Graph-Based Design of the Shortest Shaded Path Connecting All Buildings at Institut Teknologi Bandung Campus of Jatinangor

Ahmad Zaky Robbani - 13524045

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jalan Ganesha 10 Bandung

E-mail: ahmadzakyrobbani@gmail.com , 13524045@std.stei.itb.ac.id

***Abstract*—Institut Teknologi Bandung (ITB), renowned for its academic excellence and vibrant campus life, is designed to be a walkable environment that encourages pedestrian movement across its facilities. However, the tropical climate and high daytime temperatures in Indonesia often make walking uncomfortable, particularly under direct sunlight. To support sustainability and enhance comfort of students, staff, and visitors, it is essential to integrate shaded pathways throughout the campus. This study uses Kruskal's algorithm and depth-first search to compute minimum spanning tree as a plan to connect every building in ITB campus of Jatinangor by shaded pathways.**

Keywords—graph; shaded pathways; institut teknologi bandung; minimum spanning tree; kruskal

I. Introduction

Institut Teknologi Bandung (ITB) is one of Indonesia's most prestigious universities. With a growing academic community, ITB now operates across multiple campuses, including its historic Ganesha campus in Bandung and the newer Jatinangor campus in Sumedang. The Jatinangor campus serves as the main campus for freshmen students during Tahap Persiapan Bersama (TPB) across various study programs. Designed with a walkable layout to encourage movement and interaction among buildings, the campus faces challenges typical of a tropical setting—particularly high temperatures and direct sunlight. These conditions highlight the urgent need for shaded pedestrian pathways to ensure student comfort and accessibility while promoting sustainable campus mobility.

Graph theory is a field of discrete mathematics that studies graphs—structures used to represent discrete objects and the relationships between them. A graph consists of a set of vertices (nodes) connected by edges (links). This concept was first formalized by Leonhard Euler in 1736 through the famous Königsberg Bridge Problem, which laid the foundation of graph theory. Since then, graphs have become essential tools for modeling and solving real-world problems in various fields.

This study is implementing graph theory to plan potential shaded pedestrian pathways to cover each and all building within ITB campus of Jatinangor. Using minimum spanning tree, the "cheapest" way to build a walkable campus can be acquired.

II. Theoretical Foundation

A. Graph

Graph is a mathematical structure which consists of a non-empty set of vertices and a set of edges. There are many types of graph depending on its edge component and its direction:

1. Simple graph : contains no loop and no parallel edge

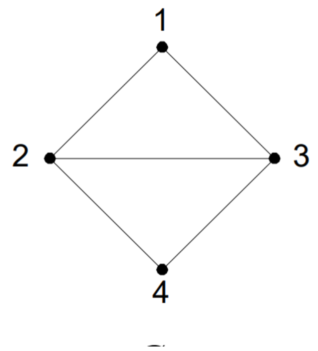


Figure 2.1 Simple graph

(source: informatika.stei.itb.ac.id)

2. Multi-graph : contains parallel edge

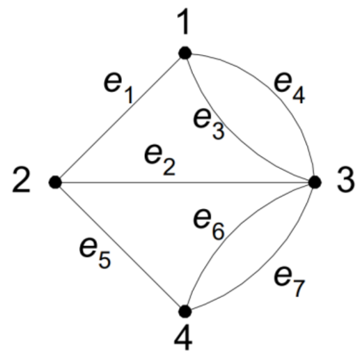
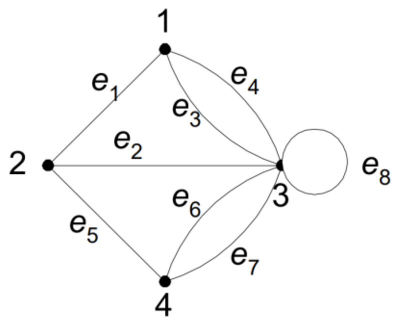


Figure 2.2 Multi-graph

(source: informatika.stei.itb.ac.id)

3. Pseudo graph : contains loop



G_3

Figure 2.3 Pseudo graph

(source: informatika.stei.itb.ac.id)

4. Undirected graph : each edge is mutual

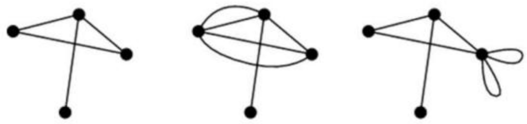


Figure 2.4 Undirected graphs

(source: informatika.stei.itb.ac.id)

5. Directed graph : each edge has direction

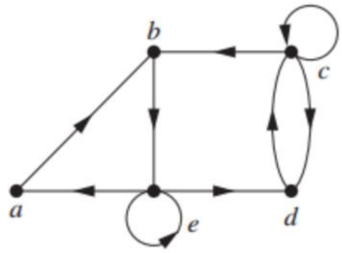


Figure 2.5 Directed graph

(source: informatika.stei.itb.ac.id)

There are many important concepts about graph. Some concepts which will be used in this research are

1. Path : a sequence of edges connecting a series of vertices
2. Cycle : a path that starts and ends at the same vertex
3. Connected graph : a graph that has path connecting each and every vertex

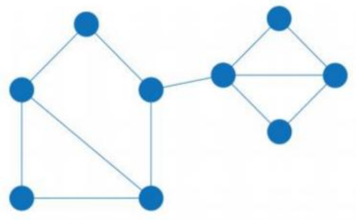


Figure 2.6 Connected graph

(source: informatika.stei.itb.ac.id)

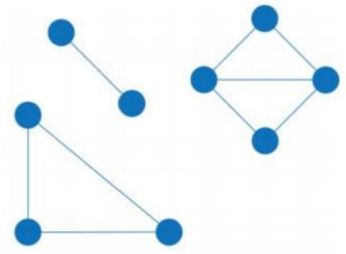


Figure 2.7 Disconnnected graph

(source: informatika.stei.itb.ac.id)

4. Subgraph : a graph which consists of a subset of edges and vertices set of parent graph

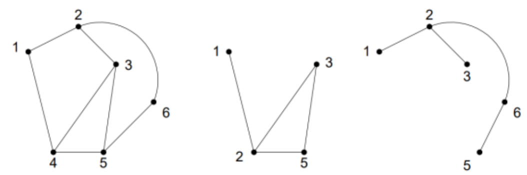


Figure 2.8 A graph and its two subgraphs

(source: informatika.stei.itb.ac.id)

5. Spanning subgraph : a subgraph that includes all vertices of the original graph

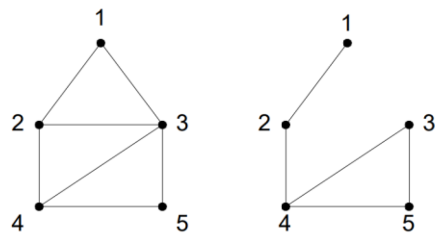


Figure 2.9 A graph and its spanning subgraph

(source: informatika.stei.itb.ac.id)

6. Weighted graph

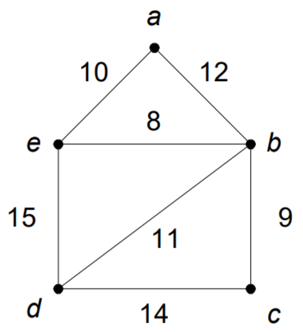


Figure 2.10 Weighted graph

(source: informatika.stei.itb.ac.id)

B. Adjacency List

Adjacency list is a way of representing graph. Adjacency list consists of each vertex and its neighboring vertices.

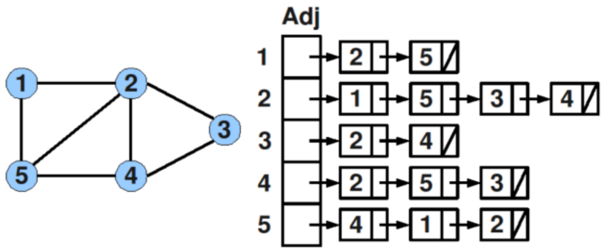


Figure 2.11 Adjacency list structure

(source: informatika.stei.itb.ac.id)

C. Tree

A tree is a special type graph which is undirected, connected graph and does not contain cycles.

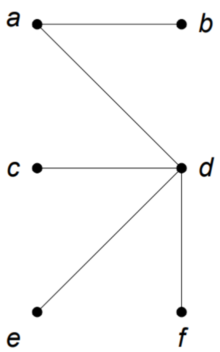


Figure 2.12 A tree
(source: informatika.stei.itb.ac.id)

D. Minimum Spanning Tree (MST)

A spanning tree is a subgraph of a connected graph that includes all the vertices and is a tree. A minimum spanning tree is a spanning tree of a weighted graph with the smallest possible total weight.

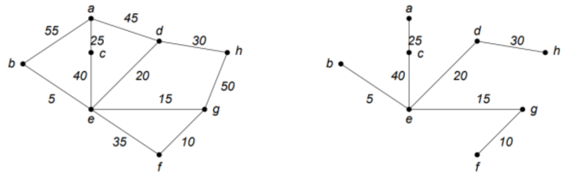


Figure 2.15 A weighted graph and its minimum spanning tree
(source: informatika.stei.itb.ac.id)

E. Kruskal's Algorithm

Kruskal's algorithm is an algorithm used to search for minimum spanning tree of a graph. Kruskal's algorithm consists of 4 steps:

1. Sort all edges in the graph in ascending order of weight.
2. Initialize an empty set T to store the MST edges.
3. Iterate through the sorted edges. For each edge, check if adding it to T would form a cycle. If no cycle is formed, add the edge to T, else skip the edge.
4. Repeat until T contains exactly n-1 edges, where n is the number of vertices.

F. Depth-First Search (DFS)

Depth-first search (DFS) is a graph traversal algorithm that explores as far as possible along each branch before backtracking. It is used to visit all vertices in a systematic, deep-first manner. DFS can be implemented to search for loops within a graph

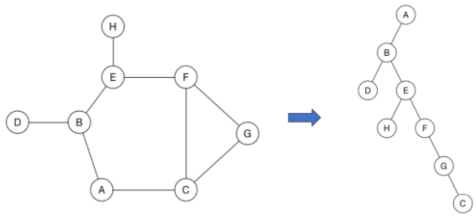


Figure 2.17 A graph and its DFS tree illustration
(source: informatika.stei.itb.ac.id)

III. Methodology

A. Assumptions

1. Two places are defined as within the same building if and only if there exists a shaded path that connects the two.
2. Shaded pathways which are already built will not be deconstructed. Therefore, buildings connected by built pathways will be defined as one building complex.

3. Planned pathways should take built road into consideration and not just a straight line between two places.
4. Planned pathways are the shortest path between two disconnected building complexes.

B. Graph representation

Based on the assumptions, connected buildings are defined as one building complex. This problem will be presented as a weighted graph problem. Each building complex will be represented by a node with a given name and code. Each potential pathway will be represented by an edge connecting two nodes with weight correspondent with the distance or length of the path.

C. Graph construction

The graph will be constructed using Google Earth. According to the assumptions, planned pathways are the shortest path possible. The distance or length of each path will be measured using Google Earth's path feature which measures up to centimeters precision.

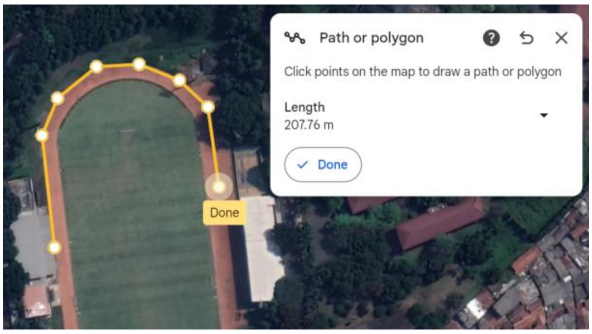


Figure 3.1 Google earth's path feature

(source: earth.google.com)

D. Technical implementation

To compute the data, python 3.13.3 will be used. Graphs will be represented by adjacency lists which will be realized using built-in dictionary structure. The building complex's code will be the key and a list of tuples of adjacent building complex and their distance.

```python
adjacency_list = {0 : [],
                  1 : [(2, 10.1), (3, 1.71)],
                  2 : [(1, 10.1)],
                  3 : [(1, 1.71)]
                  }
```

Figure 3.2 Adjacency list representation of a graph

(source: author's screenshot)

Kruskal's algorithm will be implemented using the steps below

1. Create a sorted list of edges from the graph
2. Initialize an empty dictionary (each vertex's value is an empty list)
3. Iterate through the sorted edges. Check if adding the edge forms a cycle using DFS. If it does not form a cycle, the edge is added to the dictionary's values.

```python
# Create sorted list of edges (tuple of source, destination, and distance)
def sortEdges(graph : dict) -> list:
    edges = []
    for source in graph.keys():
        for adj in graph.get(source):
            dest = adj[0]
            dist = adj[1]
            edge = set([source, dest])
            if (edge, dist) not in edges:
                edges.append((edge, dist))
    return sorted(edges, key=lambda x: x[1])
```

Figure 3.3 sortEdges subprogram

(source: author's screenshot)

```python
# DFS utility function for cycle-finding
def dfs(graph : dict, visited : set, parent : int, v : int) -> bool:
    visited.add(v)
    edges = graph[v]
    for edge in edges:
        if edge[0] not in visited:
            if dfs(graph, visited, v, edge[0]):
                return True
        elif parent != edge[0]:
            return True
    return False

# Function to check for cycle within a graph
def containCycle(graph : dict) -> bool:
    visited = set()
    for vertex in graph.keys():
        if vertex not in visited:
            if dfs(graph, visited, -1, vertex):
                return True
    return False
```

Figure 3.4 dfs and containCycle subprogram

(source: author's screenshot)

```python
# Kruskal algorithm on a graph, returns a tuple of minimum weight spanning tree and its weight
def kruskal(graph : dict) -> tuple:
    # Create empty graph with same vertices as the original graph
    spanning_tree = dict()
    for vertex in graph.keys():
        spanning_tree[vertex] = []
    total_distance = 0.0

    # Sort edges by distance ascending
    edges = sortEdges(graph)

    # Add edge if it doesn't create cycle, starting from the smallest distance
    for edge in edges:
        v1 = min(edge[0])
        v2 = max(edge[0])
        temp1 = spanning_tree.get(v1)
        temp1.append((v2, edge[1]))
        temp2 = spanning_tree.get(v2)
        temp2.append((v1, edge[1]))
        spanning_tree[v1] = temp1
        spanning_tree[v2] = temp2

        if (containCycle(spanning_tree)):
            temp1.remove((v2, edge[1]))
            temp2.remove((v1, edge[1]))
            spanning_tree[v1] = temp1
            spanning_tree[v2] = temp2
        else:
            total_distance += edge[1]

    return (spanning_tree, total_distance)
```

Figure 3.5 Kruskal's algorithm implementation

(source: author's screenshot)

IV. Results and Discussion

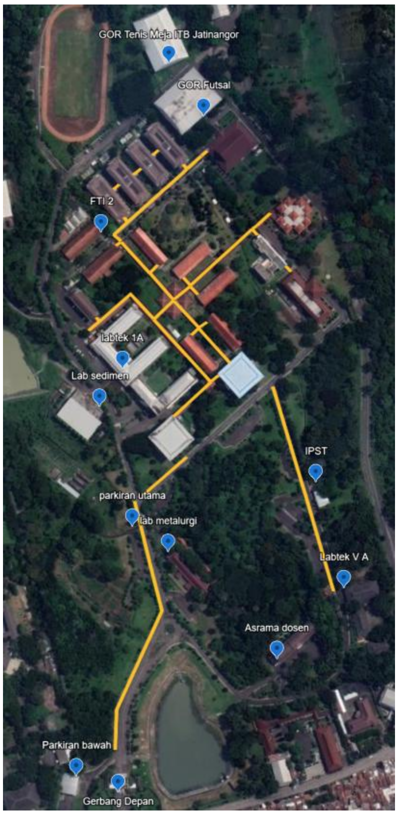


Figure 4.1 Existing pathways and building complexes of ITB campus of Jatinangor

(source: earth.google.com)

Table 4.1 Existing building complexes

Code	Complex Name	Building Members
0	Main Gate	Main Gate
1	Car Park	Car Park
2	Entrance Shaded Path	Entrance Shaded Path
3	Lecturer's Dormitory	Lecturer's Dormitory
4	Reading Room Shaded Path	Reading Room and shaded pathway near it
5	Labtek 5	Labtek 5
6	IPST	IPST
7	Metallurgy Lab	Metallurgy Lab
8	Motorcycle Park	Motorcycle Park
9	GKU 2 Shaded Path	Shaded path near GKU 2
10	Labtek 1B	Labtek 1B
11	Labtek 1A	Labtek 1A
12	Sedimentation Lab	Sedimentation Lab
13	Rectorate Parking Lot	Rectorate Parking Lot
14	Main Buildings	GKU 2, GKU 1, Canteen, Musala Al-Wasath, Gedung A, Gedung C, Gedung D, Gedung E, South FTI Building, TB 5 Dormitory, TB 4 Dormitory, TB 3 Dormitory, TB 2 Dormitory, TB 1 Dormitory, SBM Building, GKU 3, KOICA Building, Rectorate Building
15	North FTI Building	North FTI Building
16	GOR Futsal	GOR Futsal
17	GOR Tenis Meja	GOR Tenis Meja

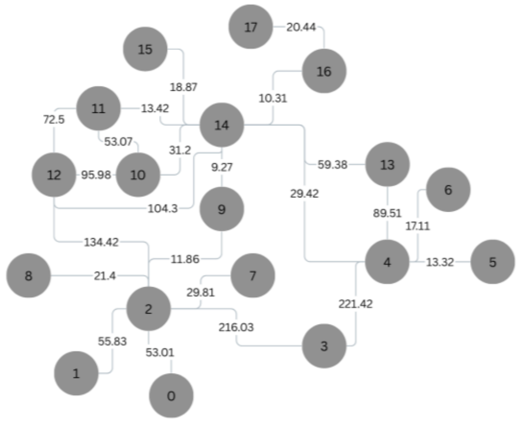
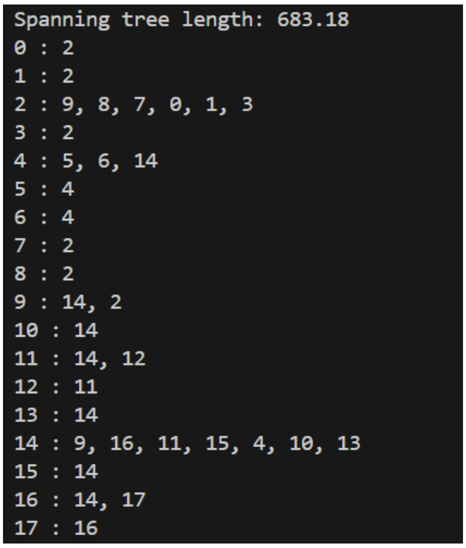


Figure 4.2 Graph of Building Complex in ITB campus of Jatinangor



```
Spanning tree length: 683.18
0 : 2
1 : 2
2 : 9, 8, 7, 0, 1, 3
3 : 2
4 : 5, 6, 14
5 : 4
6 : 4
7 : 2
8 : 2
9 : 14, 2
10 : 14
11 : 14, 12
12 : 11
13 : 14
14 : 9, 16, 11, 15, 4, 10, 13
15 : 14
16 : 14, 17
17 : 16
```

Figure 4.3 Kruskal algorithm's result

(source: author's screenshot)

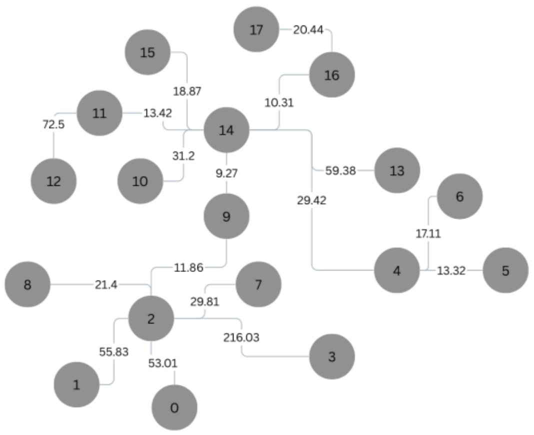


Figure 4.4 Spanning Tree of Building Complex in ITB campus of Jatinangor

Using the program that has been created, the minimum spanning tree of the graph is computed. The adjacency list of the spanning tree is shown in Fig. 4.3. The spanning tree consists of 17 edges with a total length of 683.18 m.

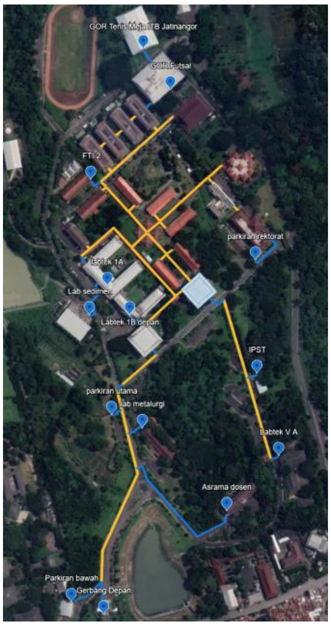


Figure 4.5 Final Map of Shaded Pathways

(source: earth.google.com)

It shall be kept in mind that several factors may cause the results to be inaccurate, such as human error in measurement and limitations of the measurement tool (Google Earth). Lastly, the calculations performed do not consider geological/geophysical/other related aspects. Further research will be necessary for planning the construction of the shaded pathways.

V. Conclusion

Graph theory has been a successful way to approach the problem of planning the shortest additional shaded pathways for ITB campus of Jatinangor. To connect each and every building within the campus, a total of 17 pathways with combined length of 683.18 m should be constructed. However, there might be measurement errors due to human error or limitations of measurement tool. The calculations also do not take geological/geophysical aspect and human comfortness. Therefore, further research will be necessary to plan the actual construction of shaded pathways.

VI. Appendix

- Complete source code of the program used to calculate minimum spanning tree: https://github.com/SleepyySloth/shaded-path-MST.git
- Google earth project for data collecting: https://earth.google.com/earth/d/1wTYQ_UhtElWAwb1gBe1Jh1u5YZUj8pYh?usp=sharing

VII. Acknowledgment

The author expresses his deepest gratitude to God Almighty for the strength, clarity, and perseverance granted throughout the process of writing this paper. The author would also like to sincerely thank Mr. Rinaldi Munir, whose dedication and expertise in teaching Discrete Mathematics have been critical in shaping the author's understanding of this subject. His guidance has not only deepened the author's knowledge but also inspired a greater appreciation for the logical beauty of mathematics. The author hopes that the insights and understanding gained from this paper can be meaningful and beneficial to others.

References

[1] R. Munir, Graf (Bag. 1), IF1220 Discrete Mathematics, School of Electrical Engineering and Informatics, Institut Teknologi Bandung, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf. [Accessed: Jun. 20, 2025].

[2] R. Munir, Graf (Bag. 2), IF1220 Discrete Mathematics, School of Electrical Engineering and Informatics, Institut Teknologi Bandung, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf. [Accessed: Jun. 20, 2025].

[3] R. Munir, Pohon (Bag. 1), IF1220 Discrete Mathematics, School of Electrical Engineering and Informatics, Institut Teknologi Bandung, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/23-Pohon-Bag1-2024.pdf. [Accessed: Jun. 20, 2025].

[4] R. Munir and N. U. Maulidevi, Breadth/Depth First Search (Bag. 1), IF2211 Algorithmic Strategies, School of Electrical Engineering and Informatics, Institut Teknologi Bandung, 2025. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/13-BFS-DFS-(2025)-Bagian1.pdf. [Accessed: Jun. 20, 2025].

[5] W3Schools, "DSA Graph Cycle Detection Algorithm," [Online]. Available: https://www.w3schools.com/dsa/dsa_algo_graphs_cycledetection.php. [Accessed: Jun. 20, 2025].

Declaration

With this, I hereby declare that this paper is my own writing, not a copy, or a translation of someone else's paper, and not a plagiarism.

Bandung, 20 Juni 2025

Ahmad Zaky Robbani - 13524045