Lecture 17: The “animal kingdom” of heuristics: Admissible, Consistent, Zero, Relaxed, Dominant

Mark Hasegawa-Johnson, February 2022

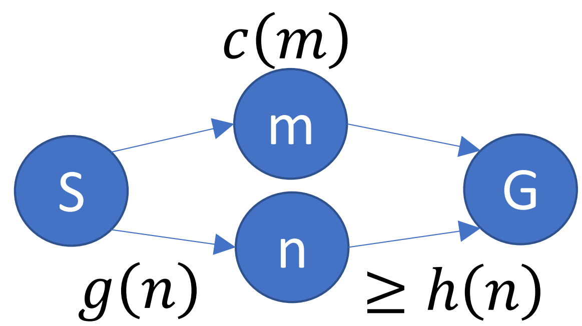
Distributed under CC-BY 4.0

Title image: Peaceable Kingdom by Edward Hicks, National Gallery of Art, Washington, DC

Outline of lecture

1. Admissible heuristics
2. Consistent heuristics
3. The zero heuristic: Uniform Cost Search
4. Relaxed heuristics
5. Dominant heuristics

A* Search

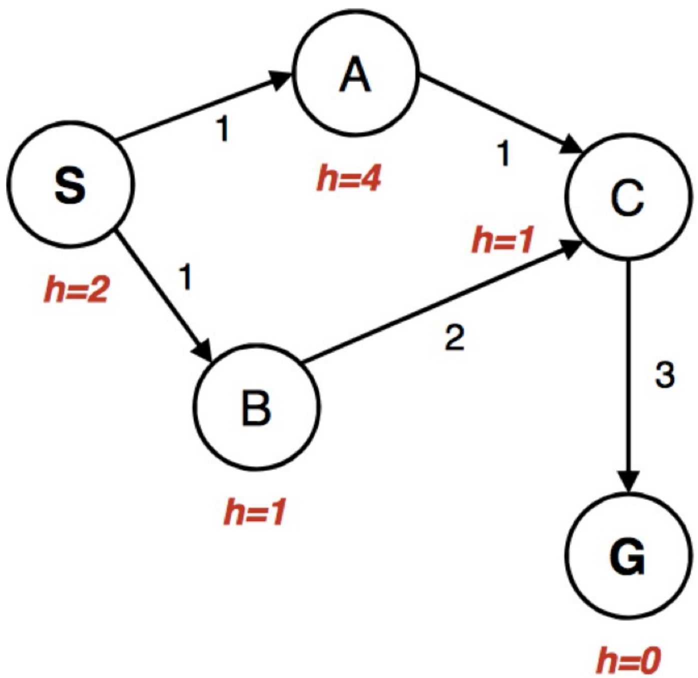


Definition: A* SEARCH

- If $h(n)$ is **admissible** $(d(n) \geq h(n))$, and
- if the frontier is a priority queue sorted according to $g(n) + h(n)$, then
- the FIRST path to goal uncovered by the tree search, path m, is guaranteed to be the SHORTEST path to goal

  ($h(n) + g(n) \geq c(m)$ for every node n that is not on path m)

Bad interaction between A* and the explored set



Explored Set:

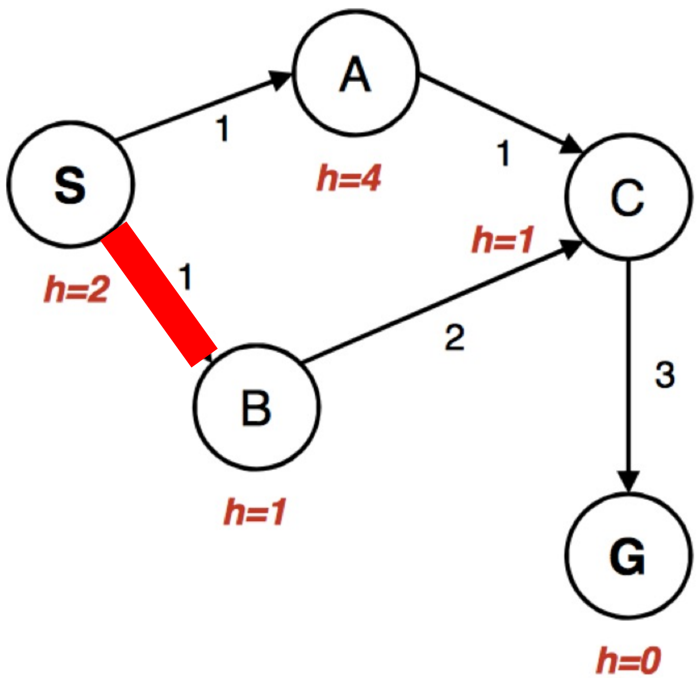
empty

Frontier

S: g(S)+h(S)=2, g(S)=0, parent=none

Expand: S, put its children A and B on the frontier.

Bad interaction between A* and the explored set



Explored Set:

S

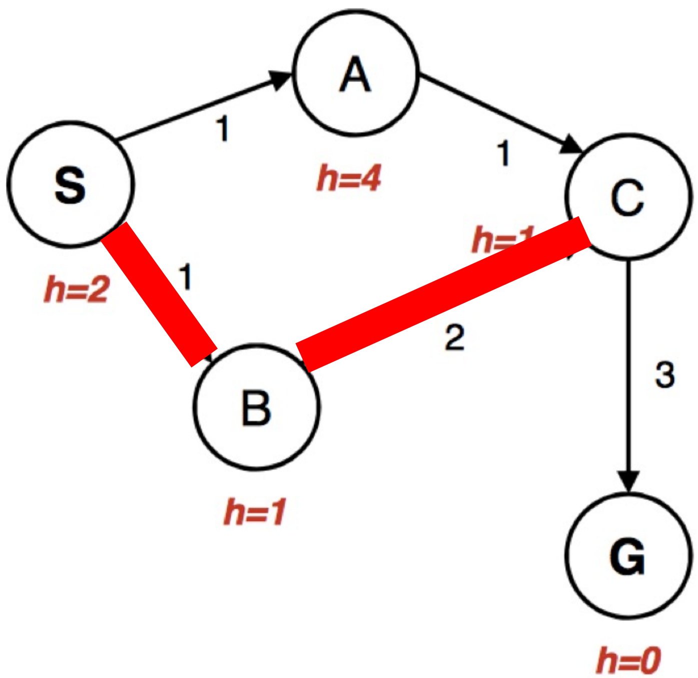
Frontier

A: g(A)+h(A)=5, g(A)=1, parent=S

B: g(B)+h(B)=2, g(B)=1, parent=S

Expand: B, put its child C on the frontier.

Bad interaction between A* and the explored set



Explored Set:

S, B

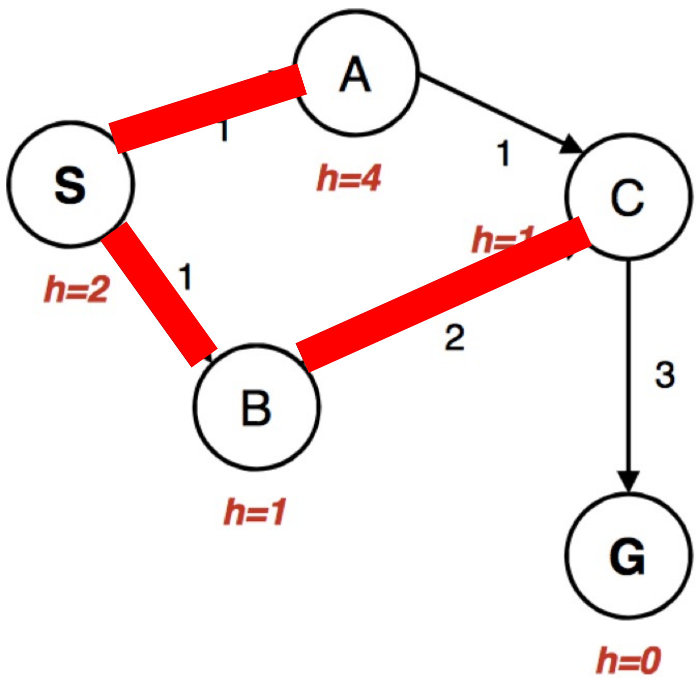
Frontier

A: g(A)+h(A)=5, g(A)=1, parent=S

C: g(C)+h(C)=4, g(C)=3, parent=B

Expand: C, put its child G on the frontier.

Bad interaction between A* and the explored set



Explored Set:

S, B, C

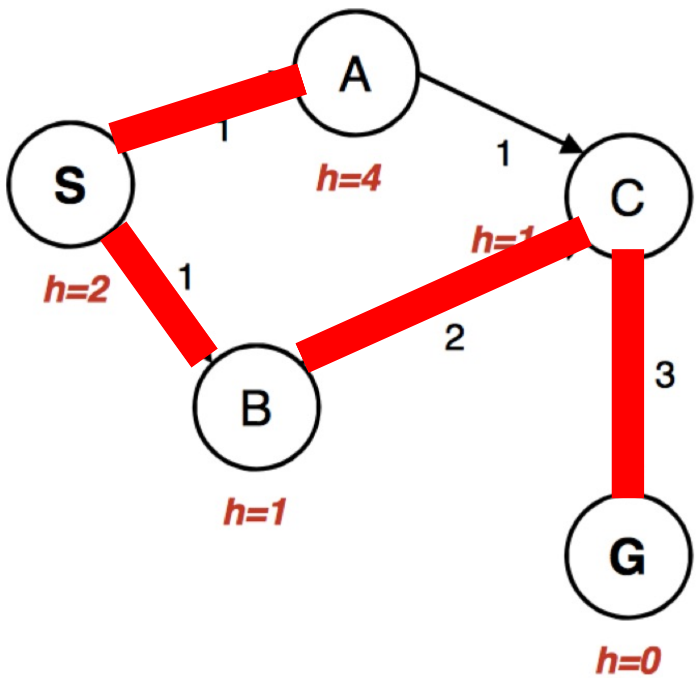
Frontier

A: g(A)+h(A)=5, g(A)=1, parent=S

G: g(G)+h(G)=6, g(G)=6, parent=C

Expand: A. But we can't put its child, C, on the frontier, because C is already in the explored set!

Bad interaction between A* and the explored set



Explored Set:

S, B, C

Frontier

G: g(G)+h(G)=6, g(G)=6, parent=C

Expand: G. Return the path SBCG, with cost 6. OOPS!

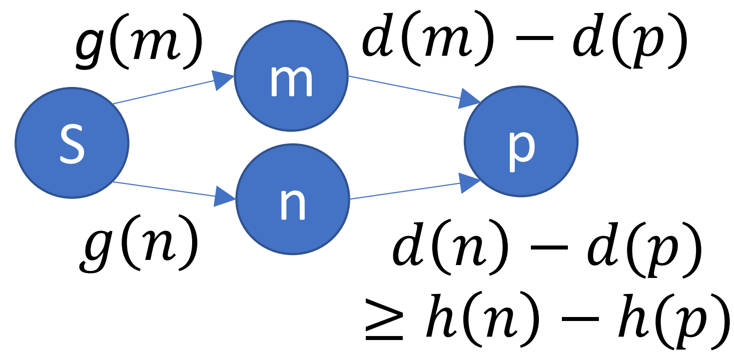
Why did this happen?

- Well, because we used an <u>**explored set**</u> instead of an <u>**explored dict**</u>.
  - <u>**explored dict**</u> lists the h(n)+g(n) for each explored state
  - If the same state shows up later, with lower h(n)+g(n), then put it back on the frontier.
  - An explored set undermines A*, but an explored dict works just fine.
- But actually, why did the higher-cost path <u>**SBC**</u> get explored before the lower-cost path <u>**SAC**</u>?
  - That never happens for goal. An <u>**admissible**</u> heuristic guarantees that the first time you pop Goal from the frontier, it will have its lowest cost.
  - Can we make the same idea true for ***every state***, not just the ***goal state?***

Outline of lecture

1. Admissible heuristics
2. Consistent heuristics
3. The zero heuristic: Uniform Cost Search
4. Relaxed heuristics
5. Dominant heuristics

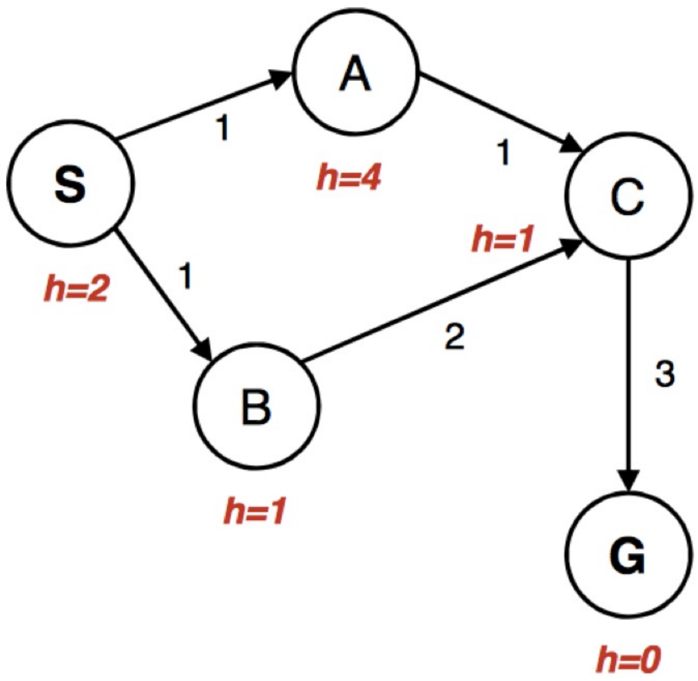
Consistent (monotonic) heuristic



<u>Definition:</u> A **<u>consistent heuristic</u>** is one for which, for every pair of nodesn and p such that $d(n) \geq d(p)$, $d(n) - d(p) \geq h(n) - h(p)$.

In words: the <u>distance between any pair of nodes</u> is **<u>greater than or equal to</u>** the <u>difference in their heuristics</u>.

A* with an inconsistent heuristic



Inconsistent because:

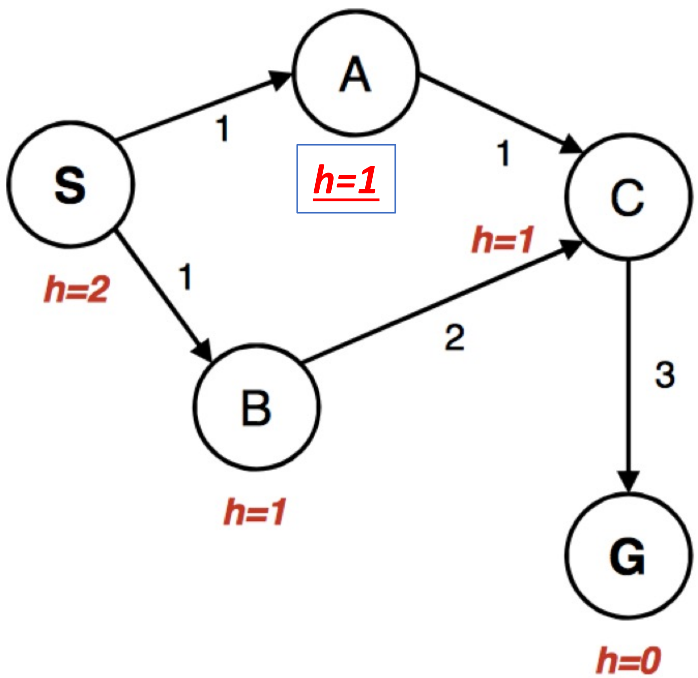
$$d(A) - d(C) = 1$$

but

$$h(A) - h(C) = 3$$

To fix this, we need to reduce $h(A)$ so that $h(A) - h(C) \leq 1$.

A* with a consistent heuristic



Consistent because:

$$d(A) - d(C) = 1$$

and

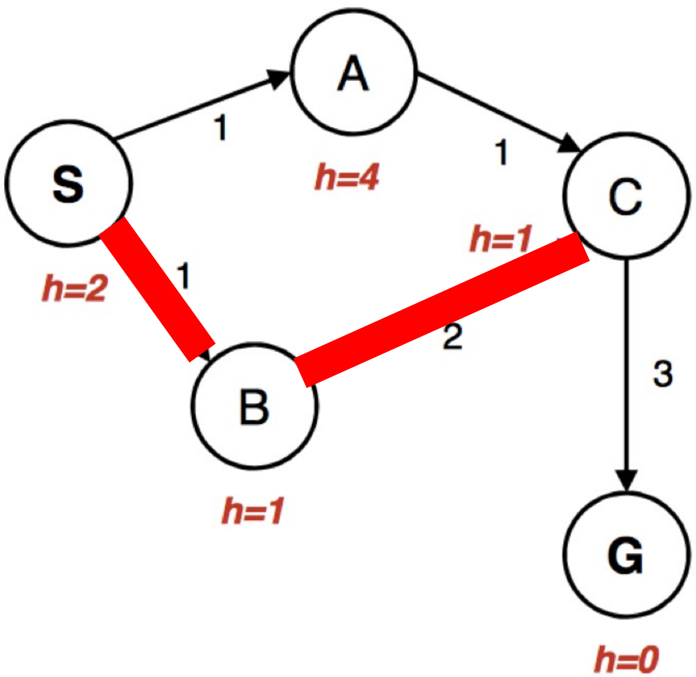
$$h(A) - h(C) = 0$$

Similarly

$$h(n) - h(p) \leq d(n) - d(p)$$

...for every pair of nodes n and p such that $d(n) \geq d(p)$.

A* with an inconsistent heuristic



Explored Set

S, B

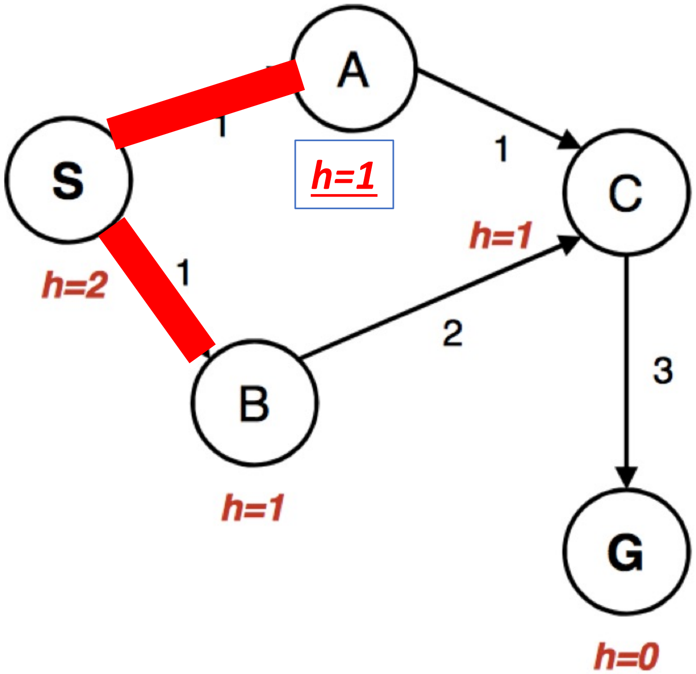
Frontier

A: g(A)+h(A)=5, g(A)=1, parent=S

C: g(C)+h(C)=4, g(C)=3, parent=B

Expand: C

A* with a consistent heuristic



Explored Set

S, B

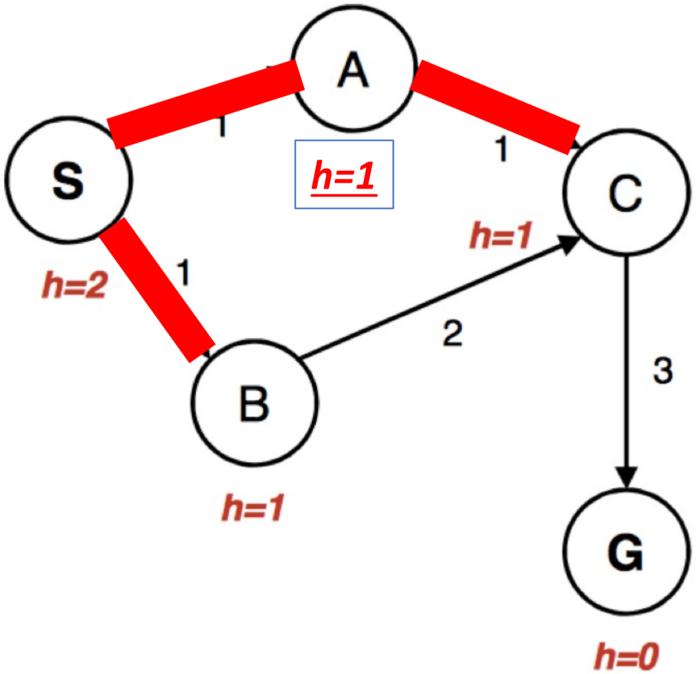
Frontier

A: g(A)+h(A)=**2**, g(A)=1, parent=S

C: g(C)+h(C)=4, g(C)=3, parent=B

Expand: **A**

A* with a **consistent** heuristic



Explored Set

S, B, **A**

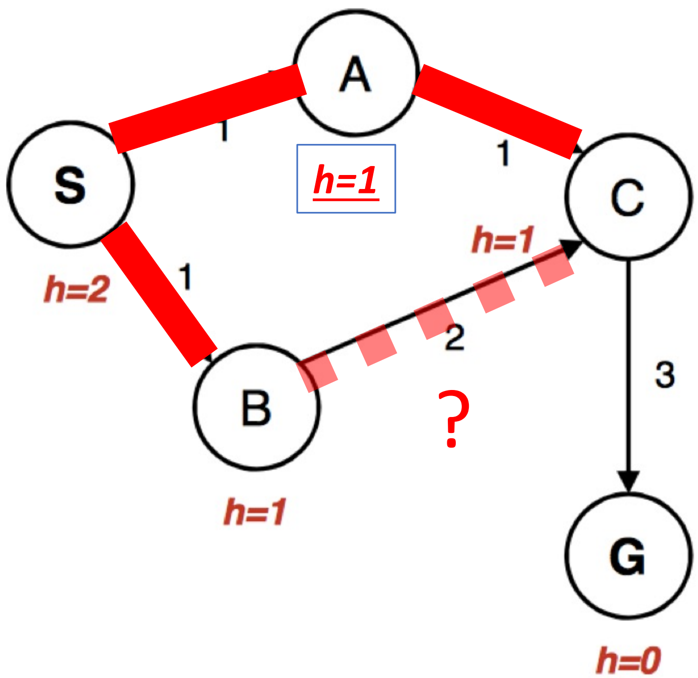
Frontier

C: g(C)+h(C)=4, g(C)=3, parent=B

C: g(C)+h(C)=**3**, g(C)=2, parent=**A**

Expand: the copy of **C** that has A as its parent.

A* with a **consistent** heuristic



Explored Set

S, B, **A,** C

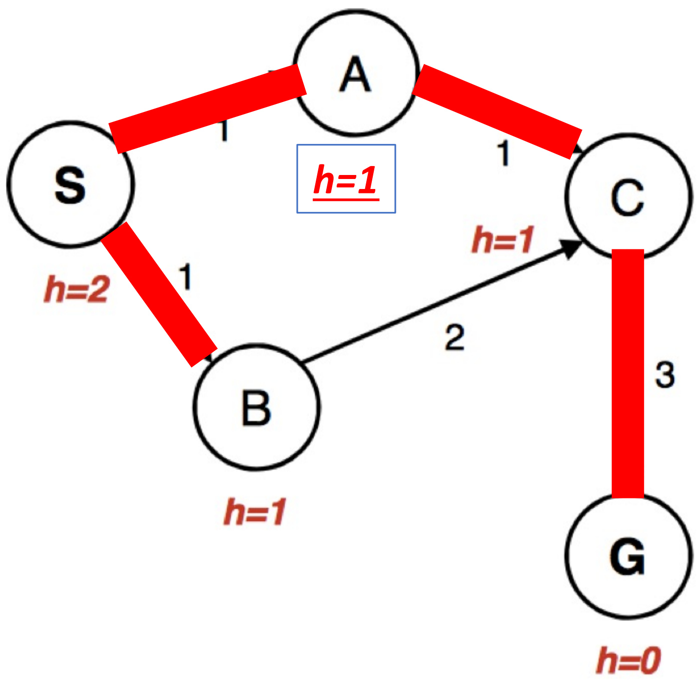
Frontier

C: g(C)+h(C)=4, g(C)=3, parent=B

G: g(G)+h(G)=**5**, parent=C

Expand: The copy of C that has B as its parent. But C is already in the explored set, and since our heuristic is consistent, we know that the new path (with B as parent) has a higher cost than the old path (with A as parent), so we can safely ignore the new path.

A* with a consistent heuristic



Explored Set

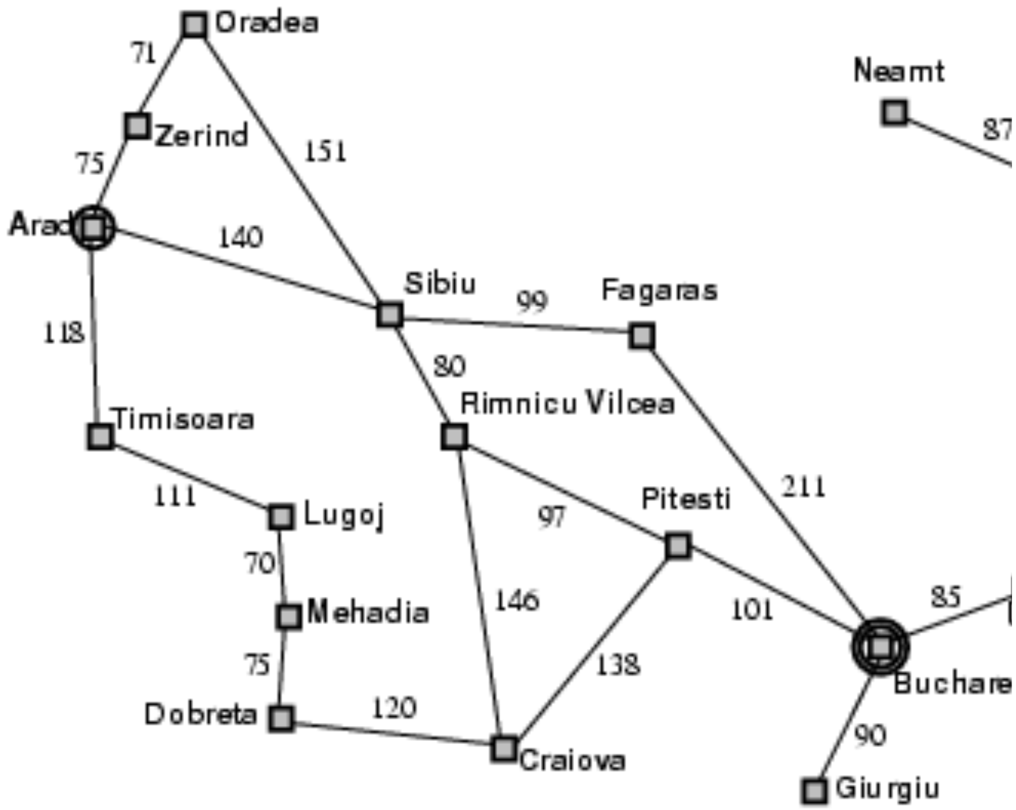
S, B, **A,** C

Frontier

G: g(G)+h(G)=**5**, parent=C

Expand: G

Admissible heuristic example: Romania



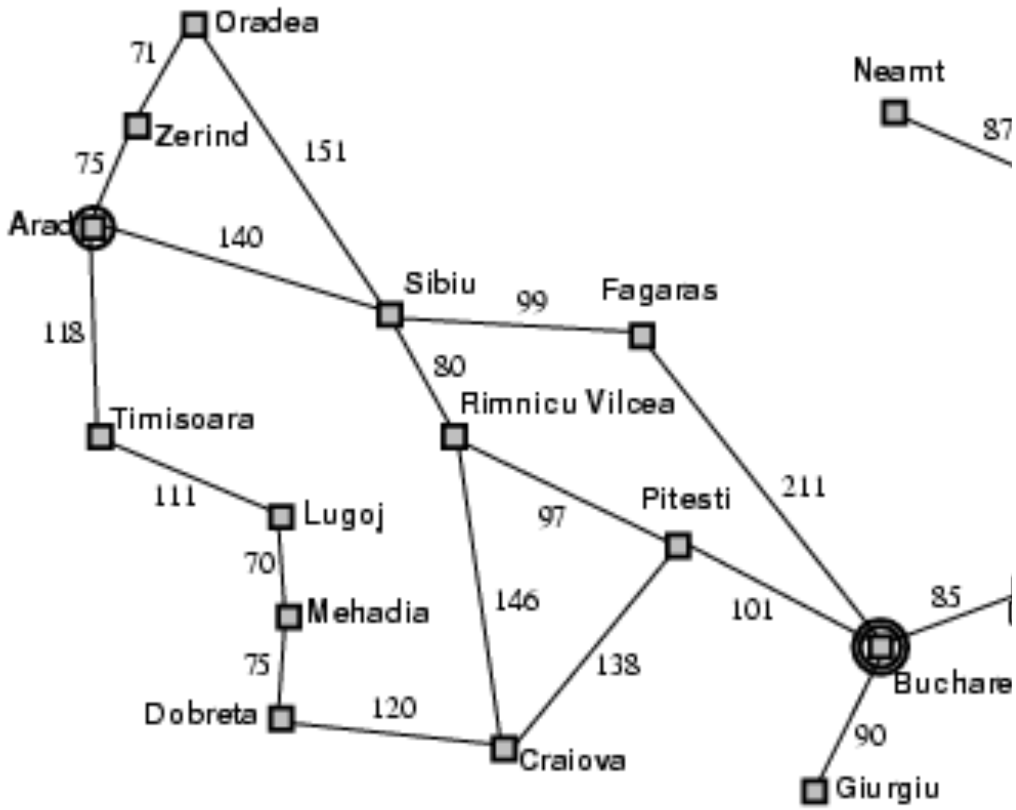
Admissible:

$$h(n) \leq d(n)$$

Example:

$$d(\text{Sibiu}) \ = \ 278$$
$$h(\text{Sibiu}) \ \leq \ 278$$

Consistent heuristic example: Romania



Consistent:

$$h(n) - h(p) \leq d(n) - d(p)$$

Example:

$$d(\text{Arad}) - d(\text{Sibiu}) = 140$$
$$h(\text{Arad}) - h(\text{Sibiu}) \leq 140$$

The heuristic difference is always less than or equal to the cost of the action.

Can you use this in the MP?

- Maybe.
- In the MP, every action has a cost of exactly 1!
- …so a consistent heuristic would be one such that, for every pair of neighboring states n and p, $h(n) - h(p) \leq 1$.
- Manhattan distance satisfies this condition.
- There are good heuristics for parts 3 and 4 that don't satisfy this condition. If your heuristic is not consistent, just make sure that you use an explored dict, instead of an explored set.

Outline of lecture

1. Admissible heuristics
2. Consistent heuristics
3. The zero heuristic: Uniform Cost Search
4. Relaxed heuristics
5. Dominant heuristics

The trivial case: h(n)=0

- A heuristic is **admissible** if and only if $d(n) \geq h(n)$ for every n.
- A heuristic is **consistent** if and only if $d(n, p) \geq h(n) - h(p)$ for every n and p.

- Both criteria are satisfied by $h(n) = 0$.

UCS = A* with h(n)=0

- Suppose we choose $h(n) = 0$
- Then the frontier is a priority queue sorted by
$$g(n) + h(n) = g(n)$$
- In other words, the first node we pull from the queue is the one that's closest to START!! (The one with minimum $g(n)$).
- **Uniform Cost Search** is **A* Search** with the heuristic $h(n) = 0$ for all states.

Outline of lecture

1. Admissible heuristics
2. Consistent heuristics
3. The zero heuristic: Uniform Cost Search
4. Relaxed heuristics
5. Dominant heuristics

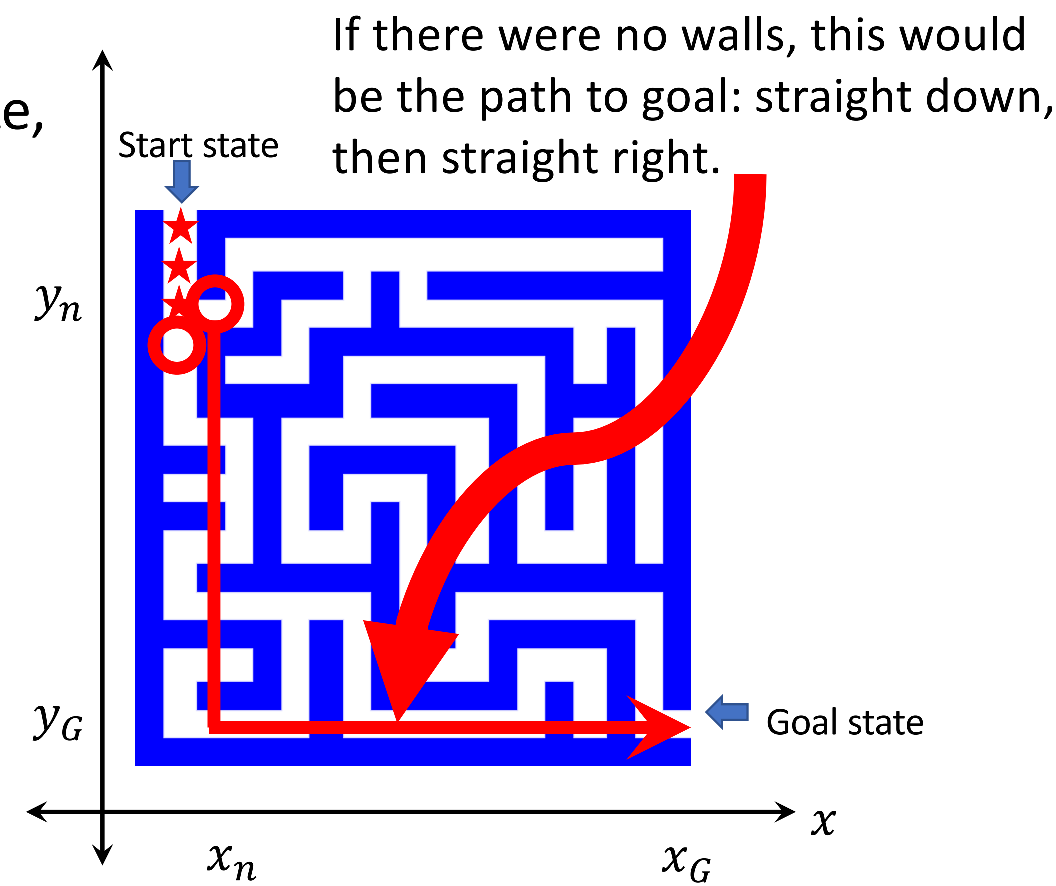
Heuristics from relaxed problems

- A problem with fewer restrictions on the actions is called a relaxed problem
- In most problems, having fewer restrictions on your action means that you can reach the goal faster.
- So designing a heuristic is usually the same as finding a relaxed problem that makes it easy to calculate the distance to goal.

Relaxed heuristic example: Manhattan distance

If there were no walls in the maze, then the number of steps from position (x_n, y_n) to the goal position (x_G, y_G) would be

$$h(n) = |x_n - x_G| + |y_n - y_G|$$

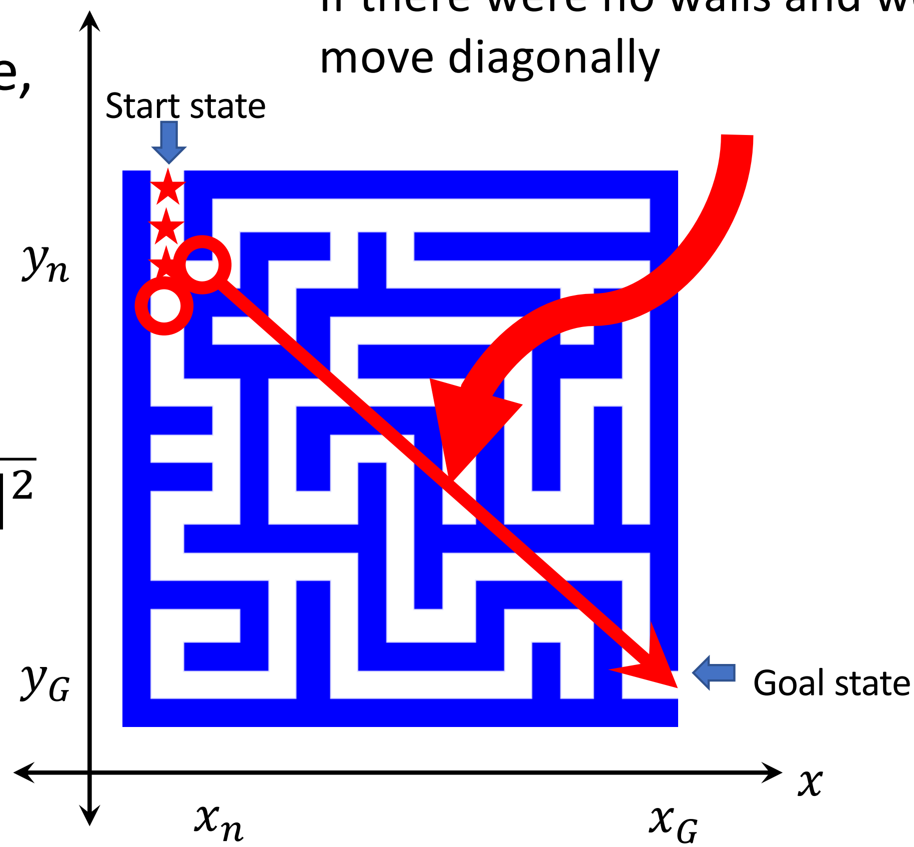


Relaxed heuristic example: Euclidean distance

If there were no walls in the maze, and we could move diagonally, then the number of steps from position (x_n, y_n) to the goal position (x_G, y_G) would be

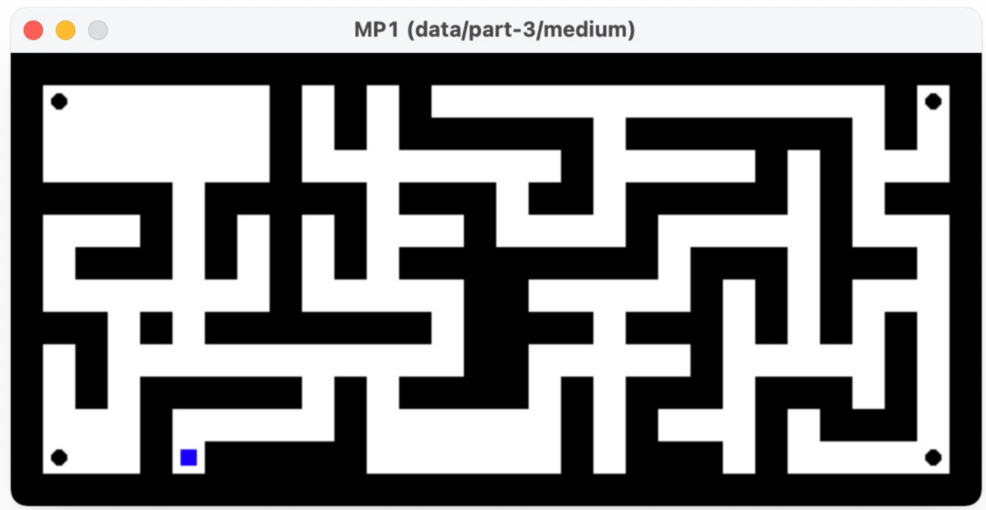
$$h(n) = \sqrt{|x_n - x_G|^2 + |y_n - y_G|^2}$$

If there were no walls and we could move diagonally



Relaxed heuristic example: Corner dots

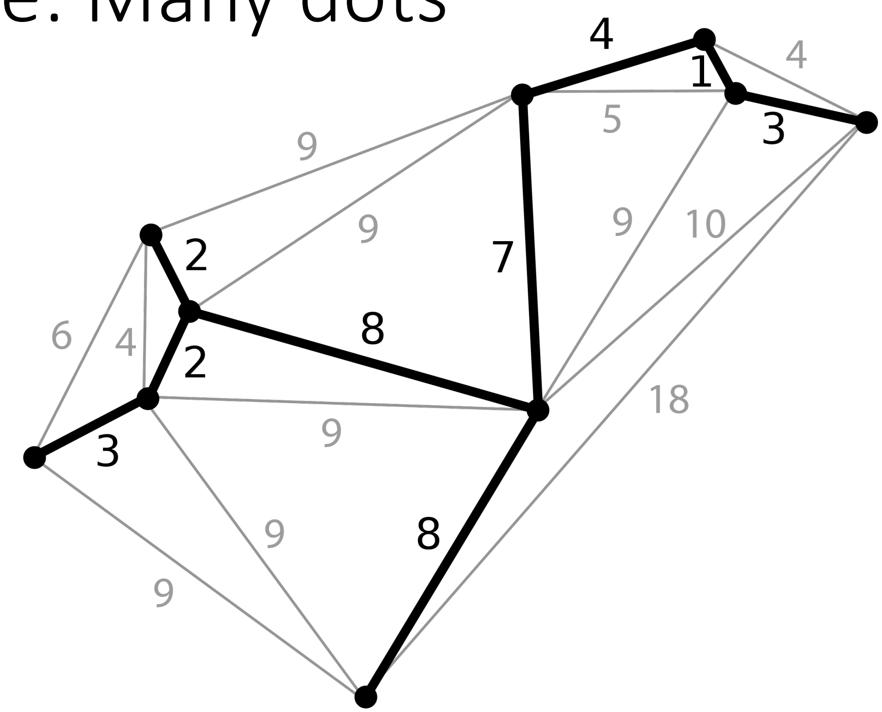
Suppose that, instead of touching ALL of the waypoints, you only had to touch the most extreme waypoints?



Relaxed heuristic example: Many dots

Suppose that, after you reached a waypoint, you could magically fly back to the nearest branch in the minimum spanning tree?

In other words, you only have to go one-way from where you are to the waypoint – you don't have to come back again.



Outline of lecture

1. Admissible heuristics
2. Consistent heuristics
3. The zero heuristic: Uniform Cost Search
4. Relaxed heuristics
5. Dominant heuristics

Which heuristic is better

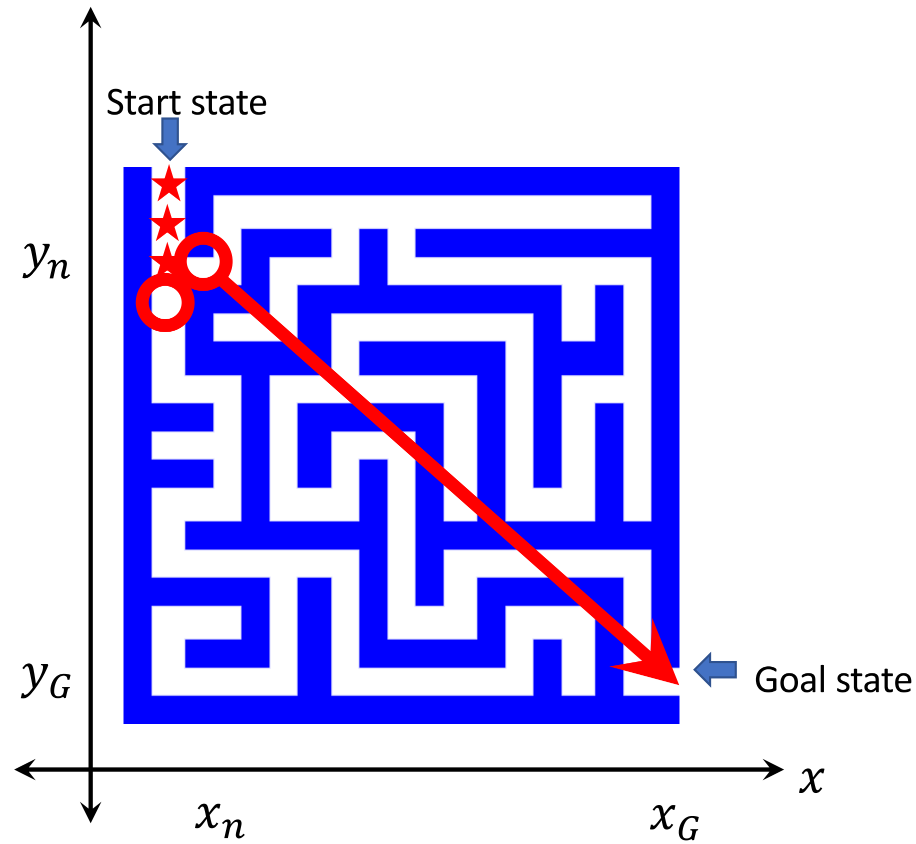
- If Euclidean distance and Manhattan distance are both admissible heuristics for the single-waypoint maze problem, which one is better?

- Computational complexity of A*: If $c(G)$ is true cost of the best path to goal, then A* evaluates every n for which $g(n) + h(n) \leq c(G)$

- How to minimize computational complexity: make $h(n)$ as large as possible, subject to the constraint that $h(n) \leq d(n)$.

Euclidean distance

$$h_2(n) = \sqrt{|x_n - x_G|^2 + |y_n - y_G|^2}$$

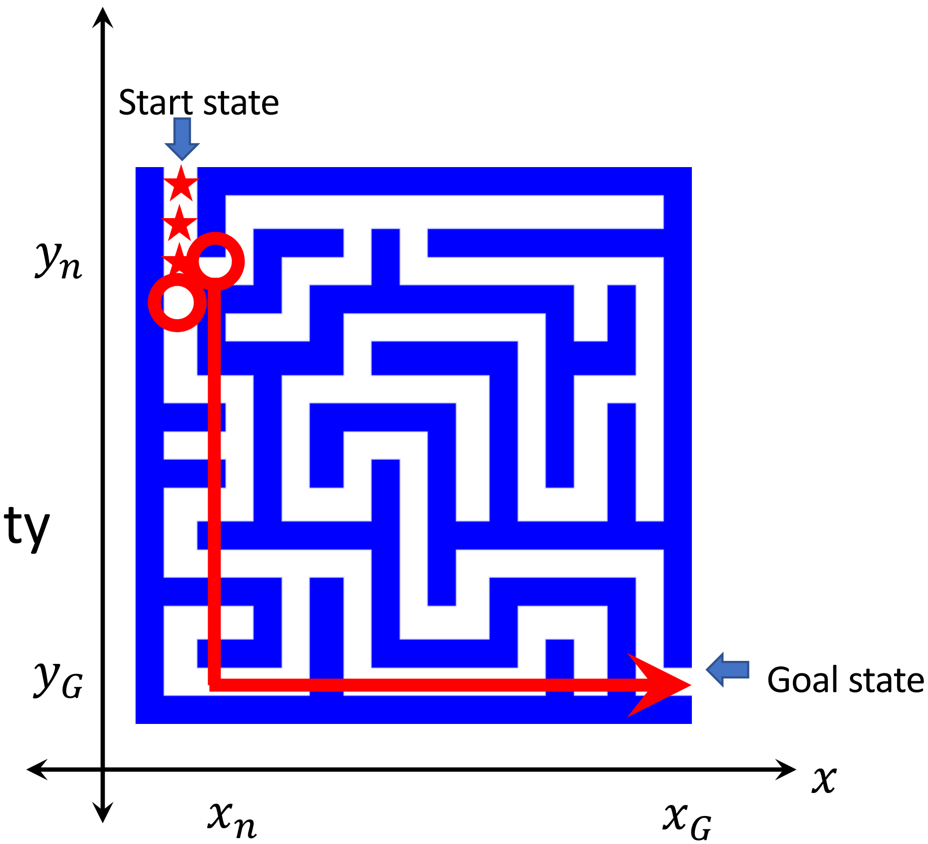


Manhattan distance

$$h_1(n) = |x_n - x_G| + |y_n - y_G|$$

$$h_1(n) \geq h_2(n)$$

Using $h_1(n)$, there will be fewer nodes with $g(n) + h(n) \leq c(G)$. Therefore, computational complexity is lower. Therefore $h_1(n)$ is better.

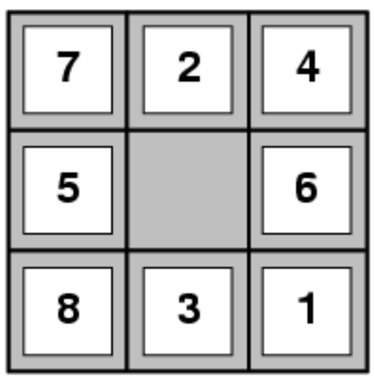


Dominance

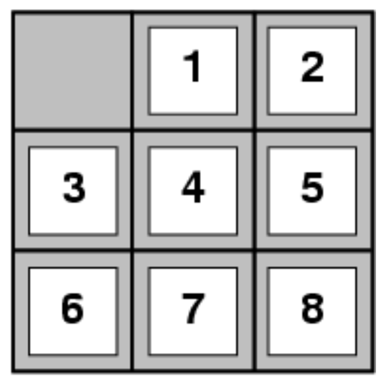
- If $h_2(n) \geq h_1(n)$ for all n, (both admissible) then h_2 **dominates** h_1
- As long as they're both admissible, they will both find the optimum path.
- But $h_2(n)$ will require less computation to find it.

Example: the 8-puzzle

- Problem statement: given a shuffled set of numbers (left), re-arrange them in order (right).
- State: ordering of the numbers and of the space.
- Possible actions: swap the space with any of its neighbors.
- Like traveling salesman, this is an NP-complete problem.



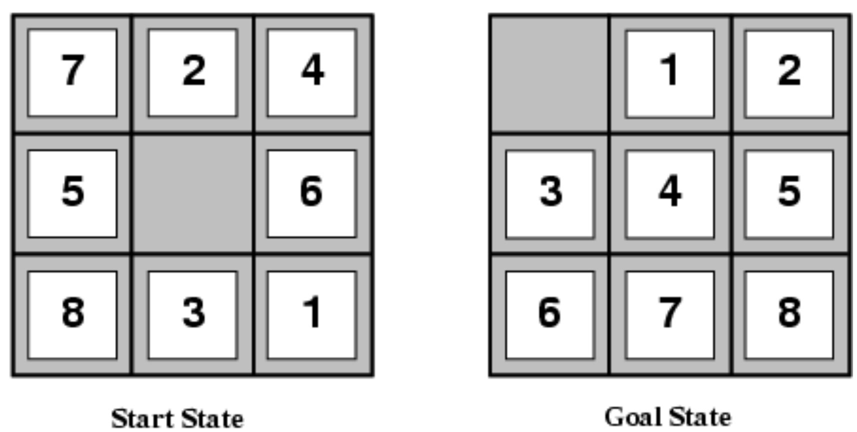
Start State



Goal State

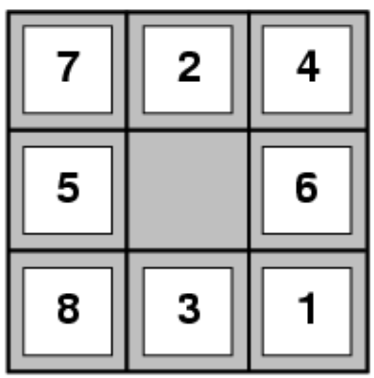
8-puzzle: Heuristic $h_1(n)$

- Suppose that, on each step, we could move any tile, anywhere on the board, regardless of where other tiles were.
- Then $h_1(n) =$ # tiles that need to be moved.
- Example below: $h_1(n) = 8$

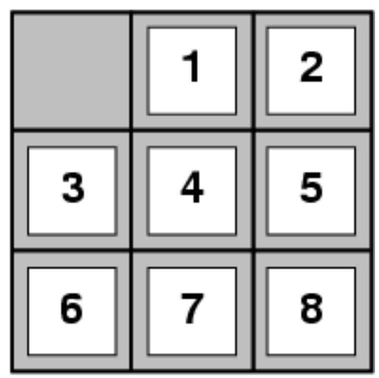


8-puzzle: Heuristic $h_2(n)$

- Suppose that, on each step, we could move any tile by just one step horizontally or vertically, regardless of whether there are other tiles in the way.
- Then $h_2(n) \; = $ sum of Manhattan distances from the current positions of each tile to their target positions (notice: $h_2(n) \geq h_1(n)$)
- Example below: $h_2(n) = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$



Start State



Goal State

Dominance

Experiment results reported by Svetlana Lazebnik

Typical search costs for the 8-puzzle (average number of nodes expanded for different solution depths):

- d=12 BFS expands 3,644,035 nodes
  $A^*(h_1)$ expands 227 nodes
  $A^*(h_2)$ expands 73 nodes

- d=24 BFS expands 54,000,000,000 nodes
  $A^*(h_1)$ expands 39,135 nodes
  $A^*(h_2)$ expands 1,641 nodes

Combining heuristics

- Suppose we have a collection of admissible heuristics $h_1(n)$, $h_2(n)$, …, $h_m(n)$, but none of them dominates the others
- How can we combine them?

$$h(n) = \max\{h_1(n), h_2(n), \ldots, h_m(n)\}$$

Outline of lecture

1. Admissible heuristics: $h(n) \leq d(n)$
2. Consistent heuristics: $h(n) - h(p) \leq d(n) - d(p)$
3. The zero heuristic: Uniform Cost Search: $h(n) = 0$
4. Relaxed heuristics: $h(n)$ is the $d(n)$ from a problem with fewer rules.
5. Dominant heuristics: if $h_2(n) \leq h_1(n)$ and both are admissible, then $h_1(n)$ has lower computational complexity

Five search strategies

Algorithm	Complete?	Optimal?	Time complexity	Space complexity	Implement the Frontier as a…
BFS	Yes	If all step costs equal	$O\{b^d\}$	$O\{b^d\}$	Queue
DFS	No	No	$O\{b^m\}$	$O\{bm\}$	Stack
UCS	Yes	Yes	#nodes s.t. $g(n) \leq c(G)$	#nodes s.t. $g(n) \leq c(G)$	Priority Queue: $g(n)$
Greedy	No	No	$O\{b^m\}$	$O\{b^m\}$	Priority Queue: $h(n)$
A*	Yes	Yes	#nodes s.t. $g(n) + h(n) \leq c(G)$	#nodes s.t. $g(n) + h(n) \leq c(G)$	Priority Queue: $h(n) + g(n)$