Graph algorithms in Swift

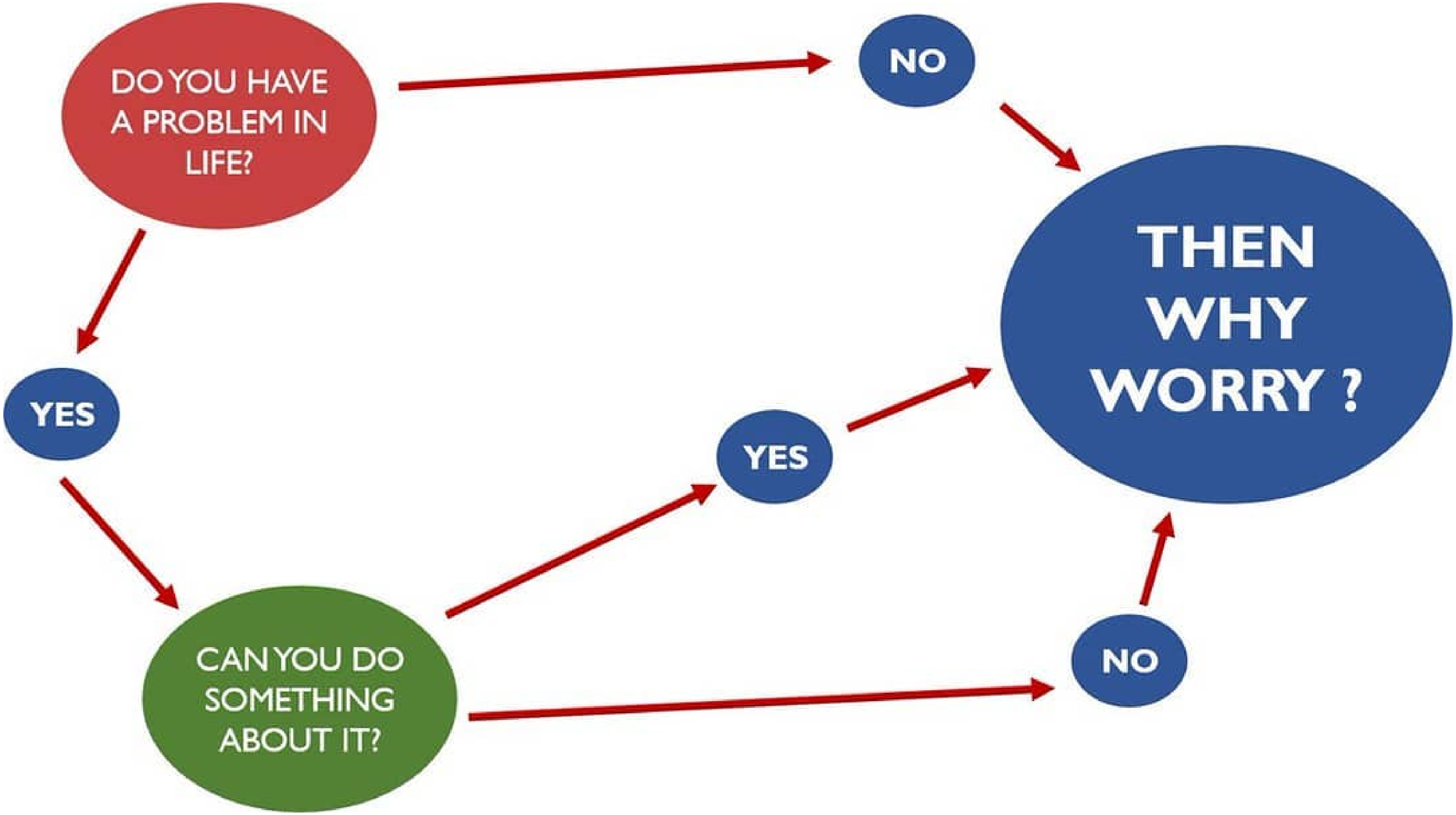
Sidharth Shah,

Fafadia Tech

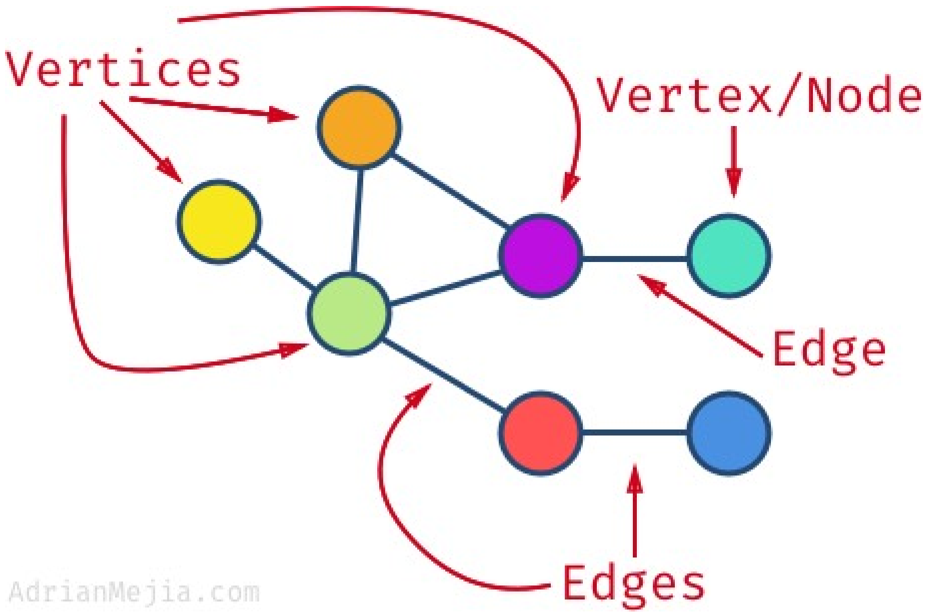
Introduction

Intutive Explaination

Graph is pretty common data-structure in computer science. It has many real-world applications.


DO YOU HAVE A PROBLEM IN LIFE?
NO
YES
THEN WHY WORRY ?
YES
CAN YOU DO SOMETHING ABOUT IT?
NO

Graphs are representation that specify:



1. Things aka **Verticies/Nodes** {which it represents}
2. Structure aka **Edges** {in form inteconnection}

Good reference for getting started: Graph Data Structures for Beginners

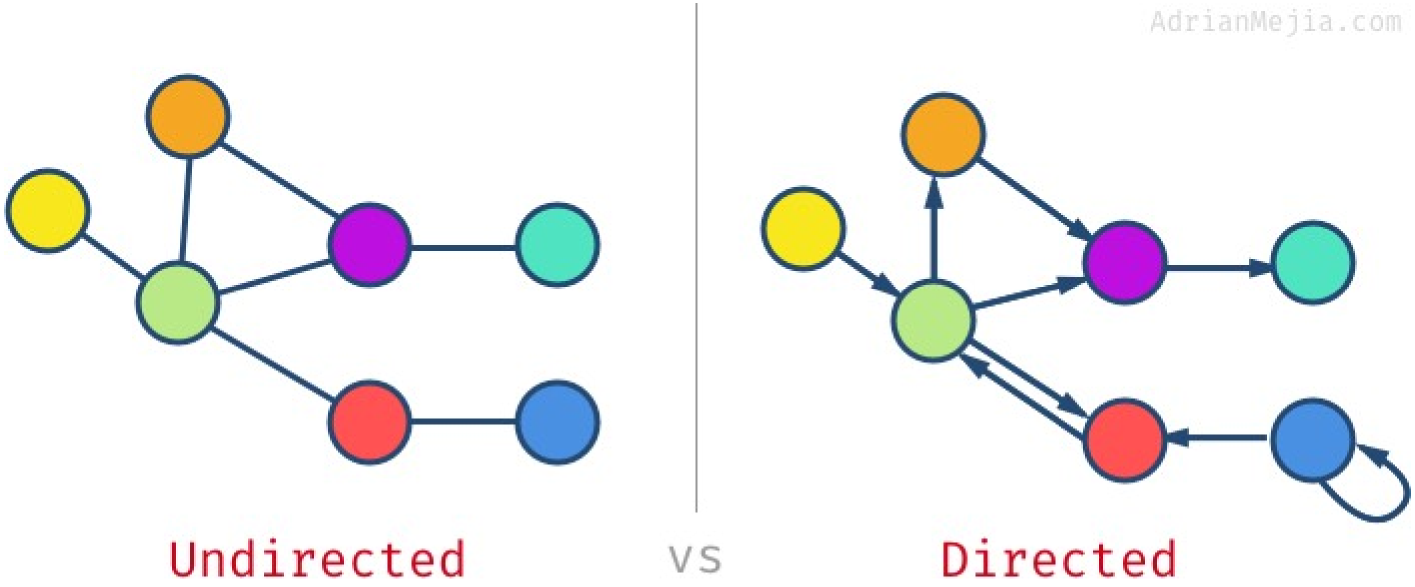
Is Tree and Graph same?

Nope, trees and graphs are **not same**. However Trees can be considered speical kind of Graphs. So Graphs is like super-set of Trees.

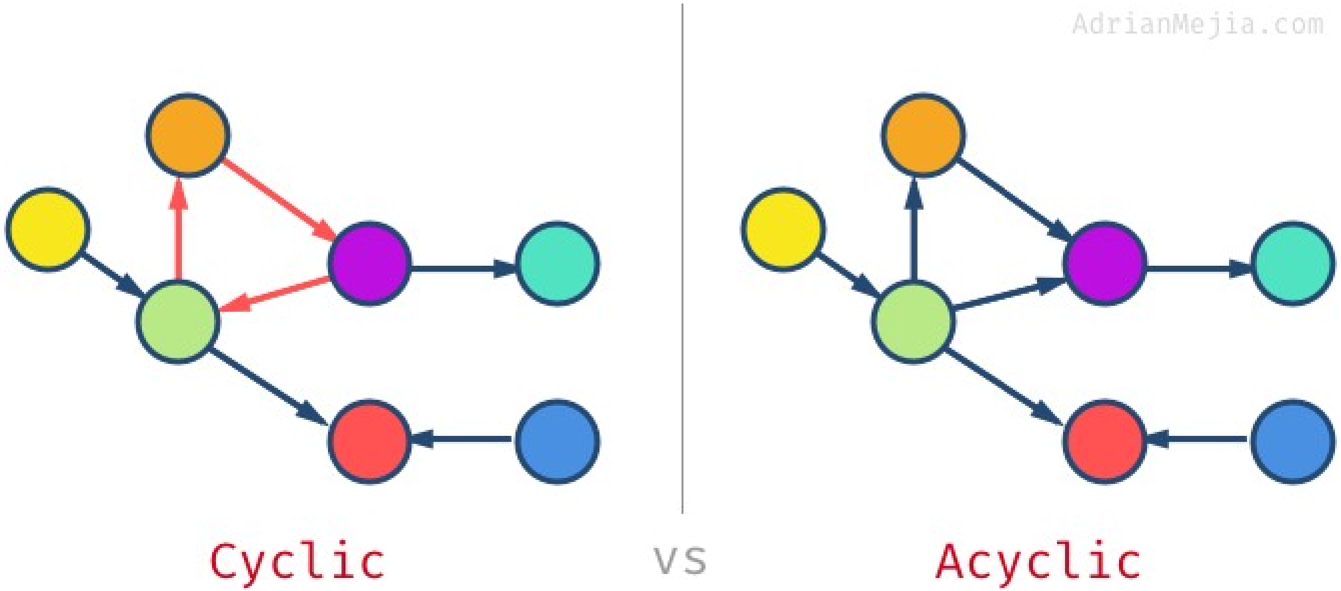
1. Graphs don't have strict levels
2. Trees mostly have stricter levels

Classification of Graphs

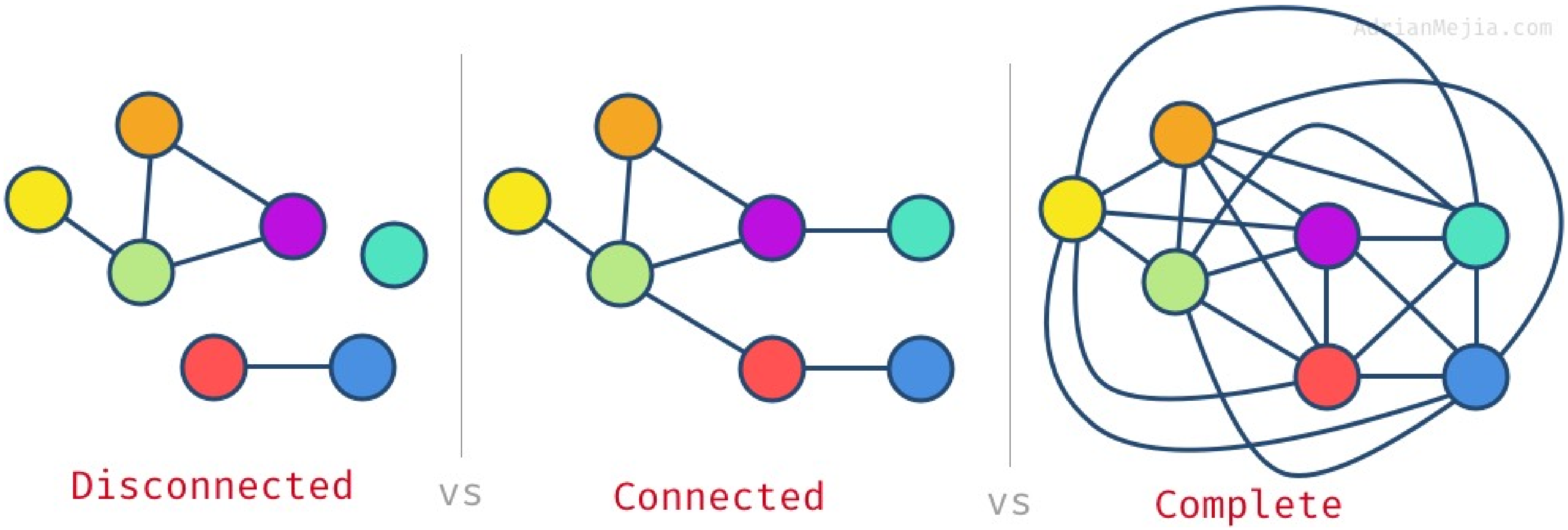
- Undirected vs Directed



- Cyclic vs Acyclic



- Disconnected vs Connected vs Complete



Few uses of Graphs in our daily life include:

1. Autocomplete
2. Google Maps
3. Facebook's "You may know" suggestions
4. XCode
    i. Detecting cyclic imports/references
    ii. Abstract Syntax Trees
    iii. Static Analysis via Call Graphs
    iv. Optimization: Notes on Graph Algorithms Used in Optimizing Compilers
5. Strategy Games
    i. StarCraft: AlphaStar
    ii. DOTA: OpenAI Five
6. Pretty much most of Deep Learning Algorithms

Store Graphs using

1. Adjacency List
2. Adjacency Matrix

Motivation

Lets say you work at one of the space agencies, your boss has asked you to build AI for robotic probe for soil-sampling different area of terrains and report back results.

Start with Simple one location

You will be given

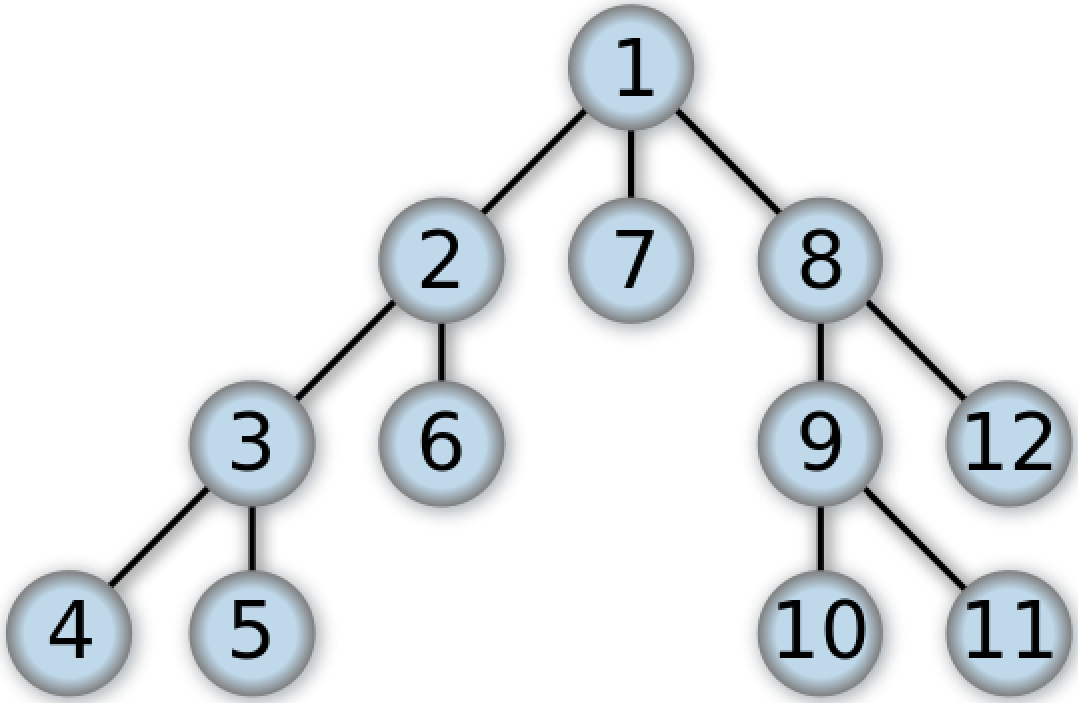
1. Starting Location
2. Map of the World {in form of a grid}, each cell
    i. Traversable
    ii. Not-traversable
3. Target Location

Useful Abstractions

1. **World**: Representation of maps as two-dimensional world
2. **State{s}**: Represent one of many possible configurations of world
3. **State Space**: Represents all of possible configurations of world
4. **Action{s}**: Mapping from current state to Child States
5. **Child States**: Valida states that be generated from *Current State*
6. **Target State**: Final state that we're trying to reach

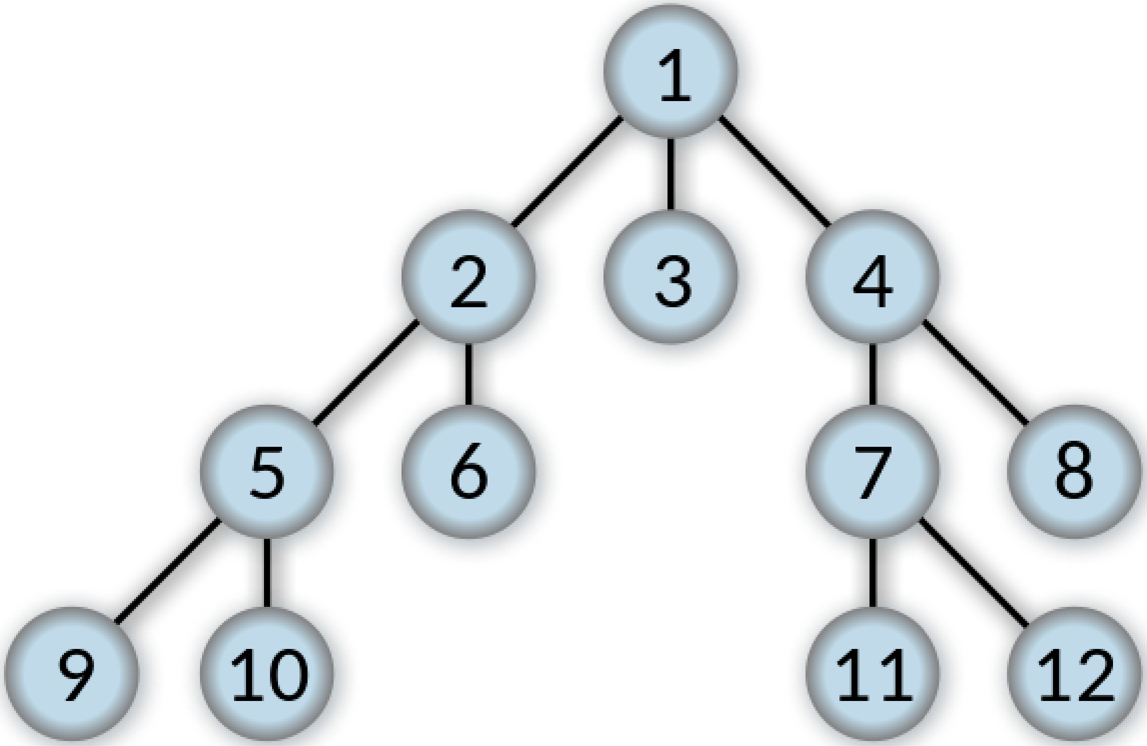
DFS Algorithm

Now that we've defined our core classes, we will start with Algorithm called DFS: Depth First Search

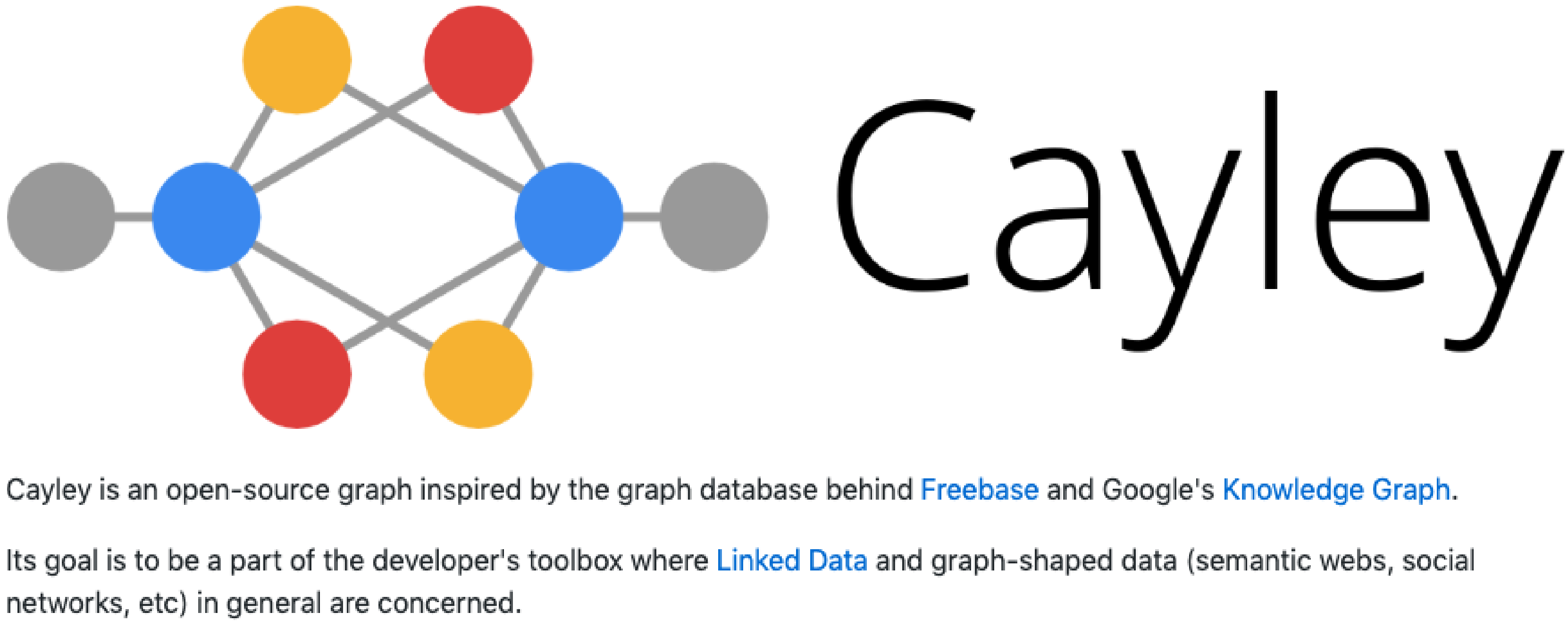


BFS Algorithm

BFS: Breath-First Search is an alternative, which expands all nodes at **current** level **before** going to **next** level



Fun Fact: Graph Databases



Cayley is an open-source graph inspired by the graph database behind Freebase and Google's Knowledge Graph.

Its goal is to be a part of the developer's toolbox where Linked Data and graph-shaped data (semantic webs, social networks, etc) in general are concerned.

https://github.com/cayleygraph/cayley

```
// Simple math
cayley> 2 + 2

// JavaScript syntax
cayley> x = 2 * 8
cayley> x

// See all the entities in this small follow graph.
cayley> graph.Vertex().All()

// See only dani.
cayley> graph.Vertex("<dani>").All()

// See who dani follows.
cayley> graph.Vertex("<dani>").Out("<follows>").All()
```

Summarize

1. Define **states** and **actions**
2. Define **child state generator**
3. Sepcify **start** and **termination** state
4. Use any **Search Algo** to find solution

References

1. Graph Data Structures for Beginners
2. Wikipedia: Depth First Search
3. Algorithms: Graph Search, DFS and BFS
4. Introduction to A* Algorithm