P: Problems
P: Programming
P: Programming Language
P: Programming Contest

Problems

The motivation of human creativity.

- Mother of every new idea.
- Human asks problems.
- And human finds solutions.

The history of human civilization is the history of problems.

- How will society handle the massive wave of unemployment as AI and automation take over human jobs?
- How does the brain create thoughts and feelings? Is brain and mind different?
- What is the secret behind the pattern of prime numbers, and can we ever fully understand it?
- How can a democracy stay strong when people are divided and trust is fading?
- How can we reduce the gap between the rich and the poor without slowing progress?
- Can we color any map using no more than four colors so that no two neighboring regions have the same color?
- $2^{136,279,841} - 1$ is a prime number?

1 + 2 + 3 + … + 100 = ?

A mathematician's solution

Using the Gauss's relations,

$$S = \frac{n(n+1)}{2}$$

Here n = 100, therefore,

$$S = \frac{100.101}{2} = 5050$$

A C Scientist's approach

```
S = 0
for i in range(101):
        S = S + i
print(S)
```

Find the roots of
$x^2 - 5x + 6 = 0.$

Mathematical approach

For **quadratic** ($ax^2 + bx + c = 0$), we have the **quadratic formula**:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Set the values for a, b, c and calculate.
Here, a = 1, b = -5, c = 6
x = 2, 3

Computational Approach I

Just Use the quadratic formula and code it! Easy-peasy!

```python
import math

# Coefficients
a = 1
b = -5
c = 6

# Calculate the discriminant
discriminant = b**2 - 4*a*c

# Check if the discriminant is non-negative
if discriminant >= 0:
    # Two real roots
    root1 = (-b + math.sqrt(discriminant)) / (2 * a)
    root2 = (-b - math.sqrt(discriminant)) / (2 * a)

    print(f"The solutions are: x1 = {root1} and x2 = {root2}")
else:
    # Complex roots
    realPart = -b / (2 * a)
    imaginaryPart = math.sqrt(-discriminant) / (2 * a)

    print(f"x1 = {realPart} + {imaginaryPart}i")
    print(f"x2 = {realPart} - {imaginaryPart}i")
```

Computational Approach II

Use Newton-Raphson method
The Newton-Raphson method for solving an equation f(x)=0 is given by the iterative/loop update formula:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Steps:
1.Define the function f(x) and its derivative f′(x).
2.Choose an initial guess for x_n .
3.Use the Newton-Raphson update rule to iteratively refine the guess for the root.

For our problem, set
$f(x) = x^2 - 5x + 6$ and $f'(x_n) = 2x - 5$

```python
# Define the quadratic function and its derivative
def f(x, a, b, c):
    return a*x**2 + b*x + c

def f_prime(x, a, b):
    return 2*a*x + b

# Newton-Raphson method to find roots of the quadratic equation
def newton_raphson(a, b, c, initial_guess, max_iterations=100, tolerance=1e-7):
    x_n = initial_guess
    for _ in range(max_iterations):
        fx_n = f(x_n, a, b, c)
        f_prime_x_n = f_prime(x_n, a, b)

        if f_prime_x_n == 0:  # Avoid division by zero
            print("Derivative is zero. Newton-Raphson method fails.")
            return None

        x_n1 = x_n - fx_n / f_prime_x_n

        # If the solution is within tolerance, stop
        if abs(x_n1 - x_n) < tolerance:
            return x_n1

        x_n = x_n1

    print("Maximum iterations reached. No convergence.")
    return None
```

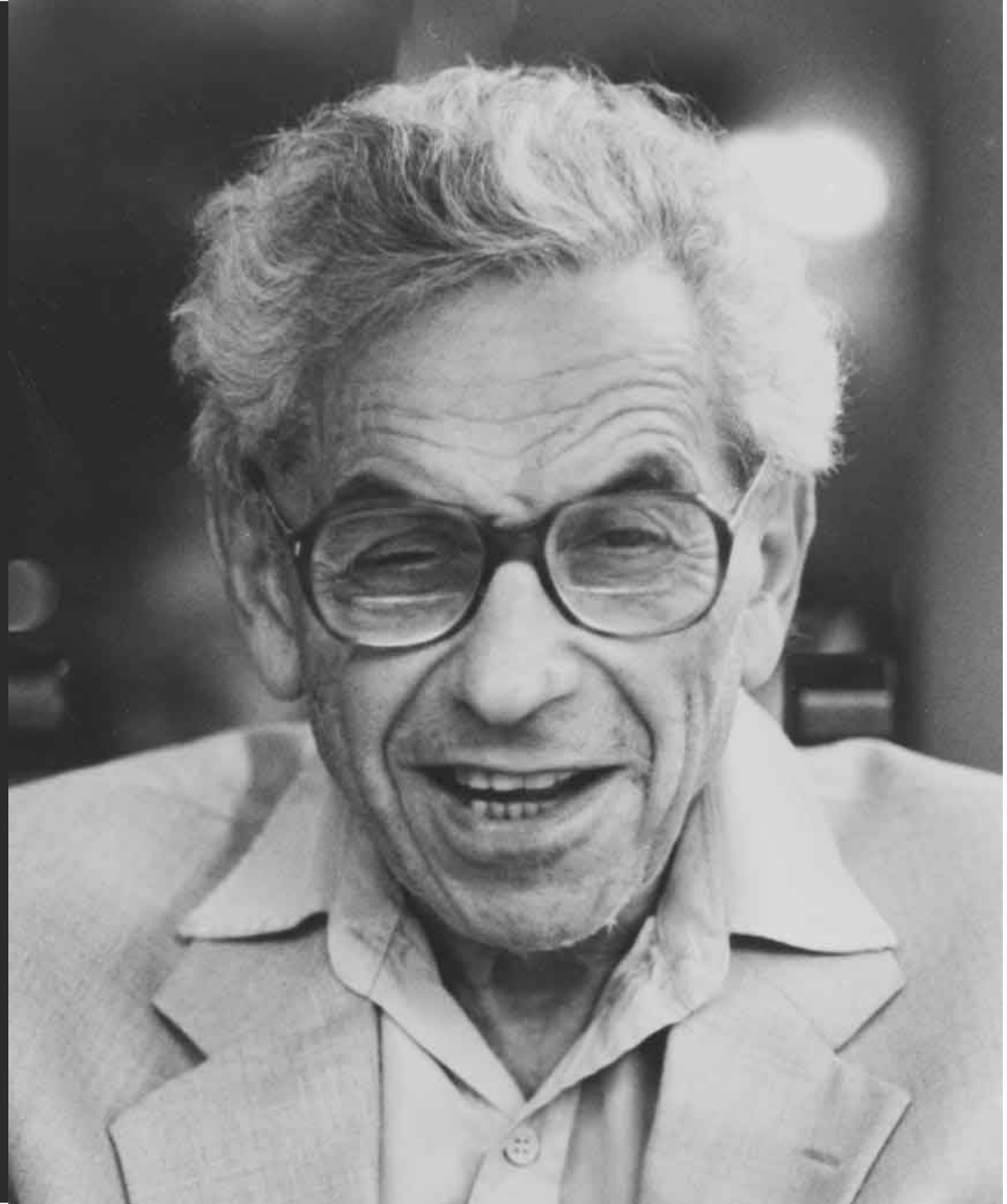
Iteration	x_n	f(x_n)	f'(x_n)	x_n+1
0	7	20	9	4.777777777777778
1	4.777777777777778	4.938271604938269	4.555555555555555	3.693766937669377
2	3.693766937669377	1.1750795014725224	2.387533875338754	3.2015940250208406
3	3.2015940250208406	0.2422341759449456	1.4031880500416811	3.028962725931777
4	3.028962725931777	0.029801565425175 98	1.057925451863553 7	3.000792909833034
5	3.000792909833034	0.0007935385390371863	1.001585819660679	3.000000627710567 5
6	3.000000627710567 5	6.277109605434816e-07	1.000001255421135	3.000000000003952
7	3.000000000003952	3.961275751862558 5e-13	1.000000000007905	2.999999999999999

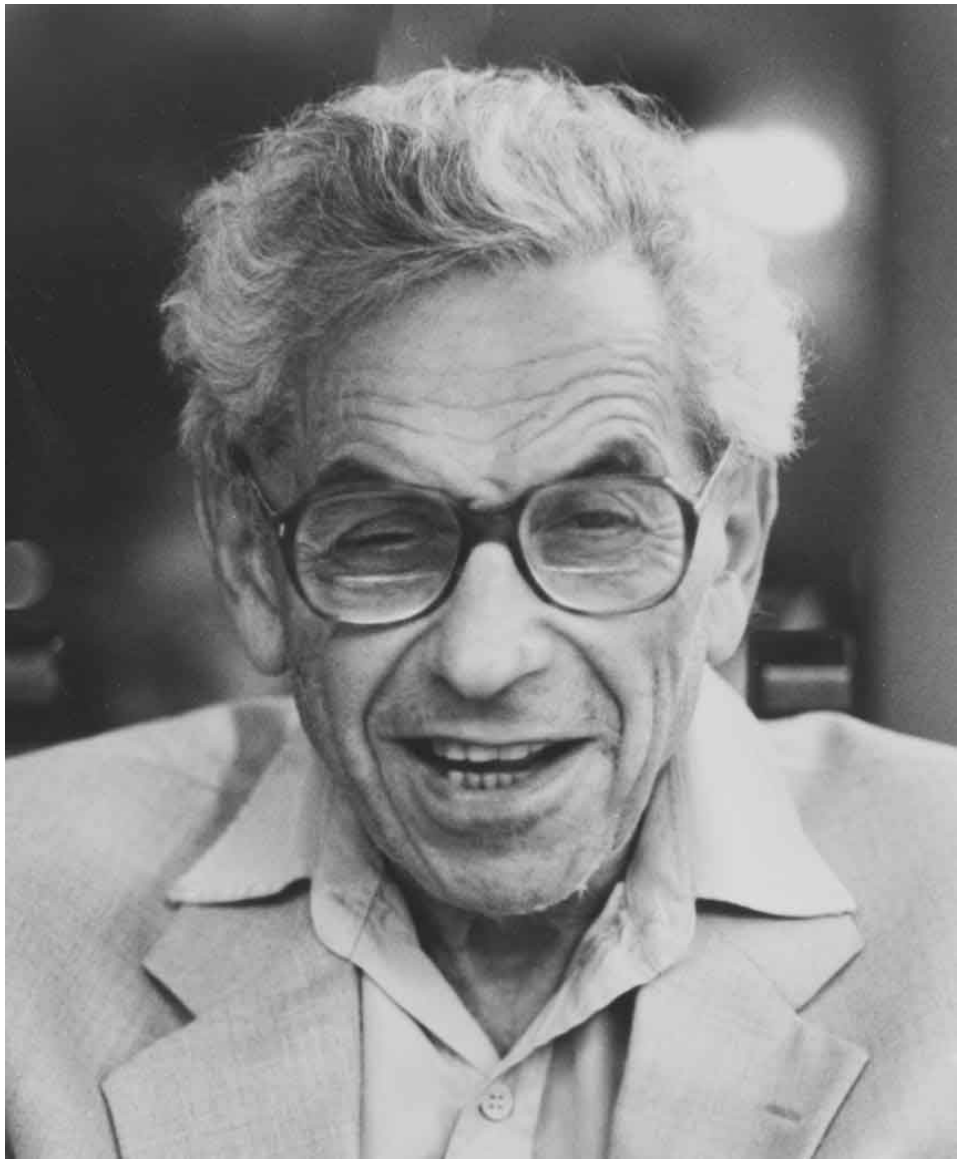
Computational Duality

- **"Mathematical-Computational Symbiosis"** – Mutual benefit.
- **"Formalism-Algorithm Duality"** – Bridge between mathematical formalism and computational implementation.
- **"Theoretical-Practical Confluence"** – Showing their merging influences.

Computer has limited/NO capacity to prove mathematical theorem. But it can help mathematical framework through faster computation.

Mathematics has many computational frameworks but is NOT time efficient.





Paul Erdős

Hungarian Nomad Mathematician

- Spent one year (1953-54) in the mathematics department at the University of Notre Dame in South Bend, Indiana
- Known for around 4000 mathematical papers.

- Best known for The Prime Number Theorem.

$$\lim_{x\to\infty} \frac{\pi(x)}{\frac{x}{\ln(x)}} = 1$$

Offered $500 for Collatz conjecture! Still an open problem. Try!!

The 3n + 1 problem vs Collatz Conjecture

3n + 1 problem: A computational problem

You will given a positive integer n.

- If n is even, repeat the process
- If n is odd, find 3n + 1 and repeat the process
- If n is 1, stop

Collatz Conjecture: A mathematical quest

This process will eventually reach the number 1, regardless of which positive integer is chosen initially.

CSES

CSES Problem Set

Weird Algorithm

TASK | STATISTICS

Time limit: 1.00 s **Memory limit:** 512 MB

Consider an algorithm that takes as input a positive integer n. If n is even, the algorithm divides it by two, and if n is odd, the algorithm multiplies it by three and adds one. The algorithm repeats this, until n is one. For example, the sequence for $n = 3$ is as follows:

$$3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$$

Your task is to simulate the execution of the algorithm for a given value of n.

Input

The only input line contains an integer n.

Output

Print a line that contains all values of n during the algorithm.

Constraints

- $1 \leq n \leq 10^6$

Example

Input:
```
3
```

Output:
```
3 10 5 16 8 4 2 1
```

8-Queen Problem

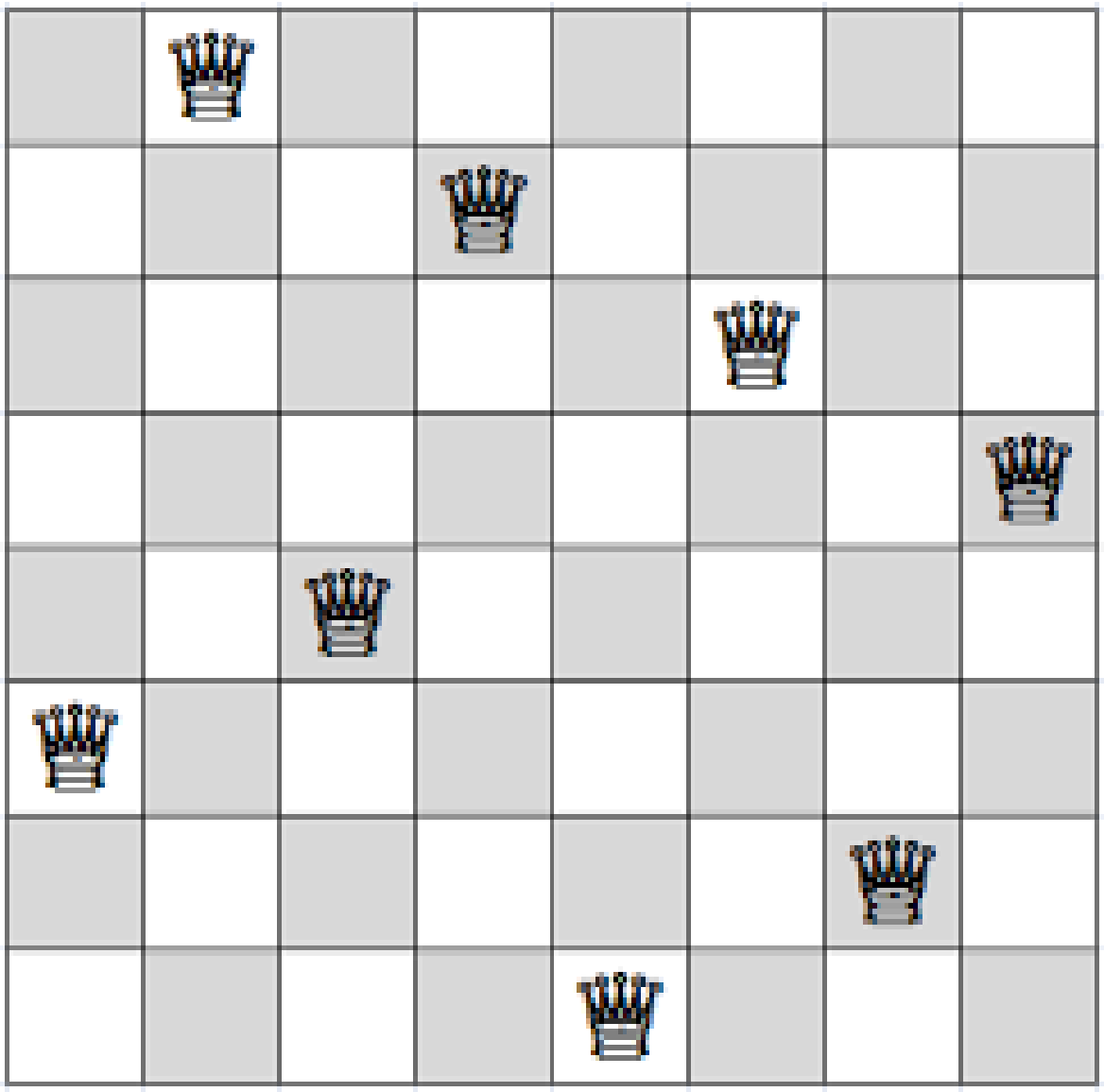
Most famous variant:

You are given an 8×8 chess board. You have to place eight queens on it so that no two queens are attacking each other. Find the number of valid configuration.

Variant 2:

You are given a configuration. Now you have to verify if it is a valid one.

Gauss first solved this problem using backtracking.



Beyond 8 Queen problem: N-Queen problem

Now, you are given an integer for N. You have to find the number of valid configuration.

- Can we find the number of valid configurations for any arbitrary n?
- It’s a pure computational problem. So far we computed up to N = 27. But it requires parallel computing.
- Algorithm is NOT always enough.

n	fundamental	all
1	1	1
2	0	0
3	0	0
4	1	2
5	2	10
6	1	4
7	6	40
8	12	92
9	46	352
10	92	724
11	341	2,680
12	1,787	14,200
13	9,233	73,712
14	45,752	365,596
15	285,053	2,279,184
16	1,846,955	14,772,512
17	11,977,939	95,815,104
18	83,263,591	666,090,624
19	621,012,754	4,968,057,848
20	4,878,666,808	39,029,188,884
21	39,333,324,973	314,666,222,712
22	336,376,244,042	2,691,008,701,644
23	3,029,242,658,210	24,233,937,684,440
24	28,439,272,956,934	227,514,171,973,736
25	275,986,683,743,434	2,207,893,435,808,352
26	2,789,712,466,510,289	22,317,699,616,364,044
27	29,363,495,934,315,694	234,907,967,154,122,528

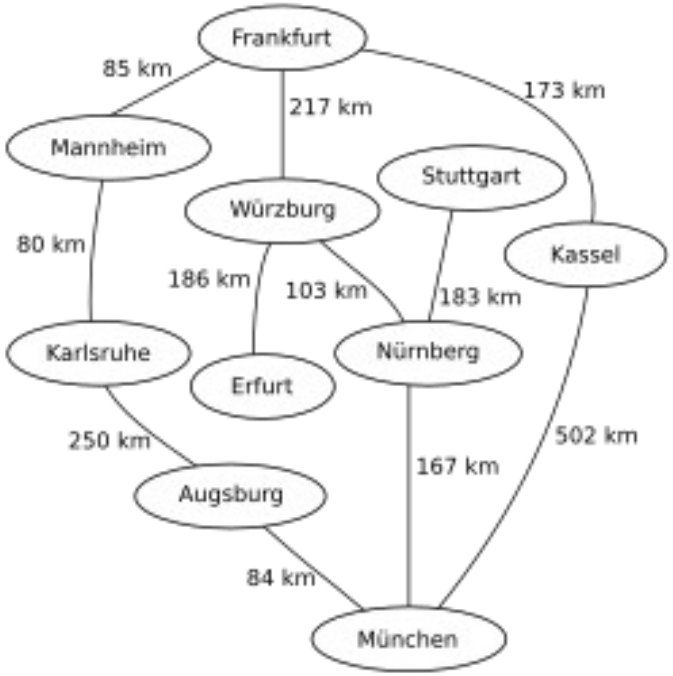
Two problems from graph search

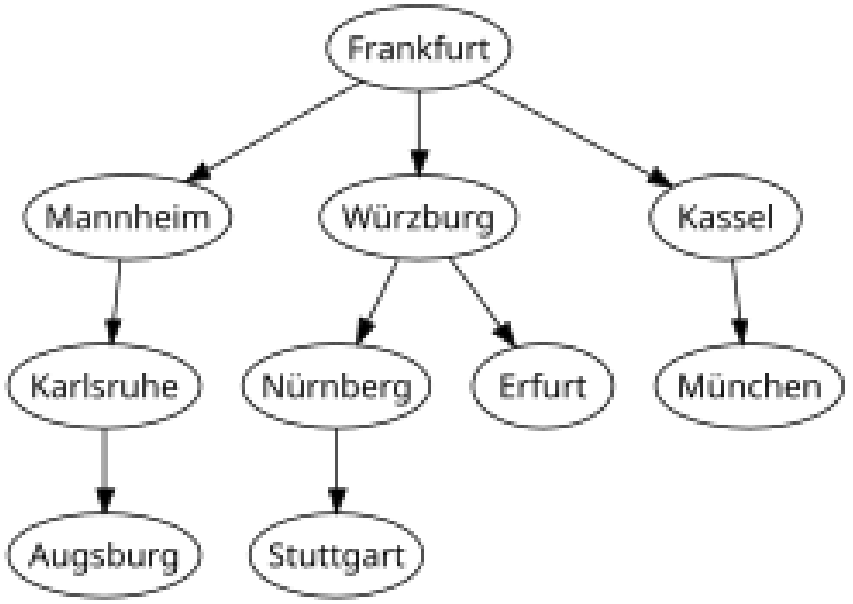
1. Minimum number of hop of a city from Frankfurt.

- This requires a BFS. Need to restructure the graph.
- DON’T need all those distance values.

2. What is the minimum distance of a city from Frankfurt?

- This is almost BFS, but modified BFS.
- The algorithm is called Dijkstra.
- Need all the distances values.





Another Graph search problem Finding out the number of rooms.

You are given a map of floor. There are walls. A room is a place where you can freely move. Now find out the number of rooms.

- Use DFS.
- A fancy name for this problem is flood-fill. You use it a lot in painting.

CSES

CSES Problem Set

Counting Rooms

TASK | STATISTICS

Time limit: 1.00 s **Memory limit:** 512 MB

You are given a map of a building, and your task is to count the number of its rooms. The size of the map is $n \times m$ squares, and each square is either floor or wall. You can walk left, right, up, and down through the floor squares.

Input

The first input line has two integers n and m: the height and width of the map.

Then there are n lines of m characters describing the map. Each character is either . (floor) or # (wall).

Output

Print one integer: the number of rooms.

Constraints

- $1 \le n, m \le 1000$

Example

Input:
```
5 8
########
#..#...#
####.#.#
#..#...#
########
```

Output:
```
3
```

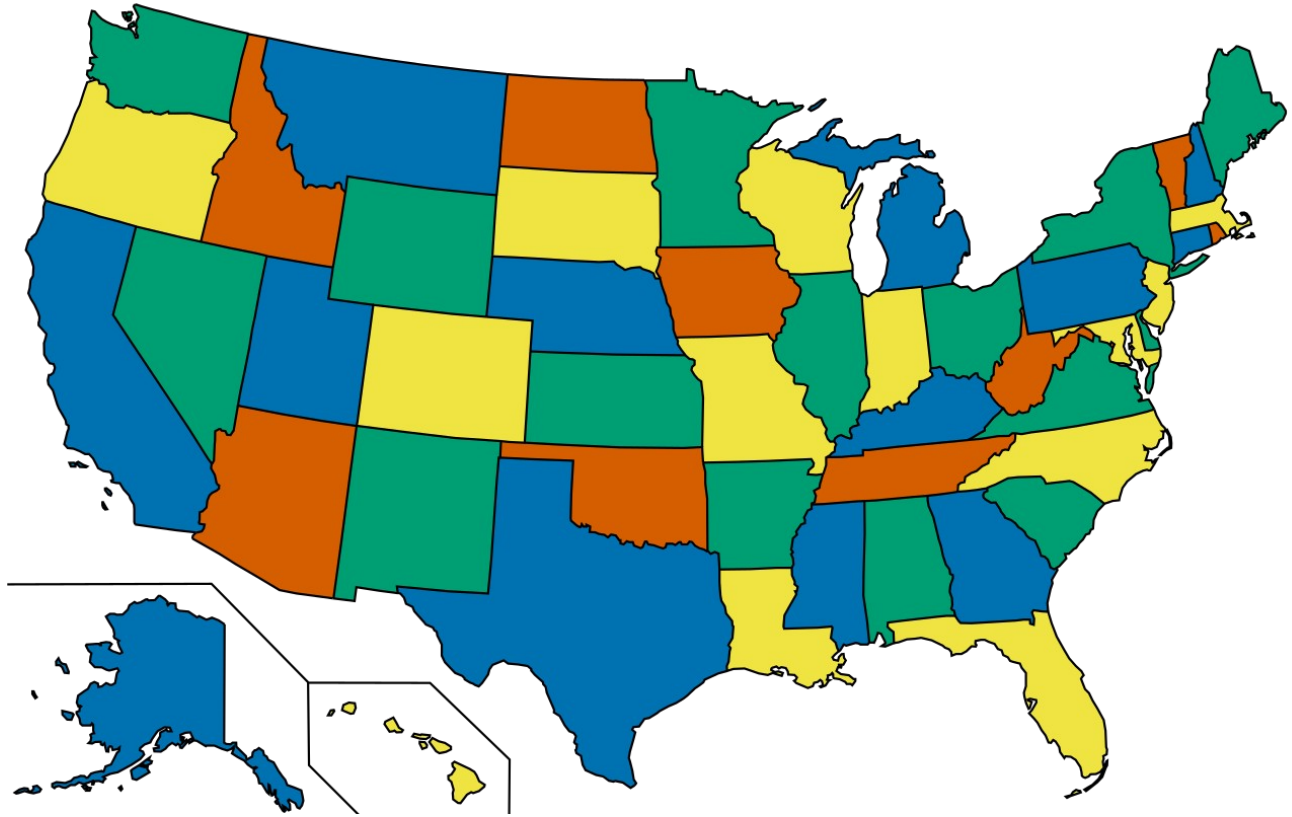
The Four Color Problem

(Not a problem anymore)
It’s solved!

Theorem: Any map drawn on a plane can be colored with at most four colors in such a way that no two adjacent regions share the same color.

A purely mathematical problem.

But not solved purely by human. Needed computational help. The first proof of human-computer collaboration.



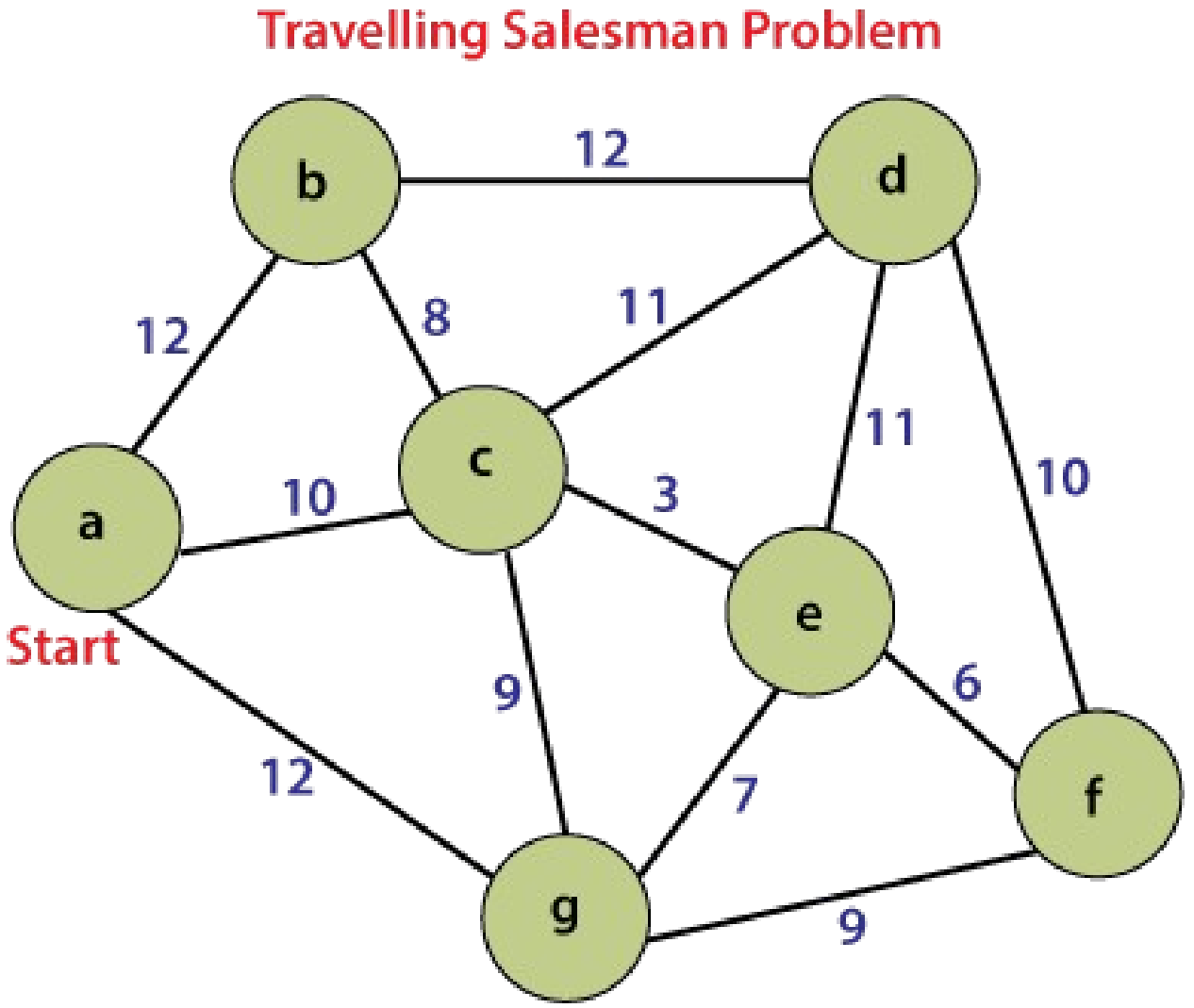
The TSP: A really Hard Problem

A salesman needs to visit n cities, starting from a given city (a) , visiting each city **exactly once**, and returning to the starting city.

The **goal** is to **find the shortest possible route that minimizes the total travel distance (or cost).**

Solution Approaches
Algorithms (for small n):

1. **Brute Force**: Check all (n−1)! possible routes (exponential time).
2. **Dynamic Programming** (**Held-Karp Algorithm**): Solves in **$O(n^2 2^n)$** time.



Problems in Primes

1. Generate all the primes within 100, 1000, 10000…
2. What is the largest prime. (solved!)
3. What is the largest prime so far…
4. How to verify a number is a prime or not?

We can generate primes by Sieve of Eratosthenes. Its simple yet beautiful!

There is no largest prime! We have a proof of that. One of the earliest and fanciest proof. You know that?

The largest prime so far:

$$2^{136,279,841} - 1$$

Found in October 12, 2024 by GIMPS(Great Internet Mersenne Prime Search). The man is Luke Durant.

Primality Testing
How do you test a number n is prime or not?

Easy Solution:
Just divide all the number from 2 to n-1. If found any number then n is composite, otherwise prime.

But this is slow.

Summary of Complexity Comparison

Algorithm	Type	Time Complexity	Practical Use?
Trial Division	Deterministic	$O(\sqrt{n})$	No (too slow)
Optimized Trial Division	Deterministic	$O(\sqrt{n})$	No (still slow)
Fermat Test	Probabilistic	$O(k \log^2 n)$	No (not reliable)
Miller-Rabin Test	Probabilistic	$O(k \log^3 n)$	Yes (widely used)
AKS Algorithm	Deterministic	$O(\log^6 n)$	No (too slow)
ECPP	Deterministic	$O(\log^4 n)$	Yes (best for very large primes)

Programming Language

The real way to use a computer

For programming competition:

Choose any programming language that has three qualities:

1. Easy to use
2. Fast Compilation
3. Big Community

- What is language?!
  - Natural Language
  - Formal Language

- There is language, that’s why there are problems!
- We live in a language mediated reality.

- Make your language better!

The three Programming Languages

- The C languages : Interface to the machine
- The Python : The interface to the AI/ML
- The JavaScript : The interface to the web

Lisp: The other kind of Programming Language

- Declarative
  - Focuses what to compute NOT how to compute
- Functional
  - Prefers functions, therefore recursion rather than looping.
- Uses in AI research
- Immutability
  - Memory friendly

CP: Competitive Programming

Why Programming Contest? There are two reasons:

1. To ensure competitiveness
2. To foster problem solving capacity, therefore creativity

What is my suggestion?
Why do I do programming?

Most likely it will **NOT** give you:

- The best job
- A Nobel Prize
- A ministry

But it will make you:

- Smarter
- Visionary
- Thoughtful

How to CP

The three books:

1. Competitive Programming 3: Steven Halim
2. Programming Challenges
3. Competitive Programmer's Handbook

Easy problems topics:

What is an easy problem? The problem you can solve immediately. Do:

Searching: Linear, Binary
Sorting: Quick Sort, Merge sort, Bubble Sort
Basic Math: Number theoretic Problem
Geometry: Easy Geometry

1. Start learning a programming language. Start with python. Learn everything up to function. Don't hurry! Understand every bit! But only learning is NOT enough. Need Practice.
2. Buy three books.

3. Start solving programming. Take easy problems first. Spend one/two months with easy problems. Don't stay there for long or forever!

4. Now start reading data structures. Start with linked list, queue, priority queue and stack. Try to solve problems that requires these data structures.

5. But you will find eventually that you need to learn graphs. All these data structures are almost useless alone but powerful with graphs.
6. But before going to the graph, practice some problems with backtracking. It will give the power of recursion. In this stage you may want to explore dynamic programming or DP. But please don't.

Graph Before DP:

- Start with Search algorithms. Whatever First Search. Shortest Paths. Circuits.

- Tree: Binary Tree, Heap, BST

- MST, Top Sort
- Graph Coloring Problems

Graph After DP:

- Articulation Point

- Connectivity

- Flow: Max flow, min cuts and variants.

7. Start exploring graph.

8. Now Start exploring DP. Don’t touch DP before you understand the DAGs

9. Go deep in graph and DP. Start solving hard problems.

10. Explore some geometric problems, game theory and other spinoffs.

Where to practice: The online judges

- A lot of problems
- And an automated judging system

1. Start with cses.fi
2. When you are done with introductory problems of cses then start codeforces division 3 and 2. After each contest analyze the solution. Codeforces is a key player in PC and will be a key player on WW3
3. Don’t forget about Uva from the very beginning of your CP journey.
4. BTW, tocoder is another platform. They are American!

Community

- Practice in a group.
- Create discussion group.
- Create reading group.
- CLUB
- Create your own online community.

Onsite Contest

Create your own team.

- Target for NCPC 2026.
- Win ICPC within 2030