Solving Problems by Searching

Dr. Steven Bethard

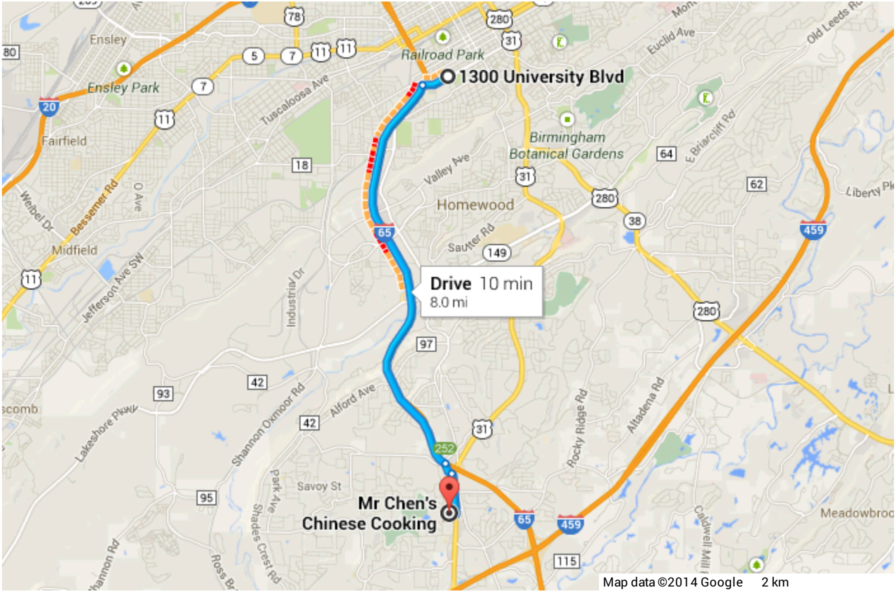
Computer and Information Sciences
University of Alabama at Birmingham

14 Jan 2016

Outline

1. Search Problems
   - Describing Search Problems
   - Search Trees and Search Nodes
2. Uninformed Search Strategies
   - Breadth-first Search
   - Uniform-cost Search
   - Depth-first Search
   - Iterative Deepening Search
3. Informed Search
   - Best-First Search
   - A* Search
   - Heuristic Functions

Outline


1300 University Blvd
Drive 10 min
8.0 mi
Mr Chen's
Chinese Cooking
Railroad Park
Birmingham
Botanical Gardens
Homewood
Ensley
Ensley Park
Fairfield
Midfield
Meadowbrook
Map data ©2014 Google 2 km

Properties of Search Problems

- Fully or Partially Observable?
- Deterministic or Stochastic?
- Episodic or Sequential?
- Static or Dynamic?
- Discrete or Continuous?
- Single or Multi-Agent?

Properties of Search Problems

- Fully or Partially Observable?
- Deterministic or Stochastic?
- Episodic or Sequential?
- Static or Dynamic?
- Discrete or Continuous?
- Single or Multi-Agent?

Properties of Search Problems

- Fully or Partially Observable?
- Deterministic or Stochastic?
- Episodic or Sequential?
- Static or Dynamic?
- Discrete or Continuous?
- Single or Multi-Agent?

Properties of Search Problems

- Fully or Partially Observable?
- Deterministic or Stochastic?
- Episodic or Sequential?
- Static or Dynamic?
- Discrete or Continuous?
- Single or Multi-Agent?

Properties of Search Problems

- Fully or Partially Observable?
- Deterministic or Stochastic?
- Episodic or Sequential?
- Static or Dynamic?
- Discrete or Continuous?
- Single or Multi-Agent?

Properties of Search Problems

- Fully or Partially Observable?
- Deterministic or Stochastic?
- Episodic or Sequential?
- Static or Dynamic?
- Discrete or Continuous?
- Single or Multi-Agent?

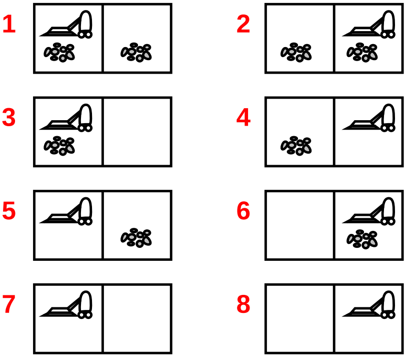
Properties of Search Problems

- Fully or Partially Observable?
- Deterministic or Stochastic?
- Episodic or Sequential?
- Static or Dynamic?
- Discrete or Continuous?
- Single or Multi-Agent?

Example: Vacuum Problem

Problem

- Start in **1**
- Left square actions: **SUCK** or **RIGHT**
- Right square actions: **SUCK** or **LEFT**
- Success: **7** or **8**
- Optimal: fewest actions



Solution

Example: Vacuum Problem

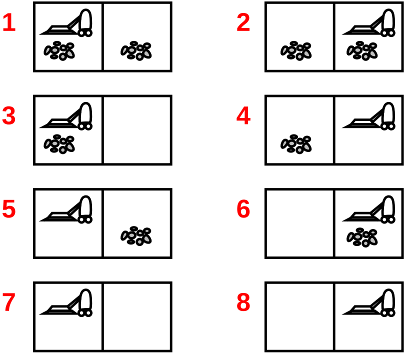
Problem

- Start in **1**
- Left square actions: **Suck** or **Right**
- Right square actions: **Suck** or **Left**
- Success: **7** or **8**
- Optimal: fewest actions

Solution

[Suck, Right, Suck]

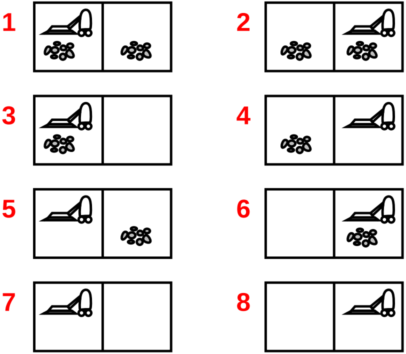
i.e., [1, 5, 6, 8]



Example: Vacuum Problem

Problem

- Start in **1**
- Left square actions: **Suck** or **Right**
- Right square actions: **Suck** or **Left**
- Success: **7** or **8**
- Optimal: fewest actions



Solution

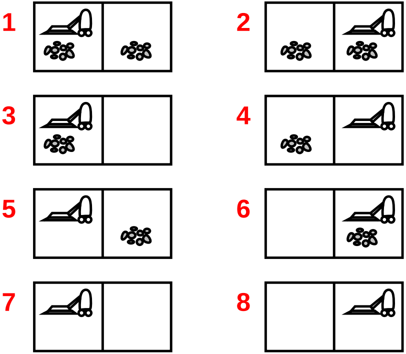
[**Suck**, **Right**, **Suck**]

i.e., [1, 5, 6, 8]

Example: Vacuum Problem

Problem

- Start in **1**
- Left square actions: **SUCK** or **RIGHT**
- Right square actions: **SUCK** or **LEFT**
- Success: **7** or **8**
- Optimal: fewest actions



Solution

[**SUCK**, **RIGHT**, **SUCK**]
i.e., [**1**, **5**, **6**, **8**]

Defining a Search Problem

Components

- Initial state
- Actions
- Goal test
- Path cost

Solution

Path from initial state to goal state

Optimal solution

Path with lowest cost

Defining the Vacuum Problem

Initial State

1

Actions

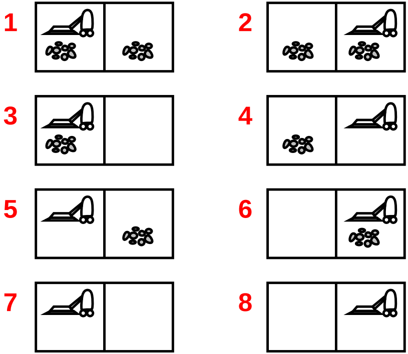
$S(1) = \{(\textsc{Right}, 2), (\textsc{Suck}, 5)\}$
$S(2) = \{(\textsc{Left}, 1), (\textsc{Suck}, 4)\}$
...

Goal Test

$G(s) = s \in \{7, 8\}$

Path Cost

1 per state



Defining the 8-Puzzle Problem

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

States Mappings of tile numbers to tile locations

Initial {1:9, 2:2, 3:8, 4:3, 5:4, 6:6, 7:1, 8:7}

Actions Move blank left, right, up, down

Goal {1:1, 2:2, 3:3, 4:4, 5:5, 6:6, 7:7, 8:8}

Cost 1 per move

Defining the 8-Puzzle Problem

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

States	Mappings of tile numbers to tile locations
Initial	{1:9, 2:2, 3:8, 4:3, 5:4, 6:6, 7:1, 8:7}
Actions	Move blank left, right, up, down
Goal	{1:1, 2:2, 3:3, 4:4, 5:5, 6:6, 7:7, 8:8}
Cost	1 per move

Defining the 8-Puzzle Problem

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

States	Mappings of tile numbers to tile locations
Initial	{1 : 9, 2 : 2, 3 : 8, 4 : 3, 5 : 4, 6 : 6, 7 : 1, 8 : 7}
Actions	Move blank left, right, up, down
Goal	{1 : 1, 2 : 2, 3 : 3, 4 : 4, 5 : 5, 6 : 6, 7 : 7, 8 : 8}
Cost	1 per move

Defining the 8-Puzzle Problem

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

States Mappings of tile numbers to tile locations

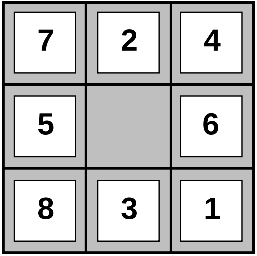
Initial $\{1:9, 2:2, 3:8, 4:3, 5:4, 6:6, 7:1, 8:7\}$

Actions Move blank left, right, up, down

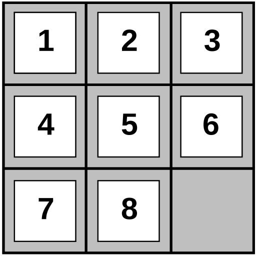
Goal $\{1:1, 2:2, 3:3, 4:4, 5:5, 6:6, 7:7, 8:8\}$

Cost 1 per move

Defining the 8-Puzzle Problem



Start State



Goal State

States Mappings of tile numbers to tile locations

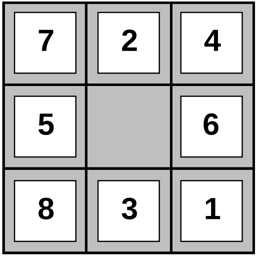
Initial $\{1\!:\!9, 2\!:\!2, 3\!:\!8, 4\!:\!3, 5\!:\!4, 6\!:\!6, 7\!:\!1, 8\!:\!7\}$

Actions Move blank left, right, up, down

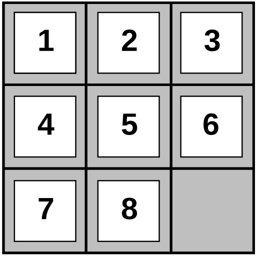
Goal $\{1\!:\!1, 2\!:\!2, 3\!:\!3, 4\!:\!4, 5\!:\!5, 6\!:\!6, 7\!:\!7, 8\!:\!8\}$

Cost 1 per move

Defining the 8-Puzzle Problem



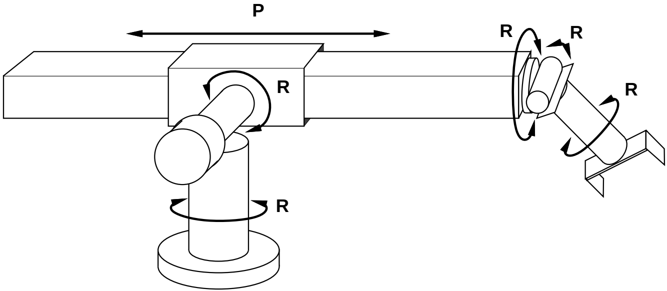
Start State



Goal State

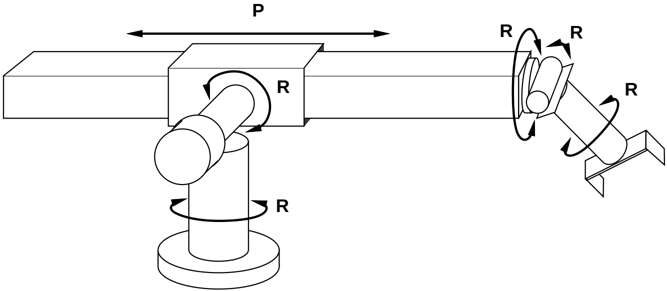
States	Mappings of tile numbers to tile locations
Initial	$\{1\!:\!9, 2\!:\!2, 3\!:\!8, 4\!:\!3, 5\!:\!4, 6\!:\!6, 7\!:\!1, 8\!:\!7\}$
Actions	Move blank left, right, up, down
Goal	$\{1\!:\!1, 2\!:\!2, 3\!:\!3, 4\!:\!4, 5\!:\!5, 6\!:\!6, 7\!:\!7, 8\!:\!8\}$
Cost	1 per move

Defining a Robotic Assembly Problem



States	real-valued joint angles, parts to assemble
Initial	many possible
Actions	real-valued adjustments to joint angles
Goal	fully assembled part
Cost	total duration of all movements

Defining a Robotic Assembly Problem



States real-valued joint angles, parts to assemble

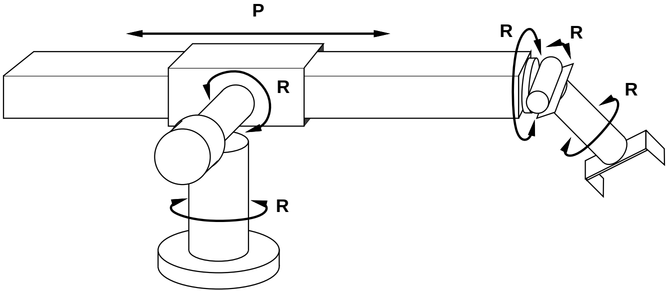
Initial many possible

Actions real-valued adjustments to joint angles

Goal fully assembled part

Cost total duration of all movements

Defining a Robotic Assembly Problem



States real-valued joint angles, parts to assemble

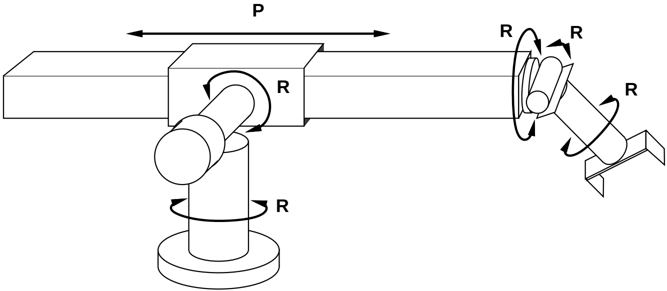
Initial many possible

Actions real-valued adjustments to joint angles

Goal fully assembled part

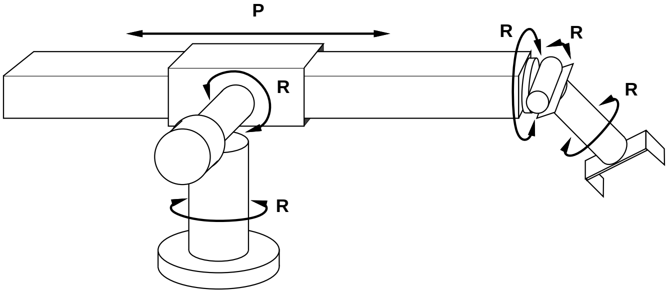
Cost total duration of all movements

Defining a Robotic Assembly Problem



States	real-valued joint angles, parts to assemble
Initial	many possible
Actions	real-valued adjustments to joint angles
Goal	fully assembled part
Cost	total duration of all movements

Defining a Robotic Assembly Problem



States real-valued joint angles, parts to assemble

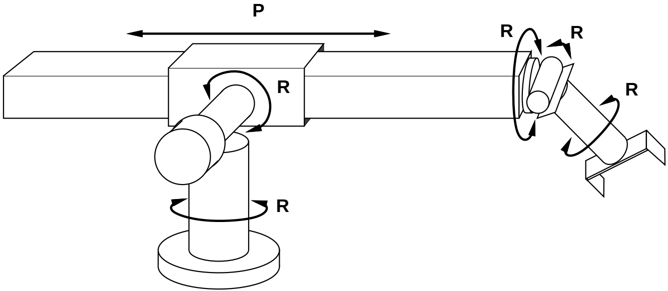
Initial many possible

Actions real-valued adjustments to joint angles

Goal fully assembled part

Cost total duration of all movements

Defining a Robotic Assembly Problem



States	real-valued joint angles, parts to assemble
Initial	many possible
Actions	real-valued adjustments to joint angles
Goal	fully assembled part
Cost	total duration of all movements

Defining a Machine Translation Problem

Input	Comí la manzana roja porque tenía hambre.
Output	I ate the red apple because I was hungry.

States

Initial

Actions

Goal

Cost

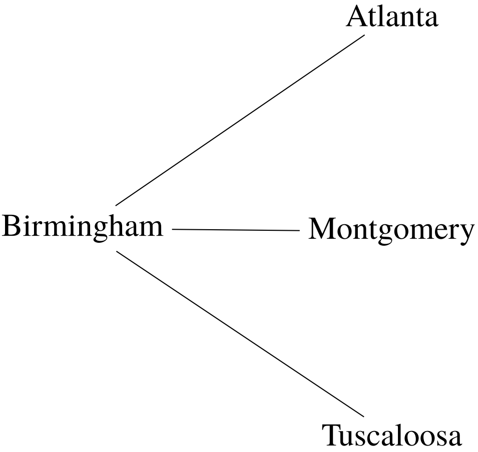
Tree Search

```
def tree_search(problem, strategy):

    strategy.add(...problem.initial_state...)

    for node in strategy:

        if problem.is_goal(node.state):
            return node.get_actions()

        items = problem.get_successors(node.state)
        for state, action, ... in items:
            strategy.add(...state...action...)

    return None
```

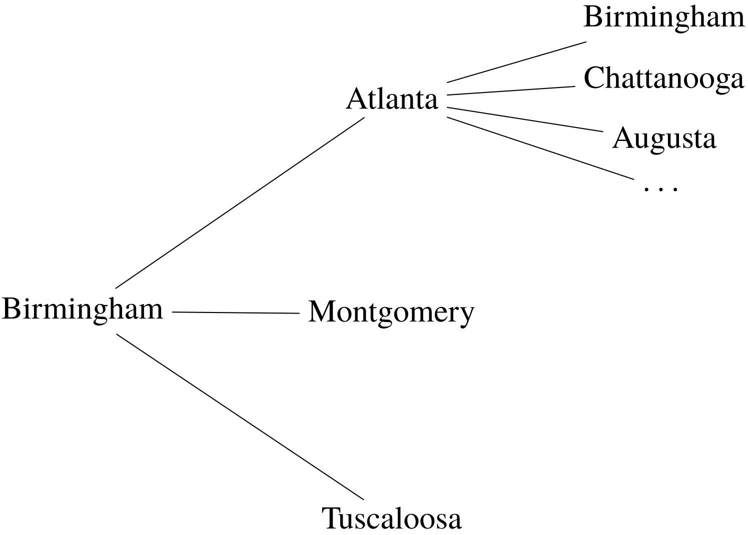
Search Tree Example

Birmingham

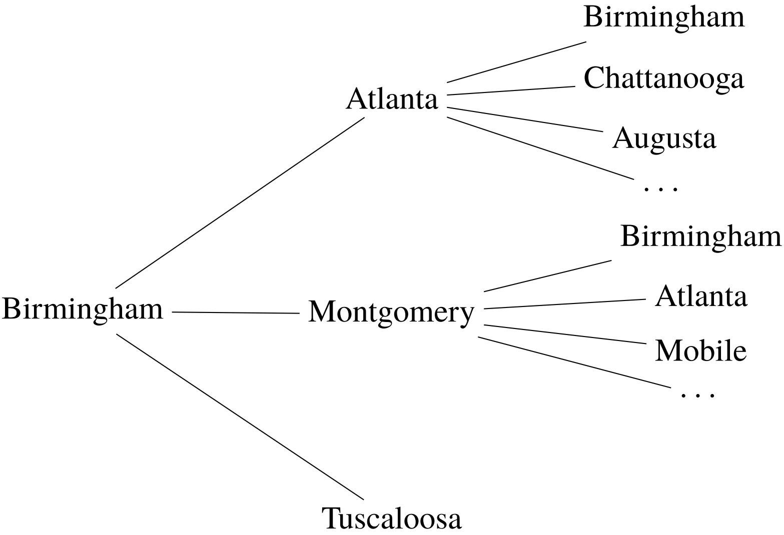
Search Tree Example



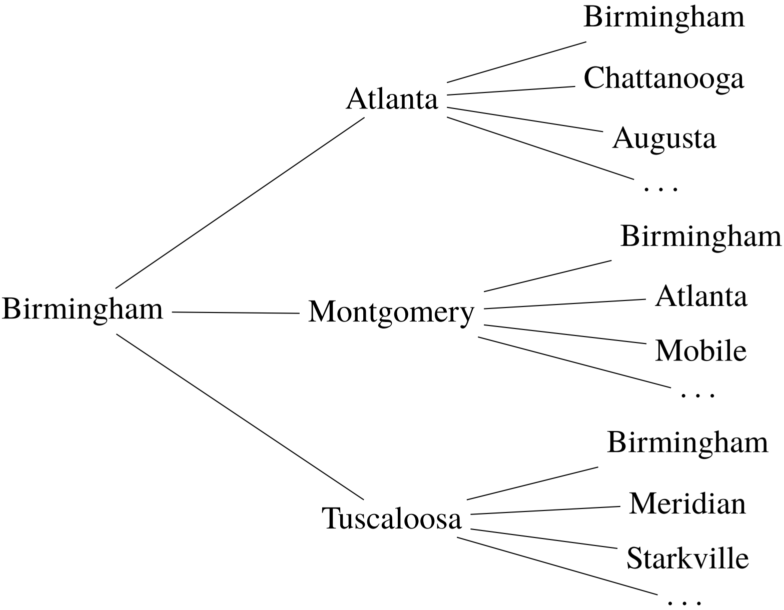
Search Tree Example



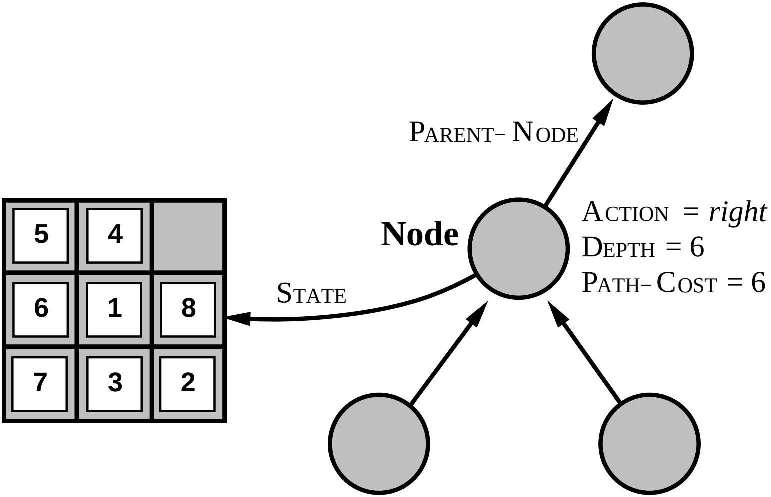
Search Tree Example



Search Tree Example



The Node Data Structure



Tree Search Revisited

```python
def tree_search(problem, strategy):
    strategy.add(Node(problem.initial_state))
    for node in strategy:
        if problem.is_goal(node.state):
            return node.get_actions()
        succs = problem.get_successors(node.state)
        for state, action, cost in succs:
            strategy.add(Node(
                state=state,
                action=action,
                parent=node,
                cost=node.cost + cost,
                depth=node.depth + 1))
    return None
```

Outline

1. Search Problems
   - Describing Search Problems
   - Search Trees and Search Nodes
2. Uninformed Search Strategies
   - Breadth-first Search
   - Uniform-cost Search
   - Depth-first Search
   - Iterative Deepening Search
3. Informed Search
   - Best-First Search
   - A* Search
   - Heuristic Functions

Search Strategy Properties

Completeness

If a solution exists, is it always found?

Optimality

Is the solution found always the lowest cost?

Time Complexity

How many search nodes will be generated?

Space Complexity

How many search nodes must stay in main memory?

Search Strategy Properties

Completeness

If a solution exists, is it always found?

Optimality

Is the solution found always the lowest cost?

Time Complexity

How many search nodes will be generated?

Space Complexity

How many search nodes must stay in main memory?

Search Strategy Properties

Completeness

If a solution exists, is it always found?

Optimality

Is the solution found always the lowest cost?

Time Complexity

How many search nodes will be generated?

Space Complexity

How many search nodes must stay in main memory?

Search Strategy Properties

Completeness

If a solution exists, is it always found?

Optimality

Is the solution found always the lowest cost?

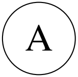
Time Complexity

How many search nodes will be generated?

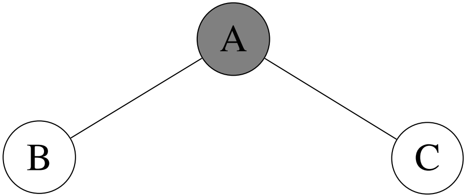
Space Complexity

How many search nodes must stay in main memory?

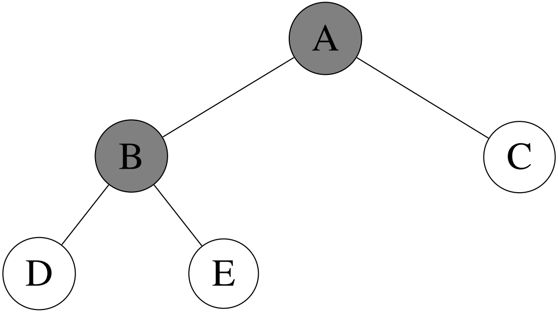
Breadth-first Search



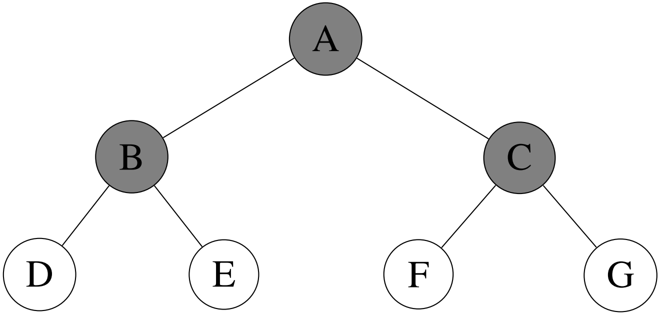
Breadth-first Search



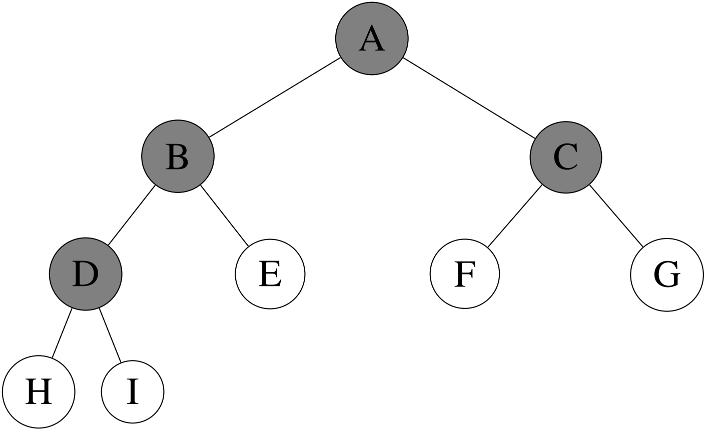
Breadth-first Search



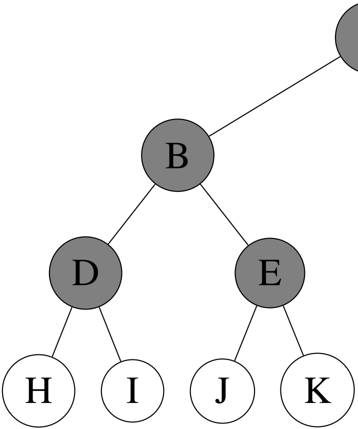
Breadth-first Search



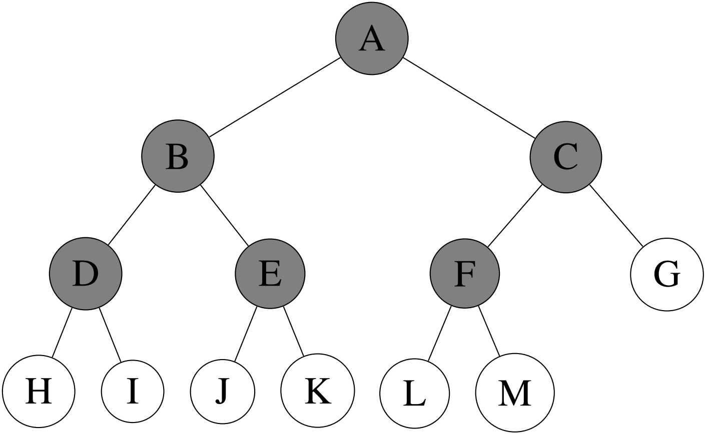
Breadth-first Search



Breadth-first Search



Breadth-first Search



Breadth-first Properties

Strategy? ▸ Example

First-in First-Out Queue

Complete?

Yes, if number of branches is finite

Optimal?

Yes, if step costs are all identical

Worst Case Time Complexity?

$O(b^{d+1})$, branching factor b, depth of goal state d

Worst Case Space Complexity?

$O(b^{d+1})$, branching factor b, depth of goal state d

Breadth-first Properties

Strategy? ▸ Example

First-in First-Out Queue

Complete?

Yes, if number of branches is finite

Optimal?

Yes, if step costs are all identical

Worst Case Time Complexity?

$O(b^{d+1})$, branching factor b, depth of goal state d

Worst Case Space Complexity?

$O(b^{d+1})$, branching factor b, depth of goal state d

Breadth-first Properties

Strategy? ▸ Example

First-in First-Out Queue

Complete?

Yes, if number of branches is finite

Optimal?

Yes, if step costs are all identical

Worst Case Time Complexity?

$O(b^{d+1})$, branching factor b, depth of goal state d

Worst Case Space Complexity?

$O(b^{d+1})$, branching factor b, depth of goal state d

Breadth-first Properties

Strategy? ▸ Example

First-in First-Out Queue

Complete?

Yes, if number of branches is finite

Optimal?

Yes, if step costs are all identical

Worst Case Time Complexity?

$O(b^{d+1})$, branching factor b, depth of goal state d

Worst Case Space Complexity?

$O(b^{d+1})$, branching factor b, depth of goal state d

Breadth-first Properties

Strategy?

Example

First-in First-Out Queue

Complete?

Yes, if number of branches is finite

Optimal?

Yes, if step costs are all identical

Worst Case Time Complexity?

$O(b^{d+1})$, branching factor b, depth of goal state d

Worst Case Space Complexity?

$O(b^{d+1})$, branching factor b, depth of goal state d

Breadth-first Properties

Strategy?

Example

First-in First-Out Queue

Complete?

Yes, if number of branches is finite

Optimal?

Yes, if step costs are all identical

Worst Case Time Complexity?

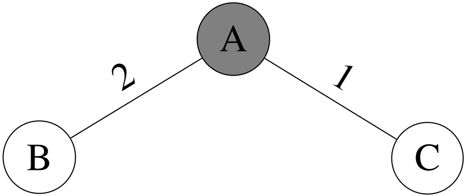
$O(b^{d+1})$, branching factor b, depth of goal state d

Worst Case Space Complexity?

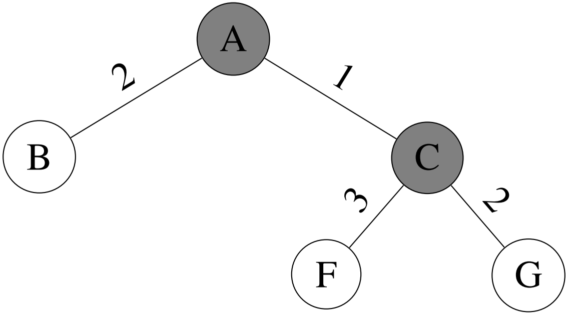
$O(b^{d+1})$, branching factor b, depth of goal state d

Uniform-cost Search

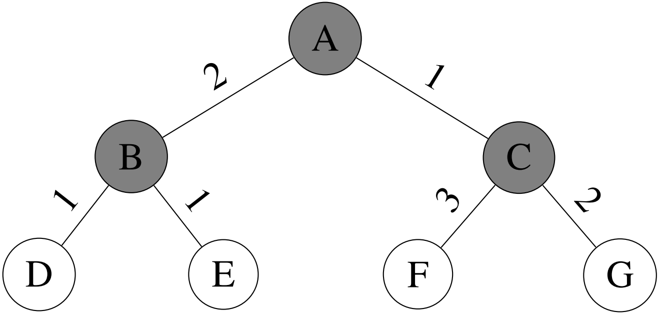
Uniform-cost Search



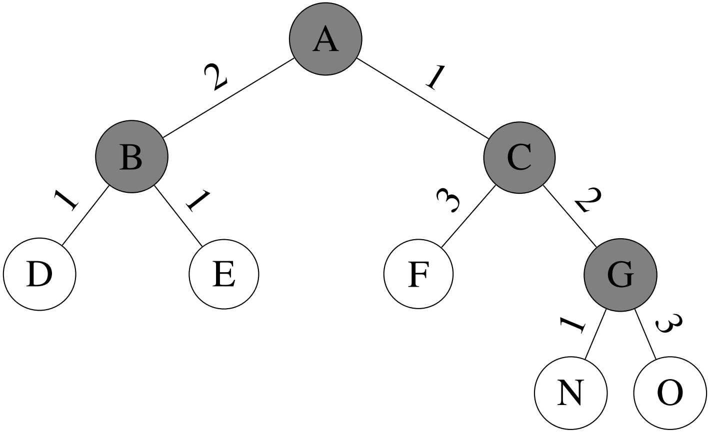
Uniform-cost Search



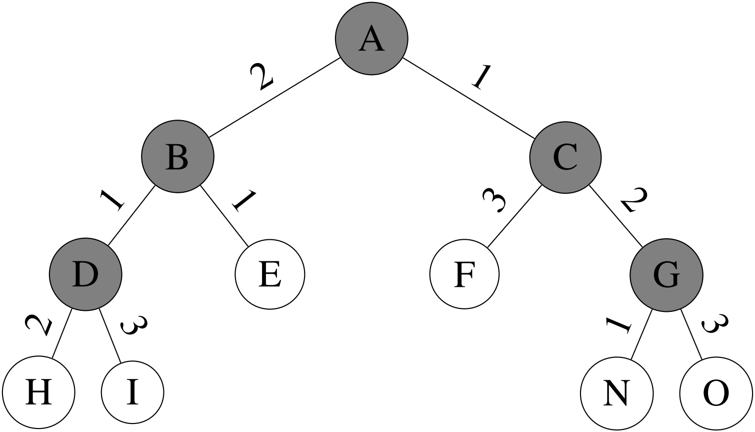
Uniform-cost Search



Uniform-cost Search



Uniform-cost Search



Uniform-cost Search

Strategy? Example

Lowest Cost First Priority Queue

Complete?

Yes, if number of branches is finite and steps are all positive

Optimal?

Yes

Worst Case Time Complexity?

$O(b^{\lfloor C^*/\epsilon \rfloor + 1})$, optimal cost C^*, minimum step cost ϵ

Worst Case Space Complexity?

$O(b^{\lfloor C^*/\epsilon \rfloor + 1})$, optimal cost C^*, minimum step cost ϵ

Uniform-cost Search

Strategy? ▸ Example

Lowest Cost First Priority Queue

Complete?

Yes, if number of branches is finite and steps are all positive

Optimal?

Yes

Worst Case Time Complexity?

$O(b^{\lfloor C^*/\epsilon \rfloor + 1})$, optimal cost C^*, minimum step cost ϵ

Worst Case Space Complexity?

$O(b^{\lfloor C^*/\epsilon \rfloor + 1})$, optimal cost C^*, minimum step cost ϵ

Uniform-cost Search

Strategy? ▸ Example

Lowest Cost First Priority Queue

Complete?

Yes, if number of branches is finite and steps are all positive

Optimal?

Yes

Worst Case Time Complexity?

$O(b^{\lfloor C^*/\epsilon \rfloor + 1})$, optimal cost C^*, minimum step cost ϵ

Worst Case Space Complexity?

$O(b^{\lfloor C^*/\epsilon \rfloor + 1})$, optimal cost C^*, minimum step cost ϵ

Uniform-cost Search

Strategy? Example

Lowest Cost First Priority Queue

Complete?

Yes, if number of branches is finite and steps are all positive

Optimal?

Yes

Worst Case Time Complexity?

$O(b^{\lfloor C^*/\epsilon \rfloor + 1})$, optimal cost C^*, minimum step cost ϵ

Worst Case Space Complexity?

$O(b^{\lfloor C^*/\epsilon \rfloor + 1})$, optimal cost C^*, minimum step cost ϵ

Uniform-cost Search

Strategy? ▸ Example

Lowest Cost First Priority Queue

Complete?

Yes, if number of branches is finite and steps are all positive

Optimal?

Yes

Worst Case Time Complexity?

$O(b^{\lfloor C^*/\epsilon \rfloor + 1})$, optimal cost C^*, minimum step cost ϵ

Worst Case Space Complexity?

$O(b^{\lfloor C^*/\epsilon \rfloor + 1})$, optimal cost C^*, minimum step cost ϵ

Uniform-cost Search

Strategy? ‣ Example

Lowest Cost First Priority Queue

Complete?

Yes, if number of branches is finite and steps are all positive

Optimal?

Yes

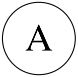
Worst Case Time Complexity?

$O(b^{\lfloor C^*/\epsilon \rfloor + 1})$, optimal cost C^*, minimum step cost ϵ

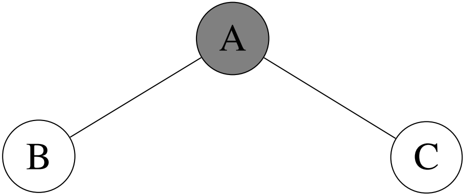
Worst Case Space Complexity?

$O(b^{\lfloor C^*/\epsilon \rfloor + 1})$, optimal cost C^*, minimum step cost ϵ

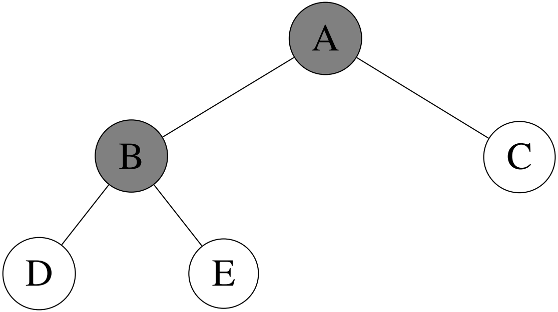
Depth-first Search



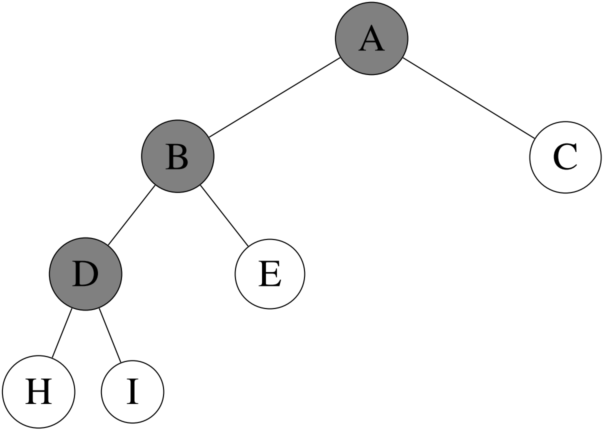
Depth-first Search



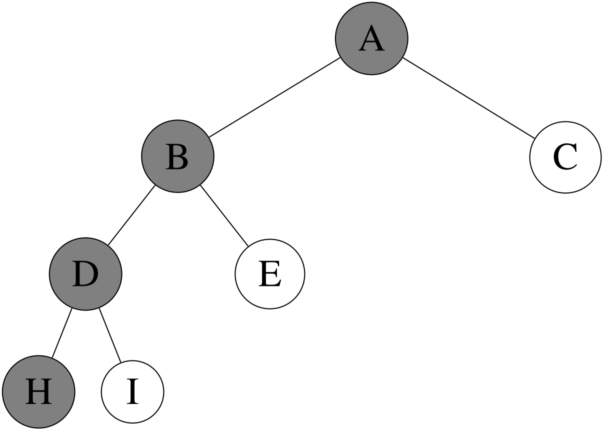
Depth-first Search



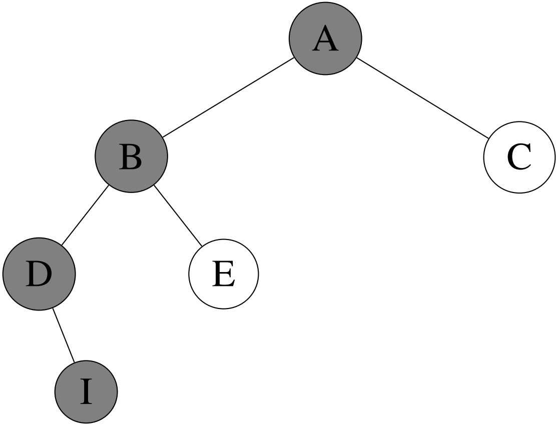
Depth-first Search



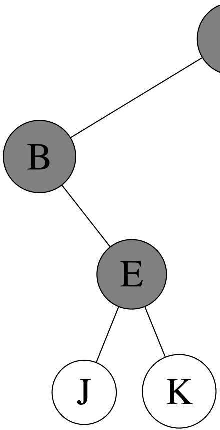
Depth-first Search



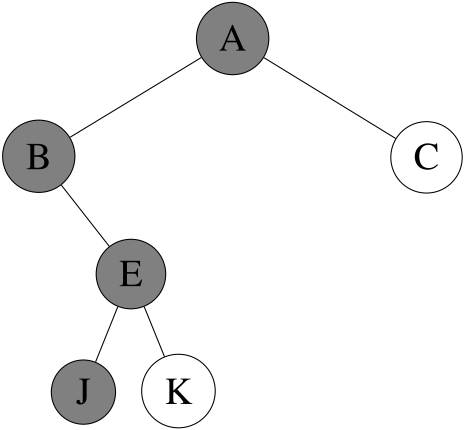
Depth-first Search



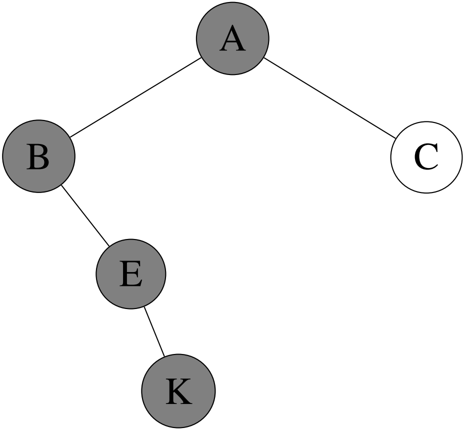
Depth-first Search



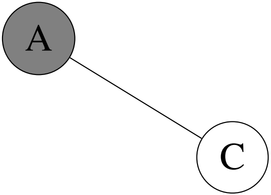
Depth-first Search



Depth-first Search



Depth-first Search



Depth-first Properties

Strategy? ▸ Example

First-in Last-Out Stack

Complete?

Yes, if finite number of states and no cyclic paths

Optimal?

No, lower goal states may be found first

Worst Case Time Complexity?

$O(b^m)$, branching factor b, maximum depth m

Worst Case Space Complexity?

$O(bm)$, branching factor b, maximum depth m

Depth-first Properties

Strategy? Example

First-in Last-Out Stack

Complete?

Yes, if finite number of states and no cyclic paths

Optimal?

No, lower goal states may be found first

Worst Case Time Complexity?

$O(b^m)$, branching factor b, maximum depth m

Worst Case Space Complexity?

$O(bm)$, branching factor b, maximum depth m

Depth-first Properties

Strategy? Example

First-in Last-Out Stack

Complete?

Yes, if finite number of states and no cyclic paths

Optimal?

No, lower goal states may be found first

Worst Case Time Complexity?

$O(b^m)$, branching factor b, maximum depth m

Worst Case Space Complexity?

$O(bm)$, branching factor b, maximum depth m

Depth-first Properties

Strategy? ‣ Example

First-in Last-Out Stack

Complete?

Yes, if finite number of states and no cyclic paths

Optimal?

No, lower goal states may be found first

Worst Case Time Complexity?

$O(b^m)$, branching factor b, maximum depth m

Worst Case Space Complexity?

$O(bm)$, branching factor b, maximum depth m

Depth-first Properties

Strategy? ▸ Example

First-in Last-Out Stack

Complete?

Yes, if finite number of states and no cyclic paths

Optimal?

No, lower goal states may be found first

Worst Case Time Complexity?

$O(b^m)$, branching factor b, maximum depth m

Worst Case Space Complexity?

$O(bm)$, branching factor b, maximum depth m

Depth-first Properties

Strategy? ▸ Example

First-in Last-Out Stack

Complete?

Yes, if finite number of states and no cyclic paths

Optimal?

No, lower goal states may be found first

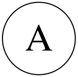
Worst Case Time Complexity?

$O(b^m)$, branching factor b, maximum depth m

Worst Case Space Complexity?

$O(bm)$, branching factor b, maximum depth m

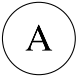
Iterative Deepening Search



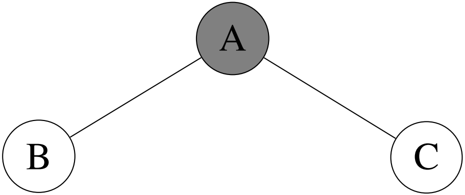
Iterative Deepening Search

Iterative Deepening Search

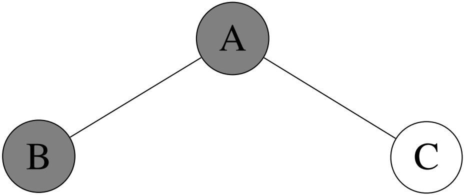
Iterative Deepening Search



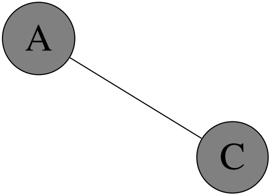
Iterative Deepening Search



Iterative Deepening Search

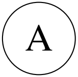


Iterative Deepening Search

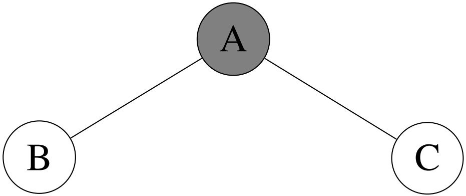


Iterative Deepening Search

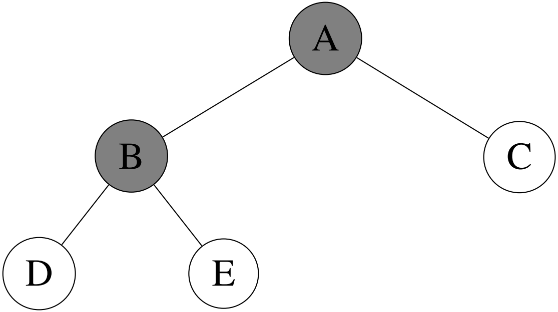
Iterative Deepening Search



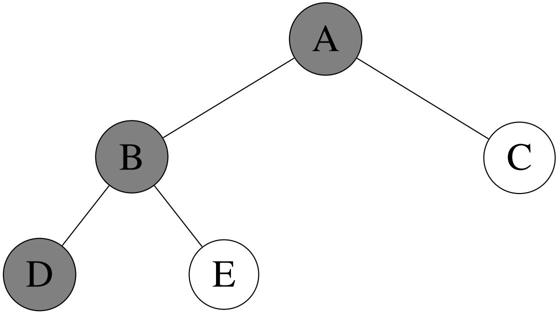
Iterative Deepening Search



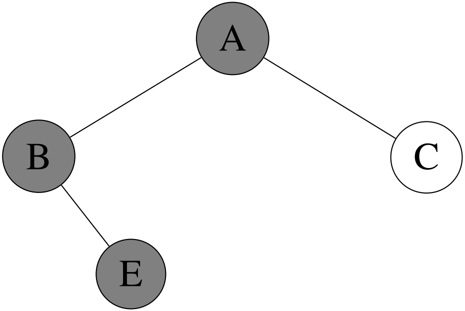
Iterative Deepening Search



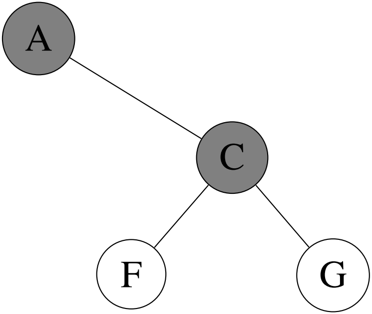
Iterative Deepening Search



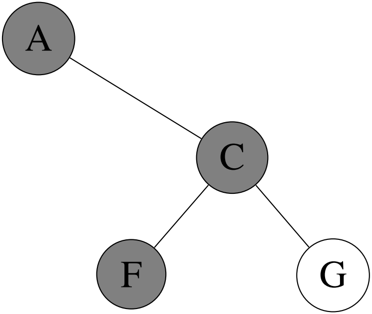
Iterative Deepening Search



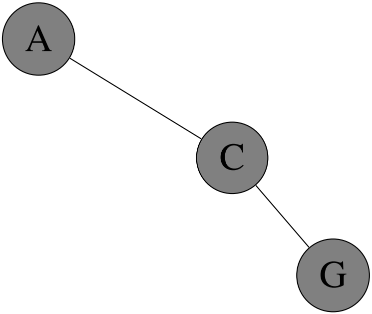
Iterative Deepening Search



Iterative Deepening Search

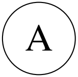


Iterative Deepening Search



Iterative Deepening Search

Iterative Deepening Search



Iterative Deepening Properties

Strategy? ▸ Example

First-in Last-Out Stack with depth limit

Complete?

Yes, if number of branches is finite

Optimal?

Yes, if step costs are all identical

Worst Case Time Complexity?

$O(b^d)$, branching factor b, depth of goal state d

Worst Case Space Complexity?

$O(bd)$, branching factor b, depth of goal state d

Iterative Deepening Properties

Strategy? ▸ Example

First-in Last-Out Stack with depth limit

Complete?

Yes, if number of branches is finite

Optimal?

Yes, if step costs are all identical

Worst Case Time Complexity?

$O(b^d)$, branching factor b, depth of goal state d

Worst Case Space Complexity?

$O(bd)$, branching factor b, depth of goal state d

Iterative Deepening Properties

Strategy? ▸ Example

First-in Last-Out Stack with depth limit

Complete?

Yes, if number of branches is finite

Optimal?

Yes, if step costs are all identical

Worst Case Time Complexity?

$O(b^d)$, branching factor b, depth of goal state d

Worst Case Space Complexity?

$O(bd)$, branching factor b, depth of goal state d

Iterative Deepening Properties

Strategy? ‣ Example

First-in Last-Out Stack with depth limit

Complete?

Yes, if number of branches is finite

Optimal?

Yes, if step costs are all identical

Worst Case Time Complexity?

$O(b^d)$, branching factor b, depth of goal state d

Worst Case Space Complexity?

$O(bd)$, branching factor b, depth of goal state d

Iterative Deepening Properties

Strategy?

▸ Example

First-in Last-Out Stack with depth limit

Complete?

Yes, if number of branches is finite

Optimal?

Yes, if step costs are all identical

Worst Case Time Complexity?

$O(b^d)$, branching factor b, depth of goal state d

Worst Case Space Complexity?

$O(bd)$, branching factor b, depth of goal state d

Iterative Deepening Properties

Strategy? ▸ Example

First-in Last-Out Stack with depth limit

Complete?

Yes, if number of branches is finite

Optimal?

Yes, if step costs are all identical

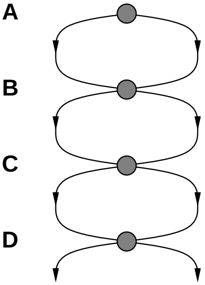
Worst Case Time Complexity?

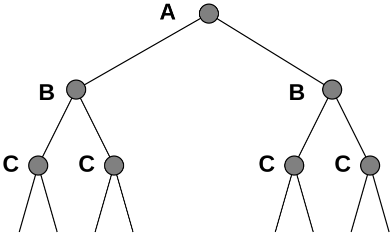
$O(b^d)$, branching factor b, depth of goal state d

Worst Case Space Complexity?

$O(bd)$, branching factor b, depth of goal state d

Exponential Costs of Repeated States



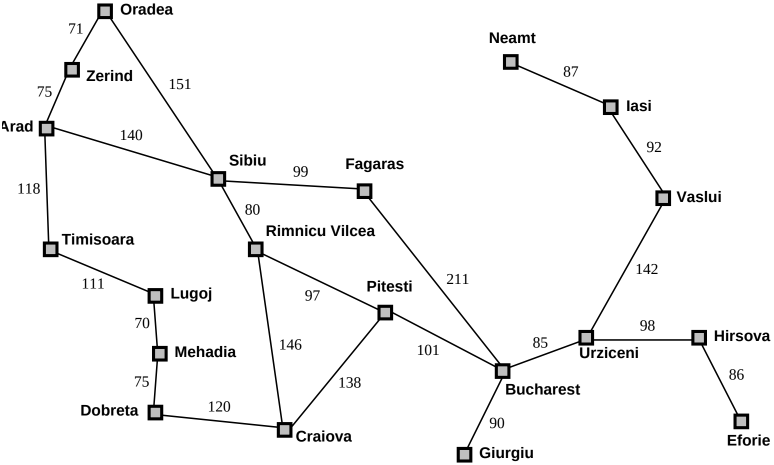


Graph Search

```python
def graph_search(problem, strategy):
    seen = set()
    strategy.add(Node(problem.initial_state))
    for node in strategy:
        if problem.is_goal(node.state):
            return node.get_actions()
        if node not in seen:
            seen.add(node)
            succs = problem.get_successors(node.state)
            for state, action, cost in succs:
                strategy.add(Node(
                    state=state,
                    action=action,
                    parent=node,
                    cost=node.cost + cost,
                    depth=node.depth + 1))
```

Outline

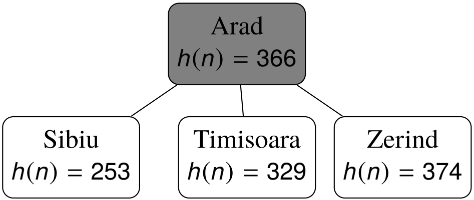
All States are not Equal



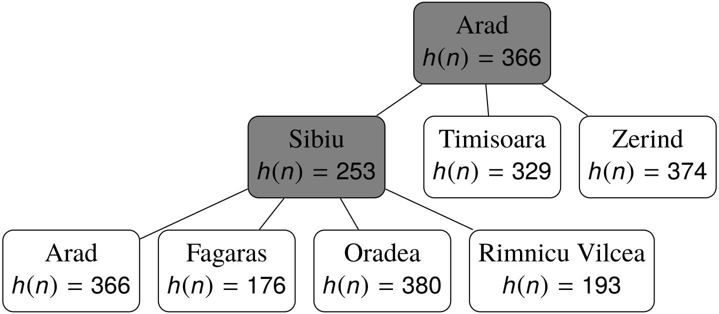
Best-First (Greedy) Search

Arad
$h(n) = 366$

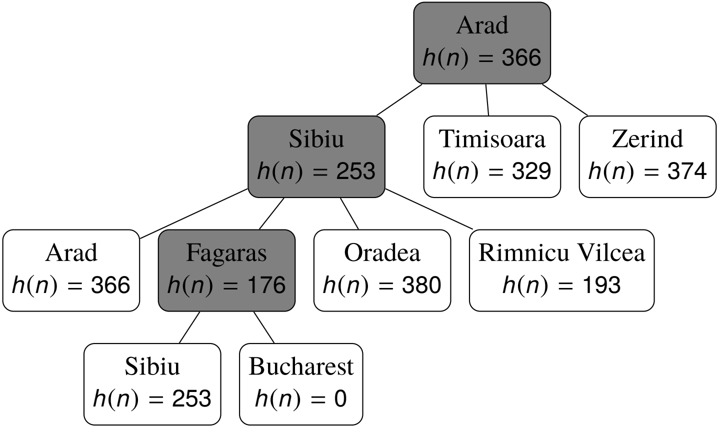
Best-First (Greedy) Search



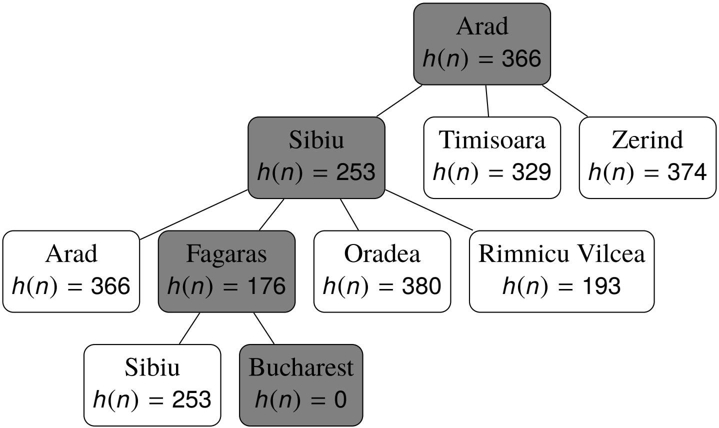
Best-First (Greedy) Search



Best-First (Greedy) Search



Best-First (Greedy) Search



Best-First (Greedy) Properties

Strategy?

Priority Queue, $f(n) = h(n)$

Complete?

Yes, if finite number of states and no cyclic paths

Optimal?

No

Worst Case Time Complexity?

$O(b^m)$, but better with good heuristic

Worst Case Space Complexity?

$O(b^m)$

Best-First (Greedy) Properties

Strategy?

Priority Queue, $f(n) = h(n)$

Complete?

Yes, if finite number of states and no cyclic paths

Optimal?

No

Worst Case Time Complexity?

$O(b^m)$, but better with good heuristic

Worst Case Space Complexity?

$O(b^m)$

Best-First (Greedy) Properties

Strategy?

Priority Queue, $f(n) = h(n)$

Complete?

Yes, if finite number of states and no cyclic paths

Optimal?

No

Worst Case Time Complexity?

$O(b^m)$, but better with good heuristic

Worst Case Space Complexity?

$O(b^m)$

Best-First (Greedy) Properties

Strategy?

Priority Queue, $f(n) = h(n)$

Complete?

Yes, if finite number of states and no cyclic paths

Optimal?

No

Worst Case Time Complexity?

$O(b^m)$, but better with good heuristic

Worst Case Space Complexity?

$O(b^m)$

Best-First (Greedy) Properties

Strategy?

Priority Queue, $f(n) = h(n)$

Complete?

Yes, if finite number of states and no cyclic paths

Optimal?

No

Worst Case Time Complexity?

$O(b^m)$, but better with good heuristic

Worst Case Space Complexity?

$O(b^m)$

Best-First (Greedy) Properties

Strategy?

Priority Queue, $f(n) = h(n)$

Complete?

Yes, if finite number of states and no cyclic paths

Optimal?

No

Worst Case Time Complexity?

$O(b^m)$, but better with good heuristic

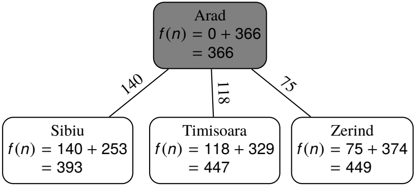
Worst Case Space Complexity?

$O(b^m)$

A* Search

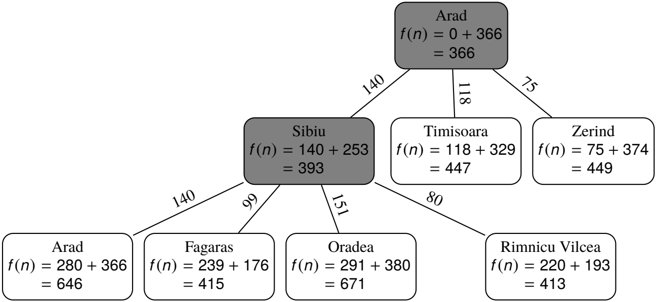
Arad
$f(n) = 0 + 366$
$= 366$

A* Search



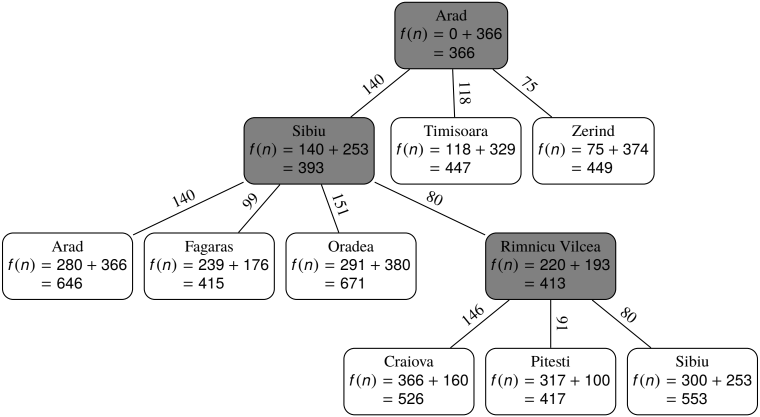
Arad
$f(n) = 0 + 366$
$= 366$
140
118
75
Sibiu
$f(n) = 140 + 253$
$= 393$
Timisoara
$f(n) = 118 + 329$
$= 447$
Zerind
$f(n) = 75 + 374$
$= 449$

A* Search

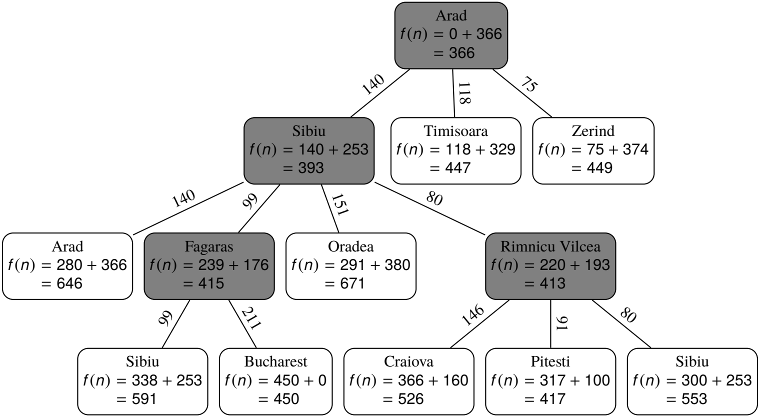


Arad
$f(n) = 0 + 366$
$= 366$
140
118
75
Sibiu
$f(n) = 140 + 253$
$= 393$
Timisoara
$f(n) = 118 + 329$
$= 447$
Zerind
$f(n) = 75 + 374$
$= 449$
140
99
151
80
Arad
$f(n) = 280 + 366$
$= 646$
Fagaras
$f(n) = 239 + 176$
$= 415$
Oradea
$f(n) = 291 + 380$
$= 671$
Rimnicu Vilcea
$f(n) = 220 + 193$
$= 413$

A* Search

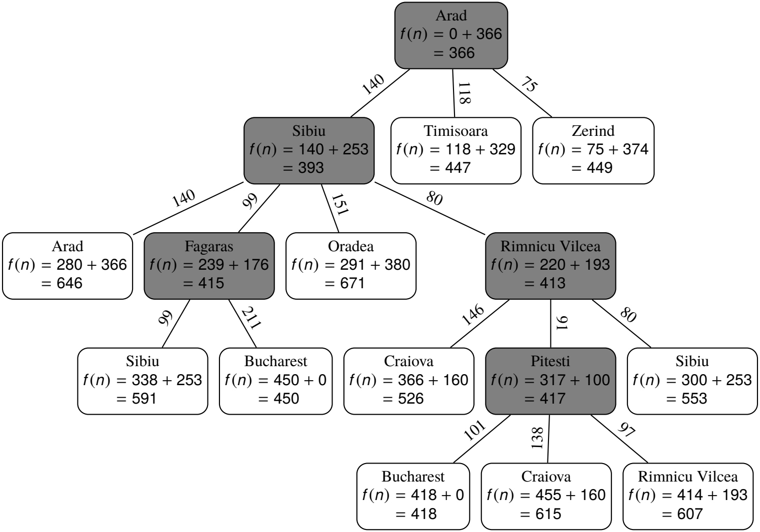

Arad
$f(n) = 0 + 366$
$= 366$
140
118
75
Sibiu
$f(n) = 140 + 253$
$= 393$
Timisoara
$f(n) = 118 + 329$
$= 447$
Zerind
$f(n) = 75 + 374$
$= 449$
140
99
151
80
Arad
$f(n) = 280 + 366$
$= 646$
Fagaras
$f(n) = 239 + 176$
$= 415$
Oradea
$f(n) = 291 + 380$
$= 671$
Rimnicu Vilcea
$f(n) = 220 + 193$
$= 413$
146
91
80
Craiova
$f(n) = 366 + 160$
$= 526$
Pitesti
$f(n) = 317 + 100$
$= 417$
Sibiu
$f(n) = 300 + 253$
$= 553$

A* Search



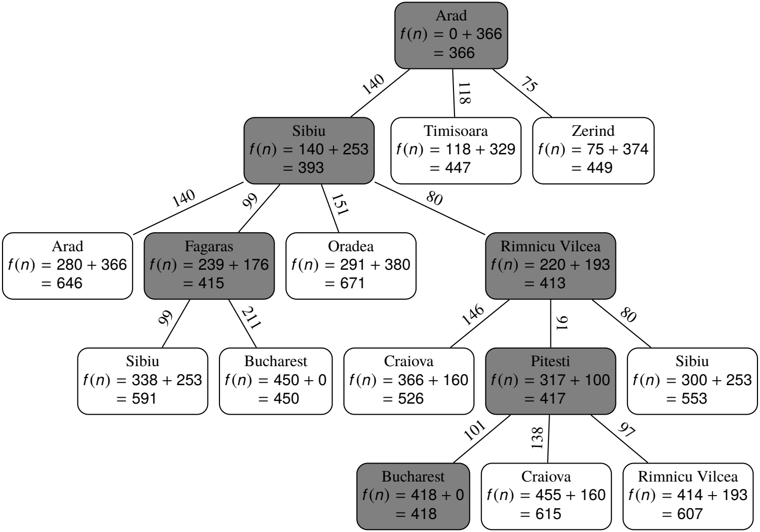
Arad
$f(n) = 0 + 366$
$= 366$
140
118
75
Sibiu
$f(n) = 140 + 253$
$= 393$
Timisoara
$f(n) = 118 + 329$
$= 447$
Zerind
$f(n) = 75 + 374$
$= 449$
140
99
151
80
Arad
$f(n) = 280 + 366$
$= 646$
Fagaras
$f(n) = 239 + 176$
$= 415$
Oradea
$f(n) = 291 + 380$
$= 671$
Rimnicu Vilcea
$f(n) = 220 + 193$
$= 413$
99
211
146
97
80
Sibiu
$f(n) = 338 + 253$
$= 591$
Bucharest
$f(n) = 450 + 0$
$= 450$
Craiova
$f(n) = 366 + 160$
$= 526$
Pitesti
$f(n) = 317 + 100$
$= 417$
Sibiu
$f(n) = 300 + 253$
$= 553$

A* Search



Arad
$f(n) = 0 + 366$
$= 366$
140
118
75
Sibiu
$f(n) = 140 + 253$
$= 393$
Timisoara
$f(n) = 118 + 329$
$= 447$
Zerind
$f(n) = 75 + 374$
$= 449$
140
99
151
80
Arad
$f(n) = 280 + 366$
$= 646$
Fagaras
$f(n) = 239 + 176$
$= 415$
Oradea
$f(n) = 291 + 380$
$= 671$
Rimnicu Vilcea
$f(n) = 220 + 193$
$= 413$
99
211
146
97
80
Sibiu
$f(n) = 338 + 253$
$= 591$
Bucharest
$f(n) = 450 + 0$
$= 450$
Craiova
$f(n) = 366 + 160$
$= 526$
Pitesti
$f(n) = 317 + 100$
$= 417$
Sibiu
$f(n) = 300 + 253$
$= 553$
101
138
97
Bucharest
$f(n) = 418 + 0$
$= 418$
Craiova
$f(n) = 455 + 160$
$= 615$
Rimnicu Vilcea
$f(n) = 414 + 193$
$= 607$

A* Search



Arad
$f(n) = 0 + 366$
$= 366$
140
118
75
Sibiu
$f(n) = 140 + 253$
$= 393$
Timisoara
$f(n) = 118 + 329$
$= 447$
Zerind
$f(n) = 75 + 374$
$= 449$
140
99
151
80
Arad
$f(n) = 280 + 366$
$= 646$
Fagaras
$f(n) = 239 + 176$
$= 415$
Oradea
$f(n) = 291 + 380$
$= 671$
Rimnicu Vilcea
$f(n) = 220 + 193$
$= 413$
99
211
146
97
80
Sibiu
$f(n) = 338 + 253$
$= 591$
Bucharest
$f(n) = 450 + 0$
$= 450$
Craiova
$f(n) = 366 + 160$
$= 526$
Pitesti
$f(n) = 317 + 100$
$= 417$
Sibiu
$f(n) = 300 + 253$
$= 553$
101
138
97
Bucharest
$f(n) = 418 + 0$
$= 418$
Craiova
$f(n) = 455 + 160$
$= 615$
Rimnicu Vilcea
$f(n) = 414 + 193$
$= 607$

A* Properties

Strategy?

Priority Queue, $f(n) = g(n) + h(n)$

Complete?

Yes, if there are finite nodes with $f(n) < C^*$

Optimal?

Yes, if h is consistent

Worst Case Time Complexity?

All nodes with $f(n) < C^*$, exponential in `len(path)`

Worst Case Space Complexity?

All nodes with $f(n) < C^*$

A* Properties

Strategy?

Priority Queue, $f(n) = g(n) + h(n)$

Complete?

Yes, if there are finite nodes with $f(n) < C^*$

Optimal?

Yes, if h is consistent

Worst Case Time Complexity?

All nodes with $f(n) < C^*$, exponential in `len(path)`

Worst Case Space Complexity?

All nodes with $f(n) < C^*$

A* Properties

Strategy?

Priority Queue, $f(n) = g(n) + h(n)$

Complete?

Yes, if there are finite nodes with $f(n) < C^*$

Optimal?

Yes, if h is consistent

Worst Case Time Complexity?

All nodes with $f(n) < C^*$, exponential in `len(path)`

Worst Case Space Complexity?

All nodes with $f(n) < C^*$

A* Properties

Strategy?

Priority Queue, $f(n) = g(n) + h(n)$

Complete?

Yes, if there are finite nodes with $f(n) < C^*$

Optimal?

Yes, if h is consistent

Worst Case Time Complexity?

All nodes with $f(n) < C^*$, exponential in `len(path)`

Worst Case Space Complexity?

All nodes with $f(n) < C^*$

A* Properties

Strategy?

Priority Queue, $f(n) = g(n) + h(n)$

Complete?

Yes, if there are finite nodes with $f(n) < C^*$

Optimal?

Yes, if h is consistent

Worst Case Time Complexity?

All nodes with $f(n) < C^*$, exponential in `len(path)`

Worst Case Space Complexity?

All nodes with $f(n) < C^*$

A* Properties

Strategy?

Priority Queue, $f(n) = g(n) + h(n)$

Complete?

Yes, if there are finite nodes with $f(n) < C^*$

Optimal?

Yes, if h is consistent

Worst Case Time Complexity?

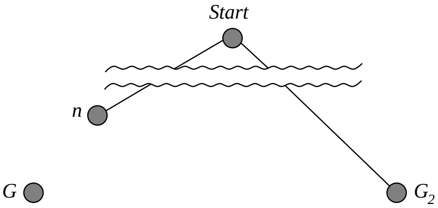
All nodes with $f(n) < C^*$, exponential in `len(path)`

Worst Case Space Complexity?

All nodes with $f(n) < C^*$

Proof that A* is Optimal

Suppose some suboptimal goal G_2 has been generated and is in the queue. Let n be an unexpanded node on a shortest path to an optimal goal G_1.

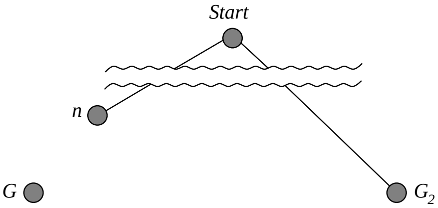


$$
\begin{aligned}
f(G_2) &= g(G_2) && \text{since } h(G_2) = 0 \\
&> g(G_1) && \text{since } G_2 \text{ is suboptimal} \\
&\geq f(n) && \text{since } h \text{ is consistent}
\end{aligned}
$$

Since $f(G_2) > f(n)$, A* will never select G_2 for expansion.

Proof that A* is Optimal

Suppose some suboptimal goal G_2 has been generated and is in the queue. Let n be an unexpanded node on a shortest path to an optimal goal G_1.

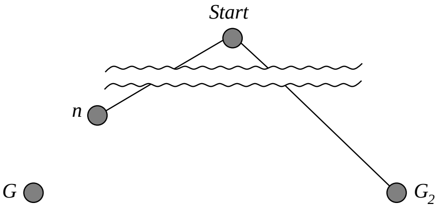


$$
\begin{aligned}
f(G_2) &= g(G_2) && \text{since } h(G_2) = 0 \\
&> g(G_1) && \text{since } G_2 \text{ is suboptimal} \\
&\geq f(n) && \text{since } h \text{ is consistent}
\end{aligned}
$$

Since $f(G_2) > f(n)$, A* will never select G_2 for expansion.

Proof that A* is Optimal

Suppose some suboptimal goal G_2 has been generated and is in the queue. Let n be an unexpanded node on a shortest path to an optimal goal G_1.

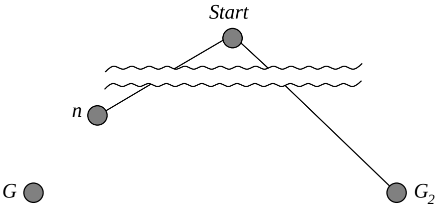


$$
\begin{aligned}
f(G_2) &= g(G_2) && \text{since } h(G_2) = 0 \\
&> g(G_1) && \text{since } G_2 \text{ is suboptimal} \\
&\geq f(n) && \text{since } h \text{ is consistent}
\end{aligned}
$$

Since $f(G_2) > f(n)$, A* will never select G_2 for expansion.

Proof that A* is Optimal

Suppose some suboptimal goal G_2 has been generated and is in the queue. Let n be an unexpanded node on a shortest path to an optimal goal G_1.

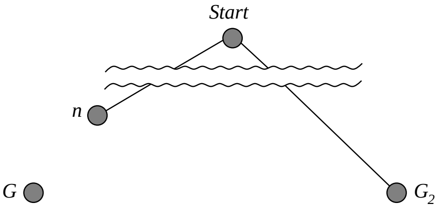


$$
\begin{array}{rcll}
f(G_2) & = & g(G_2) & \text{since } h(G_2) = 0 \\
 & > & g(G_1) & \text{since } G_2 \text{ is suboptimal} \\
 & \geq & f(n) & \text{since } h \text{ is consistent}
\end{array}
$$

Since $f(G_2) > f(n)$, A* will never select G_2 for expansion.

Proof that A* is Optimal

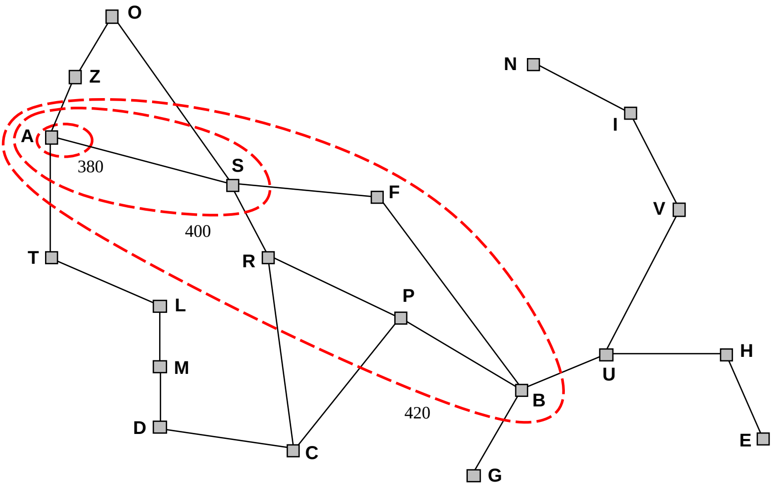
Suppose some suboptimal goal G_2 has been generated and is in the queue. Let n be an unexpanded node on a shortest path to an optimal goal G_1.



$$
\begin{aligned}
f(G_2) &= g(G_2) && \text{since } h(G_2) = 0 \\
&> g(G_1) && \text{since } G_2 \text{ is suboptimal} \\
&\geq f(n) && \text{since } h \text{ is consistent}
\end{aligned}
$$

Since $f(G_2) > f(n)$, A* will never select G_2 for expansion.

A* Contours



8-Puzzle Heuristics

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

Misplaced Tiles $h(n) = 6$

Manhattan Distance $h(n) = 4 + 0 + 3 + 3 + 1 + 0 + 2 + 1$

8-Puzzle Heuristics

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

Misplaced Tiles $h(n) = 6$

Manhattan Distance $h(n) = 4 + 0 + 3 + 3 + 1 + 0 + 2 + 1$

8-Puzzle Heuristics

7	2	4
5		6
8	3	1

Start State

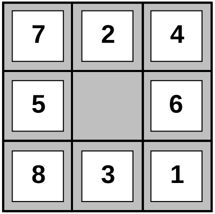
1	2	3
4	5	6
7	8	

Goal State

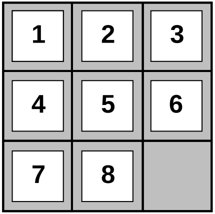
Misplaced Tiles $h(n) = 6$

Manhattan Distance $h(n) = 4 + 0 + 3 + 3 + 1 + 0 + 2 + 1$

8-Puzzle Heuristics



Start State



Goal State

Misplaced Tiles $h(n) = 6$

Manhattan Distance $h(n) = 4 + 0 + 3 + 3 + 1 + 0 + 2 + 1$

8-Puzzle Heuristics

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

Misplaced Tiles $h(n) = 6$

Manhattan Distance $h(n) = 4 + 0 + 3 + 3 + 1 + 0 + 2 + 1$

Heuristic Quality

All heuristics were not created equal

	Misplaced Tiles	Manhattan Distance
4 moves	13 nodes	12 nodes
8 moves	39 nodes	25 nodes
12 moves	227 nodes	73 nodes
16 moves	1301 nodes	211 nodes
20 moves	7276 nodes	676 nodes

Heuristic Dominance

Dominance

h_1 **dominates** h_2 if for all n, $h_1(n) \geq h_2(n)$

Which one dominates?

- Misplaced Tiles
- Manhattan Distance

Dominance = Efficiency

A* with h_1 will never expand more nodes than A* with h_2

Why? Every node with $h(n) < C^* - g(n)$ is expanded.

Heuristic Dominance

Dominance

h_1 **dominates** h_2 if for all n, $h_1(n) \geq h_2(n)$

Which one dominates?

- Misplaced Tiles
- Manhattan Distance

Dominance = Efficiency

A* with h_1 will never expand more nodes than A* with h_2

Why? Every node with $h(n) < C^* - g(n)$ is expanded

Heuristic Dominance

Dominance

h_1 **dominates** h_2 if for all n, $h_1(n) \geq h_2(n)$

Which one dominates?

- Misplaced Tiles
- Manhattan Distance

Dominance = Efficiency

A* with h_1 will never expand more nodes than A* with h_2

Why? Every node with $h(n) < C^* - g(n)$ is expanded

Heuristic Dominance

Dominance

h_1 **dominates** h_2 if for all n, $h_1(n) \geq h_2(n)$

Which one dominates?

- Misplaced Tiles
- Manhattan Distance

Dominance = Efficiency

A* with h_1 will never expand more nodes than A* with h_2

Why? Every node with $h(n) < C^* - g(n)$ is expanded

Relaxed Problems

The 8-Puzzle

Problem	**Heuristic**
Tiles move anywhere	Misplaced Tiles
Tiles move to adjacent squares	Manhattan Distance

Generating heuristics

Exact solution to relaxed problem $\Rightarrow$ consistent heuristic

Why?

- Solution in original problem is also solution in relaxed
- Heuristic is exact cost in relaxed $\Rightarrow$ triangle inequality

Relaxed Problems

The 8-Puzzle

Problem	Heuristic
Tiles move anywhere	Misplaced Tiles
Tiles move to adjacent squares	Manhattan Distance

Generating heuristics

Exact solution to relaxed problem $\Rightarrow$ consistent heuristic

Why?

- Solution in original problem is also solution in relaxe
- Heuristic is exact cost in relaxed $\Rightarrow$ triangle inequalit

Relaxed Problems

The 8-Puzzle

Problem	**Heuristic**
Tiles move anywhere	Misplaced Tiles
Tiles move to adjacent squares	Manhattan Distance

Generating heuristics

Exact solution to relaxed problem $\Rightarrow$ consistent heuristic

Why?

- Solution in original problem is also solution in relaxed
- Heuristic is exact cost in relaxed $\Rightarrow$ triangle inequality

Relaxed Problems

The 8-Puzzle

Problem	**Heuristic**
Tiles move anywhere	Misplaced Tiles
Tiles move to adjacent squares	Manhattan Distance

Generating heuristics

Exact solution to relaxed problem $\Rightarrow$ consistent heuristic

Why?

- Solution in original problem is also solution in relaxed
- Heuristic is exact cost in relaxed $\Rightarrow$ triangle inequality

Relaxed Problems

The 8-Puzzle

Problem	**Heuristic**
Tiles move anywhere	Misplaced Tiles
Tiles move to adjacent squares	Manhattan Distance

Generating heuristics

Exact solution to relaxed problem $\Rightarrow$ consistent heuristic

Why?

- Solution in original problem is also solution in relaxed
- Heuristic is exact cost in relaxed $\Rightarrow$ triangle inequality

Relaxed Problems

The 8-Puzzle

Problem	**Heuristic**
Tiles move anywhere	Misplaced Tiles
Tiles move to adjacent squares	Manhattan Distance

Generating heuristics

Exact solution to relaxed problem $\Rightarrow$ consistent heuristic

Why?

- Solution in original problem is also solution in relaxed
- Heuristic is exact cost in relaxed $\Rightarrow$ triangle inequality

Relaxed Problems

The 8-Puzzle

Problem	Heuristic
Tiles move anywhere	Misplaced Tiles
Tiles move to adjacent squares	Manhattan Distance

Generating heuristics

Exact solution to relaxed problem $\Rightarrow$ consistent heuristic

Why?

- Solution in original problem is also solution in relaxed
- Heuristic is exact cost in relaxed $\Rightarrow$ triangle inequality

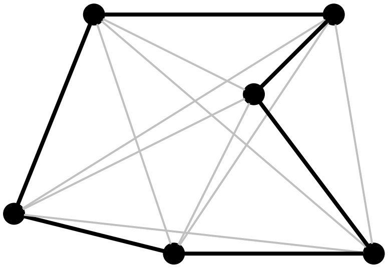
Traveling Salesman Problem

Problem

Visit all cities exactly once, minimum distance

Heuristic

Minimum spanning tree
Solvable in $O(n^2)$



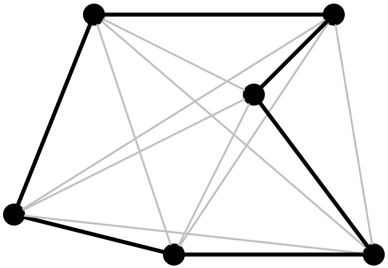
Traveling Salesman Problem

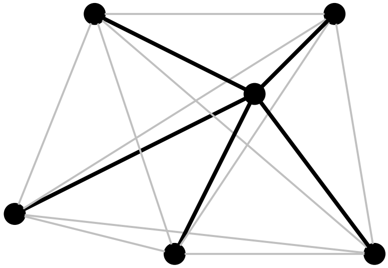
Problem

Visit all cities exactly once, minimum distance

Heuristic

Minimum spanning tree
Solvable in $O(n^2)$





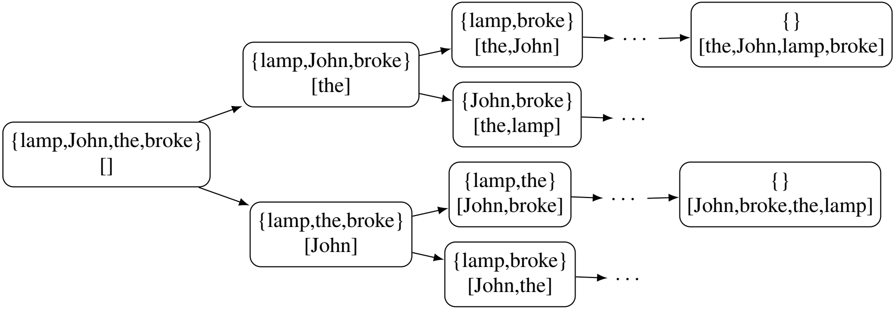
Bag Generation

Order of a bag of words

Initial	Full bag, empty sentence
Actions	Pop from bag, add to sentence
Goal	Empty bag, full sentence
Cost	$c(w_1, w_2) + c(w_2, w_3) + \ldots + c(w_{n-1}, w_n)$

$c(v, w)$

...		
John	broke	3.5
...		
the	the	25.1
...		



A Bag Generation Heuristic

Node

`{lamp, the}`
`[John, broke]`

Bag-Word Estimates

lamp
the

h(n)

John	lamp	7.6
broke	lamp	6.9
the	lamp	3.5
lamp	lamp	23.0

John	the	7.1
broke	the	3.2
the	the	25.1
lamp	the	6.2

A Bag Generation Heuristic

Node

`{lamp, the}`
`[John, broke]`

Bag-Word Estimates

$$\min_{w \in \{\text{broke,the}\}} s(w, \text{lamp})$$
$$\min_{w \in \{\text{broke,lamp}\}} s(w, \text{the})$$

h(n)

John	lamp	7.6
broke	lamp	6.9
the	lamp	3.5
lamp	lamp	23.0

John	the	7.1
broke	the	3.2
the	the	25.1
lamp	the	6.2

A Bag Generation Heuristic

Node

`{lamp, the}`

`[John, broke]`

Bag-Word Estimates

$$\min_{w \in \{\text{broke},\text{the}\}} s(w, \text{lamp}) \quad 3.5$$

$$\min_{w \in \{\text{broke},\text{lamp}\}} s(w, \text{the})$$

h(n)

John	lamp	7.6
broke	lamp	6.9
the	lamp	3.5
lamp	lamp	23.0

John	the	7.1
broke	the	3.2
the	the	25.1
lamp	the	6.2

A Bag Generation Heuristic

Node

`{lamp, the}`
`[John, broke]`

Bag-Word Estimates

$$\min_{w \in \{\text{broke,the}\}} s(w, \text{lamp}) \quad 3.5$$
$$\min_{w \in \{\text{broke,lamp}\}} s(w, \text{the}) \quad 3.2$$

h(n)

John	lamp	7.6
broke	lamp	6.9
the	lamp	3.5
lamp	lamp	23.0

John	the	7.1
broke	the	3.2
the	the	25.1
lamp	the	6.2

A Bag Generation Heuristic

Node

`{lamp, the}`
`[John, broke]`

Bag-Word Estimates

$$\min_{w \in \{\text{broke}, \text{the}\}} s(w, \text{lamp}) \quad 3.5$$

$$\min_{w \in \{\text{broke}, \text{lamp}\}} s(w, \text{the}) \quad 3.2$$

h(n)

$3.5 + 3.2 = 6.7$

John	lamp	7.6
broke	lamp	6.9
the	lamp	3.5
lamp	lamp	23.0

John	the	7.1
broke	the	3.2
the	the	25.1
lamp	the	6.2

Key Points

Search Problems

- Initial State, Actions, Goal Test, Path Cost

Search Strategies

- Breadth-first
- Uniform-cost
- Depth-first
- Iterative Deepening
- A* Search

Heuristics

- Dominance, Relaxed Problems