QALAAI ZANISTSCIENTIFIC JOURNAL

A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq

Vol. (10), No (1), Spring 2025

ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

An Analytical Study of Graph Algorithms: An Overview of Graph Theory

Kamaran Jamal Hamad

Department of Mathematics, Faculty of Science, Soran University, Soran, Kurdistan Region, Iraq.

Email: Kamaranmath93@gmail.com

Salim S. Mahmood

Department of Mathematics, Faculty of Science, Soran University, Soran, Kurdistan Region, Iraq.

Email: salimsaeedmahmood@gmail.com

ARTICLE INFO

Article History:

Received: 16/11/2023

Accepted: 11/1/2024

Published: Spring2025

Keywords:

Depth-First Search,

Breadth-First Search,

Prim's algorithm,

Minimum Spanning Tree,

Kruskal's algorithm

Doi:

10.25212/lfu.qzj.10.1.43

ABSTRACT

Graph theory is a fundamental branch of mathematics with diverse applications in computer science and various other fields. This research provides an analytical overview of essential graph algorithms, focusing on the Depth-First Search, Breadth-First Search, Prim's algorithm, and Kruskal's algorithm for finding minimum spanning trees. Through an in-depth exploration of these algorithms, this study aims to shed light on their principles, advantages, and applications. The insights gained from this research can be valuable in solving real-world problems that involve networks and optimization.

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

1. Introduction

Graphs serve as a powerful mathematical representation for modeling relationships, networks, and structures in diverse domains [18]. The field of graph theory has given rise to a multitude of algorithms, each designed to address specific graph-related problems. In this research, we delve into the world of graph algorithms, focusing on four fundamental ones: Depth-First Search (DFS), Breadth-First Search (BFS), Prim's algorithm, and Kruskal's algorithm [15]. Our objective is to provide a comprehensive understanding of these algorithms and highlight their significance in solving problems related to connectivity, traversal, and optimization in graphs. These algorithms form the backbone of numerous applications, from computer networking to route planning and resource allocation [16]. By exploring the principles, characteristics, and applications of these algorithms, we aim to equip researchers and practitioners with a better understanding of how to harness the power of graph theory.

2. Undirected Graphs

An undirected graph is a pair G(V,E) where V is vertices set and E is edges set. Vertices are often called nodes and the elements of the set E are called edges [4]. Figure (1) shows an example of an undirected graph where: V={a,b,c,d} and E={(a,b), (a,d), (b,c),(b,d) ,(c,d)}.

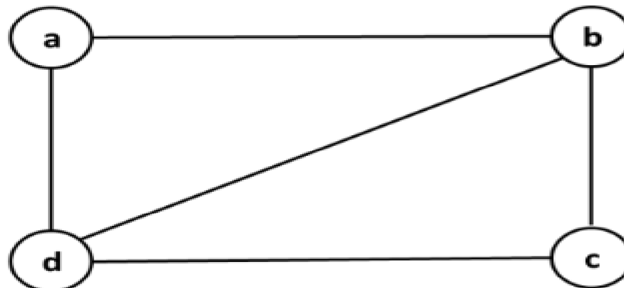


Figure (1) an example of an undirected graph

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

3. Traversing graphs:

Most algorithms used to solve graph problems examine every vertex and every edge [3]. There are two graph traversal strategies that each provide a way to visit every vertex and every edge only once:

- Depth-first search [13].
- Breadth-first search [11].

3.1 Depth-first search:

Assuming G(V,E) is an undirected graph, applying the depth-first search method is as follows:

- Initially, we designate a starting vertex as v and explore it.
- Next, we opt for any edge (v, w) associated with vertex v and proceed to explore w via this edge.
- In a more general context, assuming x is vertex which most visited, the exploration continues through selecting the edge between x and y that has not been traversed yet and is connected to x. If y is visited earlier, we seek another edge (unvisited) originating from x. If y hasn't been visited, we explore y, and the process recommences from vertex y.
- After investigating all possible paths stemming from y, we backtrack to x, which served as the starting point for reaching y.
- This process of choosing unexplored edges linked to x persists until all such edges have been exhausted. The depth-first search algorithm can be described as follows:

Inputs: G(V,E) graph is represented through adjacency lists L(v) where $v \in V$.
Outputs: The set of edges E is divided into two groups:

- T: The set of tree edges.
- B: set of inverse edges (visited edges)

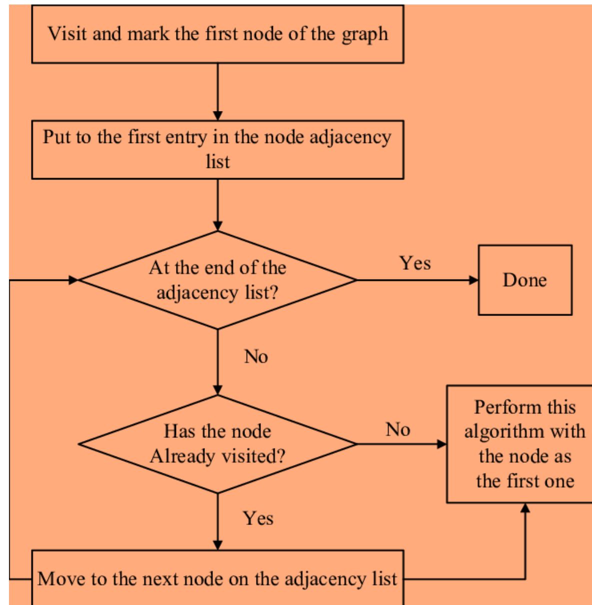
QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

Procedure:

```
Void DFS (G, v)
{
        //G: Graph
        //v: Start Node
        v.visited= true
        for each w∈v. Adjacency_matrix
                if w.visited==false
                        DFS(G, w)
}
Void main()
{
        for each v∈G. vertexes
                v.visited=false
        for each v∈G. vertexes
                DFS(G, v)
}
```

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

The following flowchart shows the steps of DFS algorithm [17]:



3.2 Analysis of DFS:

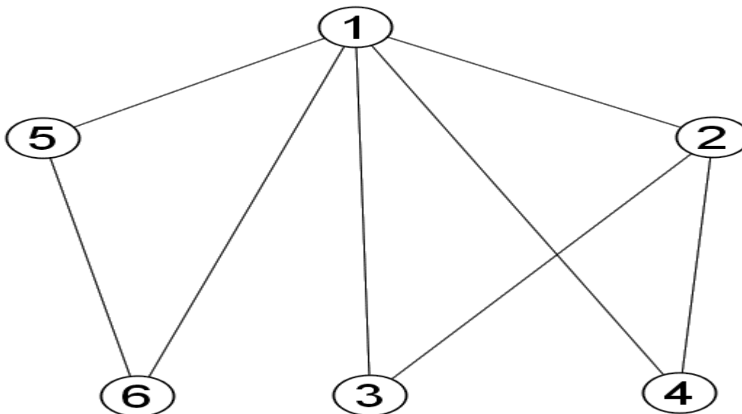
We will now calculate the complexity of the algorithm as a function of n the number of vertices of the graph and m the number of edges. We note that the two basic operations in the algorithm are traversing the edges of the diagram and assigning orders to the label vector. Regarding the address assignment process, we note that n operations are performed in the main program and n operations are performed in the DFS procedure. As for the edge traversal process, we note that the DFS procedure is called once for each vertex, and we note that within DFS, the adjacency list for any vertex is traversed only once, so we have O(m) traversal process as a total value.
We conclude from the above that the total complexity is O(m+n), while the remaining steps require a smaller number of operations.

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

Theorem 1: A depth-first search algorithm to traverse an undirected graph which contains vertices (n) and edges (m) requires O[max(n,m)] steps.

The proof:
Assuming we have a list of vertices in an undirected graph and scan it only once, searching for "new" vertices requires O(n) steps. The time taken by the DFS procedure is directly suitable with the vertices number which near to vertex v, and the DFS procedure is called only once for a given vertex v, because vertex v is addressed as “old” when DFS is invoked for the first time. Thus, the complexity taken by DFS is O[max(n,m)] [5].

Example: Suppose we have the undirected graph shown in the following figure:



Initially, all vertices will be in the “new” state. Assuming vertex “1” is chosen, then DFS(1) will be executed. Assuming vertex “2” is chosen, since vertex “2” is in the “new” state, (1,2) will be added.) to T and DFS(2) will be called. The procedure DFS(2) may select vertex "1", but vertex "1" is in the "old" state and will be ignored. Assuming vertex "3" is selected, since vertex "3" is in the "new" state, (2,3) will be added. To T then DFS(3) will be called and now all vertices adjacent to vertex “3” are in “old” state so DFS(2) will be referenced. By continuing to execute DFS(2), edge (2,4) will be selected and added to T, then DFS(4) will be called. Now there are no vertices adjacent

to vertex “4” in the “new” state, so DFS(2) will be returned as well. There are no vertices adjacent to vertex "2" in the "new" state, so DFS(1) will be returned. Continuing the search, DFS(1) finds vertex "5" and applies the procedure DFS(5) to find vertex "6". Thus, all vertices on the tree become in the “old” state, and thus the algorithm reaches its end.

3.3 Breadth-first search:

We saw previously in the depth-first search method that to traverse the graph, every new vertex in the “new” state is visited, so we continue searching, heading deeper into the tree, then we go back through the last traversed vertex to branch in another direction, and this leads to visiting all vertices. Existing in a subgraph adjacent to vertex v before going to a new subgraph adjacent to vertex v.

In the breadth-first search method, vertices are visited in the order of increasing distances from the starting point v, for example, where the distance is simply the number of characters from the shortest path. By the shortest path that can be reached from vertex v to vertex w, we mean the path that contains the least number of edges. In other words, the algorithm visits all vertices that are a distance d away from vertex v before visiting the vertices that are a distance d+1 away from vertex v [7].

3.4 Comparison between Depth-first and Breadth-first algorithms:

DFS and BFS represent two fundamental graph traversal algorithms employed for the exploration and analysis of graphs or trees. Here's a comparative analysis between these methods [2]:

1) Order of Exploration:
   - DFS delves as far as it can along each branch before backtracking, implying that it traverses deep into the graph before examining sibling nodes.

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

- BFS examines all neighboring nodes at the current depth before progressing to nodes at the next depth, effectively covering the entire breadth of the graph.

2) Data Structures Used:
   - DFS is frequently implemented utilizing a stack, either explicitly or through recursion.
   - BFS relies on a queue data structure for its implementation.

3) Completeness:
   - Both algorithms are complete, meaning they will find a solution if one exists.

4) Memory Usage:
   - In specific scenarios, DFS may consume less memory than BFS because it doesn't require the storage of all nodes at the current depth in a queue.
   - BFS can consume more memory, especially in graphs with a wide branching factor, as it needs to store all nodes at the current depth.

5) Optimality:
   - BFS is guaranteed to find the shortest path in an unweighted graph, making it optimal for such scenarios.
   - DFS may not always discover the shortest path and depends on the order of exploration.

6) Applications:
   - DFS is often used in problems involving searching, backtracking, and traversal of hierarchical structures like trees.
   - BFS is frequently used in scenarios where you need to discover the shortest path, the shortest distance, or minimal steps between nodes, such as in navigation or puzzles.

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

7) Time Complexity:
   - DFS and BFS have a time complexity of O(V + E), where V is the number of vertices, and E is the number of edges.

8) Space Complexity:
   - DFS has a space complexity of O(V), where V is the maximum depth of the recursion stack.
   - BFS has a space complexity of O(V), where V is the maximum number of nodes at a single level in the search.

In summary, the selection between DFS and BFS hinges on the particular problem and the attributes of the graph or tree under consideration. If you need to discover the shortest path or explore graph in more systematic way, BFS is usually preferred. If you're interested in exploring all possibilities or looking for a specific target, DFS may be more suitable.

4. Graph optimization problems:

In this section, two algorithms will be studied to find a Minimum Spanning Tree (MST) for an undirected graph [12], which are Prim algorithm and Kruskal algorithm [8][1].

4.1 Prim algorithm to find MST:

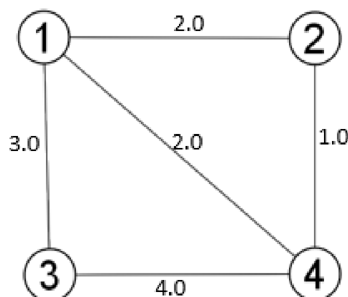
We will study the problem for finding MST for a connected, weighted, undirected graph. For disconnected graphs, the natural extension of the problem is to find MST for each connected component each. It is known that it is possible to find connected components in linear time. MST are only meaningful for undirected graphs with character weights, so any reference to "graph" below will mean "undirected graph," and "weights" will always mean "character weights." (weights edge). And remember that the convention G(V,E,W) means that W is a function that assigns a weight to each edge in E. This is the mathematical description. As for implementation (there is usually

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

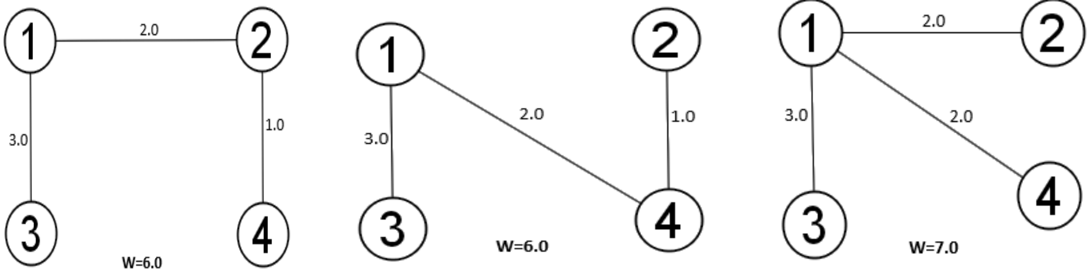
no "function", but rather the weight of each edge is stored in the data structure for that edge.

Definition: **Minimum Spanning Tree (MST):** Assume that G(V,E) is a connected, undirected graph. A graph G1 is said to be the MST of a graph G if G1 is a partial graph of G and represents an undirected tree containing all G vertices. The graph weight is the sum of all weights for all edge, and the MST of the graph Weighted is the tree with the lowest weight [6].
There are numerous scenarios in which the need arises to discover the MST. For instance, it can be applied to identify the lowest way cost for linking a collection of terminal units, be they cities, electrical terminals, computers, factories, or the means of communication, such as roads or wires. The solution involves extracting the MST from a graph where each edge is weighted to represent the connection cost. Additionally, finding the MST plays a crucial role in routing algorithms for determining the most efficient path.

Example: The following figure represents a weighted graph G and three minimum generative trees.



QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)



Prim algorithm initiates through selecting initial vertex and after that expands from the current tree's part by picking a new vertex and an edge during each iteration. This new edge links the new vertex to the existing tree [9]. While running the algorithm, we can categorize the vertices into three classes arranged as follows:

Tree vertices: encompass the vertices within the currently constructed tree.

Fringing vertices: are those that are adjacent to a tree vertex but aren't part of the tree.

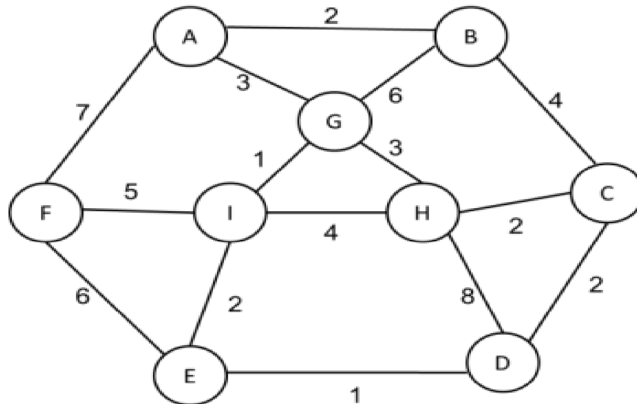
Invisible heads: are all other heads.

The pivotal step in the algorithm involves selecting a vertex from the fringe and the letter associated with that vertex. Since the weights are associated with the letters, the primary focus is on the letter rather than the vertex itself. The algorithm consistently opts for the letter with the lowest weight, moving from the top of the tree to the top of the fringe. The algorithm is described as follows:

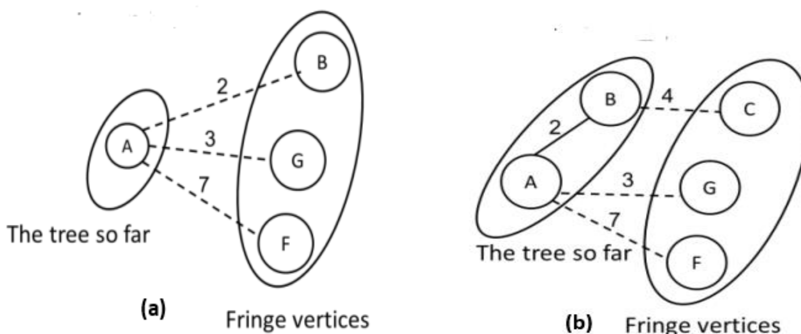
```
void primMST (G, n) //OUTLINE
        {
            Initialize all vertices as unseen.
            Select an arbitrary vertex s to start the tree; reclassify it as tree.
            Reclassify all vertices adjacent to s as fringe.
            while ( there are fringe vertices )
            {
                    Select an edge of minimum weight between a tree vertex
                     t and a fringe vertex v;
                     Reclassify v as tree; add the edge tv to the tree;
                     Reclassify all unseen vertices adjacent to v as fringe.
            }
        }
```

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

We will perform one iteration of Prim's algorithm to illustrate the idea. Assume we have the weighted graph shown in the following figure:



We assume that we will choose vertex A as the starting vertex. The vertices adjacent to vertex A are B, G, and F, so they form the vertices of the fringe as shown in Figure (a). In the first iteration of the While loop, we find that the edge (A, B) is the edge with the least weight to the vertex of the fringe. Therefore, vertex B is added to the tree, and the invisible vertices adjacent to vertex B enter the fringe, leading to the shape (b), and we notice that the edge (B, G) It is not shown in Figure (b) because the edges (A, G) are a better choice to reach vertex G because its weight is less. We also notice in the figure that the solid lines are the edges of a tree, while the dashed lines are the edges of the fringe vertices.

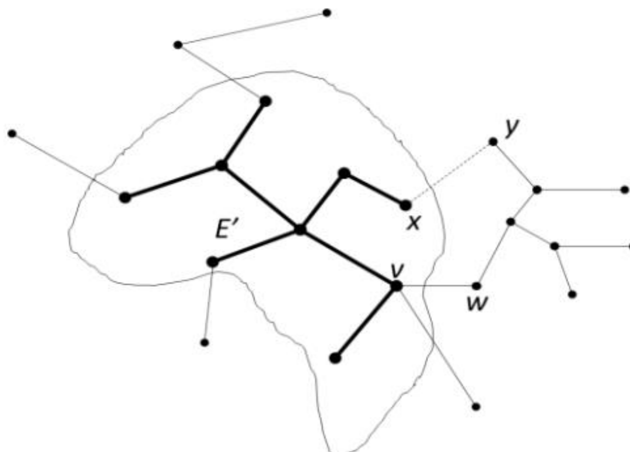


QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

Prim's algorithm is an example of what is called a greedy algorithm. Greedy algorithms are algorithms for optimal solution problems (i.e. problems in which we want the value of a quantity to be a local minimum or maximum) at each step of the algorithm in the hope that these choices will lead to an optimal solution. Comprehensive [10]. This method works in many cases but not in all cases. Prim's algorithm is one of the cases where this strategy works, and this can be easily demonstrated by induction. Since the algorithm starts without selecting any edge, the edges that will be initially selected are a subset of the edges present in the MST. Then we prove that after adding a smaller edge (i.e., the edge with the lowest weight) from the tree to the vertex of Fringe, the chosen set of edges remains contained in the MST.

Theorem 2: We assume that G(V,E,W) is a weighted connected graph, and that E' is a subset of the edges of the MST tree T(V,E_T) of the graph G. We assume that V' is a vertices set located on the edges of E'. If (x,y) is an edge with the least weight such that ($y \notin V'$) ($x \in V'$) then ($E' \cup (x, y)$) is a subset of MST.

The Proof: If the edge (x,y) is in the set ET, then the requirement is produced directly. Assume that (x,y) is not in ET. There is a path from x toy in tree T because the tree is connected. We assume that (v,w) is the first edge in this path, as there is one vertex of this edge that falls within the set of vertices V', and let this vertex be v. The following figure shows a partial MST tree from a graph G.



QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

Since v lies within V' and w is not within V', therefore by choosing the edge (x,y) then $W(x,y) \leq W(v,w)$ and therefore $W(T') \leq W(T)$ and so that T' are the MST tree and thus the theorem is proven.

Now we can summarize Prim algorithm for generating MST as follows:

Inputs: n is an integer number greater than or equal to 2, an un-directed weighted connected graph. The graph is showed as matrix W with two dimensional of size n x n, where, W[i][j] is the weight of the edge connecting vertex i and vertex j.

Outputs: edges' set F in a minimal generative tree of a graph.

Procedure:

```
void prim ( int n, const number W[ ][ ], set_of_ edges & F )
{
        index i , vnear;
        number min;
        edge e;
        index nearest [2..n];
        number distance [2..n];
        F = φ ;
        for (i = 2; i <= n; i+ +)
          {
             nearest [i]=1 // for all vertices, initialize v1 to
             distance[i] = W[1][i]; // be the nearest vertex in Y and
          }

        repeat ( n - 1 times )
          {
             min = ∞;
             for ( i = 2; i <= n; i++) // check each vertex for
                 if ( 0 ≤ distance [i] < min ) { // being nearest to Y.
                   min = distance [i];
                   vnear = i ;}
             e = edge connecting vertices indexed by vnear and
             nearest[vnear];
             add e to F;
            distance [vnear] = -1; // add vertex indexed by
            for ( i = 2; i <= n; i++) // vnear to Y.
                if ( W[i][vnear] < distance[i] ) { // for each vertex not in Y,
                   distance[i] = W[i] [vnear]; // update its distance from Y.
                   nearest[i] = vnear;}

          }
}
```

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

4.2 Analysis of Prim algorithm:

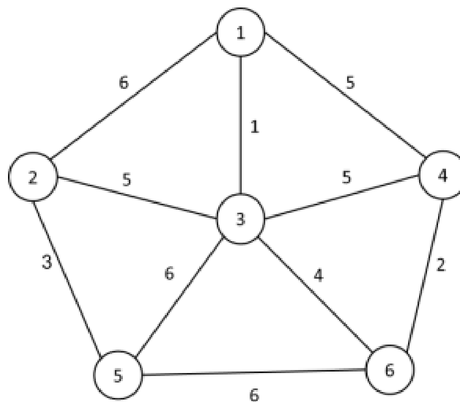
In Prim algorithm, it's important to observe that there are two distinct loops, each comprising (n-1) iterations. The execution of instructions within each loop can be regarded as a single execution of the fundamental process. With each loop having (n-1) iterations, the complexity can be expressed as follows:

$$T(n) = 2 \times (n-1) \times (n-1) \in O(n^2)$$

4.3 Kruskal algorithm to find MST:

Assume that, we have an undirected weighted graph G(V,E,W). The fundamental concept behind the Kruskal algorithm is that, at each stage, it chooses the edge with the lowest weight that has not been selected yet from any location in the graph. However, it refrains from selecting any edge which creating a cycle when combined with chosen edges [14].

<u>Example:</u> The Kruskal algorithm will be applied to discover MST for the following graph:

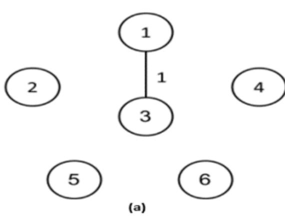


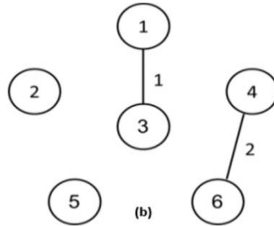
First, the edges of the graph G are arranged in ascending order according to their weights:

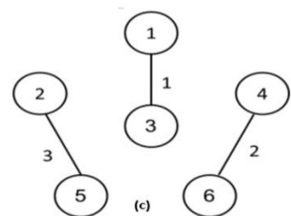
QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

Edge	(1,3)	(4,6)	(2,5)	(3,6)	(1,4)	(3,4)	(2,3)	(1,2)	(3,5)	(5,6)
Weight	1	2	3	4	5	5	5	6	6	6

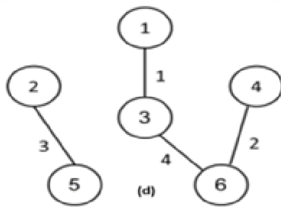
The edges with the least weight are chosen incrementally, and the edge is ignored if it forms a cycle, even if it represents the lowest weight. Accordingly, the results are as shown in the following figures:

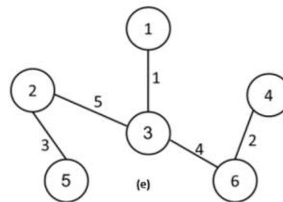






(a) **(b)** **(c)**





(d) **(e)**

We note that the weight of the graph G is W(G) = 43, while the weight of the resulting tree is W(F) = 15. We also note that the resulting tree contains 6 vertices and 5 edges.

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

Therefore, in general, any tree consisting of n vertices and (n-1) edges, if any edge is added to it, will lead to the formation of a cycle.

In general, Kruskal algorithm to finding MST begins by generating distant sub-sets of the set of vertices V. Then the algorithm tests the edges successively based on their weights. For letters of equal weight, they are chosen randomly provided that they do not form a cycle. If there are two vertices connecting through an edge in separate sub-sets, it is included in the edge set F, and the two sub-sets are merged into a single set. This procedure is iterated until all sub-sets become a single set [14].

We can summarize Kruskal algorithm for generating an MST as follows:

Inputs: $n \geq 2$, integer positive m, and an undirected weighted connected graph with vertices (n) and edges (m).

Outputs: edges set F in a minimal generative tree of a graph.

Procedure:

```
void kruskal ( int n, int m, set_of_edges E, set_of_edges& F)
    {
      index i, j;
      set_pointer p,q;
      edge e;
      sort the m edges in E by weight in nondecreasing order;
      F = φ;
      initial (n); // initialize n disjoint subsets.
      while ( number of edges in F is less than n-1 )
        {
           e= edge with least weight not yet considered ;
           i, j = indices of vertices incident upon e ;
           p = find(i) ;
           q = find(j) ;
           if ( ! equal ( p, q ) )
            {
              merge ( p, q);
              add e to F;
            } // end if
        } // end while
    }
```

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

4.4 Analysis of Kruskal algorithm:

There are three points in this algorithm that must be taken into consideration:

1) The time required to arrange the edges: The Merge sort algorithm can be used to arrange the edges, and its degree of complexity is O(m log m). Therefore, the degree of complexity for this step is: $W(m) \in O(m \log m)$.
2) The time spent within the while loop is predominantly determined by the processing of distant sets, as everything else remains constant. In the worst-case scenario, the loop iterates through all edges before exiting, which occurs m times. Consequently, the complexity of this stage is determined by: $W(m,n) \in O(m + n \log n)$.
3) The time required to give initial values for n-distant groups: The degree of complexity required to give initial values is:

$$T(n) \in O(n)$$

Since m is greater or equal to (n-1), the process of sorting and processing distant groups overcomes the instatement time, which means that:

$$W(m,n) \in O(m + n \log n + m \log m) = O(m \log m)$$

It might appear that the worst-case scenario is independent of n, but in this worst-case situation, every vertex can be connected to every other vertex, leading to:

$$m = \frac{n(n-1)}{2} \in O(n^2)$$

Hence, we can compose the most pessimistic scenario as follows:

$$W(m,n) \in O(n_2 \log n^2) = O(n^2 2 \log n) = O(n^2 \log n)$$

4.5 Comparison between Prim and Kruskal algorithms:

We obtained the following two-time complexity scores:

$$T(n) \in O(n^2) \qquad \text{Pr}\mathit{ime}\ A\text{lg}\mathit{orithm}$$

$$W(m,n) \in O(m \log m) \quad \mathit{Kruskal}\ A\text{lg}\mathit{orithm}$$

In any connected graph, the relationship always holds:

$$n - 1 \le m \le \frac{n(n-1)}{2}$$

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

Therefore, for any graph whose number of edges m is close to the minimum (i.e. the graph is sparse), the complexity of Kruskal's algorithm is O(nlogn), which means that it becomes faster than Prim's algorithm. If the number of edges of the graph is close to the highest (the graph is highly connected), the complexity of the Kruskal algorithm is $O(n^2 \log n)$, which means that the Prim algorithm is faster.

5. Conclusions:

In conclusion, this research has offered a comprehensive overview of key graph algorithms. We have examined the underlying principles, advantages, and real-world applications of each algorithm. DFS and BFS are fundamental tools for traversing graphs, while Prim and Kruskal algorithms provide elegant solutions for finding minimum spanning trees. These algorithms have been instrumental in solving a wide range of problems, from finding efficient routes in transportation networks to optimizing resource allocation in various industries. By understanding the intricacies of these algorithms, researchers and practitioners can apply them effectively to address complex problems in their respective fields. This research serves as a stepping stone for further exploration and application of graph theory in practical scenarios. The contribution of our research lies in providing a comprehensive overview and analysis of key graph algorithms, including DFS, BFS, Prim algorithm, and Kruskal algorithm. Our research contributes to the field of computer science and graph theory by:

1. Providing an educational resource: Our research offers an educational resource that can help students, researchers, and professionals understand these fundamental graph algorithms, their principles, and their applications.
2. Comparative analysis: This comparison can help readers understand the strengths and weaknesses of each algorithm and when to use them in different scenarios.

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

3. Contribution to the graph theory community: Our work contributes to the ongoing discussions and research in the field of graph theory, helping researchers and students build a strong foundation in this area.

In summary, our research contributes by offering a comprehensive and analytical exploration of essential graph algorithms, aiding in the dissemination of knowledge and fostering a better understanding of these algorithms in both academic and practical contexts.

References:

[1] Guttoski, P. B., Sunye, M. S., & Silva, F. (2007, September). Kruskal's algorithm for query tree optimization. In 11th International Database Engineering and Applications Symposium (IDEAS 2007) (pp. 296-302). IEEE.

[2] Akanmu, T. A., Olabiyisi, S. O., Omidiora, E. O., Oyeleye, C. A., Mabayoje, M. A., & Babatunde, A. O. (2010). Comparative study of complexities of breadth-first search and depth-first search algorithms using software complexity measures. In Proceedings of the World Congress on Engineering (Vol. 1).

[3] Lattanzi, S., Moseley, B., Suri, S., & Vassilvitskii, S. (2011, June). Filtering: a method for solving graph problems in mapreduce. In Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures (pp. 85-94).

[4] Yang, S., Yuan, L., Lai, Y. C., Shen, X., Wonka, P., & Ye, J. (2012, August). Feature grouping and selection over an undirected graph. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 922-930).

[5] van de Pol, J. C. (2015, June). Automated verification of nested DFS. In International Workshop on Formal Methods for Industrial Critical Systems (pp. 181-197). Cham: Springer International Publishing.

[6] Singh, M., & Lau, L. C. (2015). Approximating minimum bounded degree spanning trees to within one of optimal. Journal of the ACM (JACM), 62(1), 1-19.

[7] Banerjee, N., Chakraborty, S., & Raman, V. (2016, July). Improved space efficient algorithms for BFS, DFS and applications. In International Computing and Combinatorics Conference (pp. 119-130). Cham: Springer International Publishing.

[8] Dey, A., & Pal, A. (2016). Prim's algorithm for solving minimum spanning tree problem in fuzzy environment. Annals of Fuzzy Mathematics and Informatics, 12(3), 419-430.

[9] Latha, N. R., Shyamala, G., & Prasad, G. R. (2017, March). Exploring the parallel implementations of the three classical MST algorithms. In 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT) (pp. 340-346). IEEE.

[10] Furqan, M., Mawengkang, H., Sitompul, O. S., Siahaan, A., & Nasution, N. (2018). A review of prim and genetic algorithms in finding and determining routes on connected weighted graphs. Int. J. Civ. Eng. Technol, 9(9), 1755-1765.

[11] Gaihre, A., Wu, Z., Yao, F., & Liu, H. (2019, June). XBFS: eXploring runtime optimizations for breadth-first search on GPUs. In Proceedings of the 28th International symposium on high-performance parallel and distributed computing (pp. 121-131).

[12] Dey, A., Broumi, S., Son, L. H., Bakali, A., Talea, M., & Smarandache, F. (2019). A new algorithm for finding minimum spanning trees with undirected neutrosophic graphs. Granular Computing, 4, 63-69.

[13] Palanisamy, V., & Vijayanathan, S. (2020, October). A novel agent based depth first search Algorithm. In 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA) (pp. 443-448). IEEE.

[14] Rachmawati, D., & Pakpahan, F. Y. P. (2020, July). Comparative analysis of the Kruskal and Boruvka algorithms in solving minimum spanning tree on complete graph. In 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA) (pp. 55-62). IEEE.

[15] Rathnayake, B. R. M. S. R. B., Marzuk, H., Senadheera, R. I. A., Vijeyakumar, S., & Abeygunawardhana, P. K. (2021, August). Analysis of Searching Algorithms in Solving Modern Engineering Problems. In 2021 10th International Conference on Information and Automation for Sustainability (ICIAfS) (pp. 123-128). IEEE.

[16] Erciyes, K., & Erciyes, K. (2021). Trees and Traversals. Discrete Mathematics and Graph Theory: A Concise Study Companion and Guide, 243-262.

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (1), Spring 2025
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

[17] Ansari, M. N., Singh, R. K., & Gupta, A. K. (2021, December). Optimal PMU Positioning Considering the Effect of Zero Injection Buses in Advanced Grid Monitoring. In 2021 IEEE 2nd International Conference on Electrical Power and Energy Systems (ICEPES) (pp. 1-5). IEEE.

[18] Das, R., & Soylu, M. (2023). A key review on graph data science: The power of graphs in scientific studies. Chemometrics and Intelligent Laboratory Systems, 104896.

لێکۆڵینەوەیەکی شیکاری ئەلگۆریتمەکانی گراف: تێڕوانینێکی گشتی بۆ تیۆری گراف

پـوختـە:

تیۆری گراف لقێکی بنەڕەتی بیرکارییە کە بەکارهێنانی جۆراوجۆری هەیە لە زانستی کۆمپیوتەر و بوارە جیاوازەکانی تردا. ئەم توێژینەوەیە تێڕوانینێکی گشتی شیکاری بۆ ئەلگۆریتمەکانی گرافیکی بنەڕەتی دەدات، کە تیایدا سەرنج دەخاتە سەر گەڕانی قووڵی-یەکەم، گەڕانی فراوانی-یەکەم، ئەلگۆریتمەکەی پریم و ئەلگۆریتمەکەی کروسکاڵ بۆ دۆزینەوەی کەمترین دارە فراوانەکان. لە ڕێگەی گەڕانێکی قووڵی ئەم ئەلگۆریتمانە، ئەم توێژینەوەیە ئامانجیەتی ڕۆشنایی بخاتە سەر بنەما و سوود و بەکارهێنانەکانیان. ئەو تێگەیشتنانەی لەم لێکۆڵینەوەیەدا بەدەست هاتوون دەتوانن بەنرخ بن لە چارەسەرکردنی کێشەکانی جیهانی ڕاستەقینە کە تۆڕ و باشکردن لەخۆدەگرن.

دراسة تحليلية لخوارزميات الرسم البياني: نظرة عامة على نظرية الرسم البياني

الملخص:

نظرية الرسم البياني هي فرع أساسي من الرياضيات ولها تطبيقات متنوعة في علوم الكمبيوتر ومختلف المجالات الأخرى. يقدم هذا البحث نظرة عامة تحليلية لخوارزميات الرسم البياني الأساسية، مع التركيز على البحث عن العمق أولاً، والبحث عن العرض أولاً، وخوارزمية بريم، وخوارزمية كروسكال للعثور على الحد الأدنى من الأشجار الممتدة. ومن خلال التعمق في هذه الخوارزميات، تهدف هذه الدراسة إلى تسليط الضوء على مبادئها ومزاياها وتطبيقاتها. يمكن أن تكون الأفكار المكتسبة من هذا البحث ذات قيمة في حل مشكلات العالم الحقيقي التي تتضمن الشبكات والتحسين.