CPSC 322 - Announcements

- One of the "informative posts" is on Canvas. Please read and respond.
- Assignment 1 due next Tuesday

Review: Searching

A search problem (representing a state-space problem) consists of:

- A set of nodes (that represent states)
- A set of arcs, where an arc is an ordered pair of nodes (that represent actions)
- A start node (that represents the agent's initial state)
- A goal predicate that is true of a node if it is a goal node

Graph Search Algorithm

Input: a graph
a start node s
Boolean procedure $goal(n)$ that tests if n is a goal node

$frontier := \{\langle s \rangle\}$
while *frontier* is not empty:
select and **remove** path $\langle n_0, \ldots, n_k \rangle$ from *frontier*
if $goal(n_k)$
return $\langle n_0, \ldots, n_k \rangle$
$Frontier := Frontier \cup \{\langle n_0, \ldots, n_k, n \rangle : \langle n_k, n \rangle \in A\}$
end while

Depth-first Search

- Depth-first search treats the frontier as a stack. (First-in last-out)

Depth-first Search

- Depth-first search treats the frontier as a stack. (First-in last-out)
- It always selects one of the last elements added to the frontier.

Breadth-first Search

- Breadth-first search treats the frontier as a queue (first-in, first-out).
- It always selects one of the earliest elements added to the frontier.

Lowest-cost-first Search

- At each stage, lowest-cost-first search selects a path on the frontier with lowest cost.
- The frontier is a priority queue ordered by path cost.

Clicker Question

Suppose the frontier contains the following paths. The paths are listed in order of being added to the frontier, (so that (i) was the first of these added (ii) was the second of these added, etc.)

i) $\langle a, b, c \rangle$ with cost 6

ii) $\langle a, e, f \rangle$ with cost 5

iii) $\langle a, e, w \rangle$ with cost 5

iv) $\langle a, d, x \rangle$ with cost 9

v) $\langle a, d, z \rangle$ with cost 7

Which path will be expanded next for breath-first search

A (i) because it was added first

B (v) because it was added last

C either (ii) or (iii) because they have least cost

D (iv) because it has the greatest cost

E we can't tell; any of them could be chosen

Clicker Question

Suppose the frontier contains the following paths. The paths are listed in order of being added to the frontier, (so that (i) was the first of these added (ii) was the second of these added, etc.)

i) $\langle a, b, c \rangle$ with cost 6

ii) $\langle a, e, f \rangle$ with cost 5

iii) $\langle a, e, w \rangle$ with cost 5

iv) $\langle a, d, x \rangle$ with cost 9

v) $\langle a, d, z \rangle$ with cost 7

Which path will be expanded next for least-cost-first search

- A (i) because it was added first
- B (v) because it was added last
- C either (ii) or (iii) because they have least cost
- D (iv) because it has the greatest cost
- E we can't tell; any of them could be chosen

Clicker Question

Suppose the frontier contains the following paths. The paths are listed in order of being added to the frontier, (so that (i) was the first of these added (ii) was the second of these added, etc.)

i) $\langle a, b, c \rangle$ with cost 6

ii) $\langle a, e, f \rangle$ with cost 5

iii) $\langle a, e, w \rangle$ with cost 5

iv) $\langle a, d, x \rangle$ with cost 9

v) $\langle a, d, z \rangle$ with cost 7

Which path will be expanded next for depth-first search

A (i) because it was added first

B (v) because it was added last

C either (ii) or (iii) because they have least cost

D (iv) because it has the greatest cost

E we can't tell; any of them could be chosen

Properties of Depth-first Search

- Does depth-first search guarantee to find a solution (path from a start node to a goal node) with fewest arcs?

Properties of Depth-first Search

- Does depth-first search guarantee to find a solution (path from a start node to a goal node) with fewest arcs?
- Does depth-first search guarantee to find a solution with least cost?

Properties of Depth-first Search

- Does depth-first search guarantee to find a solution (path from a start node to a goal node) with fewest arcs?
- Does depth-first search guarantee to find a solution with least cost?
- What happens on infinite graphs or on graphs with cycles if there is a solution?

Properties of Depth-first Search

- Does depth-first search guarantee to find a solution (path from a start node to a goal node) with fewest arcs?
- Does depth-first search guarantee to find a solution with least cost?
- What happens on infinite graphs or on graphs with cycles if there is a solution?
- How does the goal affect the search?

Properties of Breadth-first Search

- Does breadth-first search guarantee to find a solution with fewest arcs?

Properties of Breadth-first Search

- Does breadth-first search guarantee to find a solution with fewest arcs?
- Does breadth-first search guarantee to find a solution with least cost?

Properties of Breadth-first Search

- Does breadth-first search guarantee to find a solution with fewest arcs?
- Does breadth-first search guarantee to find a solution with least cost?
- What happens on infinite graphs or on graphs with cycles if there is a solution?

Properties of Breadth-first Search

- Does breadth-first search guarantee to find a solution with fewest arcs?
- Does breadth-first search guarantee to find a solution with least cost?
- What happens on infinite graphs or on graphs with cycles if there is a solution?
- How does the goal affect the search?

Properties of Lowest-Cost-first Search

- Does lowest-cost-first search guarantee to find a solution with fewest arcs?

Properties of Lowest-Cost-first Search

- Does lowest-cost-first search guarantee to find a solution with fewest arcs?
- Does lowest-cost-first search guarantee to find a solution with least cost?

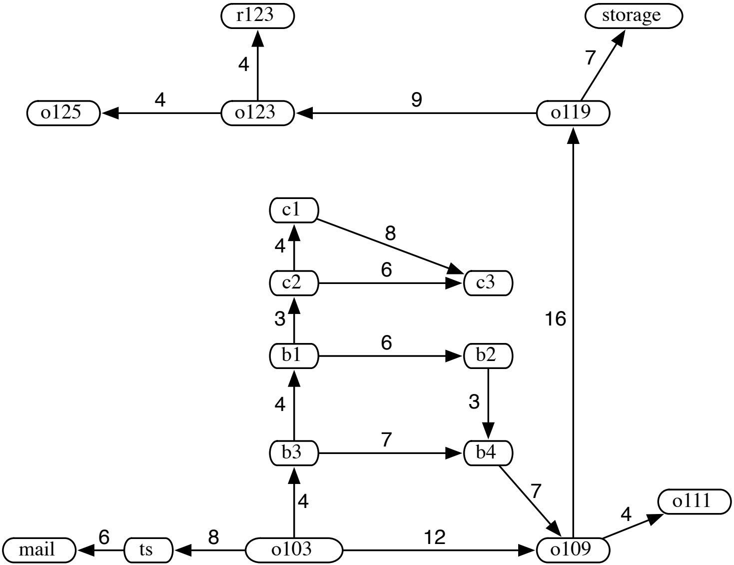
Properties of Lowest-Cost-first Search

- Does lowest-cost-first search guarantee to find a solution with fewest arcs?
- Does lowest-cost-first search guarantee to find a solution with least cost?
- What happens on infinite graphs or on graphs with cycles if there is a solution?

Properties of Lowest-Cost-first Search

- Does lowest-cost-first search guarantee to find a solution with fewest arcs?
- Does lowest-cost-first search guarantee to find a solution with least cost?
- What happens on infinite graphs or on graphs with cycles if there is a solution?
- How does the goal affect the search?

State-Space Graph for the Delivery Robot



Summary of Search Strategies

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added			
Breadth-first	First node added			
Lowest-cost-first	Minimal $cost(p)$			

Complete — guaranteed to find a solution if there is one (for graphs with finite number of neighbours, even on infinite graphs)
Halts — on finite graph (perhaps with cycles).
Space — as a function of the length of current path

Summary of Search Strategies

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No		
Breadth-first	First node added			
Lowest-cost-first	Minimal $cost(p)$			

Complete — guaranteed to find a solution if there is one (for graphs with finite number of neighbours, even on infinite graphs)
Halts — on finite graph (perhaps with cycles).
Space — as a function of the length of current path

Summary of Search Strategies

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No		
Breadth-first	First node added	Yes		
Lowest-cost-first	Minimal $cost(p)$			

Complete — guaranteed to find a solution if there is one (for graphs with finite number of neighbours, even on infinite graphs)

Halts — on finite graph (perhaps with cycles).

Space — as a function of the length of current path

Summary of Search Strategies

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No		
Breadth-first	First node added	Yes		
Lowest-cost-first	Minimal $cost(p)$	Yes		

Complete — guaranteed to find a solution if there is one (for graphs with finite number of neighbours, even on infinite graphs)

Halts — on finite graph (perhaps with cycles).

Space — as a function of the length of current path

Summary of Search Strategies

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No	No	
Breadth-first	First node added	Yes		
Lowest-cost-first	Minimal $cost(p)$	Yes		

Complete — guaranteed to find a solution if there is one (for graphs with finite number of neighbours, even on infinite graphs)

Halts — on finite graph (perhaps with cycles).

Space — as a function of the length of current path

Summary of Search Strategies

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No	No	
Breadth-first	First node added	Yes	No	
Lowest-cost-first	Minimal $cost(p)$	Yes		

Complete — guaranteed to find a solution if there is one (for graphs with finite number of neighbours, even on infinite graphs)
Halts — on finite graph (perhaps with cycles).
Space — as a function of the length of current path

Summary of Search Strategies

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No	No	
Breadth-first	First node added	Yes	No	
Lowest-cost-first	Minimal $cost(p)$	Yes	No	

Complete — guaranteed to find a solution if there is one (for graphs with finite number of neighbours, even on infinite graphs)
Halts — on finite graph (perhaps with cycles).
Space — as a function of the length of current path

Summary of Search Strategies

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No	No	Linear
Breadth-first	First node added	Yes	No	
Lowest-cost-first	Minimal $cost(p)$	Yes	No	

Complete — guaranteed to find a solution if there is one (for graphs with finite number of neighbours, even on infinite graphs)
Halts — on finite graph (perhaps with cycles).
Space — as a function of the length of current path

Summary of Search Strategies

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No	No	Linear
Breadth-first	First node added	Yes	No	Exp
Lowest-cost-first	Minimal $cost(p)$	Yes	No	

Complete — guaranteed to find a solution if there is one (for graphs with finite number of neighbours, even on infinite graphs)

Halts — on finite graph (perhaps with cycles).

Space — as a function of the length of current path

Summary of Search Strategies

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No	No	Linear
Breadth-first	First node added	Yes	No	Exp
Lowest-cost-first	Minimal $cost(p)$	Yes	No	Exp

Complete — guaranteed to find a solution if there is one (for graphs with finite number of neighbours, even on infinite graphs)

Halts — on finite graph (perhaps with cycles).

Space — as a function of the length of current path

Complexity

n is the path length to closest goal
m is the maximum path length in graph (could be infinite)
b is the branching factor — the max num of neighbors

Method	Time Complexity	Space Complexity
Depth-first		
Breadth-first		
Lowest-cost-first		

Complexity

n is the path length to closest goal
m is the maximum path length in graph (could be infinite)
b is the branching factor — the max num of neighbors

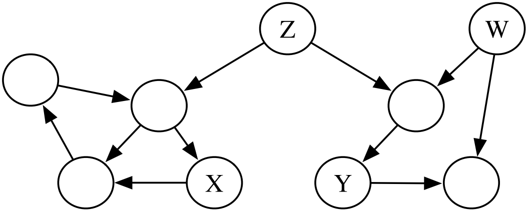
Method	Time Complexity	Space Complexity
Depth-first	$O(b^m)$	$O(bm)$
Breadth-first		
Lowest-cost-first		

Complexity

n is the path length to closest goal
m is the maximum path length in graph (could be infinite)
b is the branching factor — the max num of neighbors

Method	Time Complexity	Space Complexity
Depth-first	$O(b^m)$	$O(bm)$
Breadth-first	$O(b^n)$	$O(b^n)$
Lowest-cost-first	$O(b^n)$	$O(b^n)$

Clicker Question



The start node is *Z*. The goal is *Y*. Children are expanded left to right. Cost is distance. Which of the following is true:

A breadth-first, depth-first and least-cost-first (LCF) all halt

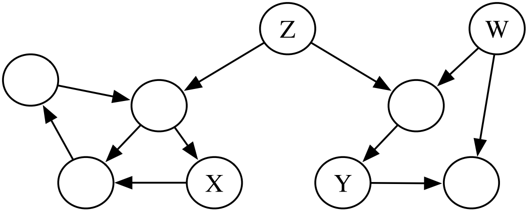
B none of breadth-first, depth-first and LCF halt

C depth-first and LCF halt, but breadth-first search doesn't halt

D breadth-first and LCF searches halt, but depth-first search doesn't halt

E none of the above

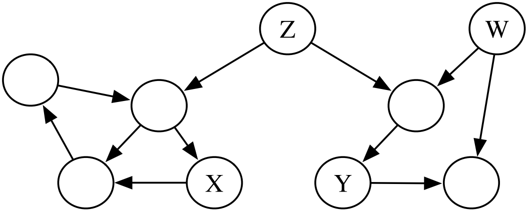
Clicker Question



The start node is Z. The goal is W. Children are expanded left to right. Cost is distance. Which of the following is true:

- A breadth-first, depth-first and least-cost-first (LCF) all halt
- B none of breadth-first, depth-first and LCF halt
- C depth-first and LCF halt, but breadth-first search doesn't halt
- D breadth-first and LCF searches halt, but depth-first search doesn't halt
- E none of the above

Clicker Question



The start node is W. The goal is X. Children are expanded from left to right. Cost is distance. Which of the following is true:

- A breadth-first, depth-first and least-cost-first (LCF) all halt
- B none of breadth-first, depth-first and LCF halt
- C depth-first and LCF halt, but breadth-first search doesn't halt
- D breadth-first and LCF searches halt, but depth-first search doesn't halt
- E none of the above

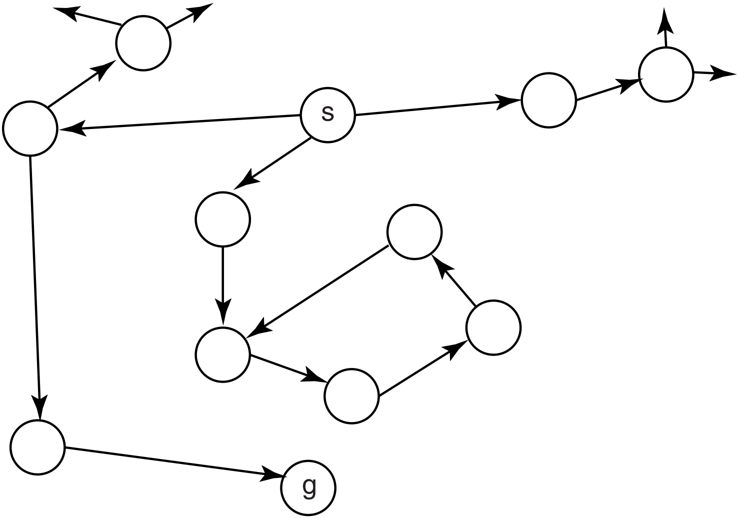
Heuristic Search

- $h(n)$ is a nonnegative estimate of the cost of the least-cost path from node n to a goal node.
- $h(n)$ is an underestimate if there is no path from n to a goal with cost less than $h(n)$.
- An admissible heuristic is a heuristic function that is an underestimate for every node.

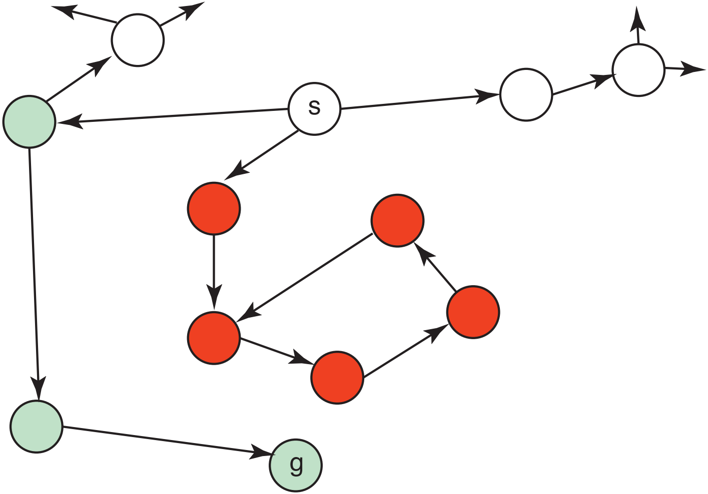
Best-first Search

- Idea: select the path whose end is closest to a goal according to the heuristic function.
- Best-first search selects a path on the frontier with minimal h-value.
- It treats the frontier as a priority queue ordered by h.

Illustrative Graph



Illustrative Graph



Complexity of Best-first Search

- Does best-first search guarantee to find the path with fewest arcs?

Complexity of Best-first Search

- Does best-first search guarantee to find the path with fewest arcs?
- What happens on infinite graphs or on graphs with cycles if there is a solution?

Complexity of Best-first Search

- Does best-first search guarantee to find the path with fewest arcs?
- What happens on infinite graphs or on graphs with cycles if there is a solution?
- What is the time complexity as a function of length of the path selected?

Complexity of Best-first Search

- Does best-first search guarantee to find the path with fewest arcs?
- What happens on infinite graphs or on graphs with cycles if there is a solution?
- What is the time complexity as a function of length of the path selected?
- What is the space complexity as a function of length of the path selected?

Complexity of Best-first Search

- Does best-first search guarantee to find the path with fewest arcs?
- What happens on infinite graphs or on graphs with cycles if there is a solution?
- What is the time complexity as a function of length of the path selected?
- What is the space complexity as a function of length of the path selected?
- How does the goal affect the search?

A^* Search

- A^* search uses both path cost and heuristic function.

A^* Search

- A^* search uses both path cost and heuristic function.
- $cost(p)$ is the cost of path p.

A^* Search

- A^* search uses both path cost and heuristic function.
- $cost(p)$ is the cost of path p.
- $h(p)$ estimates the cost from the end of p to a goal.

A^* Search

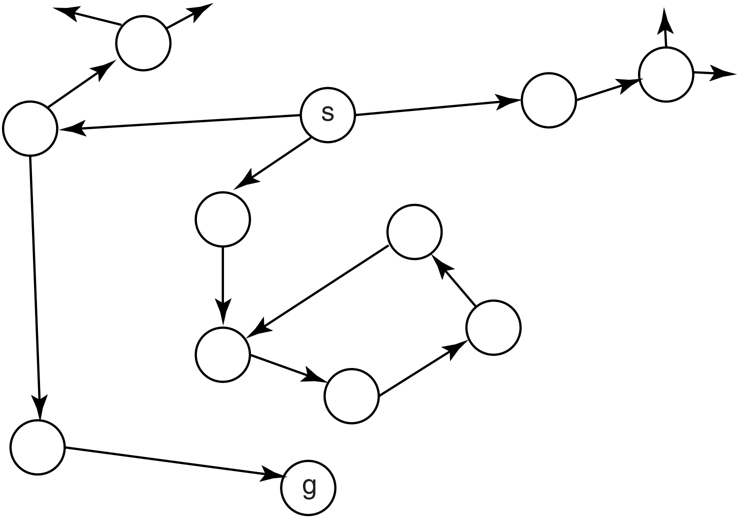
- A^* search uses both path cost and heuristic function.
- $cost(p)$ is the cost of path p.
- $h(p)$ estimates the cost from the end of p to a goal.
- Let $f(p) = cost(p) + h(p)$.

A^* Search

- A^* search uses both path cost and heuristic function.
- $cost(p)$ is the cost of path p.
- $h(p)$ estimates the cost from the end of p to a goal.
- Let $f(p) = cost(p) + h(p)$.
  $f(p)$ estimates the total path cost of going from a start node to a goal via p.

$$\underbrace{\underbrace{start \xrightarrow{\text{path } p} n}_{cost(p)} \underbrace{\xrightarrow{\text{estimate}} goal}_{h(p)}}_{f(p)}$$

Illustrative Graph



Clicker Question

Given a path, p, that is longer than one arc, which of the following is the correct way of calculating $f(p)$ for the A^* search algorithm:

A f(p) = cost(p)

B f(p) = cost(p) + heuristic values of all nodes in the path

C f(p) = cost(p) + heuristic value of first node in the path

D f(p) = cost(p) + heuristic value of last node in the path

Complexity of A^* Search

- Does A^* search guarantee to find the path with fewest arcs?

Complexity of A^* Search

- Does A^* search guarantee to find the path with fewest arcs?
- Does A^* search guarantee to find the least-cost path?

Complexity of A^* Search

- Does A^* search guarantee to find the path with fewest arcs?
- Does A^* search guarantee to find the least-cost path?
- What happens on infinite graphs or on graphs with cycles if there is a solution?

Complexity of A^* Search

- Does A^* search guarantee to find the path with fewest arcs?
- Does A^* search guarantee to find the least-cost path?
- What happens on infinite graphs or on graphs with cycles if there is a solution?
- What is the time complexity as a function of length of the path selected?

Complexity of A^* Search

- Does A^* search guarantee to find the path with fewest arcs?
- Does A^* search guarantee to find the least-cost path?
- What happens on infinite graphs or on graphs with cycles if there is a solution?
- What is the time complexity as a function of length of the path selected?
- What is the space complexity as a function of length of the path selected?

Complexity of A^* Search

- Does A^* search guarantee to find the path with fewest arcs?
- Does A^* search guarantee to find the least-cost path?
- What happens on infinite graphs or on graphs with cycles if there is a solution?
- What is the time complexity as a function of length of the path selected?
- What is the space complexity as a function of length of the path selected?
- How does the goal affect the search?

Admissibility of A^*

If there is a solution, A^* always finds an optimal solution —the first path to a goal selected— if

- the branching factor is finite
- arc costs are bounded above zero (there is some $\epsilon > 0$ such that all of the arc costs are greater than ϵ), and
- $h(n)$ is nonnegative and an underestimate of the cost of the least-cost path from n to a goal node.

Why is A^* admissible?

- If a path p to a goal is selected from a frontier, can there be a lower cost path to a goal?
- Suppose path p' is on the frontier. Because p was chosen before p', and $h(p) = 0$:

Why is A^* admissible?

- If a path p to a goal is selected from a frontier, can there be a lower cost path to a goal?
- Suppose path p' is on the frontier. Because p was chosen before p', and $h(p) = 0$:

$$cost(p) \leq cost(p') + h(p').$$

- Because h is an underestimate:

for any path p'' to a goal that extends p'.

Why is A^* admissible?

- If a path p to a goal is selected from a frontier, can there be a lower cost path to a goal?
- Suppose path p' is on the frontier. Because p was chosen before p', and $h(p) = 0$:

$$cost(p) \le cost(p') + h(p').$$

- Because h is an underestimate:

$$cost(p') + h(p') \le cost(p'')$$

  for any path p'' to a goal that extends p'.
- So $cost(p) \le cost(p'')$ for any other path p'' to a goal.

Why is A^* admissible?

A^* can always find a solution if there is one:

- The frontier always contains the initial part of a path to a goal, before that goal is selected.
- A^* halts, as the costs of the paths on the frontier keeps increasing, and will eventually exceed any finite number.

How do good heuristics help?

Suppose c is the cost of an optimal solution. What happen to a path p where

- $cost(p) + h(p) < c$

How do good heuristics help?

Suppose c is the cost of an optimal solution. What happen to a path p where

- $cost(p) + h(p) < c$
- $cost(p) + h(p) = c$

How do good heuristics help?

Suppose c is the cost of an optimal solution. What happen to a path p where

- $cost(p) + h(p) < c$
- $cost(p) + h(p) = c$
- $cost(p) + h(p) > c$

How can a better heuristic function help?

Summary of Search Strategies

Assume the graph follows the assumptions the the admissibility theorem.

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added			
Breadth-first	First node added			
Lowest-cost-first	Minimal $cost(p)$			
Best-first	Minimal $h(p)$			
A^*	Minimal $f(p)$			

Complete — if there a path to a goal, it can find one, even on infinite graphs.

Halts — on finite graph (perhaps with cycles).

Space — as a function of the length of current path

Summary of Search Strategies

Assume the graph follows the assumptions the the admissibility theorem.

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No	No	Linear
Breadth-first	First node added	Yes	No	Exp
Lowest-cost-first	Minimal $cost(p)$	Yes	No	Exp
Best-first	Minimal $h(p)$			
A^*	Minimal $f(p)$			

Complete — if there a path to a goal, it can find one, even on infinite graphs.

Halts — on finite graph (perhaps with cycles).

Space — as a function of the length of current path

Summary of Search Strategies

Assume the graph follows the assumptions the the admissibility theorem.

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No	No	Linear
Breadth-first	First node added	Yes	No	Exp
Lowest-cost-first	Minimal $cost(p)$	Yes	No	Exp
Best-first	Minimal $h(p)$	No		
A^*	Minimal $f(p)$			

Complete — if there a path to a goal, it can find one, even on infinite graphs.

Halts — on finite graph (perhaps with cycles).

Space — as a function of the length of current path

Summary of Search Strategies

Assume the graph follows the assumptions the the admissibility theorem.

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No	No	Linear
Breadth-first	First node added	Yes	No	Exp
Lowest-cost-first	Minimal $cost(p)$	Yes	No	Exp
Best-first	Minimal $h(p)$	No		
A^*	Minimal $f(p)$	Yes		

Complete — if there a path to a goal, it can find one, even on infinite graphs.

Halts — on finite graph (perhaps with cycles).

Space — as a function of the length of current path

Summary of Search Strategies

Assume the graph follows the assumptions the the admissibility theorem.

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No	No	Linear
Breadth-first	First node added	Yes	No	Exp
Lowest-cost-first	Minimal $cost(p)$	Yes	No	Exp
Best-first	Minimal $h(p)$	No	No	
A^*	Minimal $f(p)$	Yes		

Complete — if there a path to a goal, it can find one, even on infinite graphs.

Halts — on finite graph (perhaps with cycles).

Space — as a function of the length of current path

Summary of Search Strategies

Assume the graph follows the assumptions the the admissibility theorem.

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No	No	Linear
Breadth-first	First node added	Yes	No	Exp
Lowest-cost-first	Minimal $cost(p)$	Yes	No	Exp
Best-first	Minimal $h(p)$	No	No	
A^*	Minimal $f(p)$	Yes	No	

Complete — if there a path to a goal, it can find one, even on infinite graphs.

Halts — on finite graph (perhaps with cycles).

Space — as a function of the length of current path

Summary of Search Strategies

Assume the graph follows the assumptions the the admissibility theorem.

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No	No	Linear
Breadth-first	First node added	Yes	No	Exp
Lowest-cost-first	Minimal $cost(p)$	Yes	No	Exp
Best-first	Minimal $h(p)$	No	No	Exp
A^*	Minimal $f(p)$	Yes	No	

Complete — if there a path to a goal, it can find one, even on infinite graphs.

Halts — on finite graph (perhaps with cycles).

Space — as a function of the length of current path

Summary of Search Strategies

Assume the graph follows the assumptions the the admissibility theorem.

Strategy	Frontier Selection	Complete	Halts	Space
Depth-first	Last node added	No	No	Linear
Breadth-first	First node added	Yes	No	Exp
Lowest-cost-first	Minimal $cost(p)$	Yes	No	Exp
Best-first	Minimal $h(p)$	No	No	Exp
A^*	Minimal $f(p)$	Yes	No	Exp

Complete — if there a path to a goal, it can find one, even on infinite graphs.
Halts — on finite graph (perhaps with cycles).
Space — as a function of the length of current path