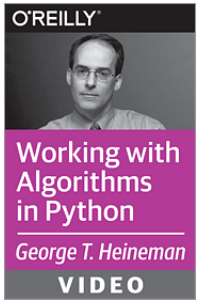
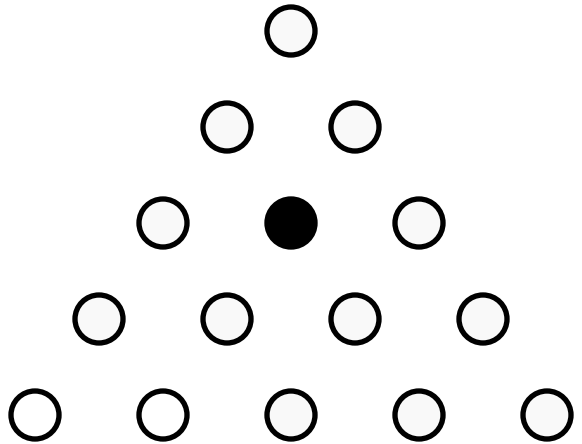

O'REILLY®

Brute Force Algorithms



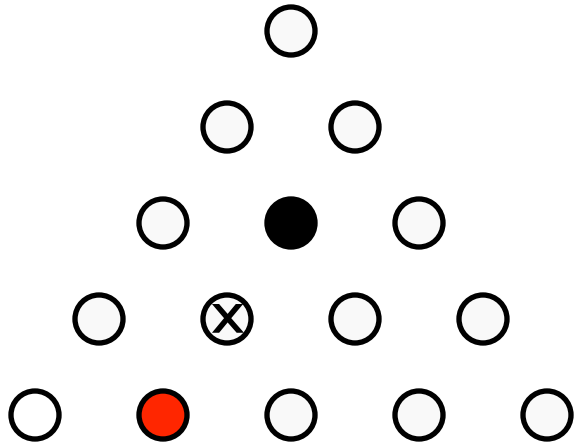
Solitaire Games

- Common solitaire games are challenging
  - Simple rules with no simple solution
- Consider peg-jumping game
  - 14 pegs with one empty hole
  - Remove peg by jumping a peg to an empty hole over that peg



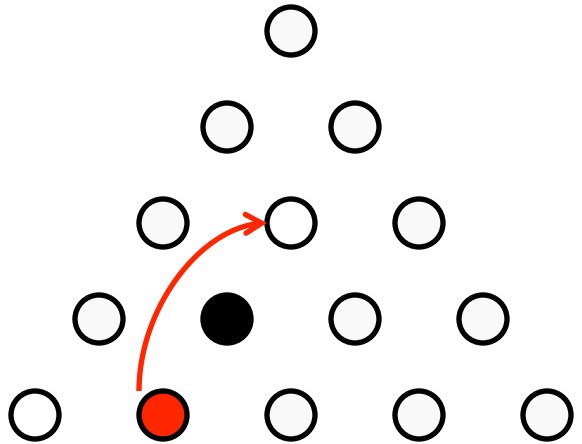
Solitaire Games

- Common solitaire games are challenging
  - Simple rules with no simple solution
- Consider peg-jumping game
  - 14 pegs with one empty hole
  - Remove peg by jumping a peg to an empty hole over that peg



Solitaire Games

- Common solitaire games are challenging
  - Simple rules with no simple solution
- Consider peg-jumping game
  - 14 pegs with one empty hole
  - Remove peg by jumping a peg to an empty hole over that peg
  - Continue until one peg left



Brute Force Algorithms

- Don’t try to intelligently solve the game
  - Develop strategy to blindly try all possible move sequences
- Observations
  - With each **move** the number of pegs is reduced
  - **Undo** moves to try different path when hit a dead end
  - Can identify solution easily (just one peg left)

Brute Force Algorithms

- Consider recursive approach
  - Construct a path of executed moves
  - Board state updated as moves are made
  - Tries all possible attempts

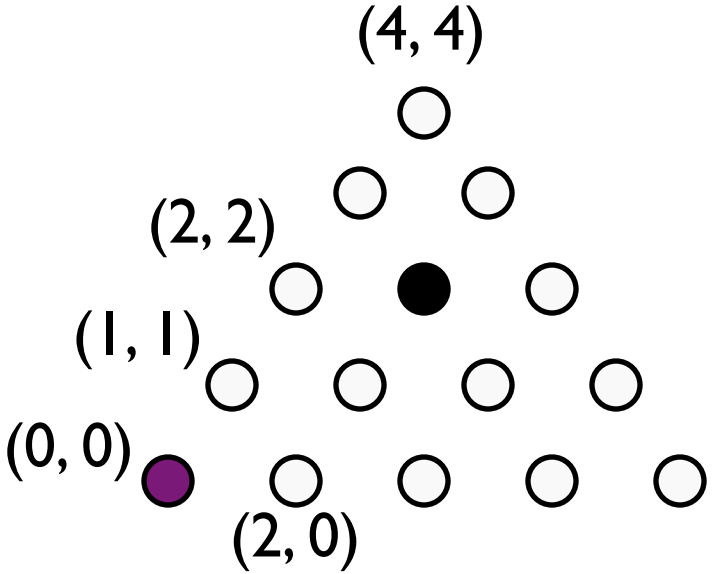
```
solve (board, path)
  if board is solved return True

  for all possible moves m in board
    add m to path and make move on board
    if solve (board, path) return True
    undo m on board and remove from path

  return False
```

Brute Force Issues

- Board state representation
  - Space efficient
  - Easy to compute possible moves
- Move representation
  - Support execute and undo
- State Explosion
  - Limitations imposed by problem

Peg Solitaire Game State Representation

- Python dictionary for board storage
  - key = (*c*,*r*)
  - Value = True/False
- Move Decisions
  - E/W, SW/NE, SE/NW
  - Execute on board
  - Undo on board



Magic Square
State Representation

- Use Brute Force approach to build a Magic Square
  - Two dimensional arrangement of n^2 numbers
  - Sum of rows, columns, diagonals is the same
  - Most people can arrive at 3x3 solution
  - What about 4x4? Or higher orders?

8	**1**	**6**
3	**5**	**7**
4	**9**	**2**

Brute Force Algorithm Summary

- Useful when hard to create intelligent solutions
- Works best when
  - Representation of state is small
  - Easy to determine available moves
  - Problem size is manageable
- Your mileage will vary
  - Overall approach works on countless problems