Programming Abstractions

CS106B

Cynthia Lee

Graphs Topics

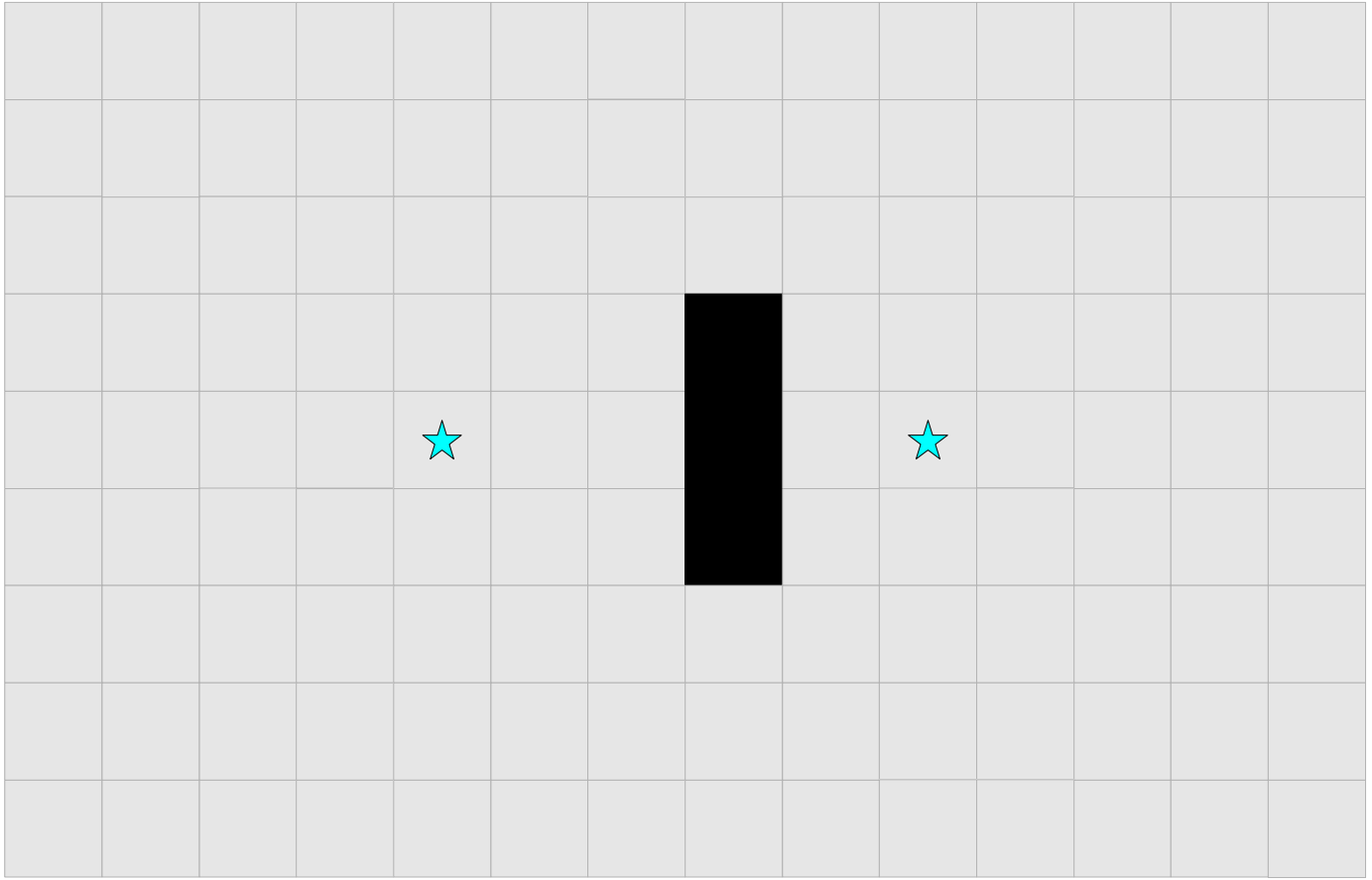
Graphs!

1. **Basics**
   - What are they? How do we represent them?
2. **Theorems**
   - What are some things we can prove about graphs?
3. **Breadth-first search on a graph**
   - Spoiler: just a very, very small change to tree version
4. **Dijkstra's shortest paths algorithm**
   - Spoiler: just a very, very small change to BFS
5. **A* shortest paths algorithm**
   - Spoiler: just a very, very small change to Dijkstra's
6. **Minimum Spanning Tree**
   - Kruskal's algorithm

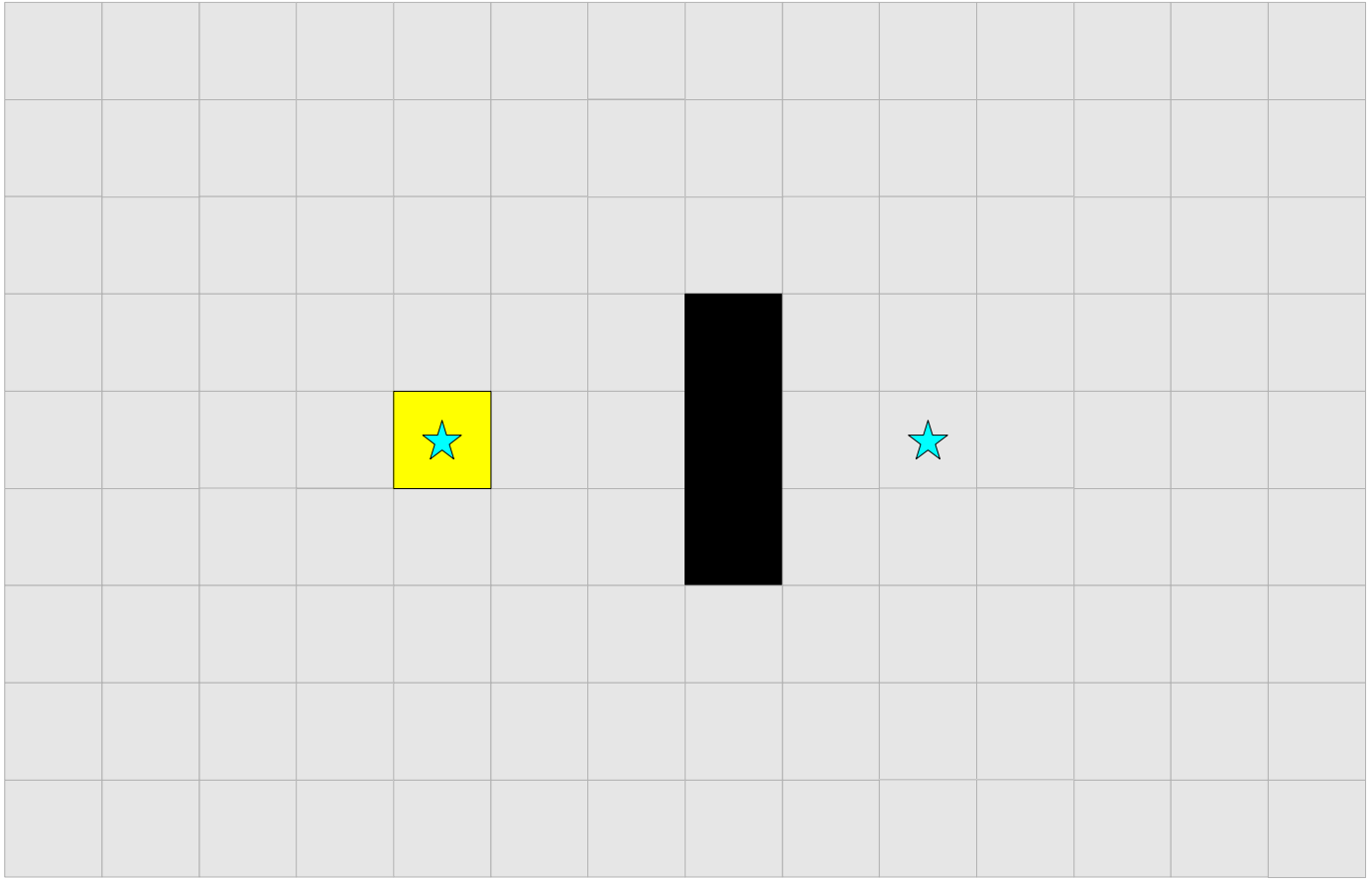
A* Search

- Mark all nodes as gray.
- Mark the initial node ***s*** as yellow and at candidate distance **0**.
- Enqueue ***s*** into the priority queue with priority **h(s,t)**.
- While not all nodes have been visited:
- Dequeue the lowest-cost node ***u*** from the priority queue.
- Color ***u*** green. The candidate distance ***d*** that is currently stored for node ***u*** is the length of the shortest path from ***s*** to ***u***.
- If ***u*** is the destination node ***t***, you have found the shortest path from ***s*** to ***t*** and are done.
- For each node ***v*** connected to ***u*** by an edge of length ***L***:
  - If ***v*** is gray:
    - Color ***v*** yellow.
    - Mark ***v***'s distance as ***d + L***.
    - Set ***v***'s parent to be ***u***.
    - Enqueue ***v*** into the priority queue with priority ***d + L +*** **h(v,t)**.
  - If ***v*** is yellow and the candidate distance to ***v*** is greater than ***d + L***:
    - Update ***v***'s candidate distance to be ***d + L***.
    - Update ***v***'s parent to be ***u***.
    - Update ***v***'s priority in the priority queue to ***d + L +*** **h(v,t)**.

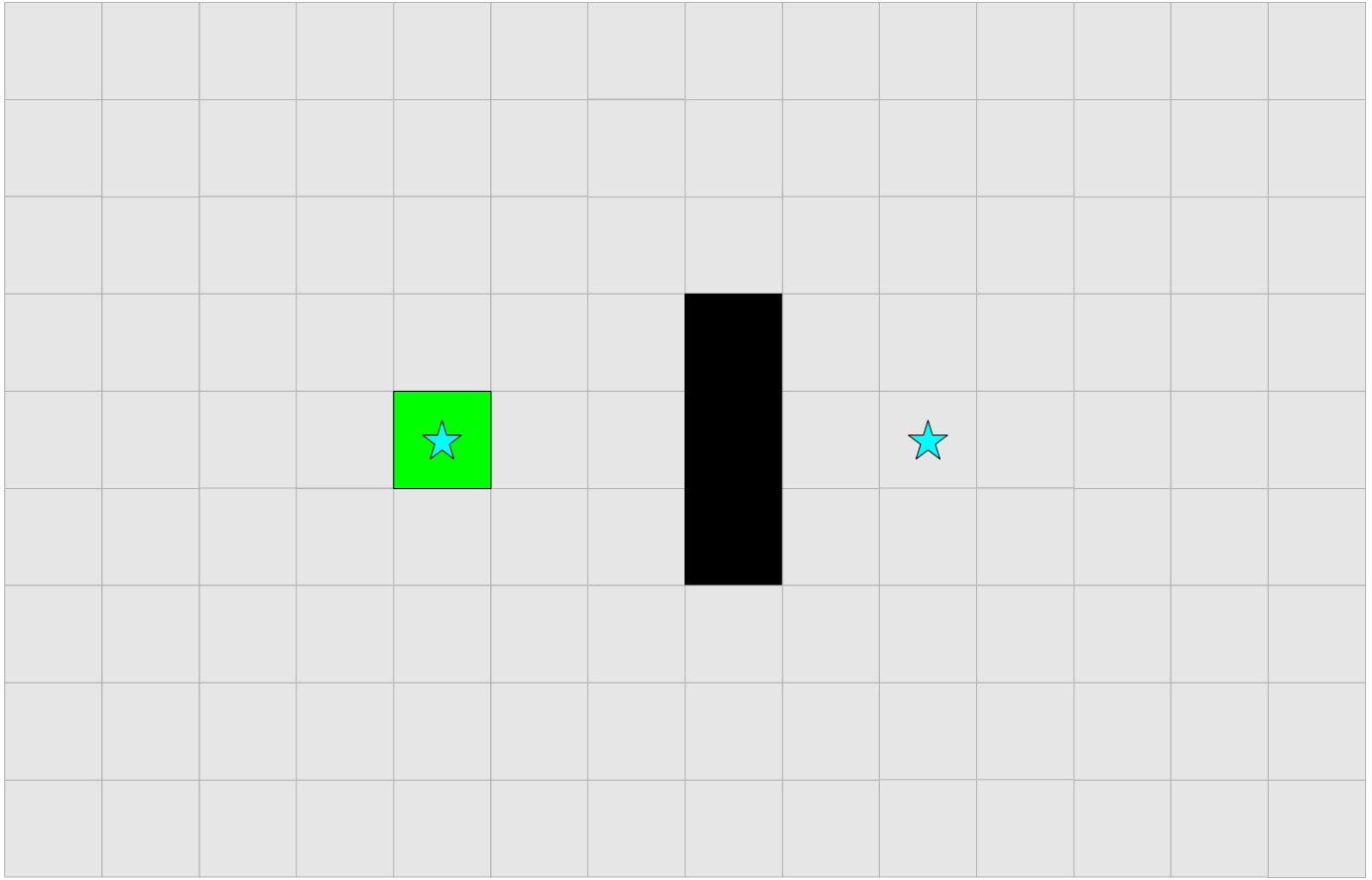
A* on two points where the heuristic is slightly misleading due to a wall blocking the way



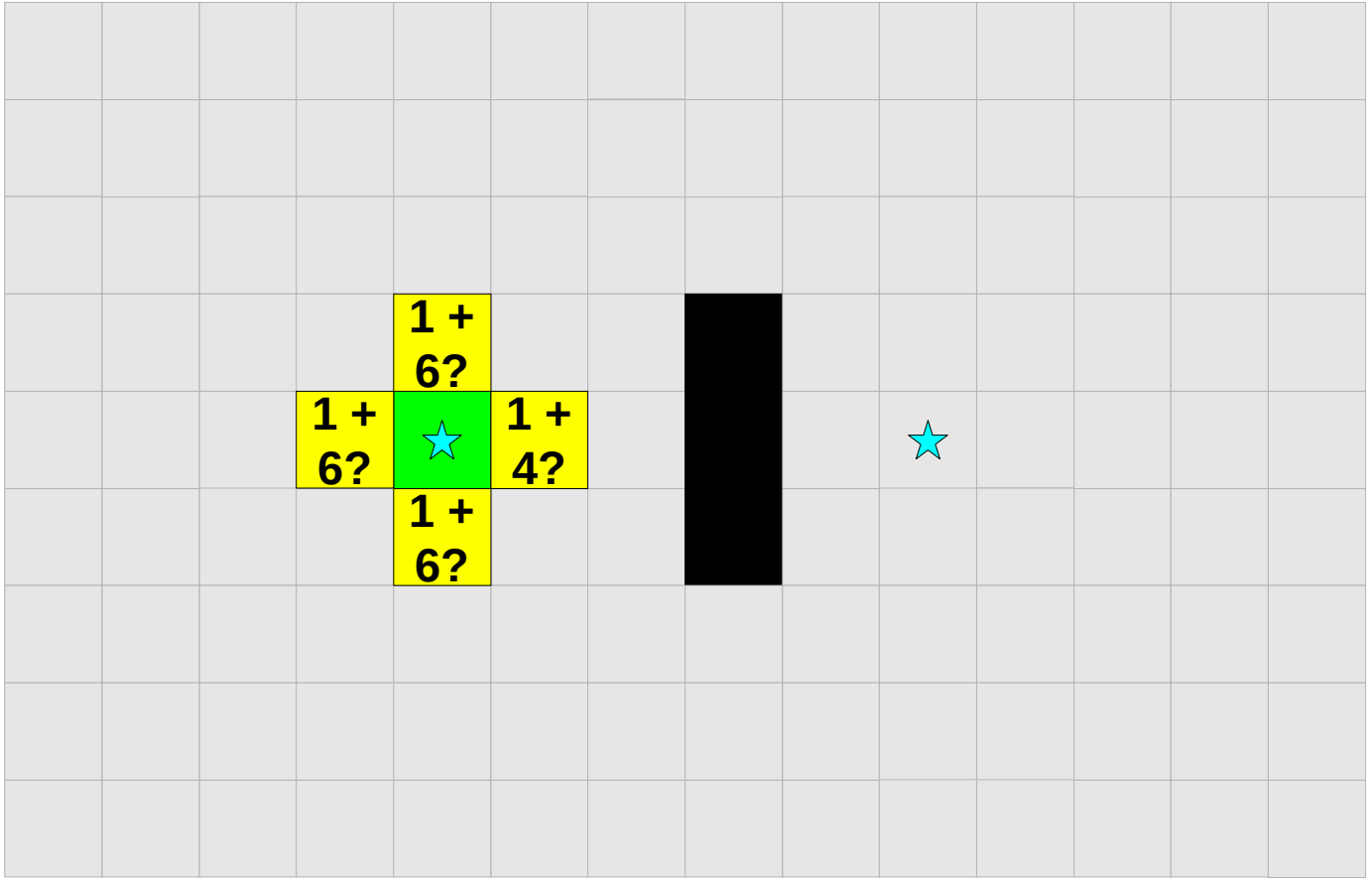
A* starts with start node yellow, other nodes grey.



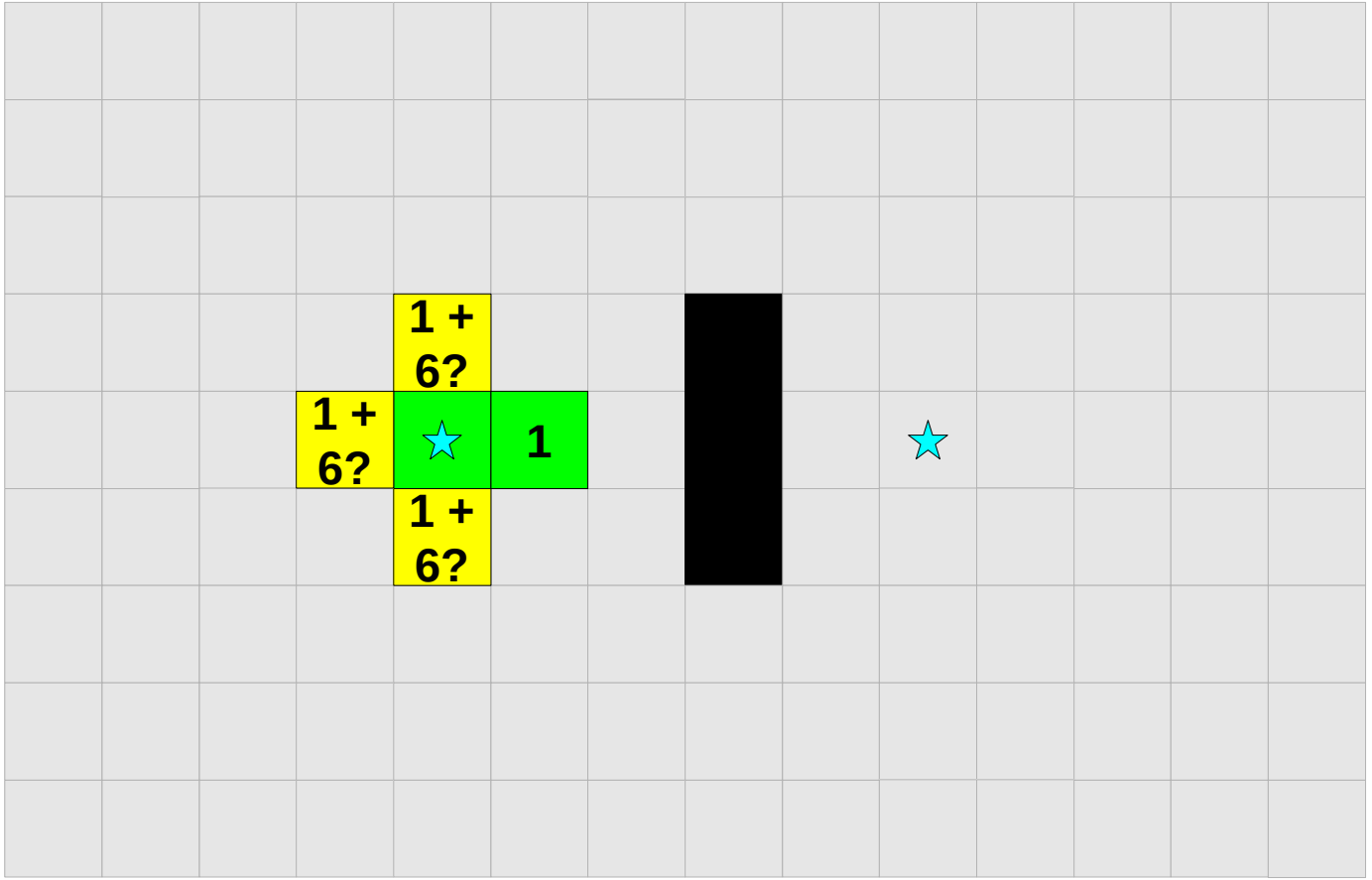
A*: dequeue start node, turns green.

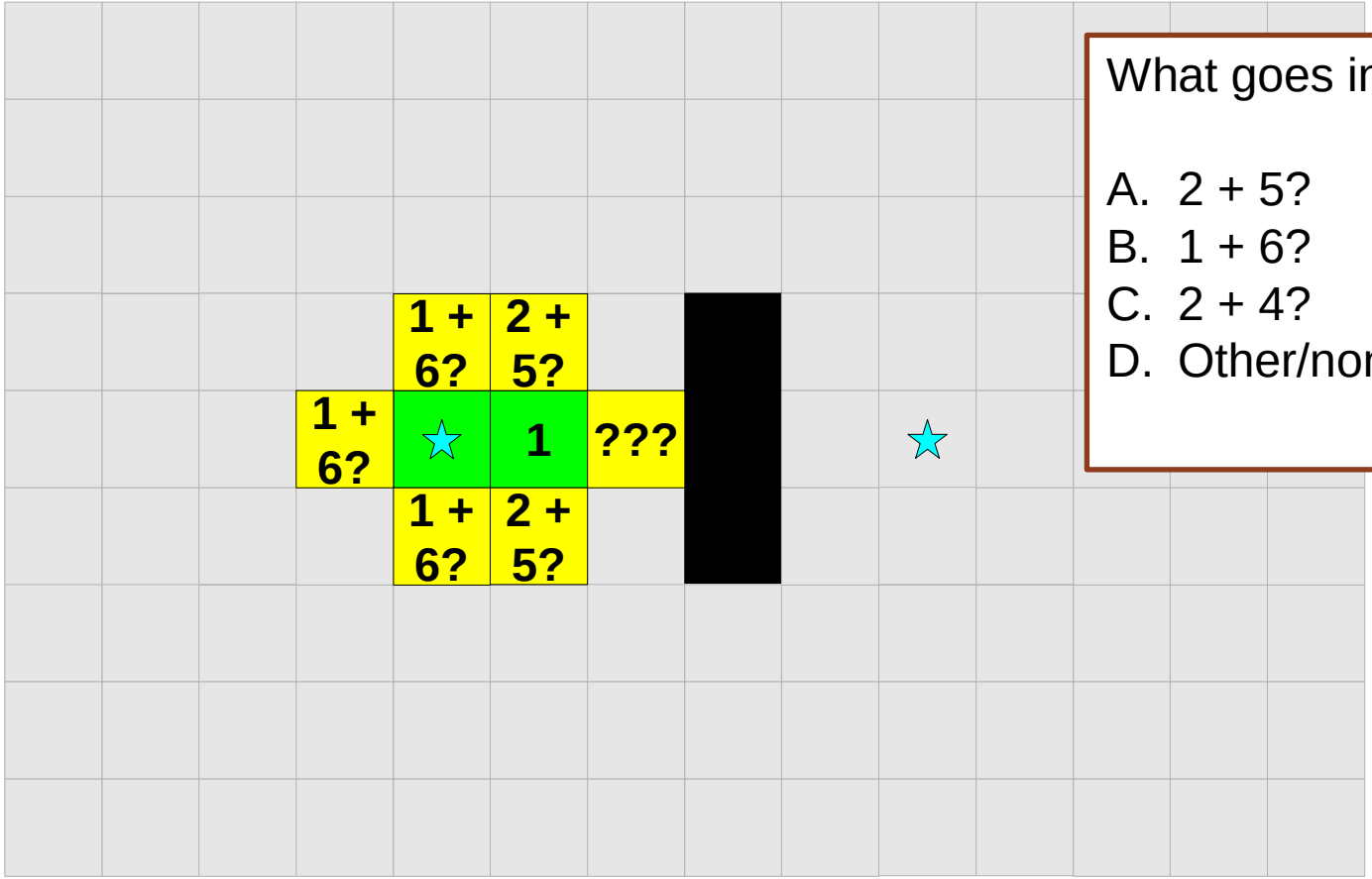


A*: enqueue neighbors with candidate distance + heuristic distance as the priority value.



A*: dequeue min-priority-value node.

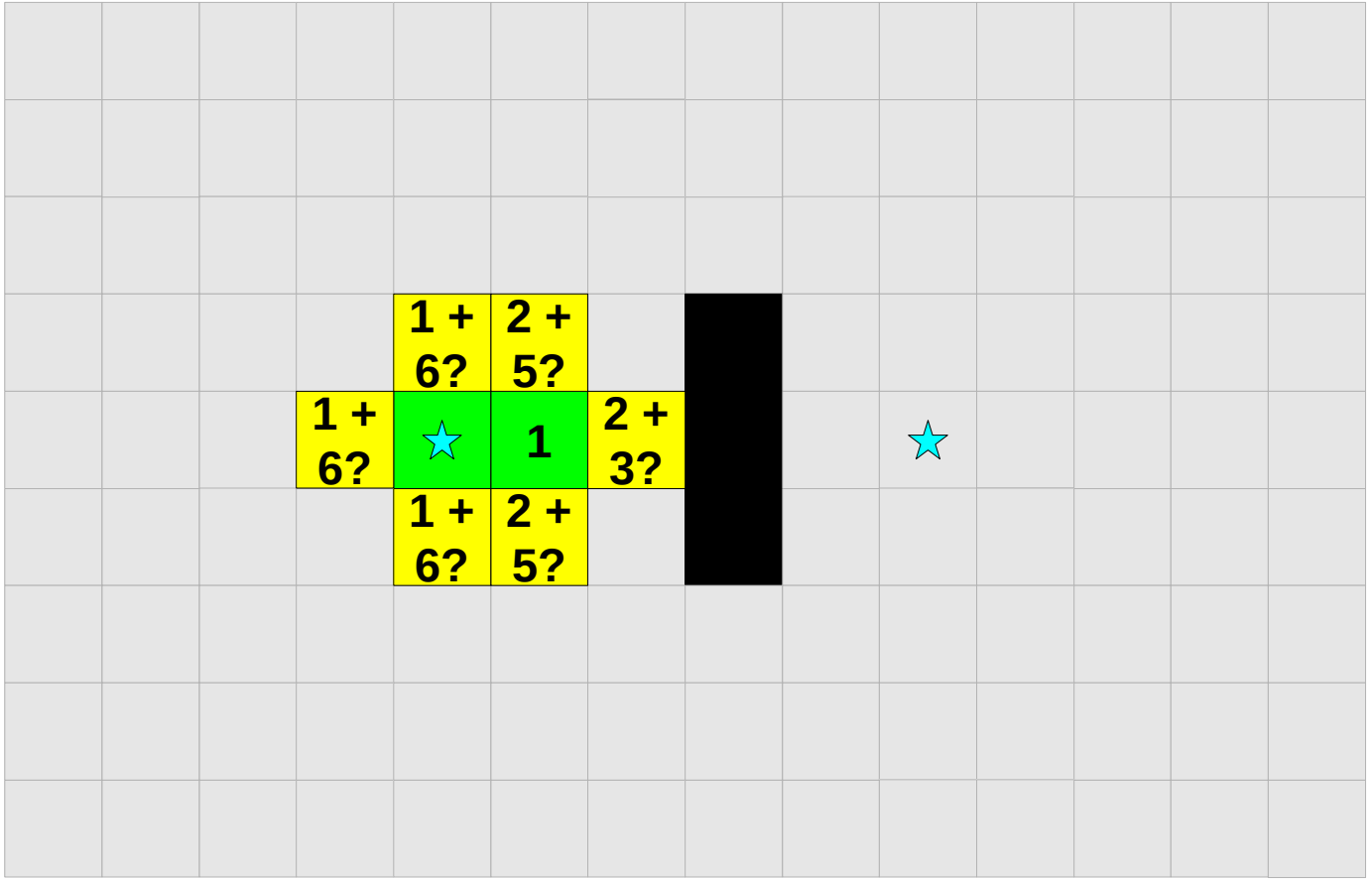


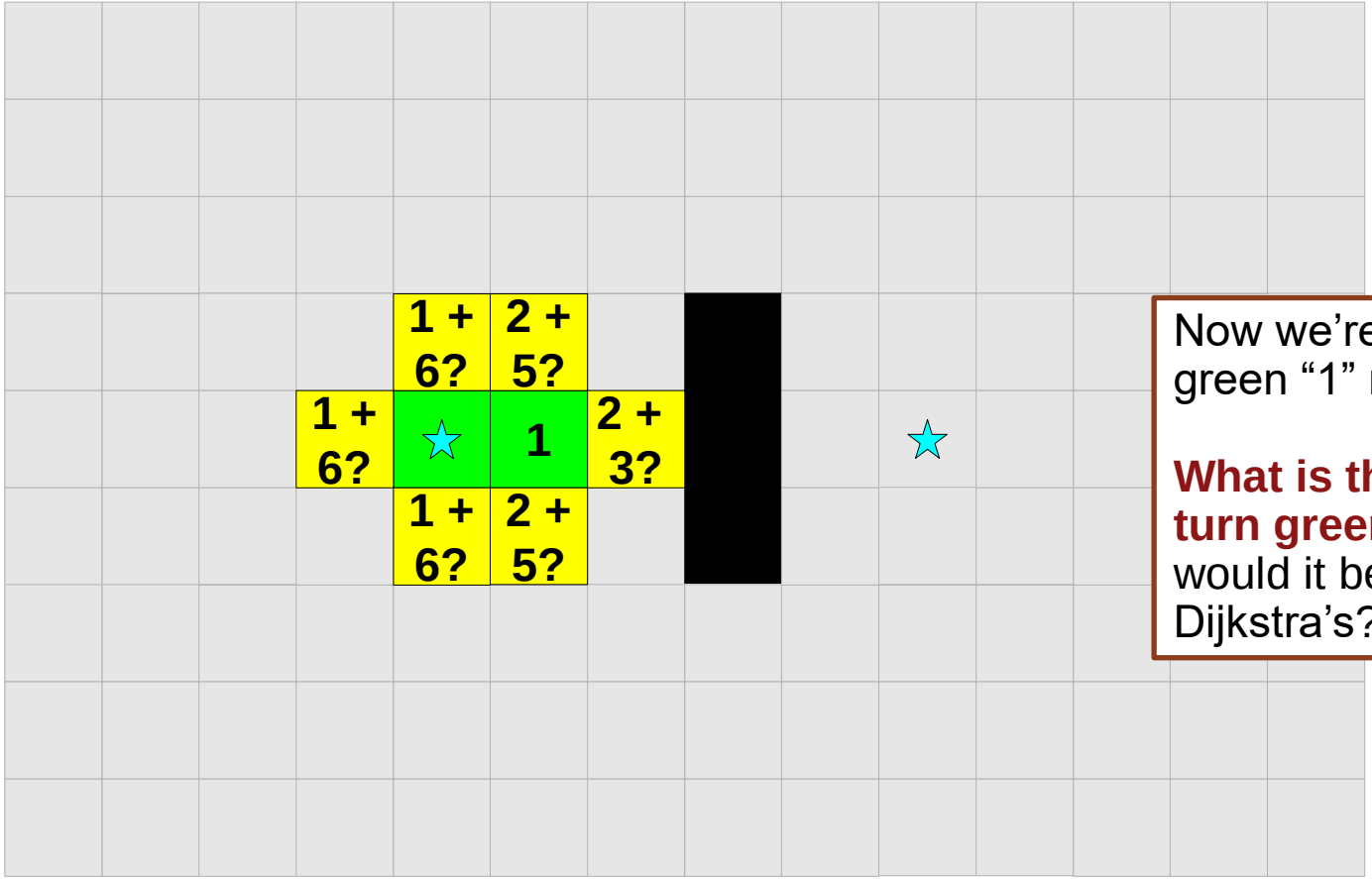


What goes in the **???** ?

A. 2 + 5?
B. 1 + 6?
C. 2 + 4?
D. Other/none/more

A*: enqueue neighbors.

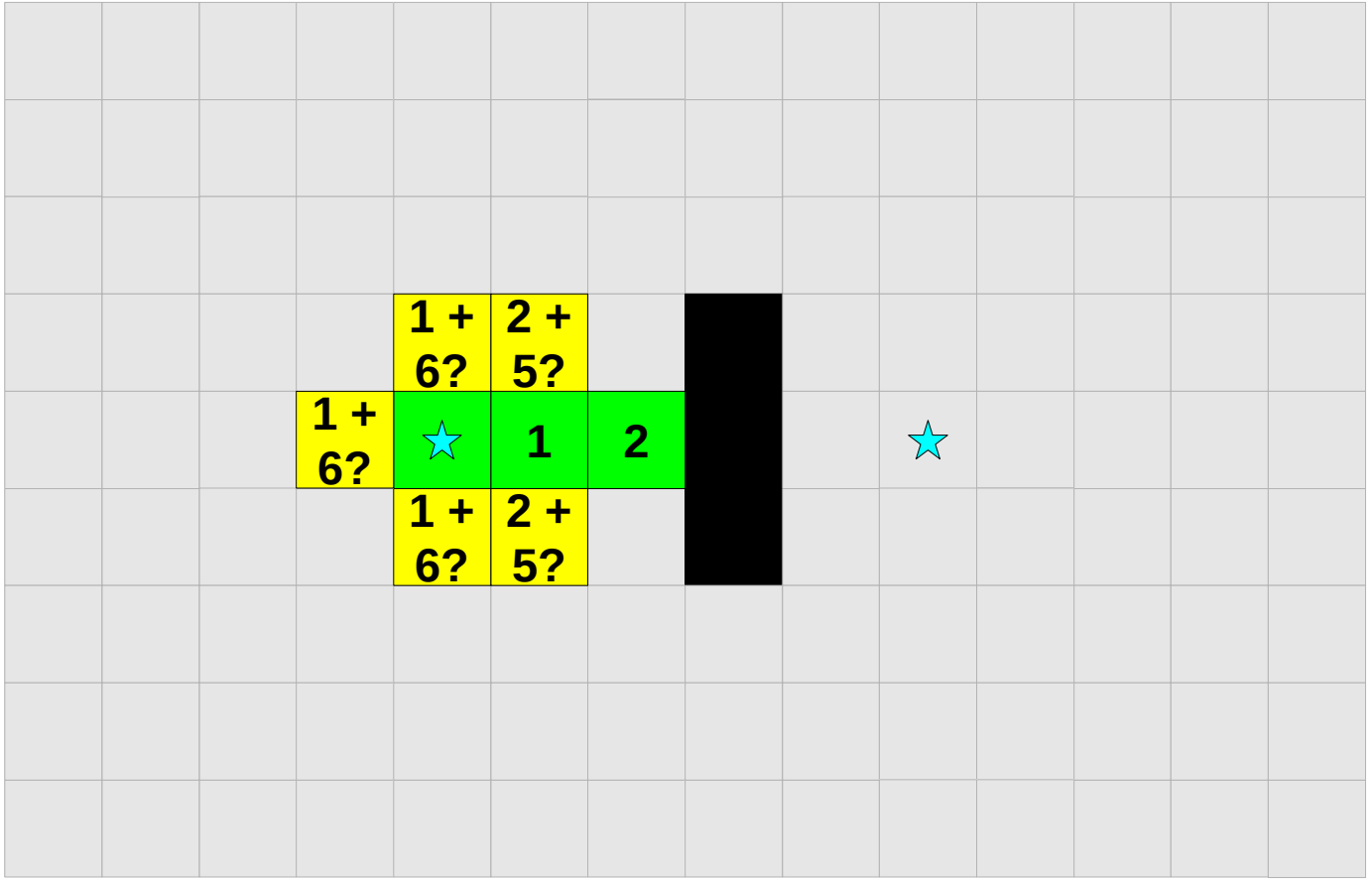




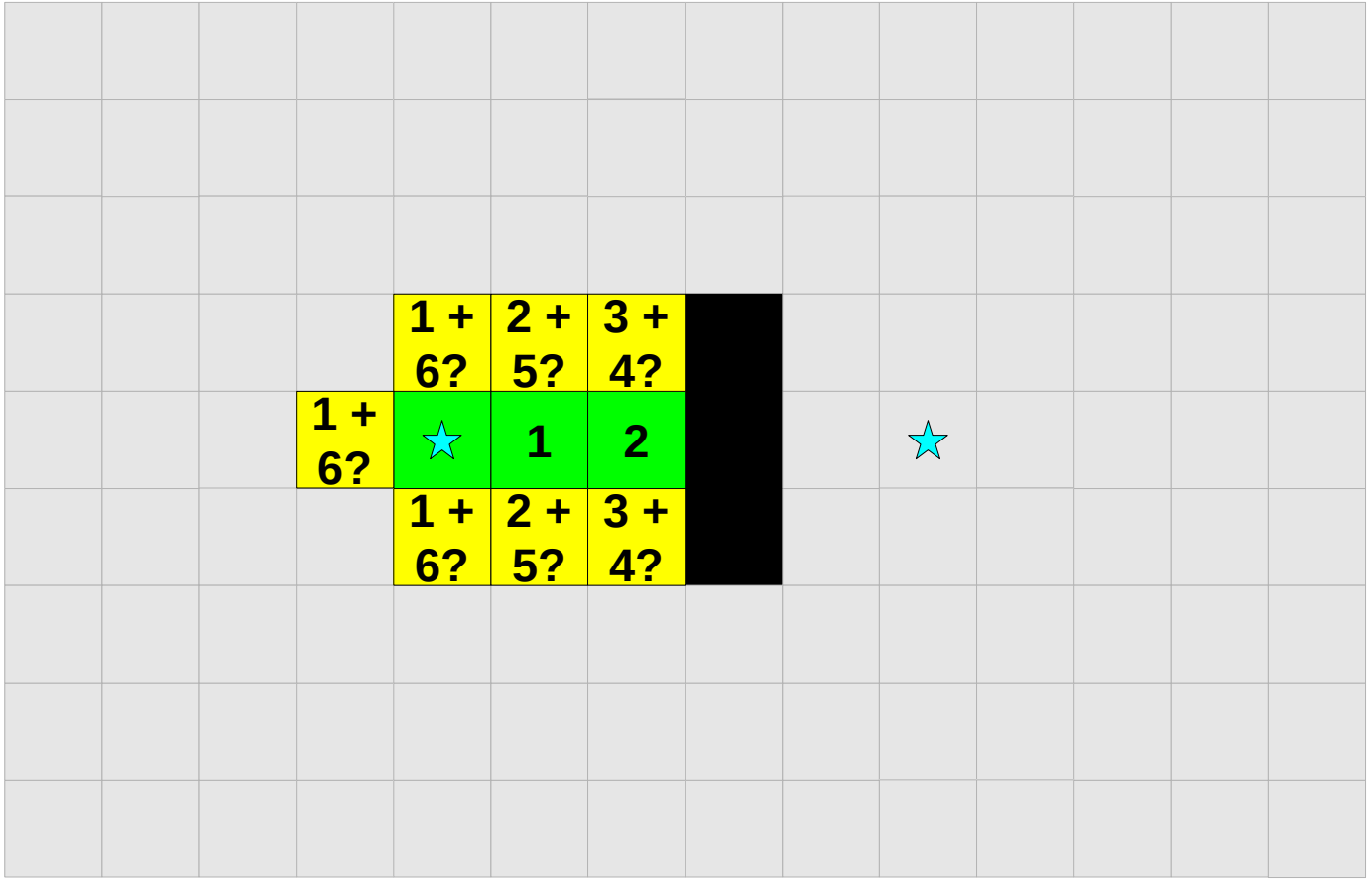
Now we're done with the green "1" node's turn.

What is the next node to turn green? (and what would it be if this were Dijkstra's?)

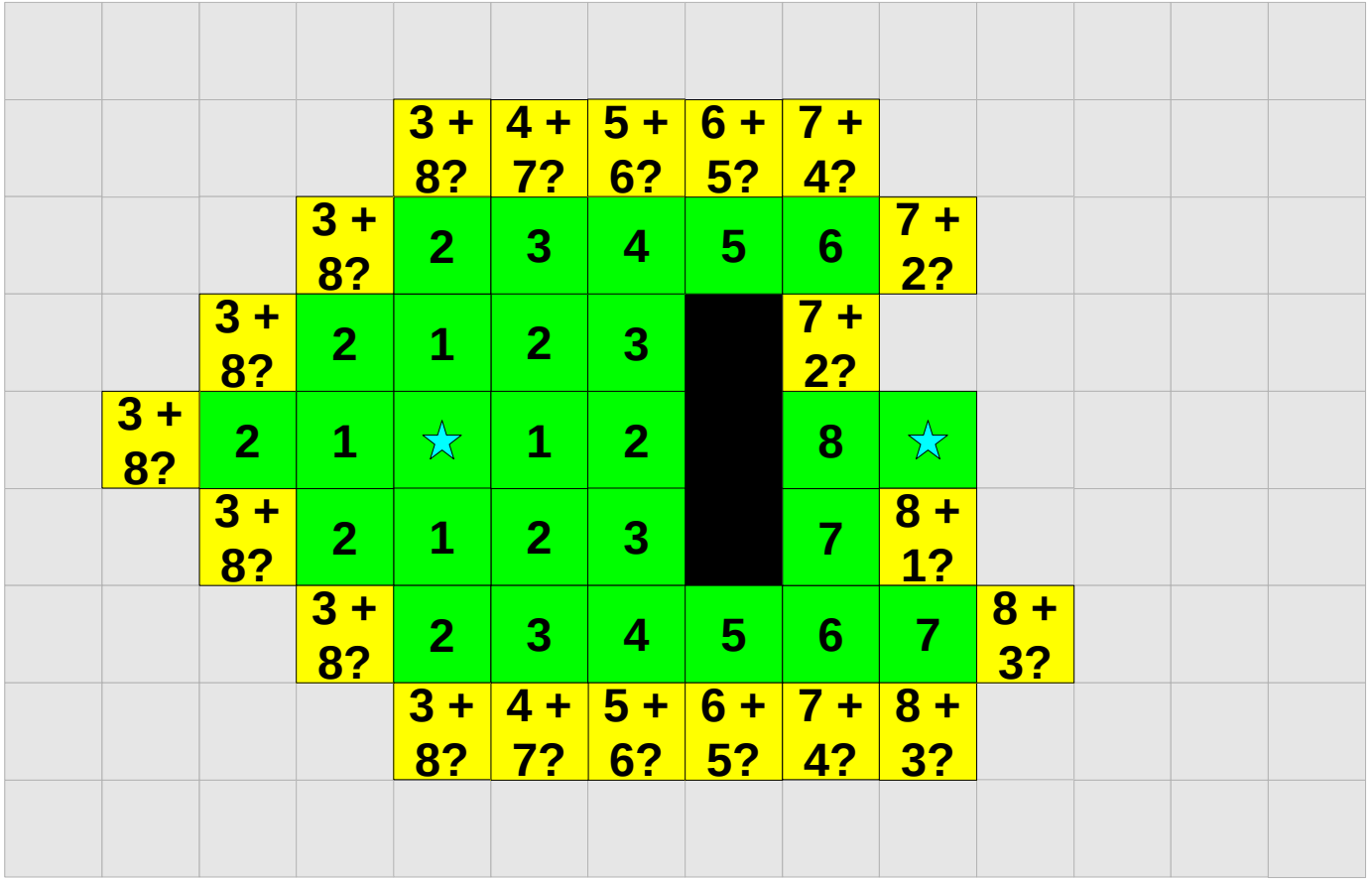
A*: dequeue next lowest priority value node. Notice we are making a straight line right for the end point, not wasting time with other directions.



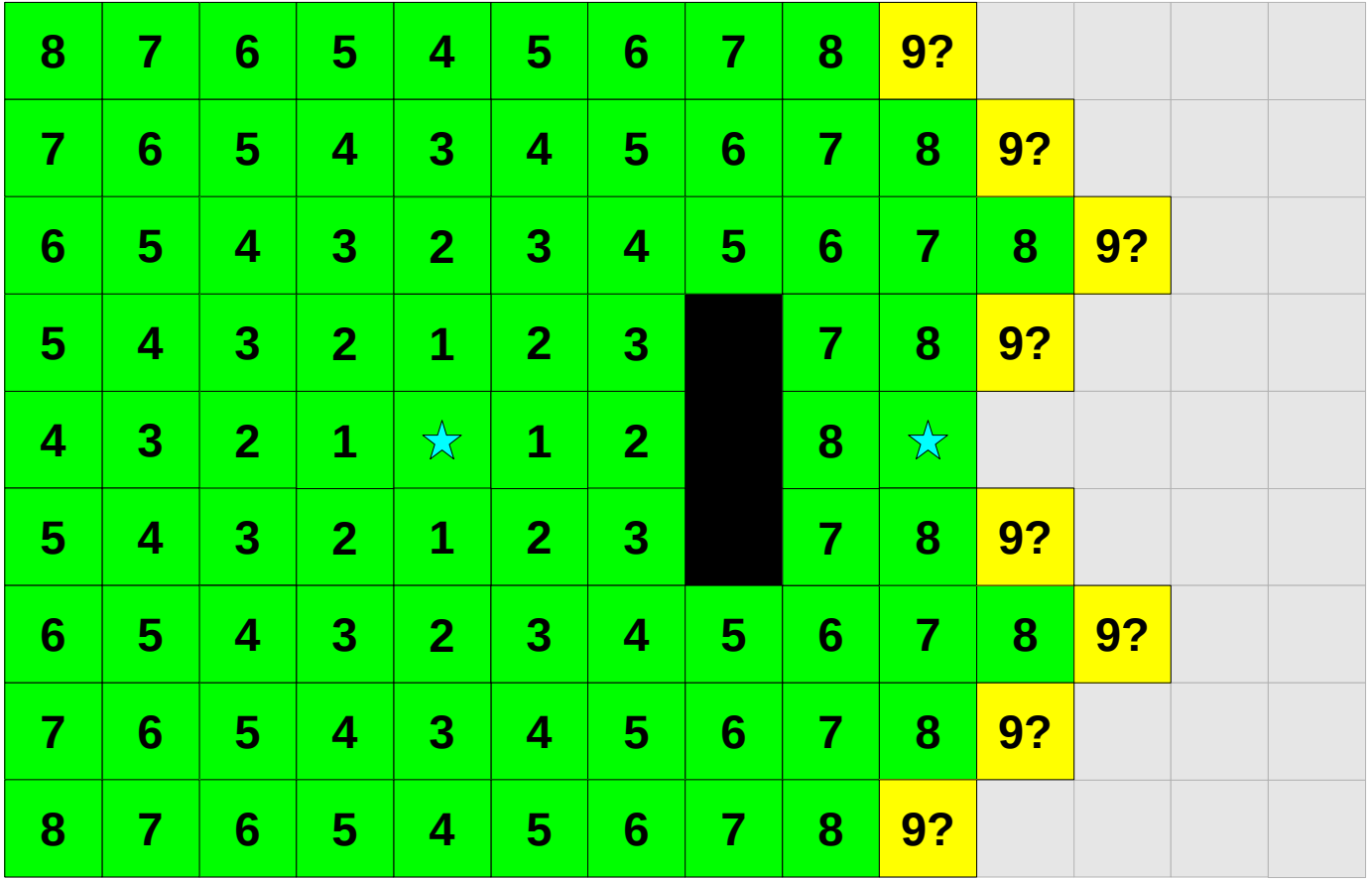
A*: enqueue neighbors—uh-oh, wall blocks us from continuing forward.



A*: eventually figures out how to go around the wall, with some waste in each direction.



For Comparison: What Dijkstra's Algorithm Would Have Searched



A* Search

- Mark all nodes as gray.
- Mark the initial node ***s*** as yellow and at candidate distance **0**.
- Enqueue ***s*** into the priority queue with priority **h(s,t)**.
- While not all nodes have been visited:
- Dequeue the lowest-cost node ***u*** from the priority queue.
- Color ***u*** green. The candidate distance ***d*** that is currently stored for node ***u*** is the length of the shortest path from ***s*** to ***u***.
- If ***u*** is the destination node ***t***, you have found the shortest path from ***s*** to ***t*** and are done.
- For each node ***v*** connected to ***u*** by an edge of length ***L***:
  - If ***v*** is gray:
    - Color ***v*** yellow.
    - Mark ***v***'s distance as ***d + L***.
    - Set ***v***'s parent to be ***u***.
    - Enqueue ***v*** into the priority queue with priority ***d + L +*** **h(v,t)**.
  - If ***v*** is yellow and the candidate distance to ***v*** is greater than ***d + L***:
    - Update ***v***'s candidate distance to be ***d + L***.
    - Update ***v***'s parent to be ***u***.
    - Update ***v***'s priority in the priority queue to ***d + L +*** **h(v,t)**.

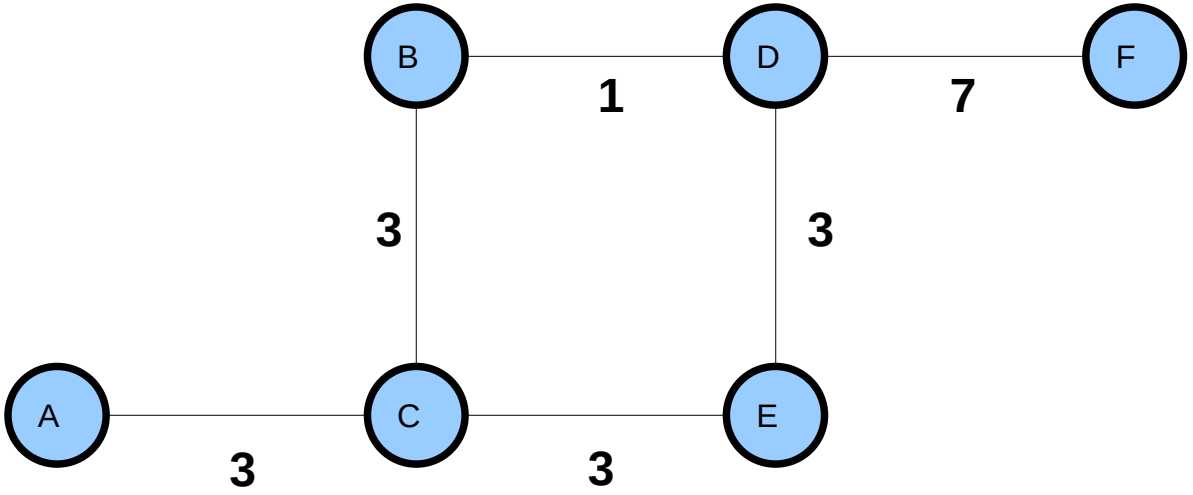
A* solves Super Mario game

https://youtu.be/DlkMs4ZHHr8

Minimum Spanning Tree

A **spanning tree** in an undirected graph is a set of edges with no cycles that connects all nodes.

A **minimum spanning tree** (or **MST**) is a spanning tree with the least total cost.



How many distinct minimum spanning trees are in this graph?

A. 0-1
B. 2-3
C. 4-5
D. 6-7
E. >7

Kruskal's algorithm

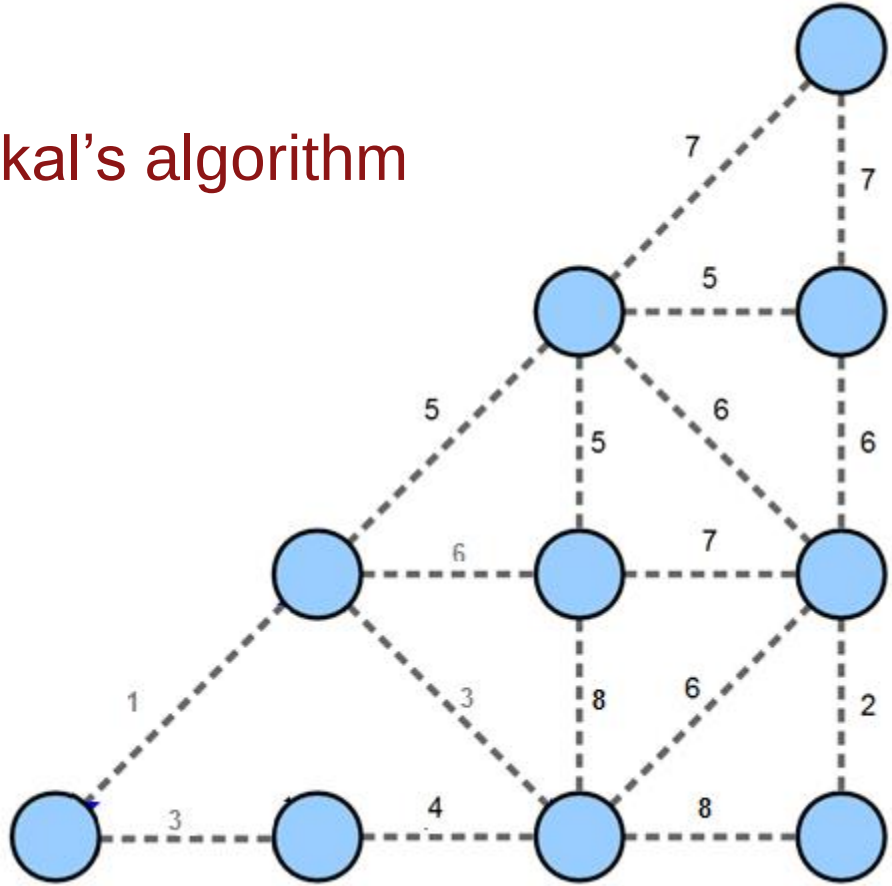
Remove all edges from graph

Place all edges in a PQ based on length/weight

While !PQ.isEmpty():

- Dequeue edge
- If the edge connects previous disconnected nodes or groups of nodes, keep the edge
- Otherwise discard the edge

Kruskal's algorithm



The Good Will Hunting Problem

Video Clip

https://www.youtube.com/watch?v=N7b0cLn-wHU

“Draw all the homeomorphically irreducible trees with n=10.”

“Draw all the homeomorphically irreducible trees with n=10.”

- **“trees”** as used here simply means **graphs with no cycles**
- **“with n = 10”** means has **10 nodes**
- **“homeomorphically irreducible”** boils down to needing to meet two criteria:
  1. **No nodes of degree 2** allowed in your solutions
  - For this problem, nodes of degree 2 are useless in terms of tree structure—they just act as a blip on an edge—and are therefore banned
  2. **Have to be actually different structure**
  - Ignore superficial differences in rotation or angles of drawing