Search Problems
Informed Search

Prof. Dr. Eduardo Noronha
Inteligência Artificial Aplicada
Instituto Federal de Goiás (IFG)

Topics Overview

Search in Artificial Intelligence

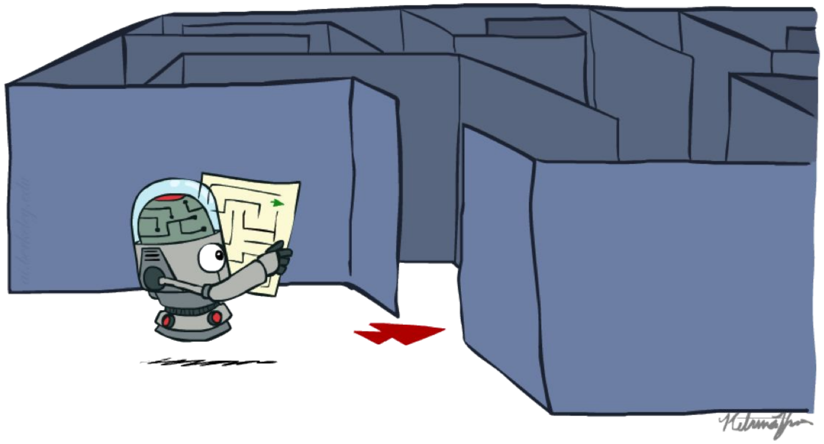
- **Uninformed Search**
  - Depth First Search
  - Breadth First Search
  - Uniform Cost Search
- **Informed Search**
  - A star Search
  - Greedy Search
  - Heuristics
- **Evolutionary Search**
  - Genetic Algorithm

Machine Learning

- Perceptron
- Neural Network
- Neural Network and Decision Trees
- Backpropagation

Uninformed Search - Recap

- Search problem:
  - States (configurations of the world)
  - Actions and costs
  - Successor function (world dynamics)
  - Start state and goal test
- Search tree:
  - Nodes: represent plans for reaching states
  - Plans have costs (sum of action costs)
- Search algorithm:
  - Systematically builds a search tree
  - Chooses an ordering of the fringe (unexplored nodes)
  - Optimal: finds least-cost plans



[This slide was adapted from Dan Klein and Pieter Abbeel at UC Berkeley]

Uninformed Search - Recap

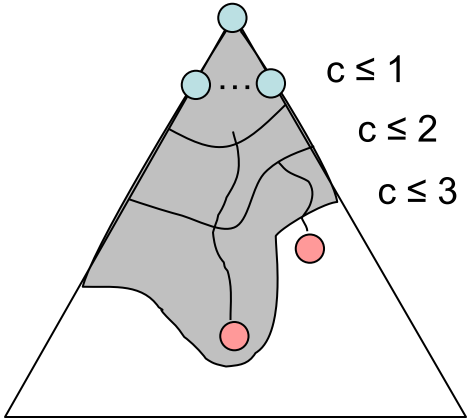
- DFS: Depth first search
- BFS: breadth first search
- UCS: Uniform cost search
- Analysis: Complete, optimal, time and space

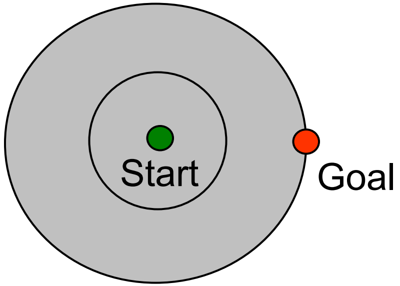
Uniform Cost Search

- Strategy: expand lowest path cost

- The good: UCS is complete and optimal!

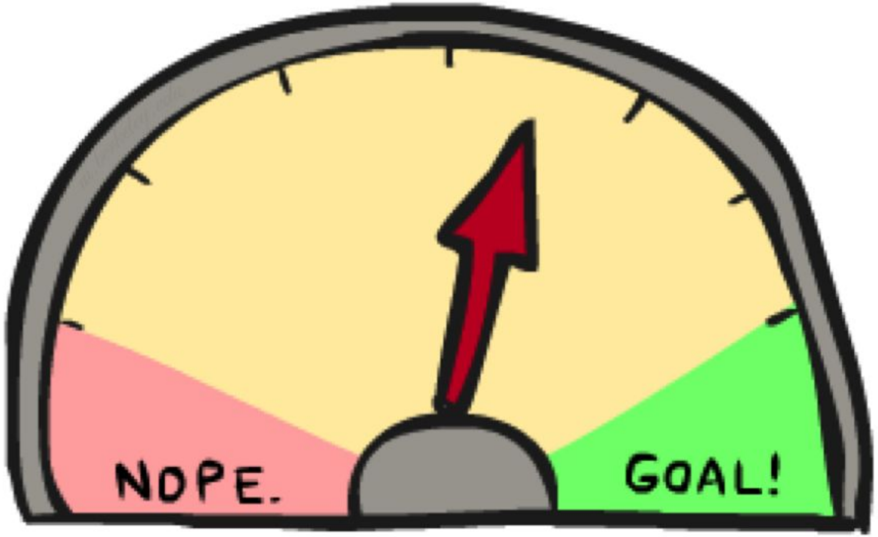
- The bad:
  - Explores options in every “direction”
  - No information about goal location





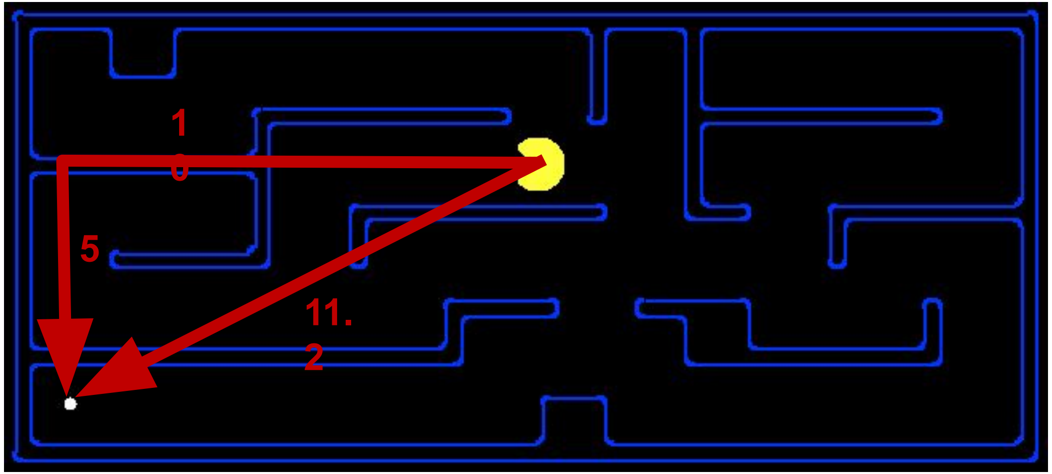
[This slide was adapted from Dan Klein and Pieter Abbeel at UC Berkeley]

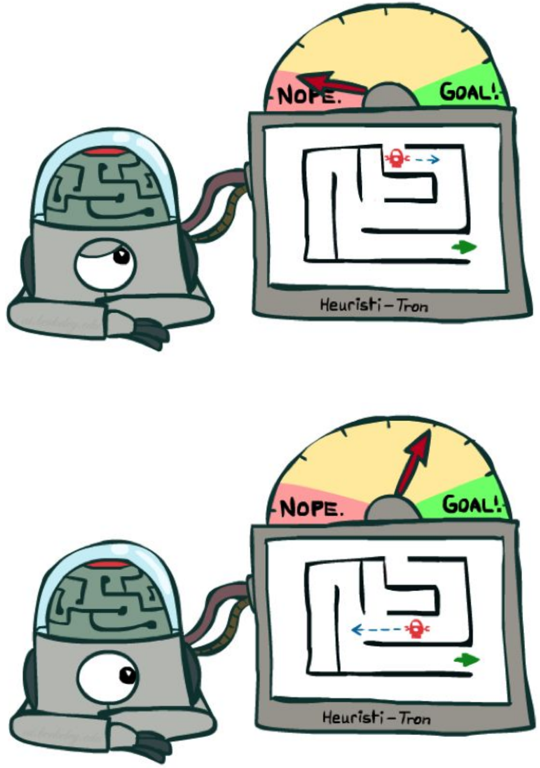
Informed Search



Search Heuristics

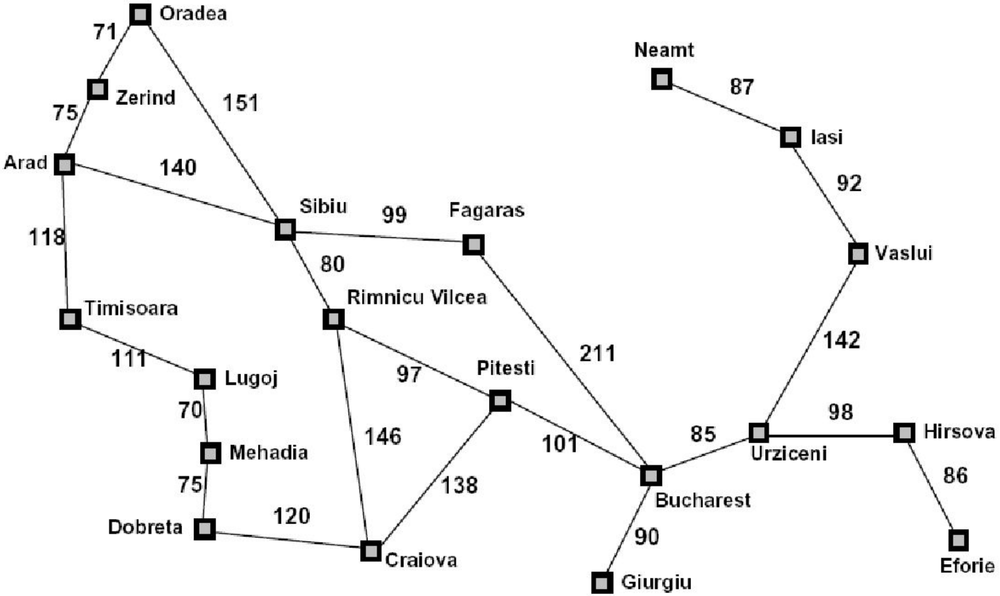
- A heuristic is:
  - A function that *estimates* how close a state is to a goal
  - Designed for a particular search problem
  - Examples: Manhattan distance, Euclidean distance for pathing





[This slide was adapted from Dan Klein and Pieter Abbeel at UC Berkeley]

Example: Heuristic Function



Straight−line distance to Bucharest

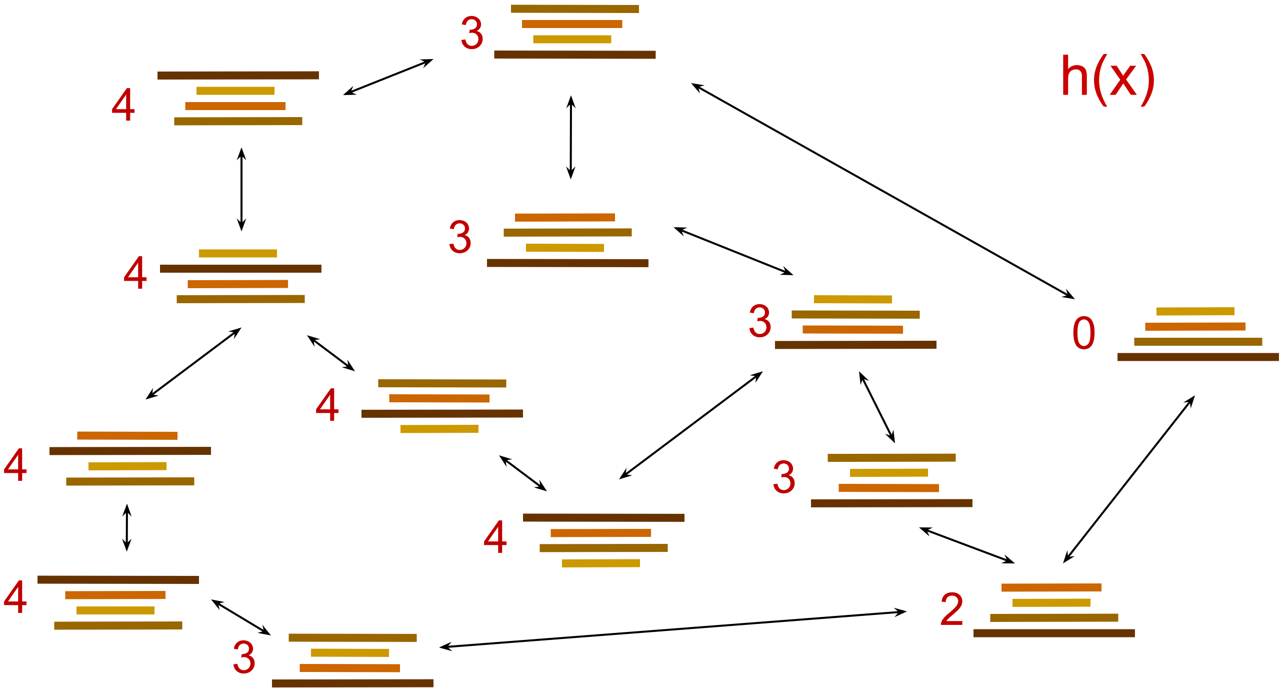
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

h(x)

[This slide was adapted from Dan Klein and Pieter Abbeel at UC Berkeley]

Example: Heuristic Function

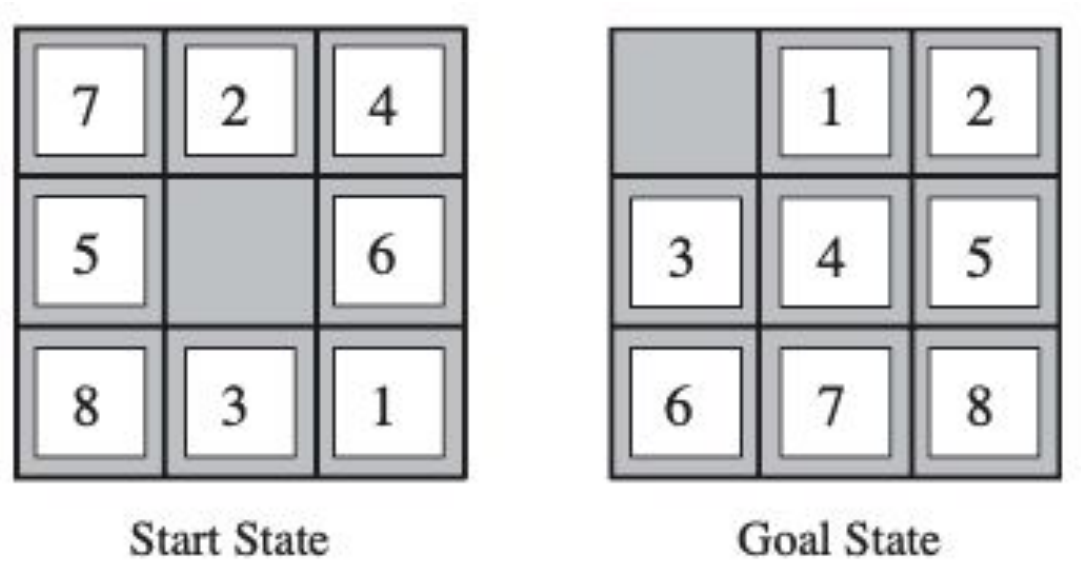
Heuristic: the number of the largest pancake that is still out of place



[This slide was adapted from Dan Klein and Pieter Abbeel at UC Berkeley]

Example: Heuristic Function

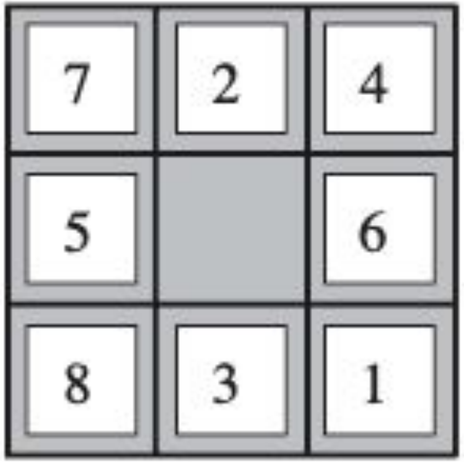
Heuristic (***<u>h1</u>***): the number of misplaced tiles.



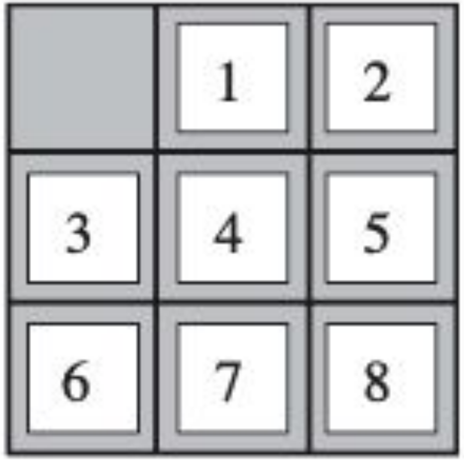
h1 = 1+1+1+1+1+1+1+1 = 8

Example: Heuristic Function

Heuristic (<u>*h2*</u>): the sum of the distances of the tiles from their goal positions.



Start State

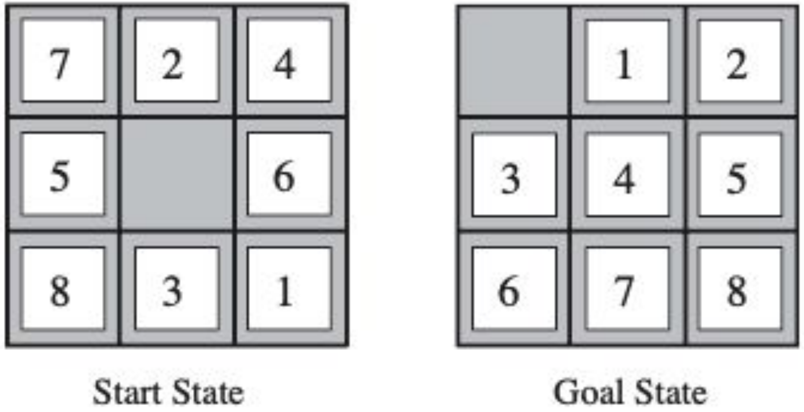


Goal State

$$h_2 = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$$

Example: 8-puzzle

- The average solution cost for a randomly generated 8-puzzle instance is about 22 steps.
- The branching factor is about 3.
- exhaustive tree search to depth 22 would look at about $3^{22} \approx 3.1 \times 10^{10}$ states



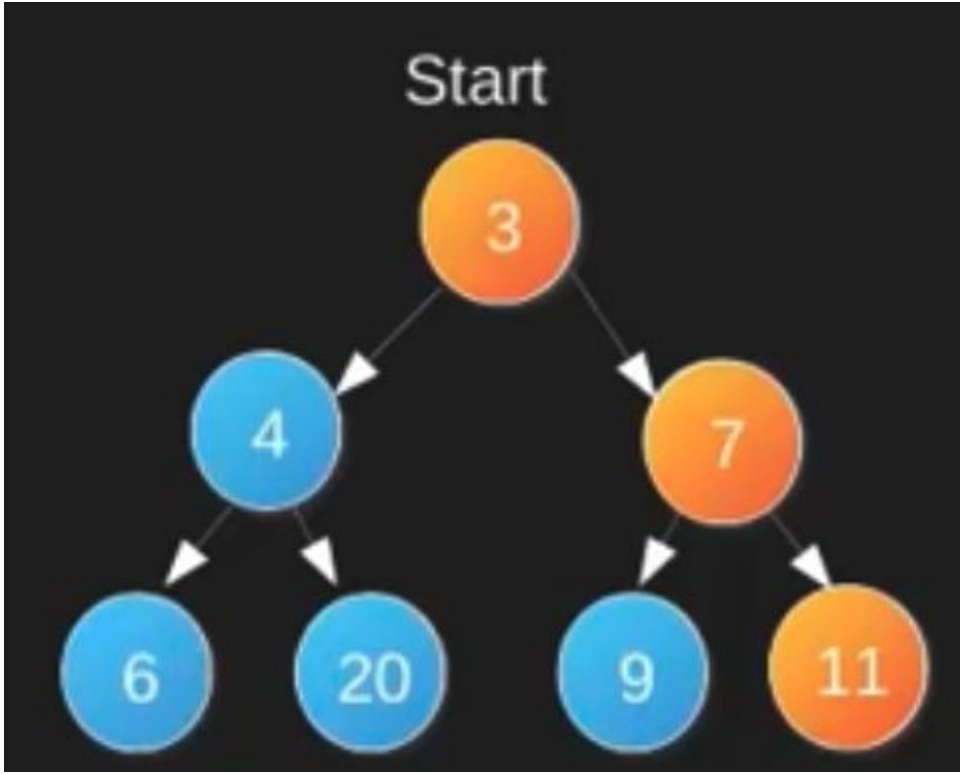
Greedy Search

An algorithmic paradigm that follows the problem solving approach of making the <u>locally optimal choice</u> at each stage with the <u>hope of finding</u> a global optimum.

<u>Pros</u>: simple, easy to implement, run fast

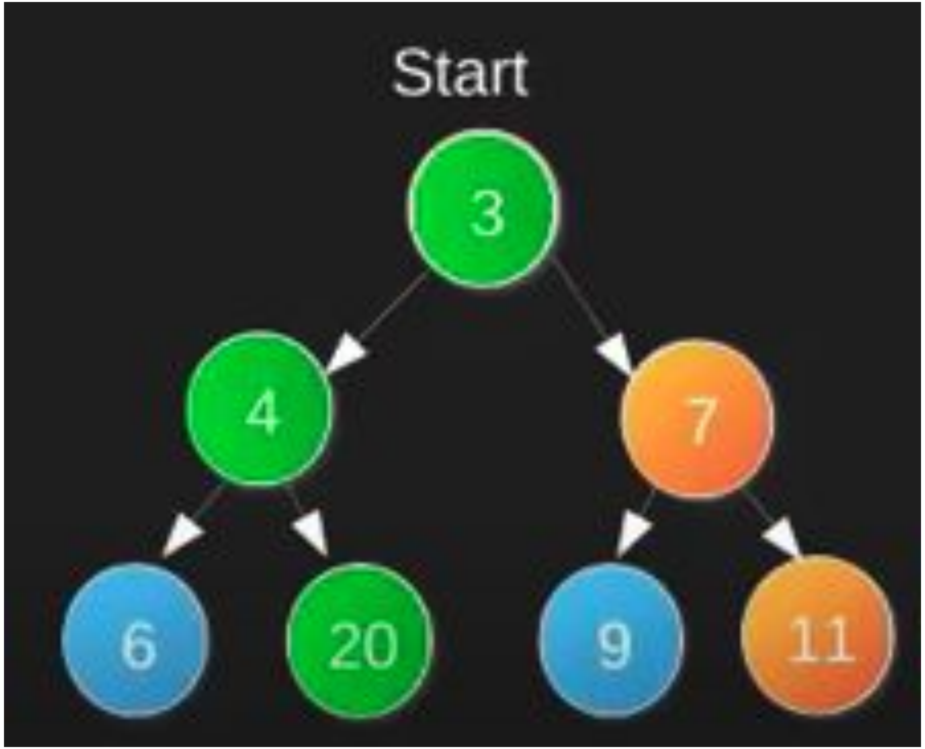
<u>Cons</u>: Very often they don't provide the global optimal solution

Greedy Search



Greedy answer: 3 + 7 + 11 = 21

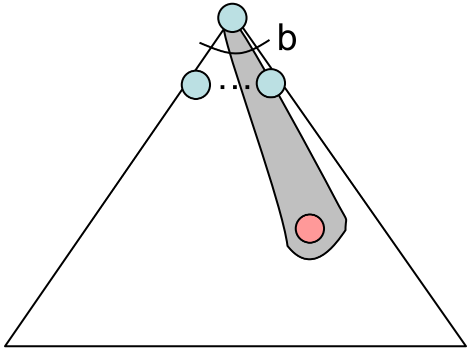
Greedy Search



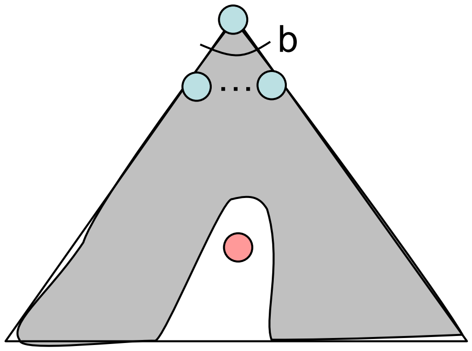
Optimal answer: 3 + 4 + 20 = 27

Greedy Search

- Strategy: expand a node that you think is closest to a goal state
  - Heuristic: estimate of distance to nearest goal for each state



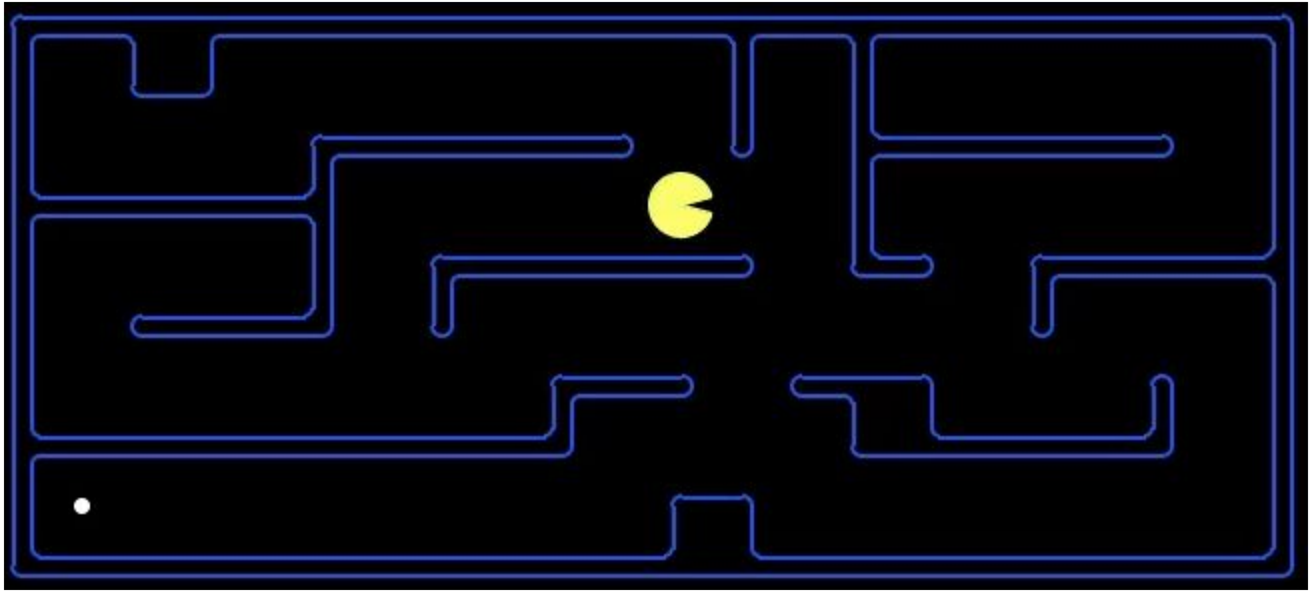
- A common case:
  - Best-first takes you straight to the (wrong) goal (as we see in the previous example)



- Worst-case: like a badly-guided DFS

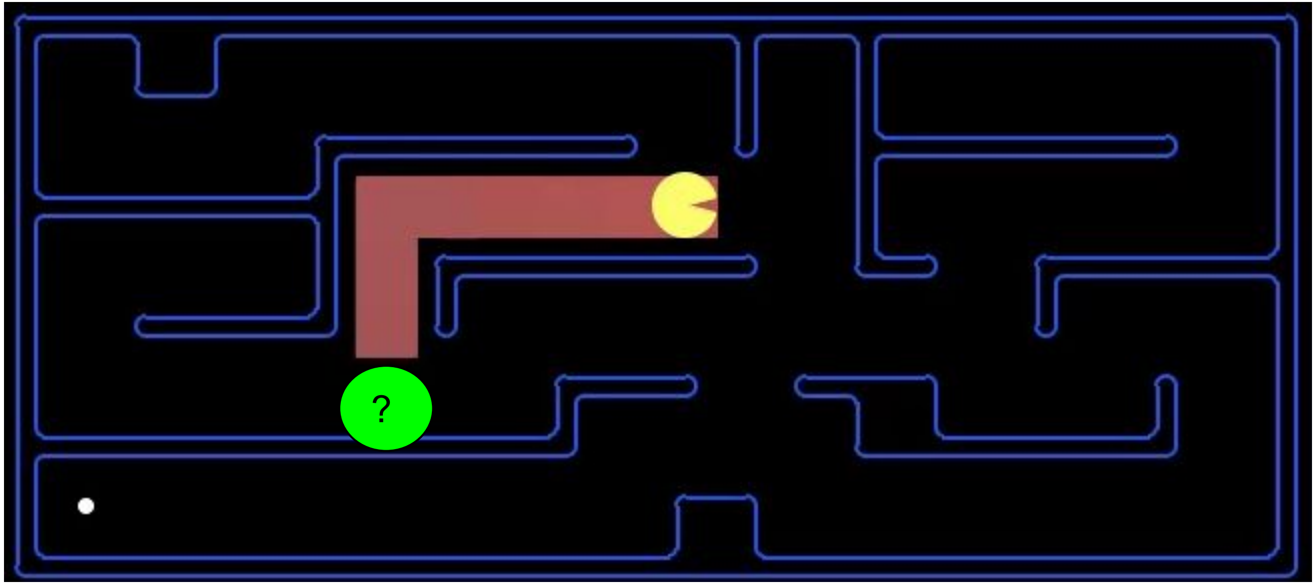
Greedy Search

Think about **heuristic** as Euclidean Distance.

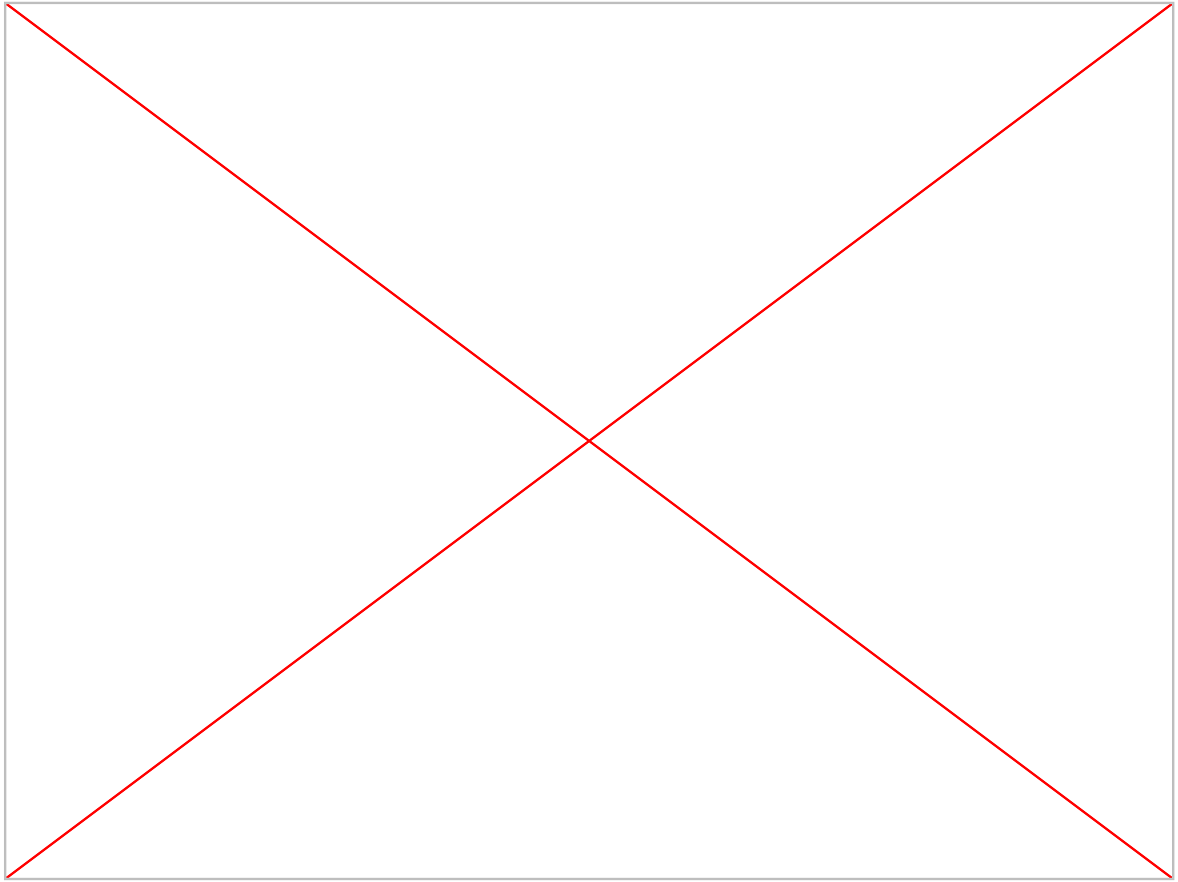


Greedy Search

What decision to take here, as Pac-man is guided by Euclidean Distance?



Greedy Search

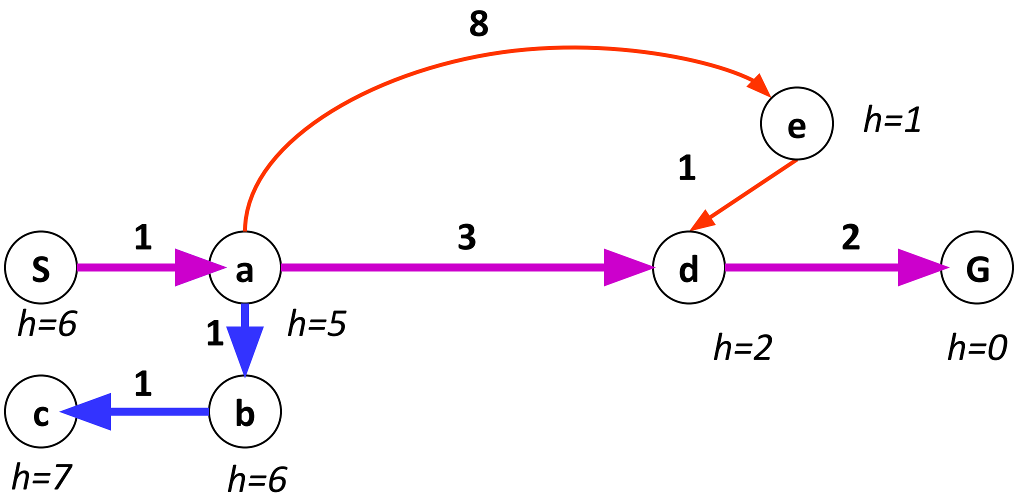


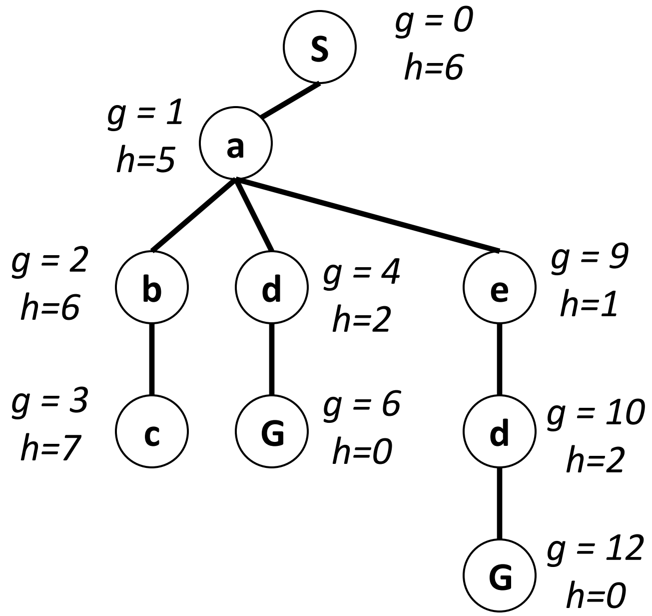
A* Search

[This slide was adapted from Dan Klein and Pieter Abbeel at UC Berkeley]

Combining UCS and Greedy

- Uniform-cost orders by path cost, or *backward cost* g(n)
- Greedy orders by goal proximity, or *forward cost* h(n)





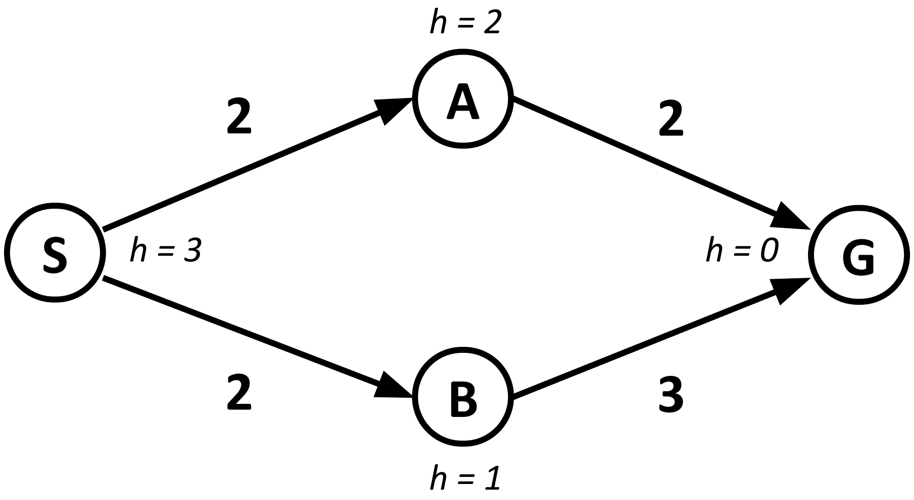
- A* Search orders by the sum: f(n) = g(n) + h(n)

[This slide was adapted from Dan Klein and Pieter Abbeel at UC Berkeley]

Example: Teg Grenager

When should A* terminate?

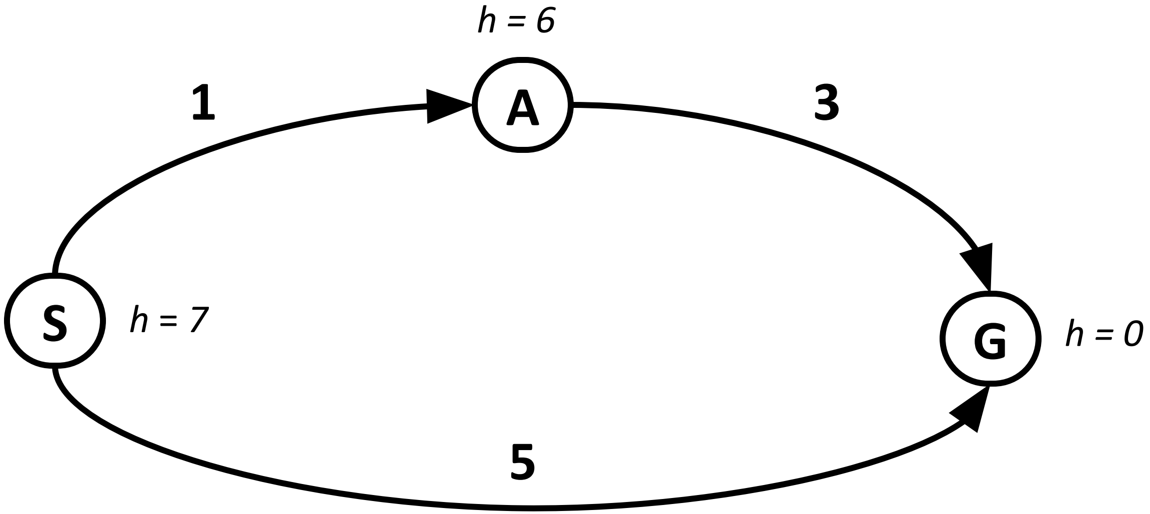
- Should we stop when we enqueue a goal?



Path	*G*	*H*	*F*
S	0	3	3
S→A	2	2	4
S→B	2	1	3
S→B→G	**2+3=5**	**0**	**5**
S→A→G	**2+2=4**	**0**	**4**

- No: A* terminates when the path it chooses to extend is a path from start to goal **AND** if there are no paths eligible to be extended

Is A* Optimal?



- What went wrong?
- Actual bad goal cost < estimated good goal cost
- We need estimates to be less than actual costs!

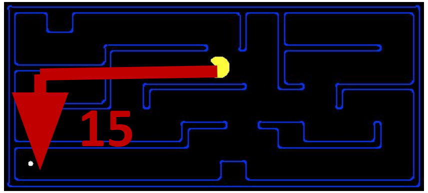
Admissible Heuristics

- A heuristic h is *admissible* (optimistic) if:

$$0 \leq h(n) \leq h^*(n)$$

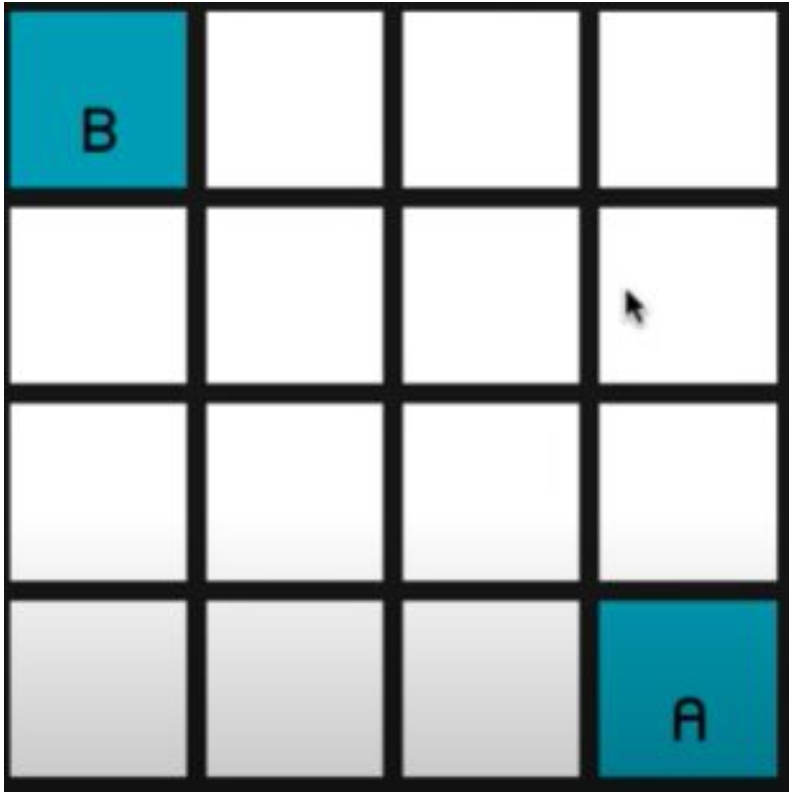
where $h^*(n)$ is the true cost to a nearest goal

- Examples:



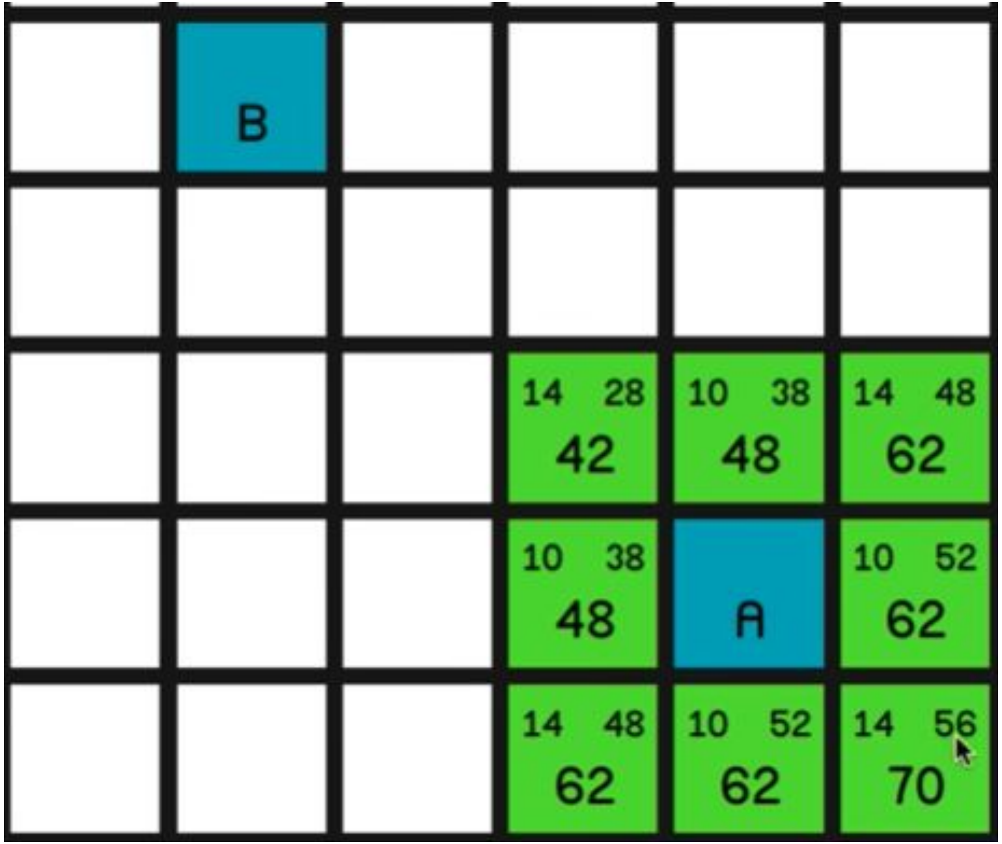
- Coming up with admissible heuristics is most of what's involved in using A* in practice.

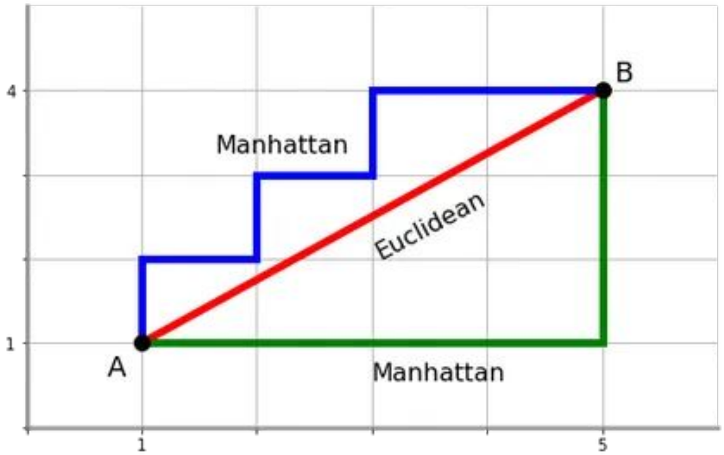
A* Search - Example



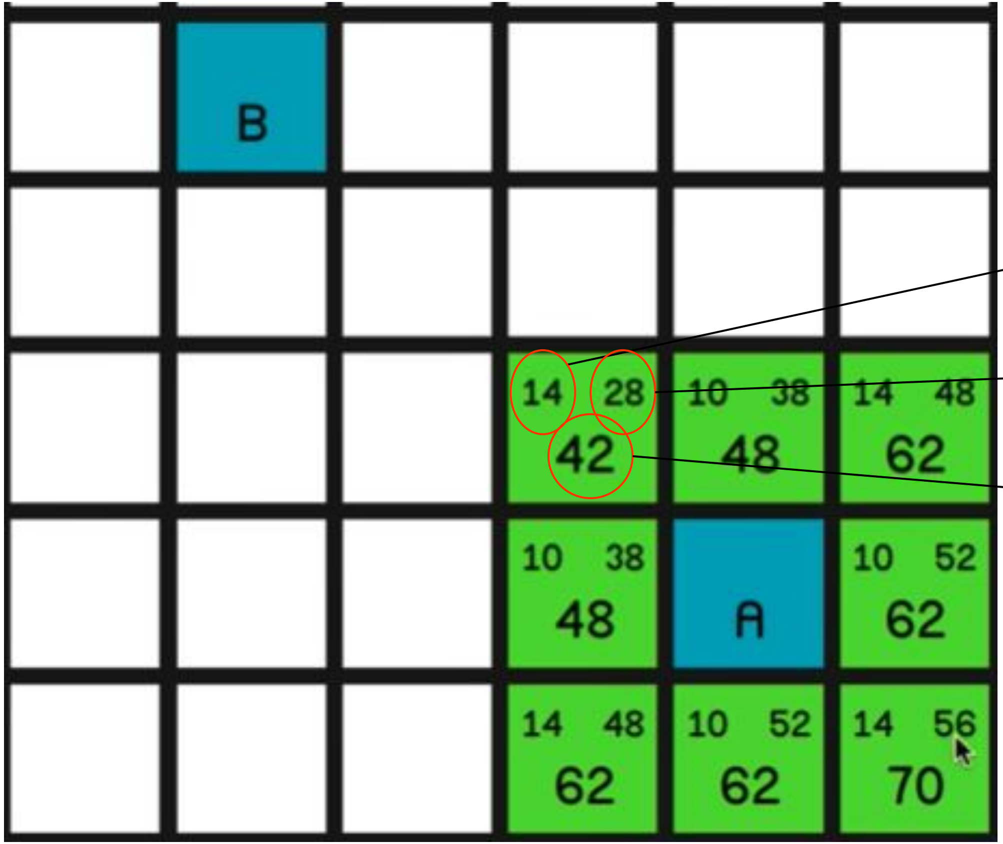
Example adapted from: https://www.youtube.com/watch?v=-L-WgKMFuhE

A* Search - Example





A* Search - Example

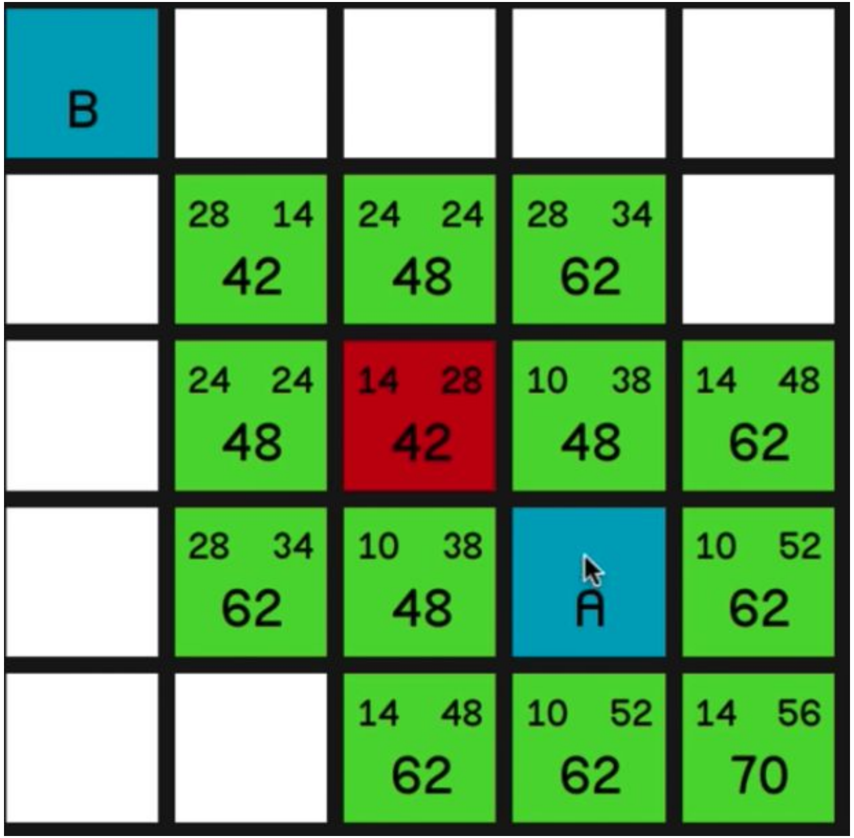


Gcost: distance from starting node

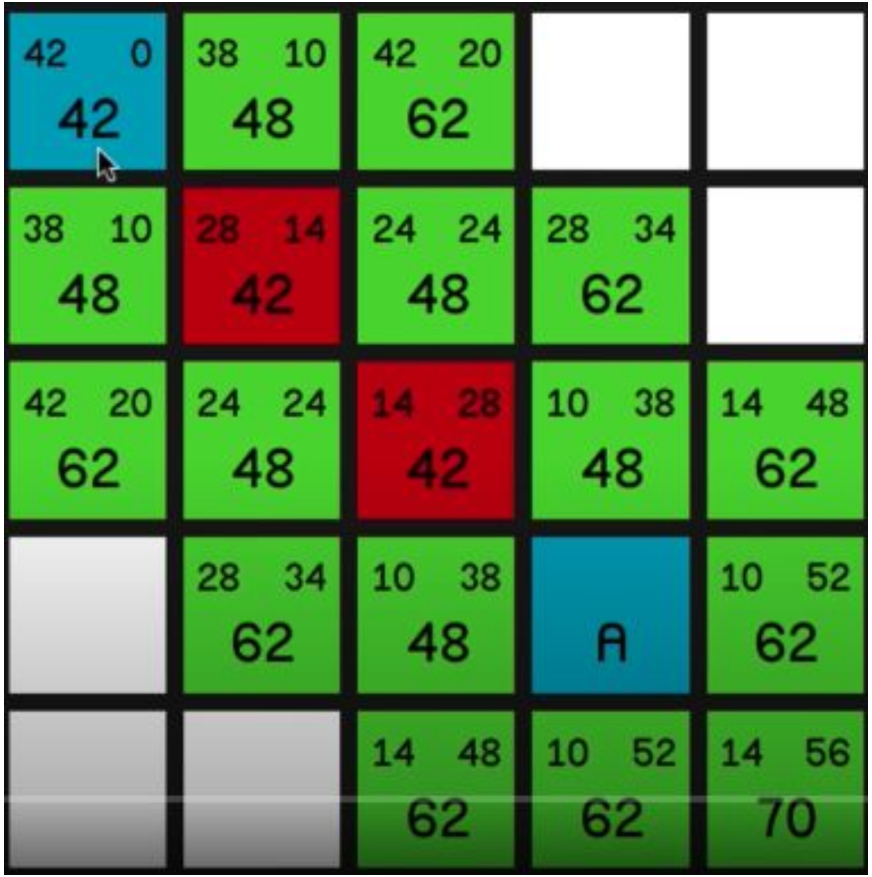
Hcost: distance from end node (heuristic)

Fcost: <u>Gcost</u> + <u>Hcost</u>

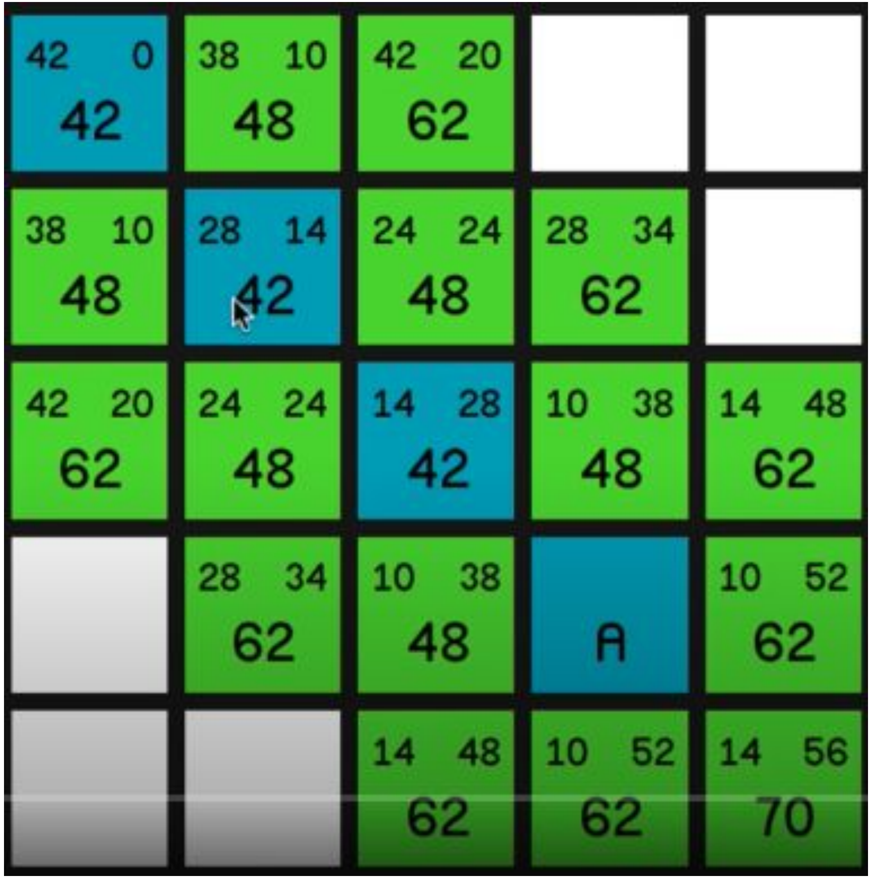
A* Search - Example



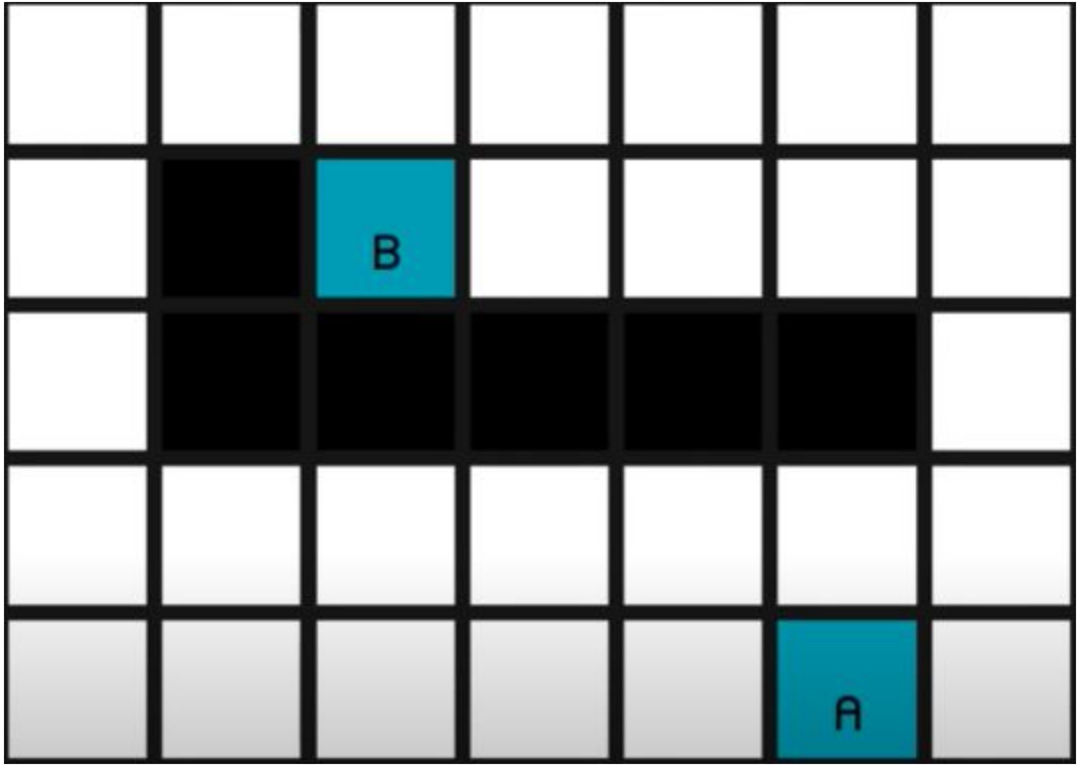
A* Search - Example



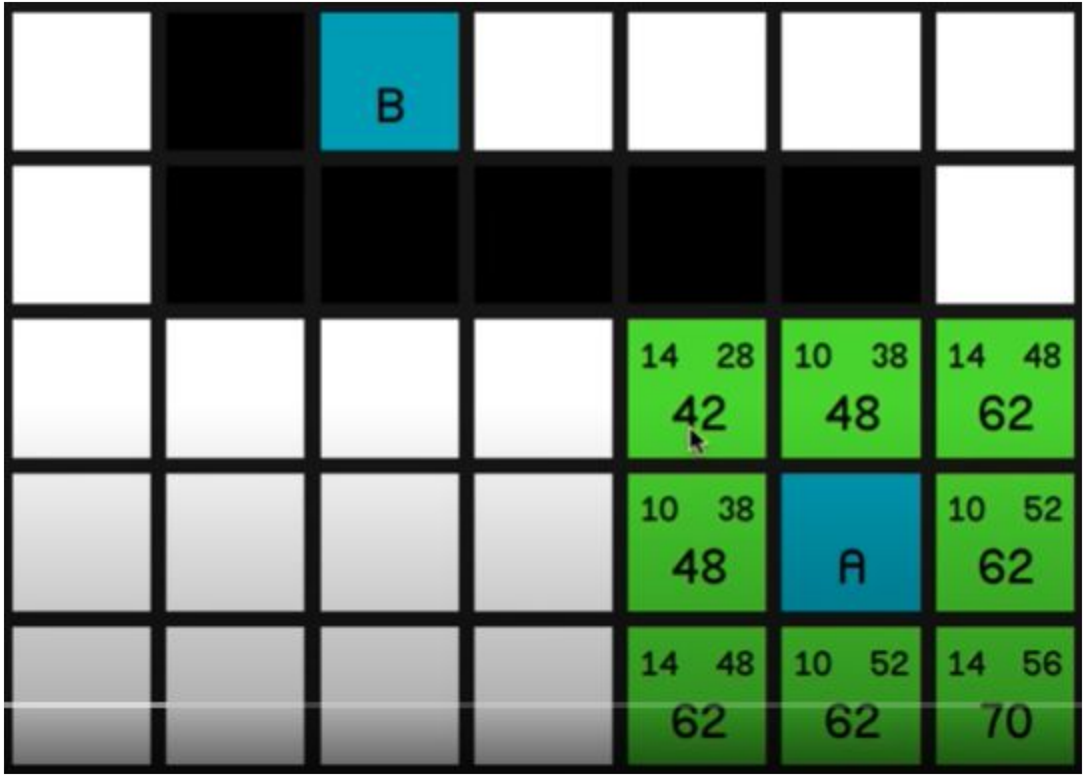
A* Search - Example



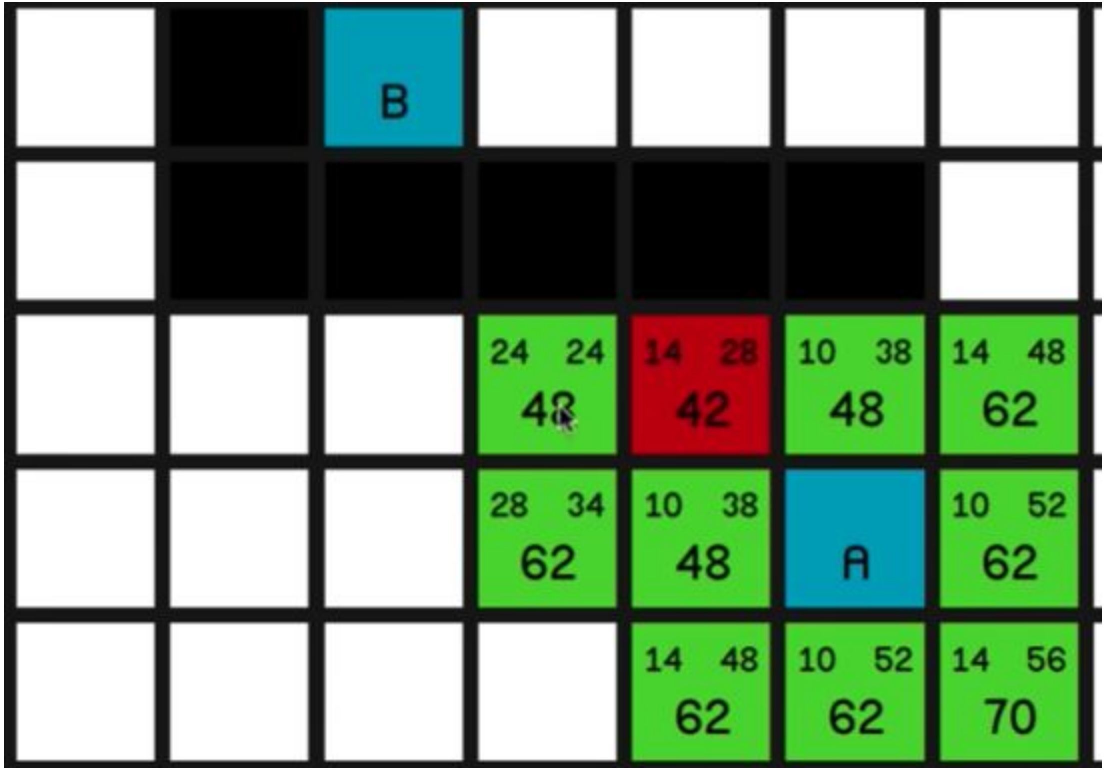
A* Search - Example



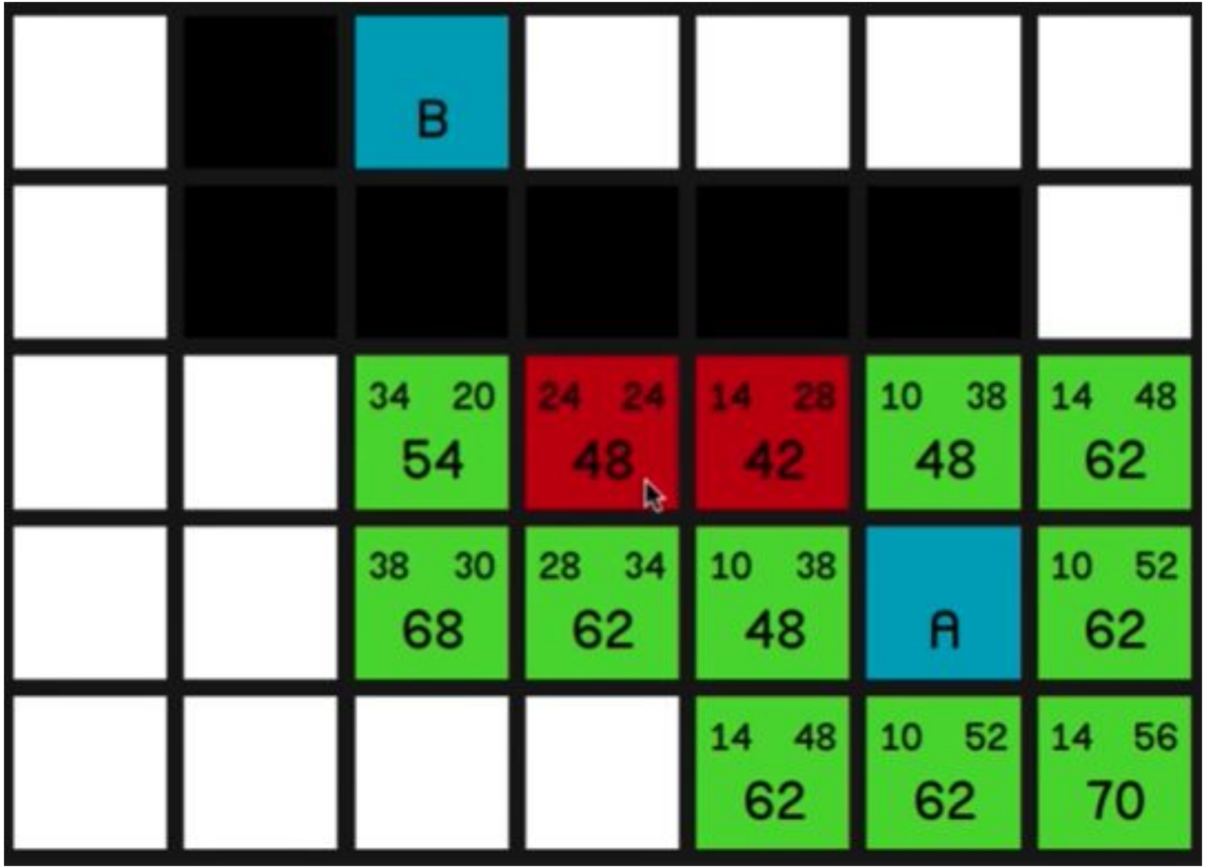
A* Search - Example



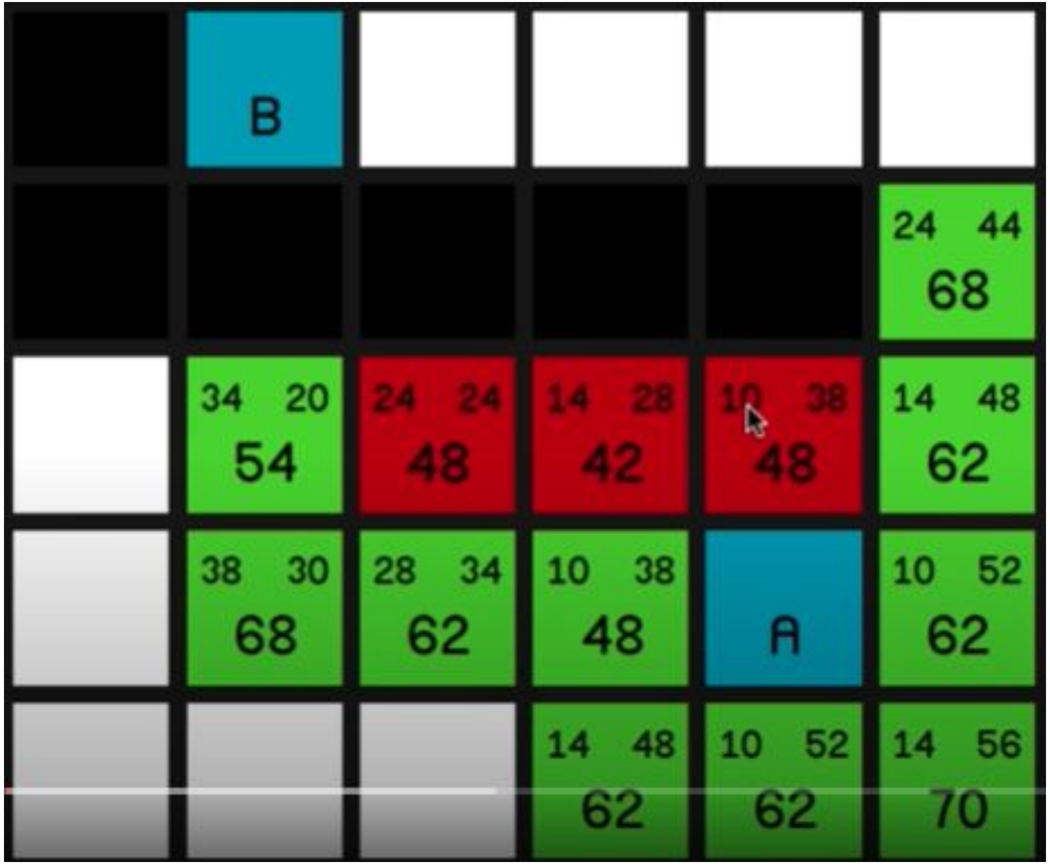
A* Search - Example



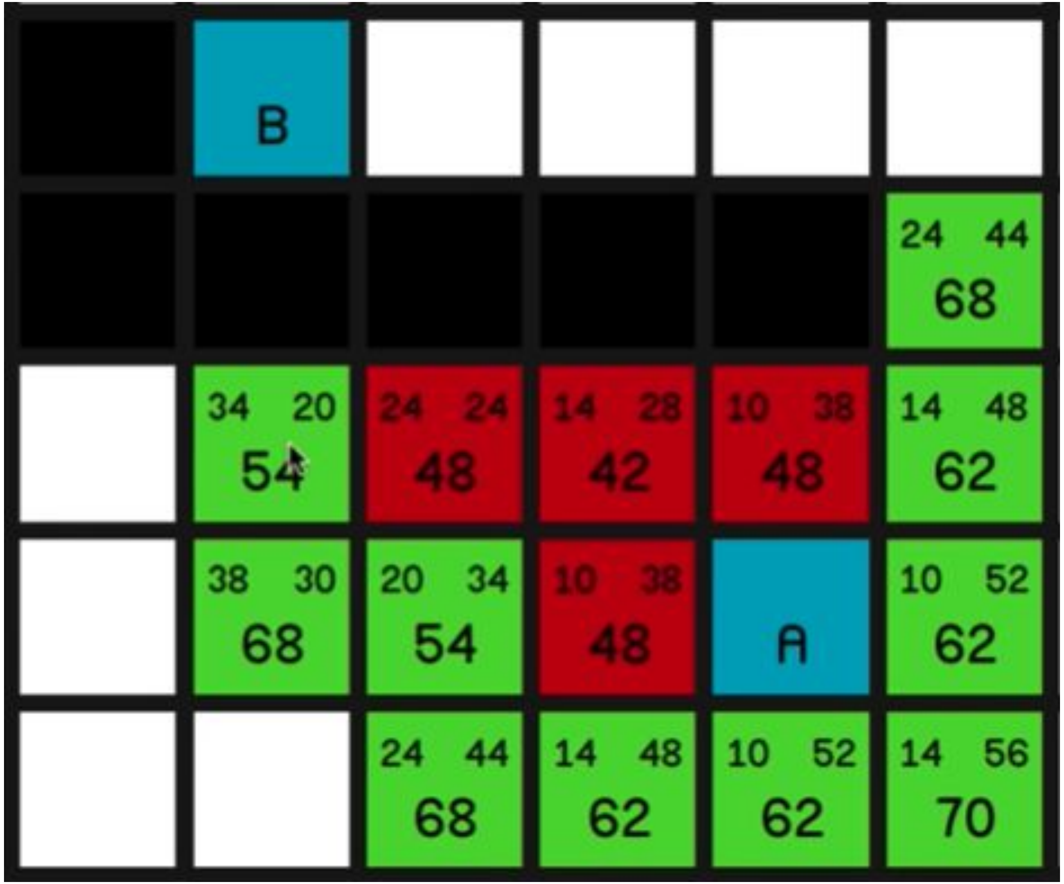
A* Search - Example



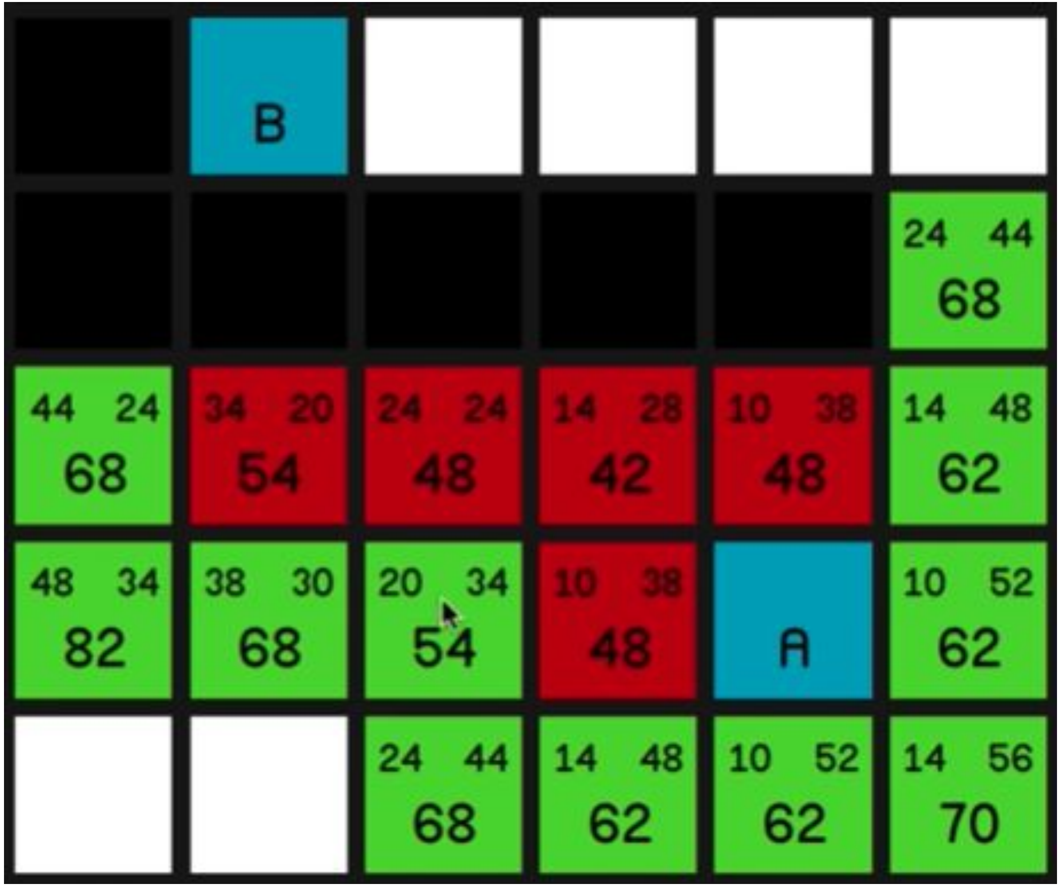
A* Search - Example



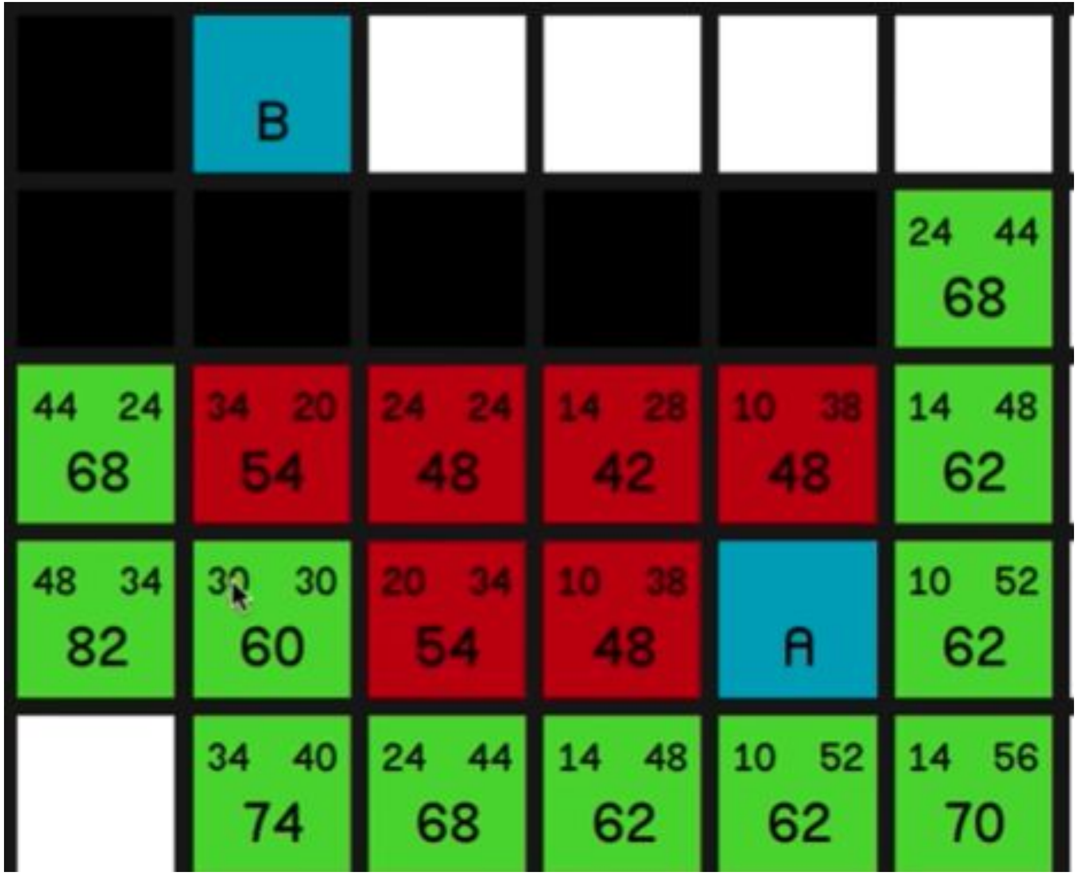
A* Search - Example



A* Search - Example



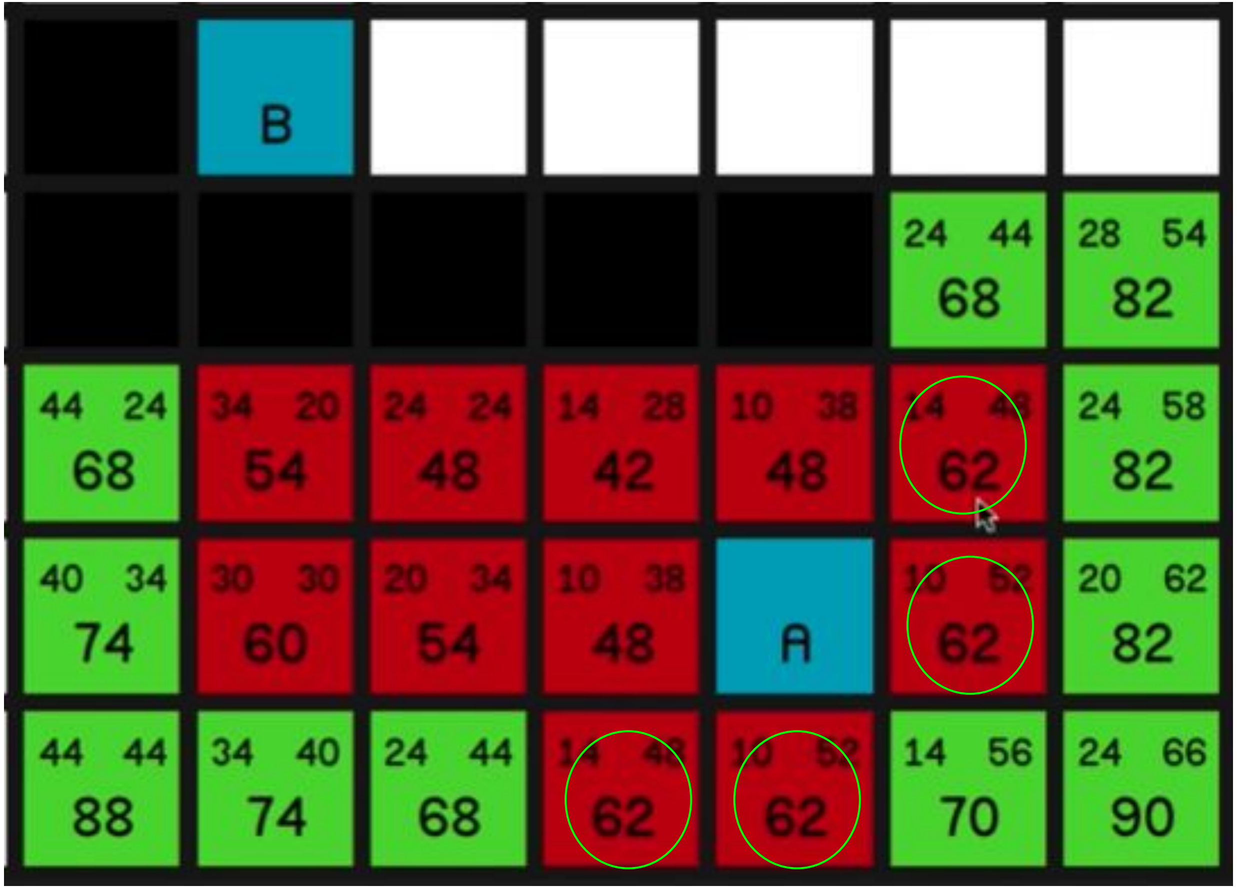
A* Search - Example



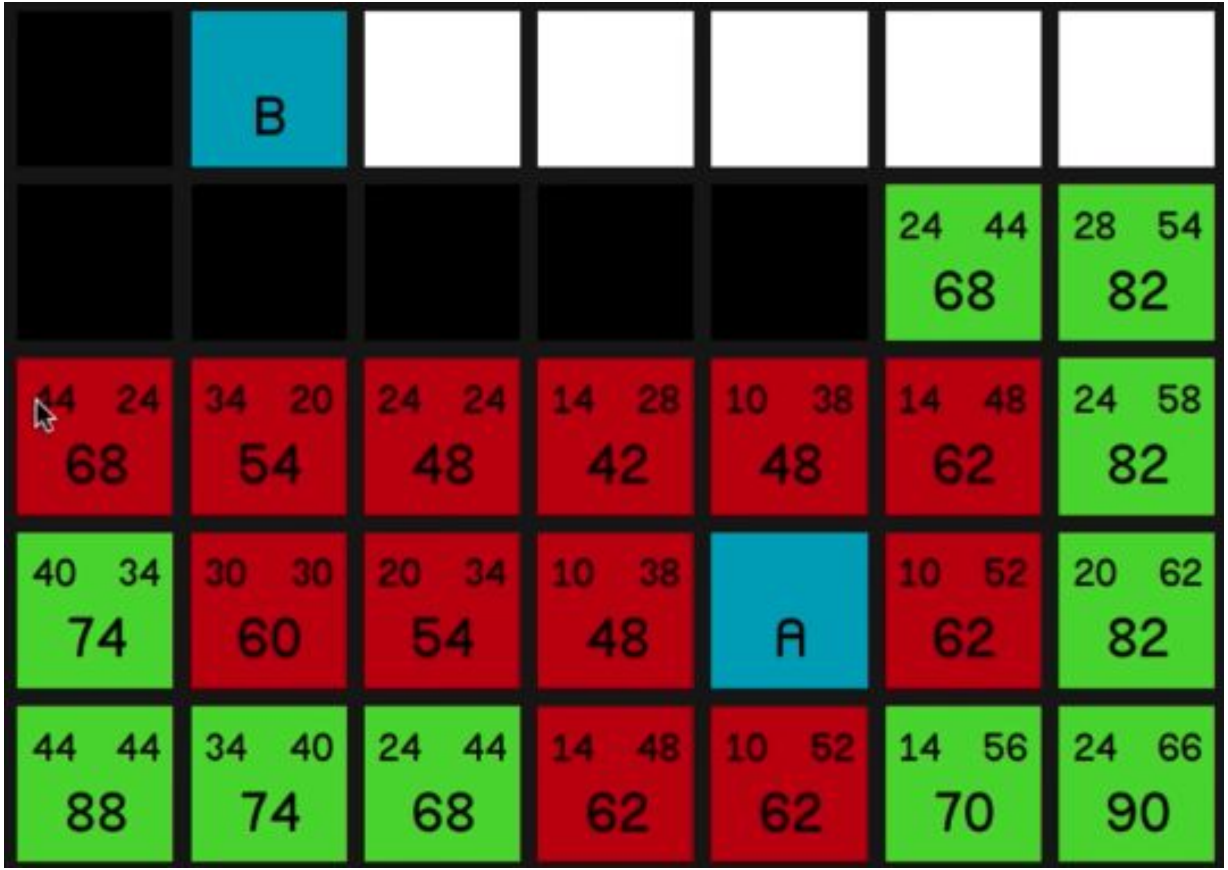
A* Search - Example

A* Search - Example

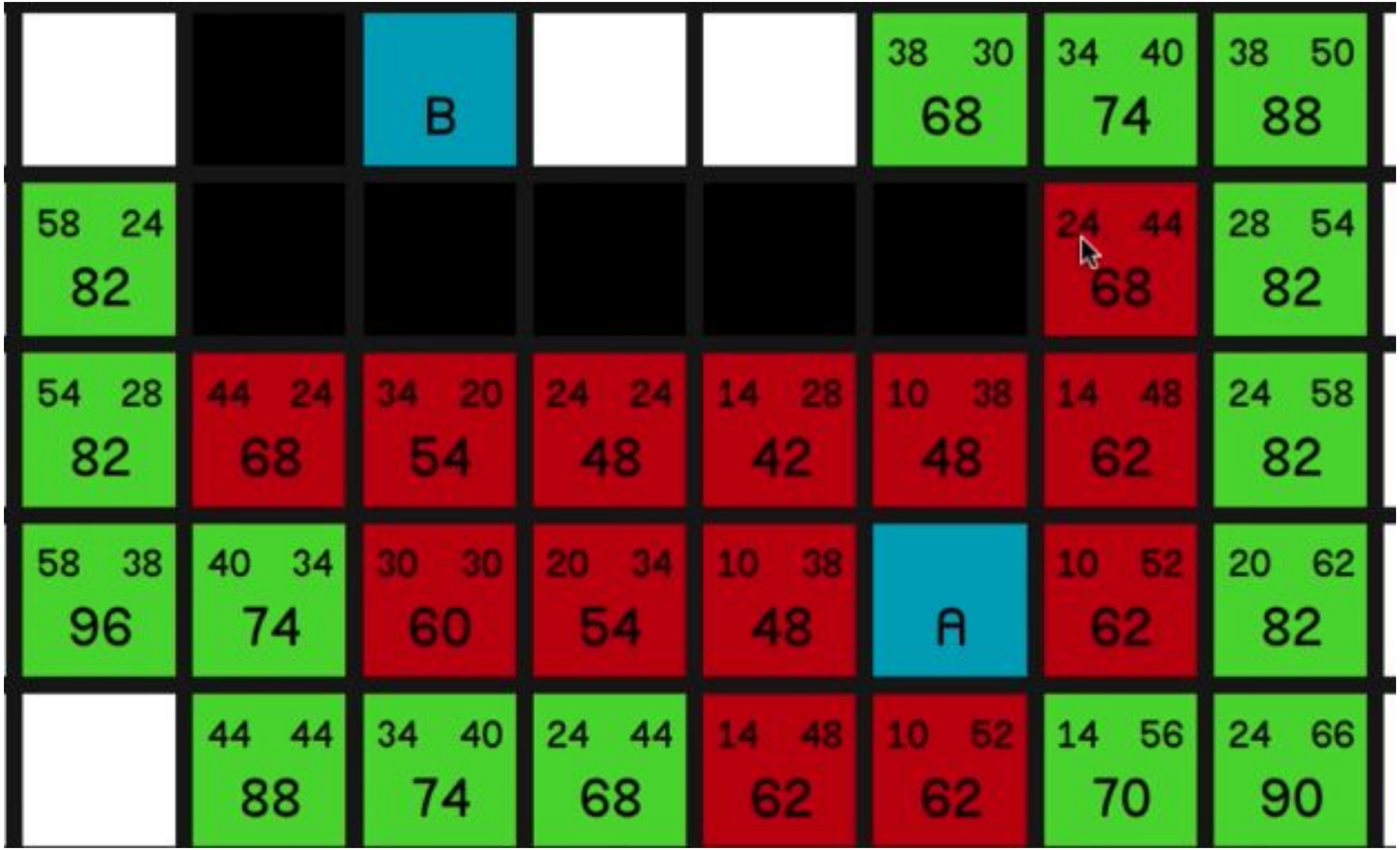
A* Search - Example



A* Search - Example

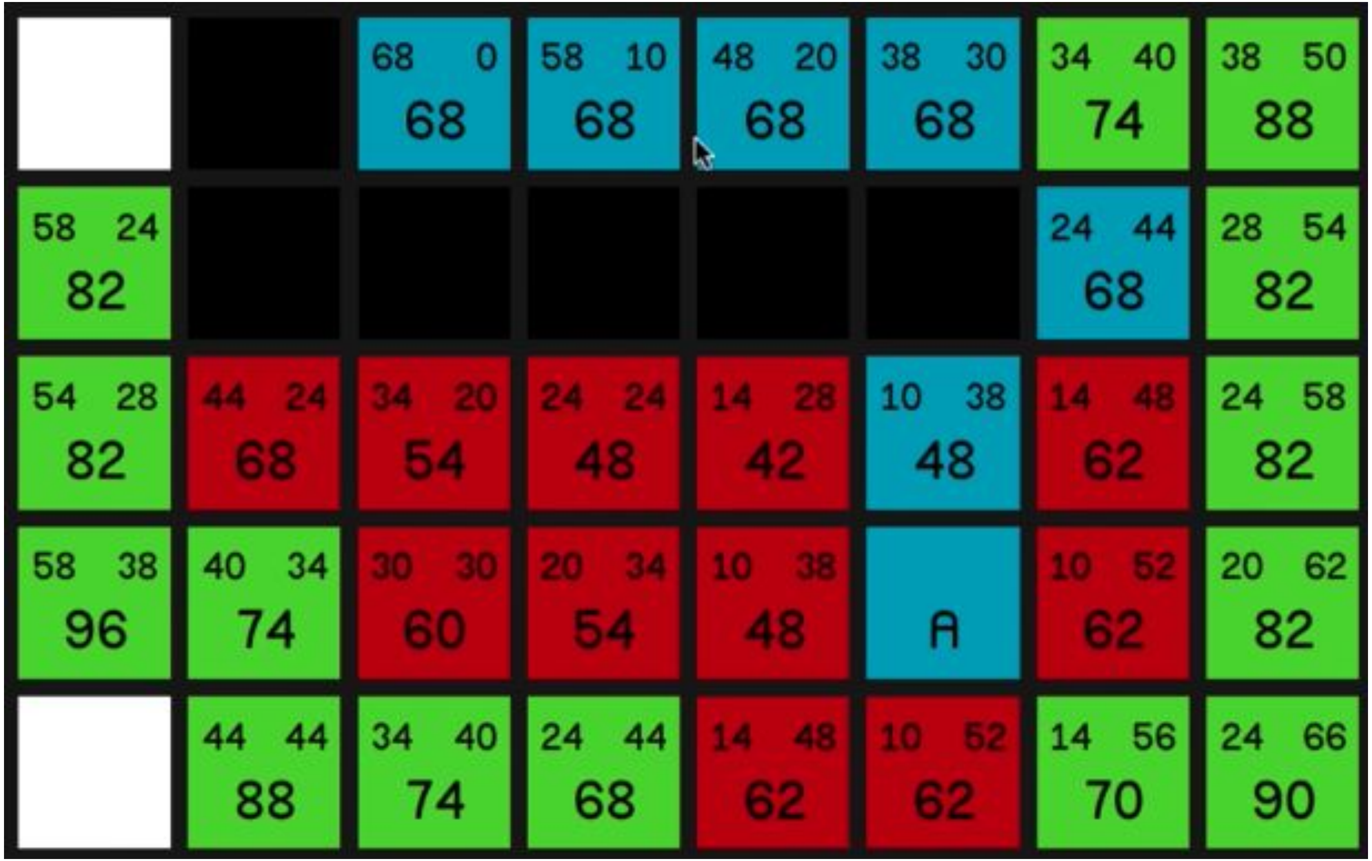


A* Search - Example

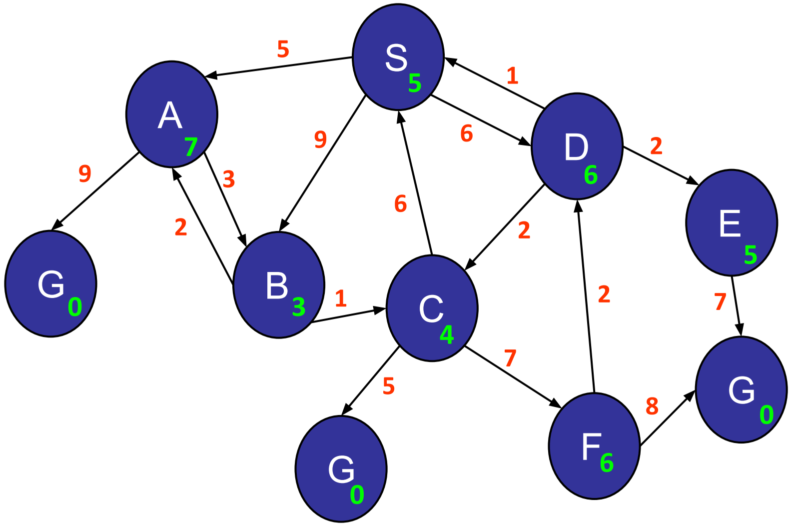


A* Search - Example

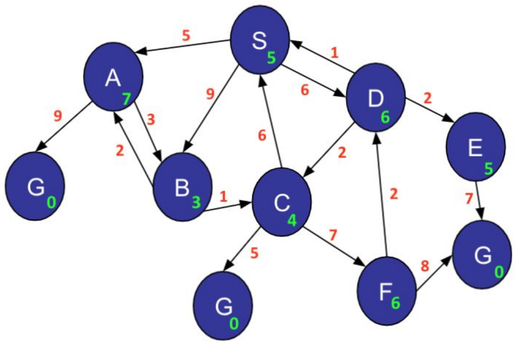
A* Search - Example

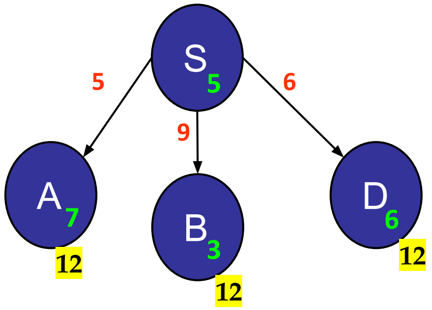


A* Search



A* Search - Example





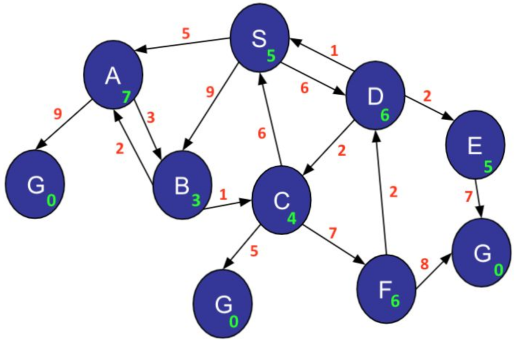
Visited: S(5)

Gcost: distance from starting node

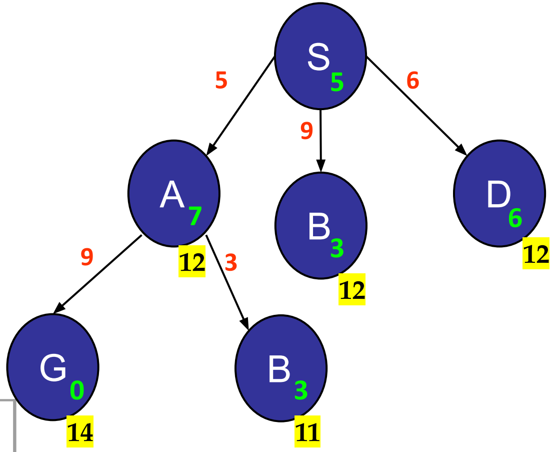
Hcost: distance from starting node (Heuristic)

A* = Gcost + Hcost

Path	G	H	A* = G + H
$S \rightarrow A$	5	7	12
$S \rightarrow B$	9	3	12
$S \rightarrow C$	6	6	12



A* Search - Example



Visited: S(5), A(12)

Visited: S(5)

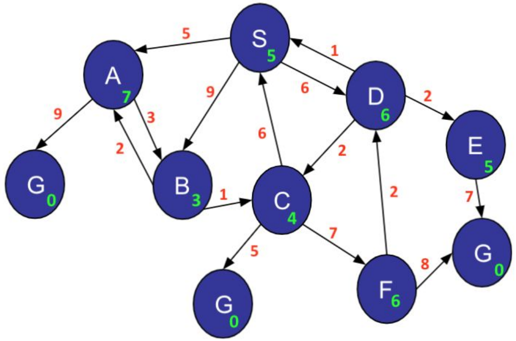
Gcost: distance from starting node

Hcost: distance from starting node (Heuristic)

A* = Gcost + Hcost

Path	G	H	A* = G + H
S → A	5	7	12
S → B	9	3	12
S → C	6	6	12
S → A → G	5+9=14	0	14
S → A → B	5+3=8	3	11

A* Search - Example

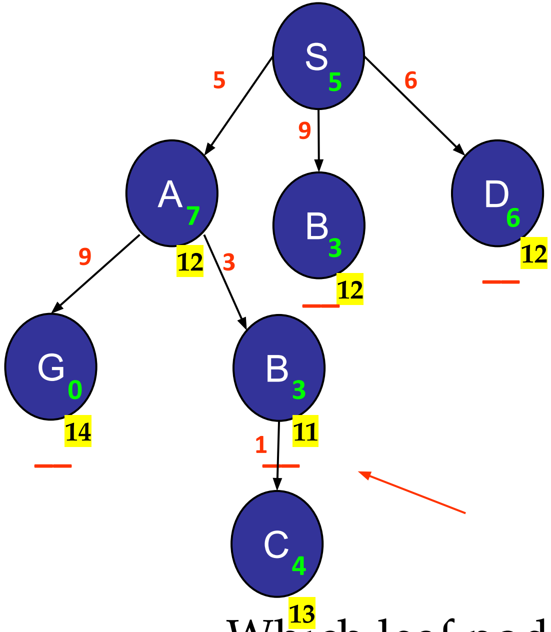


Gcost: distance from starting node

Hcost: distance from starting node (Heuristic)

A* = Gcost + Hcost

Path	G	H	A* = G + H
S → A	5	7	12
S → B	9	3	12
S → C	6	6	12
S → A → G	**5+9=14**	**0**	**14**
S → A → B	**5+3=8**	**3**	**11**
S → A → B → C	**5+3+1=9**	**4**	**13**

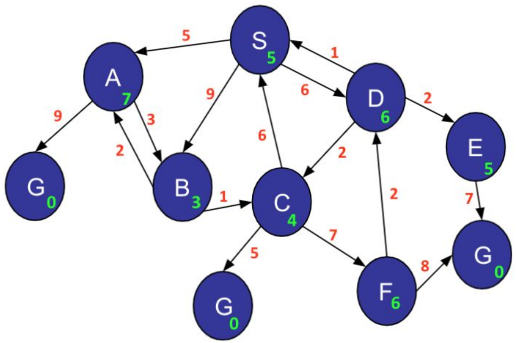


Visited: S(5), A(12), **B(11)**

Which leaf nodes were not visited yet?

Which is the most promising?

A* Search - Example

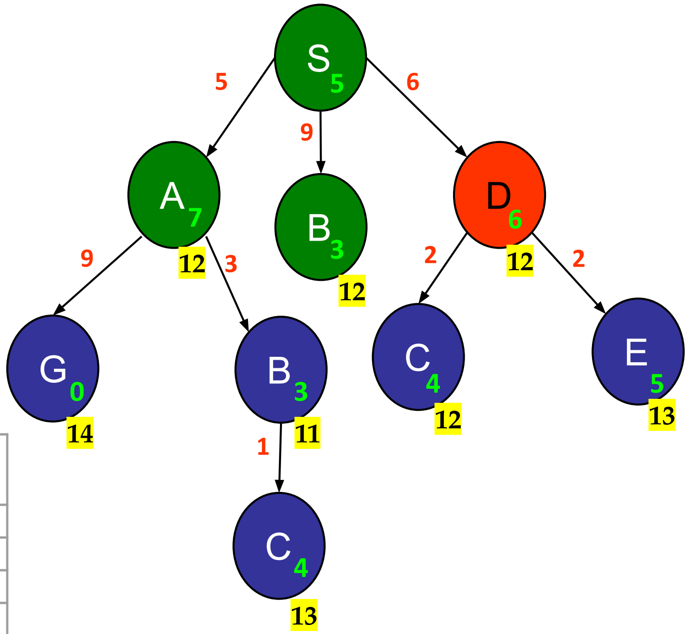


Gcost: distance from starting node

Hcost: distance from starting node (Heuristic)

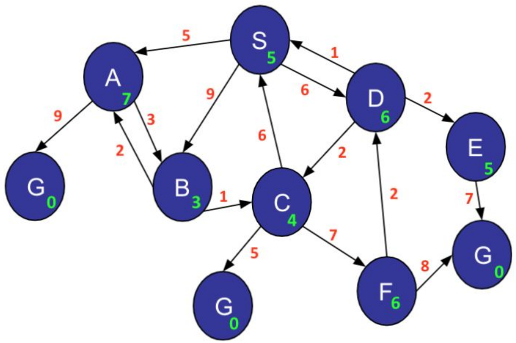
A* = Gcost + Hcost

Path	G	H	A* = G + H
S → A	5	7	12
S → B	9	3	12
S → C	6	6	12
S → A → G	**5+9=14**	**0**	**14**
S → A → B	**5+3=8**	**3**	**11**
S → A → B → C	**5+3+1=9**	**4**	**13**
S → D → C	**6+2=8**	**4**	**12**
S → D → E	**6+2=8**	**5**	**13**



Visited: **S**(5), **A**(12), **B**(11)

A* Search - Example

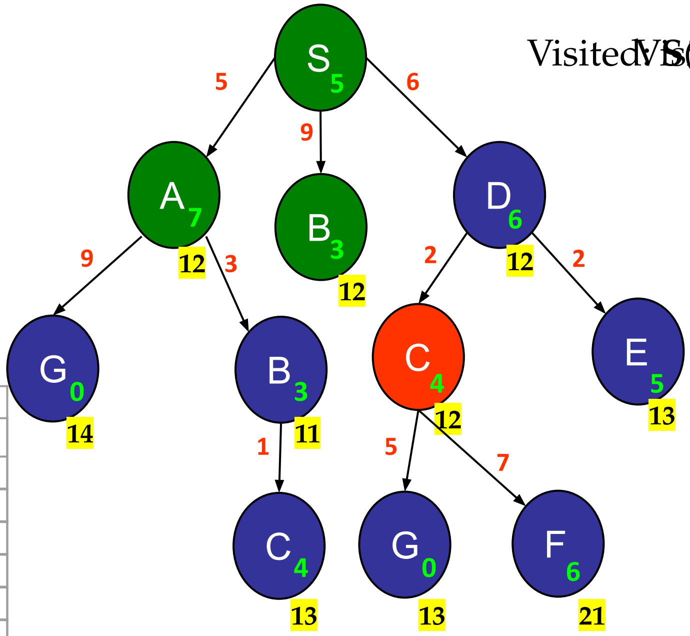


Gcost: distance from starting node

Hcost: distance from starting node (Heuristic)

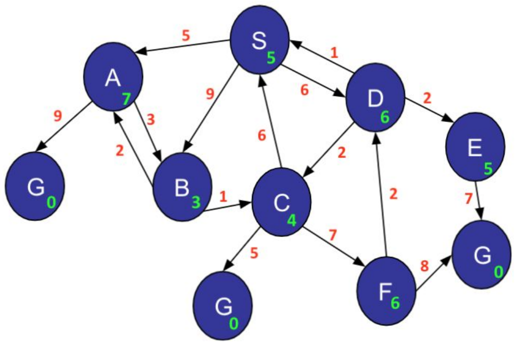
A* = Gcost + Hcost

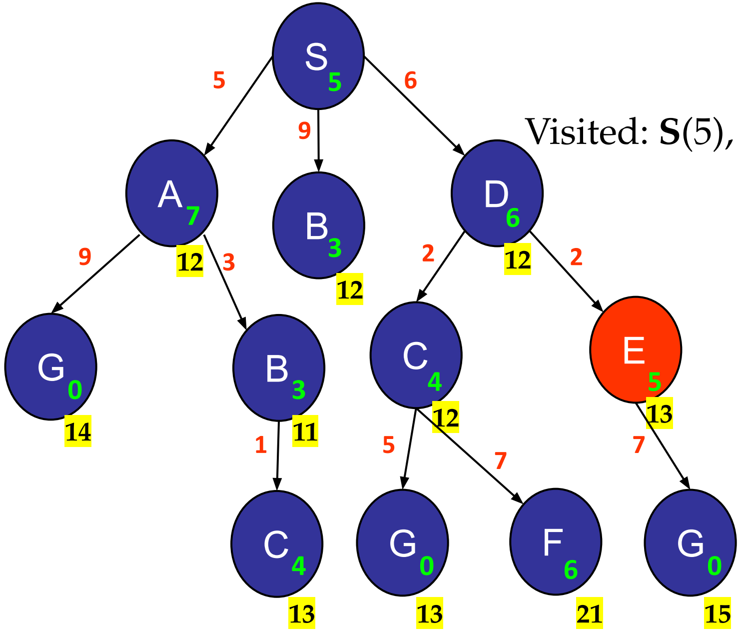
Path	G	H	A* = G + H
S → A	5	7	12
S → B	9	3	12
S → D	6	6	12
S → A → G	5+9=14	0	14
S → A → B	5+3=8	3	11
S → A → B → C	5+3+1=9	4	13
S → D → C	6+2=8	4	12
S → D → E	6+2=8	5	13
S → D → E	6+2=8	5	13
S → D → C → G	6+2+5=13	0	13
S → D → C → F	6+2+7=15	6	21



Visited: S(5), A(12), B(12), D(12), C(12)

A* Search - Example





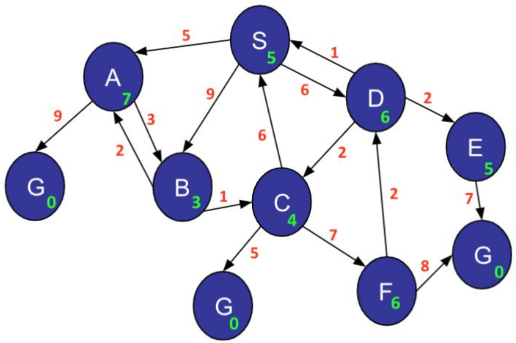
Visited: **S**(5), **A**(12), **B**(11), **D**(12), **C(12)**

Gcost: distance from starting node

Hcost: distance from starting node (Heuristic)

A* = Gcost + Hcost

Path	G	H	A* = G + H
S → A	5	7	12
S → B	9	3	12
S → C	6	6	12
S → A → G	5+9=14	0	14
S → A → B	5+3=8	3	11
S → A → B → C	5+3+1=9	4	13
S → D → C	6+2=8	4	12
S → D → E	6+2=8	5	13
S → D → C → G	**6+2+5=13**	**0**	**13**
S → D → C → F	**6+2+7=15**	**6**	**21**
S → D → C → F → G	**6+2+7=15**	**0**	**15**



A* Search - Example

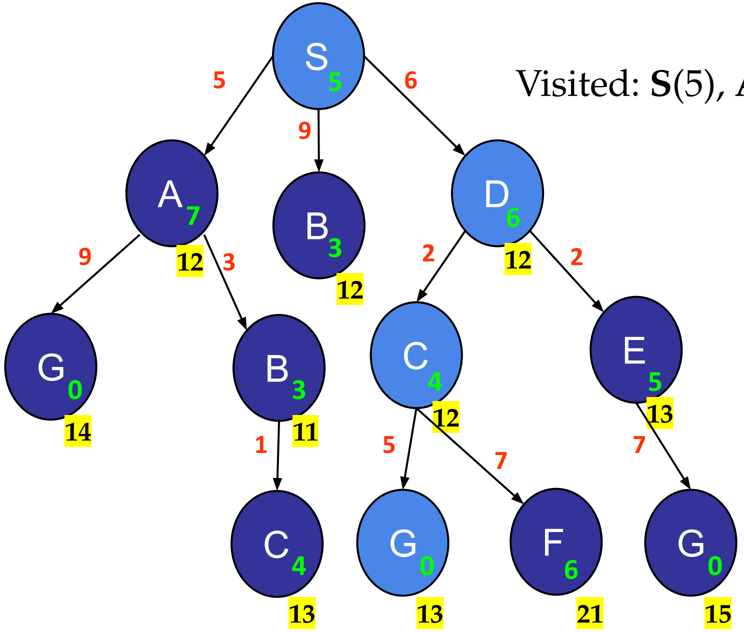
Gcost: distance from starting node

Hcost: distance from starting node (Heuristic)

A* = Gcost + Hcost

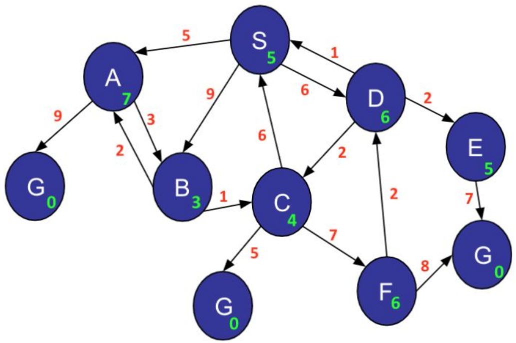
Path	G	H	A* = G + H
S → A	5	7	12
S → B	9	3	12
S → C	6	6	12
S → A → G	5+9=14	0	14
S → A → B	5+3=8	3	11
S → A → B → C	5+3+1=9	4	13
S → D → C	6+2=8	4	12
S → D → E	6+2=8	5	13
S → D → C → G	**6+2+5=13**	**0**	**13**
S → D → C → F	**6+2+7=15**	**6**	**21**
S → D → C → F → G	**6+2+7=15**	**0**	**15**
S → D → C → G	**6+2+5=13**	**0**	**13**

Visited: **S**(5), **A**(12), **B**(11), **D**(12), **C**(12), E(13)



GOAL!

Comparing G=14, **G=13**, and G=15, the best was 13.



Exercise

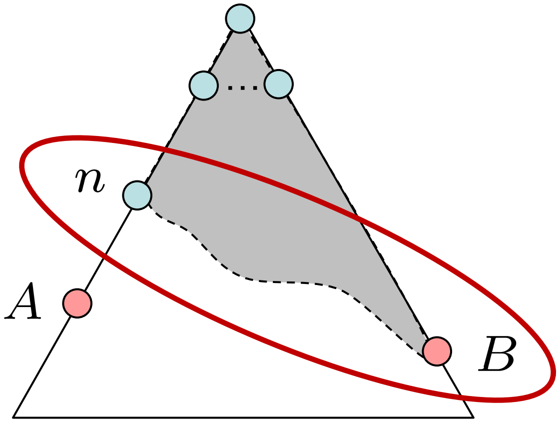
Comparison: DFS, BFS, UCS, Greedy and A*

	DFS	**BFS**	**UCS**	**Greedy**	**A***
Complete					
Optimal					
Time					
Space					
Path					

Optimality of A* Tree Search: Blocking

Proof:

- Imagine B is on the fringe
- Some ancestor *n* of A is on the fringe, too (maybe A!)
- Claim: *n* will be expanded before B
  1. f(n) is less or equal to f(A)
  2. f(A) is less than f(B)
  3. *n* expands before B
- All ancestors of A expand before B
- A expands before B
- A* search is optimal



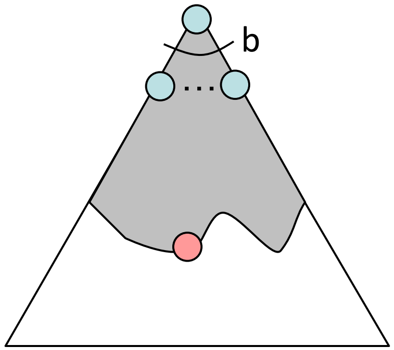
$$f(n) \leq f(A) < f(B)$$

[This slide was adapted from Dan Klein and Pieter Abbeel at UC Berkeley]

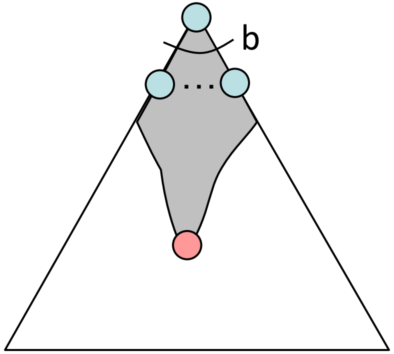
Properties of A*

Properties of A*

Uniform-Cost



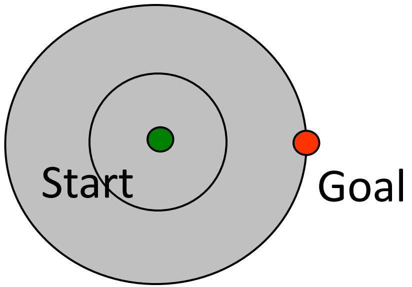
A*



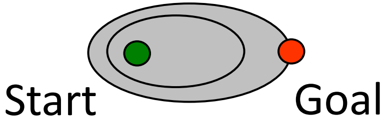
[This slide was adapted from Dan Klein and Pieter Abbeel at UC Berkeley]

UCS vs A* Contours

- Uniform-cost expands equally in all "directions"

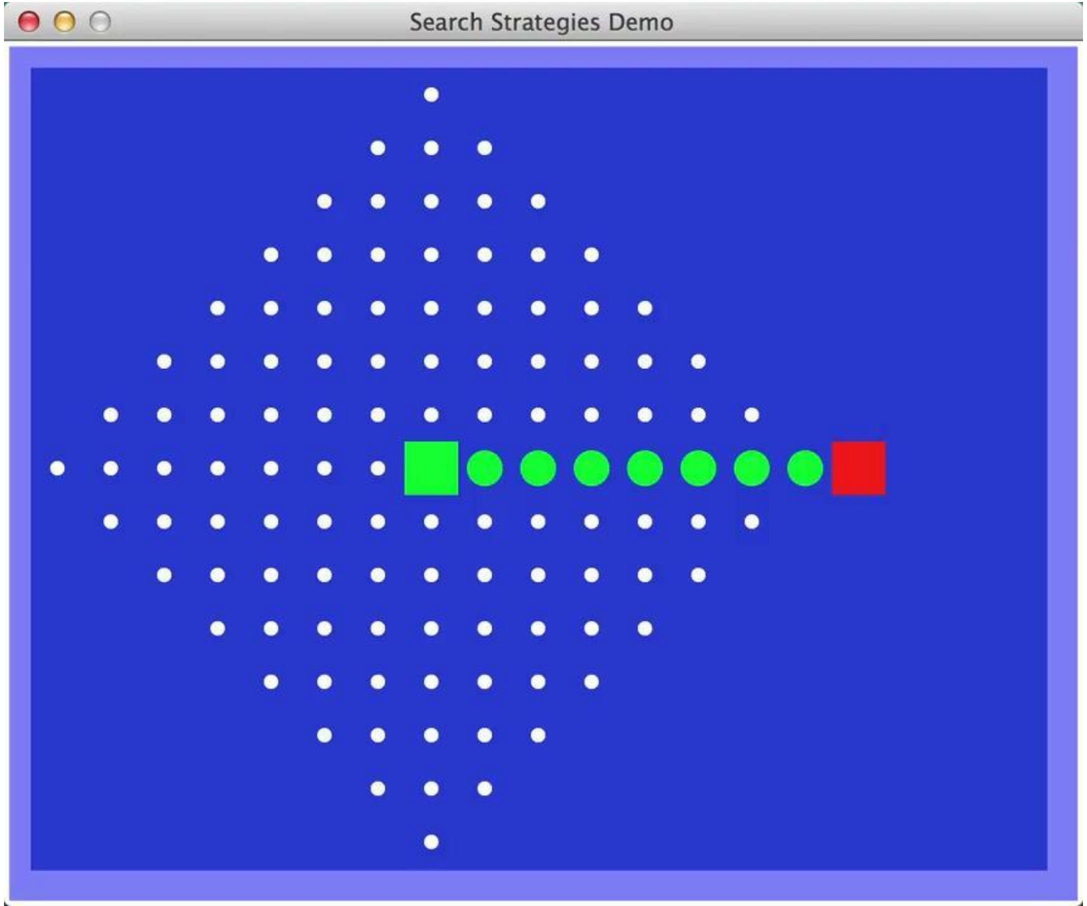


- A* expands mainly toward the goal, but does hedge its bets to ensure optimality

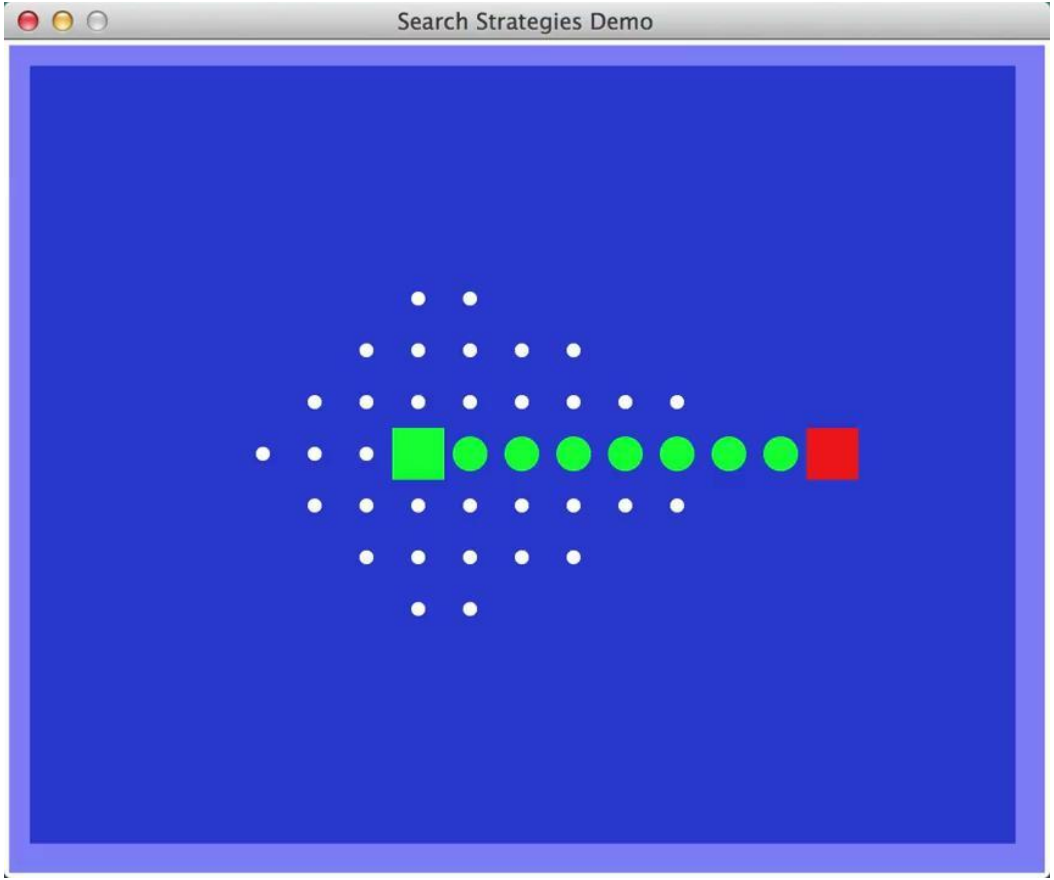


Greedy Search

UCS Search



A* Search (UCS + Greedy / 2)



The **BEST** part

Challenge

- Encontrar um problema real em algum segmento de atuação (ex: segurança, saúde, transporte, pecuária, etc.), e aplicar (implementar em um contexto real e simples) alguma técnica de busca aprendida (informada ou não informada).

Entrega: 03/06/2025