

1810705 - Search and ML Clustering

by Kristian Cole

Submission date: 04-Jan-2022 01:49PM (UTC+0000)

Submission ID: 167522346

File name: 12521_Kristian_Leigh_Cole_1810705_-_Search_and_ML_Clustering_393826_69308948.docx
(312.08K)

Word count: 3267

Character count: 14041

Contents

1.0 Task 1	1
1.0.1 Data Structures Used In Program	1
1.0.2 Depth First Search Algorithm	1
2.0 Task 2	2
2.0.1 K- Means Method	2
2.0.2 K- Means Algorithm	4
2.0.3 Criticism Of K-Means Clustering Algorithm	4
3.0 Task 3	4
References	10
Appendix	11
1.0.1 User Guide	11
1.0.2 Test Schedule	12
1.0.3 Program Listing	12

1.0 Task 1

1.0.1 Data Structures Used In Program

The data structures used within the application:

1. dStruct - A Struct entitled 'dStruct' containing three variables (int x, in y, char letter), on visiting a successful route in the map the x,y coordinates are added. If a sign is found, the sign is added to the struct via the letter variable.
2. signStack - A stack of dStructs that collects the signs found information; this is added to the struct then added to the sign stack.
3. dfsVisited - A stack that holds the path taken by the application; this is the same struct mentioned in point 1. Each path is added to the dfsVisited stack to chart the path to each sign. This stack is emptied when a sign is found and its data displayed on screen as the path to the sign.
4. dfsToVisited - A to visit stack that holds paths with more than one choice of direction, to be checked after one node path has been traversed using a recursive Depth First Search function.
5. FloorMapVect – a two-dimensional vector array that holds the map and that has been read into the application. Vectors are designed to hold arrays of object types.

1.0.2 Depth First Search Algorithm

The algorithm used is the depth first search (DFS) algorithm:

1. Check current coordinate in path (node) and all other paths (branches).
2. If path has more than one node, such as a fork in the path, add path node to unvisited path to be able to check after current path is traversed. Each node is treated as a branch

3. Revisit all unvisited paths after current path is exhausted. [1]

In programming this can be viewed as B-Tree structure due to tree like nature of the nodes. At any given fork in the current path, one path is to be followed until exhausted of entries, then the other paths in the fork are followed until exhausted of entries also. The depth first search makes sure all entries are explored and followed to the extent of the maps edges. By placing all searched entries into a stack, this imitates a node structure:

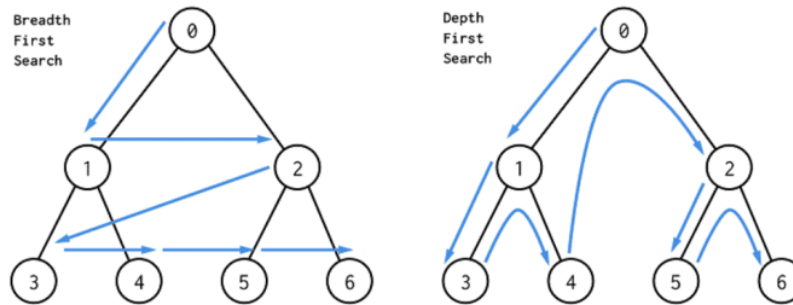


Fig 1.1 Node tree structure of searching [2]

Each path that is traversed is added to the visited stack, if there is a path that forks in the node/path, displayed in Fig 1.1 as node 1 and 2. Then node 1 is traversed and all its path exhausted, one completed node 2 is traversed until all paths exhausted.

2.0 Task 2

2.0.1 K- Means Method

The K-Means Clustering algorithm is an unsupervised learning algorithm that plots the points in a data source where the values are similar in clusters [3].

The user is required to choose the number of clusters required before running the algorithm. There are two methods to find number of clusters, The First is the 'Elbow For K Means Clustering'. This can be based on the data points presented within a line graph and finding the largest dip, the elbow, of the graph [4]. In fig 2.1 the main elbow can be charted at point 3 so there amount of clusters for this algorithm will be set as 3 clusters [5].



Fig 2.1 Elbow Graph

The second method is the Silhouette Method. This method entails measuring the distance between clusters and averaging out the number of datapoints from each cluster:

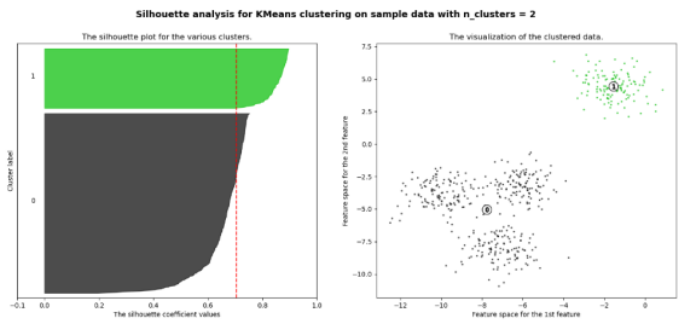


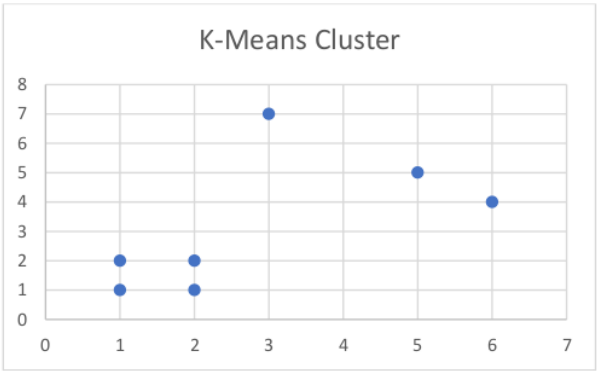
Fig 2.3 Silhouette Analysis [6]

Based on the below dataset the K-Means Clustering algorithm can be applied. The X and Y axis are measured to 8 in scale:

Data Point	X Co-Ordinate	Y Co-Ordinate
A1	5	5
A2	1	1
A3	1	2
A4	2	2
A5	2	1
A6	3	7
A7	6	4

Fig 2.2 Data set for Clustering

Placing the data within a graph:



The use of centroids is the random choice of data points within the dataset and calculating the distance between centroids chosen using Euclidian distance:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad [7]$$

The random centroid dataset points are chosen repeatedly until all data points have been checked. Each Euclidian distance result can be added together and the mean result derived where the

centroid can be worked out for each cluster. In this case 1,1 is the centroid of the first cluster and 5,5 is the centroid of the second.

2.0.2 K- Means Algorithm

The algorithm then runs as follows:

1. Generate Data for the application – an inputted dataset written into a two dimensional vector.
2. Define cluster level to analyse – random number or defined by the Silhouette method or Elbow method.
3. Define centroid of each cluster – the centre of each cluster gives an accurate distance from other datapoints and clusters.
4. Calculate distance of each centroid from surround datapoints – the desired answer can be concluded by this method. For example a survey is created to gather a simple true or false answer; the survey is split by age. Each age than answers true or false would sit on the graph in a simple two clustered dataset.
5. Each cluster can then be measured for its amount of data points surrounding it.

2.0.3 Criticism Of K-Means Clustering Algorithm

Criticism of K-Means Clustering Algorithm:

1. A human eye could work out the clustering amount needed quicker than the algorithm if the data was presented in a scatter graph. An improvement to this would be to pinpoint whether this type of algorithm is best used for a particular sized data source.
2. The algorithm must work out the placement of every data point in relation to other data points to successfully realise the entire dataset and conclude which centroid is the more beneficial to use. An improvement to the multiple stages of the process could be to set the cluster amount manually not to rely on Elbow or Silhouette methods.
3. The data clusters are analysed better by a shape, obviously highly scattered datasets are more difficult to cluster together. When data is closer together in large amounts and there are spaces between the clusters it is easier to define clusters and their data. An improvement to this could be to use a dataset with a large amount of data within it.

3.0 Task 3

The authors centroid data points allocated:

	Student Code	Centroid			
		1		2	
		x	y	x	y
14	1810705/1	1.80	1.51	4.13	2.79

Data source:

Label	X	Y
A	5	5
B	0	5
C	2	3
D	2	2
E	5	1
F	3	3
G	1	4
H	5	3
I	1	2
J	4	2
K	1.8	1.51
L	4.13	2.79

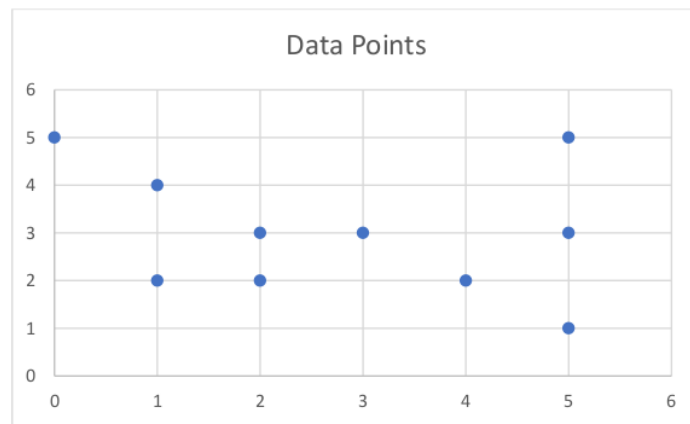


Fig 3.1 Data points displayed in graph

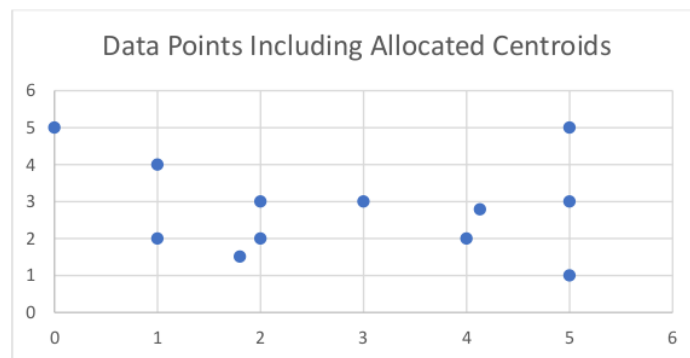


Fig 3.2 Data points displayed in graph with Author centroids

Centroid One

The distance from centroid one K where $X = 1.8$ and $Y = 1.59$.

Distance	X	Y
KA	3.2	3.49
KB	-1.8	3.49
KC	0.2	1.49
KD	0.2	0.49
KE	3.2	-0.51
KF	1.2	1.49
KG	-0.8	2.49
KH	3.2	1.49
KI	-0.8	0.49
KJ	2.2	0.49

Centroid One Averaged

Distance	x	y
KA	3	3
KB	-2	3
KC	0	1
KD	0	0
KE	3	-1
KF	1	1
KG	-1	2
KH	3	1
KI	-1	0
KJ	2	0

Centroid One Distance To Nearest Data point

If both X and Y are between -1 and 1 radius to the K centroid

Distance	x	y	Nearest to X	Nearest to y
KA	3	3	FALSE	FALSE
KB	-2	3	FALSE	FALSE
KC	0	1	TRUE	TRUE
KD	0	0	TRUE	TRUE
KE	3	-1	FALSE	TRUE
KF	1	1	TRUE	TRUE
KG	-1	2	TRUE	FALSE
KH	3	1	FALSE	TRUE
KI	-1	0	TRUE	TRUE
KJ	2	0	FALSE	TRUE

Centroid Two

The distance from centroid two L where X = 4.13. and Y = 2.79.

Distance	X	Y
LA	0.87	2.21
LB	-4.13	2.21
LC	-2.13	0.21
LD	-2.13	-0.79
LE	0.87	-1.79
LF	-1.13	0.21
LG	-3.13	1.21
LH	0.87	0.21
LI	-3.13	-0.79
LJ	-0.13	-0.79

Centroid Two Averaged

Distance	X	Y
LA	1	2
LB	-4	2
LC	-2	0
LD	-2	-1
LE	1	-2
LF	-1	0
LG	-3	1
LH	1	0
LI	-3	-1
LJ	0	-1

Centroid Two Distance To Nearest Data point

If both X and Y are between -1 and 1 radius to the L centroid

Distance	X	Y	Nearest to X	Nearest to y
LA	1	2	TRUE	FALSE
LB	-4	2	FALSE	FALSE
LC	-2	0	FALSE	TRUE
LD	-2	-1	FALSE	TRUE
LE	1	-2	TRUE	FALSE
LF	-1	0	TRUE	TRUE
LG	-3	1	FALSE	TRUE
LH	1	0	TRUE	TRUE
LI	-3	-1	FALSE	TRUE
LJ	0	-1	TRUE	TRUE

True Centroid Calculation for All Datapoint on X For Cluster One

Each datapoint is calculated as being proposed as the centre of the first cluster

Label	X										Total
		AB	AC	AD	AE	AF	AG	AH	AI	AJ	
A	5	5	3	3	0	2	4	0	4	1	4
		BA	BC	BD	BE	BF	BG	BH	BI	BJ	
B	0	-5	-2	-2	-5	-3	-1	-5	-1	-4	2
		CA	CB	CD	CE	CF	CG	CH	CI	CJ	
C	2	2	2	0	-3	-1	1	-3	1	-2	4
		DA	DB	DC	DE	DF	DG	DH	DI	DJ	
D	2	-3	2	0	-3	-1	1	-3	1	-2	4
		EA	EB	EC	ED	EF	EG	EH	EI	EJ	
E	5	0	5	3	3	2	4	0	4	1	2
		FA	FB	FC	FD	FE	FG	FH	FI	FJ	
F	3	-2	3	1	1	-2	2	-2	2	-1	3
		GA	GB	GC	GD	GE	GF	GH	GI	GJ	
G	1	-4	1	-1	-1	-4	-2	-4	0	-3	4
		HA	HB	HC	HD	HE	HF	HG	HI	HJ	
H	5	0	5	3	3	0	2	4	4	1	3
		IA	IB	IC	ID	IE	IF	IG	IH	IJ	
I	1	-4	1	-1	-1	-1	-4	0	-4	-3	5
		JA	JB	JC	JD	JE	JF	JG	JH	JI	
J	4	-1	4	2	2	-1	1	0	-1	3	4

True Centroid Calculation for All Datapoint on Y

Label	Y										Total
		AB	AC	AD	AE	AF	AG	AH	AI	AJ	
A	5	0	2	3	4	2	1	2	3	3	2
		BA	BC	BD	BE	BF	BG	BH	BI	BJ	
B	5	0	2	3	4	2	1	2	3	3	2
		CA	CB	CD	CE	CF	CG	CH	CI	CJ	
C	3	-2	-2	1	2	0	-1	0	1	1	5
		DA	DB	DC	DE	DF	DG	DH	DI	DJ	
D	2	-3	-3	-1	1	-1	-2	-1	0	0	6
		EA	EB	EC	ED	EF	EG	EH	EI	EJ	
E	1	-4	-4	-2	-1	-2	-3	-2	-1	-1	3
		FA	FB	FC	FD	FE	FG	FH	FI	FJ	
F	3	-2	-2	0	1	2	-1	0	1	1	5
		GA	GB	GC	GD	GE	GF	GH	GI	GJ	
G	4	-1	-1	1	2	3	1	1	2	2	5
		HA	HB	HC	HD	HE	HF	HG	HI	HJ	
H	3	-2	-2	0	1	2	0	-1	1	1	4
		IA	IB	IC	ID	IE	IF	IG	IH	IJ	
I	2	-3	-3	-1	-1	0	1	-2	-1	0	5
		JA	JB	JC	JD	JE	JF	JG	JH	JI	
J	2	-3	-3	-1	0	1	-1	-2	-1	0	5

True Centroid Calculation for All Datapoint on X For Cluster Two

Each datapoint is calculated as being proposed as the centre of the second cluster.

Label	X										Total
		AB	AC	AD	AE	AF	AG	AH	AI	AJ	
A	5	5	3	3	0	2	4	0	4	1	5
		BA	BC	BD	BE	BF	BG	BH	BI	BJ	
B	0	-5	-2	-2	-5	-3	-1	-5	-1	-4	4
		CA	CB	CD	CE	CF	CG	CH	CI	CJ	
C	2	2	2	0	-3	-1	1	-3	1	-2	2
		DA	DB	DC	DE	DF	DG	DH	DI	DJ	
D	2	-3	2	0	-3	-1	1	-3	1	-2	3
		EA	EB	EC	ED	EF	EG	EH	EI	EJ	
E	5	0	5	3	3	2	4	0	4	1	5
		FA	FB	FC	FD	FE	FG	FH	FI	FJ	
F	3	-2	3	1	1	-2	2	-2	2	-1	3
		GA	GB	GC	GD	GE	GF	GH	GI	GJ	
G	1	-4	1	-1	-1	-4	-2	-4	0	-3	4
		HA	HB	HC	HD	HE	HF	HG	HI	HJ	
H	5	0	5	3	3	0	2	4	4	1	5
		IA	IB	IC	ID	IE	IF	IG	IH	IJ	
I	1	-4	1	-1	-1	-1	-4	0	-4	-3	3
		JA	JB	JC	JD	JE	JF	JG	JH	JI	
J	4	-1	4	2	2	-1	1	0	-1	3	2

True Centroid Calculation for All Datapoint on Y For Cluster Two

Label	Y										Total
		AB	AC	AD	AE	AF	AG	AH	AI	AJ	
A	5	0	2	3	4	2	1	2	3	3	4
		BA	BC	BD	BE	BF	BG	BH	BI	BJ	
B	5	0	2	3	4	2	1	2	3	3	4
		CA	CB	CD	CE	CF	CG	CH	CI	CJ	
C	3	-2	-2	1	2	0	-1	0	1	1	5
		DA	DB	DC	DE	DF	DG	DH	DI	DJ	
D	2	-3	-3	-1	1	-1	-2	-1	0	0	2
		EA	EB	EC	ED	EF	EG	EH	EI	EJ	
E	1	-4	-4	-2	-1	-2	-3	-2	-1	-1	2
		FA	FB	FC	FD	FE	FG	FH	FI	FJ	
F	3	-2	-2	0	1	2	-1	0	1	1	0
		GA	GB	GC	GD	GE	GF	GH	GI	GJ	
G	4	-1	-1	1	2	3	1	1	2	2	1
		HA	HB	HC	HD	HE	HF	HG	HI	HJ	
H	3	-2	-2	0	1	2	0	-1	1	1	0
		IA	IB	IC	ID	IE	IF	IG	IH	IJ	
I	2	-3	-3	-1	-1	0	1	-2	-1	0	2
		JA	JB	JC	JD	JE	JF	JG	JH	JI	
J	2	-3	-3	-1	0	1	-1	-2	-1	0	2

References

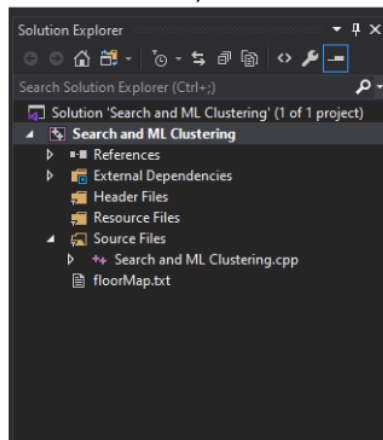
- [1] G. Dickers, "Depth First Search and Transversal," Trinity College, Swansea, 2021.
- [2] D. Zaltsman, "Difference between Depth First Search and Breadth First Search," Dev, 1 03 2020. [Online]. Available: <https://dev.to/danimal92/difference-between-depth-first-search-and-breadth-first-search-6om>. [Accessed 14 12 2021].
- [3] G. Dickers, "K Means Clustering," Trinity College, Swansea, 2021.
- [4] T. s.-y. developers, "Elbow Method," The scikit-yb developers, 2019. [Online]. Available: <https://www.scikit-yb.org/en/latest/api/cluster/elbow.html>. [Accessed 11 12 2021].
- [5] G. Vickers, "Selecting K in K Means," Trinity College, Swansea, 2021.
- [6] s.-l. developers, "Selecting the number of clusters with silhouette analysis on KMeans clustering," scikit-learn developers, 2021. [Online]. Available: https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html. [Accessed 11 12 2021].
- [7] Sunaina, "Steps to calculate centroids in cluster using K-means clustering algorithm," TechTarget, Inc, 07 03 2018. [Online]. Available:

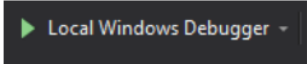
<https://www.datasciencecentral.com/profiles/blogs/steps-to-calculate-centroids-in-cluster-using-k-means-clustering>. [Accessed 10 12 2021].

Appendix

1.0.1 User Guide

1. Open 'Search and ML Clustering.sln' in Visual Studio 19 and over
2. Use the floorMap.txt as the map for the application or alternatively delete floorMap.txt content and insert your own.



3. Press the local windows debugger button to run the program:

4. Application runs within command prompt.
5. The floorMap.txt get analysed and the first Depth First Search is applied.
6. With any forks in the path they are added to the ToVisit Stack object
7. The second Depth First Search check runs through the ToVisit Stack.

1.0.2 Test Schedule

```

First Depth First Search
S Q Q Q Q Q Q Q A *
* * * * * Q * * * * *
* Q * Q Q Q Q Q Q
* Q * * * * * Q *
* Q * Q Q Q Q Q Q
* Q * * * * * Q
* Q B * Q Q * D *
* Q * * * * * Q *
* Q * Q Q Q Q Q *
* Q * * * * * Q
* Q * * * * * Q Q
* Q * * * * * Q
* Q Q Q Q Q Q Q Q Q Q Q
* * * * * Q * * * * *
Second Depth First Search Visiting Second Path.
S Q Q Q Q Q Q Q Q Q Q Q *
* * * * * Q * * * * *
* Q * Q Q Q Q Q Q Q Q Q *
* Q * Q Q Q Q Q Q Q Q Q *
* Q * * * * * Q * * * * *
* Q Q Q Q Q * Q Q *
* Q * * * * * Q * * * * *
* Q * Q Q Q Q Q Q Q * Q *
* Q * * * * * Q * * * * *
* Q * * * * * Q Q Q Q Q *
* Q * * * * * Q * * * * *
* Q * * * * * Q
* Q Q Q Q Q Q Q Q Q Q Q
D Sign Placed At Map Coordinates (7,14)
A Sign Placed At Map Coordinates (1,10)
B Sign Placed At Map Coordinates (7,5)
C Sign Placed At Map Coordinates (3,1)
E Sign Placed At Map Coordinates (14,1)
F Sign Placed At Map Coordinates (14,14)

```

1.0.3 Program Listing

```
2 // Search and ML Clustering.cpp : This file contains the 'main' function.  
3 Program execution begins and ends there.  
4  
5 #include <iostream>  
6 #include <vector>  
7 #include <fstream>  
8 #include <string>  
9 #include <stack>  
10 #include <ctime>  
11 #include <cstdlib>  
12 using namespace std;  
13  
14 //Function declaration  
15 bool floorMapRead();  
16 void dfs(int ro, int dep);  
17 void emptyStack();  
18 void emptytoVisitStack();  
19 void outputMap();  
20 void checkSign();  
21 void checkAlpha(int x, int y);
```

```

21 void toVisit();
22 void currentMap();
23
24 //struct to hold map visited points
25 typedef struct {
26     string letter;
27     int x;
28     int y;
29 } dStruct;
30 //Stack to hold positive map entries
31 std::stack<dStruct> dfsVisited;
32 std::stack<dStruct> dfsToVisit;
33 std::stack<dStruct> signStack;
34 //Read map from file
35 std::ifstream Map("floorMap.txt");
36 //vector to hold map data
37 std::vector<vector<char>> floorMapVect;
38 //String to hold line to convert to char
39 std::string lineText;
40 //Map alphabet array comparison
41 char alphabet[26] = {
    'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','r','s','t','u',
    'v','w','x','y','z' };
42 //Initialize struct coordinate object
43 dStruct viz;
44 dStruct toViz;
45 int direction = 1;
46 int row;
47 int depth;
48
49 int main()
50 {
51     //Read Map from file
52     floorMapRead();
53     //Perform Depth First Search
54     dfs(1,0);
55     //Output Searched Map
56     cout << "First Depth First Search" << endl;
57     outputMap();
58     //Recursive DFS
59     cout << "Second Depth First Search Visiting Unvisited Paths." << endl;
60     emptytoVisitStack();
61     //Check for sign on map
62     checkSign();
63     //Output Searched Map
64     outputMap();
65     //Empty Stack Containing Coordinates Of Signs
66     emptyStack();
67     // Close the file
68     Map.close();
69 }
70
71 void emptyStack()
72 {
73     //empties stack
74     while (!signStack.empty()) {
75         viz = signStack.top();
76         cout << viz.letter << " Sign Placed At Map Coordinates (" << viz.x <<
77         ", " << viz.y << ")" << endl;
78         signStack.pop();
79     }
80 }

```

```

80
81 void emptytoVisitStack()
82 {
83     //empties stack
84     while (!dfsToVisit.empty()) {
85         viz = dfsToVisit.top();
86         dfsToVisit.pop();
87         dfs(viz.x, viz.y);
88     }
89 }
90
91 bool floorMapRead()
92 {
93
94     //Check file is valid
95     if (!Map)
96     {
97         std::cerr << "Cannot open the File" << std::endl;
98         return false;
99     }
100     //Read floorMap into string line by line
101     int count = 0;
102     while (getline(Map, lineText)) {
103         //Read file into Vector
104         std::vector<char> fileLine;
105
106         for (char& c : lineText) {
107             if (c != ',') {
108                 fileLine.push_back(c);
109             }
110         }
111
112         floorMapVect.push_back(fileLine);
113     }
114 }
115
116 void outputMap()
117 {
118     for (int i = 0; i < sizeof(floorMapVect); i++)
119     {
120         for (int j = 0; j < sizeof(floorMapVect[0]); j++)
121         {
122             cout << floorMapVect[i][j] << " ";
123         }
124         cout << endl;
125     }
126 }
127
128 void checkSign()
129 {
130     while (!dfsVisited.empty())
131     {
132         viz = dfsVisited.top();
133         dfsVisited.pop();
134         //Check For Sign
135         for (int p = 0; p < sizeof(alphabet); p++)
136         {
137             if (tolower(floorMapVect[viz.x][viz.y]) == tolower(alphabet[p]))
138             {
139                 viz.letter = toupper(alphabet[p]);
140                 signStack.push(viz);
141             }

```

```

142     }
143 }
144 }
145
146 void dfs(int ro, int dep)
147 {
148     row = ro;
149     depth = dep;
150     int direction = 1;
151     int count = 0;
152
153     while (count != 1000)
154     {
155
156         switch (direction)
157         {
158             //go forward
159             case 1:
160                 depth++;
161                 if (depth > 0 &&
162                     depth < sizeof(floorMapVect[0]) &&
163                     floorMapVect[row][depth] != '*')
164                 {
165                     checkAlpha(row, depth);
166                     floorMapVect[row][depth] = 'Q';
167                     //cout << "Count: " << count << " " << row << " " << depth
168                     << " Forward" << endl;
169                     viz.x = row;
170                     viz.y = depth;
171                     dfsVisited.push(viz);
172                     //Check right
173                     if (floorMapVect[row + 1][depth] == ' ')
174                     {
175                         direction = 2;
176                     }
177                 }
178             else
179             {
180                 direction = 3;
181             }
182             break;
183             //go right
184             case 2:
185                 row++;
186                 if (row > 0 &&
187                     row < sizeof(floorMapVect) &&
188                     floorMapVect[row][depth] != '*')
189                 {
190                     checkAlpha(row, depth);
191                     floorMapVect[row][depth] = 'Q';
192                     //cout << "Count: " << count << " " << row << " " << depth
193                     << " Right" << endl;
194                     viz.x = row;
195                     viz.y = depth;
196                     dfsVisited.push(viz);
197                     if (floorMapVect[row+1][depth] == ' ')
198                     {
199                         direction = 1;
200                         row++;
201                     }
202                 }
203             }
204         }
205     }
206 }

```

```

202         else
203         {
204             direction = 1;
205             row--;
206         }
207         ;
208         break;
209         //go backward
210     case 3:
211         depth--;
212         if (depth > 0 &&
213             depth < sizeof(floorMapVect[0]) &&
214             floorMapVect[row][depth] != '*')
215         {
216             checkAlpha(row, depth);
217             floorMapVect[row][depth] = 'Q';
218             //cout << "Count: " << count << " " << row << " " << depth
<< " Backward" << endl;
219             viz.x = row;
220             viz.y = depth;
221             dfsVisited.push(viz);
222             //check right
223             if (floorMapVect[row + 1][depth] == ' ')
224             {
225                 direction = 2;
226             }
227         }
228         else
229         {
230             direction = 4;
231             depth++;
232         }
233         ;
234         break;
235     case 4:
236         row--;
237         if (row > 0 &&
238             row < sizeof(floorMapVect) &&
239             floorMapVect[row][depth] != '*')
240         {
241             checkAlpha(row, depth);
242             floorMapVect[row][depth] = 'Q';
243             //cout << "Count: " << count << " " << row << " " << depth
<< " Left" << endl;
244             viz.x = row;
245             viz.y = depth;
246             dfsVisited.push(viz);
247             //left
248             if (floorMapVect[row - 1][depth] == ' ')
249             {
250                 direction = 1;
251                 row--;
252             }
253         }
254         ;
255         break;
256     }
257     count++;
258     toVisit();
259 }
260 }
261

```

```

262 void checkAlpha(int x, int y)
263 {
264     if (isalpha(floorMapVect[x][y]))
265     {
266         for (int a = 0; a < sizeof(alphabet); a++)
267         {
268             if (tolower(floorMapVect[x][y]) == tolower(alphabet[a]))
269             {
270                 viz.letter = floorMapVect[x][y];
271                 viz.x = x;
272                 viz.y = y;
273                 signStack.push(viz);
274             }
275         }
276     }
277 }
278
279 void toVisit()
280 {
281     int x = 0, y = 0;
282     int paths = 0;
283     while (!dfsVisited.empty())
284     {
285         toViz = dfsVisited.top();
286         dfsVisited.pop();
287         x = toViz.x;
288         y = toViz.y;
289
290         //check left
291         if (x > 1 && x < sizeof(floorMapVect))
292         {
293             if (floorMapVect[x - 1][y] != 'Q' && floorMapVect[x - 1][y] !=
294                 '*)')
295             {
296                 paths++;
297                 if (paths > 0)
298                 {
299                     toViz.x = x - 1;
300                     toViz.y = y;
301                     dfsToVisit.push(toViz);
302                 }
303             }
304         }
305
306         //check right
307         if (x > 1 && x < sizeof(floorMapVect))
308         {
309             if (floorMapVect[x + 1][y] != 'Q' && floorMapVect[x + 1][y] !=
310                 '*)')
311             {
312                 paths++;
313                 if (paths > 0)
314                 {
315                     toViz.x = x + 1;
316                     toViz.y = y;
317                     dfsToVisit.push(toViz);
318                 }
319             }
320         }
321     }

```

```

322
323         //check forward
324
325         if (y > 1 && y < sizeof(floorMapVect[0]))
326         {
327             if (floorMapVect[x][y + 1] != 'Q' && floorMapVect[x][y + 1] !=
328                 'x')
329             {
330                 paths++;
331                 if (paths > 0)
332                 {
333                     toViz.x = x;
334                     toViz.y = y + 1;
335                     dfsToVisit.push(toViz);
336                 }
337             }
338
339             //check back
340             if (y > 1 && y < sizeof(floorMapVect[0]))
341             {
342                 if (floorMapVect[x][y - 1] != 'Q' && floorMapVect[x][y - 1] !=
343                     'x')
344                 {
345                     paths++;
346                     if (paths > 0)
347                     {
348                         toViz.x = x;
349                         toViz.y = y - 1;
350                         dfsToVisit.push(toViz);
351                     }
352                 }
353             }
354         }
355
356         void currentMap()
357         {
358             cout << "[" << row << "]"[" << depth << "]" << endl;
359             cout << floorMapVect[row][depth] << endl;
360         }

```

1810705 - Search and ML Clustering

GRADEMARK REPORT

FINAL GRADE

45 / 100

GENERAL COMMENTS

Instructor

A basic report submitted.

Task 1:

Code builds but crashes on my machine. No video provided- checked email.

Screen shot provided.

Essential dfs in evidence there.

Task 2.

Clustering is used not just in visualisation, its used elsewhere in inference. That is given a new point, it answers the question "what cluster does it belong to"?

Distance calc is incorrectly described in your report- the ref needs checking for correctness. Should be $\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$. Use only reliable references.

Include stand texts and papers - eg Norvig and Russell.

Task 3:

Attempted, but some errors here in calcs and method.

PAGE 1

PAGE 2



Comment 1

Not necessarily



Comment 2

no - its $\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$



Comment 3

Yes but its used elsewhere to for example inference. Given a new point, what cluster does it belong to?

GRADING FORM: SMU GENERIC_GJ

KRISTIAN COLE

0

RESEARCH-INFORMED LITERATURE

KNOWLEDGE AND UNDERSTANDING

ANALYSIS

PRACTICAL APPLICATION AND DEP.

SKILLS FOR PROFESSIONAL PRAC.

