GRAPH LABELING USING ALGORITHMIC APPROACH

Design & Analysis of Algorithms
(COMP-SCI 5592)

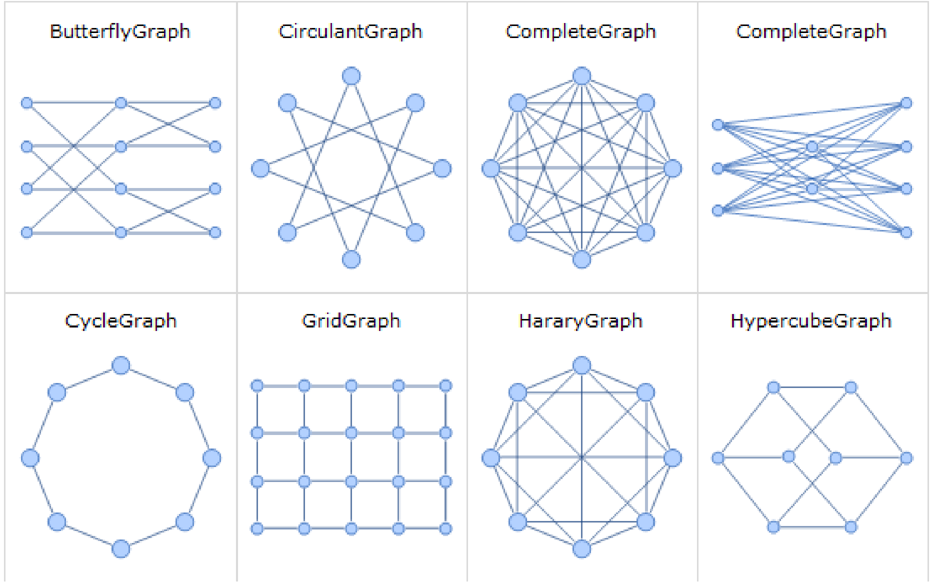
Dr. Ahsan Asim

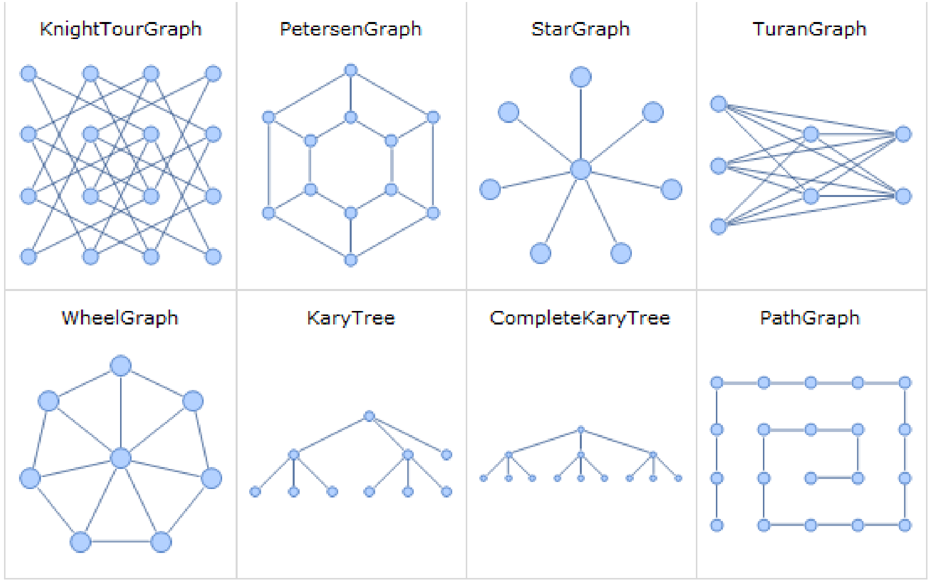
Graph Concepts

Definition:

- A graph G = (V , E) consists of finite sets of vertices and edges.
- The cardinality of vertex set V(G) and edge set E(G) of a graph is called the order, and size of the graph respectively.
- In graphs degree or valency of any vertex is the number of incident edges, it is denoted as deg(v).
- Studying different types of graphs based on symmetries. (families of graphs)
- Devising new algorithms using different design strategies.

Types of Graphs or Graph Families





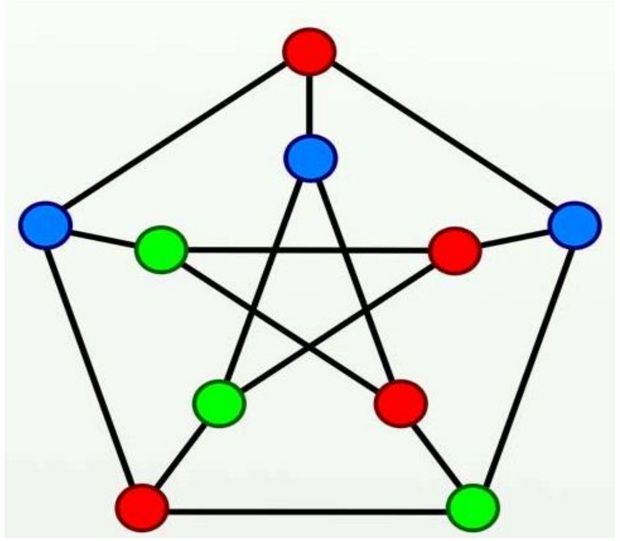
Graph Labeling

An assignment of integers to the vertex set or edge set, or both.

Three common characteristics:

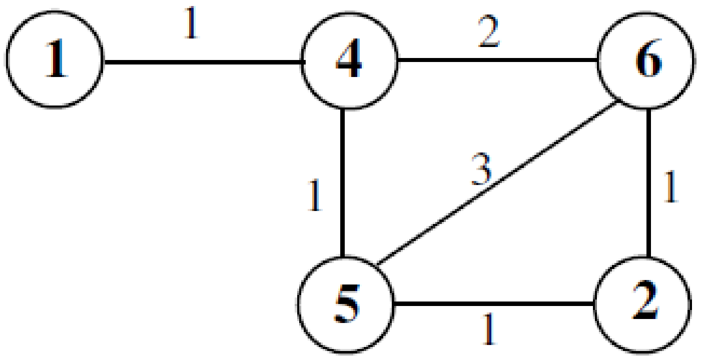
- A set of integers as labels
- A rule that assigns the labels
- Some condition(s) that these labels must satisfy.

Graph coloring is a special type of graph labeling, it is an assignment of colors to elements of a graph. If no two adjacent vertices are of the same color this is called a vertex coloring. Similarly, an edge coloring assigns a color to edges.



Edge Labeling

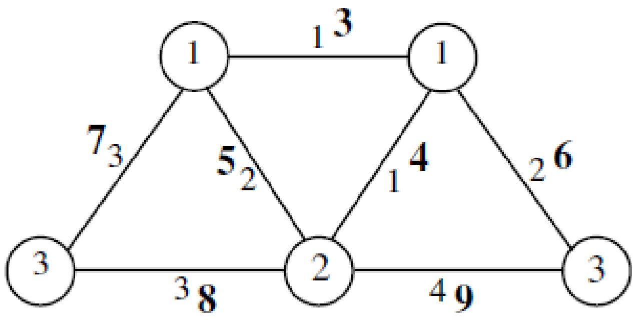
In 1988 **Chartrand** introduced edge *k*-labeling of a graph such that $\boldsymbol{w_{\varphi}(x) \neq w_{\varphi}(y)}$ *for* all vertices $\boldsymbol{x, y \in V(G)}$. Such labeling was called irregularity strength $\boldsymbol{s(G)}$.



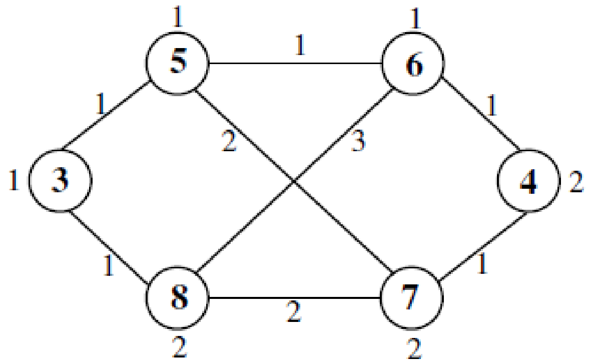
G. Chartrand, M.S. Jacobson, J. Lehel, O.R. Oellermann, S. Ruiz and F. Saba, Irregular networks, Congr. Numer. 64 (1988), 187–192.

Total k-Labeling : Edge Irregular & Vertex Irregular

- In 2007, **Baca** introduced two types of total k-labelings:
  - ➢ Edge irregular total k-labeling tes(G)
  - ➢ Vertex irregular total k-labeling tvs(G)



Example for tes(G)



Example for tvs(G)

Baˇca, M., Jendroˇl , S., Miller, M. and Ryan, J., On irregular total labelings, Discrete Math. 307 (2007), 1378–1388.

Vertex k-labeling or Edge Irregularity Strength: es(G)

In 2014 **Ahmad** introduced vertex *k*-labeling

$$\phi : V(G) \to \{1, 2, \ldots\ldots, k\}$$

- For every two different edges *e* and *f*

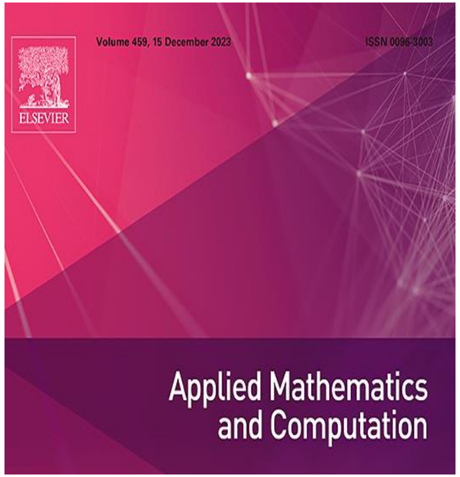
$$w_\phi(e) \neq w_\phi(f)$$

- Edge weights:

$$w_\phi(xy) = \phi(x) + \phi(y)$$

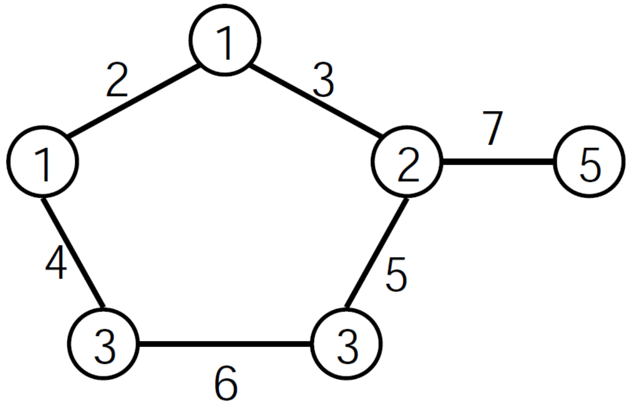
- Minimum ***k*** is called the edge irregularity strength ***es(G)***.

$$es(G) \geq \max\left\{\left\lceil \frac{|E(G)|+1}{2} \right\rceil, \Delta(G)\right\}$$

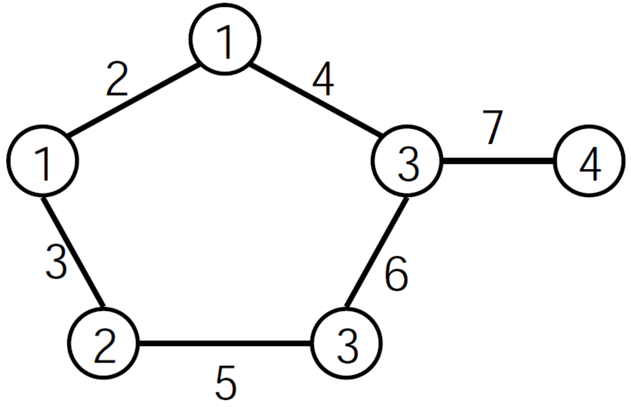

Volume 459, 15 December 2023
ISSN 0096-3003
ELSEVIER
Applied Mathematics and Computation

Ahmad, O. Al-Mushayt, M. Baca, On edge irregularity strength of graphs, *Appl. Math. Comput.* **243**(2014), 607–610

Edge Irregularity Strength: es(G) or Vertex k-labeling



V(G) = 6
E(G) = 6
Labels = {1,1,2,3,3,**5**}
Max Edge Weight = 7



V(G) = 6
E(G) = 6
Labels = {1,1,2,3,3,**4**}
Max Edge Weight = 7

Algorithmic Solutions for Graph Labeling

Algorithms can solve many problems, where other mathematical solutions are very complex / impossible.

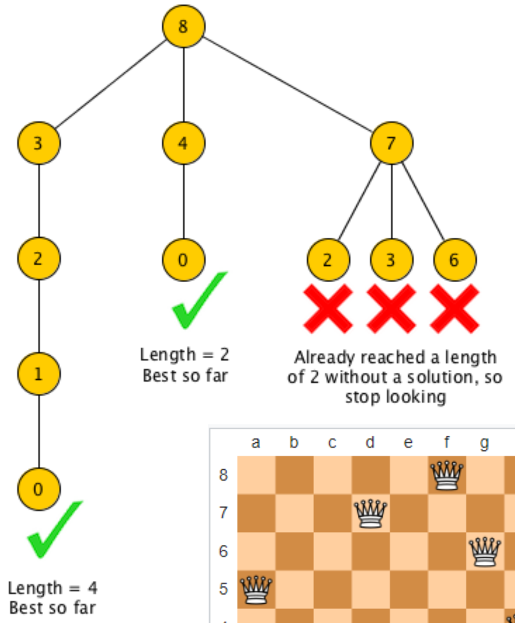
Computations can tackle the exhaustive issues of numerical calculations by providing comprehensive results.

Algorithmic Design Strategies (Helpful in Assig-2)

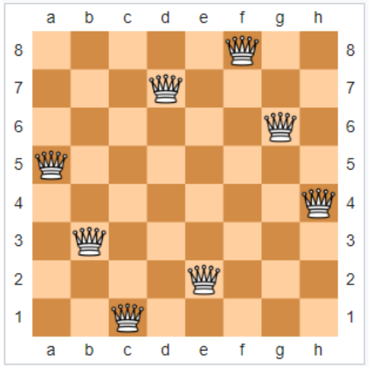
- Brute Force
- Divide & Conquer
- Backtracking
- Branch & Bound
- Dynamic Programming

Backtracking Paradigm

- Backtracking mainly used in computational problems with possible constraint satisfactions. This helps to make decision valid or abandoned solutions.
- Is a useful technique to solve combinatorial problems that have exactly one solution. To find that correct combination, problem is dealt in multiple stages and there are multiple options at each stage. So, each option is explored at every stage one-by-one to find the correct combination. But if any option does not satisfy at any point, then program backtracks one step and starts exploring the next option. In this way, the program explores all possible actions to find the correct combination.
- A classic example is eight queens puzzle.



8
3
4
7
2
0
2
3
6
1
0
Length = 2
Best so far
Already reached a length of 2 without a solution, so stop looking
Length = 4
Best so far



- This paradigm is used to model decision trees.
- The depth of the tree can be represented by the depth of the recursive stack.
- The breadth of the tree can be represented by the number of iterations performed by the for loop within each recursive stack frame.

Branch & Bound

- Is a useful technique to solve combinatorial optimization problems that have multiple solutions, and we are interested to find the most optimum solution.
- Problem is solved in same fashion of backtracking and entire solution is represented in the form of a state space tree*. As the program progresses each state combination is explored, and the previous solution is replaced by new one if it is not the optimal than the current solution.
- * A **space state tree** is a tree representing all the possible states (solution or nonsolution) of the problem from the root as an initial state to the leaf as a terminal state.

- **Difference:** Backtracking is used to solve the decision problem whereas the Branch-N-Bound is used to solve the optimization problem. Both techniques follow the brute force method and are used to generate the State Space tree.

Binary Tree Labeling (Combinatorics)

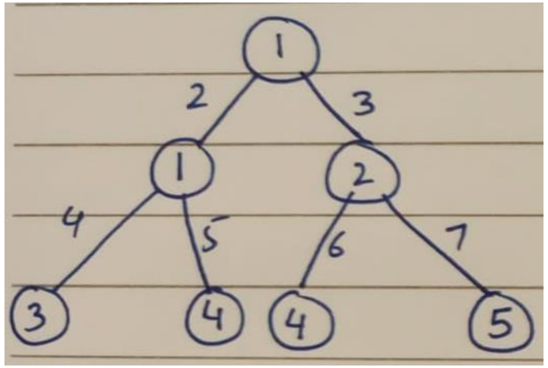
$$es(G) \geq \max\left\{\left\lceil \frac{|E(G)|+1}{2}, \Delta(G) \right\rceil\right\}$$

V = 7, E = 6

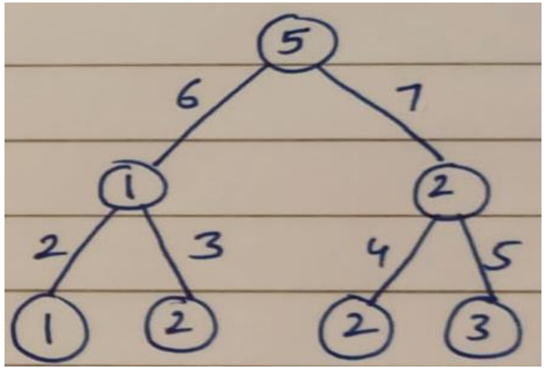
k= ⌈ V/2 ⌉ = **4**

Min weight = 2

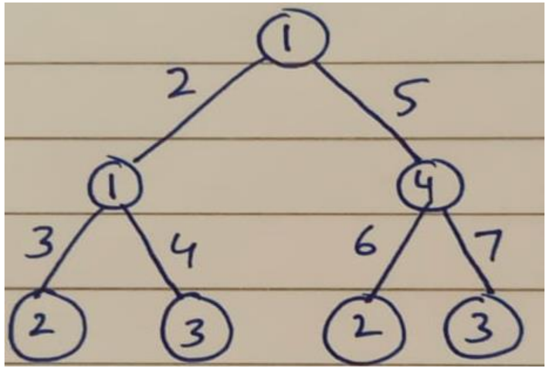
Max weight = 7



✗



✗

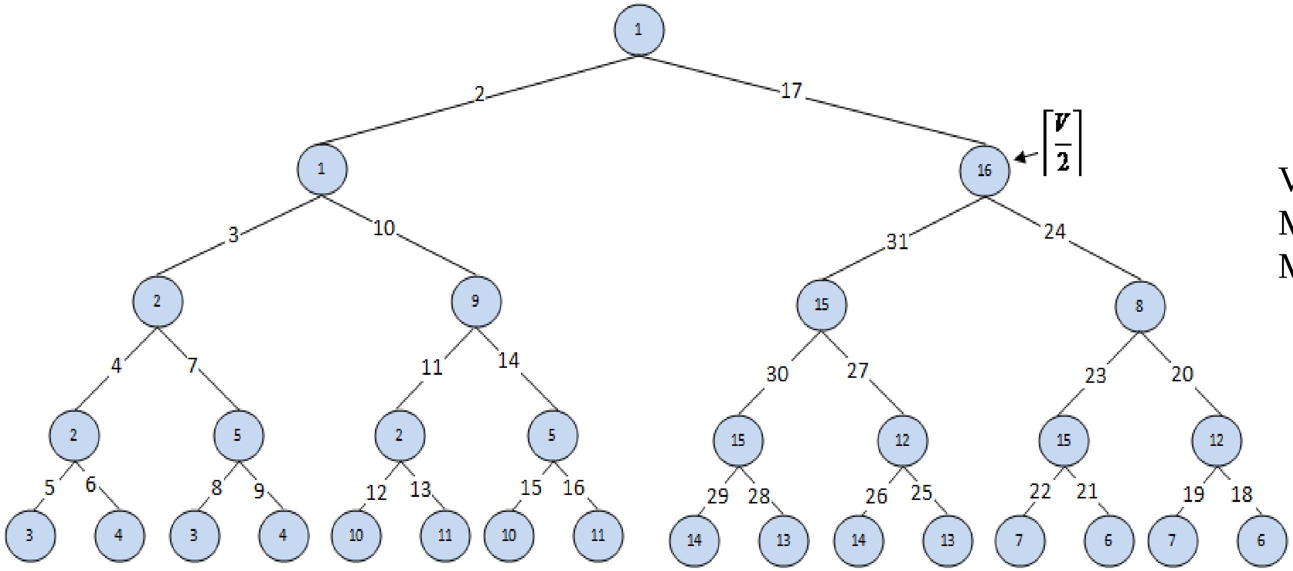


✓

Results of Computer Based Experiment

- Value of ***k*** is exactly $= \left\lceil \frac{V}{2} \right\rceil$

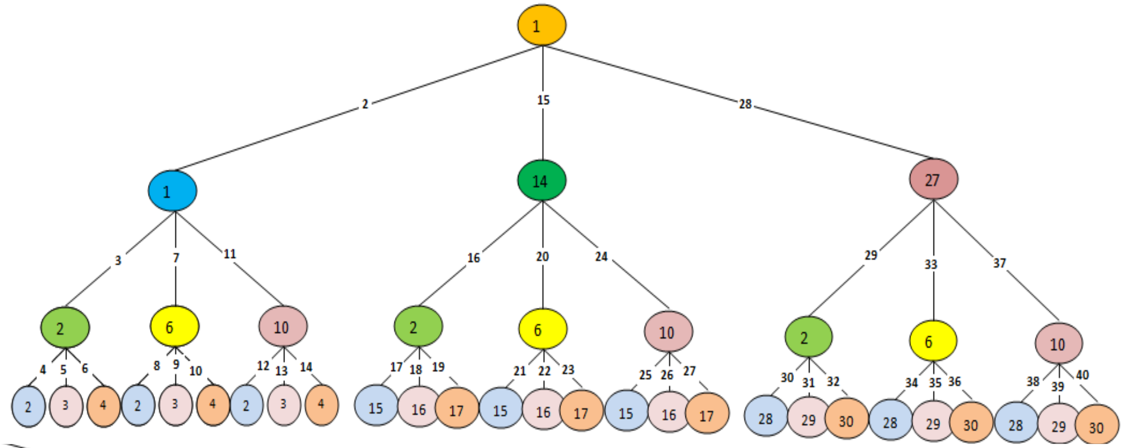
- Value of ***k*** always found at ***m***^{*th*} child of root vertex.
- Left tree follows ascending order whereas right tree follows descending order.

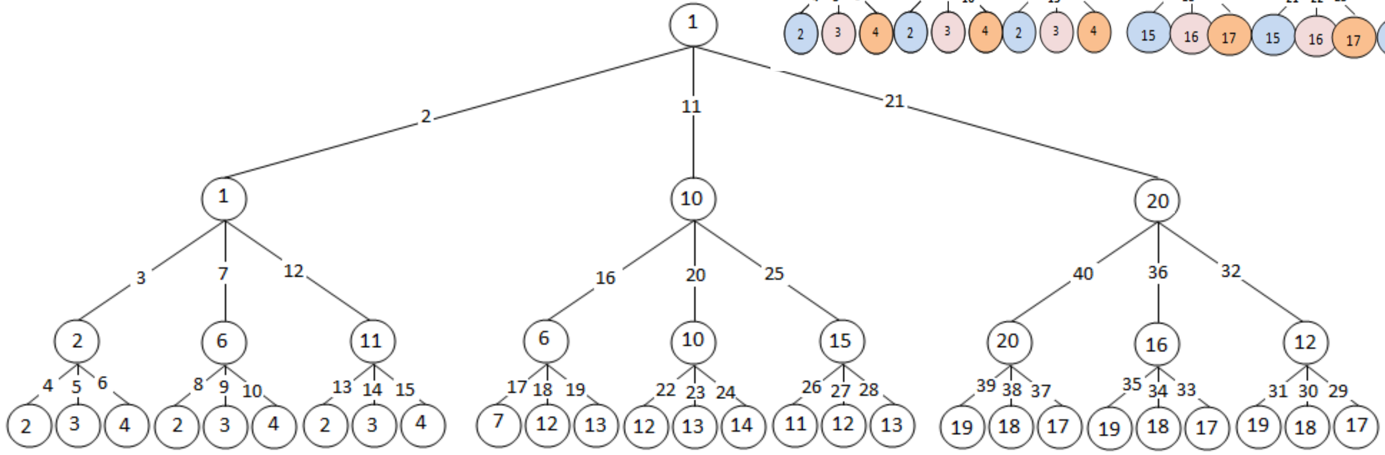


V=31, E=30, k=16
Min weight = 2
Max weight = 31

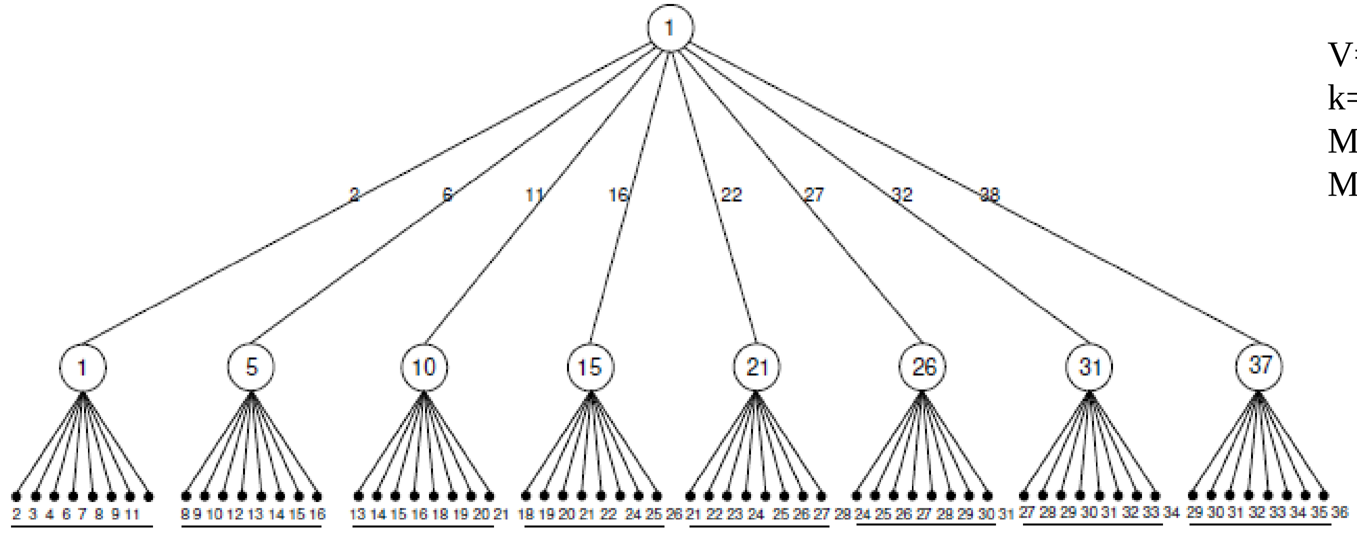
Complete Ternary Tree :($T_{3,3}$)

V=40, E=39, k=20
Min weight = 2,
Max weight = 40





Complete m-ary Trees



V=73, E=78,
k=37
Min weight = 2,
Max weight = 73

$$d = \left\lfloor \frac{\left\lceil \frac{V}{2} \right\rceil}{m-1} \right\rfloor$$

Level 1	Childs of root	Left most	2	3	....	m^th
	Labels	1	$val_1 = 0 + d$	$val_2 = val_1 + d$	….	k
	Edges Weights	2	$val_1 + 1$	$val_2 + 1$	….	k+1

A. Ahmad, M. Baca, M.A. Asim, R. Hasni, "Computing Edge Irregularity Strength of Complete m-ary Trees using Algorithmic Approach", UPB Scientific Bulletin, Applied Mathematics and Physics 80(3), 2018.

Complete Graphs: es(Kn)

- Complete graph is a famous type of regular graphs denoted as K_n.
- In complete graph every two distinct vertices are adjacent.
- Complete graph K_n is a regular graph of degree $r = n\text{-}1$ and size is $K_n = \frac{(n(n-1))}{2}$

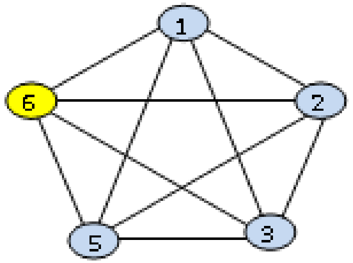
Known Result: Let G be a complete graph K_n of order n and Fibonacci sequence F_n.

$$es(K_n) \le F_n$$

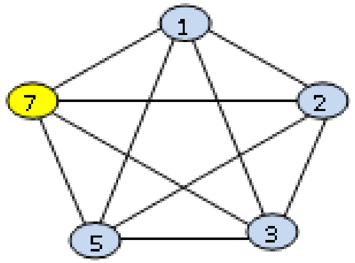
A. Ahmad, O. Al-Mushayt, M. Baca, On edge irregularity strength of graphs, Appl. Math. Comput. 243(2014), 607–610.

Algorithm for K_5 Adjacency Matrix

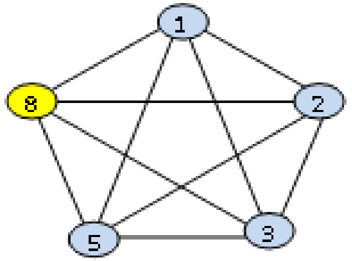
	1	2	3	5	6
1	0	3	4	6	7
2		0	5	7	8
3			0	8	9
5				0	11
6					0



	1	2	3	5	7
1	0	3	4	6	8
2		0	5	7	9
3			0	8	10
5				0	12
6					0

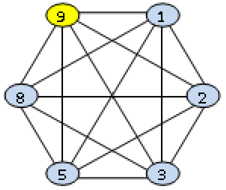


	1	2	3	5	8
1	0	3	4	6	9
2		0	5	7	10
3			0	8	11
5				0	13
6					0

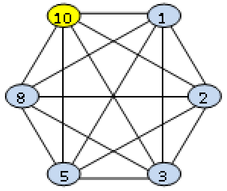


Algorithm for K_6 Adjacency Matrix

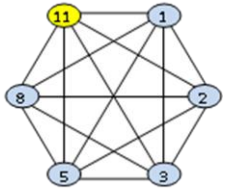
	1	2	3	5	8	9
1	0	3	4	6	9	10
2		0	5	7	10	11
3			0	8	11	12
5				0	13	14
6					0	15
9						0



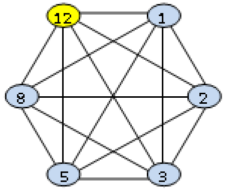
	1	2	3	5	8	10
1	0	3	4	6	9	11
2		0	5	7	10	12
3			0	8	11	13
5				0	13	15
6					0	16
9						0



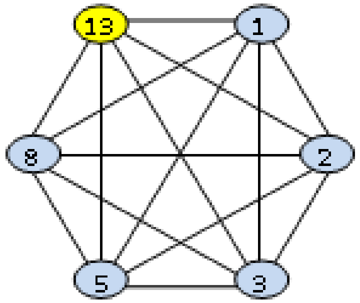
	1	2	3	5	8	11
1	0	3	4	6	9	12
2		0	5	7	10	13
3			0	8	11	14
5				0	13	16
6					0	17
9						0



	1	2	3	5	8	12
1	0	3	4	6	9	13
2		0	5	7	10	14
3			0	8	11	15
5				0	13	17
6					0	18
9						0



	1	2	3	5	8	13
1	0	3	4	6	9	14
2		0	5	7	10	15
3			0	8	11	16
5				0	13	18
6					0	19
9						0

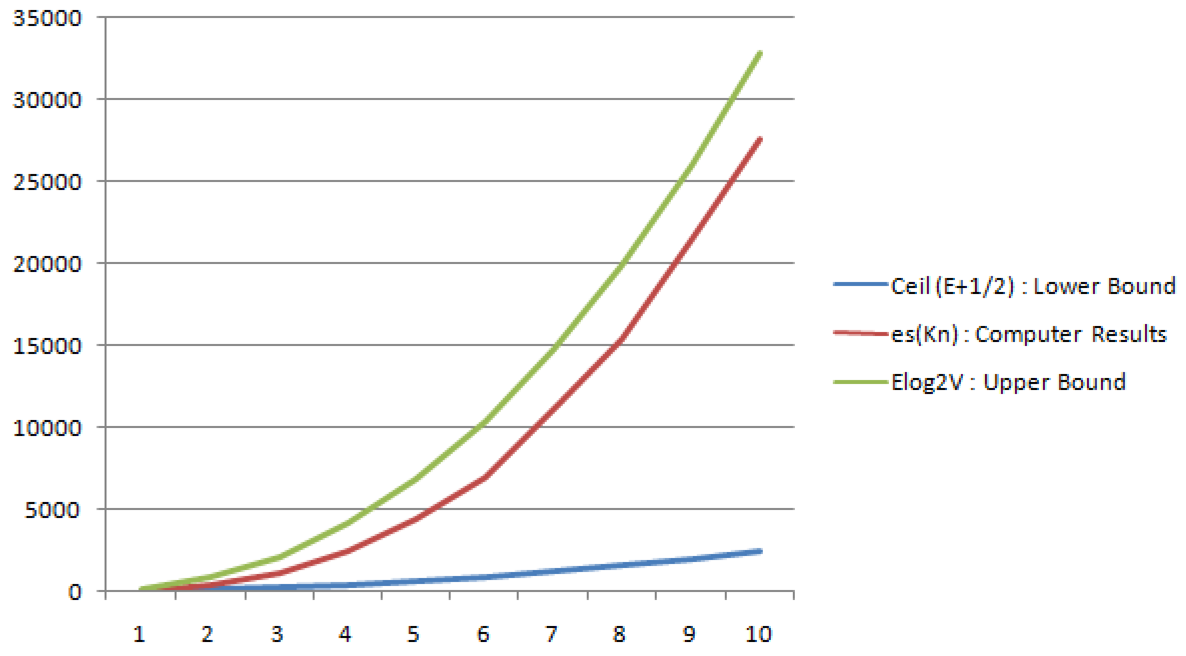


Improved upper bound for es(Kn)

V	E	$es(K_n)$: Computer Results	$es(K_n) \leq E\log_2 V$	F_n
10	45	53	149	55
20	190	413	821	6765
30	435	1161	2134	832040
40	780	2497	4151	102334155
50	1225	4447	6913	12586269025
60	1770	6980	10455	1548008755920
70	2415	11110	14802	190392490709135
80	3160	15470	19977	23416728348467600
90	4005	21492	25999	2880067194370810000
100	4950	27602	32887	354224848179261000000

Asim, M. A., Ahmad, A. and Hasni, R., Iterative algorithm for computing irregularity strength of complete graph, Ars Combin. 138 (2018), 17-24

Improved upper bound for es(Kn)

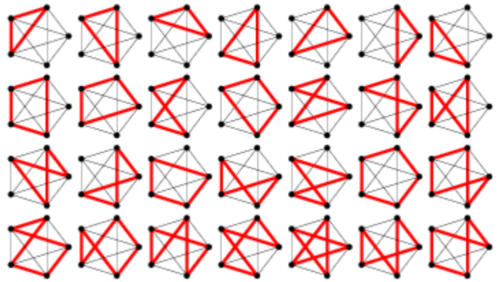


International Conference on
Graph Theory and Information Security
ICGTIS-2017 Indonesia.

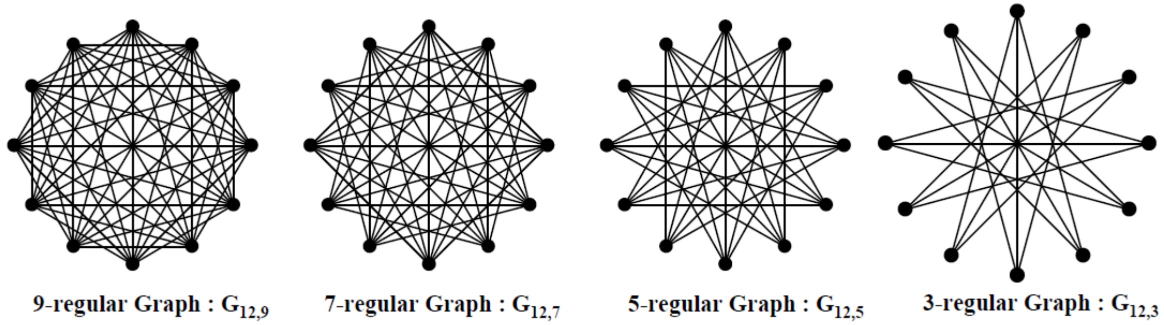
Prof. Baca, accepted the power of
Algorithmic results.
$E\log_2 V << F_n$

Graph De-composition

By deleting edges from complete graph, sub-graphs can be extracted like paths, cycles, star graphs and disjoint graphs.



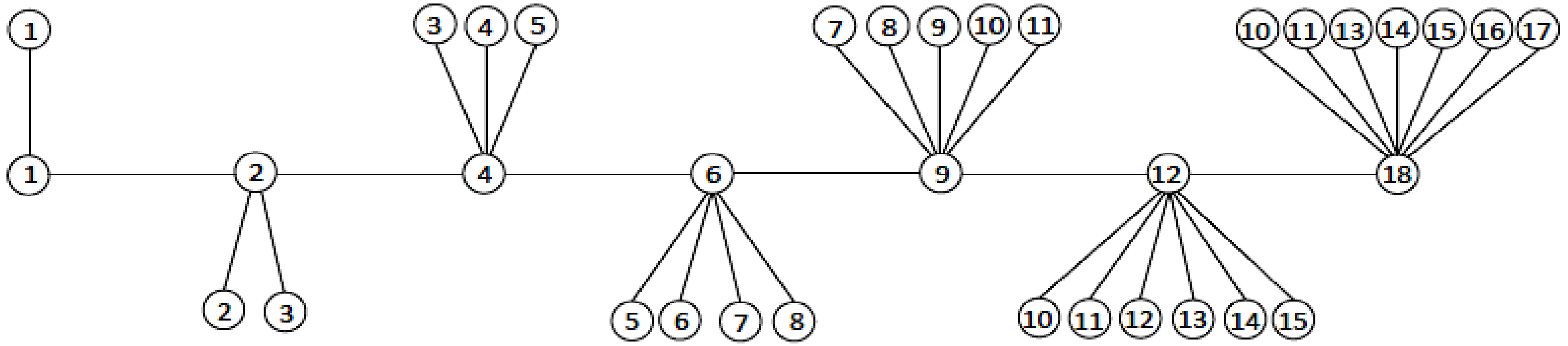
More interested types of ***r*-Regular graphs** can be extracted by deleting Hamiltonian cycles or *j*-factors from complete graph.


9-regular Graph : $G_{12,9}$
7-regular Graph : $G_{12,7}$
5-regular Graph : $G_{12,5}$
3-regular Graph : $G_{12,3}$

M. A. Asim, R. Hasni, A. Ahmad, B. Assiri, A. S. Fenovcikova, "Irregularity Strength of Circulant Graphs using Algorithmic Approach", IEEE Access, 2021.

Non - Homogeneous Caterpillar

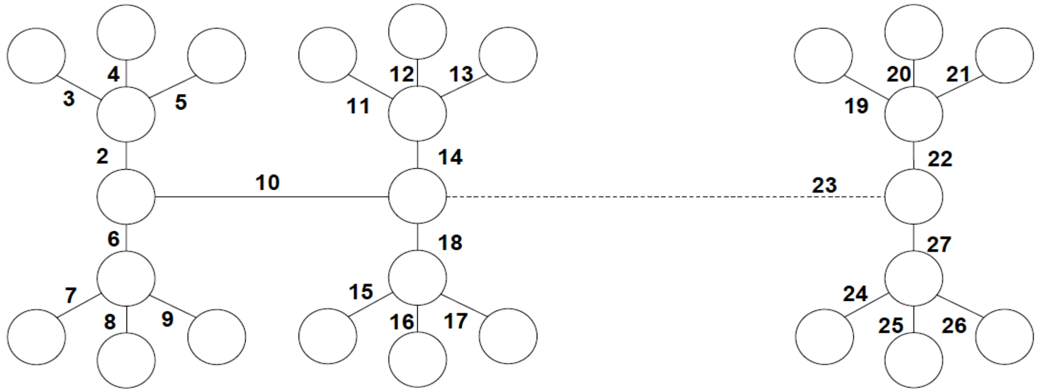
- T = $CT_n(m_1, m_2, \ldots .m_n)$ is a non-homogeneous caterpillar.
- Order of T is ***n(n+3)/2***



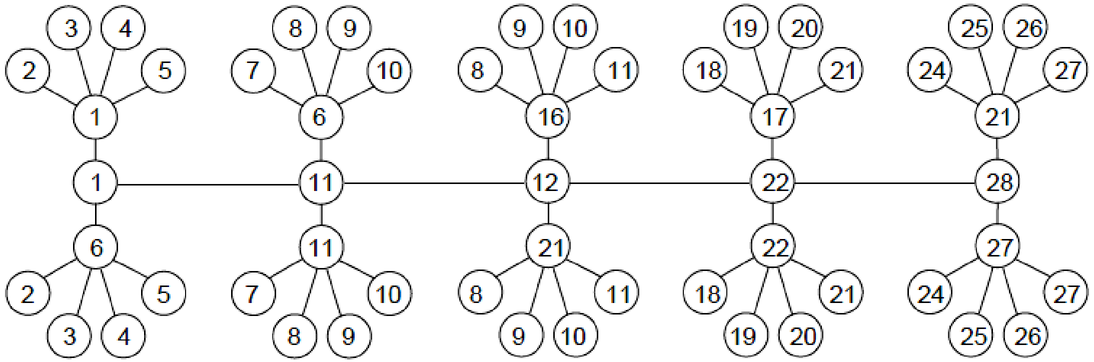
Vertex label is at most ***k*** and the edge weights are unique thus T admits edge irregular $k = \left\lceil \frac{\left\lfloor \frac{n(n+3)}{2} \right\rfloor}{2} \right\rceil - labeling$

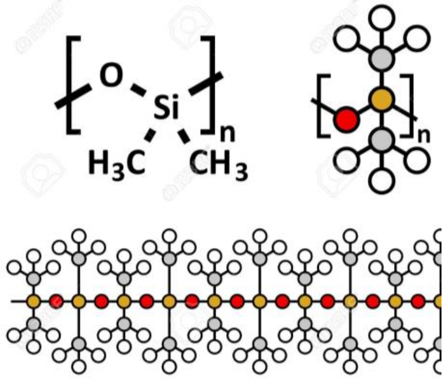
Homogeneous Lobster

- For any star graph of order ***p,*** where ***p ≥ 2.***
- Internal vertices of two S_p are connected with each vertex of path graph P_n.
- New structure will become lobster ***Lob(n, p)***.
- Order of Lob(n, p) = ***n(2p+1)***.



Homogeneous Lobster





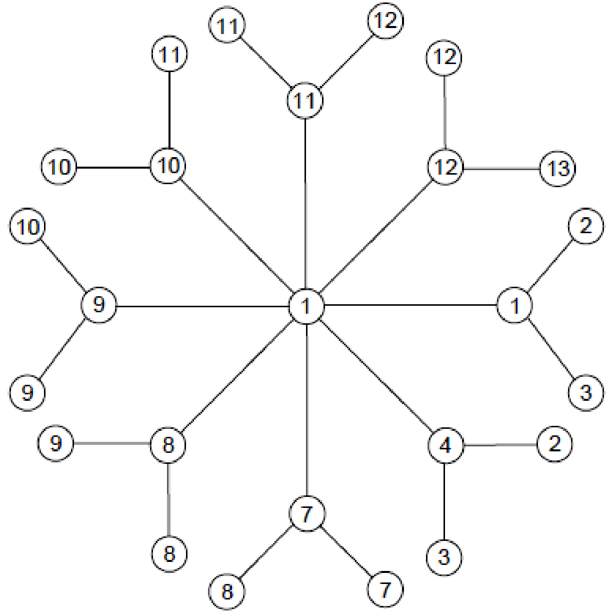
All vertex labels are at most ***k*** and the edge weights are distinctive, thus ***Lob(n,p)*** admits the edge irregular *k*-labeling.

$$\boldsymbol{k} = \left\lceil \frac{|\boldsymbol{n}(\mathbf{2}\boldsymbol{p} + \mathbf{1})|}{\mathbf{2}} \right\rceil$$

Homogenous amalgamated Star : $S_{n,3}$

- For $\boldsymbol{n \geq 3}$, homogenous amalgamated star $\boldsymbol{S_{n,3}}$ admits the edge irregular k-labeling.
- Order of $S_{n,3}$ = 3n+1.
- Vertex label is at most ***k*** and the edge weights are distinctive, thus $\boldsymbol{S_{n,3}}$ admits the edge irregular *k*-labeling.

$$\boldsymbol{k = \left\lceil \frac{3n+1}{2} \right\rceil}$$



UMKC

Assignment-2:

Vertex *k*-Labeling of Graphs using Algorithmic Approach

Submission Date : Nov 8th 2024

Assignment-2 (Research Assignment)	25 %
Presentation (Work done in Assignment-2)	10 %

Assig-2: Tasks to Do for Problem-1 & 2

1. Find out the best data-structure to represent / store the graph in memory.
2. Devise an algorithm to assign the labels to the vertices using vertex k-labeling definition. (Main Task)
3. What design strategy you will apply, also give justifications that selected strategy is most appropriate.
4. How traversing will be applied?
5. Store the labels of vertices and weights of the edges as an outcome.
6. Compare your results with mathematical property and tabulate the outcomes for comparison.
7. Hardware resources supported until what maximum value of V,E.
8. Compute the Time Complexity of your algorithm *T(V,E)* or *T(n)*.

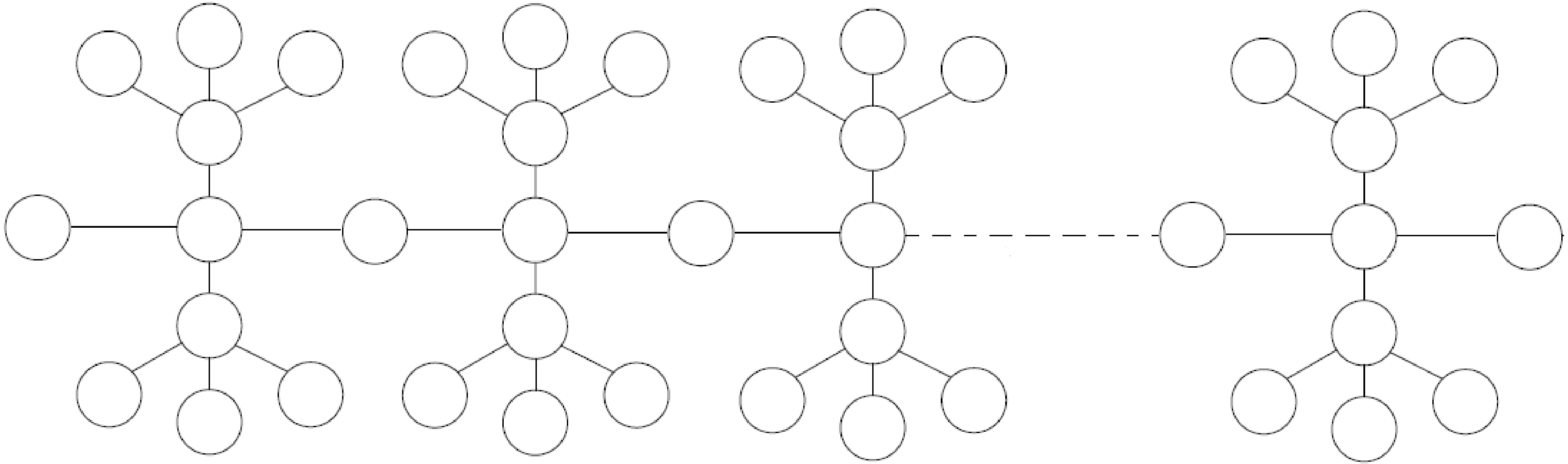
Problem-1: Homogeneous Lobster+1

Inputs: ***n*** and ***p***

For any star graph of order ***p*,** where ***p* ≥ *2*.**

Internal vertices of two $\boldsymbol{S_p}$ are connected with each vertex of path graph $\boldsymbol{P_n}$.

New structure will become lobster ***Lob(n, p)***.



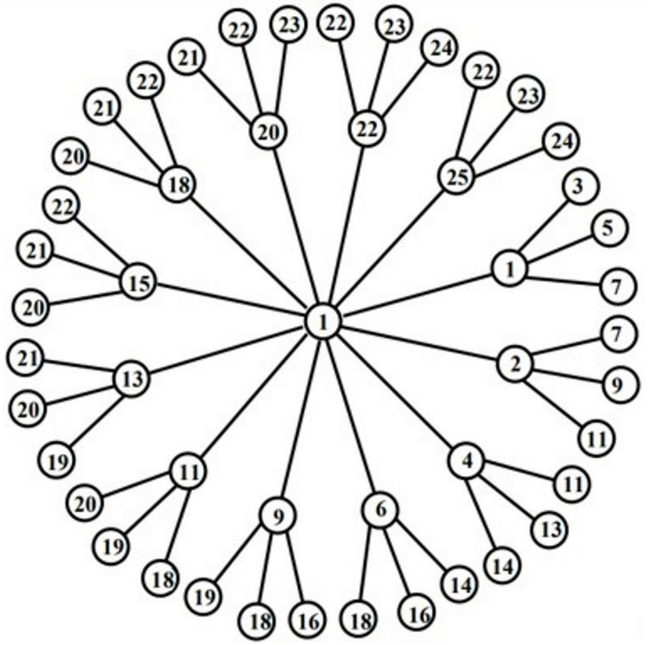
Problem-2: Homogenous amalgamated Star : $S_{n,m}$

- For $n \geq 3$, homogenous amalgamated star $S_{n,m}$ admits the edge irregular k-labeling.

- Order of $\mathbf{S}_{n,m}$ = $\boldsymbol{m}$ x $\boldsymbol{n}$ + 1

- Vertex label is at most $\boldsymbol{k}$ and the edge weights are distinctive, thus $S_{n,m}$ admits the edge irregular k-labeling.

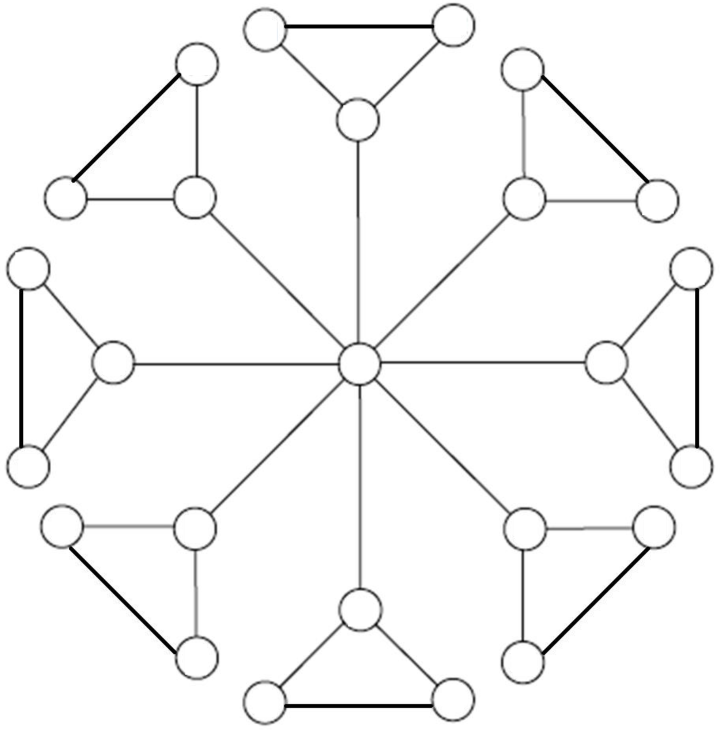
$$\boldsymbol{k} = \left\lceil \frac{\boldsymbol{m} \text{ x } \boldsymbol{n} + \mathbf{1}}{\mathbf{2}} \right\rceil$$



Problem-3: For any number n as branches with centroid vertex.

Tasks to Do:

1. Suggest a suitable name.
2. Devise the formulae for calculating order and size of the graph.
3. Data-structure to store the graph.
4. Assign the labels using algorithm.
5. Store the labels of vertices and weights of the edges as an outcome.



Problem-4: For any number n as branches with centroid vertex.

Tasks to Do:

1. Suggest a suitable name.
2. Devise the formulae for calculating order and size of the graph.
3. Data-structure to store the graph.
4. Assign the labels using algorithm.
5. Store the labels of vertices and weights of the edges as an outcome.