Application of Tree Structure & Algorithms in Different Fields of Sciences

Assig-4
ACT & Prog. (5501)

Dr. Ahsan Asim

Assig-4 : Research + Programming Assignment

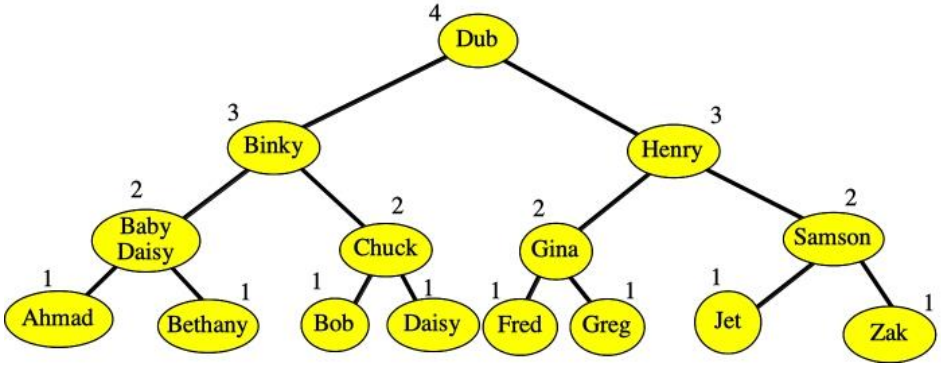
Group Assignment = 5 Members (self sign-up) @ Canvas
Discussion in class : Nov 5th, 2025
Report Submission Date : Dec 5th, 2025
Class Presentations: (on Zoom)
Group-1 & 2 : Dec 8th , 2025
Group-3 & 4 : Dec 10th , 2025

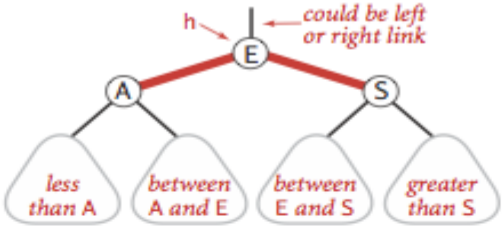
Report with proper template: Abstract, Introduction, Experiment Objectives, Results, Explanation, Conclusion, References	20 Points
Presentation (Work done in Assignment-4)	5 Points

Problem-1: Implementation of AVL and Red & Black Trees

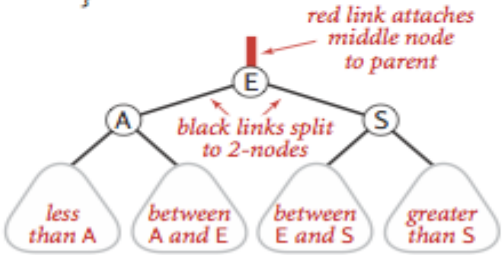
Tasks:

- Read the rules/concept of Red & Black Tree
- Store 1000 frequent words from the given file in both Trees
- Search ***same*** 100 random words from both data structures and count comparisons.
- Prove the statement, ***AVL trees*** offer faster search due to their strict height balancing whereas Red-Black trees are faster while insert and delete operations.






```
void flipColors(Node h)
{
   h.color = RED;
   h.left.color = BLACK;
   h.right.color = BLACK;
}
```

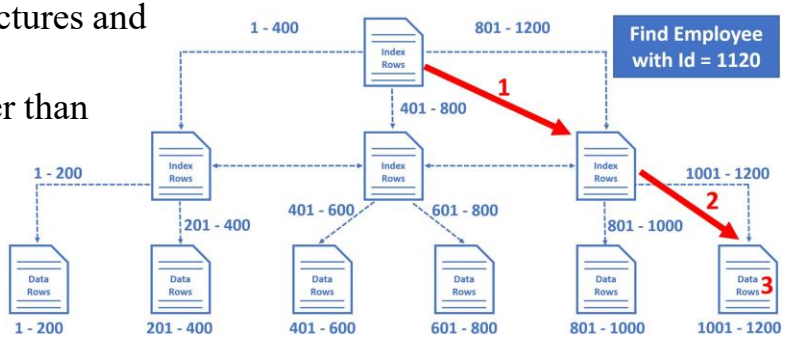


Flipping colors to split a 4-node

Problem-2: Implementation of B Tree

Tasks:

- Store customer data from the given file in B Tree using ***First-Name*** as key
- Store the same data in any built-in Collection.
- Search ***same*** 100 records using the same key from both data structures and count comparisons.
- Print the results, ideally searching phenomenon in B Tree is faster than any other data-structure.



SQL Data is stored as "Data Pages on Physical Drive"

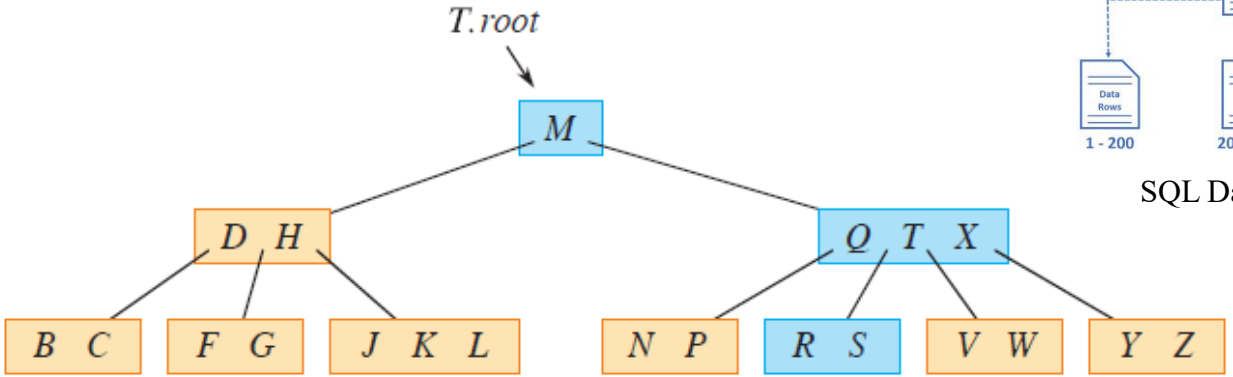
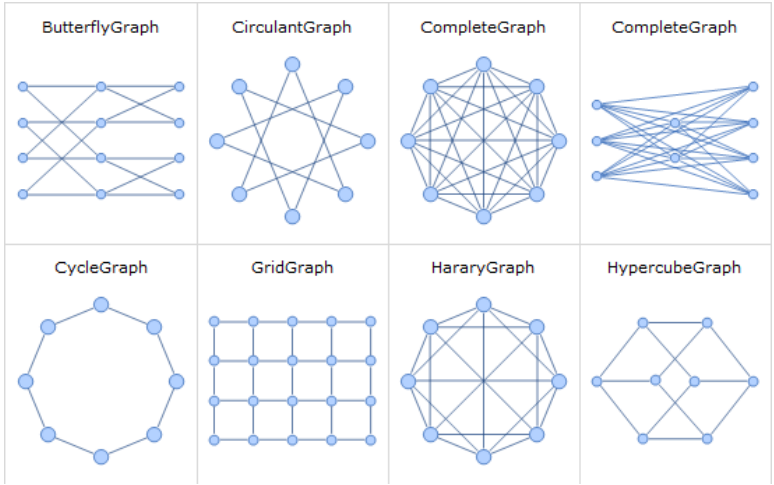
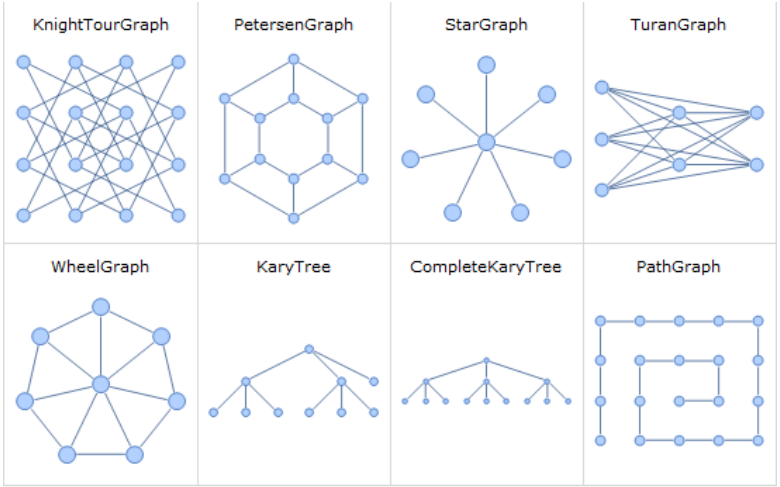


Fig: B-Tree whose keys are the consonants of English. An internal node ***x*** containing ***x:n*** keys has ***x:n+1*** children. All leaves are at the same depth in the tree. Blue shaded nodes are examined in a search for the letter R.

Graph

- Graph is a non-linear data structure consists of vertices (nodes) and edges(links) that connect any two nodes.
- A graph $G = (V , E)$ consists of finite sets of vertices (*order*) and edges (*size*).





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

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

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

- For every two different edges e and f

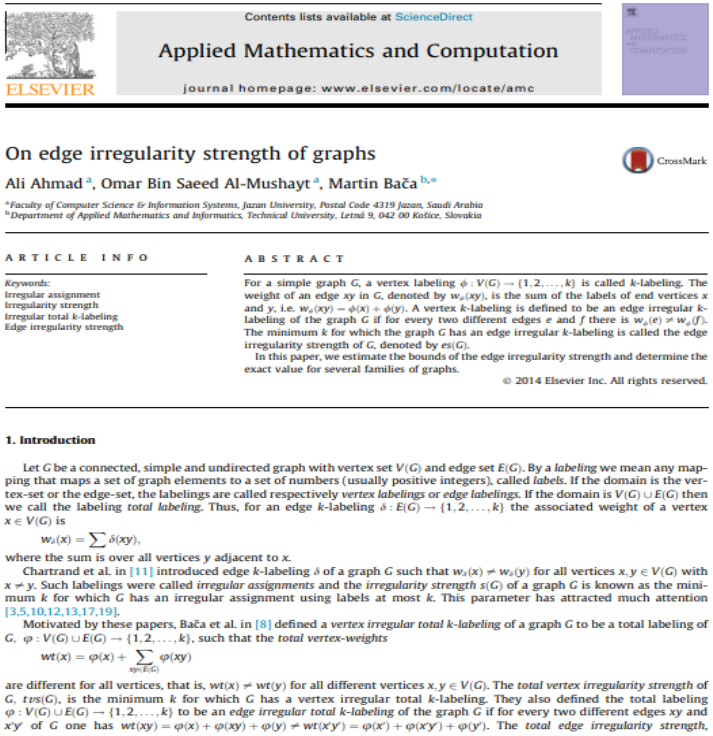
$$w_\phi(e) \neq w_\phi(f)$$

- Edge weights:

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

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

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



Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

ELSEVIER

On edge irregularity strength of graphs

Ali Ahmad [a], Omar Bin Saeed Al-Mushayt [a], Martin Bača [b,*]

[a] Faculty of Computer Science & Information Systems, Jazan University, Postal Code 4319 Jazan, Saudi Arabia
[b] Department of Applied Mathematics and Informatics, Technical University, Letná 9, 042 00 Košice, Slovakia

ARTICLE INFO

Keywords:
Irregular assignment
Irregularity strength
Irregular total k-labeling
Edge irregularity strength

ABSTRACT

For a simple graph G, a vertex labeling $\phi : V(G) \to \{1, 2, \ldots, k\}$ is called k-labeling. The weight of an edge xy in G, denoted by $w_\phi(xy)$, is the sum of the labels of end vertices x and y, i.e. $w_\phi(xy) = \phi(x) + \phi(y)$. A vertex k-labeling is defined to be an edge irregular k-labeling of the graph G if for every two different edges e and f there is $w_\phi(e) \neq w_\phi(f)$. The minimum k for which the graph G has an edge irregular k-labeling is called the edge irregularity strength of G, denoted by $es(G)$.

In this paper, we estimate the bounds of the edge irregularity strength and determine the exact value for several families of graphs.

1. Introduction

Let G be a connected, simple and undirected graph with vertex set $V(G)$ and edge set $E(G)$. By a *labeling* we mean any mapping that maps a set of graph elements to a set of numbers (usually positive integers), called *labels*. If the domain is the vertex-set or the edge-set, the labelings are called respectively *vertex labelings* or *edge labelings*. If the domain is $V(G) \cup E(G)$ then we call the labeling *total labeling*. Thus, for an edge k-labeling $\delta : E(G) \to \{1, 2, \ldots, k\}$ the associated weight of a vertex $x \in V(G)$ is

$$w_\delta(x) = \sum \delta(xy),$$

where the sum is over all vertices y adjacent to x.

Chartrand et al. in [11] introduced edge k-labeling δ of a graph G such that $w_\delta(x) \neq w_\delta(y)$ for all vertices $x, y \in V(G)$ with $x \neq y$. Such labelings were called *irregular assignments* and the *irregularity strength* $s(G)$ of a graph G is known as the minimum k for which G has an irregular assignment using labels at most k. This parameter has attracted much attention [3,5,10,12,13,17,19].

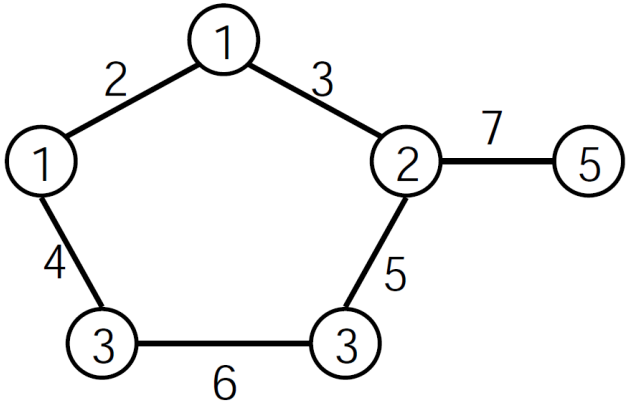
Motivated by these papers, Bača et al. in [8] defined a *vertex irregular total k-labeling* of a graph G to be a total labeling of G, $\varphi : V(G) \cup E(G) \to \{1, 2, \ldots, k\}$, such that the *total vertex-weights*

$$wt(x) = \varphi(x) + \sum_{xy \in E(G)} \varphi(xy)$$

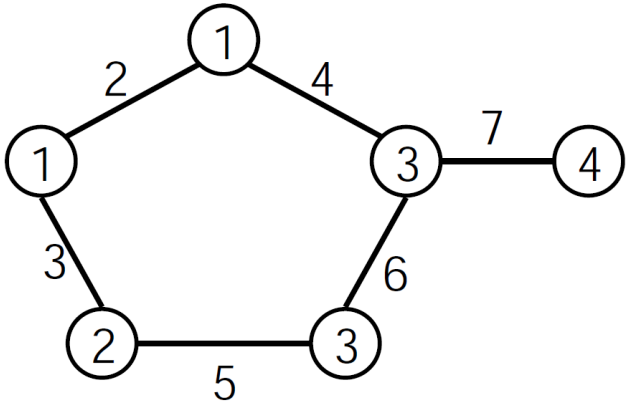
are different for all vertices, that is, $wt(x) \neq wt(y)$ for all different vertices $x, y \in V(G)$. The *total vertex irregularity strength* of G, $tvs(G)$, is the minimum k for which G has a vertex irregular total k-labeling. They also defined the total labeling $\varphi : V(G) \cup E(G) \to \{1, 2, \ldots, k\}$ to be an *edge irregular total k-labeling* of the graph G if for every two different edges xy and $x'y'$ of G one has $wt(xy) = \varphi(x) + \varphi(xy) + \varphi(y) \neq wt(x'y') = \varphi(x') + \varphi(x'y') + \varphi(y')$. The *total edge irregularity strength*,

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

Binary Tree Labeling (Combinatorics)

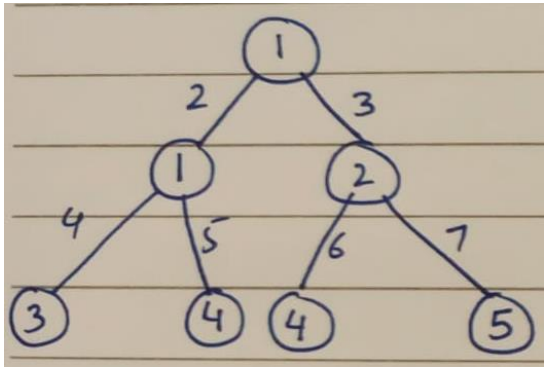
$$es(G) \geq \max\left\{\left\lceil \frac{|E(G)|+1}{2}, \Delta(G) \right\rceil\right\}$$

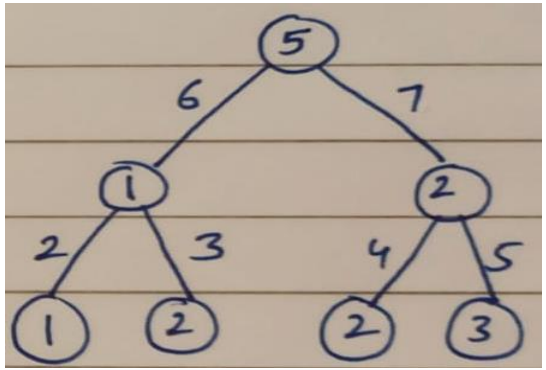
V = 7, E = 6

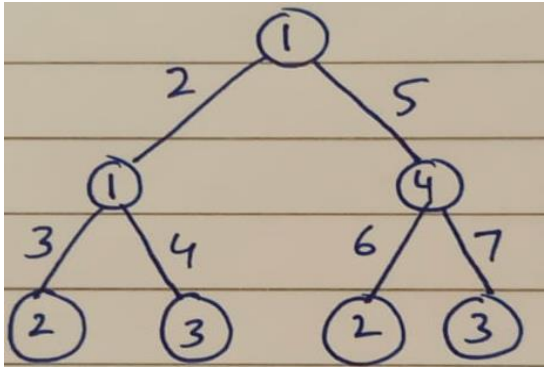
k= [V/2] = **4**

Min weight = 2

Max weight = 7

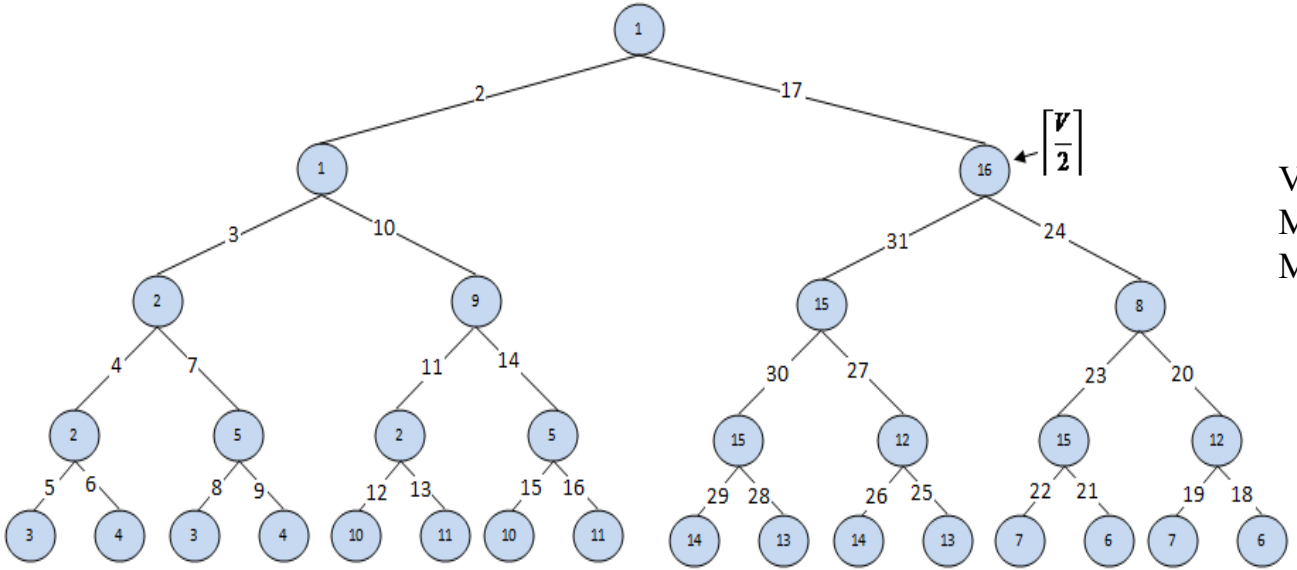






Algorithmic Results for Perfect Binary Tree

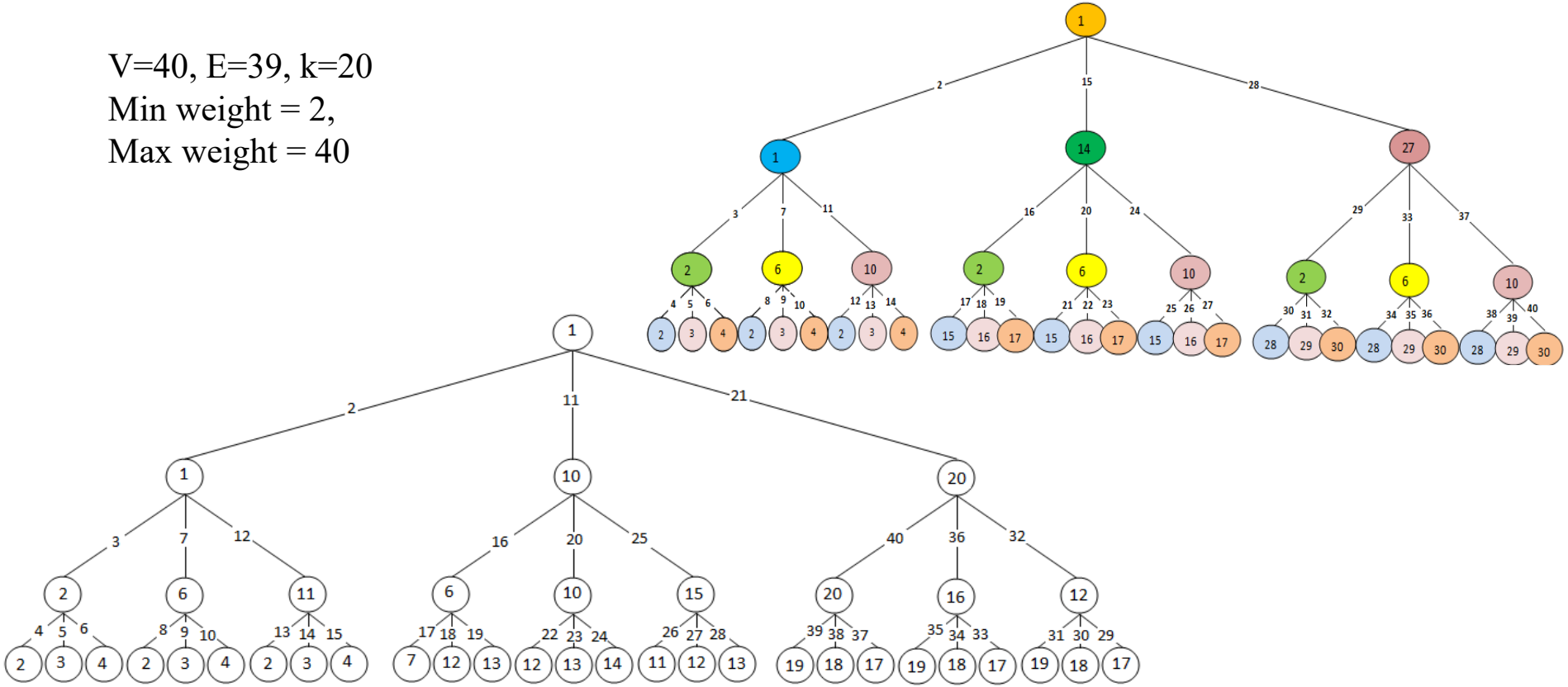
- Value of ***k*** is exactly $= \left\lceil \frac{V}{2} \right\rceil$
- Value of ***k*** always found at m^{th} child of root vertex.
- Left tree follows ascending order whereas right tree follows descending order.



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

Problem-3: Vertex k-labeling for Perfect Ternary Tree ($T_{m,h} = T_{3,5}$)

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



Challenge Problem:
Vertex k-labeling Homogeneous Amalgamated Star ($S_{m,n}$)

Algorithm 1 Star-Labeling (m, n)

1: **Input:** Two positive integers n and m, where $m, n \geq 2$
2: **Output:** Labels of vertices $\{1, 2, \ldots, k\}$ stored in arrays L1 and L2.
3: $V = m * n + 1$
4: $Diff = \left\lfloor \frac{\lceil |V|/2 \rceil}{m-1} \right\rfloor$
5: $L0 = 1$
6: $L1[1-3][1] = 0, 1, 2$
7: **for** $i = 2$ to m **do**
8: $L1[1][i] = L1[1][i-1] + Diff$
9: $L1[2][i] = \lfloor L1[1][i] \rfloor$
10: $L1[3][i] = L1[2][i] + 1$
11: **end for**
12: $L1[3][m+1] = NIL$
13: $h, j = 1, 2$
14: **for** $wt = 3$ to V **do**
15: **if** $(wt \neq L1[3][j])$ **then**
16: $L2[1][h] = wt$
17: $L2[2][h] = wt - L1[2][\lfloor (h + (n-2))/(n-1) \rfloor]$
18: $h = h + 1$
19: **else**
20: $j = j + 1$
21: **end if**
22: **end for**

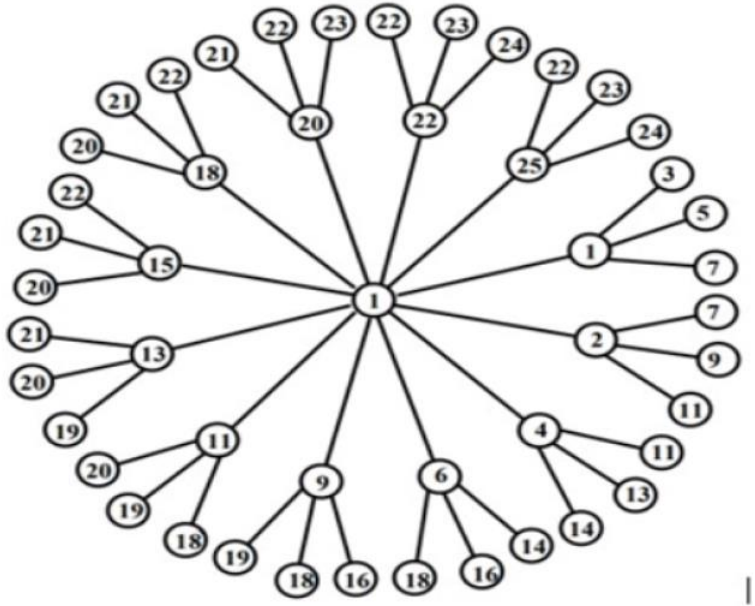


Fig: Vertex k-labeling of $S_{12,4}$

Note: 3 Points Extra for this problem whoever want to attempt this question as individual or as group. Extra points may compensate your quizzes grade.

Tasks to Do for Problem 3 and Challenge Problem

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. How traversing will be applied?
4. Store the labels of vertices and weights of the edges as an outcome.
5. Compare your results with mathematical property and tabulate the outcomes for comparison.