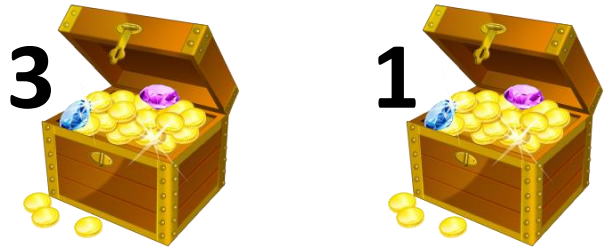
Fun with Search

EECS 348: Artificial Intelligence

- You have keys of various types (numbers)

- Each key unlocks a chest of matching type
  - But then breaks!



- Inside chests: treasure. Also: keys.
- You know the chest types, and which keys are where
  - Question: can you open all the chests?

Example

- You start with:  1

Chest Number	Key Type to Open Chest	Keys types inside chest
1	1	None
2	1	1, 3
3	2	None
4	3	2

- Can unlock all, in order (2, 1, 4, 3)
  - (2, 4, 1, 3) also possible
  - stipulation: return "**lexicographically smallest**"

Backtracking Search!

```
Backtrack(Keys, Chests)
    For i = 1 to |Chests|
        If you can open Chest i :
            Keys += Keys(Chest i)
            Chests[i] = Open
            If Backtrack(Keys, Chests)
                return true
                Else back out changes
    return false
```

Issues

- Too slow
- Can we use CSP heuristics?
  - Not exactly
  - What are the variables?
    - X_i = chest opened at step i?
    - Values = possible chest IDs
    - Constraints are complex
      - AllDiff over variables, sequential ordering, etc…
      - The “lexicographically smallest” requirement implies we can’t return **just any** satisfying assignment

Search problem

- We have to find the “lexicographically smallest” solution
  - Different from the costs we’ve considered in search problems before
  - Code Jam says this is a “red herring”
    - Say you can answer “is there a solution from here?”
    - Then open lowest # chest, ask the above question, backtrack as necessary
    - Sidenote: what’s the worst case complexity introduced by this? For n = 200…?

Let’s proceed

- Say all we want to do is answer “is there a solution from here?”
- Ingenious (in my opinion) observation: There is a solution if and only if
  - There are enough key types available (anywhere) to open all unopened chests, and
  - You can find a path to *some* key of any type for which there’s an unopened chest

Leads to a fast algorithm

- There are enough key types available (anywhere) to open all unopened chests?
  - Just counting

- You can find a path to *some* key of any type for which there's an unopened chest
  - Just BFS/DFS on max 200 nodes, easy

Proof

- Solvable => conditions (easy)
- Conditions => solvable (not easy)
  - Key idea: you can always open at least one chest such that the conditions still hold

What else could we do?

- Say we're not quite smart enough to come up with the previous line of reasoning in time
- Borrow an idea from CSPs
  - forward checking
- Forward checking:
  1. Are there enough keys of the right types available – anywhere – to open all unopened chests?
  2. Pretend the keys didn't break. Can we open all the chests?

  …thanks to Thanapon Noraset ☺

Those heuristics work!

- Solves both small and large data sets in the Code Jam
  - Not a lot of people were able to solve these problems
- Something to think about: can you break the heuristics?