CS 4100/5100: Foundations of AI
Search

Instructor: Rob Platt
rplatt@ccs.neu.edu

College of Computer and information Science
Northeastern University

Fall, 2014
Examples of search problems

Suppose the problem is to find a path to the x.

▶ state space \((n = 56)\)?
▶ action space?
▶ transition function?
▶ goal test / path cost?
Examples of search problems

Suppose the problem is to reach the x while avoiding ghosts.

- state space ($56^3 = 175k$)?
- action space?
- transition function?
- goal test / path cost?
Examples of search problems

Suppose the problem is to eat all the dots and avoid the ghosts?

▶ state space $56^3 + 2^5\cdot6 > 7.2^{16}$?
▶ action space?
▶ transition function?
▶ goal test / path cost?
Examples of search problems

- state space (no greater than 9! = 362k states)?
- action space?
- transition function?
- goal test / path cost?
Examples of search problems
function TREE-SEARCH\((\text{problem})\) returns a solution, or failure
initialize the frontier using the initial state of \text{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 corresponding solution
  expand the chosen node, adding the resulting nodes to the frontier

function GRAPH-SEARCH\((\text{problem})\) returns a solution, or failure
initialize the frontier using the initial state of \text{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 corresponding solution
  add the node to the explored set
  expand the chosen node, adding the resulting nodes to the frontier
  only if not in the frontier or explored set
Breadth First Search (BFS)

```
function BREADTH-FIRST-SEARCH(problem) returns a solution, or failure
    node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
    if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
    frontier ← a FIFO queue with node as the only element
    explored ← an empty set
    loop do
        if EMPTY?(frontier) then return failure
        node ← POP(frontier) /* chooses the shallowest node in frontier */
        add node.STATE to explored
        for each action in problem.ACTIONS(node.STATE) do
            child ← CHILD-NODE(problem, node, action)
            if child.STATE is not in explored or frontier then
                if problem.GOAL-TEST(child.STATE) then return SOLUTION(child)
                frontier ← INSERT(child, frontier)
```
Uniform Cost Search (UCS)

**function** UNIFORM-COST-SEARCH(`problem`) **returns** a solution, or failure

- `node` ← a node with `STATE = problem.INITIAL-STATE`, `PATH-COST = 0`
- `frontier` ← a priority queue ordered by `PATH-COST`, with `node` as the only element
- `explored` ← an empty set

**loop**

- **if** EMPTY?(`frontier`) **then** return failure
- `node` ← POP(`frontier`) /* chooses the lowest-cost node in `frontier` */
- **if** `problem.GOAL-TEST(node.STATE)` **then** return SOLUTION(`node`)
- add `node.STATE` to `explored`

**for each** `action` in `problem.ACTIONS(node.STATE)` **do**

- `child` ← CHILD-NODE(`problem`, `node`, `action`)
- **if** `child.STATE` is not in `explored` or `frontier` **then**
  - `frontier` ← INSERT(`child`, `frontier`)
- **else if** `child.STATE` is in `frontier` with higher `PATH-COST` **then**
  - replace that `frontier` node with `child`
Depth Limited Depth First Search (DLDFS)

function Depth-Limited-Search(problem, limit) returns a solution, or failure/cutoff
return Recursive-DLS(Make-Node(problem.INITIAL-STATE), problem, limit)

function Recursive-DLS(node, problem, limit) returns a solution, or failure/cutoff
if problem.GOAL-Test(node.STATE) then return SOLUTION(node)
else if limit = 0 then return cutoff
else
cutoff Occurred? ← false
for each action in problem.ACTIONS(node.STATE) do
    child ← Child-Node(problem, node, action)
    result ← Recursive-DLS(child, problem, limit - 1)
    if result = cutoff then cutoff Occurred? ← true
    else if result ≠ failure then return result
if cutoff Occurred? then return cutoff else return failure
Bidirectional search