ČVUT
v Praze
in Prague

State space search

A* Algorithm and way to it via Breath-first search and Dijkstra algorithms

Václav Hlaváč

Czech Technical University in Prague (ČVUT)

Czech Institute of Informatics, Robotics, and Cybernetics (CIIRC)

Prague 6, Zikova 4, Czech Republic

hlavac@ciirc.cvut.cz
http://people.ciirc.cvut.cz/hlavac/

Courtesy: Antonín Vobecký and authors of several presentation on the web

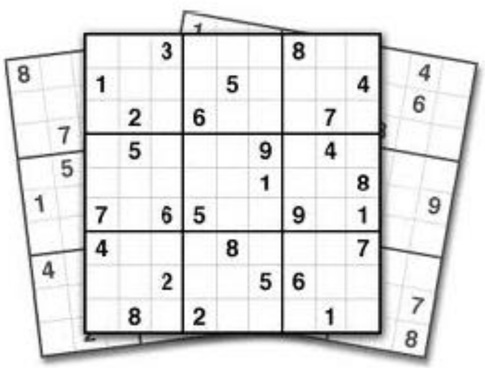
Motivation

- Many analytical tasks can be solved by searching through a space of possible states.
- Starting from an initial state, we try reaching a goal state.
- Sequence of actions leading from initial to goal state is the solution to the problem.
- The issues: large number of states a
many choices to make in each state.
- Search has to be performed in a systematic manner.

Typical search tasks

ČVUT
v Praze
'n Prague



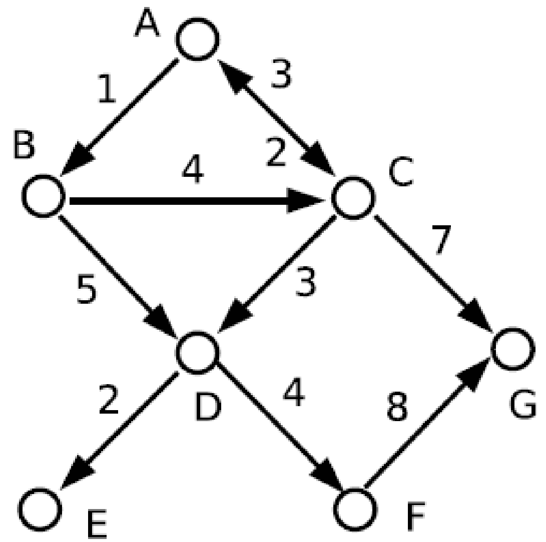



State space search, the basic idea

ČVUT
v Praze
in Prague

- State space search amounts to a search through a directed graph.
  - graph nodes = states
  - arcs (directed edges) = transitions between states.
- Graph may be defined explicitly or implicitly.
- Graph may contain cycles.

- If we also need the transition costs, we work with a weighted directed graph.



Size of the search space

ČVUT
v Praze
in Prague

- The state space can be HUGE! (Combinatorial explosion)
- Right representation helps.
  - Eight puzzle: 181,440
  - Draughts / Checkers / *in Czech dáma*: 10^{40}
  - Chess: 10^{120} (in an average length game)
  - Theorem Proving: Infinite!
- Control strategy helps choose which operators to apply:
  - Small # of operators: general, but bushy tree.
  - Large #: perhaps overly specific, but less bushy trees.

Search tree

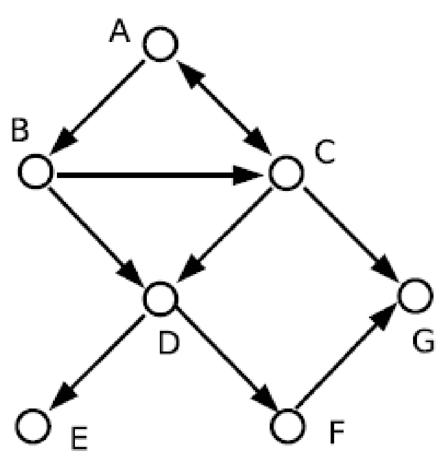
ČVUT
v Praze
in Prague

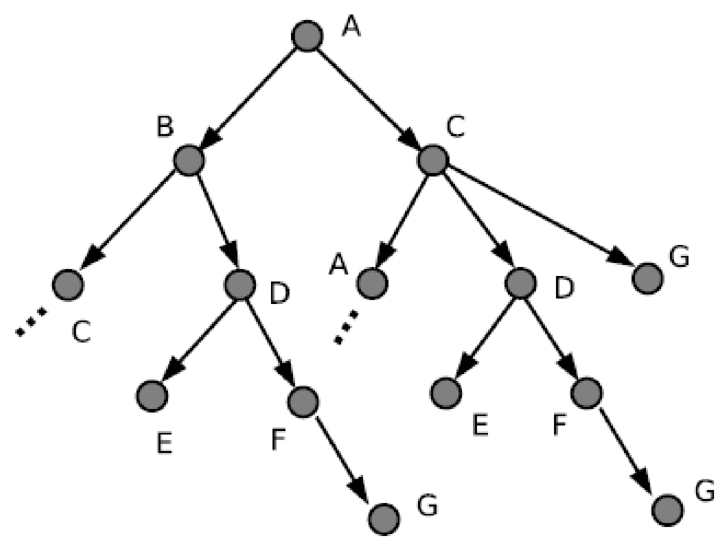
- By searching through a directed graph, we gradually construct a search tree.
- We do this by expanding one node after the other: we use the successor function to generate the descendants of each node.
- Open nodes or “the frontier”: nodes that have been generated, but have not yet been expanded.
- Closed nodes: already expanded nodes.
- Search strategy is defined by the order in which the nodes are expanded. Different orders yield different strategies.

State space vs. search tree

ČVUT
v Praze
in Prague

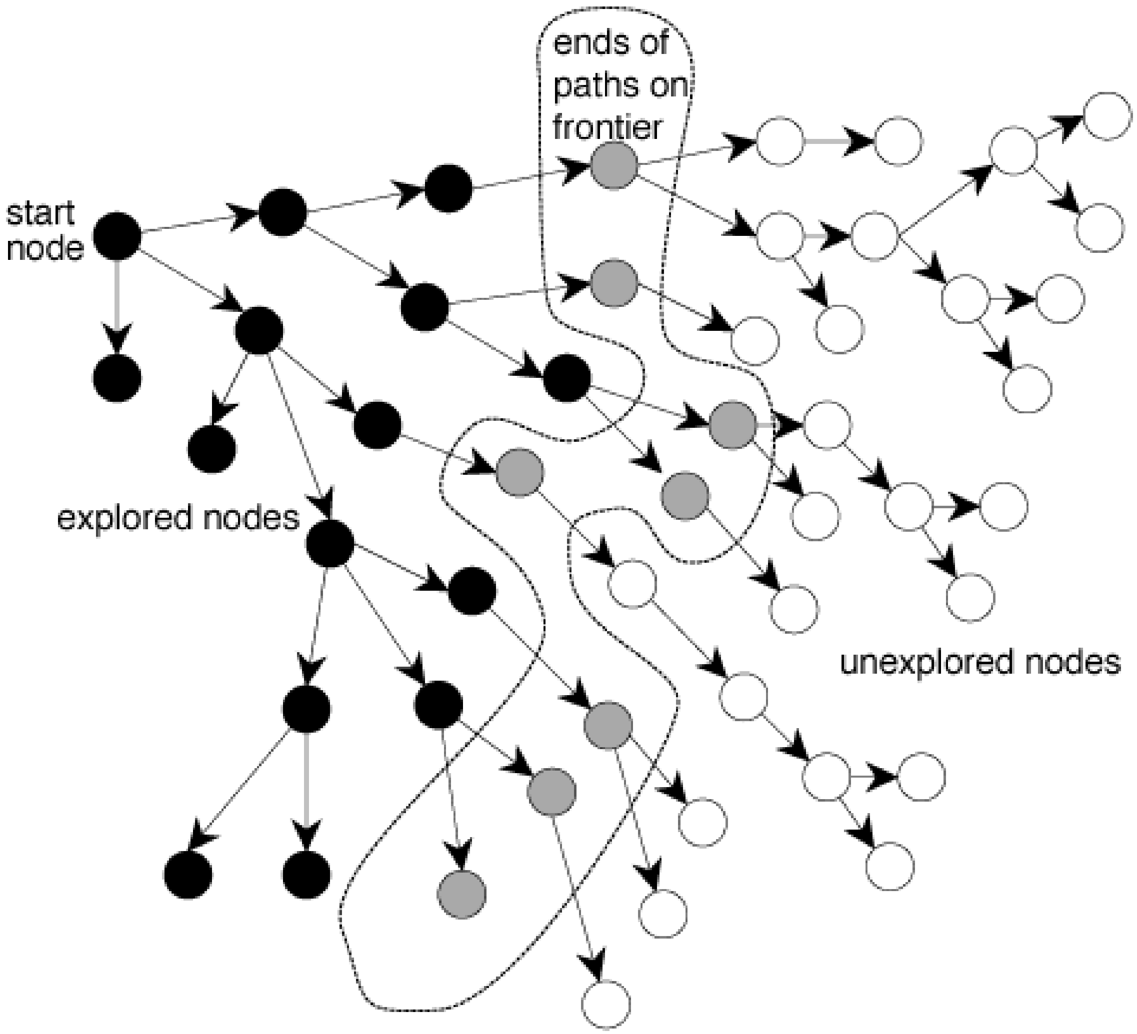
- Search tree is created while searching through the state space.
- Search tree can be infinite even if the state space is finite. E.g. if the state space contains cycles → search tree is infinite.





Open nodes, pictorial illustration

ČVUT
v Praze
in Prague



The basic search algorithm

ČVUT
v Praze
in Prague

```
Initialize: put the start node into OPEN
while OPEN is not empty
    take a node N from OPEN
    if N is a goal node, report success
    put the children of N onto OPEN
Report failure
```

- If OPEN is a stack, this is a depth-first search.
- If OPEN is a queue, this is a breadth-first search.
- If OPEN is a *priority queue*, sorted according to *most promising first*, we have a best-first search (Dijkstra algorithm).

Breadth-first search

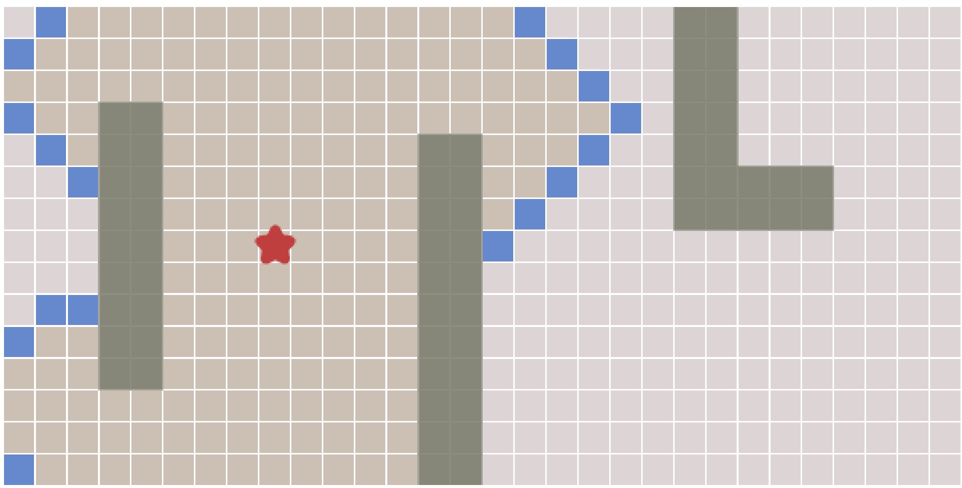
ČVUT
v Praze
in Prague

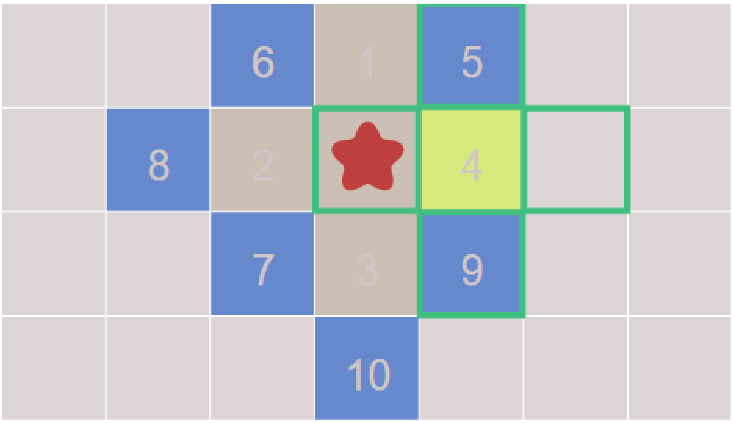
(abbrev. BFS)

Implementation:

- Pick and remove a location from the OPEN (frontier).
- Mark the location as visited so that we know not to process it again.
- Expand it by looking at its neighbors. Any neighbors we haven't seen yet we add to the frontier.

Breadth-first search (2)





Breadth-first search (3)

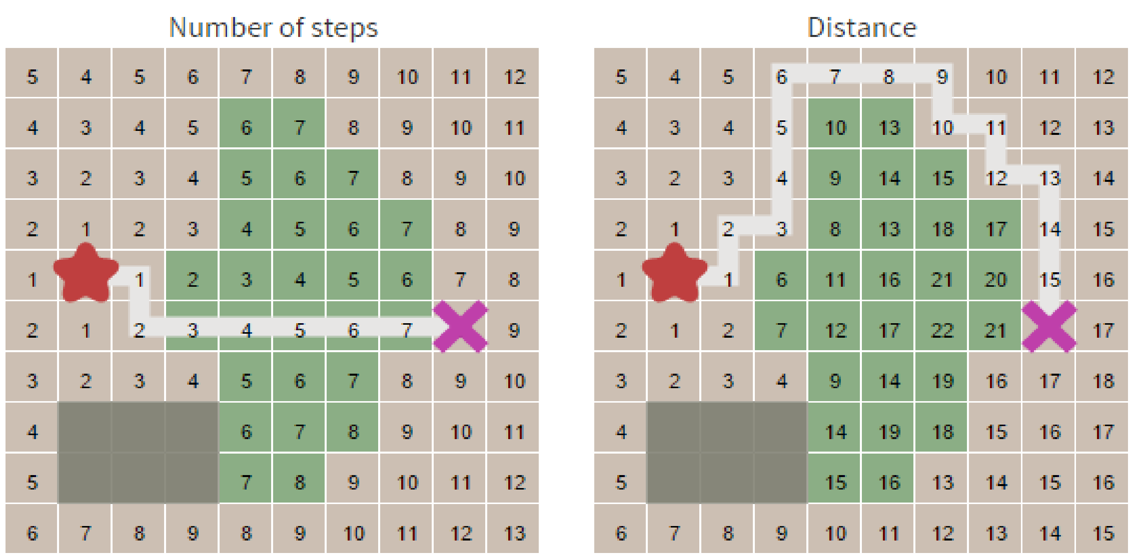
ČVUT
v Praze
in Prague

- visits all reachable places
- efficiency:
  - time: $O(b^d)$
  - space: $O(b^d)$
  - b=branching factor, d=depth of goal
- no priority
- possible improvements:
  - **early exit** = search stops when the goal is reached
  - movement cost → Dijkstra algorithm

Dijkstra algorithm

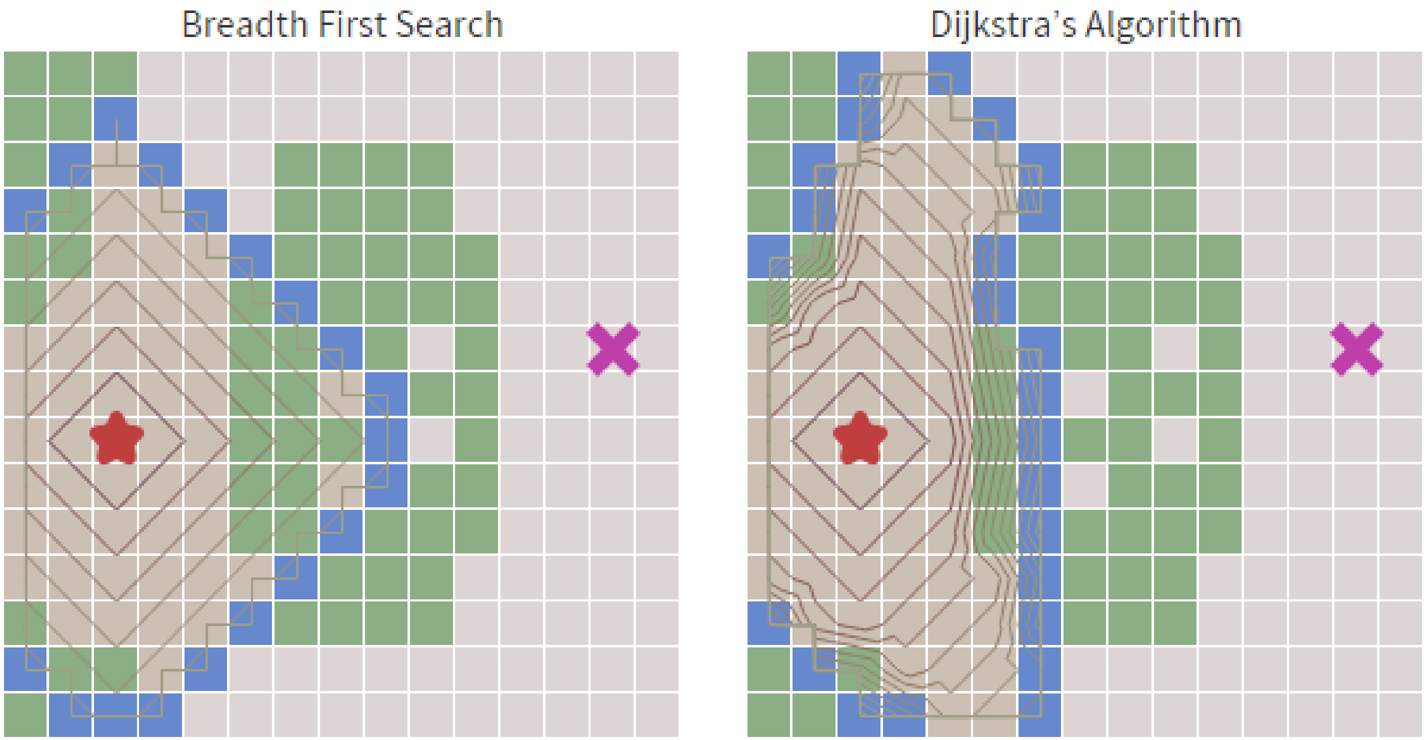
ČVUT
v Praze
in Prague

- Adding movement cost to Breath-first search algorithm, expands in all directions
- Using priority queue
  - Choosing move with the lowest cost
- Time efficiency: $O(|E|+|V|\ log|V|)$, V=number of nodes, E=number of edges



Dijkstra algorithm vs. BFS

ČVUT
v Praze
in Prague



Greedy best first search

ČVUT
v Praze
in Prague

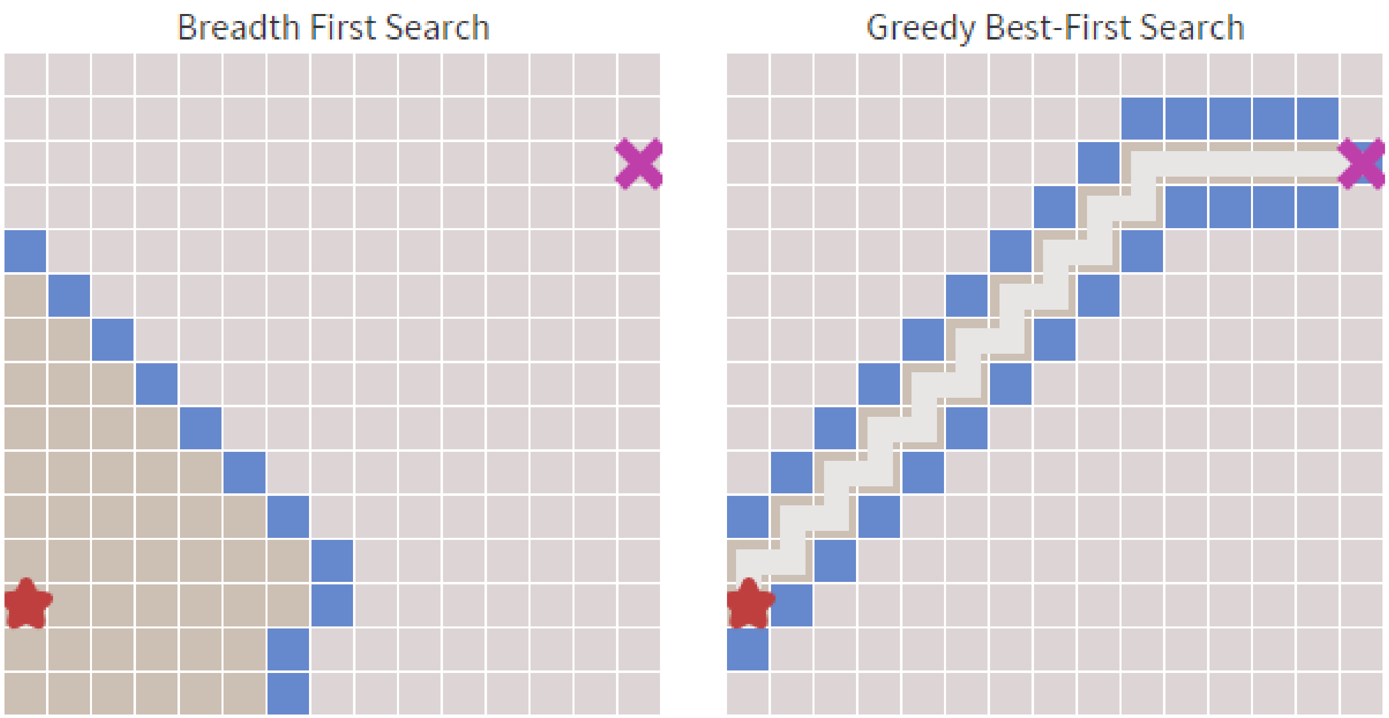
- better for finding path to one exact location
- use of heuristics:
  - distance to the goal
  - e.g.:

```
def heuristics(a,b):
        return abs(a.x - b.x) + abs(a.y + b.y)
```

- time/space efficiency: $O(b^m)$
  - good heuristics can give huge improvements
- priority queue
  - priority = distance to goal

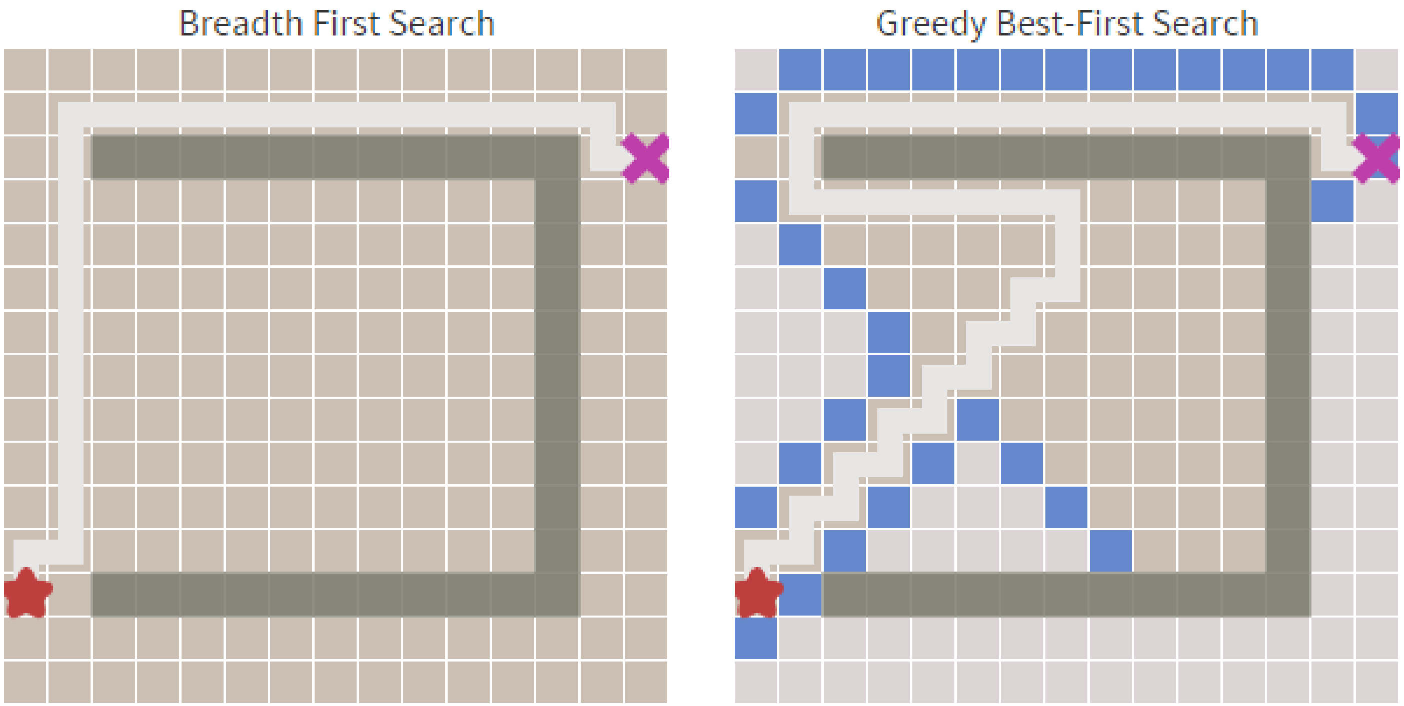
Greedy best-first search - examples

ČVUT
v Praze
in Prague



Greedy best-first search - examples

- Problem with obstacles.
- May not find the shortest path.

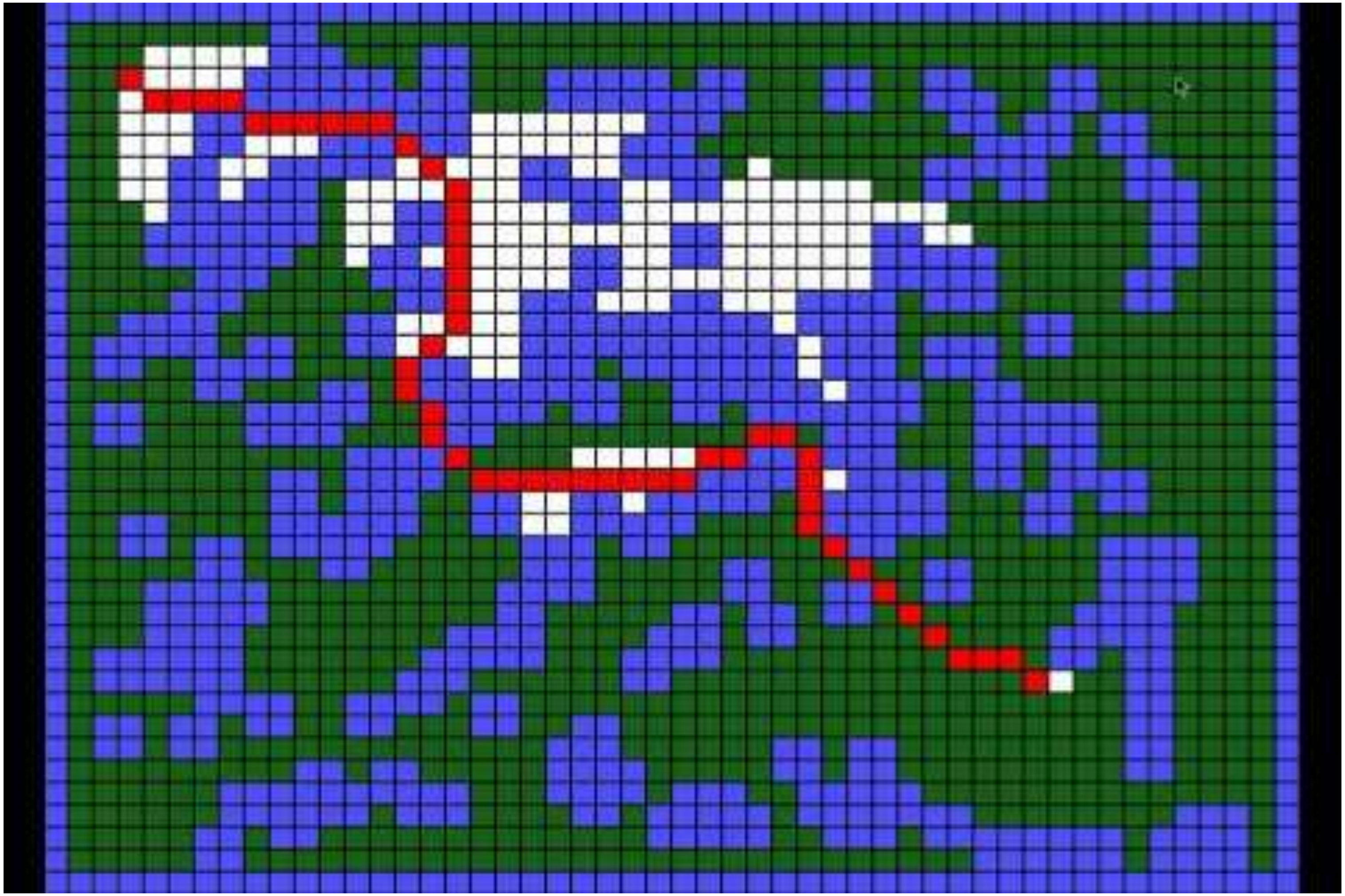


A* algorithm (read “A star”)

ČVUT
v Praze
in Prague

- Using the best of both Djikstra and Greedy algorithms, worst time/space: $O(b^d)$
- Expanding based on:
  - distance from start
  - distance to goal (=heuristics)

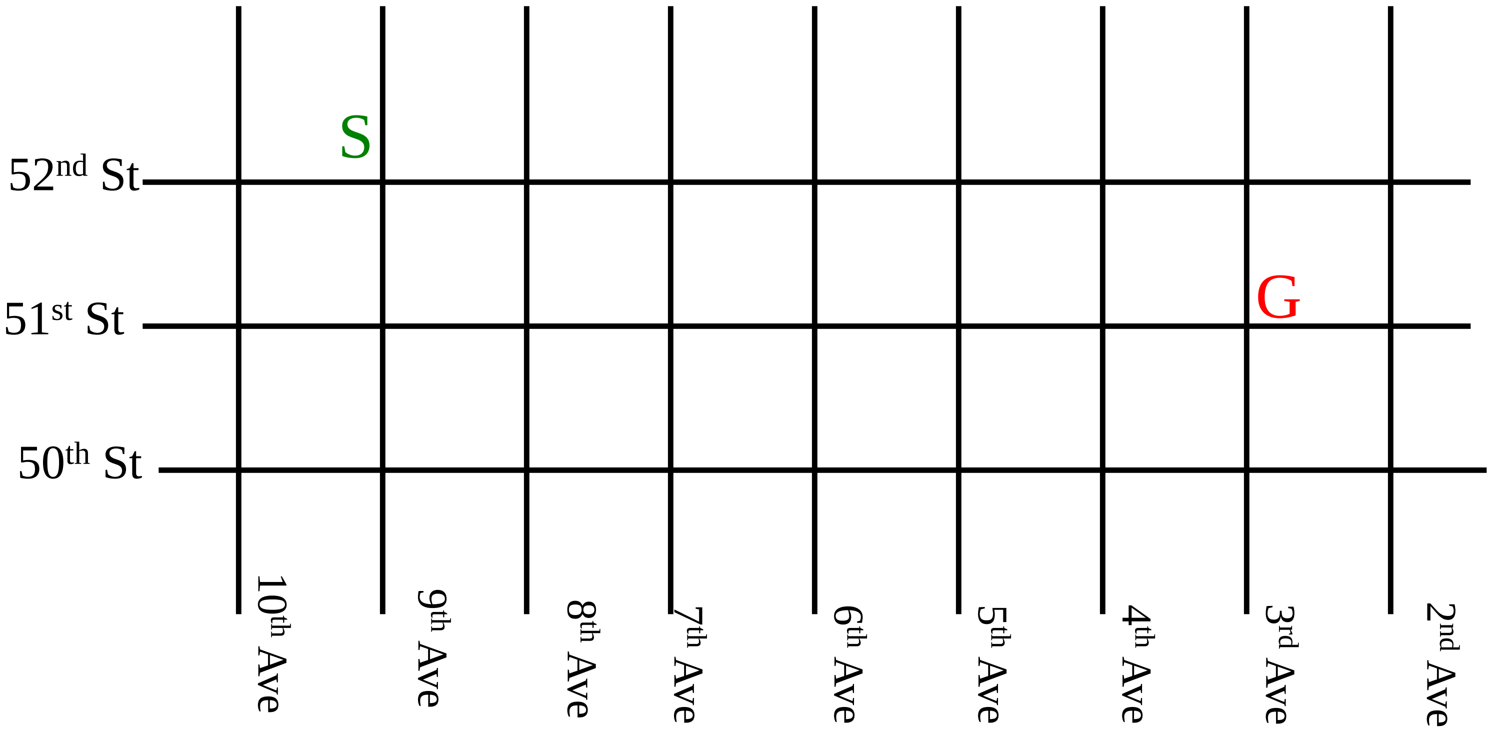
A* algorithm



Map of Manhattan

ČVUT
v Praze
in Prague

- How would you find a path from S to G?



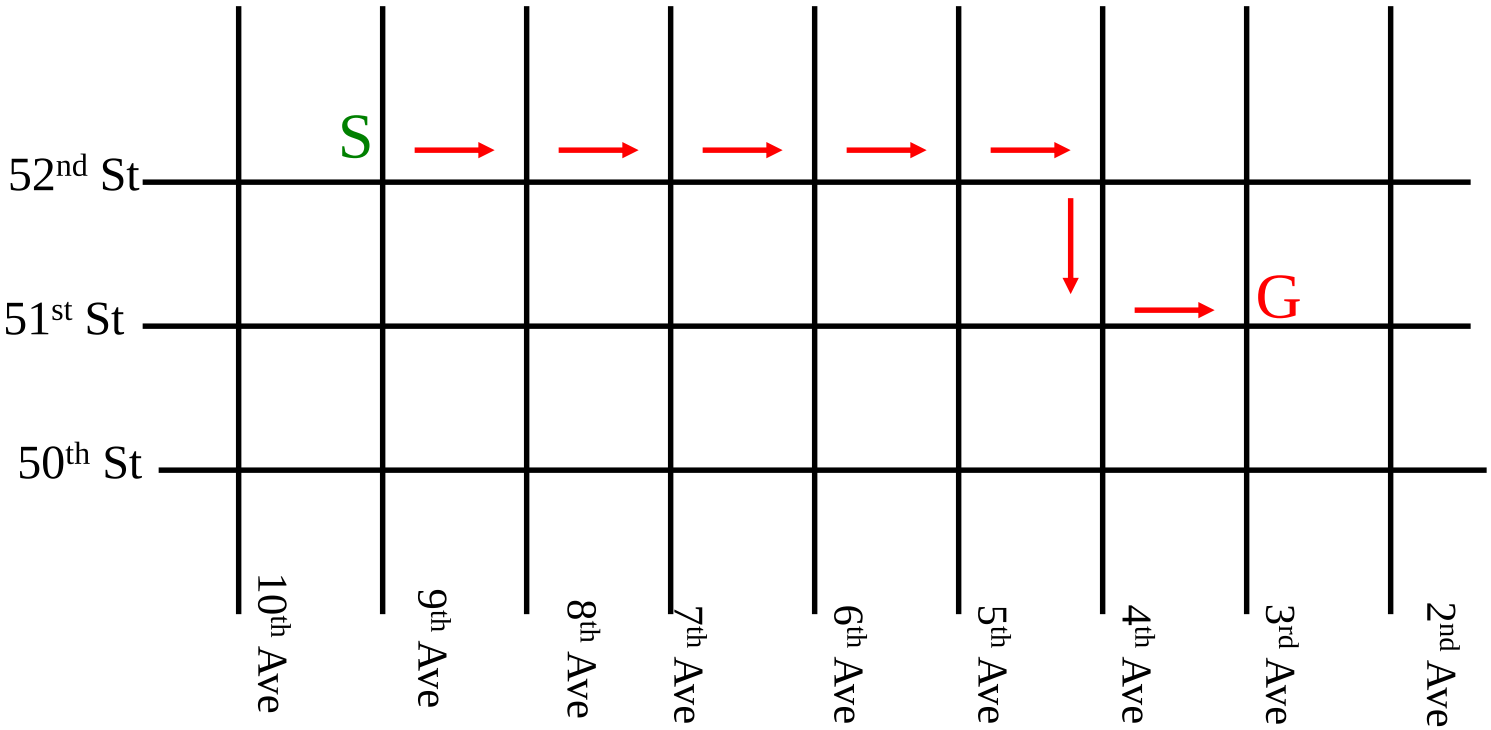
Best-First Search

ČVUT
v Praze
in Prague

- The *Manhattan distance* ($\Delta x + \Delta y$) is an estimate of the distance to the goal
  - It is a heuristic function
- Best-First Search
  - Order nodes in priority queue to minimize estimated distance to the goal h(n)
- Compare: Dijkstra
  - Order nodes in priority queue to minimize distance from the start

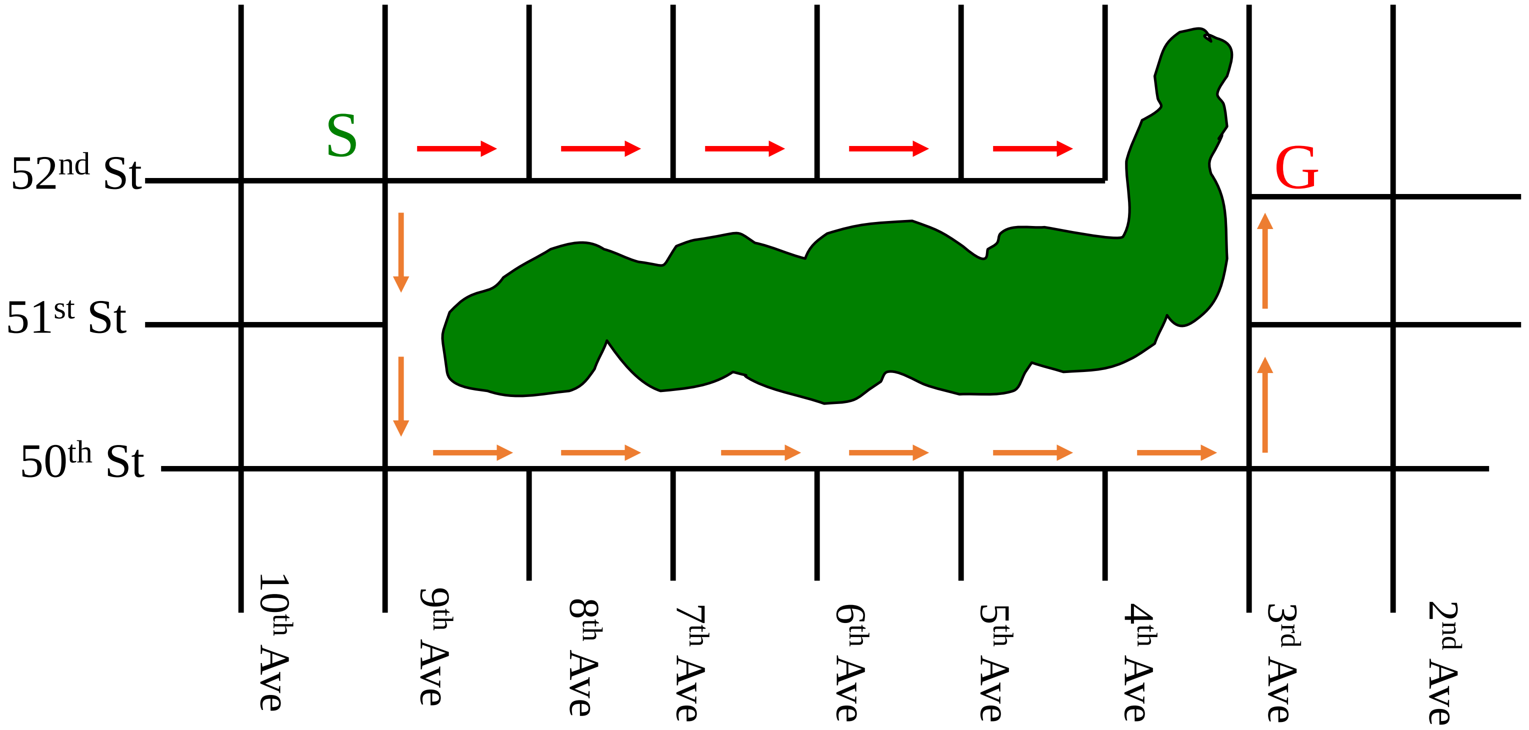
Best First in action

- How would you find a path from S to G?



Problem 1: Led astray

- Eventually will expand vertex to get back on the right track



ČVUT
v Praze
in Prague

Problem 2: Optimality

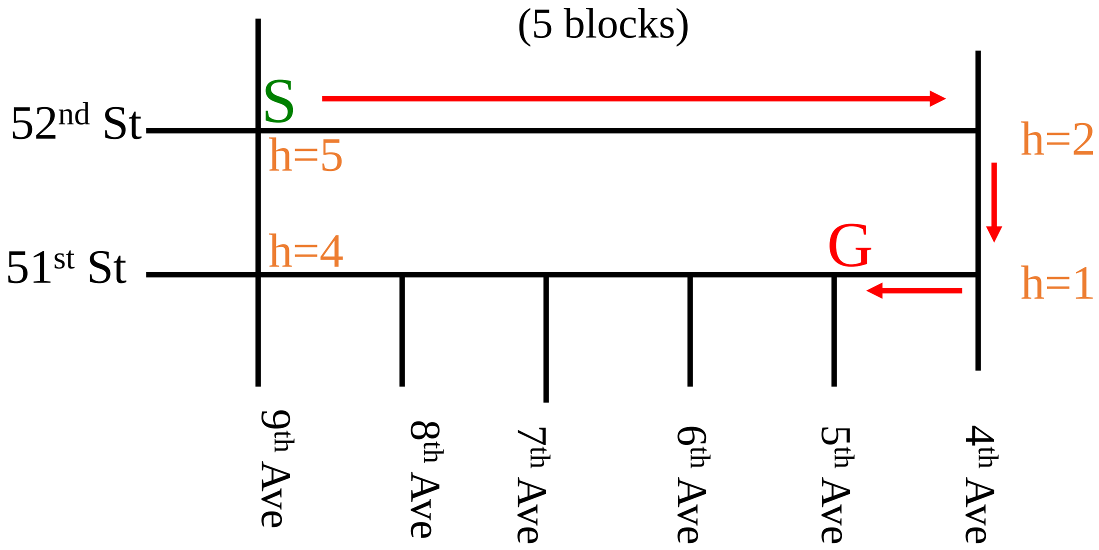
ČVUT
v Praze
in Prague

- With Best-first search, are you *guaranteed* a shortest path is found when
  - goal is first seen?
  - when goal is removed from priority queue (as with Dijkstra?)

Sub-optimal solution

ČVUT
v Praze
in Prague

- No! Goal is by definition at distance 0: will be removed from priority queue immediately, even if a shorter path exists!



Synergy?

- Dijkstra / Breadth First guaranteed to find *optimal* solution
- Best First often visits *far fewer* vertices, but may not provide optimal solution

  - *Can we get the best of both?*

A*, heuristics

ČVUT
v Praze
in Prague

Order vertices in priority queue to minimize

(distance from start) + (estimated distance to goal)

$$f(n) = g(n) + h(n)$$

$f(n)$ = priority of a node

$g(n)$ = true distance from start

$h(n)$ = heuristic distance to goal

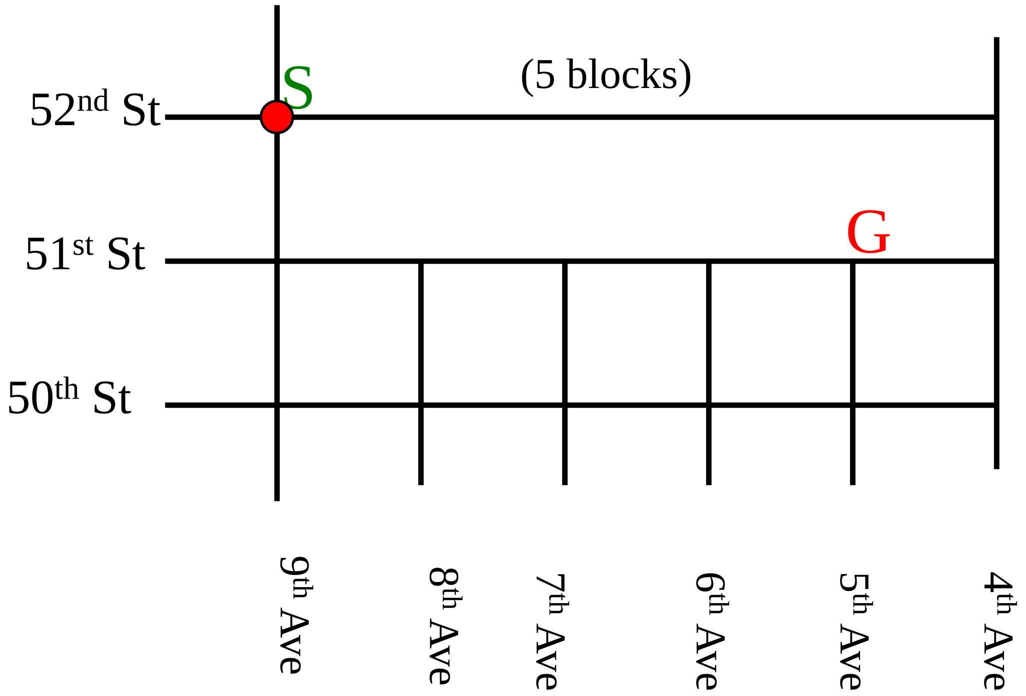
Optimality

- Suppose the estimated distance (h) is *always* less than or equal to the true distance to the goal
  - heuristic is a *lower bound on true distance*
  - *heuristic is admissible*

- Then: when the goal is removed from the priority queue, we are guaranteed to have found a shortest path!

A* in action

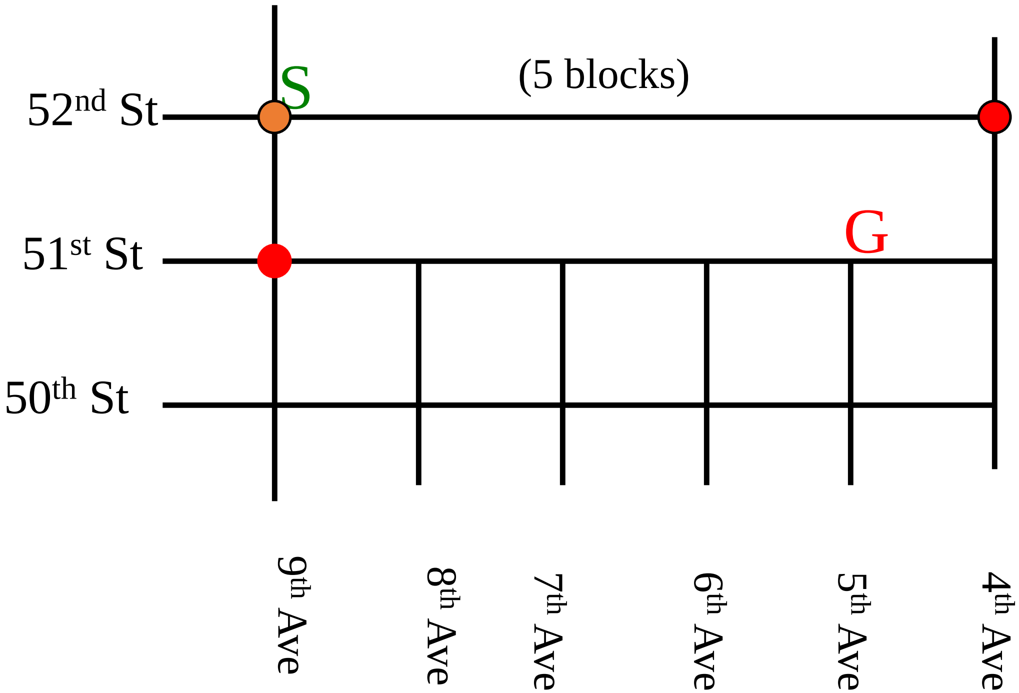
ČVUT
v Praze
in Prague



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

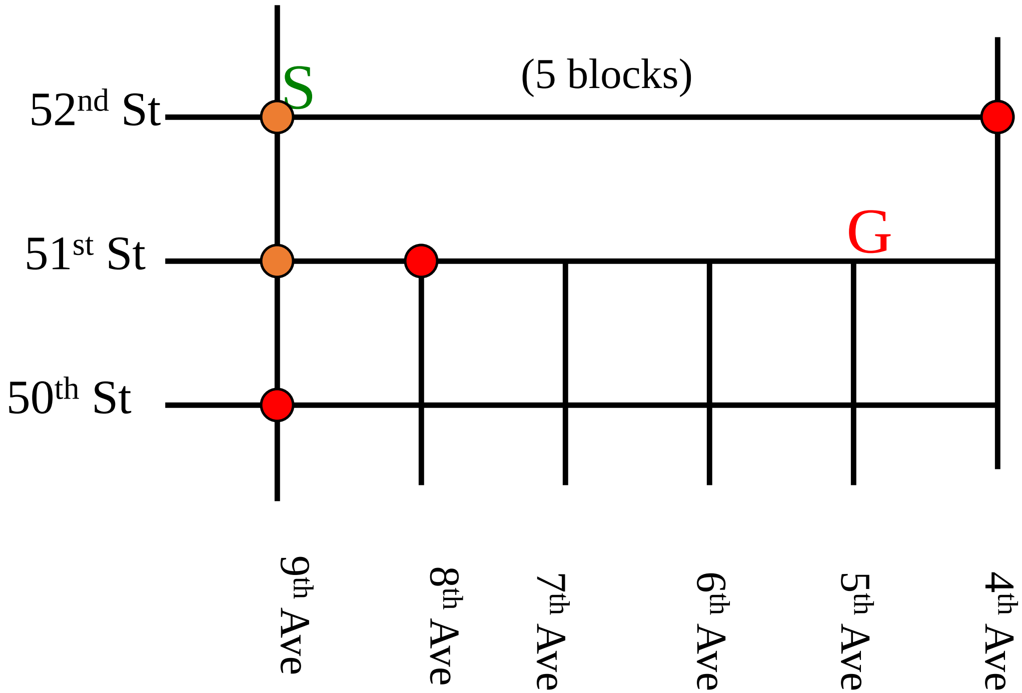
A* in action

ČVUT
v Praze
in Prague



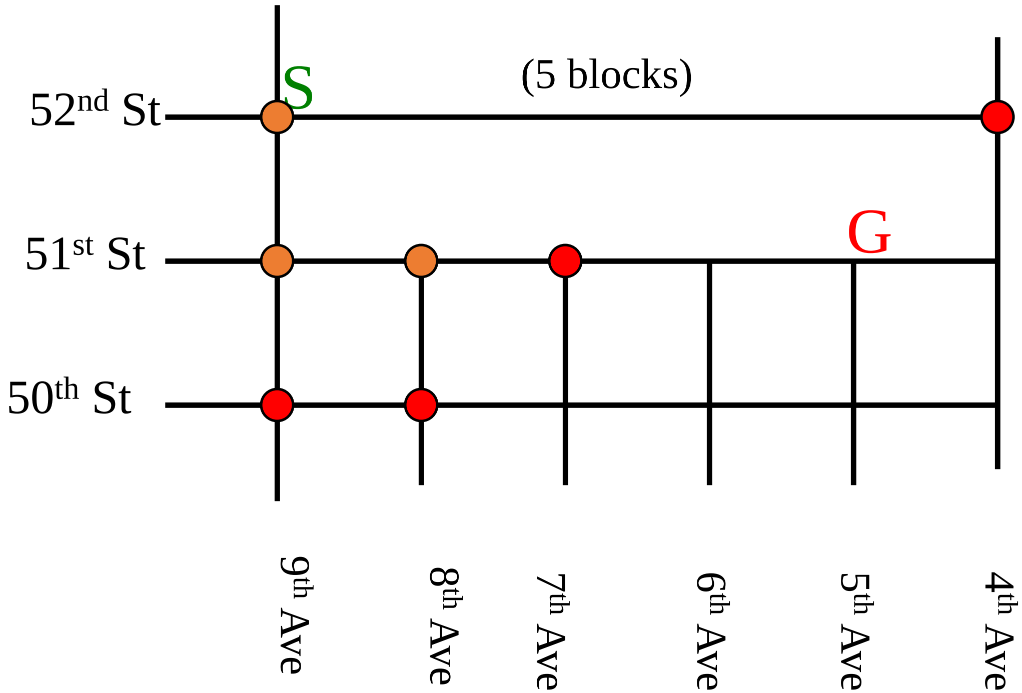
vertex	g(n)	h(n)	f(n)
52nd & 4th	5	2	7
51st & 9th	1	4	5

A* in action



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

A* in action

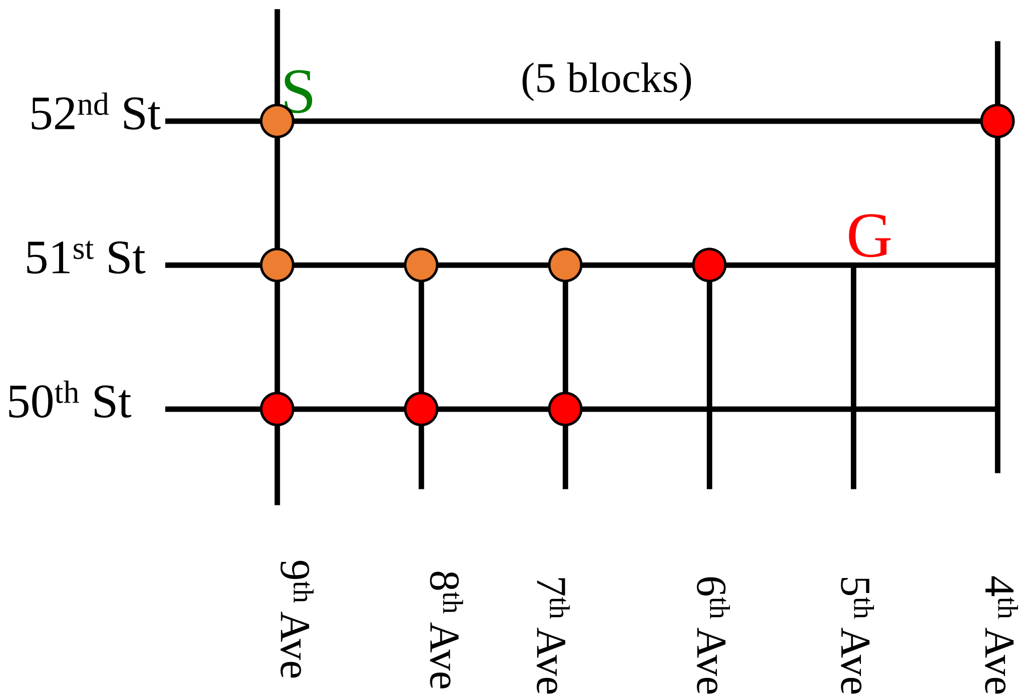


vertex	g(n)	h(n)	f(n)
52nd & 4th	5	2	7
51st & 7th	3	2	5
50th & 9th	2	5	7
50th & 8th	3	4	7

ČVUT
v Praze
in Prague

A* in action

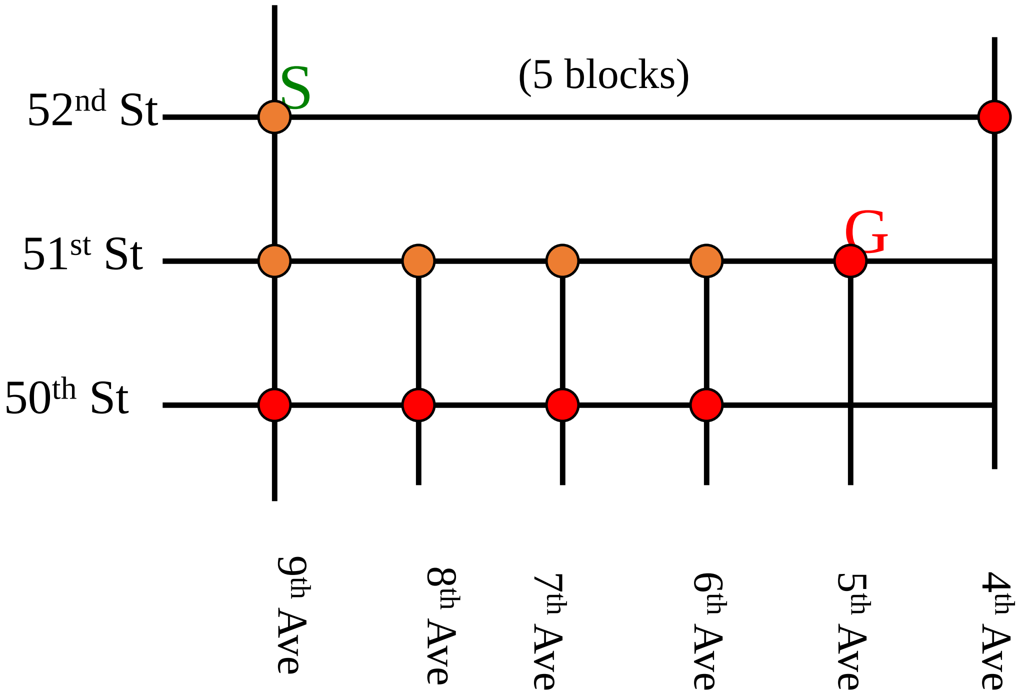
ČVUT
v Praze
in Prague



vertex	g(n)	h(n)	f(n)
52nd & 4th	5	2	7
51st & 6th	4	1	5
50th & 9th	2	5	7
50th & 8th	3	4	7
50th & 7th	4	3	7

A* in action

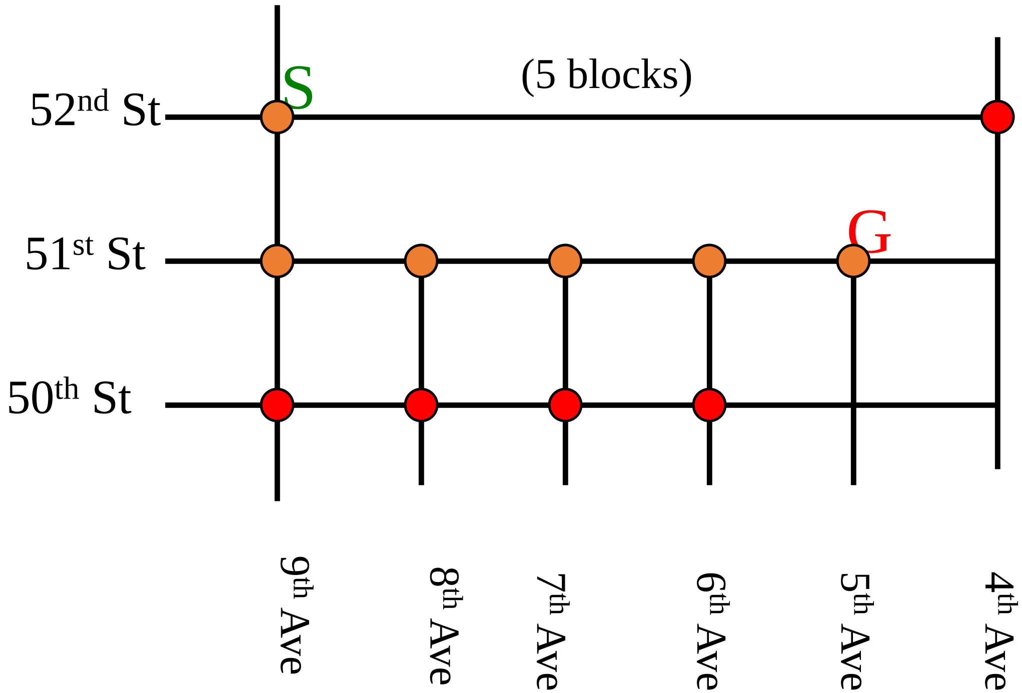
ČVUT
v Praze
in Prague



vertex	g(n)	h(n)	f(n)
52nd & 4th	5	2	7
51st & 5th	5	0	5
50th & 9th	2	5	7
50th & 8th	3	4	7
50th & 7th	4	3	7

A* in action

ČVUT
v Praze
in Prague

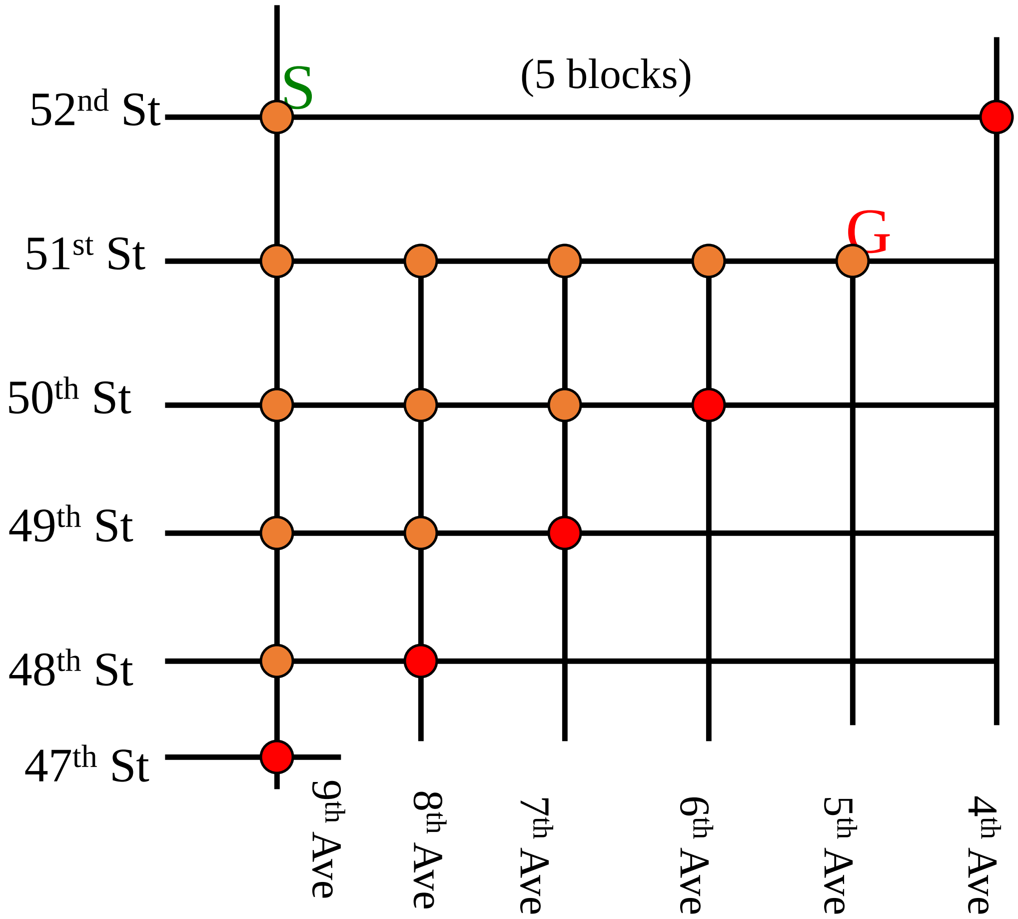


vertex	g(n)	h(n)	f(n)
52nd & 4th	5	2	7
50th & 9th	2	5	7
50th & 8th	3	4	7
50th & 7th	4	3	7

DONE!

What would Dijkstra have done?

ČVUT
v Praze
in Prague



Importance of Heuristics

ČVUT
v Praze
in Prague

- h1 = number of tiles in the wrong place
- h2 = sum of distances of tiles from correct location

D	IDS	A*(h1)	A*(h2)
2	10	6	6
4	112	13	12
6	680	20	18
8	6384	39	25
10	47127	93	39
12	364404	227	73
14	3473941	539	113
18		3056	363
24		39135	1641

Summary

ČVUT
v Praze
in Prague

Finding path to ALL locations:

- Same cost
  → Breadth-first search
- Costs vary →
  Dijkstra algorithm

Finding path to ONE location:

- Preferably use A* algorithm