Heuristic Search

CPSC 322 Lecture 7

A question to ponder

Suppose you are trying to drive to Edmonton from Vancouver, and you are using search to guide you. You are still in Vancouver, and you have two options, **X** and **Y**, for your next “step” of the journey.

*Where should you go: **X** or **Y**?*

What if you knew that **X** is 3km away, but **Y** is only 1km away?

What if you knew that **X** is an on-ramp for the Trans-Canada Highway, but **Y** is Vancouver City Hall?

Lecture Overview

- **Recap**
  - **Search with Costs**
  - **Summary of Uninformed Search**
- Heuristic Search

Recap: Search with Costs

- Sometimes there are costs associated with arcs.
  - The cost of a path is the sum of the costs of its arcs.

- Optimal solution: not the one that minimizes the ***number of arcs***, but the one that minimizes ***cost***
- Lowest-Cost-First Search: expand paths from the frontier in order of their costs.

Summary of Uninformed Search

	complete?	optimal?	time O( )	space O( )
DFS	No	No	b^m	mb
BFS	Yes	Yes*	b^m	b^m
IDS	Yes	Yes*	b^m	mb
LCFS	Yes**	Yes**	b^m	b^m

* Assuming arc costs are equal

** Assuming arc costs are positive

Recap Uninformed Search

Why are all these strategies called uninformed?

Because they **do not consider any information about the states (end nodes)** to decide which path to expand first on the frontier

In other words, they are **general**; they do not take into account the **specific nature of the problem**.

Lecture Overview

- **Recap**
  - **Search with Costs**
  - **Summary Uninformed Search**
- Heuristic Search

Learning Goals for this class

- **Construct admissible heuristics** for a given problem
- **Verify Heuristic Dominance**
- **Combine admissible heuristics**

Beyond uninformed search….

What information we could use to better select paths from the frontier?

A. an estimate of the distance/cost from the last node on the path to the goal
B. an estimate of the distance/cost from the start state to the goal
C. an estimate of the cost of the path so far
D. None of the above
E. Just Google it

Heuristic Search

Uninformed/blind search algorithms do not take the goal into account until they are at a goal node.

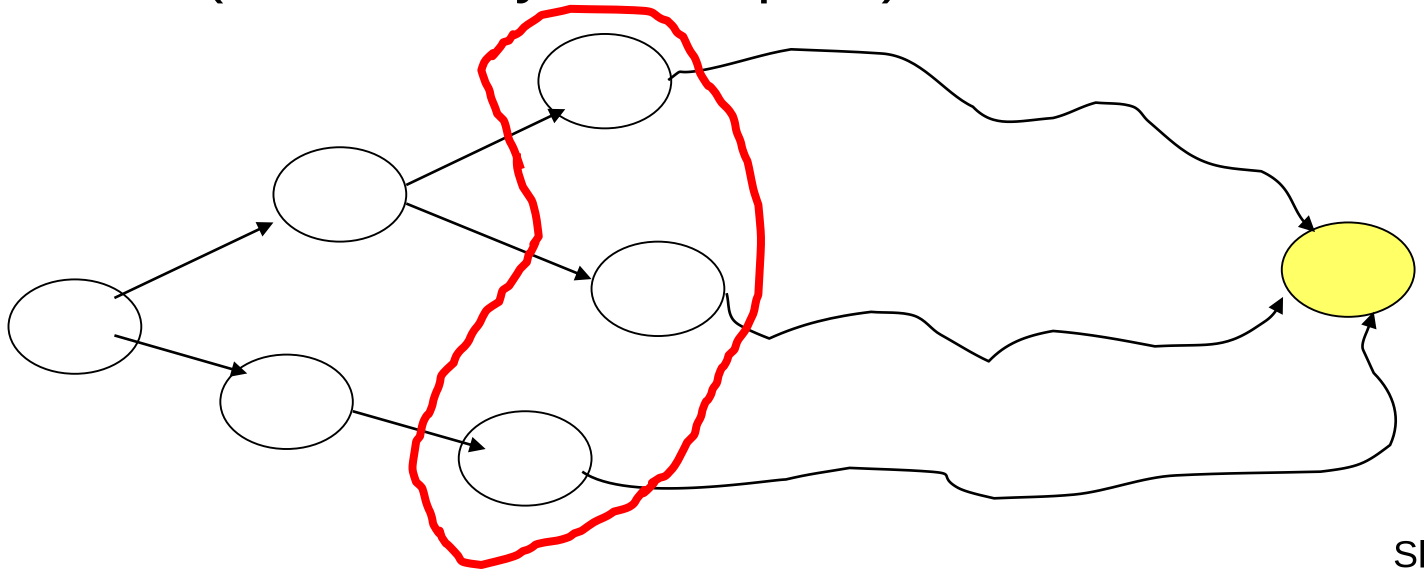
Often there is extra knowledge that can be used to guide the search: **an *estimate* of the distance from node *n* to a goal node.**

This is called a ***heuristic***

More formally

> Definition (**search heuristic**)
>
> A search heuristic *h(n)* is an estimate of the cost of the lowest-cost path from node *n* to a goal node.

- *h* can be extended to paths: $h(\langle n_0,\ldots,n_k \rangle)=h(n_k)$
- For now think of *h(n)* as only using readily obtainable information (that is easy to compute) about a node.



More formally (cont.)

Definition (**admissible heuristic**)

A search heuristic *h(n)* is admissible if it is never an overestimate of the minimum cost from *n* to a goal.

- There is never a path from *n* to a goal that has path cost less than *h(n)*.
- another way of saying this: *h(n)* is a lower bound on the cost of getting from *n* to the nearest goal.

How to Construct an Admissible Heuristic

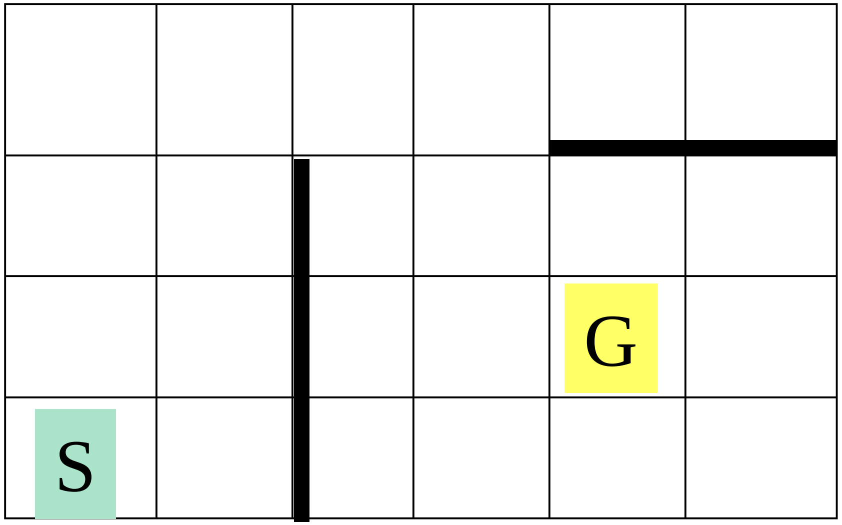
Identify a relaxed version of the problem:

- some constraints on "valid" states have been dropped
- fewer restrictions on the actions

Example Admissible Heuristic Functions

Search problem: robot has to find a route from start location to goal location on a grid (discrete space with obstacles)

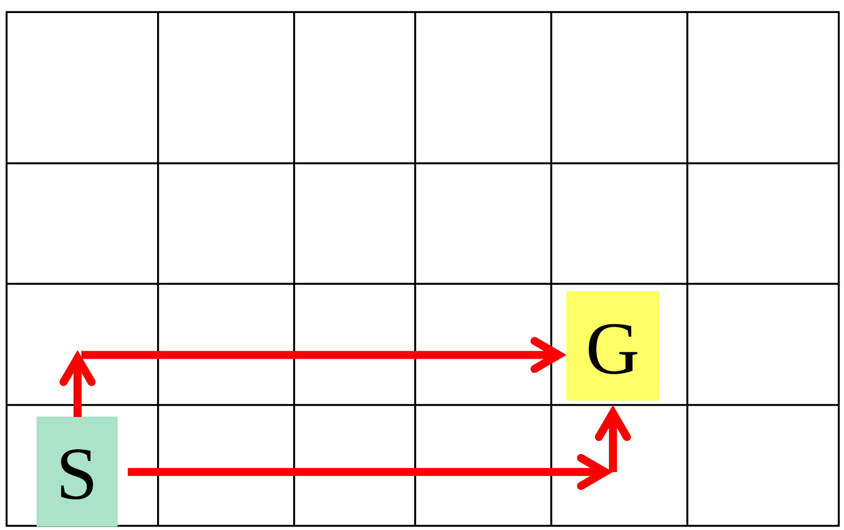
Final cost (quality of the solution) is the number of steps



Example Admissible Heuristic Functions

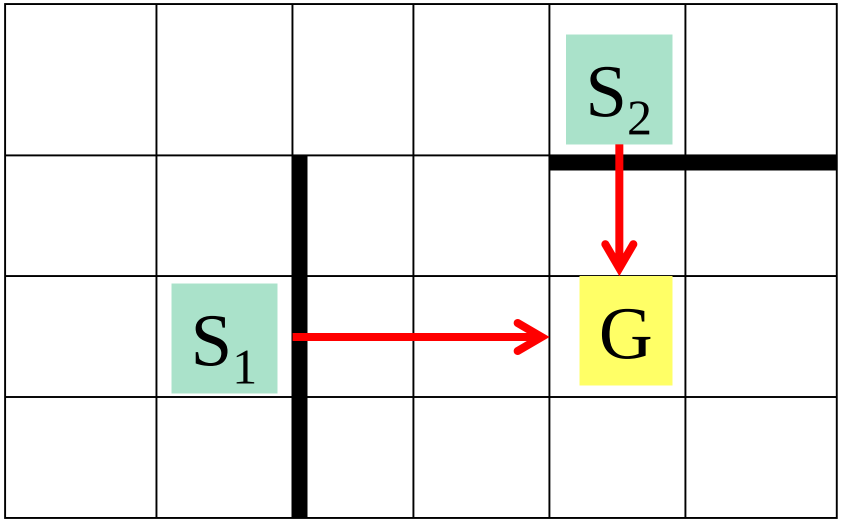
If there are no obstacles, the cost of an optimal solution is…

(this is known as the ***Manhattan distance***)



Example Admissible Heuristic Functions

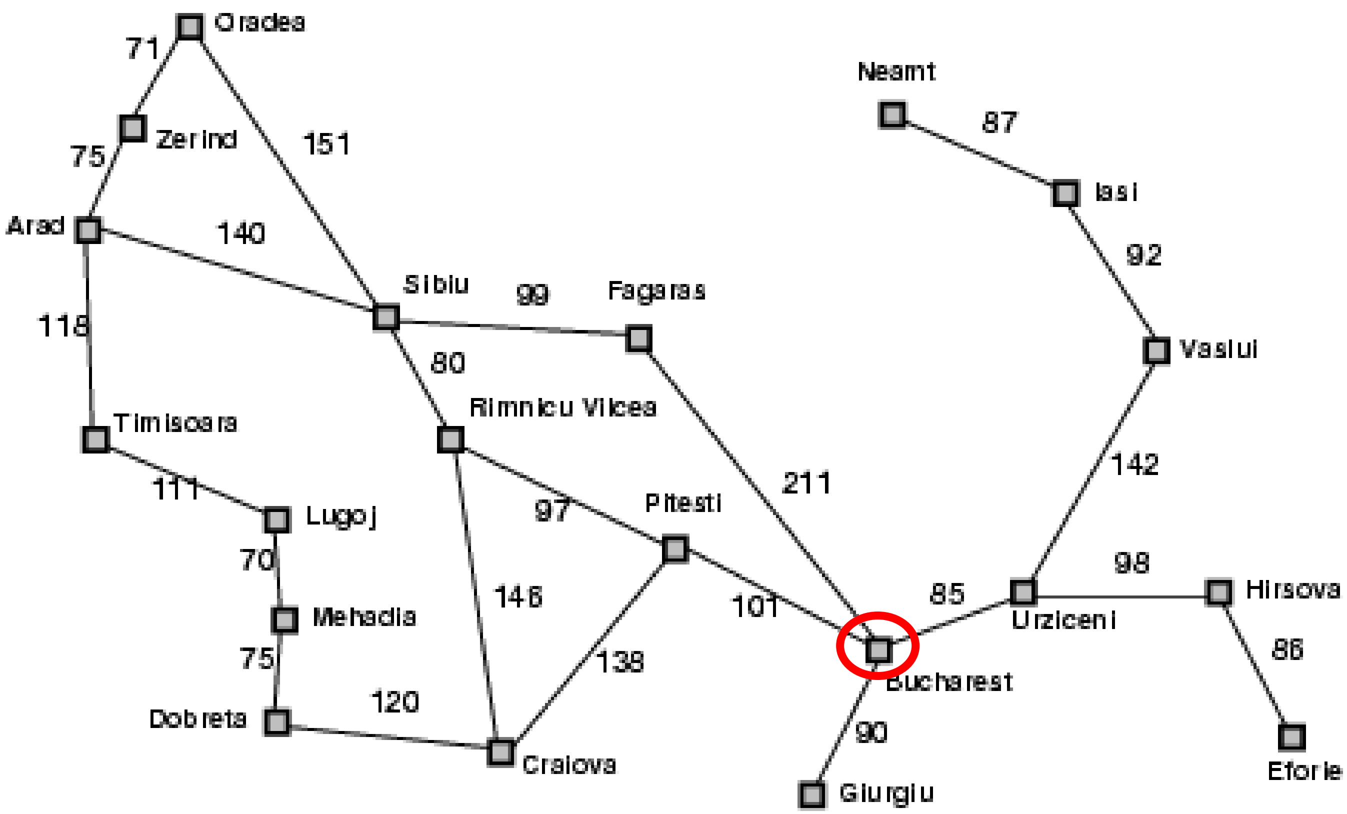
If there are obstacles, the **cost** of the optimal solution ***without*** obstacles is an **admissible heuristic**



$h(S_1)=3$
$h(S_2)=2$

Example Admissible Heuristic Functions

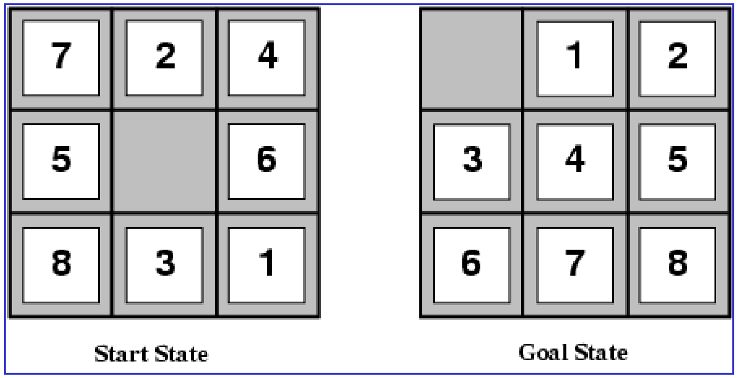
Similarly, If the nodes are points on a Euclidean plane and the cost is the distance, we can use the straight-line distance from *n* to the closest goal as the value of *h(n)*.



Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Admissible Heuristic for 8-puzzle



A reasonable admissible heuristic for this is:

A. Number of misplaced tiles plus number of correctly placed tiles

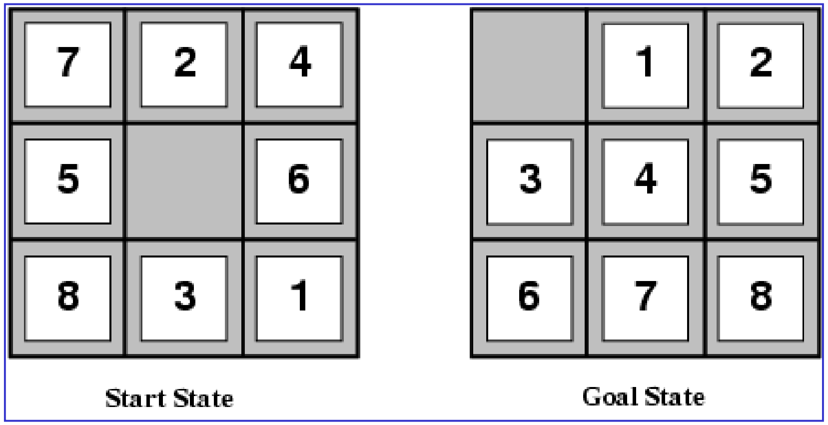
B. Number of misplaced tiles

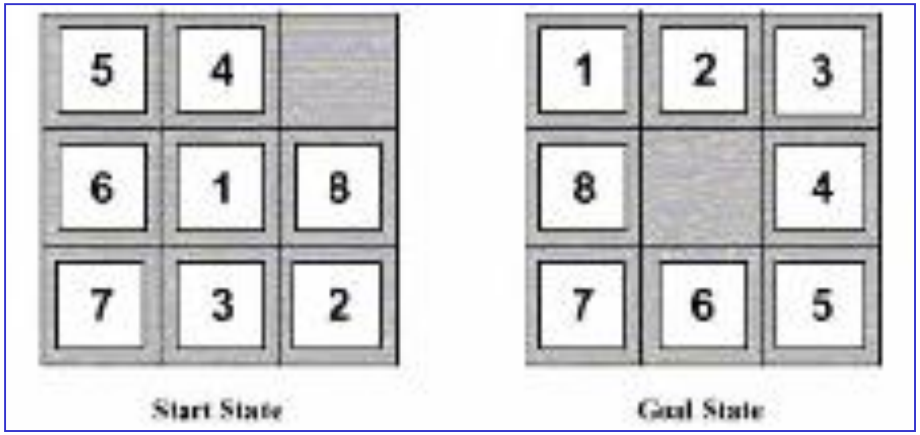
C. Number of correctly placed tiles

D. Number of tiles we haven’t moved yet

E. None of the above

Admissible Example Heuristic Functions

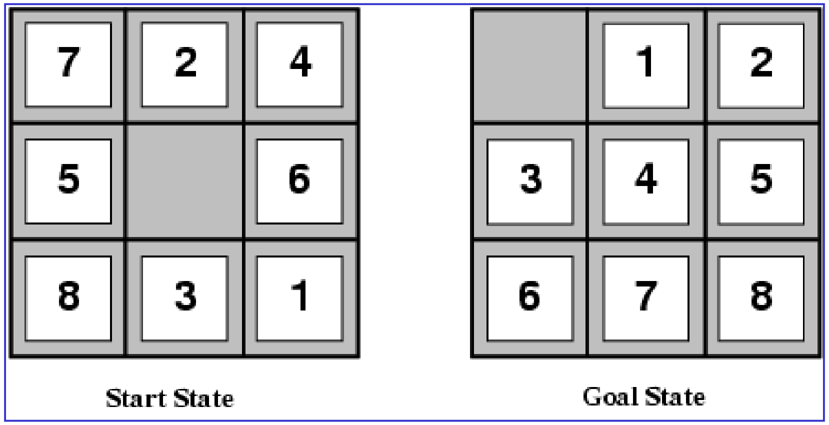


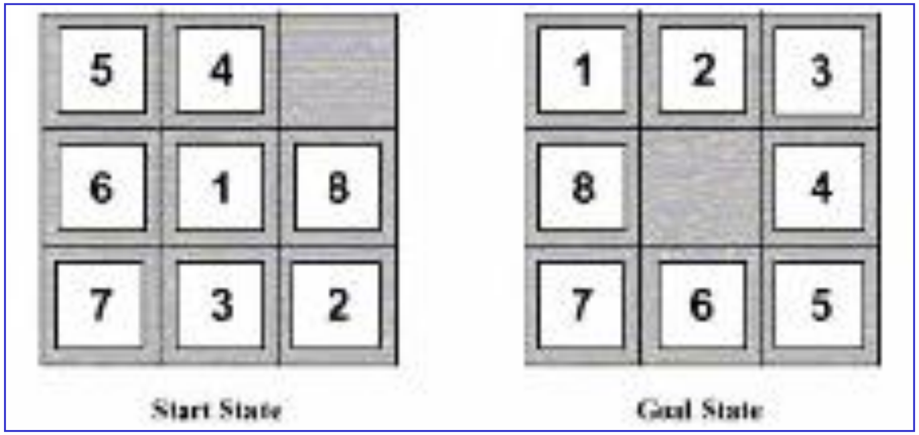


Using the number of misplaced tiles is equivalent to assuming a tile **can go anywhere**, whether or not another tile is already at that location

Example Admissible Heuristic Functions

Another one we can use: the sum of the number of moves between each tile's current position and its position in the solution





This is equivalent to assuming a tile **can go to any adjacent position**, whether or not another tile is already at that location

Which is better?

How to Construct an Admissible Heuristic

You identify a relaxed version of the problem:

- where one or more constraints have been dropped
- problem with fewer restrictions on the actions

Robot: the agent can move through walls

Driver: the agent can move straight towards the destination

8puzzle: (1) tiles can move anywhere

(2) tiles can move to any adjacent square

Result: The cost of an optimal solution in the relaxed problem is an admissible heuristic for the original problem (because it is always weakly less costly to solve a less constrained problem!)

How to Construct an admissible Heuristic

You should identify constraints which, when dropped, make the problem **extremely easy** to solve

- this is important because heuristics are not useful if they're as hard to solve as the original problem!

This was the case in our examples

Robot: ***allow*** movement through walls - Manhattan distance

Driver: ***allow*** “off-road” movement - straight-line distance

8puzzle:

1. tiles **can move anywhere** - number of misplaced tiles
2. tiles can move to **any adjacent square….**

Another approach to construct heuristics

Solution cost for a subproblem

Original Problem

	1	3
8	2	5
7	6	4

Current node

1	2	3
8		4
7	6	5

Goal node

SubProblem

	1	3
@	2	@
@	@	4

1	2	3
@		4
@	@	@

Heuristics: Dominance

If $h_2(n) \geq h_1(n)$ for every state n (both admissible)

then h_2 dominates h_1

Which one is better for search ?

A. h_1

B. h_2

C. it depends

D. 42

E. Too busy playing Fortnite to click

Heuristics: Dominance

$h_2 \geq h_1$ (why?)

8puzzle: (1) tiles can move anywhere

(2) tiles can move to any adjacent square

(Original problem: tiles can move to an adjacent square if it is empty)

search costs for the 8-puzzle (average number of paths expanded):

d=12 IDS = 3,644,035 paths
$A^*(h_1)$ = 227 paths
$A^*(h_2)$ = 73 paths

d=24 IDS = too many paths
$A^*(h_1)$ = 39,135 paths
$A^*(h_2)$ = 1,641 paths

Data are averaged over 100 instances of the 8-puzzle, for various solutions

Combining Admissible Heuristics

Eg. In 8-puzzle, solution cost for the 1,2,3,4 subproblem is substantially more accurate than Manhattan distance **in some cases**

So, **which one should we pick?**

Combining Admissible Heuristics

How to combine heuristics when there is no dominance?

If $h_1(n)$ and $h_2(n)$ are both admissible, then

h(n)= ______ is also admissible **and dominates all its components**

A. h1 + h2
B. |h1 – h2|
C. max(h1, h2)
D. min(h1, h2)
E. avg(h1, h2)

Next Class

- Best-First Search
- Combining LCFS and BestFS: **A*** (finish 3.6)
- A* Optimality