Backtracking

Optimized Bruteforce

General Idea

General Idea

- We have complex problem

General Idea

- We have complex problem
  - Multiple decisions are needed

General Idea

- We have complex problem
  - Multiple decisions are needed
  - Each decisions requires making one of many possible choices

General Idea

- We have complex problem
  - Multiple decisions are needed
  - Each decisions requires making one of many possible choices
  - Few solutions exist

General Idea

- We have complex problem
  - Multiple decisions are needed
  - Each decisions requires making one of many possible choices
  - Few solutions exist
  - Certain, partial solutions can be tested for validity

General Idea

- We have complex problem
  - Multiple decisions are needed
  - Each decisions requires making one of many possible choices
  - Few solutions exist
  - Certain, partial solutions can be tested for validity
- Example

General Idea

- We have complex problem
  - Multiple decisions are needed
  - Each decisions requires making one of many possible choices
  - Few solutions exist
  - Certain, partial solutions can be tested for validity
- Example
  - Sudoku

Sudoku

Sudoku

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

Sudoku

- What decisions do we have?

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

Sudoku

- What decisions do we have?
  - Empty spots

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

Sudoku

- What decisions do we have?
  - Empty spots
- What options are there?

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

Sudoku

- What decisions do we have?
  - Empty spots
- What options are there?
  - Values (typically between 1 and 9)

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

Sudoku

- What decisions do we have?
  - Empty spots
- What options are there?
  - Values (typically between 1 and 9)
- Trying all possible boards might be much…

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

Sudoku

- What decisions do we have?
  - Empty spots
- What options are there?
  - Values (typically between 1 and 9)
- Trying all possible boards might be much…
- How many possible finished boards are there?

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

Sudoku

- What decisions do we have?
  - Empty spots
- What options are there?
  - Values (typically between 1 and 9)
- Trying all possible boards might be much…
- How many possible finished boards are there?
  - A lot

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

Sudoku

- What decisions do we have?
  - Empty spots
- What options are there?
  - Values (typically between 1 and 9)
- Trying all possible boards might be much…
- How many possible finished boards are there?
  - A lot
  - $9^{\#\text{ empty spots}}$

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

Sudoku

- What decisions do we have?
  - Empty spots
- What options are there?
  - Values (typically between 1 and 9)
- Trying all possible boards might be much…
- How many possible finished boards are there?
  - A lot
  - $9^{\#\text{ empty spots}}$
  - In our case 9^{58}

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

Sudoku

- What decisions do we have?
  - Empty spots
- What options are there?
  - Values (typically between 1 and 9)
- Trying all possible boards might be much…
- How many possible finished boards are there?
  - A lot
  - $9^{\#\text{ empty spots}}$
  - In our case 9^{58}
  - ~2e55

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

Solving!

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

Solving!

- Let’s try the first decision

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

Solving!

- Let's try the first decision

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

Solving!

- Let's try the first decision
- And the second...

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

Solving!

- Let's try the first decision
- And the second...

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

Solving!

- Let's try the first decision
- And the second…
- We could keep trying decisions

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

Solving!

- Let's try the first decision
- And the second…
- We could keep trying decisions
  - Probably a bad idea

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

Solving!

- Let's try the first decision
- And the second…
- We could keep trying decisions
  - Probably a bad idea
  - So we won't

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

Solving!

- Let's try the first decision
- And the second…
- We could keep trying decisions
  - Probably a bad idea
  - So we won't
- Represent our choices graphically

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

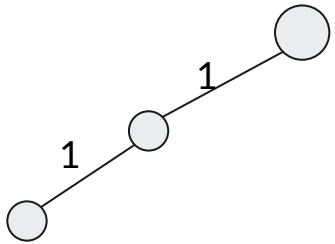
Solving!

- Let's try the first decision
- And the second…
- We could keep trying decisions
  - Probably a bad idea
  - So we won't
- Represent our choices graphically
  - Decision tree

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

Solving!

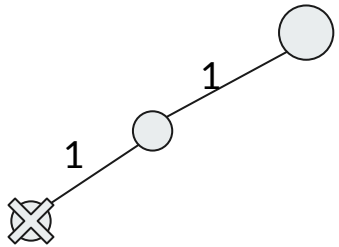
- Let's try the first decision
- And the second…
- We could keep trying decisions
  - Probably a bad idea
  - So we won't
- Represent our choices graphically
  - Decision tree



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

Solving!

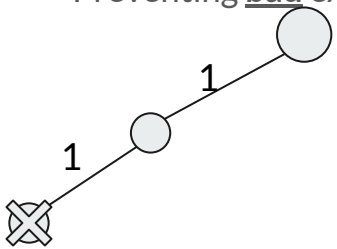
- Let's try the first decision
- And the second…
- We could keep trying decisions
  - Probably a bad idea
  - So we won't
- Represent our choices graphically
  - Decision tree



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

Solving!

- Let's try the first decision
- And the second…
- We could keep trying decisions
  - Probably a bad idea
  - So we won't
- Represent our choices graphically
  - Decision tree
- Preventing **bad** explorations reduces runtime



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

Pseudo-code

Pseudo-code

- First decision : for 1 to 9

Pseudo-code

- First decision : for 1 to 9
  - If !good continue
  - Second decision : for 1 to 9

Pseudo-code

- First decision : for 1 to 9
  - If !good continue
  - Second decision : for 1 to 9
    - If !good continue
    - Third decision : for 1 to 9

Pseudo-code

- First decision : for 1 to 9
  - If !good continue
  - Second decision : for 1 to 9
    - If !good continue
    - Third decision : for 1 to 9
      - ...

Pseudo-code

- First decision : for 1 to 9
  - If !good continue
  - Second decision : for 1 to 9
    - If !good continue
    - Third decision : for 1 to 9
      - ...
- Code might get a little intense

Pseudo-code

- First decision : for 1 to 9
  - If !good continue
  - Second decision : for 1 to 9
    - If !good continue
    - Third decision : for 1 to 9
      - ...
- Code might get a little intense
- New idea!

Pseudo-code

- First decision : for 1 to 9
  - If !good continue
  - Second decision : for 1 to 9
    - If !good continue
    - Third decision : for 1 to 9
      - ...
- Code might get a little intense
- New idea!
- Solve Recursively

Pseudo-code

- First decision : for 1 to 9
  - If !good continue
  - Second decision : for 1 to 9
    - If !good continue
    - Third decision : for 1 to 9
      - …
- Code might get a little intense
- New idea!
- Solve Recursively
  - Recurse (decision)
    - IF decision is at the end THEN Do Base Case
    - FOR each choice in decision
      - Try choice
      - Recurse (decision + 1)

Pseudo-code

- First decision : for 1 to 9
  - If !good continue
  - Second decision : for 1 to 9
    - If !good continue
    - Third decision : for 1 to 9
      - …
- Code might get a little intense
- New idea!
- Solve Recursively
  - Recurse (decision)
    - IF decision is at the end THEN Do Base Case
    - FOR each choice in decision
      - Try choice **// Could be a bad choice!**
      - Recurse (decision + 1)

Backtracking!

Backtracking!

- Recurse (decision)
  - IF decision is at the end THEN Do Base Case
  - FOR each choice in decision
    - Try choice
    - Recurse (decision + 1)

Backtracking!

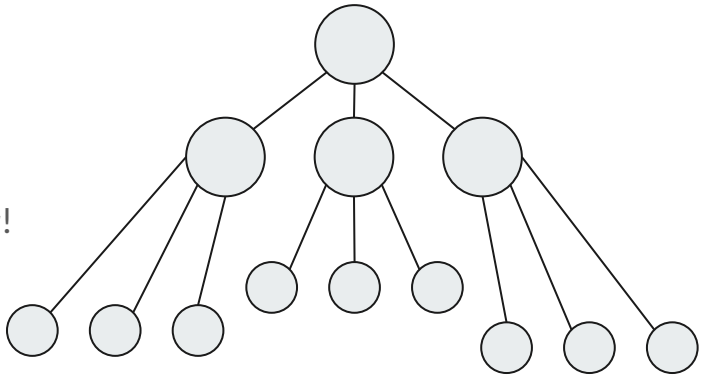
- Recurse (decision)
  - **IF choices are not valid THEN Do first base case**
  - IF decision is at the end THEN Do Base Case
  - FOR each choice in decision
    - Try choice
    - Recurse (decision + 1)

Backtracking!

- Recurse (decision)
  - **IF choices are not valid THEN Do first base case**
  - IF decision is at the end THEN Do Base Case
  - FOR each choice in decision
    - Try choice
    - Recurse (decision + 1)
- The earlier we make an invalid decision the better!

Backtracking!

- Recurse (decision)
  - **IF choices are not valid THEN Do first base case**
  - IF decision is at the end THEN Do Base Case
  - FOR each choice in decision
    - Try choice
    - Recurse (decision + 1)
- The earlier we make an invalid decision the better!



Sudoku Coding Time!

Other Problems

Other Problems

- N Queens

Other Problems

- N Queens
  - Place N queens on a N by N chess board

Other Problems

- N Queens
  - Place N queens on a N by N chess board
  - No two queens can attack each other

Other Problems

- N Queens
  - Place N queens on a N by N chess board
  - No two queens can attack each other

Other Problems

- N Queens
  - Place N queens on a N by N chess board
  - No two queens can attack each other
- What are the decisions?

Other Problems

- N Queens
  - Place N queens on a N by N chess board
  - No two queens can attack each other
- What are the decisions?
- What are the choices?

Other Problems

- N Queens
  - Place N queens on a N by N chess board
  - No two queens can attack each other
- What are the decisions?
- What are the choices?
- Note that each row contains exactly 1 queen!

Other Problems

- N Queens
  - Place N queens on a N by N chess board
  - No two queens can attack each other
- What are the decisions?
- What are the choices?
- Note that each row contains exactly 1 queen!
  - Select the position for the queen in a given row starting with the first

Other Problems

- N Queens
  - Place N queens on a N by N chess board
  - No two queens can attack each other
- What are the decisions?
- What are the choices?
- Note that each row contains exactly 1 queen!
  - Select the position for the queen in a given row starting with the first
- No two queens have the same column

Other Problems

- N Queens
  - Place N queens on a N by N chess board
  - No two queens can attack each other
- What are the decisions?
- What are the choices?
- Note that each row contains exactly 1 queen!
  - Select the position for the queen in a given row starting with the first
- No two queens have the same column
  - They form a permutation

Other Problems

- N Queens
  - Place N queens on a N by N chess board
  - No two queens can attack each other
- What are the decisions?
- What are the choices?
- Note that each row contains exactly 1 queen!
  - Select the position for the queen in a given row starting with the first
- No two queens have the same column
  - They form a permutation
- How do we check for validity?

Other Problems

- N Queens
  - Place N queens on a N by N chess board
  - No two queens can attack each other
- What are the decisions?
- What are the choices?
- Note that each row contains exactly 1 queen!
  - Select the position for the queen in a given row starting with the first
- No two queens have the same column
  - They form a permutation
- How do we check for validity?
  - A validity function is easiest

Other Problems

- N Queens
  - Place N queens on a N by N chess board
  - No two queens can attack each other
- What are the decisions?
- What are the choices?
- Note that each row contains exactly 1 queen!
  - Select the position for the queen in a given row starting with the first
- No two queens have the same column
  - They form a permutation
- How do we check for validity?
  - A validity function is easiest
- Note this exact problem will be part of the first lab group