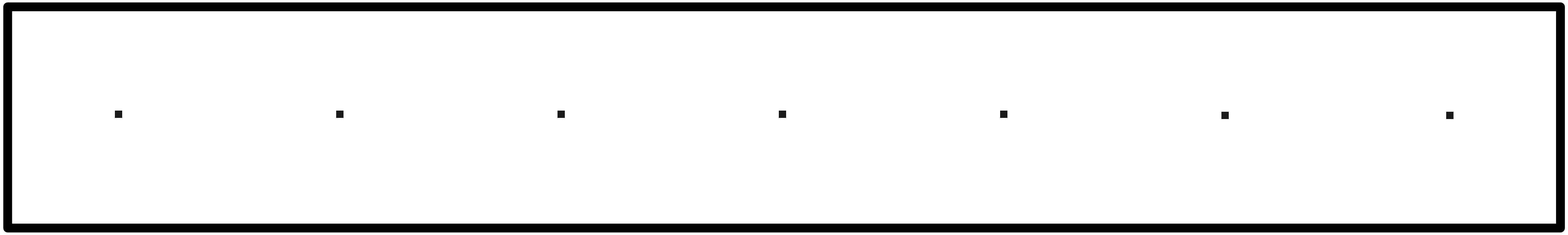


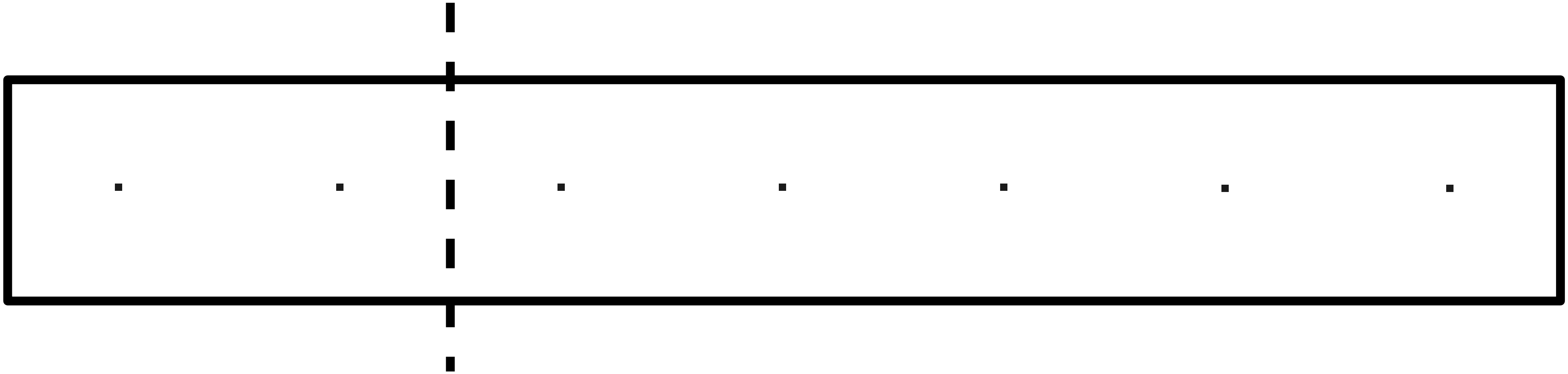
Width 1

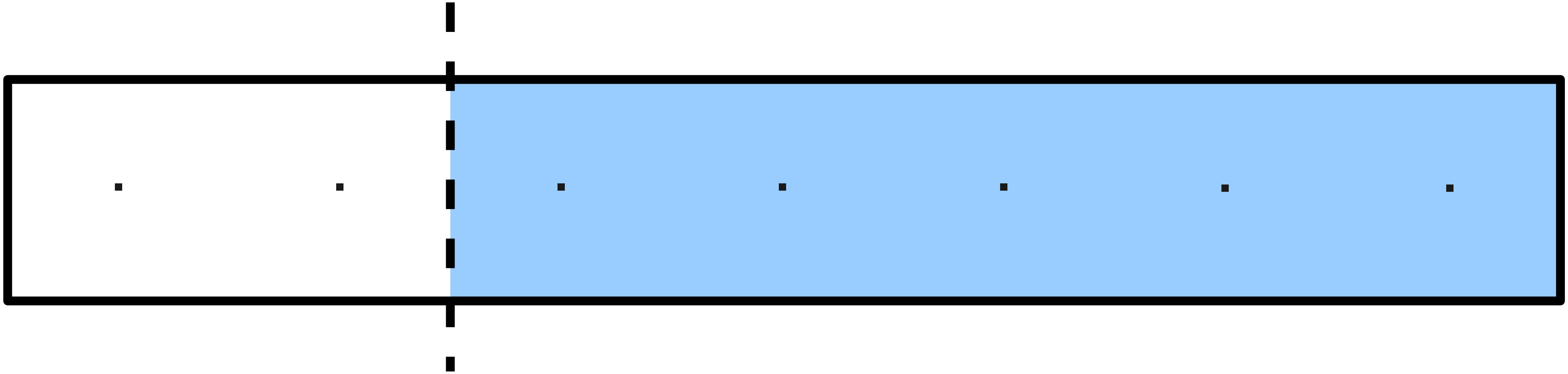


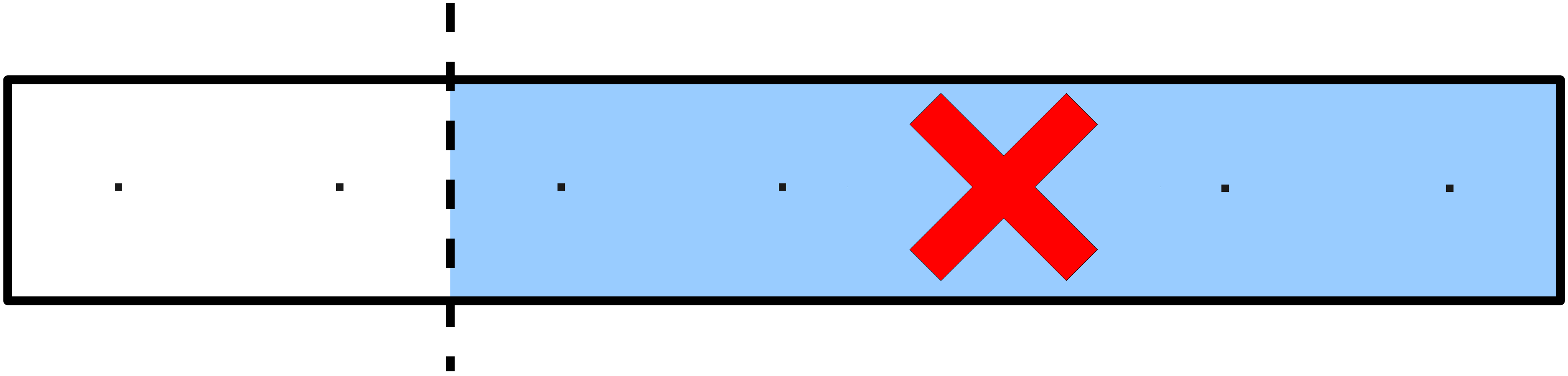
Width 2

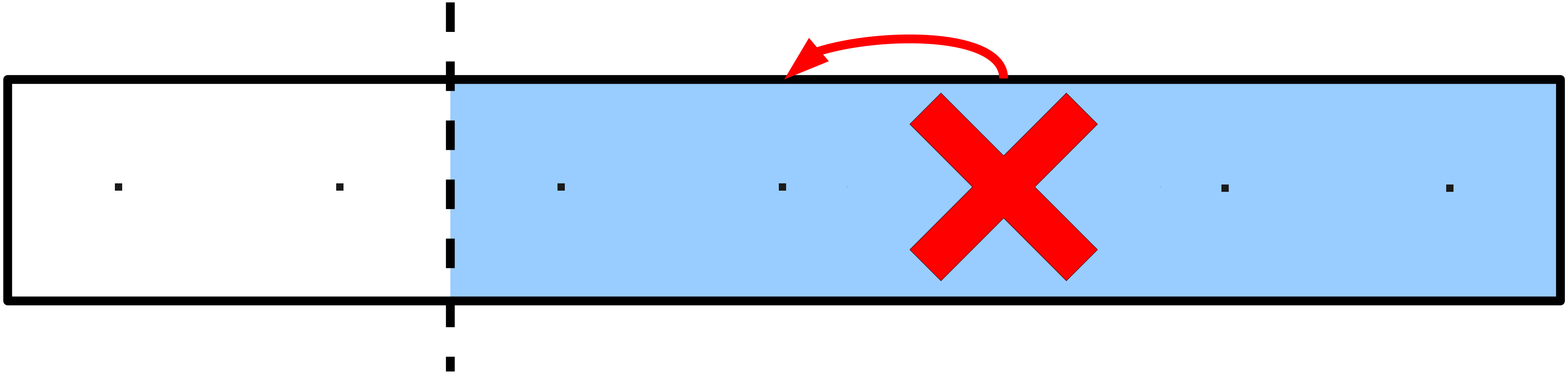


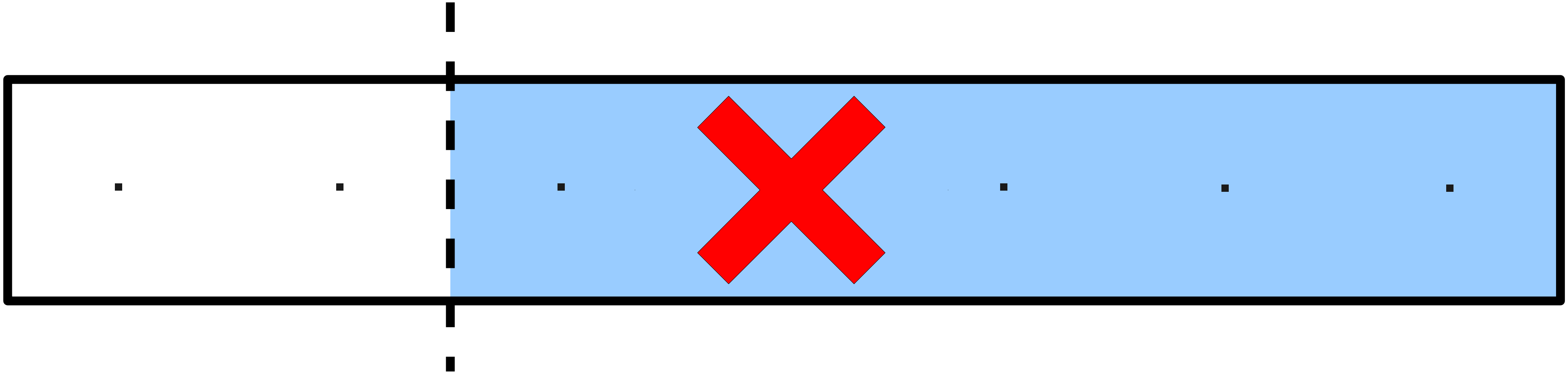


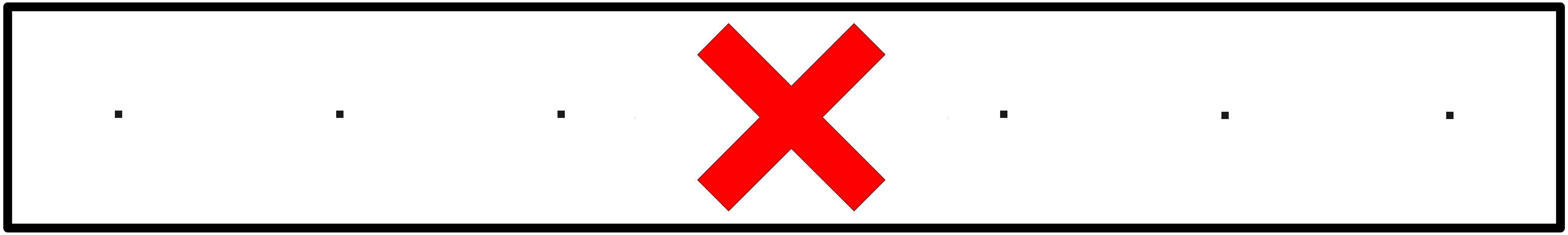












Finding the Midpoint

- If the width is 1, Karel is standing on the midpoint.
- If the width is 2, either position can be considered the midpoint.
- Otherwise:
  - Take two steps forward.
  - Find the midpoint of the rest of the world.
  - Take one step backward.

A Surprisingly Short Solution

A **recursive solution** is a solution that is defined in terms of itself.

Thinking recursively allows you to solve an enormous class of problems cleanly and concisely.

Goals for this Course

- **Learn how to model and solve complex problems with computers.**
- To that end:
  - Explore common abstractions for representing problems.
  - Harness recursion and understand how to think about problems recursively.
  - Quantitatively analyze different approaches for solving problems.

Goals for this Course

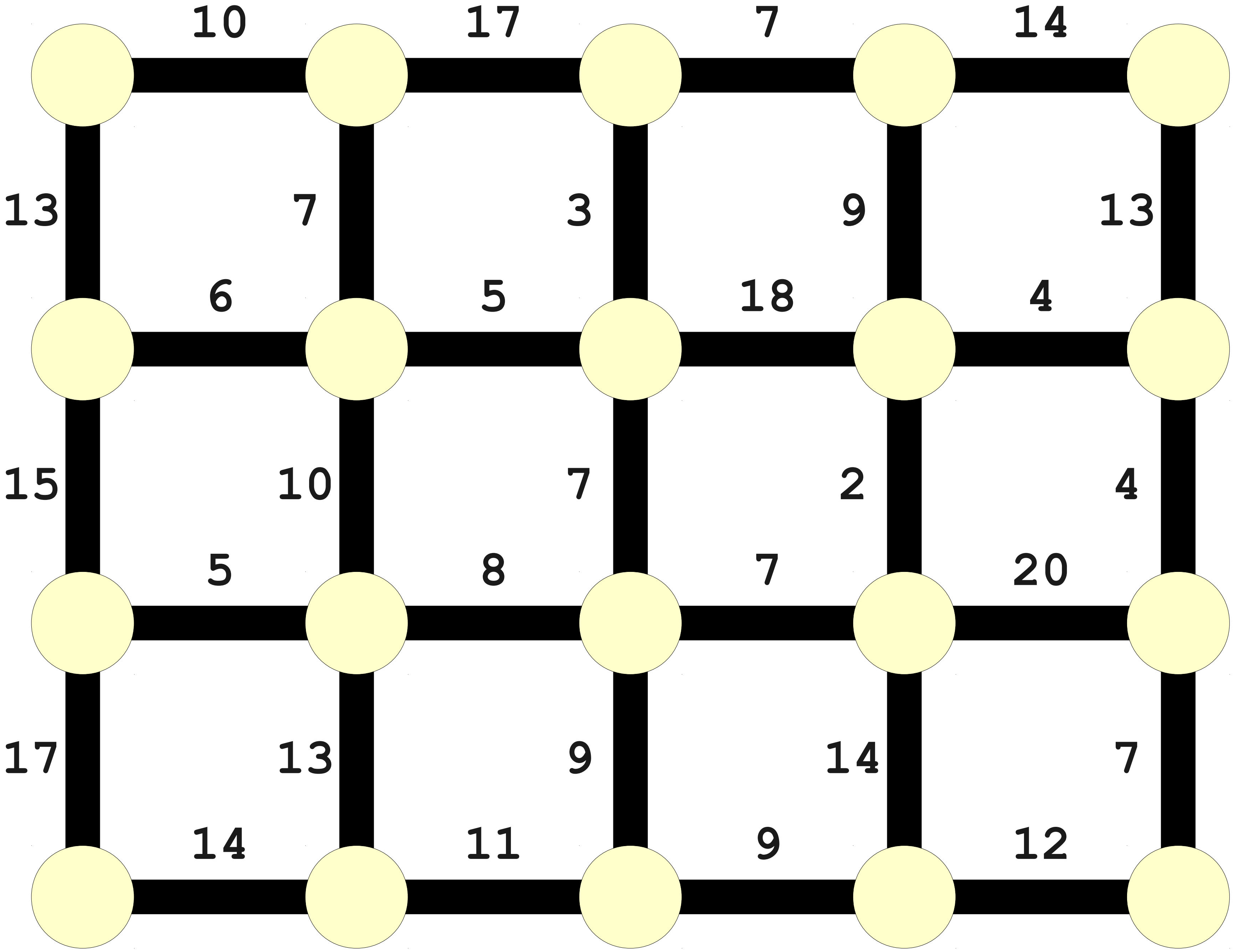
Learn how to model and solve complex problems with computers.

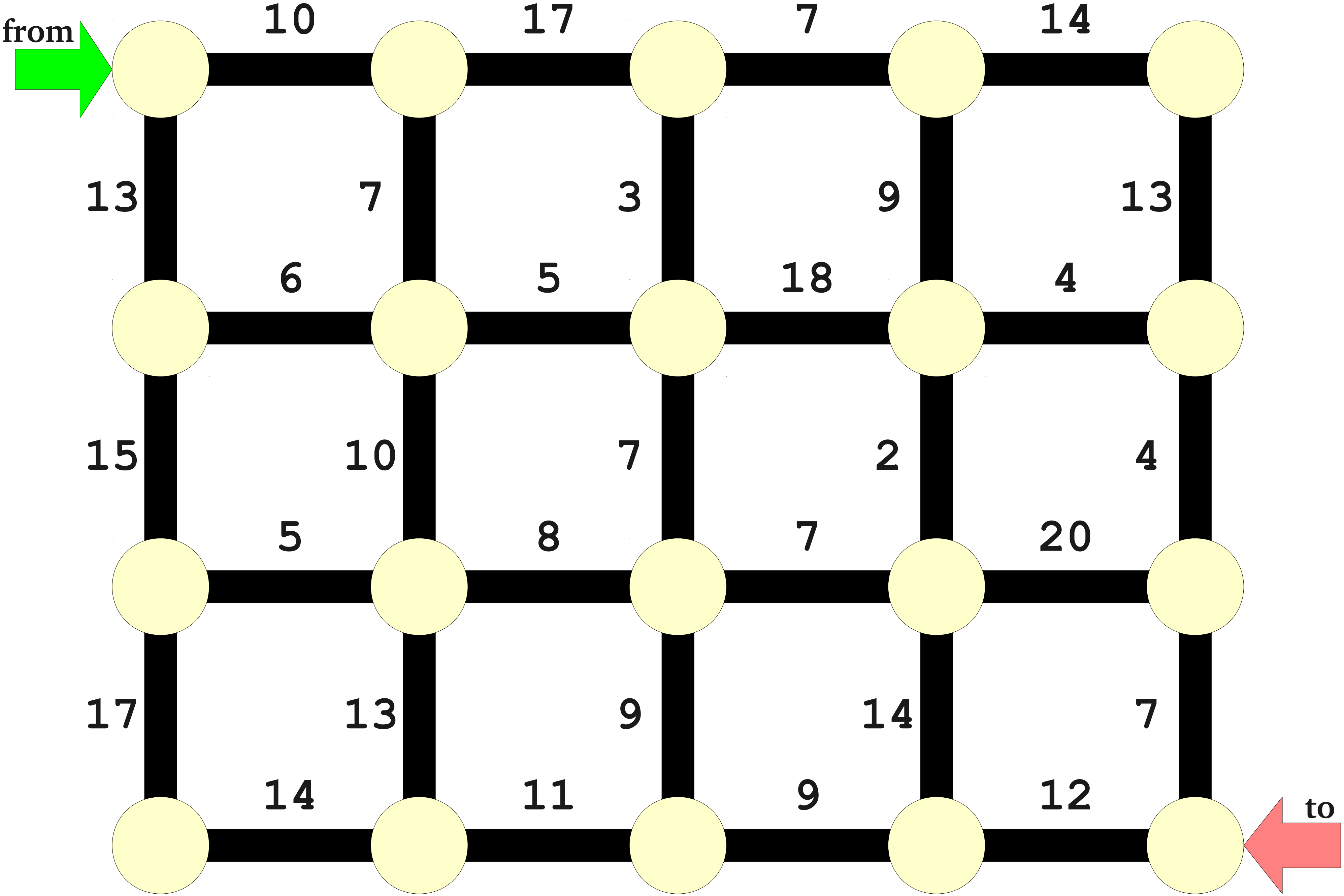
To that end:

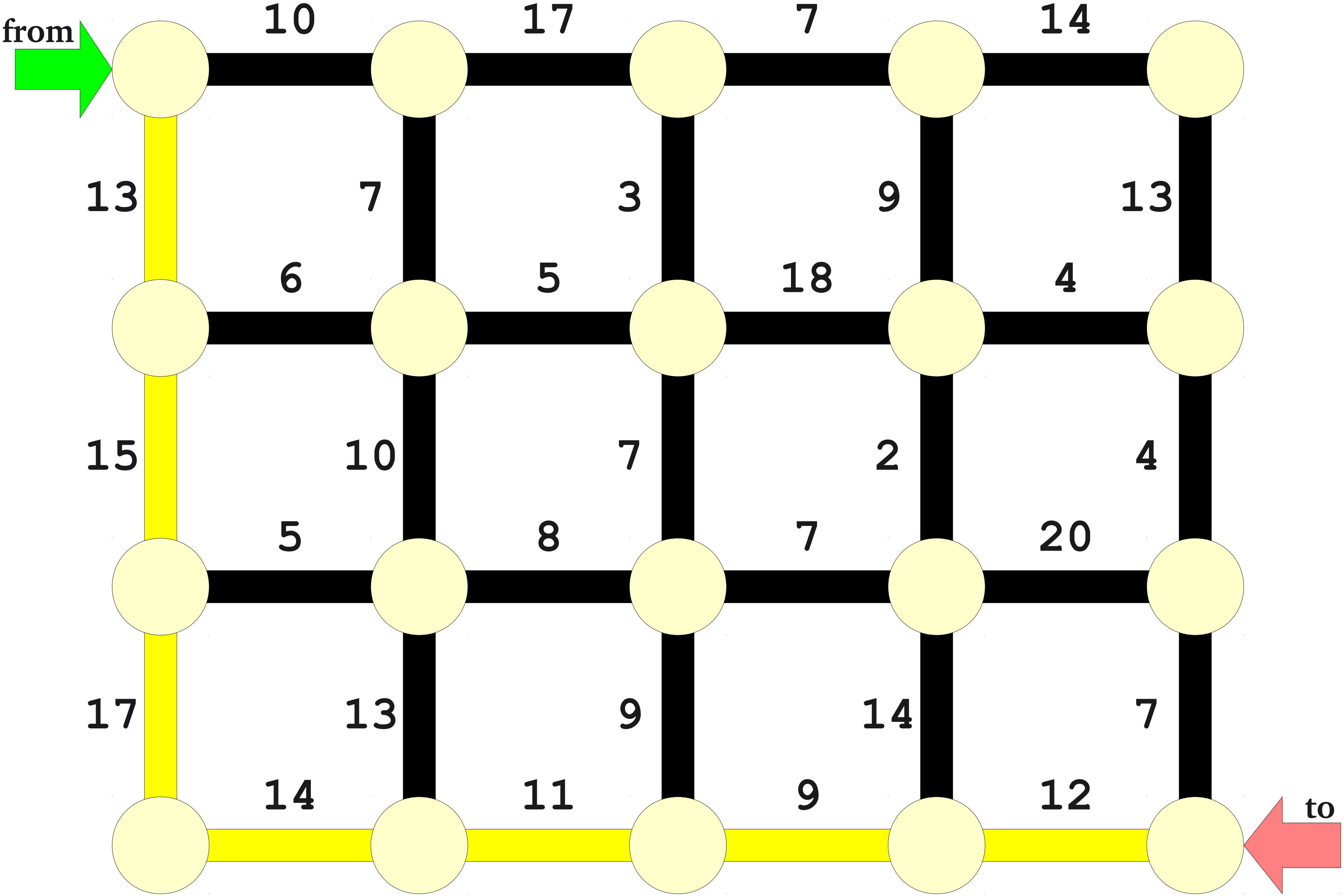
Explore common abstractions for representing problems.

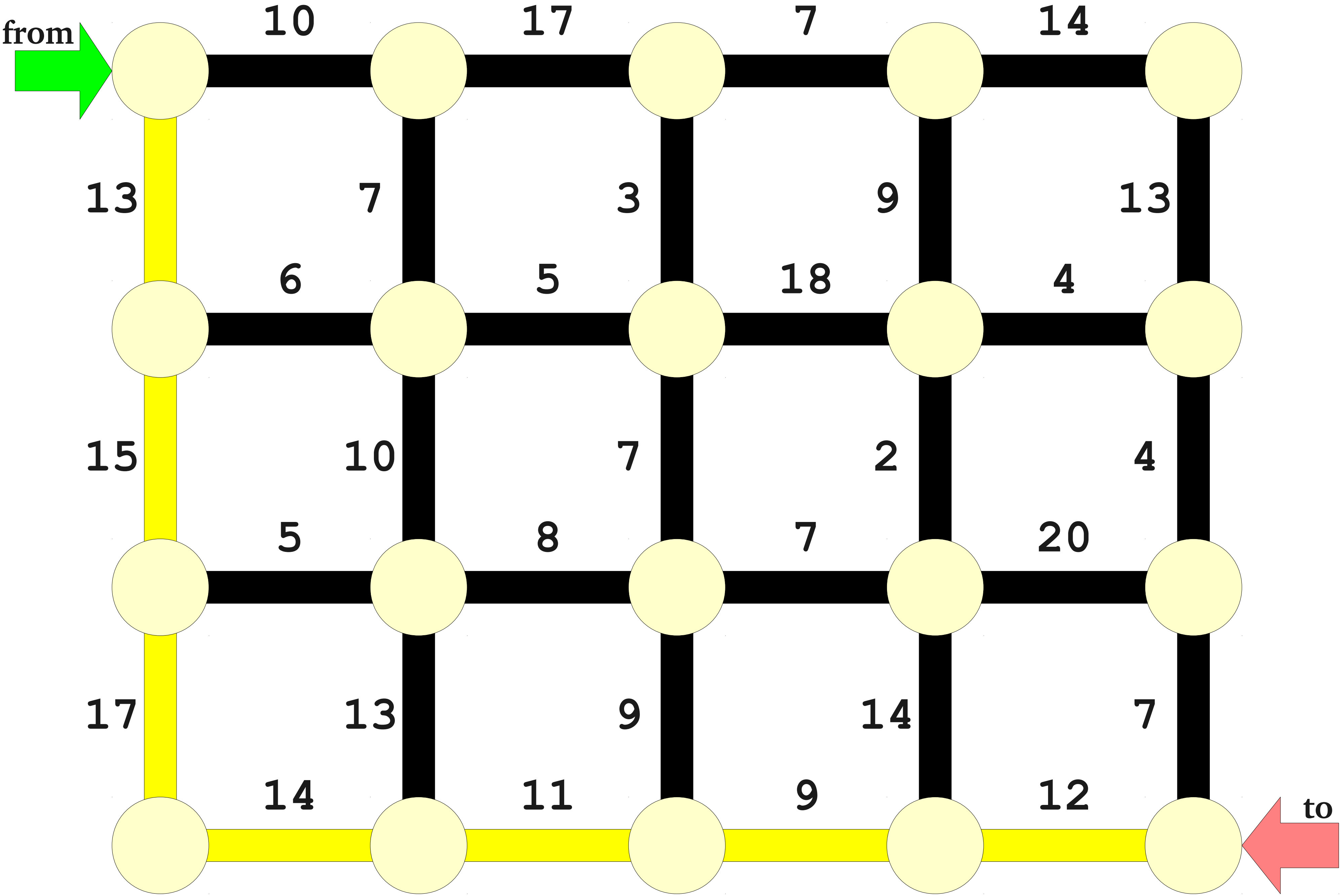
Harness recursion and understand how to think about problems recursively.

- Quantitatively analyze different approaches for solving problems.

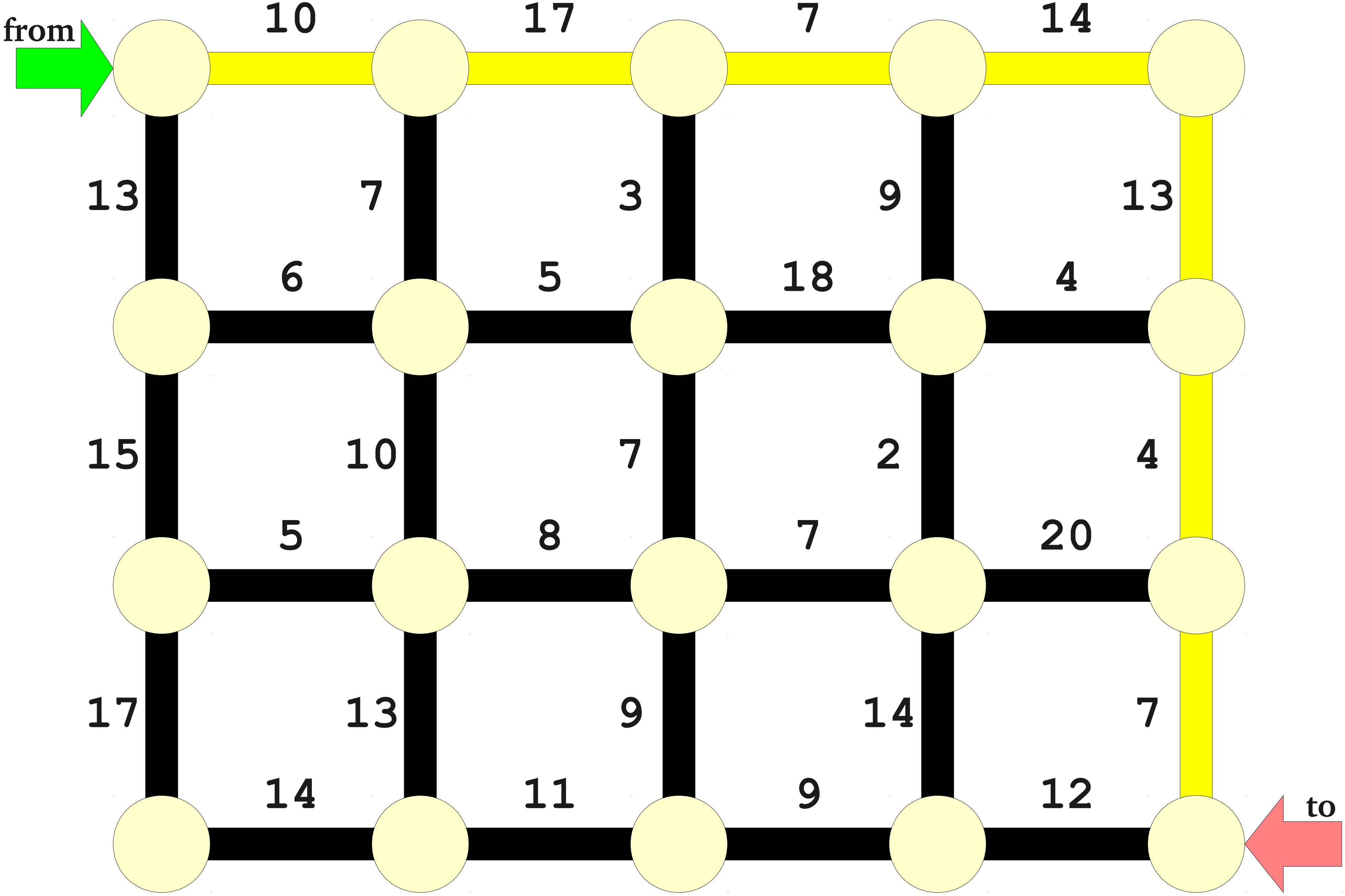

10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12

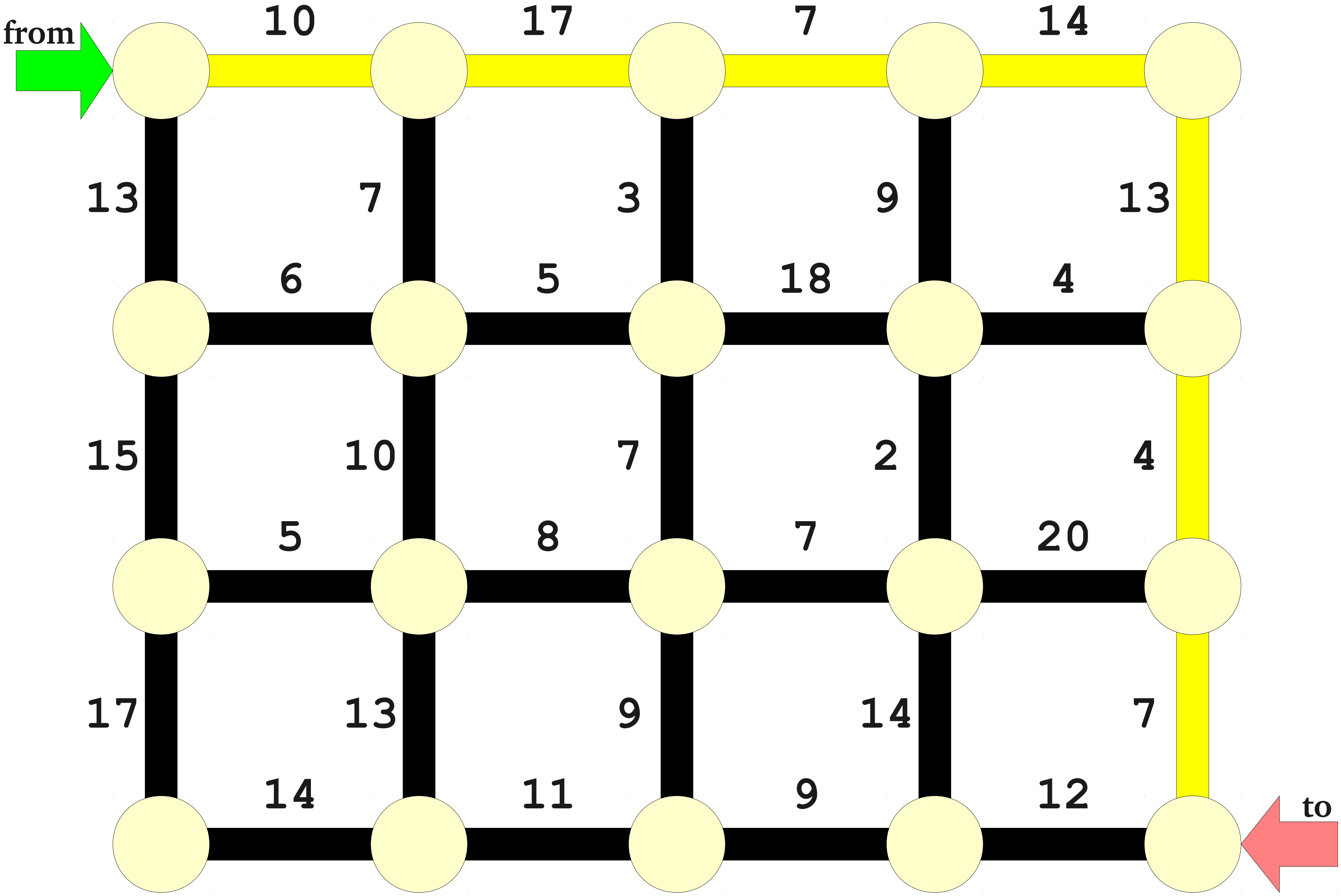

from
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to


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

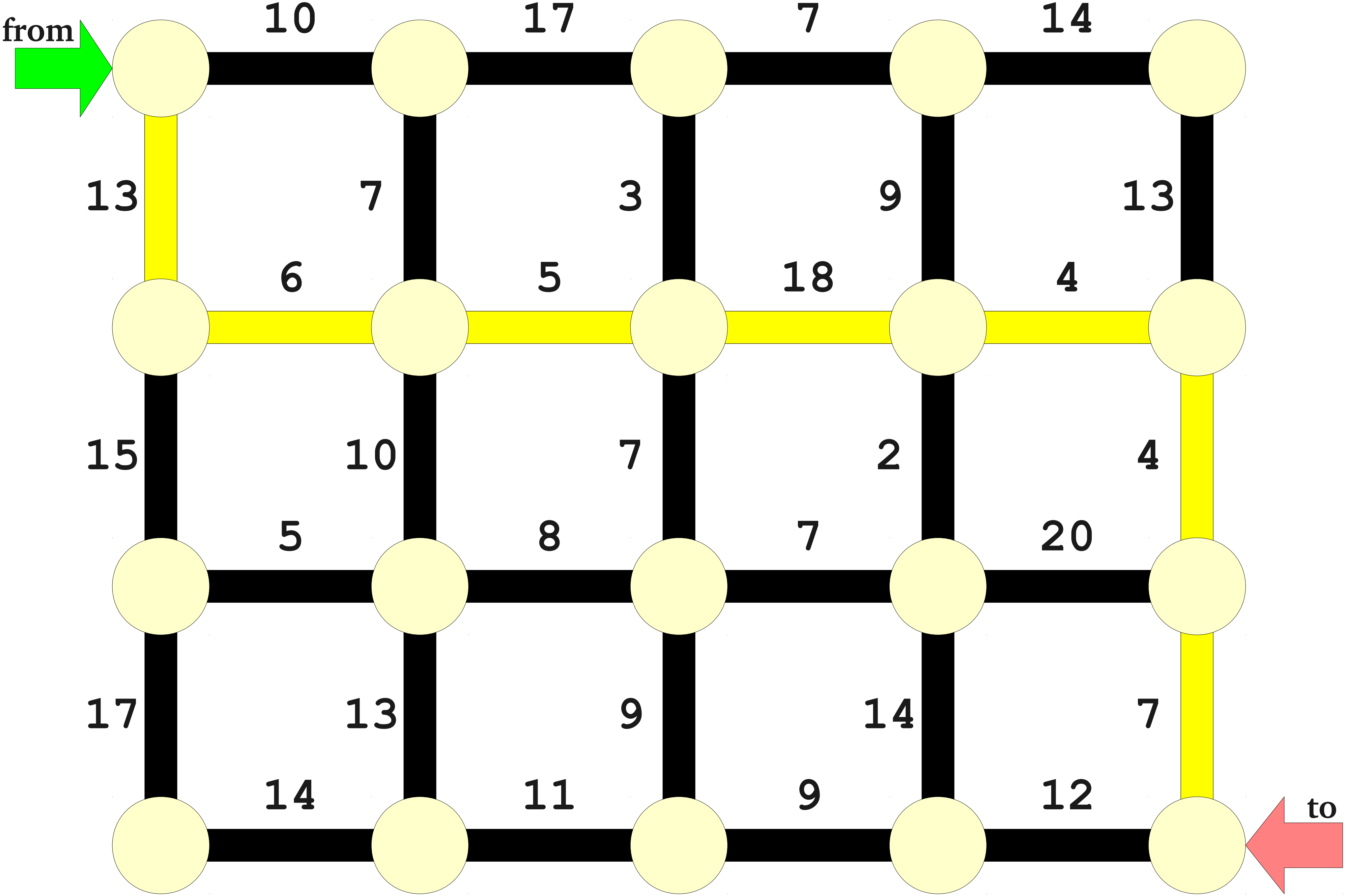


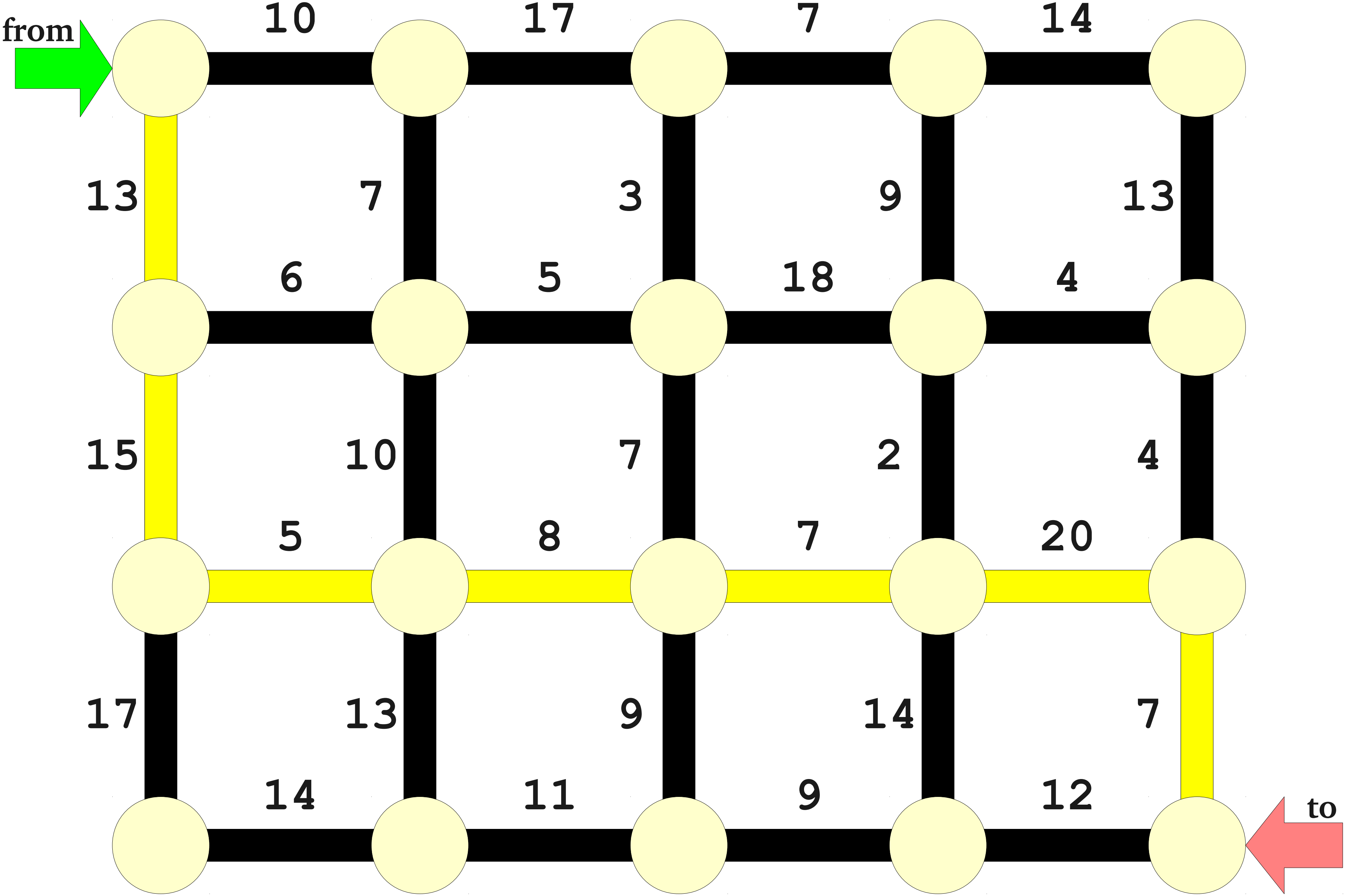
Travel Time: 13 + 15 + 17 + 14 + 11 + 9 + 12 = **91**

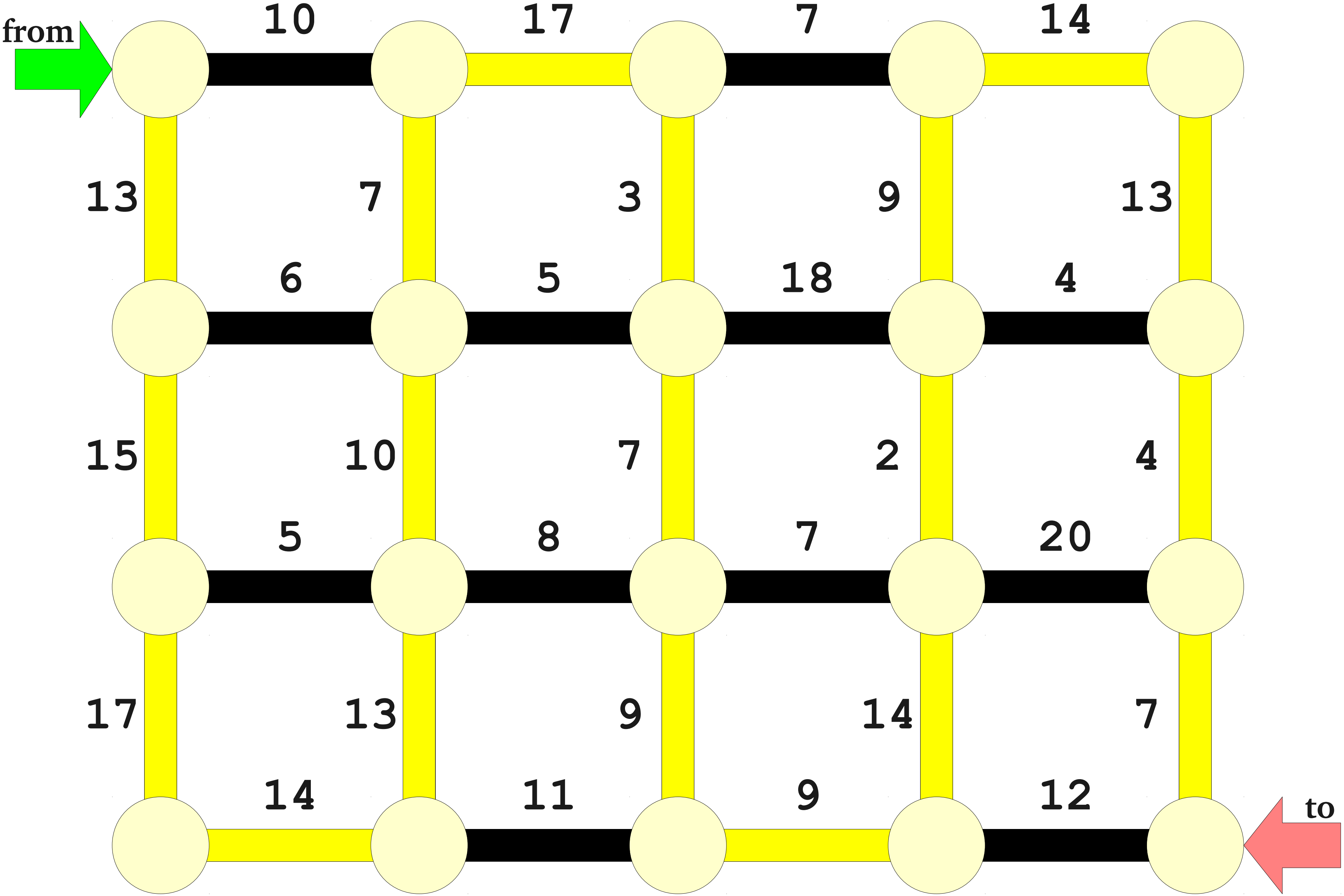

from
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to

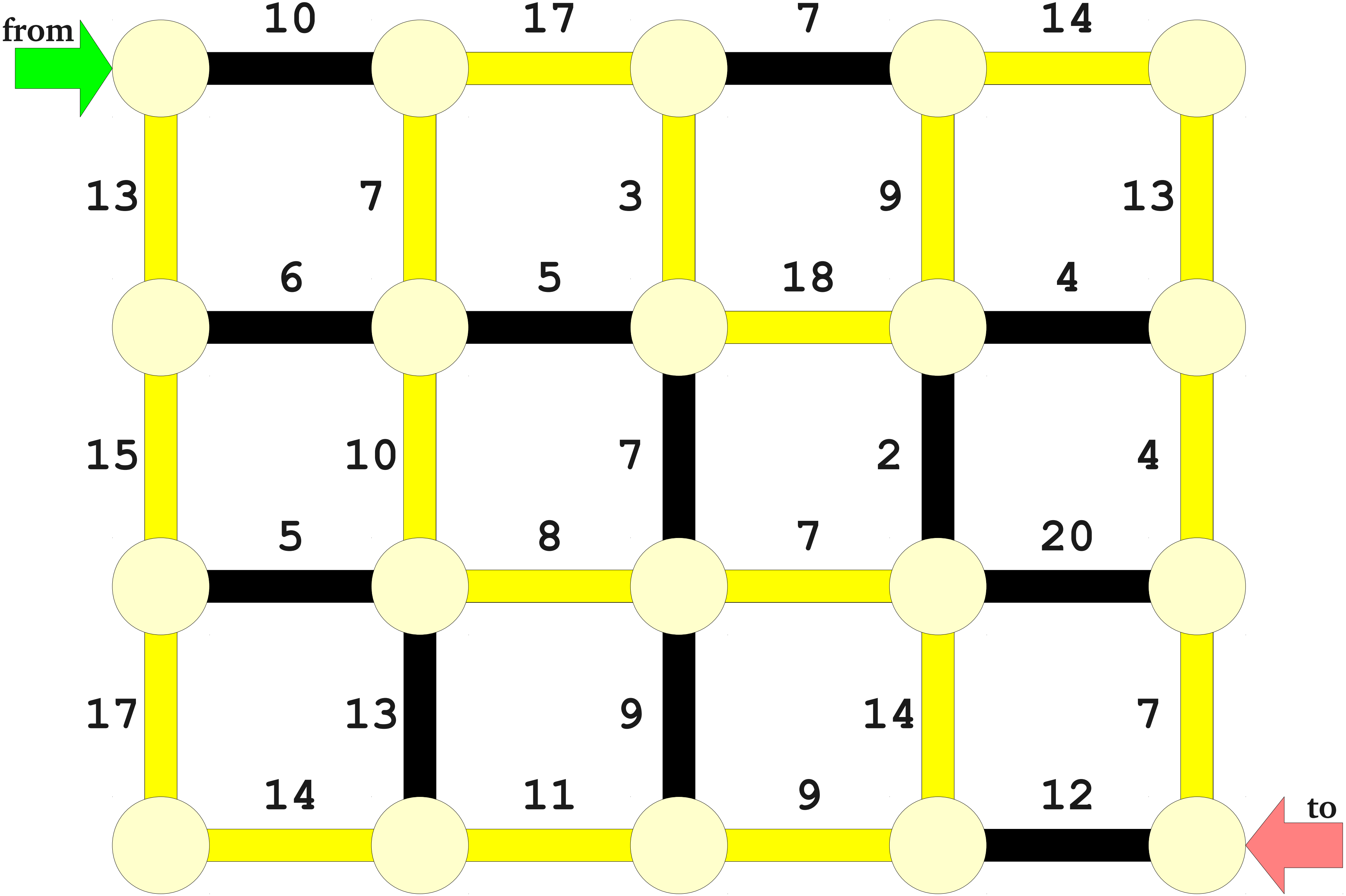


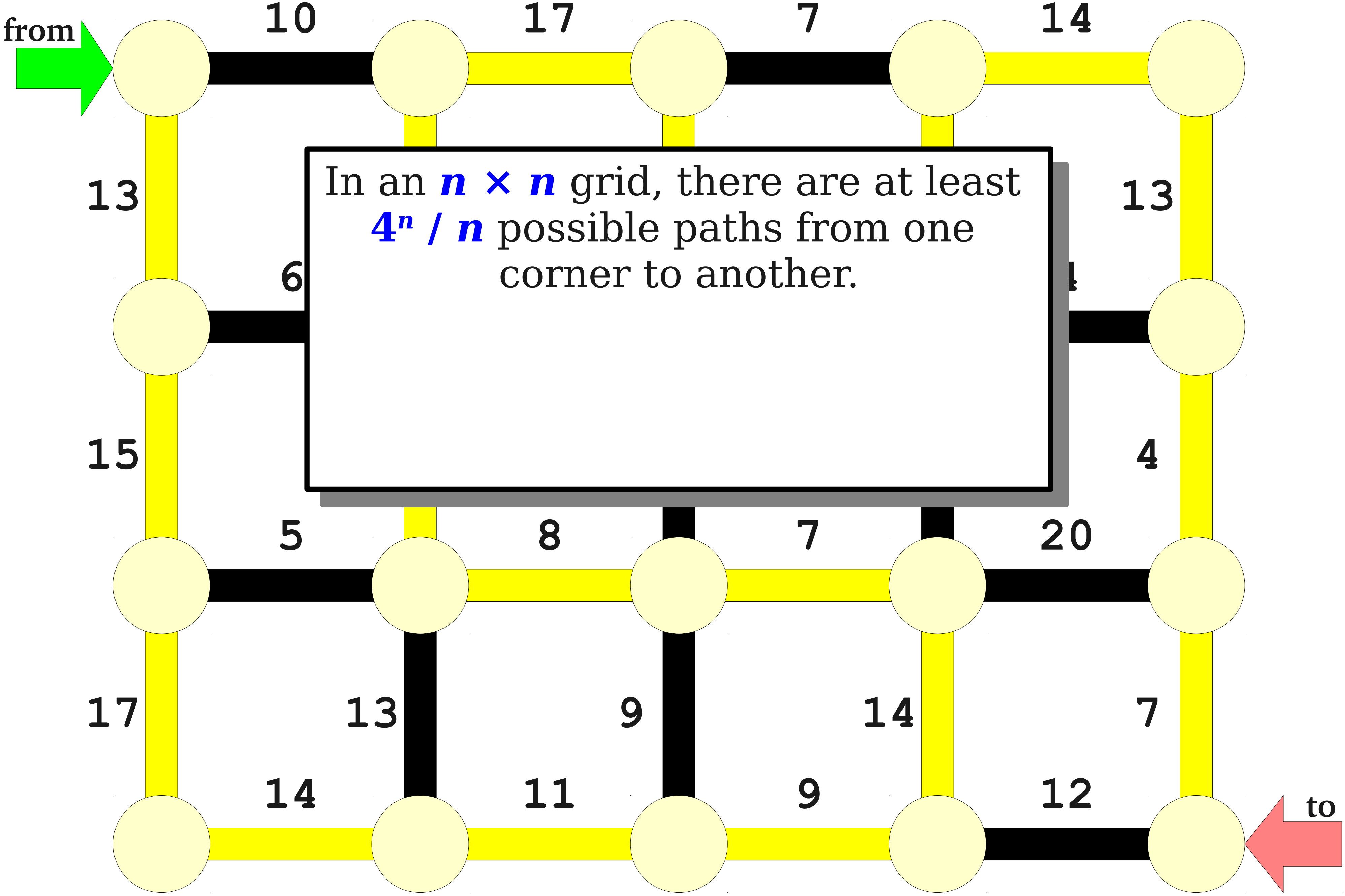
Travel Time: 10 + 17 + 7 + 14 + 13 + 4 + 7 = **72**


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

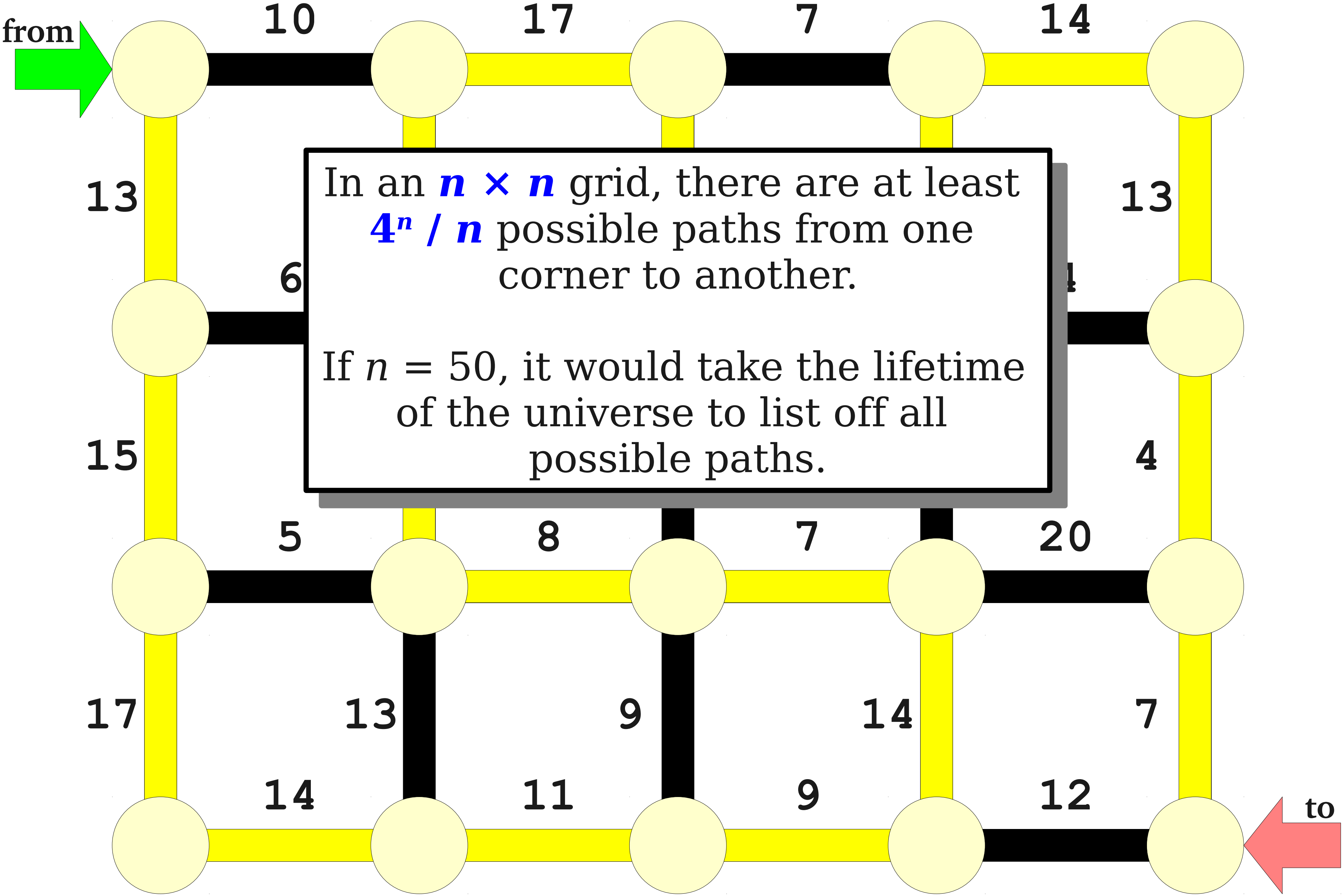

from
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to


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

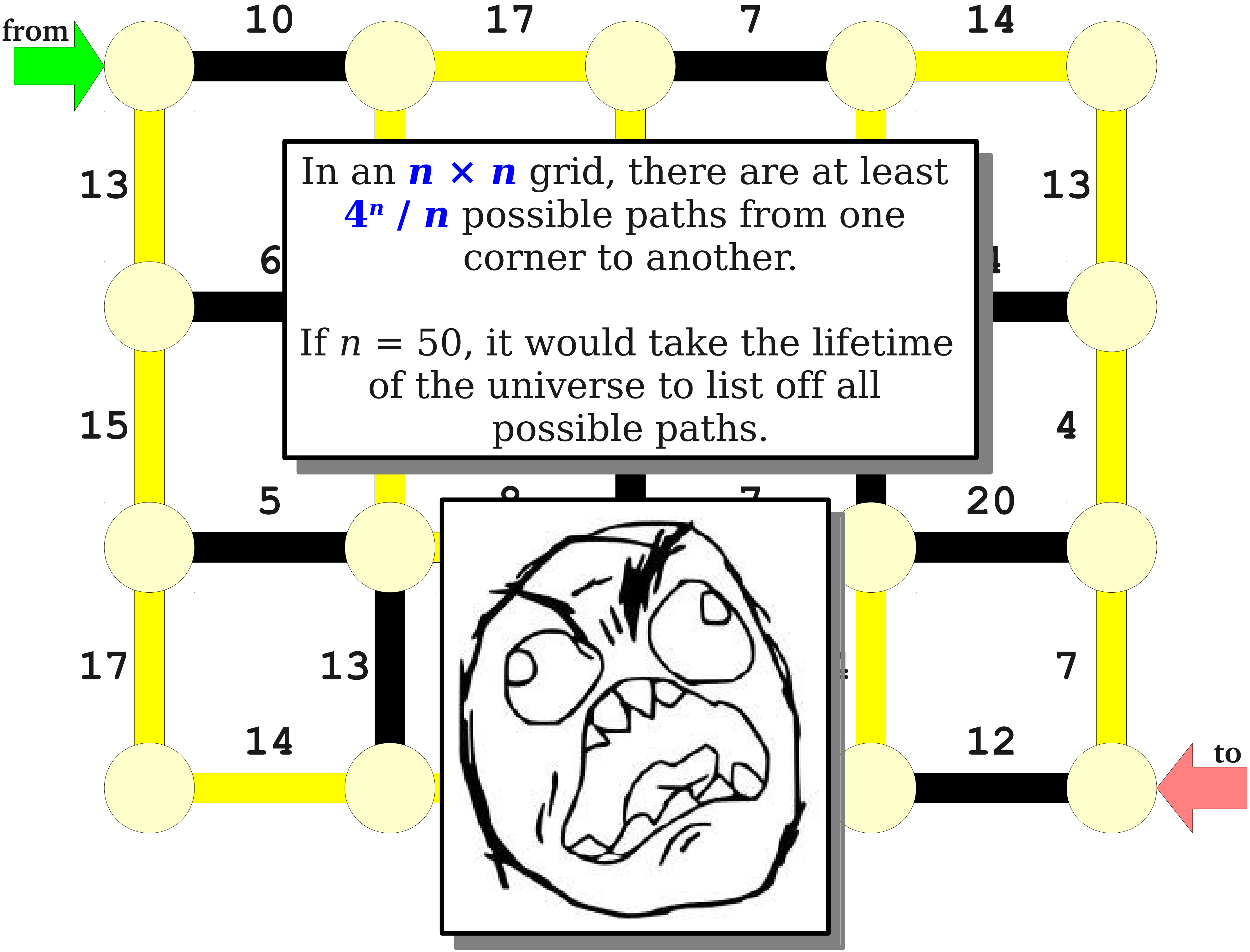


In an $n \times n$ grid, there are at least $4^n / n$ possible paths from one corner to another.



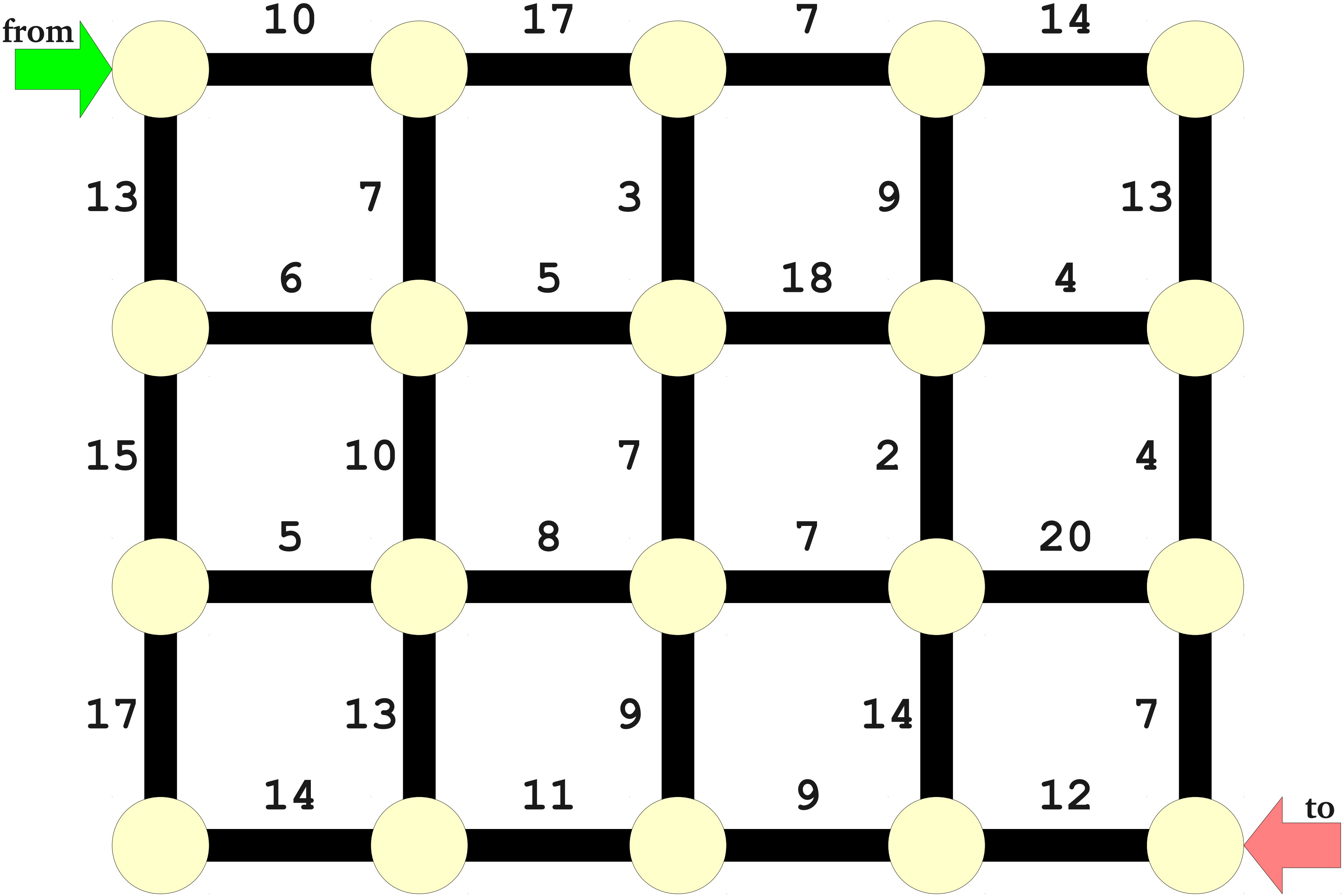
In an ***n* × *n*** grid, there are at least **4^n / *n*** possible paths from one corner to another.

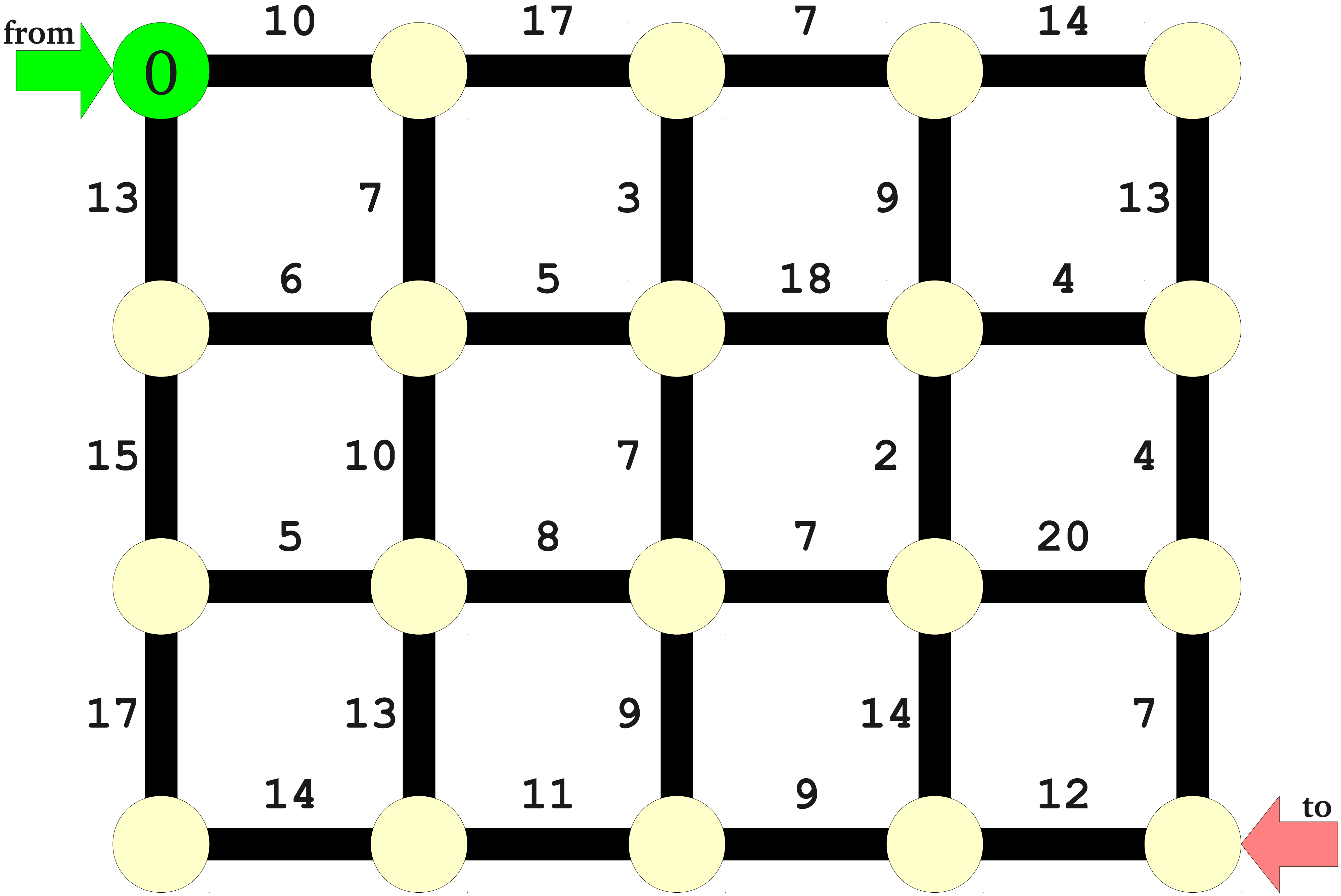
If n = 50, it would take the lifetime of the universe to list off all possible paths.

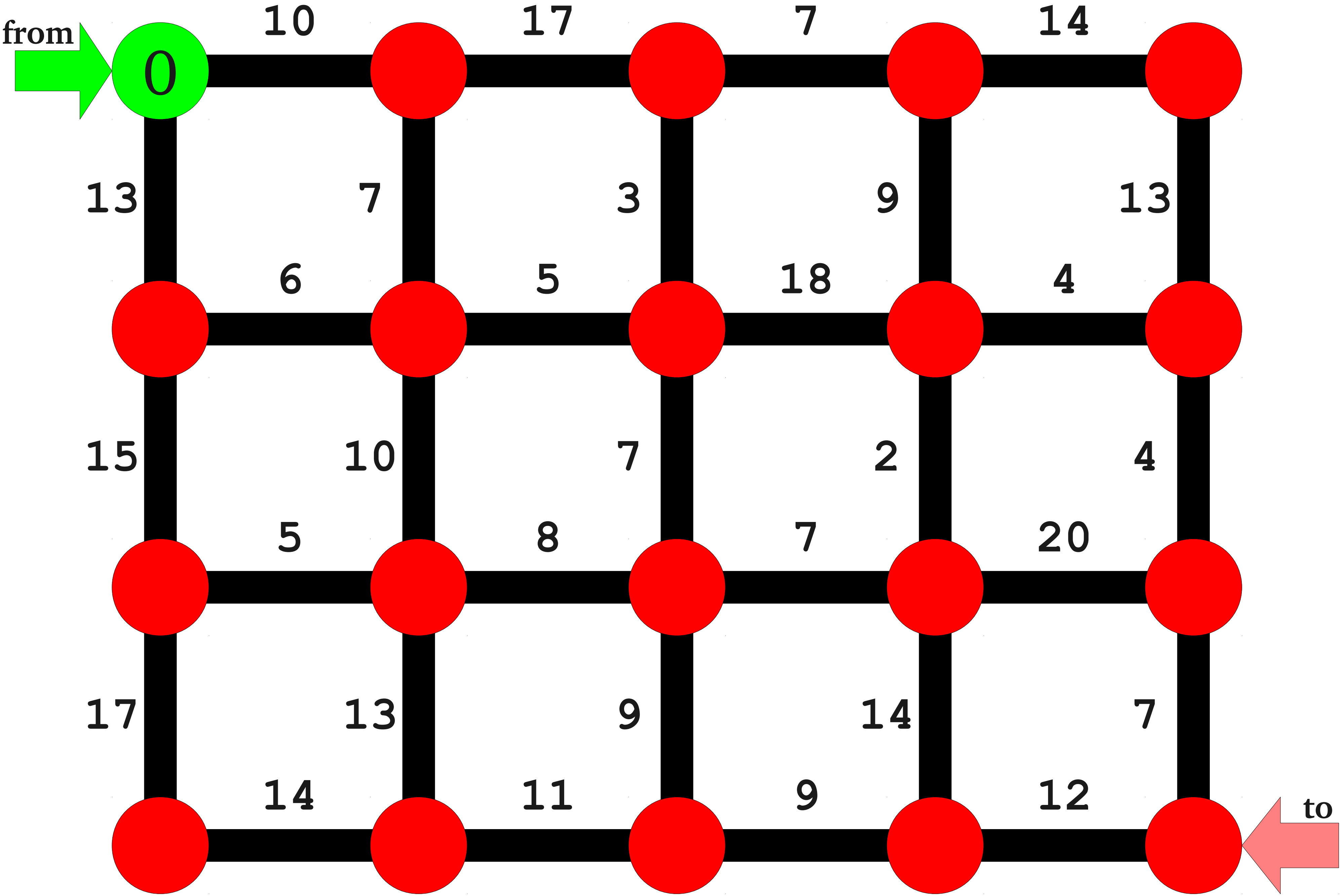


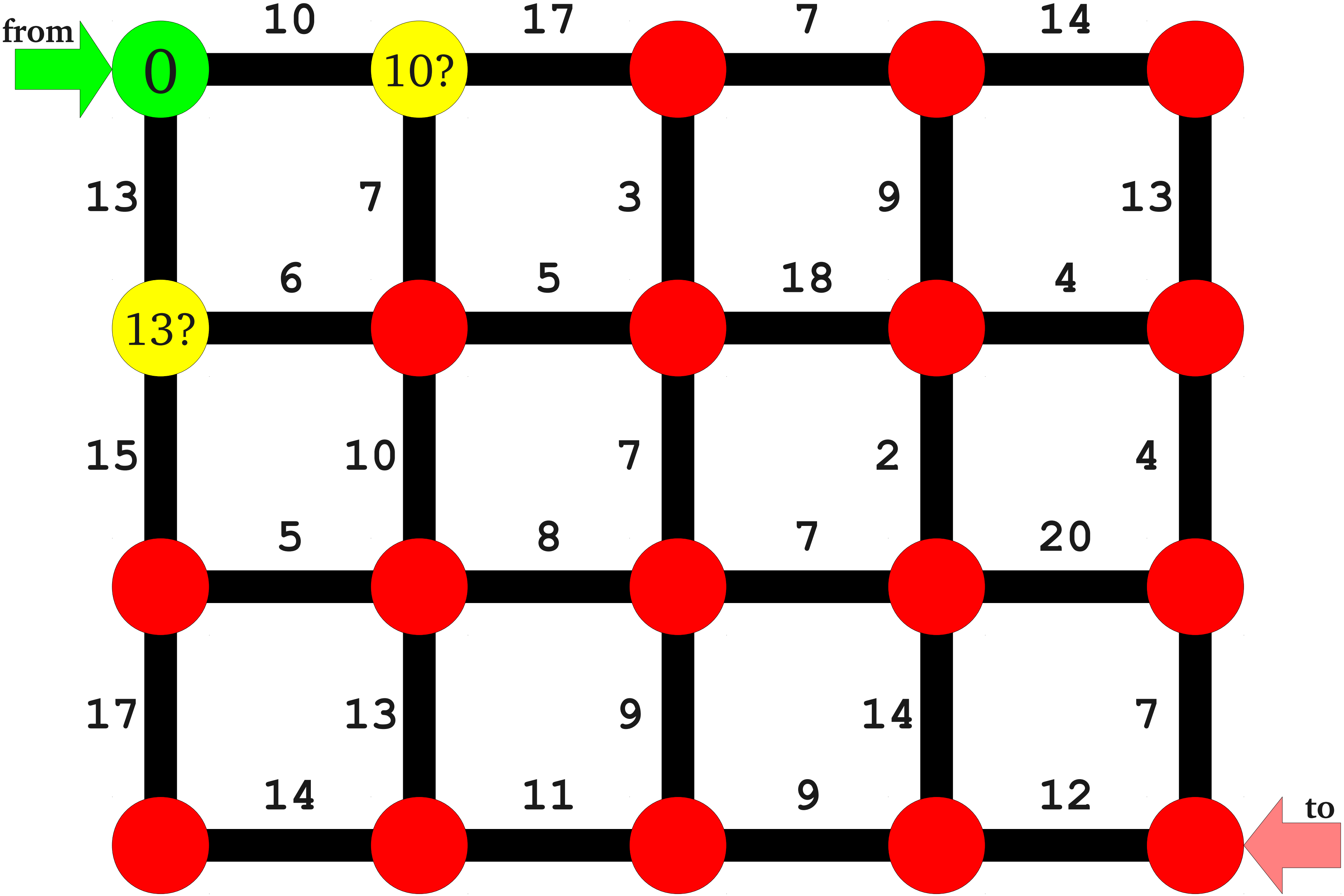
In an $n \times n$ grid, there are at least $4^n / n$ possible paths from one corner to another.

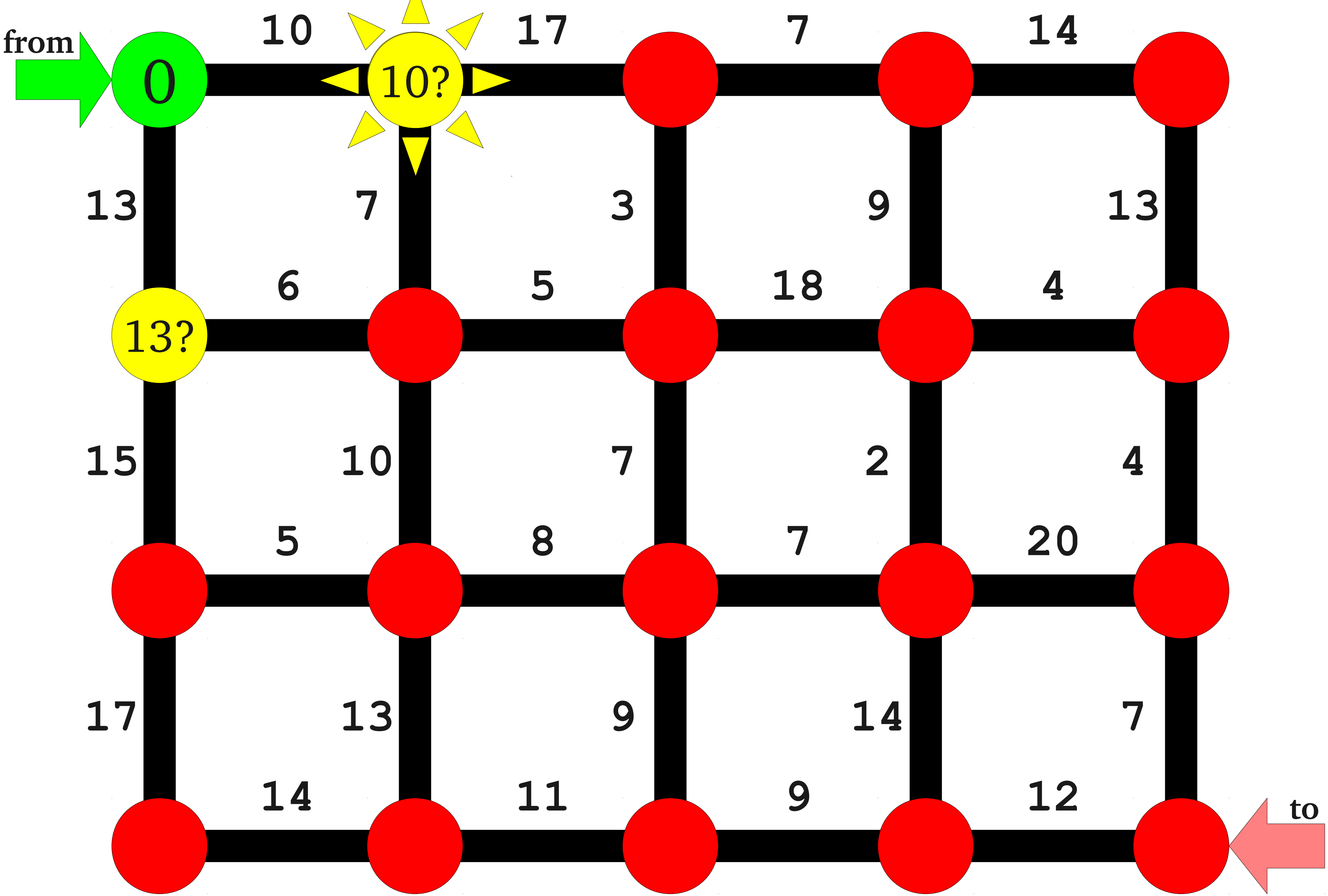
If $n = 50$, it would take the lifetime of the universe to list off all possible paths.

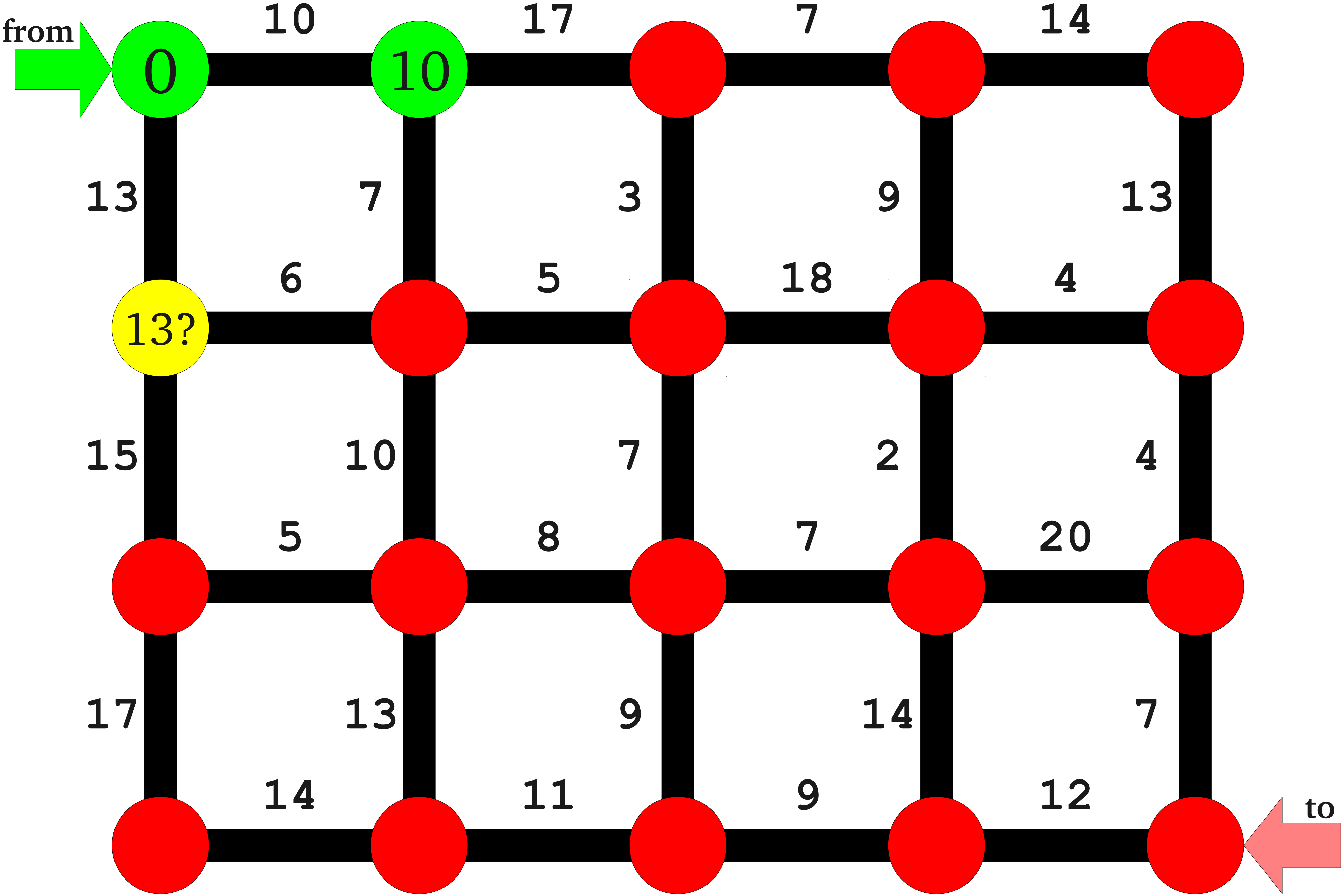

from
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to

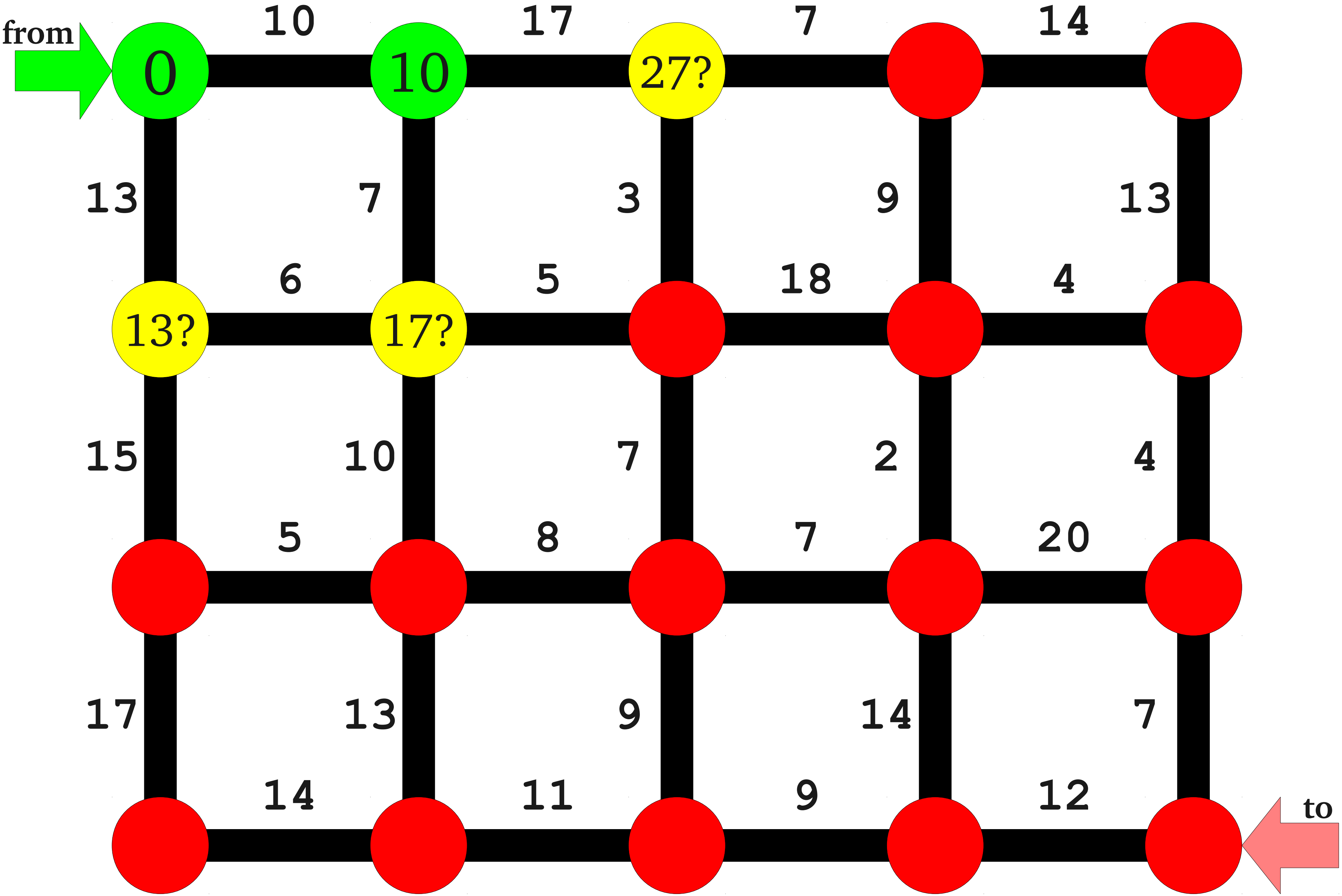

from
0
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to

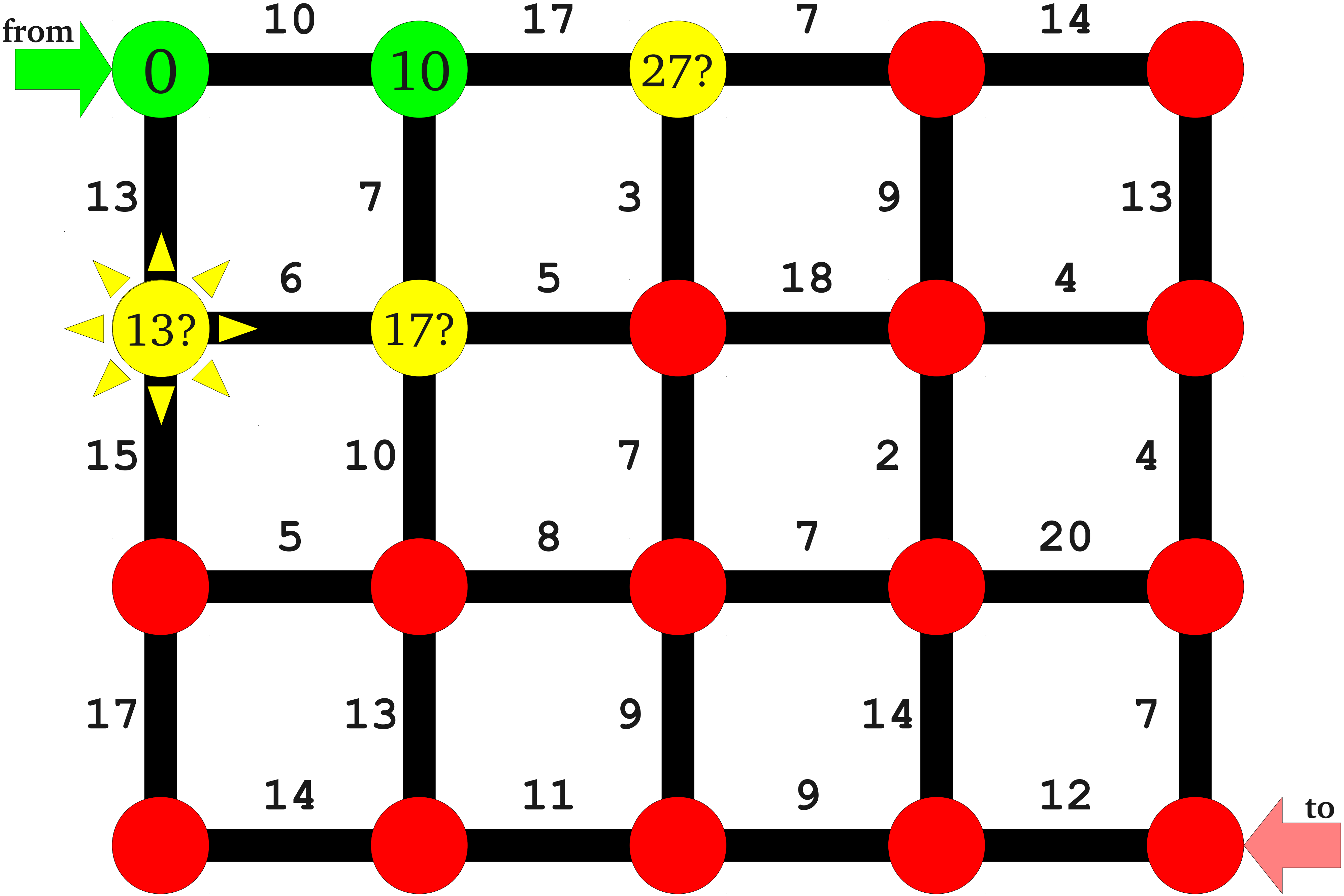

from
10
17
7
14
0
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to

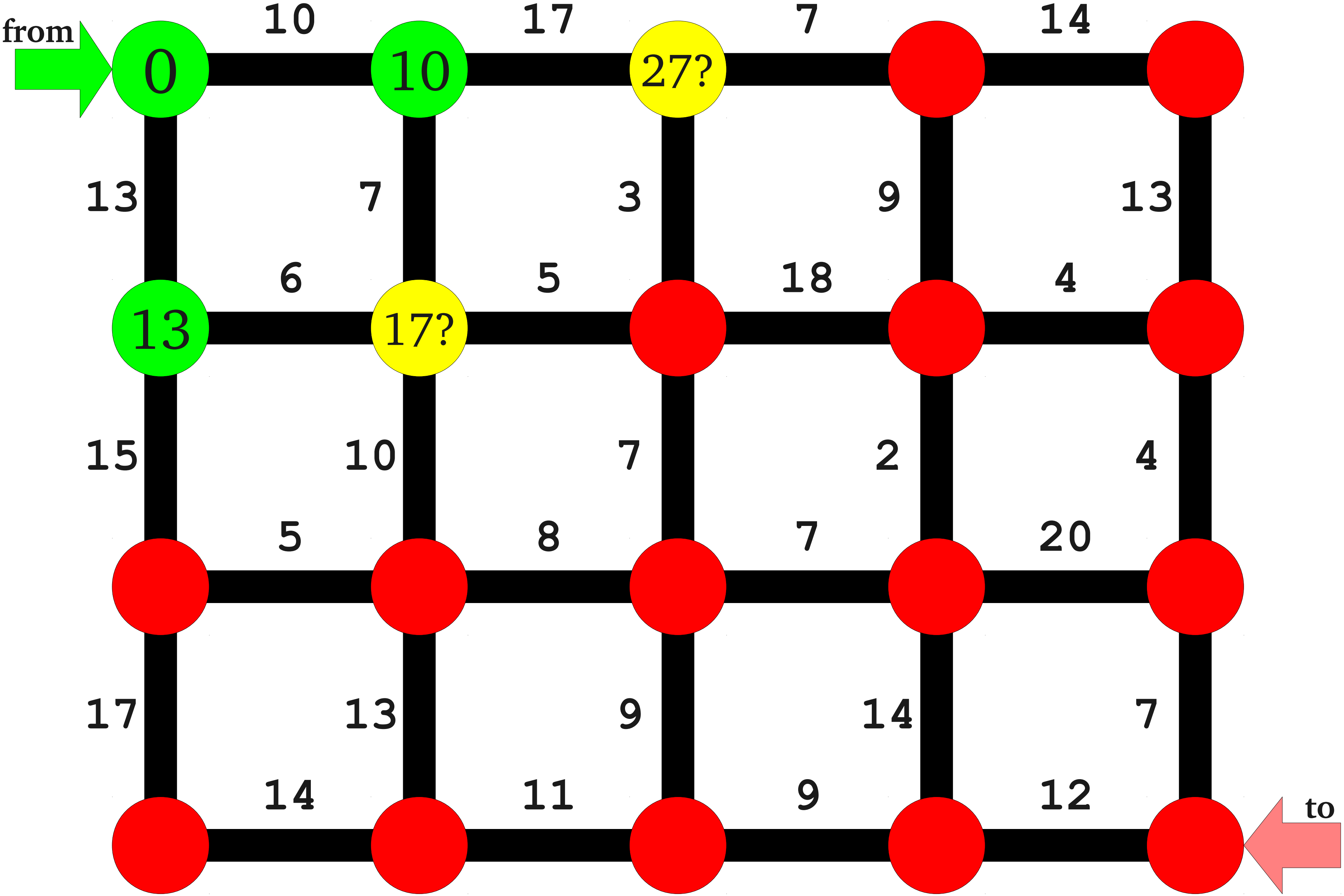

from
0
10
10?
17
7
14
13
7
3
9
13
13?
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to

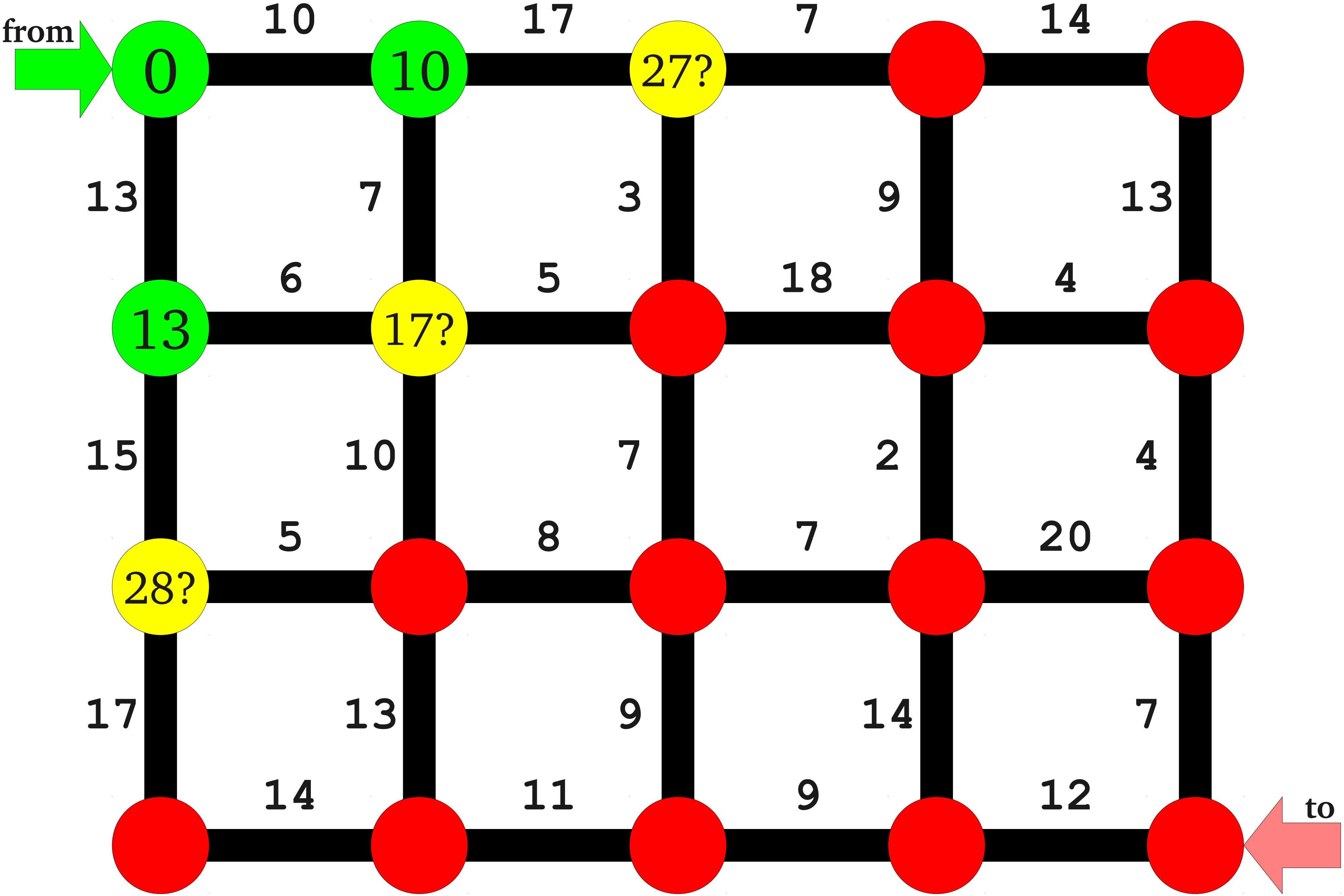

from
0
10
10?
17
7
14
13
7
3
9
13
13?
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to

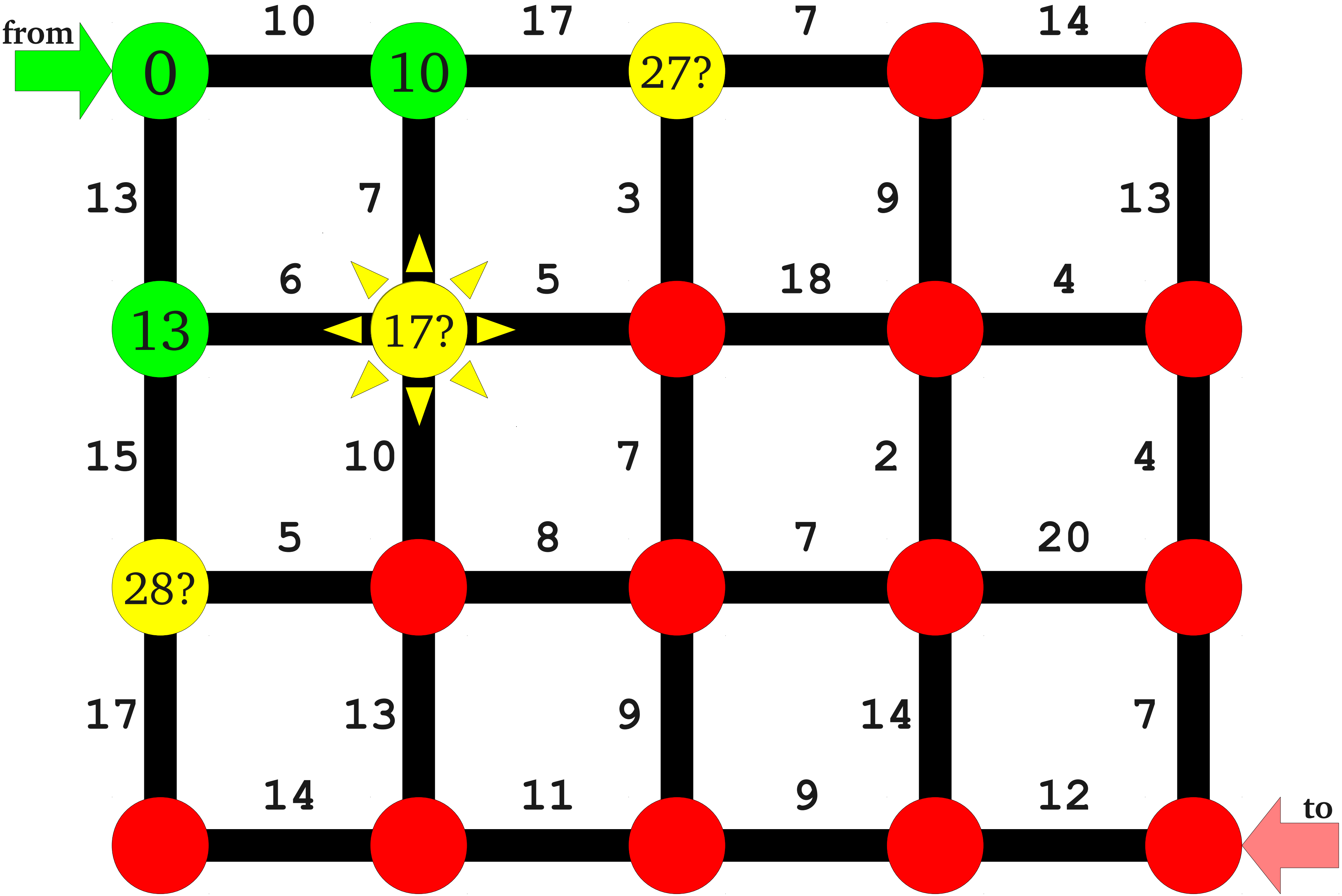

from
0
10
13?
to
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12

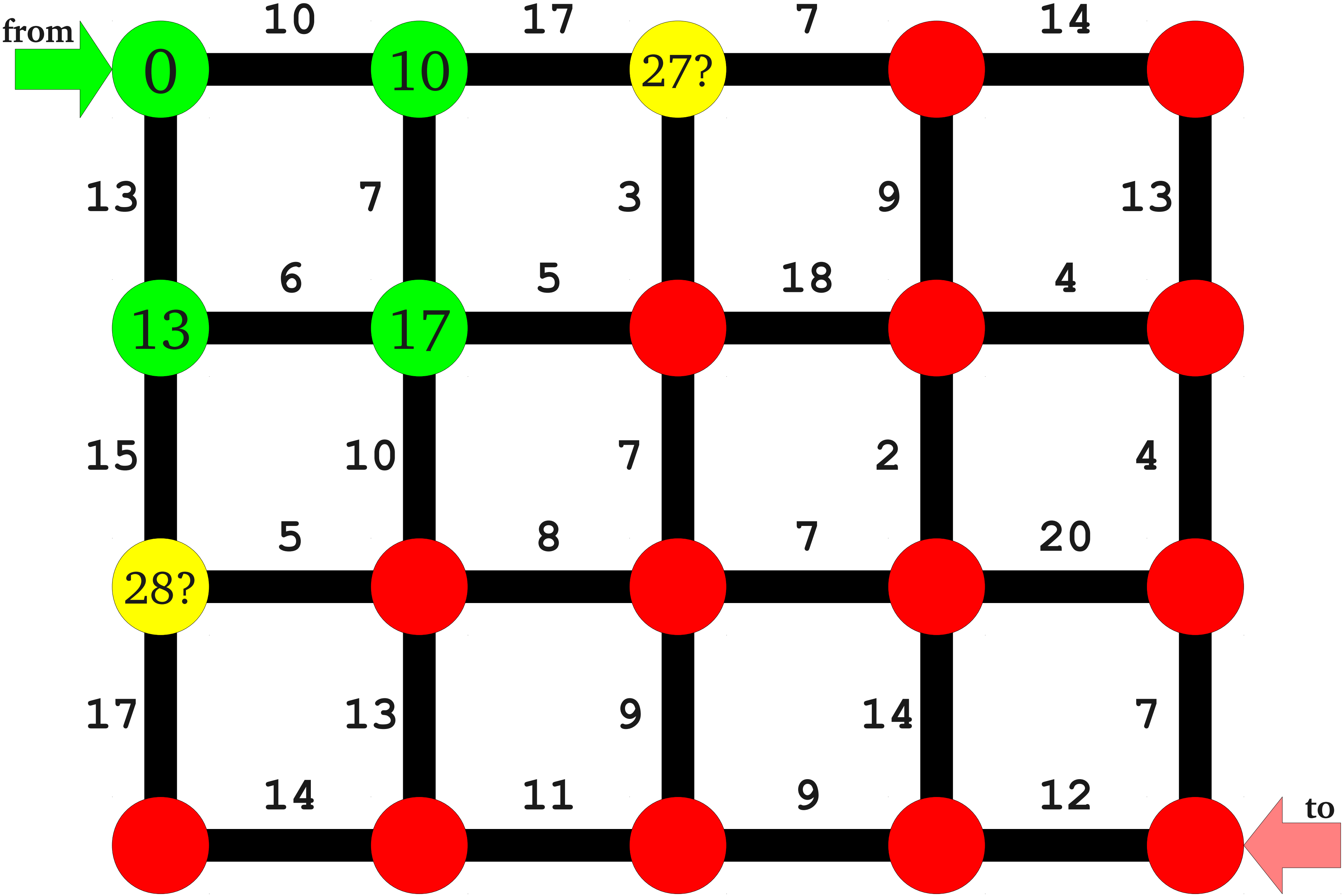

from
10
17
7
14
0
10
27?
13
7
3
9
13
6
5
18
4
13?
17?
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to

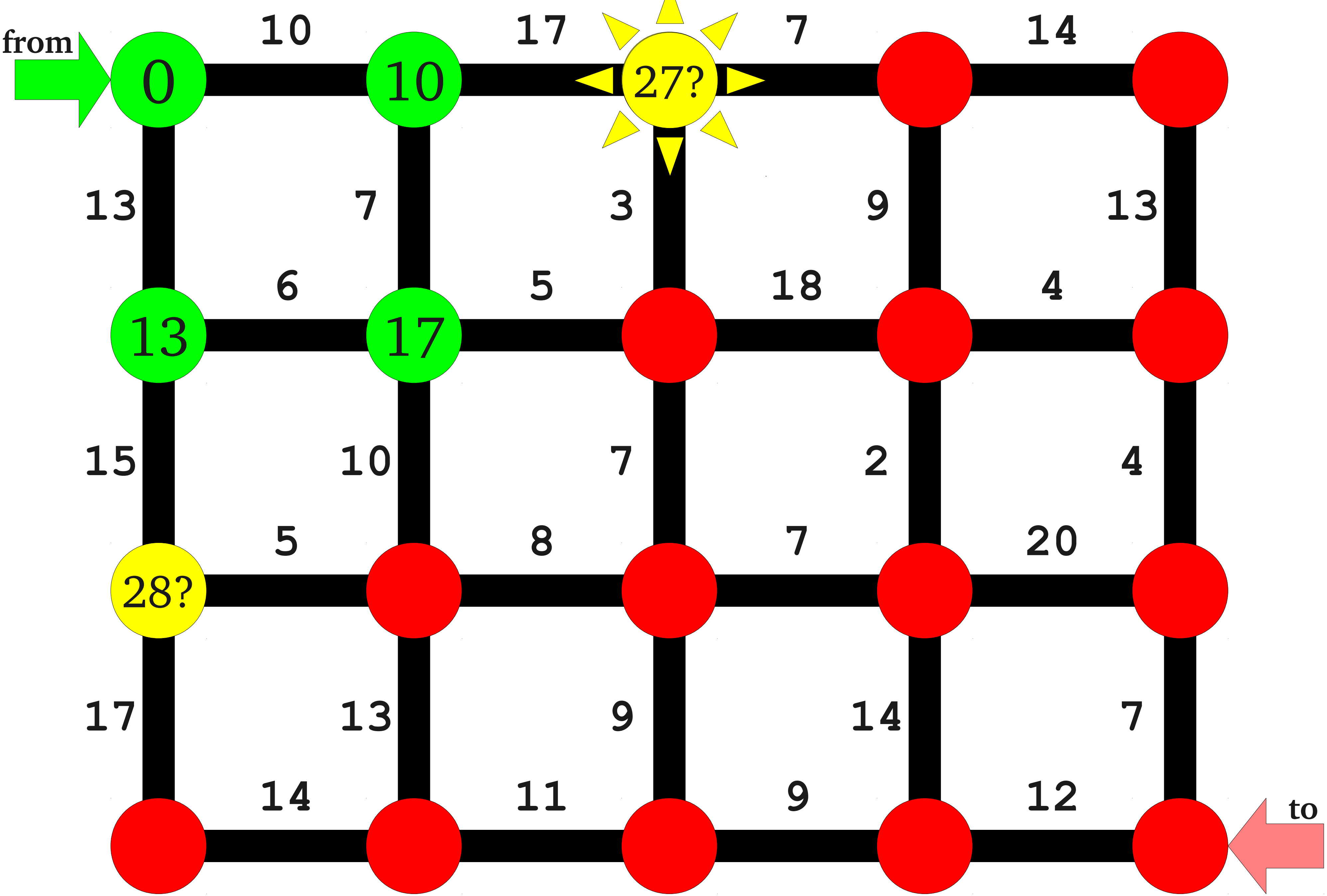

from
10
17
7
14
0
10
27?
13
7
3
9
13
6
5
18
4
13?
17?
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to

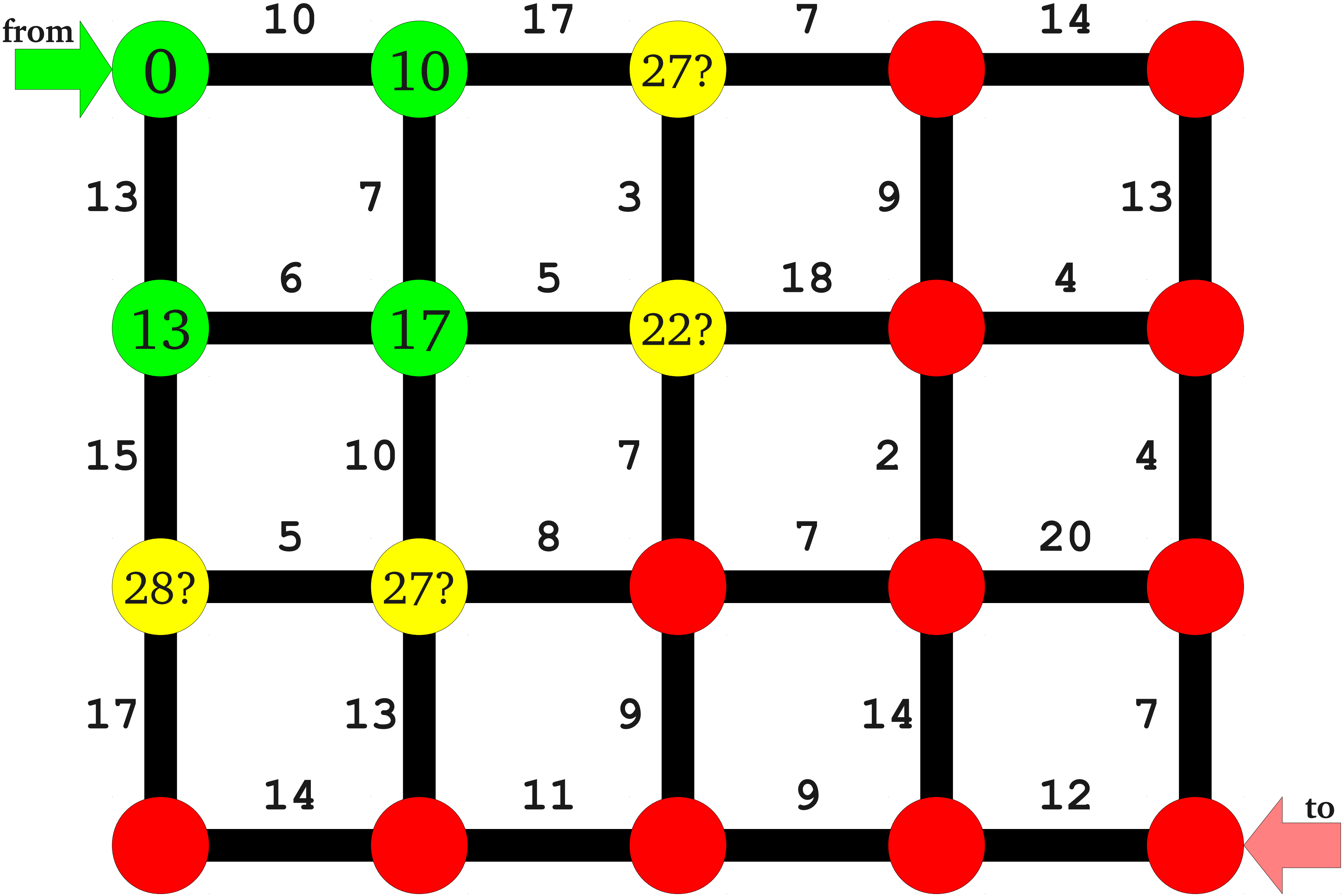

from
10
17
7
14
0
10
27?
13
7
3
9
13
6
5
18
4
13
17?
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
to

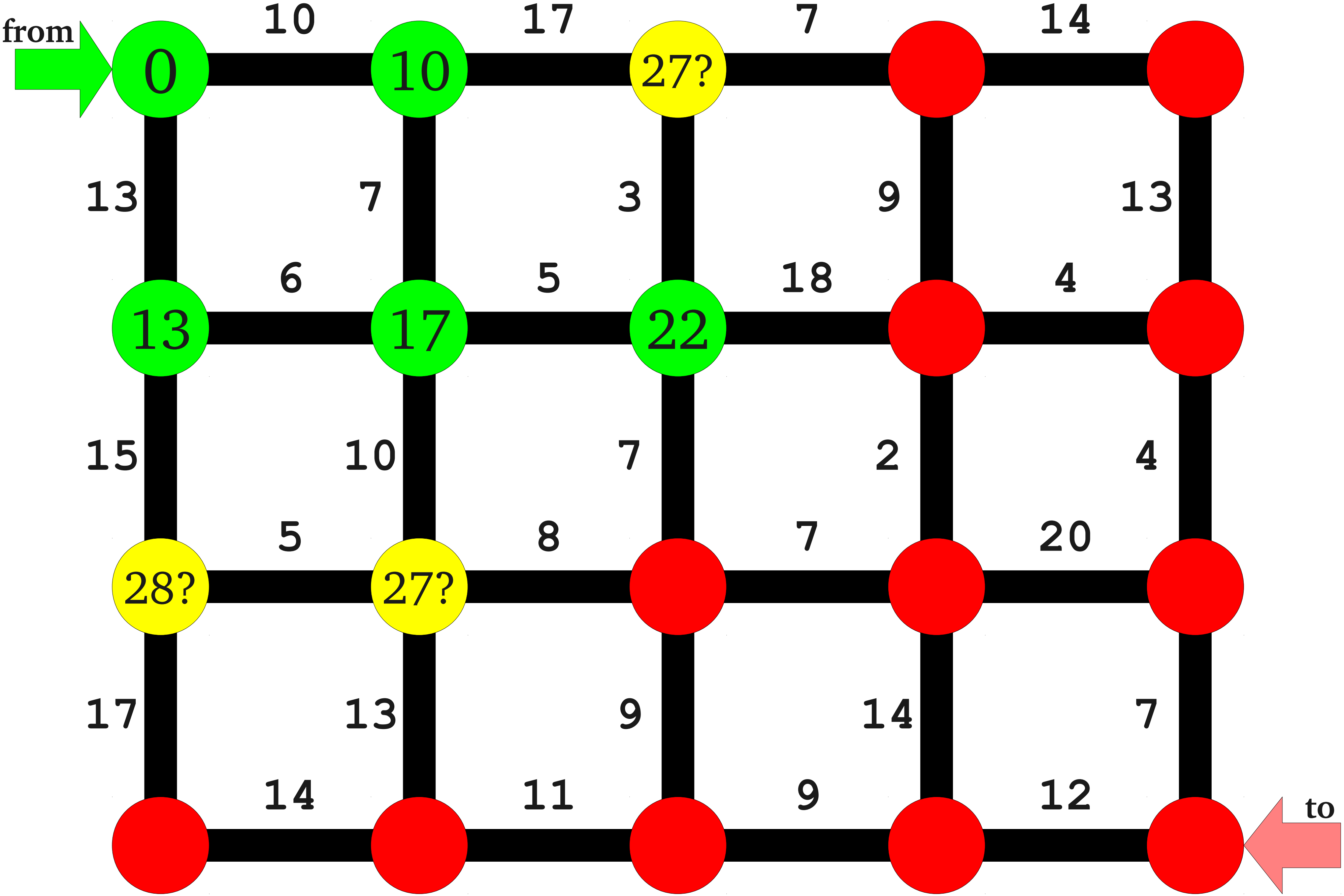

from
10
17
7
14
0
10
27?
13
7
3
9
13
6
5
18
4
13
17?
15
10
7
2
4
5
8
7
20
28?
17
13
9
14
7
14
11
9
12
to

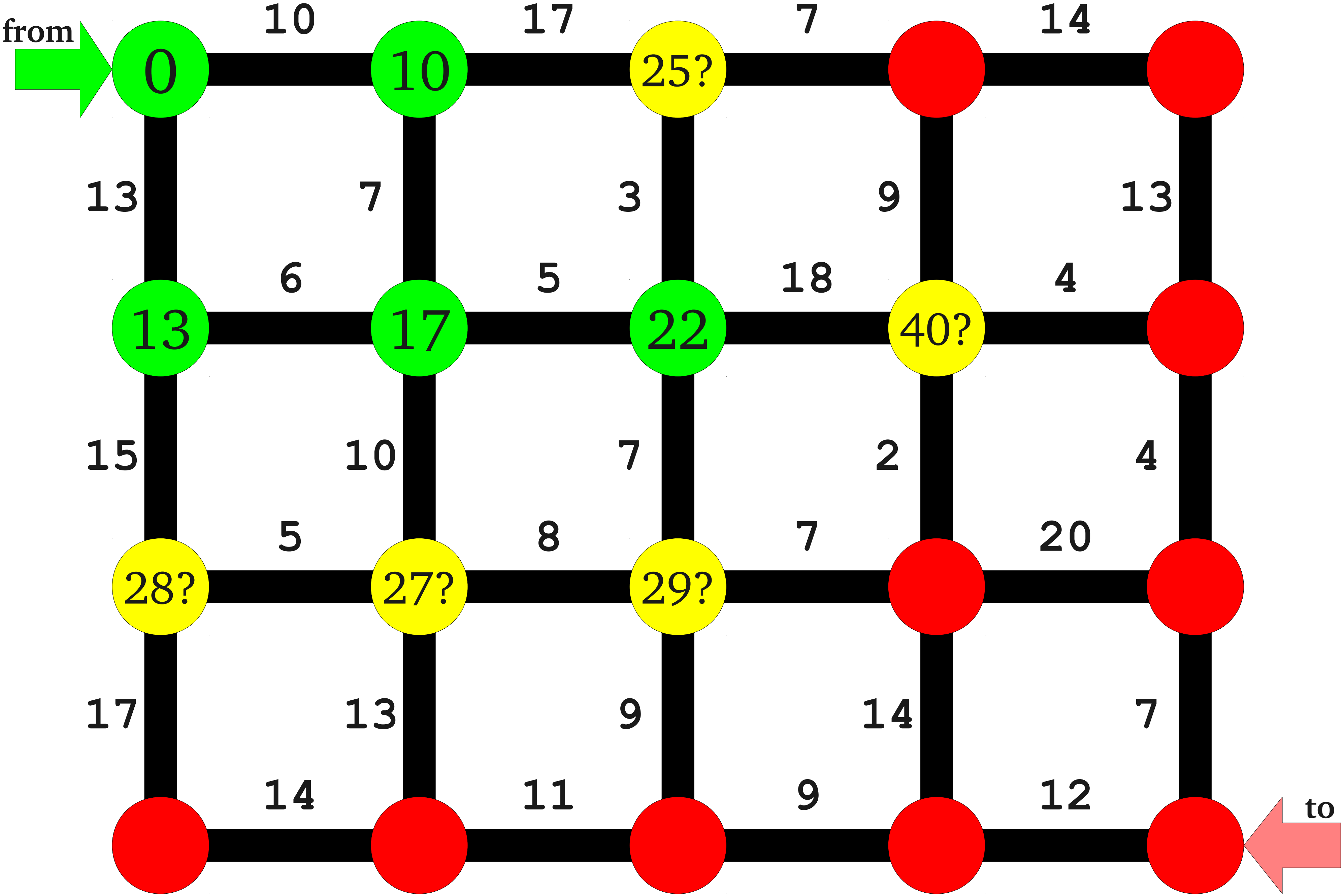

from
10
17
7
14
0
10
27?
13
7
3
9
13
6
5
18
4
13
17?
15
10
7
2
4
5
8
7
20
28?
17
13
9
14
7
14
11
9
12
to

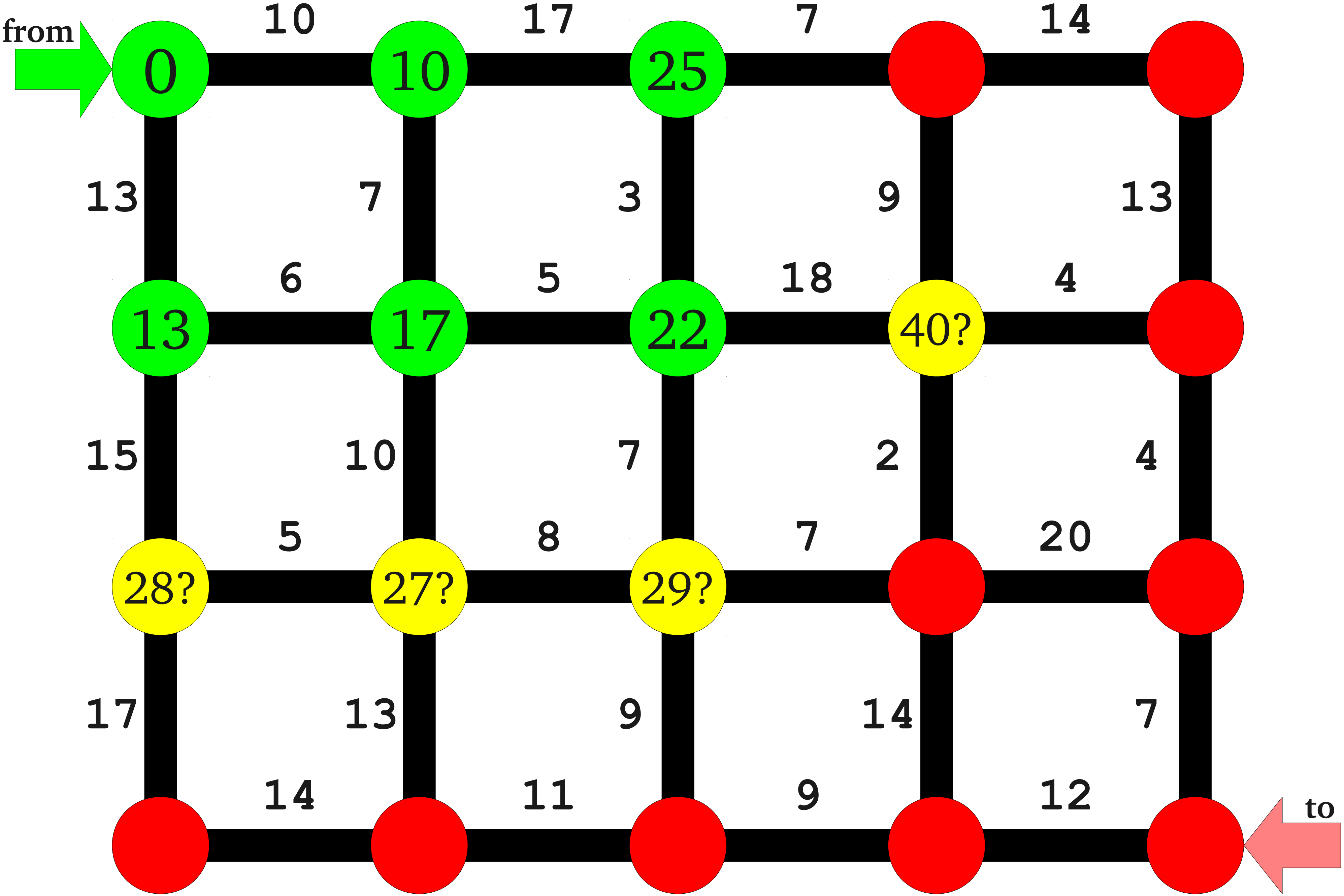

from
10
17
7
14
0
10
27?
13
7
3
9
13
6
5
18
4
13
17
15
10
7
2
4
5
8
7
20
28?
17
13
9
14
7
14
11
9
12
to

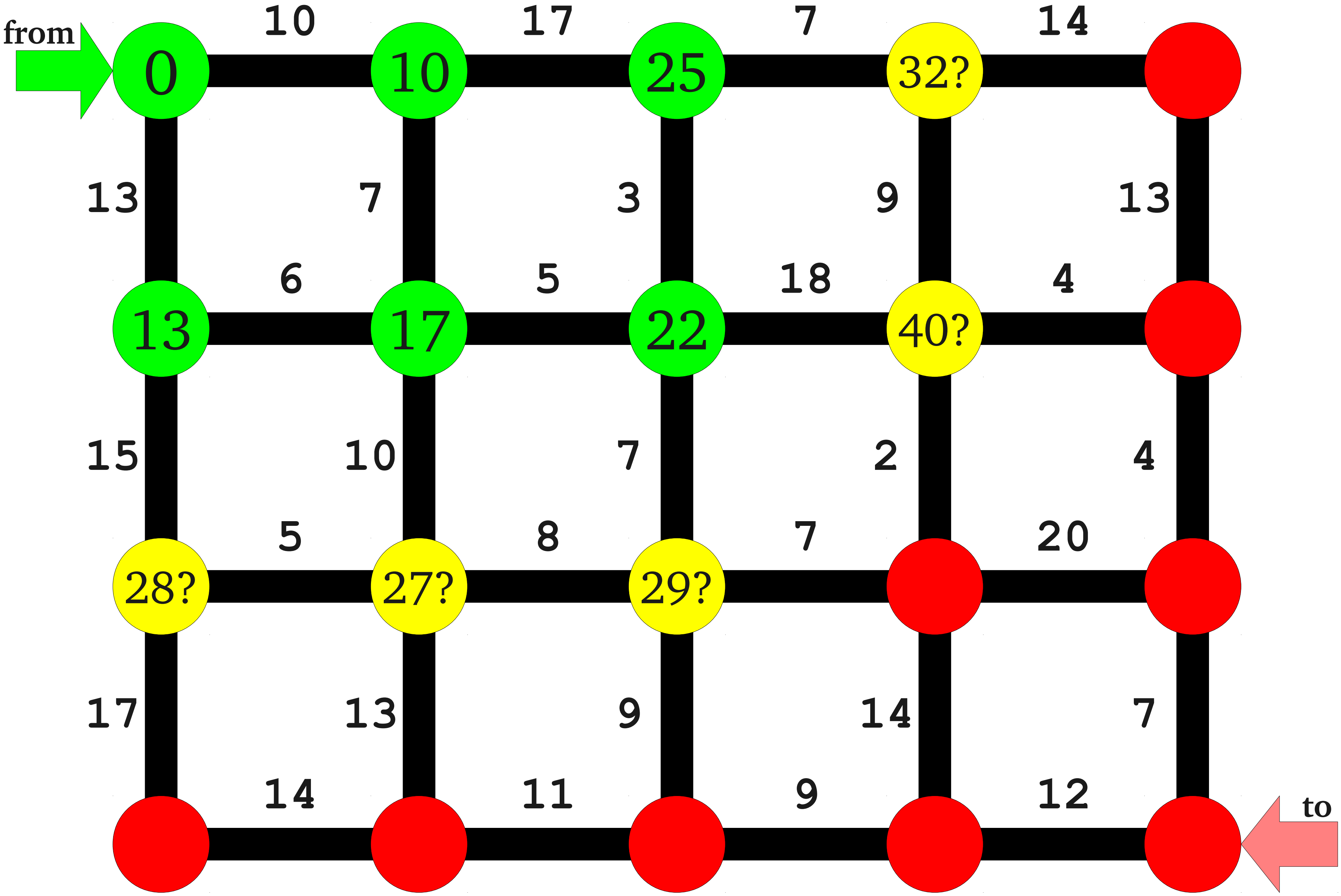

from
10
17
7
14
0
10
27?
13
7
3
9
13
6
5
18
4
13
17
15
10
7
2
4
5
8
7
20
28?
17
13
9
14
7
14
11
9
12
to

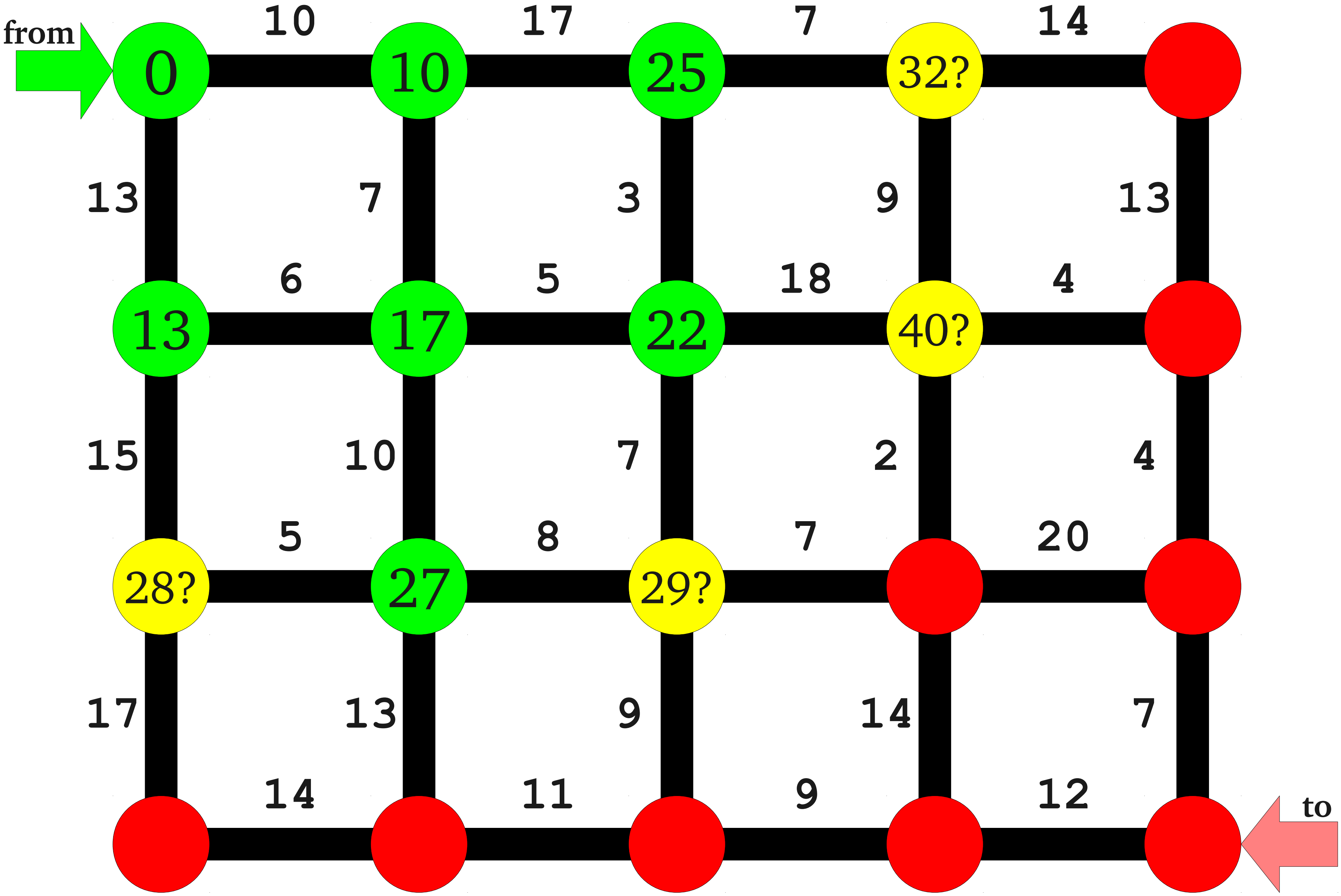

from
10
17
7
14
0
10
27?
13
7
3
9
13
6
5
18
4
13
17
22?
15
10
7
2
4
5
8
7
20
28?
27?
17
13
9
14
7
14
11
9
12
to

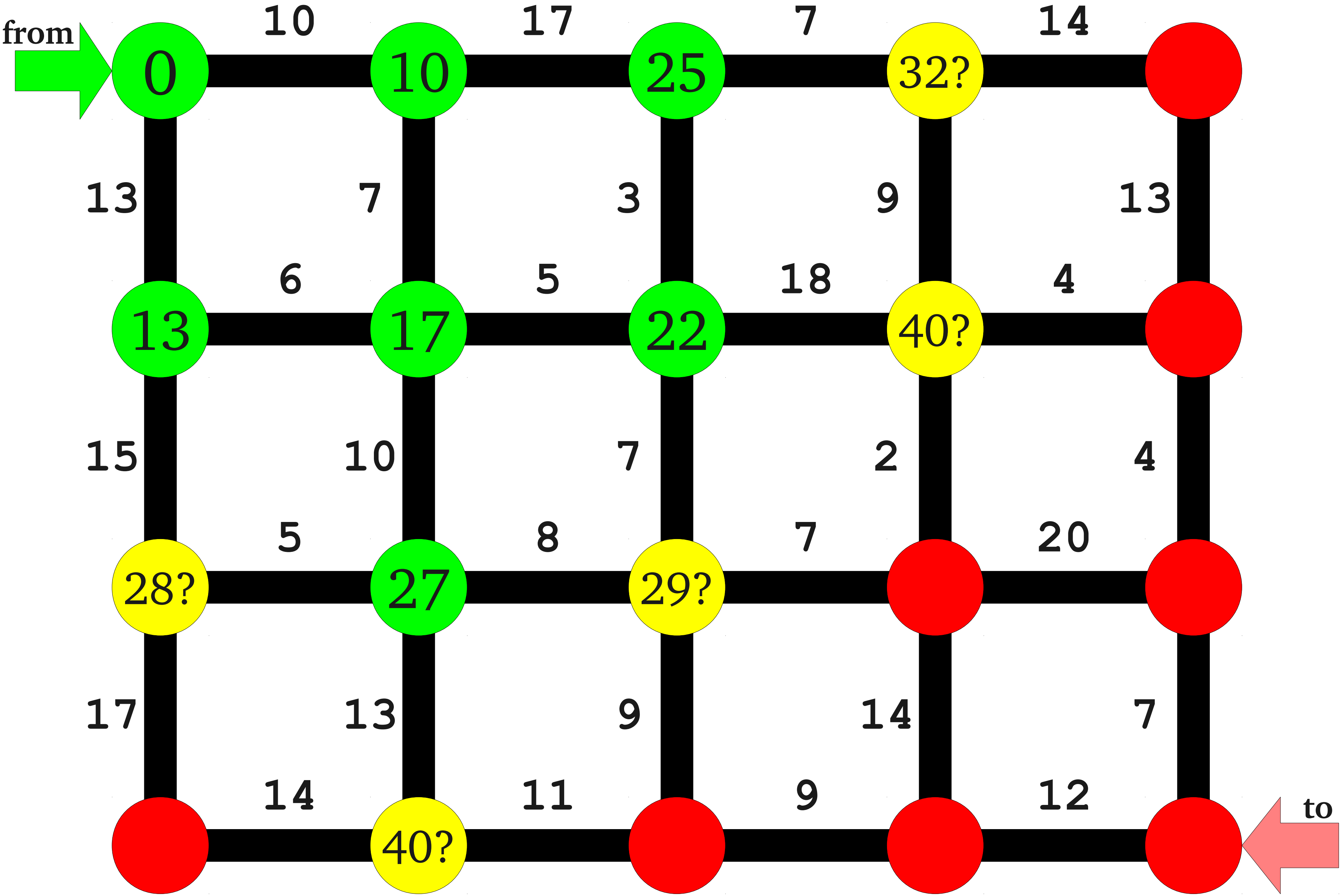

from
to
0
10
27?
13
17
22
28?
27?
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12

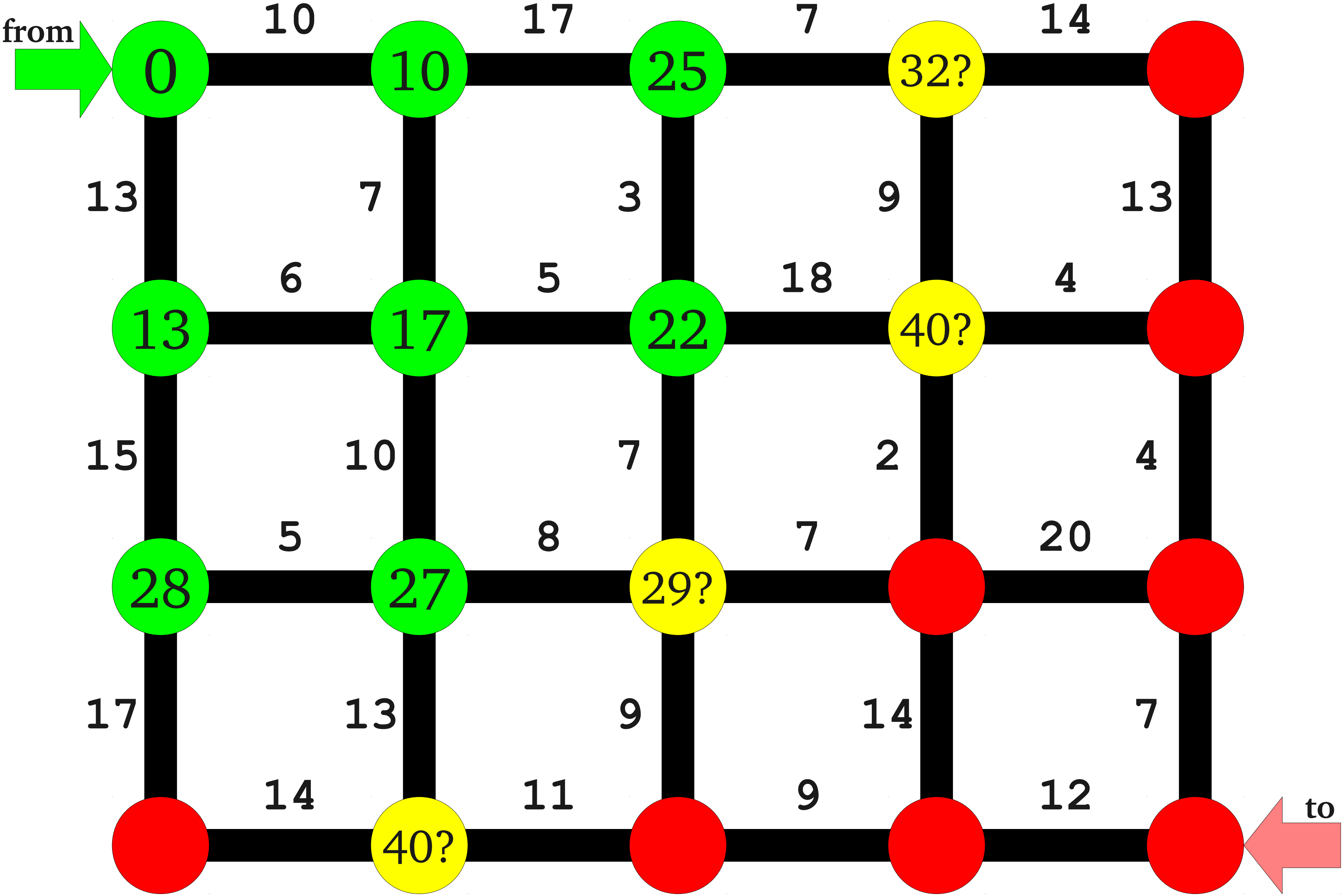

from
10
17
7
14
0
10
25?
13
7
3
9
13
6
5
18
4
13
17
22
40?
15
10
7
2
4
5
8
7
20
28?
27?
29?
17
13
9
14
7
14
11
9
12
to

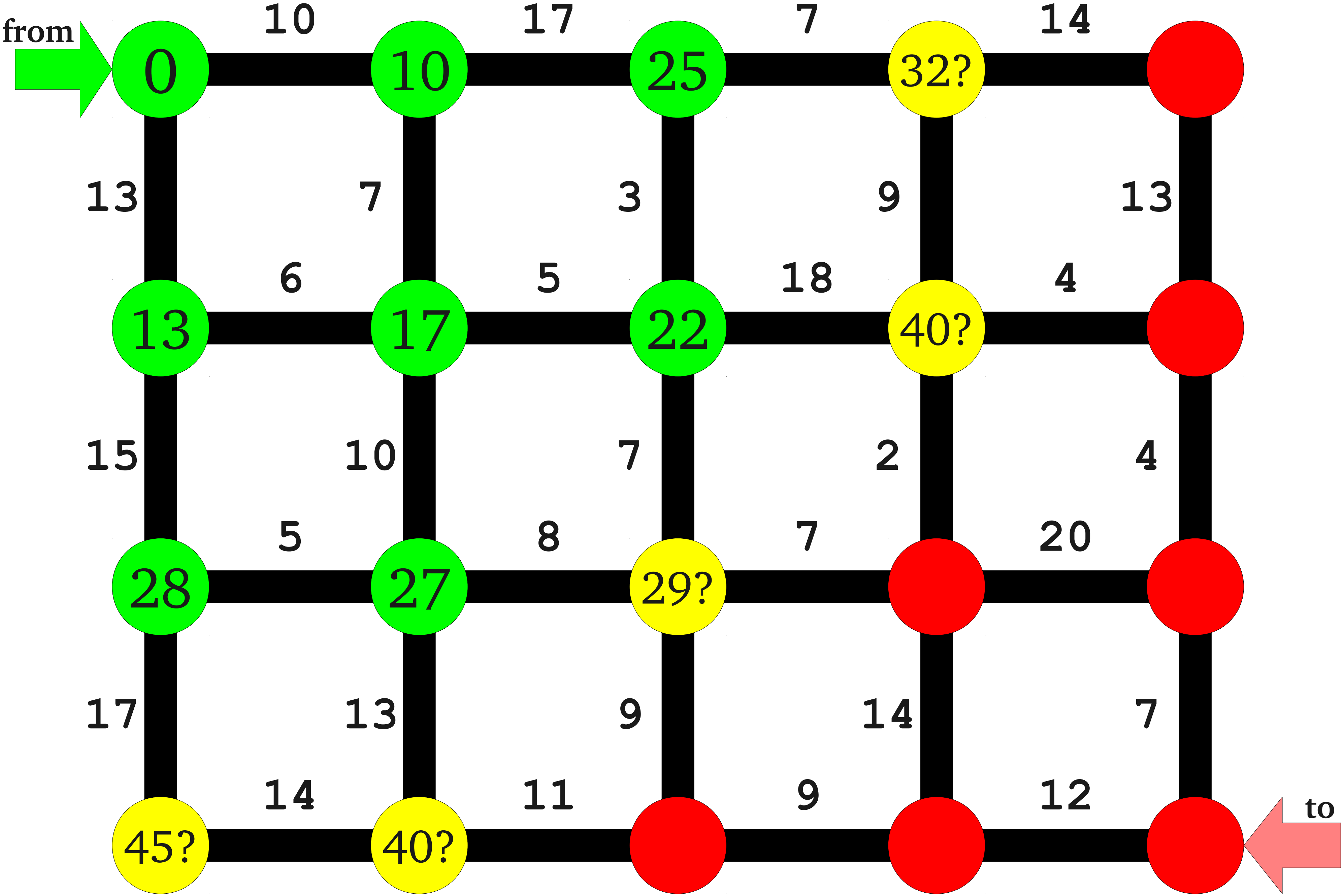

from
10
17
7
14
0
10
25
13
7
3
9
13
6
5
18
4
13
17
22
40?
15
10
7
2
4
5
8
7
20
28?
27?
29?
17
13
9
14
7
14
11
9
12
to

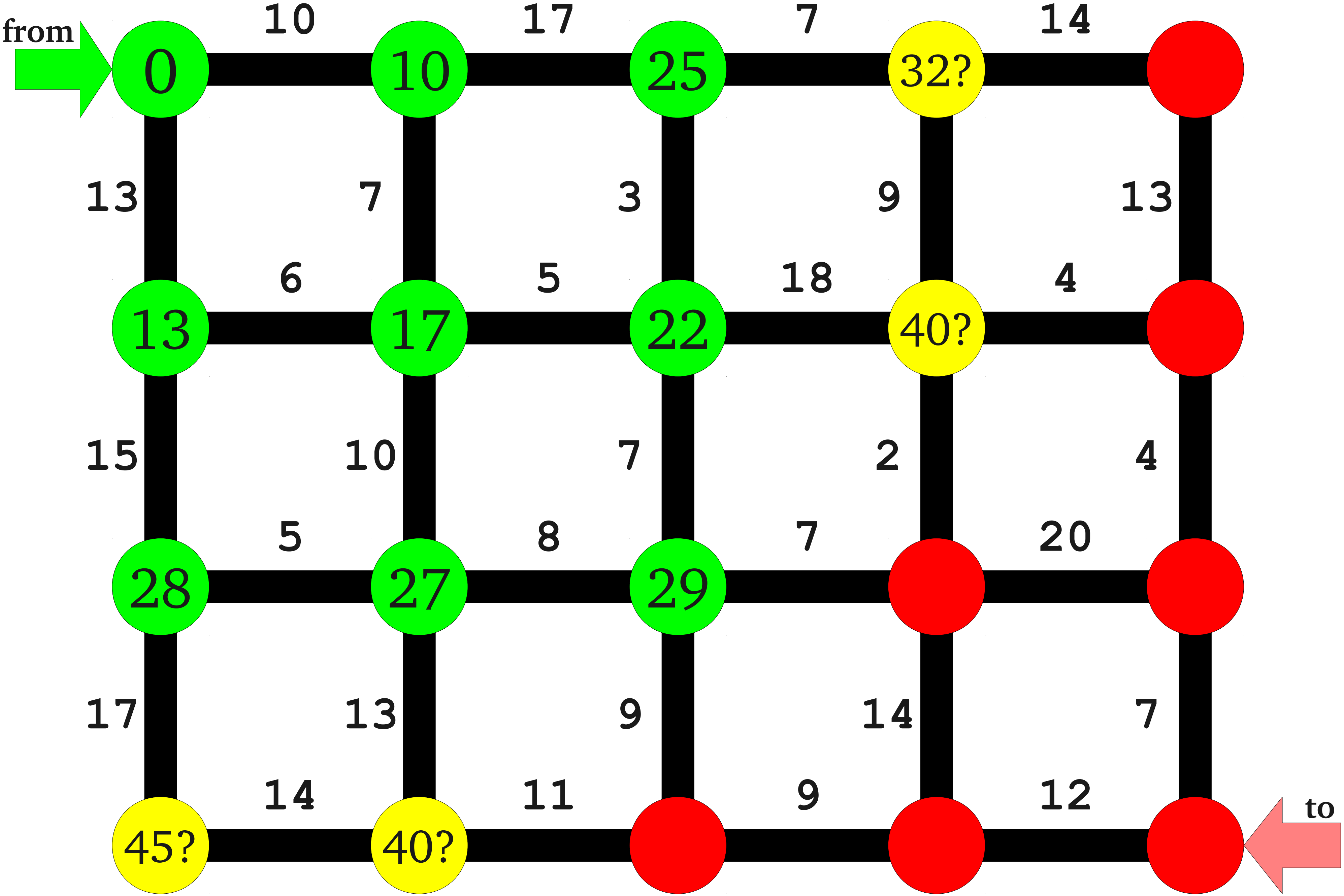

from
10
17
7
14
0
10
25
32?
13
7
3
9
13
6
5
18
4
13
17
22
40?
15
10
7
2
4
5
8
7
20
28?
27?
29?
17
13
9
14
7
14
11
9
12
to

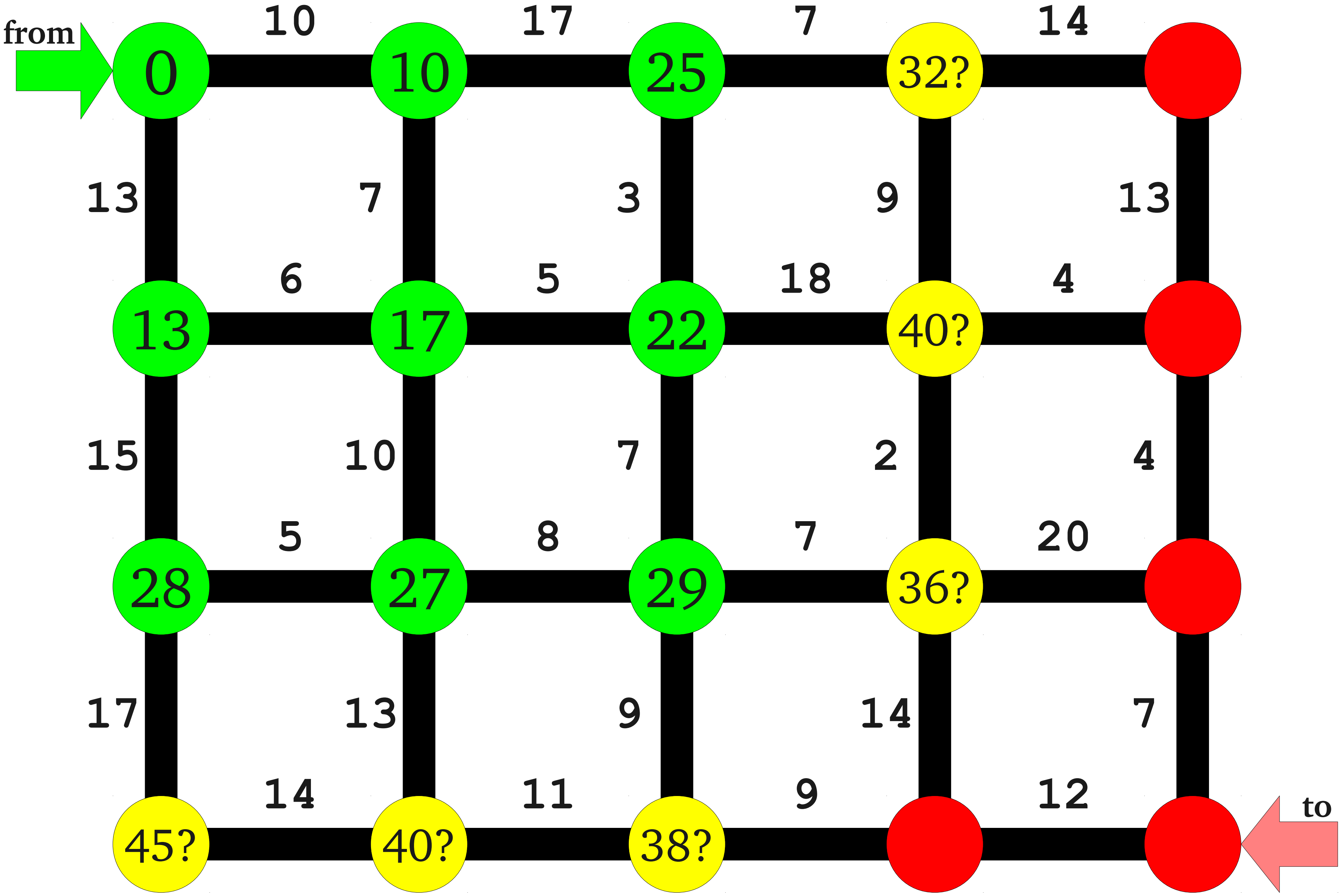

from
10
17
7
14
0
10
25
32?
13
7
3
9
13
6
5
18
4
13
17
22
40?
15
10
7
2
4
5
8
7
20
28?
27
29?
17
13
9
14
7
14
11
9
12
to

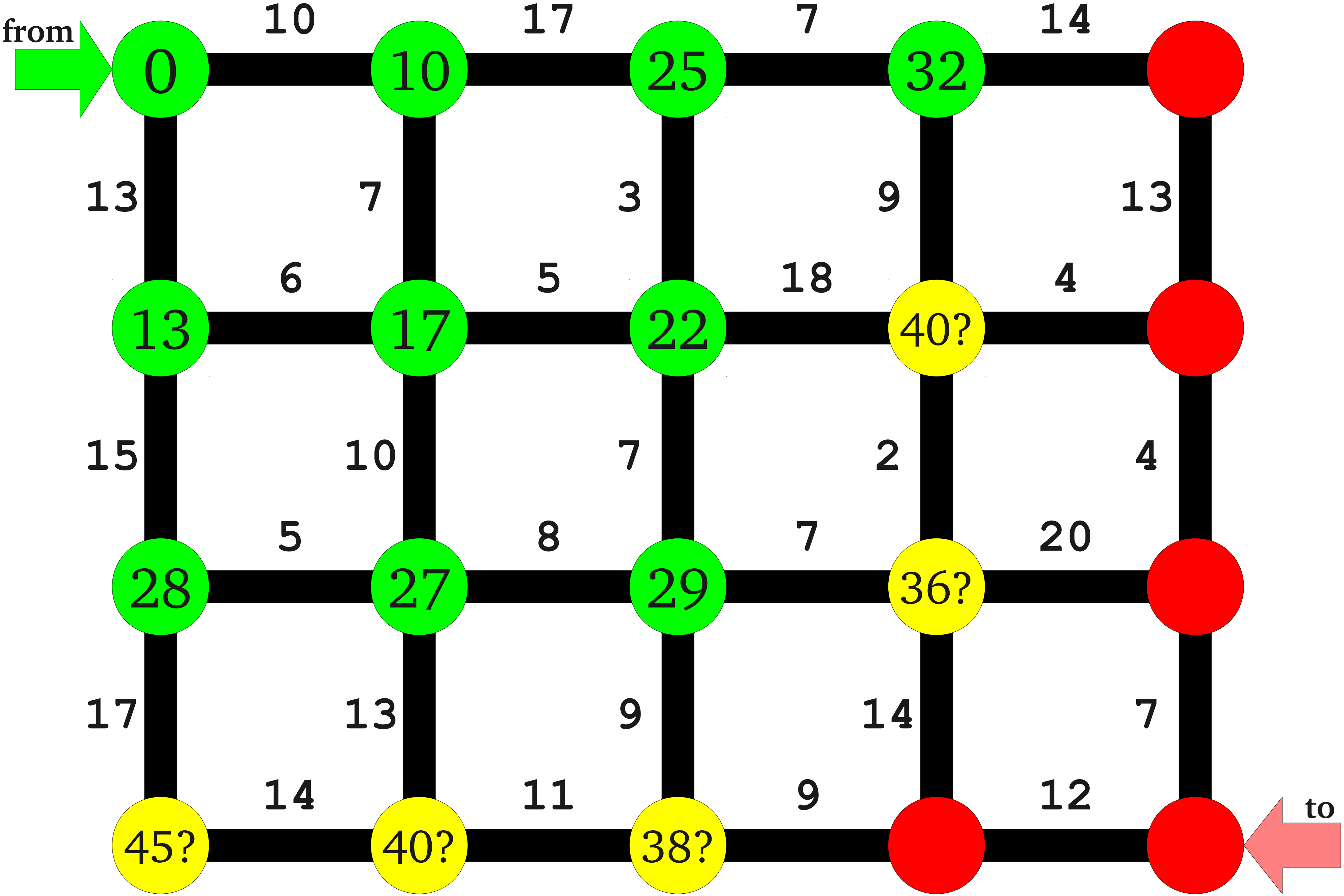

from
10
17
7
14
0
10
25
32?
13
7
3
9
13
6
5
18
4
13
17
22
40?
15
10
7
2
4
5
8
7
20
28?
27
29?
17
13
9
14
7
14
11
9
12
40?
to

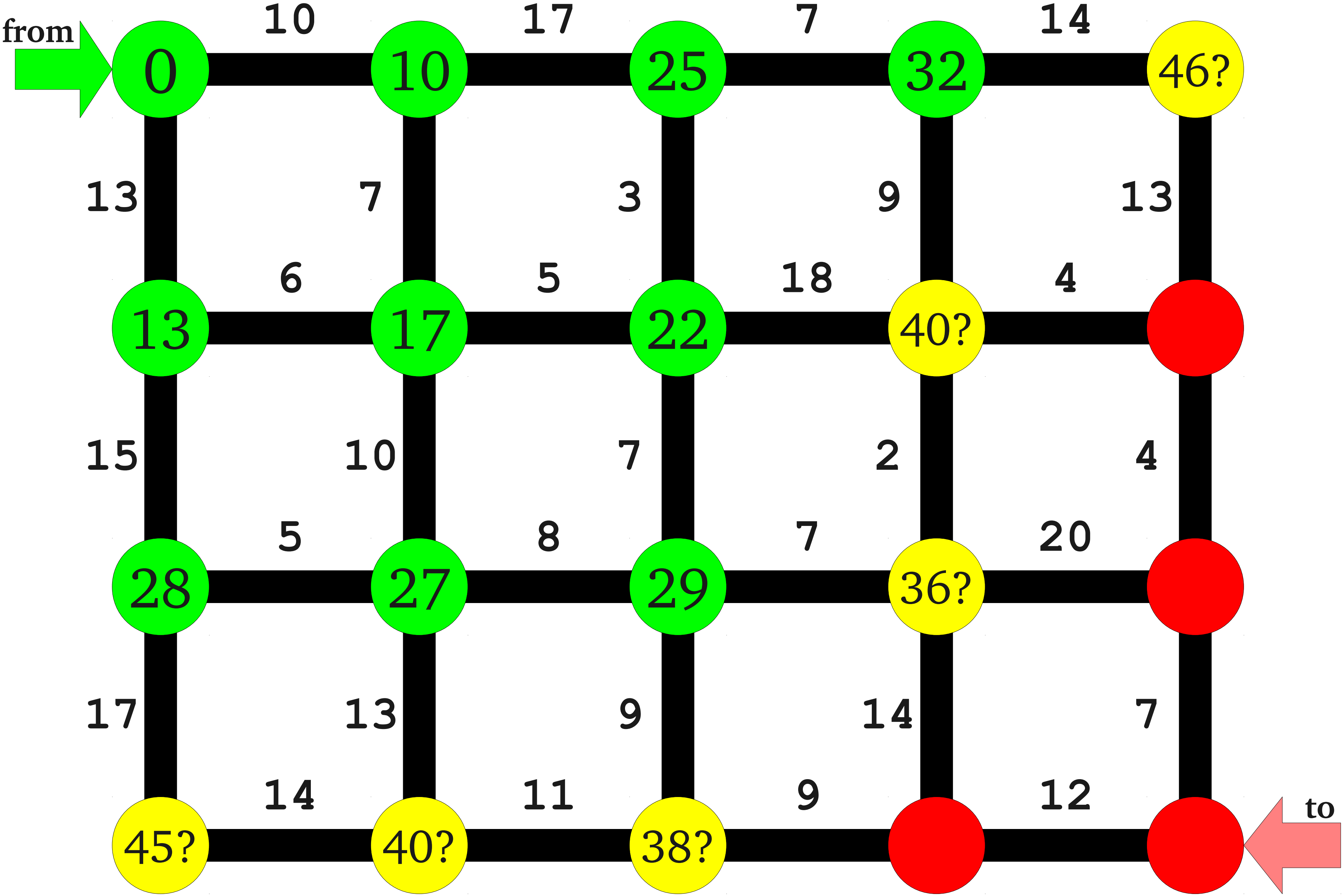

from
10
17
7
14
0
10
25
32?
13
7
3
9
13
6
5
18
4
13
17
22
40?
15
10
7
2
4
5
8
7
20
28
27
29?
17
13
9
14
7
14
11
9
12
to
40?

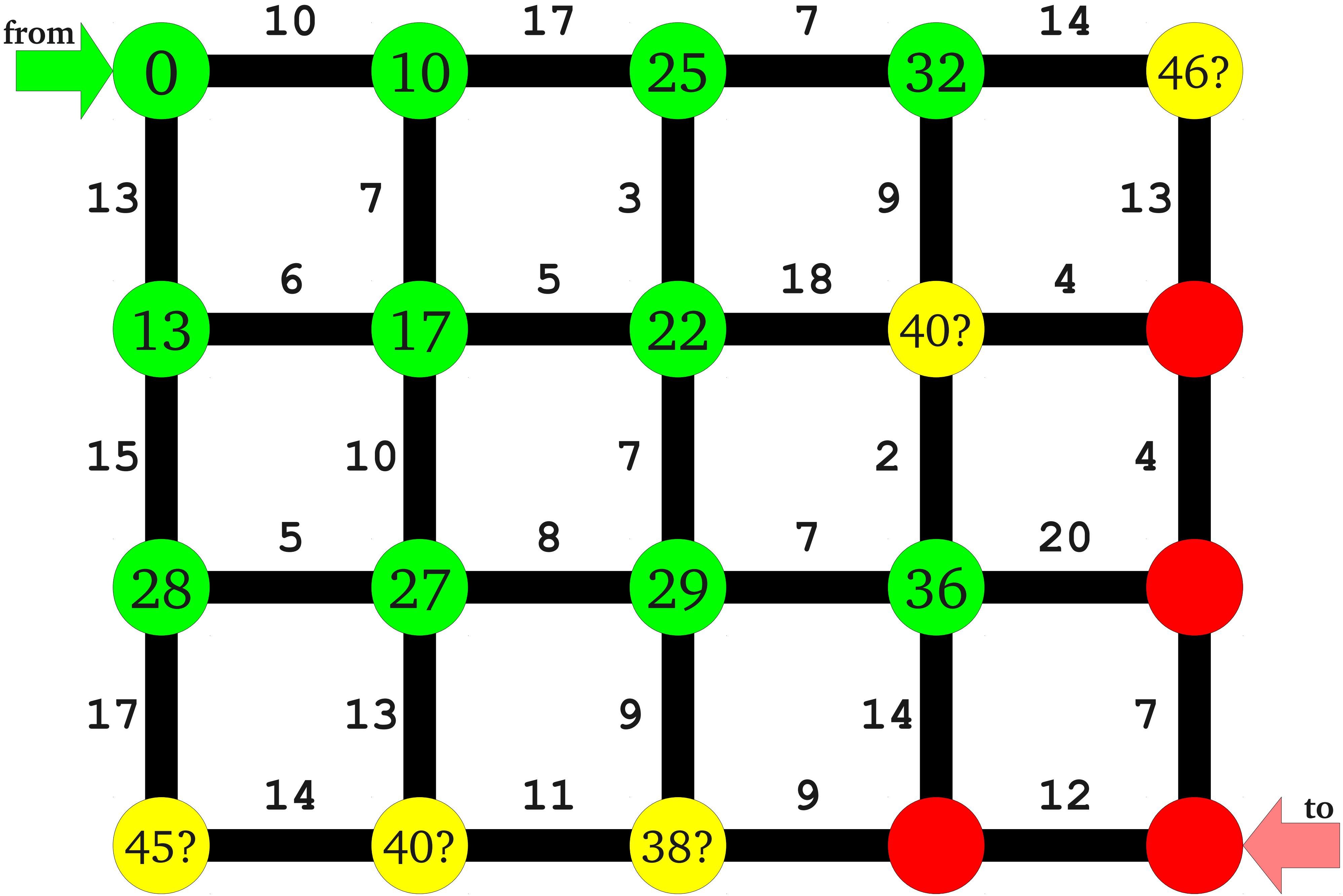

from
10
17
7
14
0
10
25
32?
13
7
3
9
13
6
5
18
4
13
17
22
40?
15
10
7
2
4
5
8
7
20
28
27
29?
17
13
9
14
7
14
11
9
12
to
45?
40?

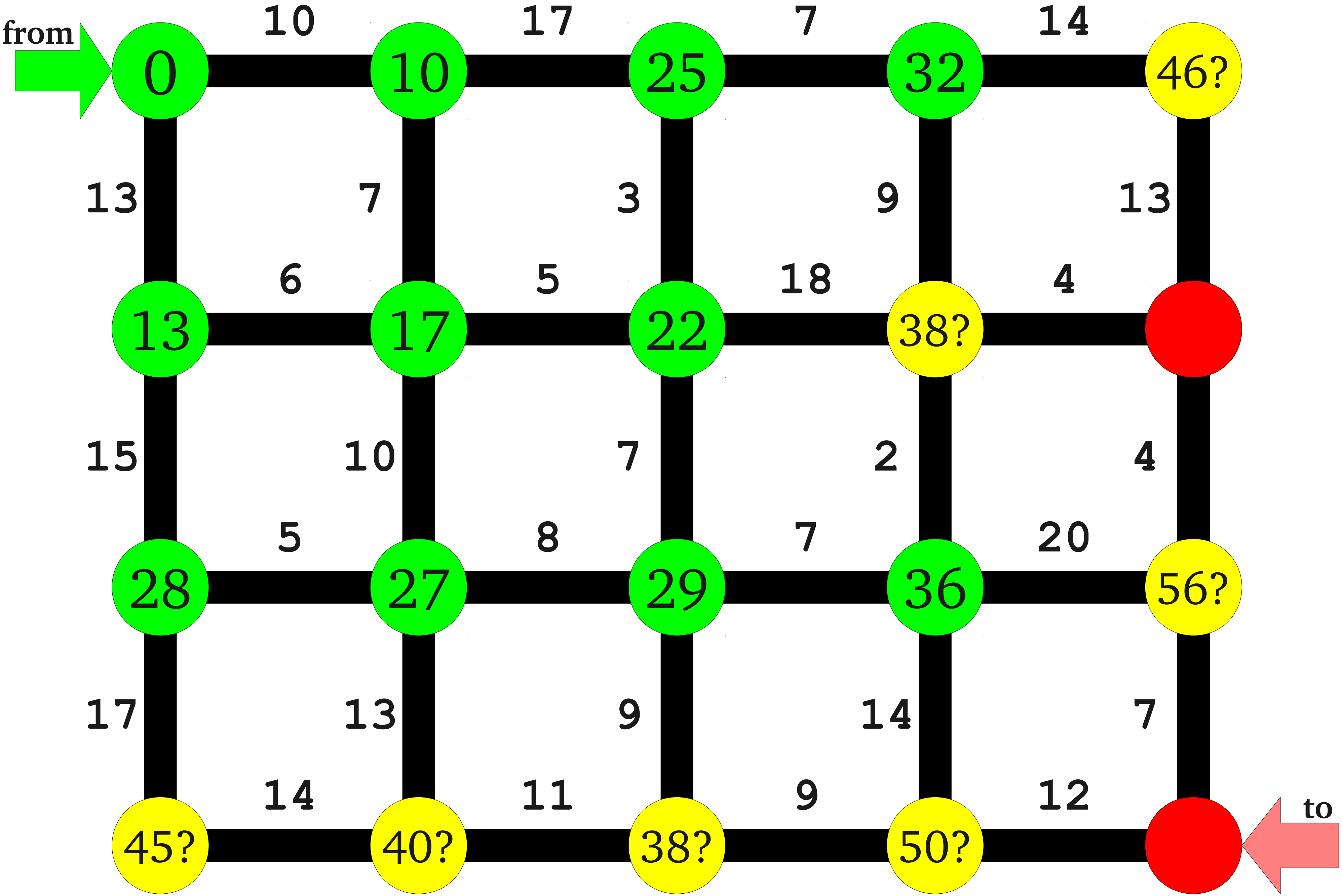

from
10
17
7
14
0
10
25
32?
13
7
3
9
13
6
5
18
4
13
17
22
40?
15
10
7
2
4
5
8
7
20
28
27
29
17
13
9
14
7
14
11
9
12
to
45?
40?

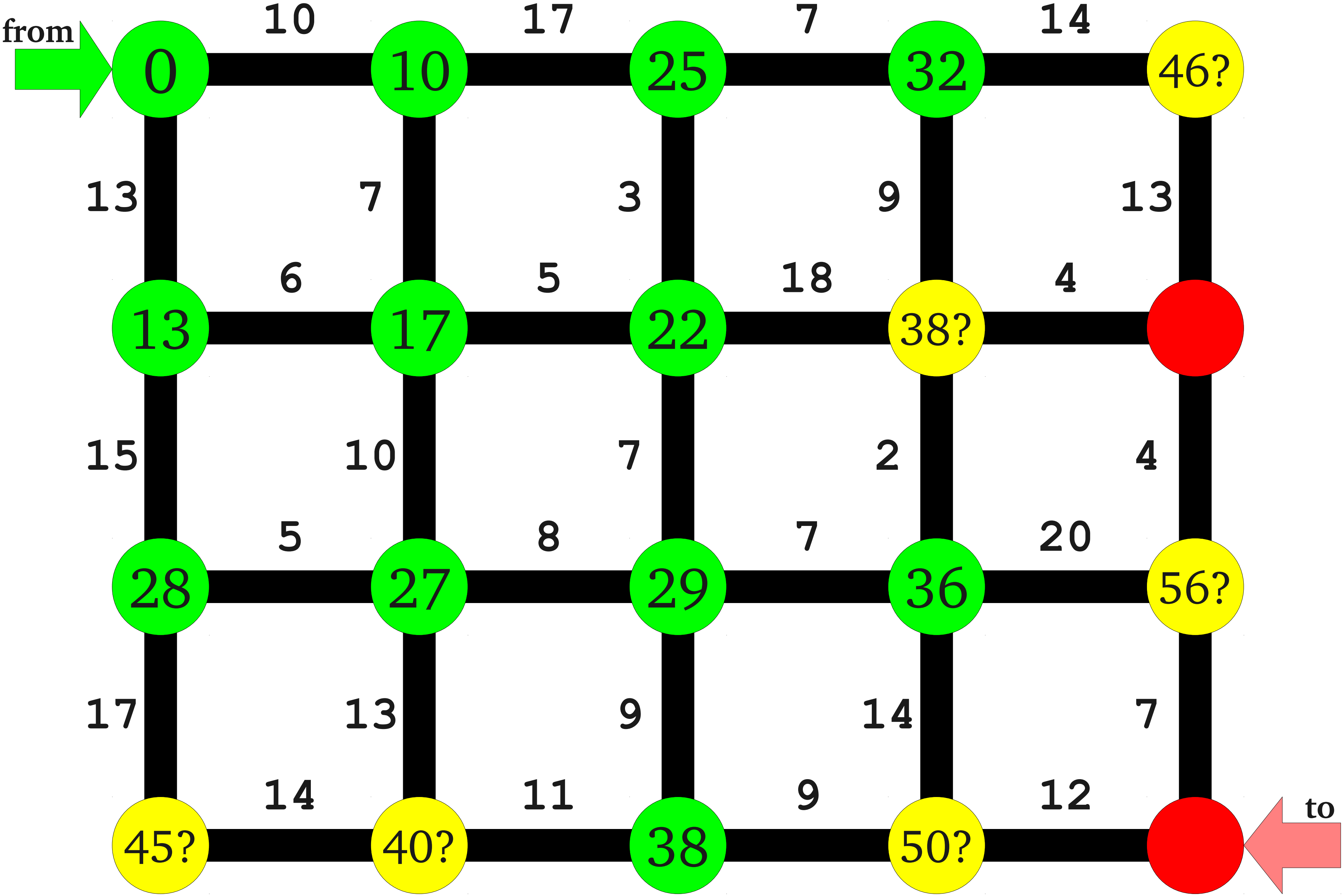

from
10
17
7
14
0
10
25
32?
13
7
3
9
13
6
5
18
4
13
17
22
40?
15
10
7
2
4
5
8
7
20
28
27
29
36?
17
13
9
14
7
14
11
9
12
to
45?
40?
38?

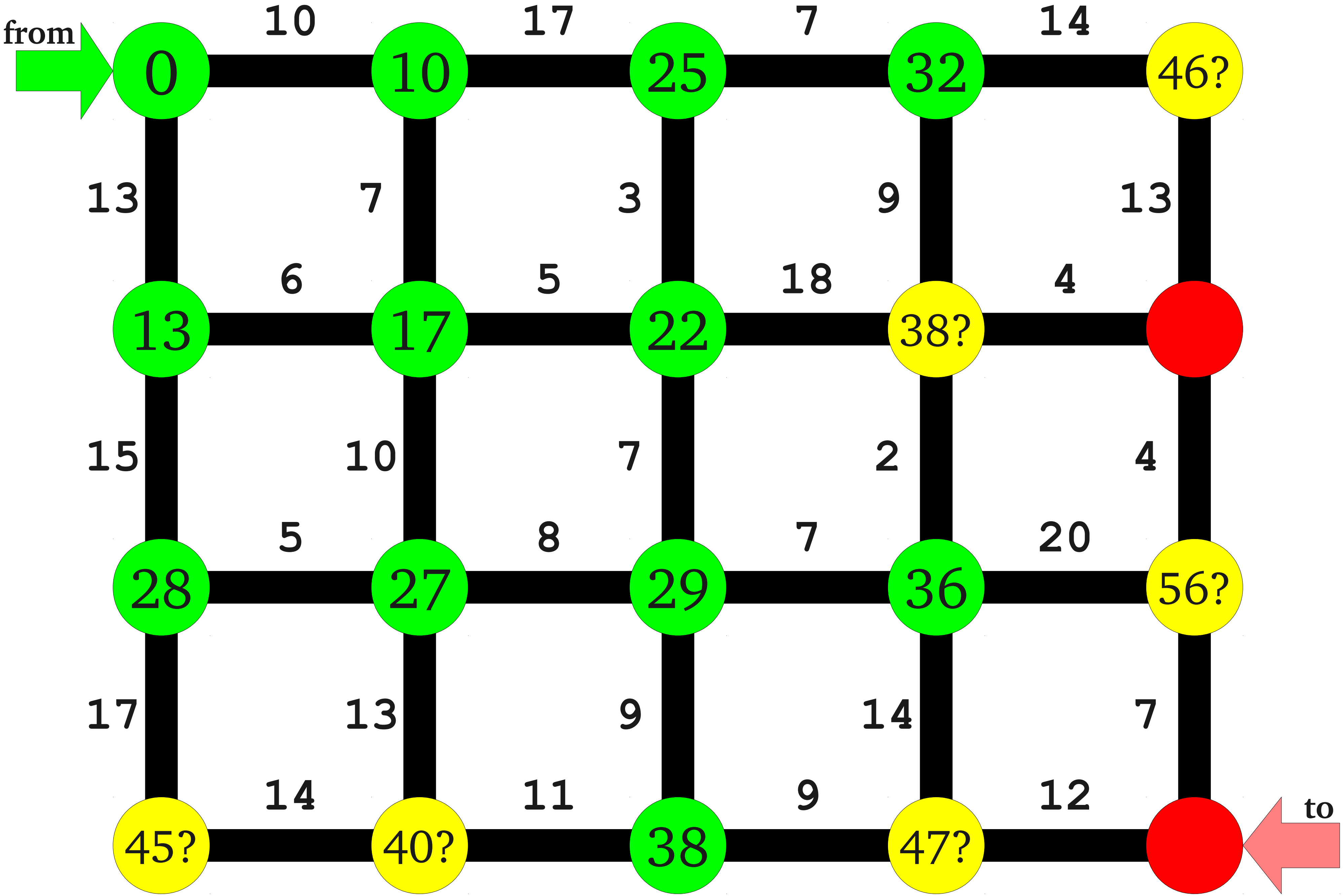

from
10
17
7
14
0
10
25
32
13
7
3
9
13
6
5
18
4
13
17
22
40?
15
10
7
2
4
5
8
7
20
28
27
29
36?
17
13
9
14
7
14
11
9
12
45?
40?
38?
to

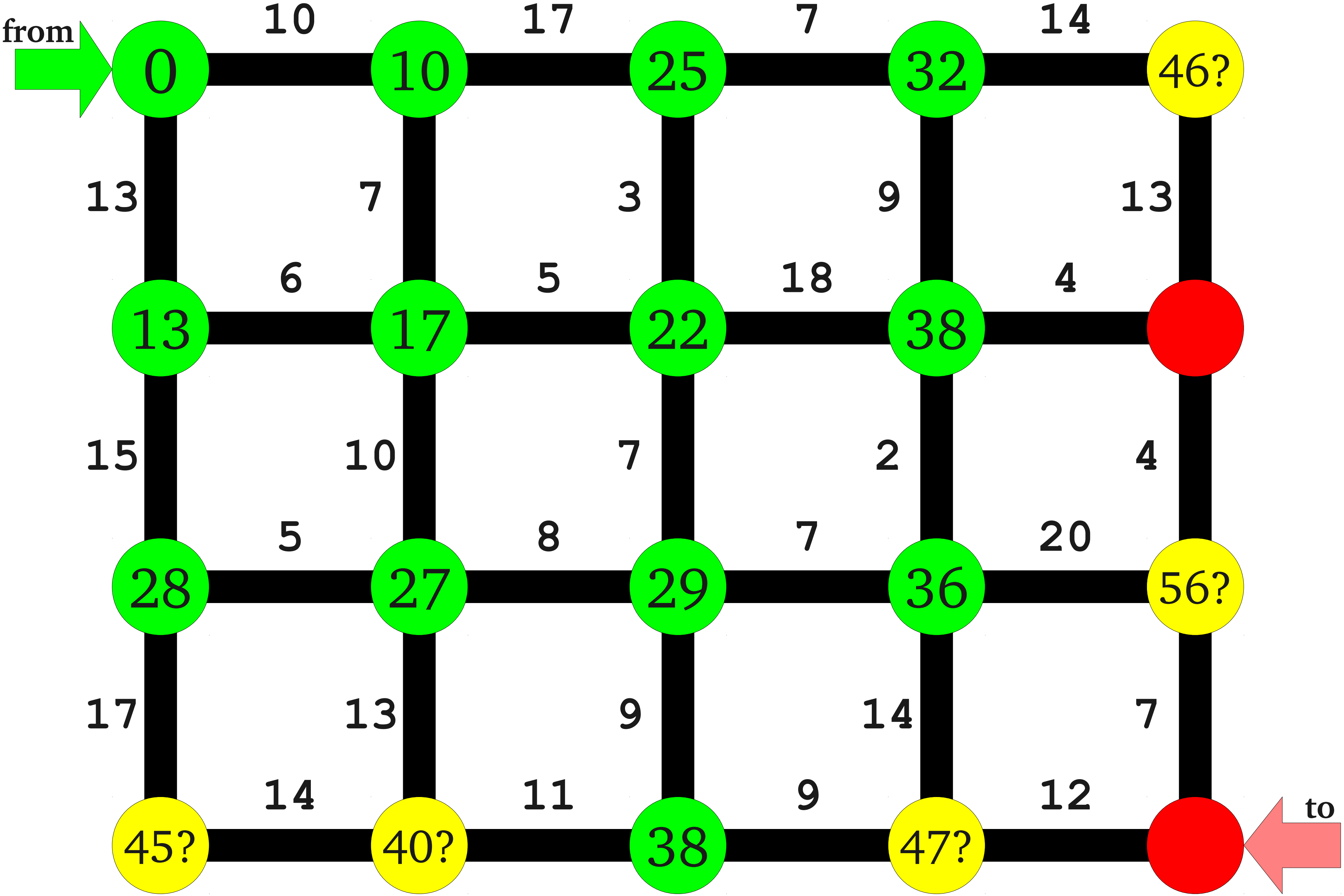

from
10
17
7
14
0
10
25
32
46?
13
7
3
9
13
6
5
18
4
13
17
22
40?
15
10
7
2
4
5
8
7
20
28
27
29
36?
17
13
9
14
7
14
11
9
12
to
45?
40?
38?

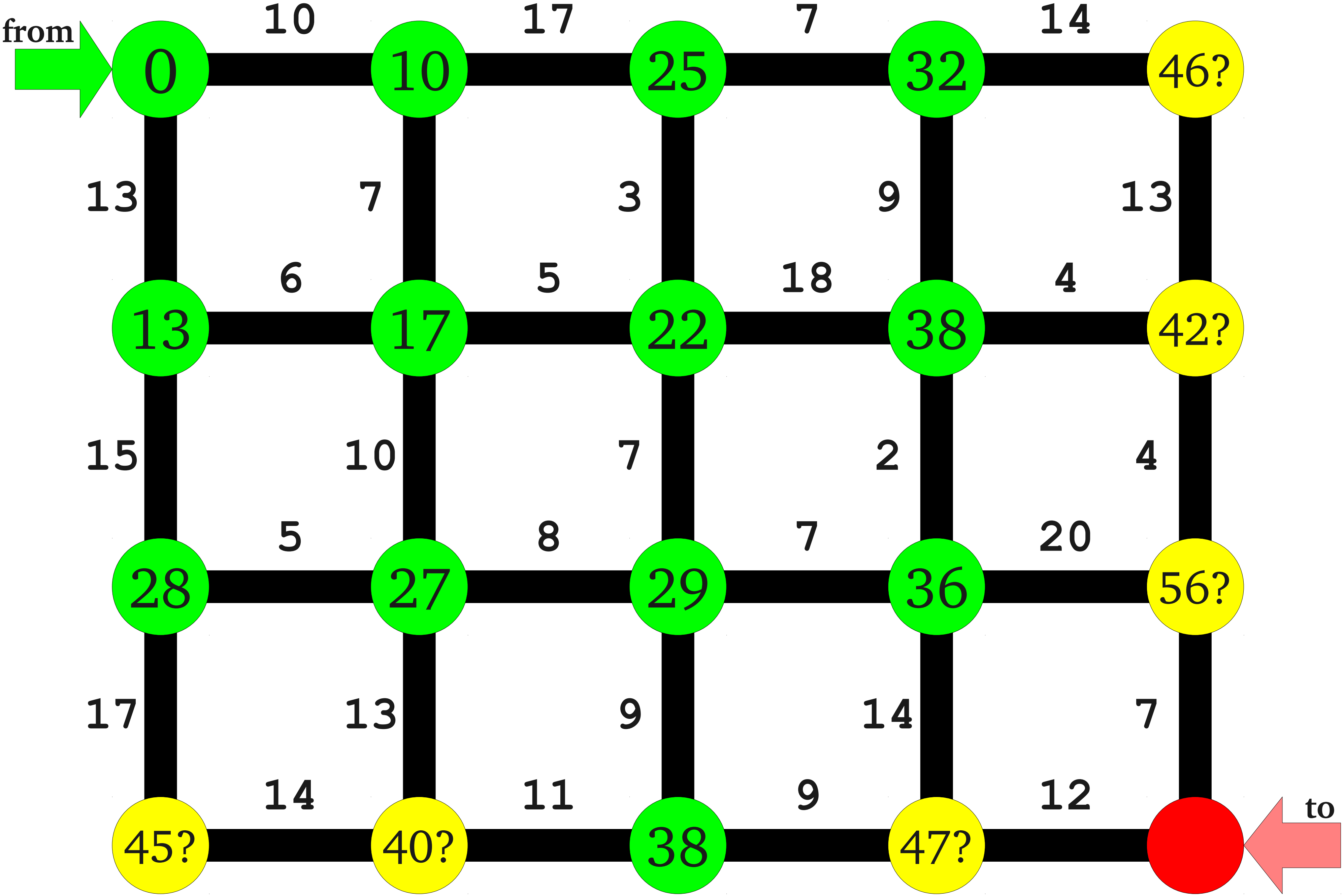

from
to
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12
0
10
25
32
46?
13
17
22
40?
28
27
29
36
45?
40?
38?

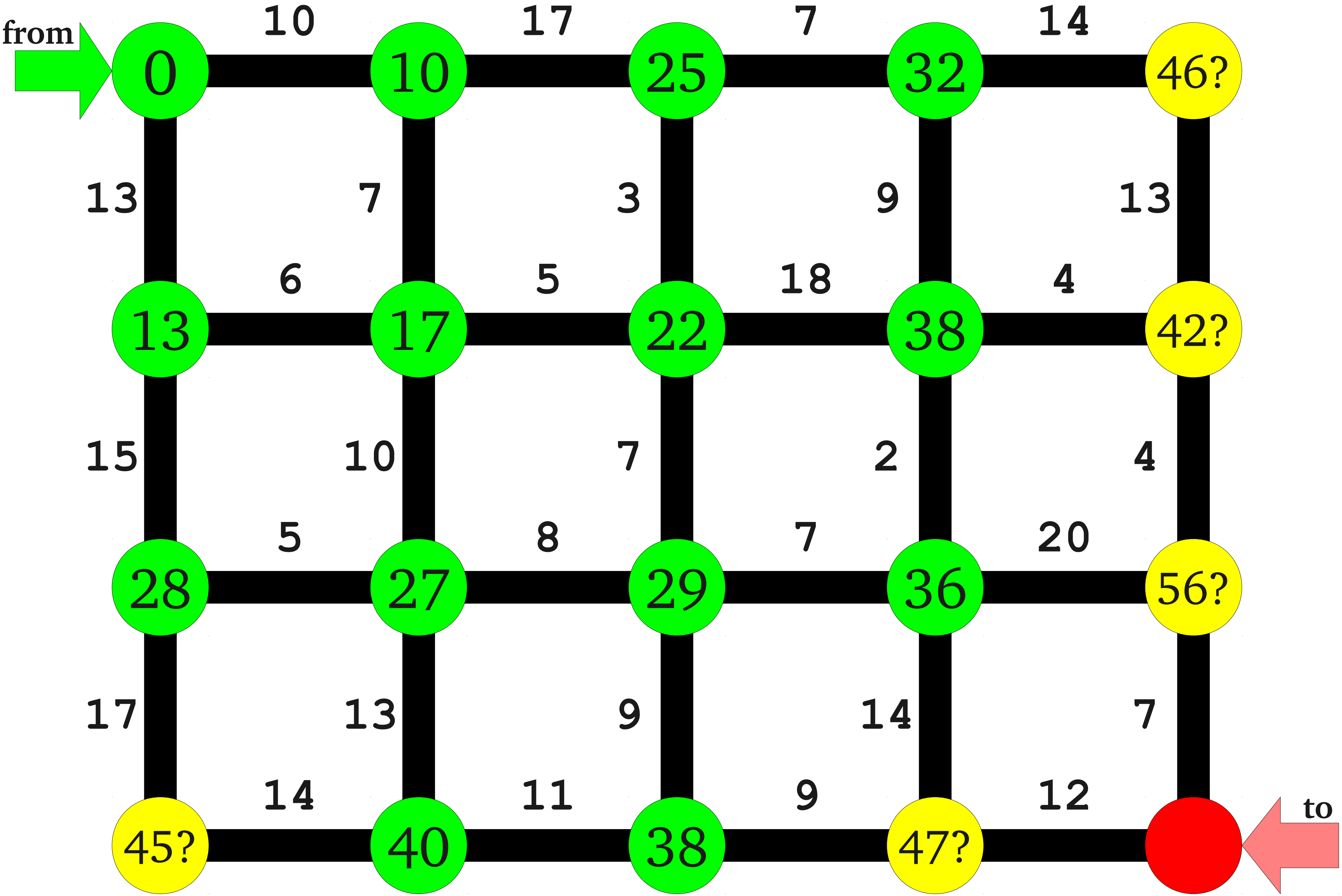

from
10
17
7
14
0
10
25
32
46?
13
7
3
9
13
6
5
18
4
13
17
22
38?
15
10
7
2
4
5
8
7
20
28
27
29
36
56?
17
13
9
14
7
14
11
9
12
to
45?
40?
38?
50?

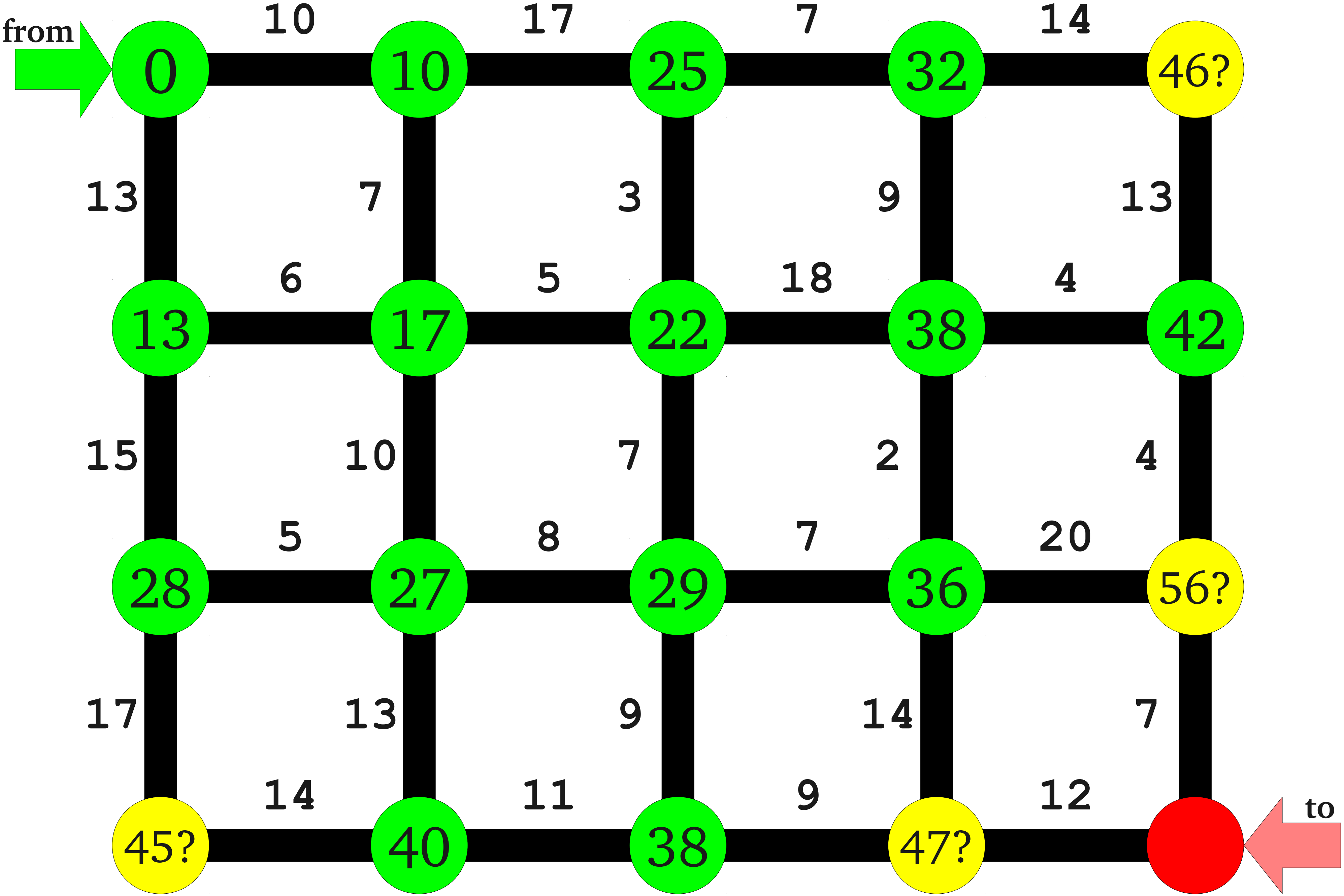

from
10
17
7
14
0
10
25
32
46?
13
7
3
9
13
6
5
18
4
13
17
22
38?
15
10
7
2
4
5
8
7
20
28
27
29
36
56?
17
13
9
14
7
14
11
9
12
to
45?
40?
38
50?

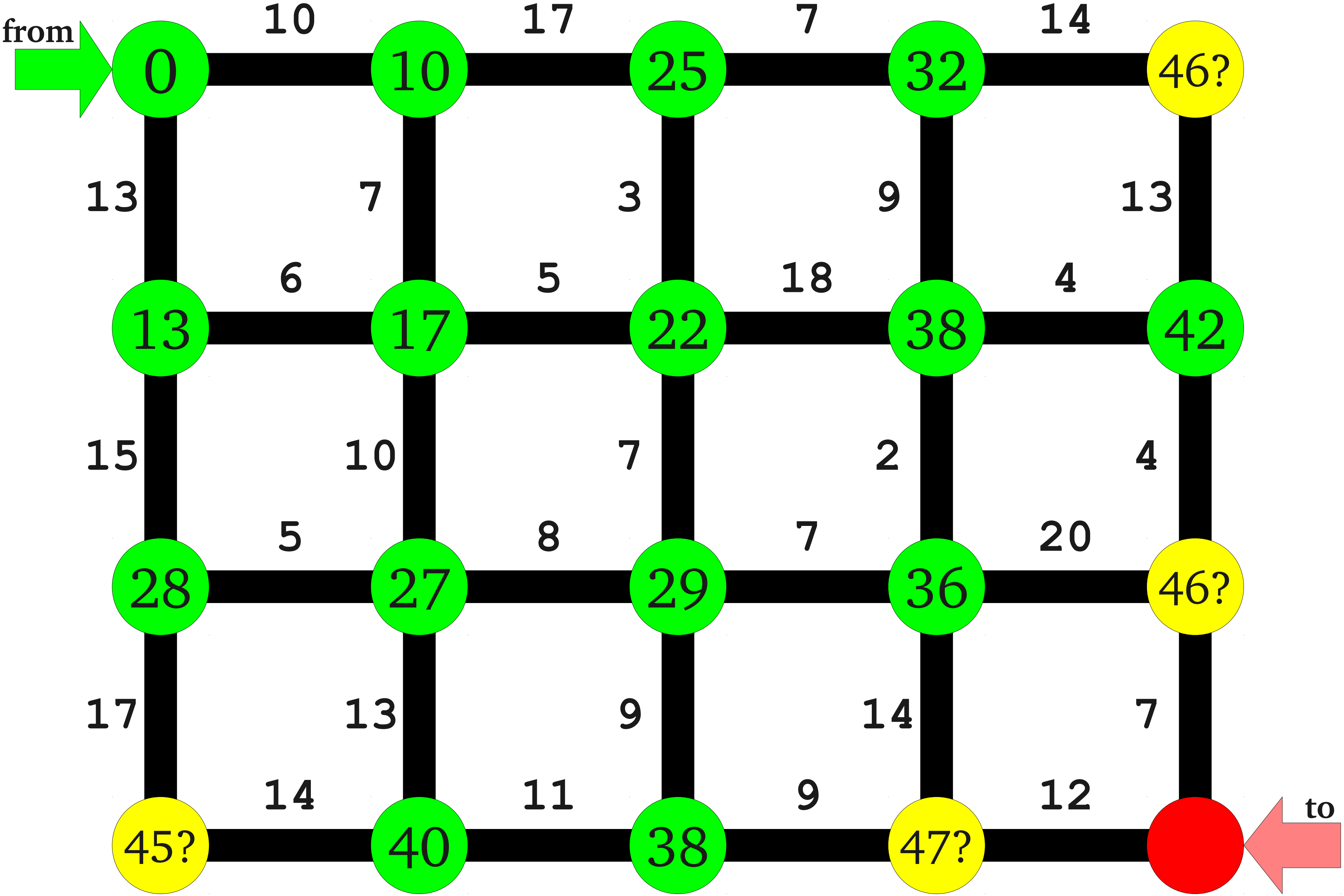

from
10
17
7
14
0
10
25
32
46?
13
7
3
9
13
6
5
18
4
13
17
22
38?
15
10
7
2
4
5
8
7
20
28
27
29
36
56?
17
13
9
14
7
14
11
9
12
to
45?
40?
38
47?

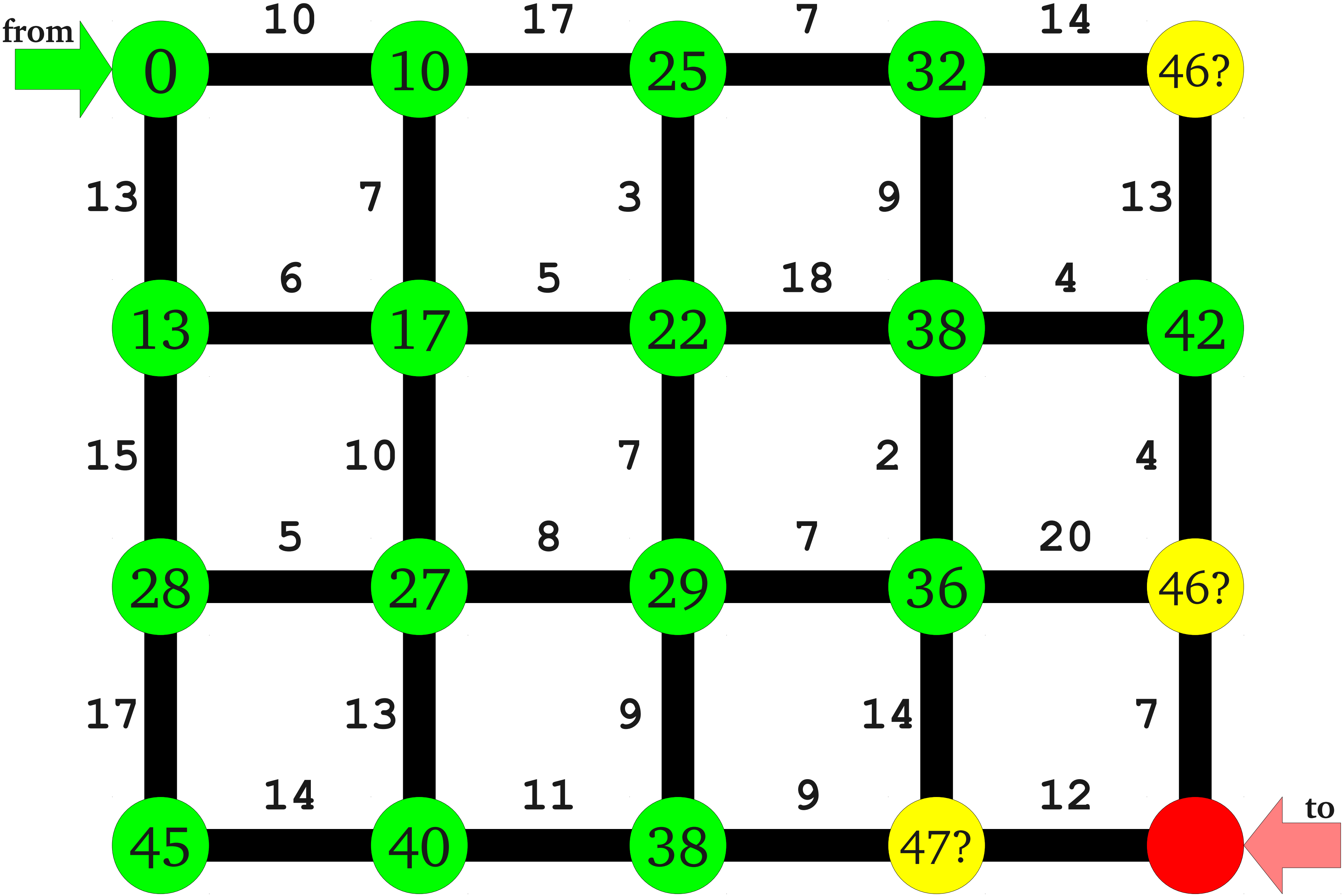

from
10
17
7
14
0
10
25
32
46?
13
7
3
9
13
6
5
18
4
13
17
22
38
15
10
7
2
4
5
8
7
20
28
27
29
36
56?
17
13
9
14
7
14
11
9
12
45?
40?
38
47?
to

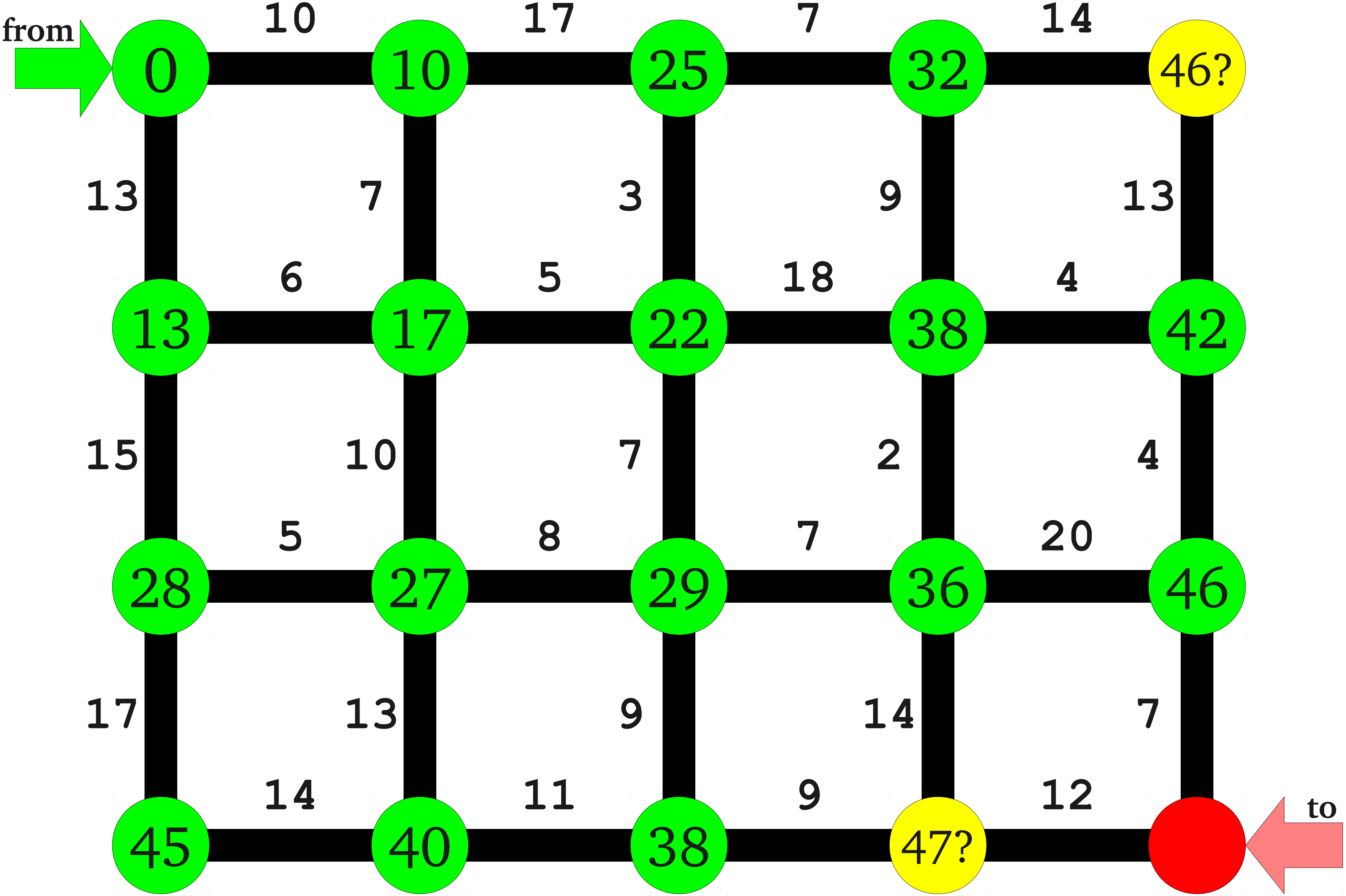

from
10
17
7
14
0
10
25
32
46?
13
7
3
9
13
6
5
18
4
13
17
22
38
42?
15
10
7
2
4
5
8
7
20
28
27
29
36
56?
17
13
9
14
7
14
11
9
12
to
45?
40?
38
47?

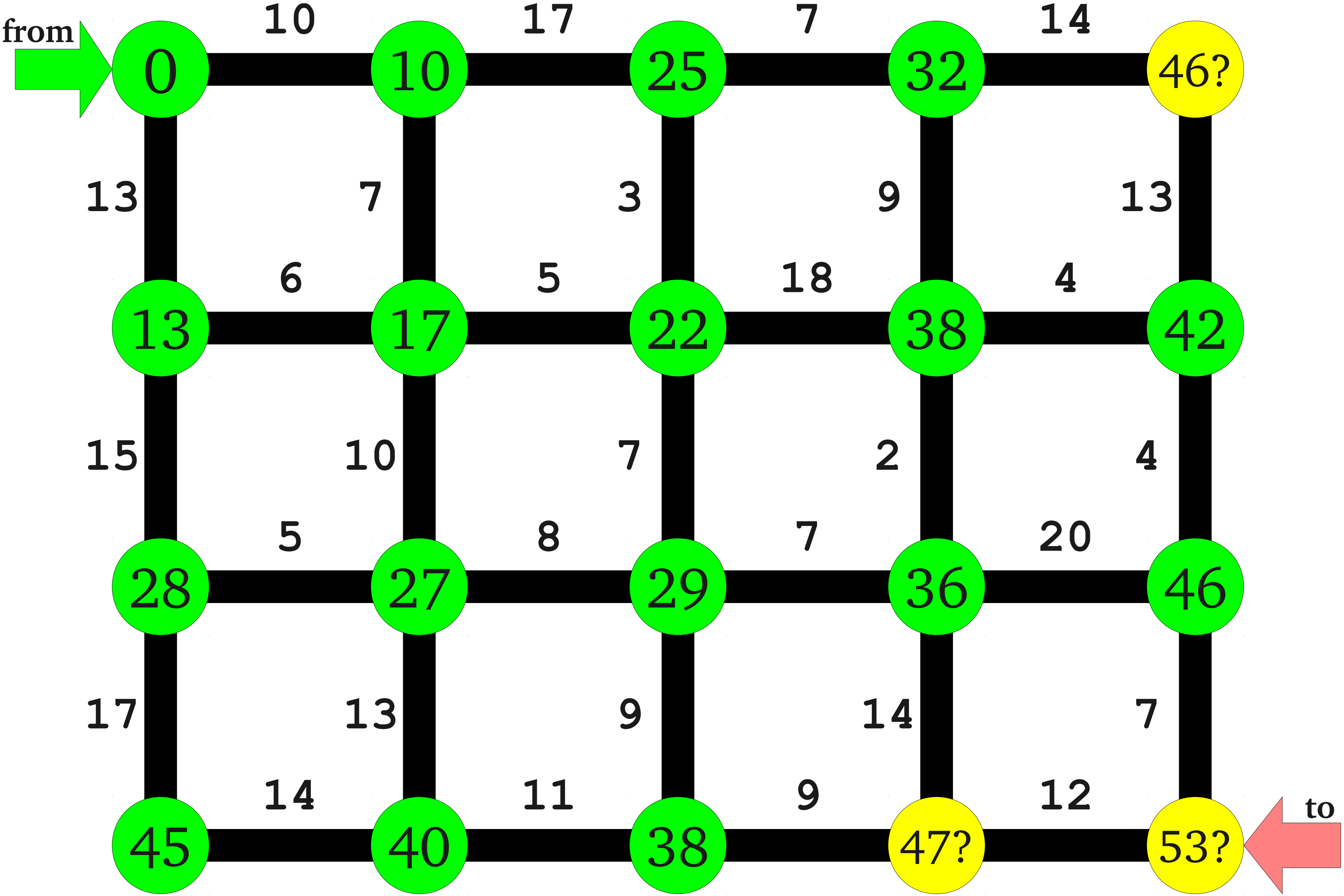

from
10
17
7
14
0
10
25
32
46?
13
7
3
9
13
6
5
18
4
13
17
22
38
42?
15
10
7
2
4
5
8
7
20
28
27
29
36
56?
17
13
9
14
7
14
11
9
12
45?
40
38
47?
to

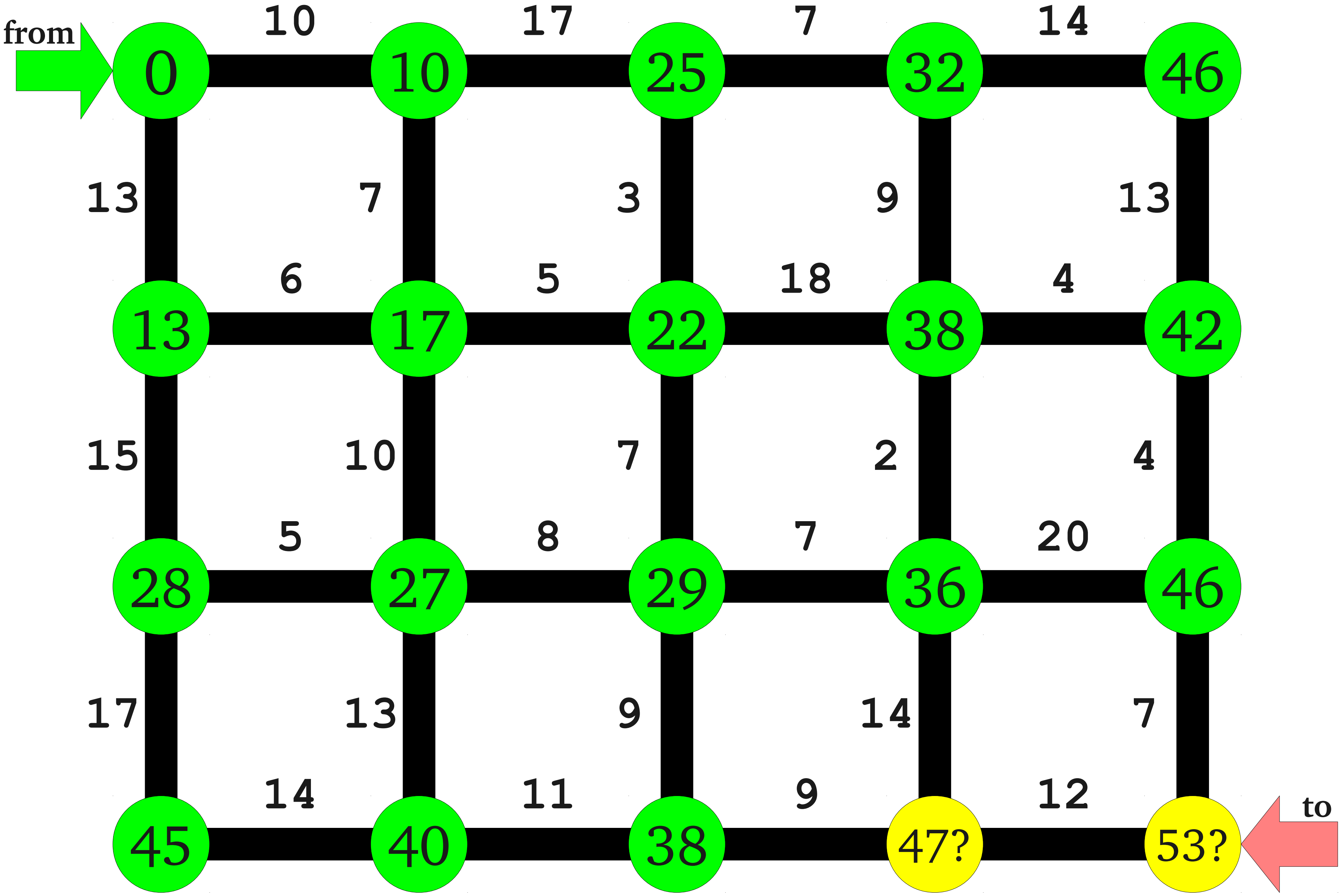

from
to
0
10
25
32
46?
13
17
22
38
42
28
27
29
36
56?
45?
40
38
47?
10
17
7
14
13
7
3
9
13
6
5
18
4
15
10
7
2
4
5
8
7
20
17
13
9
14
7
14
11
9
12

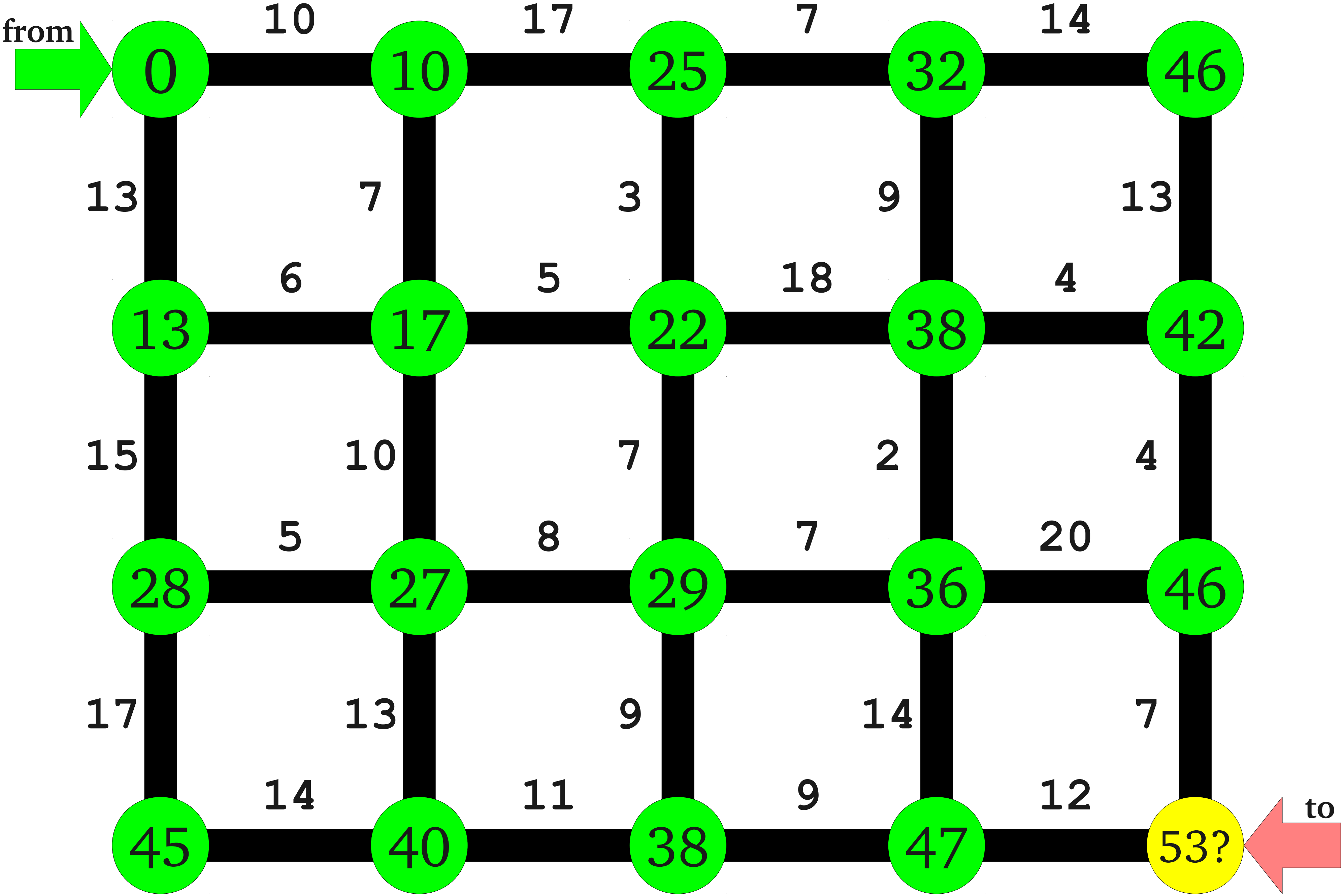

from
10
17
7
14
0
10
25
32
46?
13
7
3
9
13
6
5
18
4
13
17
22
38
42
15
10
7
2
4
5
8
7
20
28
27
29
36
46?
17
13
9
14
7
14
11
9
12
45?
40
38
47?
to

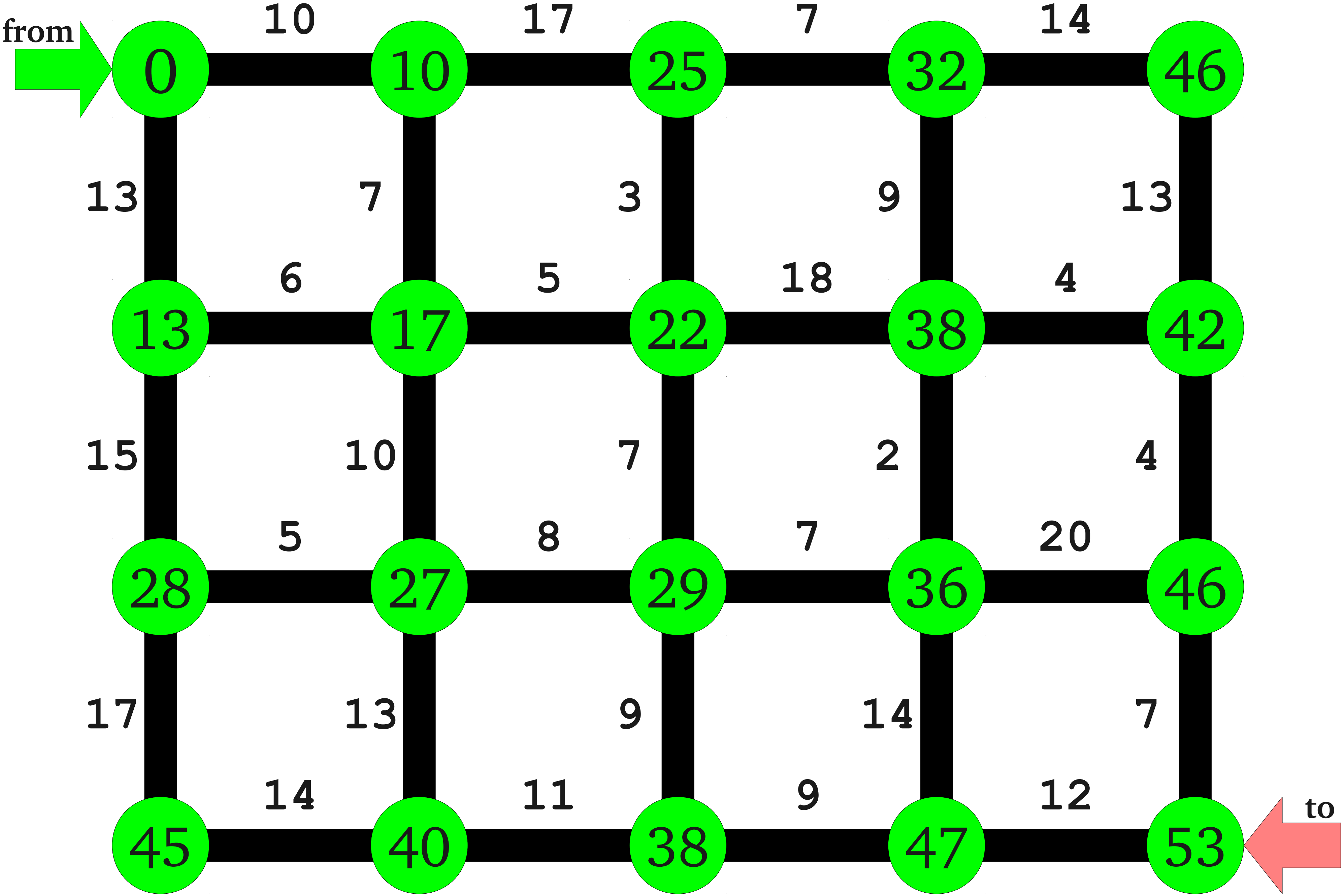

from
10
17
7
14
0
10
25
32
46?
13
7
3
9
13
6
5
18
4
13
17
22
38
42
15
10
7
2
4
5
8
7
20
28
27
29
36
46?
17
13
9
14
7
14
11
9
12
to
45
40
38
47?

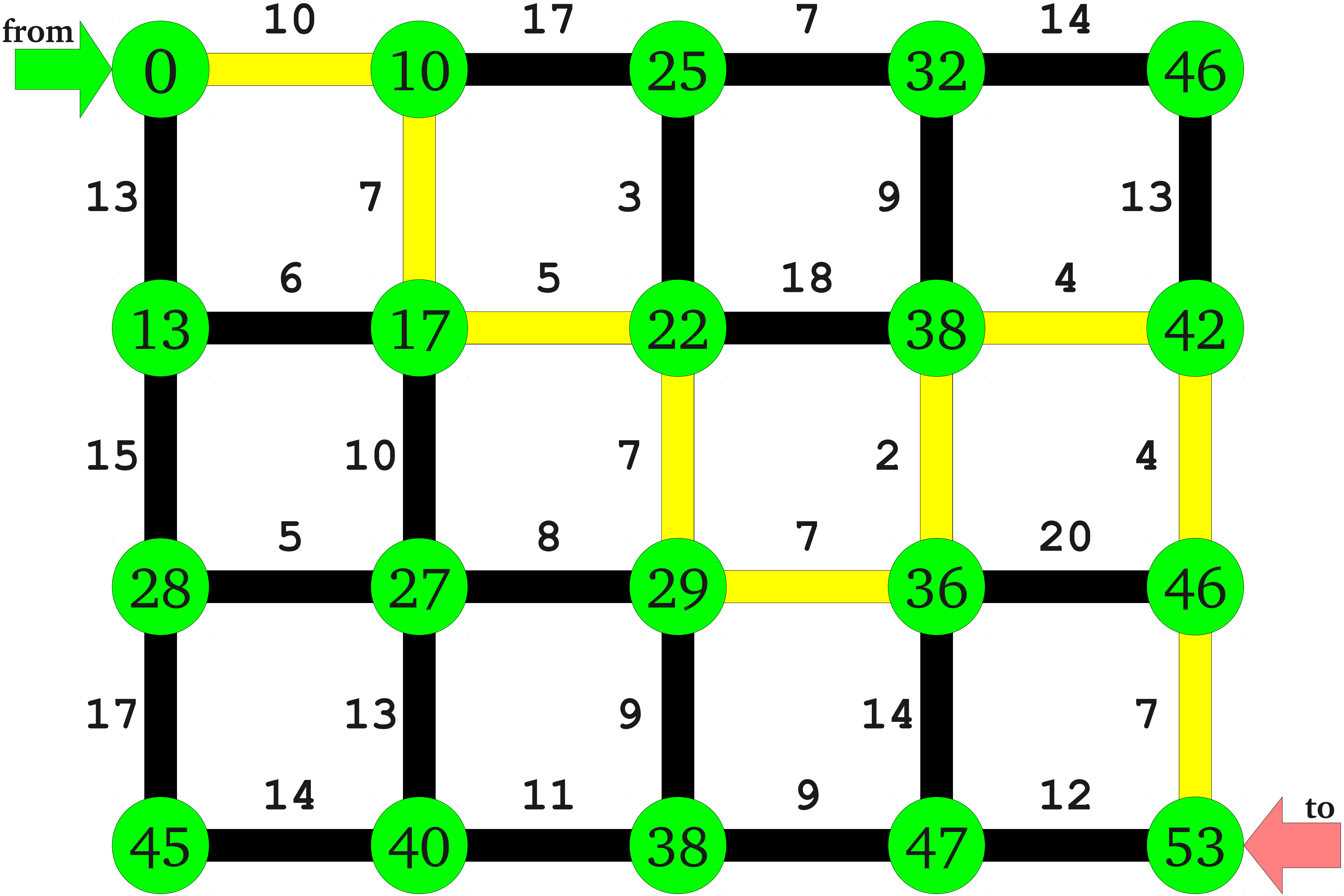

from
10
17
7
14
0
10
25
32
46?
13
7
3
9
13
6
5
18
4
13
17
22
38
42
15
10
7
2
4
5
8
7
20
28
27
29
36
46
17
13
9
14
7
14
11
9
12
to
45
40
38
47?

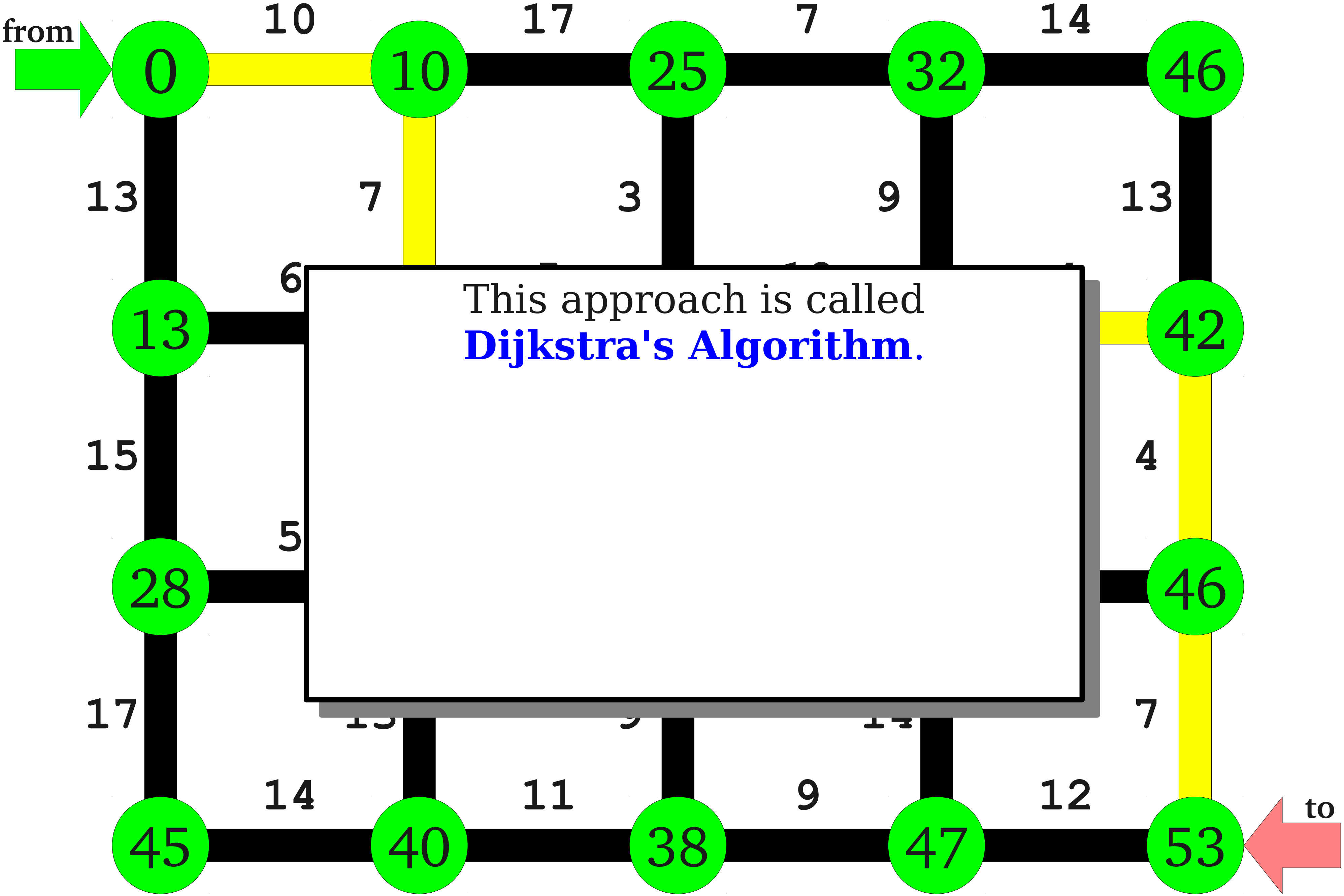

from
10
17
7
14
0
10
25
32
46?
13
7
3
9
13
6
5
18
4
13
17
22
38
42
15
10
7
2
4
5
8
7
20
28
27
29
36
46
17
13
9
14
7
14
11
9
12
to
45
40
38
47?
53?


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
45
40
38
47?
53?
to

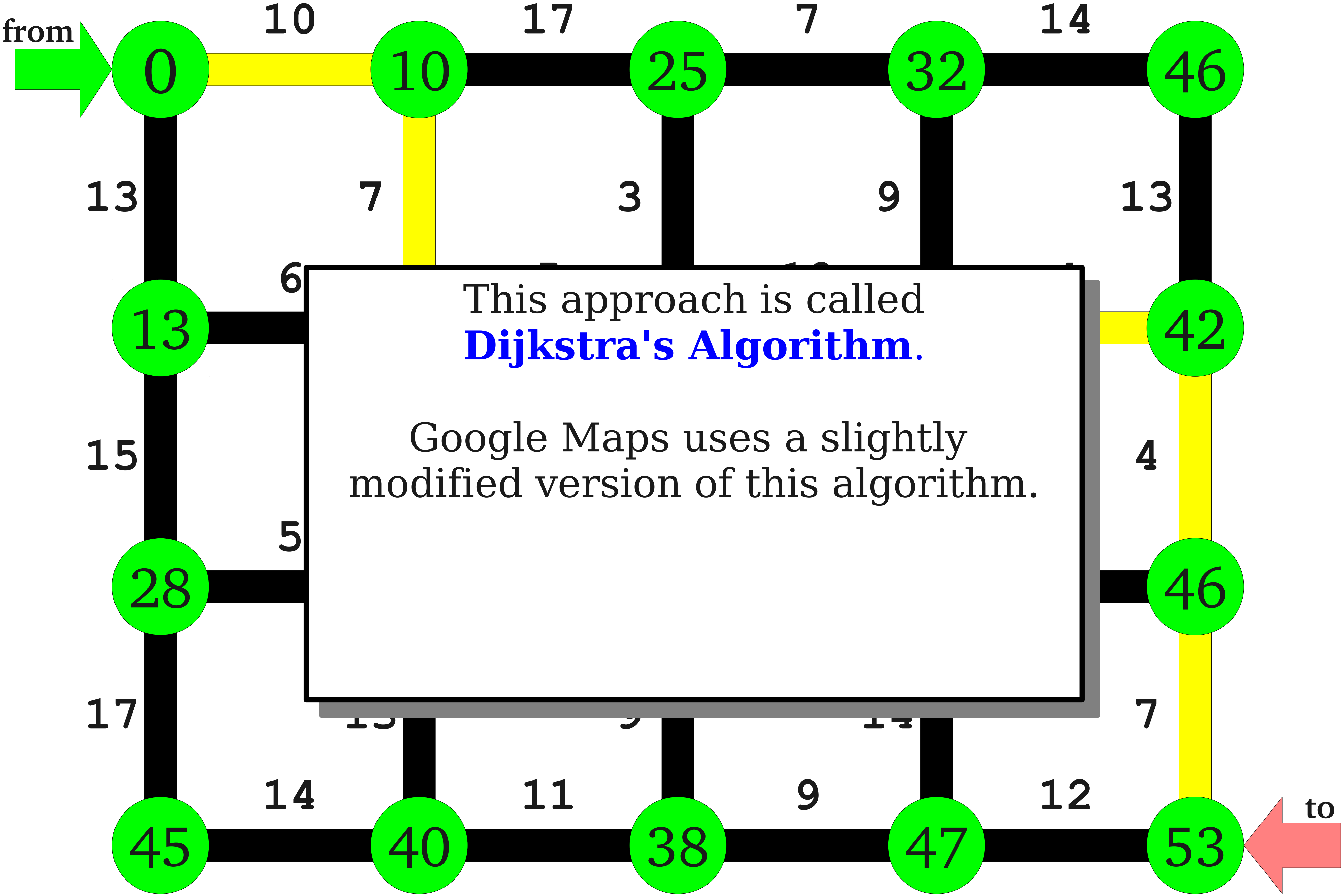

from
10
17
7
14
0
10
25
32
46
13
7
3
9
13
6
5
18
4
13
17
22
38
42
15
10
7
2
4
5
8
7
20
28
27
29
36
46
17
13
9
14
7
14
11
9
12
to
45
40
38
47
53?


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


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

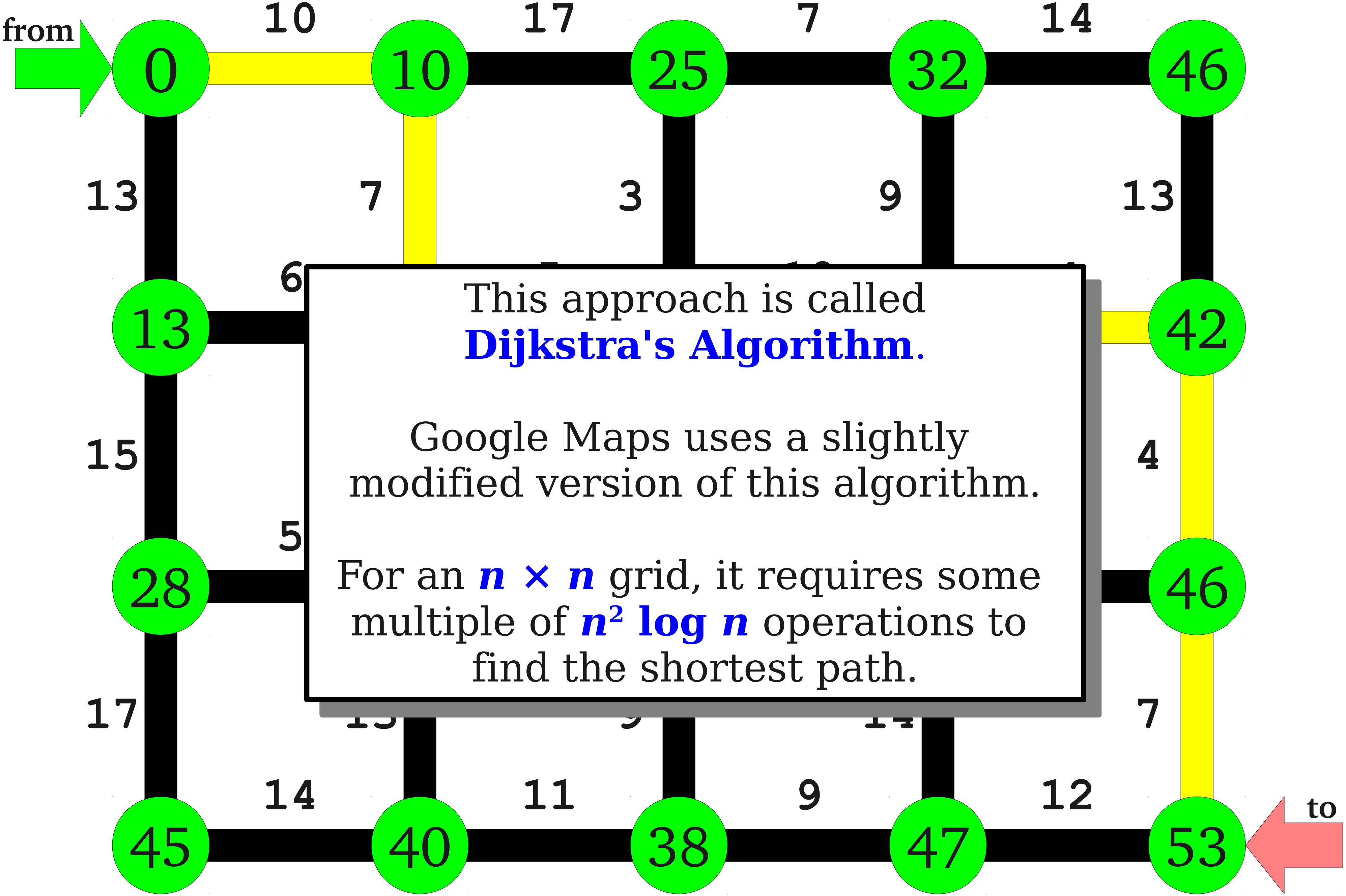


This approach is called
Dijkstra's Algorithm.



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 $n \times n$ grid, it requires some multiple of $n^2 \log n$ operations to find the shortest path.

Goals for this Course

- **Learn how to model and solve complex problems with computers.**
- To that end:
  - Explore common abstractions for representing problems.
  - Harness recursion and understand how to think about problems recursively.
  - Quantitatively analyze different approaches for solving problems.

Who's Here Today?

- African Studies
- Applied Physics
- 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!

One more detail...

C++

What is C++?

- Programming language developed in 1983 by Bjarne Stroustrup.
- Widely used for general programming when performance is important.
- Supports a variety of programming styles.

```cpp
/* File: hello-world.cpp
 *
 * A canonical Hello, world! program
 * in C++.
 */

#include <iostream>
using namespace std;

int main() {
    cout << "Hello, world!" << endl;
}
```

```cpp
/* File: retain-evens.cpp
 *
 * A program to filter out odd numbers from a list.
 */
#include <iostream>
#include "vector.h"
using namespace std;

Vector<int> retainEvens(Vector<int> values) {
    Vector<int> result;
    for (int i = 0; i < values.size(); i++) {
        if (values[i] % 2 == 0)
            result += values[i];
    }
    return result;
}

int main() {
    Vector<int> values;
    values += 1, 2, 3, 4, 5;

    Vector<int> processed = retainEvens(values);

    for (int i = 0; i < processed.size(); i++) {
        cout << processed[i] << endl;
    }
}
```

CS106L

- Optional, one-unit companion course to CS106B.
- In-depth treatment of C++'s libraries and language features.
- Excellent complement to the material from CS106B; highly recommended!
- Not a replacement for section; it's purely an add-on.

Next Time

- **Welcome to C++!**
  - Defining functions.
  - Reference parameters.
  - Introduction to recursion.