Optimalisasi Jaringan Komputer Melalui *Weighted Graph*: Implementasi *Spanning Tree* dan Relasi Rekurens

Muhammad Raihan Nazhim Oktana - 13523021[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]m.raihannazhimoktana@gmail.com, 13523021@std.stei.itb.ac.id

Abstrak—Matematika diskrit adalah suatu cabang materi dalam matematika yang punya banyak aplikasi dalam dunia teknologi seperti pada informatika. Matematika diskrit akan mengkaji objek-objek yang dihitung secara diskrit dan tidak koninu. Matematika diskrit memiliki beragam cabang materi didalamnya, mulai dari logika, himpunan, relasi, teori bilangan, kombinatorika, aljabar boolean, graf, pohon, kompleksitas algoritma, dan lainnya. Penggabungan seluruh cabang ini, menjadi satu materi yang cukup menarik dalam dunia informatika untuk dimanfaatkan dalam dunia nyata. Dua ilmu penting dalam matematika diskrit adalah graf dan pohon. Secara umum, graf adalah sebuah materi yang membahas tentang representasi objek-objek diskrit dan hubungan antara objek-objek tersebut. Selain itu, pohon adalah sebuah materi yang membahas tentang representasi Kumpulan graf tak-berarah yang terhubung dan tidak mengandung sirkuit. Kedua materi graf dan pohon adalah suatu struktur data yang fundamental karena punya banyak penerapan dan implementasinya. Hingga saat ini, dunia teknologi telah berkembang dengan sangat pesat. Dalam hal ini, perkembangan dunia informatika dan kaitannya terhadap dunia teknologi sangatlah penting. Jaringan Komputer menjadi salah satu perkembangan teknologi yang menarik untuk dianalisis. Penerapan ilmu informatika yang lebih baik tentu dapat mengoptimalkan struktur jaringan komputer yang lebih baik lagi. Pada kesempatan kali ini, penulis ingin mencari tahu dan menganalisis implementasi graf berbobot dalam pengoptimalan jaringan komputer untuk proses yang lebih baik. Dari hasil percobaan yang dilkaukan, ditemukan bahwa terbukti aplikasi graf berbobot dengan *spanning-tree* dan relasi rekurens pada optimalisasi jaringan komputer sangat baik dan bermanfaat.

Kata kunci—matematika diskrit, graf, jaringan komputer, relasi rekurens, spanning-tree.

I. PENDAHULUAN

Matematika diskrit adalah suatu cabang materi dalam matematika yang punya banyak aplikasi dalam dunia teknologi seperti pada informatika. Matematika diskrit akan mengkaji objek-objek yang dihitung secara diskrit dan tidak koninu. Matematika diskrit memiliki beragam cabang materi didalamnya, mulai dari logika, himpunan, relasi, teori bilangan, kombinatorika, aljabar boolean, graf, pohon, kompleksitas algoritma, dan lainnya. Kegunaan dari setiap cabang materi ini sangat banyak dan beragam, mulai dari pemecahan masalah, hingga memberikan dasar penting untuk digabungkan dengan ilmu yang lain. Penggabungan seluruh cabang ini, menjadi satu materi yang cukup menarik dalam dunia informatika untuk dimanfaatkan dalam dunia nyata [1].

Pada materi matematika diskrit, terdapat banyak sekali cabang-cabang ilmu. Beberapa di antaranya adalah graf dan pohon. Secara umum, graf adalah sebuah materi yang membahas tentang representasi objek-objek diskrit dan hubungan antara objek-objek tersebut [5]. Selain itu, pohon adalah sebuah materi yang membahas tentang representasi Kumpulan graf tak-berarah yang terhubung dan tidak mengandung sirkuit [8]. Kedua materi graf dan pohon adalah suatu struktur data yang fundamental karena punya banyak penerapan dan implementasinya.

Hingga saat ini, dunia teknologi telah berkembang dengan sangat pesat. Dalam hal ini, perkembangan dunia informatika dan kaitannya terhadap dunia teknologi sangatlah penting. Dapat dilihat di berbagai teknologi modern, hampir seluruhnya menerapkan ilmu-ilmu dalam dunia informatika. Jaringan Komputer menjadi salah satu perkembangan teknologi yang menarik untuk dianalisis. Penerapan ilmu informatika yang lebih baik tentu dapat mengoptimalkan struktur jaringan komputer yang lebih baik lagi. Hal ini akan berdampak pada proses yang lebih baik dalam jaringan-jaringan komputer tersebut [12].

Matematika diskrit punya peran penting dalam memajukan berbagai teknologi yang ada di dunia. Penerapan graf dan pohon yang tepat akan sangat membantu pembuatan teknologi yang lebih baik lagi. Masih banyak pengembangan untuk optimalisasi dan inovasi baru yang dapat dilakukan. Oleh karena itu, pada kesempatan kali ini, penulis ingin mencari tahu dan menganalisis implementasi *spanning-tree* dan relasi rekurens dalam pengoptimalan jaringan komputer yang lebih baik. Penulis akan mencoba melakukan penerapan graf berbobot dalam pengoptimalan struktur jaringan komputer dengan menggunakan materi-materi pada matematika diskrit yang berkaitan [13].

II. DASAR TEORI

A. Matematika Diskrit

Matematika diskrit adalah suatu cabang materi dalam matematika yang punya banyak aplikasi dalam dunia teknologi seperti pada informatika. Matematika diskrit akan mengkaji objek-objek yang dihitung secara diskrit dan tidak koninu. Matematika diskrit memiliki beragam cabang materi didalamnya, mulai dari logika, himpunan, relasi, teori bilangan, kombinatorika, aljabar boolean, graf, pohon, kompleksitas algoritma, dan lainnya. Kegunaan dari setiap cabang materi ini sangat banyak dan beragam,

mulai dari pemecahan masalah, hingga memberikan dasar penting untuk digabungkan dengan ilmu yang lain. Penggabungan seluruh cabang ini, menjadi satu materi yang cukup menarik dalam dunia informatika untuk dimanfaatkan dalam dunia nyata [1].

B. Relasi

Relasi merupakan suatu materi dalam ilmu matematika diskrit. Relasi membahas tentang hubungan antar 2 buah objek yang pada umumnya direpresentasikan dalam bentuk himpunan. Pada penerapannya, relasi dapat diperluas secara umum menjadi hubungan antar 2 buah hal yang saling terkait satu sama lain. Relasi memiliki banyak kegunaan, seperti pada relasi rekurens dan berbagai jenis algoritma yang terkait dengan relasi [2].

C. Relasi Rekurens

Secara definisi, rekursi berarti memanggil dirinya sendiri, dan relasi adalah keterkaitan antar 2 buah hal yang saling terkait satu sama lain. Oleh karena itu, relasi rekurens berarti suatu proses relasi yang dilakukan secara rekursif. Relasi rekurens menjadi penting dalam beberapa hal, seperti dalam penerapan beberapa algoritma pencarian jalan terpendek [3].

Dalam penerapannya, relasi rekurens dimodelkan dalam suatu bentuk operasi matematis yang tepat. Umumnya, suatu ekspresi akan bergantung pada ekspresi sebelumnya, dan akan mempengaruhi ekspresi-ekspresi selanjutnya. Contoh paling umum relasi rekurens adalah barisan fibonacci [4].

$$f_n = f_{n-1} + f_{n-2}$$

Gambar 1. Ilustrasi Relasi Rekurens [4]

D. Graf

Graf adalah suatu himpunan atau kumpulan dari berbagai simpul dan sisi yang menunjukkan relasi antar simpul. Graf memiliki 2 jenis, yaitu graf sederhana dan graf tidak sederhana. Terdapat beberapa terminologi khusus yang ada pada graf, yaitu simpul dan sisi [5].

Graf dapat direpresentasikan dalam berbagai hal, seperti matriks ketetanggaan (*adjacency matrix*), matriks bersisisan (*incidency matrix*), dan senarai ketetanggaan (*adjacency list*) [6]. Dalam teori graf, dikenal istilah graf euler dan graf hamilton. Secara umum, graf euler berarti melewati setiap sisi sebanyak 1x dan berakhir pada simpul yang sama, atau jika tidak disebut lintasan euler. Secara umum, graf hamilton berarti melewati setiap simpul sebanyak 1x dan berakhir pada simpul yang bersisian dengan simpul awal, atau jika tidak disebut lintasan hamilton [7].

Dalam algortima pencarian jalan terpendek, dikenal beberapa algoritma, seperti Algoritma Dijkstra dan Algoritma A-Star. Algoritma Dijkstra bekerja dengan rekursi normal, sedangkan Algoritma A-Star dibantu oleh suatu fungsi heuristic [14][15].

E. Pohon

Pohon merupakan suatu struktur data pengembangan dari graf. Pohon merupakan suatu jenis graf yang tidak memiliki bentuk siklik. Terdapat berbagai jenis pohon, umumnya dikenal dengan *binary tree* dan *n-ary tree*. Dikenal pula terminologi dari pohon, seperti *parent*, *children*, *leaf*, dan berbagai terminologi lainnya [9].

Pada pembahasan pohon, dikenal pula pohon merentang. Pohon merentang merupakan suatu bentuk pohon dimana setiap simpul terhubung pada upapohon (subpohon). Lebih spesifik, dikenal pohon merentang minimum (minimum *spanning-tree*), yaitu pohon merentang dengan bobot minimum. Ada 2 algoritma besar yang dikenal, yaitu Algoritma Prim dan Algortima Kruskal. Algoritma Prim bekerja dengan urut dari sisi dengan bobot paling kecil, lalu bobot sisi bersisian yang paling kecil, dan seterusnya. Sedangkan Algoritma Kruskal mengurutkan sisi-sisi tersebut dari yang paling ringan, lalu membentuk pohon merentang minimum dari yang bobot terkecil tanpa membuat siklik dan memenuhi tujuan dari pohon merentang minimum tersebut [8].

F. Kompleksitas Algoritma

Kompleksitas algoritma merupakan suatu hal penting dalam pembahasan algoritma. Hal ini karena kompleksitas algoritma menggambarkan secara umum tingkat kompleks dari suatu algoritma. Secara harfiah, kompleksitas ini berarti menghitung banyaknya operasi yang dilakukan oleh suatu algoritma [10].

Dalam pembahasan kompleksitas algoritma yang lebih lanjut, dikenal notasi dalam kompleksitas algoritma. Notasi utama tersebut dibagi menjadi 3, yaitu Big-O, Big-Omega, dan Big-Theta. Ketiganya saling berkaitan dan punya fungsi masing-masing dalam menggambarkan kompleksitas suatu algoritma [11].

G. Jaringan Komputer

Jaringan komputer adalah suatu kumpulan atas ragam perangkat komputer yang saling terhubung satu sama lain. Keterkaitan ini memungkinkan mereka untuk dapat bertukar data, sumber daya, dan aplikasi. Jaringan komputer menggunakan protokol komunikasi seperti untuk melakukan transmisi informasi melalui media kabel ataupun nirkabel [12].

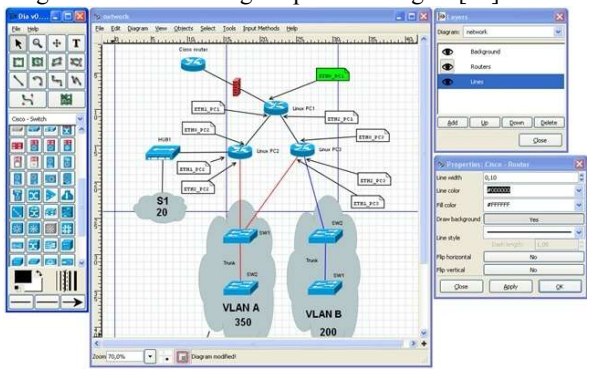
Jaringan komputer punya berbagai manfaat. Mulai dari memudahkan komunikasi, memeudahkan pengiriman data, hingga meningkatkan keamanan data. Contoh dari jaringan komputer adalah LAN, PAN, MAN, CAN, VPN, WAN.

Gambar 2. Ilustrasi Jaringan Komputer

Sumber: Ilustrasi Jaringan Komputer

H. Aplikasi Graf pada Jaringan Komputer

Dalam konteks jaringan komputer, graf dapat menjadi media aplikasi yang cukup baik. Hal ini memungkinkan graf untuk menjadi media visualisasi yang baik atas suatu jaringan komputer. Pada penerapannya, graf tersebut dapat merepresentasikan bobot biaya, waktu transmisi, waktu proses, dan sebagainya dalam bentuk graf berbobot. Dengan ini, proses pembuatan jaringan komputer akan sangat dimudahkan dengan aplikasi dari graf [13].



Gambar 3. Ilustrasi Graf Jaringan Komputer
Sumber: Ilustrasi Graf Jaringan Komputer

III. PEMBAHASAN

A. Perhitungan Pohon Merentang Minimum

Pohon merentang minimum (minimum *spanning-tree*) adalah sebuah upagraf (subgraf) dari suatu graf awal yang dapat bertujuan untuk menghubungkan semua simpul yang ada pada graf tersebut dengan total bobot terkecil dan tanpa membentuk suatu sirkuit. Algoritma yang akan digunakan dalam membuat minimum *spanning-tree* pada percobaan ini adalah Algoritma Prim dan Algoritma Kruskal. Algoritma Prim akan bekerja dengan memulai pada suatu sisi dengan bobot terkecil, selanjutnya akan bekerja dengan mencari simpul terkecil yang berhubungan dengan upagraf yang sudah terbentuk dan tidak membentuk siklik, proses diulang sampai upagraf yang sesuai terbentuk. Sedangkan Algoritma Kruskal akan bekerja dengan cara mengurutkan semua sisi graf dari yang terkecil hingga terbesar, lalu menyusunnya agar membentuk upagraf yang sesuai dan tidak membentuk siklik. Penerapan ini menjadi penting karena desain jaringan komputer yang baik tentu memerlukan aspek terhadap struktur jaringan yang baik, hal ini untuk meminimalisasi waktu yang dibutuhkan.

B. Pencarian Jarak Terpendek

Pencarian jarak terpendek adalah suatu masalah penting yang perlu dibahas dalam permasalahan jaringan komputer ini. Hal ini penting untuk meminimalisasi waktu yang ditempuh antar komponen dalam jaringan komputer. Algoritma yang digunakan dalam percobaan ini untuk mencari jalan terpendek adalah Algoritma Dijkstra dan A-Star (A*). Kedua algoritma ini memiliki kelebihan dan kekurangan masing-masing untuk implementasinya. Algoritma Dijkstra akan lebih efektif pada graf berbobot non-negatif, sementara Algoritma A-Star menggabungkan heuristik untuk membantunya memperceat pencarian rute tersebut. Algoritma pencarian jarak terpendek menjadi salah satu hal yang sangat krusial dalam struktur jaringan komputer dan perlu diberi perhatian lebih.

C. Pengaturan Struktur Jaringan Komputer

Struktur jaringan komputer menjadi hal penting karena akan sangat berkaitan dengan waktu yang dibutuhkan untuk menempuh jarak-jarak antar komponen jaringan komputer. Rancangan yang tepat dapat dilakukan dengan bantuan dari teori graf dan pohon untuk melakukan perencanaan yang tepat. Efisiensi yang baik dalam komunikasi akan sangat memberikan dampak positif kepada kualitas sekumpulan jaringan komputer yang terkait seutuhnya.

D. Dampak Positif pada Jaringan Komputer

Penerapan konsep graf dan pohon dari matematika diskrit dalam jaringan komputer akan memberikan banyak dampak positif, mulai dari peningkatan efisiensi waktu, peningkatan efisiensi pengelolaan data, optimalisasi sumber daya jaringan yang ada, mengurangi waktu latensi, meningkatkan kecepatan transmisi, dan berbagai dampak positif lainnya. Perkembangan dunia jaringan komputer akan semakin pesat dengan pemanfaatan teori-teori dasar dalam matematika diskrit yang baik seperti ini.

IV. IMPLEMENTASI PROGRAM

Proses implementasi program implementasi graf berbobot dalam jaringan komputer untuk optimalisasi jaringan komputer membutuhkan beberapa urutan tahap implementasi. Tahapan yang akan dijelaskan di sini mengabaikan tahapan kecil seperti menginput graf, atau semacamnya. Secara umum, tahap implementasi ini meliputi proses deklarasi struktur tipe data, perhitungan pohon merentang minimum, serta proses pencarian jarak terpendek antara sumber dan tujuan.

Untuk implementasi pertama, dapat dilihat seperti pada program berikut. Menggunakan struktur *list of edge* dan *list of node* dan merepresentasikannya dalam bentuk matriks keterkaitan.

```
class Node:
    def __init__(self , info , x , y) :
        self.info = info
        self.x = x
        self.y = y

    def constructNode(self , info , x , y) :
        self.info = info
        self.x = x
        self.y = y
        return self
```

Gambar 4. Struktur Node
Sumber: Dokumen Penulis / Lampiran

```python
class Edge:
    def __init__(self , info , node1 , node2 , weight) :
        self.info = info
        self.node1 = node1
        self.node2 = node2
        self.weight = weight

    def constructEdge(self , x , ujung1 , ujung2 , weight) :
        self.info = x
        self.node1 = ujung1
        self.node2 = ujung2
        self.weight = weight
        return self
```

Gambar 5. Struktur Edge
Sumber: Dokumen Penulis / Lampiran

```python
class Graph :
    def __init__(self , list_of_node , list_of_edge , matrix) :
        self.nodes = list_of_node
        self.edges = list_of_edge
        self.graph = matrix

    def constructGraph(self , list_node , list_edge) :
        self.nodes = list_node
        self.edges = list_edge
        matrix = [[0 for _ in range (len(list_node))] for _ in range (len(list_node))]
        for i in range (len(list_edge)) :
            matrix[list_edge[i][1]][list_edge[i][2]] = 1
            matrix[list_edge[i][2]][list_edge[i][1]] = 1
        self.graph = matrix
        return self
```

Gambar 6. Struktur Graph
Sumber: Dokumen Penulis / Lampiran

Untuk implementasi kedua, dapat dilihat seperti pada program berikut. Proses pengecekan pohon dilakukan terlebih dahulu karena algoritma prim dan kruskal ataupun semacamnya bertujuan untuk mengecek pohon. Selanjutnya dilakukan implementasi algoritma prim dan kruskal untuk menghitung pohon merentang minimum.

```python
def is_tree(self) :
    def dfs(current_index , parent_index) :
        visited[current_index] = True
        for i in range (len(self.nodes)) :
            if (self.graph[current_index][i] != 0) :
                if (not(visited[i])):
                    if (not(dfs(i , current_index))) :
                        return False
                elif (i != parent_index) :
                    return False
        return True

    visited = [False] * len(self.nodes)
    if (not(dfs(0 , -1))) :
        return False
    else :
        return all(visited)
```

Gambar 7. Pengecekan Pohon
Sumber: Dokumen Penulis / Lampiran

```python
def prim(self) :
    num_nodes = len(self.nodes)
    in_mst = [False] * num_nodes
    min_edge = [(float('inf') , -1)] * num_nodes
    min_edge[0] = (0 , -1)
    total_weight = 0
    mst_edges = []
    for _ in range (num_nodes) :
        u = -1
        for i in range (num_nodes) :
            if ((not(in_mst[i])) and ((u == -1) or (min_edge[i][0] < min_edge[u][0]))) :
                u = i
        if ((min_edge[u][1] == -1) and (u != 0)) :
            return None
        else :
            in_mst[u] = True
            total_weight += min_edge[u][0]
            if (min_edge[u][1] != -1) :
                mst_edges.append((u , min_edge[u][1] , min_edge[u][0]))
            for v in range (num_nodes) :
                if ((self.graph[u][v] != 0) and (not(in_mst[v])) and (self.graph[u][v] < min_edge[v][0])) :
                    min_edge[v] = (self.graph[u][v] , u)
    return mst_edges , total_weight
```

Gambar 8. Implementasi Algoritma Prim
Sumber: Dokumen Penulis / Lampiran

```python
def kruskal(self) :
    class UnionFind :
        def __init__(self , size) :
            self.parent = list(range(size))
            self.rank = [0] * size

        def find(self , p) :
            if (self.parent[p] != p) :
                self.parent[p] = self.find(self.parent[p])
            return self.parent[p]

        def union(self , p , q) :
            rootP = self.find(p)
            rootQ = self.find(q)
            if (rootP != rootQ) :
                if (self.rank[rootP] < self.rank[rootQ]) :
                    self.parent[rootP] = rootQ
                elif (self.rank[rootP] > self.rank[rootQ]) :
                    self.parent[rootQ] = rootP
                else :
                    self.parent[rootQ] = rootP
                    self.rank[rootP] += 1
                return True
            return False
```

Gambar 24. Implementasi Algoritma Kruskal 1
Sumber: Dokumen Penulis / Lampiran

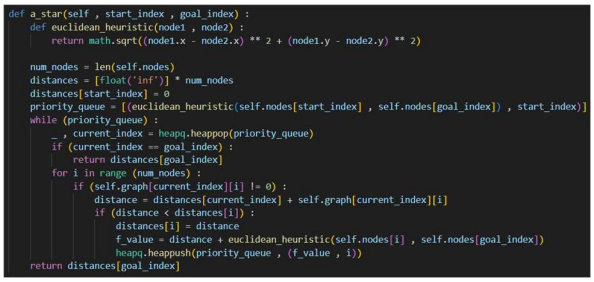
```python
    edges = []
    for u in range (len(self.nodes)) :
        for v in range (u + 1 , len(self.nodes)) :
            if (self.graph[u][v] != 0) :
                edges.append((self.graph[u][v] , u , v))
    edges.sort()
    uf = UnionFind(len(self.nodes))
    mst_edges = []
    total_weight = 0
    for (weight , u , v) in (edges) :
        if (uf.union(u , v)) :
            mst_edges.append((u , v , weight))
            total_weight += weight
    if (len(mst_edges) != len(self.nodes) - 1) :
        return None
    else :
        return mst_edges , total_weight
```

Gambar 9. Implementasi Algoritma Kruskal 2
Sumber: Dokumen Penulis / Lampiran

Untuk implementasi ketiga, dapat dilihat seperti pada program berikut. Proses algoritma dijkstra dan a-star dilakukan untuk menghitung jalan terpendek antara 2 titik sesuai dengan teorinya. Proses ini menggunakan bantuan library heapq dan math untuk memudahkan proses algoritma dan perhitungan.

```python
def dijkstra(self , start_index , goal_index) :
    num_nodes = len(self.nodes)
    distances = [float('inf')] * num_nodes
    distances[start_index] = 0
    priority_queue = [(0 , start_index)]
    while (priority_queue) :
        current_distance , current_index = heapq.heappop(priority_queue)
        if (current_index == goal_index) :
            return distances[goal_index]
        elif (current_distance > distances[current_index]) :
            continue
        else :
            for i in range (num_nodes) :
                if (self.graph[current_index][i] != 0) :
                    distance = current_distance + self.graph[current_index][i]
                    if (distance < distances[i]) :
                        distances[i] = distance
                        heapq.heappush(priority_queue , (distance , i))
    return distances[goal_index]
```

Gambar 10. Implementasi Algoritma Dijkstra
Sumber: Dokumen Penulis / Lampiran



```
def a_star(self , start_index , goal_index) :
    def euclidean_heuristic(node1 , node2) :
        return math.sqrt((node1.x - node2.x) ** 2 + (node1.y - node2.y) ** 2)

    num_nodes = len(self.nodes)
    distances = [float('inf')] * num_nodes
    distances[start_index] = 0
    priority_queue = [(euclidean_heuristic(self.nodes[start_index] , self.nodes[goal_index]) , start_index)]
    while (priority_queue) :
        _ , current_index = heapq.heappop(priority_queue)
        if (current_index == goal_index) :
            return distances[goal_index]
        for i in range (num_nodes) :
            if (self.graph[current_index][i] != 0) :
                distance = distances[current_index] + self.graph[current_index][i]
                if (distance < distances[i]) :
                    distances[i] = distance
                    f_value = distance + euclidean_heuristic(self.nodes[i] , self.nodes[goal_index])
                    heapq.heappush(priority_queue , (f_value , i))
    return distances[goal_index]
```

Gambar 11. Implementasi Algoritma A-Star
Sumber: Dokumen Penulis / Lampiran

Dengan seluruh implementasi tersebut, program ini dapat berjalan dengan baik sesuai harapan. Tentunya, perlu ada banyak penyesuaian tambahan yang menghubungkan antar subprogram fungsi-fungsi tersebut. Keseluruhan implementasi program lengkap dapat dilihat pada source code di lampiran.

V. KESIMPULAN

Setelah melakukan rangkaian penelitian, penulis dapat menyimpulkan pembahasan atas berbagai percobaan dan pengamatan yang telah dilakukan sebagai berikut:

1. Penerapan materi matematika diskrit untuk memberikan optimalisasi jaringan komputer dengan mengimplementasi materi graf berbobot dan pohon berbobot melalui bantuan *spanning tree* dan relasi rekurens pada jaringan komputer yang optimal terbukti sangat banyak dan bermanfaat.
2. Implementasi *spanning tree* sebagai salah satu teknik yang dapat dilakukan untuk menentukan pohon merentang minimum terbukti merupakan suatu teknik yang sangat membantu penentuan jaringan komputer yang optimal.
3. Implementasi relasi rekurens dalam optimalisasi pencarian jalur terpendek sebagai salah satu teknik yang dapat dilakukan untuk menentukan jarak minimum terbukti merupakan suatu teknik yang sangat membantu penentuan jaringan komputer yang optimal.
4. Perkembangan dunia teknologi akan berkembang dengan sangat pesat seiring waktu. Perkembangan internet dan jaringan komputer juga akan ikut di dalam perkembangan teknologi tersebut.

VI. UCAPAN TERIMA KASIH

Alhamdulillah, serangkaian proses penyusunan makalah yang berjudul " Optimalisasi Jaringan Komputer Melalui Weighted Graph: Implementasi Spanning Tree dan Relasi Rekurens" di mata kuliah Matematika Diskrit telah penulis selesaikan dengan baik.

Penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada Bapak Dr. Ir. Rinaldi, M.T., sebagai dosen dan pembimbing yang telah memberikan arahan dan dukungan dalam penyusunan makalah ini. Terima kasih juga kepada semua pihak yang telah membantu dalam proses penyelesaian laporan ini, meskipun tidak dapat penulis sebutkan satu persatu.

Penulis berharap semoga hasil penulisan makalah ini dapat memberikan manfaat bagi kita semua dan mendorong kemajuan ilmu pengetahuan di masa yang akan datang. Di sisi lain, penulis menyadari bahwa makalah ini masih jauh dari sempurna. Oleh karena itu, penulis terbuka dan menghargai segala kritik dan saran yang konstruktif. Sekian dan terima kasih.

REFERENSI

[1] Munir, Rinaldi. (2024). Pengantar Matematika Diskrit. 8 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/00-Pengantar-Matematika-Diskrit-2024.pdf

[2] Munir, Rinaldi. (2024). Relasi dan Fungsi. 8 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/05-Relasi-dan-Fungsi-Bagian1-(2024).pdf

[3] Munir, Rinaldi. (2024). Deretan, Rekursi, dan Relasi Rekurens Bagian 1. 8 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/10-Deretan,%20rekursi-dan-relasi-rekurens-(Bagian1)-2024.pdf

[4] Munir, Rinaldi. (2024). Deretan, Rekursi, dan Relasi Rekurens Bagian 2. 8 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/11-Deretan,%20rekursi-dan-relasi-rekurens-(Bagian2)-2024.pdf

[5] Munir, Rinaldi. (2024). Graf Bagian 1. 8 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

[6] Munir, Rinaldi. (2024). Graf Bagian 2. 8 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf

[7] Munir, Rinaldi. (2024). Graf Bagian 3. 8 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/22-Graf-Bagian3-2024.pdf

[8] Munir, Rinaldi. (2024). Pohon Bagian 1. 8 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/23-Pohon-Bag1-2024.pdf

[9] Munir, Rinaldi. (2024). Pohon Bagian 2. 8 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/24-Pohon-Bag2-2024.pdf

[10] Munir, Rinaldi. (2024). Kompleksitas Algoritma Bagian 1. 8 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/25-Kompleksitas-Algoritma-Bagian1-2024.pdf

[11] Munir, Rinaldi. (2024). Kompleksitas Algoritma Bagian 2. 8 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/26-Kompleksitas-Algoritma-Bagian2-2024.pdf

[12] Liu, Q., Wang, J., Geng, C. (2024). Network Transformation of Telecontrol Device Based on Computer Network and Optimization of Acceptance Technology. 8 Januari 2025, dari https://doi.org/10.1109/BDICN62775.2024.00040

[13] Li, Hanju. (2021). Computer Network Connection Enhancement Optimization Algorithm Based on Convolutional Neural Network. 8 Januari 2025, dari https://doi.org/10.1109/NetCIT54147.2021.00063

[14] Bunaen, M.C., Pratiwi, H., Riti, Y.F. (2022). Penerapan Algoritma Dijkstra untuk Menentukan Rute Terpendek dari Pusat Kota Surabaya ke Tempat Bersejarah. 8 Januari 2025, dari https://doi.org/10.47233/jteksis.v4i1.407

[15] Dalem, Ida Bagus Gede Wahyu Antara. (2018). Penerapan Algoritma A* (Star) Menggunakan Graph untuk Menghitung Jarak Terpendek. 8 Januari 2025, dari https://media.neliti.com/media/publications/235453-penerapan-algoritma-a-star-menggunakan-g-fa0b1902.pdf

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, 8 Januari 2025

Muhammad Raihan Nazhim Oktana
13523021

LAMPIRAN

[1] Source Code:
https://github.com/RNXFreeze/Makalah-IF1220-MatematikaDiskrit