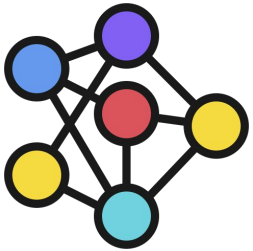
Path-Finding Panther



Rob Farmer & Erin Lee

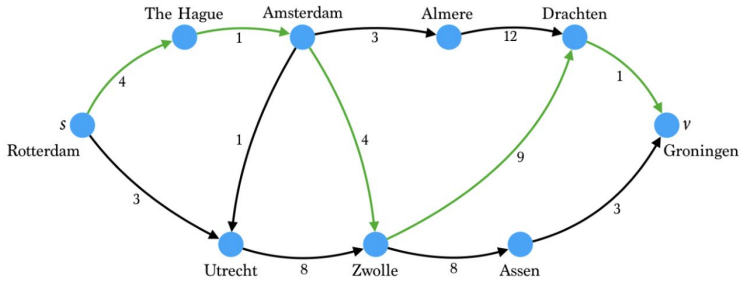
• CPSC 236-01: Visual Programming •

I. Intro

What is Dijkstra's algorithm?

- Edsger Dijkstra developed the algorithm.
- It was inspired by a map of the Netherlands.
- The algorithm computes the shortest route, or least-cost path through a network of nodes.
    - Note: This does not mean the algorithm is choosing the path with the fewest nodes!

The fastest **path** from Rotterdam to Groningen goes through The Hague, Amsterdam, Zwolle and Drachten.



The path found by the Dijkstra algorithm is highlighted with green

Image Source:
https://medium.com/@ndmitry/understanding-the-dijkstra-algorithm-by-intuition-and-step-by-step-2d132813b248

I. About Our App: “Path-Finding Panther”

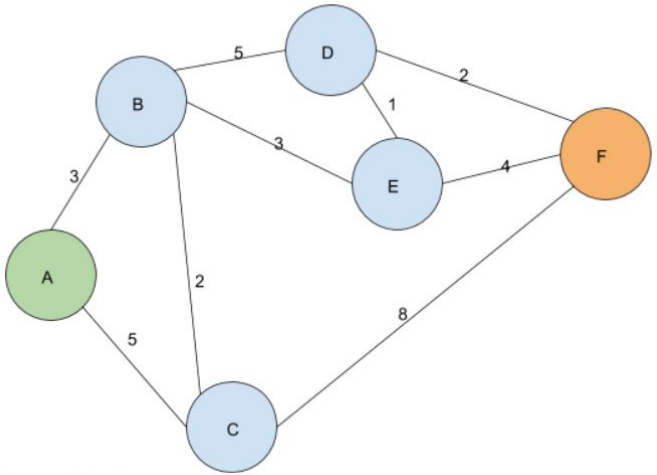
What is “Path-Finding Panther”?

- a game in Unity that uses Dijkstra’s algorithm
- **a simple implementation of a path-finding algorithm in game format**

App objective:

- to introduce students to network routing algorithms
- Our game sets up a network of paths, or roads, for the player -- in this case, our panther. (Chapman pride!)
- The ultimate goal is to find the shortest path from Node A to Node F.

II. System 1



Sample Mapping

Superclass: Game

Variables:
sprite panther;
int playerLives;

Methods:
PantherJump();
PantherKeyInput();
LifeCount();

Class: Platform

Methods:
Default();
PopUpPlatform();
DayModeMap();
NightModeMap();
AlgorithmicMap();

Class: PantherObjects

Variables:
int coins;
int count;
sprite apple;
sprite orange;

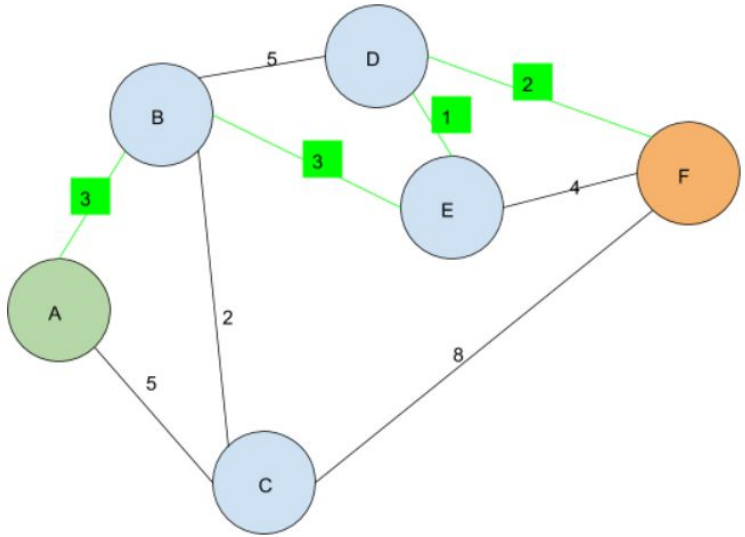
Methods:
CoinStorage();
CollectCoin();
RandomThrowAp[illegible]();
SetTrapFloor();

Class: Screens

Variable(s):
int levelSelect;

Methods:
showMenus();
hideMenus();
Pause();

II. System 2



Optimal Route

Superclass: Game

Variables:
sprite panther;
int playerLives;

Methods:
PantherKeyInput();
LifeCount();

Class: Paths

Methods:
EasyLeastCostPath();
DijkstrasPath();
TrolleyCartProblem();

Class: PantherObjects

Variables:
int currentPoints;
int totalPoints;

Methods:
PointScoreBoard();

Class: Screens

Variable(s):
int levelSelect;

Methods:
showMenus();
hideMenus();
Pause();
Play();
LeaderScoreBoard();
QuitGame();

II. System 3

Superclass:
GameManager

Variables:
Public List<Node> node;
Public GameObject
Button;

Methods:
Start ();
Restart();
Quit();
ShowButton();
HideButton();
OnButtonClick();
NumAttempts();
PromptUser();

Class: Node

Variables:
int vertex_a;
int vertex_b;
int vertex_c;
int vertex_d;
int vertex_e;
int vertex_f;
int costOfPath;
bool isNodeOccupied;
int attempts;

Methods:
void priorityQueue();
int NeighborsOfA();
int NeighborsOfB();
int NeighborsOfC();
int NeighborsOfD();
int NeighborsOfE();
int NeighborsOfF();
DisplayPaths();

IV. Conclusions

Currently, we have an indeterminate number of solutions.

- Areas to improve:
  - Implement a node generation system that can accept (x) number of nodes
  - Prefab connections