## Intro

- really busy airport in atlanta, how do they schedule flight
  - constriaint satisfaction
- the techniques we can go over to make "TWO" + "TWO" = "FOUR"
  - Backtracking, forward checking, mimum remaining values, least constraining value
  - 734 + 734 = 1468

## Map Coloring

- classic example of constriant satisfaction
- mimimum colors as possible without coloring two adjacent territory the same color
  - Variables: WA, NT, Q, NSW, V, SA, T (territories)
  - Domains Di = {green, blue, orange}
  - Constraints: adjacent regions must have different colors
  - Can list all the individual pair constraints (like Q != NSW)
  - Can list all the accepted pair constraints (like WA can == Q)

## Constraint Graph

```
* binary constraints, 2 variables, represented by the constraint graph
* the arcs show the constrains between variables
```

## CSP Examples

- possible to have contraints with 3 or more variables or soft constraints (prefer something rather than require it)
- contraint optimization problems (solved by linear programming)
  - constraint satisfaction problems (sudoku, floor planing)

## Constraint Hypergraph

- two + two = four problem
  - global contraint: none of the letters can == each other
  - hypergraph contains boxes for constraints (like O+O = R+10*X1)

## Backtracking search

- This stupid algorithm brute force
- recursively: see if we reached soluton, if not then get next unassigned variable and try out a number, if it meets contraints then call function again with new assignent, if not then try something else
  - if we reach a dead end, then we backtrack
- not too efficient, can make it more efficent

## Imporving Backtracking efficiency

- least contraining value
  - select options that mimize contraints for future variables (choose to paint with a color that's already been used instead of a new color)

- mimimum remaining values
  - chose options that have the tightest contraints or options and take care of them first
- choose variables to look at with the most contraints

## Forward checking

- when assigning a variable, keep track of the other varibles to ensure that we dont fuck them over
- early warning system that a search is going down the wrong path

## Constraint propagation and Arc Consistency

- can use constrain propagation with forward checking
- arc consistant, remove colors from neighbors if it's the only one color left with forward checking.
  - this can cause chain reaction
- can save a lot of unecessary deep searching in complicated problems

## Structured CSPs

- look at the structure and see if we can make smaller separeate problems
- instead of 80 variables (binary), can make it 4 20 variable subproblems
  - goes from 2 ˆ 80 to 4*2 ˆ 20, much fucking better
- if theres a CSP with no loops, can solve in O(n*dˆ2) instead of O(dˆn)
  - choose a root and order variables from root to leaves such that parents precede in the ordering
  - then keep going up the tree being all arc compliant, if you fail then report failure
- if shit isn't a tree, then you can solve for one node, then the rest of the nodes can be structed into tree

## Iterative Algorithms

- algoritms work well when there are a lot of moves and very littles. Didn't really understand this section :S

## Readings on Constraint satisfaction

- AIMA: Chapter 6

## Lab

- n-queens except more badass