Sp18 CS 61B Discussion 12

Welcome!

Wayne Li

wli2@berkeley.edu

https://wayne-li2.github.io/

Administrivia

- Project 3 spec is out (Due 4/18)! Please get started on it as soon as possible.
- Post midterm 2 one-on-ones are still open. (@3258)
- Remember to fill out the mandatory proj2 partner survey here

Quiz Instructions

- If you haven’t yet, please also **neatly** put your email address **outside the name box** if you want to be emailed!
- Bubble number **41**.

Forms

- Want to meet with me specifically?
  - https://goo.gl/forms/3HyKxt0ZAWHKRmij2
- Want to meet with **any** TA (not me)? @3258
- Survey:
  - https://goo.gl/forms/qxZ8qzS47HB62wiE3

Aside

Travelling Salesman Problem

- Given a list of cities and the distances between each pair of cities, what is the shortest possible route that **visits each city** and **returns to the origin city**?
- Such a similar sounding problem to Dijkstra's...

Travelling Salesman Problem

- Given a list of cities and the distances between each pair of cities, what is the shortest possible route that **visits each city** and **returns to the origin city**?
- Such a similar sounding problem to Dijkstra's…
  - But is actually incredibly hard! $O(n2^n)$ with dynamic programming.

Travelling Salesman Problem

- What do we do when presented with a difficult problem?

Travelling Salesman Problem

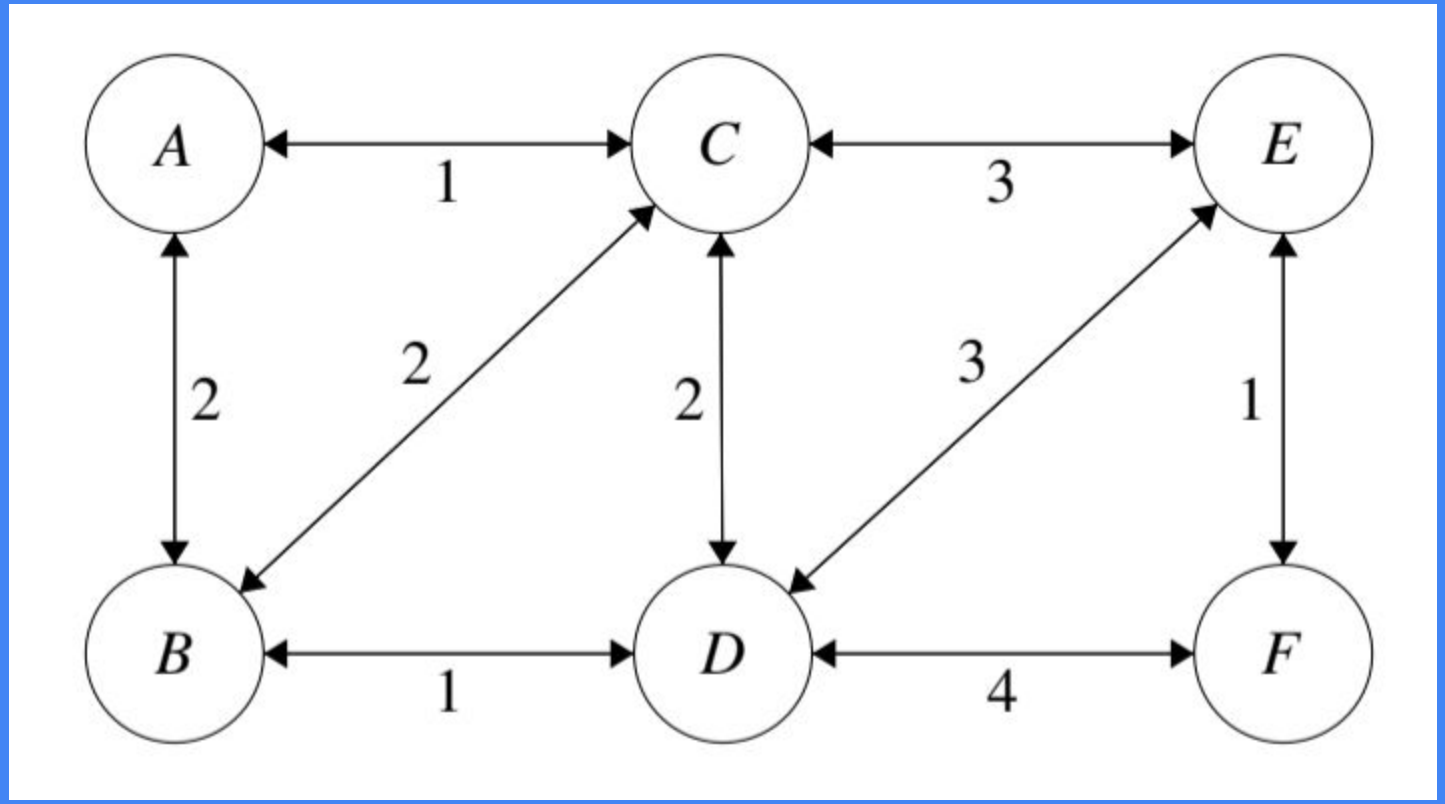
- What do we do when presented with a difficult problem?
  - Solve an approximation.
  - Use heuristics.

Travelling Salesman Problem

- https://www.youtube.com/watch?v=SC5CX8drAtU

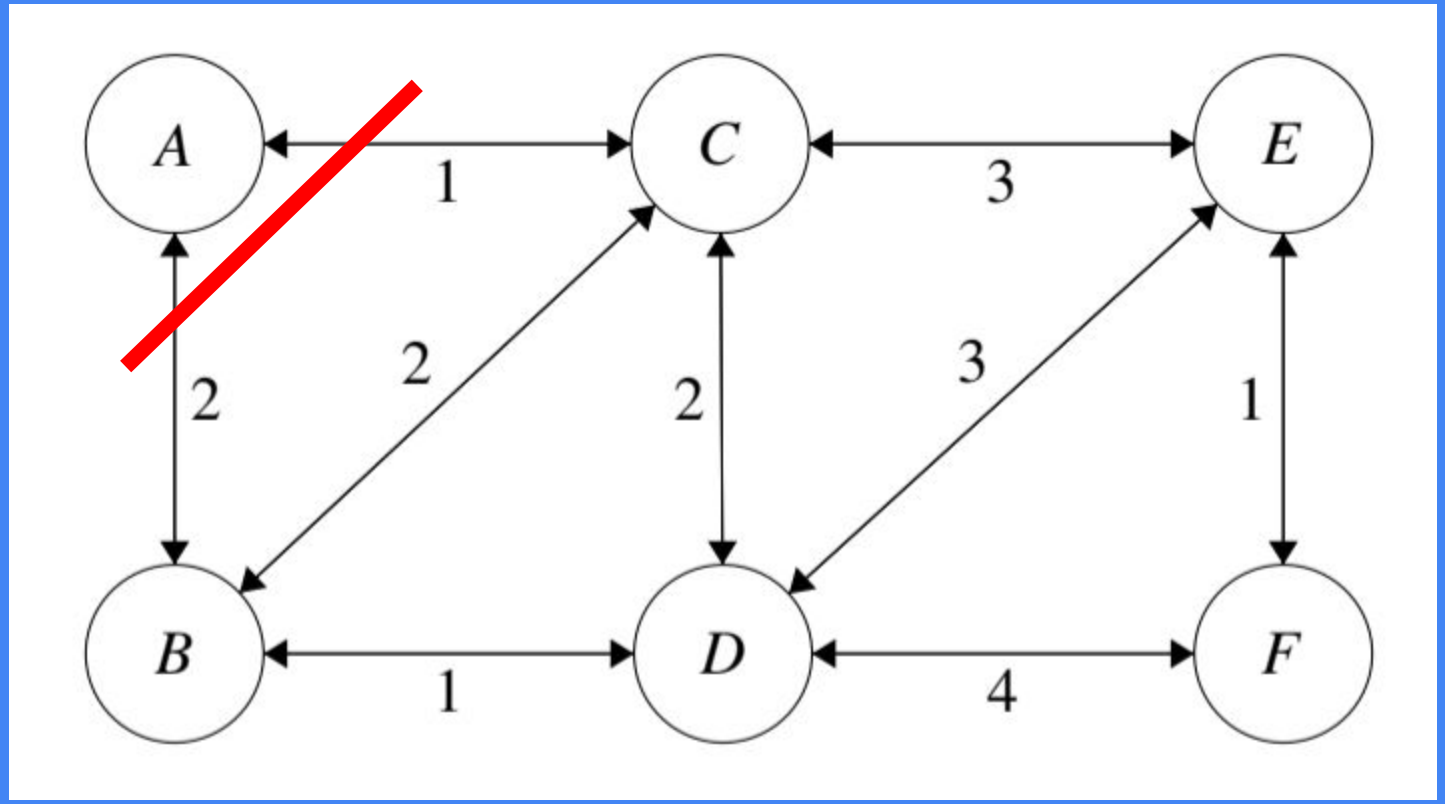
Onto Discussion

Prim’s Algorithm


A
C
E
B
D
F
1
3
2
2
2
3
1
1
4

Prim’s Algorithm Intuition

- Let’s say we start at A.

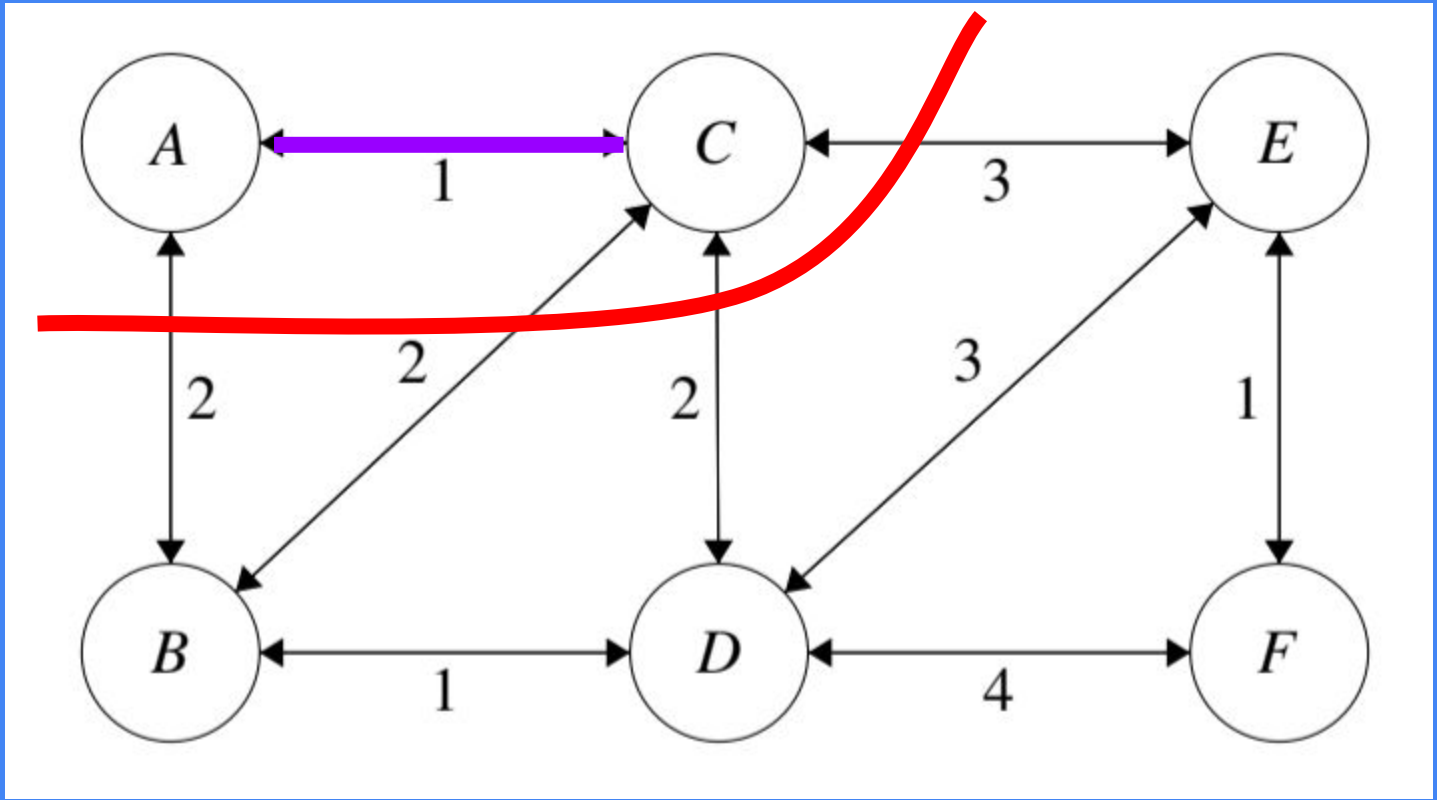

A
C
E
B
D
F
1
3
2
2
2
3
1
1
4

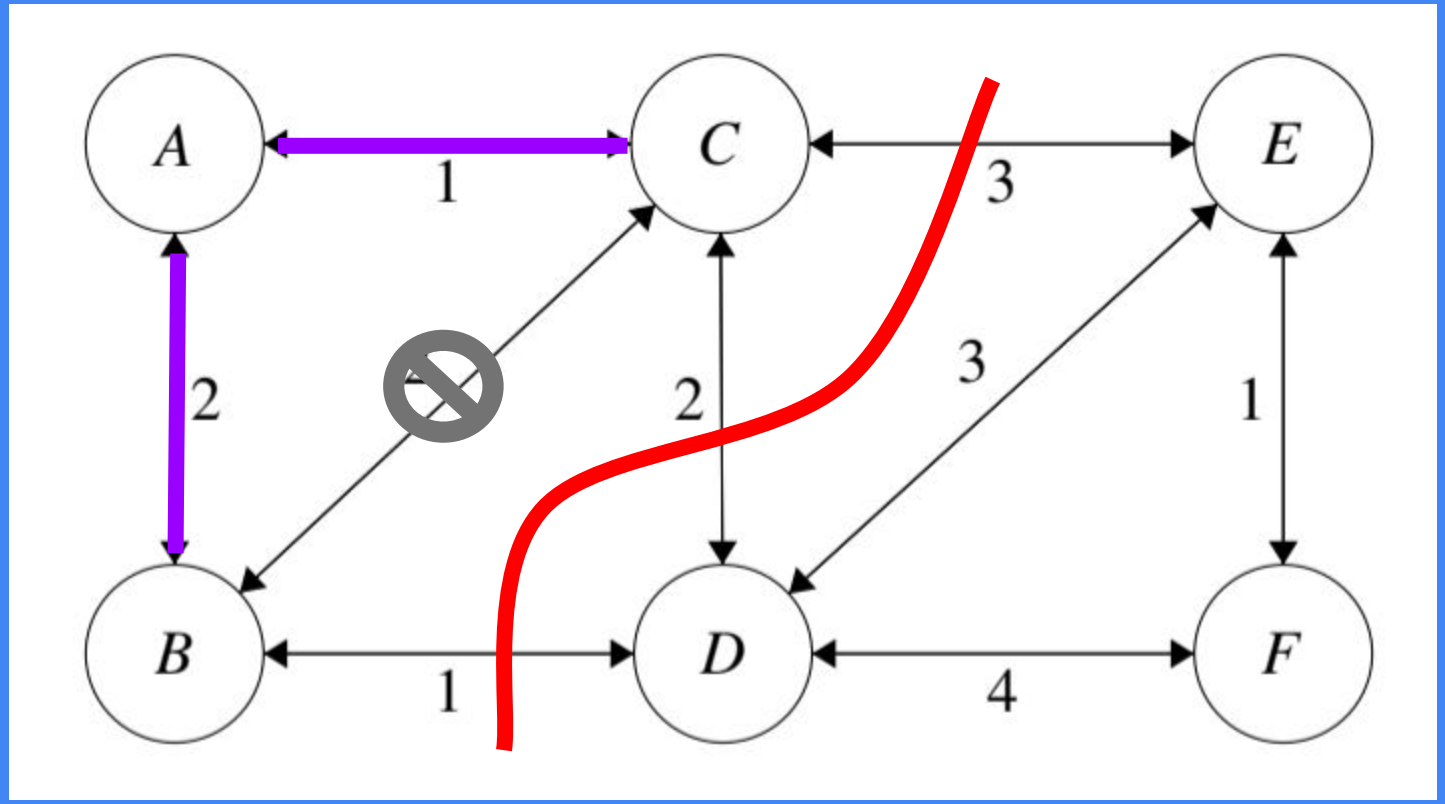
Prim’s Algorithm Intuition

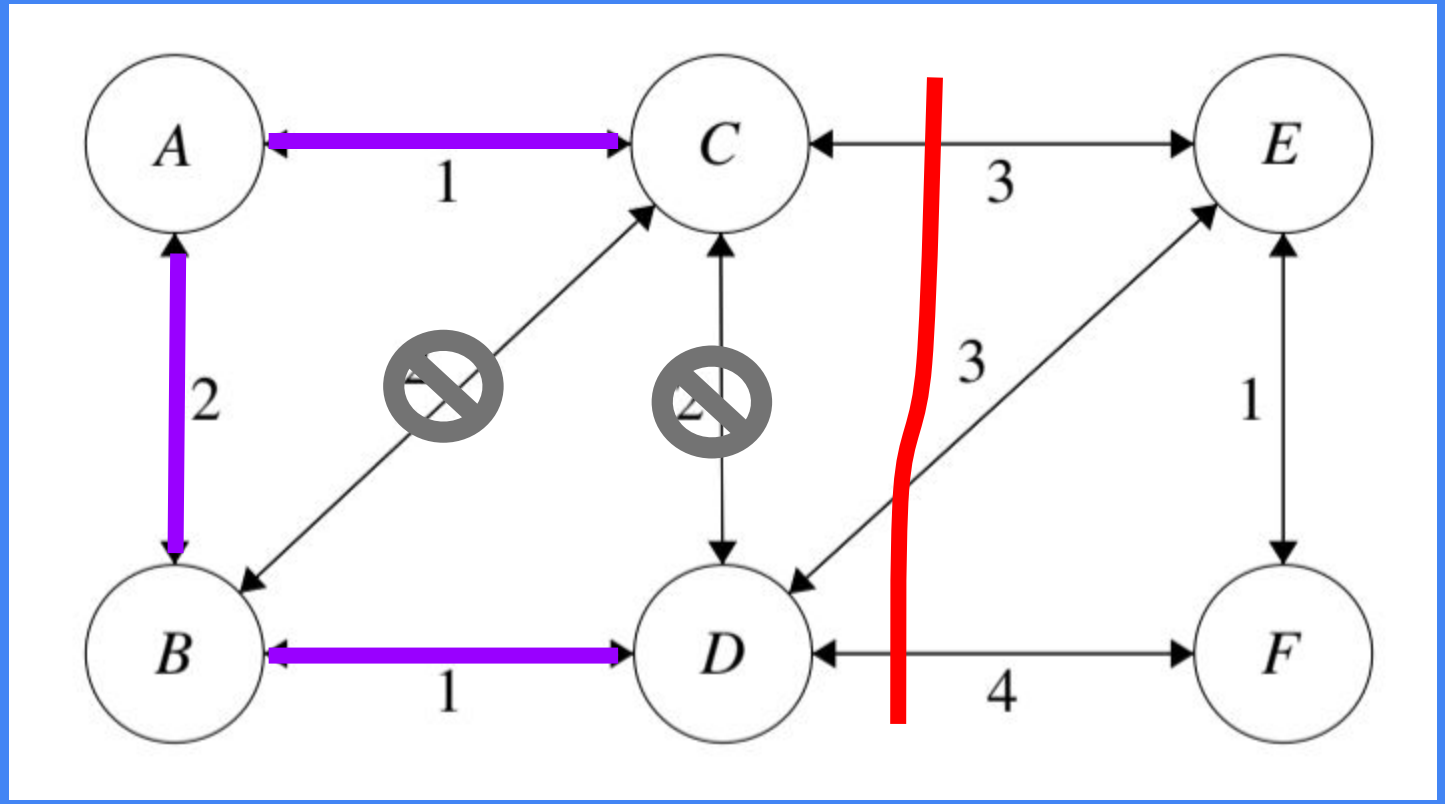
- If I told you I already have an MST made for the graph containing B, C, D, E, and F.
- Meaning we just need to connect A to the rest of the graph via A->B or A->C.
- Which one would you pick?

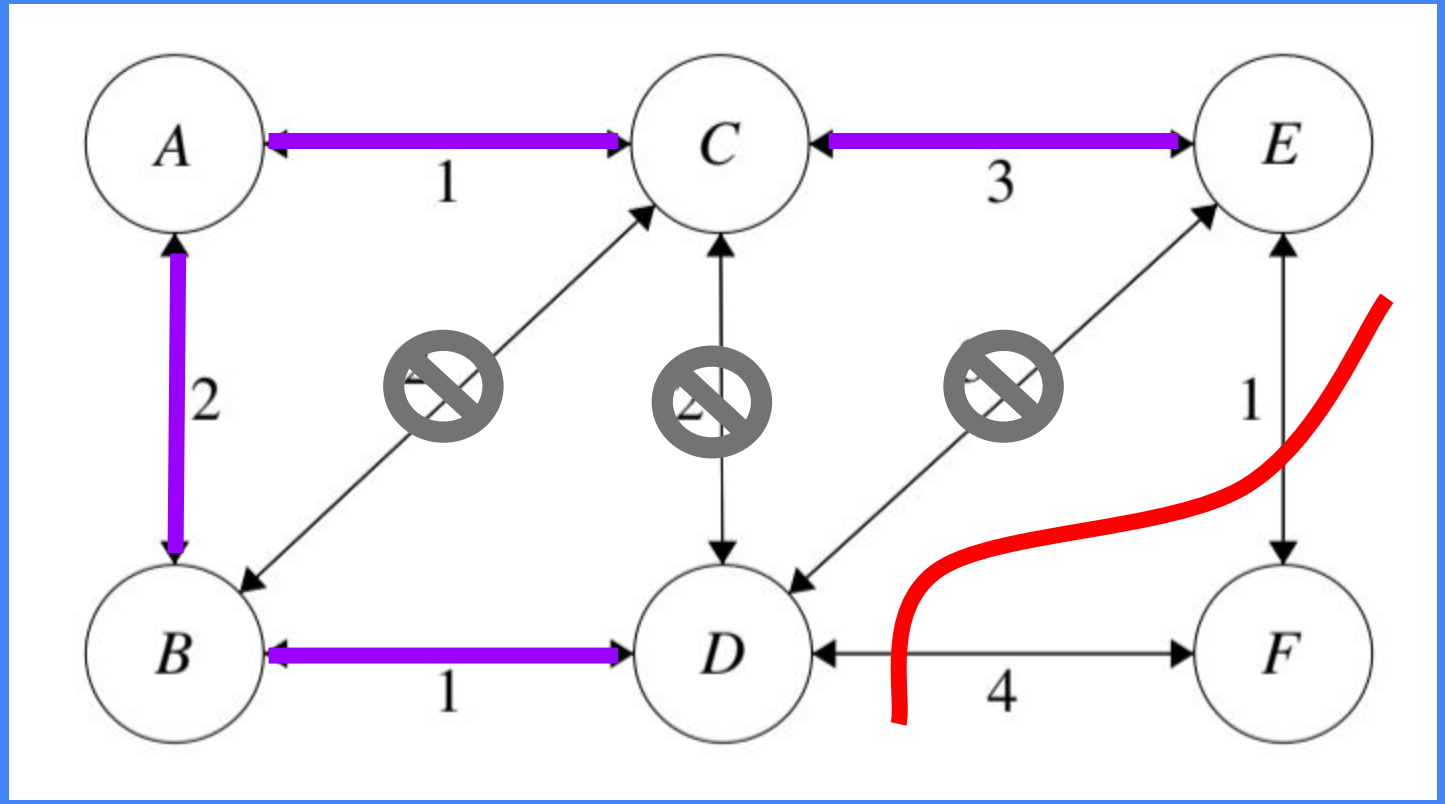
Prim’s Algorithm Intuition

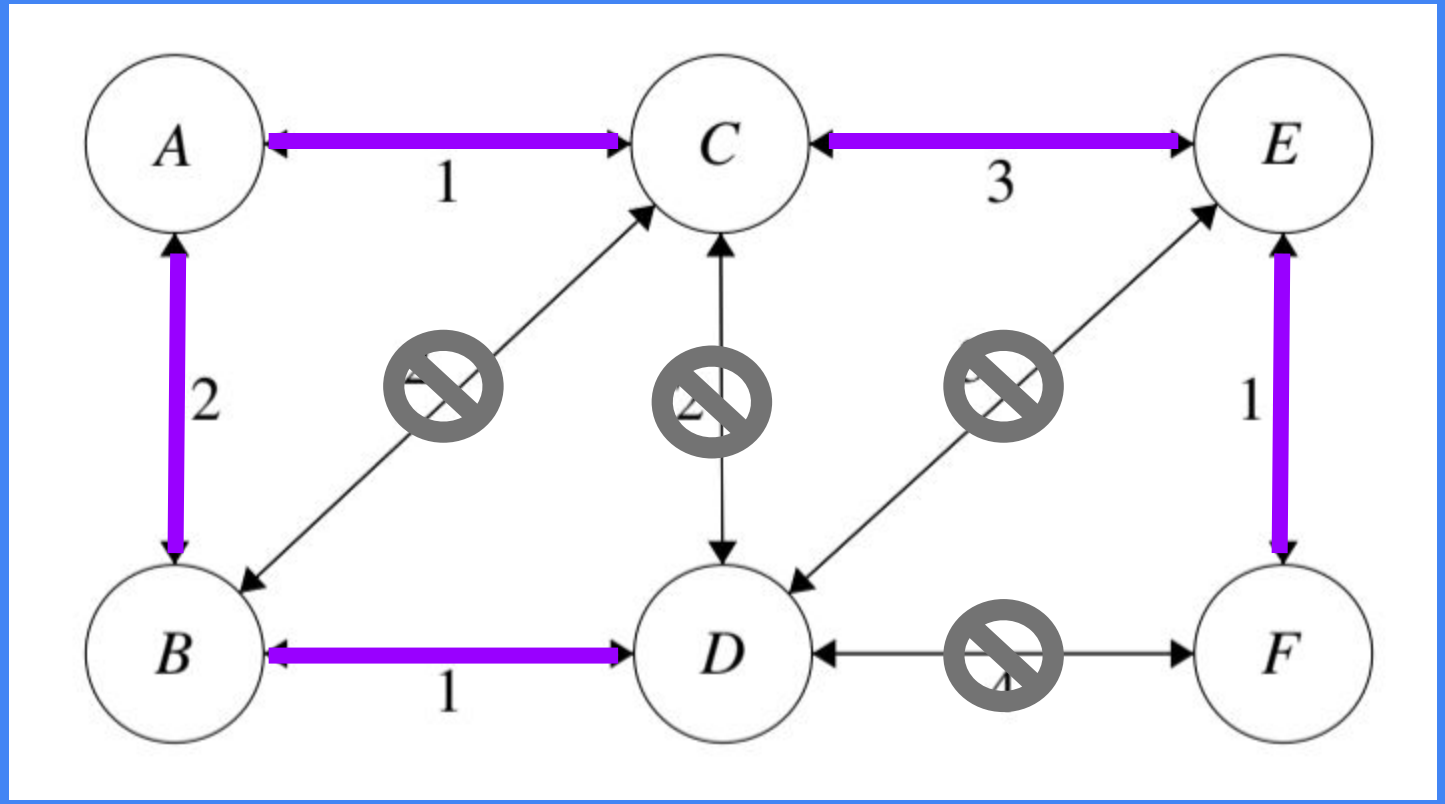
- If I told you I already have an MST made for the graph containing B, C, D, E, and F.
- Meaning we just need to connect A to the rest of the graph via A->B or A->C.
- Which one would you pick?
  - Pick edge **A->C**, because it’s cheaper!

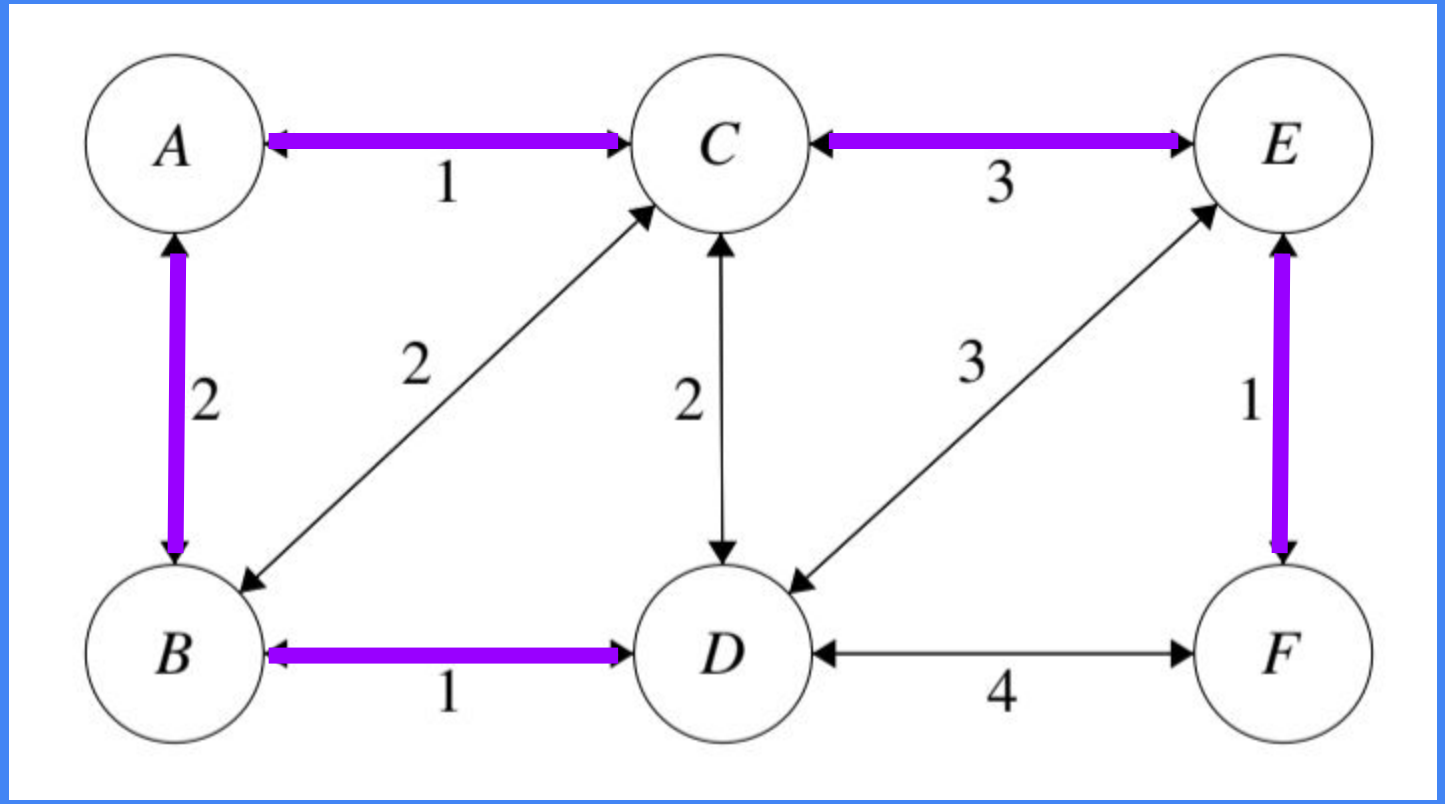

A
C
E
B
D
F
1
3
2
2
2
3
1
1
4


A
C
E
B
D
F
1
3
2
2
3
1
1
4


A
C
E
B
D
F
1
3
2
3
1
1
4


A
C
E
B
D
F
1
3
2
1
1
4

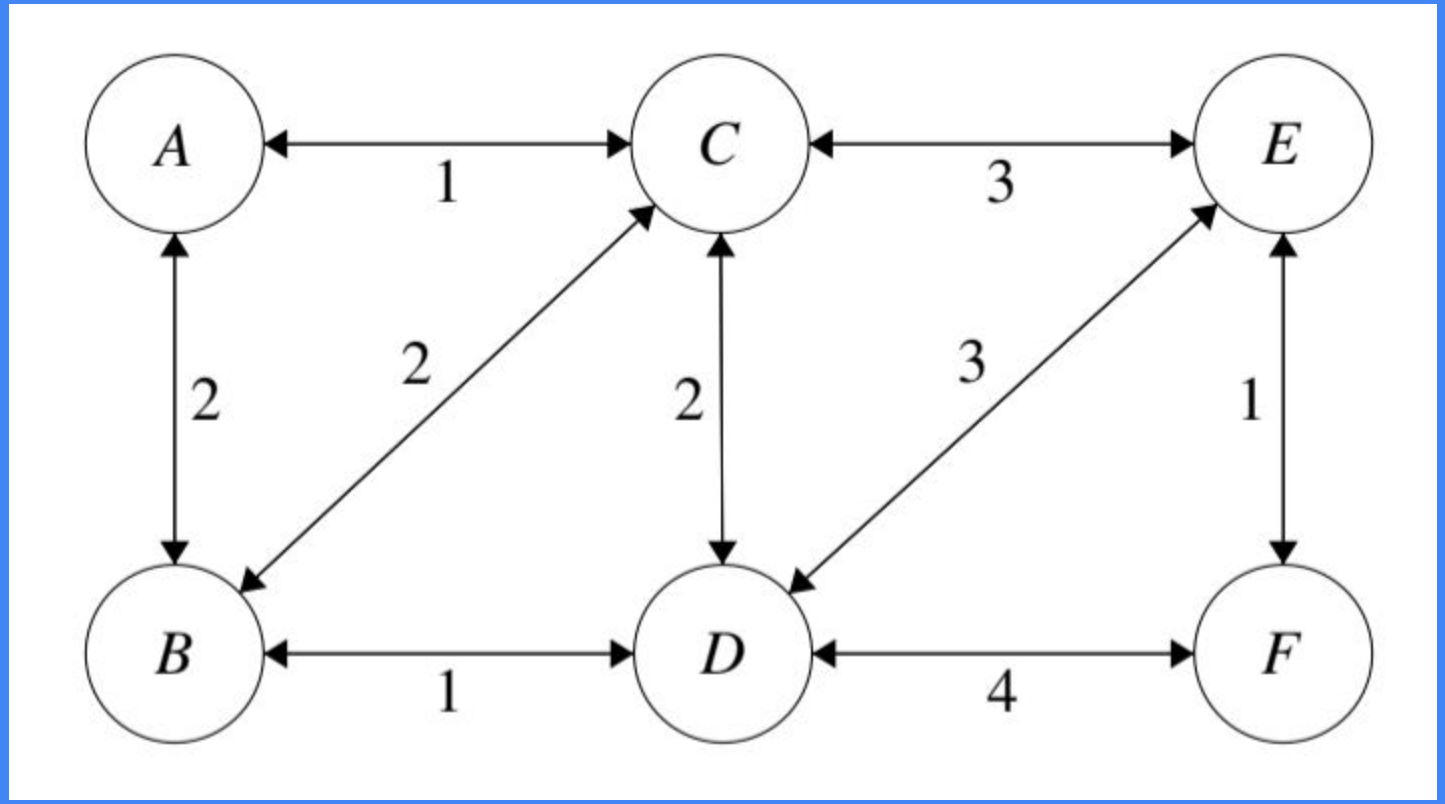

A
C
E
B
D
F
1
3
2
1
1

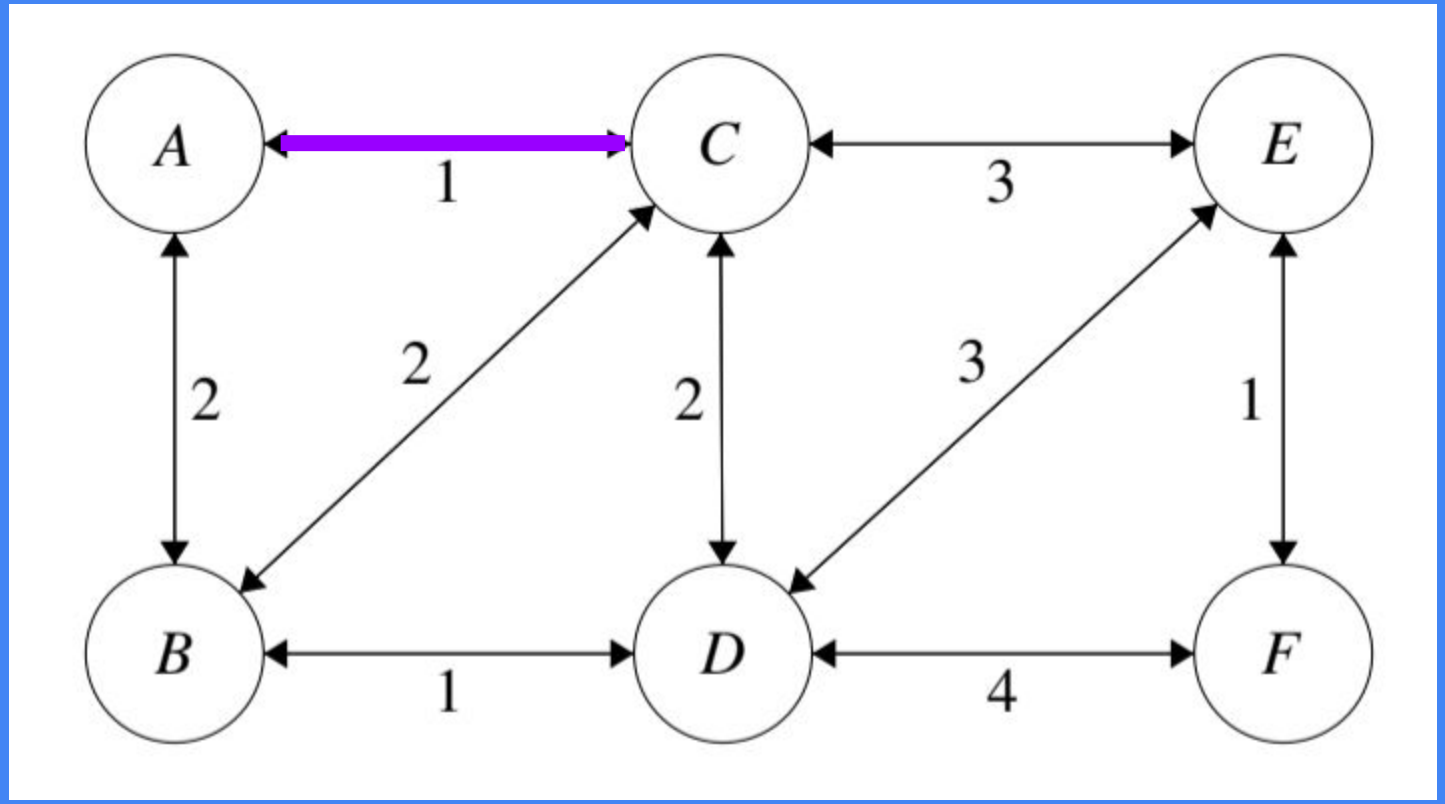

A
C
E
B
D
F
1
3
2
2
2
3
1
1
4

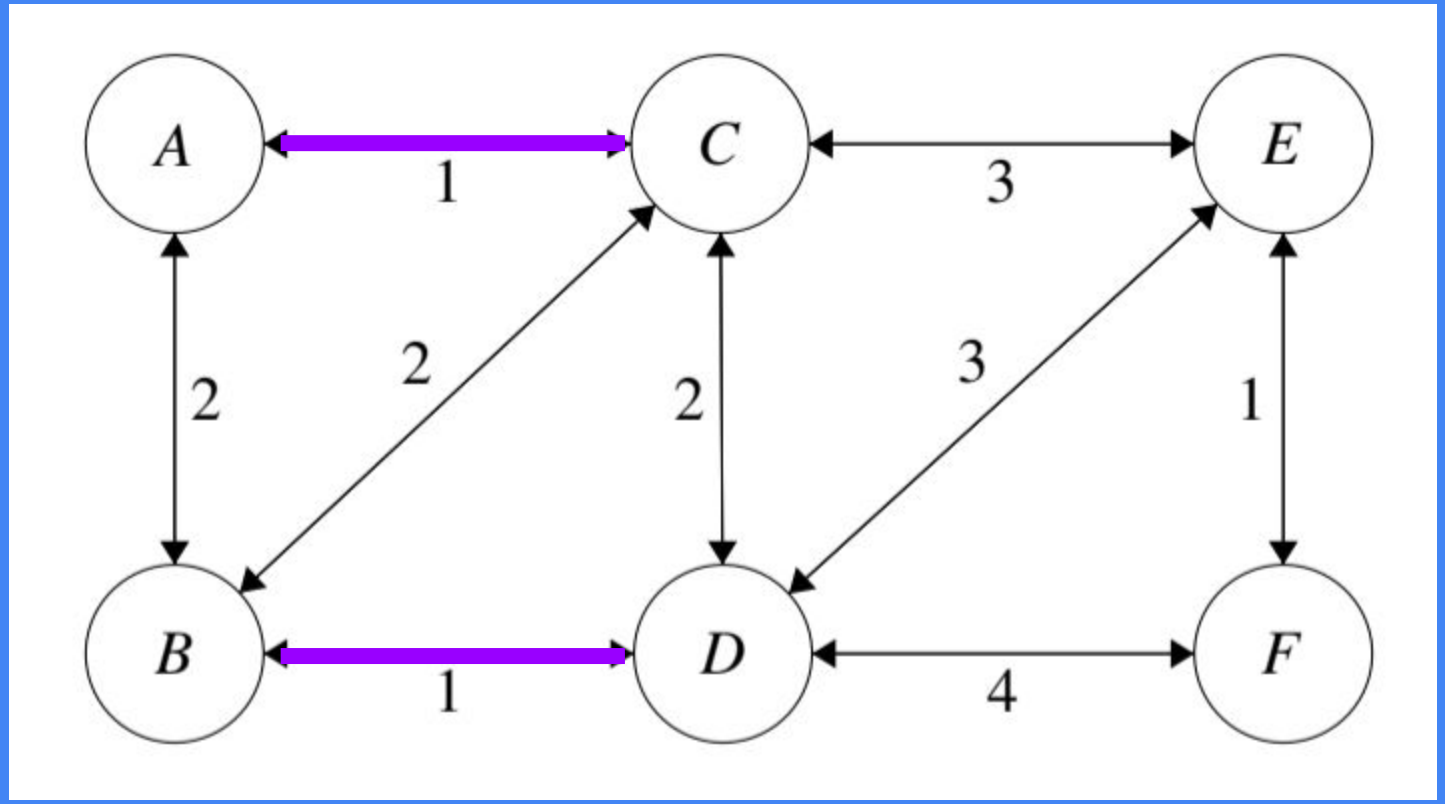
Kruskal’s Algorithm

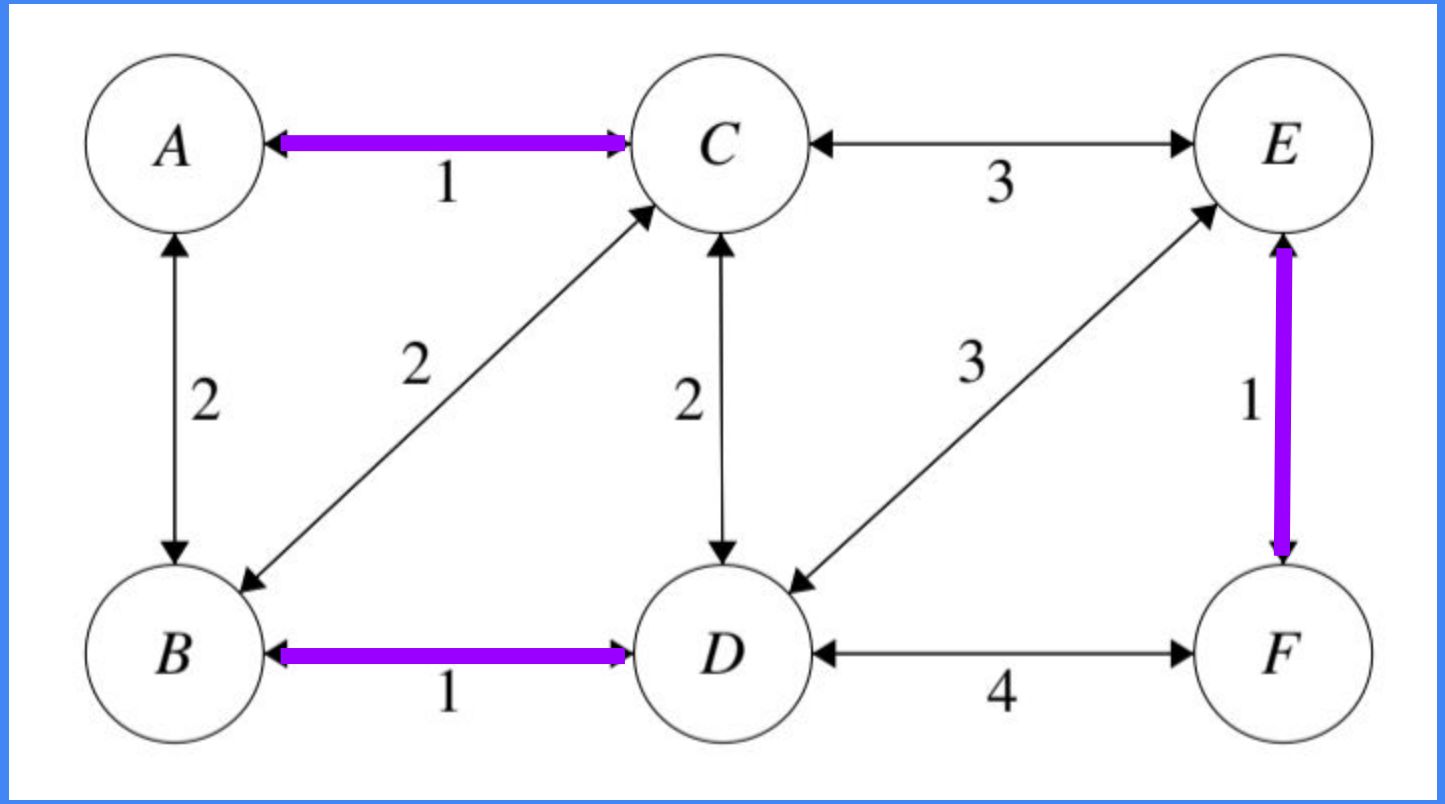
Kruskal’s Algorithm

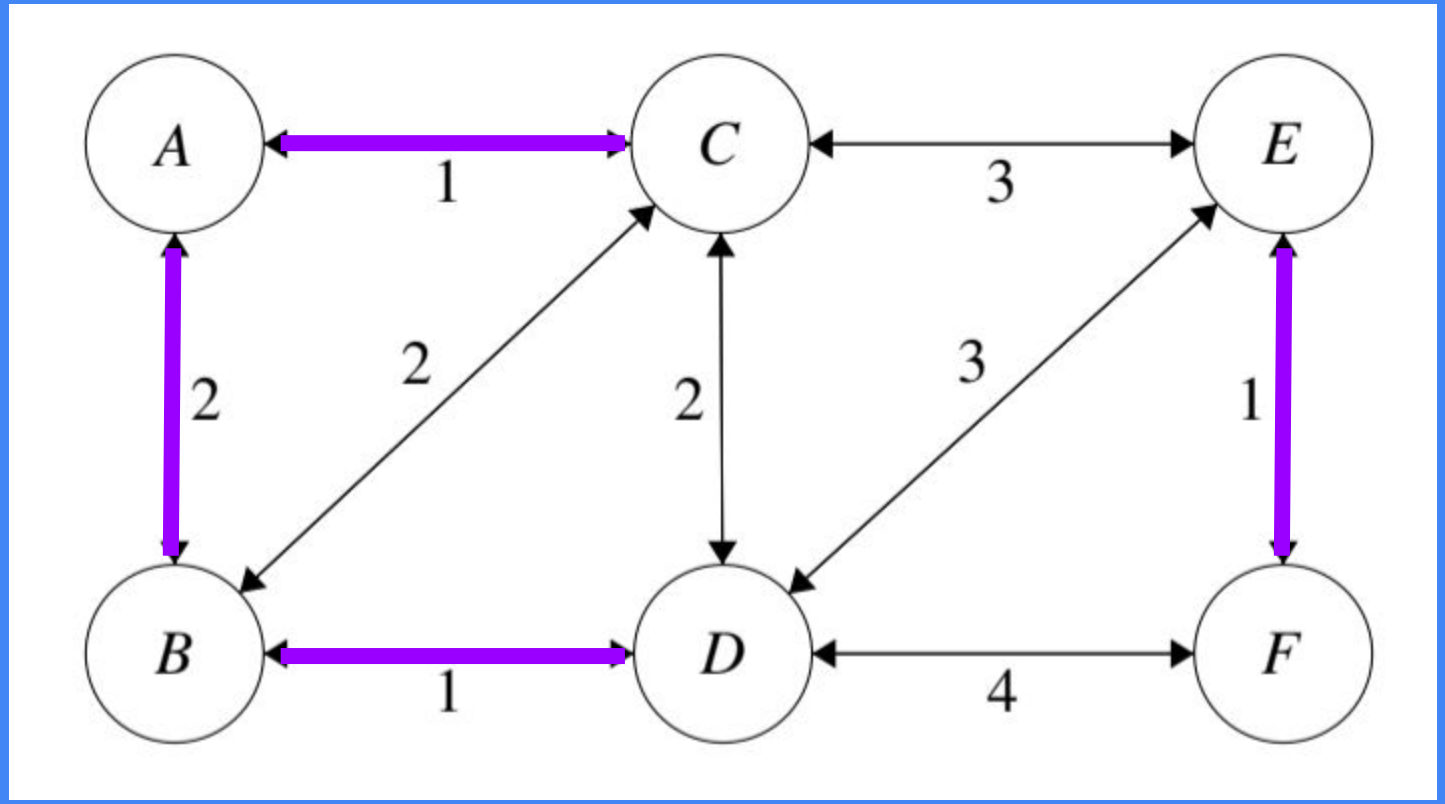
- Kruskal’s algorithm says, we can be even lazier.
- As long as the edge doesn’t create a cycle, just add the smallest edge.

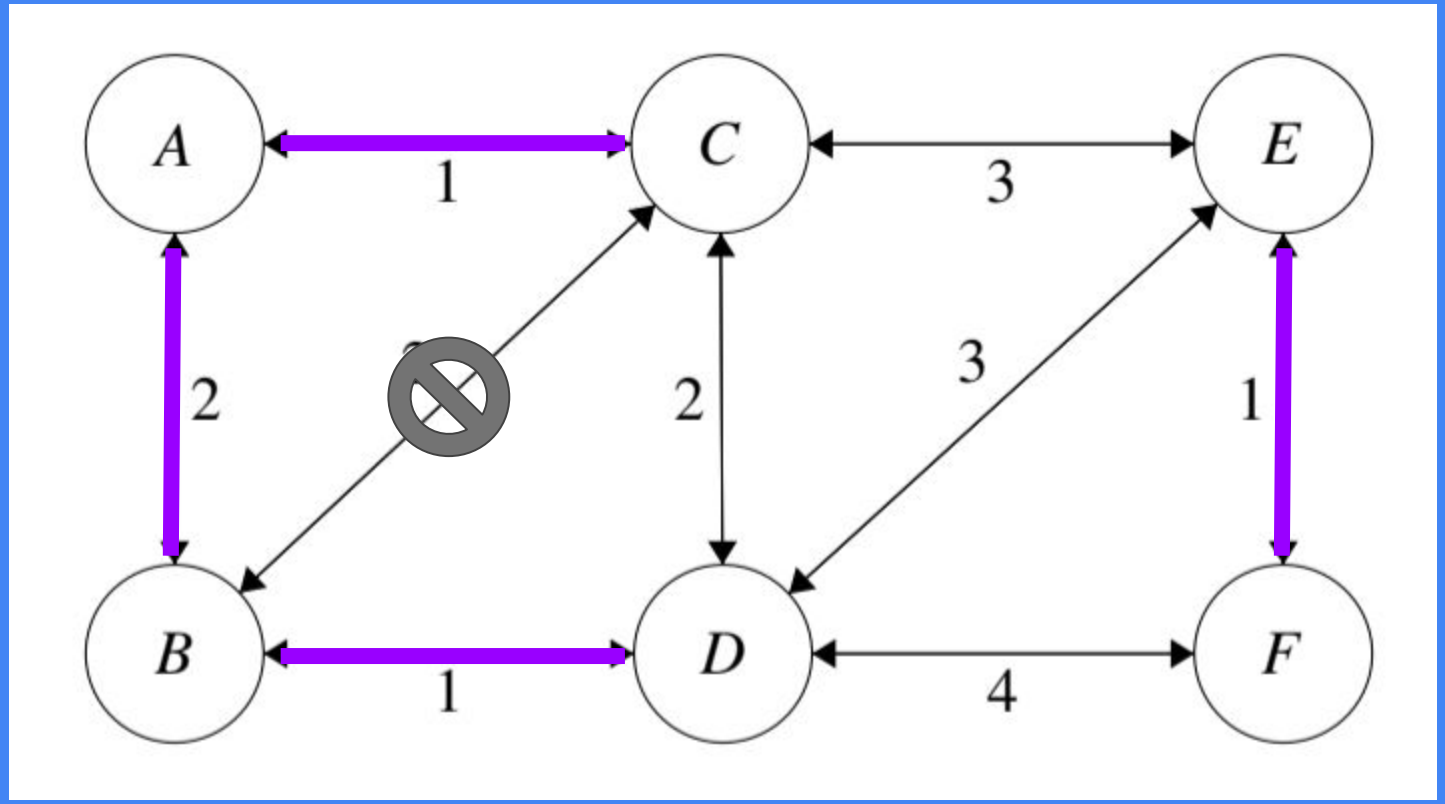

A
C
E
B
D
F
1
3
2
2
2
3
1
1
4

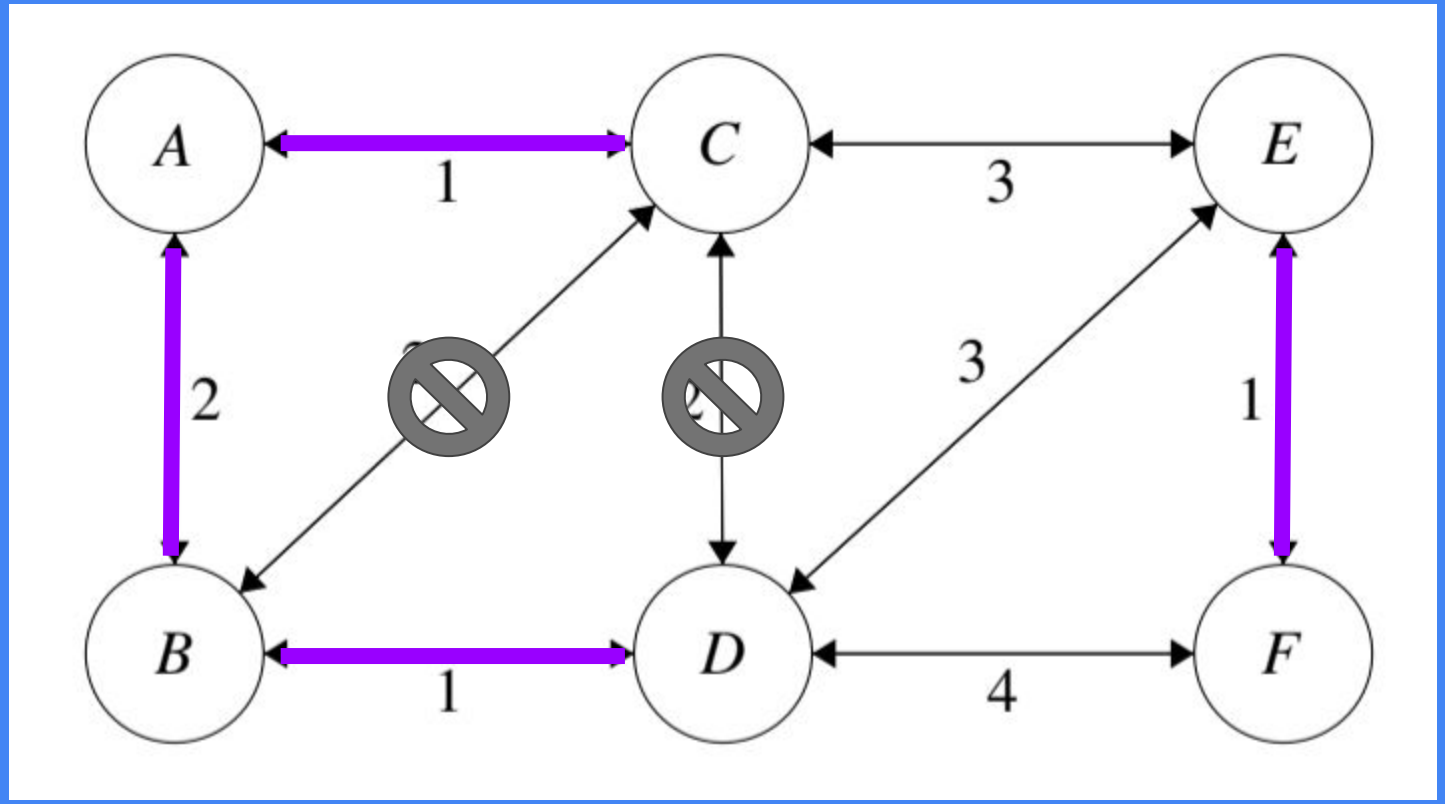

A
C
E
B
D
F
1
3
2
2
2
3
1
1
4

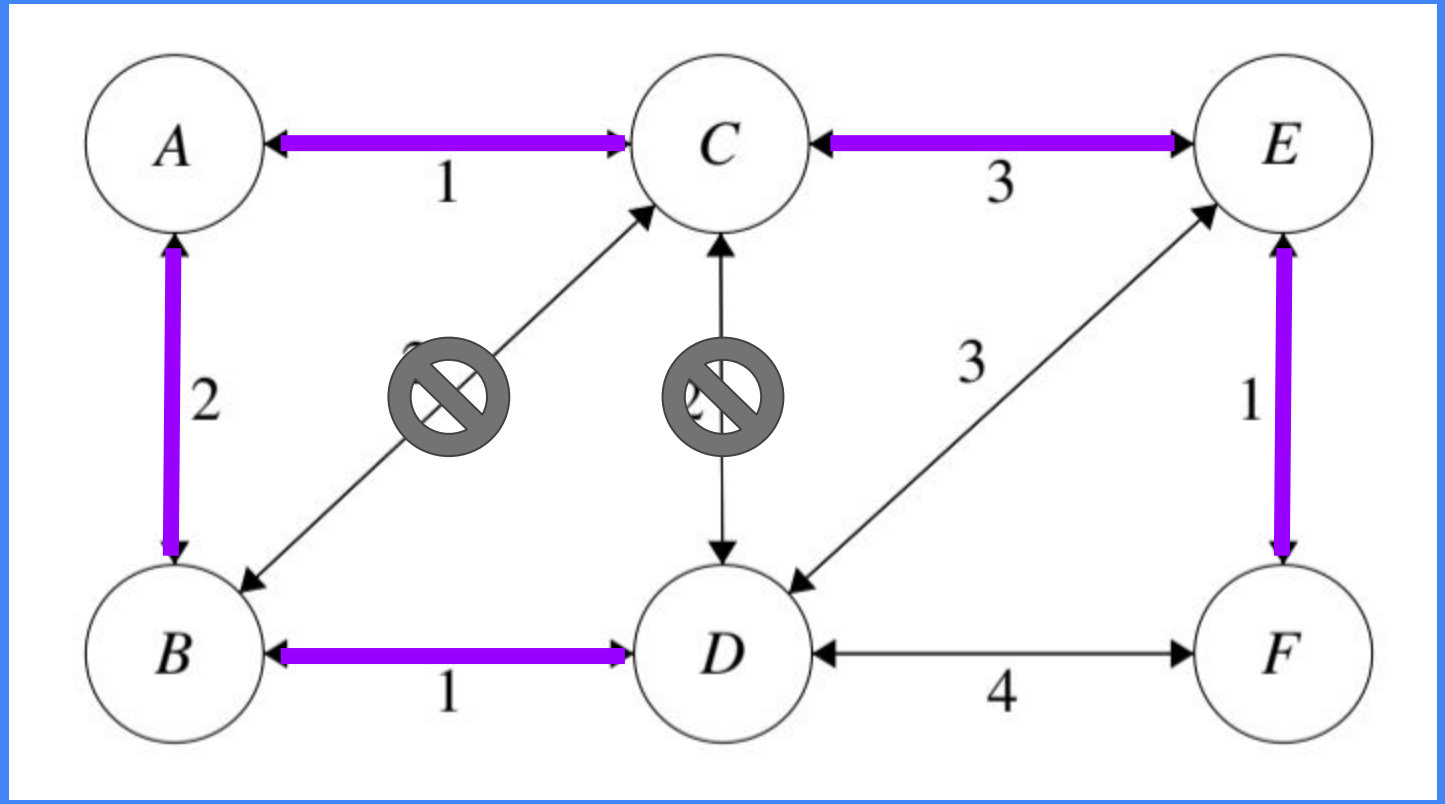

A
C
E
B
D
F
1
3
2
2
2
3
1
1
4

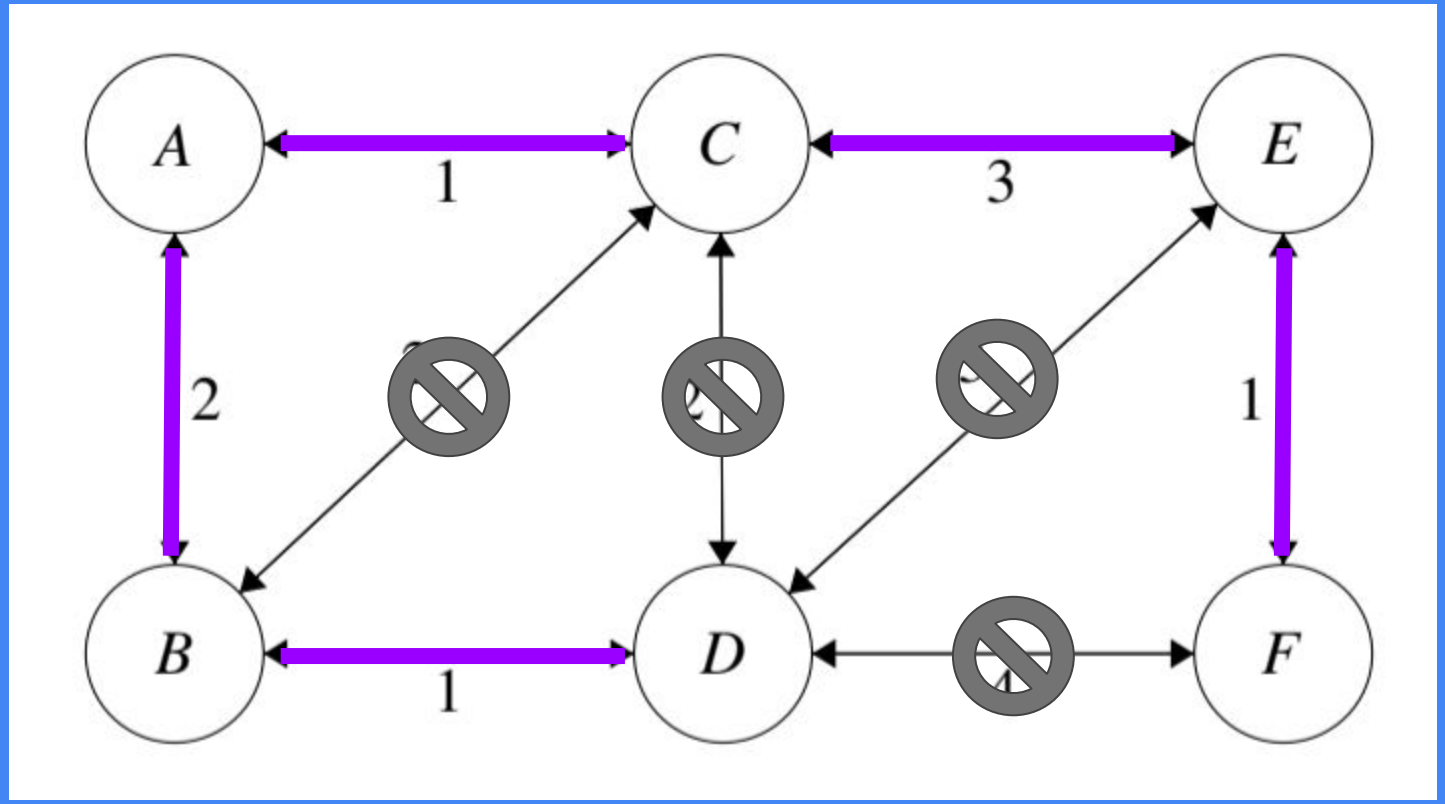

A
C
E
B
D
F
1
3
2
2
2
3
1
1
4

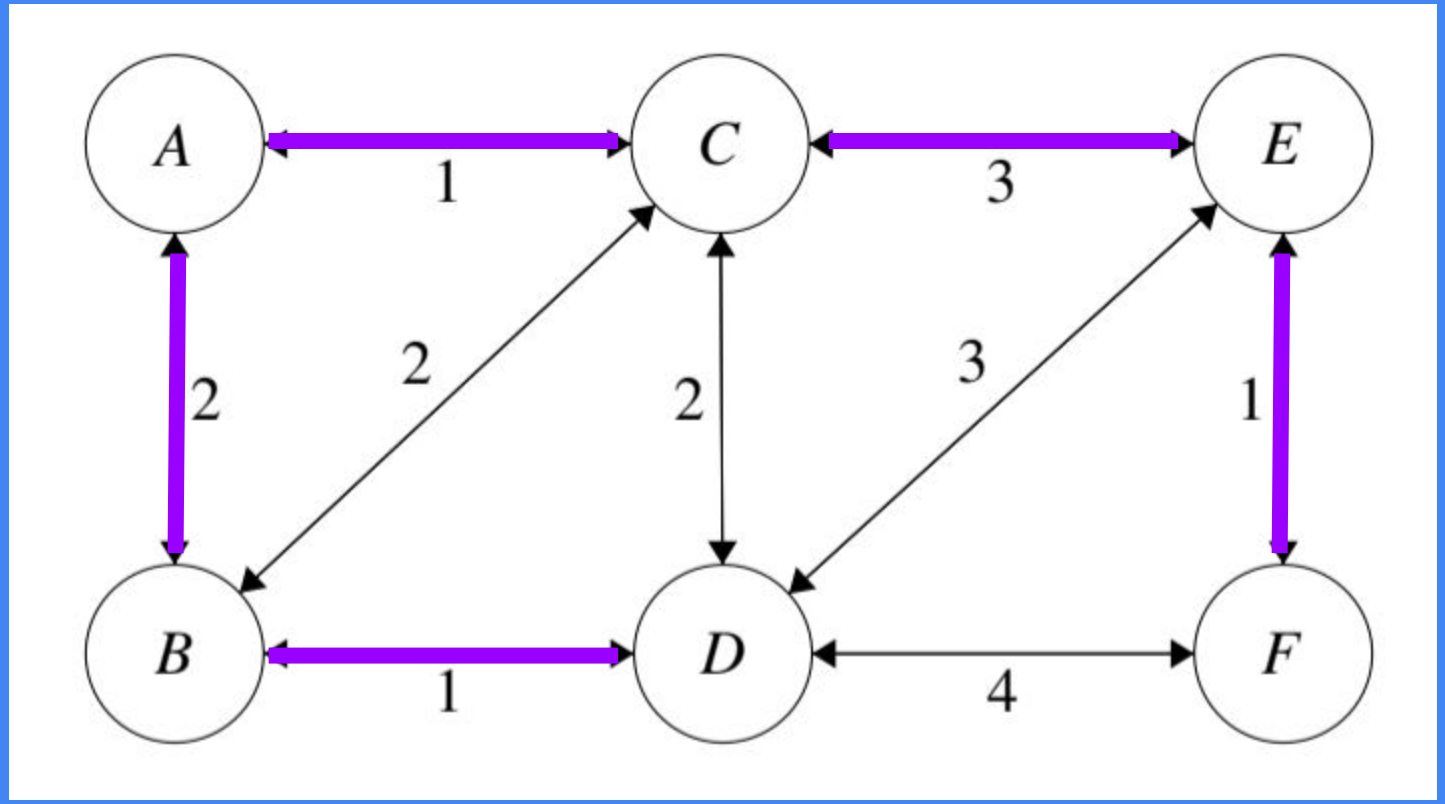

A
C
E
B
D
F
1
3
2
2
2
3
1
1
4


A
C
E
B
D
F
1
3
2
2
3
1
1
4


A
C
E
B
D
F
1
3
2
3
1
1
4


A
C
E
B
D
F
1
3
2
3
1
1
4


A
C
E
B
D
F
1
3
2
1
1


A
C
E
B
D
F
1
3
2
2
2
3
1
1
4

Kruskal’s Algorithm

- How do we model this in code?
  - Usually we use a **disjoint set**.

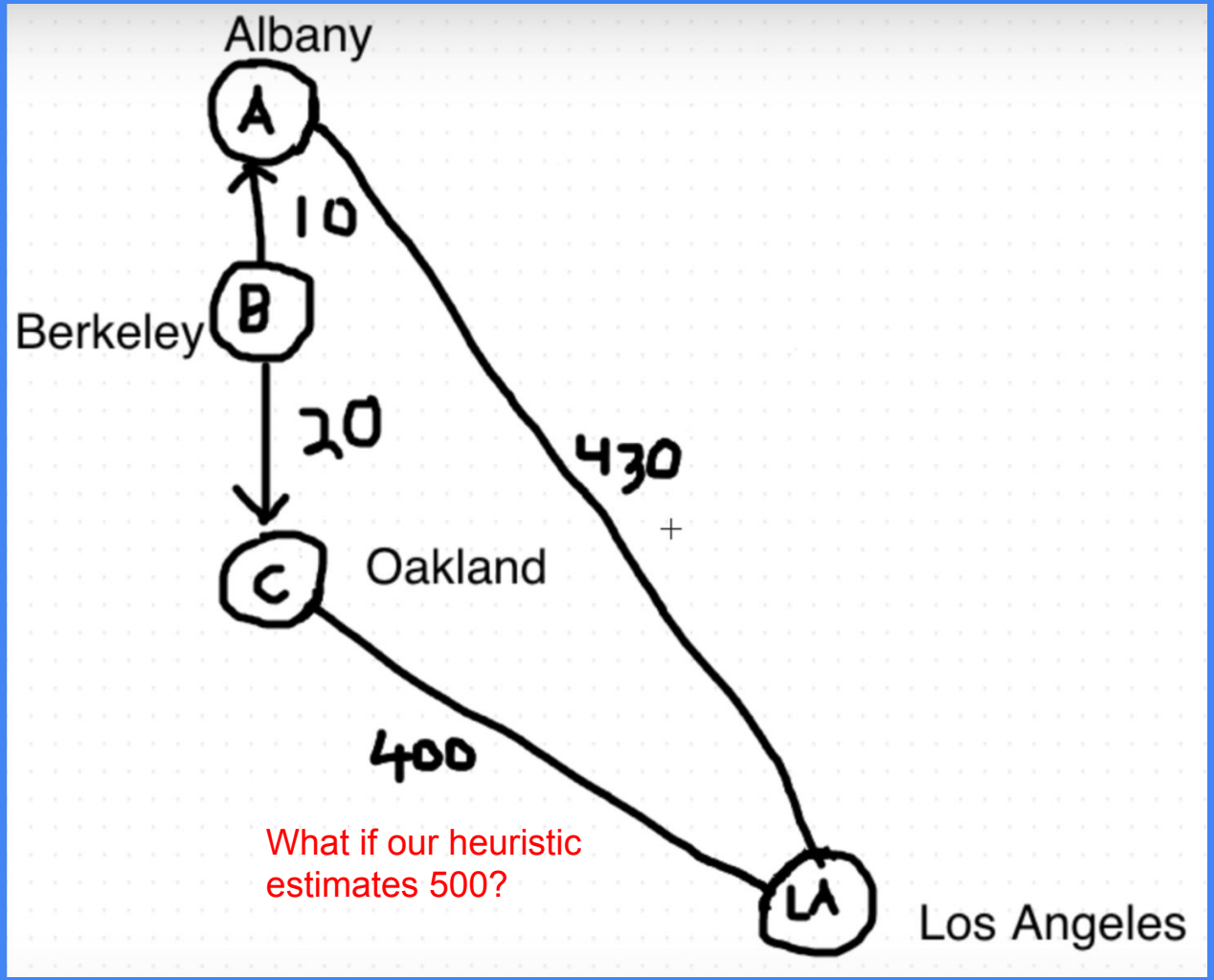
Q1 (Warmup)

A* Admissibility

- Problem: How can we **still** maintain Dijkstra's invariant that for each node we pop off the priority queue, **we have guaranteed the shortest path from the start node to that node?**
  - After all, we're adding this heuristic value to our priority queue comparator!

A* Admissibility

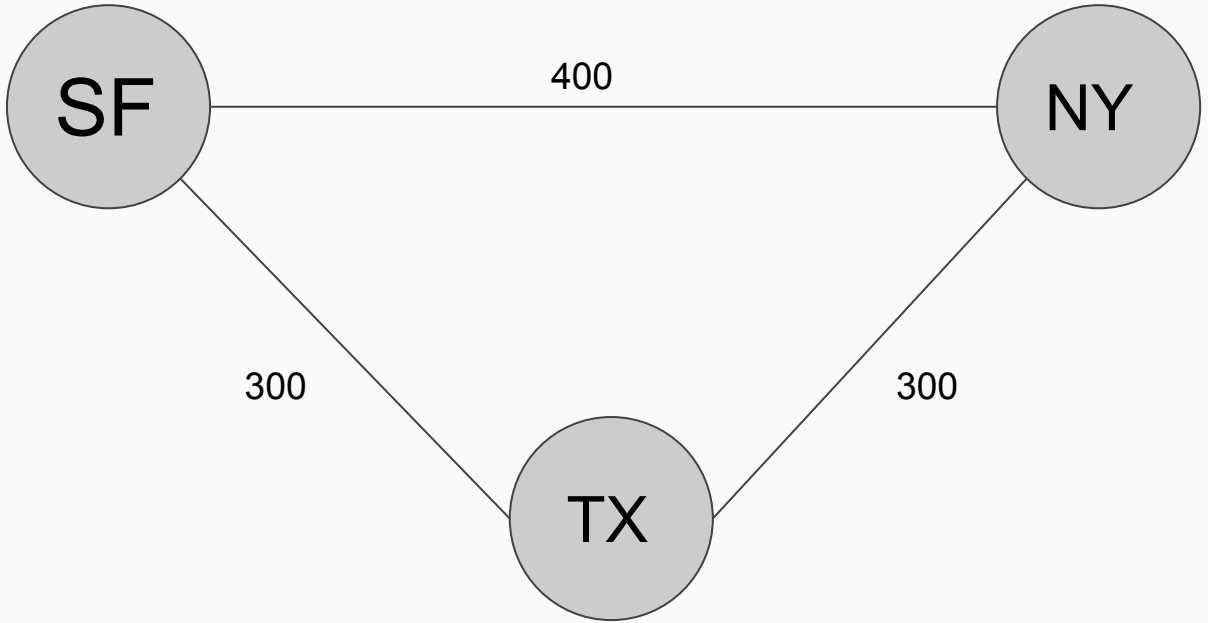
- Solution: Our heuristic must never **overestimate** our distance.
  - This is called an **admissible heuristic**.
  - If our heuristic is always **0**, then we just have regular Dijkstra's!
- What happens if our heuristic overestimates?


Albany
A
10
Berkeley
B
20
430
C
Oakland
400
What if our heuristic estimates 500?
LA
Los Angeles

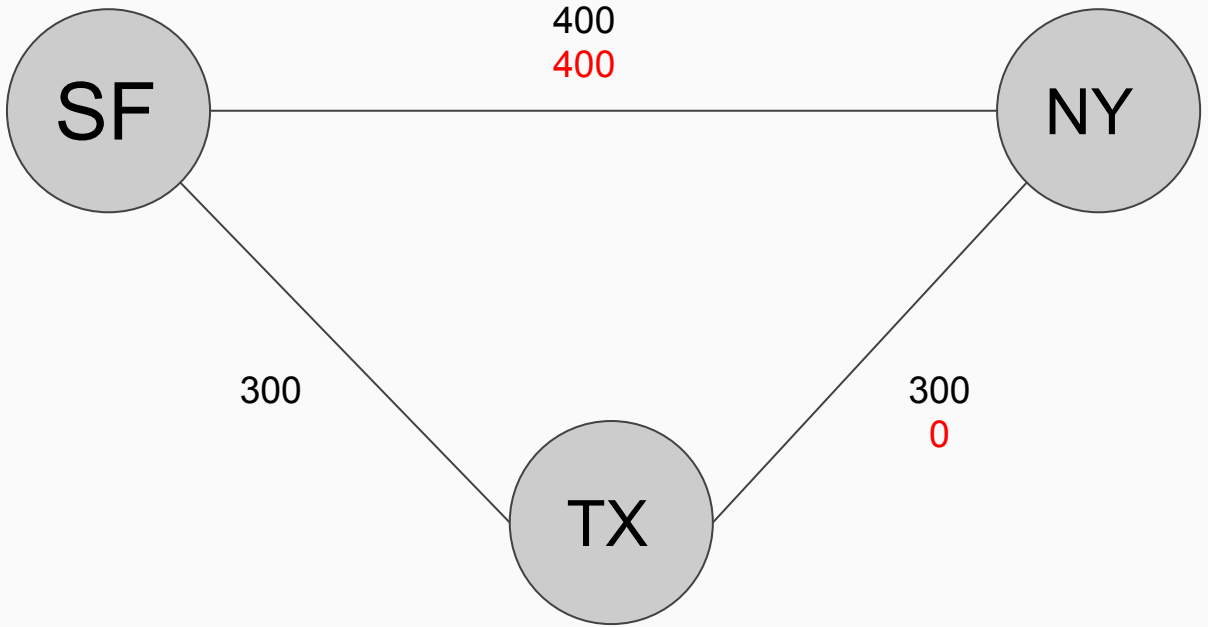
A* Admissibility

- In this case, our A* algorithm will go to Albany and then to LA.
- Thus, we don’t find our shortest path because our heuristic **overestimated** the distance from Oakland to LA.

A* Consistency



A* Consistency



Q2

Q3