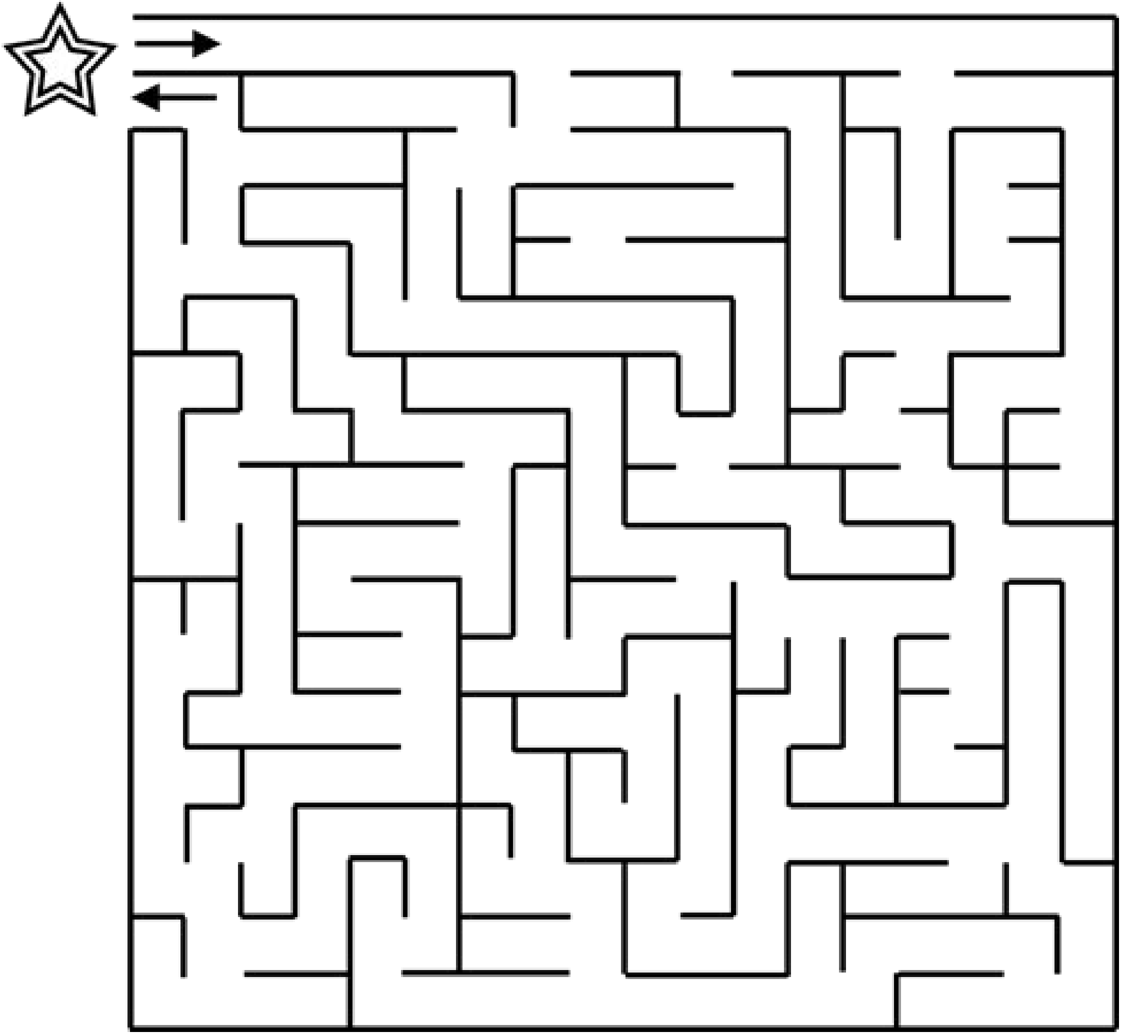
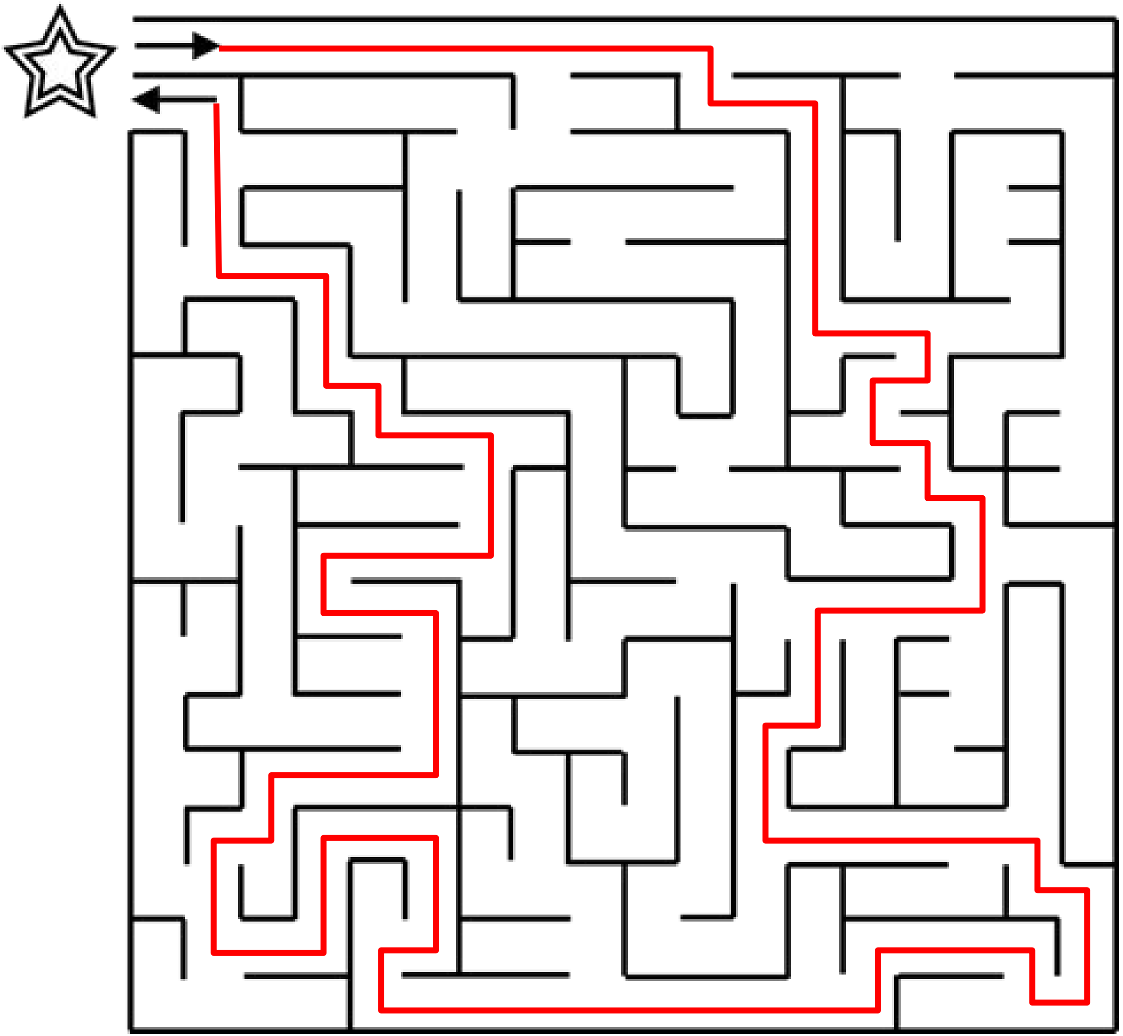
Graph Search Algorithms

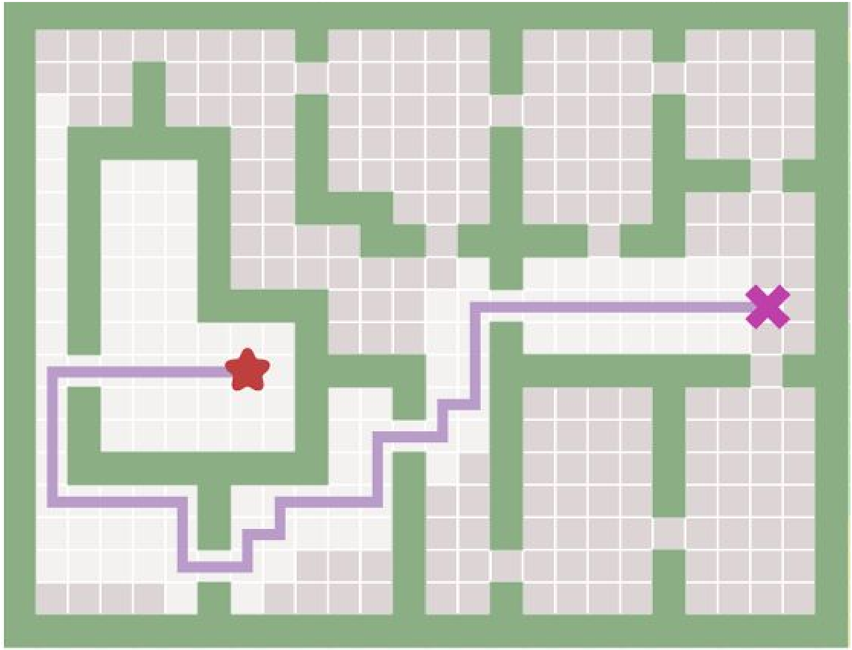
Steve Mussmann and Abi See



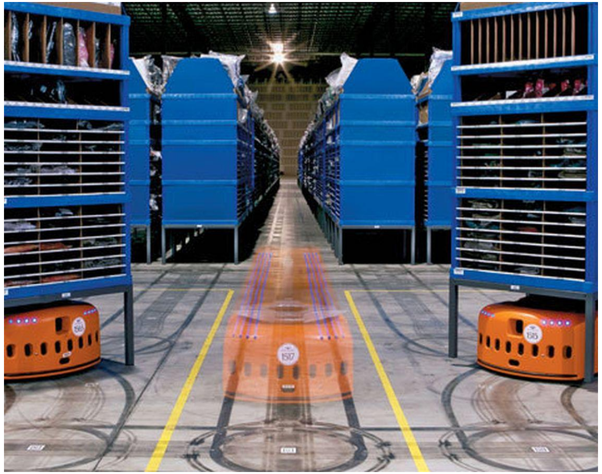


Shortest Path Problems

Find the shortest path from source ★ to target ✕

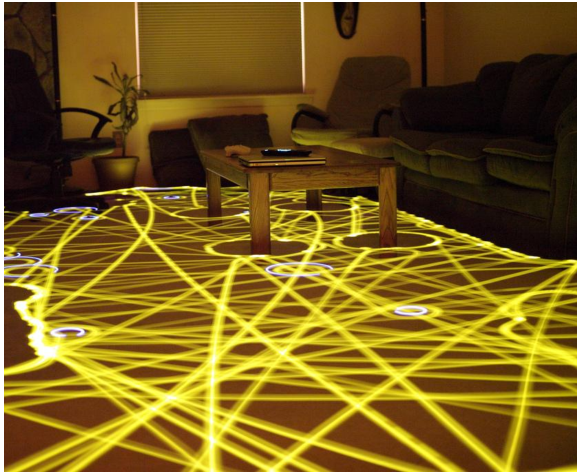


Applications: Robotics



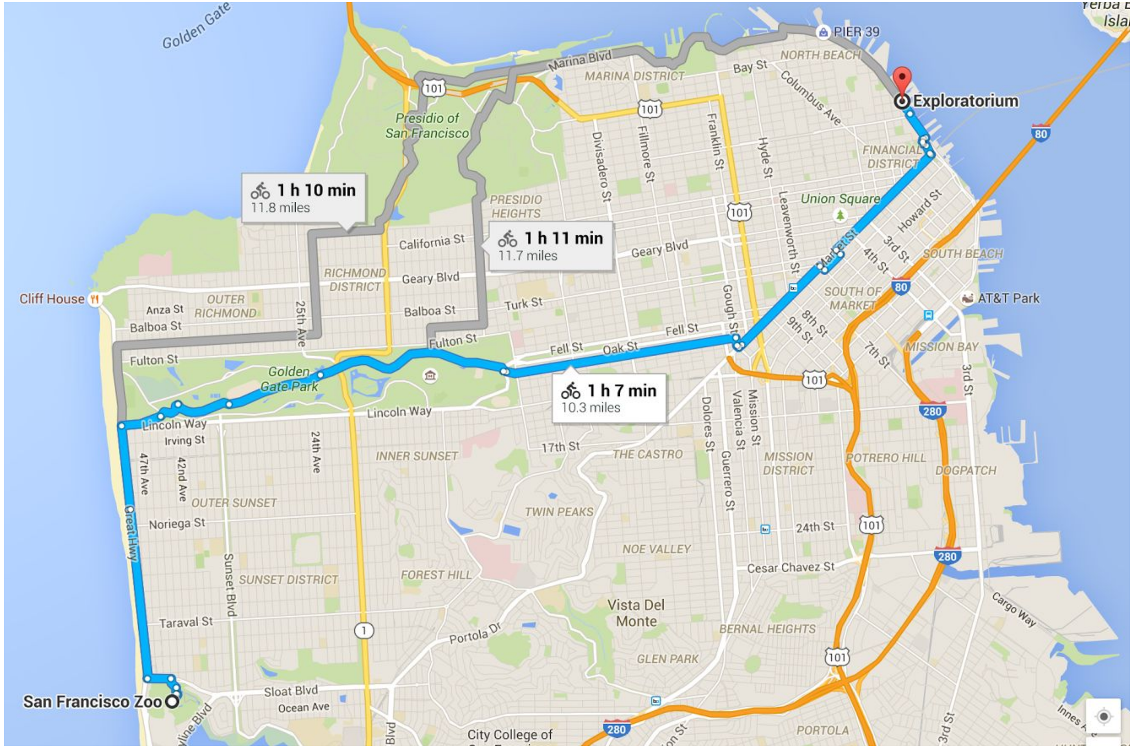
Commercial

Search & Rescue

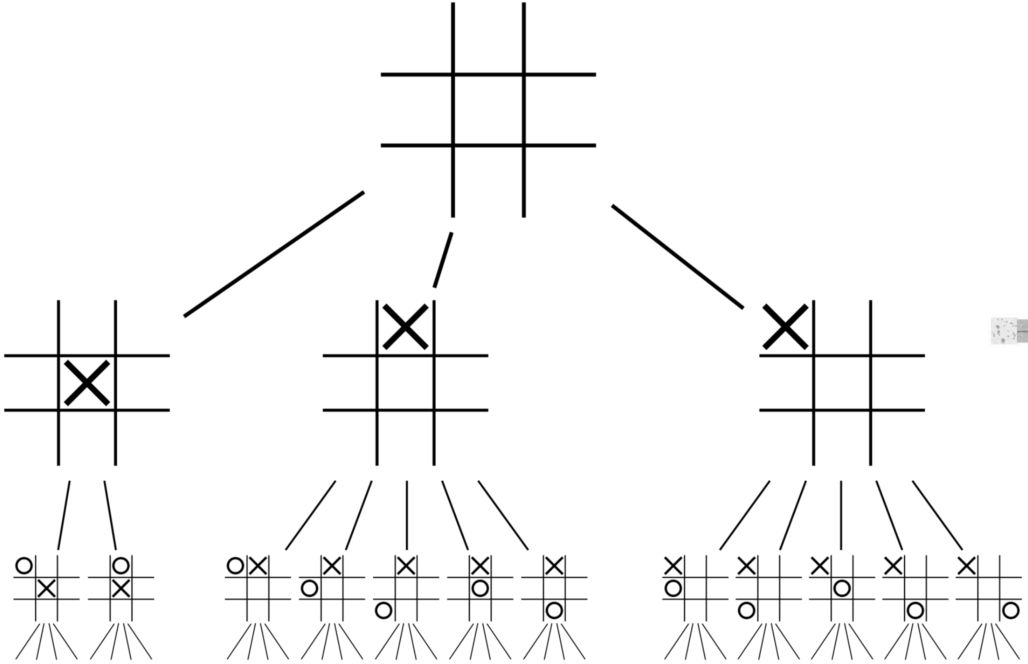


Domestic

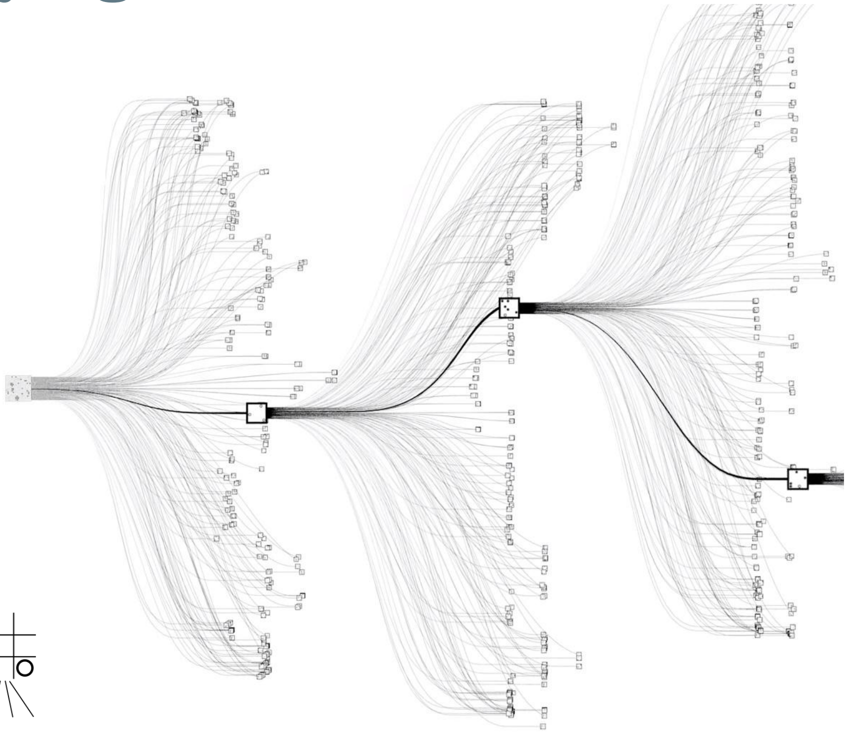
Applications: Route-Planning



Applications: Game-playing



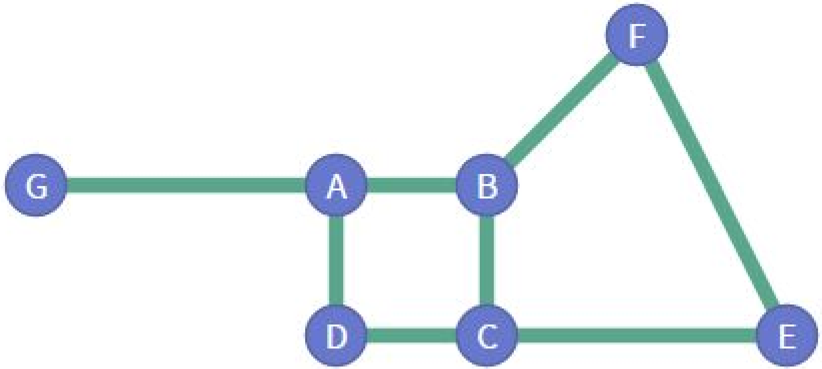
Tic-tac-toe



Go

Graphs

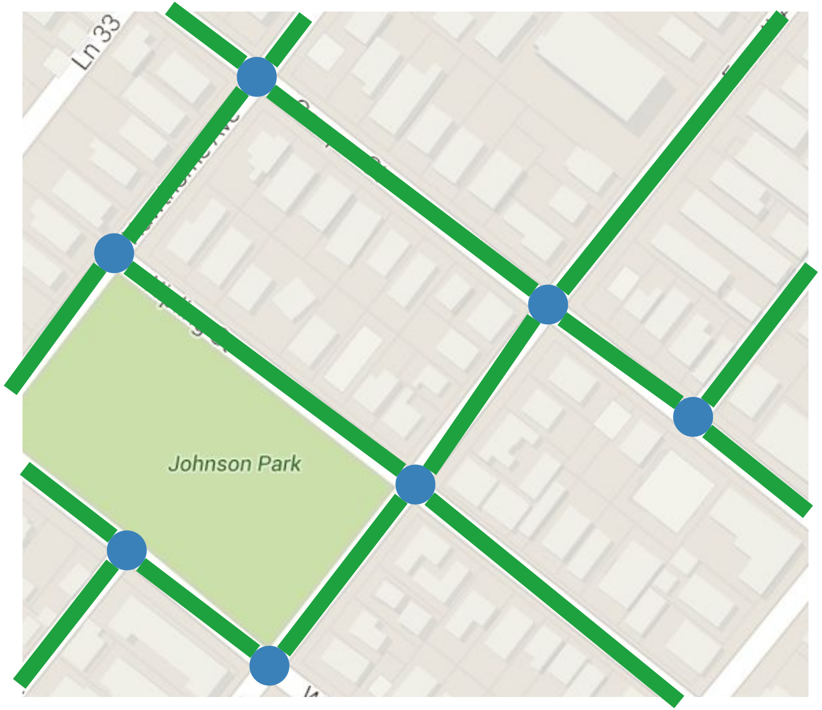
Graphs have nodes and edges.



How many nodes are there?
How many edges?

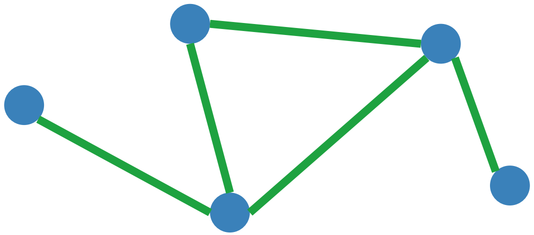
Graphs

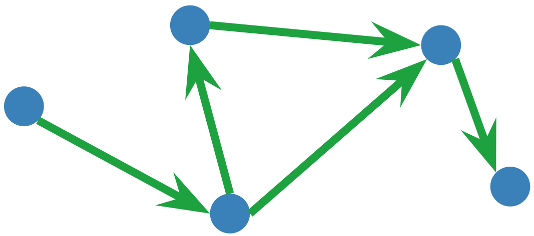
We cast real-world problems as graphs.



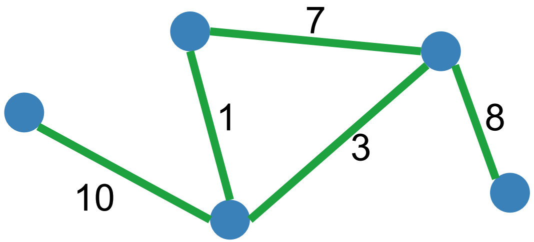
Graphs

Graphs can be *undirected* or *directed*.

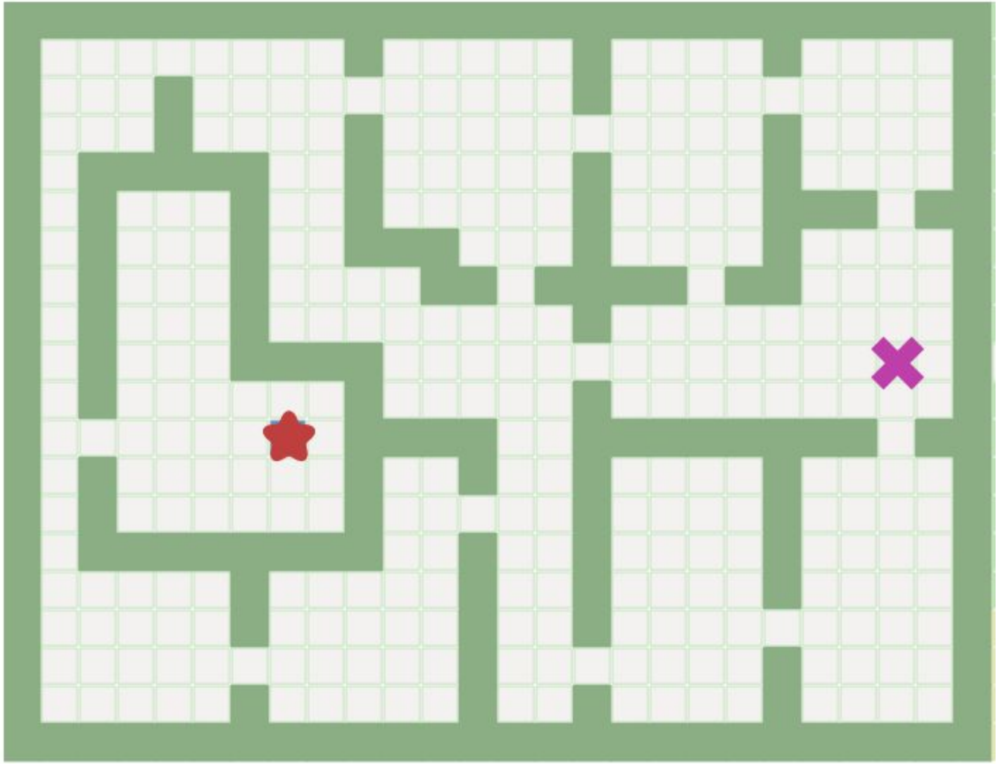




Edges can have *weights*.

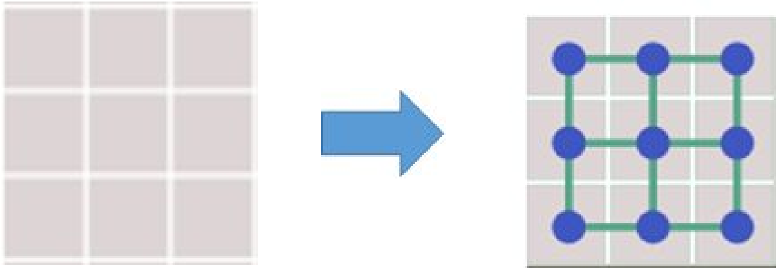


How to represent grids as graphs?

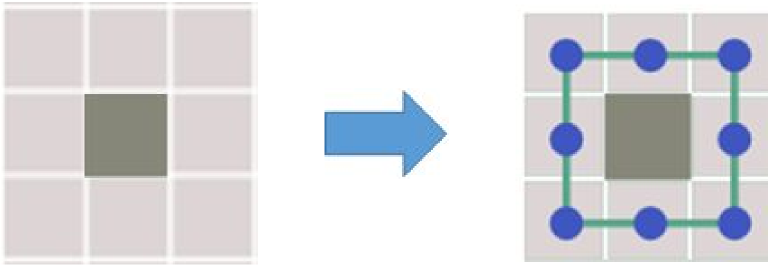


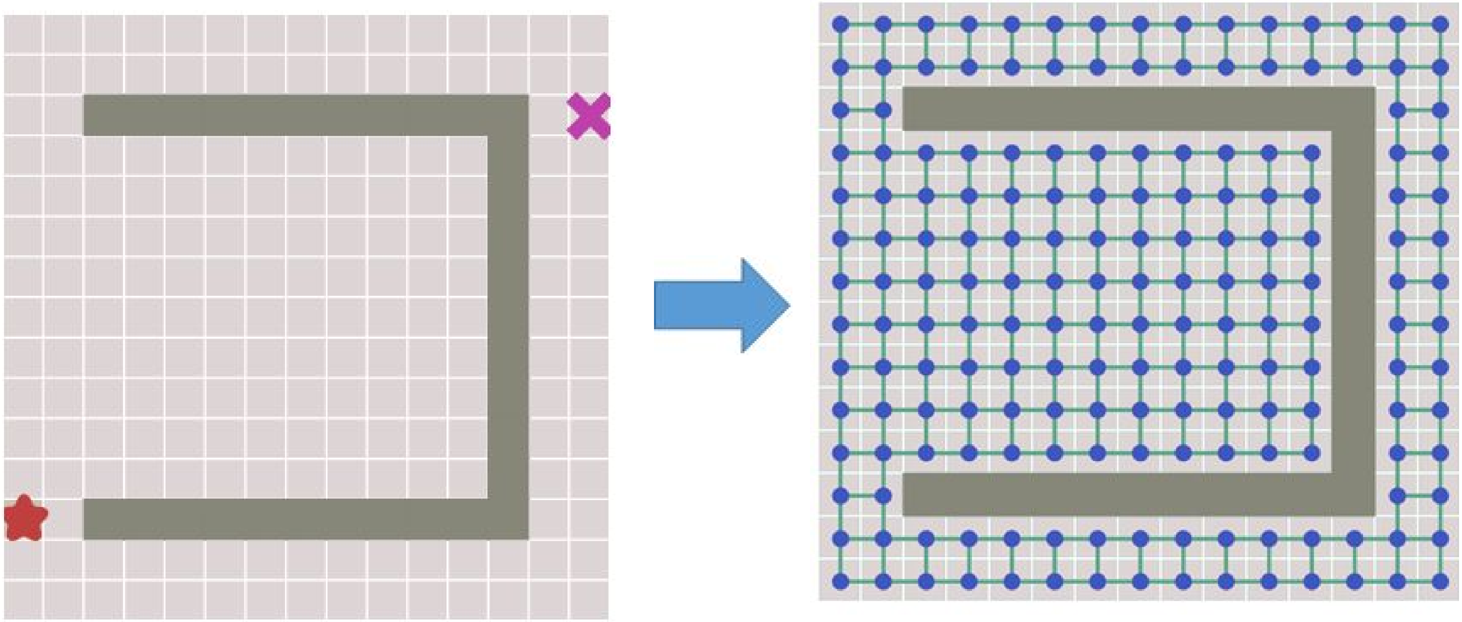
How to represent grids as graphs?

Each cell is a node. Edges connect adjacent cells.



Walls have no edges





Graph Traversal Algorithms

Graph Traversal Algorithms

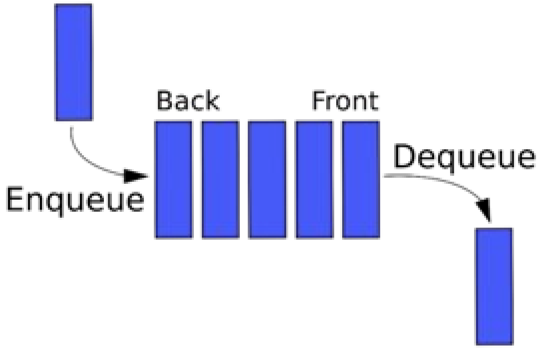
- These algorithms specify an *order* to search through the nodes of a graph.
- We start at the source node and keep searching until we find the target node.
- The *frontier* contains nodes that we've seen but haven't explored yet.
- Each iteration, we take a node off the frontier, and add its neighbors to the frontier.

Breadth First Search Demo

cs.stanford.edu/people/abisee/tutorial/bfs.html

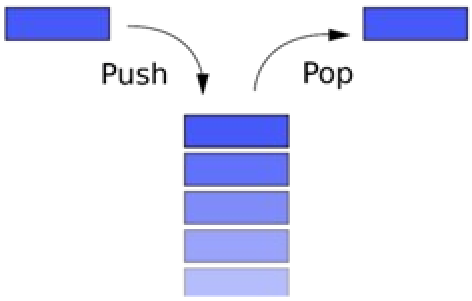
Breadth First Search vs. Depth First Search

BFS uses "first in first out".



This is a queue.

DFS uses "last in first out".



This is a stack.

Depth First Search Demo

cs.stanford.edu/people/abisee/tutorial/dfs.html

Activity: BFS vs DFS

cs.stanford.edu/people/abisee/tutorial/bfsdfs.html

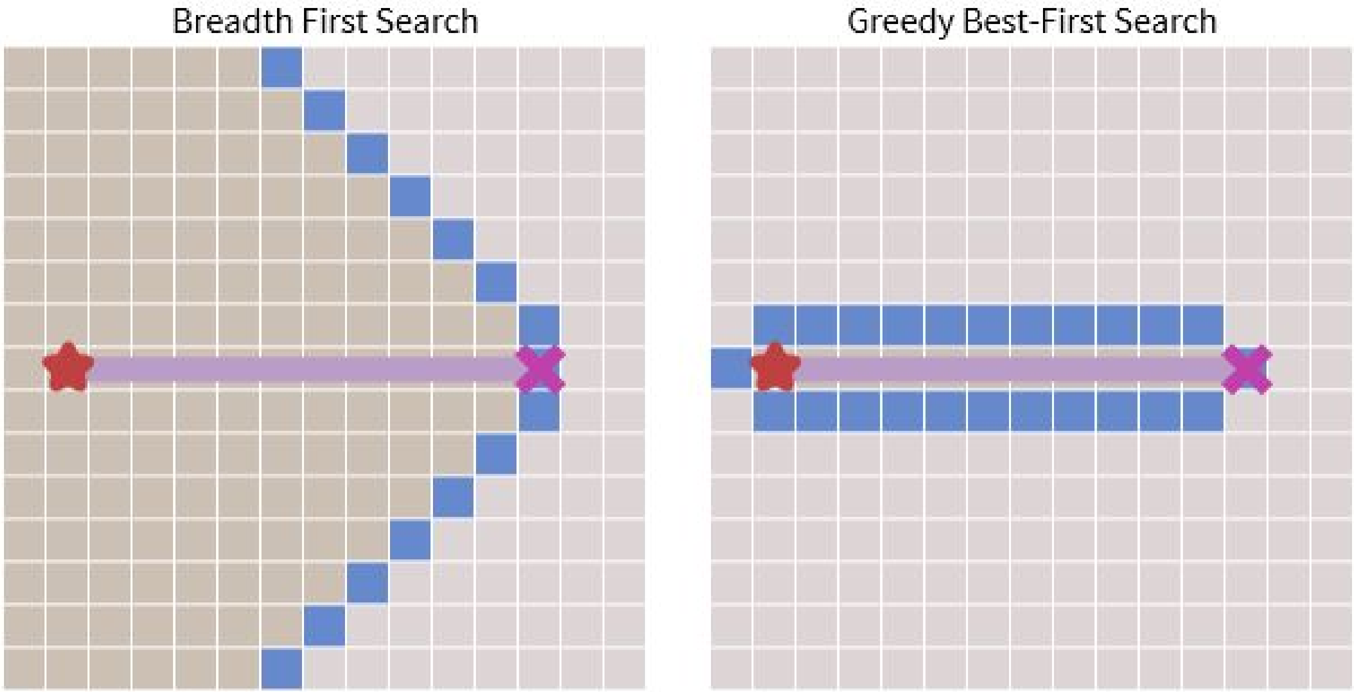
Explore:

- Try moving the source and target
- Try drawing walls

Discussion

- Does BFS necessarily return the shortest path?
  - Note that BFS explores nodes in the order of increasing distance.
- Does DFS necessarily return the shortest path?
- Once the target is found, how does the algorithm obtain the path itself?
- Disadvantages of BFS?
- Disadvantages of DFS?

Greedy Best First Search



Every step, Greedy Best First moves in the direction of the target.

A *greedy algorithm* is one that chooses the best-looking option at each step.

Greedy Best First Algorithm

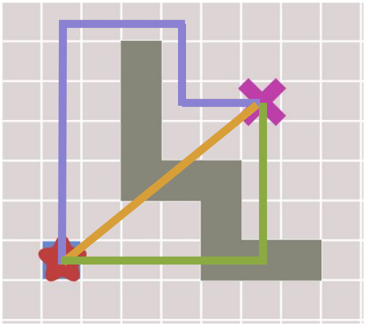
- Recall: BFS and DFS pick the next node off the frontier based on which was "first in" or "last in".
- Greedy Best First picks the "best" node according to some rule of thumb, called a *heuristic*.

Definition: A *heuristic* is an approximate measure of how close you are to the target.

A heuristic guides you in the right direction.

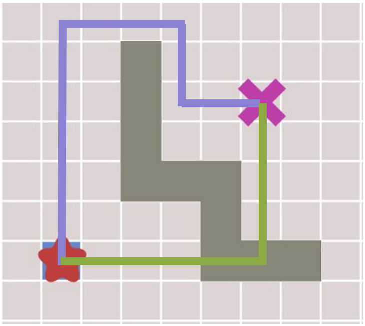
Heuristics for Greedy Best First

- We want a heuristic: a measure of how close we are to the target.
- A heuristic should be easy to compute.



- Try Euclidean distance or Manhattan distance.
- These are approximations for the actual shortest path, but easier to compute.

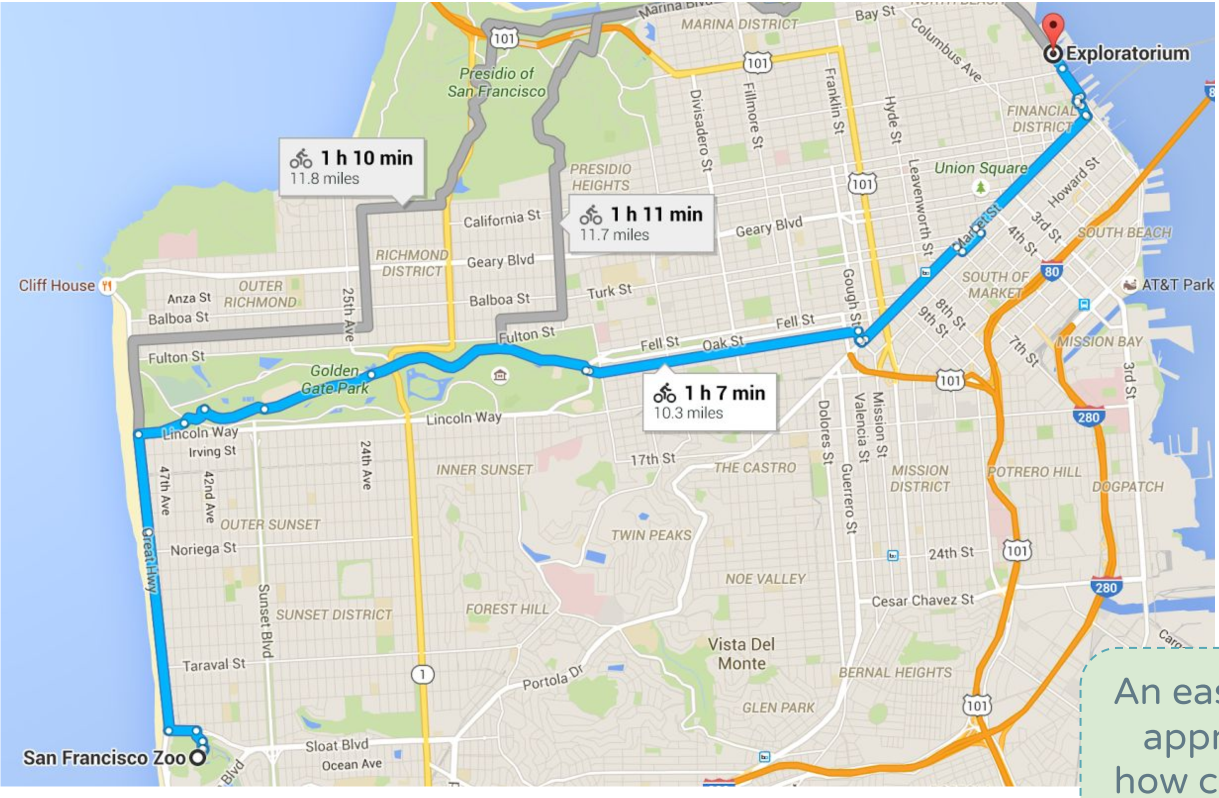
Heuristics for Greedy Best First



- Why is the Manhattan distance heuristic only an *approximation* for the true shortest path?
  - Answer: walls!
- A heuristic is often the solution for an <u>easier</u> version of the problem, that leaves out the <u>constraints</u> (e.g. walls)

Activity

We name a problem, you suggest a heuristic

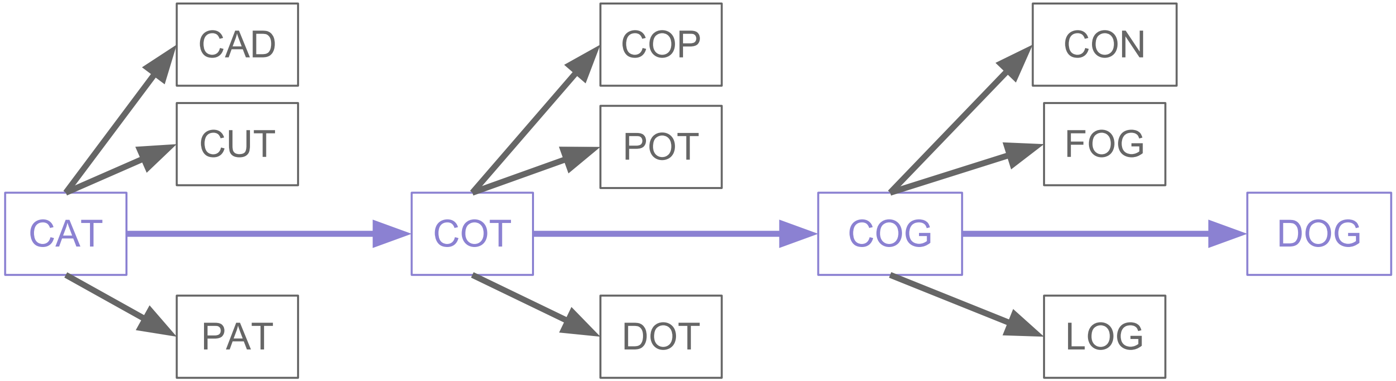


Problem: Google Maps route-planning

What is a possible heuristic?

An easy-to-compute approximation of how close you are to the target

Want: CAT ⟶ DOG

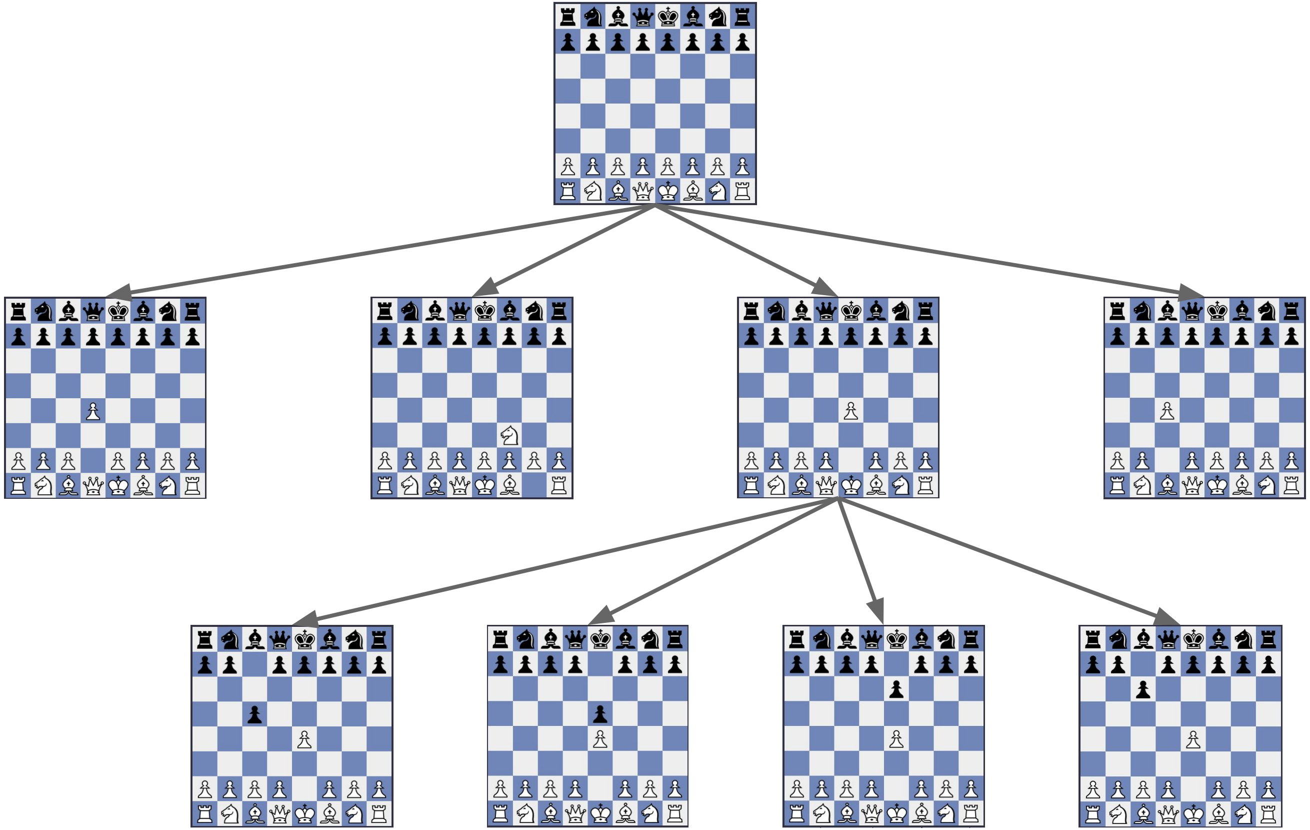


Problem: "Mutate the word" game

What is a possible heuristic?

Problem: Find the shortest chain of Facebook friends that goes from Person A to Person B

What is a possible heuristic?



Problem: Find a sequence of moves to win

What is a possible heuristic?

Greedy Best First Demo and activity

cs.stanford.edu/people/abisee/tutorial/greedy.html

Challenge: trick Greedy Best First!

Can you draw the walls so that Greedy Best First comes up with a path that is *much longer* than Breadth First Search?

Discussion

- Recall: Breadth First Search is *optimal* (always returns the shortest path). Is Greedy Best First also optimal?
- Strengths of Greedy Best First?
- Weaknesses of Greedy Best First?
- How might you improve Greedy Best First?

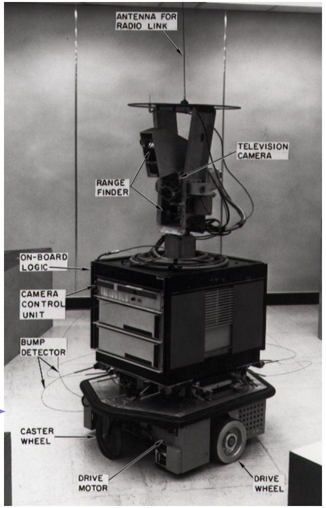
A* Search

Not so easily tricked...

A* Search

Developed by { Peter Hart, Nils Nilsson, Bertram Raphael } at Stanford

in 1968 to help Shakey the Robot navigate a room of obstacles.



Now in the Computer History Museum!

A* Search

- A* Search combines the strengths of Breadth First Search and Greedy Best First.
- Like BFS, it finds the shortest path, and like Greedy Best First, it's fast.
- Each iteration, A* chooses the node on the frontier which minimizes:

steps from source + approximate steps to target

Like BFS, looks at nodes close to source first (thoroughness)

Like Greedy Best First, uses heuristic to prioritize nodes closer to target (speed)

A* Search Demo and activity

cs.stanford.edu/people/abisee/tutorial/astar.html

Explore:

- Try moving the source and target
- Try drawing the walls

Discussion

- Which algorithm was fastest?
- Which explored the most area before finding the target?
- Do A* and BFS always find the same path?

A* is optimal

Theorem: If the heuristic function is a lower bound for the true shortest path to target, i.e.

heuristic(node) ≤ shortest_path(node,target)

for all nodes, then A* search is optimal (always finds the shortest path).

Proof Idea: The heuristic is optimistic so it never ignores a good path. As all good paths are explored, we therefore discover the optimal path.

Algorithms for Weighted Graphs

Example: Google Maps

Weight of edge = time to travel

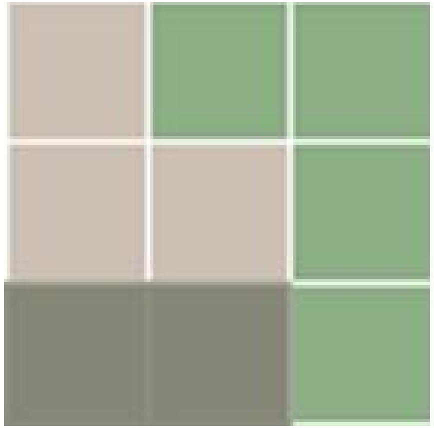
Incorporates information like:

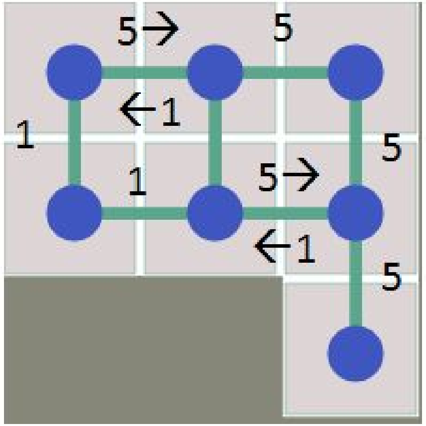
- length of road
- speed limit
- current traffic conditions

Now we want the minimum *cost* path

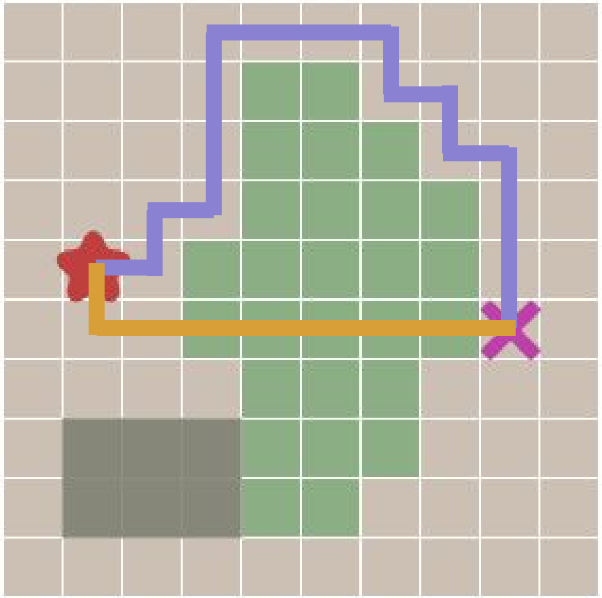
Terrain to weighted graph

Terrain	Cost
plain	1
hill	5
wall	Infinity





How to alter our algorithms?



Terrain		Cost
plain		1
hill		5
wall		Infinity

Minimum number of steps

Minimum cost path

Dijkstra's algorithm

- Like BFS for weighted graphs.
  - If all costs are equal, Dijkstra = BFS!
- Explores nodes in increasing order of *cost* from source.
- Let’s work through some examples on the board!

Dijkstra contour demo

cs.stanford.edu/people/abisee/tutorial/dijkstra.html

Weighted A*

Regular A* priority function:

> steps from source + approximate steps to target

Weighted A* priority function:

> cost from source + approximate cost to target

Activity: Dijkstra vs weighted A*

cs.stanford.edu/people/abisee/tutorial/customize.html

Explore:

- Can you alter the map so that A* finishes much more quickly than Dijkstra?
- Do Dijkstra and weighted A* ever find paths of different lengths?
- Do Dijkstra and weighted A* ever find different paths?
- Is Dijkstra or weighted A* faster?
  - Always or just sometimes?

Recap

Search algorithms for unweighted and weighted graphs

Breadth First Search	First in first out, optimal but slow
Depth First Search	Last in first out, not optimal and meandering
Greedy Best First	Goes for the target, fast but easily tricked
A* Search	"Best of both worlds": optimal and fast

Dijkstra	Explores in increasing order of cost, optimal but slow
Weighted A*	Optimal and fast

Questions?

about this or anything else...