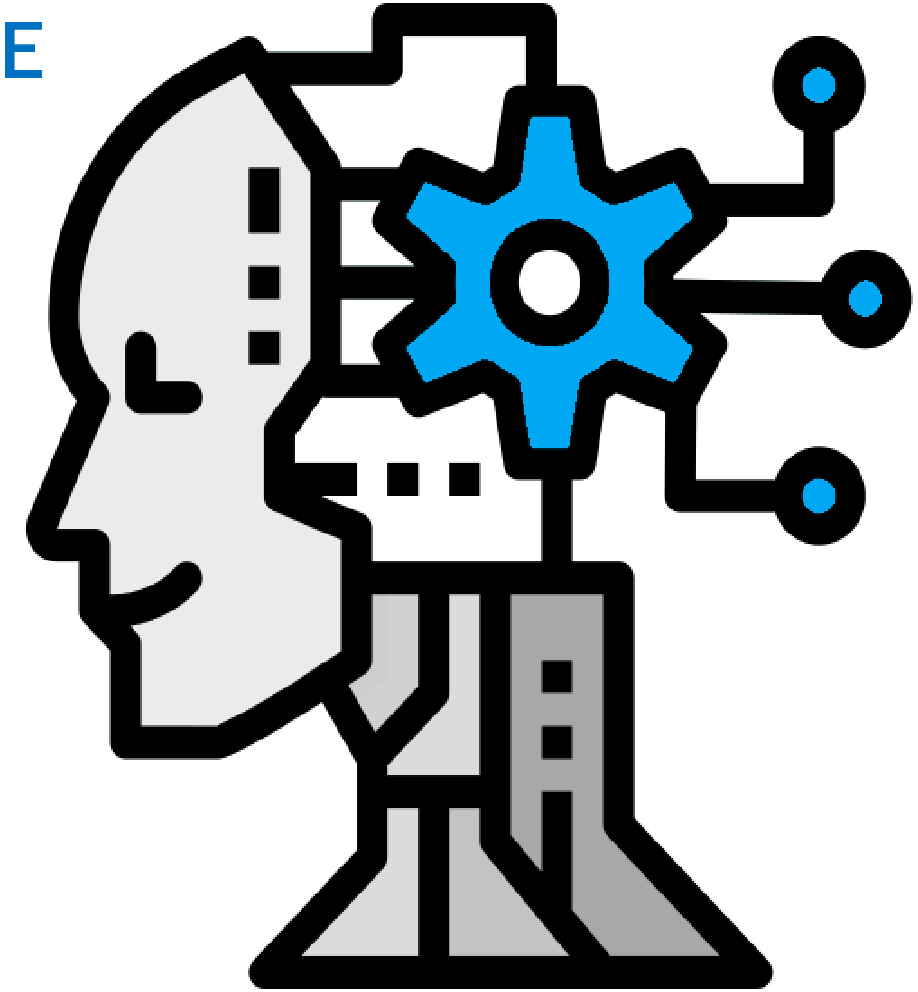
WELCOME TO COMPUTER SCIENCE

THE ADVENTURE BEGINS

DR. VICTOR MATOS

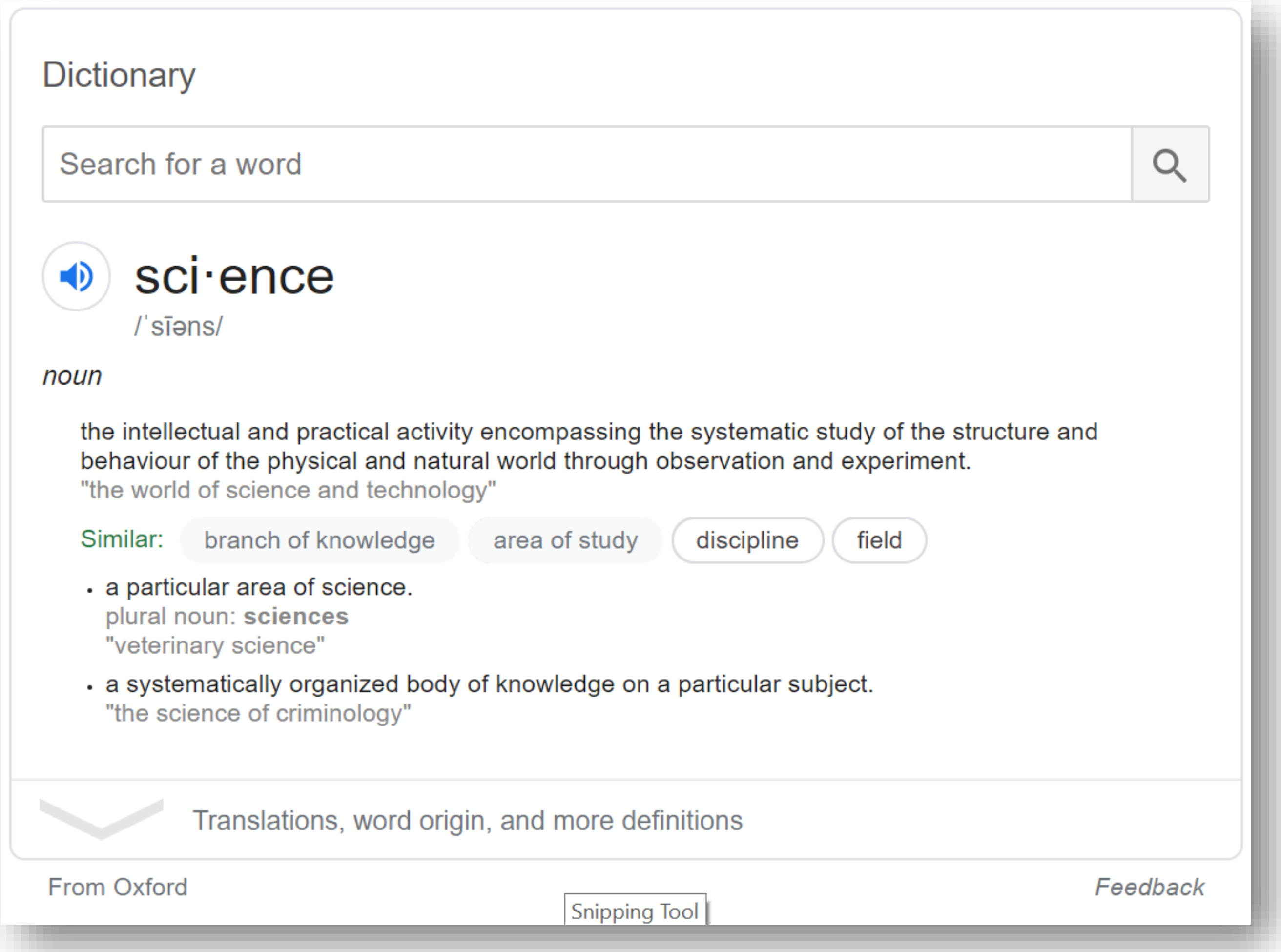


WHAT IS IN THIS TALK FOR YOU?

- A clear understanding of what Computer Science is really about.
- Insights into how computer scientists think and solve problems.
- An overview of job opportunities in 2025 and beyond.
- How El Camino College can help you begin your journey as a CS professional.

PREAMBLE

What is science?



Dictionary

Search for a word

sci·ence

/ˈsīəns/

noun

the intellectual and practical activity encompassing the systematic study of the structure and behaviour of the physical and natural world through observation and experiment.
"the world of science and technology"

Similar: branch of knowledge | area of study | discipline | field

- a particular area of science.
  plural noun: **sciences**
  "veterinary science"
- a systematically organized body of knowledge on a particular subject.
  "the science of criminology"

Translations, word origin, and more definitions

From Oxford

Feedback

PREAMBLE

What is Science?

Alternative viewpoint

According to the Austrian-British philosopher Karl Popper:

For a field to be considered a science, its claims must be **falsifiable** — that is, testable by experiment and potentially proven wrong.

- **Physics, chemistry, biology** → Their theories make predictions that can be confirmed or rejected through experiments.

- **Mathematics** → Works through logical deduction from axioms, not experiments. A mathematical theorem isn't tested in a lab; once proven, it is *necessarily* true under a given set of axioms.

PREAMBLE

Is Computer Science a science?

Computer Science Is a Hybrid Field

1. Theoretical Computer Science → Not a science under Popper

Parts of computer science behave like **mathematics**:

- Algorithms
- Computability theory
- Complexity theory
- Formal languages
- Logic and automata
- Cryptographic proofs

These areas rely on **formal reasoning, mathematical proofs**, and **axioms** — not experiments.

Example:

The correctness of Dijkstra's algorithm is proven mathematically,
not tested experimentally.
→ Therefore, **theoretical CS does not meet Popper's criterion.**
It is better classified as **a branch of mathematics.**

PREAMBLE

Is Computer Science a science?

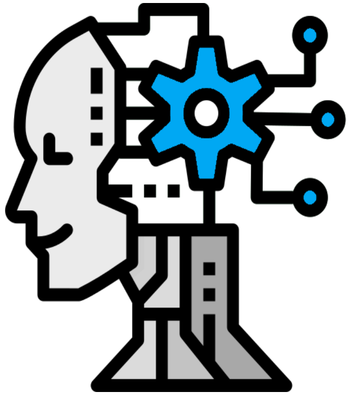
Computer Science Is a Hybrid Field

2. Applied / Empirical Computer Science →
Can be a science under Popper, meaning that some claims can be tested and falsified by observation:

- Computer architecture
- Operating systems
- Networking
- Distributed systems
- Human–computer interaction
- Performance engineering
- Machine learning
- Software engineering experiments

Example:
A hypothesis such as
“*Protocol A achieves lower latency than protocol B under condition X*”
can be experimentally tested and potentially shown false.

PREAMBLE

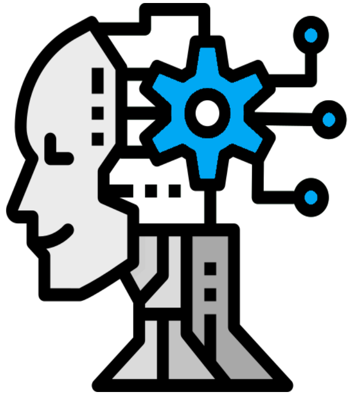


➢ What do computer scientists do?

- Computer scientists **design, develop, and analyze the theoretical foundations of information and computation.**

- CS professionals use this knowledge to **create algorithms, software, and systems** that enable technology to solve complex problems efficiently and correctly.

PROBLEM SOLVING (PROGRAMMERS)



- From a programmer's viewpoint, **problem solving**, is the ability to take (or create) a problem description and write an original computer code to solve it.

- Computer coding is easy! - Repetitive task!
- Problem solving is hard! - Creative activity

- Creative thinking is **mysterious**; nobody really knows how it works.

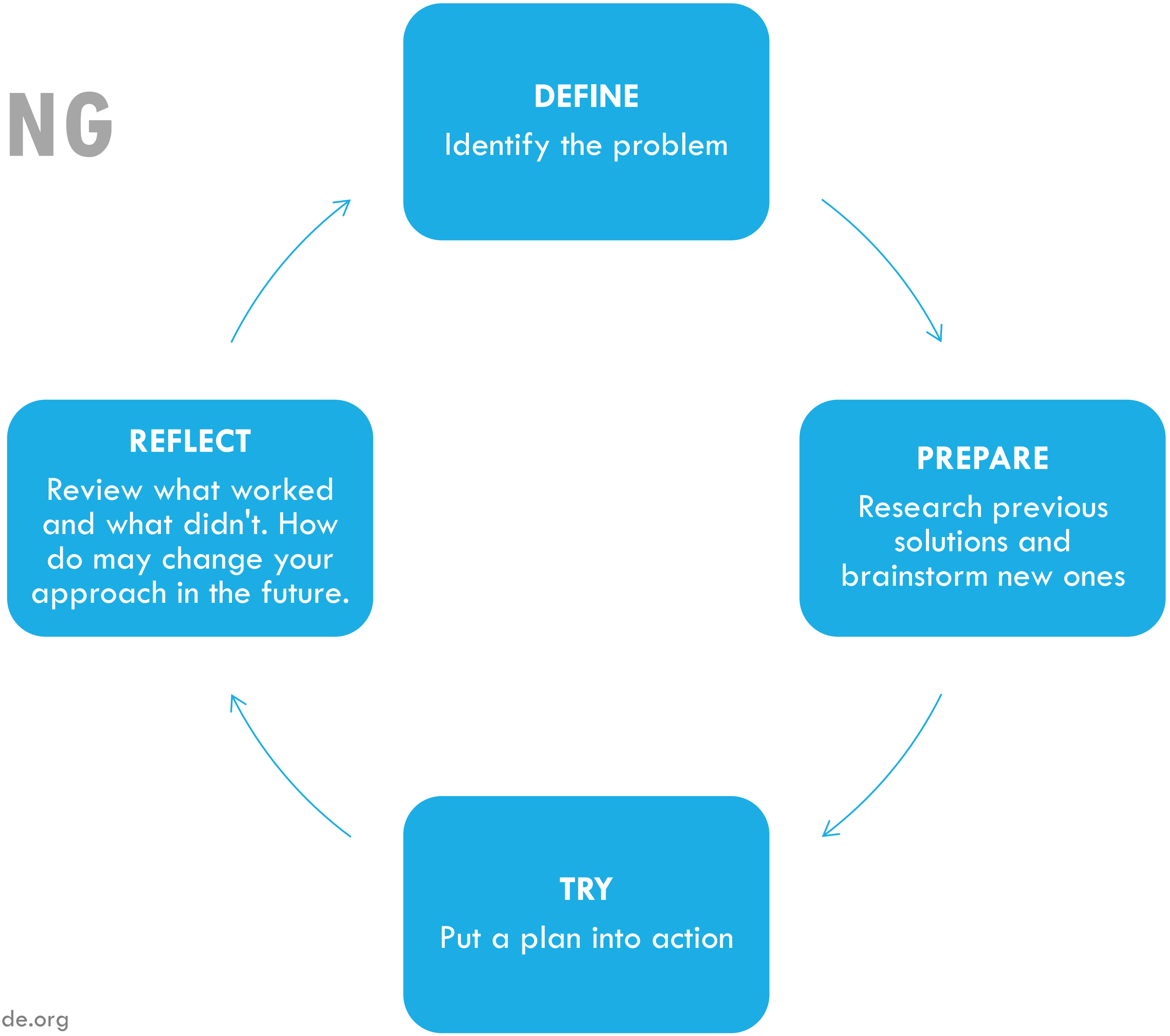
PROBLEM SOLVING - STRATEGIES

1. Brute Force
2. Divide and Conquer
3. Greedy Algorithms
4. Dynamic Programming
5. Backtracking
6. Depth-First Search (DFS)
7. Breadth-First Search (BFS)
8. Binary Search
9. Heuristic Algorithms
10. Branch and Bound
11. Randomized Algorithms
12. Graph Algorithms
13. Linear Programming
14. Network Flow Algorithms
15. String Matching Algorithms
16. Machine Learning Algorithms
17. **etc.**

PROBLEM SOLVING

Universal Framework

DPTR



DEFINE

Identify the problem

PREPARE

Research previous solutions and brainstorm new ones

TRY

Put a plan into action

REFLECT

Review what worked and what didn't. How do may change your approach in the future.

Resource. 4:02min The Problem-Solving Process with Zipline
https://www.youtube.com/watch?v=z7RaFPT3DTE&ab_channel=Code.org

COMPUTATIONAL THINKING

Is the process of formulating a problem,
finding a solution to the problem and
expressing it in such a way
that humans or machines can understand the solution.

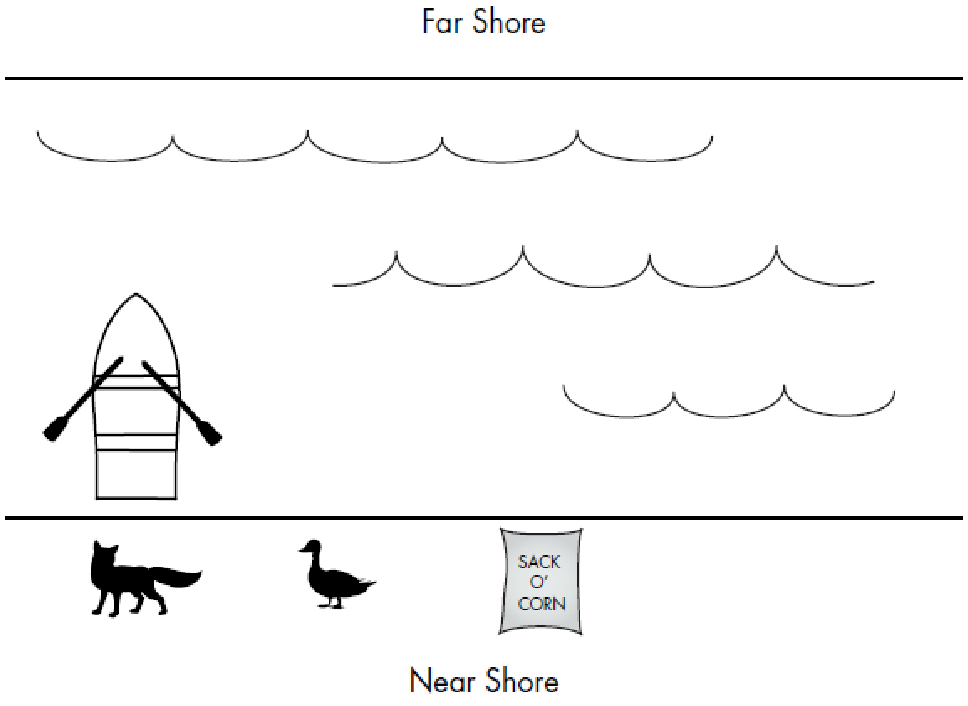
Observation
Problems include **constraints**, unbreakable rules about the problem or the way in which the problem must be solved.

CASE 1 – A CLASSIC PUZZLE (THE FOX, THE GOOSE, AND THE CORN BAG PROBLEM)

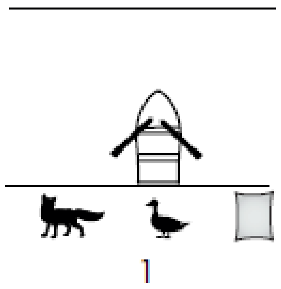
PROBLEM: HOW TO CROSS THE RIVER?

A farmer with a fox, a goose, and a sack of corn needs to cross a river.

- The farmer has a rowboat, but there is room for only the farmer and one of his three items.
- Unfortunately, both the fox and the goose are hungry.
  - The fox cannot be left alone with the goose, or the fox will eat the goose.
  - Likewise, the goose cannot be left alone with the sack of corn, or the goose will eat the corn.
- How does the farmer get everything across the river?



CASE 1 – A CLASSIC PUZZLE (THE FOX, THE GOOSE, AND THE CORN BAG PROBLEM)

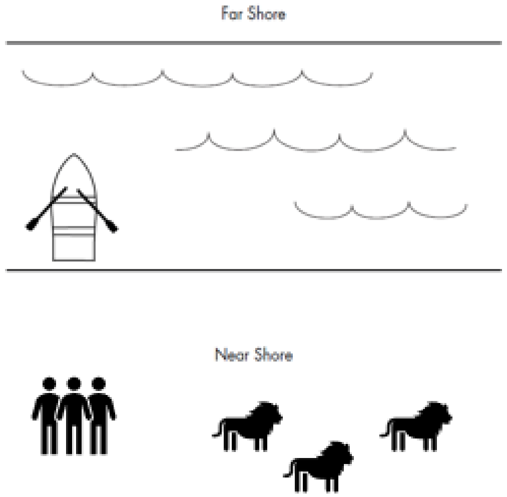


Step-by-step pictograms showing the solution of the fox, the goose, and the corn bag puzzle

CASE 1B – A VARIATION OF THE PREVIOUS PROBLEM BBBB

The three men and lion problem – How to transport men & lion to the other side of the river?

Constraint: At no time, on either side of the river, can the number of lions outnumber the number of men.



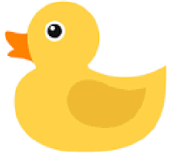
CASE 1 – A CLASSIC PUZZLE (THE FOX, THE GOOSE, AND THE CORN BAG PROBLEM)

Lessons Learned

What can we learn from the fox, the goose, and the corn puzzle?

- **Formally specifying** the problem is a great technique for gaining **deeper insight** and understanding its true constraints (consider **inputs, outputs, and edge cases).**

- **Articulating** the problem aloud to a peer — even if they offer no direct solution — often triggers new and useful thoughts (the *'rubber duck debugging' principle*).



- **Analyzing** the problem space is often as productive as immediately thinking about the solution.

- **Extrapolate**. Connect the current problem with a similar one whose solution is known.

CASE 2 – DYNAMIC GRAPH CONNECTIVITY

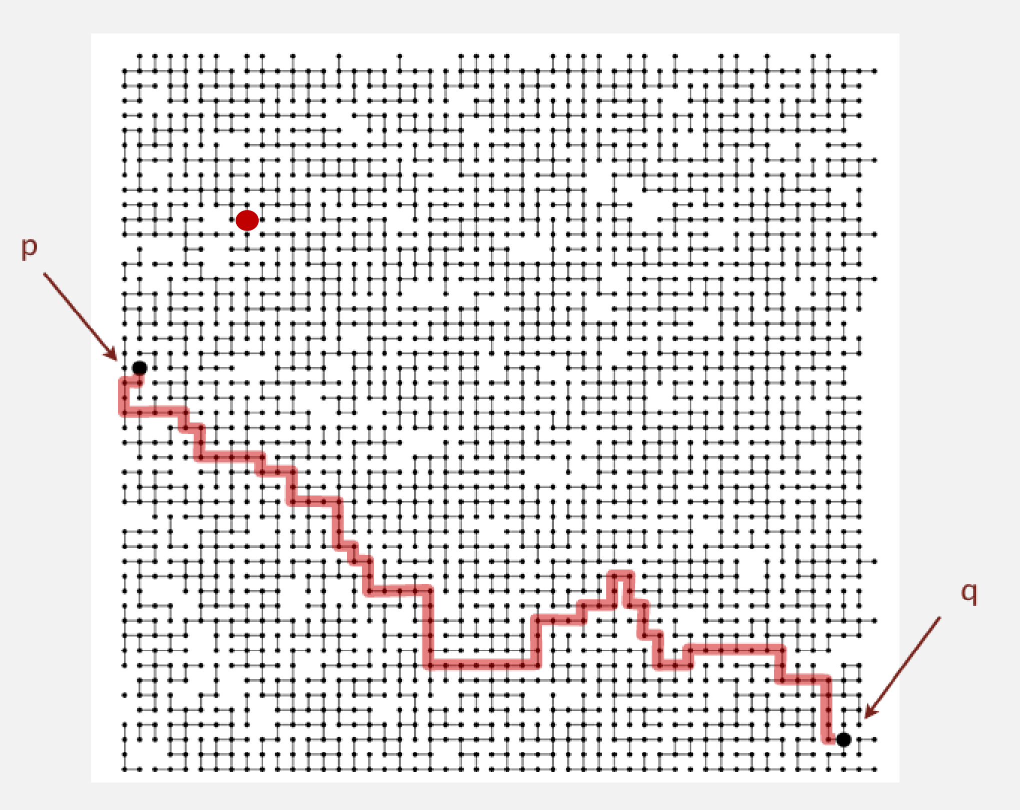
Questions.

- Are two graph points – say p and q- connected?

- Find shortest path between p and q.

Application Areas

- Pixels in a digital photo.
- Computers in a network.
- Oil pipeline grid.
- Friends in a social network.
- Transistors in a computer chip.
- Elements in a mathematical set.
- Metallic sites in a composite system.



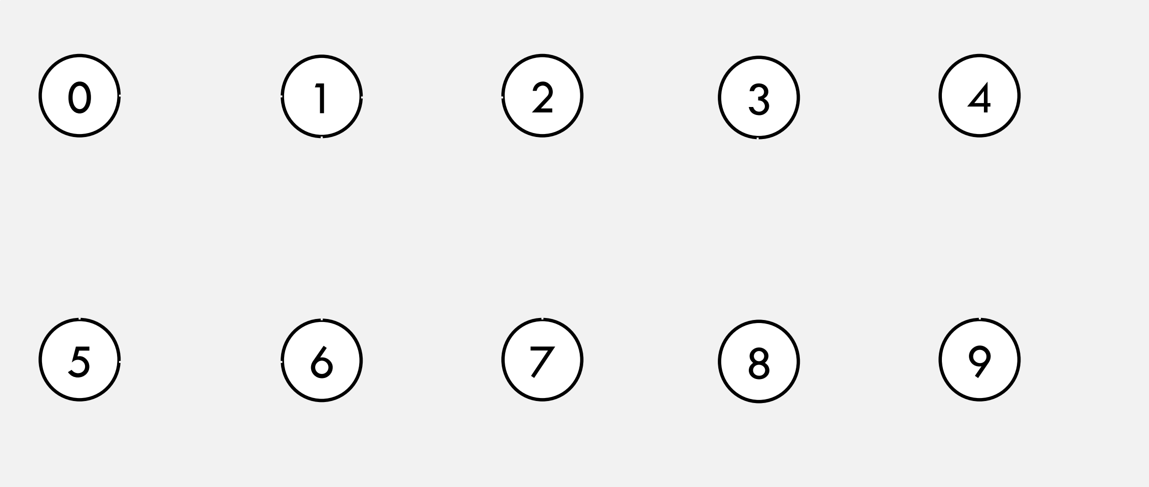
Source: Algorithms 4 Ed. By Robert Sedgewick and Kevin Wayne. Addison-Wesley

CASE 2 – GRAPH CONNECTIVITY

Dynamic connectivity problem

Given a set of N objects, support two operations

- Connect two objects.
- Is there a path connecting the two objects?



Connect 4 and 3

Connect 3 and 8

Connect 6 and 5

Connect 9 and 4

Connect 8 and 4

Connect 2 and 1

Are 0 and 7 connected?

Are 8 and 9 connected?

Connect 5 and 0

Connect 7 and 2

Connect 6 and 1

Connect 1 and 0

Are 0 and 7 connected?

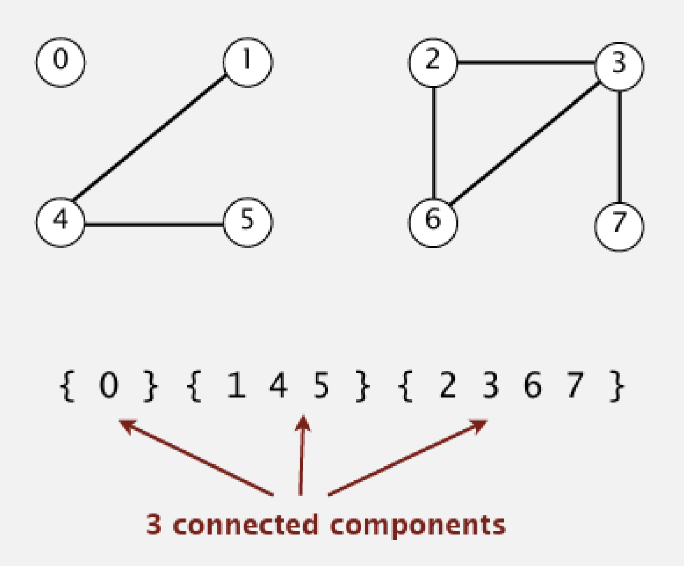
CASE 2 – GRAPH CONNECTIVITY

Implementing the graph operations

Find. In which component is object p found?

Connected. Are objects p and q in the same component?

Union. Replace components containing objects p and q with their union



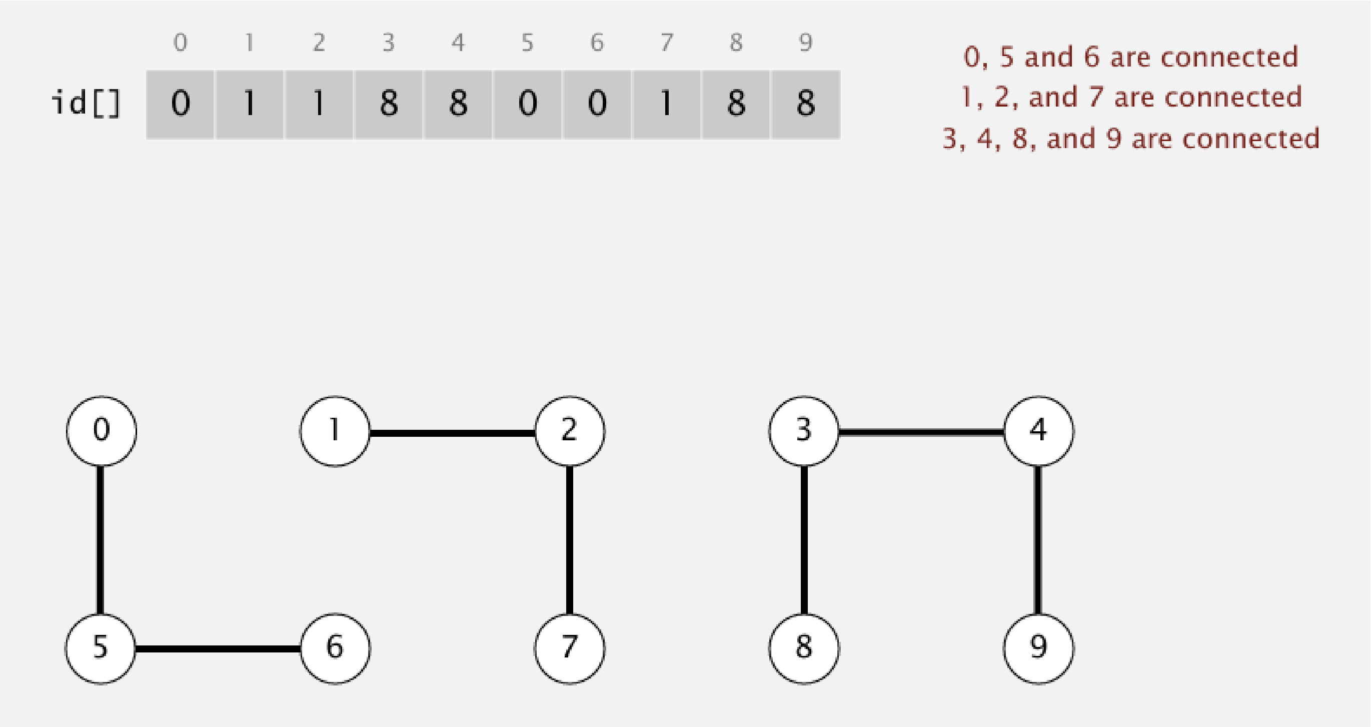
{ 0 } { 1 4 5 } { 2 3 6 7 }

3 connected components

CASE 2 – GRAPH CONNECTIVITY

Data structure.

- Integer array id[ ] of length N.
- Interpretation: id[p] is the id of the component containing p.



	0	1	2	3	4	5	6	7	8	9
id[]	0	1	1	8	8	0	0	1	8	8

0, 5 and 6 are connected
1, 2, and 7 are connected
3, 4, 8, and 9 are connected

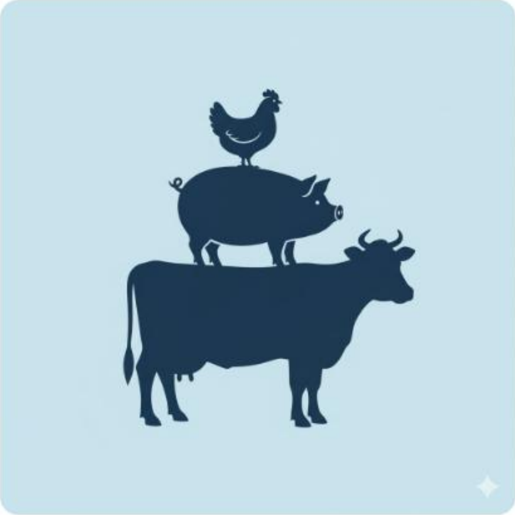
CASE 2 – GRAPH CONNECTIVITY

```cpp
class QuickFindUF {
private:
    vector<int> id;
public:
    QuickFindUF(int n) {
        id.resize(n);
        for (int i = 0; i < n; i++)
            id[i] = i;
    }

    bool connected(int p, int q) {
        return id[p] == id[q];
    }
```

```cpp

    void union(int p, int q) {
        if (connected(p, q)) return;
        int pid = id[p];
        int qid = id[q];
        for (int i = 0; i < id.size(); i++)
            if (id[i] == pid) { id[i] = qid; }
    }

    void print() {
        for (int x : id) cout << x << " ";
        cout << endl;
    }

};
```

CASE 3 – MATHEMATICAL REASONING THE COWS, THE PIGS, AND THE CHICKEN'S PROBLEM

- A farmer buys 100 animals for a total of $100.00
- He purchases only cows, pigs, and chickens, and he buys at least one of each.
- Cows cost $10.00 each, pigs cost $3.00 each, and chickens cost $0.50 each.
- How many of each type of animal did he buy?



CASE 3 – MATHEMATICAL REASONING
THE COWS, THE PIGS, AND THE CHICKEN'S PROBLEM

Brute-Force. List all possible animal combinations

Cow	Pig	Chicken	Feasible?
1 1 … 1	1 1 … 1	1 2 … 100	No No … No
1 … 1	2 … 2	1 … 100	No … No
…			
5 …	**1** …	**94** …	**Yes** . . .
100	100	100	No

Constraints

Cow + Pig + Chicken = 100

10 Cow + 3 Pig + .5 Chicken = 100

and cow > 0, pig >0, chicken > 0

Solution

5 cows, 1 pig, 94 chickens

CASE 4 – PATTERN RECOGNITION

If 8 = 56
7 = 42
6 = 30
5 = 20
3 = ?

CASE 4 – PATTERN RECOGNITION

If 8 = 56
7 = 42
6 = 30
5 = 20
4 = 12
3 = 6
2 = 2
1 = 0
0 = 0
-1 = 2
-2 = 6

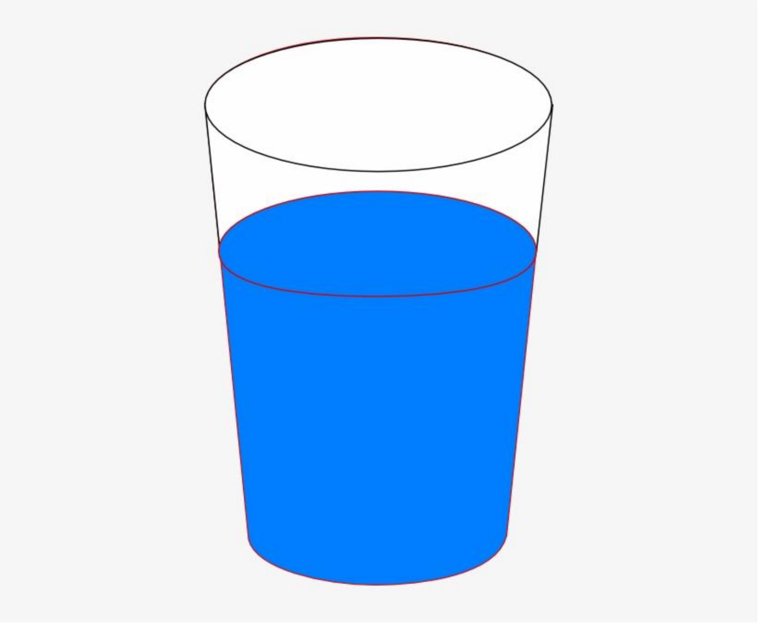
Generalizing

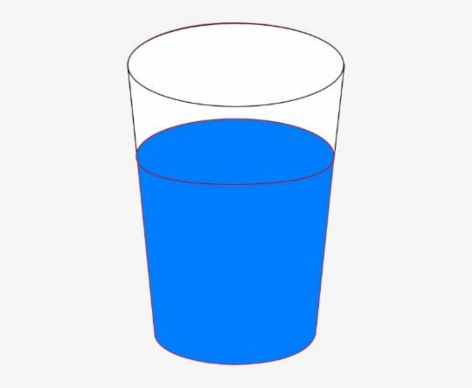
$$f(n) = n \times (n - 1)$$

CASE 5 – MEASURING CUPS

We have two graduated cups: the first has a capacity of 5 oz., and the second has a capacity of 3 oz.

How do we exactly measure 2 oz. of liquid?





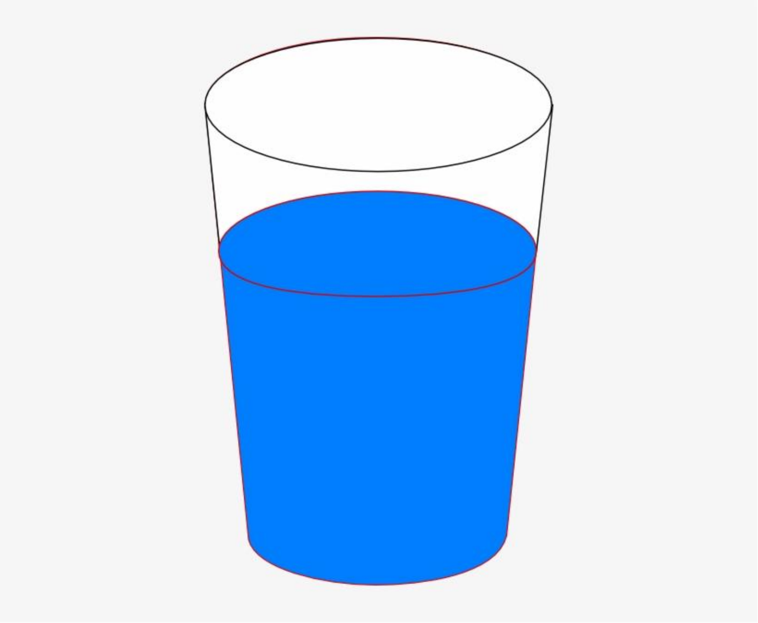
5 oz.

3 oz.

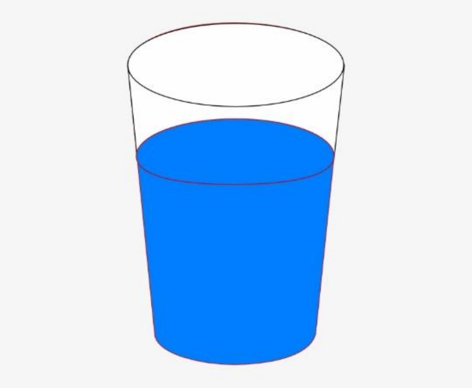
CASE 5 – MEASURING CUPS

Scheduling

1. Empty both containers.
2. Fill up larger container with 5oz. of fluid.
3. Transfer liquid from larger container to the smaller cup.
4. Repeat step 3 until small container is full (3oz.)
5. The larger container contains 2oz.



5 oz.



3 oz.

CASE 6 – SLIDING TILES PUZZLES

PROBLEM: THE SLIDING EIGHT

- A 3×3 grid is filled with eight tiles, numbered 1 through 8, and one empty space.
- At the start, the grid is arranged randomly.
- A tile can be slid into an adjacent empty space, leaving the tile's previous location empty.
- The goal is to slide the tiles to place the grid in an ordered configuration, from tile 1 in the upper left.

4	7	2
8	6	1
3	5	

Some starting configuration

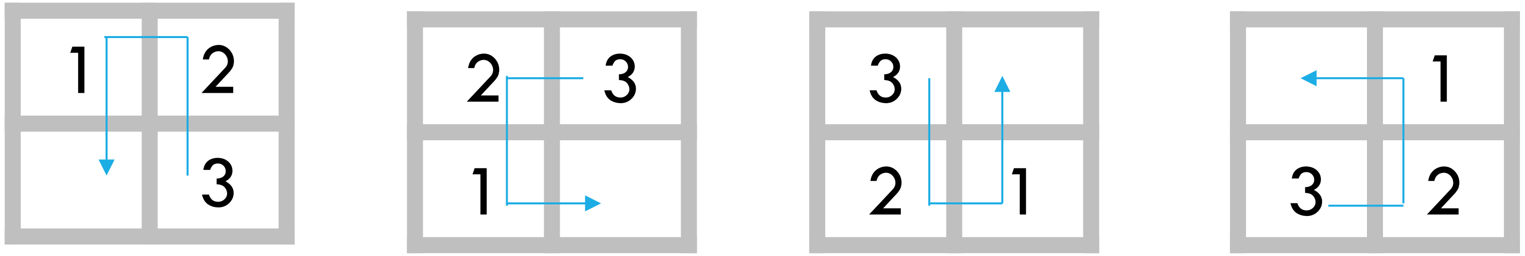
1	2	3
4	5	6
7	8	

Goal

CASE 6 – SLIDING TILES PUZZLES

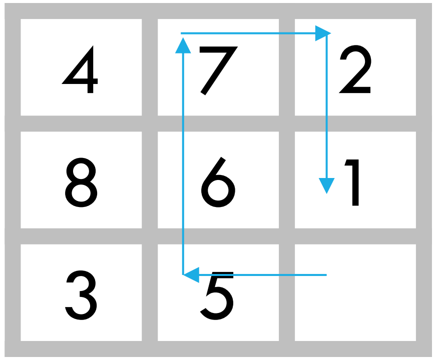
OBSERVATION

A circuit of tile positions that includes the empty space forms a **train** of tiles that can be rotated anywhere along the circuit while preserving the relative ordering of the tiles.



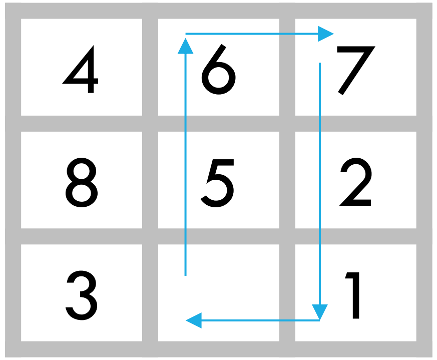
A “train,” is a path of tiles that begins adjacent to the empty square and can slide like a train of cars through the puzzle. In the example, 1 is the engine and 2 and 3 are the connected cars.

CASE 6 – SLIDING TILES PUZZLES



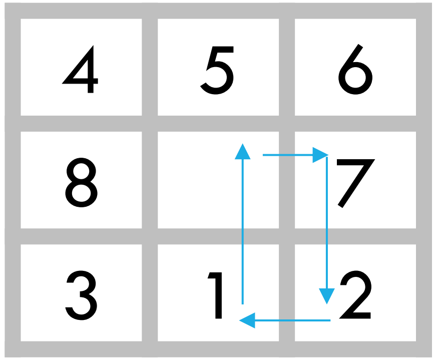
4	7	2
8	6	1
3	5	

Configuration 1



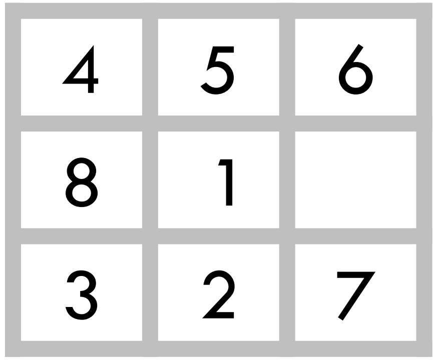
4	6	7
8	5	2
3		1

Configuration 2



4	5	6
8		7
3	1	2

Configuration 3

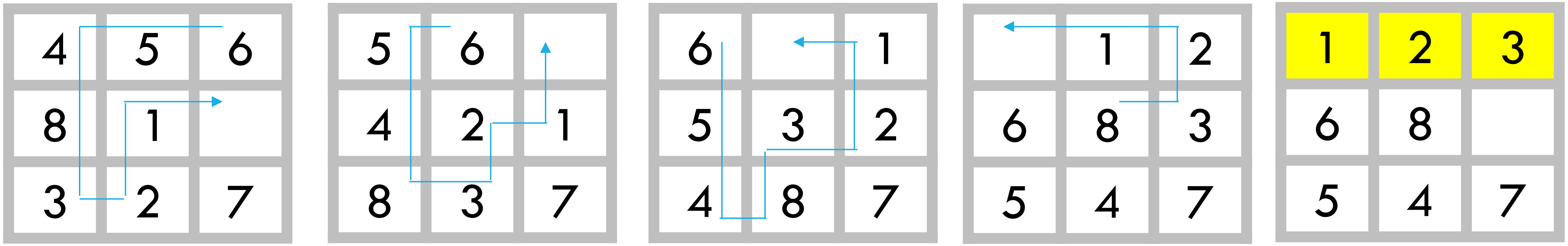


4	5	6
8	1	
3	2	7

Configuration 4

Step1. From configuration 1, rotate train (size 6) consisting of 1, 2, 7, 6, 5, Blank.
Step2. From configuration 2, rotate train (size 6) consisting of 1, 2, 7, 6, 5, Blank.
Step3. From configuration 3, rotate train (size 4) consisting of 1, 2, 7, Blank.
This produces Configuration 4. Observe that 1, 2, and 3 are close to each other.

CASE 6 – SLIDING TILES PUZZLES



4	5	6
8	1	
3	2	7

5

5	6	
4	2	1
8	3	7

6

6		1
5	3	2
4	8	7

7

	1	2
6	8	3
5	4	7

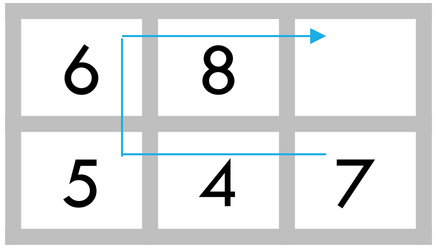
8

1	2	3
6	8	
5	4	7

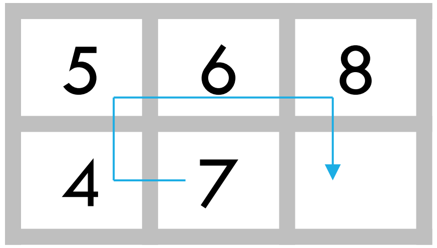
9

- From configuration 5, tiles are rotated to reach configuration 9, in which tiles 1, 2, and 3 are in their correct final positions (could be ignored!)
- Now we focus on the lower two 3x2 row, see next figure.

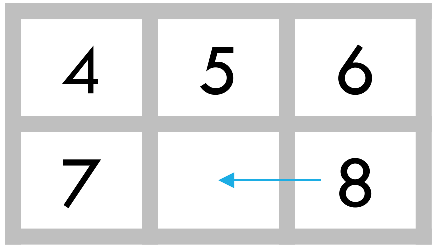
CASE 6 – SLIDING TILES PUZZLES



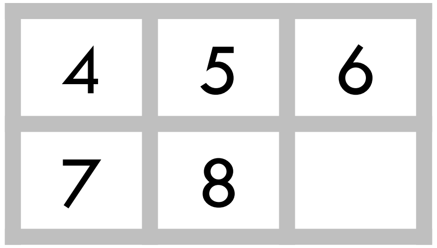
9



10



11



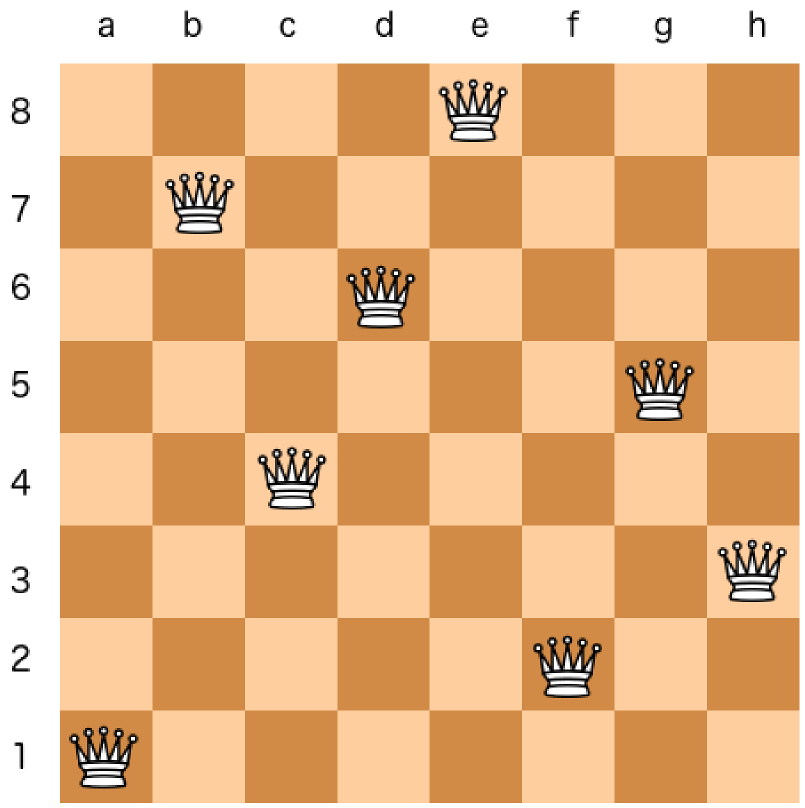
12

- From configuration 9, two rotations along the outlined “train” bring us to configuration 11. From there, a single tile slide results in the goal, configuration 12.
- Remember that the top row is already in the correct order.
- Therefore, the puzzle is solved.

CASE 7 – EIGHT QUEENS' PROBLEM

Problem:

Place eight queens on an 8x8 chess board so that the queens do not attack each other.



CASE 7 – THE EIGHT QUEENS' PROBLEM

Lessons Learned

- The main lesson is that you should start with the part that's obvious.
  If you can solve part of the problem, go ahead and do what you can.

  Backtracking. If a decision leads to an invalid/unsuccessful configuration, abandon it, go back and change the previous decision selecting another alternative. Retry the new path of possibilities

CASE 8 – A MANUFACTURING PROBLEM

A company manufactures three different electronic components for computers.

Constraints

- Component A takes two hours of fabrication and one hour of assembly
- Component B takes three hours of fabrication and one hour of assembly, and
- Component C takes two hours of fabrication and two hours of assembly.
- Suppose the company has up to a 1000 labor hours to spend on fabrication and 800 labor hours to spend on assembly each week.
- Suppose we know the profit from each component A, B and C is $7, $8, and $10 respectively.

Goal

We want to figure out how many of each part should be produced to maximize the profit.

CASE 8 – A MANUFACTURING PROBLEM

Reuse. This is a Linear Programming problem, apply the **SIMPLEX** Method.

Maximize $7a + 8b + 10c$ (Profit)

Subject to

$$\begin{cases} 2a + 3b + 2c \le 1000 & \text{(Fabrication)} \\ a + b + 2c \le 800 & \text{(Assembly)} \\ a, b, c \ge 0 & \text{(Var. constraints)} \end{cases}$$

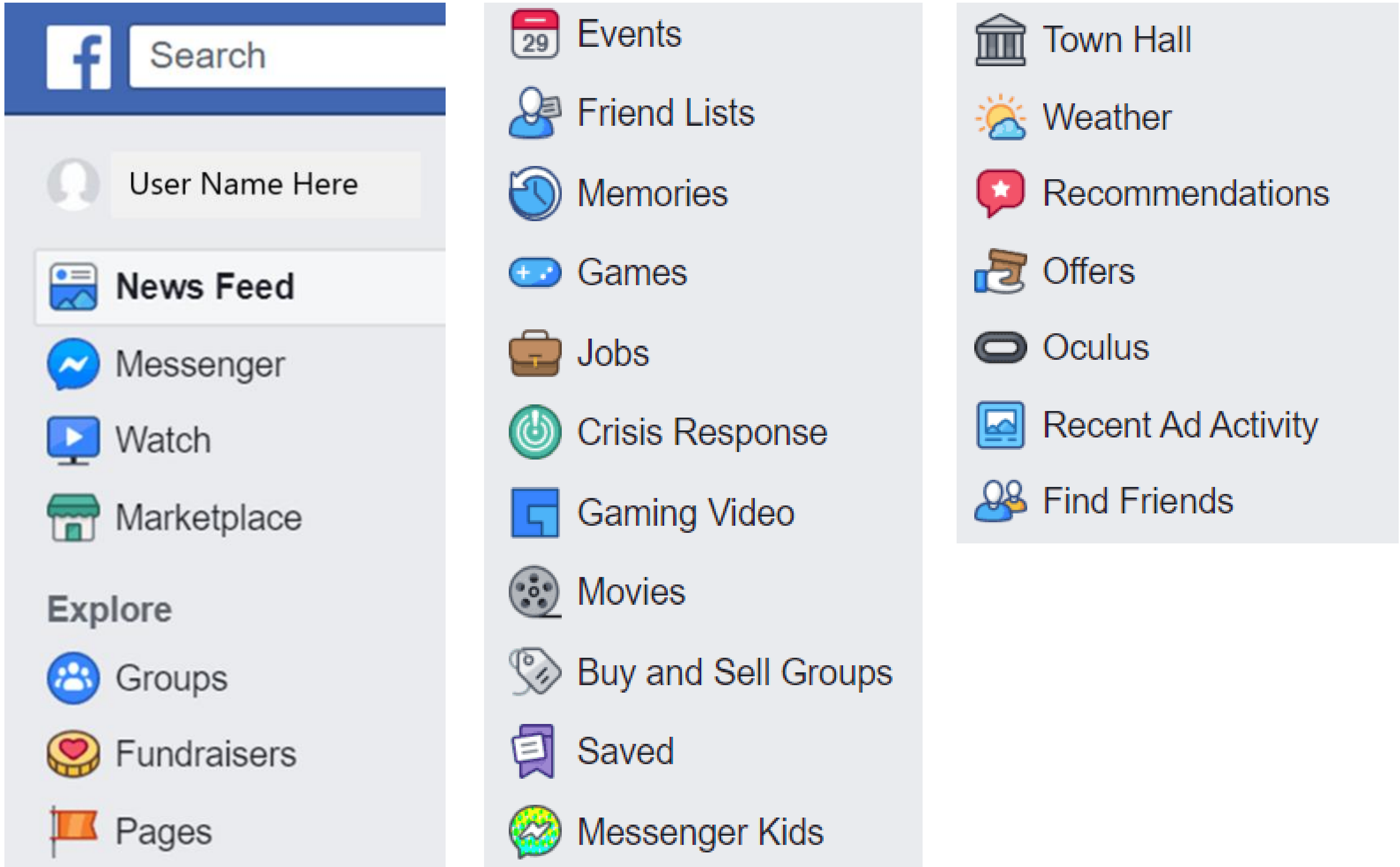
Solution:
$a = 200$, $b = 0$, $c = 300$.
Profit is $4400

Lesson learned.
Do your research. Reapply existing good algorithms!

CASE 9 – FACEBOOK ARCHITECTURE

Problem.

Based on your personal experience, apply a *Reverse* ***Engineering*** **process** to understand, then visualize, the data architecture of a large social network such as Facebook.



CASE 9 – FACEBOOK ARCHITECTURE

Brainstorming. Group discussion to produce ideas or solve problems.

User (Basic Data)

User

ID, Name, DOB, EMAIL, Friends, Common Friends, Subscriptions, Friend Requests

Posts

ID, Time, User Id, Text, Description, Content_url

Notifications

Id, Time, Message, From User, To User, Entity type, Entity Id, Status

Likes

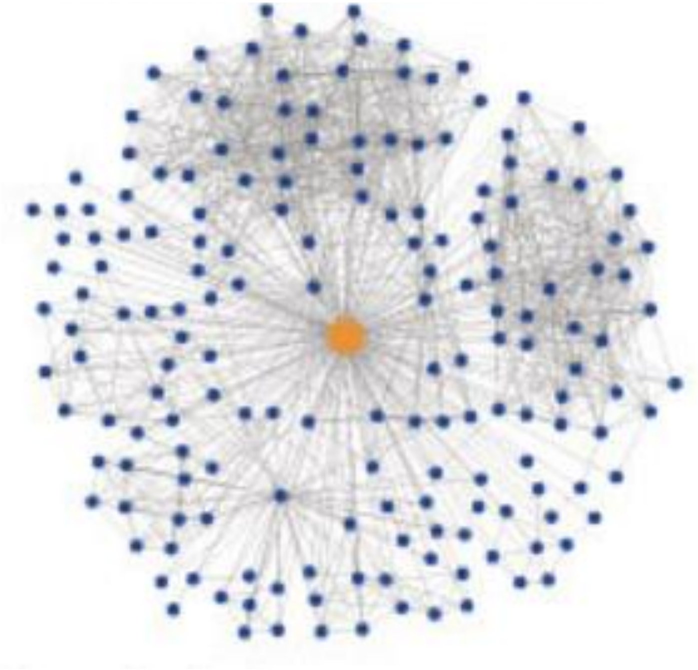
ID, Entity Type, Entity Id

Comments

ID, Entity type, Entity id

…

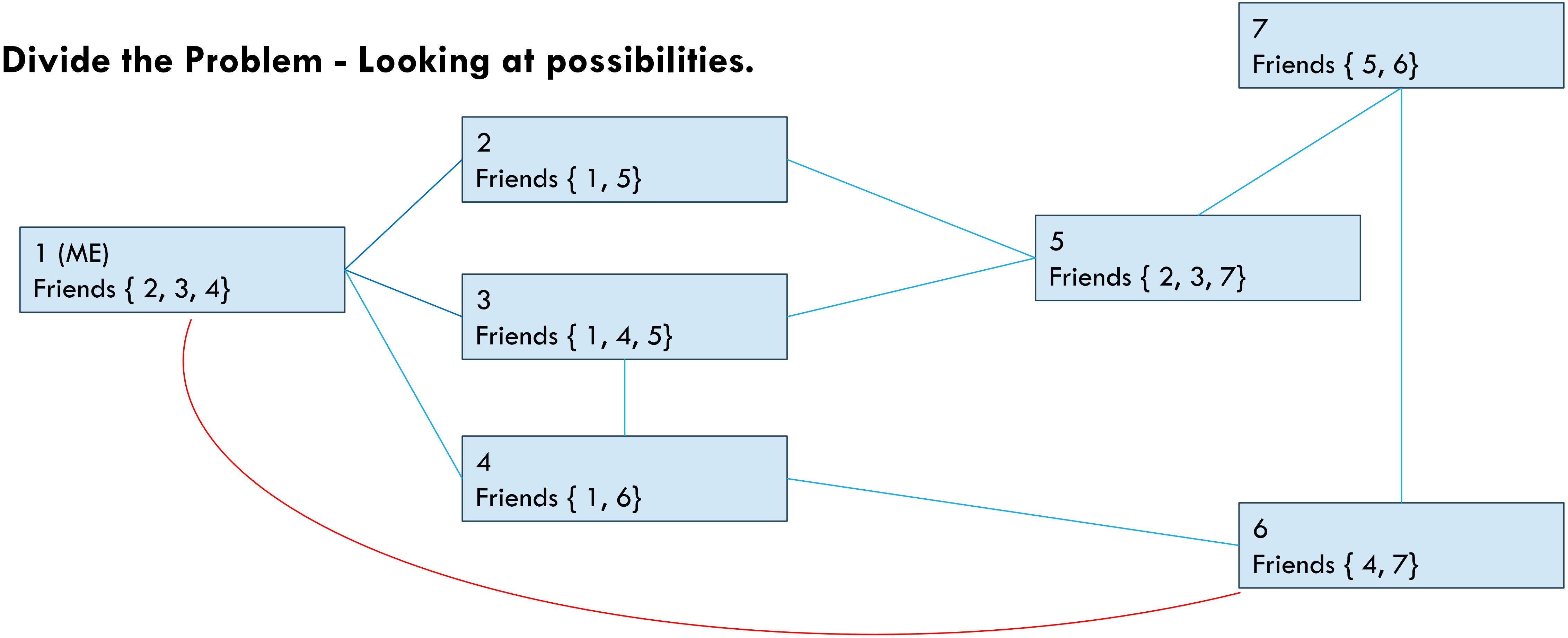
Total network: 179 friends



Source: Facebook.com

CASE 9 – FACEBOOK ARCHITECTURE

Divide the Problem - Looking at possibilities.



1 (ME)
Friends { 2, 3, 4}

2
Friends { 1, 5}

3
Friends { 1, 4, 5}

4
Friends { 1, 6}

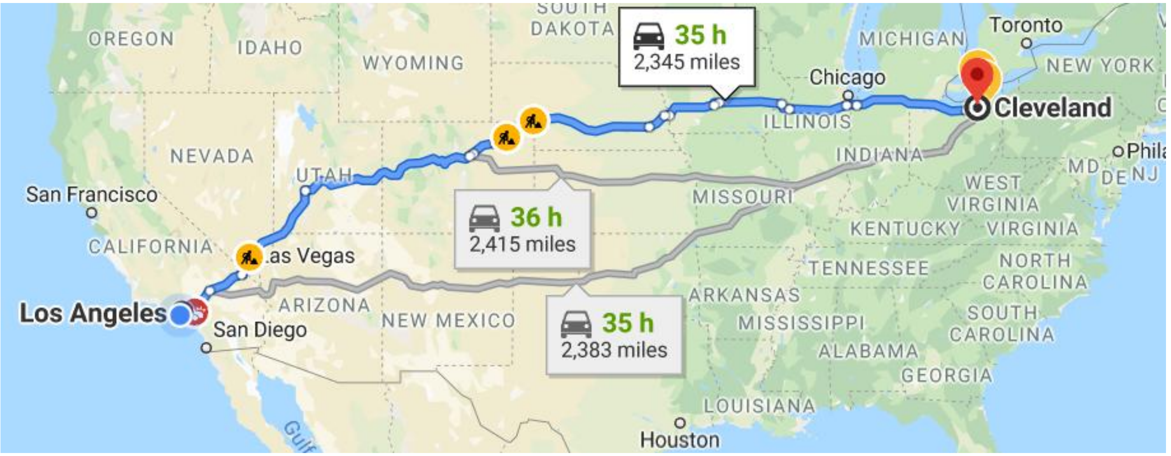
5
Friends { 2, 3, 7}

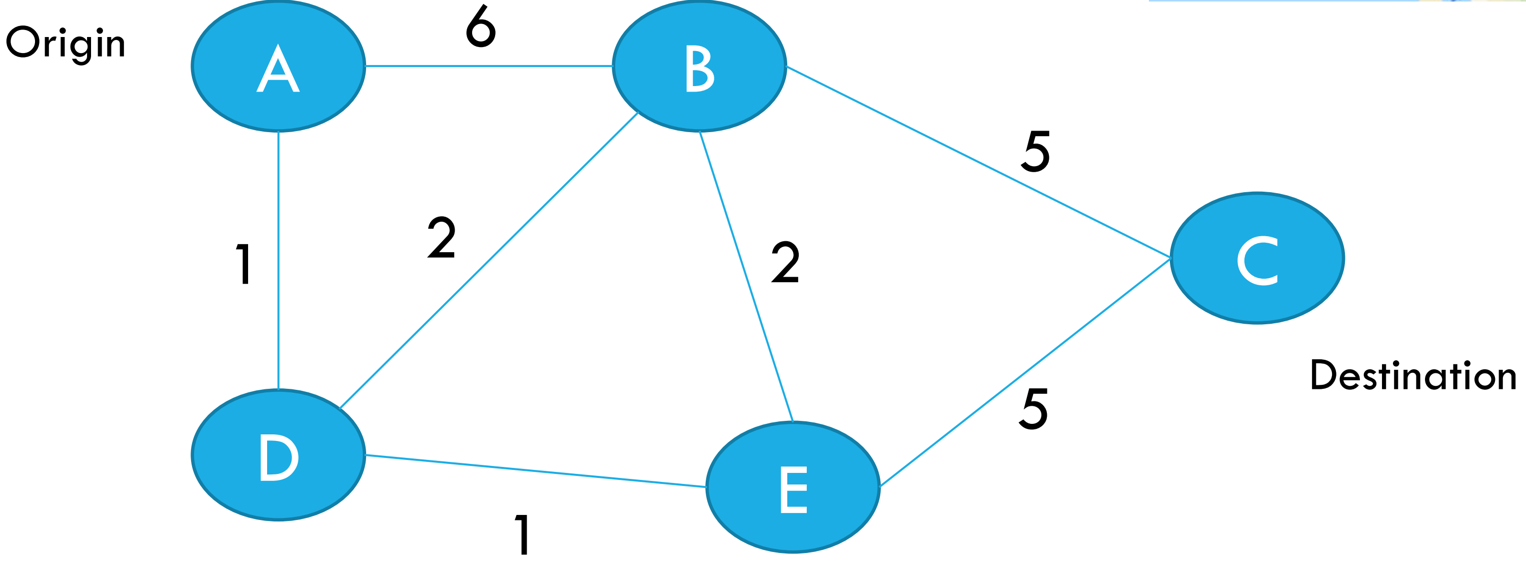
6
Friends { 4, 7}

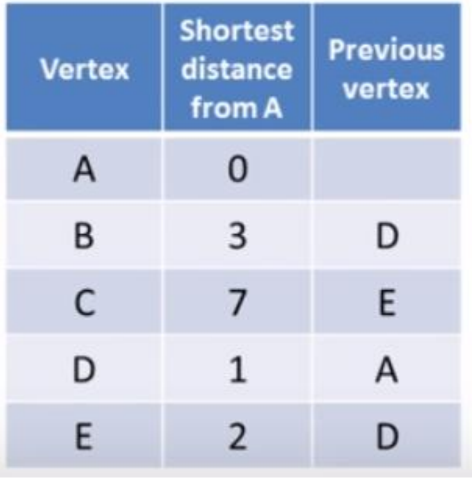
7
Friends { 5, 6}

CASE 10 – SHORTEST PATH

Find the shortest/fastest way to drive from LA to Cleveland, OH







Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

CASE 10 – SHORTEST PATH - DIJKSTRA'S ALGORITHM

ALGORITHM

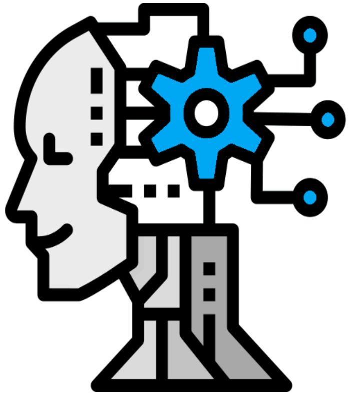
```
Let distance of start vertex from start vertex = 0
Let distance of all other vertices from start node be ∞ (infinity)
Repeat
    1. Visit the unvisited vertex with the smallest known distance from start vertex
    2. For the current vertex, examine its unvisited neighbors
    3. For the current vertex, calculate distance of each neighbor from start vertex
    4. If the calculated distance of a vertex is less than the known distance,
       update the shortest distance
    5. Update the previous vertex for each of the updated distances
    6. Add the current vertex to the list of visited vertices
Until all vertices visited
```

Vertex	Shortest Distance From A	Previous Vertex
A	0	
B	∞	
C	∞	
D	∞	
E	∞	

SUMMARY: PROBLEM SOLVING

- **Be methodical** — follow a structured approach.
- **Be curious** — ask questions and explore alternatives.
- **Be collaborative** — learn from others and share ideas.
- **Be patient** — good solutions take time.
- **Be bold** — try approaches that might fail.
- **Be persistent** — refine, debug, and keep going until it works.

WHAT THE 2025 JOB MARKET LOOKS LIKE FOR CS GRADUATES



a) Increased Competition & Entry-Level Crunch

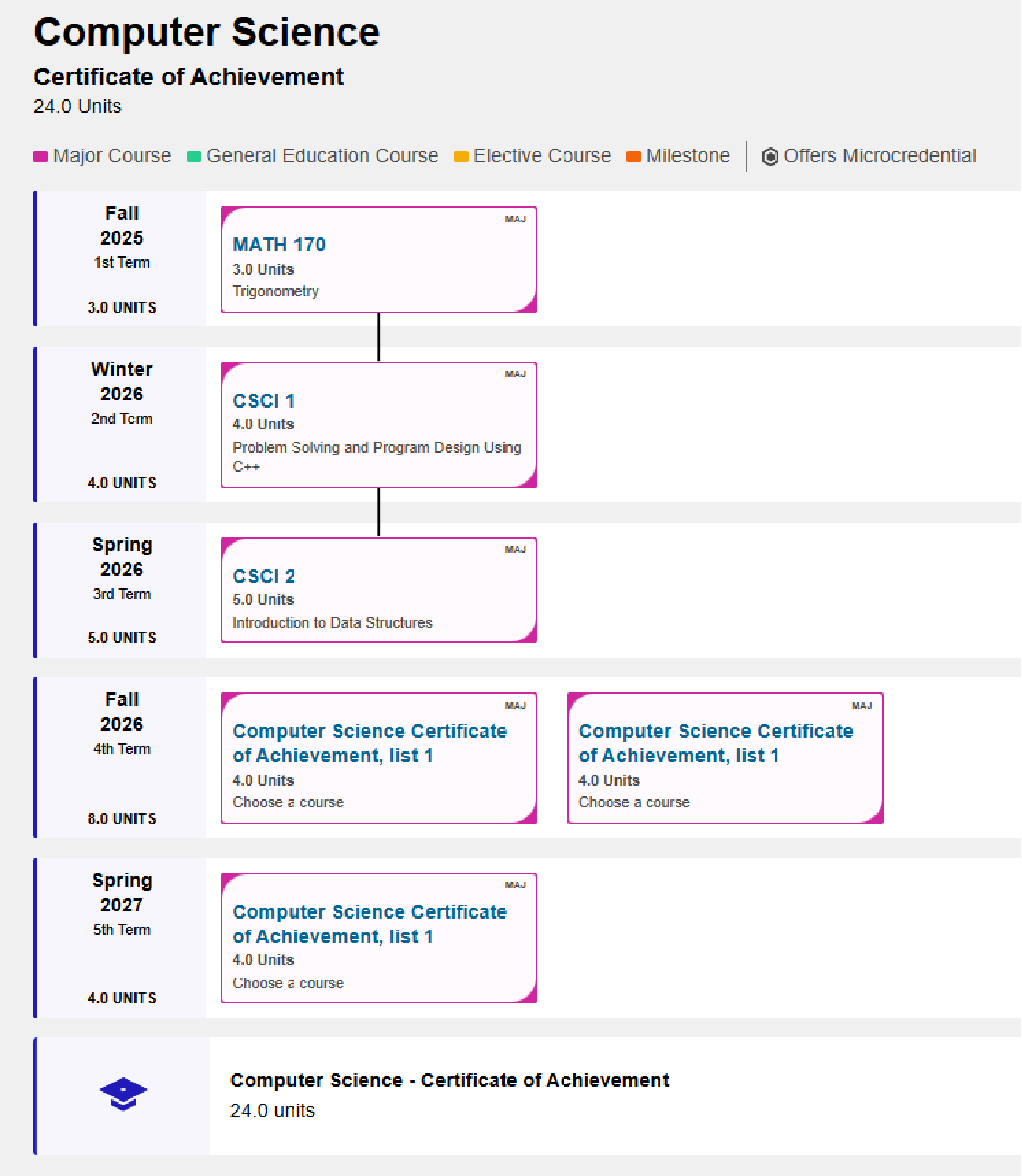
- According to recent data, new CS grads are facing a **tougher job market**.
- Students are submitting *more applications* than before.
- There's also concern about a *gap* between what's taught at universities and what employers need.
- Many instructors feel that students are *not sufficiently prepared* for the real-world roles that are currently open.

b) Salary Projections Still Look Pretty Solid

- Salary outlooks for CS grads remain relatively strong (projected salary **$76,251** on average in the U.S.)
- Companies value CS degrees, especially for more technical or in-demand roles (enterprise systems, data science, AI enabling).

c) Shift in Which Jobs Are Available

- Traditional big-tech companies are not necessarily hiring entry-level developers like before.
- **Some sectors** (finance, retail, energy, health management) are looking for CS talent, especially to build AI-enabled systems.
- Employers are increasingly demanding **specialized skills**: *not just general programming.*



Computer Science

Certificate of Achievement

24.0 Units

🟪 Major Course 🟩 General Education Course 🟨 Elective Course 🟧 Milestone | ⬢ Offers Microcredential

Fall 2025
1st Term
3.0 UNITS

MAJ
MATH 170
3.0 Units
Trigonometry

Winter 2026
2nd Term
4.0 UNITS

MAJ
CSCI 1
4.0 Units
Problem Solving and Program Design Using C++

Spring 2026
3rd Term
5.0 UNITS

MAJ
CSCI 2
5.0 Units
Introduction to Data Structures

Fall 2026
4th Term
8.0 UNITS

MAJ
Computer Science Certificate of Achievement, list 1
4.0 Units
Choose a course

MAJ
Computer Science Certificate of Achievement, list 1
4.0 Units
Choose a course

Spring 2027
5th Term
4.0 UNITS

MAJ
Computer Science Certificate of Achievement, list 1
4.0 Units
Choose a course

Computer Science - Certificate of Achievement
24.0 units

Computer Science – Certificate of Achievement

Required Core: 9 units

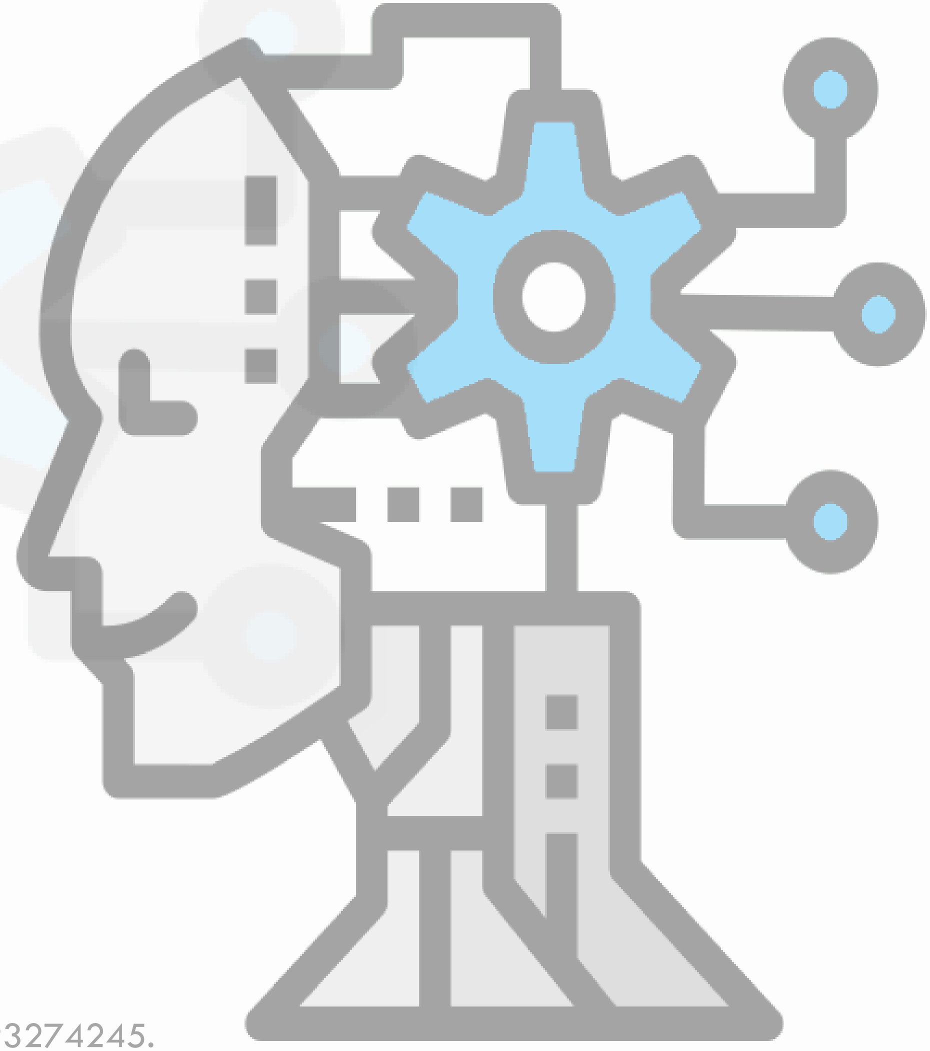
CSCI 1 - Problem Solving and Program Design Using C++
CSCI 2 - Introduction to Data Structures

Elective Courses: 12 units

CSCI 3 - Object-Oriented Programming in Java
CSCI 7 - The Beauty of Computer Science Principles
CSCI 8 - Foundations of Data Science
CSCI 12 - Programming for Internet Applications
CSCI 14 - Introduction to Programming with Python
CSCI 16 - Assembly Language Programming for x86 Processors
CSCI 30 - Advanced Programming in C++
CSCI 40 - Introduction to UNIX and LINUX Operating Systems
MATH 210 - Introduction to Discrete Structures

QUESTIONS?

Thanks for attending this talk!



References

- **Think Like a Programmer**, by V. Anton Spraul, No-Starch Press (2012), ISBN-13: 978-1593274245.
- **Algorithms** 4 Ed. By Robert Sedgewick and Kevin Wayne. Addison-Wesley.
- **Are You Smart Enough to Work at Google?** By William Poundstone. ISBN 9780316099981(2012)
- Fox5News. Lag in math preparedness at UCSD. By Chris Ponce (November 2025)
- Icons made by www.flaticon.com/authors/photo3idea-studio" title="photo3idea_studio"

DR. VICTOR MATOS

Dr. Victor Matos is a computer scientist specializing in database systems, programming languages, mobile application development, and excellence in computer science education.

He earned a Ph.D. in Computer Science from Case Western Reserve University, an M.S. in Computer Science, and a B.Sc. in Computing Engineering from Universidad Simón Bolívar.

He currently serves as Professor of Computer Sciences in the Division of Mathematical Sciences at El Camino College. Previously, he spent over three decades at Cleveland State University as Professor of Business Information Systems and Engineering, where he now holds Emeritus status.

His career also includes service as a Fulbright Scholar at the Universidad de Costa Rica, CIO at Utopia Technologies LLC, independent consultant, senior systems analyst for the Venezuelan Army, and academic roles at multiple institutions.

For a list of publications, search **"Victor Matos"** on Google Scholar.

EL CAMINO COLLEGE

CASE WESTERN RESERVE UNIVERSITY 1826

CSU Cleveland State University

Universidad Simon Bolivar

UNIVERSIDAD DE COSTA RICA

FULBRIGHT