# Pathfinding

CU Game Dev Club

# Basic Idea

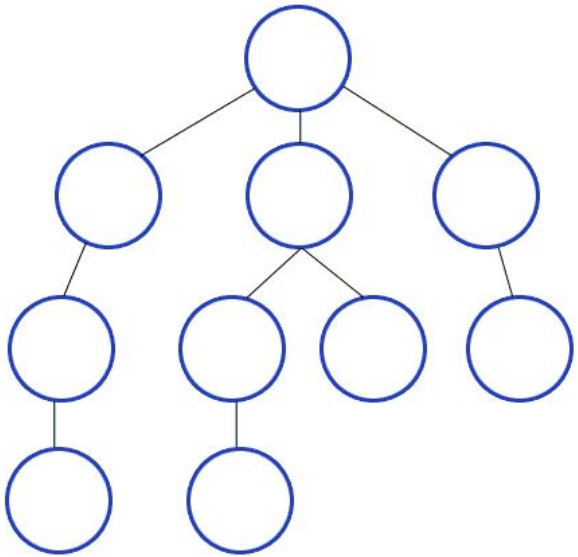
- You have some character
- That character needs to get some place
- How to get some place? Pathfinding!!!!!!

# How to Pathfind

- Graph Traversal is required
- Graph must be both mapped out between destination and target player
- For this lecture, we will not worry about making a graph, only how to traverse it

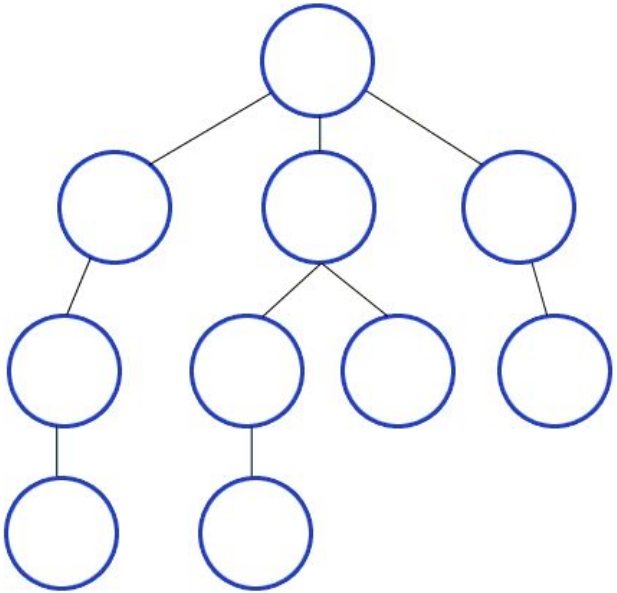
# Depth-First Search (DFS)

- Will try to find “depth” of graph - Find one solution from start to end before considering other ways
- Runtime is O(V + E)



# Breadth-First Search (BFS)

- Search Width of Tree before arriving at the end
- Runtime: O(V + E)



# Which Is Better?

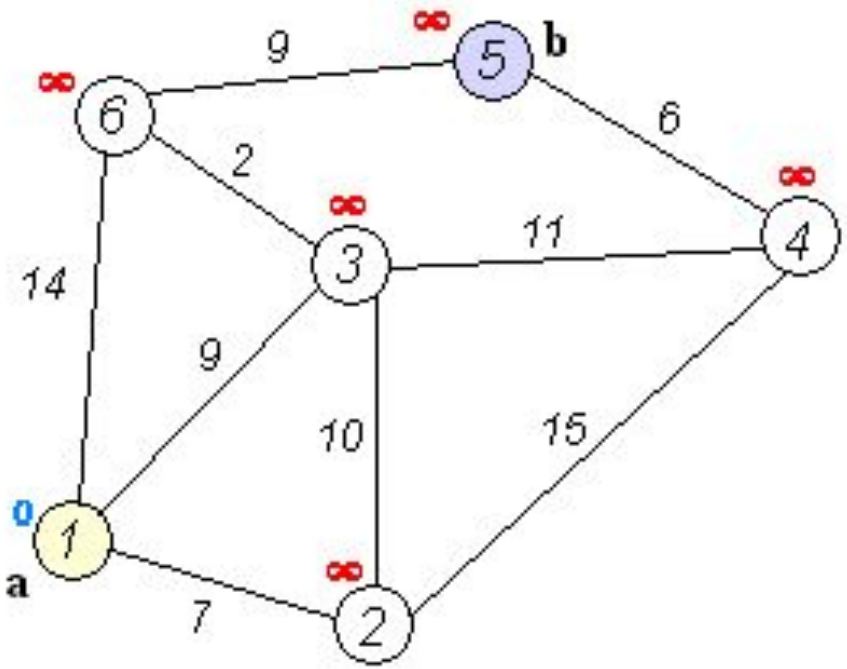
- Both DFS and BFS have the same runtime (and space-time) complexity
- But in general, one is much more used than the other …..

- BFS is a much better algorithm for pathfinding than DFS because:
  - DFS operates similar to a greedy algorithm
  - BFS is better at checking optimal solutions
  - BFS checks at each “band” of width what is the most optimal path to move on
- Djikstra’s and A* are variations on BFS
- HOWEVER, it does take longer for BFS to reach the end than DFS (in one single run)

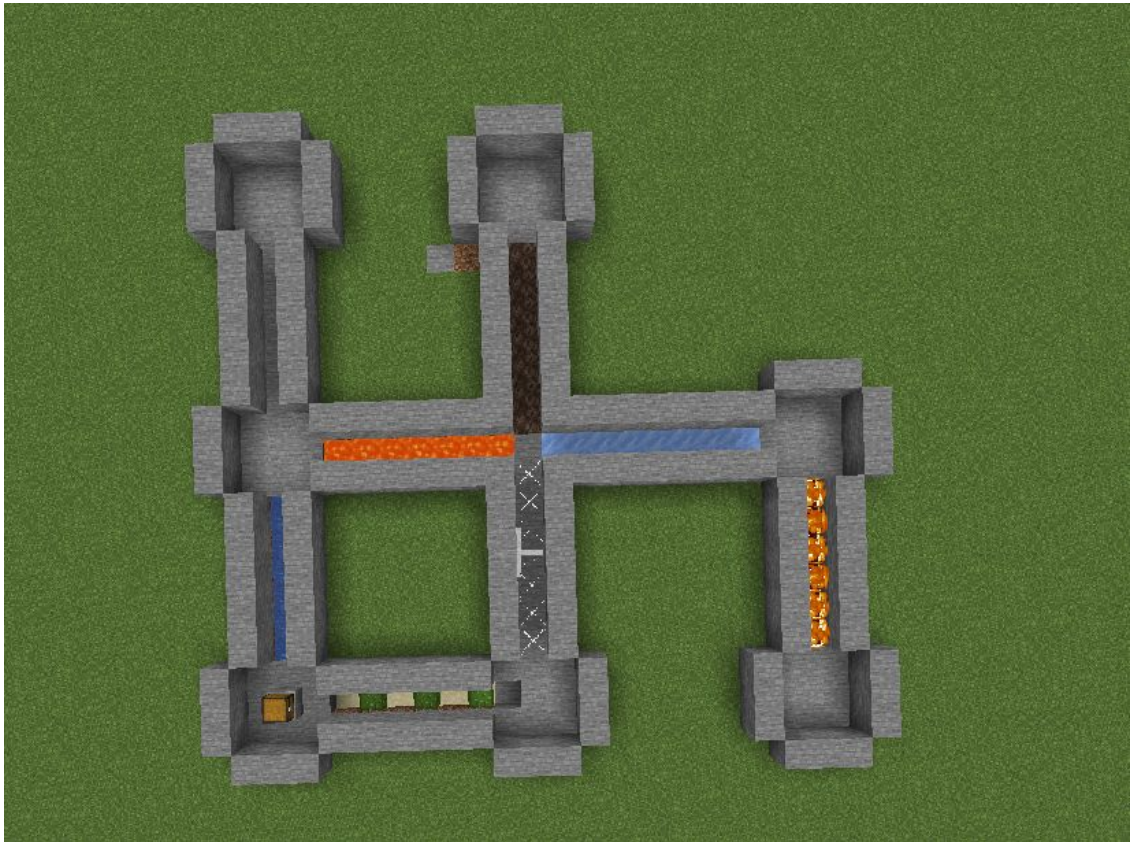
# Djikstra's Algorithm

- Assigns each node in the graph a "weight" - each node weight starts off at infinity except starting node - whose weight is zero
- The weight of the surrounding nodes are assigned according to edges connecting to starting node. The next node visited will be the surrounding node with the least weight
- Same process is repeated, except weight of surrounding nodes will now be weight of connecting edges + weight of current node
- Process is repeated until the end is reached
- On an unweighted graph, this will run exactly like BFS
- Runtime: O(E + VlogV)
- Pitfall is like BFS, still may need to check too many nodes until it finds a solution

# Djikstra's Algorithm



# In-Game Djikstra’s



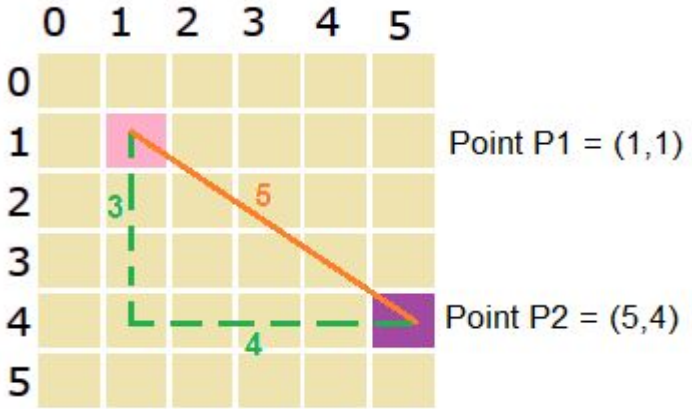
# A* Algorithm

- Possibly the most used pathfinding/graph traversal algorithm used
- Similar to an implementation of BFS/Djikstra's but introduces the concept of a heuristic
- Heuristic - rough estimation of how far you are from the end goal
- A* calculates heuristic for each surrounding node, moves to surrounding node with lowest heuristic

# Heuristics

- Many different types of heuristics for many different types of graphs
- Generally a sum of how far away a node is from starting node (G Cost) and how far away node is from the end (H Cost)
- For this example: Consider the graph to simply be a grid - each cell is a node
- There is a starting cell and an ending cell
- 2 general ways to measure G and H Costs: Euclidean vs Manhattan distance

# Heuristics




Euclidean distance = $\sqrt{(5-1)^2 + (4-1)^2} = 5$

Manhattan distance = $|5-1| + |4-1| = 7$

- Euclidian better on a completely open graph, uses distance formula directly
- Manhattan better if there are obstacles in graph, need to navigate around them

# Other Things

- Another form of Pathfinding is making Vector Fields - if there is one single target, then each node in graph is assigned a vector that will point towards target
- Graph traversal is really only the last “easy” step for pathfinding - Graph generation algorithms such as RRT and RRT* as well as spatial representation of nodes take up the bulk of the work

# Sources

- A VERY good video: https://www.youtube.com/watch?v=AKKpPmxx07w
  - His github repo for the project: https://github.com/danielmccluskey/A-Star-Pathfinding-Tutorial

Thanks!!!