**CA336 – Assignment 2**

**Graph Search [20 marks]**

**Answer all 3 Questions**

Student Name: Ronan Kelly

Student ID: 21374881

**PART 1. (4 marks) Draw your Graph & Nominate you’re A,B,C**

A diagram of different cities

Description automatically generated

Instructions: Replace this with a subset of your graph (8-10 nodes, relationships, weights etc.) and identify A, B and C in your graphs (from roughly the same area).

Figure 1: [*Graph Name + 1 line description*]

**American Sports Connections**

This is a subset of my graph showing the link between Cities who have won the Super bowl from the Central region and have population between 600k-700k.

A – Kansas City

B – Boston

C – Memphis

***(For all questions, A is your starting node)***

**PART 2 (7 marks)**

1. Why did you select that starting node?
2. Why is it important in your graph?
3. Use the Minimum Spanning Tree. What problem are you trying to solve? What is it about the minimum spanning tree that suits your problem?
4. Would the SSSP algorithm be better (or worse) for your approach? Explain your answer.
5. Using Neo4j, call gds.alpha.spanningTree (be sure to set: orientation: ’UNDIRECTED’) and click the ‘Text’ tab to display your results: screen grab or copy-and-paste the results into the slot for figure 3.

|  |
| --- |
| 1. I selected Kansas City as my starting node as its somewhat isolated from other city nodes making it good for analysis and traversal. 2. As a specific node it wasn’t necessarily important however, as a whole it allowed for analysis between the nodes as it doesn’t link directly to any of the other cities. 3. The problem the minimum spanning tree solves is linking 2 cities within the US by either sporting success, location or population. The minimum spanning tree algorithm efficiently connects these cities based on these factors. 4. The SSSP algorithm would be better for my approach as it provides the most efficient route between the 2 cities I select based on the weights provided. |

Figure 2. [*place some caption here*]═════╤═══════════════╤════╕

│City │Destination │Cost│

╞═══════════════╪═══════════════╪════╡

│"Kansas\_City" │"Central" │7.0 │

├───────────────┼───────────────┼────┤

│"Central" │"Nashville" │7.0 │

├───────────────┼───────────────┼────┤

│"Kansas\_City" │"NFL" │3.0 │

├───────────────┼───────────────┼────┤

│"NFL" │"Denver" │3.0 │

├───────────────┼───────────────┼────┤

│"NFL" │"Washington\_DC"│3.0 │

├───────────────┼───────────────┼────┤

│"Washington\_DC"│"600k-700k" │4.0 │

├───────────────┼───────────────┼────┤

│"600k-700k" │"Memphis" │3.0 │

├───────────────┼───────────────┼────┤

│"NFL" │"Las\_Vegas" │3.0 │

├───────────────┼───────────────┼────┤

│"NFL" │"Boston" │6.0 │

└───────────────┴───────────────┴────┘

**PART 3 (9 marks)**

1. Create (manually draw) a sub-graph of your graph with only the first 5 nodes (include A and *either* B or C). Insert a new (weighted) relationship where any of the nodes is unconnected. You should create a single connected graph.

A diagram of different cities

Description automatically generated

I linked the

1. Starting from the A node, compute the shortest path from this node to all other nodes. In the process, create a table like that shown on slide 3.47. If you cannot find a symbol for infinity, use INF instead.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | 1st from A | 2nd From A to B to next | 3rd from A to C to next | 4th from A to D to next | 5th from A to E to next |
| Kansas\_City (A) | inf | 0 | 0 | 0 | 0 | 0 | 0 |
| NFL (B) | inf | Inf | 3 | 3 | 3 | 3 | 3 |
| Las\_Vegas (C) | inf | Inf | 1350 | 6 | 6 | 6 | 6 |
| Denver (D) | inf | Inf | 605 | 6 | 6 | 6 | 6 |
| Boston (E) | inf | inf | 1432 | 9 | 9 | 9 | 9 |