Today

Uninformed Search Methods:

Today

Uninformed Search Methods:

- Depth-First Search

Today

Uninformed Search Methods:

- Depth-First Search
- Breadth-First Search

Today

Uninformed Search Methods:

- Depth-First Search
- Breadth-First Search
- Uniform-Cost Search

Today

Uninformed Search Methods:

- Depth-First Search
- Breadth-First Search
- Uniform-Cost Search

Informed Search:

Today

Uninformed Search Methods:

- Depth-First Search
- Breadth-First Search
- Uniform-Cost Search

Informed Search:

- A* or “A star”.

Today

Uninformed Search Methods:

- Depth-First Search
- Breadth-First Search
- Uniform-Cost Search

Informed Search:

- A* or “A star”.
  Sirius?

Today

Uninformed Search Methods:

- Depth-First Search
- Breadth-First Search
- Uniform-Cost Search

Informed Search:

- A* or “A star”.
  Sirius? Brightest star in sky.

Today

Uninformed Search Methods:

- Depth-First Search
- Breadth-First Search
- Uniform-Cost Search

Informed Search:

- A* or “A star”.
  - Sirius? Brightest star in sky.
  - No!

Today

Uninformed Search Methods:

- Depth-First Search
- Breadth-First Search
- Uniform-Cost Search

Informed Search:

A* or “A star”.
Sirius? Brightest star in sky.
No! search! Main idea:

Today

Uninformed Search Methods:

- Depth-First Search
- Breadth-First Search
- Uniform-Cost Search

Informed Search:

A* or “A star”.
Sirius? Brightest star in sky.
No! search! Main idea: Heuristics.

Today

Uninformed Search Methods:

- Depth-First Search
- Breadth-First Search
- Uniform-Cost Search

Informed Search:

A* or “A star”.
Sirius? Brightest star in sky.
No! search! Main idea: Heuristics.
Admissible.

Today

Uninformed Search Methods:

- Depth-First Search
- Breadth-First Search
- Uniform-Cost Search

Informed Search:

- A* or “A star”.
  - Sirius? Brightest star in sky.
  - No! search! Main idea: Heuristics.
  - Admissible.
- Graph Search:

Today

Uninformed Search Methods:

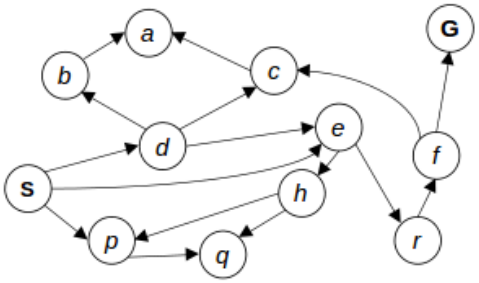
- Depth-First Search
- Breadth-First Search
- Uniform-Cost Search

Informed Search:

A* or “A star”.
Sirius? Brightest star in sky.
No! search! Main idea: Heuristics.
Admissible.
Graph Search: Consistent.

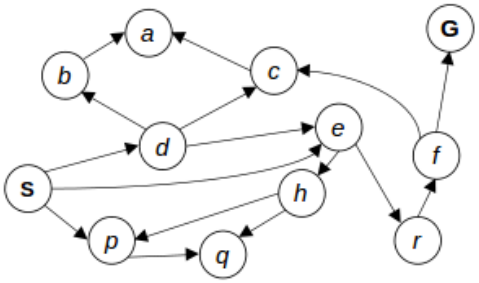
State Space Graphs

State space graph: A mathematical representation of a search problem



Tiny search graph for a tiny search problem

State Space Graphs

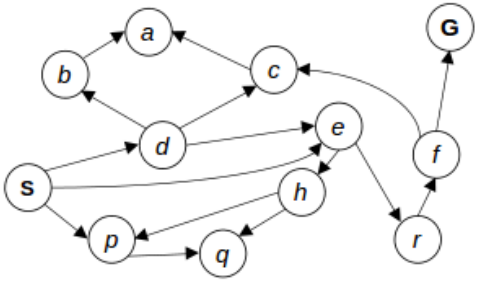


Tiny search graph for a tiny search problem

State space graph: A mathematical representation of a search problem

- Nodes are (abstracted) world configurations

State Space Graphs

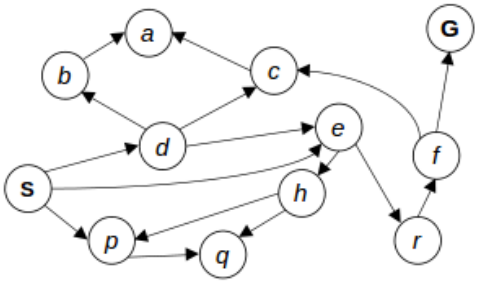


Tiny search graph for a tiny search problem

State space graph: A mathematical representation of a search problem

- Nodes are (abstracted) world configurations
- Arcs represent successors (action results)

State Space Graphs

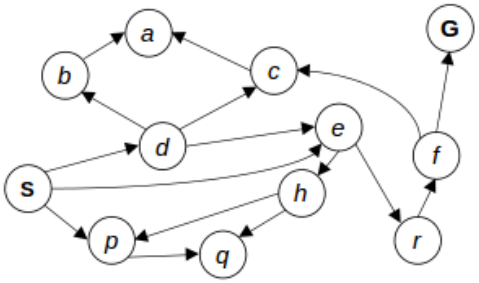


Tiny search graph for a tiny search problem

State space graph: A mathematical representation of a search problem

- Nodes are (abstracted) world configurations
- Arcs represent successors (action results)
- The goal test is a set of goal nodes (maybe only one)

State Space Graphs



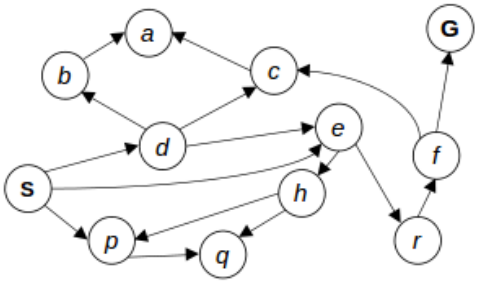
Tiny search graph for a tiny search problem

State space graph: A mathematical representation of a search problem

- Nodes are (abstracted) world configurations
- Arcs represent successors (action results)
- The goal test is a set of goal nodes (maybe only one)

In a state space graph, each state occurs only once!

State Space Graphs



Tiny search graph for a tiny search problem

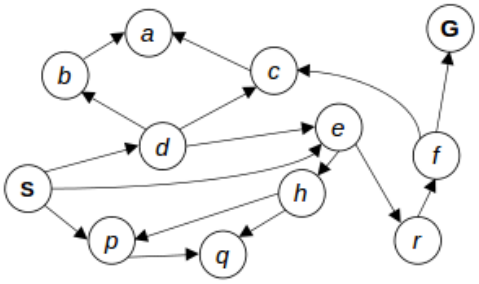
State space graph: A mathematical representation of a search problem

- Nodes are (abstracted) world configurations
- Arcs represent successors (action results)
- The goal test is a set of goal nodes (maybe only one)

In a state space graph, each state occurs only once!

We can rarely build this full graph in memory (it's too big),

State Space Graphs



Tiny search graph for a tiny search problem

State space graph: A mathematical representation of a search problem

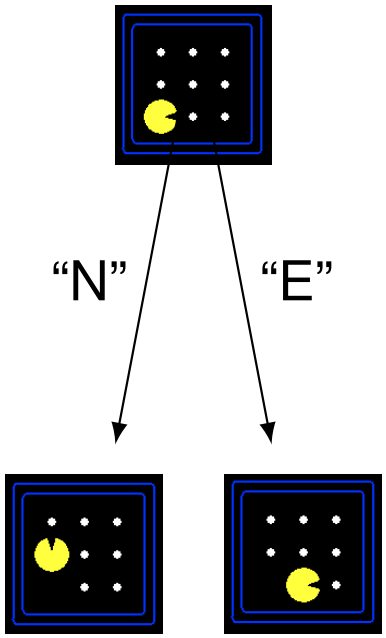
- Nodes are (abstracted) world configurations
- Arcs represent successors (action results)
- The goal test is a set of goal nodes (maybe only one)

In a state space graph, each state occurs only once!

We can rarely build this full graph in memory (it's too big), but it's a useful idea.

Search Trees

This is now / start

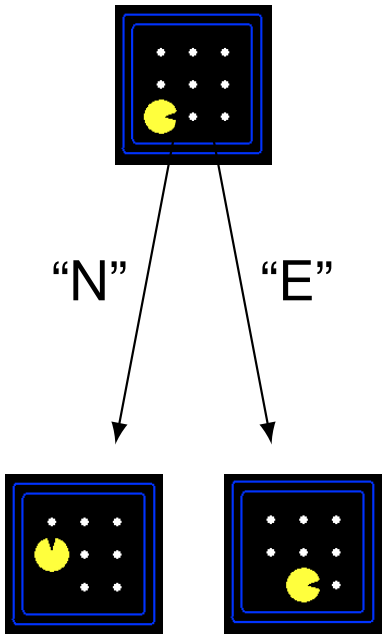


Possible futures.

A search tree:

Search Trees

This is now / start



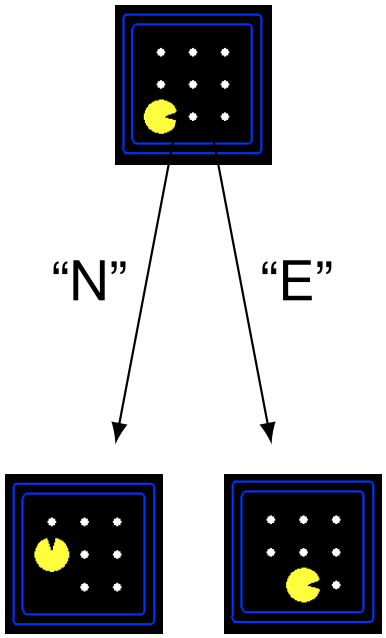
Possible futures.

A search tree:

A “what if” tree of plans and their outcomes

Search Trees

This is now / start



Possible futures.

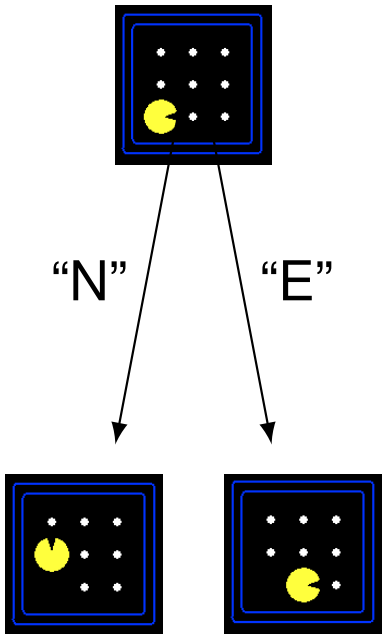
A search tree:

A “what if” tree of plans and their outcomes

The start state is the root node

Search Trees

This is now / start



Possible futures.

A search tree:

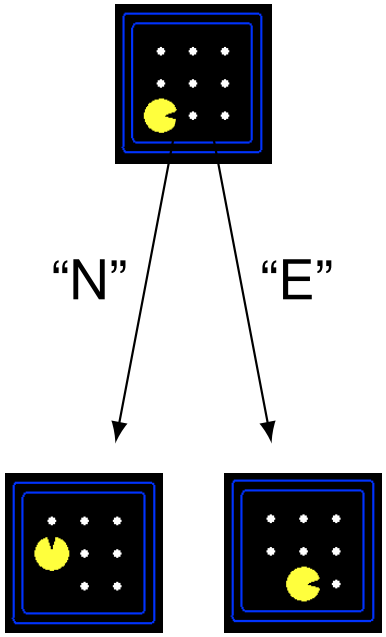
A “what if” tree of plans and their outcomes

The start state is the root node

Children correspond to successors

Search Trees

This is now / start



Possible futures.

A search tree:

A “what if” tree of plans and their outcomes

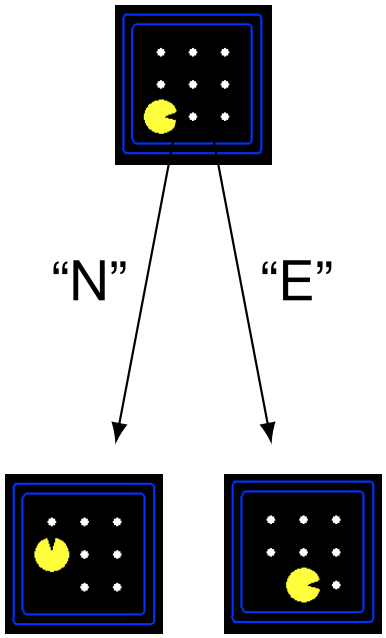
The start state is the root node

Children correspond to successors

Nodes show states, but correspond to PLANS that achieve those states

Search Trees

This is now / start



Possible futures.

A search tree:

A “what if” tree of plans and their outcomes

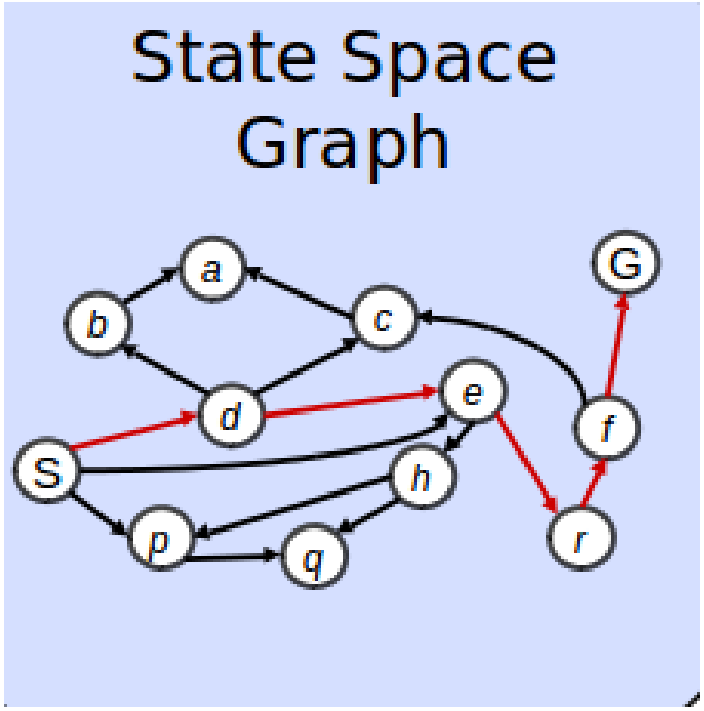
The start state is the root node

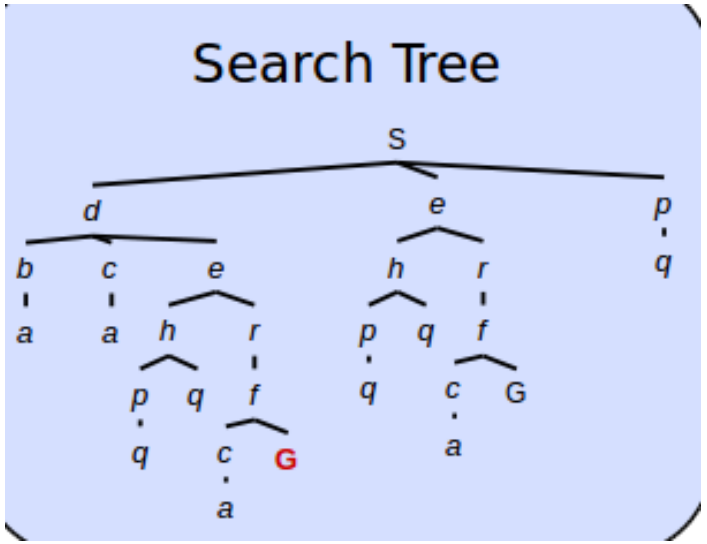
Children correspond to successors

Nodes show states, but correspond to PLANS that achieve those states

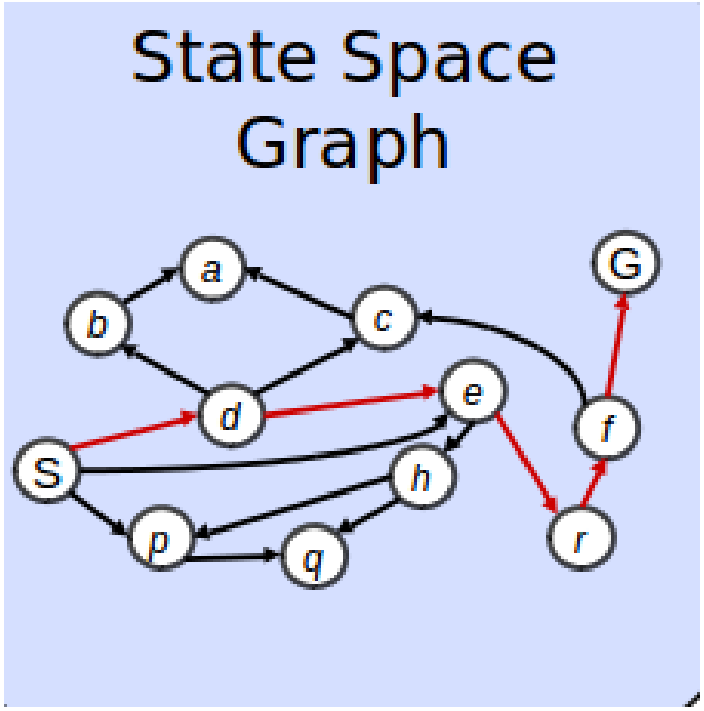
For most problems, we can never actually build the whole tree

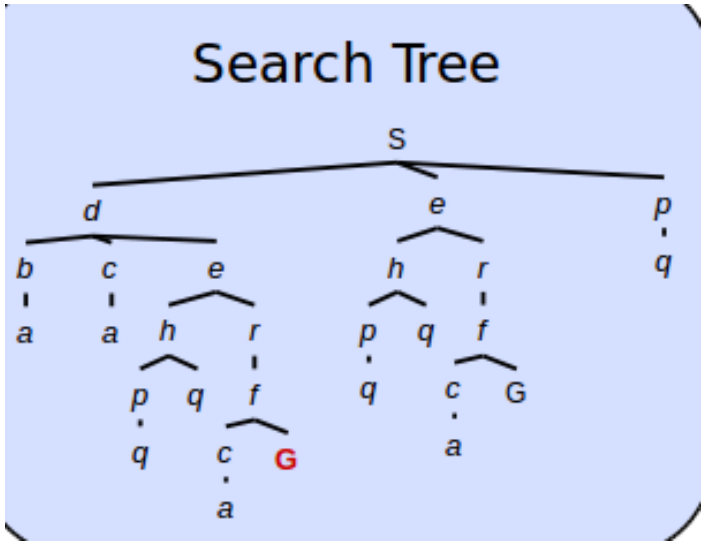
State Space Graphs vs. Search Trees





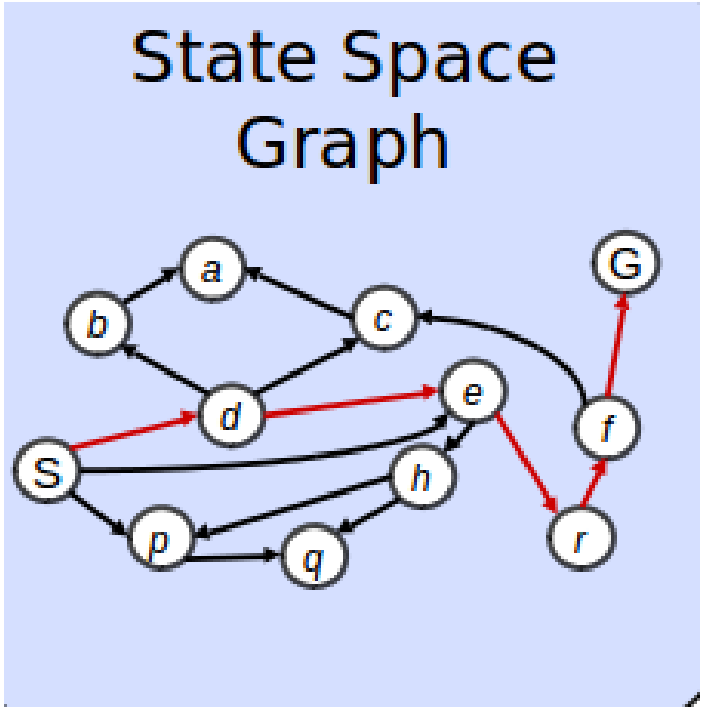
State Space Graphs vs. Search Trees

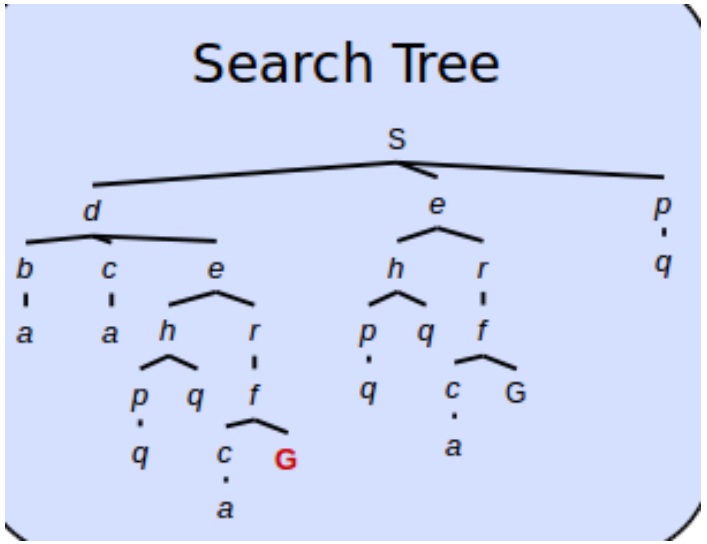




Each NODE in search tree is an entire PATH in state space graph.

State Space Graphs vs. Search Trees

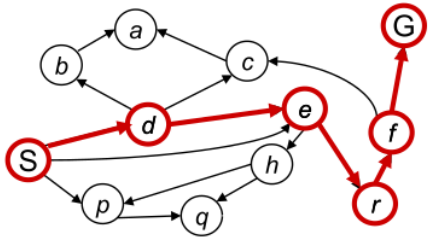


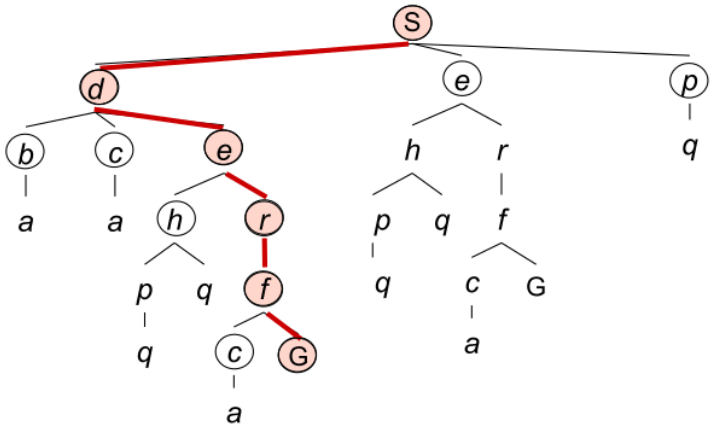


Each NODE in search tree is an entire PATH in state space graph.

We construct both on demand – and we construct as little as possible.

Tree Search: example.





~~s~~
~~s → d~~
s → e
s → p
s → d → b
s → d → c
~~s → d → e~~
s → d → e → h
~~s → d → e → r~~
~~s → d → e → r → f~~
s → d → e → r → f → c
~~s → d → e → r → f → G~~

Depth-First Search

Depth-First Search

Strategy: expand a deepest node first.

Depth-First Search

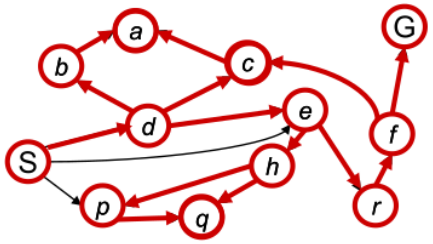
Strategy: expand a deepest node first.

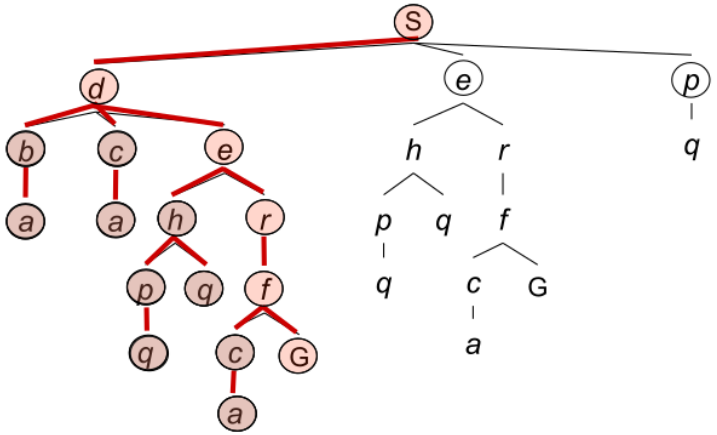
Implementation: Fringe is a LIFO stack

Tree Search: example.

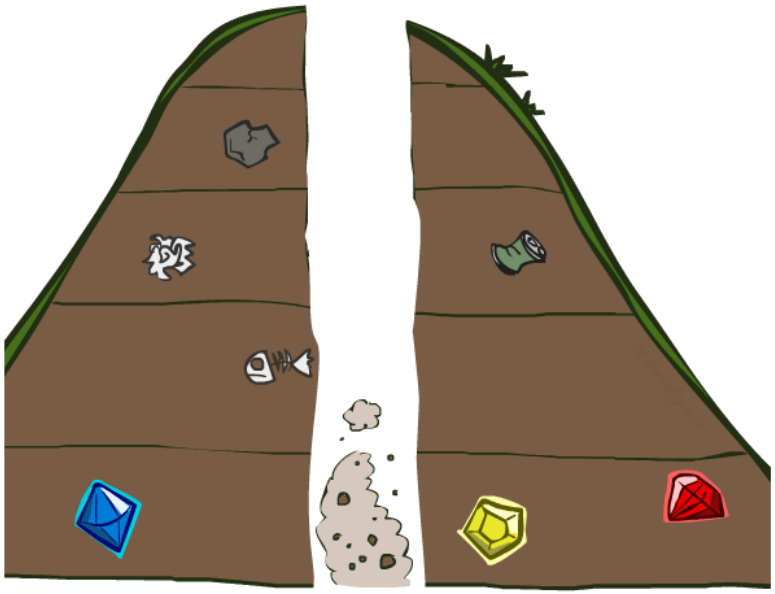
Strategy: expand a deepest node first

Implementation: Fringe is a LIFO stack



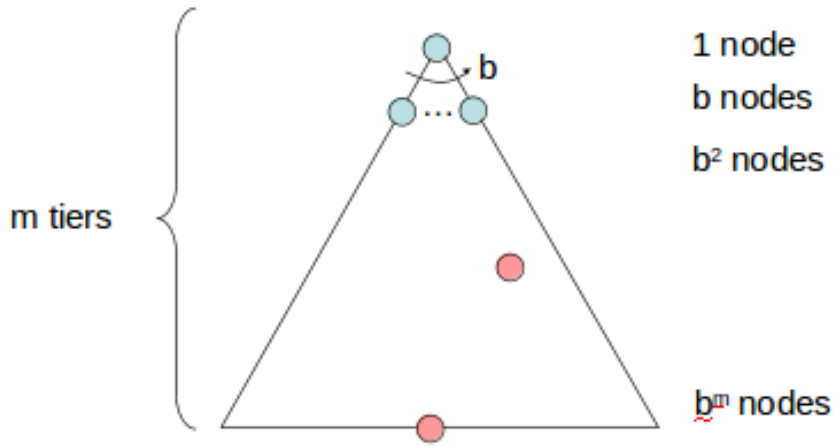


Search Algorithm Properties

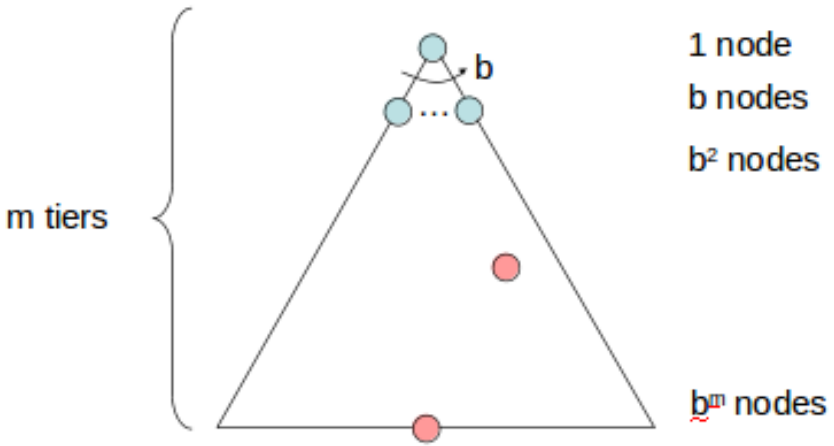


Search Algorithm Properties

Complete: Guaranteed to find a solution if one exists?



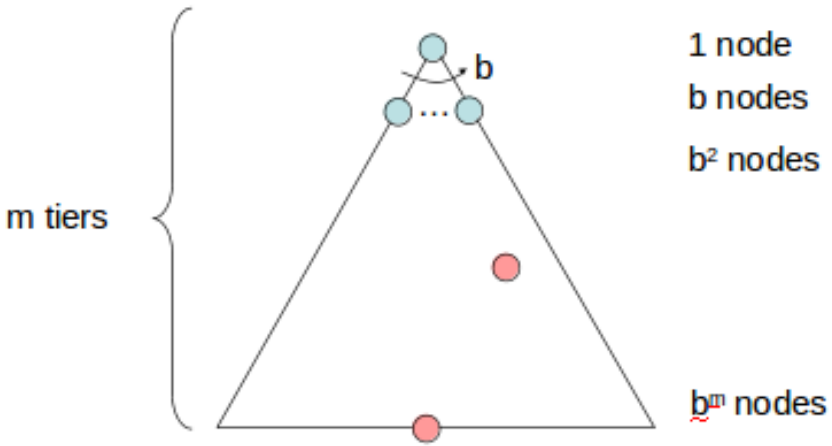
Search Algorithm Properties



Complete: Guaranteed to find a solution if one exists?

Optimal: Guaranteed to find the least cost path?

Search Algorithm Properties

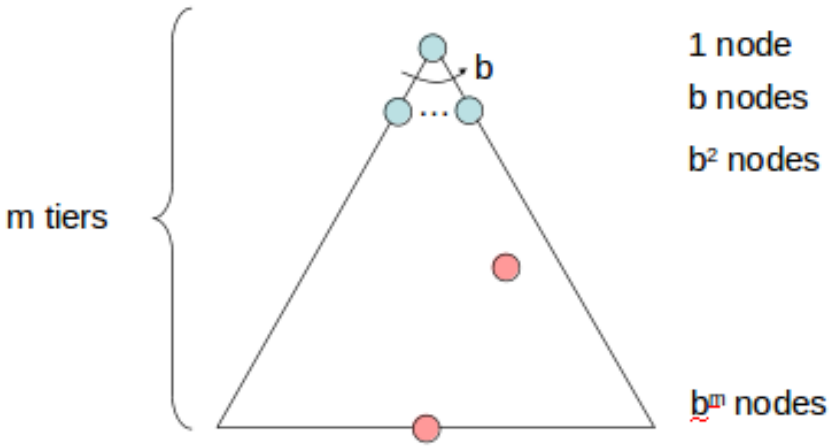


Complete: Guaranteed to find a solution if one exists?

Optimal: Guaranteed to find the least cost path?

Time complexity?

Search Algorithm Properties



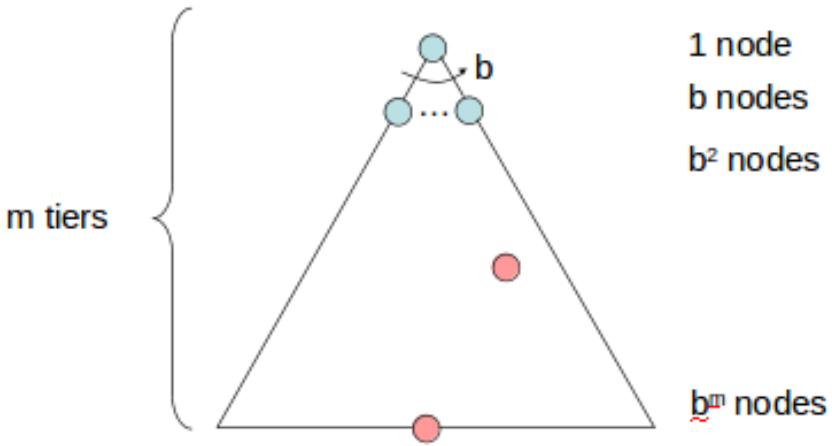
Complete: Guaranteed to find a solution if one exists?

Optimal: Guaranteed to find the least cost path?

Time complexity?

Space complexity?

Search Algorithm Properties



Complete: Guaranteed to find a solution if one exists?

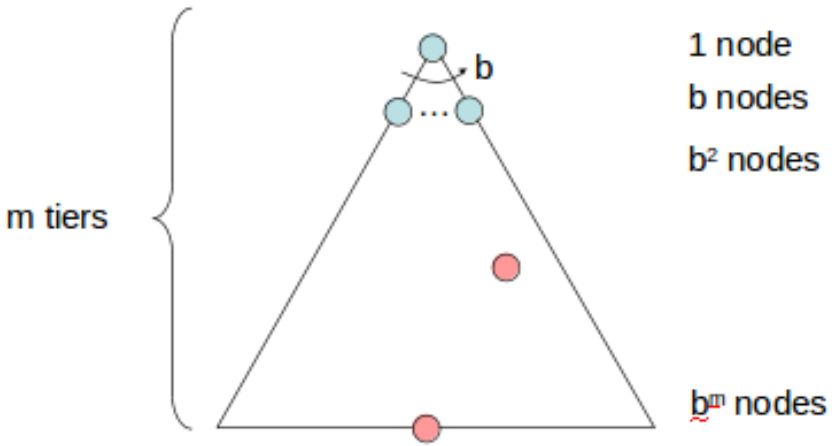
Optimal: Guaranteed to find the least cost path?

Time complexity?

Space complexity?

Sketch of search tree:

Search Algorithm Properties



Complete: Guaranteed to find a solution if one exists?

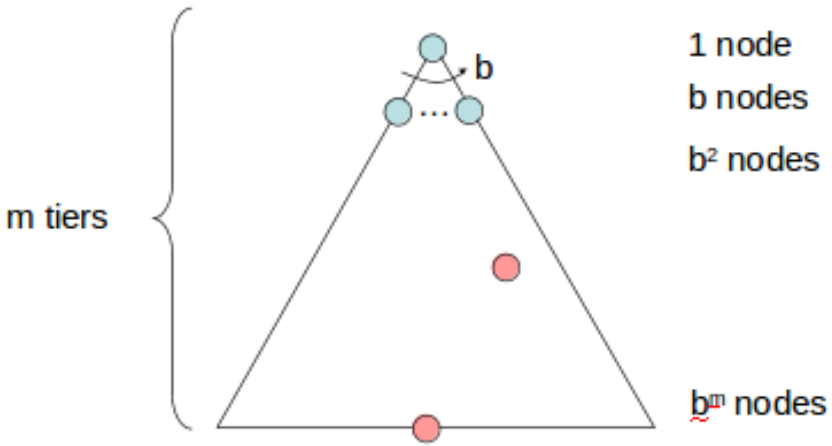
Optimal: Guaranteed to find the least cost path?

Time complexity?

Space complexity?

Sketch of search tree:
b is the branching factor

Search Algorithm Properties



Complete: Guaranteed to find a solution if one exists?

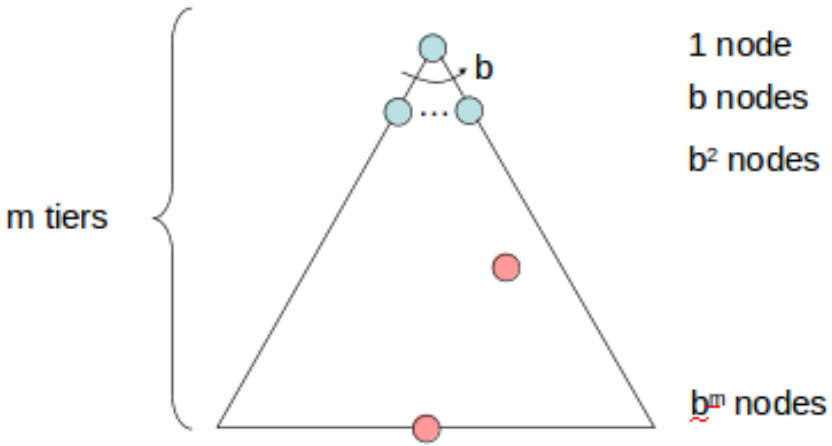
Optimal: Guaranteed to find the least cost path?

Time complexity?

Space complexity?

Sketch of search tree:
b is the branching factor
m is the maximum depth

Search Algorithm Properties



Complete: Guaranteed to find a solution if one exists?

Optimal: Guaranteed to find the least cost path?

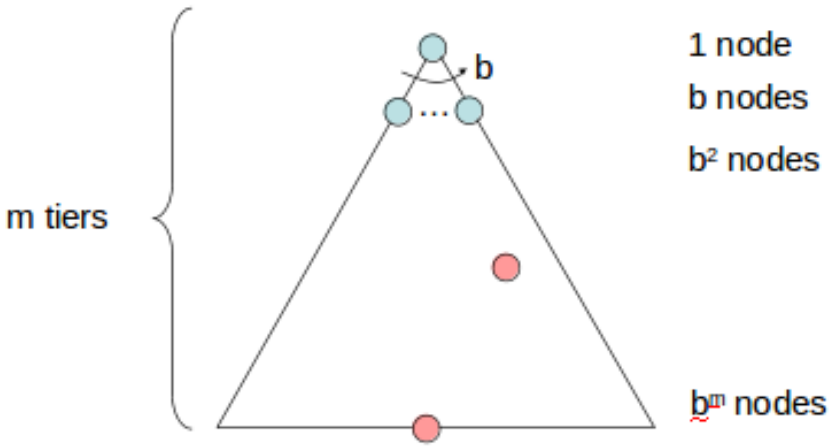
Time complexity?

Space complexity?

Sketch of search tree:

b is the branching factor
m is the maximum depth
solutions at various depths

Search Algorithm Properties



Complete: Guaranteed to find a solution if one exists?

Optimal: Guaranteed to find the least cost path?

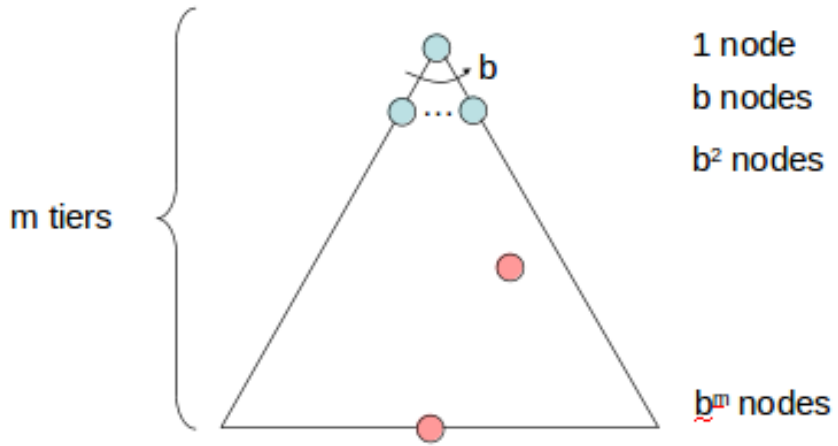
Time complexity?

Space complexity?

Sketch of search tree:
- b is the branching factor
- m is the maximum depth
- solutions at various depths

Number of nodes in entire tree?

Search Algorithm Properties



Complete: Guaranteed to find a solution if one exists?

Optimal: Guaranteed to find the least cost path?

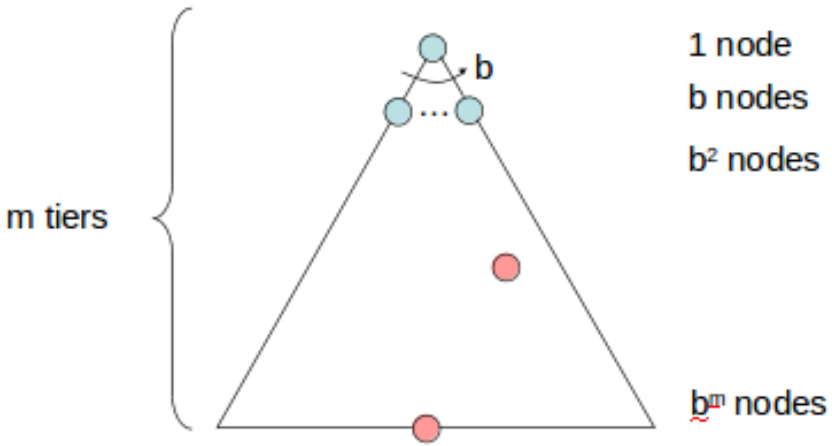
Time complexity?

Space complexity?

Sketch of search tree:
- b is the branching factor
- m is the maximum depth
- solutions at various depths

Number of nodes in entire tree?
- 1

Search Algorithm Properties



Complete: Guaranteed to find a solution if one exists?

Optimal: Guaranteed to find the least cost path?

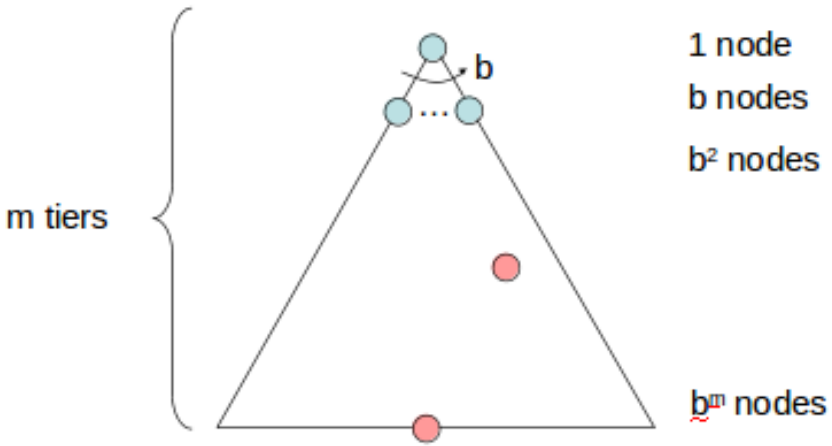
Time complexity?

Space complexity?

Sketch of search tree:
b is the branching factor
m is the maximum depth
solutions at various depths

Number of nodes in entire tree?
$1 + b$

Search Algorithm Properties



Complete: Guaranteed to find a solution if one exists?

Optimal: Guaranteed to find the least cost path?

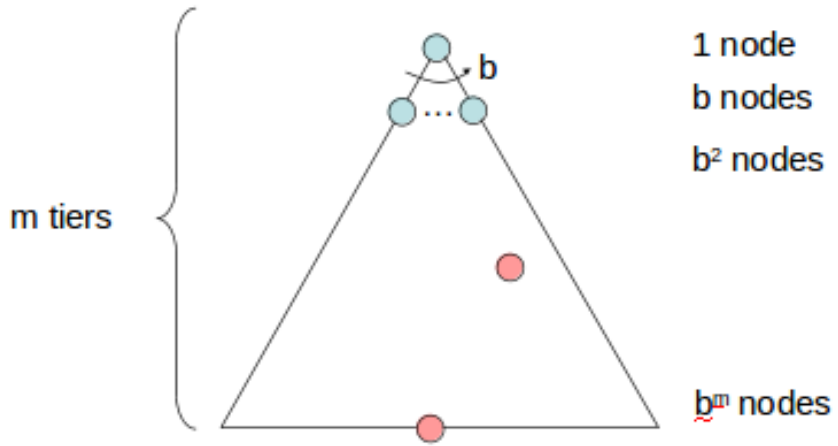
Time complexity?

Space complexity?

Sketch of search tree:
b is the branching factor
m is the maximum depth
solutions at various depths

Number of nodes in entire tree?
$1 + b + b^2 + \cdots$

Search Algorithm Properties



Complete: Guaranteed to find a solution if one exists?

Optimal: Guaranteed to find the least cost path?

Time complexity?

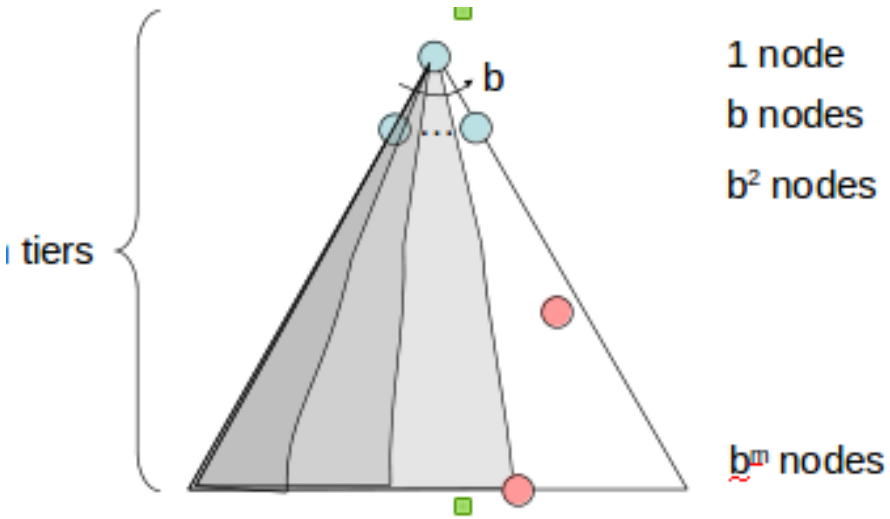
Space complexity?

Sketch of search tree:
b is the branching factor
m is the maximum depth
solutions at various depths

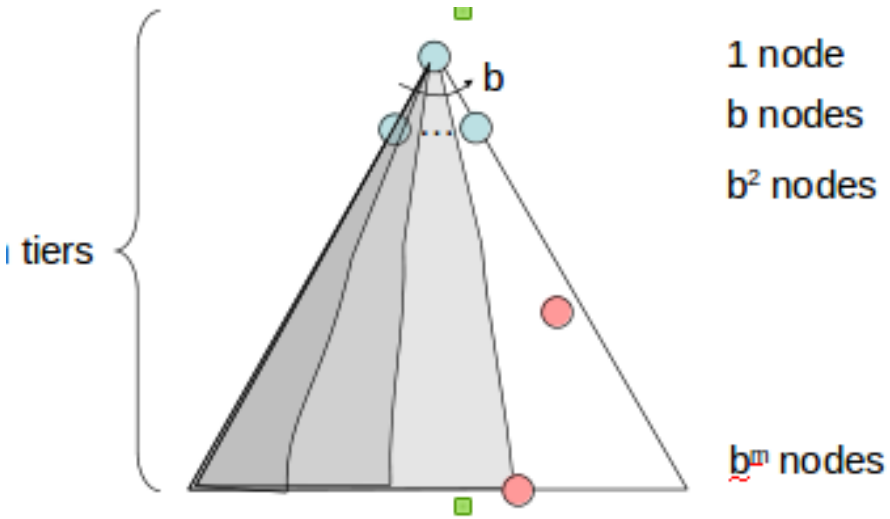
Number of nodes in entire tree?
$1 + b + b^2 + \cdots = O(b^m)$

Depth-First Search (DFS) Properties

Which nodes expanded?



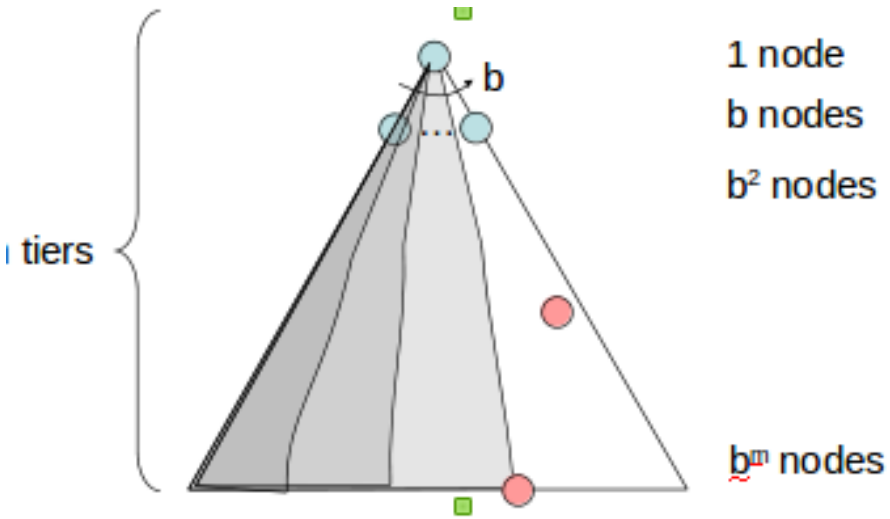
Depth-First Search (DFS) Properties



Which nodes expanded?

Some left prefix of the tree.

Depth-First Search (DFS) Properties

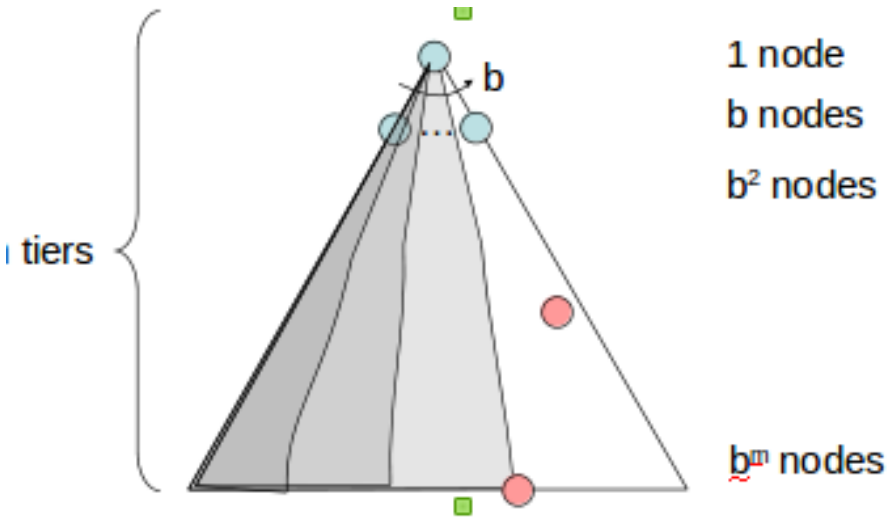


Which nodes expanded?

Some left prefix of the tree.

Could process the whole tree!

Depth-First Search (DFS) Properties



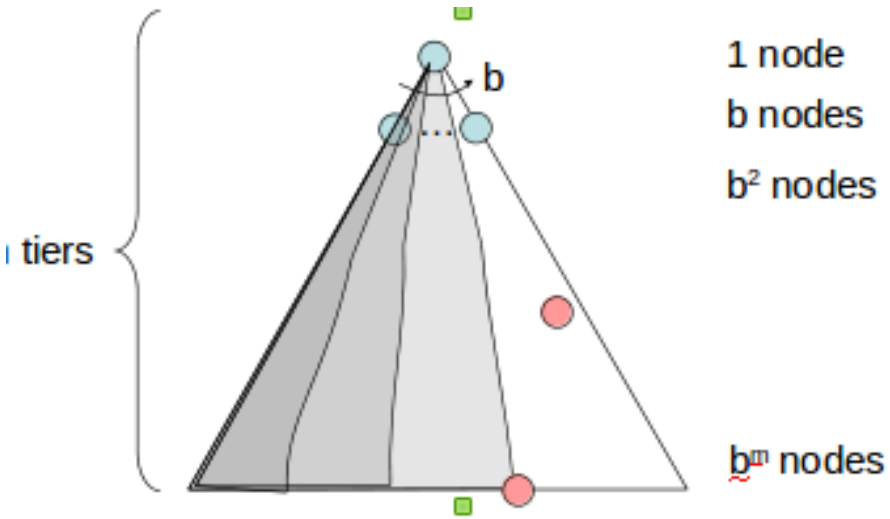
Which nodes expanded?

Some left prefix of the tree.

Could process the whole tree!

If m is finite, takes time $O(b^m)$

Depth-First Search (DFS) Properties



Which nodes expanded?

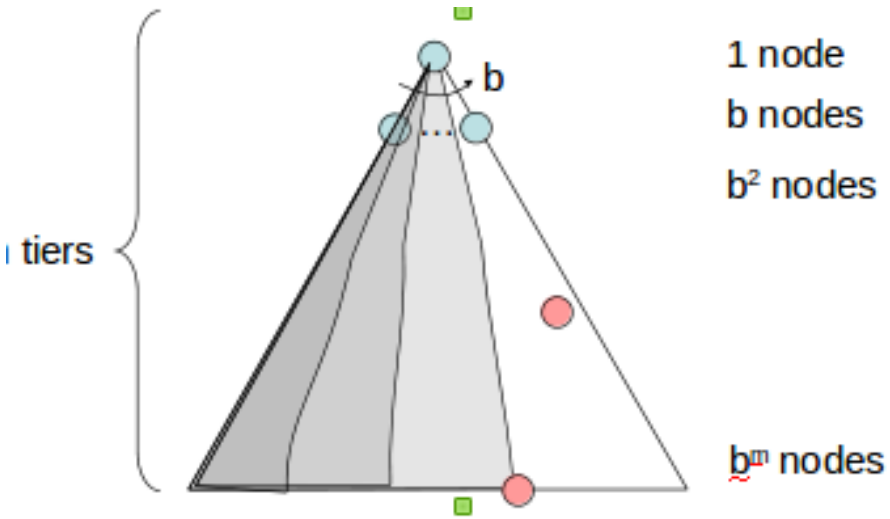
Some left prefix of the tree.

Could process the whole tree!

If m is finite, takes time $O(b^m)$

How much space does the fringe take?

Depth-First Search (DFS) Properties



Which nodes expanded?

Some left prefix of the tree.

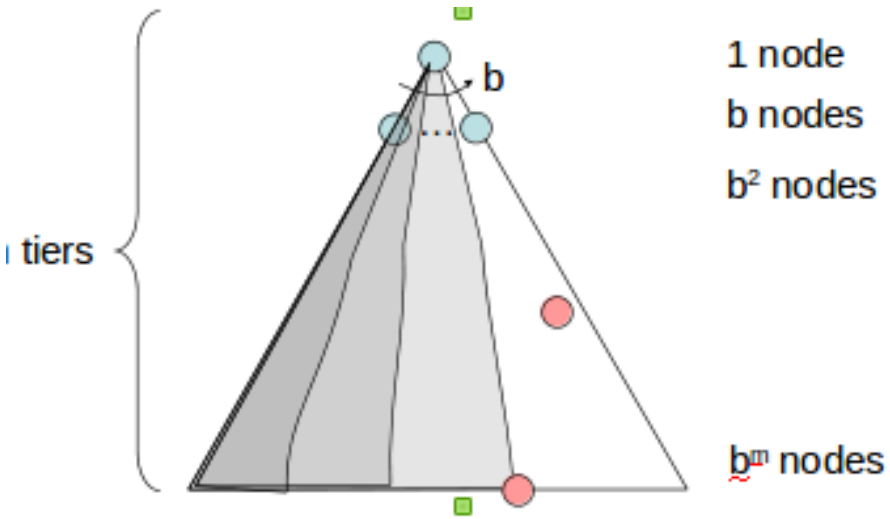
Could process the whole tree!

If m is finite, takes time $O(b^m)$

How much space does the fringe take?

Only has siblings on path to root, so $O(bm)$

Depth-First Search (DFS) Properties



Is it complete?

Which nodes expanded?

Some left prefix of the tree.

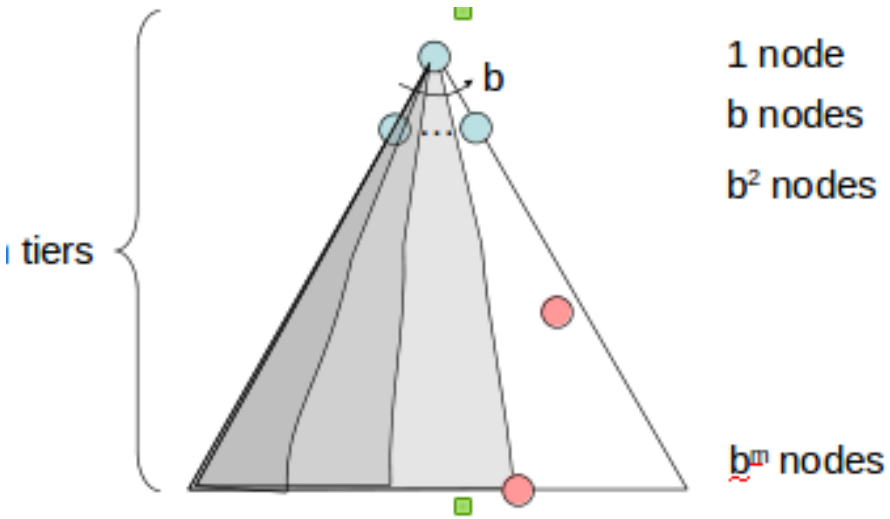
Could process the whole tree!

If m is finite, takes time $O(b^m)$

How much space does the fringe take?

Only has siblings on path to root, so $O(bm)$

Depth-First Search (DFS) Properties



Which nodes expanded?

Some left prefix of the tree.

Could process the whole tree!

If m is finite, takes time $O(b^m)$

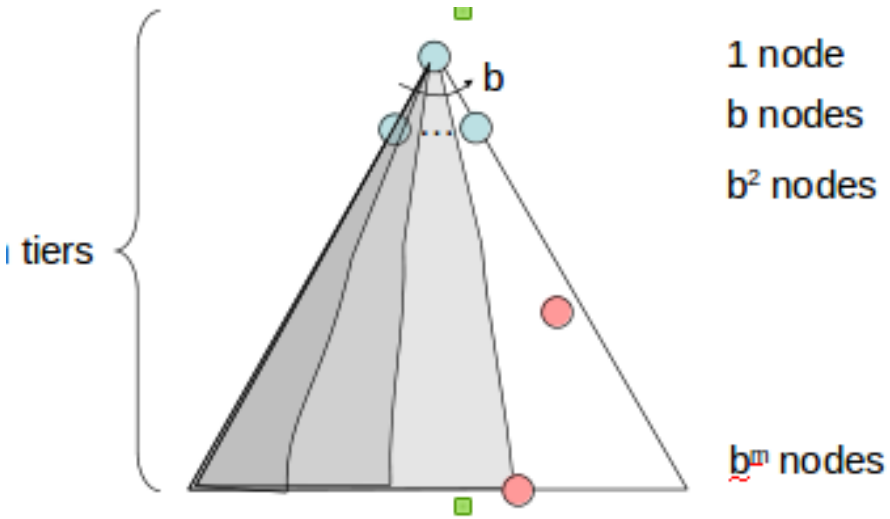
How much space does the fringe take?

Only has siblings on path to root, so $O(bm)$

Is it complete?

m could be infinite, so only if we prevent cycles (more later)

Depth-First Search (DFS) Properties



Which nodes expanded?

Some left prefix of the tree.

Could process the whole tree!

If m is finite, takes time $O(b^m)$

How much space does the fringe take?

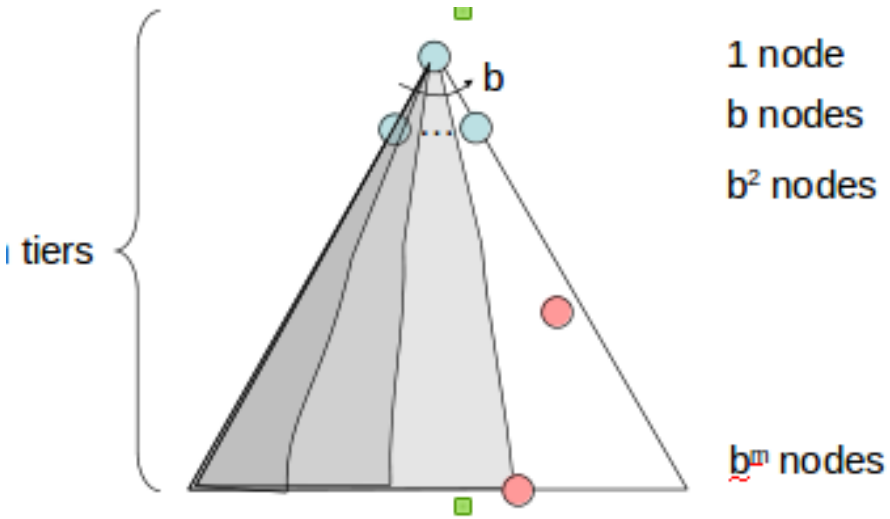
Only has siblings on path to root, so $O(bm)$

Is it complete?

m could be infinite, so only if we prevent cycles (more later)

Is it optimal?

Depth-First Search (DFS) Properties



Which nodes expanded?

Some left prefix of the tree.

Could process the whole tree!

If m is finite, takes time $O(b^m)$

How much space does the fringe take?

Only has siblings on path to root, so $O(bm)$

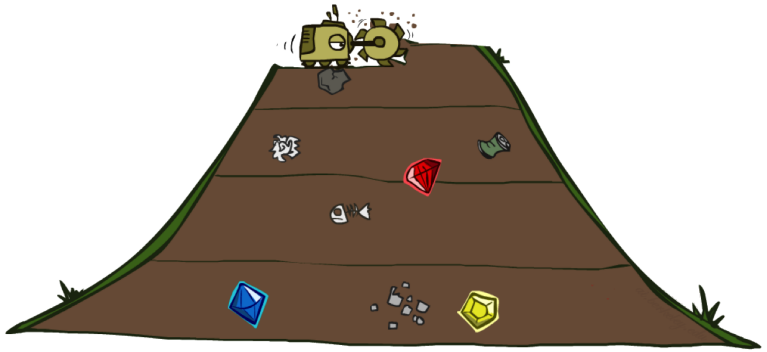
Is it complete?

m could be infinite, so only if we prevent cycles (more later)

Is it optimal?

No, it finds the “leftmost” solution, regardless of depth or cost

Breadth-First Search



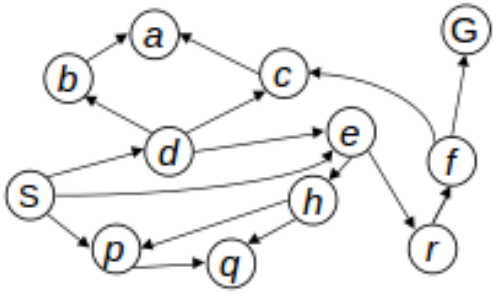
Breadth-First Search

Strategy: expand a shallowest node first

Breadth-First Search

Strategy: expand a shallowest node first

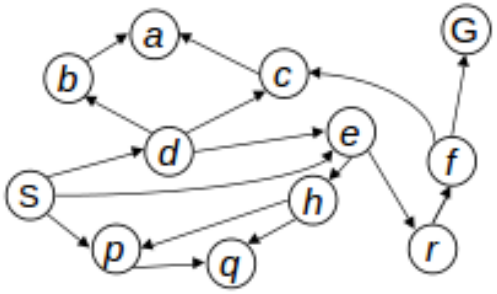
Implementation: Fringe is a FIFO queue

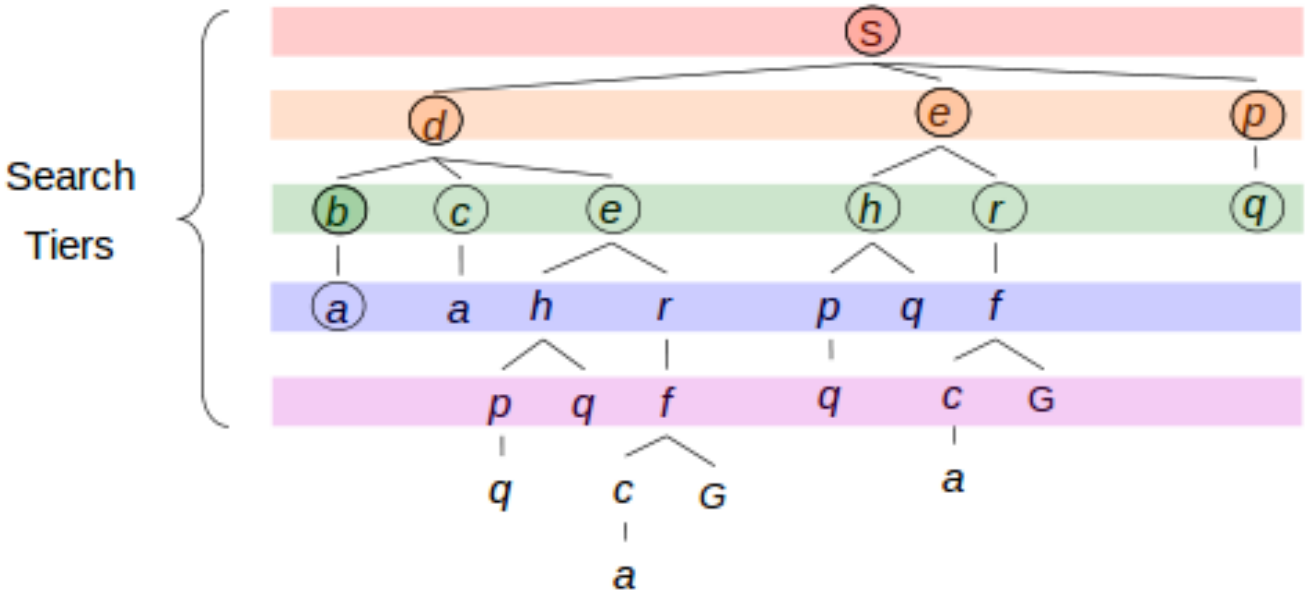


Breadth-First Search

Strategy: expand a shallowest node first

Implementation: Fringe is a FIFO queue



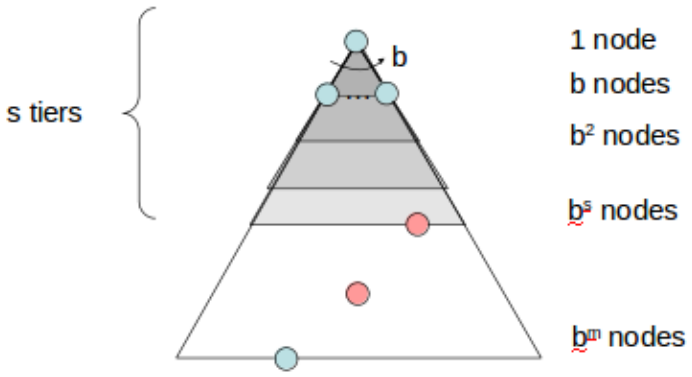


Breadth-First Search (BFS) Properties

What nodes does BFS expand? Processes all nodes above shallowest solution

Let depth of shallowest solution be s

Search takes time $O(b^s)$

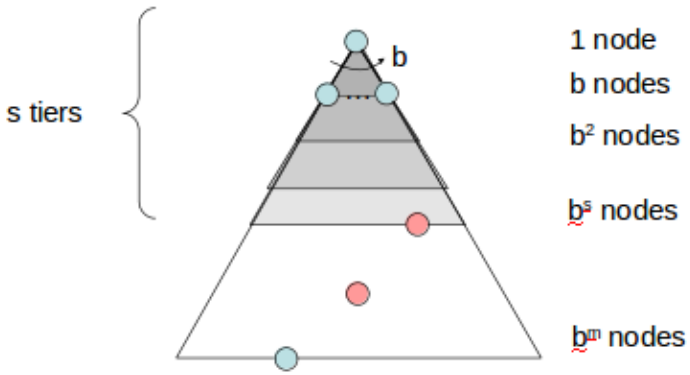


Breadth-First Search (BFS) Properties

What nodes does BFS expand? Processes all nodes above shallowest solution

Let depth of shallowest solution be s

Search takes time $O(b^s)$



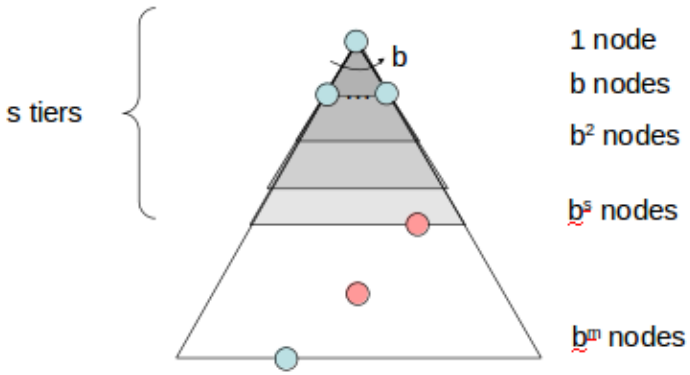
How much space does the fringe take?

Breadth-First Search (BFS) Properties

What nodes does BFS expand? Processes all nodes above shallowest solution

Let depth of shallowest solution be s

Search takes time $O(b^s)$



How much space does the fringe take?

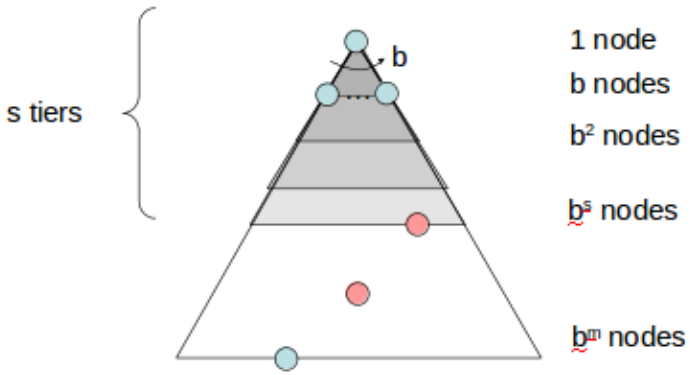
Has roughly the last tier, so $O(b^s)$

Breadth-First Search (BFS) Properties

What nodes does BFS expand? Processes all nodes above shallowest solution

Let depth of shallowest solution be s

Search takes time $O(b^s)$



How much space does the fringe take?

Has roughly the last tier, so $O(b^s)$

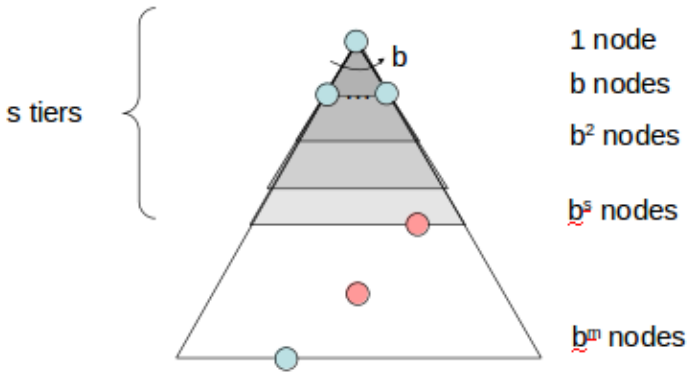
Is it complete?

Breadth-First Search (BFS) Properties

What nodes does BFS expand? Processes all nodes above shallowest solution

Let depth of shallowest solution be s

Search takes time $O(b^s)$



How much space does the fringe take?

Has roughly the last tier, so $O(b^s)$

Is it complete?

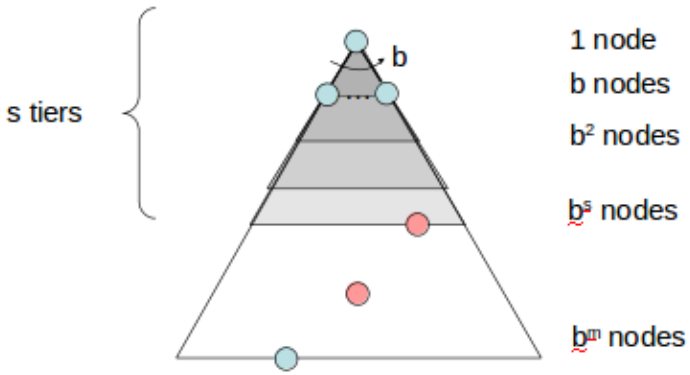
s must be finite if a solution exists,

Breadth-First Search (BFS) Properties

What nodes does BFS expand? Processes all nodes above shallowest solution

Let depth of shallowest solution be s

Search takes time $O(b^s)$



How much space does the fringe take?

Has roughly the last tier, so $O(b^s)$

Is it complete?

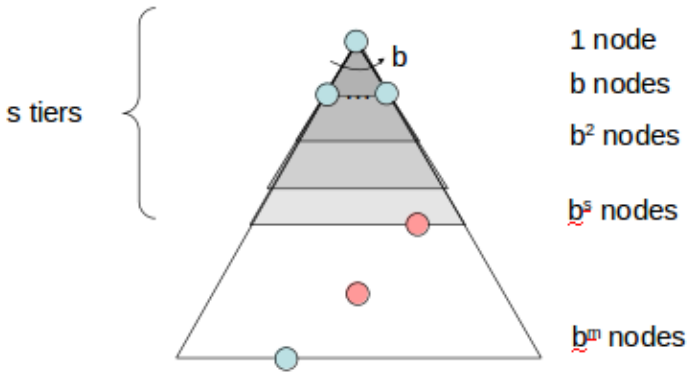
s must be finite if a solution exists, so yes!

Breadth-First Search (BFS) Properties

What nodes does BFS expand? Processes all nodes above shallowest solution

Let depth of shallowest solution be s

Search takes time $O(b^s)$



How much space does the fringe take?

Has roughly the last tier, so $O(b^s)$

Is it complete?

s must be finite if a solution exists, so yes!

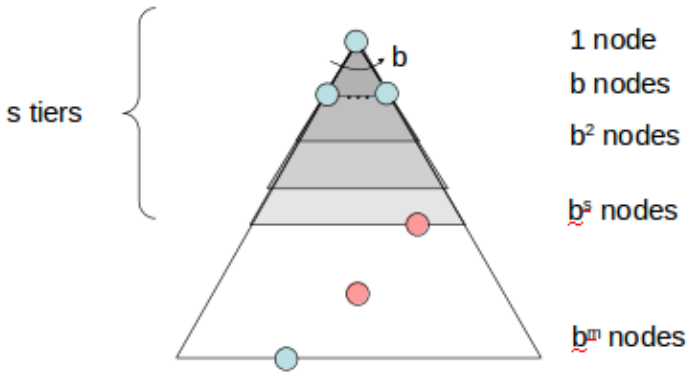
Is it optimal?

Breadth-First Search (BFS) Properties

What nodes does BFS expand? Processes all nodes above shallowest solution

Let depth of shallowest solution be s

Search takes time $O(b^s)$



How much space does the fringe take?

Has roughly the last tier, so $O(b^s)$

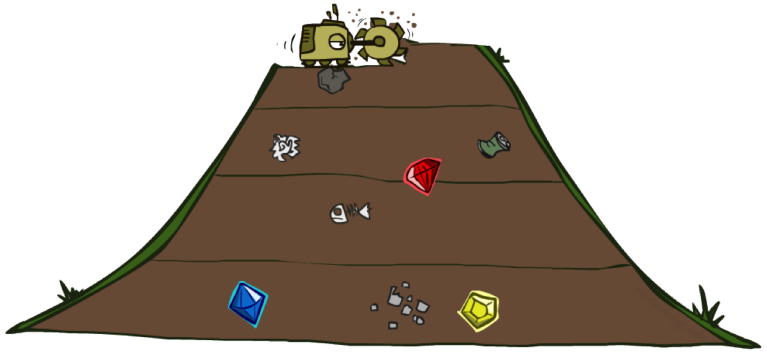
Is it complete?

s must be finite if a solution exists, so yes!

Is it optimal?

Only if costs are all 1 (more on costs later).

Quiz: DFS vs BFS



DFS vs BFS

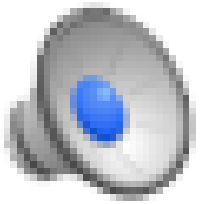
When will BFS outperform DFS?
When will DFS outperform BFS?
[Demo
:
dfs
/
bfs
maze water (L2D6)]

DFS vs BFS

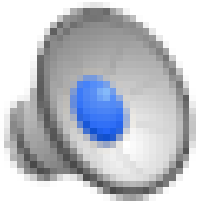
When will BFS outperform DFS?
When will DFS outperform BFS?
[Demo
:
dfs
/
bfs
maze water (L2D6)]

Space versus Time or Quality of Solution.

Video of Demo Maze Water DFS/BFS (part 1)



Video of Demo Maze Water DFS/BFS (part 2)



Next Up.

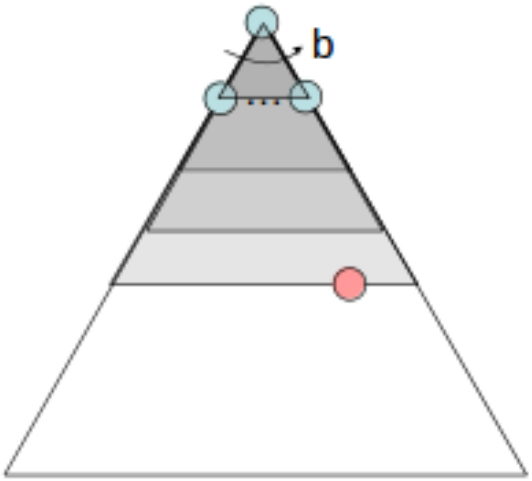
Iterative Deepening.

Next Up.

Iterative Deepening.

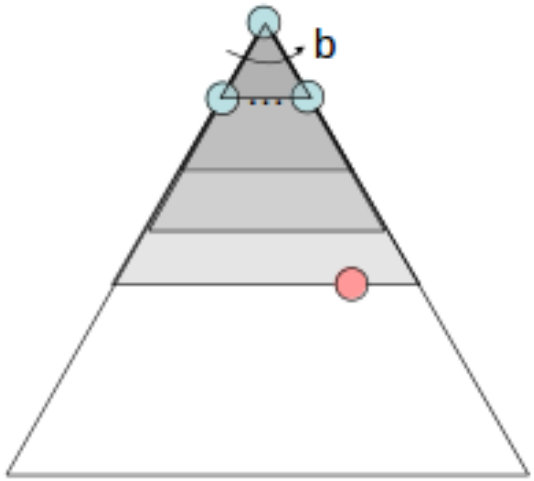
Uniform Cost Search.

Iterative Deepening



Idea: get DFS's space advantage with BFS's time / shallow-solution advantages

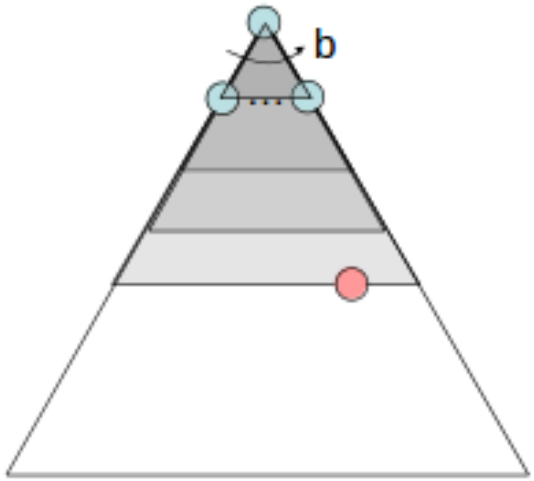
Iterative Deepening



Idea: get DFS's space advantage with BFS's time / shallow-solution advantages

Run a DFS with depth limit 1.

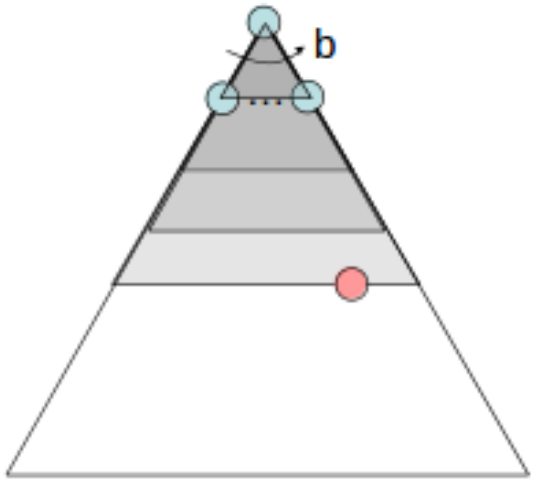
Iterative Deepening



Idea: get DFS's space advantage with BFS's time / shallow-solution advantages

Run a DFS with depth limit 1.
If no solution...

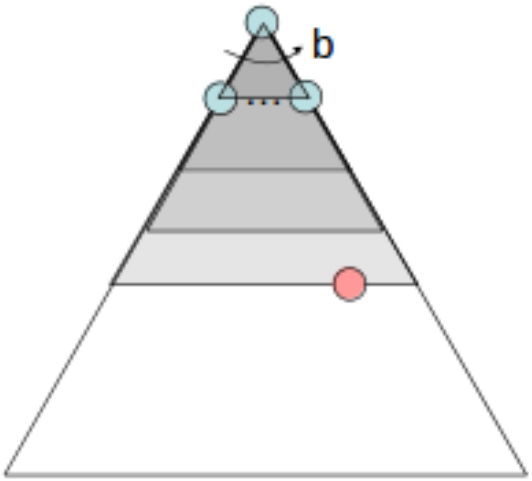
Iterative Deepening



Idea: get DFS's space advantage with BFS's time / shallow-solution advantages

Run a DFS with depth limit 1.
    If no solution...
Run a DFS with depth limit 2.

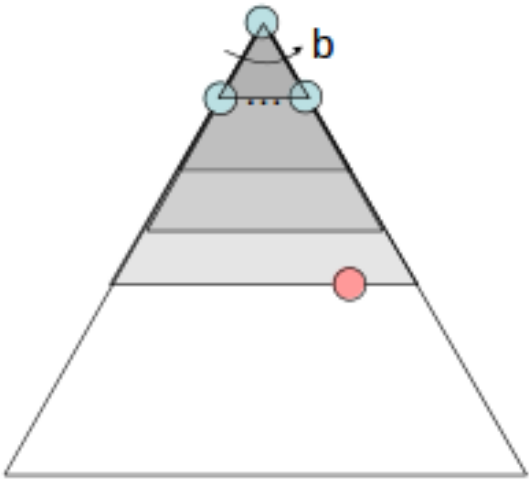
Iterative Deepening



Idea: get DFS's space advantage with BFS's time / shallow-solution advantages

Run a DFS with depth limit 1.
    If no solution...
Run a DFS with depth limit 2.
    If no solution ...

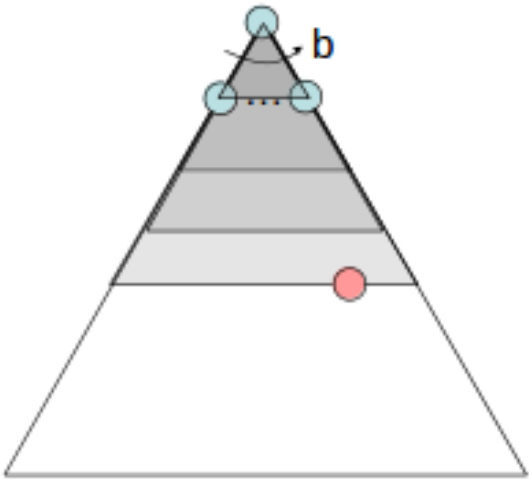
Iterative Deepening



Idea: get DFS's space advantage with BFS's time / shallow-solution advantages

Run a DFS with depth limit 1.
    If no solution...
Run a DFS with depth limit 2.
    If no solution ...
Run a DFS with depth limit 3. ....

Iterative Deepening

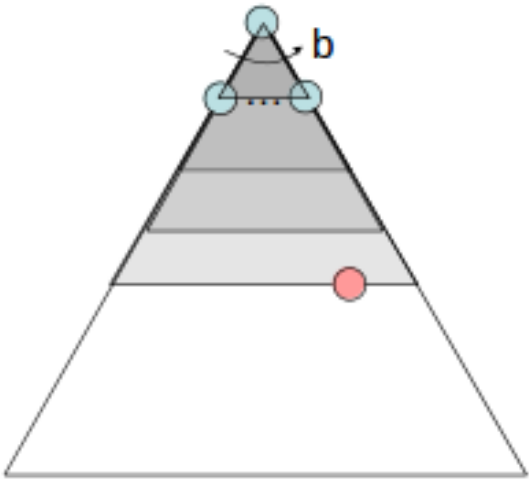


Idea: get DFS's space advantage with BFS's time / shallow-solution advantages

Run a DFS with depth limit 1.
    If no solution...
Run a DFS with depth limit 2.
    If no solution ...
Run a DFS with depth limit 3. ....

Isn't that wastefully redundant?

Iterative Deepening



Idea: get DFS's space advantage with BFS's time / shallow-solution advantages

Run a DFS with depth limit 1.
    If no solution...
Run a DFS with depth limit 2.
    If no solution ...
Run a DFS with depth limit 3. ....

Isn't that wastefully redundant?

Generally most work happens in the lowest level searched, so not so bad!

Cost-Sensitive Search

BFS finds the shortest path in terms of number of actions.

Cost-Sensitive Search

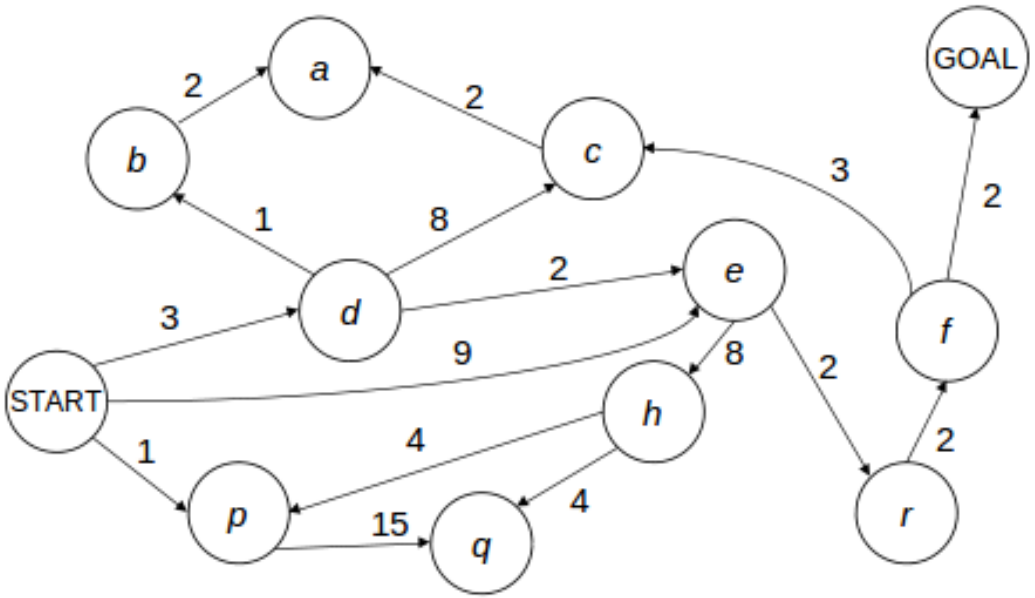
BFS finds the shortest path in terms of number of actions.

It does not find the least-cost path. We will now cover a similar algorithm which does find the least-cost path.

Cost-Sensitive Search

BFS finds the shortest path in terms of number of actions.

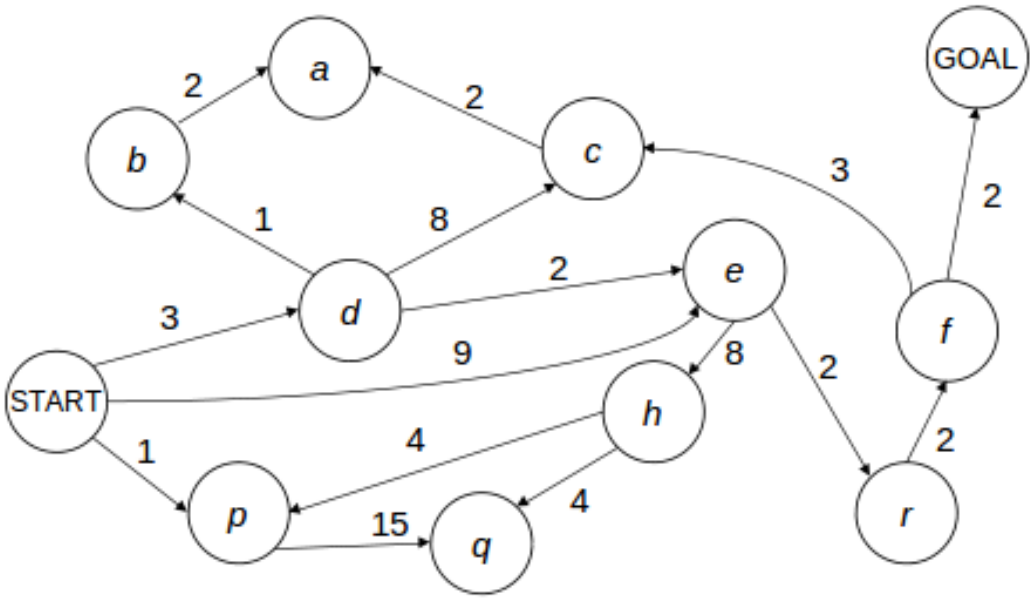
It does not find the least-cost path. We will now cover a similar algorithm which does find the least-cost path.



Cost-Sensitive Search

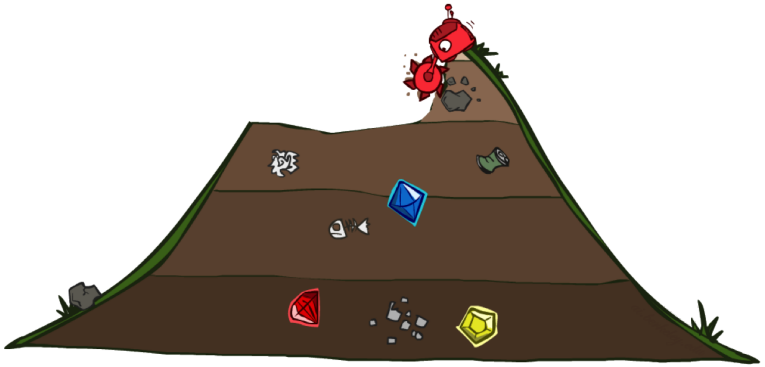
BFS finds the shortest path in terms of number of actions.

It does not find the least-cost path. We will now cover a similar algorithm which does find the least-cost path.

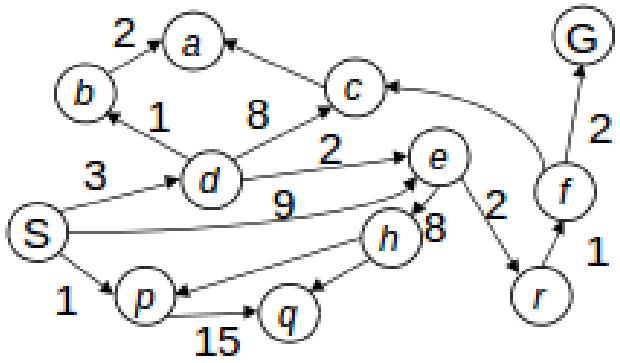


How?

Uniform Cost Search (Djikstra's algorithm.)

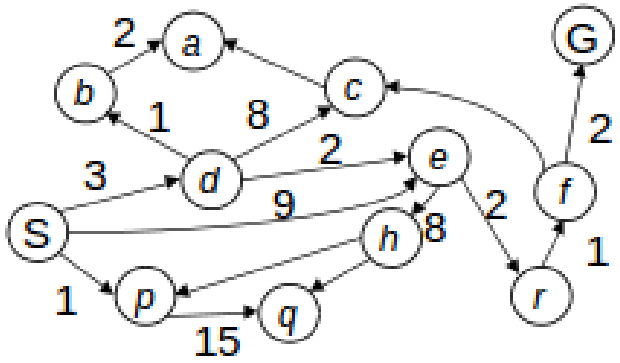


Uniform Cost Search



Strategy: expand a cheapest node first:

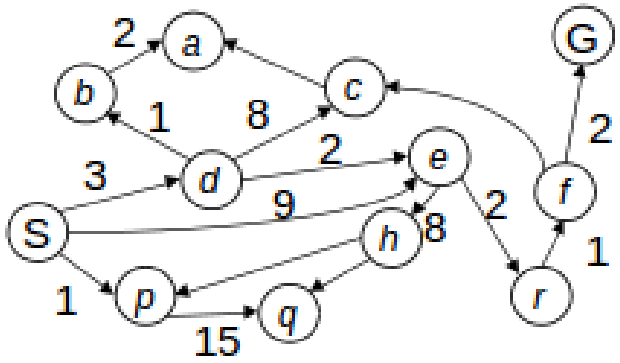
Uniform Cost Search



Strategy: expand a cheapest node first:

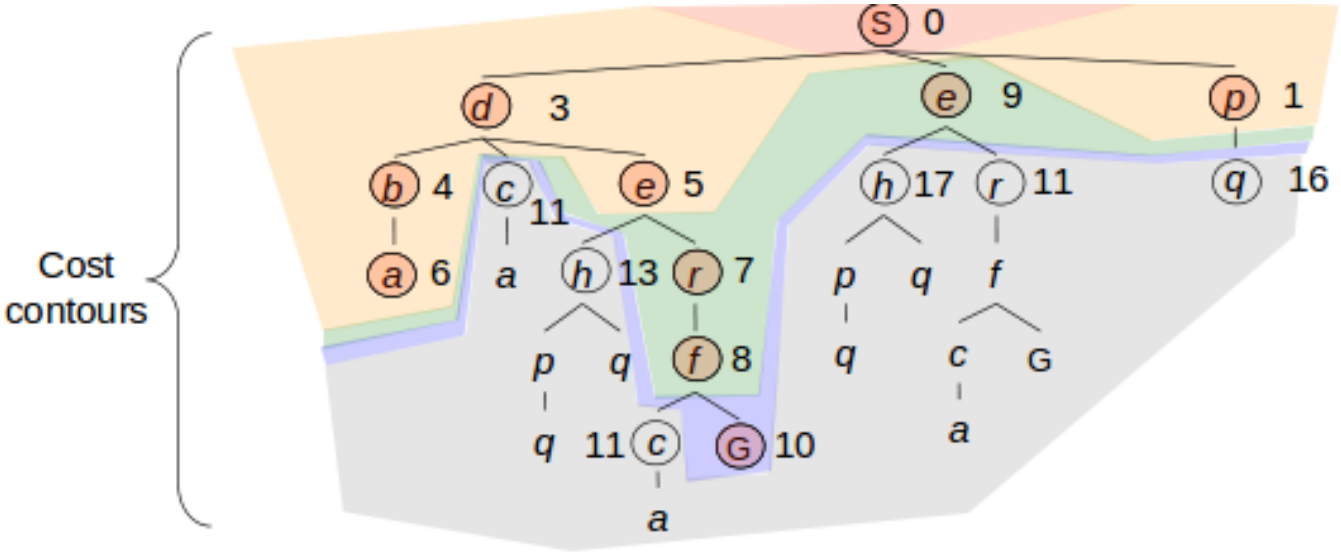
Fringe is a priority queue (priority: cumulative cost)

Uniform Cost Search



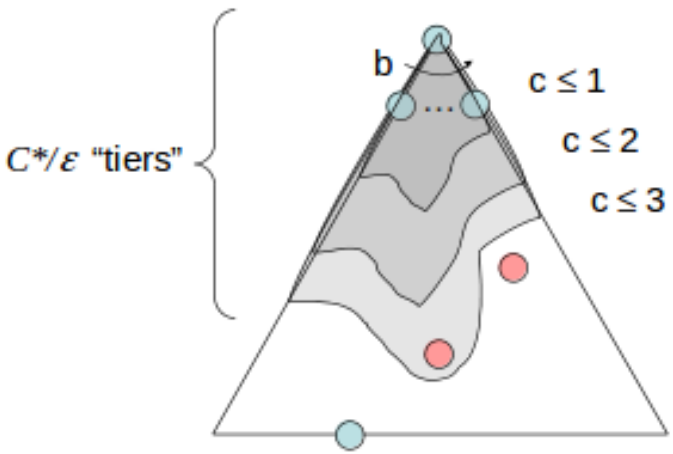
Strategy: expand a cheapest node first:

Fringe is a priority queue (priority: cumulative cost)



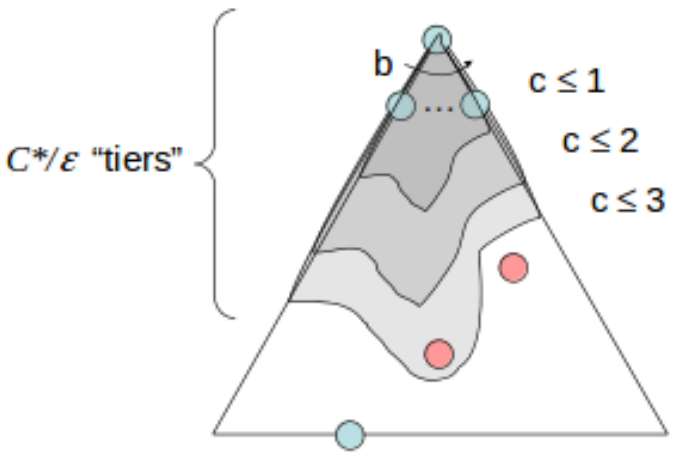
Uniform Cost Search (UCS) Properties

What nodes does UCS expand?



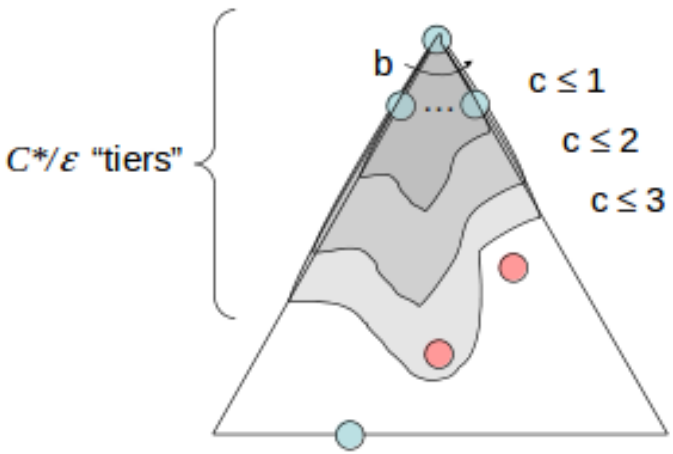
Uniform Cost Search (UCS) Properties

What nodes does UCS expand?
All nodes cheaper than solution!



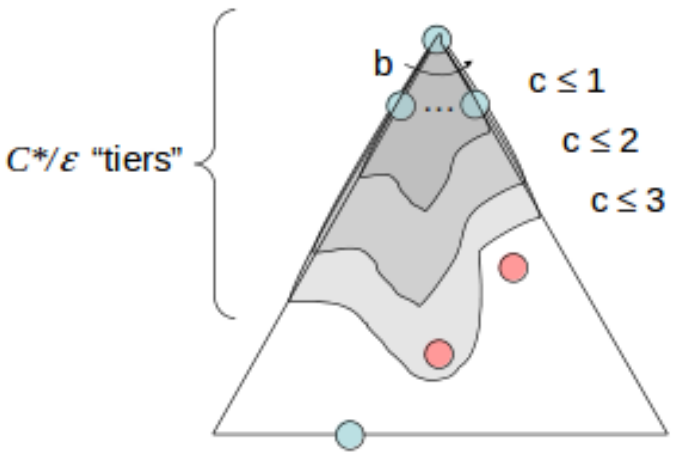
Uniform Cost Search (UCS) Properties

What nodes does UCS expand?
All nodes cheaper than solution!
Solution cost $C*$ and arc cost $\geq \varepsilon$:

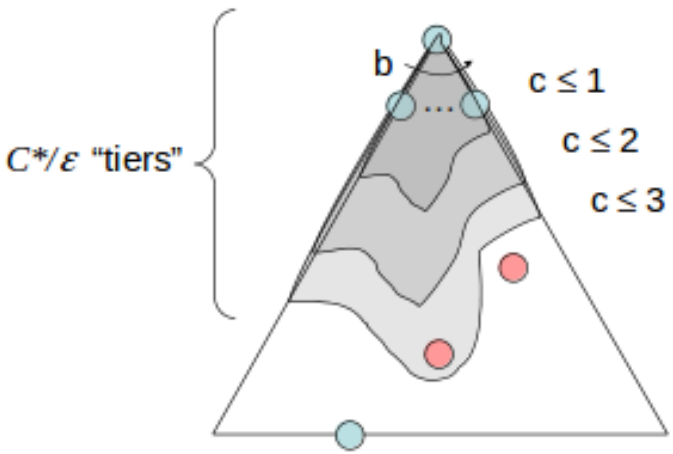


Uniform Cost Search (UCS) Properties

What nodes does UCS expand?
All nodes cheaper than solution!
Solution cost $C*$ and arc cost $\geq \varepsilon$:
depth $O(C^*/\varepsilon)$.



Uniform Cost Search (UCS) Properties



What nodes does UCS expand?

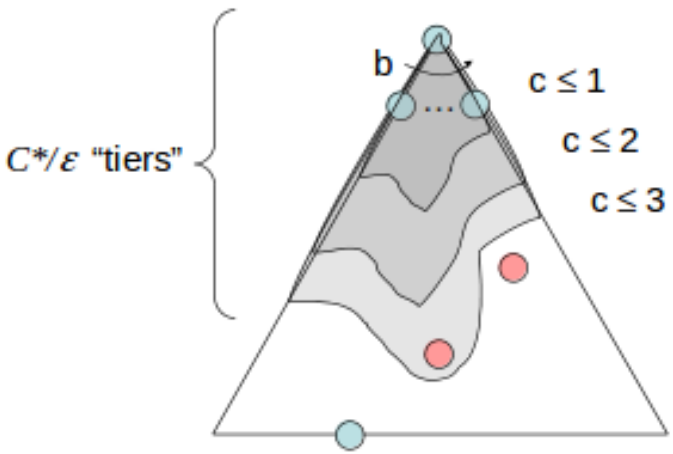
All nodes cheaper than solution!

Solution cost $C*$ and arc cost $\geq \varepsilon$:

depth $O(C^*/\varepsilon)$.

Time: $O(b^{C^*/\varepsilon})$.

Uniform Cost Search (UCS) Properties



What nodes does UCS expand?

All nodes cheaper than solution!

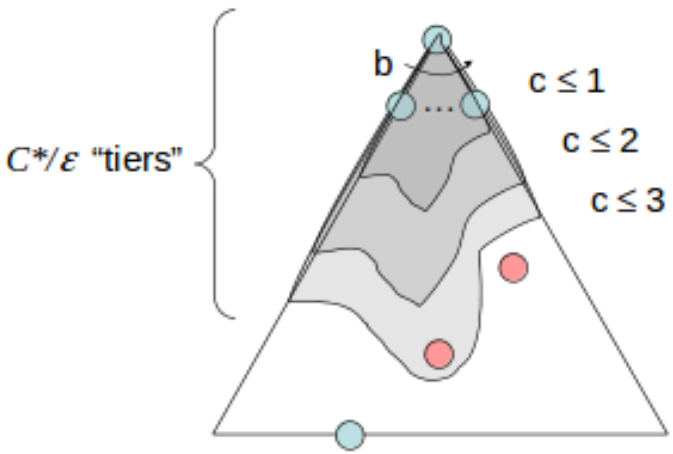
Solution cost $C*$ and arc cost $\geq \varepsilon$:

depth $O(C^*/\varepsilon)$.

Time: $O(b^{C^*/\varepsilon})$.

How much space does the fringe take?

Uniform Cost Search (UCS) Properties



What nodes does UCS expand?

All nodes cheaper than solution!

Solution cost $C*$ and arc cost $\geq \varepsilon$:

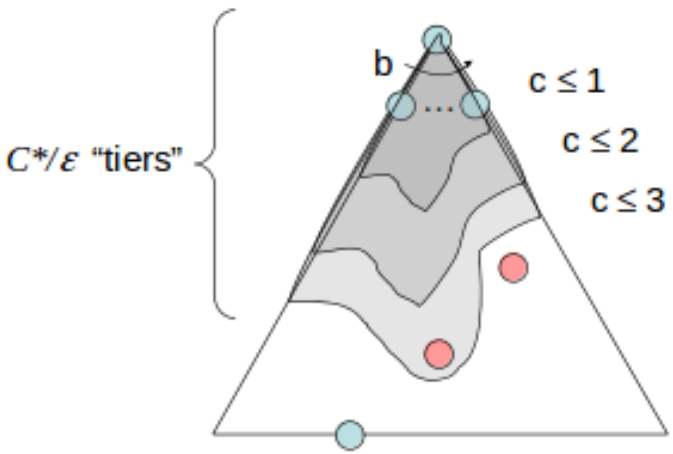
depth $O(C^*/\varepsilon)$.

Time: $O(b^{C^*/\varepsilon})$.

How much space does the fringe take?

Last Tier Space: $O(b^{C^*/\varepsilon})$.

Uniform Cost Search (UCS) Properties



What nodes does UCS expand?

All nodes cheaper than solution!

Solution cost $C*$ and arc cost $\geq \varepsilon$:

depth $O(C^*/\varepsilon)$.

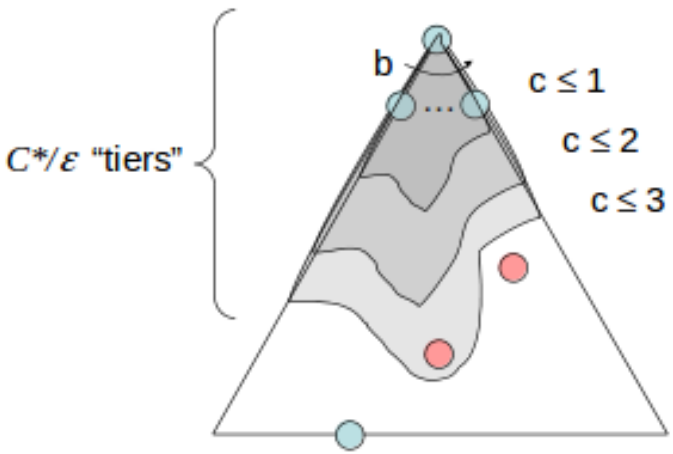
Time: $O(b^{C^*/\varepsilon})$.

How much space does the fringe take?

Last Tier Space: $O(b^{C^*/\varepsilon})$.

Is it complete?

Uniform Cost Search (UCS) Properties



What nodes does UCS expand?

All nodes cheaper than solution!

Solution cost $C*$ and arc cost $\geq \varepsilon$:

depth $O(C^*/\varepsilon)$.

Time: $O(b^{C^*/\varepsilon})$.

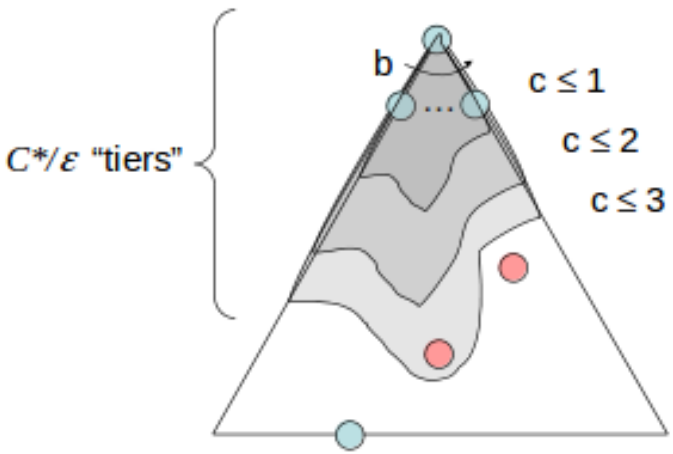
How much space does the fringe take?

Last Tier Space: $O(b^{C^*/\varepsilon})$.

Is it complete?

Finite solution cost/positive arc weights?

Uniform Cost Search (UCS) Properties



What nodes does UCS expand?

All nodes cheaper than solution!

Solution cost $C*$ and arc cost $\geq \varepsilon$:

depth $O(C^*/\varepsilon)$.

Time: $O(b^{C^*/\varepsilon})$.

How much space does the fringe take?

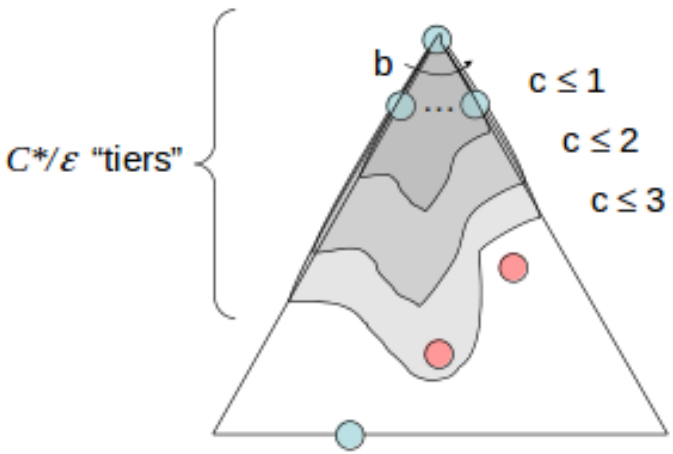
Last Tier Space: $O(b^{C^*/\varepsilon})$.

Is it complete?

Finite solution cost/positive arc weights?

Then yes.

Uniform Cost Search (UCS) Properties



What nodes does UCS expand?

All nodes cheaper than solution!

Solution cost $C*$ and arc cost $\geq \varepsilon$:

depth $O(C^*/\varepsilon)$.

Time: $O(b^{C^*/\varepsilon})$.

How much space does the fringe take?

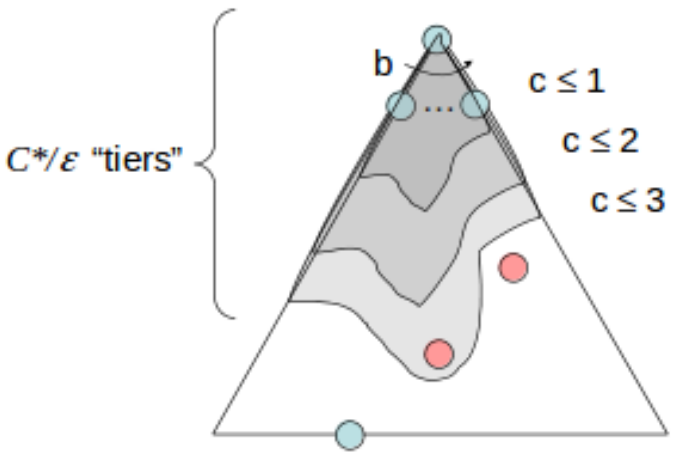
Last Tier Space: $O(b^{C^*/\varepsilon})$.

Is it complete?

Finite solution cost/positive arc weights?

Then yes.

Uniform Cost Search (UCS) Properties



What nodes does UCS expand?

All nodes cheaper than solution!

Solution cost $C*$ and arc cost $\geq \varepsilon$:

depth $O(C^*/\varepsilon)$.

Time: $O(b^{C^*/\varepsilon})$.

How much space does the fringe take?

Last Tier Space: $O(b^{C^*/\varepsilon})$.

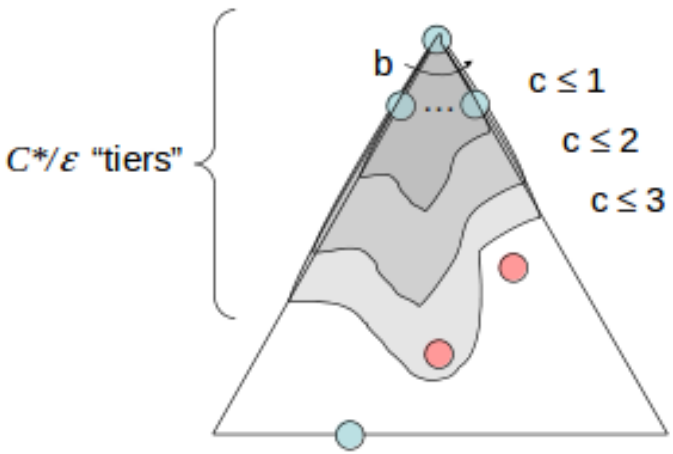
Is it complete?

Finite solution cost/positive arc weights?

Then yes.

Is it optimal?

Uniform Cost Search (UCS) Properties



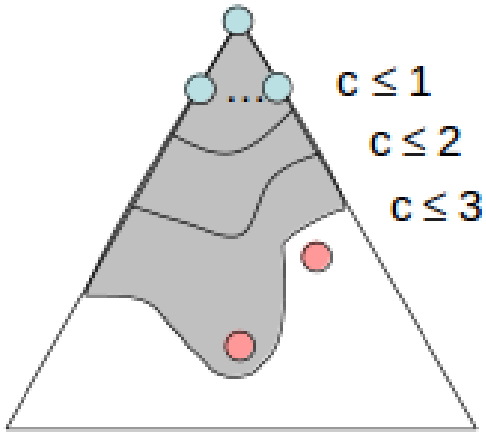
What nodes does UCS expand?
All nodes cheaper than solution!
Solution cost $C*$ and arc cost $\geq \varepsilon$:
depth $O(C^*/\varepsilon)$.
Time: $O(b^{C^*/\varepsilon})$.

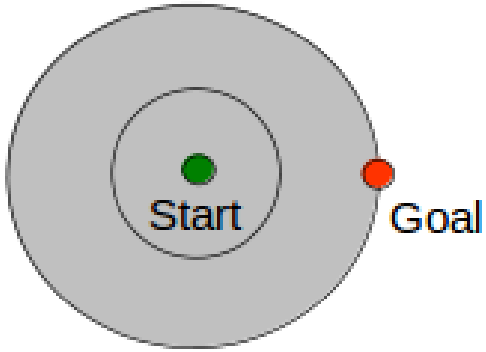
How much space does the fringe take?
Last Tier Space: $O(b^{C^*/\varepsilon})$.

Is it complete?
Finite solution cost/positive arc weights?
Then yes.

Is it optimal?
Yes.

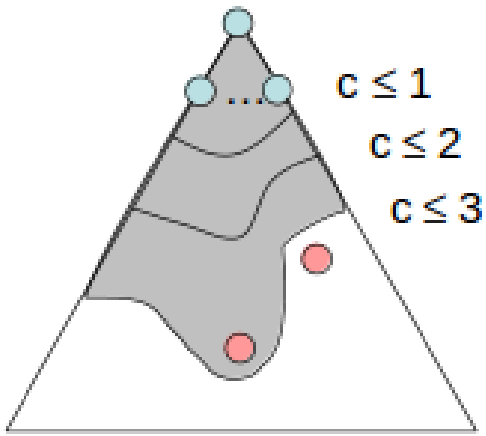
Uniform Cost Issues

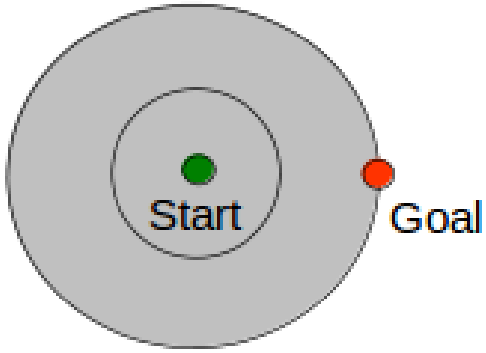




Remember: UCS explores increasing cost contours.

Uniform Cost Issues

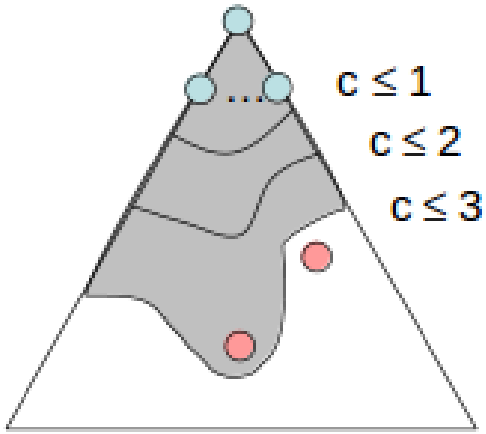


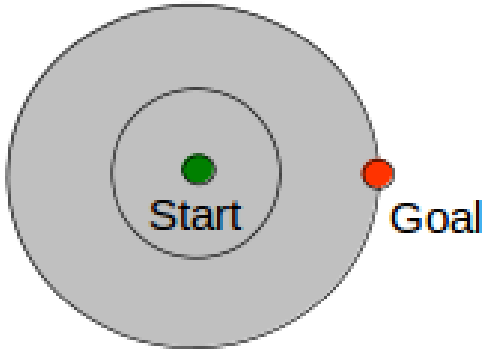


Remember: UCS explores increasing cost contours.

The good:

Uniform Cost Issues



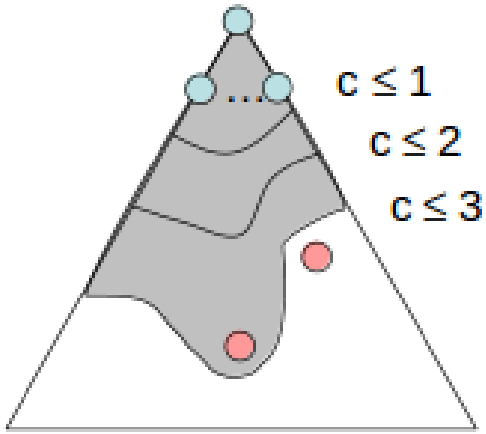


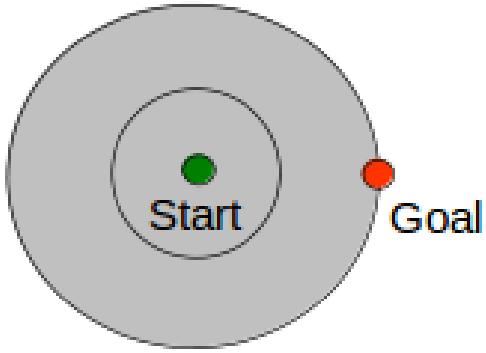
Remember: UCS explores increasing cost contours.

The good:

UCS is complete and optimal!

Uniform Cost Issues





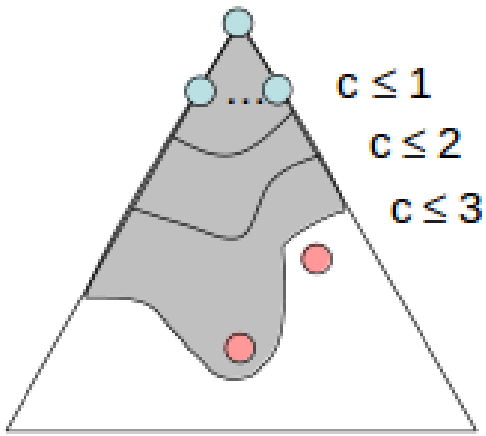
Remember: UCS explores increasing cost contours.

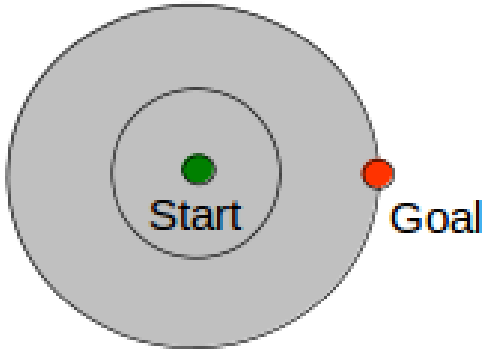
The good:

UCS is complete and optimal!

The bad:

Uniform Cost Issues



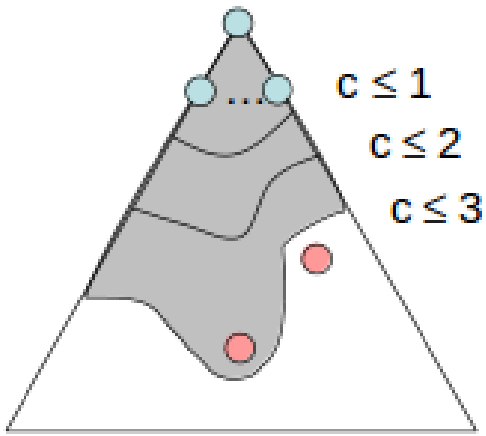


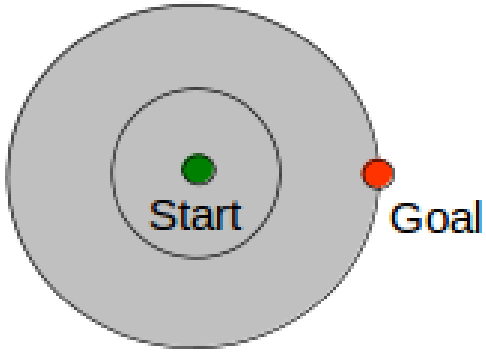
Remember: UCS explores increasing cost contours.

The good:
UCS is complete and optimal!

The bad:
Goes in every “direction”.

Uniform Cost Issues





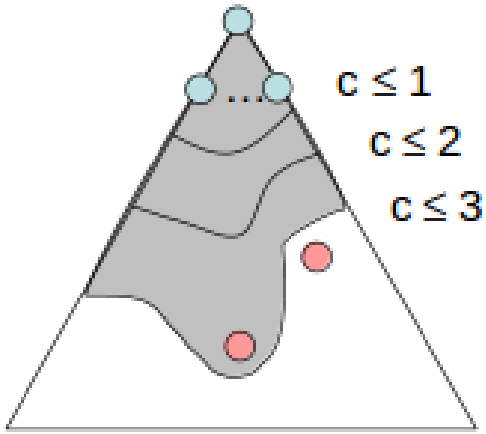
Remember: UCS explores increasing cost contours.

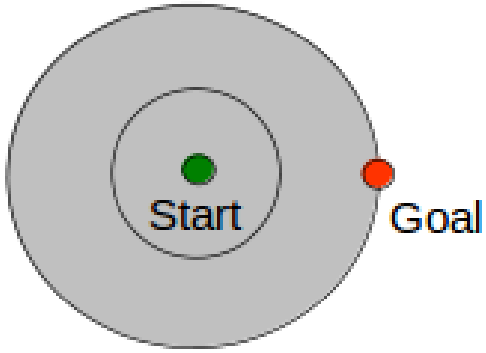
The good:
UCS is complete and optimal!

The bad:
Goes in every “direction”.

The ugly?

Uniform Cost Issues





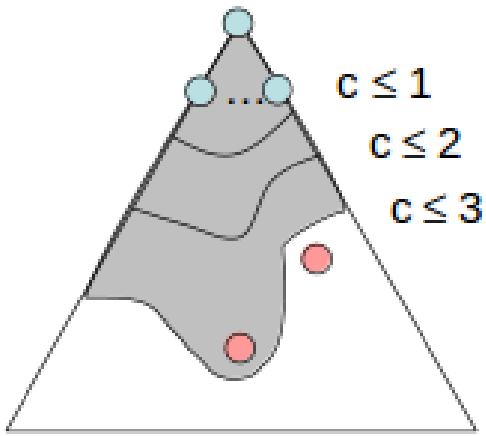
Remember: UCS explores increasing cost contours.

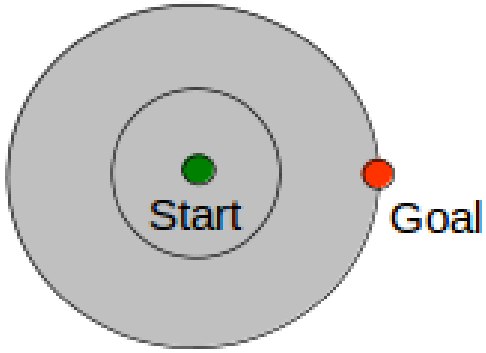
The good:
UCS is complete and optimal!

The bad:
Goes in every “direction”.

The ugly?
Huh?

Uniform Cost Issues





Remember: UCS explores increasing cost contours.

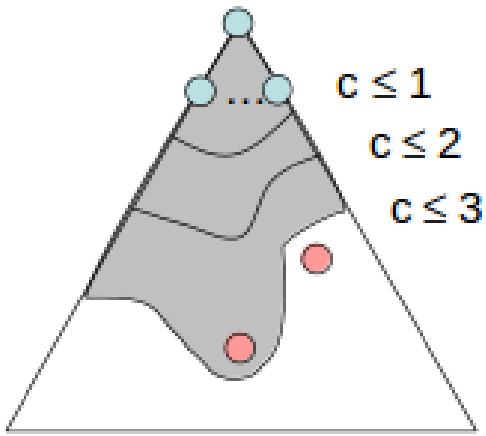
The good:
UCS is complete and optimal!

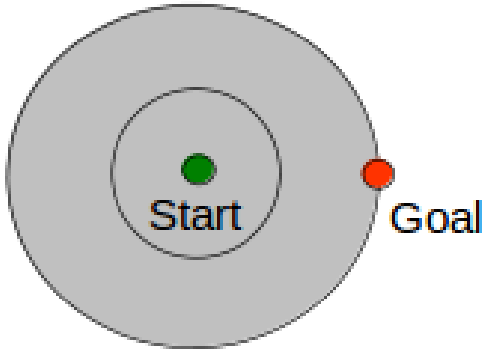
The bad:
Goes in every “direction”.

The ugly?
Huh?

No information about goal location.

Uniform Cost Issues





Remember: UCS explores increasing cost contours.

The good:
UCS is complete and optimal!

The bad:
Goes in every “direction”.

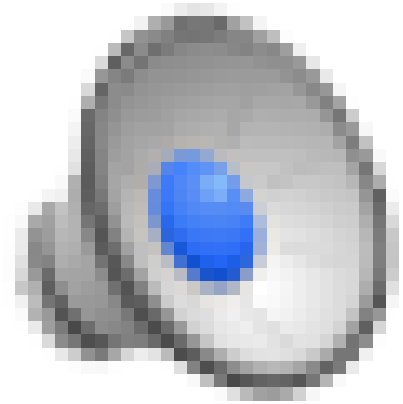
The ugly?
Huh?

No information about goal location.

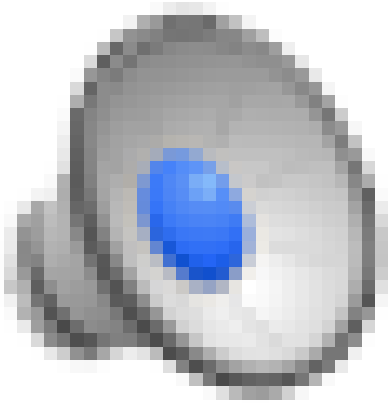
We’ll fix that soon!

[Demo: empty grid UCS (L2D5)]
[Demo: maze with deep/shallow water DFS/BFS/UCS (L2D7)]

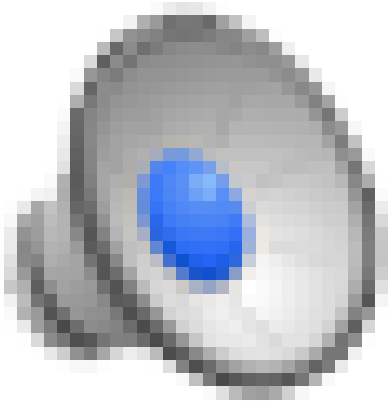
Video of Demo Empty UCS



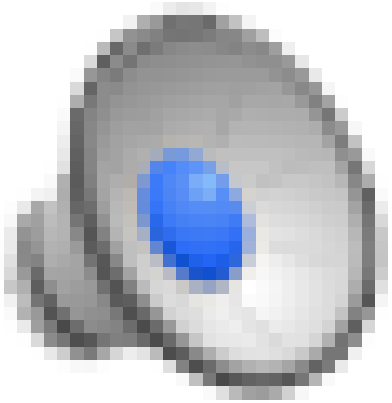
Video of Demo Maze with Deep/Shallow Water — DFS, BFS, or UCS? (part 1)



Video of Demo Maze with Deep/Shallow Water — DFS, BFS, or UCS? (part 2)



Video of Demo Maze with Deep/Shallow Water — DFS, BFS, or UCS? (part 3)



The One Queue

All these search algorithms are the same except for fringe strategies.

The One Queue

All these search algorithms are the same except for fringe strategies.

Conceptually, all fringes are priority queues (i.e. collections of nodes with attached priorities)

The One Queue

All these search algorithms are the same except for fringe strategies.

Conceptually, all fringes are priority queues (i.e. collections of nodes with attached priorities)

Practically, for DFS and BFS, you can avoid the $\log(n)$ overhead from an actual priority queue, by using stacks and queues

The One Queue

All these search algorithms are the same except for fringe strategies.

Conceptually, all fringes are priority queues (i.e. collections of nodes with attached priorities)

Practically, for DFS and BFS, you can avoid the $\log(n)$ overhead from an actual priority queue, by using stacks and queues

Can even code one implementation that takes a variable queuing object.

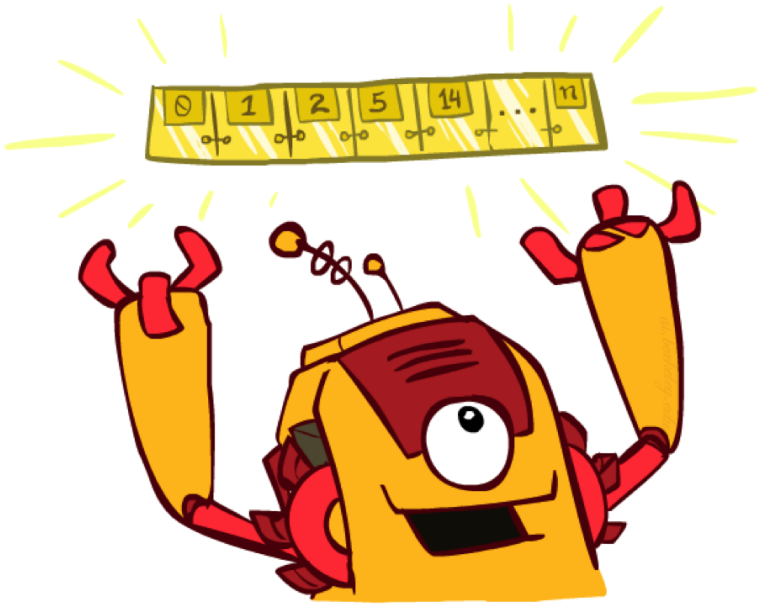
The One Queue

All these search algorithms are the same except for fringe strategies.

Conceptually, all fringes are priority queues (i.e. collections of nodes with attached priorities)

Practically, for DFS and BFS, you can avoid the $\log(n)$ overhead from an actual priority queue, by using stacks and queues

Can even code one implementation that takes a variable queuing object.



Search and Models

Search operates over models (state spaces) of the world

Search and Models

Search operates over models (state spaces) of the world

The agent doesn't actually try all the plans out in the real world!

Search and Models

Search operates over models (state spaces) of the world

The agent doesn't actually try all the plans out in the real world!

Planning is all "in simulation"

Search and Models

Search operates over models (state spaces) of the world

The agent doesn't actually try all the plans out in the real world!

Planning is all "in simulation"

Your search is only as good as your models...

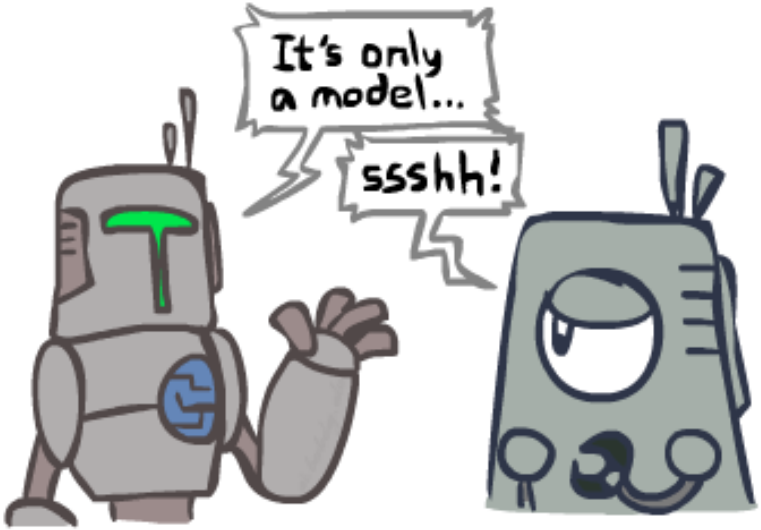
Search and Models

Search operates over models (state spaces) of the world

The agent doesn't actually try all the plans out in the real world!

Planning is all "in simulation"

Your search is only as good as your models. . .



Search Gone Wrong?

Search Gone Wrong?

Model or State Space gone wrong!

Start

Address in Norway

Place name in North Americ

Search Tips

Street Address

City: haugesund

Postal Code

End

Address in Norway

Place name in Europe

Search Tips

Street Address

City: trondheim

Postal Code

Get Directions

Units
- Miles
- Kilometers

Route Type
- Quickest
- Shortest

Map Style
- Standard
- LineDrive™

Built with MapPoint Technology

This site was created using Microsoft MapPoint technology. MapPoint technology is used

Start: Haugesund, Rogaland, Norway

Start

Address in Norway

Place name in North Americ

Search Tips

Street Address

City: haugesund

Postal Code

End

Address in Norway

Place name in Europe

Search Tips

Street Address

Units

Miles

Kilometers

Route Type

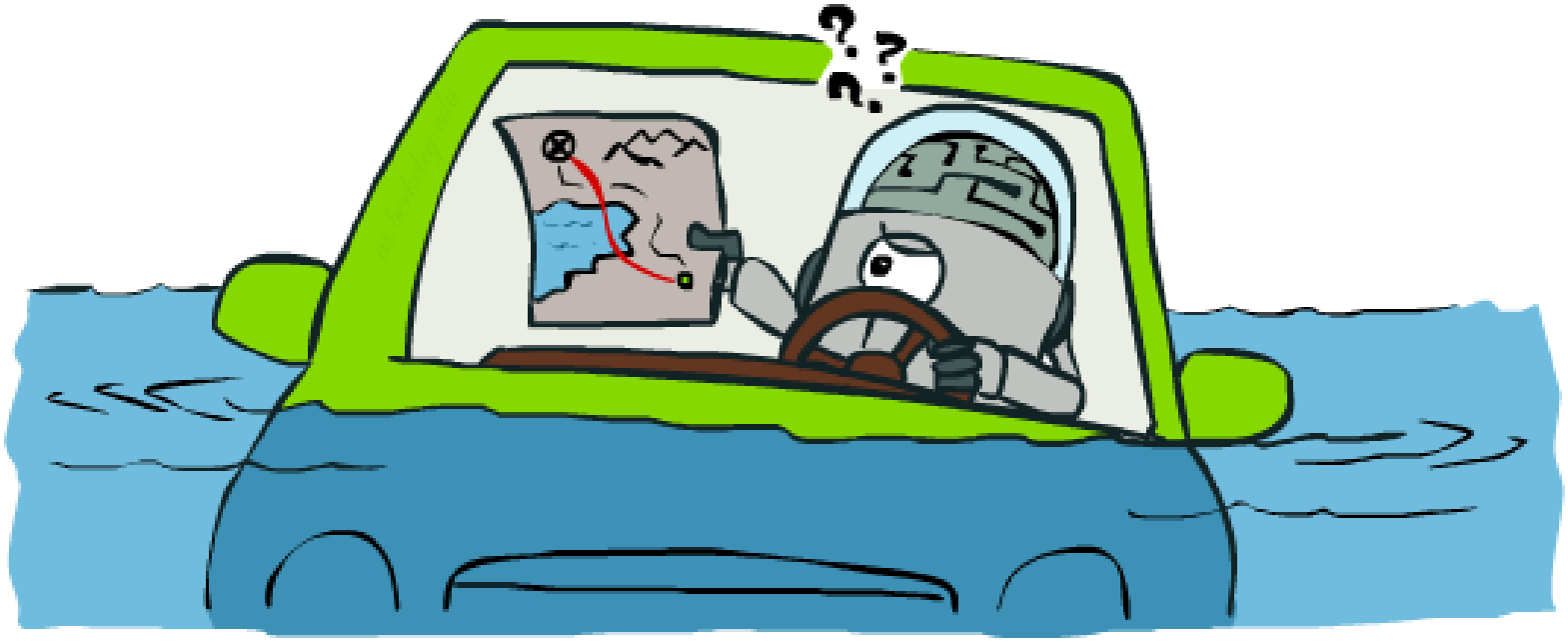
Quickest

Shortest

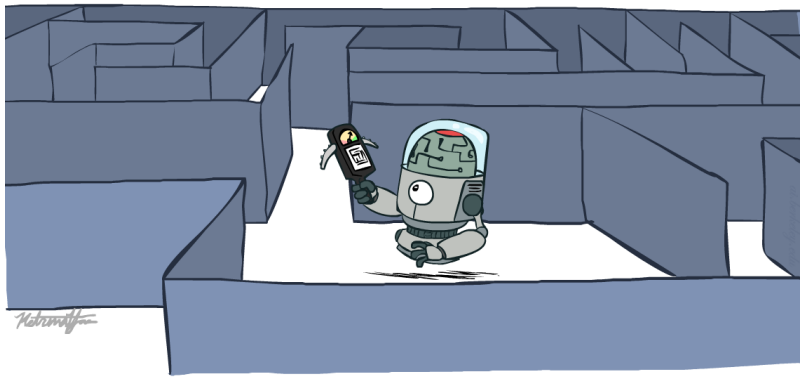
Map Style

Standard

LineDrive™



CS 188: Artificial Intelligence

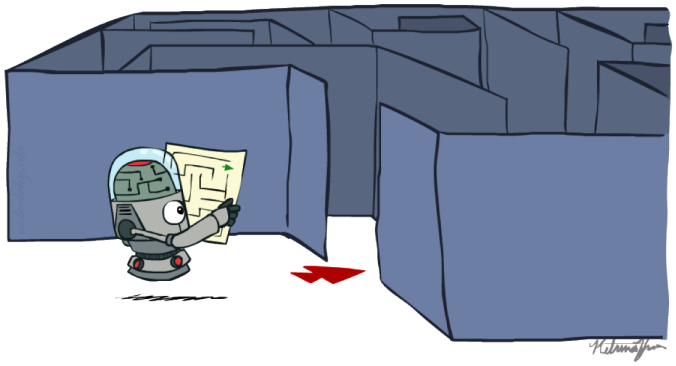


Informed Search

Informed Search

Informed Search
Heuristics
Greedy Search
A* Search
Graph Search

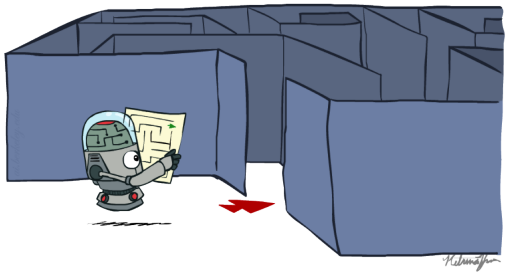
Recap: Search



Recap: Search

Search problem:

States (configurations of the world)

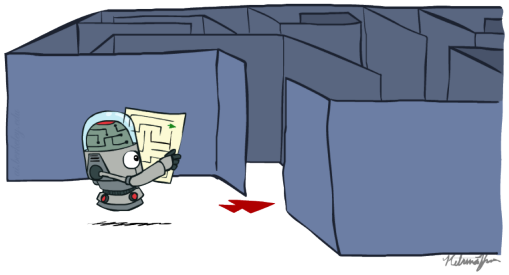


Recap: Search

Search problem:

States (configurations of the world)

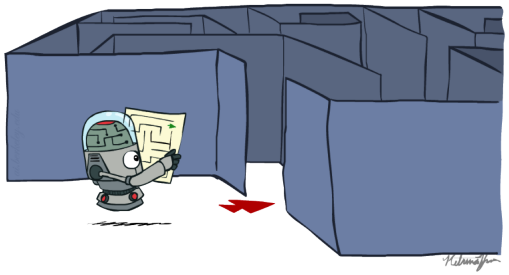
Actions and costs



Recap: Search

Search problem:

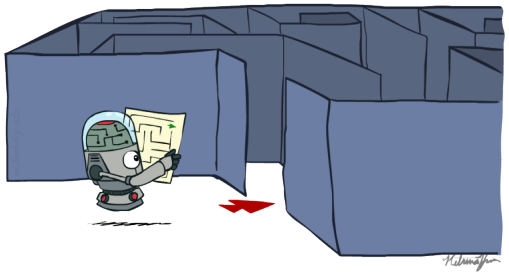
- States (configurations of the world)
- Actions and costs
- Successor function (world dynamics)



Recap: Search

Search problem:

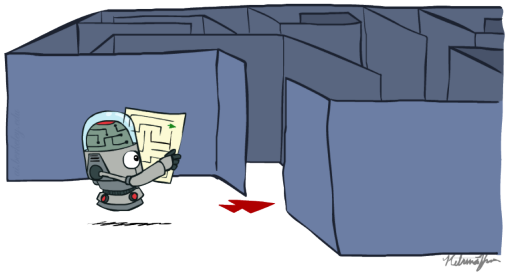
- States (configurations of the world)
- Actions and costs
- Successor function (world dynamics)
- Start state and goal test



Recap: Search

Search problem:

- States (configurations of the world)
- Actions and costs
- Successor function (world dynamics)
- Start state and goal test

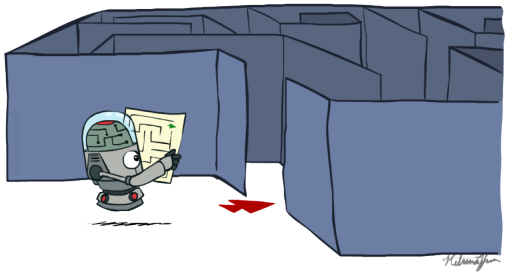


Recap: Search

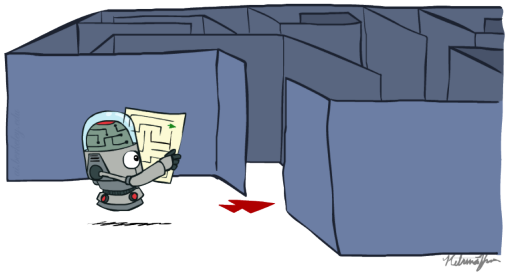
Search problem:

- States (configurations of the world)
- Actions and costs
- Successor function (world dynamics)
- Start state and goal test

Search tree:



Recap: Search



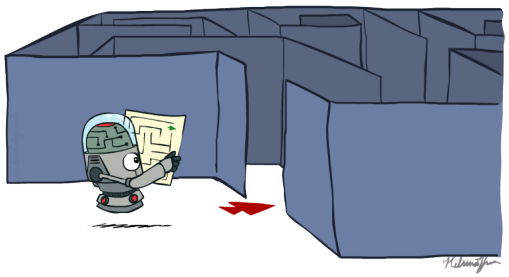
Search problem:

- States (configurations of the world)
- Actions and costs
- Successor function (world dynamics)
- Start state and goal test

Search tree:

- Nodes: represent plans for reaching states

Recap: Search



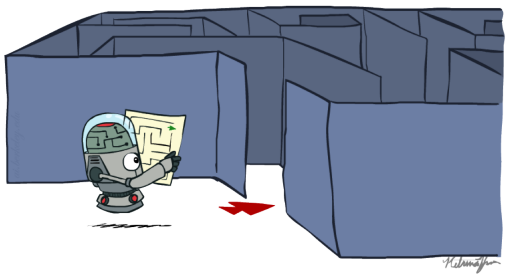
Search problem:

- States (configurations of the world)
- Actions and costs
- Successor function (world dynamics)
- Start state and goal test

Search tree:

- Nodes: represent plans for reaching states
- Plans have costs (sum of action costs)

Recap: Search



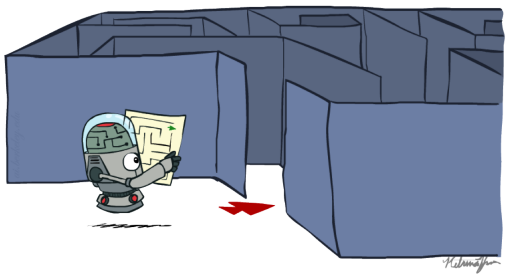
Search problem:

- States (configurations of the world)
- Actions and costs
- Successor function (world dynamics)
- Start state and goal test

Search tree:

- Nodes: represent plans for reaching states
- Plans have costs (sum of action costs)

Recap: Search



Search problem:

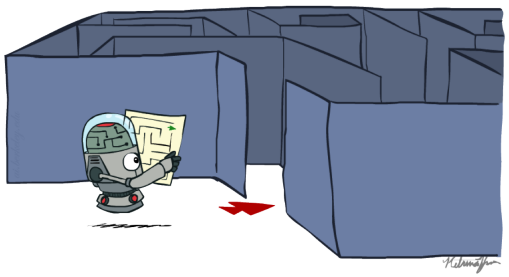
- States (configurations of the world)
- Actions and costs
- Successor function (world dynamics)
- Start state and goal test

Search tree:

- Nodes: represent plans for reaching states
- Plans have costs (sum of action costs)

Search algorithm:

Recap: Search



Search problem:

- States (configurations of the world)
- Actions and costs
- Successor function (world dynamics)
- Start state and goal test

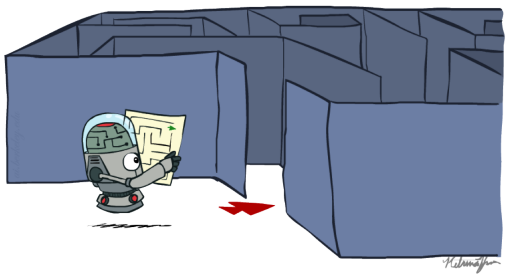
Search tree:

- Nodes: represent plans for reaching states
- Plans have costs (sum of action costs)

Search algorithm:

- Systematically builds a search tree

Recap: Search



Search problem:

- States (configurations of the world)
- Actions and costs
- Successor function (world dynamics)
- Start state and goal test

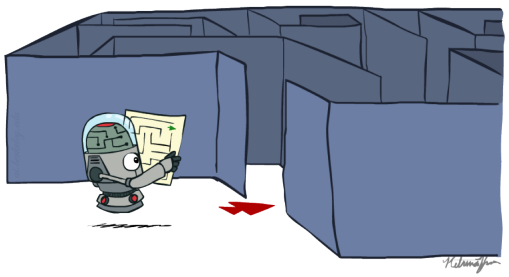
Search tree:

- Nodes: represent plans for reaching states
- Plans have costs (sum of action costs)

Search algorithm:

- Systematically builds a search tree
- Orders the fringe (unexplored nodes)

Recap: Search



Search problem:

- States (configurations of the world)
- Actions and costs
- Successor function (world dynamics)
- Start state and goal test

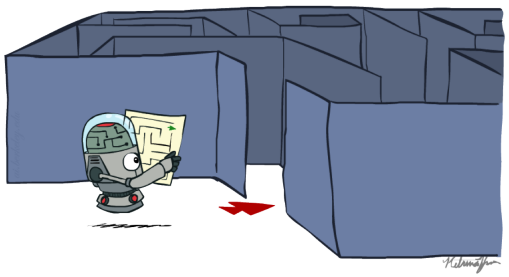
Search tree:

- Nodes: represent plans for reaching states
- Plans have costs (sum of action costs)

Search algorithm:

- Systematically builds a search tree
- Orders the fringe (unexplored nodes)

Recap: Search



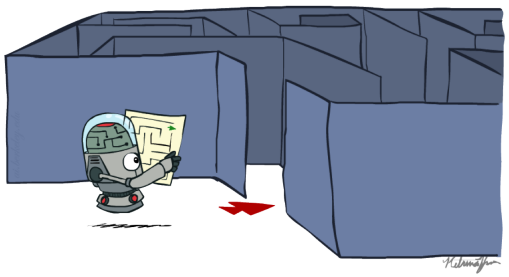
Search problem:
- States (configurations of the world)
- Actions and costs
- Successor function (world dynamics)
- Start state and goal test

Search tree:
- Nodes: represent plans for reaching states
- Plans have costs (sum of action costs)

Search algorithm:
- Systematically builds a search tree
- Orders the fringe (unexplored nodes)

Complete: finds a plan.

Recap: Search



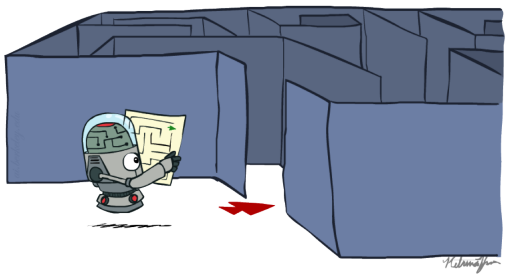
Search problem:
- States (configurations of the world)
- Actions and costs
- Successor function (world dynamics)
- Start state and goal test

Search tree:
- Nodes: represent plans for reaching states
- Plans have costs (sum of action costs)

Search algorithm:
- Systematically builds a search tree
- Orders the fringe (unexplored nodes)

Complete: finds a plan.
Optimal: finds least-cost plans

Recap: Search



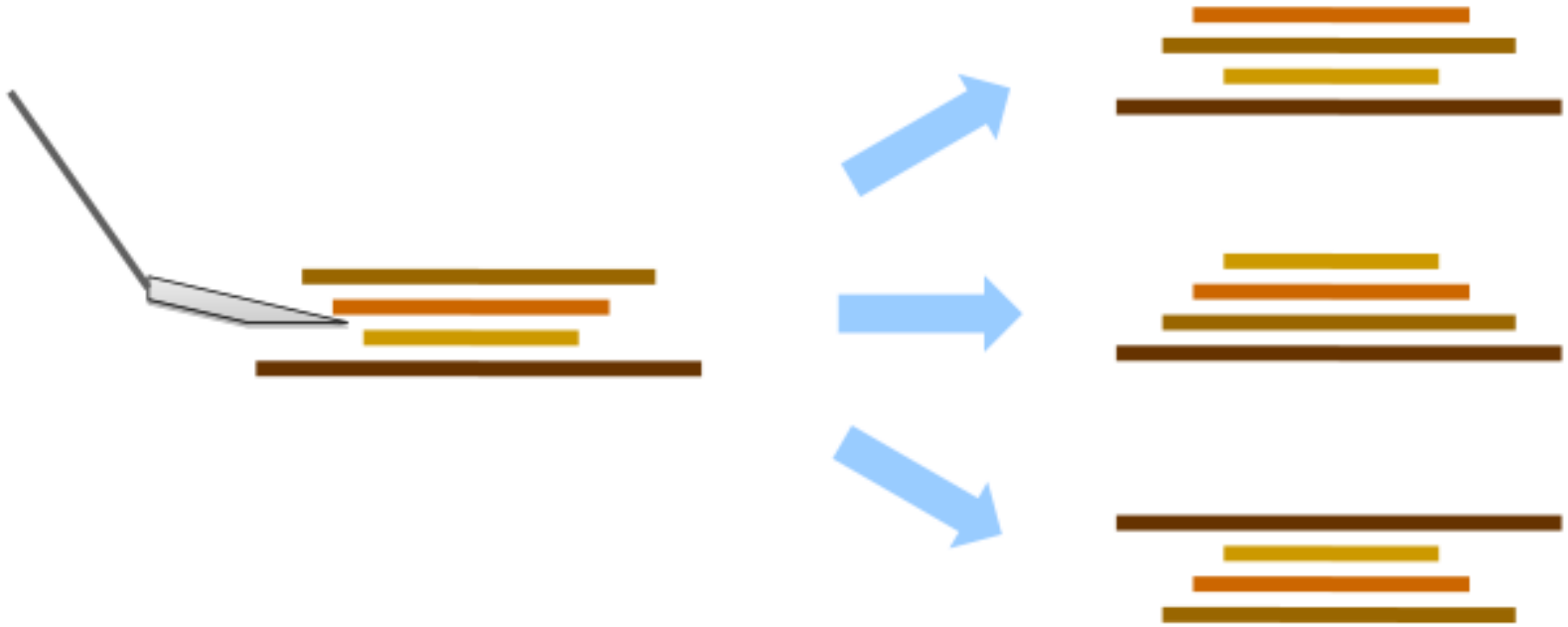
Search problem:
- States (configurations of the world)
- Actions and costs
- Successor function (world dynamics)
- Start state and goal test

Search tree:
- Nodes: represent plans for reaching states
- Plans have costs (sum of action costs)

Search algorithm:
- Systematically builds a search tree
- Orders the fringe (unexplored nodes)

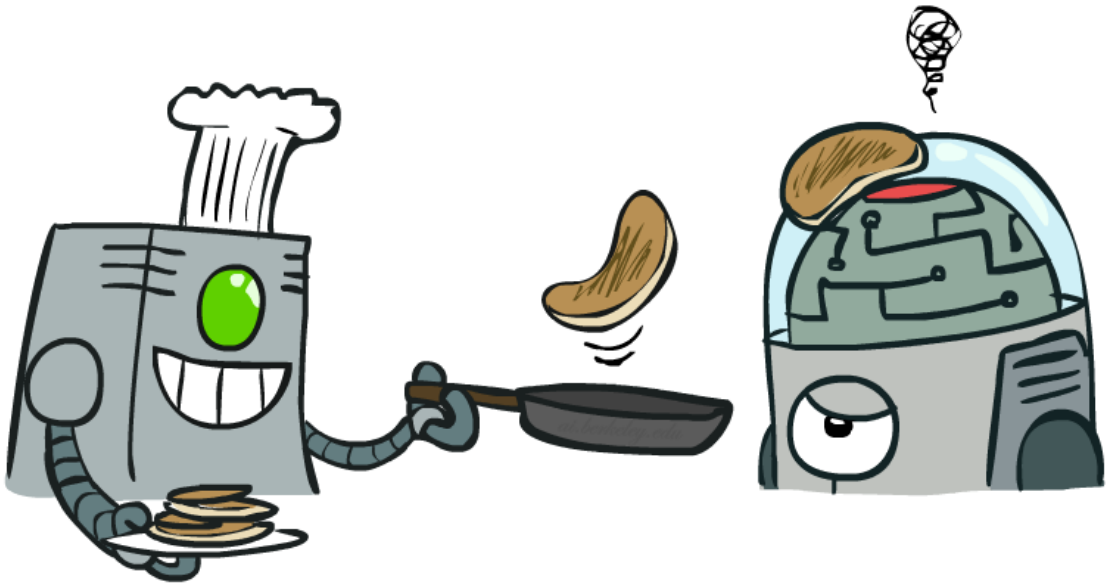
Complete: finds a plan.
Optimal: finds least-cost plans

Example: Pancake Problem



Cost: Number of pancakes flipped

Example: Pancake Problem



BOUNDS FOR SORTING BY PREFIX REVERSAL

William H. GATES

Microsoft, Albuquerque, New Mexico

Christos H. PAPADIMITRIOU*†

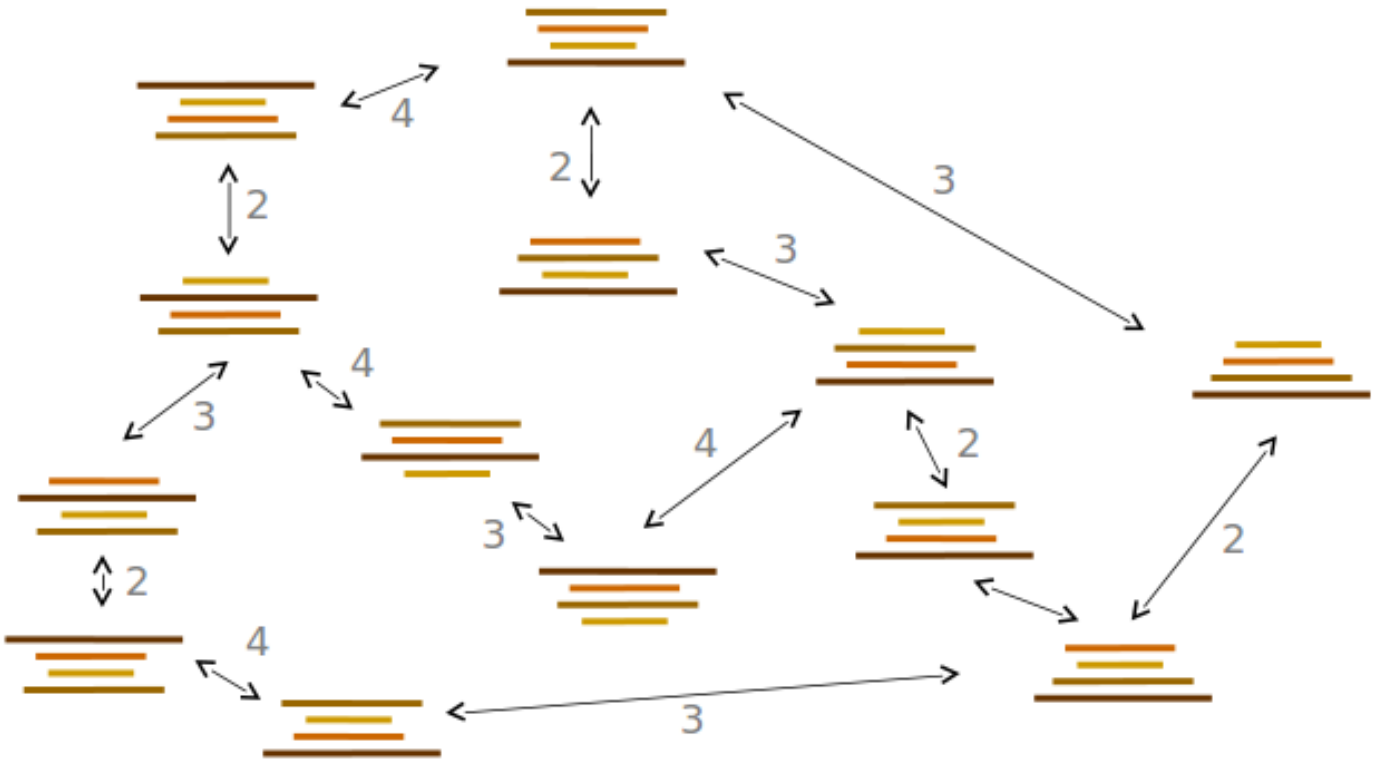
Department of Electrical Engineering, University of California, Berkeley, CA 94720, U.S.A.

Received 18 January 1978
Revised 28 August 1978

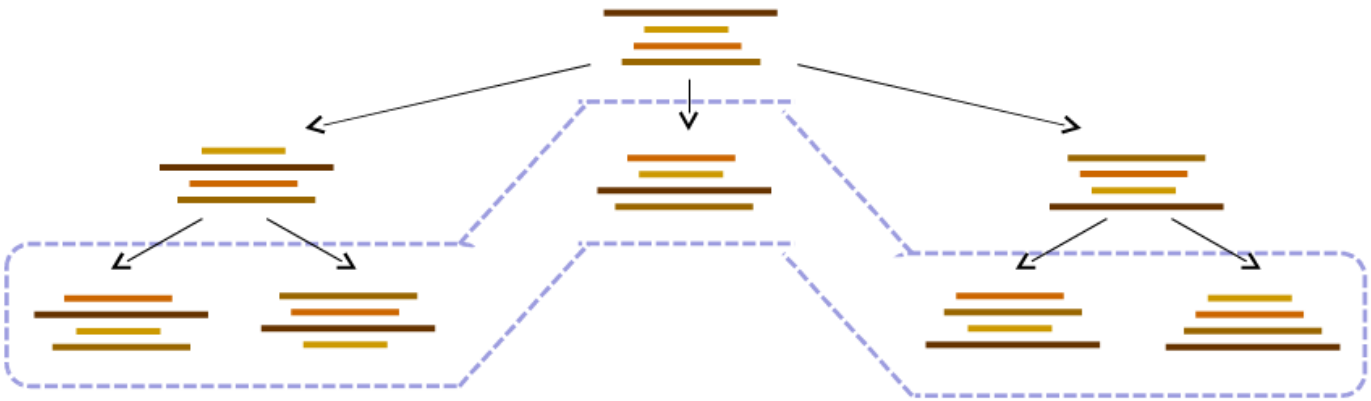
For a permutation σ of the integers from 1 to n, let $f(\sigma)$ be the smallest number of prefix reversals that will transform σ to the identity permutation, and let $f(n)$ be the largest such $f(\sigma)$ for all σ in (the symmetric group) S_n. We show that $f(n) \leqslant (5n+5)/3$, and that $f(n) \geqslant 17n/16$ for n a multiple of 16. If, furthermore, each integer is required to participate in an even number of reversed prefixes, the corresponding function $g(n)$ is shown to obey $3n/2-1 \leqslant g(n) \leqslant 2n+3$.

Example: Pancake Problem

State space graph with costs as weights.



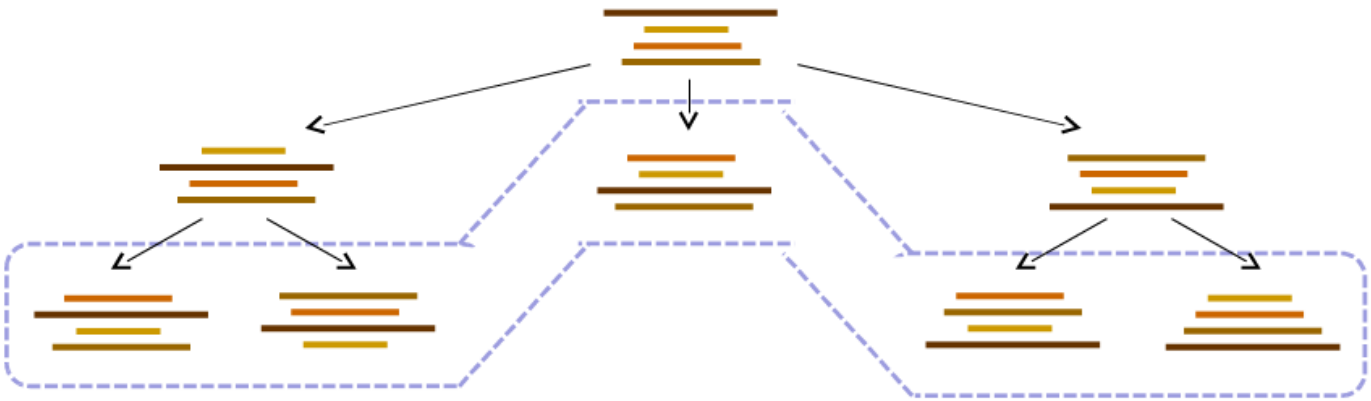
General Tree Search



Action: flip top two

```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
   initialize the search tree using the initial state of problem
   loop do
       if there are no candidates for expansion then return failure
       choose a leaf node for expansion according to strategy
       if the node contains a goal state then return the corresponding solution
       else expand the node and add the resulting nodes to the search tree
   end
```

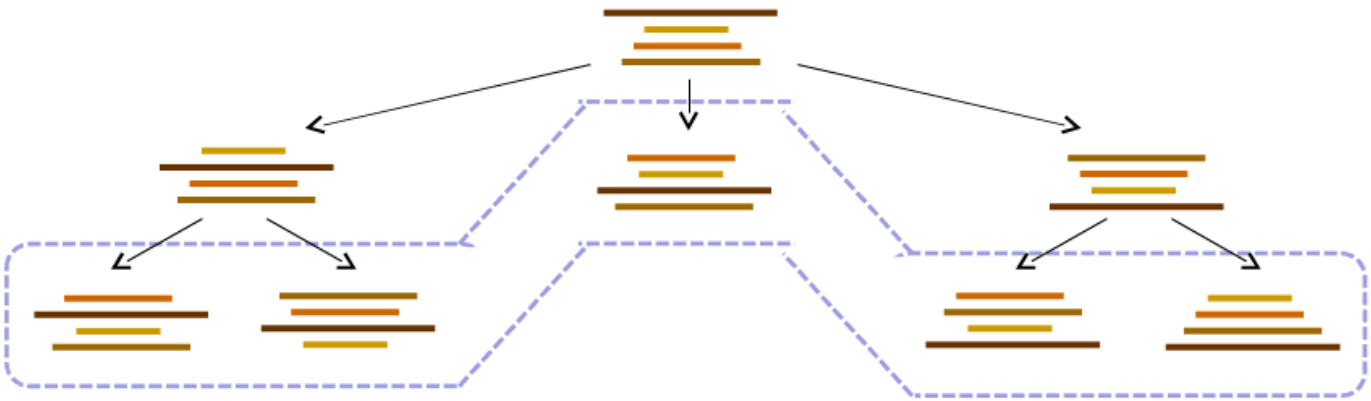
General Tree Search



Action: flip top two
Cost: 2

```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
    end
```

General Tree Search

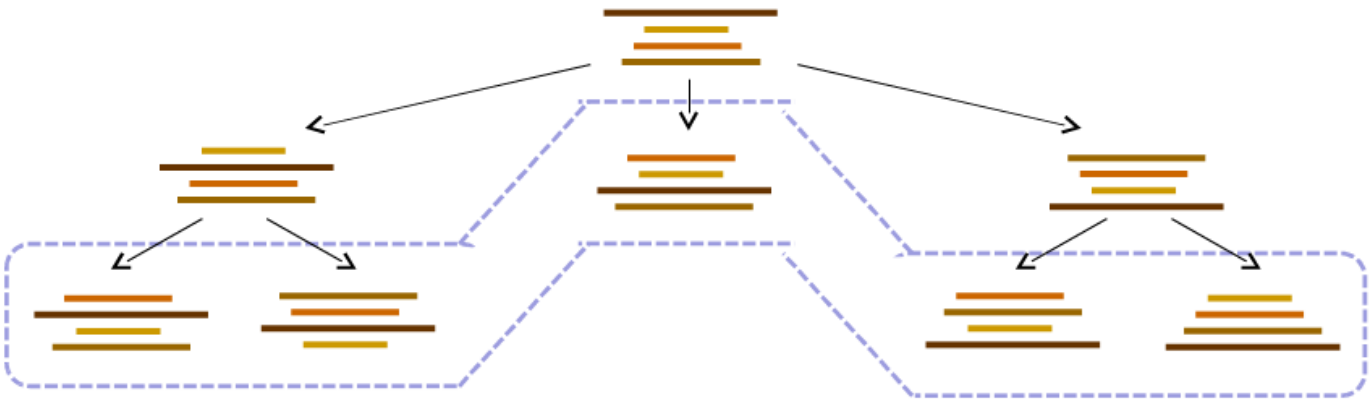


Action: flip top two
Cost: 2

Action: flip all four

```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
    end
```

General Tree Search

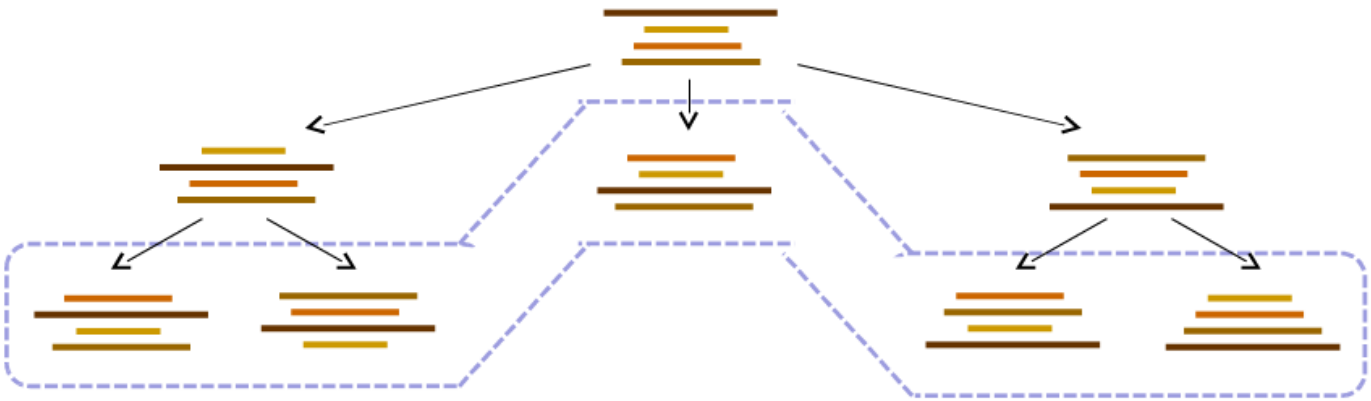


```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
    end
```

Action: flip top two
Cost: 2

Action: flip all four
Cost: 4

General Tree Search



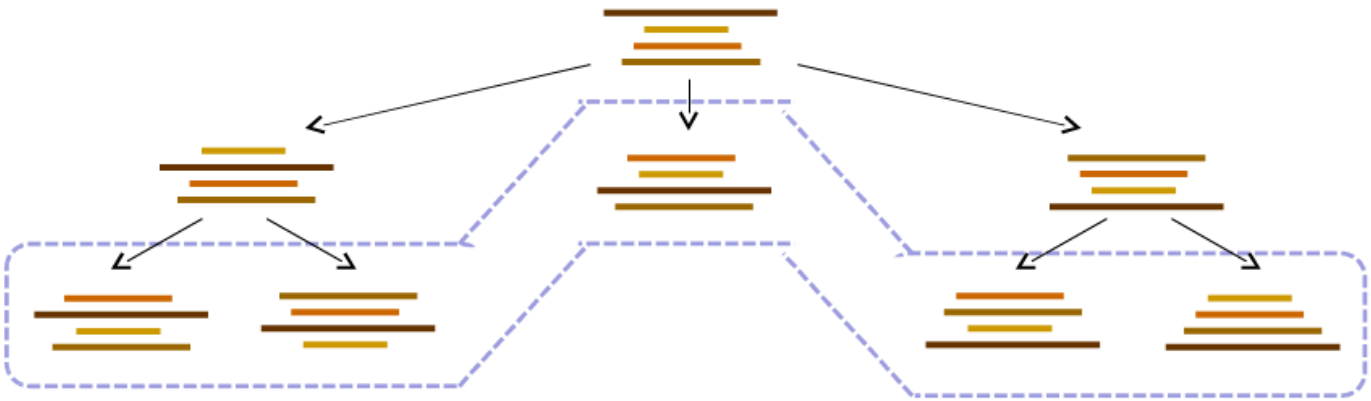
```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
    end
```

Action: flip top two
  Cost: 2

Action: flip all four
  Cost: 4

Path to reach goal:

General Tree Search



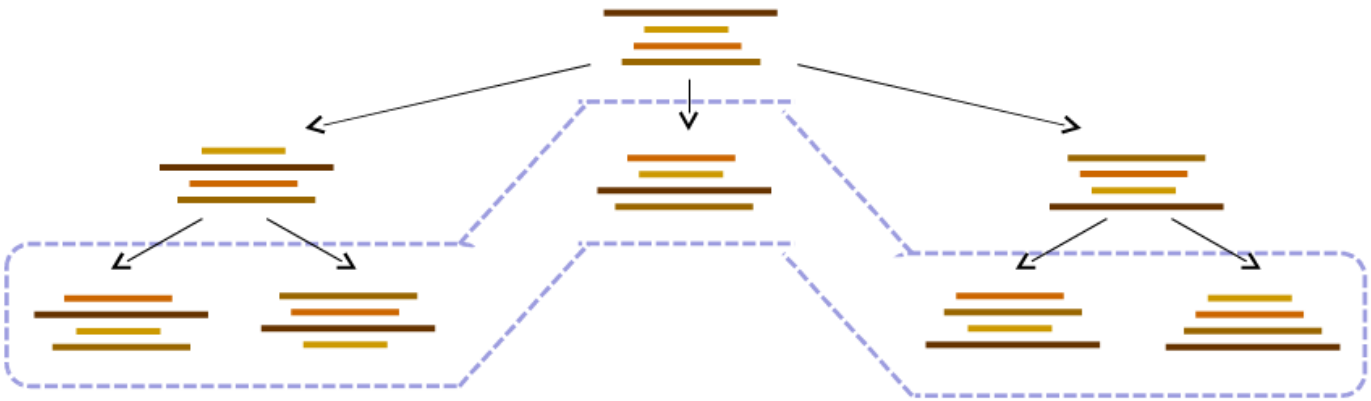
```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
    end
```

Action: flip top two
  Cost: 2

Action: flip all four
  Cost: 4

Path to reach goal:
  Flip four, flip three

General Tree Search



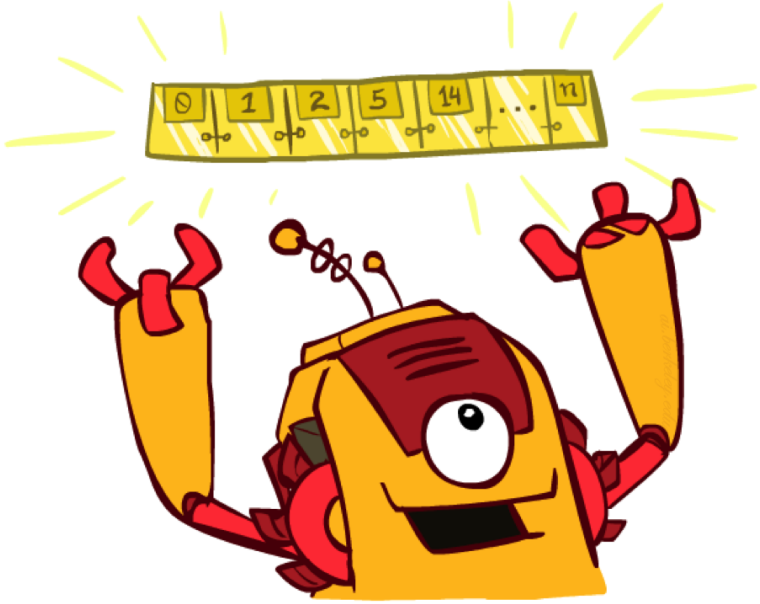
```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
    end
```

Action: flip top two
  Cost: 2

Action: flip all four
  Cost: 4

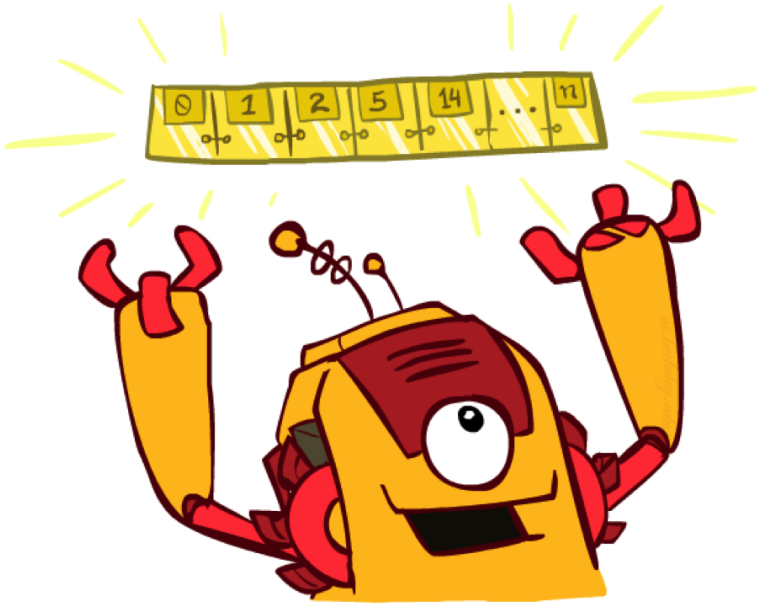
Path to reach goal:
  Flip four, flip three
Total cost: 7

The One Queue



Our search algorithms are the same except for fringe strategies

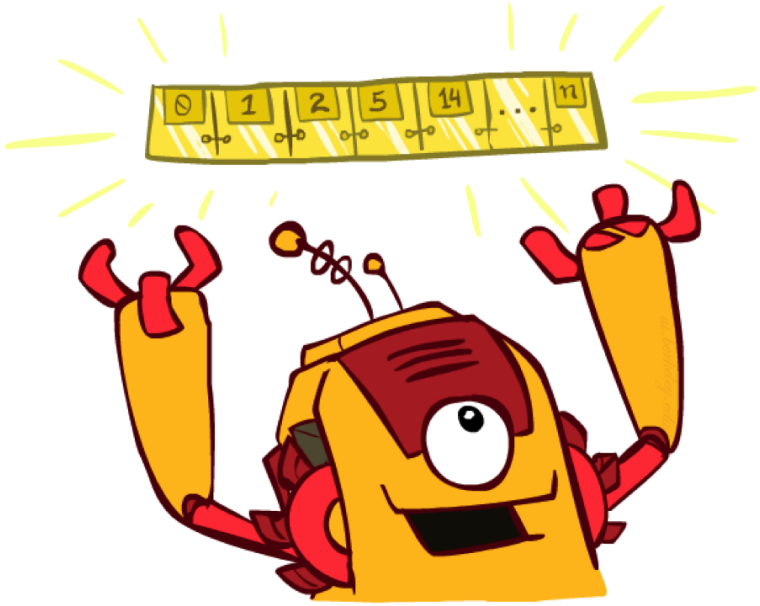
The One Queue



Our search algorithms are the same except for fringe strategies

- All fringes are priority queues: states with priorities.

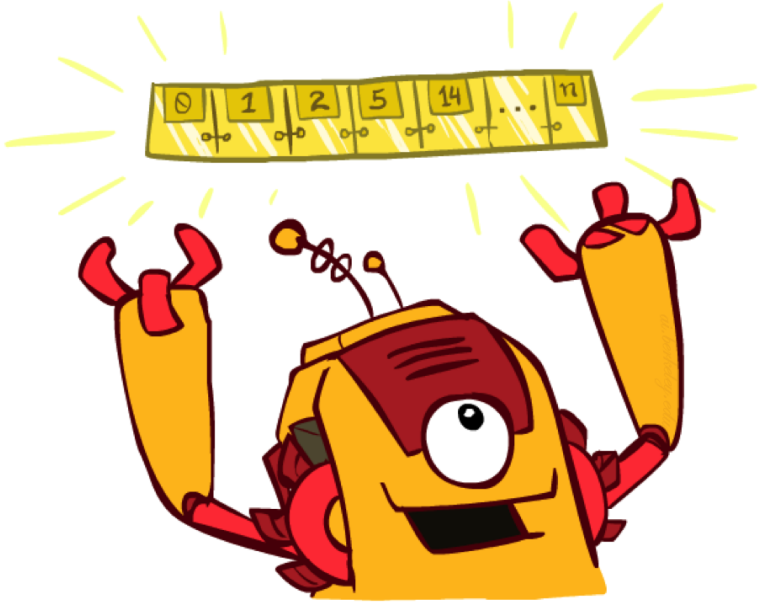
The One Queue



Our search algorithms are the same except for fringe strategies

- All fringes are priority queues: states with priorities.
- DFS, BFS a bit faster using simple stack/queues.

The One Queue

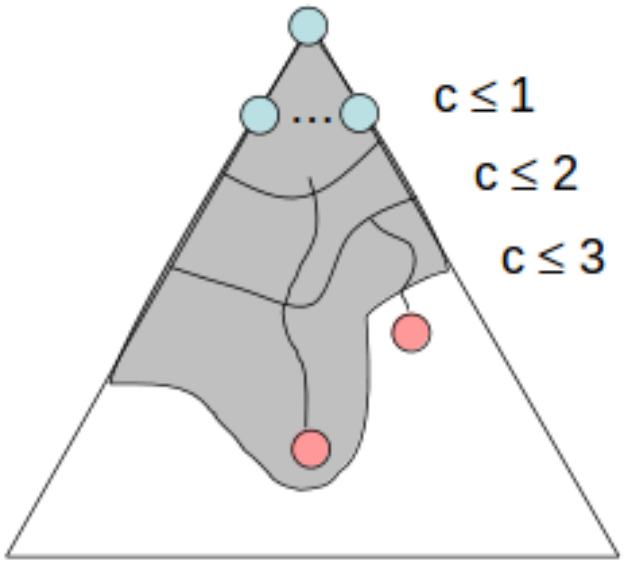


Our search algorithms are the same except for fringe strategies

- All fringes are priority queues: states with priorities.
- DFS, BFS a bit faster using simple stack/queues.
- Can code one implementation with variable queuing object.

Uninformed Search

Uniform Cost Search

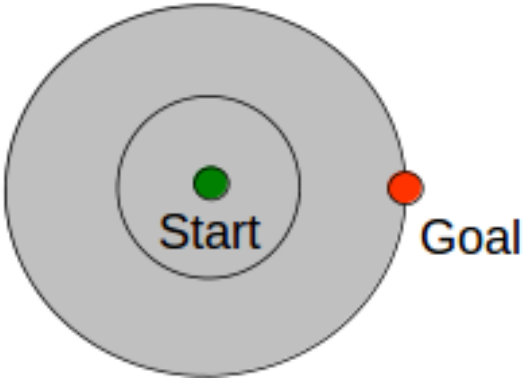


Strategy: expand lowest path cost.

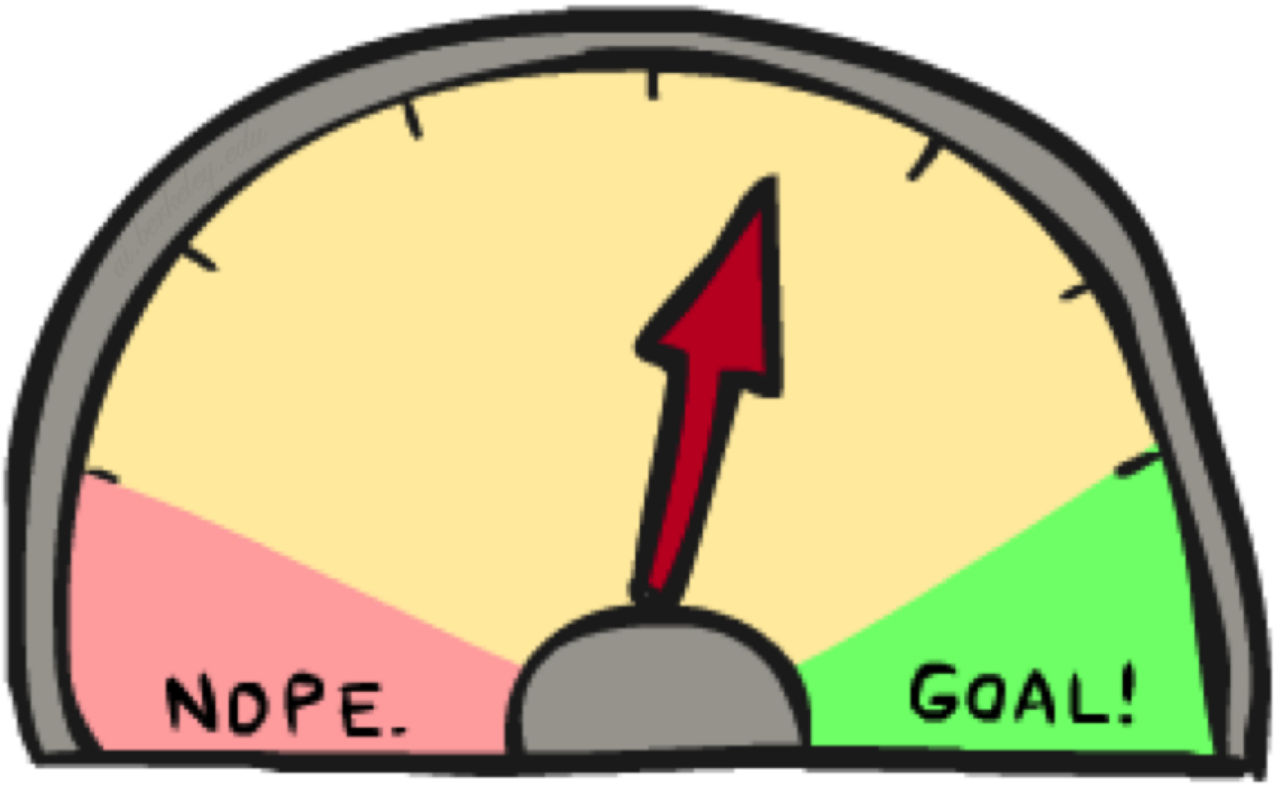
The good: Complete and optimal!

The bad: Explores options in every "direction"

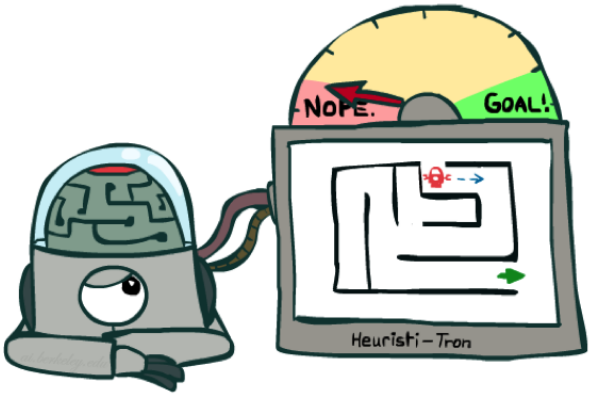
No information about goal location

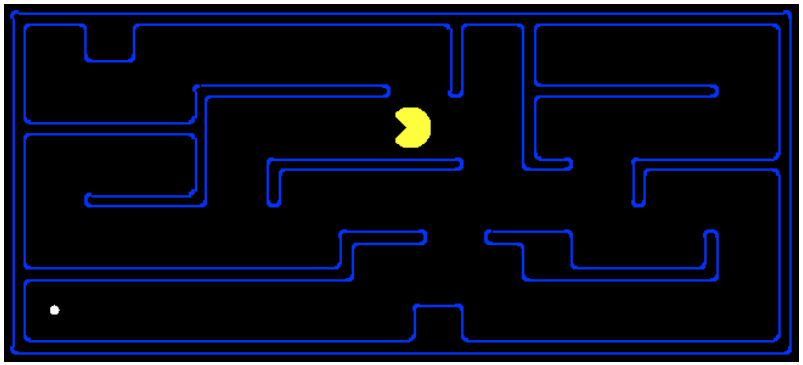


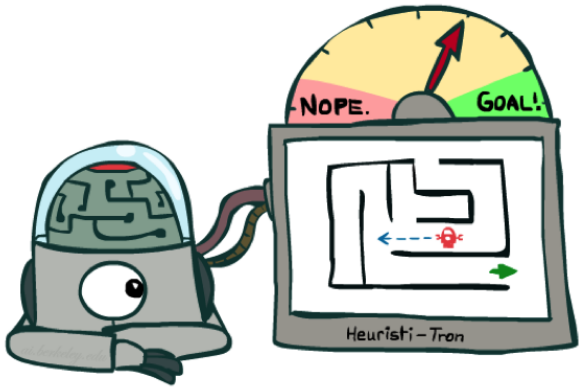
Informed Search



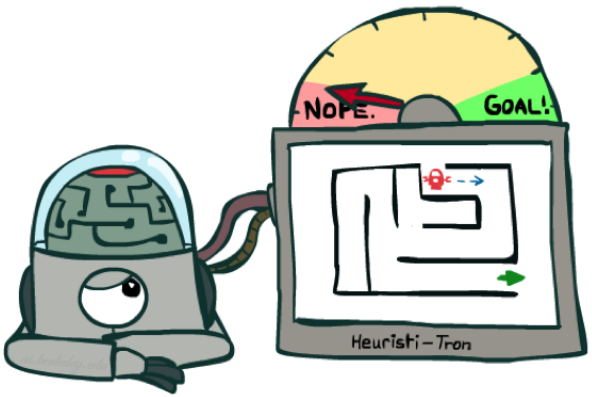
Search Heuristics

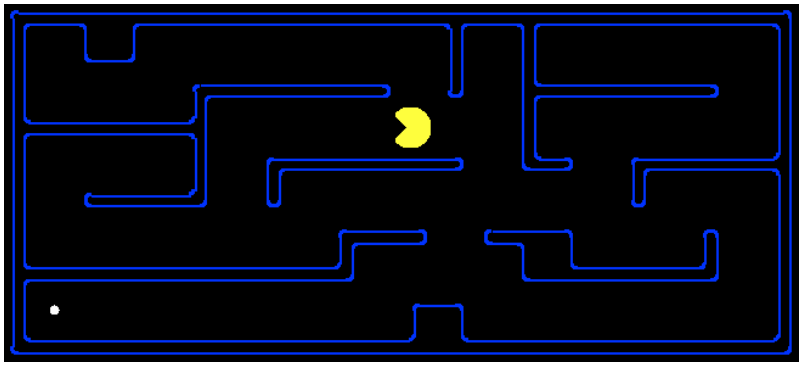






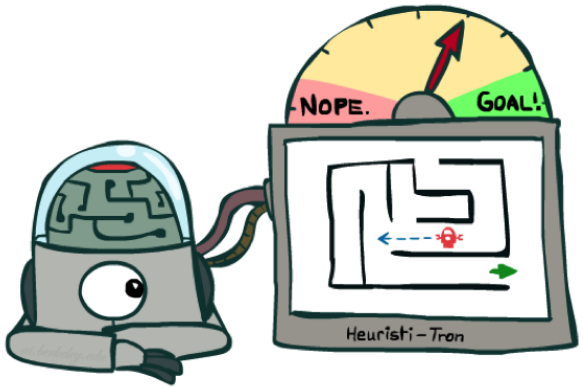
Search Heuristics



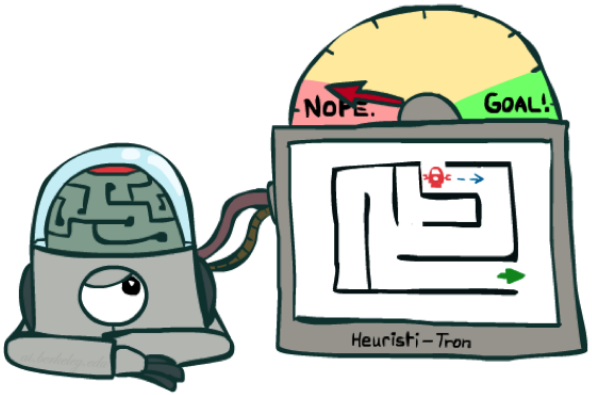


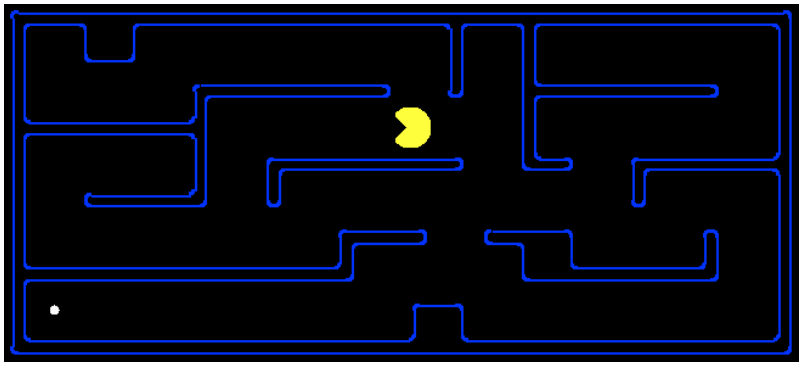
A heuristic is:

- A function that *estimates* how close a state is to a goal
- Designed for a particular search problem



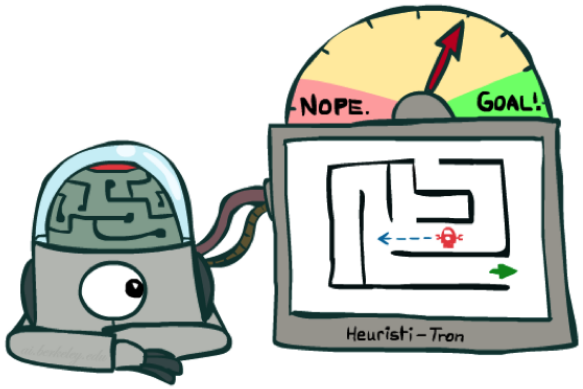
Search Heuristics



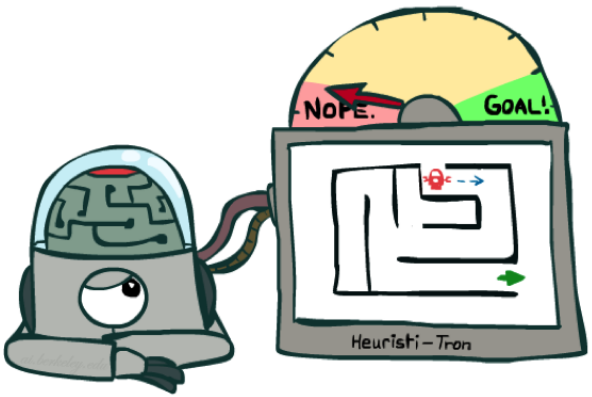


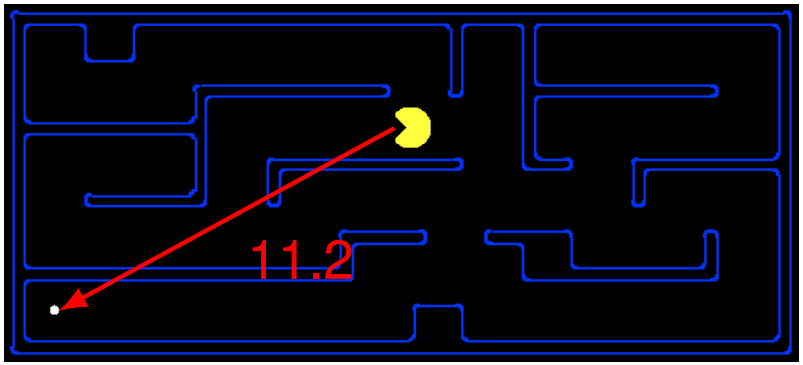
A heuristic is:

- A function that *estimates* how close a state is to a goal
- Designed for a particular search problem
- Examples:



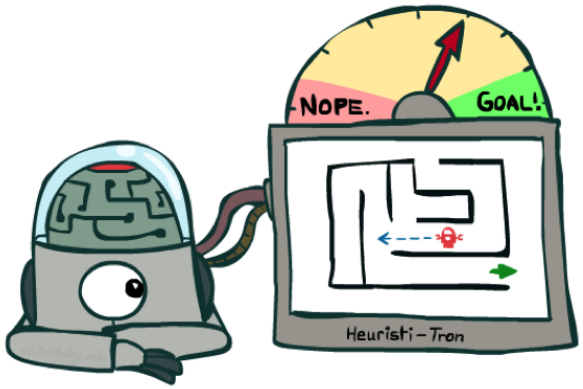
Search Heuristics



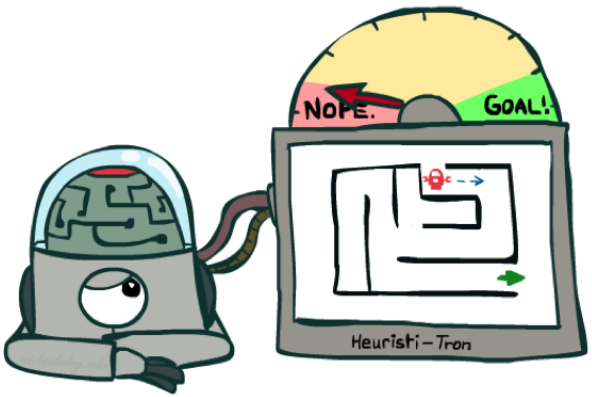


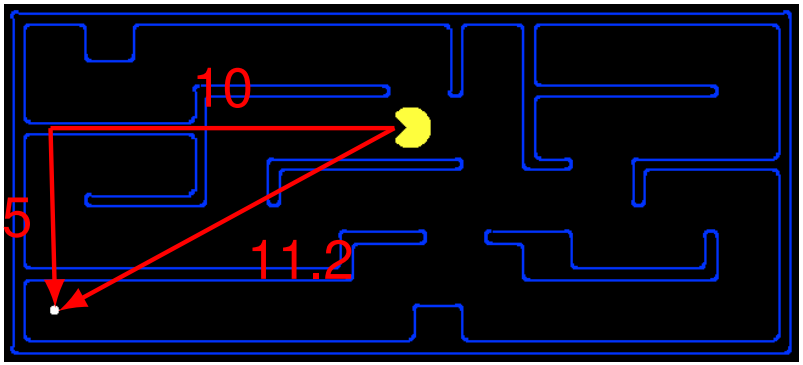
A heuristic is:

- A function that *estimates* how close a state is to a goal
- Designed for a particular search problem
- Examples: Euclidean distance for pathing.



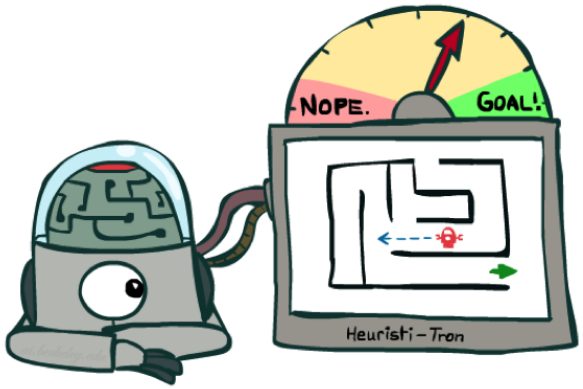
Search Heuristics



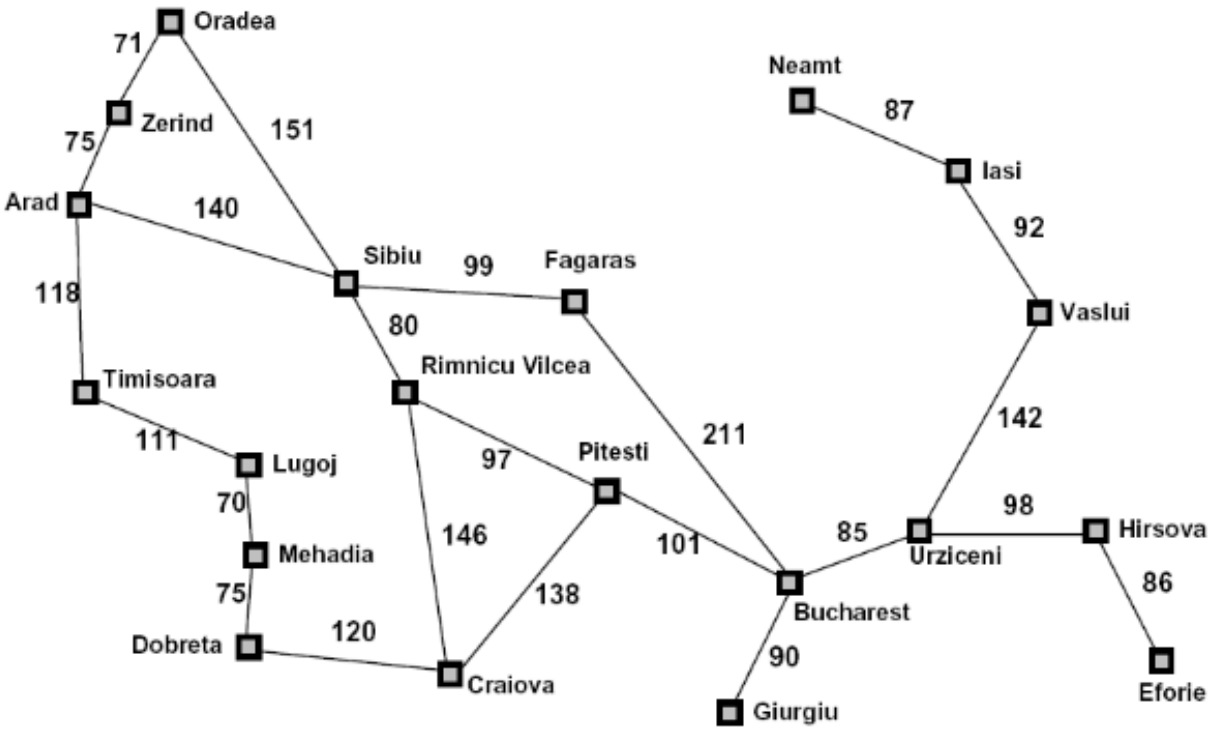


A heuristic is:

- A function that *estimates* how close a state is to a goal
- Designed for a particular search problem
- Examples: Euclidean distance for pathing. Manhattan distance.

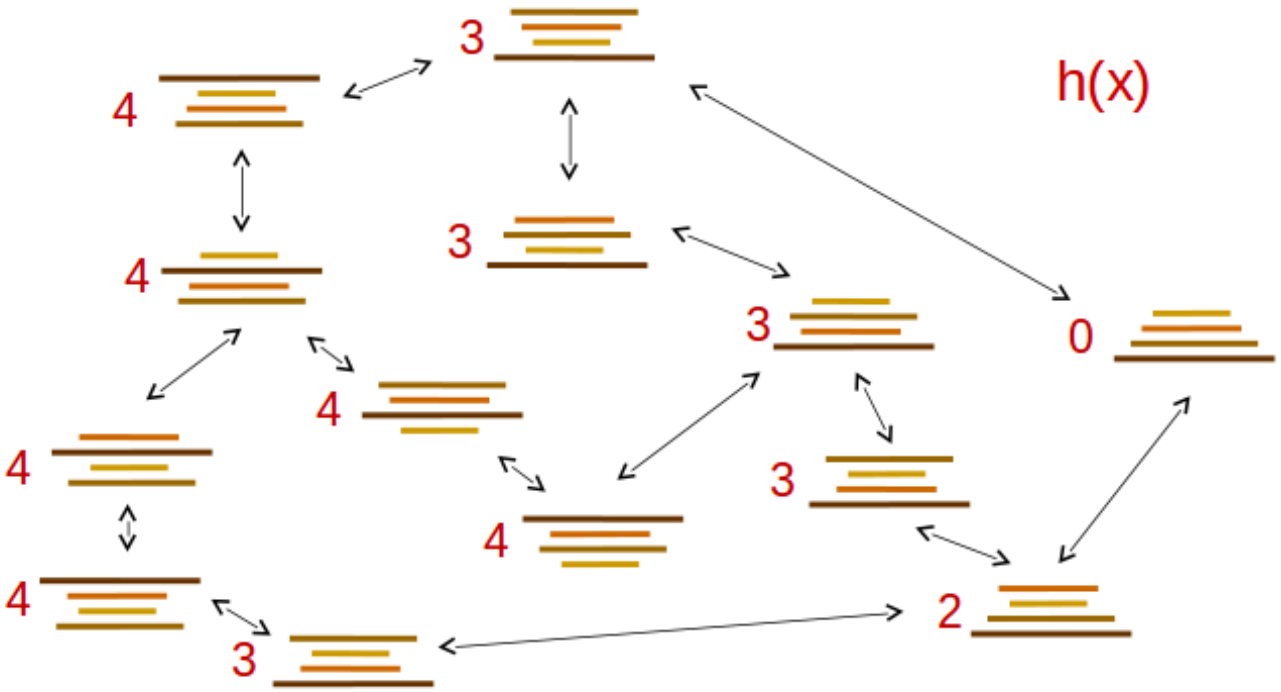


Example: Heuristic Function



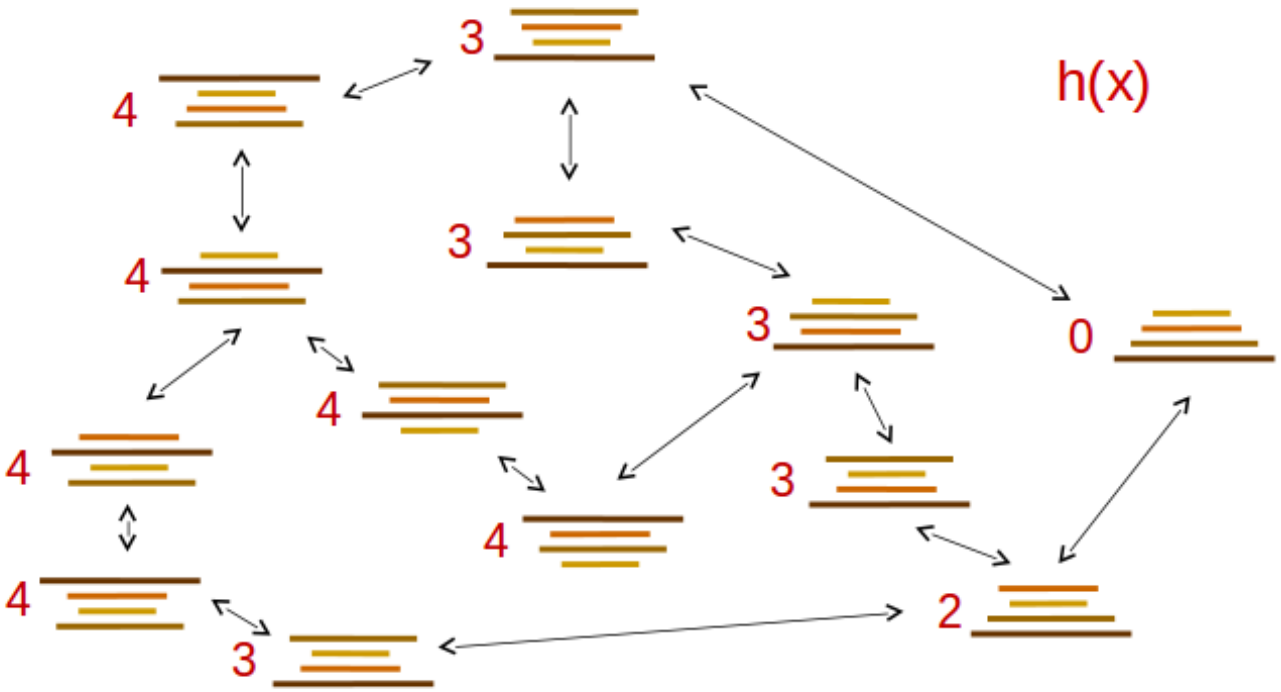
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example: Heuristic Function



Heuristic: he number of the largest pancake that is still out of place

Example: Heuristic Function



Heuristic: the number of the largest pancake that is still out of place