Data Structures with C++ : CS189

Lecture 10-2: Graphs Algorithms

Recap

- A vertex has an identifier, and lists of edges that point in and out of it
- We store this in an unordered map where you can use an identifier to get a list of edges
- HasLoop is used for operating systems to check for deadlock
    - Operating Systems is a fascinating class in Uni. So are Databses and Algorithms.
        - Basically at CSULB, if Goldstein, Foss, or Monge teach it, take it.

Counting Steps

- Graphs are the tool, the algorithm we use graphs for is what changes
- Vertexes with arrows drawn out again
- How many steps does it take to get from A to B?

Shortest Path

- Remember that for each algorithm, there are three approaches in code.
- How would we find out StepsApart(T,T)?
    - Say the start is distance 0 from itself
    - Say all other vertices are infinity/unknown away from start
    - Find the vertex with the lowest current distance that hasn't been checked already
    - Set all of its out-links as having a distance of its own score plus 1. (If that is better than what they have.)
    -
    - That's it

BestDistance

- Let's add a new field to edges - a "weight" or "value"
  - Could be cost, or distance, or time, anything
  - StepsApart is just BestDistance where all weights are 1
  - This is why our edges are structs instead of vertices just having a list of vertex identifiers
    - An idea always worth considering
- We now have an unordered map of strings to structs with lists of structs
  - Welcome to 189

Dijkstra (DJ)

- Init vertices as before, again considering the three approaches to graph problems
- For each out, add my current score to the weight of the edge, and update the target vertex if the new score is lower
- Mark me as processed so I am not picked again
- Keep going until all nodes are processed
  - Can't stop the moment we process the target because some other path may have a "shortcut"

DJ's Return

- That algorithm gets you the cost of the trip to the target
- Add in a vertex identifier that we set whenever a score is updated
- Mark down "FromWhere" whenever a score is changed
- Now you can follow "FromWhere" backwards node by node to find the actual path you took

Optimizing

- Each step through the algorithm wants to find the node with the lowest score
- We have an ADT for that
- Put all the nodes in a priority queue, and the one that comes out is always the lowest possible node. No more searching

A* (Not on Homework)

- DJ has to search every node just in case there is a shortcut somewhere
- If nodes represent points in physical space (like Google Maps) then an exhaustive search would be the whole world
- Add a "2D crow's flight estimation" to the score of a node
  - This will bias the search towards the target
  - Game pathfinding is this plus adding other weights/costs to each step
    - Distance to bad guy, rough terrain, LoS, etc

Google Maps

- Make every intersection a vertex and every edge a road
- Give each road a penalty based on speed limit and current traffic
    - Current traffic can be based on historical data, or reports from other people using same app
    - Could be better - Google Maps doesn't recognize a left turn bears an additional penalty, for example

End

Graph algos are the coolest part of algorithms class