Penerapan Algoritma Branch and Bound dalam Memutuskan Pembelian Permainan pada Google Play Store

Vincent Prasetiya Atmadja – 13520099[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, J1l. Ganesha 10 Bandung 40132, Indonesia
[1]*13520099@std.stei.itb.ac.id*

Abstrak* —Algoritma Branch and Bound merupakan salah satu materi yang dipelajari dalam strategi algoritma dan mempunyai banyak penerapan. Salah satu penerapannya adalah mengoptimisasi keputusan yang mungkin diambil. Optimisasi ini dapat digunakan salah satunya dalam keputusan pembelian *video game

***Kata Kunci* — *Video Game, Branch and Bound,* Optimisasi**

I. PENDAHULUAN

Perkembangan teknologi di era globalisasi merupakan hal yang tidak terelakkan dan mempengaruhi berbagai bidang kehidupan. Dampaknya sangat kita rasakan dalam kehidupan sehari-hari. Pengaruhnya dapat dilihat mulai dari bidang komunikasi, transportasi, pendidikan, hingga permainan.

Permainan menurut Kamus Besar Bahasa Indonesia, memiliki arti sesuatu yang digunakan untuk bermain [1]. Bentuk-bentuk permainan semakin berkembang dan berubah seturut dengan adanya perkembangan globalisasi. Pada jaman dulu, permainan yang dimainkan umumnya berupa permainan tradisional, seperti kelereng, congklak, petak umpet, tarik-tambang, speak bola, dan berbagai permainan tradisional lainnya. Permainan ini umumnya dimainkan dengan memanfaatkan bentang alam dan bergantung pada lingkungan sekitar. Globalisasi membawa era baru dalam dunia permainan dengan munculnya permainan digital, atau yang mungkin lebih dikenal dengan istilah asing *video game.*

Video Game, atau permainan digital, umumnya merupakan sebuah aplikasi yang dibuat oleh developer kemudian didistribusikan melalui berbagai platform. Platform penyebaran ini bergantung pada media *video game* tersebut dimainkan. Untuk *video game* berbasis PC atau Personal Computer, umumnya disebarkan melalui platform Steam, Epic Store, atau bisa diunduh dari internet. Untuk *video game* berbasis Android, umumnya disebarkan melalui platform Google PlayStore.

Google PlayStore merupakan salah satu aplikasi bawaan dari Google yang dapat dimanfaatkan bukan hanya untuk mengunduh game, tetapi juga aplikasi-aplikasi lainnya, misalnya aplikasi koran, layanan pesan, *browser*, dan lain-lain. *Video Game* yang disediakan oleh Google PlayStore ada yang bersifat gratis dan berbayar.

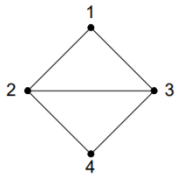
Tidak jarang, kita, sebagai manusia memiliki keinginan untuk memainkan *video game* berbayar. Untuk itu, ada perlunya kita memprioritaskan dan membuat pilihan dari berbagai *video game* yang ada yang dapat memberikan kepuasan secara maksimal dengan dana yang tersedia. Permasalahan ini merupakan permasalahan optimisasi dengan kendala biaya dan memaksimalkan kepuasan yang didapatkan. Oleh karena itu, persoalan ini dapat diselesaiakn dengan bantuan Algoritma Branch and Bound.

II. DASAR TEORI

A. Graf

Graf dapat didefinisikan sebagai struktur yang merepresentasikan objek-objek diskrit beserta dengan hubungan di antara objek-objek tersebut [2]. Secara matematis, graf dapat dituliskan sebagai pasangan himpunan (V, E) dengan,

- V = himpunan tidak kosong dari simpul-simpul, atau vertices, dapat dituliskan dengan $\{v_1, v_2, \ldots, v_n\}$.
- E = himpunan sisi, atau edges, yang menghubungkan sepasang simpul, dapat dituliskan $\{e_1, e_2, \ldots, e_n\}$.

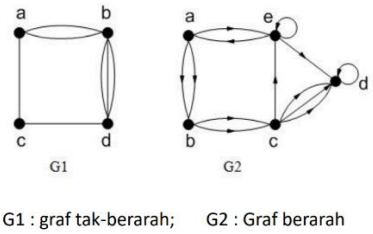


G_1

Gambar 2.1 Contoh Graf
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

Graf dapat dikelompokkan menjadi beberapa kelompok berdasarkan orientasi arah sisi graf. Berdasarkan orientasi sisi graf, graf dibagi menjadi 2 jenis, yaitu

1. Graf Tak-Berarah
   Graf tak-berarah, atau *undirected graph*, adalah graf yang sisinya tidak mempunyai orientasi arah.
2. Graf Berarah
   Graf berarah, atau *directed graph*, adalah graf yang setiap sisinya diberikan orientasi arah.

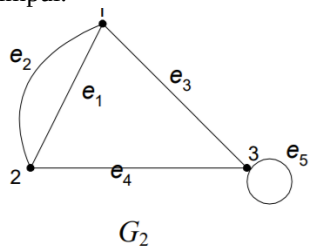


G1 : graf tak-berarah; G2 : Graf berarah

Gambar 2.2 Graf berdasarkan orientasi arah sisi graf
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

Ada beberapa terminologi atau istilah yang sering digunakan pada graf. Berikut adalah beberapa terminologi graf yang akan digunakan untuk makalah ini.

1. Ketetanggan (*adjacent)*
   Dua simpul akan dikatakan bertetangga apabila kedua simpul terhubung langsung. Misalkan dipunyai $e_1 = (v_1, v_2)$, $e_2 = (v_1, v_3)$, dan $e_3 = (v_2, v_4)$, maka v_1 dikatakan bertetangga dengan v_2 dan v_3, v_2 dikatakan bertetangga dengan v_4, tetapi v_1 dan v_3 tidak bertetangga dengan v_4.
2. Bersisian (*adjacent*)
   Suatu sisi dikatakan bersisian dengan simpul yang dihubungkannya. Misalkan dipunyai $e = (v_1, v_2)$, maka e dikatakan bersisian dengan v_1 dan v_2.
3. Graf kosong *(null graph)*
   Graf kosong adalah graf yang himpunan sisinya merupakan himpunan kosong.
4. Derajat *(degree)*
   Derajat adalah jumlah sisi yang bersisian dengan simpul.

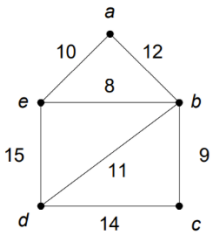


G_2

Gambar 2.3 Contoh derajat graf
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

Pada contoh graf G2 di atas, derajat simpul 1 adalah 3 atau d(1) = 3

5. Graf Berbobot *(Weighted Graph)*
   Graf berbobot adalah graf yang setiap sisinya memiliki harga atau bobot.



Gambar 2.4 Graf berbobot
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

B. Traversal Graf

Algoritma Traversal Graf merupakan algoritma mengunjungi simpul-simpul graf secara sistematik sebagai bentuk pencarian solusi [3]. Sebagai sebuah algoritma pencarian solusi, pencarian dapat dilakukan tanpa informasi maupun dengan informasi

1. Algoritma Pencarian Solusi Tanpa Informasi (*uninformed/blind search)*
   Algoritma ini melakukan pencarian solusi tanpa adanya informasi tambahan [3]. Beberapa algoritma traversal graf yang temasuk di dalamnya adalah *Breadth First Search (BFS), Depth First Search (DFS) Iterative Deeepening Search, Uniform Cost Search*.
2. Algoritma Pencarian Solusi Dengan Informasi (*informed Search)*
   Algoritma ini melakukan pencarian solusi berbasis heuristik [3]. Beberapa algoritma traversal graf yang termasuk di dalamnya adalah *Best First Search, A**.

Dalam proses pencarian traversal graf, dikenal dua pendekatan graf yang berbeda yaitu

1. Graf Statis
   Graf Stastis merupakan graf yang sudah terbentuk sebelum proses pencarian dilakukan. Graf Statis direpresentasikan sebagai struktur data.
2. Graf Dinamis
   Graf Dinamis merupakan graf yang terbentuk saat proses pencarian dilakukan. Graf Dinamis tidak tersedia sebelum pencarian, tetapi dibangun selama pencarian solusi.
   Pada Graf Dinamis ada beberepa representasi berkaitan yang akan dipakai [4].
   a. Pohon Ruang Status (*State Space Tree)*, merupakan jalur pohon yang dibangkitkan.
   b. Simpul, merupakan *state*. Apabila simpul berupa akar, maka simpul

merupakan *initial state*. Apabila simpul berupa daun, maka simpul merupakan *solution/goal state*.

c. Cabang, merupakan operator/langkah dalam persoalan
d. Ruang Status *(State Space)*, merupakan himpunan semua simpul
e. Ruang Solusi, merupakan himpunan semua status solusi.

Pada Graf Dinamis, pembangkitan status baru dilakukan dengan cara mengaplikasikan operator (langkah legal) kepada status (simpul) pada suatu jalur. Jalu dari simpul akar sampai ke simpul daun (*goal state)* menjadi rangkaian operator yang mengarah pada solusi persoalan.

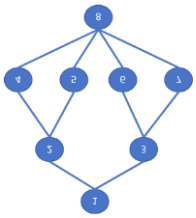
C. *Breadth First Search*

Breadth First Search merupakan algoritma traversal graf yang menekankan pada pencarian melebar. Seperti disampaikan di Traversal Graf di atas, ada dua pendekatan graf, yaitu graf statis dan graf dinamis. Hal ini juga berlaku pada *Breadth First Search (BFS)*. Pendekatan graf ini akan mempengaruhi bagaimana algoritma *BFS* dilakukan.

1. Graf Statis

   Pada graf statis, misalkan dipunyai suatu graf dan penelurusan akan dilakuakn mulai dari simpul v, makan algoritmanya adalah sebagai berikut.

   1. Kunjungi simpul v
   2. Kunjungi semua simpul yang bertetangga dengan simpul v terlebih dahulu
   3. Kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi.
   4. Ulangi proses 3 sampai semua simpul dikunjungi.



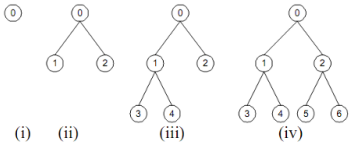
Gambar 2.5 BFS Graf Statis
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

Pada contoh graf di atas, apabila akan dilakukan *Breadth First Search* dimulai dari simpul 1, maka urutan simpul yang dikunjungi adalah 1, 2, 3, 4, 5, 6,7 , dan 8.

2. Graf Dinamis

   Pada Graf Dinamis, *Breadth First Search* mempengaruhi pembentukan pohon ruang status.

   1. Inisialisasi status awal sebagai akar, lalu tambahkan simpul anaknya.
   2. Semua simpul pada level d dibangkitkan terlebih dahulu sebelum simpul-simpul pada level d+1.



Gambar 2.6 BFS Graf Dinamis
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

D. *Branch and Bound*

Branch and Bound merupakan sebuah algoritma yang digunakan untuk persoalan optimisasi. Ini berarti, algoritma ini digunakan untuk meminimalkan atau memaksimalkan suatu fungsi objektif, dengan tidak melanggat Batasan persoalan. Algoritma ini merupakan penggabungan dair *Breadth First Search* dan *least cost search* [5].

Perbedaan antara *Branch and Bound* dan *Breadth First Search* terletak pada pilihan simpul yang akan dibangkitkan. Pada *BFS*, simpul akan dibangkitkan menurut urutan pembangkitannya atau mengikuti prinsip FIFO (*First In First Out)*. Sementara itu, pada *Branch and Bound*, proses pembangkitan didasarkan pada cost dari setiap simpul.

Setiap simpul akan diberikan nilai cost, misalnya ĉ(i),yang menyatakan taksiran lintasan termurah ke simpul status tujuan yang melalui simpul status i. Simpul berikutnya yang akan dibangkitkan akan berdasarkan *cost* ini. Pada kasus minimasi, cost yang paling kecil akan dipilih untuk dibangkitkan, sementara pada kasus maksimasi, cost yang paling besar akan dipiluh untuk dibangkitkan.

Dalam proses pembangkitan simpul, *Branch and Bound* juga memiliki pemangkasan simpul. Pemangkasan simpul akan dilakukan apabila

1. Nilai Simpul tidak lebih baik dari nilai terbaik sejauh ini (*the best solution so far)*
2. Simpul tidak merepresentasikan solusi yang *feasible* karena ada batasan yang dilanggar.
3. Solusi pada simpul hanya terdiri atas satu titik. Pada kasus ini, bandingkan nilai fungsi objektif dengan solusi terbaik saat ini, pilih yang terbaik.

Secara umum, algoritma *Branch and Bound* adalah sebagai berikut [5].

1. Masukkan simpul akar ke dalam *queue* Q. Jika simpul akar adalah simpul solusi (*goal node)*, maka solusi telah ditemukan. Jika hanya satu solusi yang diinginkan, maka stop.

2. Jika Q kosong, stop.
3. Jika Q tidak kosong, pilih dari *queue* Q simpul I yang mempunyai nilai cost ĉ(i),paling kecil. Jika terdapat beberapa simpul I yang memenuhi, pilih satu sembarang.
4. Jika I adalah simpul solusi, solusi telah ditemukan. Jika satu solusi saja yang diinginkan, stop pencarian. Pada persoalan optimasi dengan pendekatan *least cost search*, periksa cost semua simpul hidup. Jika costnya lebih besar dari cost simpul solusi, maka matikan simpul tersebut.
5. Jika simpul I bukan simpul solusi, maka bangkitkan semua anak-anaknya. Jika i tidak mempunyai anak, kembali ke langkah 2.
6. Untuk setiap anak j dari simpul i, hitung ĉ(j), dan masukkan semua anak-anak tersebut ke dalam Q.
7. Kembali ke langkah 2.

Cost untuk setiap simpul hidupnya, dapat dicari dengan formulai sebagai berikut.

$$c(i) = f(i) + g(i)$$

dengan

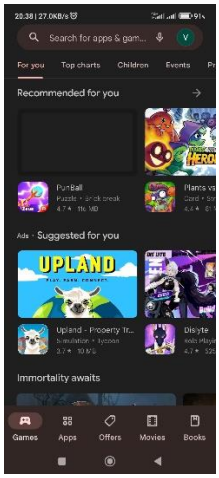
c(i) = ongkos untuk simpul i

f(i) = ongkos untuk mencapai simpul i dari akar

g(i) = ongkos untuk mencapai simpul tujuan dari simpul i.

E. *Google PlayStore*

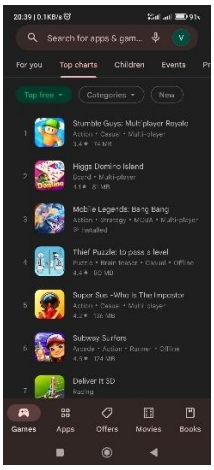
Google PlayStore merupakan salah satu aplikasi bawaan Google yang akan tersedia pada perangkat berbasis *Android*.



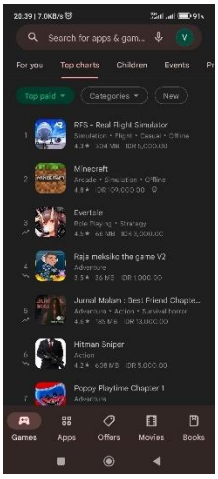
Gambar 2.7 Tampilan Utama PlayStore
Sumber : Arsip Penulis

Tampilan dia atas merupakan tampilan dari halaman utama dari *Google PlayStore*. Pada aplikasi ini, pengguna dapat mengunduh aplikasi yang tersedia. Aplikasi yang tersedia, bisa berupa aplikasi berbayar maupun aplikasi gratis.

Pada *Google PlayStore*, *rating* suatu aplikasikan ditandai dengan bintang dalam skala 5. Pengguna dapat memberikan review berupa bintang terhadap aplikasi. *Google PlayStore* menyediakan informasi *rating* tiap aplikasinya, berapa orang yang sudah memberikan *rating* dan berapa orang yang sudah mengunduhnya. Berdasarkan *rating*, *Google PlayStore* menyediakan chart untuk masing-masing *video game* berbayar maupun gratis.



Gambar 2.8 Chart Top Rated Free Games
Sumber : Arsip Penulis



Gambar 2.9 Chart Top Rated Paid Games
Sumber : Arsip Penulis

III. IMPLEMENTASI ALGORITMA BRANCH AND BOUND DALAM PEMBELIAN PERMAINAN

Permasalahan pemilihan *video game* yang akan dibeli, merupakan suatu persoalan optimisasi pemilihan dengan Batasan biaya dan memaksimalkan kepuasan yang bisa didapatkan dan meminimalkan pengeluaran. Dengan berdasarkan hal-hal ini, dapat dibangkitkan sebuah pohon ruang status, dengan definisi cost hidup, ruang solusi, dan objek-objek lebih lanjut.

Persoalan ini dapat dipandang juga sebagai suatu persoalan 1/0 Knapsack, dimana permainan yang

tersedia bisa diambil atau tidak. Pada persoalan ini, kendala yang ada adalah biaya yang tersedia, beban adalah harga permainan dan profit tingkan rating permainan yang didapatkan.

Algoritma *Branch and Bound* akan membangkitkan sebuah pohon biner, yang menggambarkan 1/0 Knapscak dengan cabang kiri menandakan *video game* ke I diambil dan cabang kanan menandakan *video game* ke I tidak diambil.

Pada contoh pengujian, penulis akan menggunakan data 5 game, berdasarkan top charts dari Google PlayStore. Harga akan dituliskan dalam satuan ribuan untuk menghindari ukuran terlalu besar. Rating akan dituliskan dalam rating 5 dan dalam persen. Sementara itu, penulis akan menggunakan budget 120 (120 000) dalam ribuan rupiah.

TABEL I. PERMAINAN

No	Nama	Harga	Rating (Skala 5)	Rating (%)
1	RFS – Real Fight Simulator	5	4.3	86
2	MineCraft	109	4.8	96
3	EverTale	3	4.5	90
4	Raja Meksiko The Game V2	1	3.5	70
5	Jurnal Malam: Best Friend Chapter 1	13	4.6	9.2

Dari data di atas, dapat dibuat suatu table rasio yang menggambarkan proporsi kepuasan, perbandingan antara rating dalam persen, dengan harganya. Rasio ini yang nantinya akan digunakan untuk perhitungan cost dari setiap simpul hidupnya.

TABEL II. PERMAINAN DAN RASIO

No	Nama	Harga	Rating (%)	Rasio
1	RFS – Real Fight Simulator	5	86	17,2
2	MineCraft	109	96	0,91
3	EverTale	3	90	30
4	Raja Meksiko The Game V2	1	70	70
5	Jurnal Malam: Best Friend Chapter 1	13	92	7,07

Perhitungan c(i) untuk setiap simpul, berdasarkan f(i) dan g(i) akan dihitung sebagai berikut

c(i) = f(i) + g(i)

dengan f(i) = c(i) simpul sebelumnya
g(i) = rasio simpul ini

Untuk memulai algoritma, perlu dibuat simpul akar yang menjadi basis dari pembangkitan pohon ruang status. Pada simpul akar, karena belum ada *video game* yang dipilih, maka c(i) = 0.

Untuk pembangkitan, simpul berikutnya, akan didasarkan pada apakah simpul tersebut dipilih atau tidak, dengan tetap memperhatikan fungsi kendala. Apabila fungsi tersebut dipilih, maka g(i) akan bernilai rasio, dan apabila tidak, g(i) akan bernilai 0. Berikut contoh pembangkitannya

1. *video game* 1 dipilih
   c(1) = 0 + 17.2 = 17.2
2. *video game* 1 tidak dipilih
   c(2) = 0 + 0 = 0

Pembangkitan simpul seperti langkah di atas akan diulang untuk setiap *video game*, sampai didapatkan maksimal rasio yang bisa didapatkan. Berikut merupakan source code yang penulis gunakan untuk mensimulasikan algoritma ini.

```
def solve(data, budget):
    firstNode = Node(0, 0, 0, False, [], 0, budget)

    front = firstNode
    for i in range(len(data)):
        parent1 = deepcopy(front.getParent())
        parent2 = deepcopy(front.getParent())

        parent1.append("True")
        parent2.append("False")

        pq.push(Node(i+1, data[i][0], data[i][1], False, parent2, front.find_cost(), front.getSisa()))
        pq.push(Node(i+1, data[i][0], data[i][1], True, parent1, front.find_cost(), front.getSisa()))

        front = pq.front()

    print(pq.front().getParent())
```

Gambar 3.1 Algoritma Branch and Bound
Sumber : Arsip Penulis

Pada source code di atas, pq merupakan sebuat Class Priorirty Queue yang berguna untuk menyimpan urutan dari Node yang akan dibangkitkan. Pengurutan Node didasarkan berdasarkan cost yang paling besar. Node merupakan Class buatan yang berguna untuk menyimpan informasi objek, beserta perhitungan yang dilakukan, seperti dapat dilihat pada gambar di bawah

```
class Node:

    def __init__(self, id, harga, rasio, chosen, parent, cost, totalSisa):
        self.id = id
        self.harga = harga
        self.rasio = rasio
        self.chosen = chosen
        self.parent = deepcopy(parent)
        self.cost = cost
        self.totalSisa = totalSisa

    def getParent(self):
        return self.parent

    def available(self):
        return self.totalSisa >= self.harga

    def getSisa(self):
        return self.totalSisa

    def find_cost(self):
        if(self.chosen and self.available()):
            self.totalSisa -= self.harga
            return self.cost + self.rasio
        elif(not self.available() and self.chosen):
            self.parent[len(self.parent) - 1] = "False"
            return self.cost
        else:
            return self.cost


    def __lt__(self, other):
        return self.find_cost() > other.find_cost()
```

Gambar 3.2 Node Classs
Sumber : Arsip Penulis

Berdasarkan program yang dijalankan di atas dan rancangan algoritma Branch and Bound di atas, akan didapatkan bahwa solusi yang diberikan adalah membeli *video game* 1, 2, 3, dan 4. Akan tetapi, apabila diperhatikan, ternyata solusi ini tidak optimal. Seharusnya solusi yang optimal adalah membeli *video* 1, 2, 3, dan 5. Hal ini terjadi karena cost function yang tidak optimal dan program yang belum mempertimbangkan simpul hidup setelah bertemu satu solusi.

IV.KESIMPULAN

Algoritma Branch and Bound dapat digunakan untuk membuat keputusan *video game* apa saja yang sebaiknya dibeli. Algoritma Branch and Bound berguna untuk optimisasi. Akan tetapi, berdasarkan uji coba penulis, ternyata pengambilan g(i) = rasio tidak berjalan baik. Alasannya adalah g(i) berlum mempertimbangkan semua kemungkinan dan rasio tidak menggambarkan kepuasan. Ada parameter lain yang perlu dipertimbangkan misalnya banyak unduh dan banyaknya yang memberi rating.

V. PENUTUP

Pertama, penulis menghaturkan puji syukur kepada Tuhan Yang Maha Esa karena atas rahmat-Nya, penulis dapat menyelesaikan makalah ini tepat waktu Penulis juga ingin mengucapkan terima kasih kepada orang tua penulis, yang telah mendukung penyusunan makalah ini. Penulis juga mengucapkan terima kasih kepada Bapak Rinaldi selaku pendamping kelas K 03 Strategi Algoritma. Tak lupa, penulis juga mengucapkan terima kasih kepada pihak-pihak lain yang mendukung tersusunnya makalah ini dan tidak bisa penulis sebutkan satu persatu.

Tak ada gading yang tak retak, begitu pula dengan makalah ini. Oleh karena itu, penulis memohon maaf apabila ada kesalahan kata ataupun ucapan. Semoga makalah ini dapat membawa manfaat bagi penulis dan pembaca.

REFERENSI

[1] KBBI. Diakses pada 22 Mei 2022, pukul 22.00 WIB, https://kbbi.web.id/main

[2] Rinaldi Munir. 2021. Graf (Bag.1). Diakses pada 22 Mei 2022 pukul 22.00 WIB, https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

[3] Rinaldi Munir.2021. BFS-DFS-2021(Bag.1). Diakses pada 23 Mei 2022 pukul 14.00 WIB, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf

[4] Rinaldi Munir.2021. BFS-DFS-2021(Bag.2). Diakses pada 23 Mei 2022 pukul 14.10WIB, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf

[5] Rinaldi Munir.2021. Algoritma-Branch-and-Bound(Bag.1). Diakses pada 23 Mei 2022 pukul 14.10WIB, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.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, 23 Mei 2022

Vincent Prasetiya Atmadja 13520099