Informed Search

CS5804 Introduction to Artificial Intelligence
Virginia Tech
function TREE-SEARCH(problem) returns a solution, or failure

initialize the frontier using the initial state of problem

loop do:

    if the frontier is empty then return failure

    choose a leaf node and remove it from the frontier

    if the node contains a goal state then return the solution

    expand the chosen node, adding the resulting nodes to frontier
**function** TREE-SEARCH(*problem*) **returns** a solution, or failure

initialize the frontier using the initial state of *problem*  

**loop** do:

**if** the frontier is empty **then** return failure

choose a leaf node and remove it from the frontier

**if** the node contains a goal state **then** return the solution

expand the chosen node, adding the resulting nodes to frontier
function GRAPH-SEARCH(problem) returns a solution, or failure

initialize the frontier using the initial state of problem

initialize the explored set to be empty

loop do:

  if the frontier is empty then return failure

choose a leaf node and remove it from the frontier

  if the node contains a goal state then return the solution

add the node to the explored set

expand the chosen node, adding the resulting nodes to frontier only if not in the frontier or explored set
function GRAPH-SEARCH(problem) returns a solution, or failure

initialize the frontier using the initial state of problem

initialize the explored set to be empty

loop do:
  if the frontier is empty then return failure

  choose a leaf node and remove it from the frontier

  if the node contains a goal state then return the solution

  add the node to the explored set

  expand the chosen node, adding the resulting nodes to frontier
    only if not in the frontier or explored set
function GRAPH-SEARCH(problem) returns a solution, or failure

initialize the frontier using the initial state of problem

initialize the explored set to be empty

loop do:
  if the frontier is empty then return failure
  choose a leaf node and remove it from the frontier
  if the node contains a goal state then return the solution
  add the node to the explored set
  expand the chosen node, adding the resulting nodes to frontier
  only if not in the frontier or explored set
function GRAPH-SEARCH(problem) returns a solution, or failure

initialize the frontier using the initial state of problem

initialize the explored set to be empty

loop do:

if the frontier is empty then return failure

choose a leaf node and remove it from the frontier

if the node contains a goal state then return the solution

add the node to the explored set

expand the chosen node, adding the resulting nodes to frontier

only if not in the frontier or explored set
function GRAPH-SEARCH(problem) returns a solution, or failure

initialize the frontier using the initial state of problem

initialize the explored set to be empty

loop do:
  if the frontier is empty then return failure
  choose a leaf node and remove it from the frontier
  if the node contains a goal state then return the solution
  add the node to the explored set
  expand the chosen node, adding the resulting nodes to frontier
    only if not in the frontier or explored set
function GRAPH-SEARCH(problem) returns a solution, or failure

initialize the frontier using the initial state of problem
initialize the explored set to be empty

loop do:
  if the frontier is empty then return failure
  choose a leaf node and remove it from the frontier
  if the node contains a goal state then return the solution
  add the node to the explored set
  expand the chosen node, adding the resulting nodes to frontier only if not in the frontier or explored set
function GRAPH-SEARCH(problem) returns a solution, or failure

initialize the frontier using the initial state of problem

initialize the explored set to be empty

loop do:

if the frontier is empty then return failure

choose a leaf node and remove it from the frontier

if the node contains a goal state then return the solution

add the node to the explored set

expand the chosen node, adding the resulting nodes to frontier only if not in the frontier or explored set
function GRAPH-SEARCH(problem) returns a solution, or failure

initialize the frontier using the initial state of problem

initialize the explored set to be empty

loop do:

if the frontier is empty then return failure

choose a leaf node and remove it from the frontier

if the node contains a goal state then return the solution

add the node to the explored set

expand the chosen node, adding the resulting nodes to frontier only if not in the frontier or explored set
function GRAPH-SEARCH(problem) returns a solution, or failure

initialize the frontier using the initial state of problem

initialize the explored set to be empty

loop do:

if the frontier is empty then return failure

choose a leaf node and remove it from the frontier

if the node contains a goal state then return the solution

add the node to the explored set

expand the chosen node, adding the resulting nodes to frontier

only if not in the frontier or explored set
function GRAPH-SEARCH(problem) returns a solution, or failure
initialize the frontier using the initial state of problem
initialize the explored set to be empty

loop do:
  if the frontier is empty then return failure
  choose a leaf node and remove it from the frontier
  if the node contains a goal state then return the solution
  add the node to the explored set
  expand the chosen node, adding the resulting nodes to frontier only if not in the frontier or explored set
function GRAPH-SEARCH(problem) returns a solution, or failure

initialize the frontier using the initial state of problem

initialize the explored set to be empty

loop do:

if the frontier is empty then return failure

choose a leaf node and remove it from the frontier

if the node contains a goal state then return the solution

add the node to the explored set

expand the chosen node, adding the resulting nodes to frontier

only if not in the frontier or explored set
function GRAPH-SEARCH(problem) returns a solution, or failure

initialize the frontier using the initial state of problem

initialize the explored set to be empty

loop do:
  if the frontier is empty then return failure

  choose a leaf node and remove it from the frontier

  if the node contains a goal state then return the solution

  add the node to the explored set

  expand the chosen node, adding the resulting nodes to frontier only if not in the frontier or explored set
function GRAPH-SEARCH(problem) returns a solution, or failure

initialize the frontier using the initial state of problem

initialize the explored set to be empty

loop do:
  if the frontier is empty then return failure
  choose a leaf node and remove it from the frontier
  if the node contains a goal state then return the solution
  add the node to the explored set
  expand the chosen node, adding the resulting nodes to frontier only if not in the frontier or explored set
Search Algorithm Properties

- Completeness
- Optimality
- Time complexity
- Space complexity
Search Algorithm Properties

- Completeness: BFS, uniform cost, finite DFS, iterative deepening…
- Optimality: uniform cost
- Time complexity: BFS $O(b^d)$, DFS terrible, iterative deep. $O(b^d)$
- Space complexity: BFS $O(b^d)$, DFS $O(bm)$, iterative deep. $O(bm)$
Search Algorithm Properties

• Completeness: BFS, uniform cost, finite DFS, iterative deepening…

• Optimality: uniform cost

• Time complexity: BFS $O(b^d)$, DFS terrible, iterative deep. $O(b^d)$

• Space complexity: BFS $O(b^d)$, DFS $O(bm)$, iterative deep. $O(bm)$

tree vs graph search?
Search Algorithms

- Breadth-first family
  - Breadth-first search
  - Uniform-cost search

- Depth-first family
  - Depth-first search
  - Depth-limited search
  - Iterative-deepening search
Example:

$R_1 L_1 S_1 R_2 L_2 S_2 R_3 L_3 S_3 R_4 L_4 S_4$
Example:

- Vacuum
- World
- State
- Space
- Graph

Chapter 3
Example:

- vacuum
- world state
- space

Chapter 3
wheels
cost so far: 1
cost so far: 1
cost so far: 1
cost so far: 1 + 2
cost so far: 1 + 2
cost so far: $1 + 2$

1 + 1 + 2 + 5

1 + 2 + 1 + 3

1 + 2 + 1 + 2
cost so far: $1 + 2 + 1$
cost so far: 1 + 2 + 1
cost so far: 1 + 2 + 1
cost so far: $1 + 2 + 1 + 2$
A* Search

• Expand node in frontier with best evaluation function score $f(n)$
  
  • $f(n) = g(n) + h(n)$
  
  • $g(n) :=$ cost to get from initial state to $n$
  
  • $h(n) :=$ heuristic estimate of cost to get from $n$ to goal

• Completeness, optimality, and time & space depend on $h$