

# Data Structures and Graph Algorithms

Gael Medina

Matheus Moreira

Mentor: Carlos Alvarado

Primes Circle 2023

Massachusetts Institute of Technology

May 12, 2023

# Introduction

---

In this presentation, We will be going over various different graph algorithms, how they compare to one another, and how we interact with them in the real world. But before all of that, what is an algorithm?

# Algorithms

---

An algorithm is a set of instructions or rules that detail or explain how to do something. It's like a recipe in the way that it lays a specific set of steps or rules that must be abided by to solve some problem.

An algorithm is comparable to a cake recipe. The recipe provides a series of instructions that include things like how to combine the ingredients, how long the cake should bake at a specific temperature, etc.

# Efficiency

---

Not all algorithms are as fast as each other, and a good way to compare the quality or speed of two separate algorithms is by checking their efficiency.

The time it takes for a computer program to complete an algorithm is called the run time, which can be affected by a variety of different things, such as the size of your input, computer program speed, and a multitude of other variables.

# Asymptotic Complexity

---

When looking at algorithms and their run times, we want to see how the algorithm deals as we vary the input size and we want to see how it works for arbitrarily large inputs.

Asymptotic complexity compares how our run times behave for large inputs of size  $n$ . We say that





$$f \in O(g) \iff \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$$

# Birthdays

---

## Problem





To think more about algorithms and efficiency, let's look at a problem. Given the image on the right, how could you sort those people from oldest to youngest?

	Name	Birthday
	Ava Berry	5/19
	Asiya Hooper	3/7
	Heather Michael	11/3
	Tony Brown	5/11

# Insertion Sort

## Steps

1. Start by looking at the first Person in this list





	Name	Birthday
	Ava Berry	5/19
	Asiya Hooper	3/7
	Heather Michael	11/3
	Tony Brown	5/11





# Insertion Sort (continued)

## Steps

2. Compare the person you are looking at with the next person in the list.

\*If the next person has a birthday before the person you started looking at, place that person at the beginning of the list and shift the other people down.

	Name	Birthday
	Ava Berry	5/19
	Asiya Hooper	3/7
	Heather Michael	11/3
	Tony Brown	5/11





	Name	Birthday
	Asiya Hooper	3/7
	Ava Berry	5/19
	Heather Michael	11/3
	Tony Brown	5/11



# Insertion Sort (continued)

## Steps

3. If a switch occurs, focus on the new first person on the list and continue comparing them with the next person in the list.





	Name	Birthday
	Asiya Hooper	3/7
	Ava Berry	5/19
	Heather Michael	11/3
	Tony Brown	5/11





	Name	Birthday
	Asiya Hooper	3/7
	Ava Berry	5/19
	Heather Michael	11/3
	Tony Brown	5/11

# Insertion Sort (continued)

## Steps

4. After completing and comparisons, focus on the next person on the list repeat the previous steps.




	Name	Birthday
	Asiya Hooper	3/7
	Ava Berry	5/19
	Heather Michael	11/3
	Tony Brown	5/11

	Name	Birthday
	Asiya Hooper	3/7
	Tony Brown	5/11
	Ava Berry	5/19
	Heather Michael	11/3

# Insertion Sort(continued)

## Steps

5. Repeat all the previous steps until all comparisons and insertions have been made and you are left with a finalized order list.

	Name	Birthday
	Asiya Hooper	3/7
	Tony Brown	5/11
	Ava Berry	5/19
	Heather Michael	11/3

# Insertion Sort(final)

---

## Efficiency


With insertion sort, in the worst case, you would have to shift an  $n$  number of items an  $n$  number of times over, which means at most you would have an  $n^2$  number of operations. This means that insertion sort runs in  $O(n^2)$ .


While this method certainly works, it is not very efficient. How can we speed it up?


# Merge Sort


## Steps

1. Split your list in half until you are left with smaller individual lists that contain only 1 element. Since these lists only have 1 element, they are already individually sorted.

	Name	Birthday
	Ava Berry	5/19

	Name	Birthday
	Heather Michael	11/3

	Name	Birthday
	Asiya Hooper	3/7

	Name	Birthday
	Tony Brown	5/11



# Merge Sort (continued)

---

## Steps

2. Group as many individual lists into groups of two as possible and recombine these groups while also sorting them.



	Name	Birthday
	Asiya Hooper	3/7
	Ava Berry	5/19



	Name	Birthday
	Tony Brown	5/11
	Heather Michael	11/3


# Merge Sort (continued)

## Steps


3. Continue grouping and sorting the lists until you are left with a fully sorted list.



	Name	Birthday
	Asiya Hooper	3/7
	Ava Berry	5/19



	Name	Birthday
	Tony Brown	5/11
	Heather Michael	11/3

	Name	Birthday
	Asiya Hooper	3/7

# Merge Sort (continued)




	Name	Birthday
	Ava Berry	5/19

	Name	Birthday
	Tony Brown	5/11
	Heather Michael	11/3

	Name	Birthday
	Asiya Hooper	3/7
	Tony Brown	5/11

	Name	Birthday
	Ava Berry	5/19

	Name	Birthday
	Heather Michael	11/3

	Name	Birthday
	Asiya Hooper	3/7
	Tony Brown	5/11
	Ava Berry	5/19



# Merge Sort (final)

---

## Run Time

Merge sort runs in  $O(n \log_2(n))$ .

Comparing sorted lists runs in  $O(n)$  since only one comparison must be made between each element in one sorted list to another.

The total number of comparisons to sort the list is proportional to the number of times the list can be divided in half until each sub list has 1 element, which is given by  $\log_2(n)$ .

# Graphs

---

A graph is a structure of ordered pairs of vertices and edges.

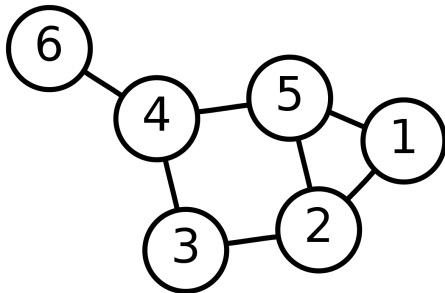


Figure: An Unweighted Graph

## Graphs (continued)

---

A graph with weighted edges is called a weighted graph.

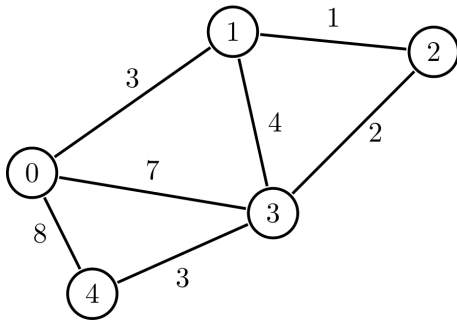


Figure: Weighted Graph

## Graphs (continued)

---

Similarly, a graph with directed edges is called a directed graph.

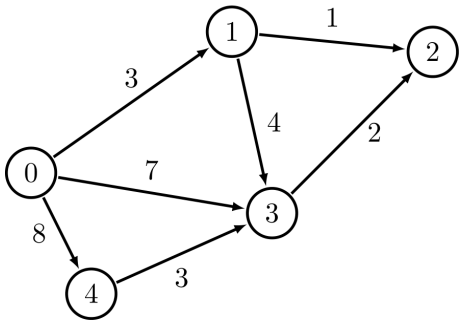


Figure: Directed Graph

# Shortest Path Problem

---

What is the shortest path problem?

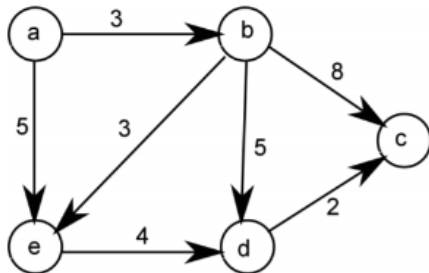


Figure: Directed Graph

# Dijkstra's Algorithm

---

Steps:

- Create a starting node and set that distance to 0
- Set the distance of every other node to infinity
- For unvisited nodes, select a new node based on the smallest distance
- Calculate the distance from the current node to reach the neighbor.
- After every neighboring node has been explored, mark the current node as visited.
- Repeat after every node has been visited

# Dijkstra's Algorithm

---

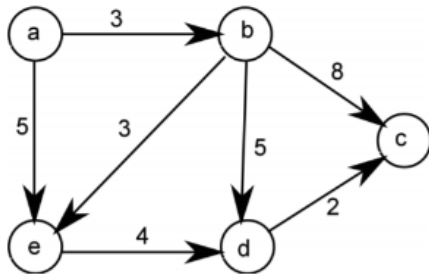


Figure: Directed Graph

## Dijkstra's Algorithm (continued)

---

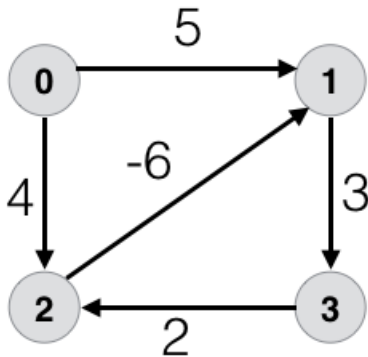
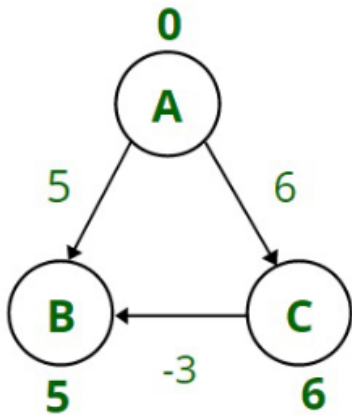


Figure: Directed Graph with a Negative Loop





# End

---

Thank you for your time and special thanks to the Primes Circle coordinators!