Programming Abstractions

CS106B

Cynthia Lee

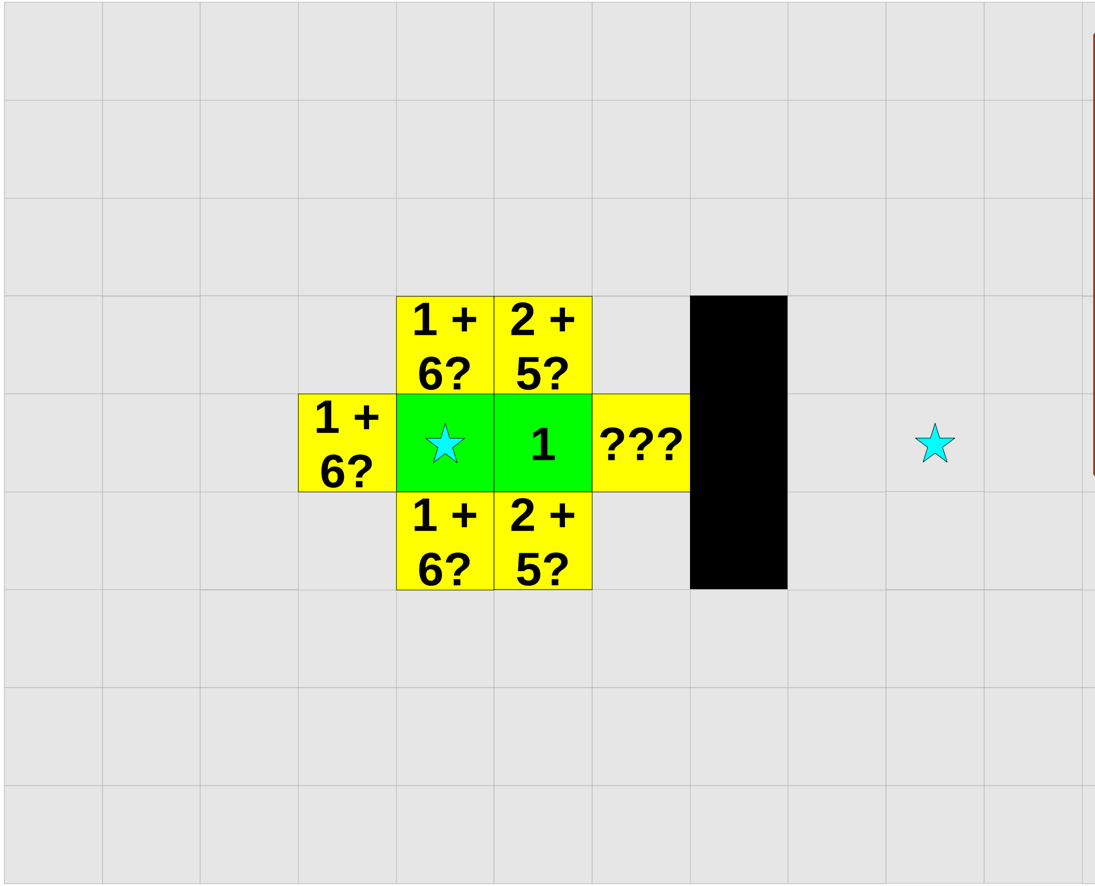
Graphs Topics

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 (continued)**
   - Spoiler: just a very, very small change to Dijkstra's
6. **Minimum Spanning Tree**
   - Kruskal's algorithm

A* Solving Super Mario (video)

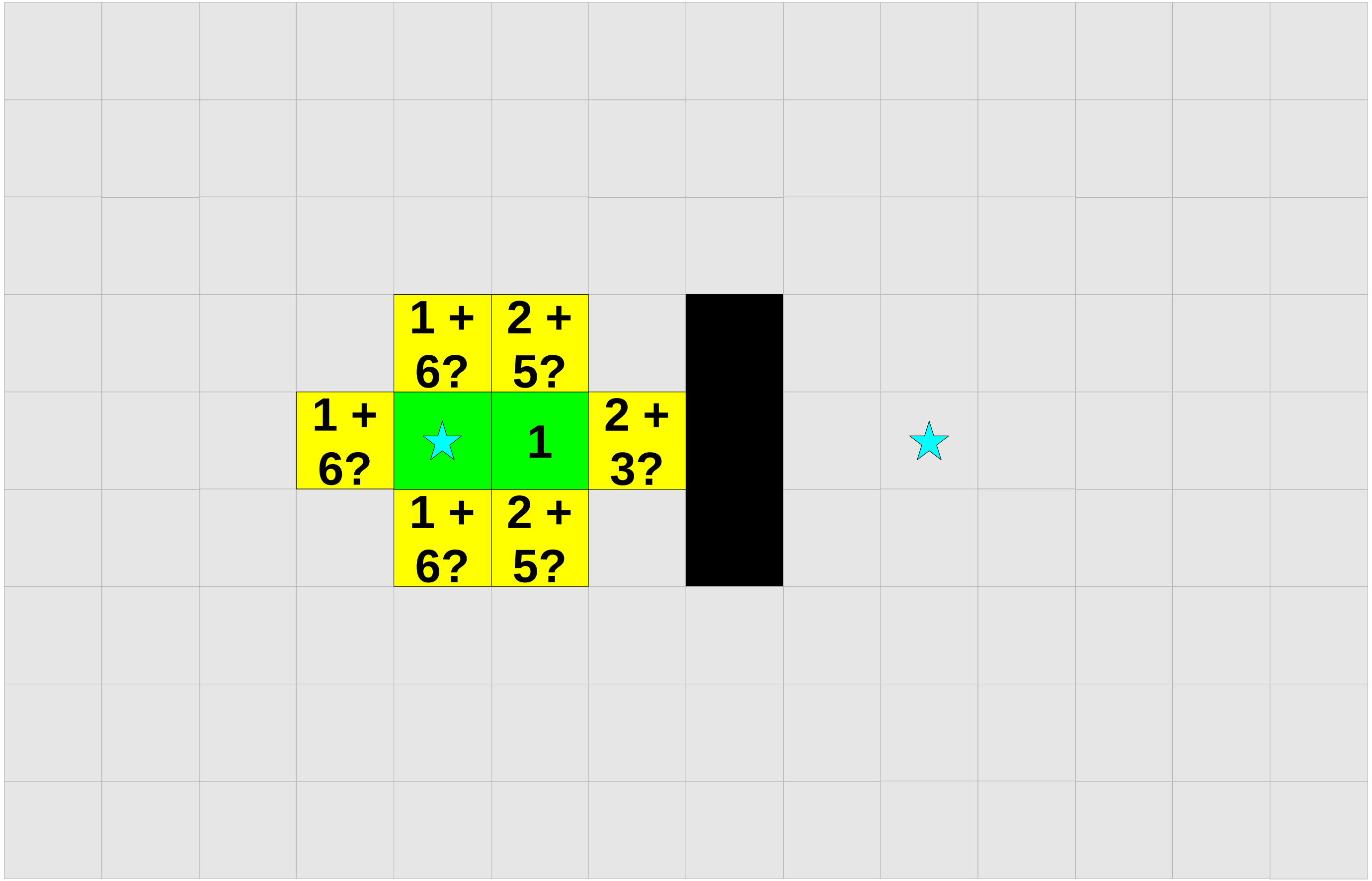
https://youtu.be/DlkMs4ZHHr8

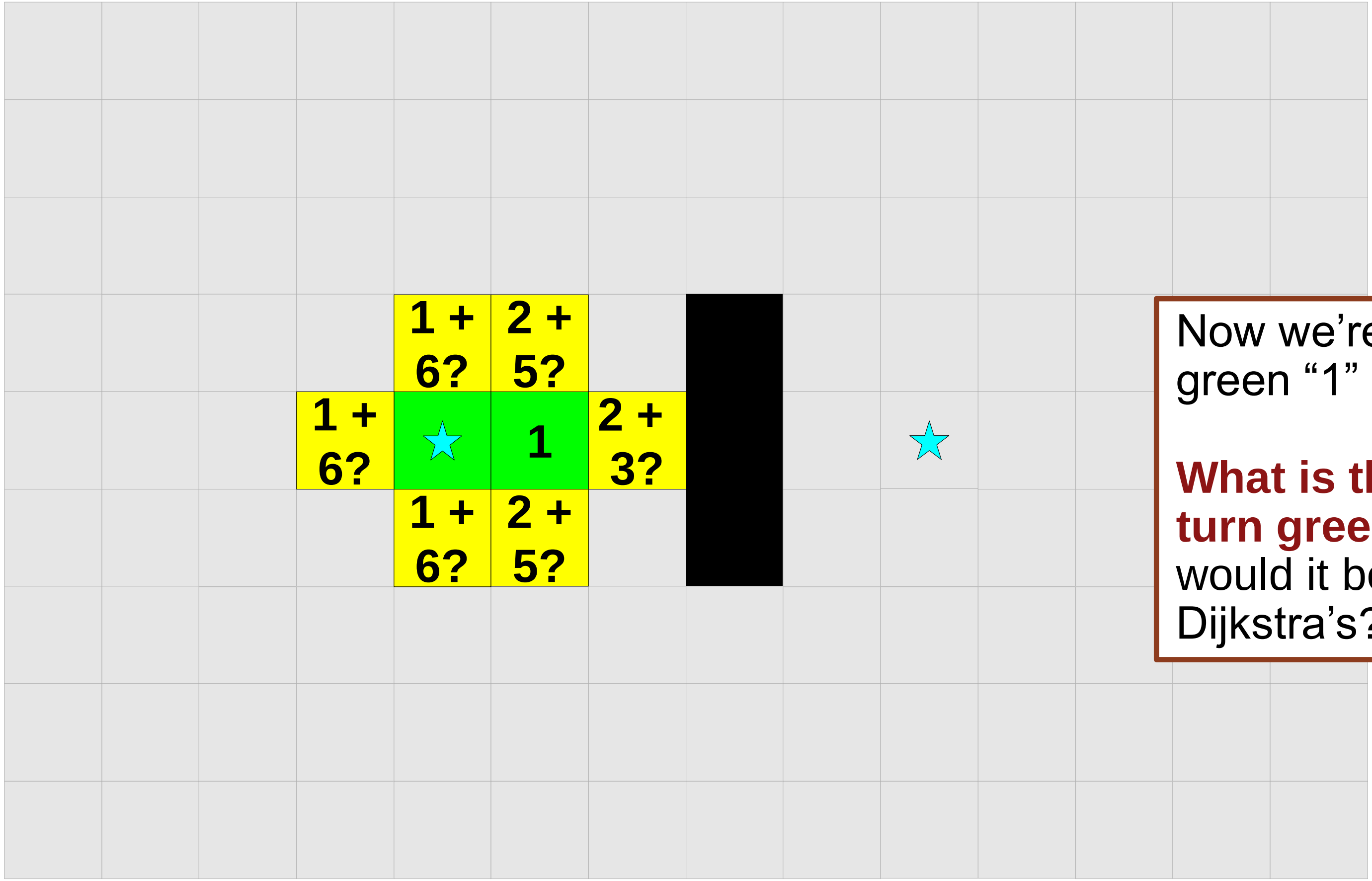


What goes in the ??? ?

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

A*: enqueue neighbors.

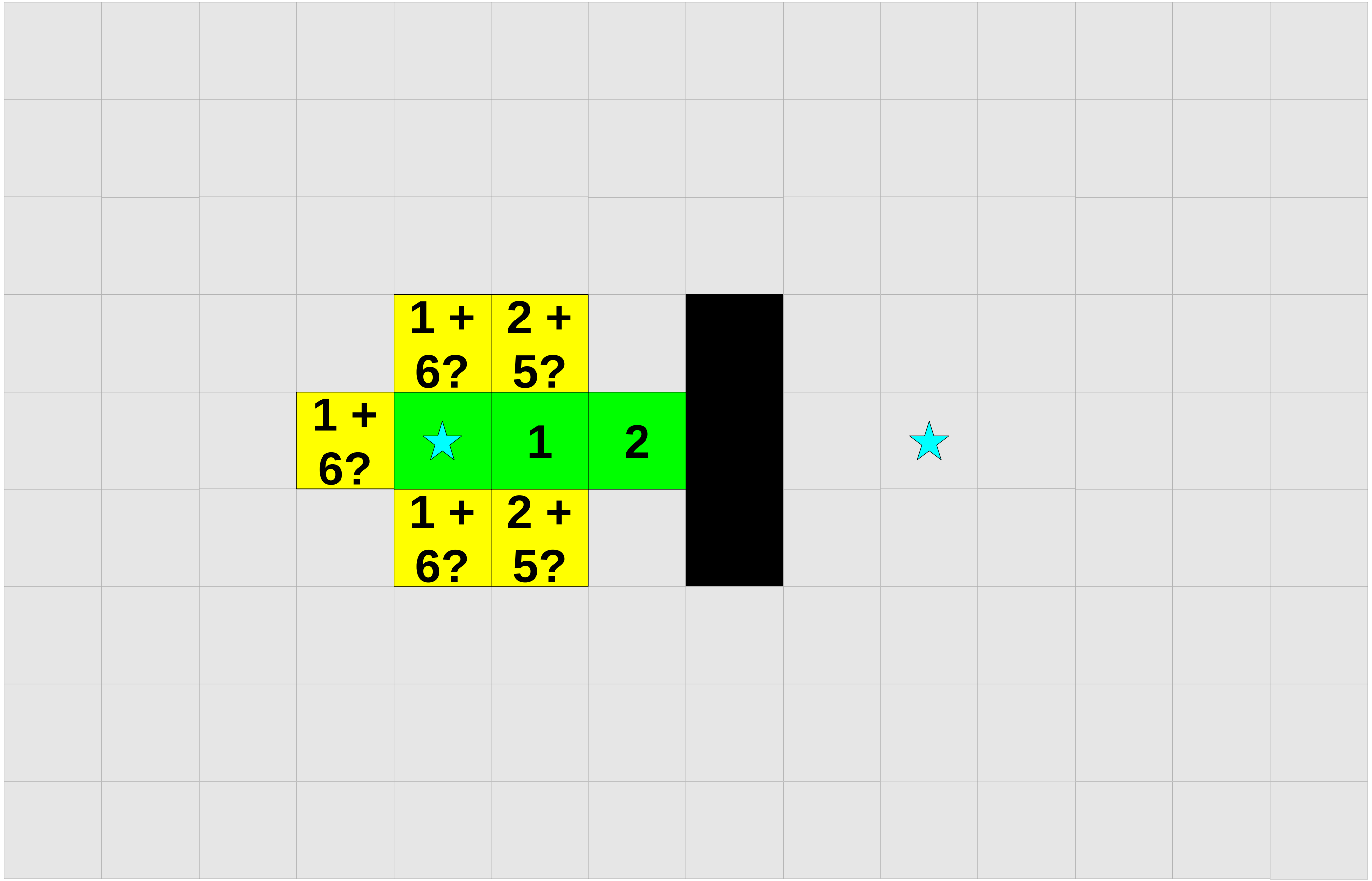




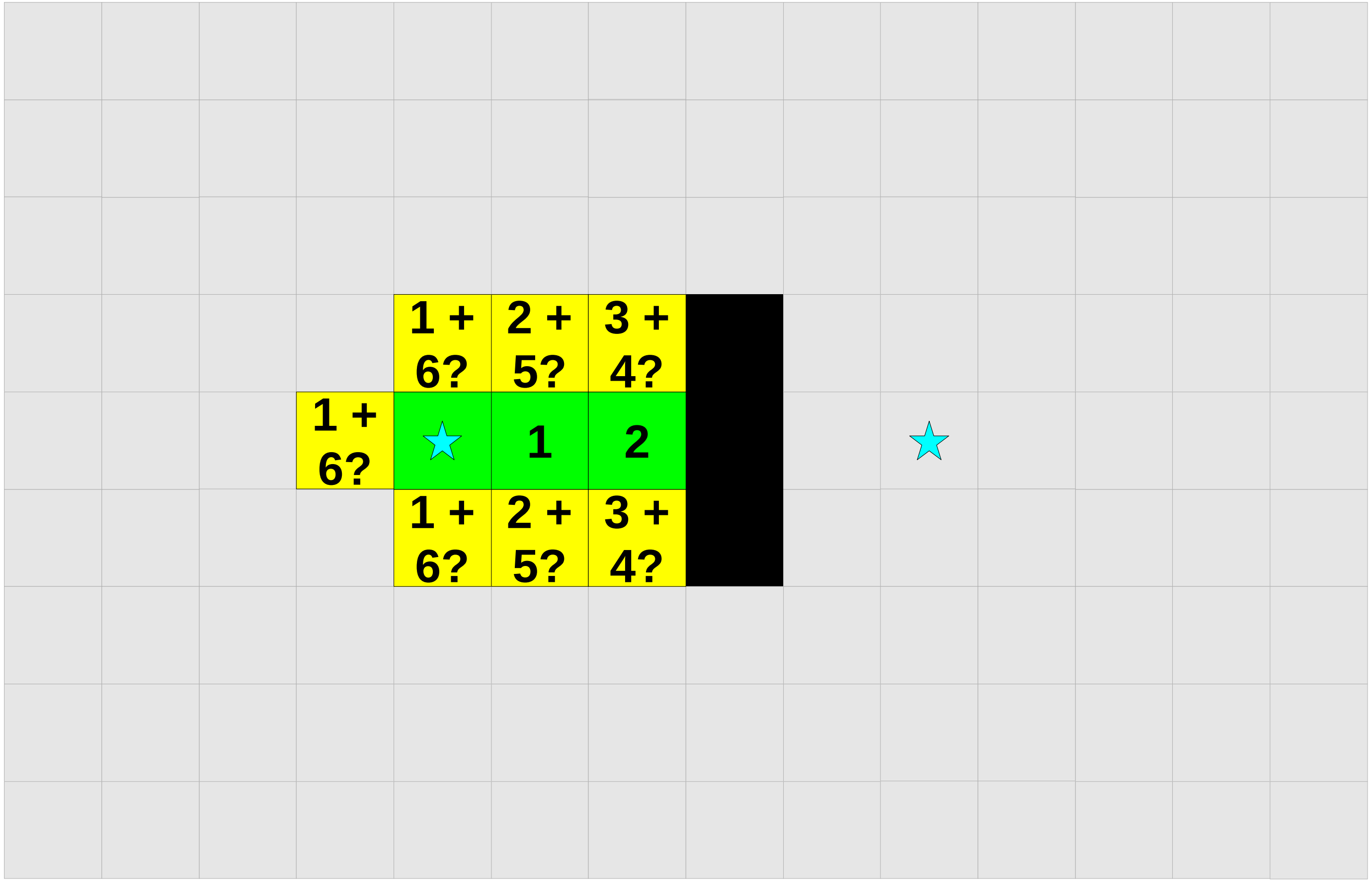
Now we're done with the green "1" node's turn.

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

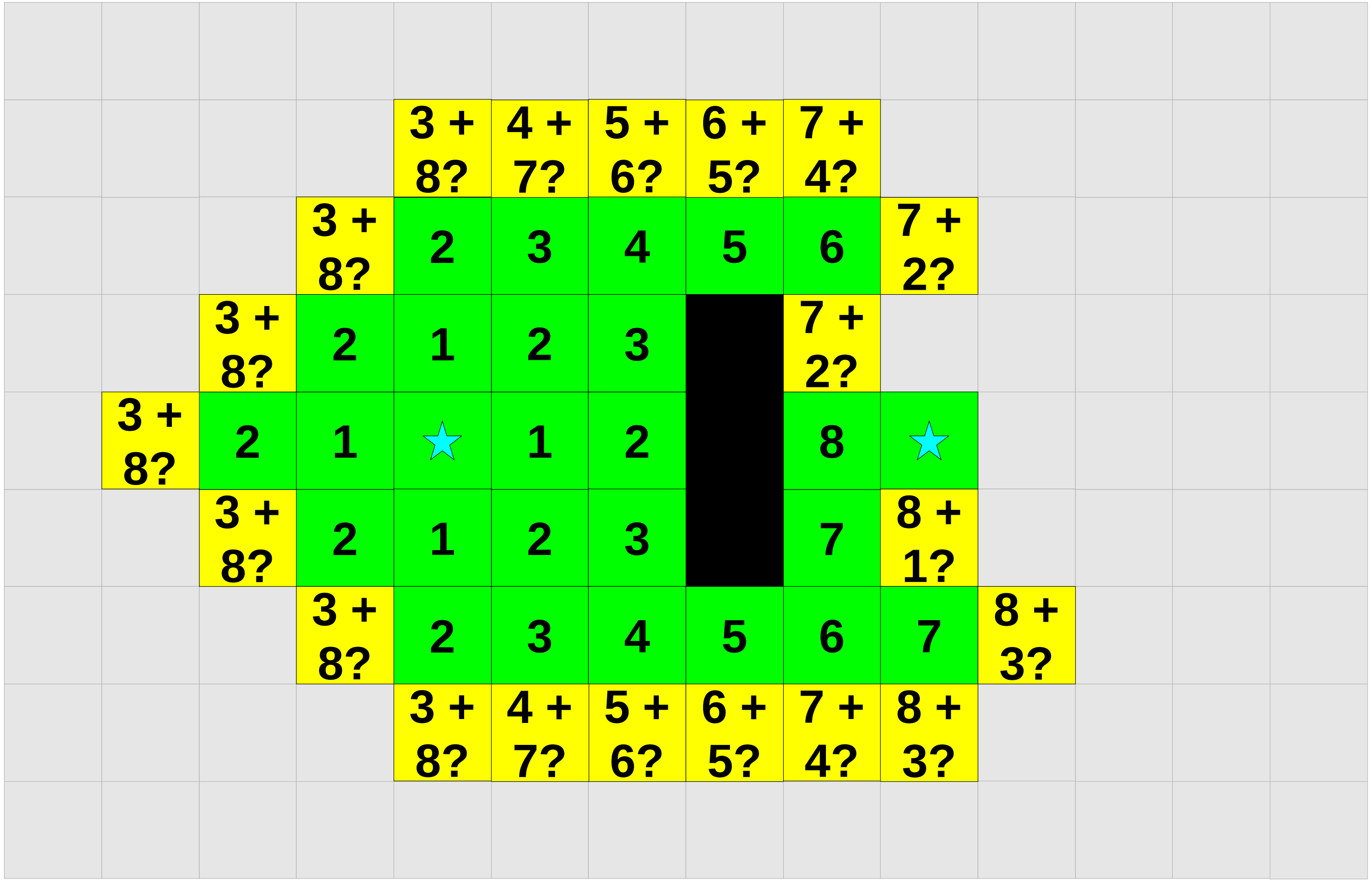
A*: dequeue next lowest priority value node. Notice we are making a straight line right for the end point, not wasting time with other directions.



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

8	7	6	5	4	5	6	7	8	9?				
7	6	5	4	3	4	5	6	7	8	9?			
6	5	4	3	2	3	4	5	6	7	8	9?		
5	4	3	2	1	2	3		7	8	9?			
4	3	2	1	★	1	2		8	★				
5	4	3	2	1	2	3		7	8	9?			
6	5	4	3	2	3	4	5	6	7	8	9?		
7	6	5	4	3	4	5	6	7	8	9?			
8	7	6	5	4	5	6	7	8	9?				

Dijkstra's Algorithm

- 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 **0**.
- 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***.
  - 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***.

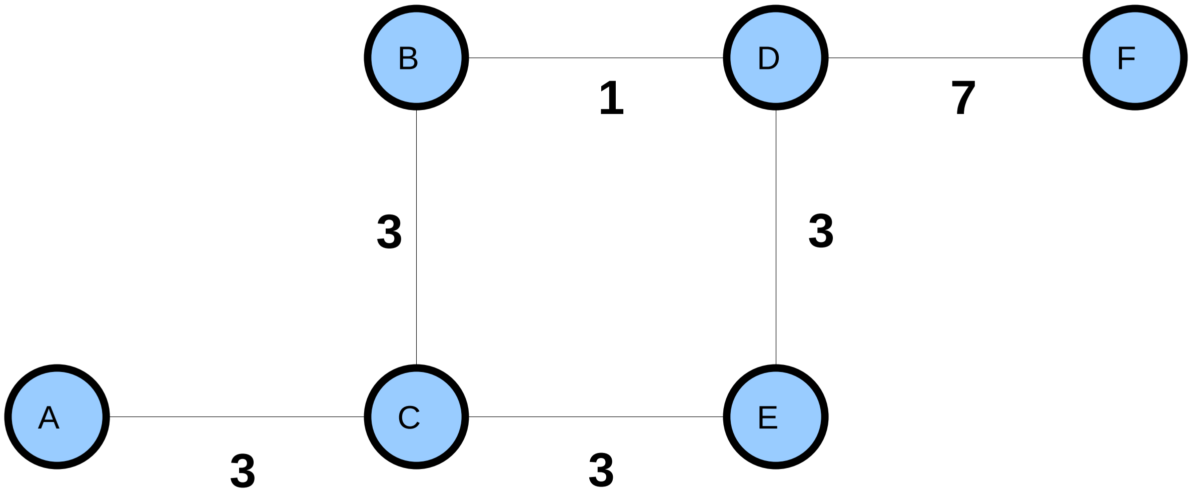
A* Search

- 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* Solving Super Mario (video)

https://youtu.be/DlkMs4ZHHr8

Minimum Spanning Tree



How many distinct minimum spanning trees are in this graph?

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

Prim's Algorithm

Prim's algorithm

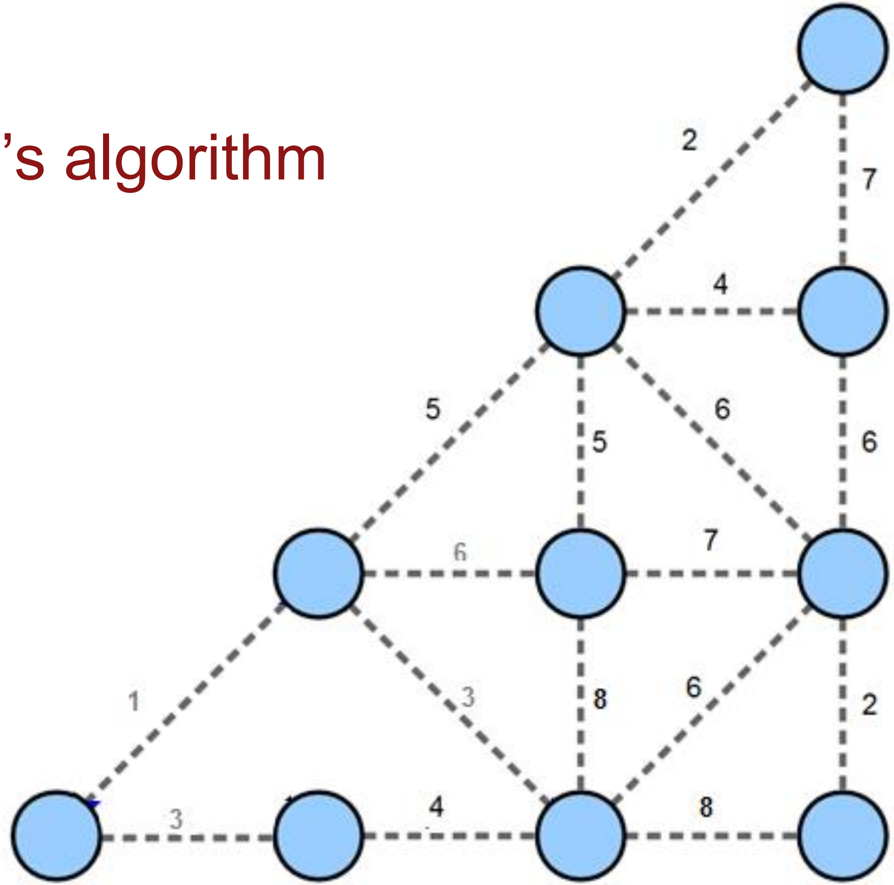
Arbitrarily choose start vertex

Add start vertex to MST

While vertices in MST < total vertices:

- Examine all edges that leave the current MST
- Choose the smallest one
- Add the end vertex of that edge to the MST

Prim's algorithm



Kruskal’s Algorithm

Kruskal's algorithm

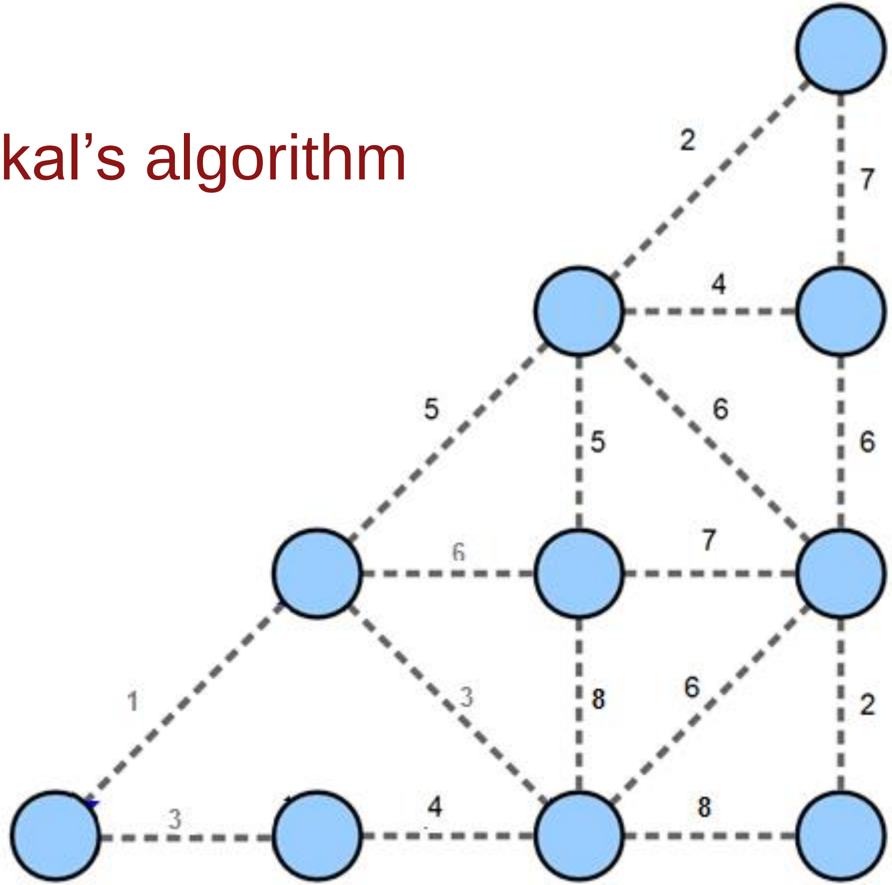
Remove all edges from graph

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



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

Efficiency of this step is key

Cluster management questions

The assignment handout asks you to consider questions such as:

- How will you keep track of which nodes are in each cluster?
- How will you determine which cluster a node belongs to?
- How will you merge together two clusters?

Cluster management strategies

[watch lecture for whiteboard hints]

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.”

In this case **“trees”** simply means **graphs with no cycles**

“with n = 10” (i.e., has **10 nodes**)

“homeomorphically irreducible”

- **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
- Have to be actually different
  - Ignore superficial changes in rotation or angles of drawing