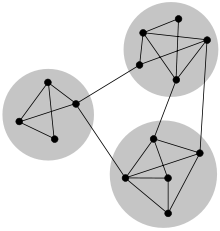
Reese McCoy

W11-12 Assignment

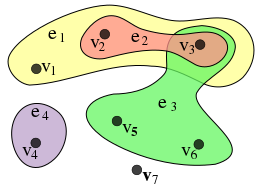
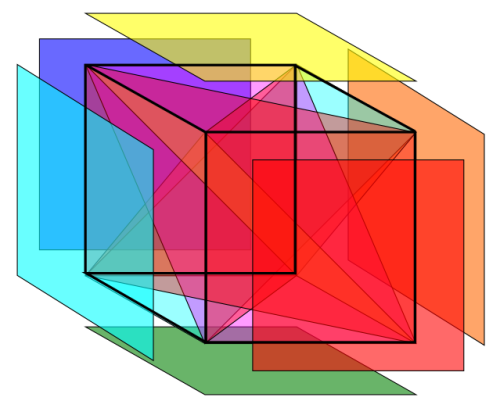
Dijkstra’s Algorithm used in Social Networking

1. **Explain the problem**
   1. Dijkstra's algorithm solves the single source shortest-paths problem on a weighted, directed graph. When Dijkstra’s algorithm is applied, it creates a tree of the shortest path from a starting vertex, to all other nodes in the graph. A social network is a social structure or graph, made of people or nodes, that have dyadic connections in relation to each cluster of relationships. These relationships are qualitatively defined as subsets of the graph where the interconnections of nodes are denser than the rest of the network. Generally, each sub relationship has weighted and modular communities that are interconnected, would be structurally dense and partition clustered. This meaning each community has sub relationships which might share interests, hobbies or simply friendships that are crucial for creating efficiently sorted networks. The goal is to find the shortest path from a specific node or individual, to a specified destination in a given network while minimizing the total cost of that path.
2. **Explain the proposed solution**
   1. Social networks can utilize Dijkstra’s algorithm by finding the modularity of the relationships between communities, all of which would solve all non-negative, weighted or non-weighted networks.



* 1. Implementing this in Java can be represented as a 3D array, in which each element and array subset are interconnected with one another. The reasons for a using a 3D array over a 2D array is mainly for creating a model of an individual in a complex network. Calling specific data such as interest[0][1] could quickly get a result across multiple people. Otherwise, you would either call each person heuristically to get the result or work from the end using tail recursion. Searching an interest first, then getting a result of people related could also work. The downsides are it becomes more complex as it is multidirectional. Or simply have an individual where they point to their interests, hobbies or friends, using a two-dimensional dataset. This is good for smaller models, but when you have millions of people in a social network, it would not be able to run as fast since it lacks complexity and depth.
  2. As an example, if one person is friends with another person, they may share common interests, do similar hobbies or are friends with other people. All of which have connections of some kind. Using tensors in multilinear algebra, these nodes can have multidirectional and connected scalars and vectors that can be used to speed up searching, predict partitioned results and know the interests and relations for future suggestions.

1. **Explain in detail how it relates to the course's explanation of the algorithm. Including a small example and the relationship to the algorithm.**
   1. The course defines Dijkstra’s algorithm as “solving the single-source shortest paths problem on a weighted, direct graph G = (V, E) for the case in which all edge weights are non-negative.” Although the book represents Dijkstra's algorithm in a two-dimensional bipartite graph, tensors can relate this algorithm in a three-dimensional bipartite hypergraph. Connecting nodes in a three-dimensional hypergraph can increase efficiency and complexity of morphic models. This in return would be capable of topologically associating weighted edges with more than two endpoints, allowing multiple graphs to interweave with one another. These connected graphs can have the same vertex yet host multiple other complex cycles.



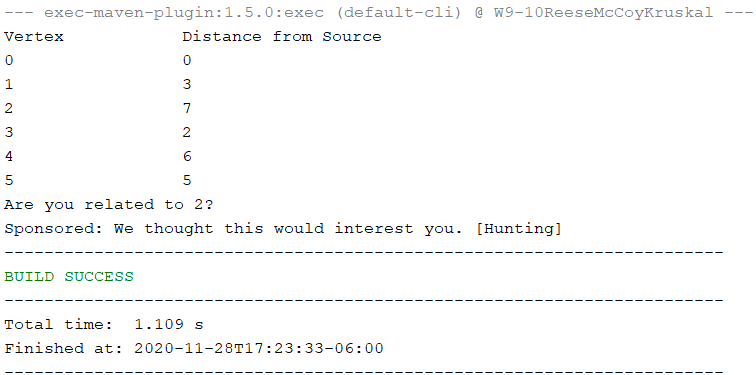
In these images, each plane is represented and connected to a single vertex. For example, an individual would be one plane or cluster that has its own unique endomorphic characteristics, while still isomorphically existing in relation to other planes or clusters.

* 1. For smaller connections or friend groups, Dijkstra’s algorithm works just fine regarding an amortized runtime of O((|E| + |V|) log|V|). This using binary min heaps and priority queues. Where connections are complex and large in dataset size, runtime begins to grow logarithmically at O(|E| + |V|^2), if not more depending on the complexity. For this reason, in order to achieve better runtime for larger datasets, we would need to implement multilinear algebra. Specifically using tensors which can be used to interconnect relationships between each individual, speeding up the interaction of these connections. Having clusters of data in subsets, all of which are interconnected, can speed up runtime since many vectors and scalars can be directed and pointed to each other’s subset, therefore reducing redundant data manipulation.
  2. Finding the shortest path using a three-dimensional hypergraph would use the same relaxation of edges as does the two-dimensional. The only difference is the relaxed values are depended upon more than one edge. First imagining a two-dimensional balanced Dijkstra graph G, where from one node to its destination node, is the most efficient and shortest path. Then add an additional dimension where the analogous path Z, is in relation to X and Y. Here, the third dimension matches across graph G, to element T as a hyperedge with the shorter path, therefore the non-adjacent edge of path T is shorter. Something to note, a three-dimensional social networking algorithm really is not prevalent in the public eye. Companies like Facebook, Google or Twitter may use something similar but there is very little to no information on the internet that talks about it.
  3. In the end, the objective is search through a large dataset as efficiently as possible, while at the same time be able to connect individuals, relations, hobbies, characteristics and other metrics together in a model, three dimensional. Utilizing and finding the shortest path in the graph or model can help narrow the scope of dataset efficiency. Heuristically solving and scaling the scope with recursion can help save on space where we also create the social network as a three-dimensional isomorphic hypergraph can help speed up searching, predict partitioned results and know the interests and relations for future suggestions.

1. **Program Evaluation**

This program utilizes Dijkstra’s algorithm where it finds the shortest path. Of the 6 nodes, it prints the shortest paths. The next two parts are finding relationships between people:

* The smaller the distance, the more likely it is you are related to that person. This would get too complex on the Facebook level, which would evaluate city, schools, last name, etc. Instead of a number, it would return names, but possibly it would be a number for indexing reasons.
* Based on the values in this example, it would suggest related results. In practicality, it would a string array, but that is not the point. In the real world, if someone is a hunter, it would connect that node to that plane or cluster of nodes where it would advertise related content.



The biggest issue I had was trying to implement a 3D array into Dijkstra’s algorithm, but was unsuccessful. The issue was it was just printing the maximum value set to infinity. That is above this assignment anyways. That file is attached. If this was actually implemented, it would be technically more than three-dimensions depending on the number of relationships. When each vertex is set to infinity, I couldn’t figure out the indexing. Chances are that is all it was. Looping through the graph[][][] with not iterating up correctly. Since that is not the whole purpose of this assignment and the number of extra stuff I included in the research, I decided to call it.