An Introduction to Graph Algorithms

Cathen Fontanilla and Zoe Guo

May 2026

Road Map

Background:

1. Introduction & the Graph Algorithm Problem
2. Algorithm Example: Breadth First Search
3. Algorithm Example: Dijkstra's Algorithm
4. Conclusion & Further Directions

Section 1: Introduction & the Graph Algorithm Problem

Graphs

Many problems in modern society can be represented through connections. For example, how do we represent large-scale cities and the roads between them? What about the digital connections we have with each other? We use **graphs**.

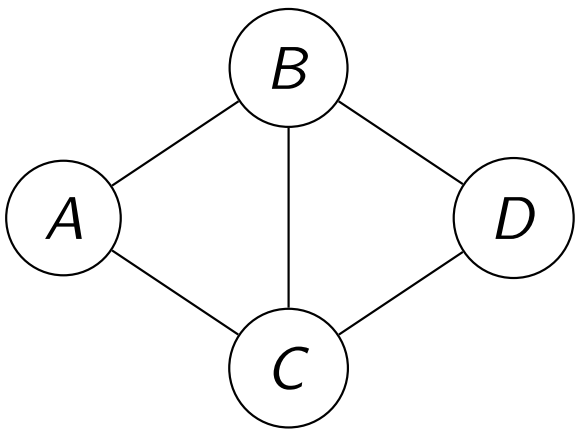
Each vertex V can represent a city or person, while edges E represent their relationship with one another.

Graphs Examples

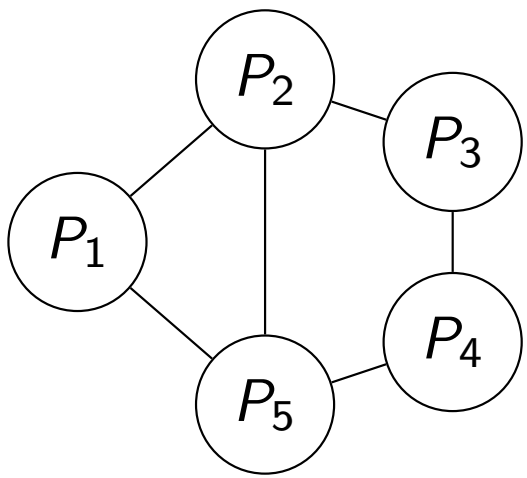
Graph

A **graph** $G = (V, E)$ is a set of vertices V which are connected via a set of edges E.

Cities Connected by Roads

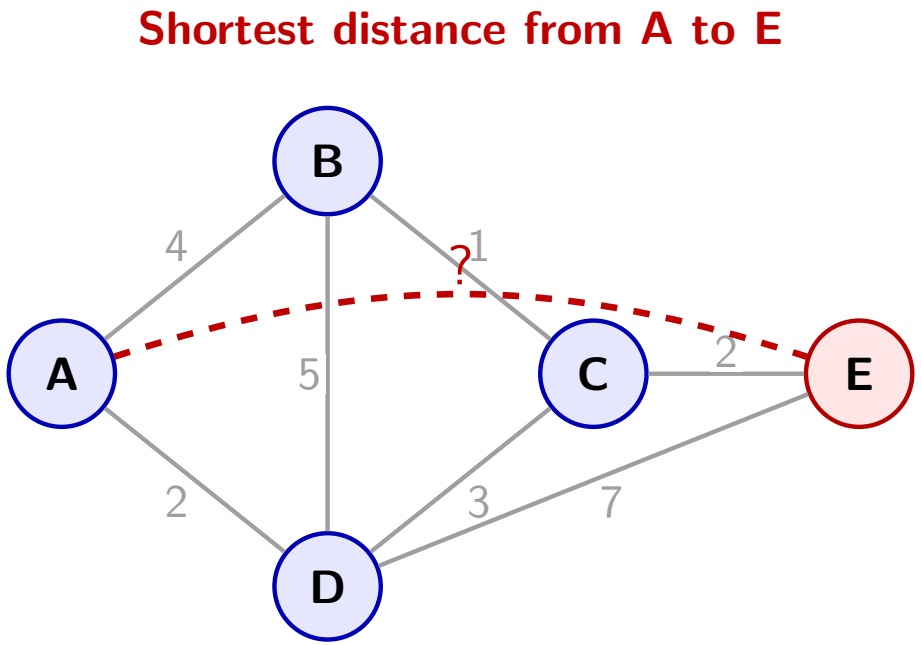


People Connected Through Social Media



Common Graph Problems

Naturally, common questions that involve graphs are inquiries like "what's the shortest path from X to Y," or "what's the cheapest way to fly between n cities." These types of problems are called **optimization problems**, specifically **shortest path optimization problems**.



Quick Definitions

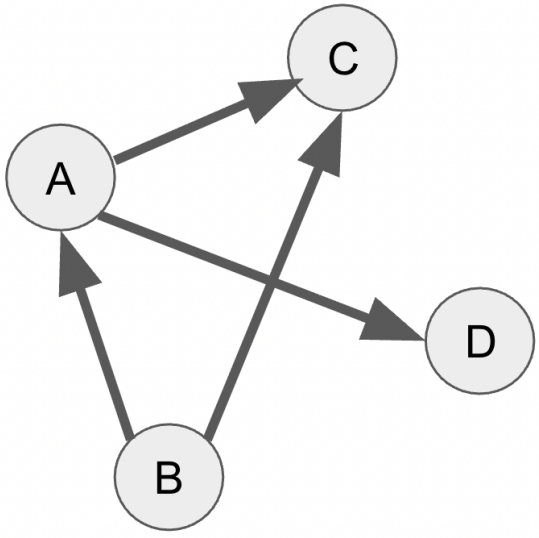


Figure: Directed Graph

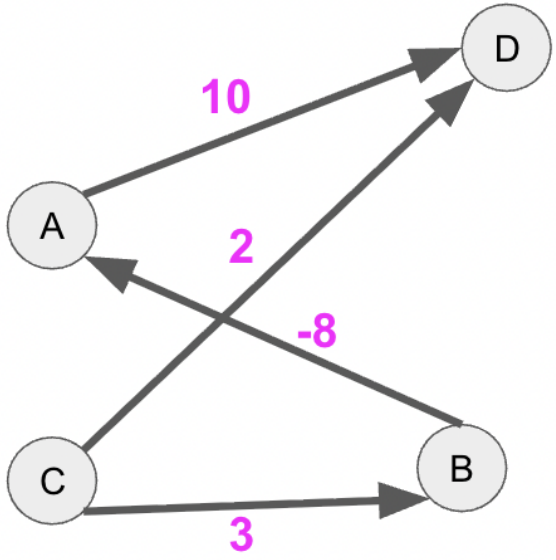


Figure: Weighted Graph

Quick Definitions

Algorithm

A graph algorithm is a series of steps or functions that are performed on a graph $G = (V, E)$ in order to accomplish a particular task.

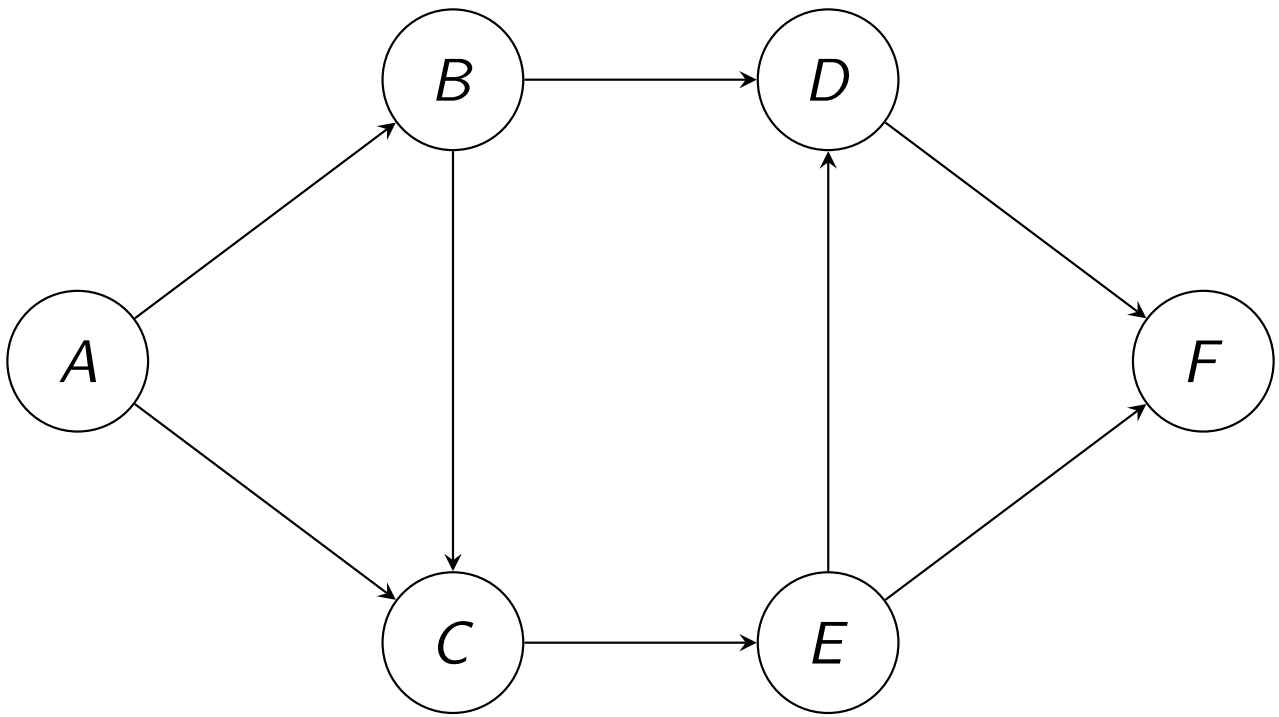
Algorithm Efficiency

Efficiency is defined as how many steps an algorithm takes to run until completion. We measure it in terms of how the runtime of the algorithm scales as the problem size increases.

Section 2: Algorithm Example: Breadth First Search

Breadth First Search: Intro

The **Breadth First Search** algorithm is an algorithm that finds the shortest path from a starting vertex to every other vertex in the graph, in a directed, *unweighted* graph, meaning every edge in our graph has the same cost, or **weight**.



Breadth First Search Algorithm

Breadth First Search Algorithm

1. Starting from a given vertex v_0, visit all vertices adjacent to v_0 and label them with distance 1, while storing those vertices in a set $v_1 = \{v_i, v_j, v_k, ...\}$.

Breadth First Search Algorithm

Breadth First Search Algorithm

1. Starting from a given vertex v_0, visit all vertices adjacent to v_0 and label them with distance 1, while storing those vertices in a set $v_1 = \{v_i, v_j, v_k, ...\}$.
2. Then, visit all vertices adjacent to a vertex in v_1 and label them with distance 2 while storing those vertices in a set $v_2 = \{v_n, v_m, v_p, ...\}$.

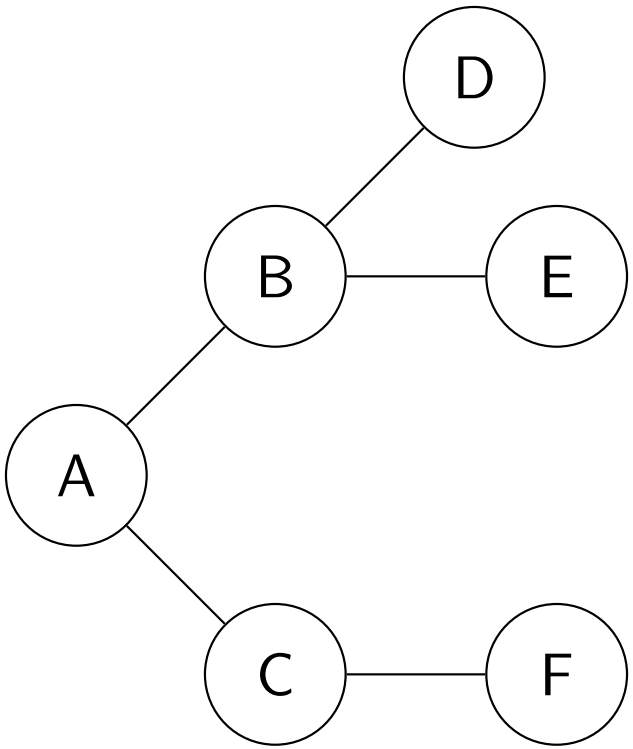
Breadth First Search Algorithm

Breadth First Search Algorithm

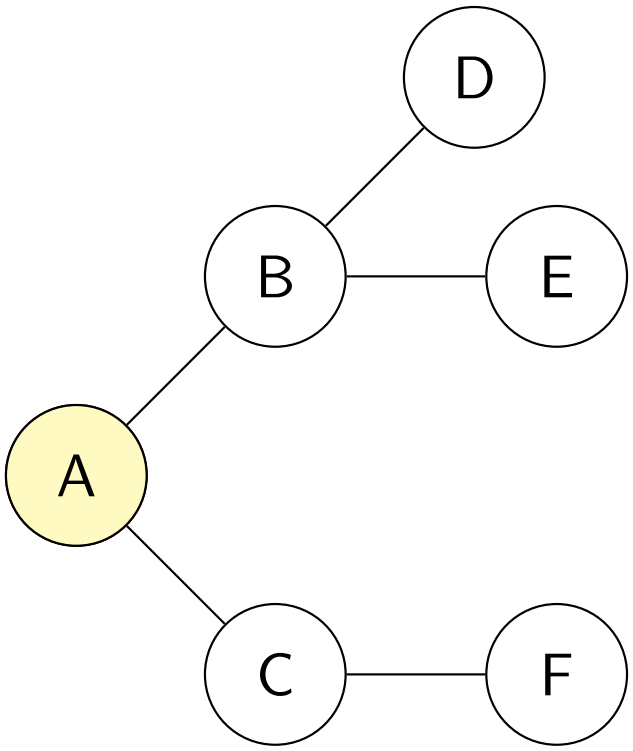
1. Starting from a given vertex v_0, visit all vertices adjacent to v_0 and label them with distance 1, while storing those vertices in a set $v_1 = \{v_i, v_j, v_k, ...\}$.
2. Then, visit all vertices adjacent to a vertex in v_1 and label them with distance 2 while storing those vertices in a set $v_2 = \{v_n, v_m, v_p, ...\}$.
3. Continue exploring the graph for $v_3, v_4, ...$ until there are no unassigned vertices.

Breadth First Search: Example

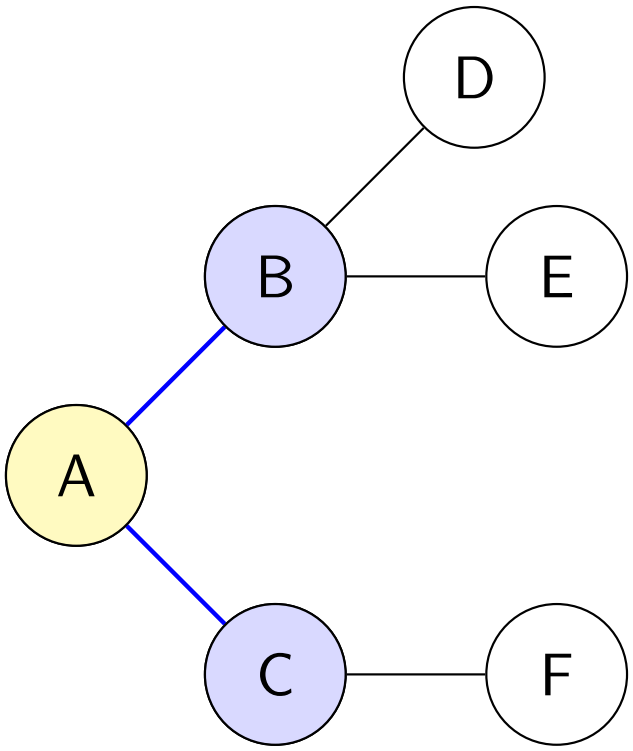
Let's see a walkthrough of this algorithm on graph G.



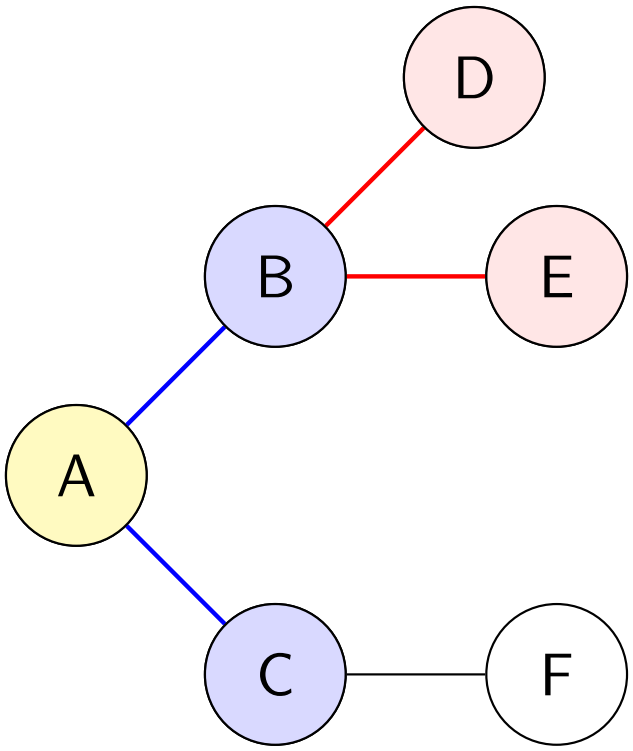
Breadth First Search: Example



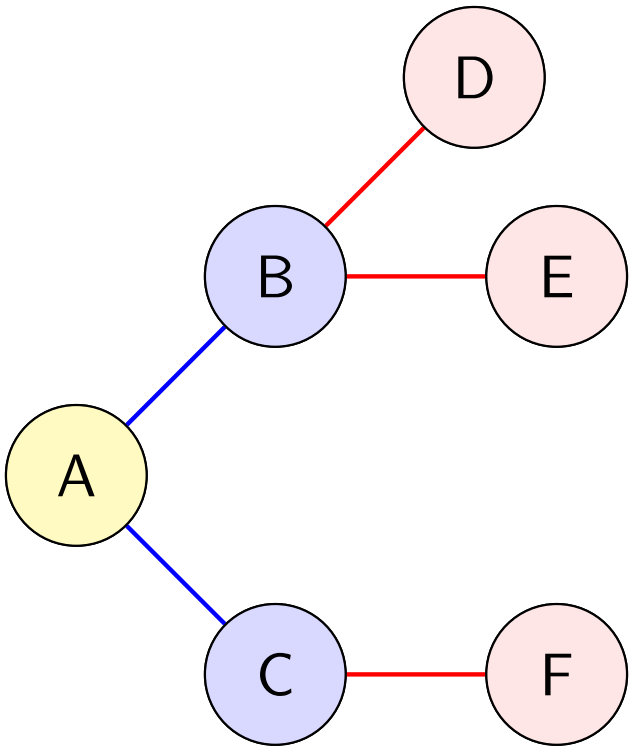
Breadth First Search: Example



Breadth First Search: Example



Breadth First Search: Example



Breadth First Search: Conclusions

Why this works

Breadth First Search explores vertices in order of their distance from the starting vertex. Therefore, the first time a vertex is reached, BFS has found the shortest possible path to that vertex in an unweighted graph.

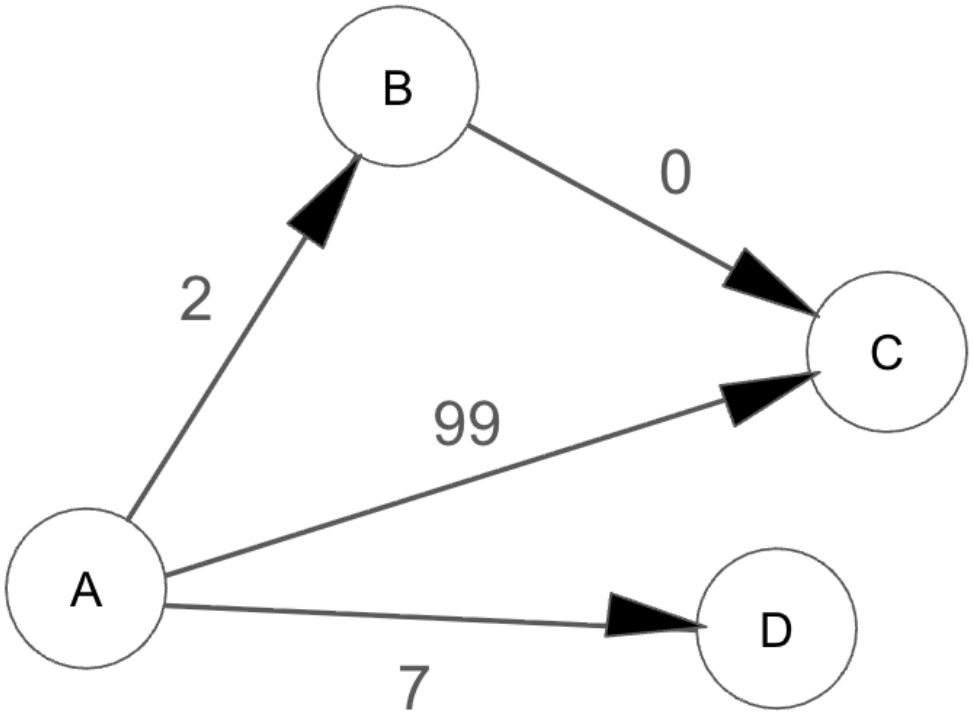
Efficiency

Since Breadth First Search visits every vertex once and travels "down" each edge once, we say Breadth First Search runs in linear $O(|V| + |E|)$ time.

Section 3: Algorithm Example: Dijkstra's Algorithm

Dijkstra's: Intro

Dijkstra's Algorithm is an algorithm that finds the shortest path in a directed, *weighted* graph where edge weights are nonnegative.



Dijkstra's Algorithm

Dijkstra's Algorithm

1. Begin at a source vertex s and estimate its distance to itself as $d(s, s) = 0$. Estimate distances to all other vertices $d(s, v)$ as infinity.

Dijkstra's Algorithm

Dijkstra's Algorithm

1. Begin at a source vertex s and estimate its distance to itself as $d(s, s) = 0$. Estimate distances to all other vertices $d(s, v)$ as infinity.
2. Repeatedly choose the unvisited vertex v with the smallest known distance.

Dijkstra's Algorithm

Dijkstra's Algorithm

1. Begin at a source vertex s and estimate its distance to itself as $d(s, s) = 0$. Estimate distances to all other vertices $d(s, v)$ as infinity.
2. Repeatedly choose the unvisited vertex v with the smallest known distance.
3. Update distances to neighboring vertices u using the formula $d(s, u) = \min\{d(s, u),\ d(s, v) + w(v, u)\}$

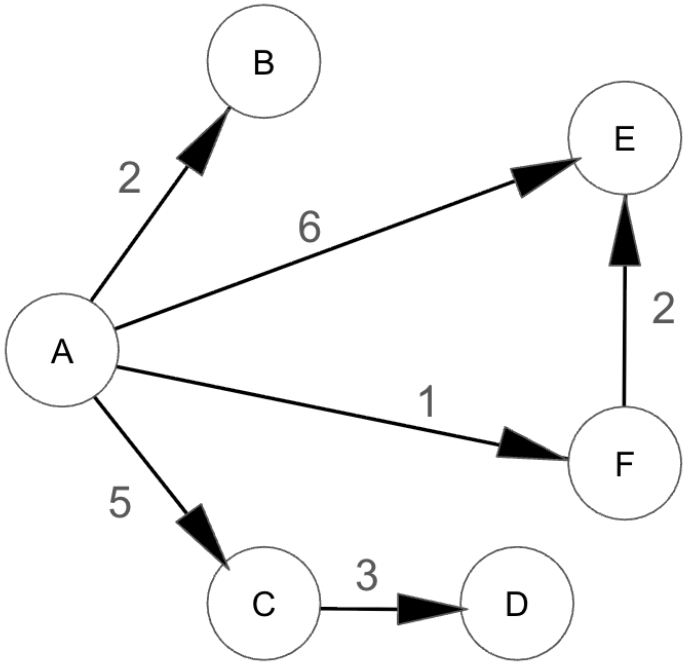
Dijkstra's Algorithm

Dijkstra's Algorithm

1. Begin at a source vertex s and estimate its distance to itself as $d(s, s) = 0$. Estimate distances to all other vertices $d(s, v)$ as infinity.
2. Repeatedly choose the unvisited vertex v with the smallest known distance.
3. Update distances to neighboring vertices u using the formula $d(s, u) = \min\{d(s, u),\ d(s, v) + w(v, u)\}$
4. Continue until all vertices are visited.

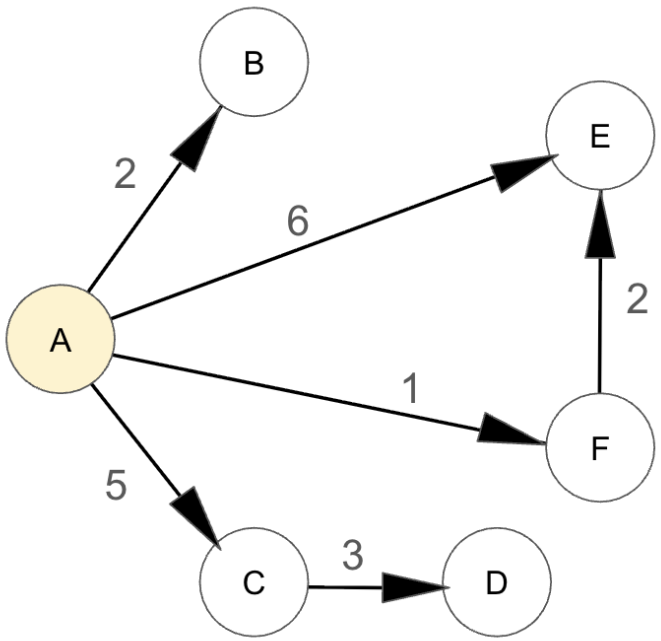
Dijkstra's: Example

Let's see a walkthrough of this algorithm.



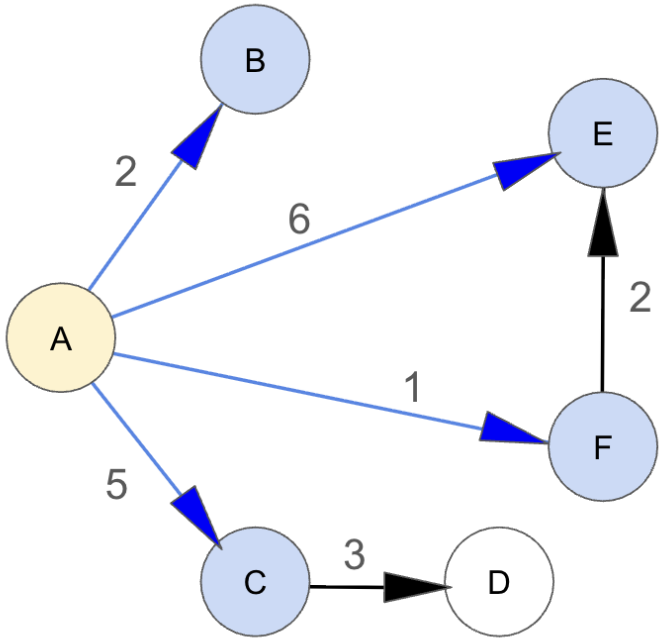
Vertex	Distance from A
A	
B	
C	
D	
E	
F	

Dijkstra's: Example



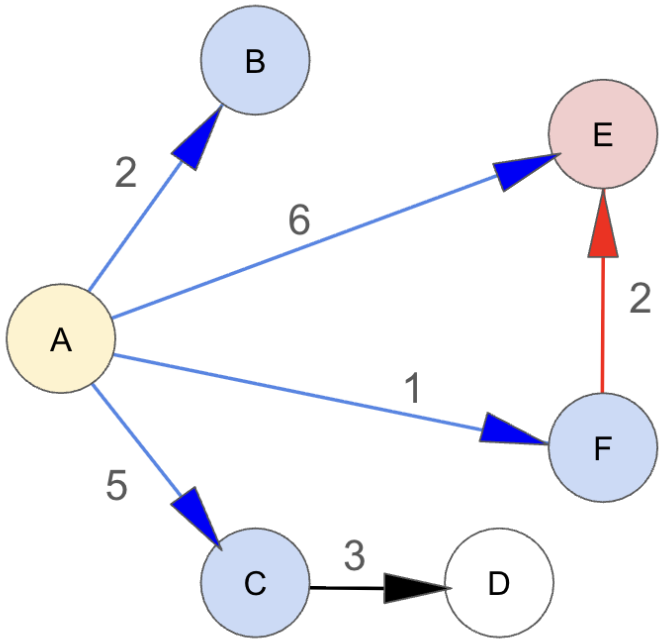
Vertex	Distance from A
A	0
B	
C	
D	
E	
F	

Dijkstra's: Example



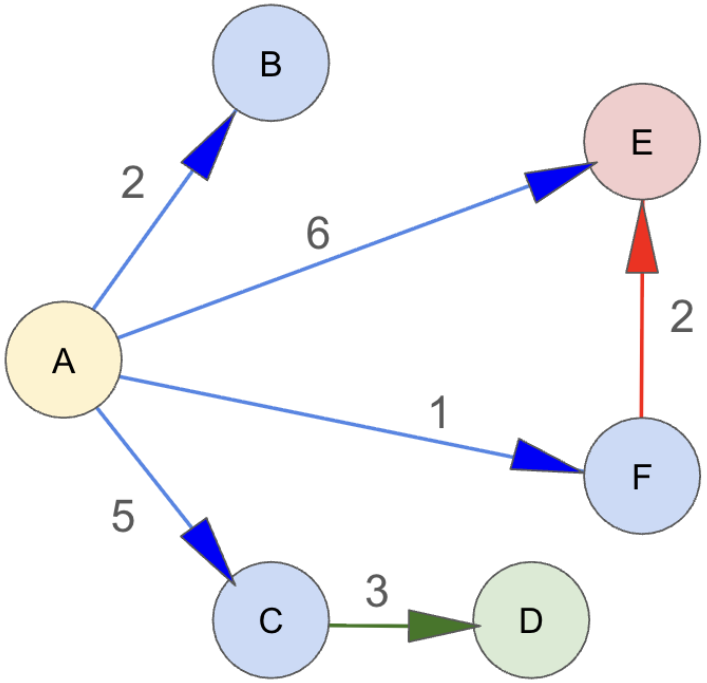
Vertex	Distance from A
A	0
B	2
C	5
D	
E	6
F	1

Dijkstra's: Example



Vertex	Distance from A
A	0
B	2
C	5
D	
E	3
F	1

Dijkstra's: Example



Vertex	Distance from A
A	0
B	2
C	5
D	8
E	3
F	1

Dijkstra's: Conclusions

Why this Works

Dijkstra's algorithm relies on a strategy called the "greedy choice" which relies on selecting the lowest value edges to form the shortest path. Since all edge weights are nonnegative, once a vertex is chosen, its shortest possible distance has been permanently determined.

Efficiency

The efficiency of Dijkstra's is approximately $O(|V|^2)$.

Conclusion & Further Directions

Final Conclusion

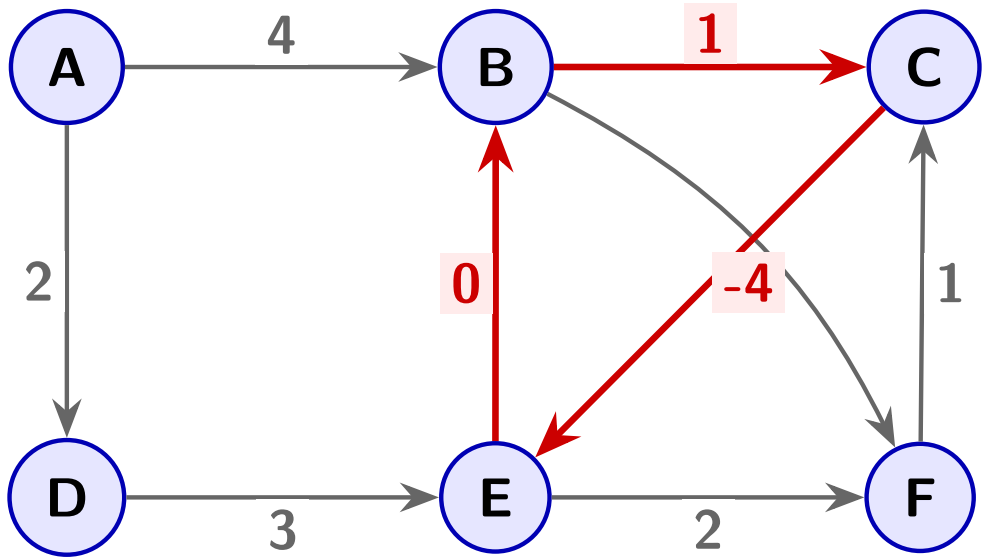
Today, we explored how graph algorithms solve shortest path optimization problems through:

- BFS for unweighted graphs
- Dijkstra's Algorithm for weighted graphs with nonnegative edge weights

Final Conclusion: Negative Cycle

However, many graph problems can even be more complex. For instance:

- Some graphs may contain **negative edge weights**, where traveling along an edge may decrease total sum. These require different algorithms, such as the Bellman-Ford Algorithm.



Negative cycle: $B \to C \to E \to B$
Total weight: $1 + (-4) + 0 = -3$

Applications

Acknowledgments

We would like to thank our mentor, Rosa Paten, for guiding us through the program and our combinatorics research. We would also like to thank the PRIMES Circle coordinators, Paige Bright and Mary Stelow, for providing us with this opportunity and for all of their support and advice throughout the program.

Acknowledgments

We would like to thank our mentor, Rosa Paten, for guiding us through the program and our combinatorics research. We would also like to thank the PRIMES Circle coordinators, Paige Bright and Mary Stelow, for providing us with this opportunity and for all of their support and advice throughout the program.

Thank you for listening!