CS 61B DISCUSSION 12

TA: Sherdil Niyaz

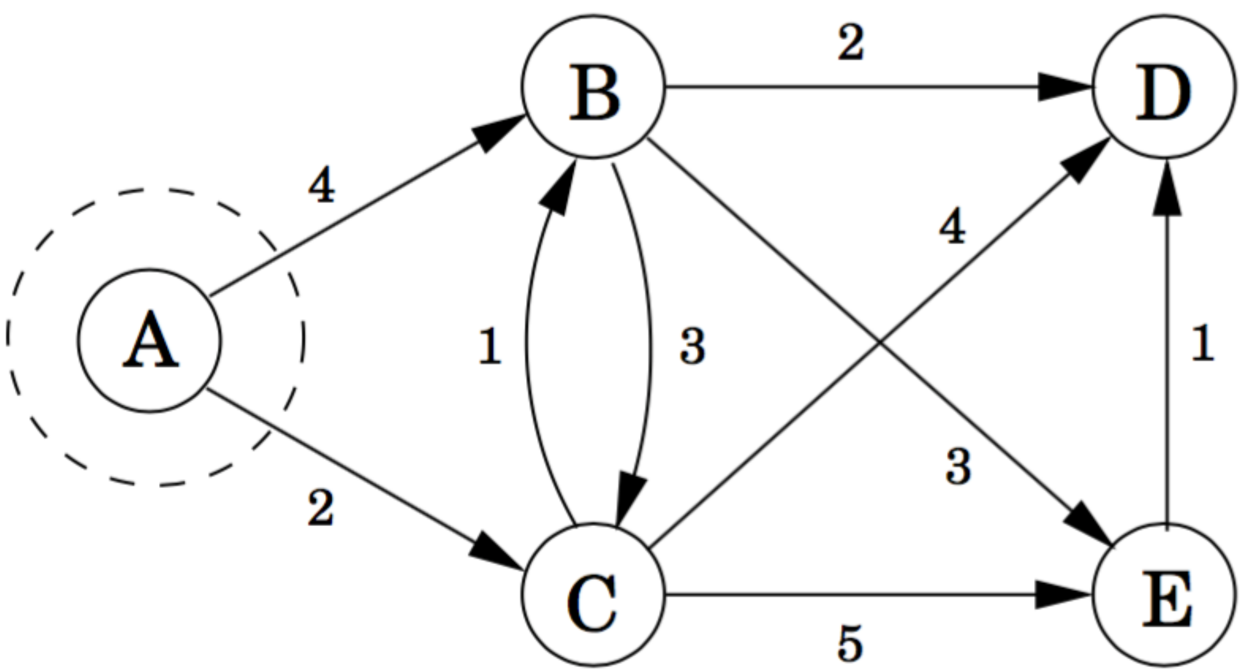
ADMINISTRIVIA

- Project 3   T__T
- A* will be covered this section, so hopefully this helps a bit.
- I made a dumb last section by skipping the Topological Sort problem. I uploaded a correction pdf on our section site.

Dijkstra’s Algorithm!

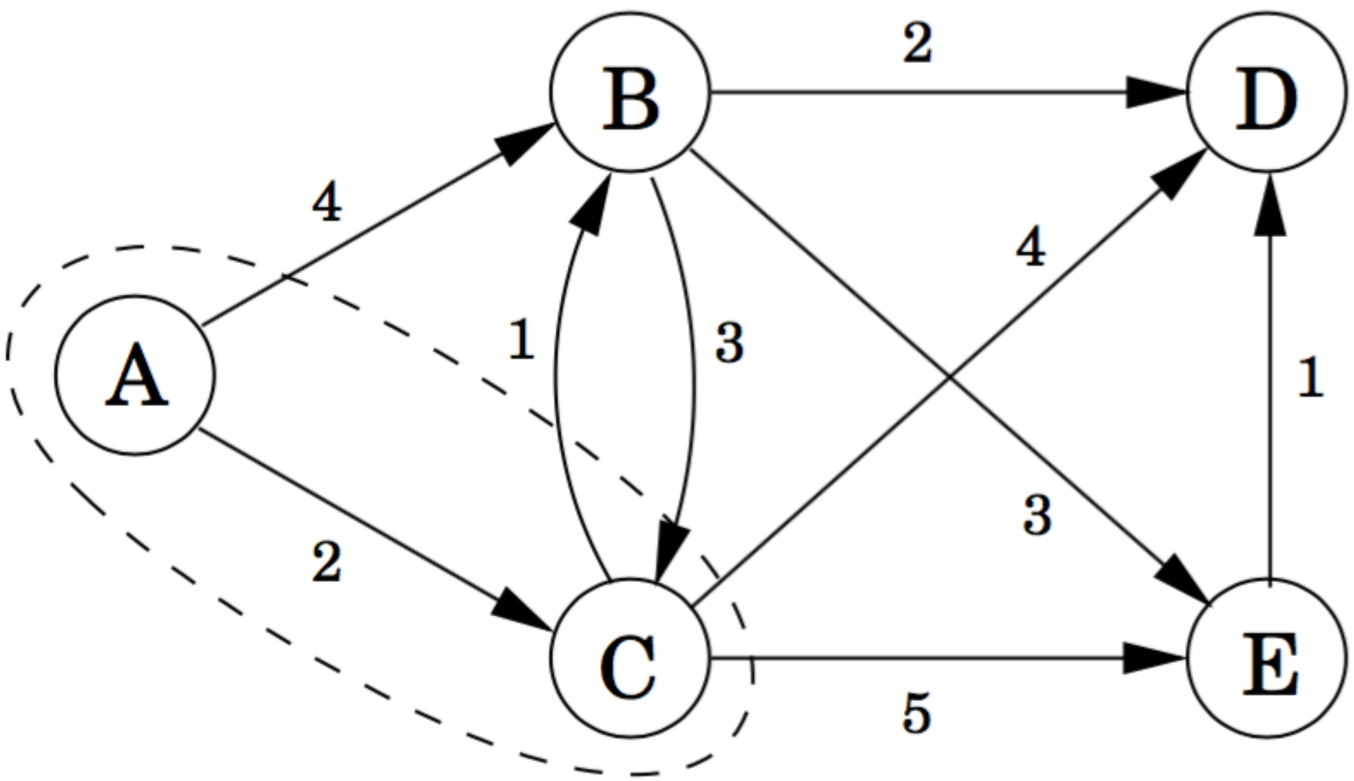
Dijkstra’s Algorithm!

- Main idea: find shortest path/shortest distance from start node in graph to every other node.
- Uses a PQueue, where priorities of nodes are their **distance from the start node**. Call this d(V).
- We pull the closest node off the queue each iteration, and update the distances for its adjacent nodes. Then repeat.

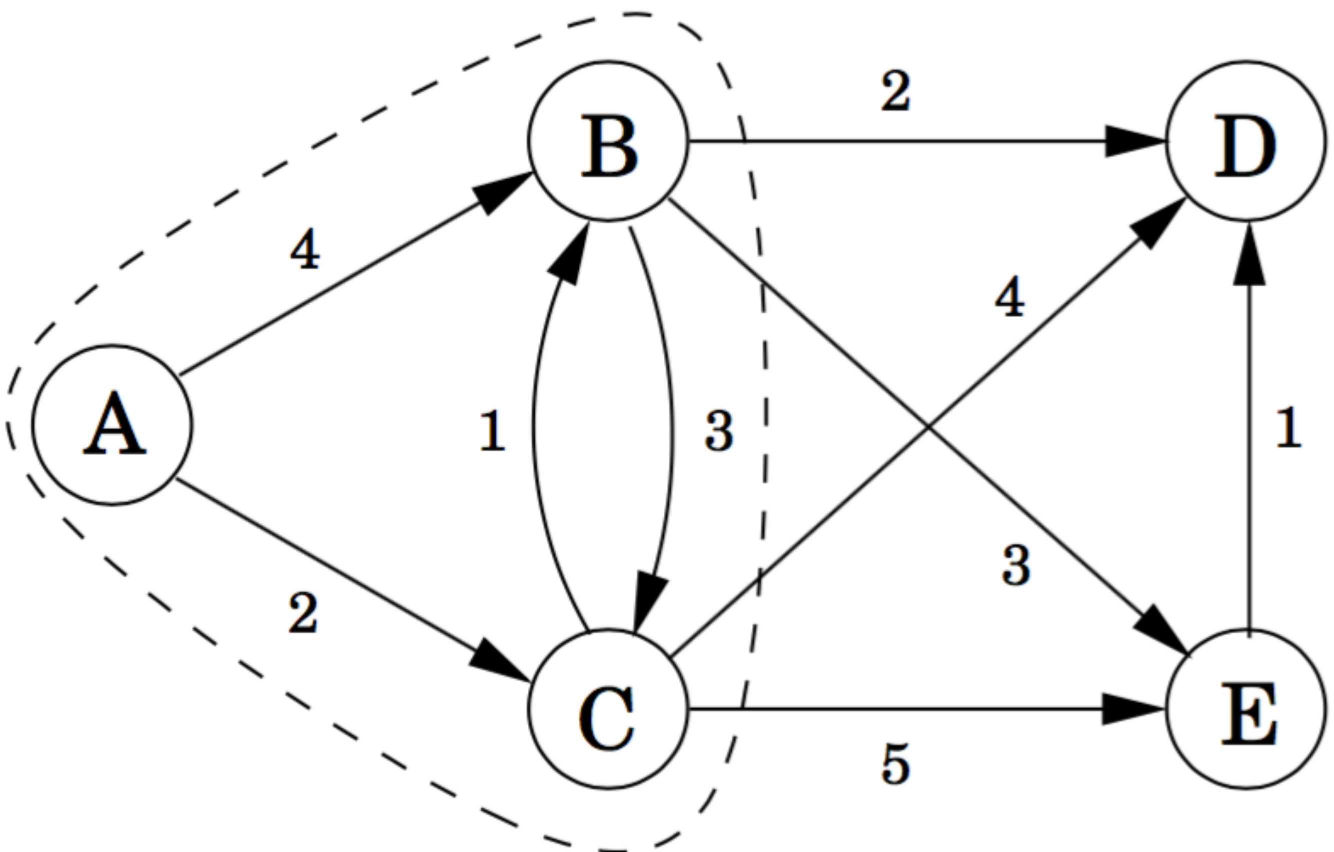


A: 0	D: ∞
B: 4	E: ∞
C: 2	

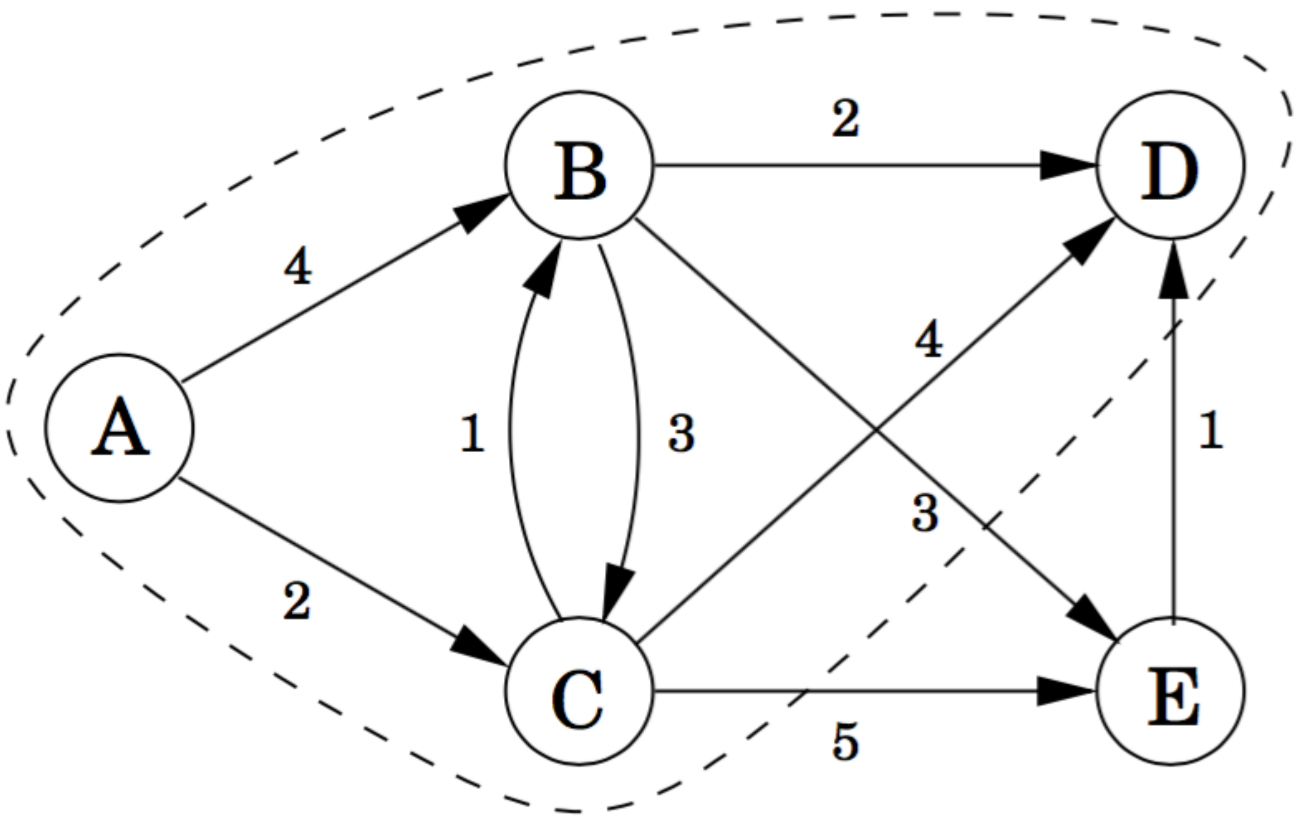
(From the CS 170 Book)



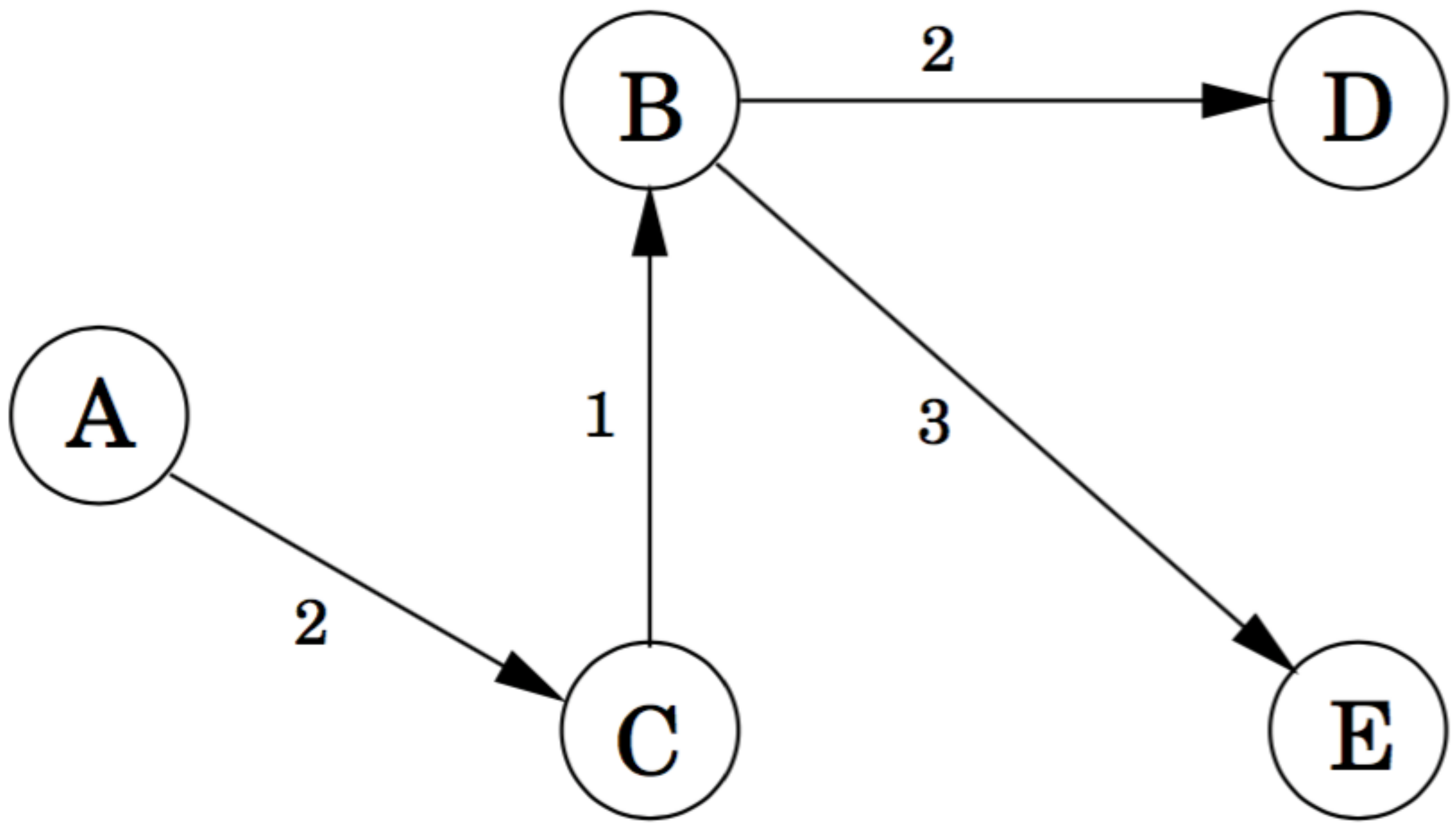
A: 0	D: 6
B: 3	E: 7
C: 2	



A: 0	D: 5
B: 3	E: 6
C: 2	



A: 0	D: 5
B: 3	E: 6
C: 2	



A
B
C
D
E
2
1
2
3
2

Dijkstra’s Algorithm!

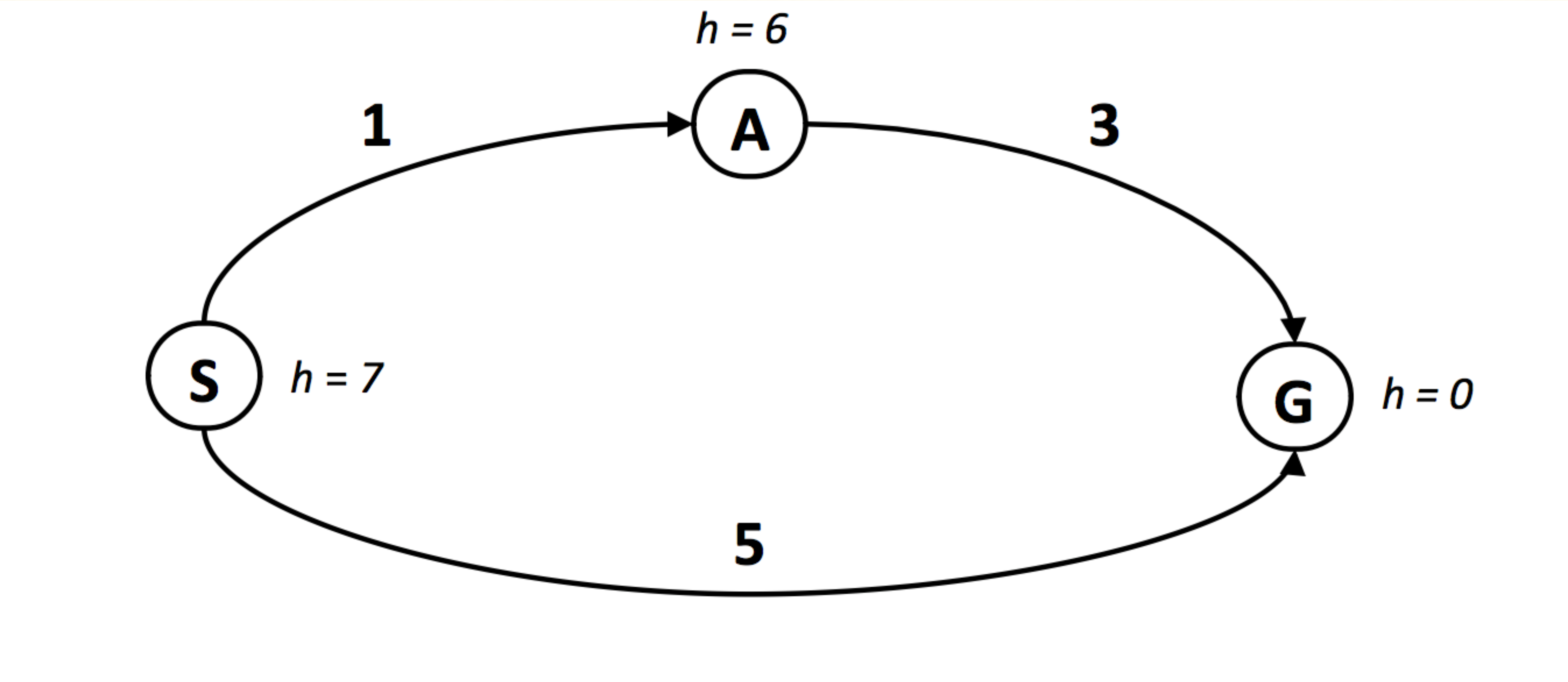
- Notice how we “grew” out an area of exploration, and updates the distances of all nodes that were not in that area. Once a node joined the area, we knew its distance was correct (you’ll prove why in CS 170).
- Runtime is O(E logV)

A*

A*

- Variant of Dijkstra's, but now we are looking for the shortest path/distance from the start node to some **goal** node, not every node in the graph!
- Each node has a **heuristic:** a guess of how far it is from the goal node. This gives A* some "direction" to start exploring from.
- Now we have to change the priorities to match our new goal. The priority of a node is now $d(V) + h(V)$.
- Updating is done in the usual way: pop a node, update the priorities of its neighbors if they can be lowered.

A*



- Whoops.

A*

- A* **only** gives us the shortest path if the heuristic for each node is **admissible**.
- This means that, for each node V in the graph, h(V) is less than or equal to the actual distance from V to the goal.
- Some people say that you need an "optimistic" heuristic because of this (one that never over-estimates the true distance).
- Proof in CS 188 (The AI class).

Minimum Spanning Trees

Minimum Spanning Trees

- Series of edges that connects all nodes in a graph, but that that have minimal total weight.
- Multiple algorithms that are used to find them.
- They use the **cut property:** If you take any cut on a graph, the minimum weight edge crossing that cut must be part of the MST (assuming all edge weights unique, which we do in 61B's proof sketch).
- **Cut:** Just any two sets of node, so long as there is at least one node in each set.

The Cut Property, Illustrated!