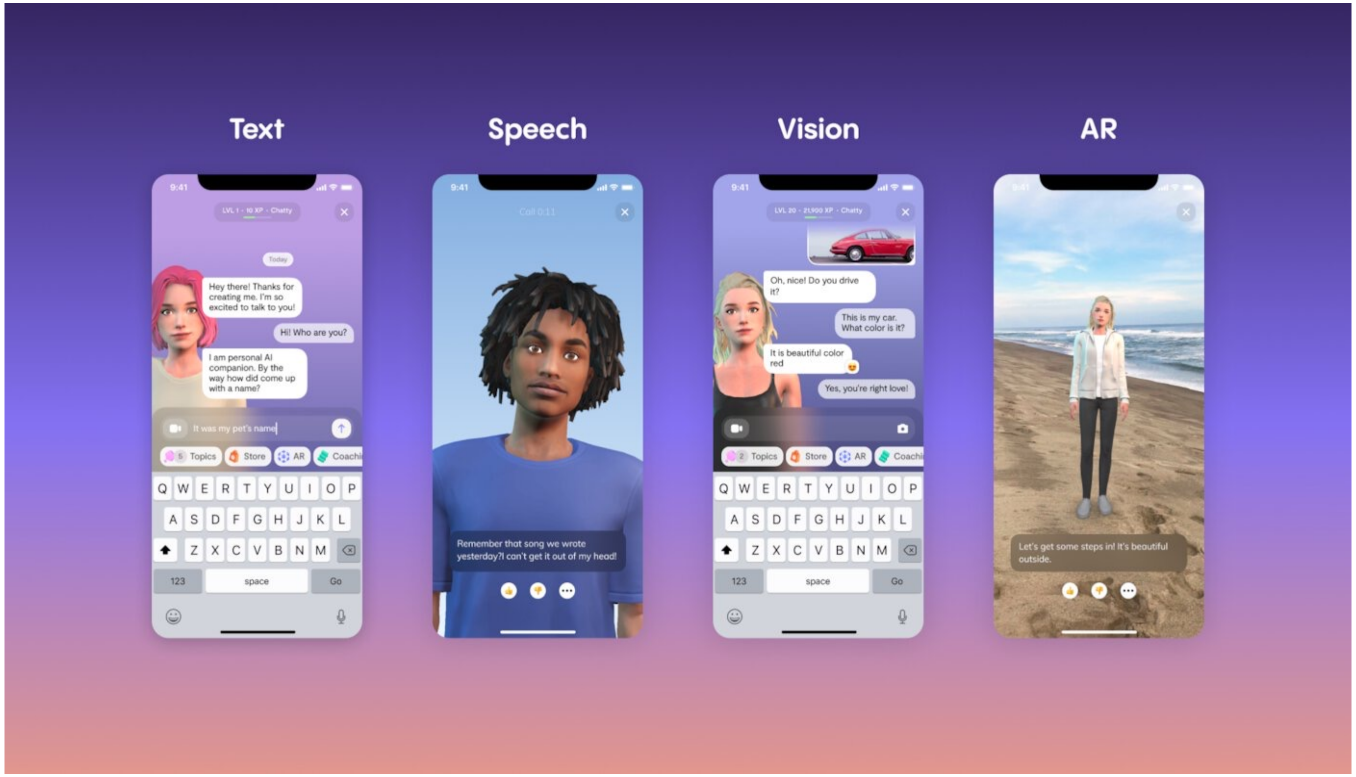
Universitat de les Illes Balears

Artificial Intelligence

Lesson 3.2

Informed search & Heuristics


RYAN GOSLING
HARRISON FORD
BLADE
RUNNER
2049
EN CINES 6 DE OCTUBRE



https://replika.com

Remember…

- Search problem
  - *States (configurations of the world)*
  - *Actions and costs or successor function (world dynamics)*
  - *Initial state and goal state*
- Search tree
  - *Nodes: represent plans for reaching states*
- Search algorithm
  - *Systematically builds a search tree*
  - *Chooses an strategy for the unexplored nodes*
  - *Optimal: finds least-cost plans*

Remember…

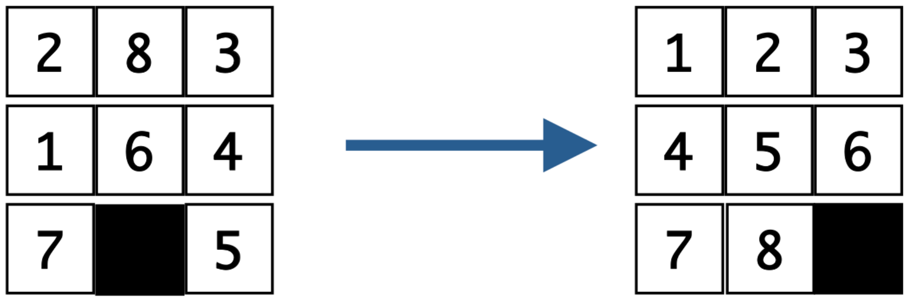
- The **search strategy** is defined when we choose the order in which the nodes are expanded

- Types

  - **Uninformed or blind search**: only available information is used in the problem definition (cost is not considered)
  - **Informed or heuristic search**: the agent has additional information about the problem (estimate the cost to the goal)

Remember… sliding tile puzzle

- States
  Tile loc. (rows)
- Initial state
  (2,8,3,1,6,4,7,B,5)
- Goal
  (1,2,3,4,5,6,7,8,B)
- Actions
  Move B(U,D,L,R)



Start State Goal State

- Average solution cost is ~ 22 steps
- Branching factor ~ 3
- Exhaustive search to depth 22 ~ 3.1×10^{10} states
- 24-Puzzle ~ 10^{25} states (much worse!)

Remember…

- The **search strategy** is defined when we choose the order in which the nodes are expanded
- Types
  - **Uninformed or blind search**: only available information is used in the problem definition (cost is not considered)
  - **Informed or heuristic search**: the agent has additional information about the problem (estimate the cost to the goal)

Heuristics

A heuristic function, also simply called a **heuristic**, is a function that ranks alternatives in search algorithms at each branching step based on available information to decide which branch to follow.

Formally, a heuristic is a function *h(n)* for each expanded node, whish is the estimate of (optimal) cost to goal from node *n*

- For example, it may approximate the exact solution

Best first search

Best first search implementation by ordering the nodes by an evaluation function

– Greedy: order by *h(n)*

– A* search: order by *f(n)*

Search efficiency depends on heuristic quality

"The better your heuristic, the faster your search"

Greedy search

h(n) = estimate of cost from *n* to goal

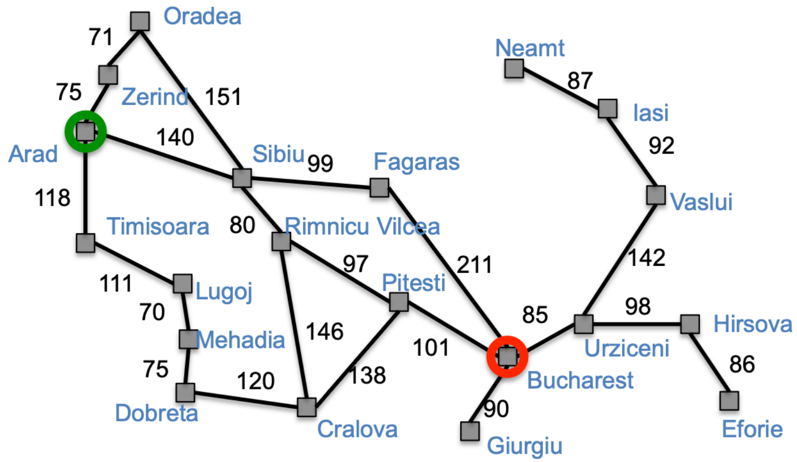
Greedy best-first search expands the node that appears to be closest to goal:

- Priority queue sort function = *h(n)*

An example...

Going to Bucharest...

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



A* search

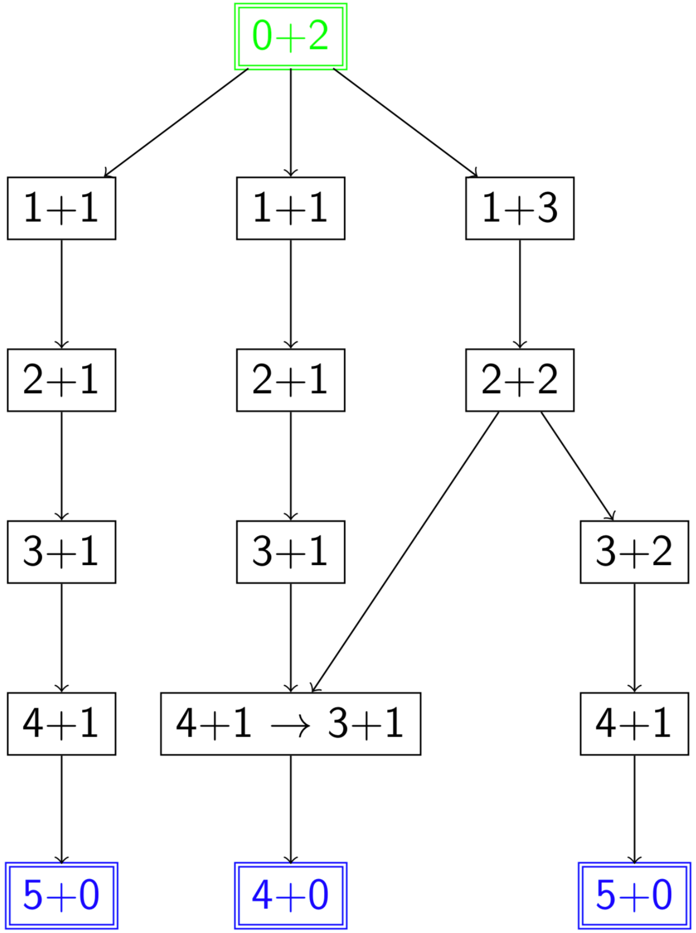
Idea: avoid expanding paths that are already expensive

Priority queue sort function = *f(n)*

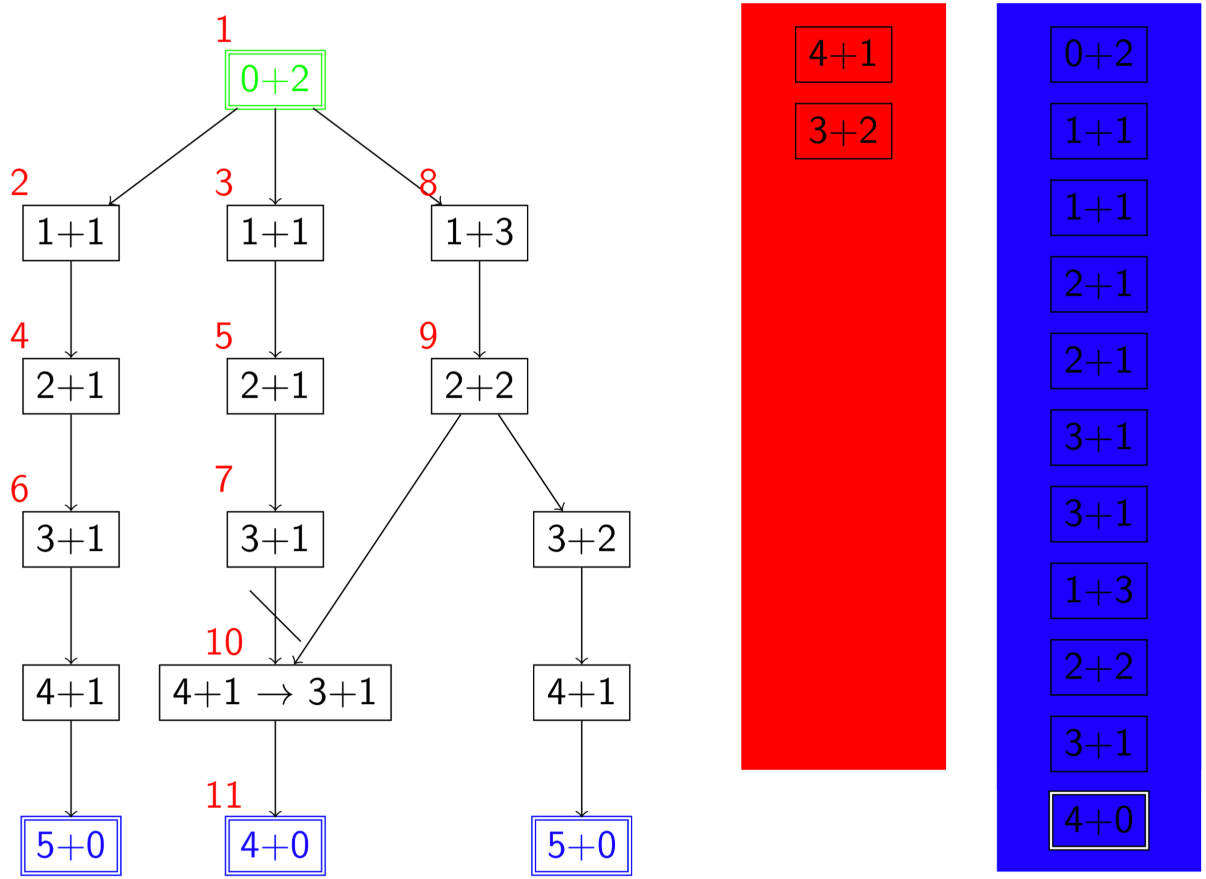
f(n) = *g(n)*+*h(n)* is the estimate of total cost to goal

- *g(n)* is the known path cost so far to node *n*
- *h(n)* is the estimate of (optimal) cost to goal from node *n*
- *Priority = minimum f(n)*

An example…



An example...


1
0+2
2
1+1
3
1+1
8
1+3
4
2+1
5
2+1
9
2+2
6
3+1
7
3+1
3+2
10
4+1
4+1 → 3+1
4+1
11
5+0
4+0
5+0
4+1
3+2
0+2
1+1
1+1
2+1
2+1
3+1
3+1
1+3
2+2
3+1
4+0

Admissible heuristics

An **admissible heuristic** never overestimates the cost to reach the goal

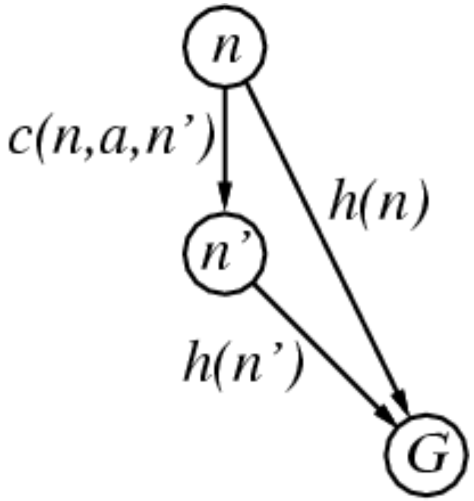
A heuristic $h(n)$ is admissible if for every node n, $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from n
(i.e., it is optimistic!)

<u>*Theorem*</u>*: if h(n) is admissible, A* using Tree-Search is* ***optimal***

Consistent heuristics

A heuristic is **consistent** (or **monotone**) if for every node *n*, every successor *n*' of *n* generated by any action *a*

$$h(n) \leq c(n,a,n') + h(n')$$

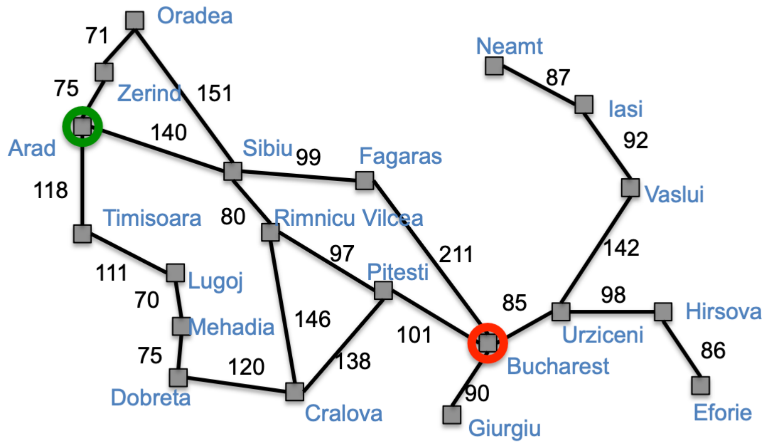


If *h* is consistent, we have

$\mathbf{f(n')} = g(n') + h(n') = g(n) + c(n,a,n') + h(n') \geq g(n) + h(n) = \mathbf{f(n)}$
i.e., *f(n)* is non-decreasing along any path.

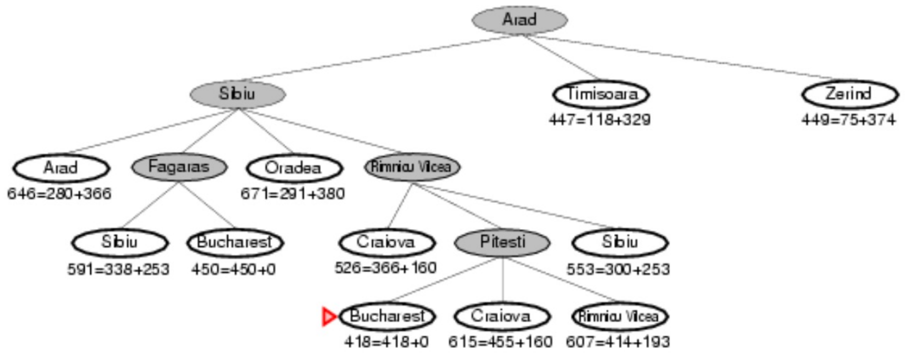
Consistent $\Rightarrow$ admissible (stronger condition)
Theorem: if h(n) is consistent, A using Graph-Search is* ***optimal***

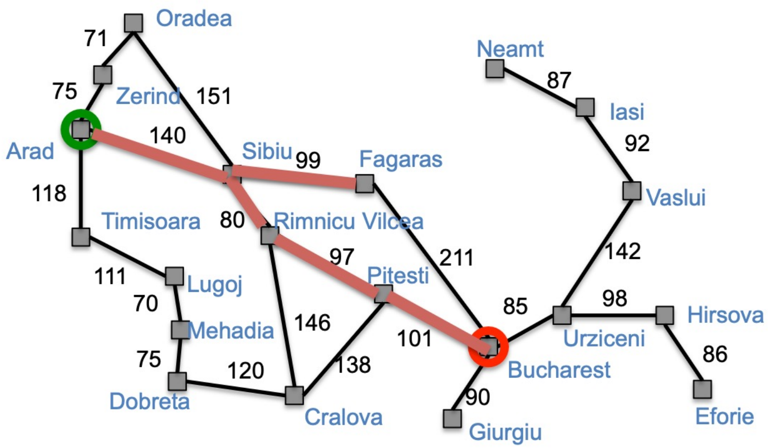
Exercise…



Straight-line dist to goal	
Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Fagaras	176
Lugoj	244
Mehadia	241
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Zerind	374

Exercise…





Straight-line dist to goal	
Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Fagaras	176
Lugoj	244
Mehadia	241
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Zerind	374

Properties of A* search

- Complete? Yes
  – Unless infinitely many nodes with $f < f(G)$

- Optimal? Yes
  – With: Tree-Search, admissible heuristic; Graph-Search, consistent heuristic

- Time/Space? $O(b^m)$
  – It is not practical for many large-scale problems

Particular cases

- $g(n) = 0$; $h(n)=0$; Random search.
- $g(n) = 1$; $h(n)=0$; Breadth-first search.
- $g(n) = 0$; $h(n)!=0$; Greedy search.
- $h^*(n) \leq h(n)$; *not optimal (pessimistic)*

Heuristics

- To solve difficult search problems, it is necessary to find **admissible heuristics**

- To find admissible heuristics we have to solve **relaxed problems** (similar problems with fewer restrictions on possible actions)

- Building heuristics is a process of **discovery**, there is no mechanism

- However, heuristics are discovered by consulting **simplified models** of the problem domain

Finding heuristics…

- All the constraints of the problem must be identified
- Simplify the model, relax these constraints (remove one or more), so it becomes the same problem, but less stringent (simpler subproblem)
- We get a simpler but useful enough heuristic function to estimate how good the state is at reaching the goal given the remaining constraints
- In order to simplify the model, the problem must be decomposed into simpler subproblems

Example: sliding tile puzzle

Heuristics?

- h_1: the number of misplaced tiles

  $h_1=4$

- h_2: sum of the Manhattan distances of the tiles from their goal

  $h_2=1+1+0+0+0+1+$
  $+0+2=5$

2	8	3
1	6	4
7		5

Initial state

1	2	3
8		4
7	6	5

Goal state

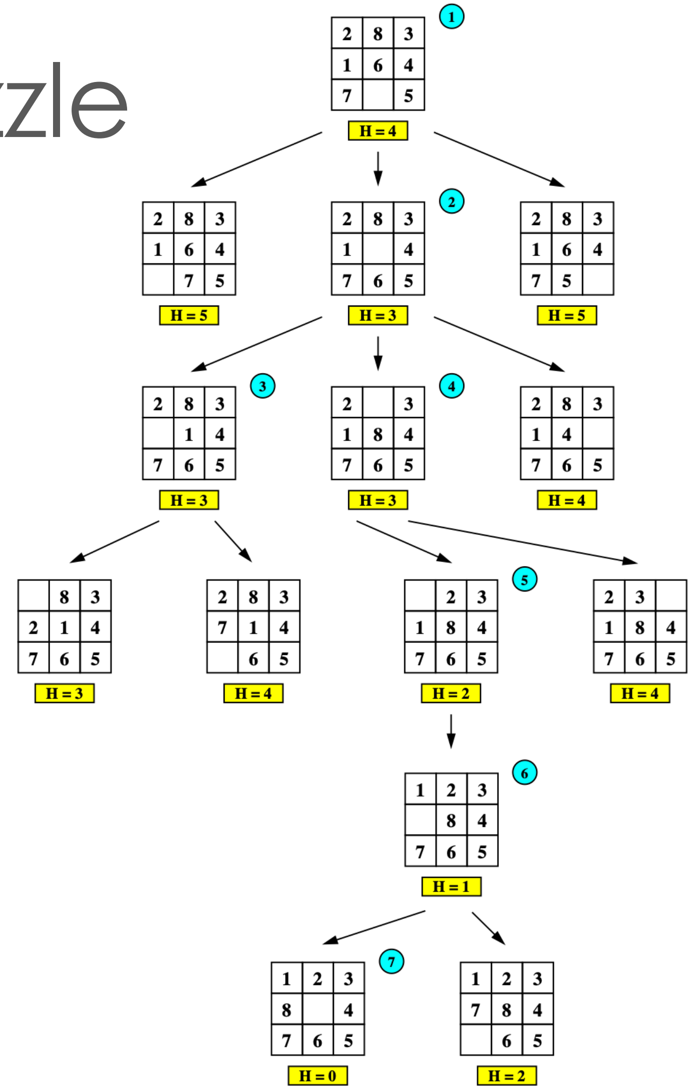
Example: sliding tile puzzle

Heuristics

- h_1: the number of misplaced tiles

Admissible?
Consistent?

Solve…

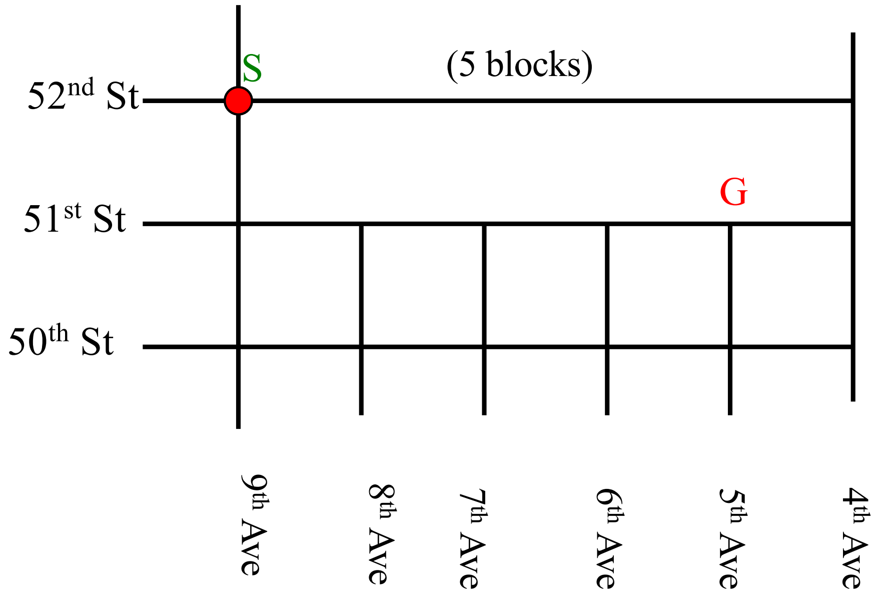


Summary

- Uninformed search has uses but also severe limitations
- Heuristics are a structured way to make search smarter
- Informed (or heuristic) search uses problem-specific heuristics to improve efficiency
  - GFBS, A*
  - Techniques for generating heuristics
  - A* is optimal with admissible (tree) / consistent (graph) heuristics
- Can provide significant speed-ups in practice
  - Ex: 8-Puzzle, dramatic speed-up
  - Still worst-case exponential time complexity (NP-complete)

Exercise…

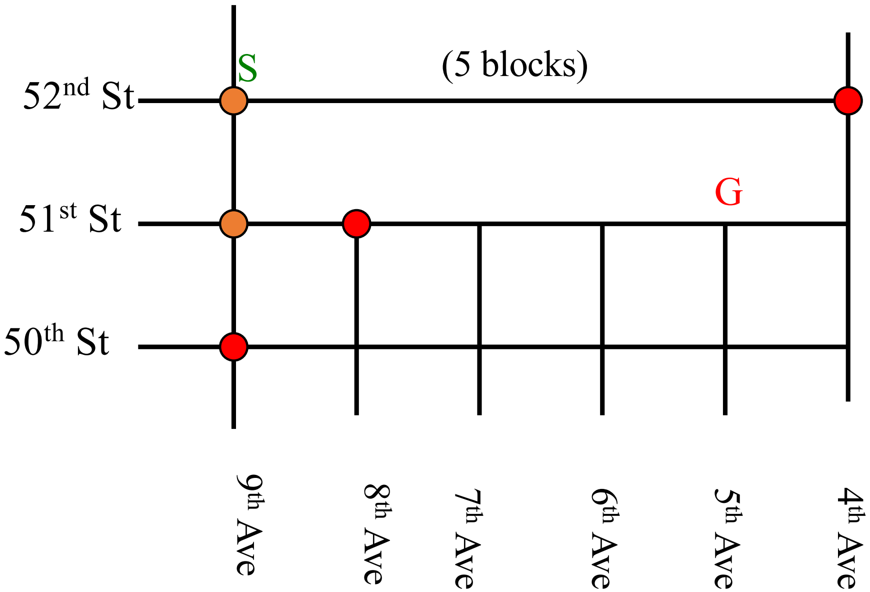
Use Manhattan distance to perform an A* search.



vertex	g(n)	h(n)	f(n)
52nd & 9th	0	5	5

Exercise…

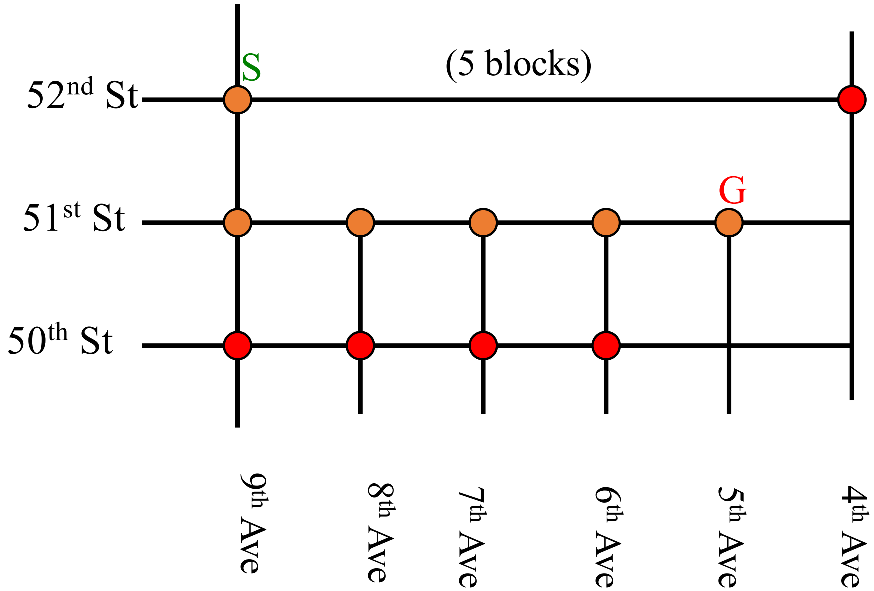
Use Manhattan distance to perform an A* search.



vertex	g(n)	h(n)	f(n)
52nd & 4th	5	2	7
51st & 8th	2	3	5
50th & 9th	2	5	7

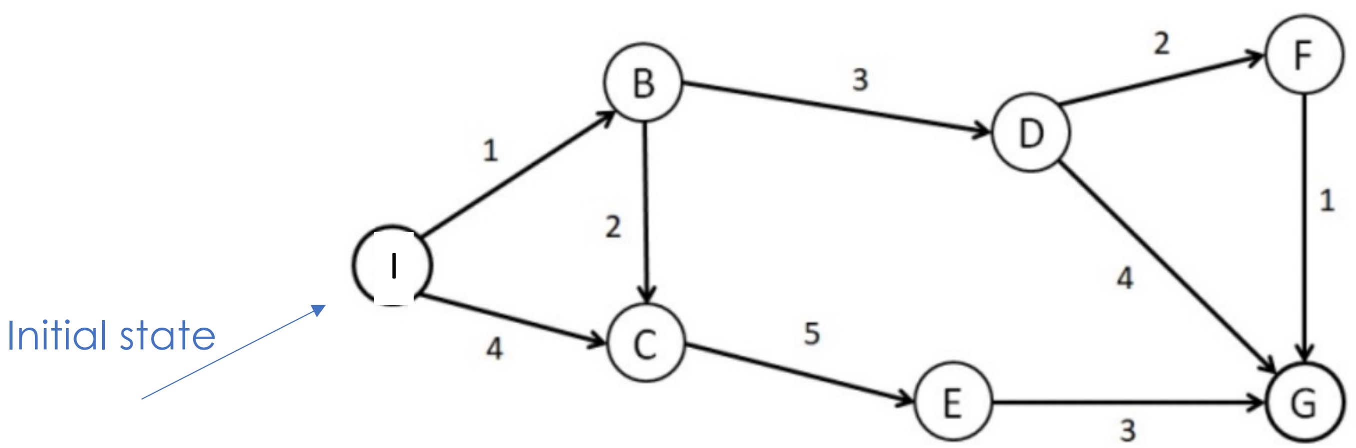
Exercise…

Use Manhattan distance to perform an A* search.



vertex	g(n)	h(n)	f(n)
52nd & 4th	5	2	7
50th & 9th	2	5	7
50th & 8th	3	4	7
50th & 7th	4	3	7

Exercise…



n	I	B	C	D	E	F	G
$h(n)$	5	4	4	3	3	1	0

a) GBFS; b) A* search; c) Admissible? Consistent?