Fire Rescue Route Optimization: Application of Prim Algorithm on Drones

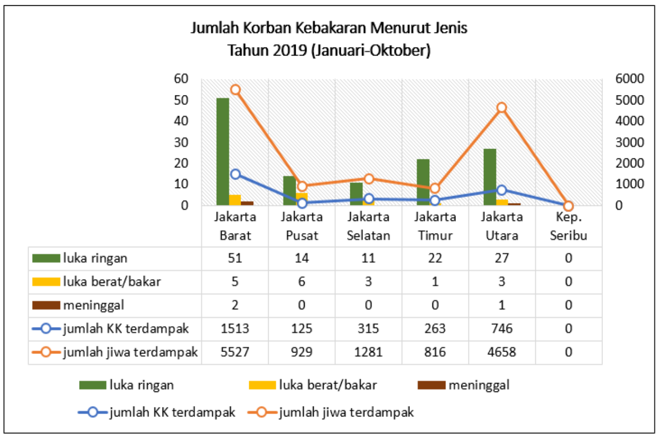
Albert - 13522081
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13522081@std.stei.itb.ac.id

***Abstract*—This paper addresses the imperative need for an efficient solution in rescuing individuals trapped within a building during a fire emergency. To optimize the rescue route, this paper propose a hypothetical scenario leveraging drone technology and applying the Prim algorithm to determine the minimum spanning tree. The classic Prim's Algorithm is employed to compute the minimum spanning tree, and the edge weights are derived from a combination of distance and probability parameters. By intricately considering these factors, the solution aims to identify the most efficient route for drone-based search and rescue operations within a single-level building. This paper contributes to the discourse on fire rescue optimization, emphasizing the integration of drone technology and algorithmic decision-making for effective emergency response.**

***Keywords*—Drone, Fire rescue route, Minimum spanning tree, Prim's Algorithm.**

I. Introduction

Indonesia contends with a recurring challenge that transcends geographical boundaries—the frequent incidence of fire-related emergencies that pose a substantial threat to public safety. Tragically, these incidents have been marked by severe consequences, leading to significant loss of life as individuals struggle to evacuate buildings swiftly and efficiently. In recent occurrences, the grim reality of these challenges has been starkly evident, underscoring the urgent need for innovative approaches to enhance fire rescue operations.



	Jakarta Barat	Jakarta Pusat	Jakarta Selatan	Jakarta Timur	Jakarta Utara	Kep. Seribu
🟩 luka ringan	51	14	11	22	27	0
🟨 luka berat/bakar	5	6	3	1	3	0
🟫 meninggal	2	0	0	0	1	0
jumlah KK terdampak	1513	125	315	263	746	0
jumlah jiwa terdampak	5527	929	1281	816	4658	0

Fig 1 Number of fire victims by type (source [4])

A notable aspect that make the gravity of these emergencies is the time-sensitive nature of fire incidents, particularly in densely populated areas or large, complex buildings. The long duration required to extinguish fires in substantial structures further magnifies the risks to occupants, necessitating expedited and precise rescue strategies. Consequently, addressing the intricacies of rescue operations in the context of formidable building structures is important to minimizing casualties.

Modern building architectures, characterized by their complexity and often labyrinthine layouts, pose a considerable obstacle to conventional rescue efforts during fire emergencies. The challenge lies not only in the identification of evacuation routes but also in locating individuals who may be trapped within the building. The clear necessity for a more advanced and flexible approach to handle these complexities shows that it's time to improve how we do things in fire rescue operations.

Recognizing this imperative, this paper endeavors to present a comprehensive solution grounded in the application of the Prim algorithm for Fire Rescue Route Optimization. This innovative approach integrates spatial distance and the probability of individuals being present in specific rooms to compute a Minimum Spanning Tree.

This paper represents a step towards mitigating the profound human cost of fire-related incidents in complex building environments. As we delve into the intricacies of fire emergencies, the proposed methodology promises to revolutionize fire rescue operations by offering a more efficient and adaptable approach, thereby reducing the time it takes to identify, locate, and evacuate individuals in buildings.

II. Basic Theory

The concept of the Minimum Spanning Tree, the main topic of this paper, is a subset within the broader and intricate topic graphs. To gain a deeper understanding of the Minimum Spanning Tree, it's imperative to dive into the foundational knowledge of graphs. Consequently, this section will provide an educational exploration of the fundamental principles of graphs.

A. Graph

A graph is a discrete set of vertices, each representing a distinct point, and edges that bridge the connection between these vertices. Formally, a graph can be mathematically described as a pair,

$$GG = (VV, EE)$$

where G is the graph itself, V a non-empty set of vertices v1, v2, ..., vn, and E a set of edges e1, e2, ..., en which connects a pair of vertices.

A graph, at a minimum, needs one vertex, but edges don't share the same constraint; vertices in a graph might not be connected by any edges. Multiple edges can link the same pair of vertices in a graph, and loops, where an edge connects both ends to the same vertex, are also possible. Furthermore, edges in a graph may have directions, indicating their starting and ending points. A graph's classification hinges on these features, encompassing multiple edges, loops, or directional edges. Graphs fall into two main categories based on these characteristics: a simple graph, such as the one depicted in Fig. 2a, which lacks multiple edges and loops, and an unsimple graph, which allows for the presence of multiple edges and/or loops.

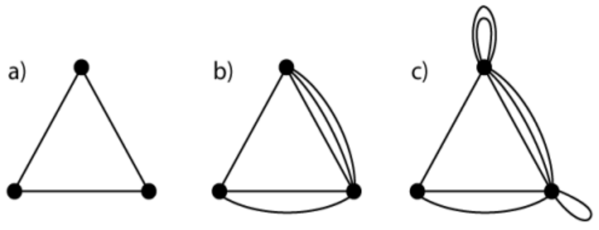


Fig. 2 (a) simple graph, (b) multigraph, and (c) pseudograph (source : [1])

Unsimple graphs are characterized by the presence of loops or multiple edges connecting the same vertices. Unsimple graphs can be further categorized into two distinct types:

a. Multipgraphs:

Multipgraphs are graphs featuring multiple edges that connect the same vertices. If there are m distinct edges associated with the same unordered pair of vertices $\{u, v\}$, then $\{u, v\}$ is considered an edge with a multiplicity of m.

b. Pseudographs:

Pseudographs are graphs that may encompass loops and potentially multiple edges connecting the same pair of vertices or a vertex to itself. Loops, in this context, refer to edges that connect a vertex directly to itself.

Graphs, based on the orientation of their edges, can be classified into two distinct types:

a. Undirected Graph:

In an undirected graph, each edge lacks a specific direction, signifying that it is impossible to determine the starting and ending points of an edge.

b. Directed Graph:

On the contrary, a directed graph (V, E) is comprised of a nonempty set of vertices V and a set of directed edges E. Each directed edge is linked to an ordered pair of vertices, with the edge represented by the ordered pair $\{u, v\}$ signifying a unidirectional connection from u to v.

The differentiation between these types is exemplified in Fig. 3a, where two edges between vertices b and d showcase the absence of direction in an undirected graph. Conversely, Fig. 3b illustrates a directed graph, emphasizing the distinct direction of edges between vertices a and e.

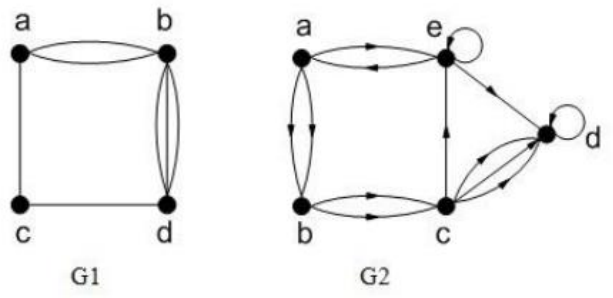


Fig. 3 a)undirected graph, b) directed graph (source : [1])

B. *Terminologies of Graph*

There are a few common graph terminologies that will be used in this paper:

1. Adjacent

The term "adjacent" in graph theory refers to the relationship between two vertices that are directly connected by an edge. Vertices A and B are considered adjacent if there exists an edge directly connecting them within the graph. When (u, v) is an edge of the graph G, u is the initial vertex of an and v is the terminal or end vertex. If an edge forms a loop, the initial and terminal vertices are the same.

2. Incidence

"Incidence" denotes the connection between an edge and a vertex in a graph. A vertex is said to be incident on an edge if the edge is directly connected to that vertex. This terminology is vital for understanding the structural arrangement of edges and vertices within a graph.

3. Degree

The degree of a vertex u is the total number of edges that are incident with u. A loop with both endpoints at u adds two to the degree of vertex u. The degree of a vertex is represented by the notation $deg(v)$. A vertex with no incident edges is referred to as an isolated vertex. When the entire graph consists solely of isolated vertices, it is called a null or empty graph.

4. Path

A path in a graph is a finite or infinite sequence of edges which joins a sequence of vertices. It has the starting node v0 and the final node vn with the long path which is counted by the number of edges in the path.

5. Cycle

A path that has the same starting and finish node is called a cycle. It has a length which is counted by the number of edges in the cycle. The length of a cycle is the number of edges it contains, and a graph can have multiple cycles of varying lengths.

6. Subgraph

A subgraph is a subset of the vertices and all the edges of a larger graph. This subset forms a smaller graph that maintains the same properties and connections as the original graph.

7. Weighted Graph

A weighted graph is a graph in which each edge is given a specific value or weight. This value can represent anything in the real world such as price or distance.

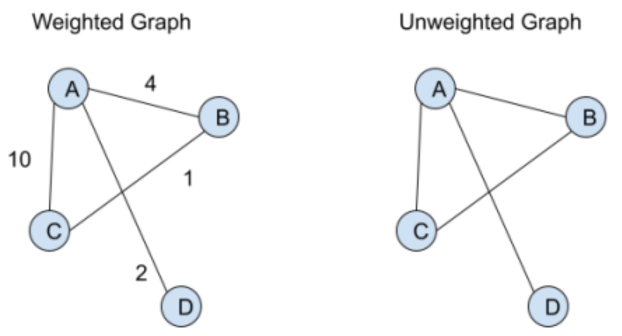


Fig. 4 weighted graph, unweighted graph (source : [1])

8. Connectedness
   Two vertices u and v are said to be connected if there exists a path that goes from u and v or vice versa.
9. Spanning subgraph
   A subgraph G1 = (V1, E1) is said to be the spanning subgraph of G = (V, E) if V1 = V, that is the vertices of G1 contain all the vertices of G.

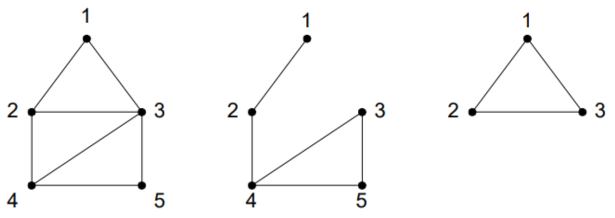


Fig. 5 Spanning Subgraph (source : [1])

C. Planar and Dual Graph
1. Planar Graph:
   A "planar graph" is characterized by its embeddability onto a two-dimensional plane or surface without any edges crossing each other, save for the common vertices at which they meet. A graph is deemed planar if it can be drawn on a plane such that no two edges intersect except at their shared vertices. The famous Kuratowski's Theorem establishes that a graph is planar if and only if it does not contain a subgraph that is a subdivision of the complete graph K_5 (the complete graph on five vertices) or the complete bipartite graph $K_{3,3}$.

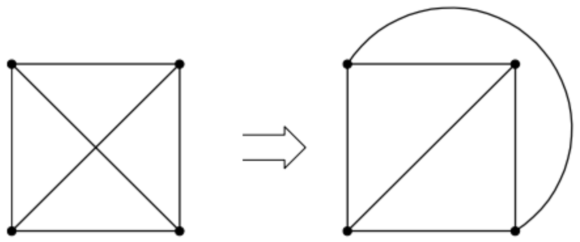


Fig. 6 Planar Graph (source : [2])

2. Dual Graph:
   The concept of a "dual graph" emerges as a powerful complement to planar graphs. Given a planar graph G, its dual graph, denoted as $G*$, is constructed by associating a face of G with a vertex in $G*$. Correspondingly, edges in $G*$ connect vertices associated with faces that share an edge in G.

C. Tree

A tree, by definition, is a connected, undirected graph that has no cycles or closed loops. The absence of cycles is a defining feature that distinguishes trees from more general graph structures, rendering them acyclic and facilitating a clear, branching pattern.

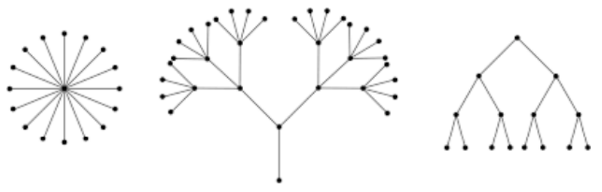


Fig. 7 Tree (source : [5])

D. <u>Minimum Spanning Tree</u>

A connected graph yields a spanning tree, which is essentially a tree-shaped subgraph encompassing all vertices of the original graph. The process of obtaining a spanning tree involves systematically removing edges until the formation of a circuit is no longer possible. A notable feature is that every connected graph possesses at least one spanning tree. In cases where the graph includes weighted edges, the concept of a minimum spanning tree arises. This minimum spanning tree is a tree-shaped subgraph with the lowest possible total weight among all potential spanning trees in the graph. Notably, a graph may exhibit multiple minimum spanning trees if certain edges share identical weights, resulting in distinct combinations of edges that contribute to the same total weight.

E. Prim's Algorithm

Prim's Algorithm, a cornerstone in graph theory and optimization, stands as an instrumental method for finding the Minimum Spanning Tree (MST) of a connected and undirected graph. The algorithm operates incrementally, selecting edges of minimum weight to grow the spanning tree until it encompasses all vertices. The iterative nature of Prim's Algorithm ensures the construction of an MST that minimizes the total weight, making it a valuable tool for network design, circuit layout, and various optimization problems.

Prim's algorithm builds a minimum spanning tree by progressively adding a new edge to the growing tree. Setting up the tree, depending on the variation of the algorithm, start either with any arbitrarily chosen vertex or the minimum weight edge of the graph. Then, iteratively take the next minimum weight edge which connects a vertex on the tree to another vertex not yet on the tree. By choosing an edge that connects a vertex on the tree to one that is not yet on the tree, it is ensured that no circuits will be formed. Repeat the iteration until all the vertices in the graph are connected, forming a complete minimum spanning tree.

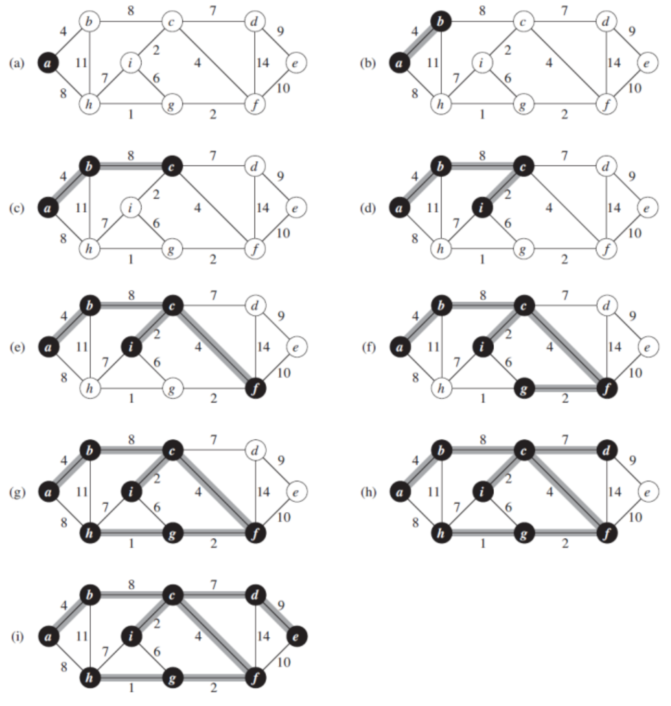


Fig. 8 Prim algorithm (source : [5])

III. Implementation

A. Limitations

Despite the efficacy of the implemented program in modeling a Fire Rescue Route Optimization using Prim's Algorithm, certain limitations constrain its applicability and realism.

1. Human Recognition

   Firstly, it is crucial to acknowledge that the program operates under the assumption that the drone is equipped with an advanced system for detecting individuals within rooms, specifically employing human recognition technology. This assumption simplifies the complexity of the real-world scenario, where the accuracy and efficiency of such detection systems may vary, impacting the overall effectiveness of the Fire Rescue Route Optimization.

2. Dummy Data

   The program relies on dummy data for distance and probability metrics associated with each room. While these values serve illustrative purposes and facilitate algorithmic execution, they lack the realism that would be present in actual rescue scenarios. In a practical setting, obtaining accurate and dynamic data, perhaps from sensors or real-time sources, would be imperative for enhancing the reliability of the generated Fire Rescue Route.

3. Static Data

   A notable limitation lies in the static nature of the data input process. Currently, the program necessitates manual input of the dataset each time it is executed. This constraint hinders the adaptability of the system to dynamic and evolving environments. In real-world scenarios, an automated or dynamically updated dataset would be preferable, reflecting the dynamic nature of emergency situations and allowing the program to respond promptly to changing conditions.

These limitations underscore the need for continuous refinement and augmentation of the program to align it more closely with the intricacies of real-world Fire Rescue Route Optimization scenarios. Addressing these constraints could involve integrating real-time data sources, enhancing the robustness of person detection mechanisms, and implementing mechanisms for dynamic dataset updates.

B. Location Data

In the pursuit of optimizing fire rescue routes through the application of Prim's Algorithm, the foundation lies in the construction of dataset. For the purpose of this study, a hospital floor plan was chosen, providing a complex yet representative framework for testing the algorithm's efficacy.

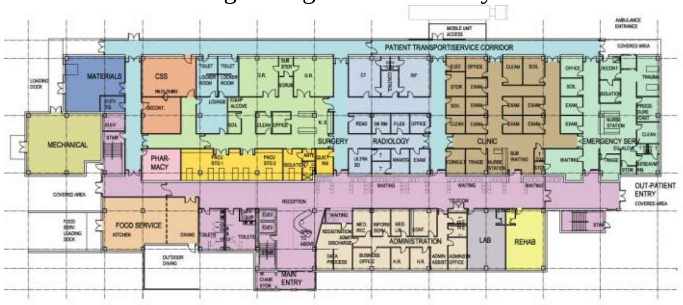
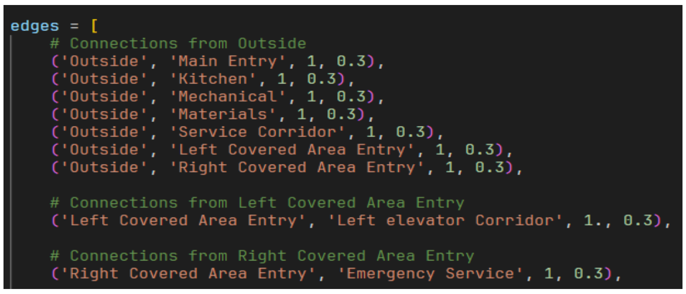


Fig. 9 Hospital blueprint (source : [6])

The chosen blueprint has been meticulously converted into a graph data structure, with each room or significant area within the building represented as a node. These nodes serve as critical waypoints in the pathfinding aspect of the rescue operation. The graph is constructed such that each node is connected to others via edges, with the adjacency denoting the physical connectivity between different areas in the hospital, such as hallways, doors, or direct paths.

The edges between nodes are weighted with distances, which are abstracted representations of the actual spatial lengths between areas in the building. For the purpose of this simulation, distances are pre-determined and static, serving as a simplified model of the real-world environment.

Alongside distance, a probability metric is assigned to each node, representing the likelihood of the presence of individuals in need of rescue within each area. This probability is informed by factors such as the typical usage of the room, its occupancy during emergency situations, and its location relative to common points of interest.

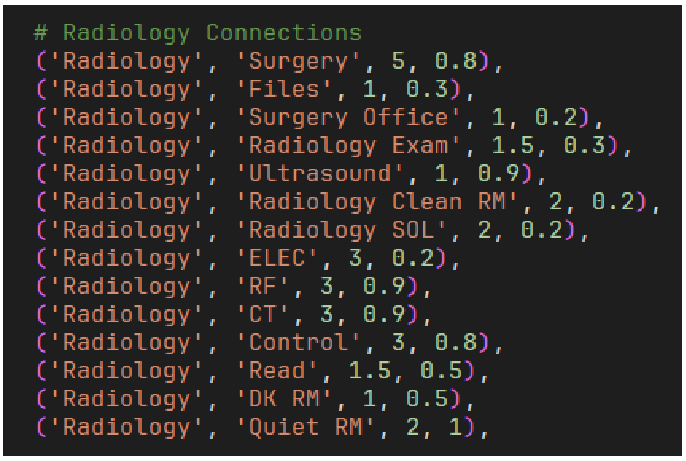


```python
edges = [
    # Connections from Outside
    ('Outside', 'Main Entry', 1, 0.3),
    ('Outside', 'Kitchen', 1, 0.3),
    ('Outside', 'Mechanical', 1, 0.3),
    ('Outside', 'Materials', 1, 0.3),
    ('Outside', 'Service Corridor', 1, 0.3),
    ('Outside', 'Left Covered Area Entry', 1, 0.3),
    ('Outside', 'Right Covered Area Entry', 1, 0.3),

    # Connections from Left Covered Area Entry
    ('Left Covered Area Entry', 'Left elevator Corridor', 1., 0.3),

    # Connections from Right Covered Area Entry
    ('Right Covered Area Entry', 'Emergency Service', 1, 0.3),
```

Fig. 10.1 Data (source : Primary)



```python
# Radiology Connections
('Radiology', 'Surgery', 5, 0.8),
('Radiology', 'Files', 1, 0.3),
('Radiology', 'Surgery Office', 1, 0.2),
('Radiology', 'Radiology Exam', 1.5, 0.3),
('Radiology', 'Ultrasound', 1, 0.9),
('Radiology', 'Radiology Clean RM', 2, 0.2),
('Radiology', 'Radiology SOL', 2, 0.2),
('Radiology', 'ELEC', 3, 0.2),
('Radiology', 'RF', 3, 0.9),
('Radiology', 'CT', 3, 0.9),
('Radiology', 'Control', 3, 0.8),
('Radiology', 'Read', 1.5, 0.5),
('Radiology', 'DK RM', 1, 0.5),
('Radiology', 'Quiet RM', 2, 1),
```

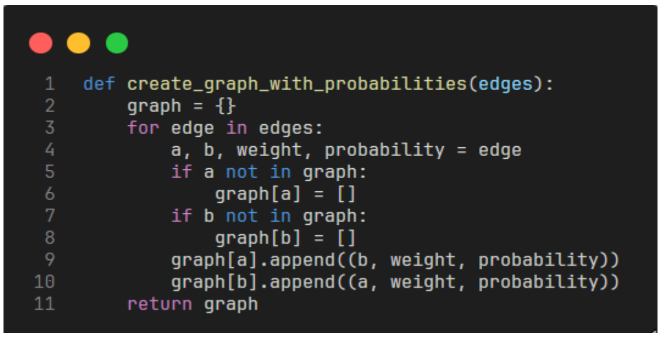
Fig. 10.1 Data (source : Primary)

The interplay between distance and probability is central to the proposed solution. The Prim's Algorithm, traditionally concerned only with the weight of the edges, is enhanced by also considering the node probabilities. This dual consideration allows for the generation of a Minimum Spanning Tree (MST) that not only spans the least distance but also maximizes the potential for successful rescue operations.

C. *Problem Modelling*

1. Create a graph dataset using the data inputted.

   The inception of our problem modeling begins with the creation of a graph dataset that meticulously mirrors the architectural complexity and interconnectedness of a building's layout. By transforming the architectural blueprints into a graph data structure, we forge a model where rooms and corridors are epitomized as vertices and edges, respectively. For the graph dataset in our fire rescue optimization problem, a dictionary data structure in Python is employed to represent the graph.



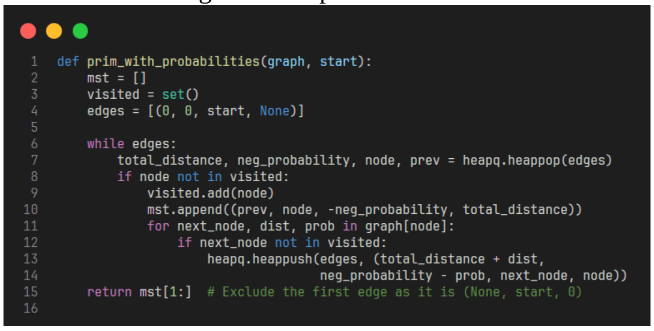
```python
def create_graph_with_probabilities(edges):
    graph = {}
    for edge in edges:
        a, b, weight, probability = edge
        if a not in graph:
            graph[a] = []
        if b not in graph:
            graph[b] = []
        graph[a].append((b, weight, probability))
        graph[b].append((a, weight, probability))
    return graph
```

Fig. 11 create graph function (source : Primary)

   In the context of our graph, each room or area within the building is represented as a key in the dictionary, and the value is a list of tuples, each tuple containing an adjacent room (b), the distance to it (weight), and the probability of finding individuals there (probability). This structure allows the program to access the adjacency list of each node directly using the node itself as a key, leading to efficient queries and updates, which are frequent operations during the execution of Prim's Algorithm.

   When the create_graph_with_probabilities function is invoked, it initializes an empty dictionary and iteratively populates it using the provided edges data. For every pair of adjacent rooms (a, b), the function ensures that both a and b are keys in the dictionary. If they are not already present, they are added, and their value is initialized to an empty list. The function then appends a tuple to the adjacency list of both a and b, representing an undirected edge between them.

2. Prim algorithm Implementation



```python
def prim_with_probabilities(graph, start):
    mst = []
    visited = set()
    edges = [(0, 0, start, None)]

    while edges:
        total_distance, neg_probability, node, prev = heapq.heappop(edges)
        if node not in visited:
            visited.add(node)
            mst.append((prev, node, -neg_probability, total_distance))
            for next_node, dist, prob in graph[node]:
                if next_node not in visited:
                    heapq.heappush(edges, (total_distance + dist,
                                           neg_probability - prob, next_node, node))
    return mst[1:]  # Exclude the first edge as it is (None, start, 0)

```

Fig. 12 Creating graph with prim (source : Primary)

   The prim_with_probabilities function is the heart of our algorithmic endeavour, where Prim's algorithm is ingeniously augmented to factor in the probability metrics alongside the traditional distance measure. This hybrid approach recalibrates the weight of each edge by deducting the probability value from the distance, thereby formulating a weighted score that serves as a beacon to guide the rescue drone.

$$\text{TotalWeight} = \text{Distance} - \text{Probablity}$$

   At the algorithm, a priority queue, represented as a heap, is initialized with a tuple comprising the total distance, negated probability (to convert a maximization problem into a minimization one), the starting node, and a placeholder for the previous node. The algorithm then enters a loop, continually evaluating the heap's smallest element, which represents the most efficacious edge to traverse next.

As each node is visited, it is marked to prevent redundancy. The algorithm then check the connected vertices and their respective edges, pushing new edges onto the heap only if they lead to unvisited vertices. This approach ensures that the resultant minimum spanning tree (MST) is not merely a path of least resistance but a route optimized for efficacy and potential for successful search and rescue.

IV. Testing

A. *Testing*

The result of the program execution after combining all the functions that could be seen in methodology chapter could be seen in following figures.

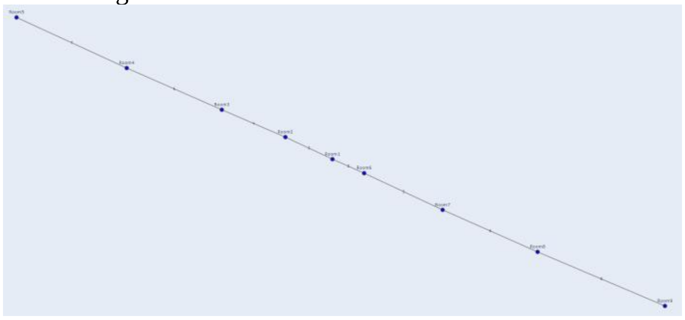


Fig. 13.1 graph visualization (source : Primary)

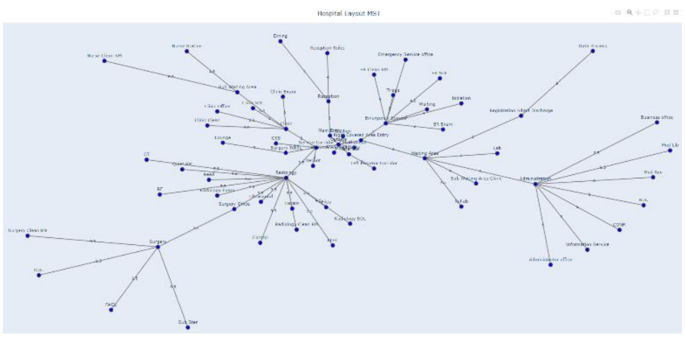


Fig. 13.2 graph visualization (source : Primary)

B. *Test Explanation*

In the initial testing phase, program was challenged with a simple model comprising nine rooms, connected in a linear fashion. The test resulted in a straightforward solution where the optimal path was to traverse from 'Room 1' to 'Room 9', as depicted in the first figure. The decision-making process in this case was fairly uncomplicated due to the limited number of nodes and edges, thus the program could easily calculate the most efficient route based on the given distances and probabilities.

The program's logic initiated at 'Room 1', presenting a binary choice to proceed to either 'Room 2' or 'Room 6'. The algorithm calculated the weights and probabilities, determining that the progression from 'Room 1' to 'Room 2' and sequentially through to 'Room 5' formed one optimal branch, while simultaneously advancing from 'Room 1' to 'Room 6' and onwards to 'Room 9' constructed the second. This dual-pathway approach ensured coverage of all rooms in a manner that minimized the total distance while considering the probability factors.

The second figure reveals the complexity that comes with scaling up from a basic model to a comprehensive hospital layout with multiple entries and a web of interconnected rooms. The hospital dataset introduces an intricate network of nodes and edges, representing rooms and pathways with varying distances and probabilities reflective of a real-world setting.

Here, the program faces a multidimensional challenge. The decision-making process becomes more nuanced as the algorithm weighs the benefits of choosing one path over another, not just on distance but also on the likelihood of encountering individuals. The program commences from multiple entry points: 'Outside' to 'Main Entry', 'Kitchen', 'Service Corridor', 'Left Covered Area Entry', 'Materials', 'Mechanical', or 'Right Covered Area Entry', each serving as a potential starting node for the rescue route.

With each decision, the program performs a calculation that extends beyond simple distance minimization. It considers the weighted probability of finding individuals in need of rescue—a crucial factor in a hospital emergency. As the program traverses the hospital graph, it selects rooms one by one based on the cumulative score of distance and probability. The program meticulously constructs a Minimum Spanning Tree (MST) that encapsulates the most optimal route.

The MST evolves with each added edge, reflecting a route that dynamically adjusts according to the dual parameters. If a room has a high probability but is farther away, the program calculates whether it is more efficient to visit it immediately or to approach it through a series of closer, less probable rooms. This decision is replicated across the graph, creating a rescue route that is both systematic and adaptable.

V. Conclusion

This research has successfully demonstrated the application of Prim's Algorithm in optimizing fire rescue routes within a complex building environment. By incorporating a probability metric that accounts for the potential presence of individuals in various rooms, the study has presented a nuanced approach to navigating drones for efficient human rescue operations. The algorithm's adeptness at adapting to a detailed hospital layout signals its robust potential for real-world implementation in emergency response systems. Despite facing limitations such as static data inputs and assumptions about detection capabilities, the study represents a significant stride in combining algorithmic precision with practical rescue operations.

The insights gleaned from this study illuminate the path forward for integrating advanced computational models with life-saving emergency services. It beckons future research to address the highlighted limitations, particularly the integration of dynamic, real-time data to enhance adaptability and responsiveness. As this paper concludes, it reinforces the imperative of continuous innovation in emergency response strategies, advocating for the synergy of technological advancements and human expertise to safeguard lives against the unpredictable nature of emergencies. In essence, the paper concludes with a forward-looking perspective, underscoring the transformative potential of algorithmic applications in emergency scenarios. The study not only contributes to academic discourse but also serves as a foundation for practical advancements in emergency response protocols.

VI. APPENDIX

The complete program of this paper can be found below.
https://github.com/AlbertChoe/Prim-Algorithm_discreteMath

VII. ACKNOWLEDGMENT

First and foremost, the author expresses gratitude to their Discrete Mathematics lecturer, Dr. Fariska Zakhralativa Ruskanda, S.T., M.T., for generously sharing knowledge with fellow students, including the researcher, whose successful completion of this research owes much to their guidance. Additionally, the researcher extends thanks to the other lecturers who contributed to and were part of the IF2120 Discrete Math class, as their dedication played a crucial role in motivating the researcher to write this research paper. Lastly, the researcher acknowledges their friends for their unwavering support throughout the Discrete Math class, fostering a collaborative learning environment that has contributed to personal growth and development over time.

REFERENCES

[1] Munir, R. (2023). Graf Bagian 1 accessed December 10, 2023 https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf
[2] Munir, R. (2023). Graf Bagian 2. accessed December 10, 2023 https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/20-Graf-Bagian2-2023.pdf
[3] Munir, R. (2023). Graf Bagian 3. accessed December 10, 2023 https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf
[4] Kejadian kebakaran beserta jumlah kerugian, korban dan penyebabnya pada tahun 2019 accessed December 10, 2023, [Online]. https://cepagram.com/index.php/2022/07/17/kejadian-kebakaran-beserta-jumlah-kerugian-korban-dan-penyebabnya-pada-tahun-2019/E. H. Miller, "A note on reflector arrays (Periodical style—Accepted for publication)," *IEEE Trans. Antennas Propagat.*, to be published.
[5] Prims algorithm accessed December 8, 2023, [Online]. https://cs.stackexchange.com/questions/50964/confusion-in-clrss-version-of-prims-algorithm.
[6] Online Hospital Design Services up to 50 Beded Hospital Anywhere in World, [Online]. https://arcmaxarchitect.com/shop/hospital-designs-architecture/buy-hospital-design-services-up-to-50-beded-hospital-anywhere-in-world
[7] Prim's Algorithm for Minimum Spanning Tree (MST), [Online]. https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 Desember 2023

Albert

Albert 13522081