CS200: Recursion

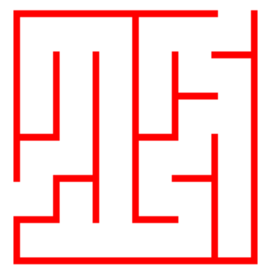
Prichard Ch. 6.1 & 6.3

Backtracking

- Problem solving technique that involves **guesses** at a solution.
- Retrace steps in reverse order and try new sequence of steps

Depth First Search

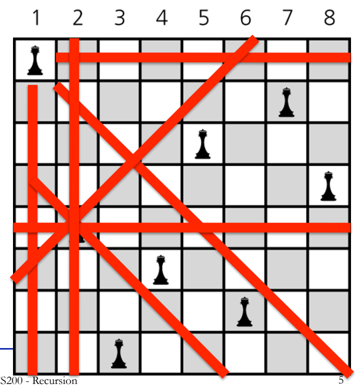
- Looking for a path in a maze
- Strategy:
  - Prioritize directions: right, straight or left.
  - At a dead end "backtrack" and try a different direction
- Recursive solution?



The Eight Queens Problem

Place 8 Queens!

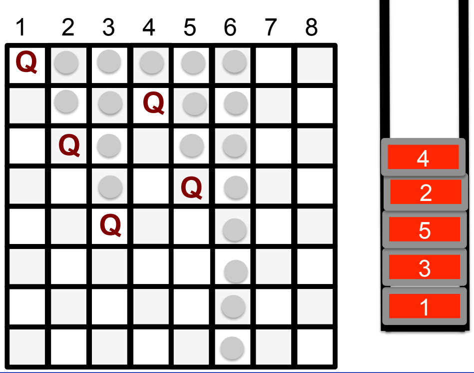
No queen can attack any other queens.



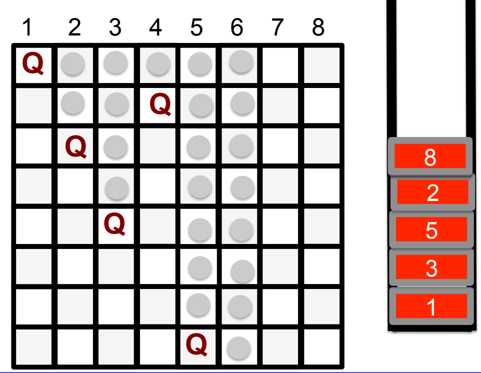
Solution with recursion and backtracking

```
placeQueen (in currColumn:integer)
if ( currColumn > 8) {
   The problem is solved
} else {
    while (unconsidered squares exist in currColumn and the
           problem is unsolved) {
        Determine if the next square is safe.
        if (such a square exists){
           place a queen in the square
           placeQueens(currColumn+1) // try next column
            if (no queen safe in currColumn+1) {
                   remove queen from currColumn
                  try the next square in that col.
            }
        }
    }
}
```

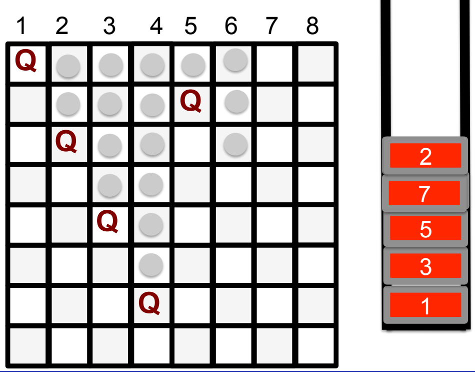

Example



Hit 'Dead End'



Backtrack



Mathematical Induction in Dominos

We have N dominos -- **If we push the 1st domino, will N dominos fall?**

We should show:

- If we push *the 1st one*, it falls
- For all dominos, if the previous domino falls, next domino falls

Process:
Show something works the first time
Assume that it works for this time
Show it will work for the next time, under the assumption
Conclusion, it works all the time

Principle of Mathematical Induction

- To prove that $P(n)$ is true for all positive integers n, where $P(n)$ is a propositional function,
- Two parts of mathematical induction
  - **Basis step**: verify that $P(1)$ is true
  - **Inductive step**: Show that the conditional statement $P(k) \rightarrow P(k+1)$ is true for all (positive, or non-negative) integers k.
- P(n): Propositional function
- P(k): Inductive hypothesis

Example

- Use mathematical induction to show that,
  1+2+3+ … + n = n(n+1)/2

for all positive integers n

Question 1. What is the propositional function here?

Question 2. What is the inductive hypothesis?

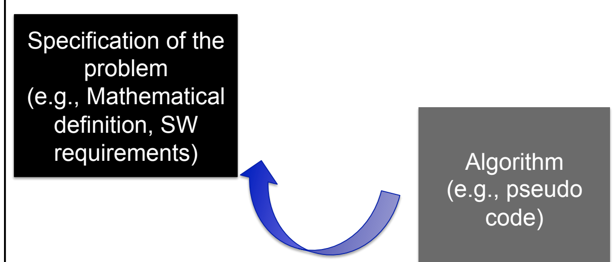
Recursion

- Specifies a solution to one or more base cases
- Then demonstrates how to derive the solution to a problem of an arbitrary size
  - From the smaller size of the same problem.

Mathematical Induction

- Proves a property about the natural numbers by
  - Proving the property about a base case and
  - Then proving that the property must be true for an arbitrary natural *N* if it is true for the natural number smaller than *N*.
- In this section, we will use MI to prove:
  - **(1) correctness of the recursive algorithm**
  - **(2) deriving the amount of recursive work it requires**

Correctness of the Recursive Factorial Method



Does your algorithm satisfy the specification of the problem?

Correctness of the Recursive Factorial Method

Definition of Factorial

factorial(n) = n (n – 1) (n – 2) … 1 for any integer $n > 0$

factorial(0) = 1

Definition of method *fact(N)*

```
fact (in n: integer): integer
    if (n is 0) {
       return 1
    } else {
       return n* fact(n-1)
    }
```


Prove that the method fact computes the factorial of its arguments

Basis step:

fact(0) = 1

Inductive Step:

Show that for an arbitrary positive integer *k*, if *fact(k)* returns *k*!, *fact(k+1)* returns *(k+1)*!

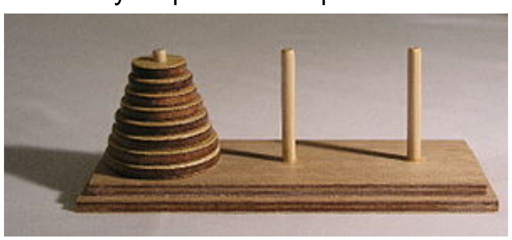
Assume that, *fact(k) = k (k-1) (k-2) … 2 1*

For *n = k+1*,

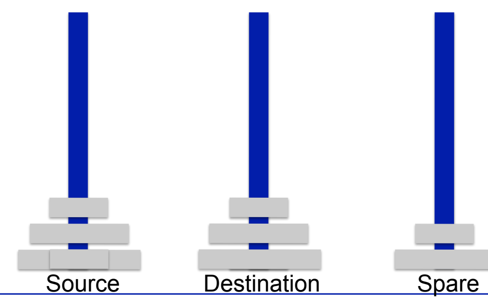
Show that *fact(k+1)* returns *(k+1) k (k-1) (k-2) … 2 1*

Deriving the amount of recursive work

- **The Towers of Hanoi Example**
- **Only one** disk may be moved at a time.
- No disk may be placed on top of a smaller disk.



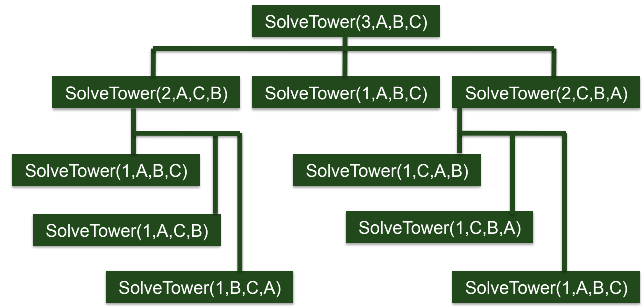
States in the Towers of Hanoi



Recursive Solution

```
solveTowers (in count: integer, in source: Pole, in
  destination: Pole, in spare:Pole)
   if (count is 1) {
       Move a disk directly from source to destination
   } else{
       solveTowers(count-1, source, spare, destination)
       solveTowers(1, source, destination, spare)
       solveTowers(count-1, spare, destination, source)
   }
```


Example with 3 disks



Cost of Towers of Hanoi

- If we have N disks, how many moves does *solveTowers()* make to solve the problem?
- From the software
  - $moves(1) = 1$
  - $move(N) = move(N\text{-}1)+1+move(N\text{-}1)$ (if $N>1$)
- A closed form formula for the number of moves that solveTowers requires for N disks:
  - $moves(N) = 2^N - 1$ (for all $N>=1$)
- **Is this true for the *solveTowers()* method with N disks?**

Proof

- Basis Step
  - Show that the property is true for N = 1.
  - $2^1 - 1 = 1$, which is consistent with the recurrence relation's specification that $moves(1) = 1$
- Inductive Step
  - Property is true for an arbitrary k ➔ property is true for $k+1$
  - Assume that the property is true for N = k
    - $moves(k) = 2^k-1$
  - Show that the property is true for $N = k + 1$

Proof – cont.

- $moves(k+1) = 2 * moves(k) + 1$

  $= 2 * (2^k - 1) + 1$

  $= 2^{k+1} - 1$

Therefore the inductive proof is complete.