Announcements

- Coding Homework 1 will be released
  - Due 1/25 at 11:59pm
- Written Homework 1 will be released
  - Due 1/25 at 11:59pm

Informed Search

Forest Agostinelli
University of South Carolina

Topics Covered in This Class

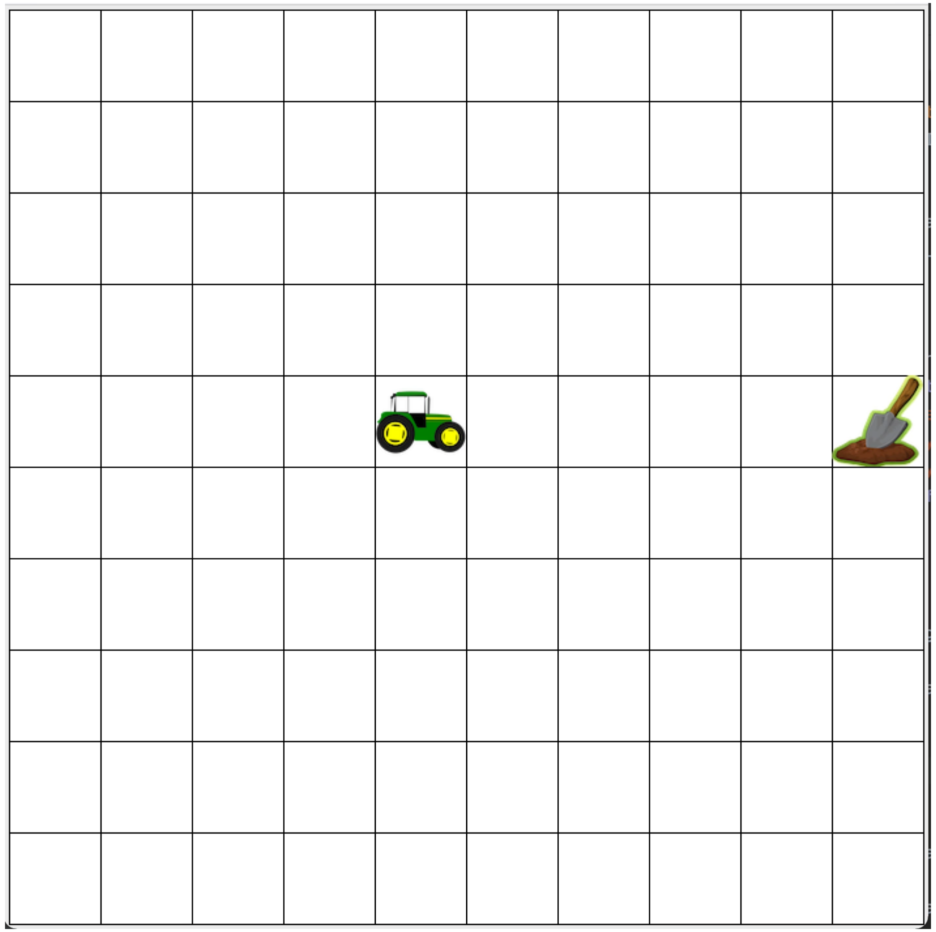
- **Part 1: Search**
  - Pathfinding
    - Uninformed search
    - Informed search
  - Adversarial search
  - Optimization
    - Local search
    - Constraint satisfaction
- **Part 2: Knowledge Representation and Reasoning**
  - Propositional logic
  - First-order logic
  - Prolog
- **Part 3: Knowledge Representation and Reasoning Under Uncertainty**
  - Probability
  - Bayesian networks
- **Part 4: Machine Learning**
  - Supervised learning
    - Inductive logic programming
    - Linear models
    - Deep neural networks
    - PyTorch
  - Reinforcement learning
    - Markov decision processes
    - Dynamic programming
    - Model-free RL
  - Unsupervised learning
    - Clustering
    - Autoencoders

Outline

- Heuristics
- Greedy Best-First search
- A* search
- Weighted A* search

Informed Search: Motivation

- Uniform cost search is guaranteed to find a shortest path
- However, it prioritizes all nodes with the same path cost equally
- We can see, intuitively, why this is undesirable
- How can we do better?



Heuristic

- A commonsense rule intended to increase the probability or efficiency of solving a problem
  - Rules of thumb
  - Educated guesses
  - Intuition
- In this context, it should estimate how close we are to solving the problem
- Problem specific
  - Reaching the shovel vs chemical synthesis vs solving the Rubik’s cube etc.

Outline

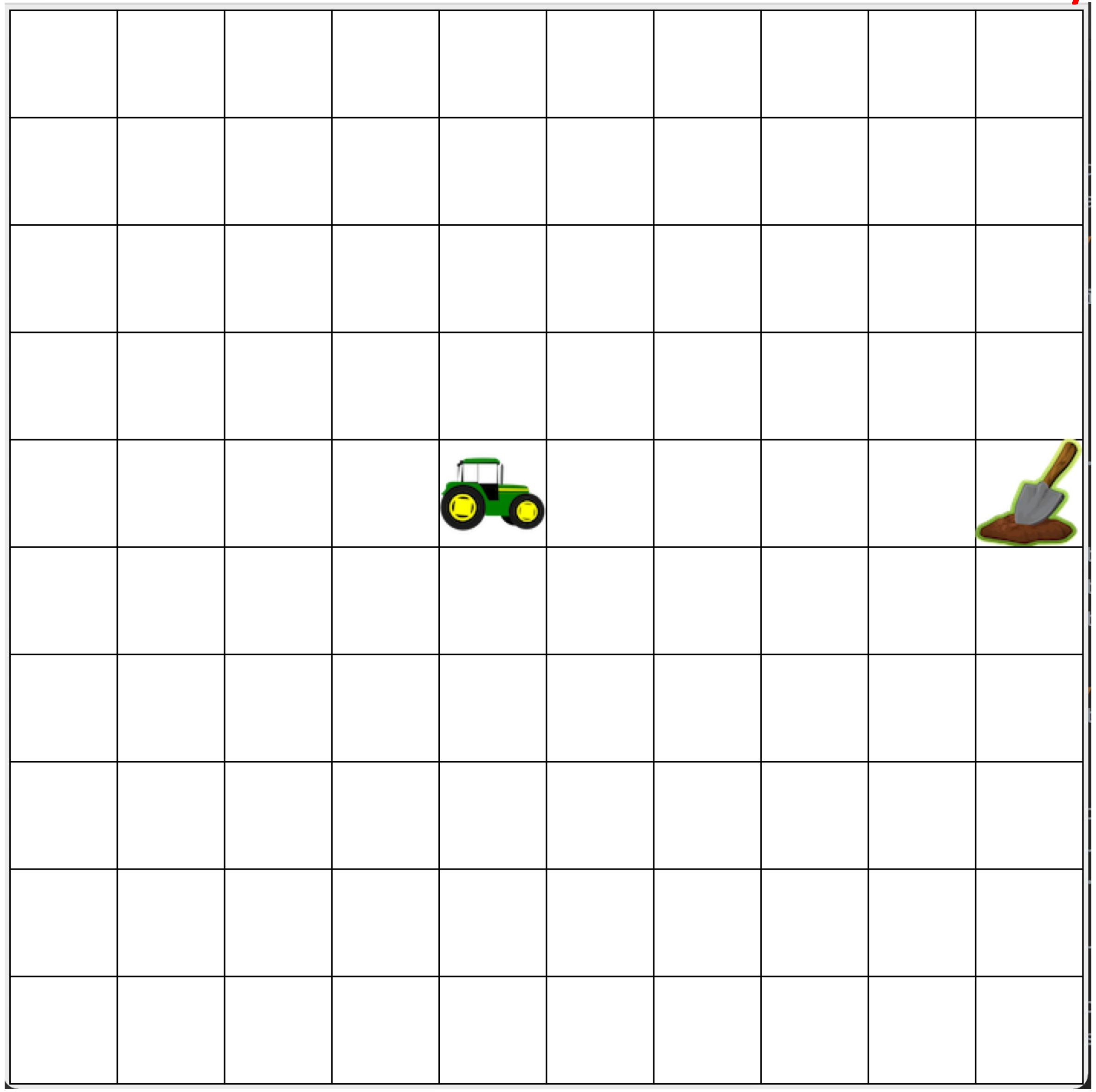
- Heuristics
- Greedy Best-First search
- A* search
- Weighted A* search

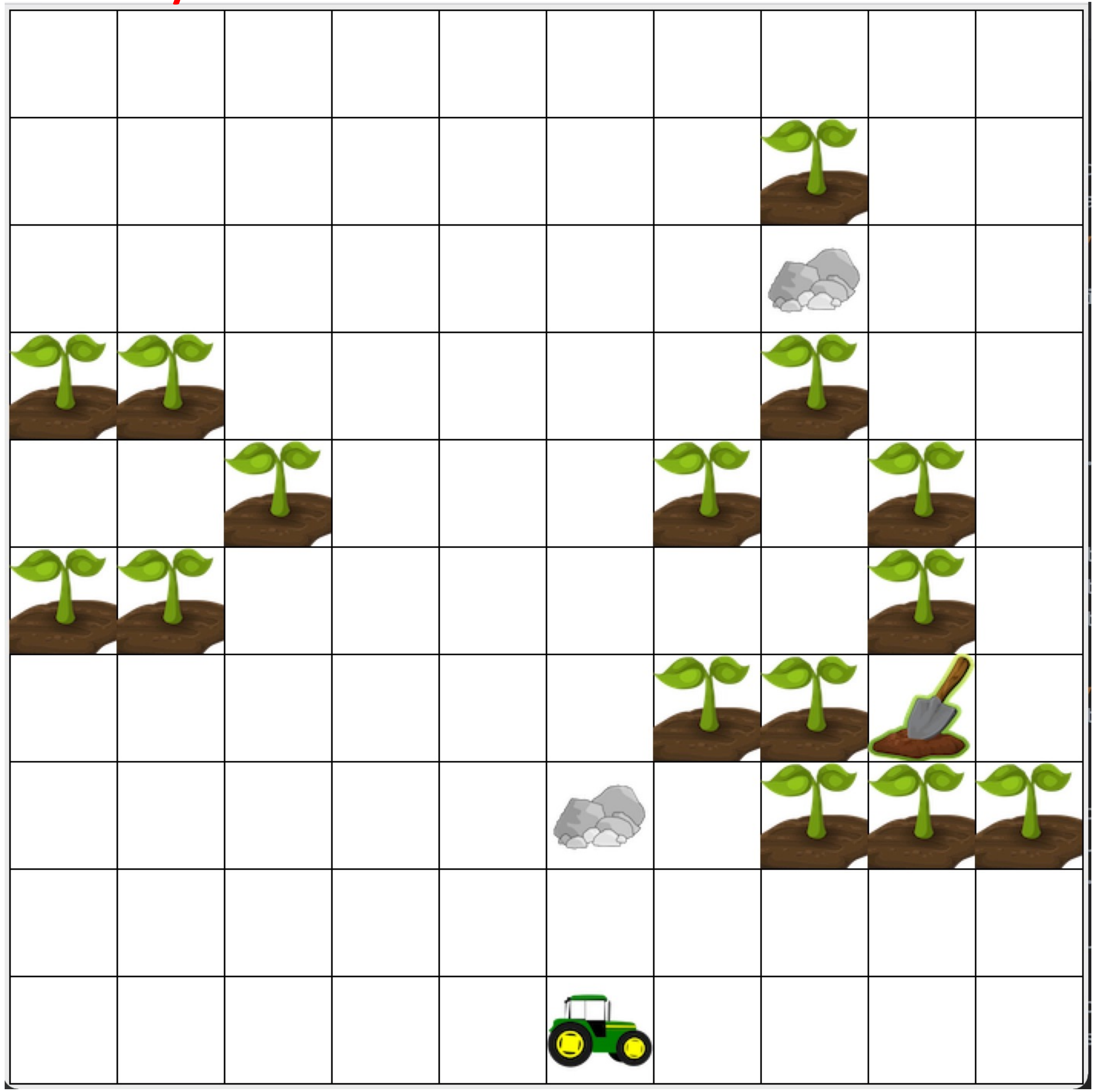
Greedy Best-First Search

- Instead of prioritizing nodes with the lowest path cost, we prioritize nodes with the lowest heuristic
- We will start with the Manhattan distance heuristic
  - $|x_1 - x_2| + |y_1 - y_2|$
  - Frequently used on grid-like problems

Greedy Best-First Search: Manhattan Distance

Not always optimal! Why?



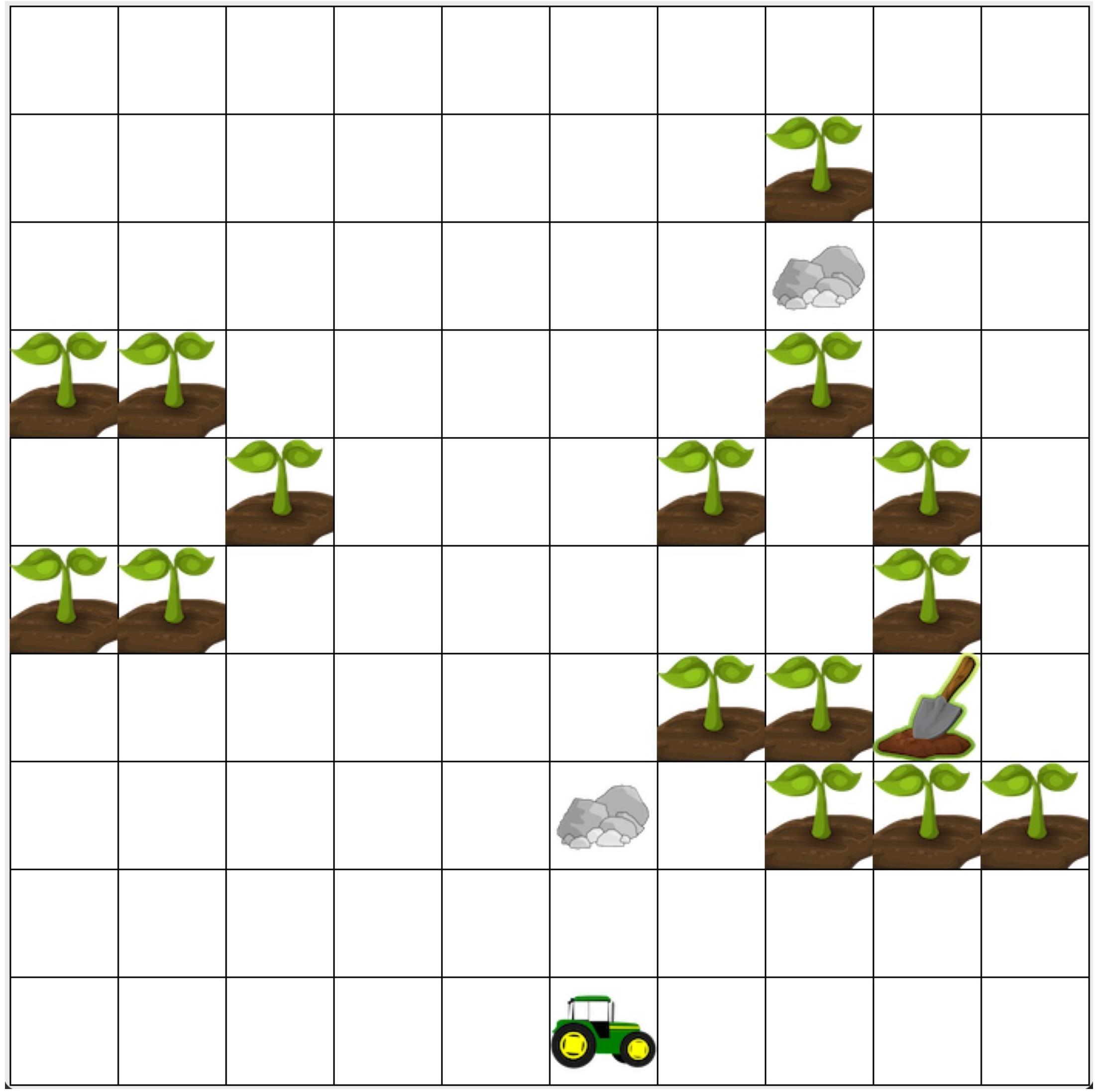


Outline

- Heuristics
- Greedy Best-First search
- A* search
- Weighted A* search

A* Search

- Priority is equal to the sum of the path cost and the heuristic
- $f(n) = g(n) + h(n)$
  - $f(n)$: cost
  - $g(n)$: path cost (cost to get from n_0 to n)
  - $h(n)$: heuristic (estimated cost to get from n to $n_g \in \mathcal{G}$)



A* Search

function BEST-FIRST-SEARCH(*problem*, *f*) **returns** a solution node or *failure*
  node ← NODE(STATE=*problem*.INITIAL)
  frontier ← a priority queue ordered by *f*, with *node* as an element
  reached ← a lookup table, with one entry with key *problem*.INITIAL and value *node*
  while not IS-EMPTY(*frontier*) **do**
    node ← POP(*frontier*)
    if *problem*.IS-GOAL(*node*.STATE) **then return** *node*
    for each *child* **in** EXPAND(*problem*, *node*) **do**
      s ← *child*.STATE
      if *s* is not in *reached* **or** *child*.PATH-COST < *reached*[*s*].PATH-COST **then**
        reached[*s*] ← *child*
        add *child* to *frontier*
  return *failure*

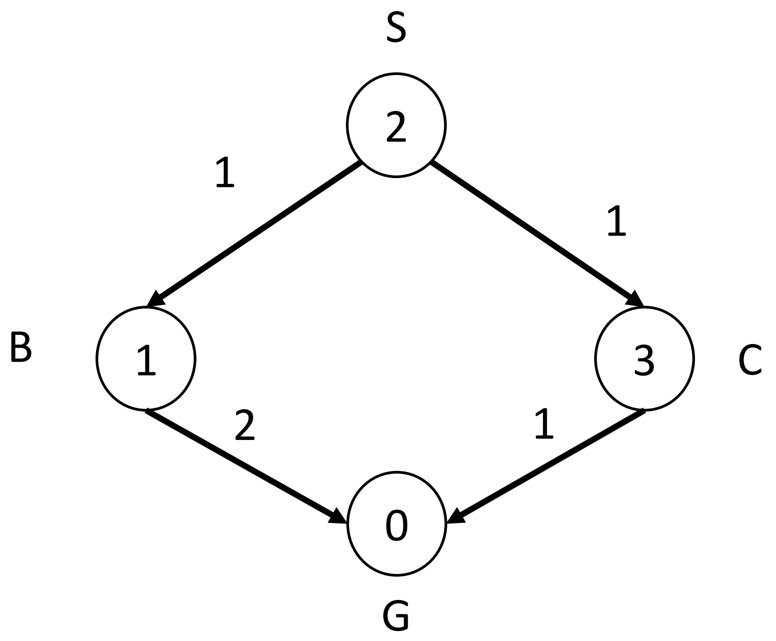
Only do a goal test when when the node is selected for expansion!

A* Search Performance

- A* search is complete
- Time and memory still has exponential complexity in the worst case
  - Given an informative heuristic, this can be significantly reduced
- A* search is guaranteed to find a shortest path when the heuristic function is admissible
  - h is admissible if and only if h never overestimates the cost of a shortest path
  - In other words, $h(n) \leq h^*(n)$ for all n
    - Where $h^*(n)$ is the cost of an optimal path from n to the goal

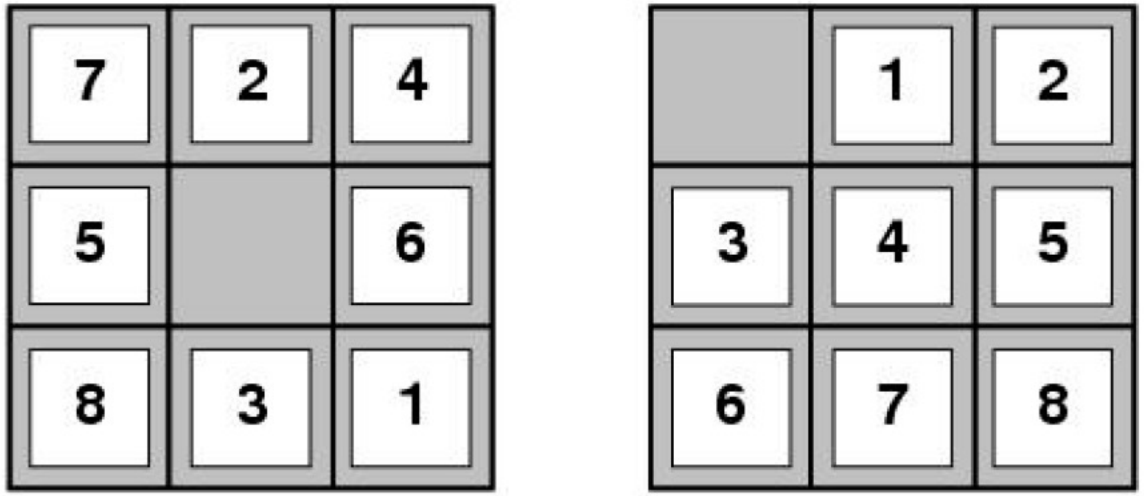
A* Search: Admissibility

- Heuristics are given inside the nodes
  - One is not admissible
- What is the shortest path?
- What is the path found when node S is the start node and node G is the goal node?



CLOSED	OPEN	Expanded
S: 0	S: 2	S
S: 0, B: 1, C: 1	B: 2, C: 4	B
S: 0, B: 1, C: 1, G: 3	G: 3, C: 4	(G)

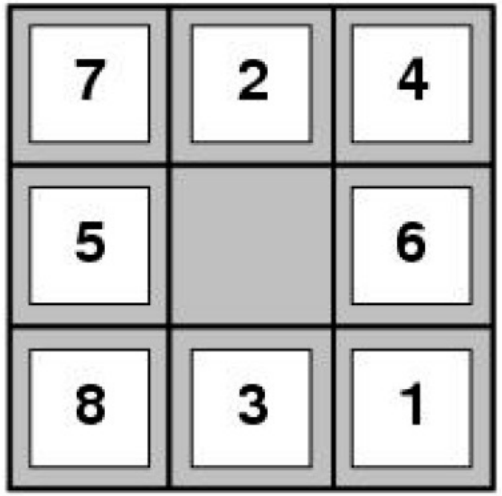
Admissible Heuristics: 8-puzzle



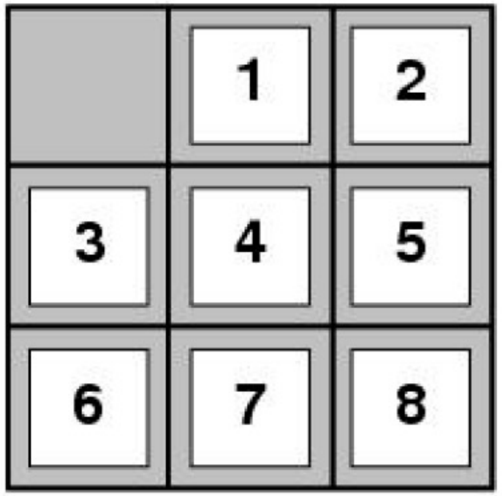
Start state Goal state

- 1.8×10^5 states
- Larger versions
  - 24-puzzle: 7.7×10^{24} states
  - 48-puzzle: 3.0×10^{62} states
- Think of some admissible heuristics

Admissible Heuristics: 8-puzzle



Start state



Goal state

- Number of misplaced tiles
  - 8
- Manhattan distance of all tiles to their goal
  - 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18
- Which is better?

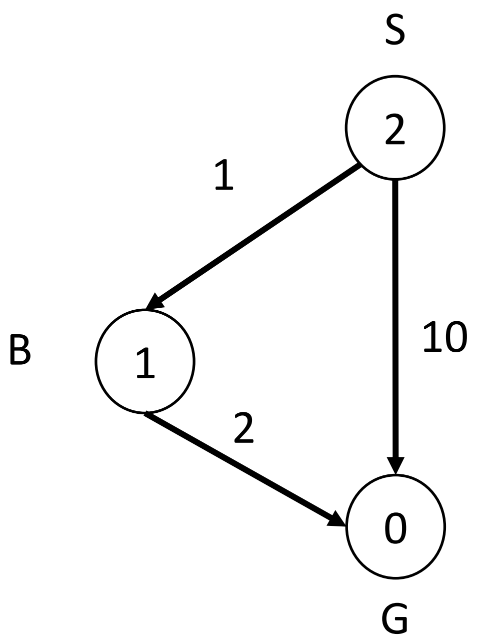
Comparing Heuristics for A* Search

- If $h_2(n) \geq h_1(n)$ for all n, then h_2 dominates h_1.
- If both are admissible, then h_2 will expand no more nodes than h_1

Performance on the 8-puzzle where h1 is the number of tiles misplaced and h2 is the sum of Manhattan distances
d is the depth of the solution and IDS is iterative deepening search

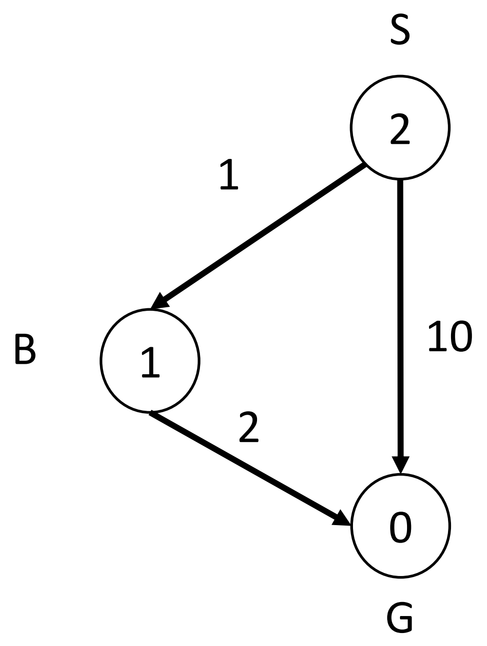
d	IDS	A*(h1)	A*(h2)
2	10	6	6
4	112	13	12
8	6384	39	25
12	364404	227	73
14	3473941	539	113
20	------------	7276	676
24	------------	39135	1641

A* Search and CLOSED/Reached



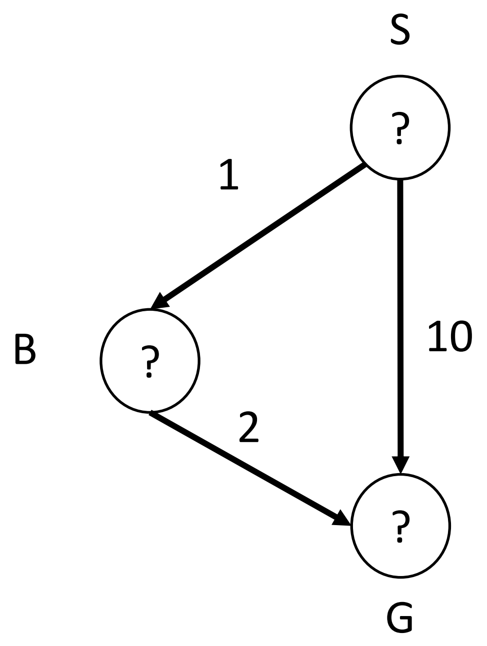
CLOSED	OPEN	Expanded

A* Search and CLOSED/Reached



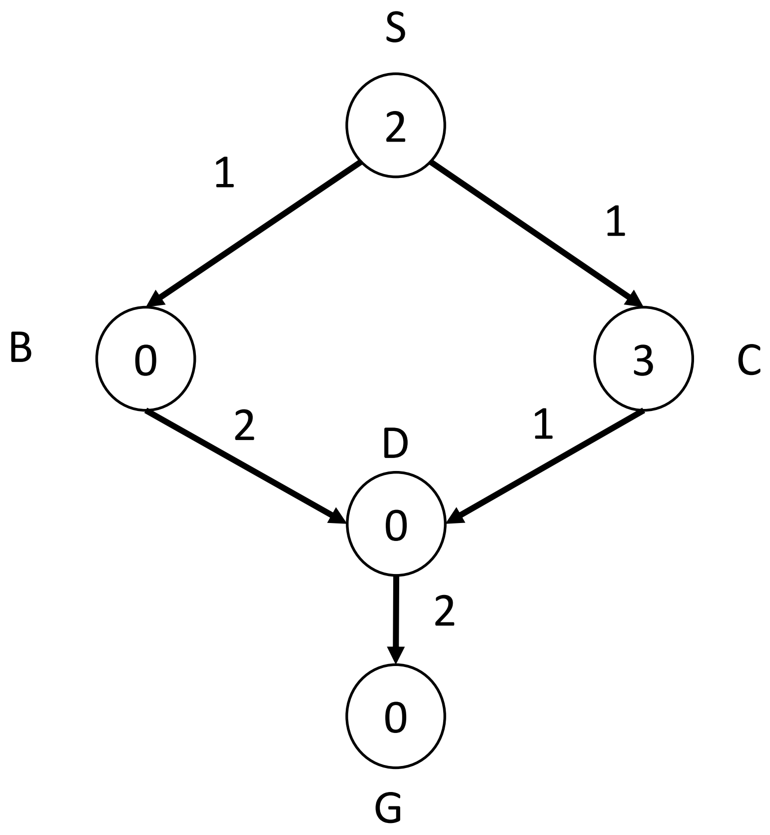
CLOSED	OPEN	Expanded
S: 0	S: 2	S
S: 0, B: 1, G: 10	B: 2, G: 10	B
S: 0, B: 1, G: 3	G: 3, G: 10	(G)

A* Search and CLOSED/Reached



- What are some inadmissible heuristic values that would cause A* search to find a suboptimal path?

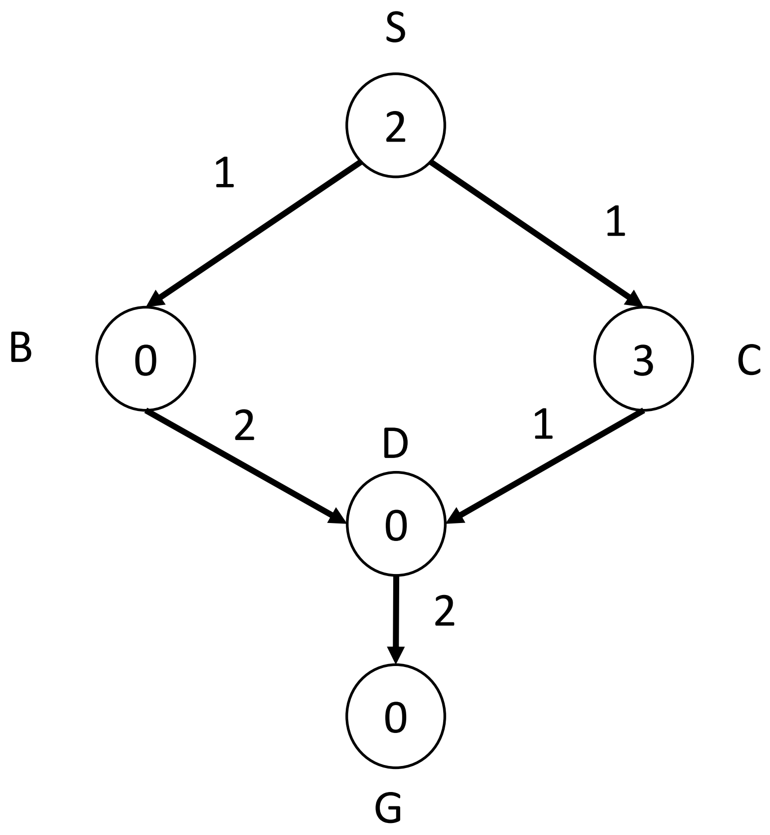
Solve



CLOSED	OPEN	Expanded

Solve

Note how we expanded D with decreasing path costs!



CLOSED	OPEN	Expanded
S: 0	S: 2	S
S: 0, B: 1, C: 1	B: 1, C: 4	B
S: 0, B: 1, C: 1, D: 3	D: 3, C: 4	D
S: 0, B: 1, C: 1, D: 3, G: 5	C: 4, G: 5	C
S: 0, B: 1, C: 1, D: 2, G: 5	D: 2, G: 5	D
S: 0, B: 1, C: 1, D: 2, G: 4	G: 4, G: 5	(G)

Consistent Heuristics

- $h(n) \leq c(n, a, n') + h(n')$
  - Obeys the triangle inequality: “The sum of the length of any two sides must be greater than or equal to the length of the remaining side”
- $h(G) = 0$ for all goal nodes G
- Ensures the cost f along any partial solution is monotonically non-decreasing
- All consistent heuristics are admissible, but not all admissible heuristics are consistent

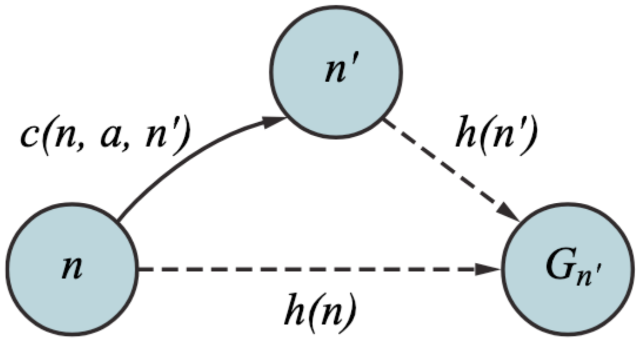


Figure 3.19 Triangle inequality: If the heuristic h is **consistent**, then the single number $h(n)$ will be less than the sum of the cost $c(n, a, a')$ of the action from n to n' plus the heuristic estimate $h(n')$.

https://en.wikipedia.org/wiki/Triangle_inequality

Consistent Heuristics

- A* search with a consistent heuristic is optimally efficient
  - Any algorithm with optimality guarantees using the same heuristic information must expand all nodes expanded by A* search that have a cost less than that of an optimal path
- With a consistent heuristic, the first time we remove a node from OPEN it will be an optimal path from the start state to that node

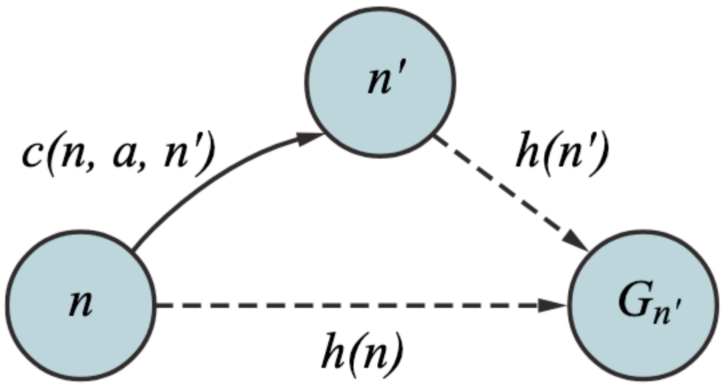


Figure 3.19 Triangle inequality: If the heuristic h is **consistent**, then the single number $h(n)$ will be less than the sum of the cost $c(n, a, a')$ of the action from n to n' plus the heuristic estimate $h(n')$.

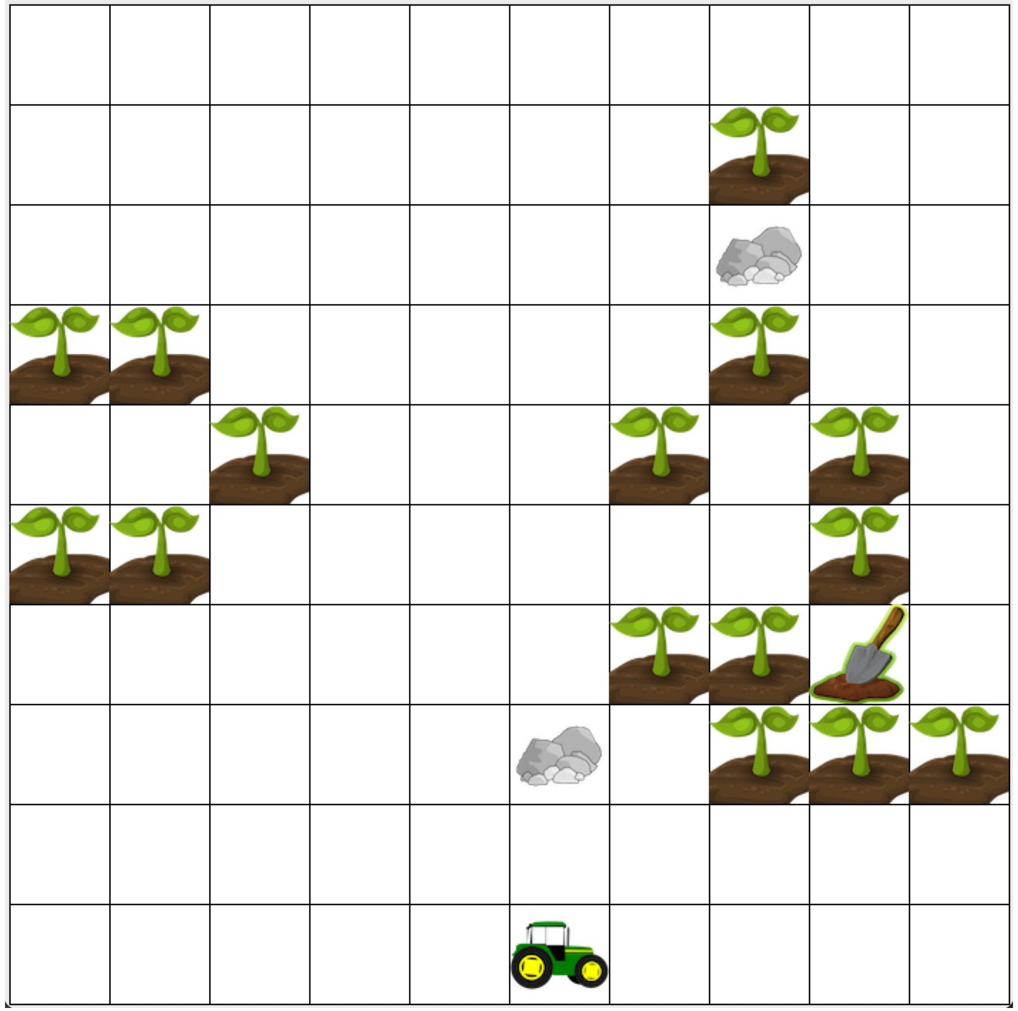
Outline

- Heuristics
- Greedy Best-First search
- A* search
- Weighted A* search

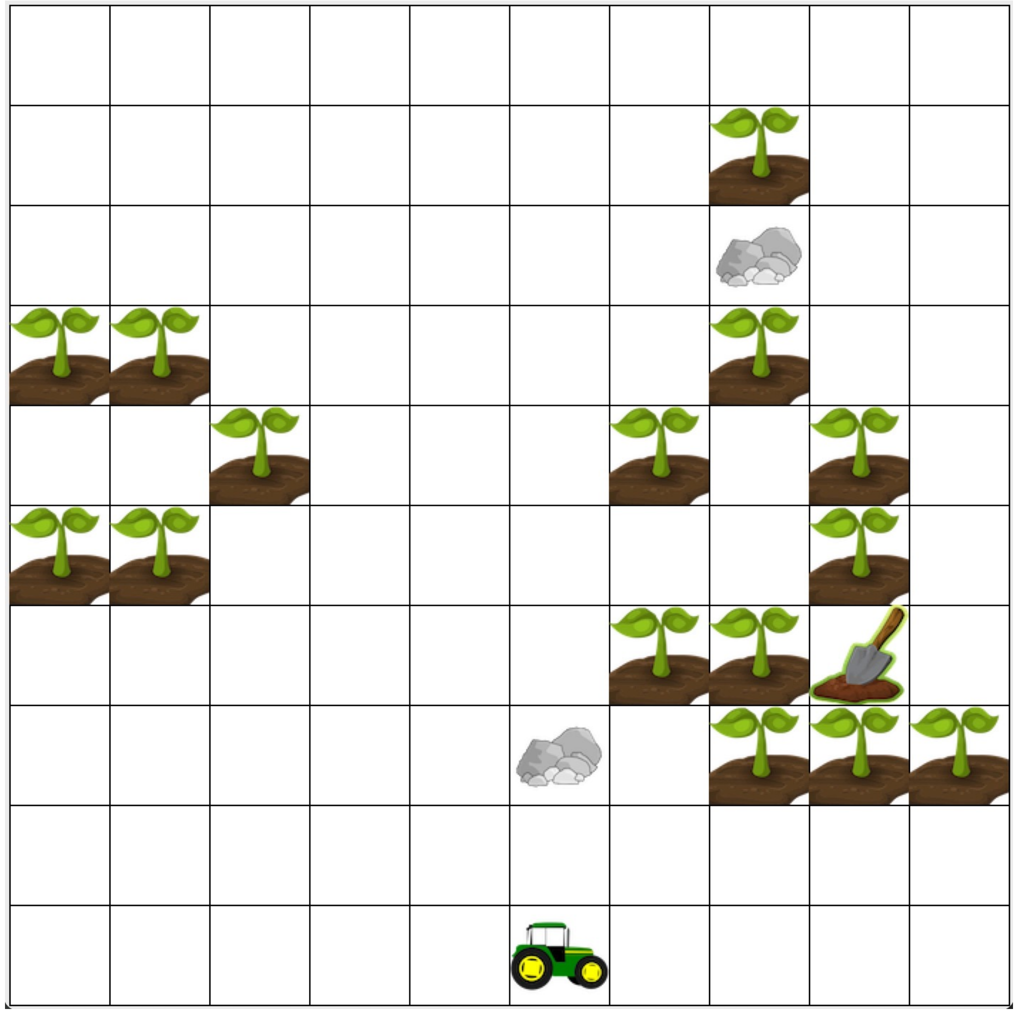
Weighted A* Search

- $f(x) = \lambda_g g(x) + h(x)$
  - $0 \leq \lambda < 1$
- $f(x) = g(x) + \lambda_h h(x)$
  - $1 < \lambda \leq \infty$
  - Bounds on suboptimal paths if h is admissible
- $f(x) = \lambda_g g(x) + \lambda_h h(x)$
  - Unifies uniform cost search, greedy best-first search, and A* search
- Can be potentially faster with less memory usage while potentially incurring a higher path cost
  - Though not always the case

Weighted A* Search



$$f(x) = \lambda_g g(x) + \lambda_h h(x)$$
$$\lambda_g = 1.0, \lambda_h = 8$$



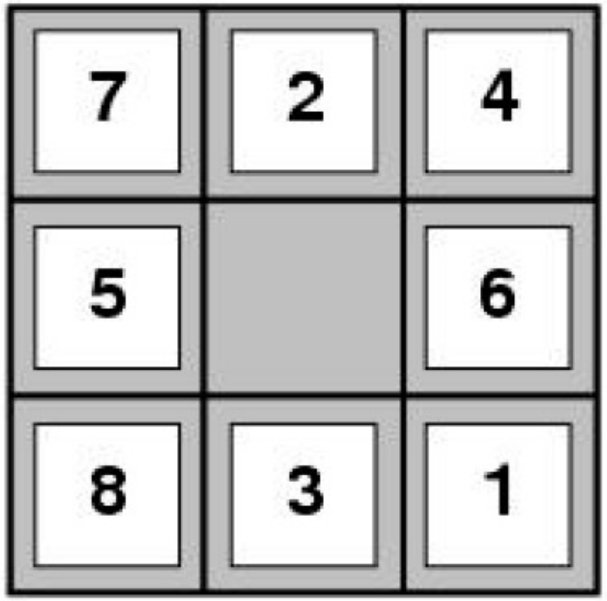
$$f(x) = \lambda_g g(x) + \lambda_h h(x)$$
$$\lambda_g = 1.0, \lambda_h = 16$$

Depression Regions

- Putting more weight on the heuristic function may lead to faster searchers, in practice
- However, there are some cases in which the search will take much longer
- This is due to depression regions
  - Regions in which the heuristic function says you are much closer than you actually are
- The path cost portion helps deprioritize depression regions
  - If the path cost is given less weight in comparison to the heuristic function, the effect depression regions has can be exacerbated

Depression Regions

- While using the misplaced tile heuristic function for the 8-puzzle results in solving the puzzle in a reasonable amount of time, using this same heuristic function for larger versions of the puzzle can lead to significant depression regions
- For example, when all but a few tiles are in the correct place, the heuristic function gives a low value
  - However, there may actually be a significant number of moves left to go
- In this situation, this heuristic does not do well in differentiating between the alternative paths to the goal
  - This results in something close to uniform cost search



Learning Heuristic Functions

- Currently, A* search relies on domain-specific knowledge for constructing the heuristic function
- However, we can use machine learning methods to learn a heuristic function given only a description of the problem

Summary

- We can get better performance using heuristics to estimate the cost to reach the goal
- A* search is guaranteed to find a shortest path if the heuristic is admissible
- We can unify four of the algorithms presented here with the cost function $f(x) = \lambda_g g(x) + \lambda_h h(x)$
  - Uniform cost search: $\lambda_g = 1, \lambda_h = 0$
  - Greedy best-first search: $\lambda_g = 0, \lambda_h = 1$
  - A* search: $\lambda_g = 1, \lambda_h = 1$
  - Weighted A* search: $\lambda_g = 1, \lambda_h > 1$ or $\lambda_g < 1, \lambda_h = 1$

Next Time

- Adversarial Search