Aplikasi Algoritma Backtracking dalam Pencarian Rute KRL Commuterline

Muhammad Rakha Athaya - 13520108
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520108@std.stei.itb.ac.id

***Abstrak*— Salah satu transportasi umum yang sering digunakan masyarakat Indonesia adalah layanan KRL Commuter Line yang disediakan oleh PT Kereta Commuter Indonesia (KAI Commuter) yang merupakan anak perusahaan dari PT Kereta Api Indonesia (PT KAI). Namun, peta rute KRL sering kali membingungkan untuk dibaca, sehingga terkadang mengakibatkan pengguna mengalami kesulitan untuk menentukan jalur yang paling efisien untuk mencapai tujuan. Dalam makalah ini, akan dibahas bagaimana algoritma *backtracking* dapat dimanfaatkan untuk mengatasi permasalahan tersebut. Algoritma *backtracking* (runut-balik) adalah algoritma yang berupa perbaikan dan *exhaustive search*. Berdasarkan implementasi yang telah dilaksanakan dalam makalah ini, algoritma *backtracking* dapat digunakan untuk mencari rute antara dua stasiun KRL Commuterline. Hal ini dapat menjadi solusi ketika mencari rute dengan melihat peta saja terlalu rumit atau sulit dilakukan. Namun, dalam makalah ini juga dibahas mengenai kekurangan dari algoritma backtracking pada penyelesaian masalah ini, yaitu solusi pertama yang didapat tidak selalu merupakan solusi yang paling optimal.**

Kata kunci*— *algoritma backtracking; KRL; peta rute; Commuterline; pemilihan rute

I. PENDAHULUAN

Bersamaan dengan menurunnya angka penularan virus COVID-19 di Indonesia, masyarakat mulai kembali memadati layanan transportasi umum. Salah satu transportasi umum yang sering digunakan masyarakat adalah layanan KRL Commuter Line yang disediakan oleh PT Kereta Commuter Indonesia (KAI Commuter) yang merupakan anak perusahaan dari PT Kereta Api Indonesia (PT KAI).

Namun, peta rute KRL sering kali membingungkan untuk dibaca, sehingga terkadang mengakibatkan pengguna mengalami kesulitan untuk menentukan jalur yang paling efisien untuk mencapai tujuan. Untuk mengatasi masalah ini, algoritma *backtracking* dapat dimanfaatkan. Algoritma *bactracking* adalah sebuah cara pemecahan masalah yang bersifat mangkus, terstruktur dengan baik, dan sistematis yang cocok untuk digunakan pada persoalan optimasi dan non-optimasi. Pengaplikasian algoritma *bactracking* dalam pemilihan rute KRL mampu membantu mempermudah masyarakat dalam memilih rute yang akan diambil sehingga membuat masyarakat merasa lebih nyaman dalam menggunakan transportasi umum, khususnya KRL.

II. LANDASAN TEORI

A. Algoritma Backtracking

Backtracking atau runut-balik adalah prosedur metodologis dimana beberapa sekuens keputusan dicoba dan diulang berkali-kali sampai ditemukan sekuens yang memenuhi kebutuhan atau mencapai solusi permasalahan. Misalkan dalam penelusuran labirin dapat dicoba sebuah sekuens arah gerakan yang diambil. Apabila sekuens gerakan tersebut berakhir pada jalan buntu, maka dicoba ulang sekuens lain sampai ditemukan sekuens yang berakhiran di ujung labirin.

Algoritma *backtracking* dapat diartikan menjadi salah satu dari dua penafsiran, yaitu:

- Salah satu fase dalam algoritma penelusuran *Depth-First Search*,
- Sebuah cara pemecahan masalah yang bersifat mangkus, terstruktur dengan baik, dan sistematis yang cocok untuk digunakan pada persoalan optimasi dan non-optimasi.

Sebagai metode pemecahan masalah, algoritma *backtracking* merupakan perbaikan dari algoritma *exhaustive search*. Jika pada exhaustive search seluruh kemungkinan solusi ditelusuri dan dievaluasi satu per satu, pada backtracking hanya pilihan yang mengarah ke solusilah yang akan ditelusuri.

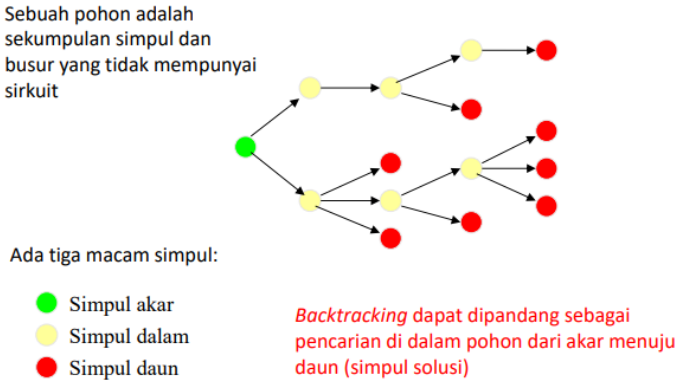
Algoritma *backtracking* ini diperkenalkan untuk pertama kalinya oleh D. H. Lehmer di tahun 1950 yang kemudian dibentuk uraian umumnya oleh R. J Walker, Golomb, dan Baumert.

Algoritma *backtracking* memiliki tiga properti dasar, yaitu:

- Solusi Persoalan, biasanya dinyatakan sebagai vector n-tuple. X = (x1, x2, x3, …, xn) dengan xi ϵ Si,
- Fungsi Pembangkit, dinyatakan sebagai predikat T(x1, x2, x3,…, xk-1) yang membangkitkan nilai solusi,

- Fungsi Pembatas, dinyatakan sebagai predikat B(x1, x2, x3,…, xk) yang memeriksa apakah pilihan mengarah ke solusi.

Semua kemungkinan solusi dari suatu permasalahan disebut ruang solusi. Ruang solusi biasanya diorganisasikan dalam struktur pohon berakar. Tiap simpul pada pohon menyatakan keadaan/status permasalahan, sedangkan sisi/cabang menyatakan nilai-nilai pada solusi persoalan. Lintasan dari akar ke daun akan menyatakan solusi yang mungkin dan seluruh lintasan dari akar ke daun akan menyatakan ruang solusi.



Gambar 1. Illustrasi ruang solusi

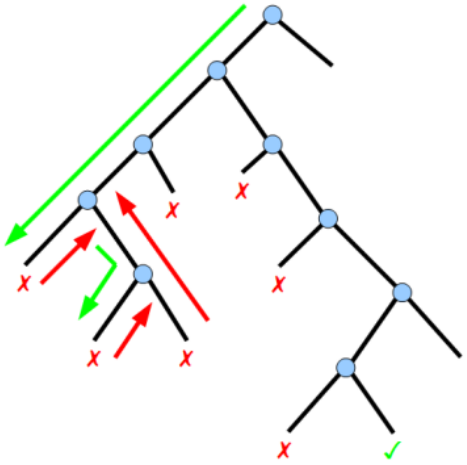
Sumber: www.cis.upenn.edu/.../35-backtracking.ppt

Aturan dalam pembangkitan simpul pada algoritma backtracking mengikuti aturan Depth-First Search. Terdapat tiga jenis simpul yang digunakan, yaitu:

- Simpul hidup, simpul yang telah dibangkitkan dan belum dimatikan,
- Simpul ekspan, simpul hidup yang sedang diperluas,
- Simpul mati, simpul yang tidak mengarah ke solusi sehingga simpul tersebut dimatikan (*prune*).

Setiap kali simpul ekspan diperluas, lintasan yang dibangun juga bertambah panjang. Jika lintasan tersebut tidak mengarah ke solusi, maka simpul ekspan tersebut akan dimatikan (*prune*) sehingga menjadi simpul mati. Cara kerja proses mematikan simpul ekspan adalah dengan menerapkan fungsi pembatas. Ketika sebuah simpul dimatikan, secara implisit kita telah memangkas simpul-simpul anaknya. Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian akan backtrack ke simpul pada tingkat di atasnya. Proses pencarian dihentikan apabila kita telah sampai pada simpul solusi (goal node).

Adanya proses mematikan (*pruning*) simpul tersebut menjadi pembeda utama antara algoritma backtracking dengan algoritma exhaustive search.



Gambar 2. Skema pencarian solusi backtracking

Sumber: https://www.w3.org/2011/Talks/01-14-steven-phenotype/

Algoritma backtracking dapat diimplementasikan dalam dua cara, yaitu secara iteratif dan secara rekursif. Skema umum algoritma backtracking versi iteratif adalah sebagai berikut:

```
procedure RunutBalikI(input k : integer)
{Mencari semua solusi persoalan dengan metode runut-balik; skema iteratif
Masukan: k, yaitu indeks komponen vektor solusi, x[k]. Diasumsikan x[1], x[2], ..., x[k – 1] sudah ditentukan nilainya.
Luaran: semua solusi x = (x[1], x[2], ..., x[n])
}
Algoritma:
  while k ≠ 0 do
    if terdapat nilai x[k] yang belum dicoba sedemikian sehingga x[k] ∈ T(x[1], x[2], …, x[k – 1]) and
       B(x[1], x[2], ... ,x[k]) = true then
       if (x[1], x[2], ... , x[k]) adalah lintasan dari akar ke simpul solusi then
          write(x[1], x[2], …, x[k])    { cetak solusi }
       endif
       k ← k + 1     { tentukan nilai x[k] selanjutnya }
    else
       k ← k – 1
    endif
  endwhile
                                                    Pemanggilan pertama kali: RunutBalikI(1)
```

20

Gambar 3. Skema umum algoritma backtracking iteratif

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-backtracking-2021-Bagian1.pdf

Sementara itu, skema umum algoritma backtracking versi rekursif adalah sebagai berikut:

```
procedure RunutBalikR(input k : integer)
{Mencari semua solusi persoalan dengan metode runut-balik; skema rekursif
Masukan: k, yaitu indeks komponen vektor solusi, x[k]. Diasumsikan x[1], x[2], ..., x[k – 1] sudah ditentukan nilainya.
Luaran: semua solusi x = (x[1], x[2], ..., x[n])
}
Algoritma:
  for setiap x[k] ∈ T(x[1], x[2], …, x[k – 1]) do
      if B(x[1], x[2], ... ,x[k]) = true then
         if (x[1], x[2], ... , x[k]) adalah lintasan dari akar ke simpul solusi then
            write(x[1], x[2], …, x[k])    { cetak solusi }
         endif
         if k < n then
            RunutBalikR(k+1)    { tentukan nilai untuk x[k+1]}
         endif
      endif
  endfor
                                                    Pemanggilan pertama kali: RunutBalikR(1)
```

18

Gambar 4. Skema umum algoritma backtracking rekursif

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-backtracking-2021-Bagian1.pdf

Dalam skema rekursif tersebut, setiap simpul bukan daun dalam pohon ruang status terasosiasi dengan sebuah pemanggilan rekursif. Jika jumlah simpul dalam pohon ruang status adalah 2^n atau n!, maka pada kasus terburuk, algoritma runut-balik membutuhkan waktu dalam $O(p(n)2^n)$ atau $O(q(n)n!)$, dengan $p(n)$ dan $q(n)$ adalah polinom derajat n yang menyatakan waktu komputasi setiap simpul.

Contoh permasalahan yang dapat diselesaikan menggunakan algoritma backtracking adalah seperti persoalan N-ratu (*the N-queens problem*), permasalahan labirin, *integer knapsack problem*, *sum of subsets problem*, pewarnaan graf, dan lain-lain.

B. KRL Commuter Line

KRL Commuter Line merupakan layanan Kereta Rel Listrik (KRL) komuter yang dikelola oleh PT Kereta Commuter Indonesia (KAI Commuter) yang merupakan anak perusahaan dari PT Kereta Api Indonesia (PT KAI). Pada tahun 1925, KRL mulai beroperasi di wilayah Jakarta, dan kemudain berkembang dan melayani rute komuter di wilayah Jabodetabek serta lintas Yogyakarta–Solo. Sejak tahun 1970-an, komuter ini diberi nama KRL Jabotabek, namun karena adanya pemekaran Kota Depok pada 1999, KRL Jabotabek diberi nama alternatif KRL Jabodetabek dengan Divisi Jabotabek sebagai operator KRL. Pada 2008, layanan KRL dioperasikan oleh PT KAI Commuter Jabodetabek yang kini menjadi KAI Commuter (PT. Kereta Commuter Indonesia, 2017).

III. IMPLEMENTASI ALGORITMA BACKTRACKING

Berdasarkan penjelasan di bagian sebelumnya, algoritma backtracking dapat digunakan untuk mencari rute antara dua stasiun KRL commuterline. Pada bagian ini akan dipaparkan mengenai proses implementasi algoritma backtracking, strategi pencarian solusi, dan hasil pengujian dengan program yang mengaplikasikan algoritma backtracking.

A. Pemetaan Properti Umum Algoritma Backtracking

Terdapat tiga properti umum dalam implementasi algoritma backtracking, yaitu solusi persoalan, fungsi pembangkit, dan fungsi pembatas. Dalam usaha pencarian rute KRL commuterline, berikut adalah pemetaan komponen-komponen tersebut:

- Solusi Persoalan

  Solusi persoalan dalam pencarian rute KRL commuterline berupa sebuah list yang berisi stasiun-stasiun yang dilewati pada rute solusi. List solusi tersebut dimulai dari stasiun keberangkatan dan diakhiri dengan stasiun tujuan.

  RouteSolution = [S0, S1, … Sn]

  Dengan S0 adalah stasiun keberangkatan dan Sn adalah stasiun tujuan.

- Fungsi Pembangkit

  Fungsi pembangkit dalam pencarian rute KRL commuterline berupa fungsi yang membangkitkan setiap adjacent stasiun dari posisi stasiun hidup saat ini.

  Penggambaran fungsinya adalah sebagai berikut:

```
procedure expand(i: Node, v: List<Node>)
{
i adalah stasiun saat ini.
for each i.adjacent do
...
end
}
```

- Fungsi Pembatas

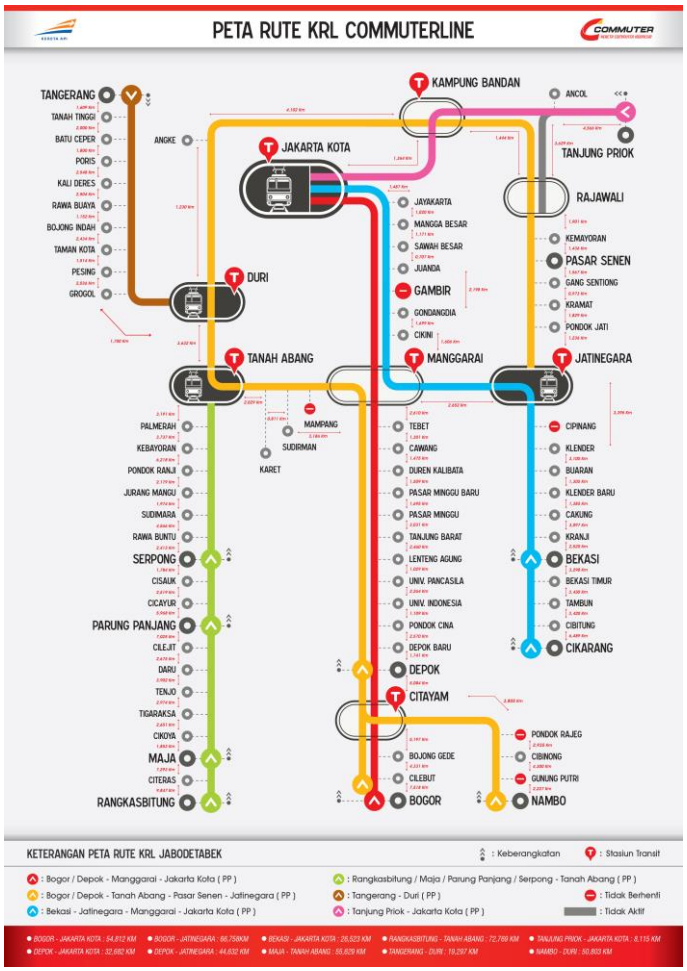
  Fungsi pembatas dalam pencarian rute KRL commuterline berupa fungsi yang menerima masukkan posisi stasiun saat ini dan *list of visited stasiun* kemudian menentukan apabila dari stasiun saat ini masih bisa berlanjut ke stasiun lain (yang belum visited) atau tidak. Apabila tidak maka akan mengembalikan nilai boolean tanda stasiun dimatikan.

  Penggambaran fungsinya adalah sebagai berikut:

```
function limit(i: Node, v: List<Node>)→ boolean
{
i adalah stasiun saat ini.
if semua adjacent stasiun dari i ada dalam v (visited), maka limit return true (matikan)
else return false (hidup)
}
```

B. Pemetaan Properti Umum Algoritma Backtracking

Jalur KRL commuterline memiliki ukuran yang besar dan kompleksitas yang tinggi. Hal tersebut menjadi standar karena jalur-jalur KRL mesti menghubungkan stasiun-stasiun penting di Jabodetabek.



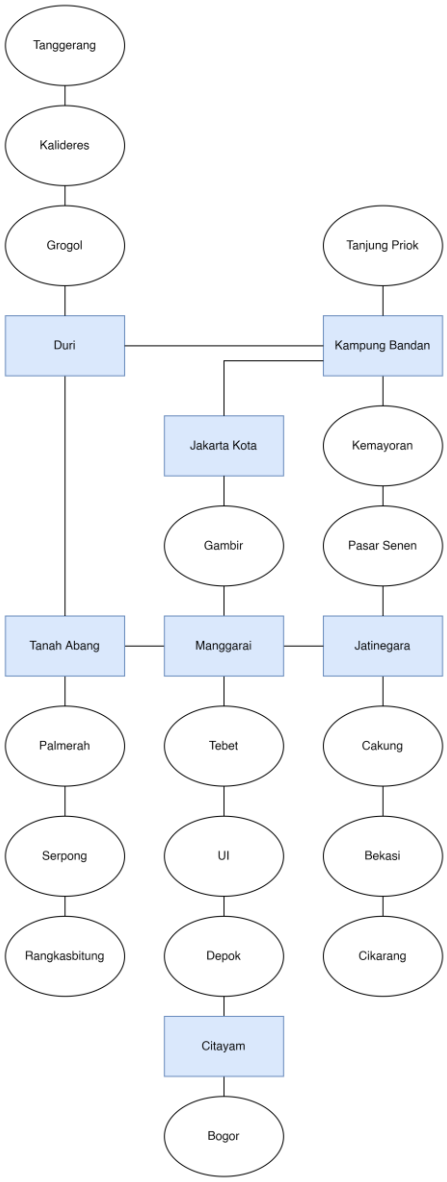
Gambar 5. Peta Rute KRL Commuterline

Sumber: www.krl.co.id

Pada makalah ini, pencarian rute tidak dilakukan pada peta asli rute KRL commuterline melihat kompleksitas dan jumlah stasiun yang harus ditambahkan. Proses pencarian rute menggunakan algoritma backtracking akan dilakukan pada model dari jalur KRL commuterline.

Jalur KRL commuterline dimodelkan ke dalam struktur data graf tidak berarah. Setiap stasiun menjadi sebuah simpul pada graf. Stasiun yang saling berhubungan digambarkan menjadi dua buah simpul yang dihubungkan oleh sebuah sisi. Jumlah stasiun yang dimasukkan ke dalam model graf tersebut lebih sedikit dari jumlah stasiun yang ada. Namun, untuk setiap stasiun transit dan hubungan antar stasiunnya tidak ada yang dirubah, sehingga model graf yang lebih kecil ini bisa merepresentasikan keadaan dari jalur KRL commuterline yang sebenarnya.

Berikut adalah model graf dari jalur KRL commuterline:



Gambar 6. Graf Model Rute KRL Commuterline

Adjacency List dari graf di atas adalah sebagai berikut:

Stasiun	Adjacent Stasiun
Citayam	Bogor, Depok
Manggarai	Jatinegara, Tanah Abang, Gambir
Jatinegara	Manggarai, Cakung, Pasar Senen
Tanah Abang	Manggarai, Palmerah, Duri
Duri	Tanah Abang, Kampung Bandan, Grogol
Jakarta Kota	Gambir, Kampung Bandan
Kampung Bandan	Jakarta Kota, Duri, Tanjung Priok, Kemayoran

Rangkasbitung	Serpong
Serpong	Rangkasbitung, Palmerah
Palmerah	Serpong, Tanah Abang
Tanggerang	Kalideres
Kalideres	Tanggerang, Grogol
Grogol	Kalideres, Duri
Tanjung Priok	Kampung Bandan
Gambir	Jakarta Kota, Manggarai
Tebet	Manggarai, UI
UI	Tebet, Depok
Depok	UI, Citayam
Bogor	Citayam
Cikarang	Bekasi
Bekasi	Cikarang, Cakung
Cakung	Bekasi, Jatinegara
Pasar Senen	Jatinegara, Kemayoran
Kemayoran	Kampung Bandan, Pasar Senen

C. Strategi Pencarian Solusi

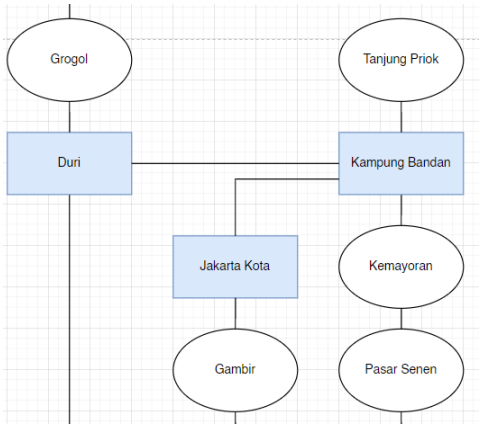
Strategi umum pencarian rute KRL commuterline adalah:

- Masukkan stasiun awal ke dalam visited stasiun.
- Bangkitkan salah satu stasiun yang adjacent dengan stasiun saat ini. Bila stasiun tersebut adalah stasiun tujuan, pencarian selesai. Apabila bukan, masukkan stasiun tersebut ke dalam visited stasiun.
- Bangkitkan kembali salah satu adjencent stasiun dari stasiun yang sebelumnya dibangkitkan. Apabila tidak ada adjacent stasiun yang belum visited, maka fungsi batas aktif dan dilakukan backtracking ke stasiun sebelumnya.
- Langkah diulang sampai ditemukan stasiun tujuan atau semua stasiun telah dikunjungi (stasiun tujuan tidak ditemukan).

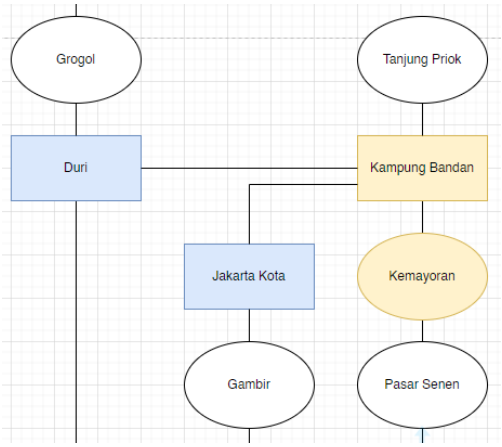
D. Pencarian Rute Menggunakan Algoritma Backtracking

Dengan menggunakan strategi dan komponen yang telah didefinisikan sebelumnya, maka algoritma backtracking sudah dapat digunakan untuk mencari rute antara dua stasiun KRL commuterline. Berikut ini adalah contoh penyelesaian dengan algoritma backtracking.

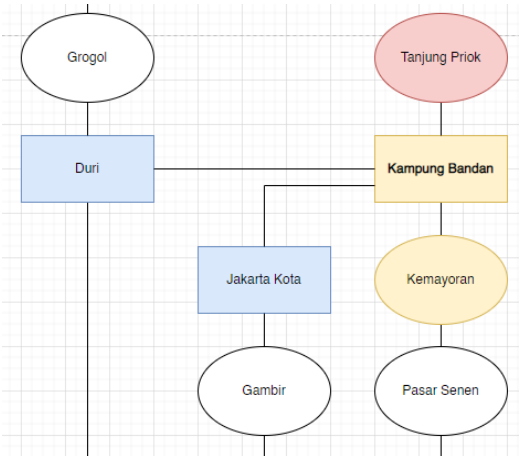
1. Kemayoran – Duri



Pencarian dimulai dari stasiun Kemayoran. Adjacent stasiun dari Kemayoran adalah Kampung Bandan dan Pasar Senen. Fungsi pembangkit dipanggil, pada kasus ini yang dibangkitkan adalah Kampung Bandan. Kemayoran dan Kampung Bandan ditandai sebagai visited.



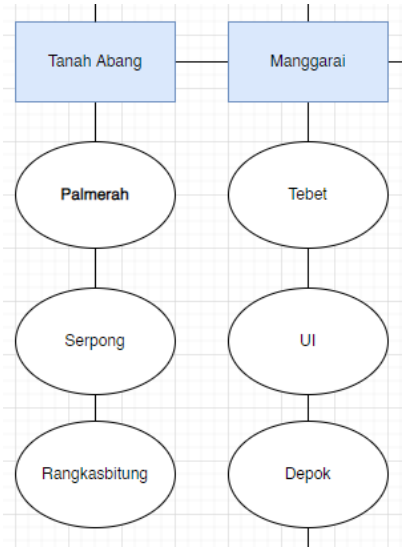
Selanjutnya yang terpanggil semisal adalah Tanjung Priok. Ketika sedang berada di Tanjung Priok, fungsi pembatas aktif karena Tanjung Priok tidak memiliki adjacent stasiun yang belum visited. Tanjung Priok dimatikan dan backtracking kembali ke Kampung Bandan.



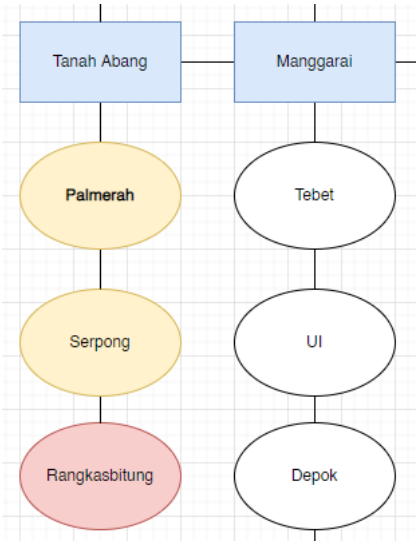
Fungsi pembangkit kembali dipanggil di Kampung Bandan. Kali ini, yang terpanggil adalah Duri. Karena Duri adalah goal

node yang dicari, maka pencarian rute selesai dengan RouteSolution = [Kemayoran, Kampung Bandan, Duri]

2. Palmerah – Manggarai



Pencarian dimulai dari stasiun Palmerah. Adjacent stasiun dari Palmerah adalah Serpong dan Tanah Abang. Fungsi pembangkit dipanggil, pada kasus ini yang dibangkitkan adalah Serpong. Palmerah dan Serpong ditandai sebagai visited. Selanjutnya yang dipanggil adalah Rangkasbitung sebagai sisa adjacent stasiun dari Serpong yang belum visited.



Di Rangkasbitung, fungsi pembatas aktif karena Rangkasbitung tidak memiliki adjacent stasiun yang belum visited. Rangkasbitung dimatikan dan diakukan backtracking kembali ke Palmerah, karena Palmerah masih bisa dibangkitkan. Selanjutnya dibangkitkan Tanah Abang, untuk pembangkitan selanjutnya dibangkitkan Manggarai sebagai salah satu adjacent stasiun di Tanah Abang. Karena Manggarai adalah goal node, maka pencarian rute selesai dengan RouteSolution = [Palmerah, Tanah Abang, Manggarai]

IV. KESIMPULAN

Algoritma *backtracking* (runut-balik) adalah algoritma yang berupa perbaikan dan *exhaustive search*. Berdasarkan implementasi yang telah dilaksanakan, algoritma backtracking dapat digunakan untuk mencari rute antara dua stasiun KRL commuterline. Hal ini dapat menjadi solusi ketika mencari rute dengan melihat peta saja terlalu rumit atau sulit dilakukan. Namun, terdapat kekurangan dari algoritma *backtracking* pada penyelesaian masalah ini, yaitu solusi pertama yang didapat tidak selalu merupakan solusi yang paling optimal.

VIDEO LINK AT YOUTUBE

https://youtu.be/9f9ispKU80w

UCAPAN TERIMA KASIH

Penulis mengucapkan puji syukur kepada Tuhan Yang Maha Esa karena atas rahmat dan ridha-Nya penulis dapat menyelesaikan makalah ini dengan baik.

Tidak lupa juga penulis mengucapkan terima kasih kepada Bapak dan Ibu dosen pengajar IF2210 Strategi Algoritma yang telah membimbing dan memberikan ilmu kepada segenap mahasiswanya termasuk penulis.

REFERENSI

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-backtracking-2021-Bagian1.pdf, diakses tanggal 22 Mei 2022 pukul 21.20

[2] PT. Kereta Commuter Indonesia. 2017. KAI Commuter. https://www.krl.co.id/ Diakses pada 23 Mei 2022

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, 20 Mei 2022

Muhammad Rakha Athaya
13520108