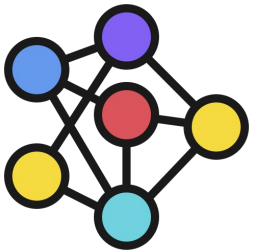
Path-Finding Panther



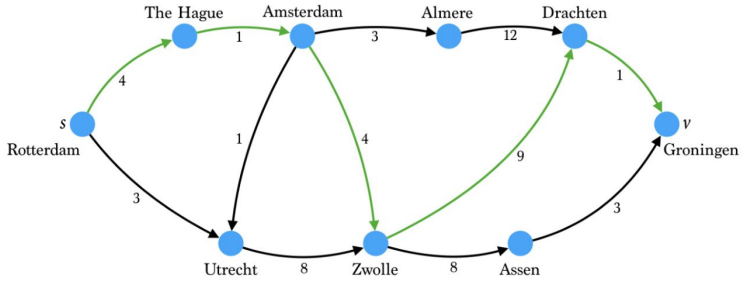
Rob Farmer & Erin Lee

CPSC 236-01

I. Intro

- **What is Dijkstra's algorithm?**
  - Edsger Dijkstra developed the algorithm.
  - It was inspired by a map of the Netherlands.
  - Computes the shortest/lowest-cost route through a series of nodes from point A to point Z
    - Doesn't mean it's the path with the fewest nodes!

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



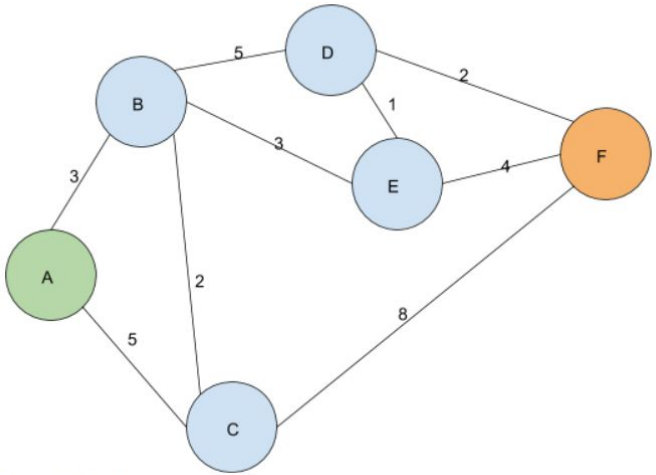
The path found by the Dijkstra algorithm is highlighted with green

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

I. About Our App

- **What is “Path-Finding Panther”?**
  - A simple implementation of Dijkstra’s algorithm meant to introduce students to cost-pathing in networks and routing.
- **App objective:**
  - **Teach a simple implementation of Dijkstra's algorithm in a game format**
  - Our game sets up a network of paths, or roads, for our panther with the end goal of finding 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;

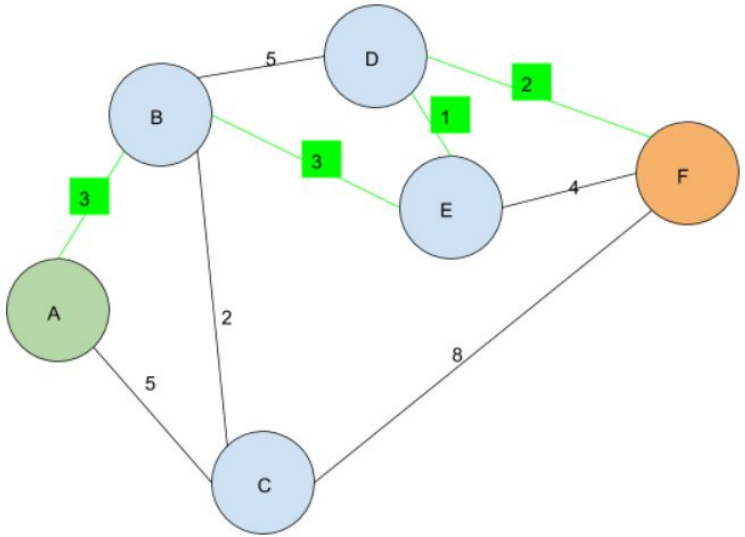
Methods:
CoinStorage();
CollectCoin();
RandomThrowApple();
SetTrapFloor();

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

1. What went well?
    a. Teamwork!
    b. C# programming!
2. What did not
    a. An indeterminate number of solutions
3. What we would have done differently
    a. Implemented a node generation system that can accept (x) number of nodes
    b. Prefab connections