MITU
الكلية التكنولوجية بالقاهرة
Technological Faculty in Cairo

MITU
MISR INTERNATIONAL TECHNOLOGICAL UNIVERSITY

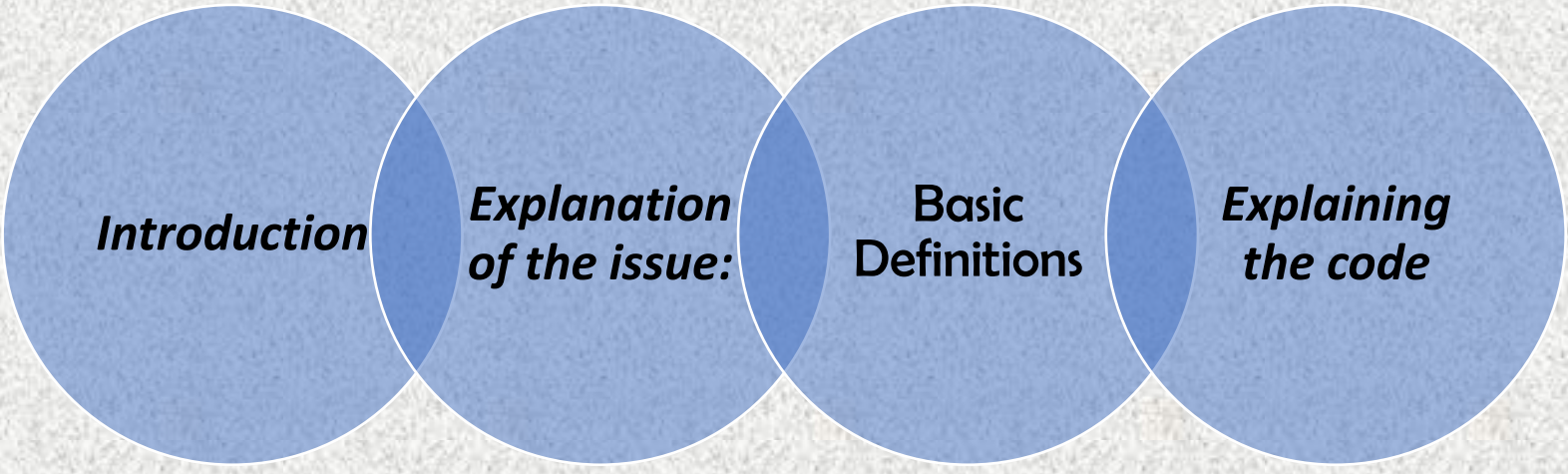
RADIO LABELING FOR RADIO STATIONS

SUPERVISOR: DR. ELSAYED BADR

BY: AMR BELAL

DES-2023

Agenda



Introduction:

Radio labeling is a critical concept in graph theory with applications in network optimization and communication systems. The goal is to assign labels to vertices in a graph such that each label reflects the distance to the farthest vertex in the graph. In this report, we explore the implementation of the Laxman Saha algorithm for radio labeling on three distinct types of graphs: Path, Cycle, and Complete Graphs.

The Laxman Saha algorithm offers an effective approach to determine optimal radio station locations by iteratively calculating distance values based on the graph's topology. This report delves into the structure and functionality of the Python code, highlighting the key components and procedures involved in generating radio labels for different graph configurations.

Through this analysis, we aim to provide a comprehensive understanding of the Laxman Saha algorithm, its application in radio labeling, and the insights gained from examining the adjacency matrix, distance matrix, cost matrix, and the final radio station location. The report concludes with an evaluation of the algorithm's efficiency, as measured by its runtime, shedding light on its practical implications in real-world scenarios.

Explanation of the issue:

Wireless communication is the transfer of information over varying distances without the use of conductors Electricity and wiring. The distances over which data is transferred vary and may be short Exceeding a few meters, such as a television remote control (or large distances) up to thousands or more Millions of kilometers away, such as radio waves.

It is an unfortunate thing to talk on the phone and find someone else on the same line. This The disturbance is caused by interference for instantaneous communications (occurring at the same time). 5. Channels near Each other may destroy the connection. We can avoid these interferences that occur between channels Assigning different (and sufficiently distant from each other) distinct frequencies for each channel.

Hale formulated this problem in the form of a problem of coloring the vertices of graphics, which was launched It has coloring-(2,1) L. We assume that we have a group of “transmitting” stations. It must be set A “frequency” station for each transmitter so that interference is as little as possible. It can be apparent The interference is so strong that if we have two different channels working at the same time, the interference will disappear This is due to the proximity of the two channels to each other. In order to reduce this interference, we must: The two close channels should be different to the maximum extent.

Assuming we have a set of correspondences, we will assign each station a channel (a positive number) In order to avoid interference between them. The smaller the distance between two stations, the greater the interference between them, that is, the greater The greater the distance between two stations, the less interference between them, and therefore the main goal is to be the distance between any two stations is as large as possible (the question is to what extent this difference is) and the answer is until the interference disappears, and the question is also whether there is a condition (restriction) that prevents this interference. To answer this Question, We can formulate the above problem into a graphics theory problem. Each vertex of the drawing It represents a station, and any pair of vertices associated with an edge corresponds to two adjacent stations.

Basic Definitions

Definition of a graph

A *graph G* comprises a set *V* of vertices and a set *E* of edges

Each *edge* in *E* is a pair (*a*,*b*) of vertices in *V*

If (*a*,*b*) is an edge in *E*, we connect *a* and *b* in the *graph drawing* of *G*

Size and order

The *order* of *G* is the number *n* of vertices in *V*

The *size* of *G* is the number *L* of edges in *E*

Minimum possible order is 0 (*empty* graph)

Maximum possible order is *n*(n-1)/2 (*complete* graph)

Adjacency matrix for a graph

The adjacency matrix $x = [x_{ab}]$ for G is a matrix with n rows and n columns and entries given by:

$x_{ab} = 1$ if (a,b) is an edge in G

0 otherwise

Degrees and degree sequence

The degree d_a of vertex a is the number of vertices to which a is linked by an edge

The minimum possible degree is 0

The maximum possible degree is n-1

The degree sequence for a graph is the vector $(d_1, d_2, \ldots, d_n)$

Subgraphs

A subgraph of G=G(V,E) is a subset W of the vertex set V together with all of the edges that connect pairs of vertices in W

E.g. if W={4,5,6,7}, the subgraph of

- ✓ If W=V then H is called

  spanning subgraph

- ✓ If W V and H maintains

  G's relations then H is called

  Induced subgraph

- ✓ Otherwise H is called A subgraph of G
- ✓ If H has not isolated vertices then

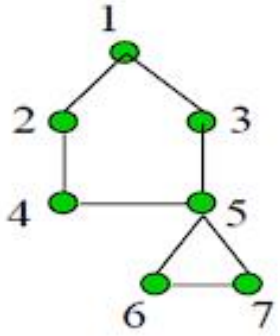
  it is called a drived subgraph

Paths

A path from vertex a to vertex b is an ordered sequence

$a=v_0, v_1, \ldots, v_m=b$

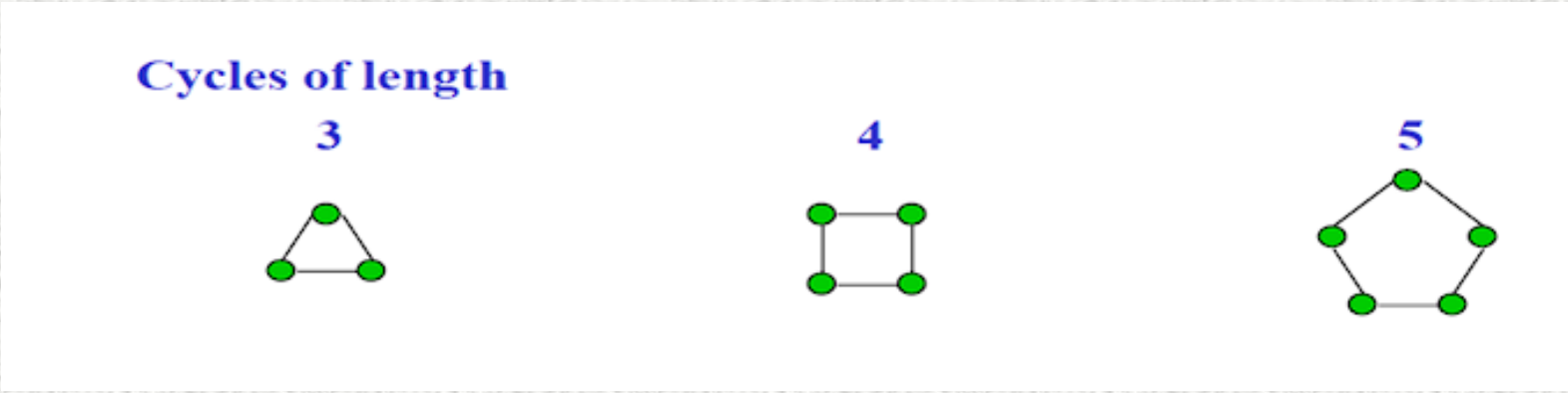
of distinct vertices in which each adjacent pair (v_{j-1},v_j) is linked by an edge. The length of the path is m



a path of length 1 from 1 to 2
a path of length 2 from 1 to 4
a path of length 3 from 1 to 4
a path of length 3 from 1 to 6

Cycles

A cycle is an ordered sequence $a=v_0, v_1, \ldots, v_m=a$ of vertices in which each adjacent pair (v_{j-1},v_j) of vertices is linked by an edge, and $v_0, v_1, \ldots, v_{m-1}$ are distinct . The length of the cycle is m



Explaining the code

1. Function `Adjacency (status, number_of_vertices):

This function creates and returns the adjacency matrix (`A`) based on the specified graph type (`status_of_graph`) and the number of vertices (`number_of_ vertices `). The function allows you to create an adjacency matrix based on the specified type of graph.

- If the graph is open (`status_of_graph == 1`): The relationship between vertices is such that vertex `i` is connected to vertices `i-1` and `i+1` (with special cases for the first and last vertices).

- If the graph is a cycle (`status_of_graph == 2`): A cycle is created where vertex `i` is connected to vertices `i-1` and `i+1` (with special cases for the first and last vertices). - If the graph is complete (`status == 3`): All vertices are connected to each other.

2. Function distance(status_of_graph, number_of_ vertices)

This function calculates the distance matrix (`D`) and determines the diameter of the graph (`diam`). The calculations depend on the type of graph:

- If the graph is open (`status_of_graph == 1`): The distance between vertices is calculated using the absolute difference between their positions, and the diameter is determined as the maximum value in the row.

- If the graph is a cycle (`status_of_graph == 2`): The distance is calculated using the rule specified in the code, and the diameter is determined as the maximum value in the row.

- If the graph is complete (`status_of_graph == 3`): The distance is the same as the distance matrix, and the diameter is determined as the maximum value in the row.

3. Function C_0(diameter, Distance)

This function creates and returns the matrix `C_0` using the graph diameter (`diameter`) and the distance matrix (`D`). This matrix is used in the next step.

4. Function calculate_X_and_FIndex(C0, number_of_vertices)

This function implements the Laxman Saha algorithm to calculate the matrices `X` and `F_Index`. `C0` is used to update values in each step, and the minimum values in each step are calculated and updated in `X`. The selected vertex index is also updated in `F_Index`.

5. Function radio_station(X)

This function calculates and returns the index of the radio station (`index_radio`) using the highest value in the `X` matrix.

Finally, the program prints the adjacency matrix, distance matrix, `C_0` matrix, `X` matrix, `F_Index` list, and the radio station index. The user is prompted to input the number of vertices (`number_of_ vertices `) and the type of graph (`status`). The program also displays the total runtime of the code.

In conclusion, the code effectively implements the Laxman Saha algorithm for radio labeling, providing an organized and labeled structure for radio stations in each graph. Further testing with various graphs and additional optimizations could be explored for a more comprehensive analysis.

Comparison among Ub0, Ub1 and Ub3 for the radio number of path and cycle graph (from X1 →X100):

m(Pn)		m(Cn)	
N	value	N	value
1	0	1	0
2	105	2	1269
3	206	3	1215
4	319	4	1161
5	420	5	1107
6	541	6	1053
7	642	7	99
8	771	8	945
9	872	9	891
10	1009	10	837
11	1110	11	783
12	1255	12	729
13	1356	13	675
14	1509	14	621
15	1610	15	567
16	1771	16	513
17	1872	17	459
18	2041	18	405
19	2142	19	351
20	2319	20	297
21	2420	21	243
22	2605	22	189
23	2706	23	135
24	2899	24	81
25	3000	25	27
26	3009	26	1296
28	3370	28	1188
29	3312	29	1134
30	2511	30	1080
31	3430	31	1026
32	2229	32	972
33	3556	33	918
34	1955	34	864
35	3690	35	810
36	1689	36	756
37	3832	37	702
38	1431	38	648
39	3982	39	594
40	1181	40	540
41	4140	41	486
42	939	42	432
43	4306	43	378
44	705	44	324
45	4480	45	270
46	479	46	216
47	4662	47	162
48	261	48	180
49	4852	49	54
50	51	50	1323
51	4950	51	1
52	155	52	1270
53	4756	53	1216
54	469	54	1162
55	4570	55	1108
56	591	56	1054
57	4392	57	1000
58	812	58	946
59	4222	59	872
60	1059	60	828
61	4060	61	784
62	1305	62	730
63	3906	63	622
64	1559	64	568
65	3760	65	514
66	1821	66	460
67	3622	67	406
68	2091	68	352
69	3492	69	298
70	2369	70	244
71	3370	71	190
72	2655	72	136
73	3256	73	82
74	2949	74	28
75	3150	75	1297
76	3049	76	1243
77	2852	77	1189
78	2751	78	1135
79	2562	79	1081
80	2461	80	1027
81	2280	81	973
82	2179	82	919
83	2006	83	919
84	1905	84	865
85	1740	85	811
86	1639	86	757
87	1482	87	703
88	1381	88	649
89	1232	89	595
90	1131	90	541
91	990	91	487
92	889	92	433
93	756	93	379
94	655	94	325
95	530	95	271
96	429	96	217
97	312	97	136
98	211	98	109
99	102	99	55
100	1	100	1324