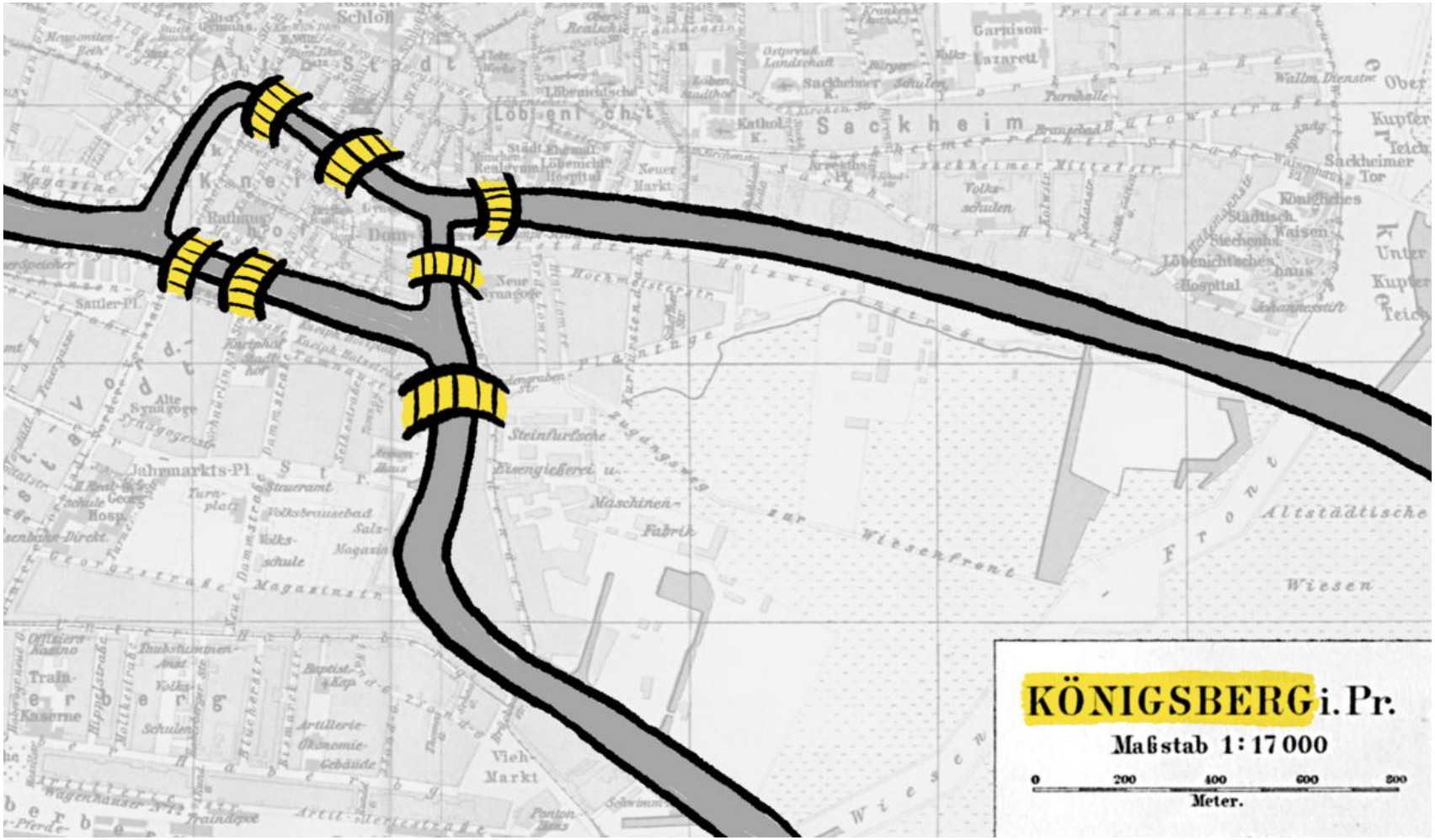
Week 5th

Prepared by Dr Syed Khaldoon Khurshid
Powered by Brilliant Application

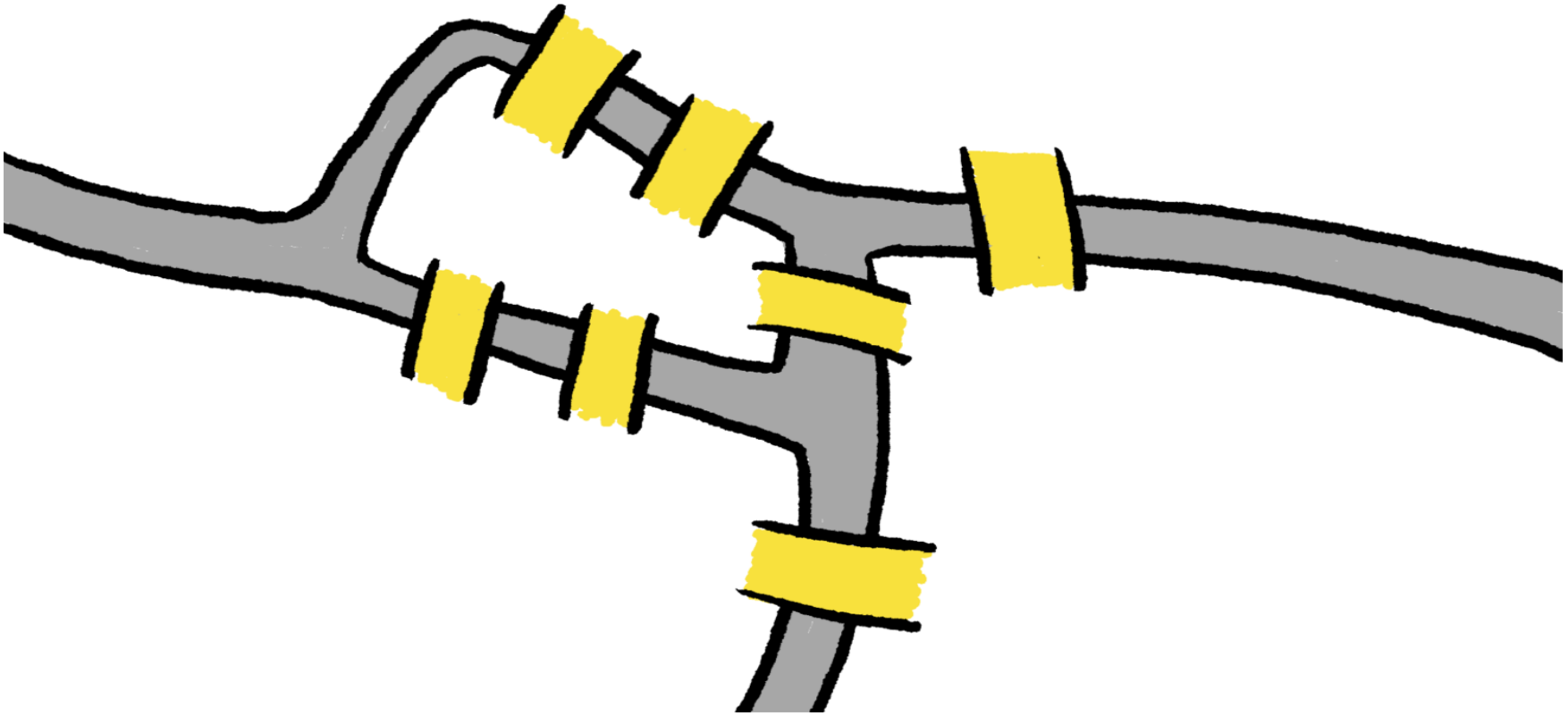
Searching for solutions

- The Russian city of Kaliningrad lies about 30 kilometers north of Poland, where the Pregolya river meets the Baltic Sea.

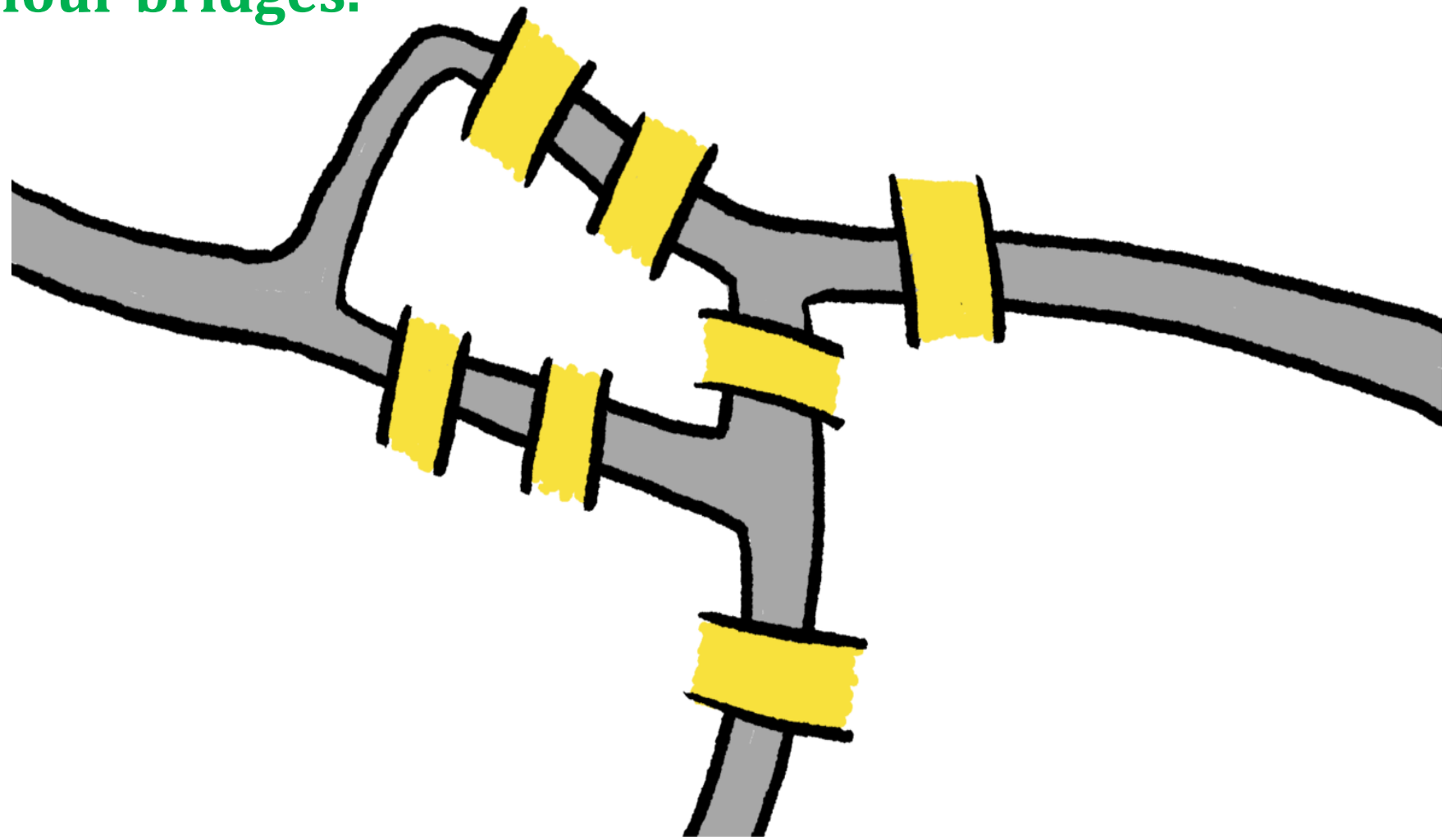
The city has a famous place in mathematical history dating back to the eighteenth century, when the city was named Königsberg and had seven bridges connecting two river islands in the heart of the city.



- **This arrangement of Königsberg's seven bridges led people to ask a simple yes-or-no question. Could you start anywhere in the city and cross every bridge exactly once? (No swimming allowed!)**

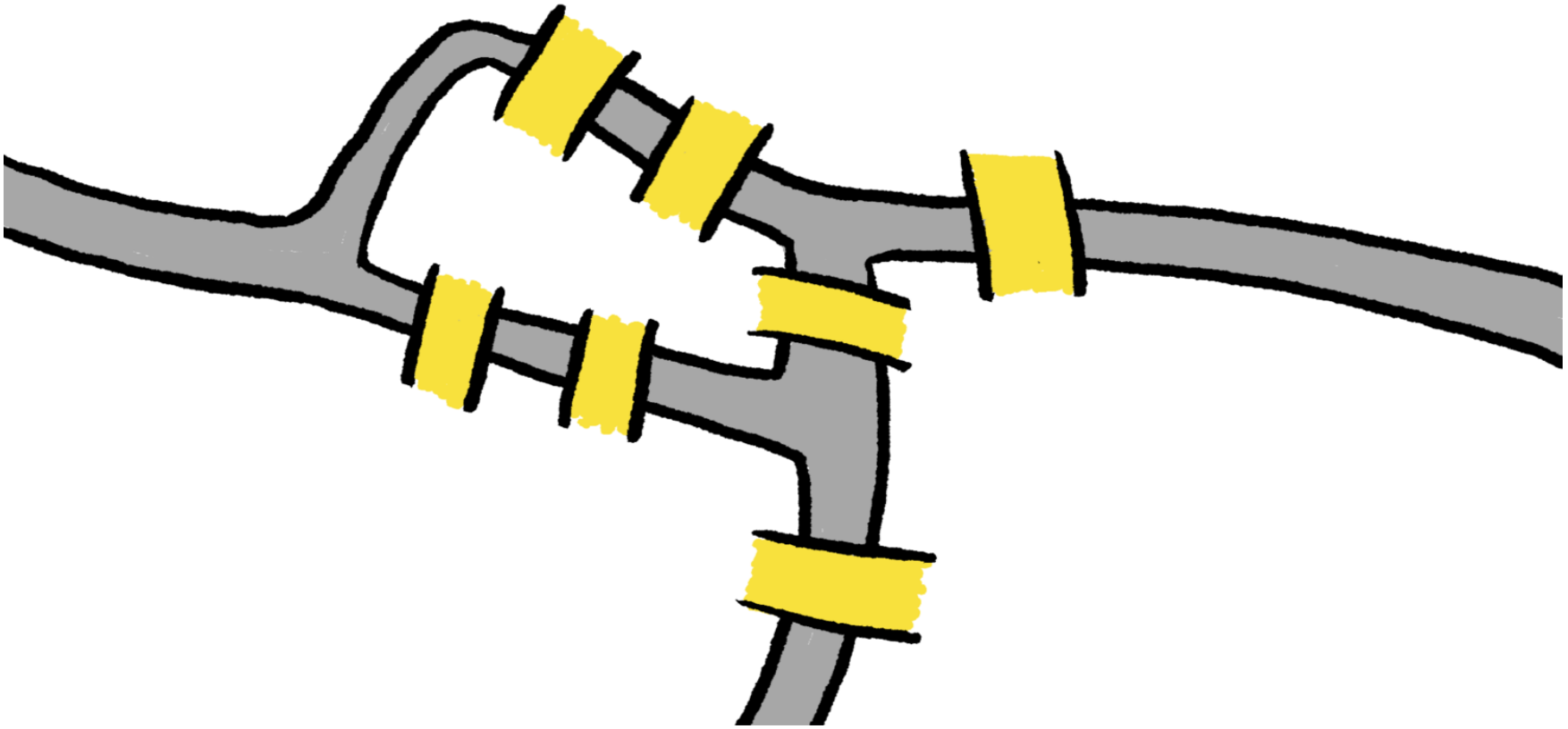


It's easy to attempt the puzzle and get stuck. **Here is a path through the city that gets stuck after crossing four bridges.**

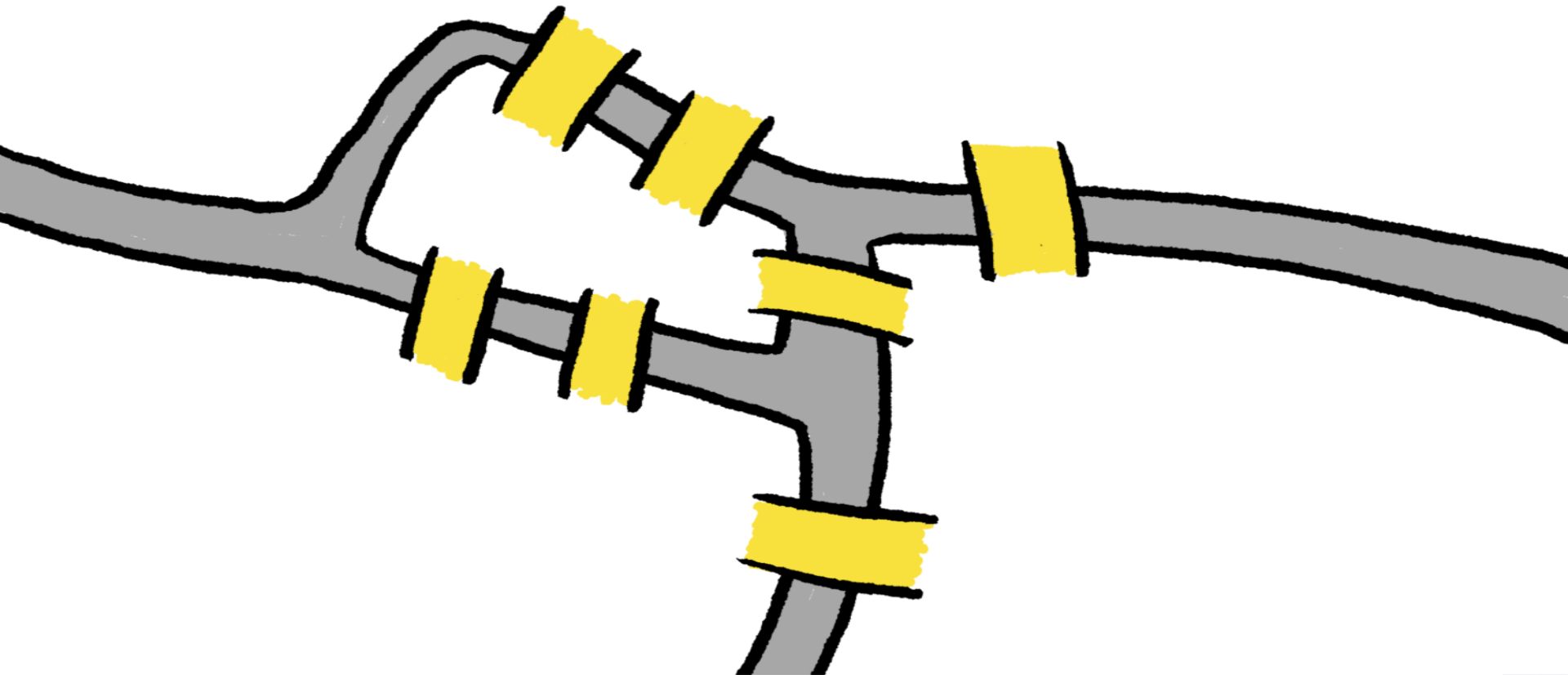


- It turns out that it was not possible to cross all seven bridges of Königsberg exactly once, even if you can start and end anywhere.
- **The interesting computer science enters the picture when you try to explain *why* you can't achieve this goal!**

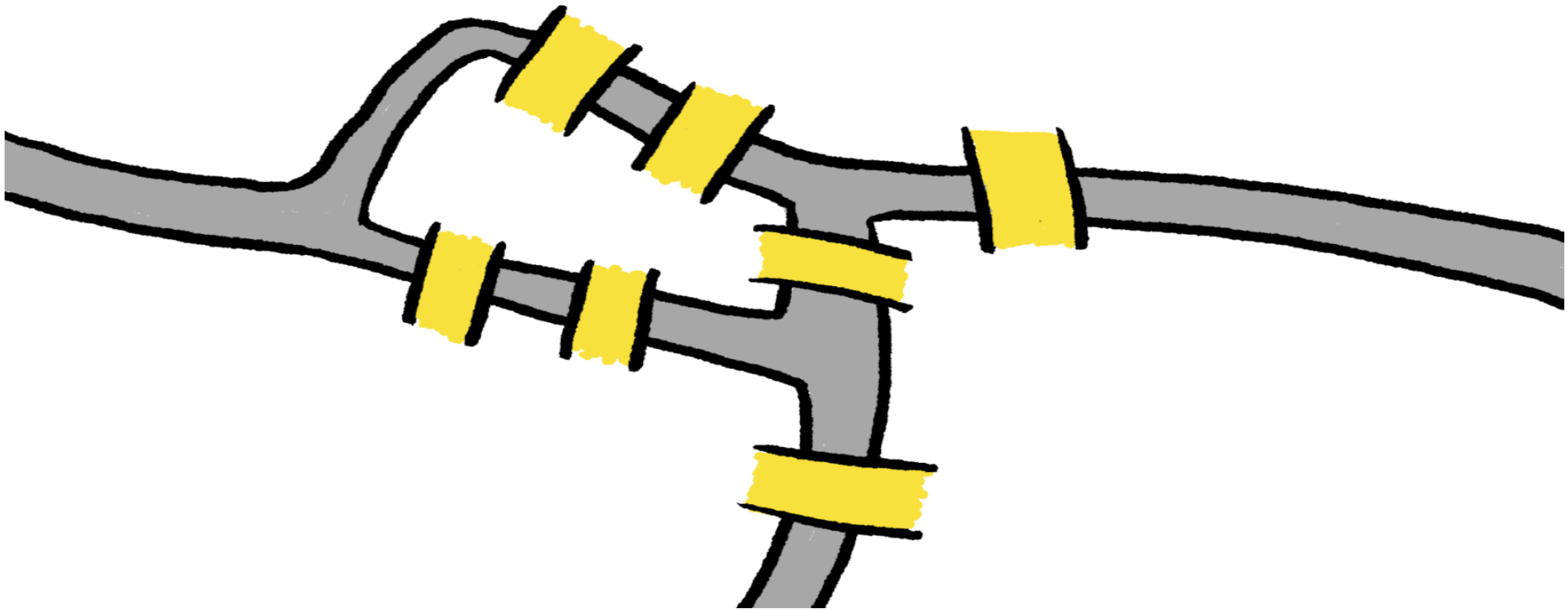
- **Can you find a path that crosses six bridges before getting stuck? Remember, you're not allowed to cross any bridge twice.**



- Let's turn the problem on its head to make sure we understand it. If you're allowed to start anywhere, but you can't cross any single bridge twice, **what is the *fewest* number of bridges you can cross before getting stuck?**



- **If you didn't know whether or not there was a solution to the puzzle of the bridges of Königsberg, you could try to find one by moving around randomly. Start at a random place, and then move across a random bridge until you succeed or get stuck.**

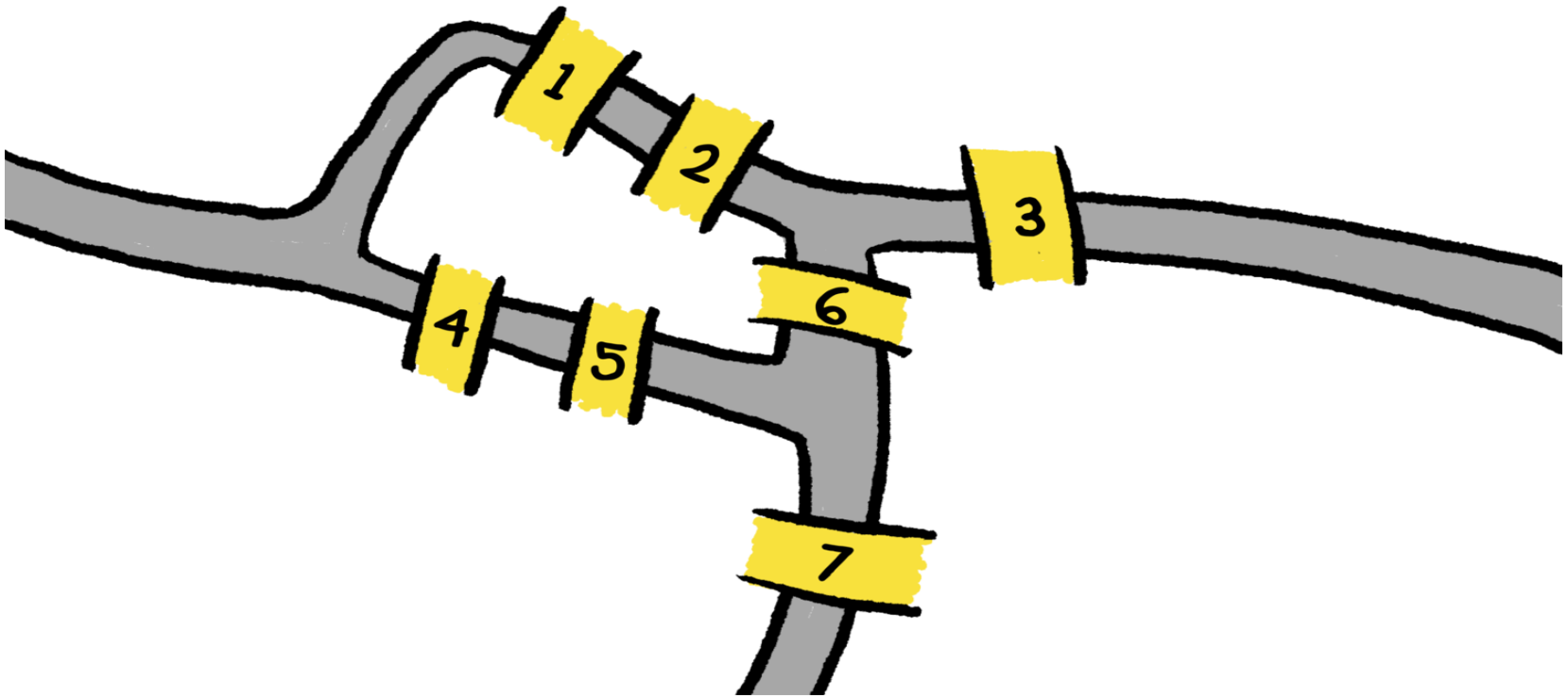


- **Repeating this process over and over will always eventually find a path if there is a path to be found. However, repeatedly performing random explorations is not a good way for us to convince ourselves that no path exists. We can never be sure that there's not some solution we've missed by getting unlucky!**

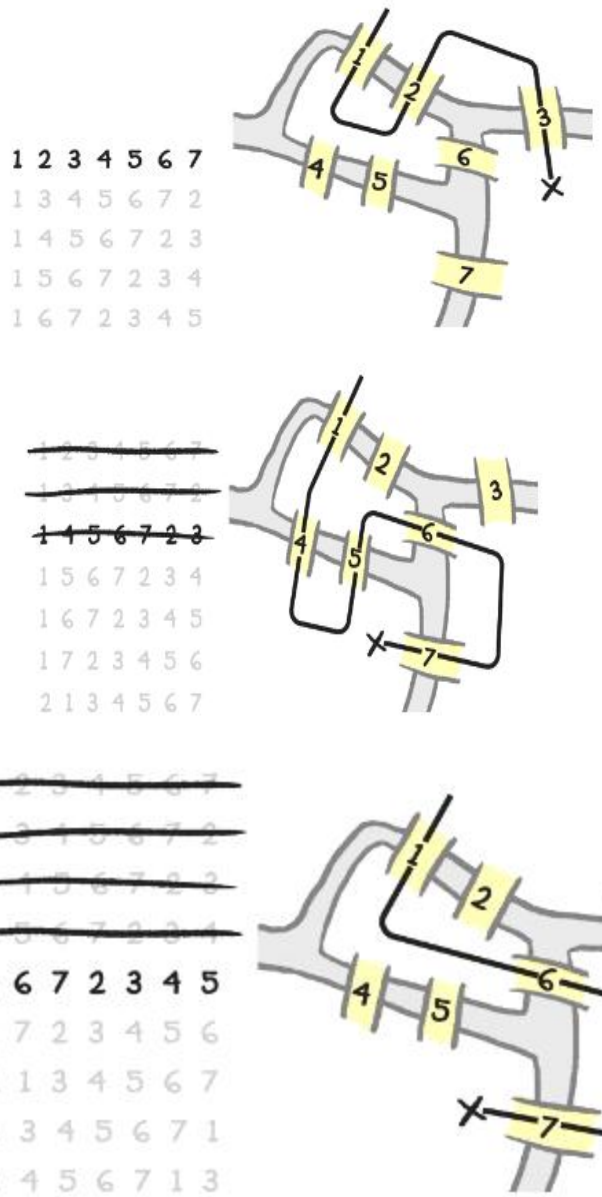
- You might try to imagine a **systematic way** to explore the city, **a method that ensures you will try *every possible combination of bridges*.** One systematic way of exploring that computer scientists use is called ***depth-first search***, but that's a topic for a different lecture/ discussion.

- In computer science, it's sometimes useful to look for a ***brute force solution***.
- **If you can describe what all the possible solutions look like, and if you can check what it means for a solution to be correct or valid, then just check all the possible solutions until you find one that works!**

- In the case of the bridges of Königsberg, we can number the seven bridges 1, 2, 3, 4, 5, 6, and 7.



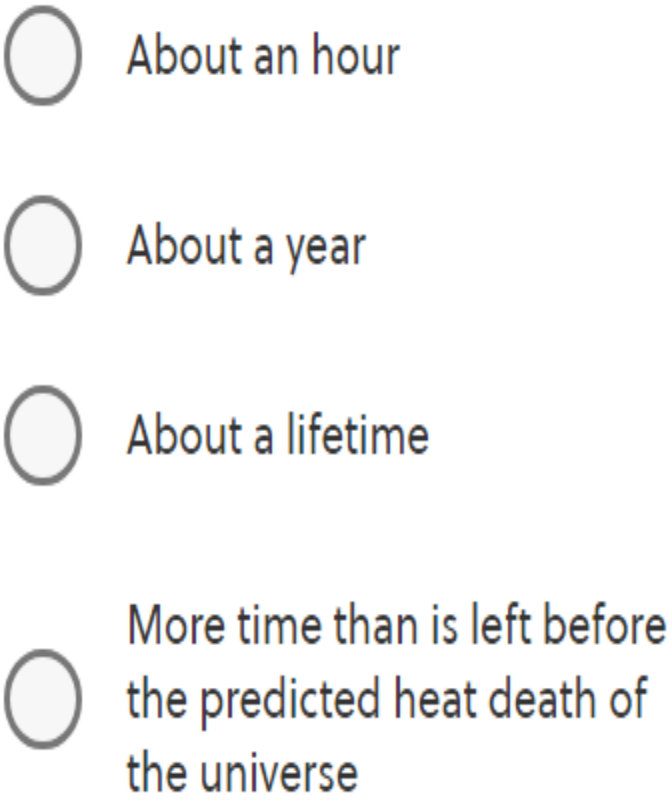
- Any solution to the puzzle is going to be some sequence of seven bridges that includes each of the seven bridges: that is, a **permutation** of the sequence 1, 2, 3, 4, 5, 6, 7.
- We can list each possible permutation, and then check each permutation in turn to see if that sequence of bridges represents a path that we can physically walk (without crossing a river). If even one of these sequences is possible to carry out, then it represents a solution to the puzzle.


1 2 3 4 5 6 7
1 3 4 5 6 7 2
1 4 5 6 7 2 3
1 5 6 7 2 3 4
1 6 7 2 3 4 5
1 7 2 3 4 5 6
2 1 3 4 5 6 7
2 3 4 5 6 7 1
2 4 5 6 7 1 3
2 5 6 7 1 3 4
2 6 7 1 3 4 5

- How many bridge permutations would we need to check in order to be sure that there is no way to walk the seven bridges of Königsberg exactly once**?**
  - **Correct answer:** About 5,000
  - The number of permutations of 7 elements is $7! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 = 5040$.
  - This is probably too many to check by hand, but with a computer we can check each of these examples pretty quickly!

- Imagine that a computer can check 5040 different permutations of bridges every second. (That would mean that it only takes a second to check all the permutations of bridges in Königsberg.)
- A 2006 study of Pittsburgh, Pennsylvania determined that the city had 446 bridges.

- If we use the same approach to see whether all the bridges in Pittsburgh can be crossed once without leaving any roadways, how long would that take?



- **Correct answer: More time than is left before the predicted heat death of the universe**
- The factorial function n! measures the number of permutations of n elements, and the factorial function gets way bigger as n gets bigger.
- The easy way to guess the answer is to use the rule of thumb that the factorial function gets really big, really fast.

- $7!$ is 5040. (We learned that from the example of the bridges of Königsberg.) That's the number of permutations you can check every second.
- In a minute, you can calculate less than $9! = 7! \cdot 8 \cdot 9$ permutations. There are 60 seconds in a minute and $60 < 8 \cdot 9 = 72$.
- In an hour, you can calculate less than $11!$ permutations by the same reasoning. $11! = 9! \cdot 10 \cdot 11$, there are 60 minutes in an hour, and $60 < 10 \cdot 11$.
- In a day, you can calculate less than $13!$ permutations, because $24 < 12 \cdot 13$.
- In a year, you can calculate less than $16!$ permutations, because $365 < 14 \cdot 15 \cdot 16$.
- In a human lifetime, which is currently expected to be less than $17 \cdot 18$ years, you could therefore calculate less than $18!$ permutations.

The calculation above means that only the last answer can be right. Here's how to justify that answer, using the estimate that the predicted heat death of the universe will be in something like 10^{100} years.

You will be able to check fewer than $116!$ permutations before the heat death of the universe. You saw that you could check less than $16!$ permutations in a year, and

$$116! = 1 \times 2 \times \cdots \times 16 \times 17 \times \cdots \times 116.$$

We know that $116!$ is bigger than

$$16! \times 10^{100} = 1 \times 2 \times \cdots 16 \times \underbrace{10 \times \cdots \times 10}_{\#\text{ of } 10\text{'s} = 100}.$$

That means that checking 116 permutations definitely takes more time than we have left before the predicted heat death of the universe, and checking the 446 permutations of Pittsburgh bridges will take much, much longer.

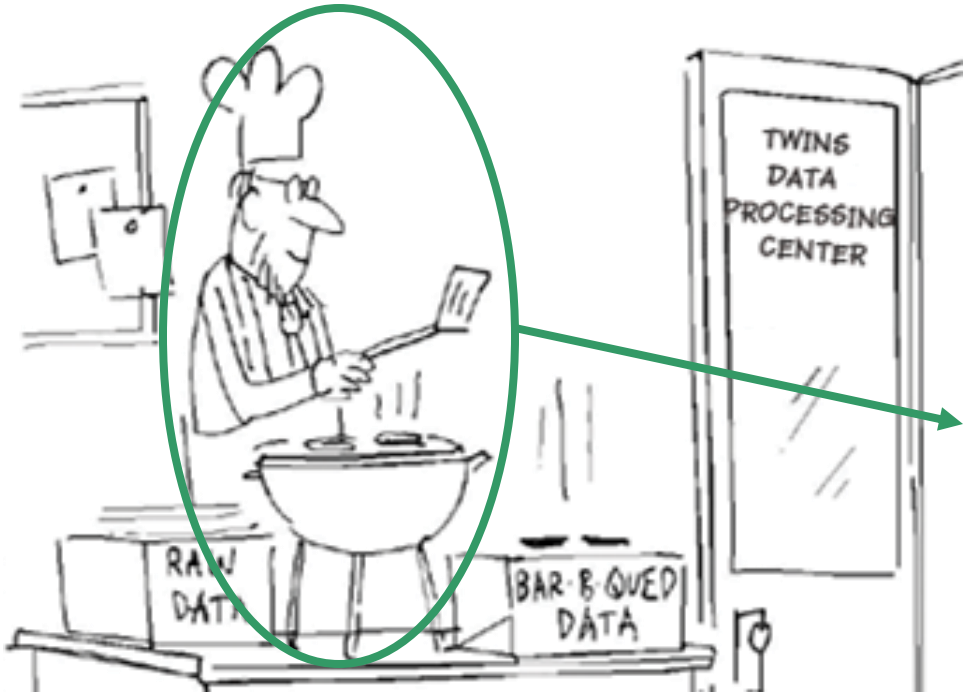
Conclusion

- **It isn't too hard to figure out if any specific sequence of bridges represents a valid way through Königsberg. We can therefore solve the puzzle of the bridges of Königsberg by listing all the permutations of bridges and checking whether each combination works.**
- **This brute force solution is not a particularly satisfying solution! It certainly doesn't help us once there are a couple more bridges.**

Conclusion

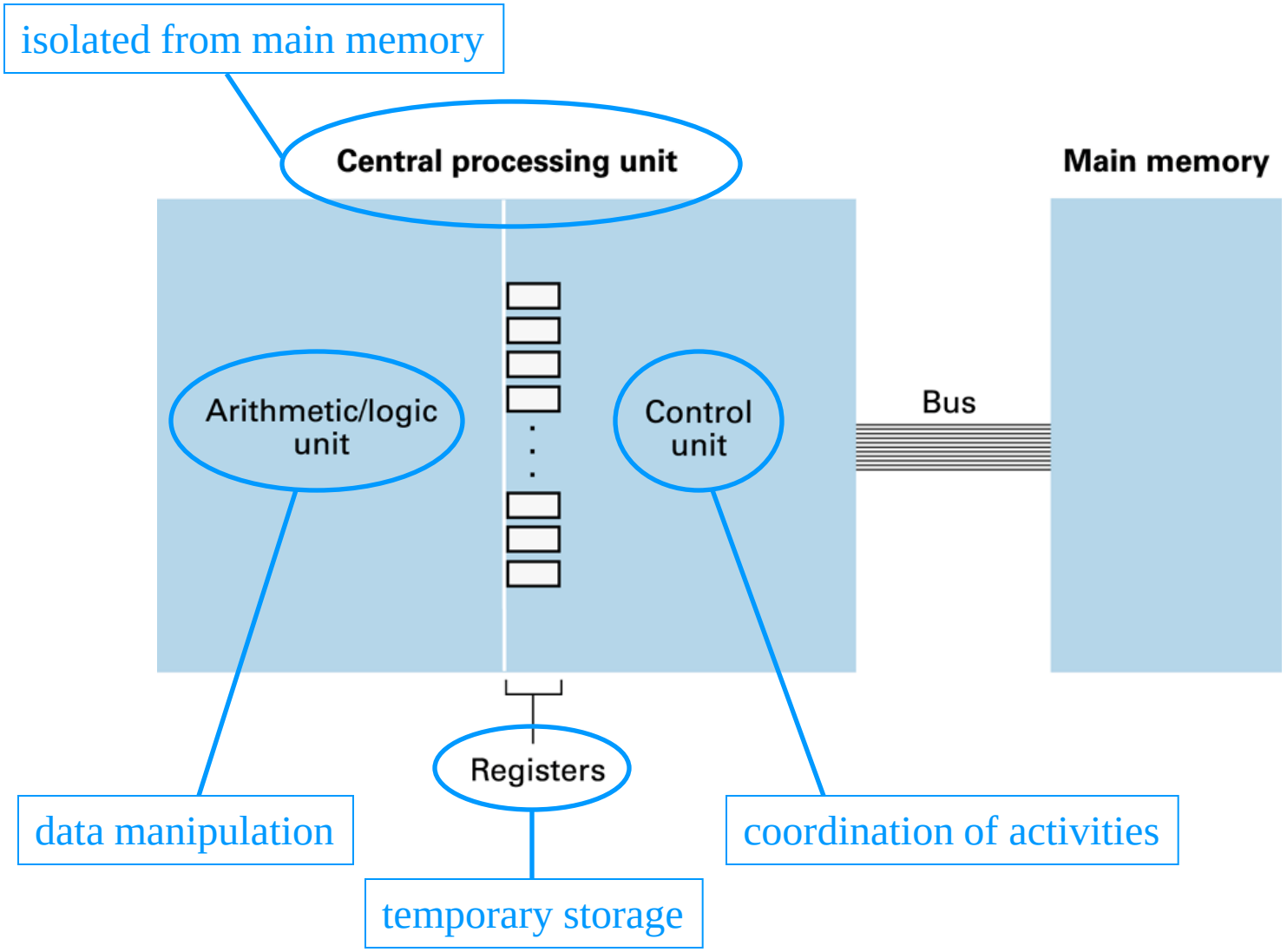
- For many important puzzles in computer science, the best way we know how to find an ideal solution is by brute force: listing all the possible solutions and then quickly checking each possible solution. (Maybe that's because there isn't any better way, but maybe that's because there is a better way and we just haven't thought of it yet! For most of the important problems involving brute force search, we don't know!)
- **Solving problems with brute force is an important tool in the computer science toolbox. However, brute force problem solving gets a lot harder when the solutions get only a little bit more complex.**

Data Manipulation



Central Processing Unit (CPU)

2.1: The 'Von Neumann Architecture': CPU and Main Memory connected via a Bus



2.1: Central role of the Control Unit

- To perform an operation on data stored in main memory, the control unit must
  - transfer the data from main memory into registers
  - inform the arithmetic/logic unit which registers hold the data
  - activate appropriate arithmetic/logic unit circuitry
  - tell the arithmetic/logic unit which register should receive the result
  - transfer the result from that register to main memory.

2.1: Flexibility of Execution

- Early computers were not flexible
  - operations were built into control unit
- Programs now encoded/stored in main memory
  - known as: *stored-program concept*
- As a consequence, the control unit must also
  - fetch (extract) the program from main memory
  - decode the set of instructions
  - execute each instruction in turn

Assignment: Real world Problems Solving by Using Mathematical Knowledge

Major Assignment :

- A student must use their knowledge of Mathematics to identify and solve real-world problem.

Rubrics of the Assignment:

- Understanding of the Real-world Practical Problem.
- Providing Solution to the Problem with simple example.
- Mapping to the Mathematical Knowledge /expression.
- Algorithm in plain English.

Note: Draw Diagrams, Figures or DFDs to enhance understanding of your assignment. Make it Self- Explanatory.


Any
Questions