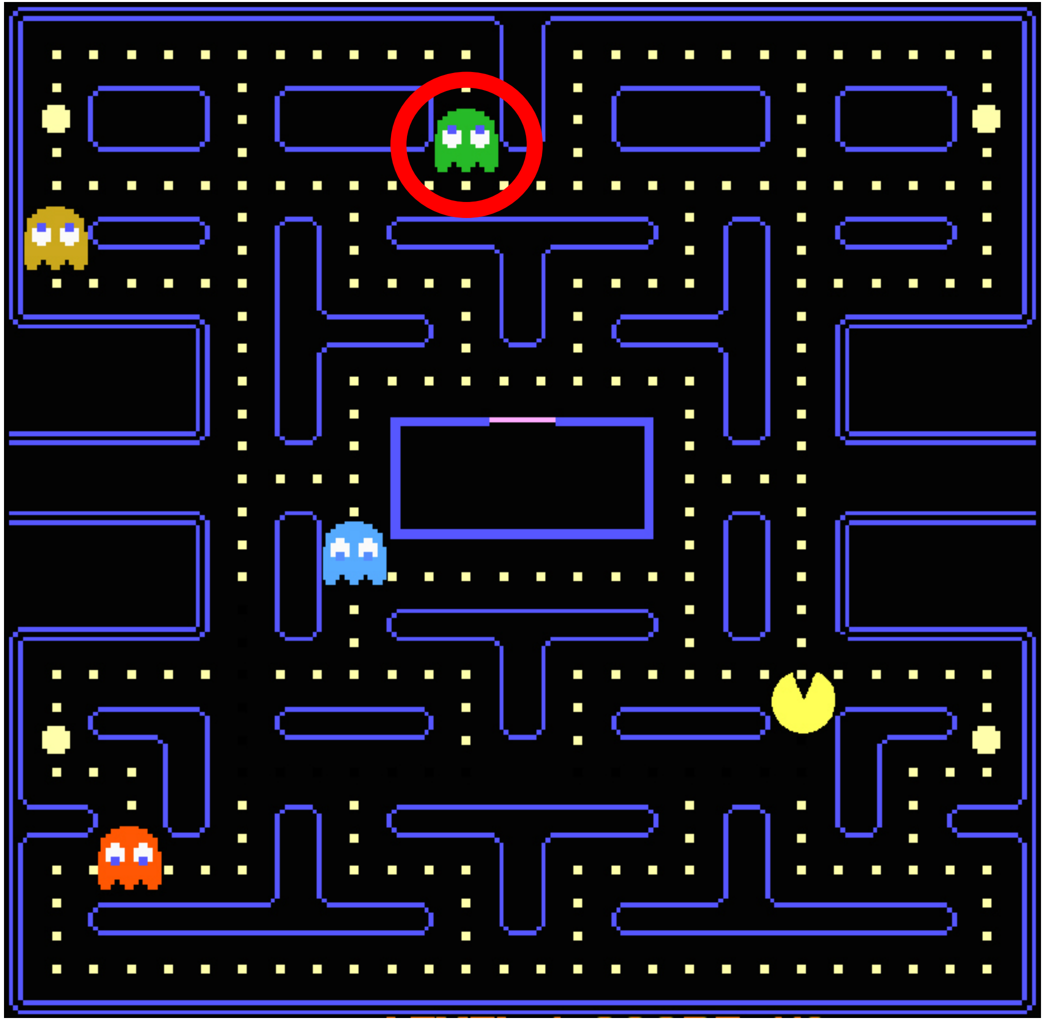
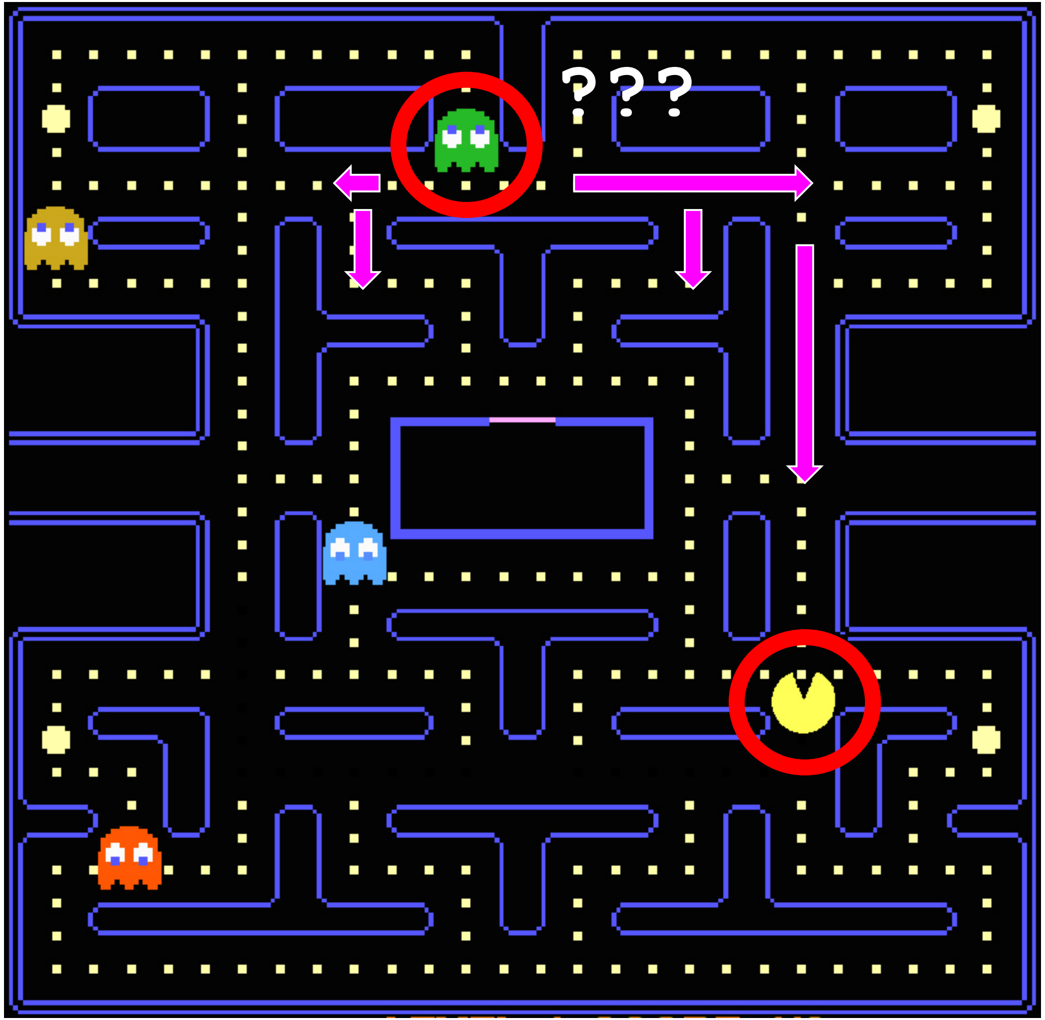
Graph Theory

A Motivating Example

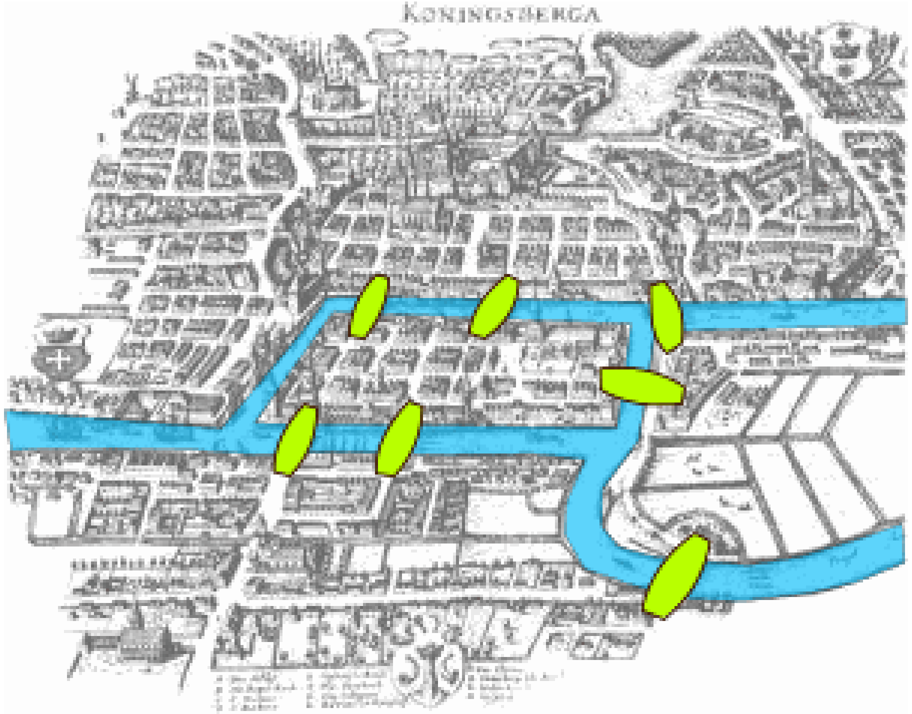


So you're Funky and your life motivation is to get to Pac-Man



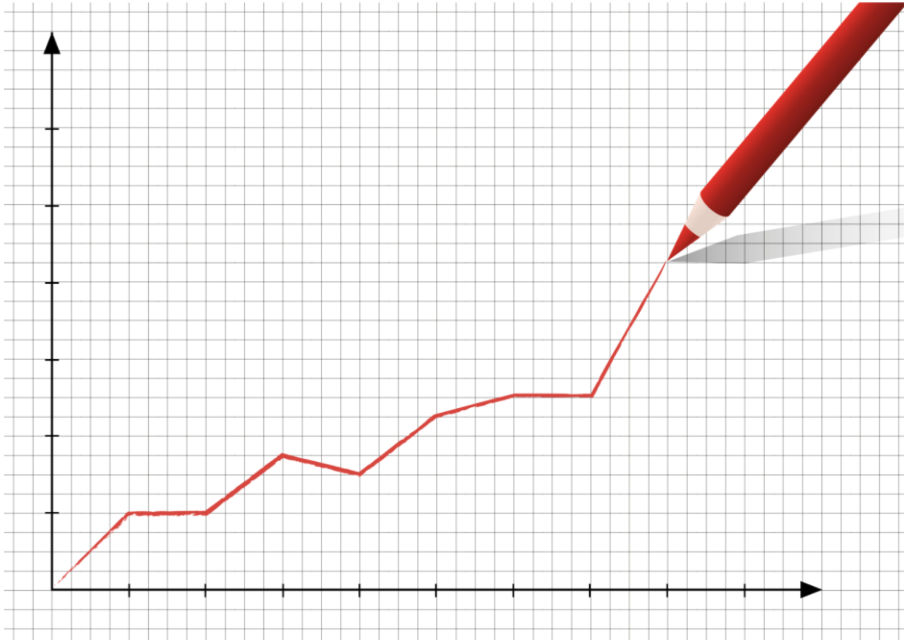
So you're Funky and your life motivation is to get to Pac-Man

But time is limited and you want to get there as soon as possible

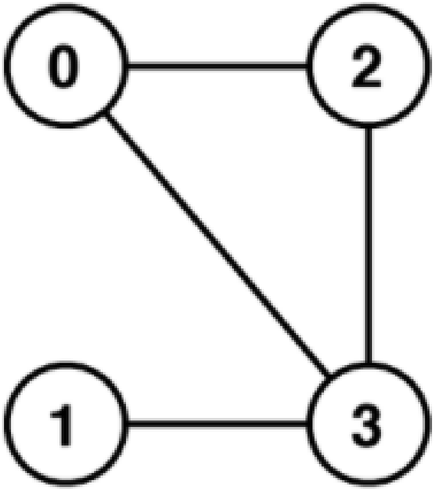


The Seven Bridges of Königsberg

So what exactly is a graph?



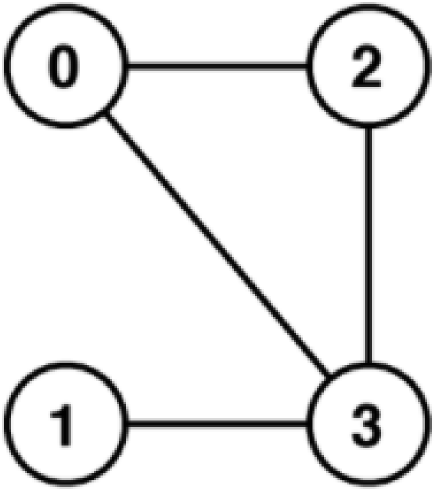
Not this.



A graph is an ordered pair of sets:

$$G = (V, E)$$

where V is the set of vertices, and E is the set of edges connecting those vertices.



A graph is an ordered pair of sets:

$$G = (V, E)$$

where V is the set of vertices, and E is the set of edges connecting those vertices.

Yes, it is a little weird.

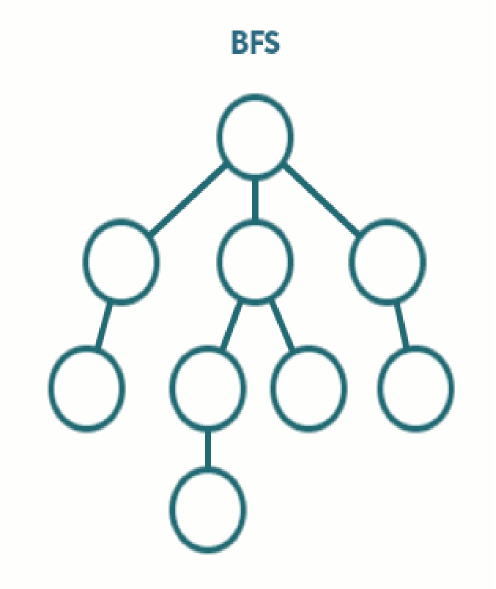
Why are we talking about graphs?

- They can represent literally anything

- Any collection of data that has relationships can be represented by a graph

- Networks, communications, data organization, social media, travel, biology, computer chip design, linguistics, chemistry, physics, sales, business, statistical analysis, psychology, sociology, mathematics, navigation, and more. Basically, actually, everything.

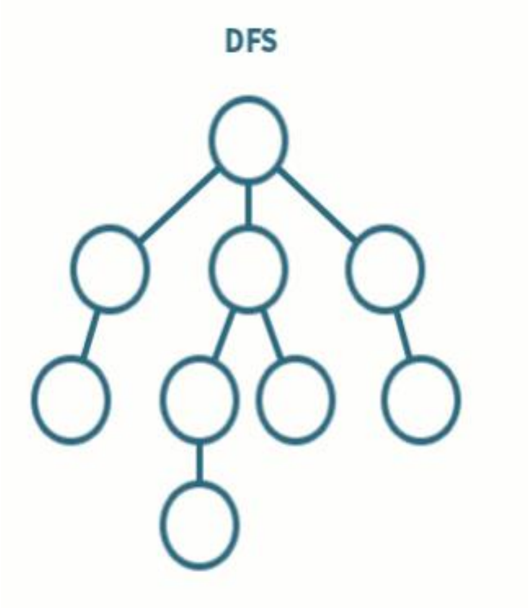
Breadth First Search



Breadth First Search

- GameAI - augment with A*
- Pathfinding
  - Directions
  - Road networks
- Other algorithms that build off BFS
  - Dijkstra
  - A*
  - Bellman-Ford
  - Floyd-Warshall

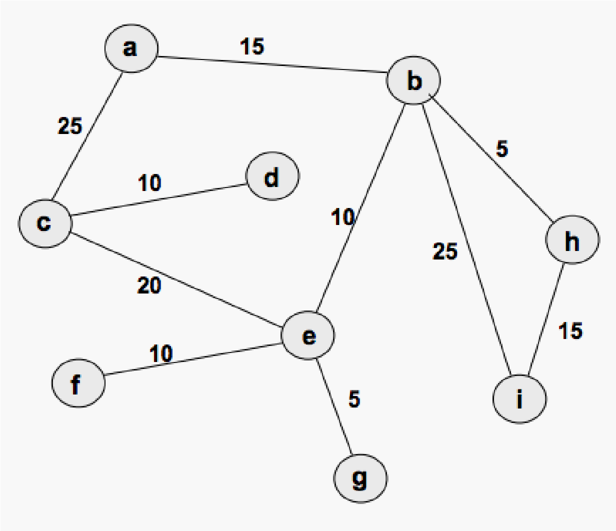
Depth First Search



Depth First Search

- Maze solving/generation
- Finding connected components
- File structure searching
- Searching for anything that can be represented as a graph
  - Searching for a friend on the social media graph
  - Searching for a location on the location maps
- Navigation systems

Weighted Graphs

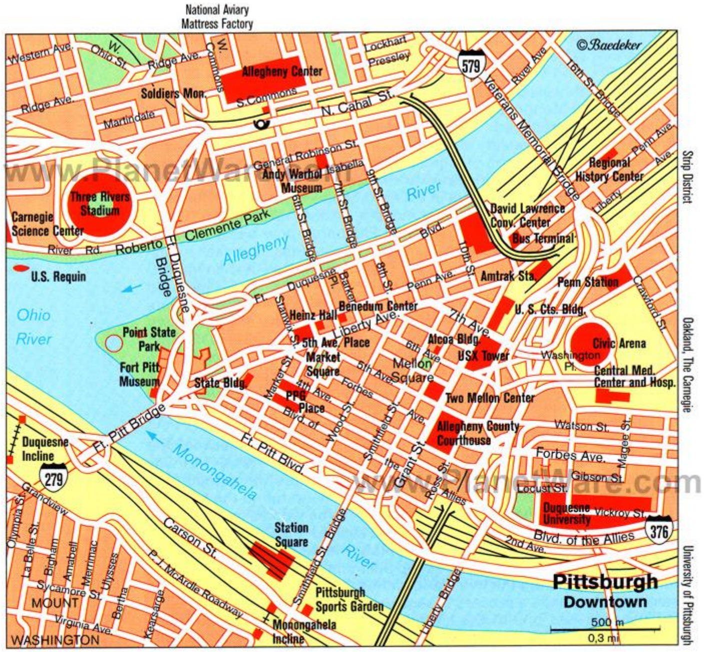


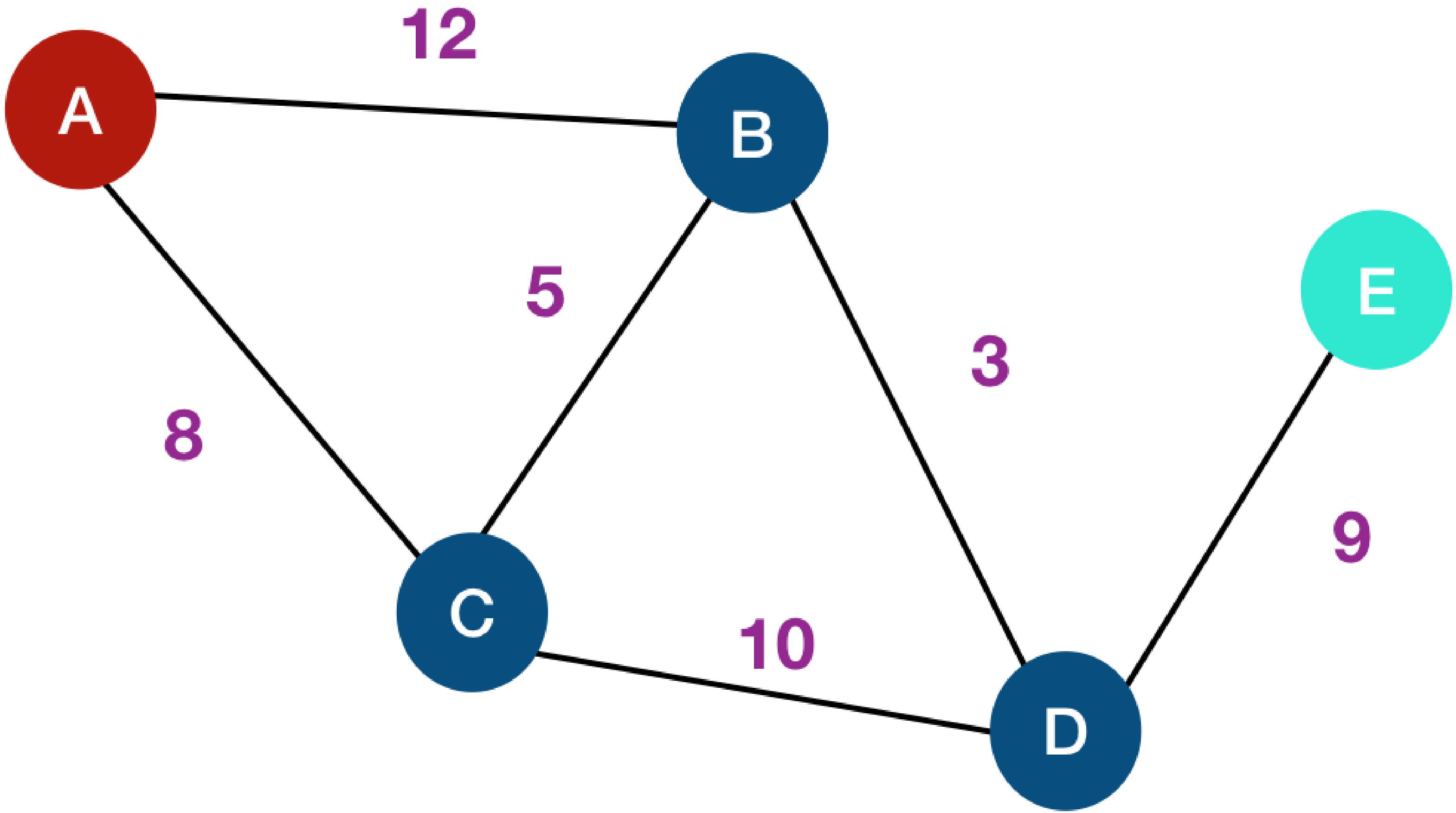
- Like normal graphs… except with edge weights
- Weights on the edges can represent anything
- For example:
  - Nodes are buildings
  - Edge weights are times it takes to get there
- Another example:
  - Nodes are people
  - Edge weights are how much money one person owes another
- Another example:
  - Nodes are airports
  - Edge weights are cost of flights from one airport to another

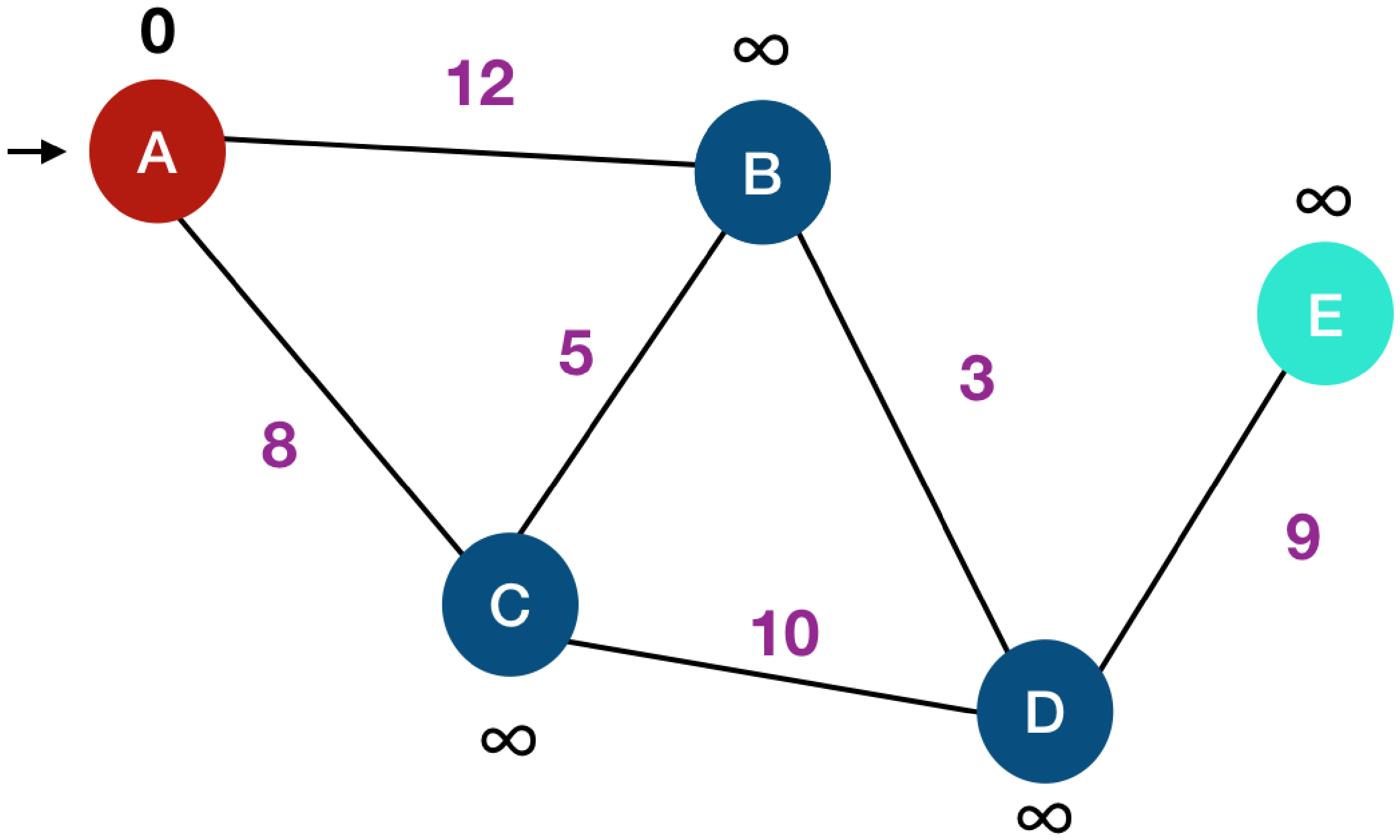
Dijkstra’s Algorithm

Path Finding Algorithms

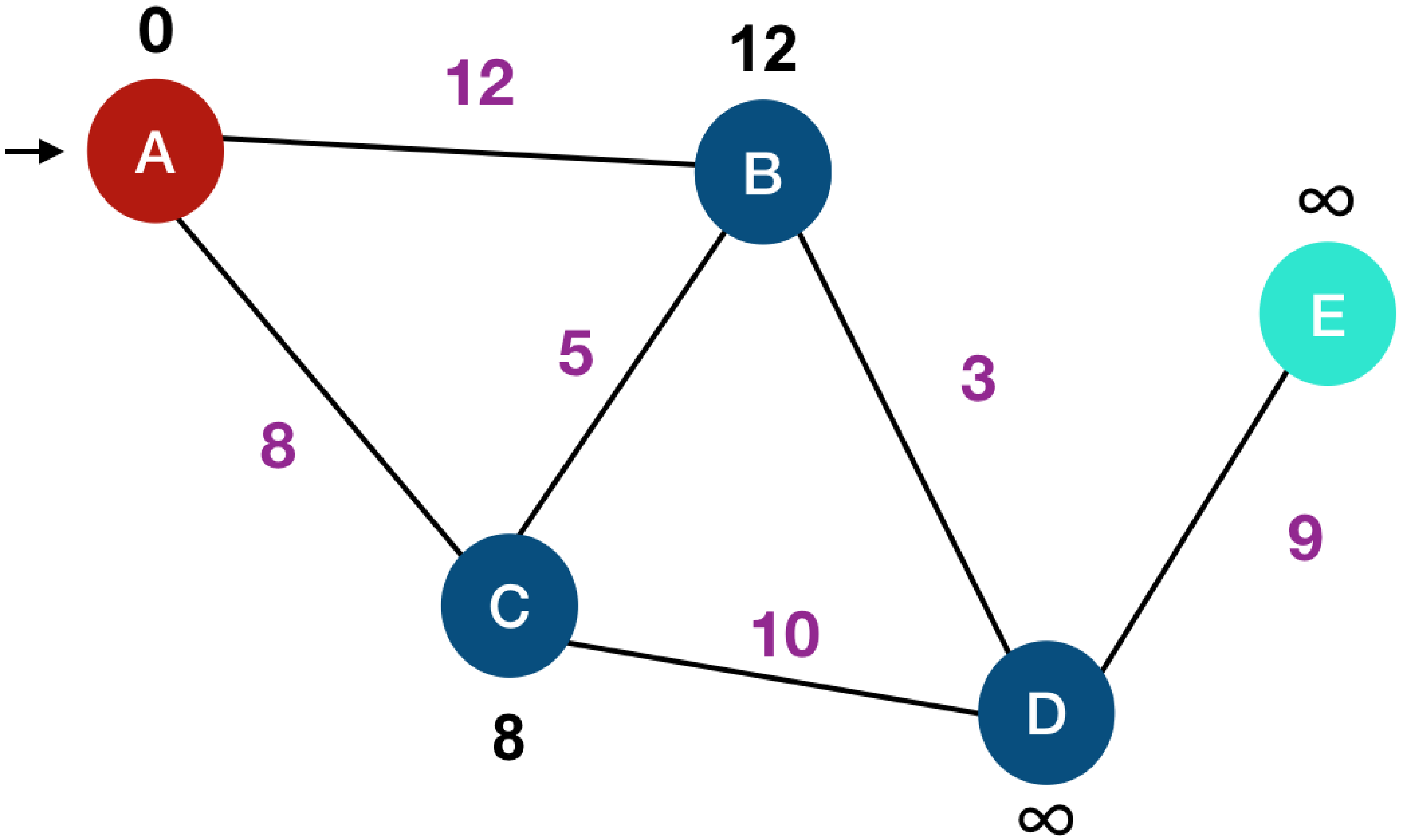
- Find the shortest path in a graph!
- Optimize distance, time, expenses

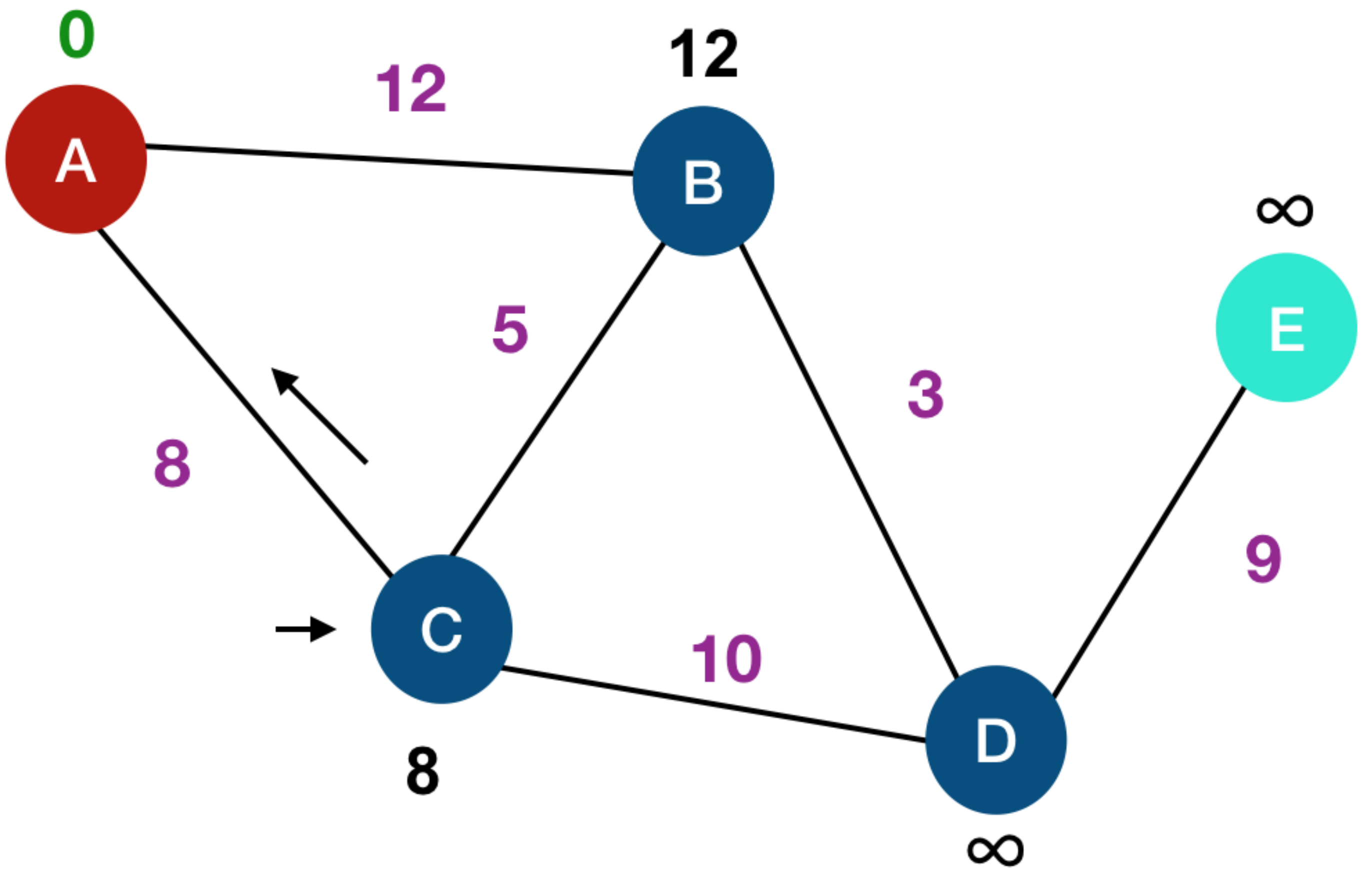


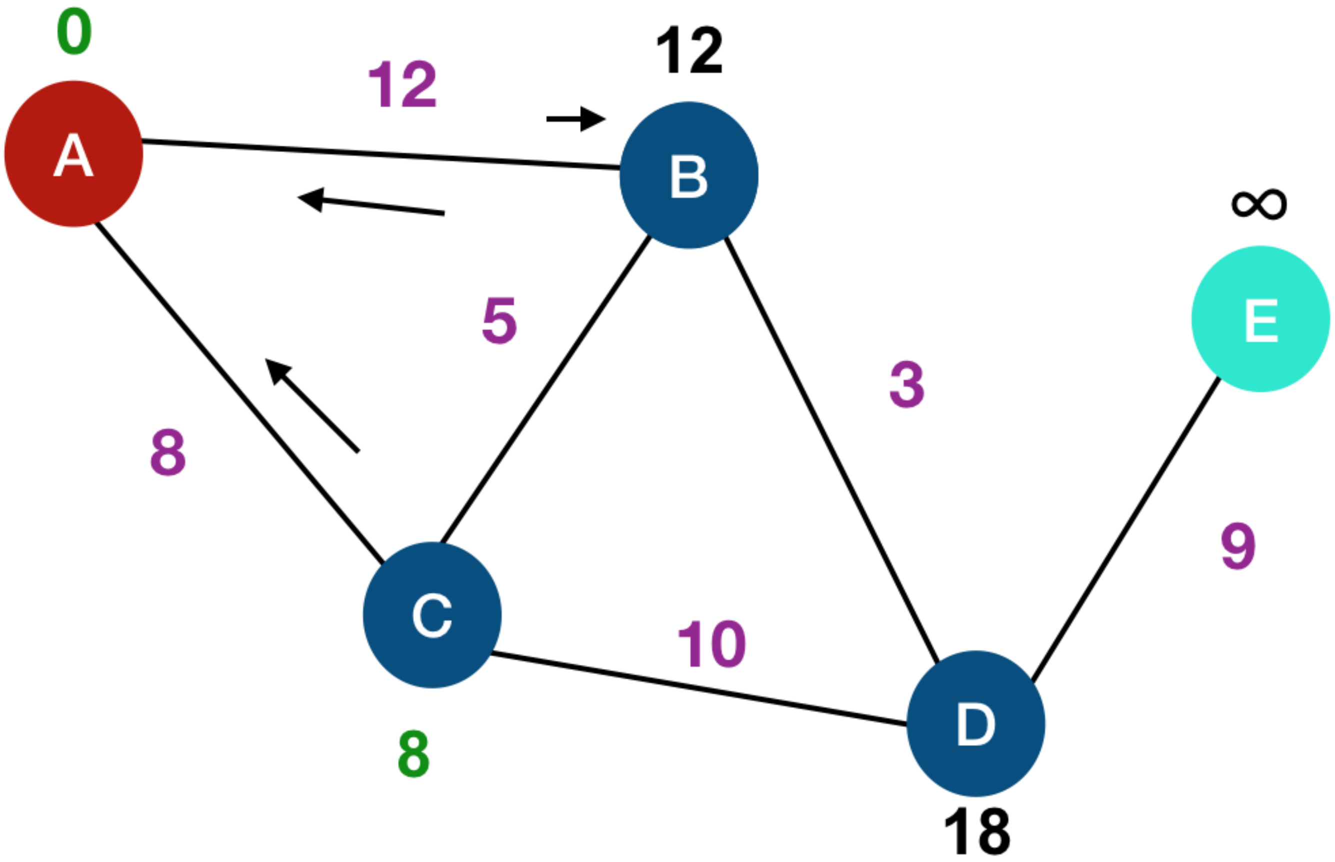

A
B
C
D
E
12
5
3
8
9
10

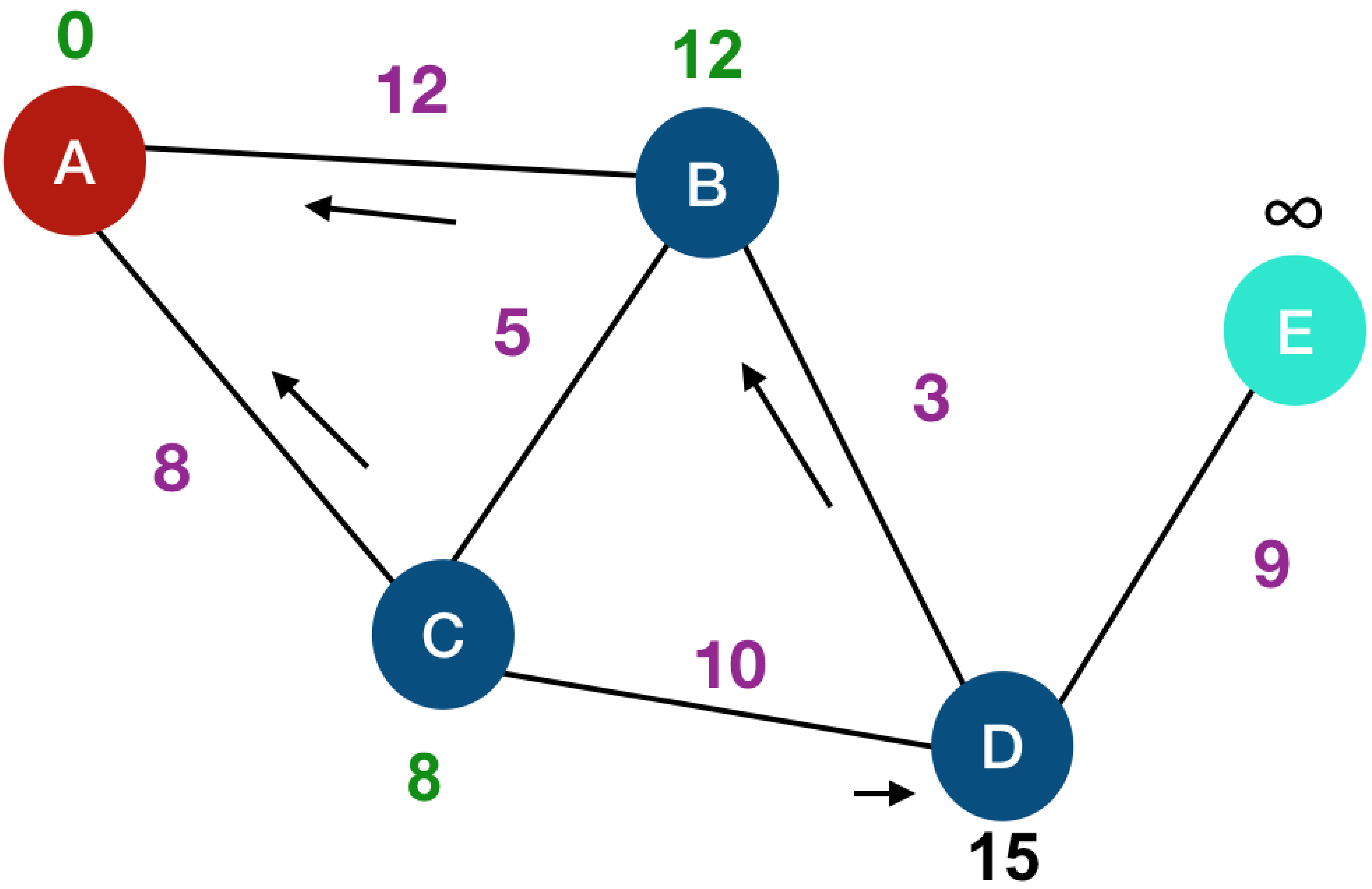


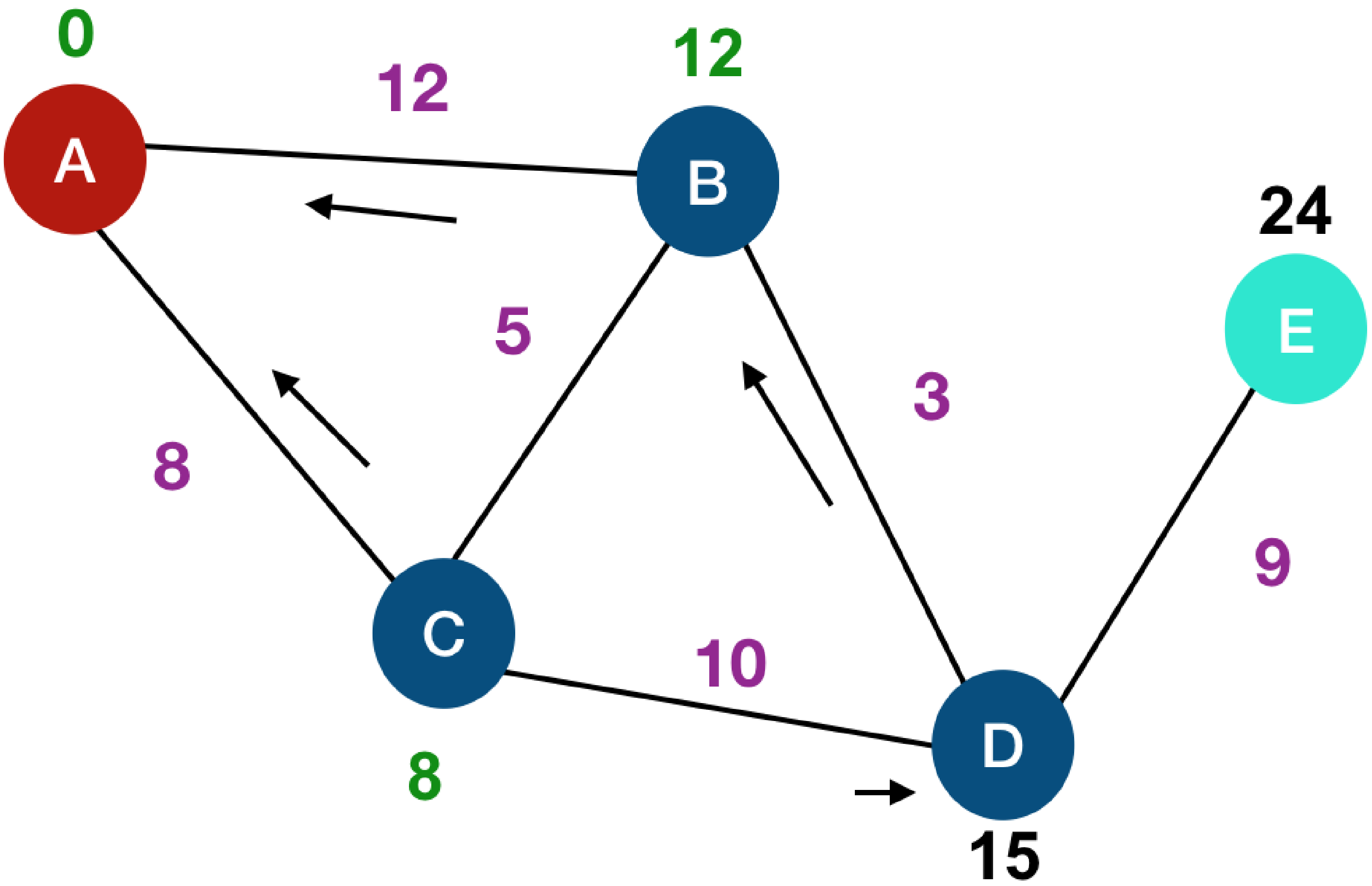
0
∞
∞
∞
∞
A
B
C
D
E
12
8
5
3
10
9

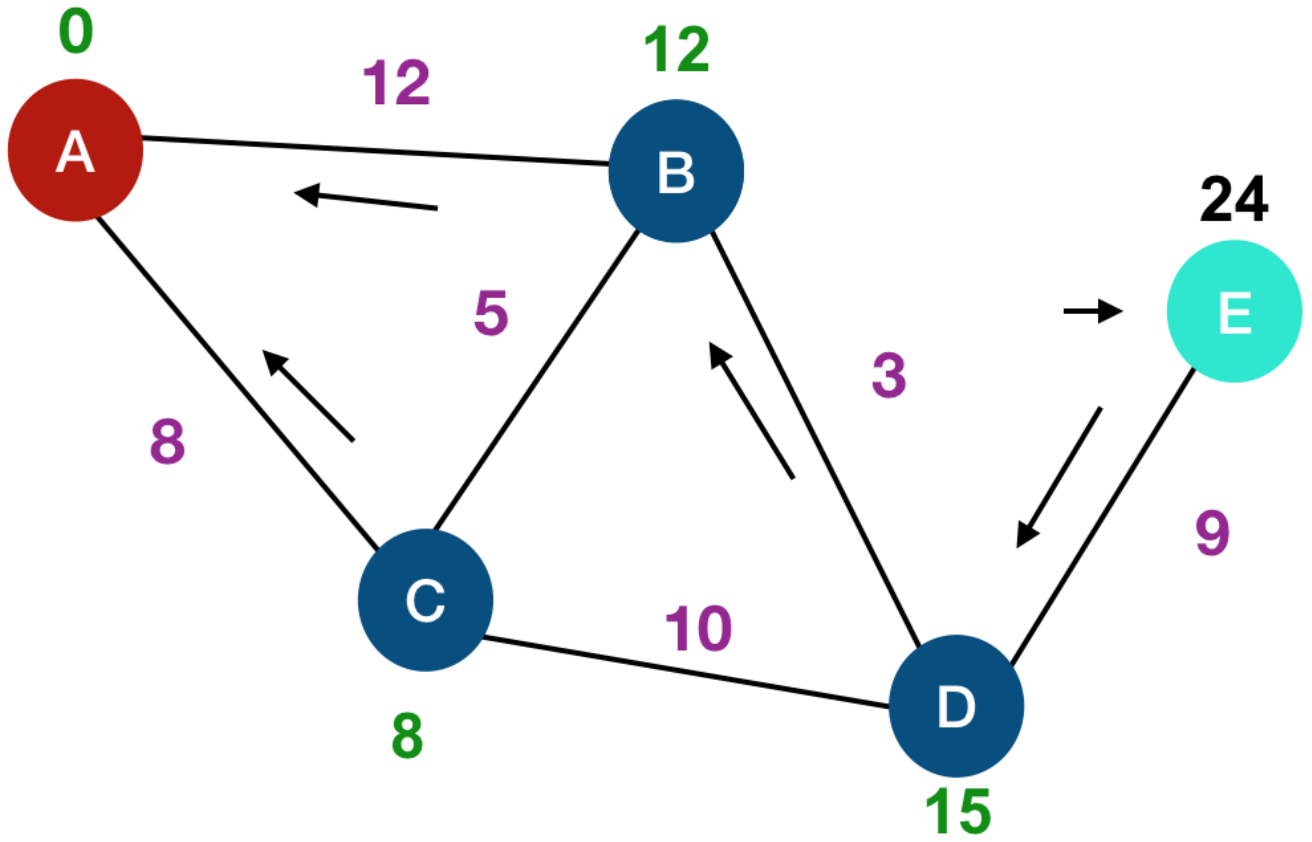

0
12
12
A
B
∞
E
5
3
8
9
C
10
D
8
∞


0
12
12
A
B
∞
E
5
3
8
9
C
10
D
8
∞


0
12
12
A
B
∞
E
5
3
8
9
C
10
D
8
18


0
A
12
12
B
∞
E
5
3
8
9
C
10
D
8
15


0
12
12
A
B
24
E
5
3
8
9
C
10
D
8
15

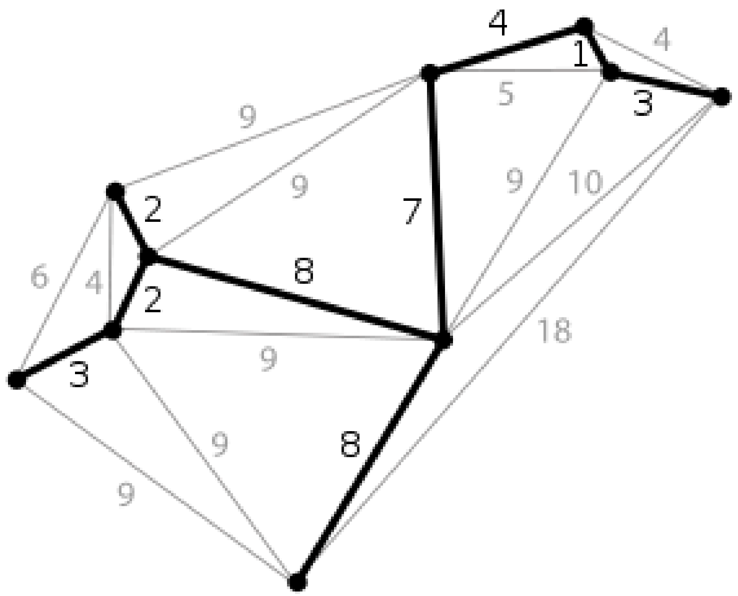


0
12
12
A
B
24
E
5
3
8
9
C
10
D
8
15

```python
def dijkstra(vertices, edges):
    visited = {}
    path = []

    while len(vertices) > 0:
        minNode = None

        for vertex in vertices:

            if vertex in visited:
                if minNode == None:
                    minNode = vertex

                elif visited[vertex] < visited[minNode]:
                    minNode = vertex

        if minNode == None:
            break

        del vertices[minNode]
        currWeight = visited[minNode]

        for edge in edges[minNode]:
            weight = currWeight + distance(minNode, edge)
            if edge not in visited or weight < visited[edge]:
                visited[edge] = weight
                path[edge] = minNode

    return visited,  path
```

Minimum Spanning Trees

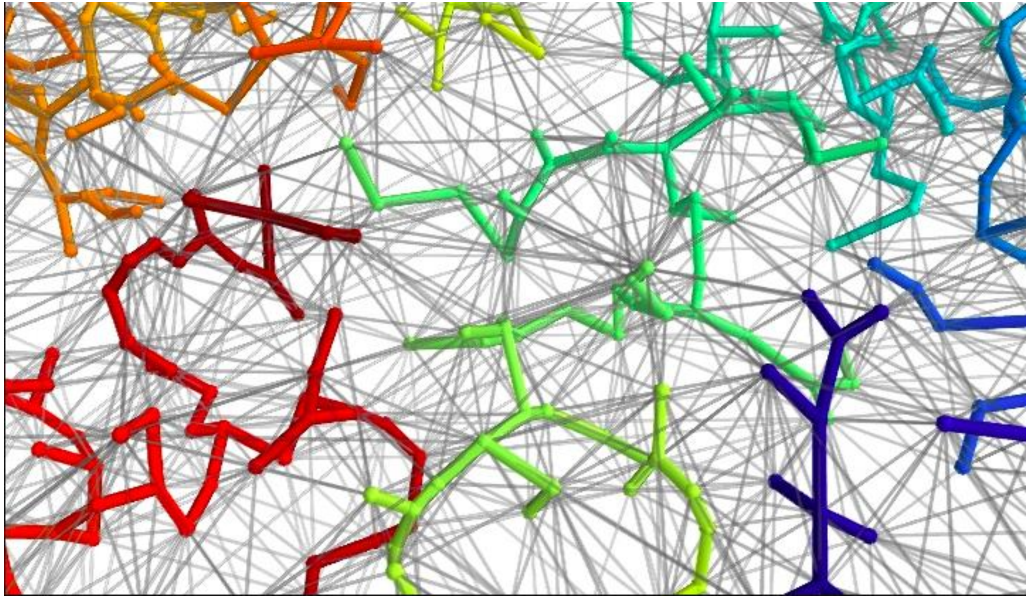
What it is



- Some of the edges on the graph
- A spanning tree:
  - Some subset of edges
  - Connects all the nodes somehow
- A minimum spanning tree?
  - A spanning tree
  - That connects all the nodes
  - But using the minimum amounts of weights for the edges

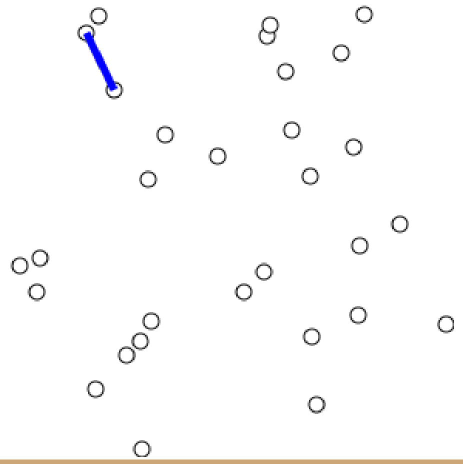
Some Algorithms For Finding MSTs

- Boruvka's Algorithm
- Prim's Algorithm
- Kruskal's Algorithm
- Tarjan's Algorithm



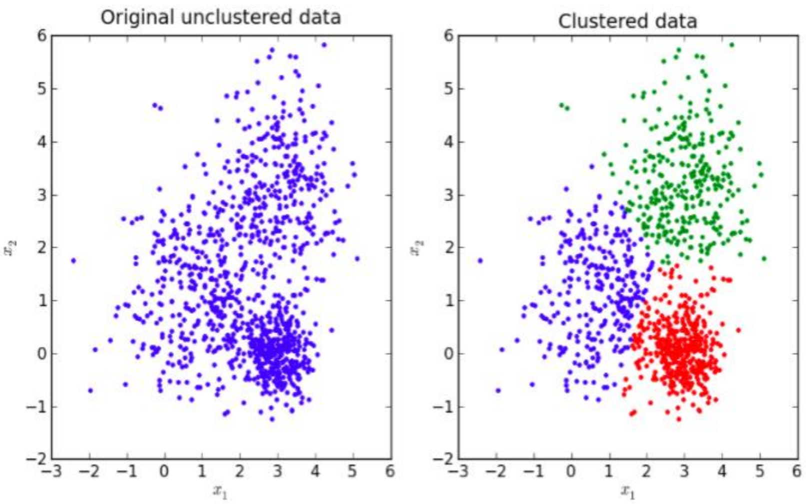
Prim's Algorithm

- Actually similar to Dijkstra's
- Start with one node as your MST
- Find the smallest edge weight coming out of it and add that to MST
- Continue (finding smallest edge weight coming out of MST)
  - Until you have all nodes



Common Uses

- Networks (any kind of network)
  - Computer, telecommunications, transportation, water supply, electrical grids, etc.
- Machine Learning
- Image processing
  - Image segmentation
- Circuit design
- Lots of markets stuff (stocks, etc)
- Computer Vision
- Maze Generation



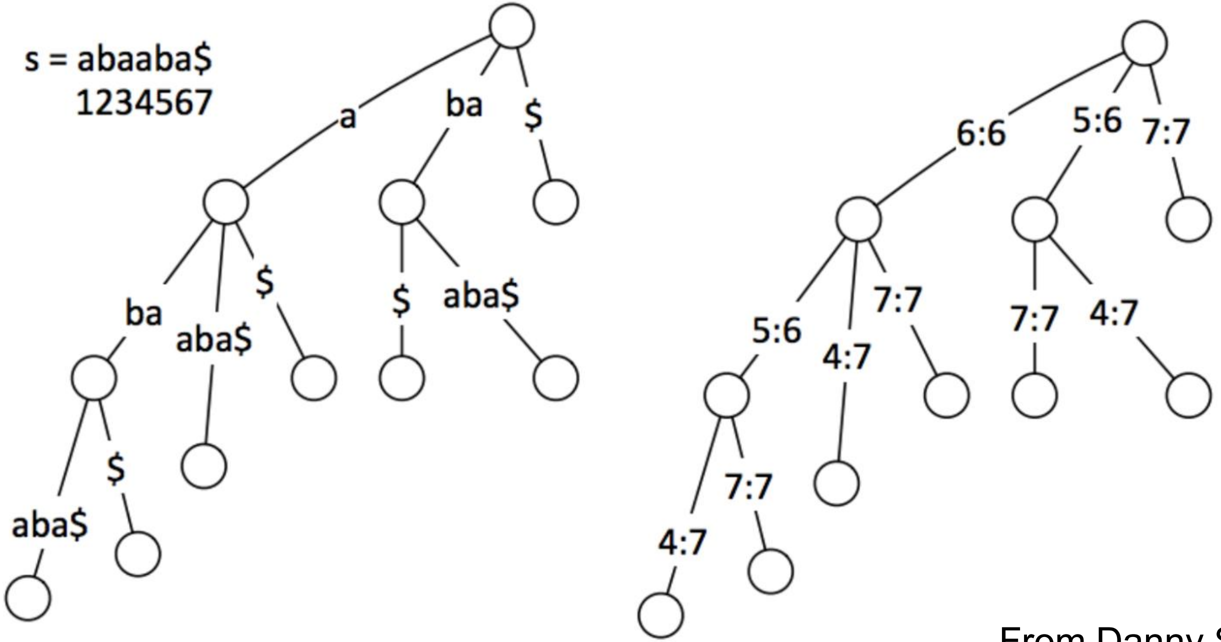
Implementation

- Wikipedia is your friend!
    - Cite code that isn't yours though, as always
- Lists! Sets!
- Other fancy data structures like: priority queues, linked lists
- Scipy
- http://www.geeksforgeeks.org/greedy-algorithms-set-5-prims-minimum-spanning-tree-mst-2/ is an awesome website
- Object oriented Programming

Suffix Trees

What it is

- A tree...of suffixes



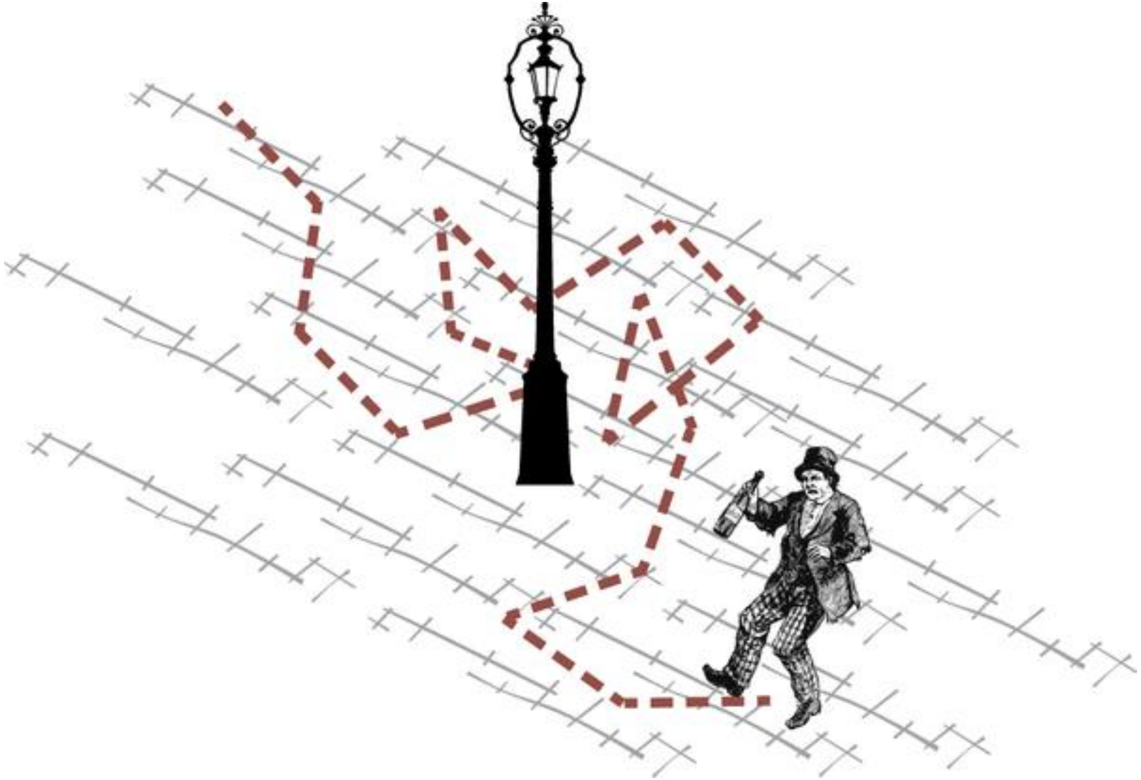
From Danny Sleator’s 451 notes

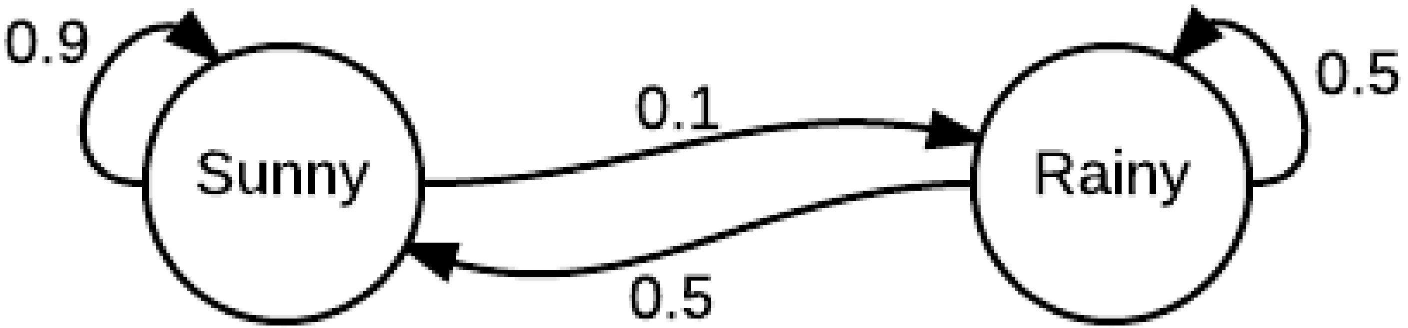
Common Uses

- Find all occurrences of P in the text T
  - O(p+k) where k is the
  - O(t) time to preprocess and store
  - Given a finite alphabet
- Check whether P is a suffix of T
- Count the number of occurrences of P in T
- Find the alphabetically first suffix
- Search for matching strings in a database
- Look for repeating sequence of nucleotides in DNA!

Adapted from Danny Sleator's 451 notes

Markov Chains




0.9
Sunny
0.1
0.5
Rainy
0.5

Common Uses

- Google PageRank!
  - The problem here is: Given n interlinked web pages, rank them in order of “importance”
- Predicting brand loyalty
- Word Prediction - @Autocorrect

https://tinyurl.com/graphTheoryAttendance