Lecture 13: Uniform-Cost Search
Last Time: Graph Search

Find path between 2 points in an arbitrary graph.
Last Time: Graph Search

Find path between 2 points in an arbitrary graph.

Represent all possible paths from A with a tree:
Labs

Last Week: Robots in Mazes
This Week: Uniform Cost Search, MapQuest
Labs

Last Week: Robots in Mazes
This Week: Uniform Cost Search, MapQuest
Graph Search Algorithm

Basic Algorithm:
- Initialize **agenda** (list of nodes to consider)
- Repeat the following:
  - Remove one node from the agenda
  - Add its children to the agenda
until **goal is found** or **agenda is empty**
- Return resulting path
Order Matters!

Strategy: Replace last node in agenda by its successors
Order Matters!

Strategy: Replace last node in agenda by its successors

Agenda: A
Order Matters!

Strategy: Replace last node in agenda by its successors

Agenda: A AB AD
Order Matters!

Strategy: Replace last node in agenda by its successors

Agenda: A AB ABD ADA ADE ADG
Order Matters!

Strategy: Replace last node in agenda by its successors

Agenda: A AB AÐ ADA ADE AÐG ADGD ADGH
Order Matters!

Strategy: Replace last node in agenda by its successors

Agenda: A AB AĐ ADA ADE AĐG ADGD ADGH

*Depth-first* Search
Order Matters!

Strategy: Remove first node and add its successors to end

A
  /   
B  C   E
  / 
D  F
  / 
G  H
Order Matters!

Strategy: Remove first node and add its successors to end

Agenda: A
Order Matters!

Strategy: Remove first node and add its successors to end

Agenda: A AB AD
**Order Matters!**

Strategy: Remove first node and add its successors to end

![Diagram of a tree structure](image)

Agenda: A AB AD ABA ABC ABE
Order Matters!

Strategy: Remove first node and add its successors to end

Agenda: A AB AD ABA ABC ABE ADA ADE ADG
Order Matters!

Strategy: Remove first node and add its successors to end

Agenda: A AB AD ABA ABC ABE ADA ADE ADG ABAB ABAD
Order Matters!

Strategy: Remove first node and add its successors to end

Agenda: A AB AĐ ABA ABC ABE ADA ADE ADG ABAB ABAD ABCB ABCF
Order Matters!

Strategy: Remove first node and add its successors to end

Agenda: A AB AD ABA ABC ABE ADA ADE ADG ABAB ABAD ABCB ABCF ABEB ABED ABEF ABEH
Order Matters!

Strategy: Remove first node and add its successors to end

Agenda: A AB AD A◇ ABA ABC ABE ADA ADE ADG ABAB ABAD ABCB ABCF ABEB ABED ABEF ABEH ADAB ADAD
Order Matters!

Strategy: Remove first node and add its successors to end

Agenda: A AB AD ADA ABA ABC ABE ADE ADA ABCB ABAB ABAD ABAD ABCF ABEB ABED ABEF ABEH ADAB ADAD ADEB ADED ADEF ADEH
Order Matters!

Strategy: Remove first node and add its successors to end

Agenda:  A  AB  AD  AĐ  ABA  ABC  ABE  AĐA  AĐE  AĐG  ABAB  ABAD  ABCB  ABCF  ABEB  ABED  ABEF  ABEH  ADAB  ADAD  ADEB  ADED  ADEF  ADEH  ADGD  ADGH
Order Matters!

Strategy: Remove first node and add its successors to end

Agenda: A AB AD ABA ABC ABE ADA ADE ADG ABAB ABAD ABCB ABCF ABEF ABED ABEH ADAB ADAD ADEB ADED ADEF ADEH ADGD ADGH

*Breadth-first* Search
Dynamic Programming

As applies to search:
(Depends slightly on which algorithm we’re using)

BFS: The shortest path $S \rightarrow X \rightarrow G$ is made up of the shortest path $S \rightarrow X$ and the shortest path $X \rightarrow G$.

DFS: A path $S \rightarrow X \rightarrow G$ is made up of a path $S \rightarrow X$ and a path $X \rightarrow G$.

The moral: once we have found a path $S \rightarrow X$, we don’t need to spend time looking for other paths through $X$. 
Dynamic Programming

Algorithm (including dynamic programming):

• Initialize **agenda** (list of nodes to consider)
• Initialize visited set (set of states visited)
• Repeat the following:
  - Remove one node from the agenda
  - For each of that node’s children:
    • If its state is in the visited list, skip it
    • Otherwise, add it to agenda and add its state to visited list
  until **goal is found** or **agenda is empty**
• Return resulting path
Python Framework

- **SearchNode class:**
  - **Attributes:**
    - state (arbitrary)
    - parent (instance of SearchNode, or None)
  - **Methods:**
    - path (returns list of states representing path from root)

- **search function:**
  - **Arguments:**
    - successor function (function state→list of states)
    - starting state
    - goal test (function state→bool)
    - dfs (True for DFS, False for BFS)
def search(successors, start_state, goal_test, dfs = False):
    if goal_test(start_state):
        return [start_state]
    else:
        agenda = [SearchNode(start_state, None)]
        visited = {start_state}
        while len(agenda) > 0:
            parent = agenda.pop(-1 if dfs else 0)
            for child_state in successors(parent.state):
                child = SearchNode(child_state, parent)
                if goal_test(child_state):
                    return child.path()
                if child_state not in visited:
                    agenda.append(child)
                    visited.add(child_state)
        return None
def search(successors, start_state, goal_test, dfs = False):
    if goal_test(start_state):
        return [start_state]
    else:
        agenda = [SearchNode(start_state, None)]
        visited = {start_state}
        while len(agenda) > 0:
            parent = agenda.pop(-1 if dfs else 0)
            for child_state in successors(parent.state):
                child = SearchNode(child_state, parent)
                if goal_test(child_state):
                    return child.path()
                if child_state not in visited:
                    agenda.append(child)
                    visited.add(child_state)
        return None
def search(successors, start_state, goal_test, dfs = False):
    if goal_test(start_state):
        return [start_state]
    else:
        agenda = [SearchNode(start_state, None)]
        visited = {start_state}
        while len(agenda) > 0:
            parent = agenda.pop(-1 if dfs else 0)
            for child_state in successors(parent.state):
                child = SearchNode(child_state, parent)
                if goal_test(child_state):
                    return child.path()
                if child_state not in visited:
                    agenda.append(child)
                    visited.add(child_state)
        return None
Example

Find path $A \rightarrow I$, BFS w/ DP
What is a Graph?

Set $V$ of vertices
Set $E$ of edges connecting vertices
Set $W$ of edge costs (or “weights”)
Example

Find path $A \to I$, *minimizing total cost*
Uniform-Cost Search

Consider searching for least-cost paths instead of shortest paths. Instead of popping from agenda based on when nodes were added, pop based on the cost of the paths they represent.

Slight change to framework:

- **SearchNode class:**
  - Attributes:
    - state (arbitrary)
    - parent (instance of SearchNode, or None)
    - cost of whole path from start
  - Methods:
    - path (returns list of states representing path from root)

- uniform_cost_search function:
  - Arguments:
    - successor function (state→list of (state,cost) tuples)
    - starting state
    - goal test (function state→bool)
def uniform_cost_search(successors, start_state, goal_test):
    if goal_test(start_state):
        return [start_state]
    agenda = [(0, SearchNode(start_state, None, cost=0))]
    expanded = set()
    while len(agenda) > 0:
        agenda.sort()
        priority, parent = agenda.pop(0)
        if parent.state not in expanded:
            expanded.add(parent.state)
            if goal_test(parent.state):
                return parent.path()
            for child_state, cost in successors(parent.state):
                child = SearchNode(child_state, parent, parent.cost+cost)
                if child_state not in expanded:
                    agenda.append((child.cost, child))
    return None

Testing for goal condition must be done at \textit{expansion} time, not at visit time. Similarly for dynamic programming.
Example

Find path $A \rightarrow I$, Uniform Cost Search
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.

We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point.
We only notice the goal when we stumble upon it.
Problem?

So far, searches have radiated outward from the starting point. We only notice the goal when we stumble upon it.

![Diagram showing a search process with the goal off the path of the search]

Too much time spent searching on the **wrong side of the goal**.
Example

Find path $E \rightarrow I$, Uniform Cost Search

![Graph Diagram]
Heuristics

So far, our searches only consider start-to-current. We can add heuristics to consider an estimate of current-to-goal as well.

\( h(x) \) estimate of cost of lowest-cost path \( X \rightarrow \text{goal} \)

- **SearchNode class:**
  - Attributes:
    - state (arbitrary)
    - parent (instance of SearchNode, or None)
    - cost of whole path from start
  - Methods:
    - path (returns list of states representing path from root)

- **uniform_cost_search function:**
  - Arguments:
    - successor function (state \( \rightarrow \) list of (state, cost) tuples)
    - starting state
    - goal test (function state \( \rightarrow \) bool)
    - heuristic (function state \( \rightarrow \) estimated cost)
def uniform_cost_search(successors, start_state, goal_test, heuristic=lambda s: 0):
    if goal_test(start_state):
        return [start_state]
    agenda = [(heuristic(start_state), SearchNode(start_state, None, cost=0))]
    expanded = set()
    while len(agenda) > 0:
        agenda.sort()
        priority, parent = agenda.pop(0)
        if parent.state not in expanded:
            expanded.add(parent.state)
            if goal_test(parent.state):
                return parent.path()
            for child_state, cost in successors(parent.state):
                child = SearchNode(child_state, parent, parent.cost+cost)
                if child.state not in expanded:
                    agenda.append((child.cost+heuristic(child_state), child))
    return None
Example

Find path $E \rightarrow I$, A*, heuristic: $h(s) = M(s, I)/2$
Admissible Heuristics

An **admissible heuristic** does not overestimate the actual cost of the shortest cost path.

If the heuristic \( h(s) \) is larger than the actual cost from \( s \) to goal, then the “best” solution may be missed!

If the heuristic is an underestimate, the search space will be larger than necessary, but we are guaranteed the shortest path.

The ideal heuristic should be:
- as close as possible to actual cost (without overestimating)
- easy to calculate

**A* (without DP) is guaranteed to find least-cost path if heuristic is admissible.**

With DP, heuristic must also be *consistent*. 
Check Yourself!

Consider searching in a four-action grid (up, down, left, right), where all actions have cost 1. Let \((r_0, c_0)\) represent the current location, and \((r_1, c_1)\) represent the goal.

Which of the following heuristics are **admissible**?

1. \(\text{abs}(r_0-r_1) + \text{abs}(c_0-c_1)\)
2. \(\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))\)
3. \(\text{max}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))\)
4. \(2*\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))\)
5. \(2*\text{max}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))\)
Consider searching in a four-action grid (up, down, left, right), where all actions have cost 1. Let \((r_0, c_0)\) represent the current location, and \((r_1, c_1)\) represent the goal.

Which of the following heuristics are **admissible**?

1. \(\text{abs}(r_0-r_1) + \text{abs}(c_0-c_1)\)
2. \(\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))\)
3. \(\text{max}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))\)
4. \(2*\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))\)
5. \(2*\text{max}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))\)
Check Yourself!

Which of the admissible heuristics minimizes the number of nodes expanded?

1. \( \text{abs}(r_0-r_1) + \text{abs}(c_0-c_1) \)
2. \( \text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \)
3. \( \text{max}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \)
4. \( 2\times\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \)
5. \( 2\times\text{max}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \)
Check Yourself!

Which of the admissible heuristics minimizes the number of nodes expanded?

1. \( \text{abs}(r_0-r_1) + \text{abs}(c_0-c_1) \)
2. \( \text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \)
3. \( \text{max}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \)
4. \( 2\times\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \)
5. \( 2\times\text{max}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
\[ \text{abs}(r_0 - r_1) + \text{abs}(c_0 - c_1) \]
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
\[ \text{abs}(r_0 - r_1) + \text{abs}(c_0 - c_1) \]
\[ \text{abs}(r0-r1) + \text{abs}(c0-c1) \]
\[ \text{abs}(r_0-r_1) + \text{abs}(c_0-c_1) \]
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
abs(r0-r1) + abs(c0-c1)
$\text{abs}(r_0-r_1) + \text{abs}(c_0-c_1)$
$\text{abs}(r0-r1) + \text{abs}(c0-c1)$
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\(\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))\)
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
min(abs(r0-r1), abs(c0-c1))
\( \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \)
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[
\text{min}(\text{abs}(r0-r1), \text{abs}(c0-c1))
\]
\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
min(abs(r0-r1), abs(c0-c1))
\text{min}(\text{abs}(r0-r1), \text{abs}(c0-c1))
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\( \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \)
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[
\min(\abs{r_0-r_1}, \abs{c_0-c_1})
\]
\[
\min(\abs{r_0-r_1}, \abs{c_0-c_1})
\]
min(abs(r0-r1), abs(c0-c1))
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\text{min}(|r_0 - r_1|, \ |c_0 - c_1|)
\min(\abs{r_0-r_1}, \abs{c_0-c_1})
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
min(abs(r0-r1), abs(c0-c1))
\[
\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\]
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\text{min}(\text{abs}(r0-r1), \text{abs}(c0-c1))
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\text{min}(\text{abs}(r0-r1), \text{abs}(c0-c1))
\[
\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\]
\[ \text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\[ \text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
$\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))$
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\text{min}(\text{abs}(r0-r1), \text{abs}(c0-c1))
\[
\min(\abs{r_0-r_1}, \abs{c_0-c_1})
\]
\[ \min(\abs{r0-r1}, \abs{c0-c1}) \]
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
min(abs(r0-r1), abs(c0-c1))
min(abs(r0-r1), abs(c0-c1))
\[
\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\]
\( \min(\abs{r_0-r_1}, \abs{c_0-c_1}) \)
\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\min(\text{abs}(r0-r1), \text{abs}(c0-c1))
$\min(\text{abs}(r0-r1), \text{abs}(c0-c1))$
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\text{min}(\text{abs}(r0-r1), \text{abs}(c0-c1))
\text{min}(\text{abs}(r0-r1), \text{abs}(c0-c1))
\[
\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\]
\[ \text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))\]
\[
\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\]
\( \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \)
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[
\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\]
\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
$$\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))$$
min(abs(r0-r1), abs(c0-c1))
\min(\abs{r_0-r_1}, \abs{c_0-c_1})
\[
\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\]
\text{min}(\text{abs}(r0-r1), \text{abs}(c0-c1))
\text{min}(\text{abs}(r0-r1), \text{abs}(c0-c1))
\[
\min(\text{abs}(r0-r1), \text{abs}(c0-c1))
\]
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\[
\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\]
\( \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \)
\[\min(\text{abs}(r_0 - r_1), \text{abs}(c_0 - c_1))\]
\texttt{min(abs(r0-r1), abs(c0-c1))}
\[ \text{min}(|a-b|, |c-d|) \]
min(abs(r0-r1), abs(c0-c1))
\[ \text{min}(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\( \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \)
min(abs(r0-r1), abs(c0-c1))
min(abs(r0-r1), abs(c0-c1))
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\( \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \)
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
$$\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))$$
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\[
\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\]
\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[
\min(\text{abs}(r0-r1), \text{abs}(c0-c1))
\]
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\[ \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
min(abs(r0-r1), abs(c0-c1))
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\min(\abs{r_0-r_1}, \abs{c_0-c_1})
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\( \min(\text{abs}(r0-r1), \text{abs}(c0-c1)) \)
\min(\text{abs}(r0-r1), \text{abs}(c0-c1))
\[ \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
$\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))$
\[ \text{min}(\text{abs(r0-r1), abs(c0-c1)}) \]
max(abs(r0-r1), abs(c0-c1))
\[ \max(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\[ \max(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\text{max}(\text{abs}(r0-r1), \text{abs}(c0-c1))
\[ \text{max}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \max(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\[ \max(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \max(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \max(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
max(abs(r0-r1), abs(c0-c1))
max(abs(r0-r1), abs(c0-c1))
\text{max}(\text{abs}(r0-r1), \text{abs}(c0-c1))
\text{max}(\text{abs}(r0-r1), \text{abs}(c0-c1))
\text{max}(abs(r0-r1), abs(c0-c1))
\[ \max(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\text{max}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\[ \text{max}(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\[ \max(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \text{max}(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\text{max}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
max(abs(r0-r1), abs(c0-c1))
max(abs(r0-r1), abs(c0-c1))
$$\max(\text{abs}(r0-r1), \text{abs}(c0-c1))$$
\text{max}\left(abs(r_0-r_1), \, abs(c_0-c_1)\right)
\[
\max(\text{abs}(r0-r1), \text{abs}(c0-c1))
\]
\max(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
max(abs(r0-r1), abs(c0-c1))
\[
\max(\text{abs}(r0-r1), \text{abs}(c0-c1))
\]
\[ \max(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \max(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \max(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\text{max}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\[ \max(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
max(abs(r0-r1), abs(c0-c1))
\text{max}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\text{max}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
max(abs(r0-r1), abs(c0-c1))
max(abs(r0-r1), abs(c0-c1))
max(abs(r0-r1), abs(c0-c1))
\[ \max(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \max(\abs{r_0-r_1}, \abs{c_0-c_1}) \]
max(abs(r0-r1), abs(c0-c1))
\[ \max(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \max(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
max(abs(r0-r1), abs(c0-c1))
\text{max}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\[ \max(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \max(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\text{max}(\text{abs}(r0-r1), \text{abs}(c0-c1))
max(abs(r0-r1), abs(c0-c1))
\[ \max(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
\[ \text{max}(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\[ \max(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
max(abs(r0-r1), abs(c0-c1))
\max(\abs{r0-r1}, \abs{c0-c1})
\text{max}(|r_0-r_1|, |c_0-c_1|)
\[ \max(\text{abs}(r0-r1), \text{abs}(c0-c1)) \]
\[
\max(\text{abs(r0-r1)}, \text{abs(c0-c1)})
\]
\( \max(\text{abs}(r0-r1), \text{abs}(c0-c1)) \)
2*min(abs(r0-r1), abs(c0-c1))
2*\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
2*\min(\abs{r_0-r_1}, \abs{c_0-c_1})
2*\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
2*\min(\abs{r0-r1}, \abs{c0-c1})
2*min(abs(r0-r1), abs(c0-c1))
2*min(abs(r0-r1), abs(c0-c1))
\[2 \times \min(|r_0 - r_1|, |c_0 - c_1|)\]
2*min(abs(r0-r1), abs(c0-c1))
2*min(abs(r0-r1), abs(c0-c1))
2*min(abs(r0-r1), abs(c0-c1))
2*\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
$2 \times \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))$
2 * min(abs(r0-r1), abs(c0-c1))
2*min(abs(r0-r1), abs(c0-c1))
2*min(abs(r0-r1), abs(c0-c1))
2*min(abs(r0-r1), abs(c0-c1))
2*min(abs(r0-r1), abs(c0-c1))
2*min(abs(r0-r1), abs(c0-c1))
\[2*\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))\]
2*\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
2*\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
2*min(abs(r0-r1), abs(c0-c1))
\[2 \times \min(|r_0 - r_1|, |c_0 - c_1|)\]
2*min(abs(r0-r1), abs(c0-c1))
2*min(abs(r0-r1), abs(c0-c1))
2*min(abs(r0-r1), abs(c0-c1))
2*min(abs(r0-r1), abs(c0-c1))
$2 \times \min(\abs{r_0 - r_1}, \abs{c_0 - c_1})$
2*min(abs(r0-r1), abs(c0-c1))
2*min(abs(r0-r1), abs(c0-c1))
\[2 \times \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))\]
2*min(abs(r0-r1), abs(c0-c1))
2*\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\[2 \times \min(\abs{r_0 - r_1}, \abs{c_0 - c_1})\]
2*\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
2*\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
2*min(abs(r0-r1), abs(c0-c1))
2*min(abs(r0-r1), abs(c0-c1))
$2\times \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))$
\[ 2 \times \min(|r_0 - r_1|, |c_0 - c_1|) \]
2*min(abs(r0-r1), abs(c0-c1))
2*min(abs(r0-r1), abs(c0-c1))
\[2 \times \min(\text{abs}(r_0-r_1), \ \text{abs}(c_0-c_1))\]
2*min(abs(r0-r1), abs(c0-c1))
2*\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
2*\text{min}(|r_0-r_1|, |c_0-c_1|)
2*\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\[2 \times \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))\]
2*\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
2*min(abs(r0-r1), abs(c0-c1))
$2 \times \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))$
$2 \times \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))$
$2 \times \min(\text{abs}(r_0 - r_1), \text{abs}(c_0 - c_1))$
2*min(abs(r0-r1), abs(c0-c1))
\[2 \times \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))\]
2*\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
\[2 \times \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))\]
$2 \times \text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))$
2*min(abs(r0-r1), abs(c0-c1))
\[ 2 \times \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1)) \]
2*\text{min}(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))
2*\text{min}(\text{abs}(r0-r1), \text{abs}(c0-c1))
2*\text{min}(\text{abs}(r0-r1), \text{abs}(c0-c1))
2*min(abs(r0-r1), abs(c0-c1))
$2\times\min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))$
2*\min(\abs{r_0-r_1}, \abs{c_0-c_1})
2*\text{min}(\text{abs}(r0-r1), \text{abs}(c0-c1))
2*min(abs(r0-r1), abs(c0-c1))
$2 \times \min(\text{abs}(r_0-r_1), \text{abs}(c_0-c_1))$
2*min(abs(r0-r1), abs(c0-c1))
2*min(abs(r0-r1), abs(c0-c1))
## Compare

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>States Expanded</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{abs}(r_0 - r_1) + \text{abs}(c_0 - c_1)$</td>
<td>42</td>
</tr>
<tr>
<td>$\text{min}(\text{abs}(r_0 - r_1), \text{abs}(c_0 - c_1))$</td>
<td>114</td>
</tr>
<tr>
<td>$\text{max}(\text{abs}(r_0 - r_1), \text{abs}(c_0 - c_1))$</td>
<td>60</td>
</tr>
<tr>
<td>$2\times\text{min}(\text{abs}(r_0 - r_1), \text{abs}(c_0 - c_1))$</td>
<td>72</td>
</tr>
</tbody>
</table>
Heuristics in Other Domains

Consider the "word ladder" problem from EX12 (and last lecture). Searching from "quiz" to "best".
quiz
quid
quip
quit
quad
quay
skit
slit
snit
suet
skid
skim
skip
skis
alit
flit
slat
slot
slid
slim
slip
knit
unit
snip
spat
spot
spin
duet
stet
sued
sues
said
shim
swim
akin
shin
ship
alia
flat
flip
plat
scat
seat
swat
slab
slag
SLAM
slap
slav
slaw
slay
blot
clot
plot
scot
shot
soot
sloe
slop
slow
sled
slum
blip
clip
knot
snap
span
spar
spay
spun
diet
dust
duel
dues
stem
step
stew
cued
hued
rued
seed
shed
sped
surd
cues
hues
rues
sees
subs
suds
sums
suns
sups
laid
maid
paid
raid
sand
sail
whim
sham
swam
swum
swig
chin
thin
shun
chip
whip
shop
ilia
aria
asia
alga
alma
fiat
flag
flak
flap
flaw
flax
flay
flop
peat
plan
play
scab
scan
scar
beat
heat
meat
neat
teat
sect
seal
seam
sear
seas
swab
swag
swan
swap
sway
blab
stab
snag
stag
slug
clam
siam
clap
soap
claw
slew
clay
shay
stay
boot
blob
bloc
blow
coot
clod
clog
clop
cloy
plod
plop
plow
ploy
SCOW
shut
shod
shoe
shoo
show
foot
hoot
loot
moot
root
toot
soft
sort
soon
aloe
floe
slue
flog
smog
stop
flow
glow
snow
stow
bled
fled
pled
glum
plum
Scum
slur
knob
know
soar
star
spur
spry
stun
spud
dint
dirt
died
diem
dies
duck
bust
gust
just
lust
must
oust
rust
dusk
fuel
dual
dull
does
dyes
dubs
duds
duns
duos
item
seem
seep
skew
spew
curd
heed
hied
hoed
reed
deed
feed
need
weed
send
seek
seen
seer
shad
aped
sure
surf
cubs
cuds
cups
curs
cuss
cuts
hies
hubs
hugs
hums
huns
huts
ryes
rubs
rugs
rums
runs
ruts
bees
fees
lees
sets
sews
nubs
pubs
tubs
sibs
sobs
buds
muds
sods
bums
gums
mums
buns
funs
guns
nuns
puns
tuns
sins
sons
sung
sunk
pups
saps
sips
sops
land
lard
laud
laic
lain
lair
mail
maim
main
pard
pail
pain
pair
rail
rain
band
hand
wand
sane
sang
sank
bail
fail
hail
jail
nail
tail
wail
soil
wham
whom
whig
whir
whit
whiz
shah
twig
cain
coin
chic
twin
than
then
this
chap
chop
ilea
area
arid
aril
alms
feet
felt
fear
fist
flew
flex
flux
fray
pelt
pert
pest
peak
peal
pear
peas
clan
elan
klan
pray
boat
brat
beet
bent
best
quiz
quit
quid
quip
suit
Words: A*
skit
slit
sidet
spit
Words: A*

suet
Words: A*

alit
Words: A*
slat
slot
Words: A*
unit
spat
spot
duet
Words: A*

stet
Words: A*

seat
Words: A*

blot
dust
Words: A*

beat
bust
Words: A*

best
Heuristics in other domains

Consider the "word ladder" problem from EX12 (and last lecture). Searching from "quiz" to "best".

**UC**
States expanded: 396
['quiz', 'quit', 'suit', 'slit', 'slat', 'seat', 'beat', 'best']

**A**
States expanded: 28
['quiz', 'quit', 'suit', 'slit', 'slat', 'seat', 'beat', 'best']
Example: 8-Puzzle

Start

1 2 3
4 5 6
7 8

Goal

1 2
3 4 5
6 7 8
Example: 8-Puzzle
Example: 8-Puzzle
Example: 8-Puzzle
Example: 8-Puzzle
Example: 8-Puzzle
Example: 8-Puzzle
Example: 8-Puzzle
Example: 8-Puzzle
Example: 8-Puzzle
Example: 8-Puzzle
Example: 8-Puzzle

```
 1 4 2
 7 3 5
8 6
```
Example: 8-Puzzle

<table>
<thead>
<tr>
<th>1</th>
<th>4</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
Example: 8-Puzzle
Example: 8-Puzzle
Example: 8-Puzzle
Example: 8-Puzzle
Example: 8-Puzzle

1 4 2
6 3 5
7 8
Example: 8-Puzzle
Example: 8-Puzzle
Example: 8-Puzzle
Example: 8-Puzzle
Example: 8-Puzzle

```
 1  2
 3  4  5
 6  7  8
```
Example: 8-Puzzle
Example: 8-Puzzle

Large number of board configurations (states):
- $9! = 362,880$ (if you count all)
- $9!/2 = 181,440$ accessible from start state

Almost half of accessible states ($84,516$) are expanded by UC.
Check Yourself!

Consider three heuristics for the “eight puzzle”:

1. 0
2. number of tiles out of place
3. sum over tiles of Manhattan distances to their goals

Let $M_i =$ num. moves in the best solution using heuristic $i$.
Let $E_i =$ num. states expanded using heuristic $i$.

Which of the following are true?

1. $M_1 = M_2 = M_3$
2. $M_1 > M_2 > M_3$
3. $E_1 = E_2 = E_3$
4. $E_1 \geq E_2 \geq E_3$
5. the same “best” solution results from all three heuristics
Check Yourself!

Consider three heuristics for the “eight puzzle”:

1. 0
2. number of tiles out of place
3. sum over tiles of Manhattan distances to their goals

Let $M_i =$ num. moves in the best solution using heuristic $i$.
Let $E_i =$ num. states expanded using heuristic $i$.

Which of the following are true?

1. $M_1 = M_2 = M_3$
2. $M_1 > M_2 > M_3$
3. $E_1 = E_2 = E_3$
4. $E_1 \geq E_2 \geq E_3$
5. the same “best” solution results from all three heuristics
Check Yourself!

Results.

Heuristics:

1. 0
2. number of tiles out of place
3. sum over tiles of Manhattan distances to their goals

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Expanded</th>
<th>Moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>84,516</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>8,329</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>1,348</td>
<td>22</td>
</tr>
</tbody>
</table>
Recap

Developed a new class of search algorithms: uniform cost
Developed a new class of optimizations: heuristics

Summary of Search Algorithms:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Agenda</th>
<th>Goal Test</th>
<th>DP</th>
<th>Guarantees†</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BFS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

† Provided a path exists
Recap

Developed a new class of search algorithms: uniform cost
Developed a new class of optimizations: heuristics

Summary of Search Algorithms:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Agenda</th>
<th>Goal Test</th>
<th>DP</th>
<th>Guarantees†</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFS</td>
<td>Stack (LIFO)</td>
<td>Visit</td>
<td>Visited Set</td>
<td>Some Path*</td>
</tr>
<tr>
<td>BFS</td>
<td>Queue (FIFO)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC</td>
<td>Priority Queue</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

† Provided a path exists
* In a finite search domain
Recap

Developed a new class of search algorithms: uniform cost
Developed a new class of optimizations: heuristics

Summary of Search Algorithms:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Agenda</th>
<th>Goal Test</th>
<th>DP</th>
<th>Guarantees(^\d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFS</td>
<td>Stack (LIFO)</td>
<td>Visit</td>
<td>Visited Set</td>
<td>Some Path(^*)</td>
</tr>
<tr>
<td>BFS</td>
<td>Queue (FIFO)</td>
<td>Visit</td>
<td>Visited Set</td>
<td>Shortest Path</td>
</tr>
<tr>
<td>UC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^\d\) Provided a path exists
\(^*\) In a finite search domain
Recap

Developed a new class of search algorithms: uniform cost
Developed a new class of optimizations: heuristics

Summary of Search Algorithms:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Agenda</th>
<th>Goal Test</th>
<th>DP</th>
<th>Guarantees†</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFS</td>
<td>Stack (LIFO)</td>
<td>Visit</td>
<td>Visited Set</td>
<td>Some Path*</td>
</tr>
<tr>
<td>BFS</td>
<td>Queue (FIFO)</td>
<td>Visit</td>
<td>Visited Set</td>
<td>Shortest Path</td>
</tr>
<tr>
<td>UC</td>
<td>Priority Queue</td>
<td>Expand</td>
<td>Expanded Set</td>
<td>Least-cost Path</td>
</tr>
</tbody>
</table>

† Provided a path exists
* In a finite search domain