CONSTRAINT SATISFACTION PROBLEMS.

WHAT IS A CSP?

•A CSP is a mathematical problem where we need to find values for variables that satisfy certain constraints
•CSPs are like puzzles where you have rules that must be followed
•Three key components: Variables, Domains, and Constraints

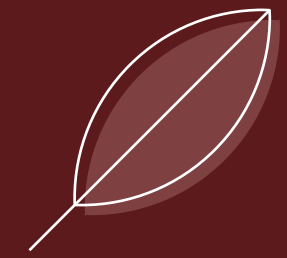
In CSPs we have a search space. What is it ?
It refers to the **set of all possible assignments of values to variables** before applying constraints. It represents all the potential solutions, including both valid and invalid ones.

- An example of a Search Space in CSPs
- 1. Map Coloring Problem
- Variables: `{Nairobi, Mombasa, Kisumu}`
- Domain: `{Red, Green, Blue}`
- Total Search Space:

  •If there are **3 cities** and **3 color choices**, then the total number of assignments is: 3^3 = 27

  •Once constraints (adjacent regions must have different colors) are applied, only a subset of these assignments remains valid.
- The main idea is to eliminate large portions of the search space all at once by identifying variable/value combinations that violate the constraints.

KEY COMPONENTS OF A CSP.



- Variables
- These are the things we need to assign values to
- Examples: Which student gets which seat? Which color goes on which region of a map?
- Variables are the "unknowns" we're trying to figure out

- Domains
- The possible values each variable can take
- For example: If our variable is "T-shirt color," the domain might be {red, blue, green, yellow}
- Domains can be:
  - Discrete (specific values like colors, numbers, etc.)
  - Continuous (any value within a range, like temperature)
  - Finite (limited number of options) or infinite (unlimited possibilities)

- Constraints
- Rules that limit which values can be assigned together
- Types of constraints:
  - Unary constraints: Affect just one variable (e.g., "Sarah cannot wear red")
  - Binary constraints: Affect pairs of variables (e.g., "Alex and Jamie cannot sit together")
- Global constraints: Affect many variables at once (e.g., "All students must have different seats")
- Precedence Constraint: Some tasks must be completed before others. A precedence constraint dictates the order in which tasks or operations must be completed, ensuring that one task finishes before another can begin.

- CSPs deal with assignments of values to variables.
- An assignment that does not violate any constraints is called a **consistent** or **legal assignment.**
- **Complete Assignment:** Every variable is assigned a value that satisfies all constraints.
- **Partial Assignment:** Only some variables are assigned values, and not all constraints are necessarily satisfied yet.

EXAMPLE PROBLEMS ON CSPS.

1. *Map Coloring Problem*.

- We have a map with different regions (like countries or states)
- We need to color each region so that no neighboring regions have the same color
- We want to use the minimum number of colors possible

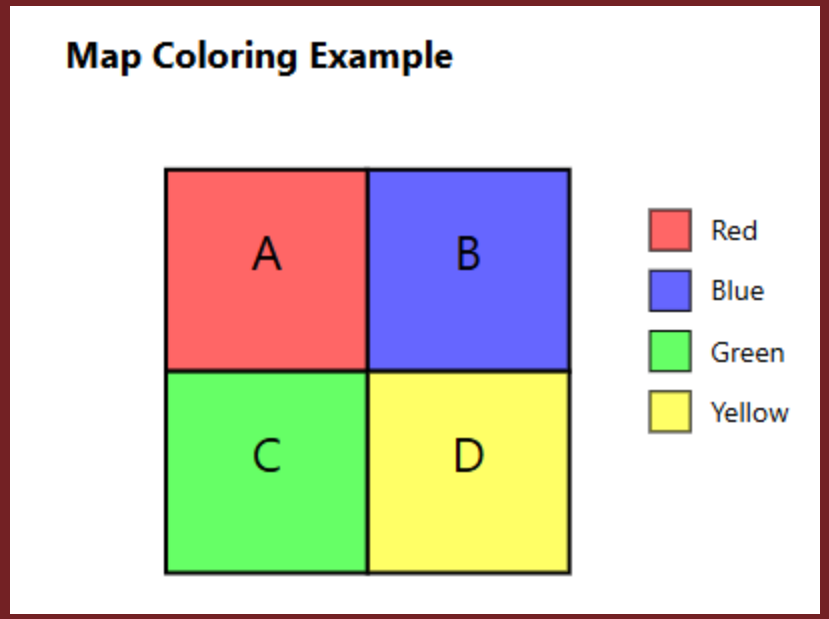
EXAMPLE 1: MAP COLORING PROBLEM

- For this example, this is what we have:
- Variables: Each region on the map i.e. {A, B, C, D}
- Domain: Available colors {Red, Blue, Green, Yellow}
- Constraints: Adjacent regions must have different colors.

- Let's say we have a simple map with 4 regions: A, B, C, and D
- A shares borders with B and C
- B shares borders with A, C, and D
- C shares borders with A, B, and D
- D shares borders with B and C
- Available colors: Red, Blue, Green, Yellow

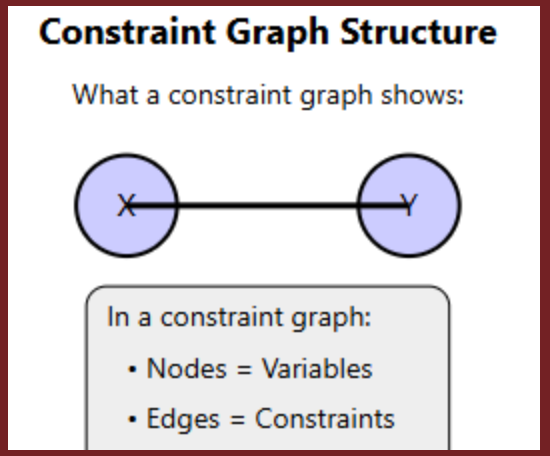
EXAMPLE 1: MAP COLORING PROBLEM

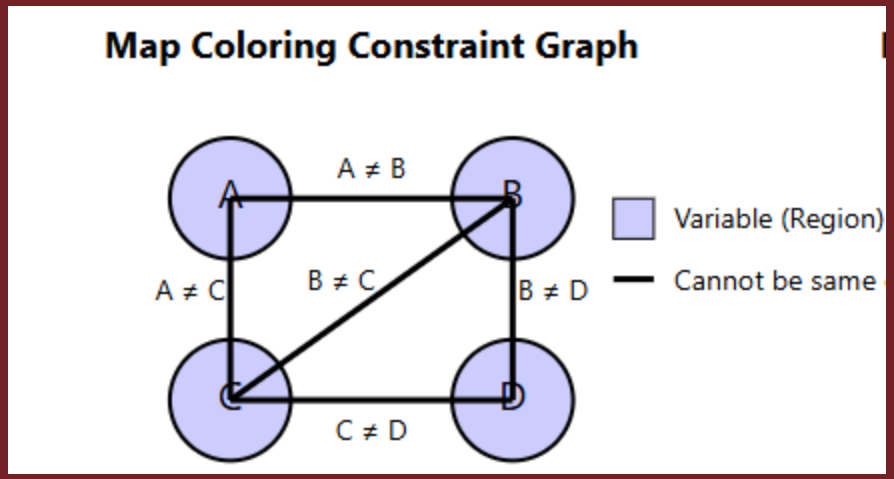
- Solution Process
- Start with region A: Let's color it Red
- Move to region B: Can't use Red (neighbor to A), so color it Blue
- Move to region C: Can't use Red (neighbor to A) or Blue (neighbor to B), so color it Green
- Move to region D: Can't use Blue (neighbor to B) or Green (neighbor to C), so color it Yellow
- Check all constraints: All neighboring regions have different colors, so our solution is valid!



EXAMPLE 1: MAP COLORING PROBLEM

- It can be helpful to visualize a CSP as a constraint graph.
- In a constraint graph, the nodes of the graph correspond to variables of the problem, and an edge connects any two variables that participate in a constraint.





- Shows regions A, B, C, and D as nodes (variables)
- Connecting lines (edges) between regions that share borders
- Each edge represents the constraint that connected regions must have different colors

EXAMPLE 2. DINNER PARTY SEATING PROBLEM

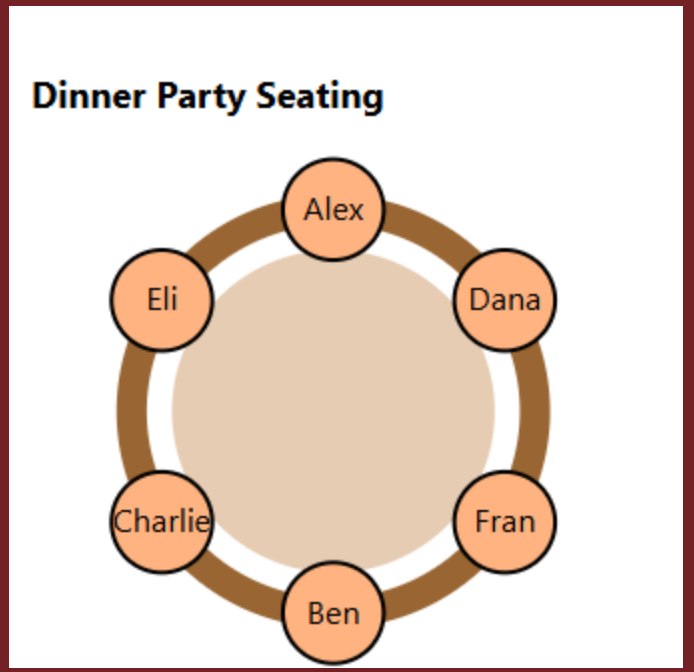
- **Problem:** At a dinner party, certain friends do not get along and cannot sit next to each other.
- **Variables:** Each seat at the table.
- **Domains:** The names of the guests.
- **Constraints:** Each person must have exactly one seat
- Each seat must have exactly one person
- Friends who don't get along cannot be seated next to each other

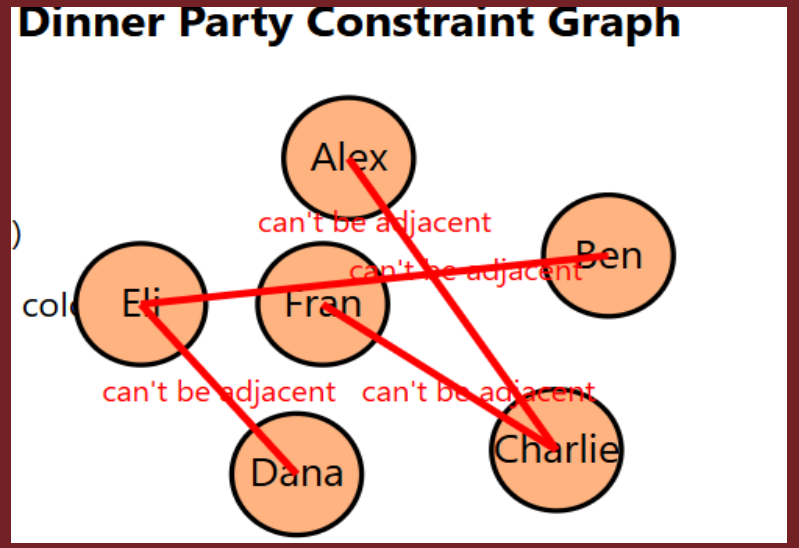
- Specific Example
- 6 friends: Alex, Ben, Charlie, Dana, Eli, and Fran
- Constraints:
  - Alex and Charlie don't get along
  - Ben and Eli don't get along
  - Charlie and Fran don't get along
  - Dana and Eli don't get along
  - Table has 6 seats arranged in a circle (seat 1 is adjacent to seats 2 and 6)

EXAMPLE 2. DINNER PARTY SEATING PROBLEM

- Solution Process
- Start with Alex in seat 1
- Ben could go in seat 2 (adjacent to Alex is fine)
- Try placing Charlie in seat 3
  - This works because Charlie isn't adjacent to Alex
- For seat 4, try Dana
  - This works as Dana has no conflicts with adjacent people
- For seat 5, try Eli
  - Problem! Eli can't sit next to Ben, and seats 2 and 5 are not adjacent
  - But Eli can't sit next to Dana either, and Dana is in seat 4
  - We have a conflict - need to backtrack
- Let's go back to step 4 and try Eli in seat 4 instead of Dana
- Now Dana could go in seat 5
- Finally, Fran in seat 6
  - Problem! Fran can't sit next to Charlie, and Charlie is in seat 3
  - Seats 3 and 6 are not adjacent in our circular table, but wait - they are!
  - We need to backtrack again
- This process continues until we find a valid arrangement, such as:
- Seat 1: Alex
- Seat 2: Dana
- Seat 3: Fran
- Seat 4: Ben
- Seat 5: Charlie
- Seat 6: Eli

EXAMPLE 2. DINNER PARTY SEATING PROBLEM

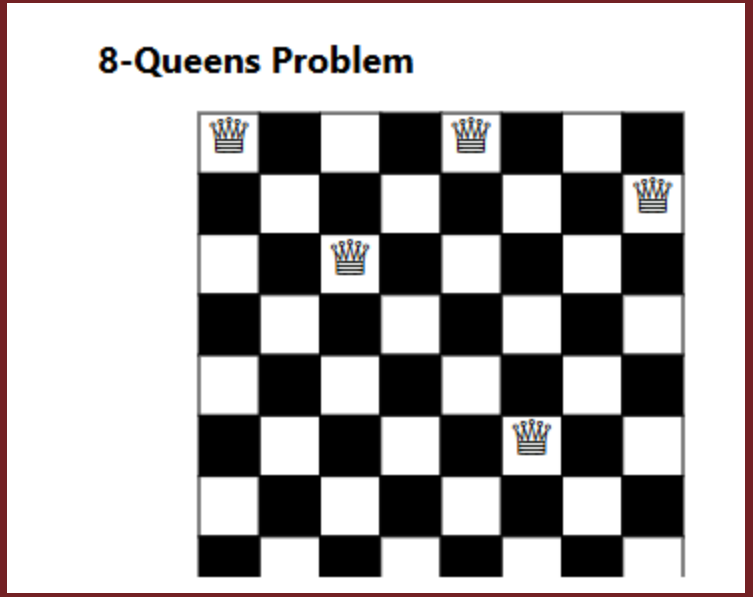




- Each person (Alex, Ben, Charlie, Dana, Eli, and Fran) is represented as a node
- Red lines connect people who cannot sit next to each other
- Shows global constraints that apply to everyone (each person needs exactly one seat)

EXAMPLE 3: CHESS PROBLEM (N-QUEENS)

- We need to place N queens on an N×N chess board
- Queens cannot attack each other (can't be in the same row, column, or diagonal)
- The challenge: How to place all N queens safely?

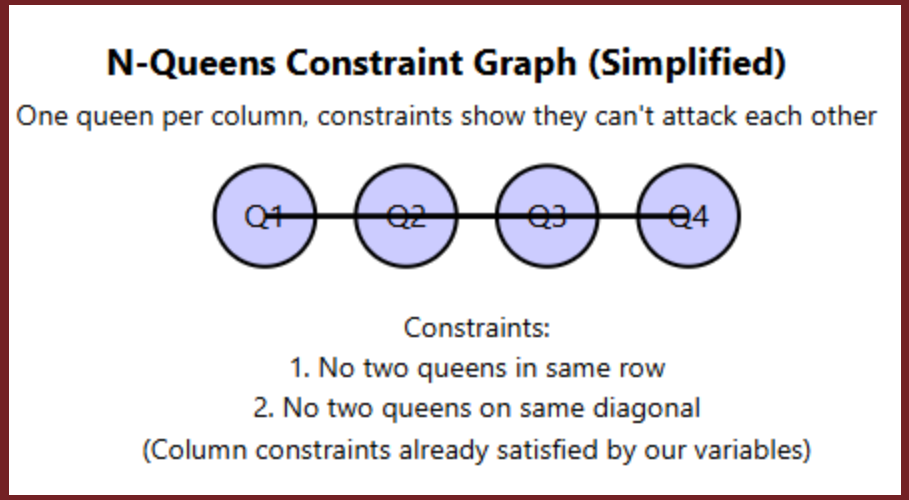


EXAMPLE 3: CHESS PROBLEM (N-QUEENS)

- Classical problem: Place 8 queens on an 8×8 chess board
- We know queens can't share columns, so we can simplify by assigning one queen to each column
- Our variables are Queens 1-8 (representing columns 1-8)
- Domain for each queen: Rows 1-8
- Constraints: No two queens can share a row or diagonal

- Place Queen 1 in column 1, row 1
- Place Queen 2 in column 2:
- Can't use row 1 (same row as Queen 1)
- Can't use row 2 (diagonal from Queen 1)
- Try row 3
- Place Queen 3 in column 3:
- Can't use rows 1, 3, or 5 (conflicts with previous queens)
- Try row 5, but that's diagonal from Queen 1
- No valid options - backtrack!
- Go back to Queen 2 and try row 4 instead of row 3
- We continue this process, backtracking when necessary

EXAMPLE 3: CHESS PROBLEM (N-QUEENS)



- Shows queens (Q1, Q2, Q3, Q4) as nodes
- All queens constrain each other (complete graph)
- Explains the three types of constraints: no two queens can share a row, column, or diagonal
- Simplified to show just 4 queens for clarity

SOLVING CSPS

- Backtracking Search
- What is Backtracking?
- Backtracking is a DFS based approach that assigns values to variables while checking constraints.

- How backtracking search works
- Step-by-step process:
- Choose a variable to assign.
- Select a value from the domain.
- Check if the value violates any constraints.
- If not, continue to the next variable.
- If a conflict arises, backtrack to the previous step and try a different option.

SOLVING CSPS

- Variable & Value Ordering
- The order in which we assign variables matters!
- Strategies include:
- Minimum-remaining-values: Pick the variable with fewest options left
- Degree heuristic: Pick the variable involved in the most constraints
- Least-constraining value: Pick values that rule out the fewest options for other variables

- Intelligent Backtracking
- Forward checking: Look ahead to see if our current assignment causes problems
- Arc consistency: Make sure that for any two related variables, every value in one variable's domain has at least one compatible value in the other's domain
- Conflict-directed backjumping: Jump back to the source of the conflict, not just the previous variable

INTELLIGENT APPROACHES TO SOLVING CSPS

- Constraint learning.
- No-Good Learning: When a particular assignment of values fails, mark it as a 'no-good' state to avoid repeating the mistake.
- After making a choice, immediately eliminate invalid options from related variables
- Helps to reduce the search space dramatically
- Example: After placing a queen, we can immediately eliminate all threatened positions.

- Intelligent Backtracking
- Chronological Backtracking: Undo assignments in the reverse order they were made.
- Conflict-Directed Backjumping: Go back directly to the source of the conflict instead of stepping back one by one.

CSPS IN THE REAL WORLD

- Scheduling Problems
- Class timetabling in schools
- Employee shift scheduling
- Sports league scheduling
- Airport gate assignment

- Assignment Problems
- Assigning doctors to patients
- Matching students to projects
- Allocating resources in a business
- Configuration Problems
- Configuring computer systems
- Product customization options
- Network design

CONCLUSION

- Key Takeaways.
- CSPs are about finding values for variables while following constraints
- They represent many everyday problems we need to solve
- Solving CSPs involves systematic search and logical thinking
- Understanding CSPs helps develop problem-solving skills
- CSPs teach us that sometimes we need to try, fail, and try again (backtracking)
- CSPs are powerful tools for solving complex problems with constraints.
- By using strategies like backtracking and constraint learning, solutions can be found efficiently.

THANK YOU

Newton Luttah

Owen Kimani

Kenneth Ruto

Kitavi Duncan

Douglas Kimani