Nolan Blevins

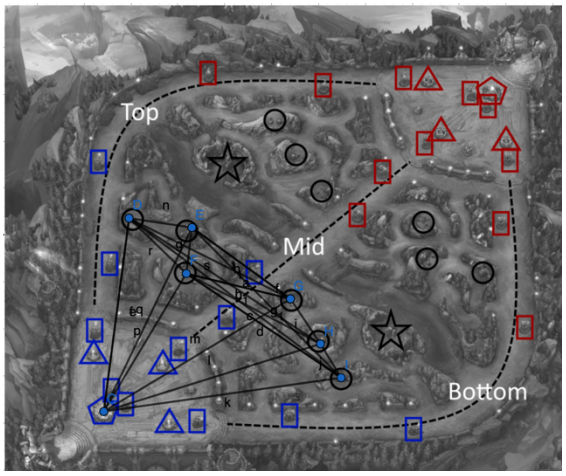
April 29, 2024

CSCE 590 Final Project – Dijkstra's Algorithm

Dijkstra's algorithm was created in 1956 by Edsger Dijkstra and has many use cases throughout different industries. Originally the algorithm was used to find the shortest path between two nodes in a weighted graph, but many variants have been made throughout time that are used to solve different issues. The algorithm is notable due to the efficiency and broad use. Some use cases include things such as network routing, pathing in game development, supply chain logistics and navigation systems such as GPS. In the perspective of optimization, Dijkstra's is relevant due to it solving a minimization problem by nature. Dijkstra's goal is to optimize pathing to provide the shortest path from a node to another. This can help with the minimization of fuel consumption, time spent traveling or even bandwidth usage. Overall, Dijkstra's algorithm is used as a tool in many optimization problems. The algorithm is fairly simple by nature but can become more complex depending on parameters. The basics of how the algorithm works is as follows. 1. Start at source node and set distance to other nodes at infinity 2. Greedily select the next node with smallest distance 3. On the current node consider all adjacent nodes and update shortest path if a shorter path is found 4. Repeat steps 1-3 until all nodes have been visited. These steps can be modified dependent on the specific problem, but Dijkstra's algorithm takes a greedy approach for solving the shortest path problem. The problem that I am choosing to use Dijkstra's to solve is in relation to a game I like to play called League of Legends. In this game one of the players responsibilities is to travel through the jungle between the three lanes and clear camps. I will be using Dijkstra's to optimize the time it takes to go through the jungle through pathing at

each of the camps. Firstly, I found a map online and then in game walked between each of the locations and timed how long it would take tracking my time in a table.

here is the map and table with the time being in seconds:

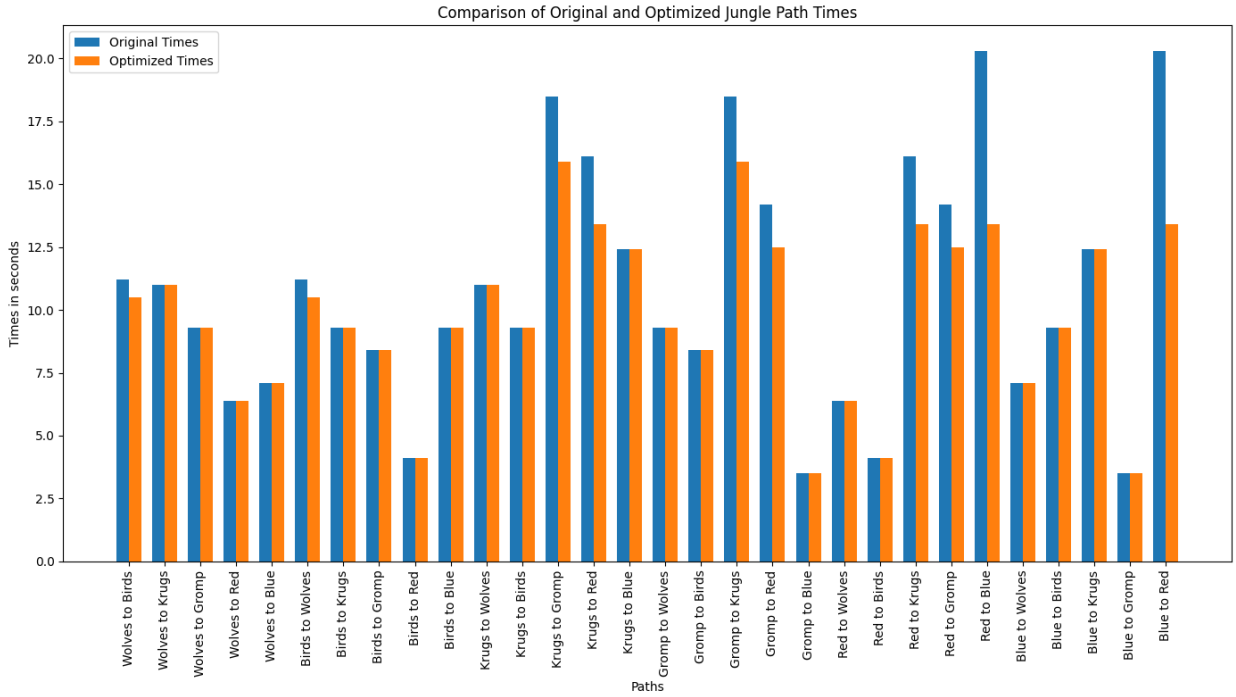


		Ending						
		Base	Wolves	Birds	Krugs	Gromp	Red	Blue
Starting	Base	0	24.2	25.4	25	26.2	26.1	25.3
	Wolves	8	0	11.2	5.3	6.4	6.8	7.1
	Birds	8	11.2	0	9.3	8.2	4.1	9.3
	Krugs	8	5.3	9.3	0	18.5	16.1	12.4
	Gromp	8	6.4	8.2	18.5	0	14.2	3.5
	Red	8	6.8	4.1	16.1	14.2	0	20.3
	Blue	8	7.1	9.3	12.4	3.5	20.3	0

In league of legends, we will be starting at base, and it always takes 8 seconds to recall which is why it is 8 all the way down, so we can exclude the base from the actual shortest path calculations, and just do the shortest path from each actual camp to another camp. This will allow us to find the best possible jungle times which could prove to be a competitive advantage.

After using this data with Dijkstra's algorithm, we can see the new shortest paths in terms of time by using a bar graph. Some of the times have not changed due to the current path being the shortest, but overall, there are a lot of improvements. One other note that should be added as well is that there is a plethora of champions (playable characters) to choose from and the one I used in this specific optimization is called Warwick. This matter because movement speeds of champion's change.

```
TERMINAL
→  Final_Project /opt/homebrew/bin/python3 /Users/nolan/Documents/school_
24/CSCE_590/Final_Project/Dijkstras_jungle.py
Shortest paths table (times in seconds):
   From/To    Wolves     Birds     Krugs     Gromp       Red      Blue
    Wolves         -      10.5      11.0       9.3       6.4       7.1
     Birds      10.5         -       9.3       8.4       4.1       9.3
     Krugs      11.0       9.3         -      15.9      13.4      12.4
     Gromp       9.3       8.4      15.9         -      12.5       3.5
       Red       6.4       4.1      13.4      12.5         -      13.4
      Blue       7.1       9.3      12.4       3.5      13.4         -
```



Link to code: https://github.com/nolanblevins/Dijkstras_Jungle