Penerapan Search Tree pada Penyelesaian Masalah Penentuan Jalur Kota Terpendek.

Arnold Nugroho Sutanto - 13507102[1)]

1) Jurusan Teknik Informatika ITB, Bandung 40132, email: if17102@students.if.itb.ac.id

Abstraksi- *Suatu permasalahan seperti penentuan jalur kota akan membentuk suatu ruang solusi di mana ruang solusi ini dapat diorganisasikan ke dalam suatu struktur pohon. Pada makalah ini, kita akan membahas penentuan suatu rute jalan terpendek menuju kota tujuan dengan pencarian suatu jalur solusi pada pohon dengan menggunakan pendekatan pencarian buta (tanpa informasi). Pencarian buta sendiri memiliki dua algoritma utama yaitu Breadth First Search dan Depth First Search. Berbeda dengan DFS,BFS menghasilkan solusi yang optimum. Solusi optimum ini diperlukan dalam penyelesaian masalah penentuan jalur kota terpendek. Tetapi solusi ini bukanlah solusi akhir dikarenakan solusi optimum yang dimaksud dalam BFS hanyalah solusi tersingkat dari state awal. Menghadapi permasalahan jalur kota yang memiliki biaya/ bobot (jarak) antar state (kota) yang berbeda, solusi tersingkat bukanlah solusi biaya terkecil. Oleh karena itu, untuk mendapatkan solusi jalur kota terpendek kita akan menggunakan pemodifikasian dari algoritma BFS yang dinamakan algoritma branch and bound sebagai algoritma yang juga memperhatikan bobot antar state.*

Kata kunci *: Penentuan jalur kota terpendek, Pohon pencarian, BFS, Branch and Bound*

1. PENDAHULUAN

Dalam kehidupan modern ini, bidang transportasi adalah bidang yang esensial untuk kelanjutan kehidupan sehari-hari. Hal ini membuat Permasalahan Penentuan Jalur Kota terpendek yang sebenarnya merupakan masalah lama namun tetap dianggap krusial. Permaslahan ini ditambah lagi dengan jalur kota yang semakin kompleks saat ini, sehingga banyak algoritma yang dibuat untuk menyelesaikan permasalahan ini, seperti algoritma Djikstra dan Exhaustion Enumeration. Pada makalah ini kita akan mencoba menggunakan Pohon Pencarian dalam menyelesaikan masalah ini.

Penetuan jalur kota terpendek ini dapat diaplikasikan tidak hanya untuk menentukan rute perjalanan, tetapi juga untuk pembangunan jalan dengan biaya yang minimum atau pembuatan rel kereta api.

2. PROBLEM SOLVING DENGAN SEARCHING

2.1. Konsep Dasar

Problem yang dimaksud di sini adalah deviasi antara current state dengan goal. Domain problem dimodelkan ke dalam state space searching (ruang pencarian) berisi kumpulan state berbeda. State adalah sebuah kondisi yang mungkin dalam obyek permasalahan tersebut.

Problem solving sendiri adalah bagaimana bergerak dari current state ke arah goal. Langkah awal dalam problem solving adalah mencari state yang cocok dengan state awal dan kemudian mecari rangkaian next state dari state awal yang cocok dan yang bisa mengantarkan pada state goal.

Konsep problem solving dengan tree searching merepresentasikan ruang pencarian dalam suatu problem menjadi struktur pohon yang dinamakan pohon pencarian. Problem solving dilakukan dengan mengunjungi simpul-simpul secara traversal dalam pohon tersebut.

Pohon pencarian adalah sebuah data struktur yang terdiri atas sebuah simpul induk utama, disebut root, tempat dimulainya pencarian. Setiap simpul dapat memiliki satu atau lebih simpul anak. Data sturktur suatu simpul :

1. Deskripsi keadaan
2. Pointer ke parent
3. Kedalaman simpul
4. Operator yang membuka simpul ini
5. Biaya total dari simpul awal sampai simpul ini

Setiap algoritma pencarian akan menyimpan semua simpul dalam pohon dalam suatu daftar List yang disebut sebagai fringe. Fringe ini dipakai sebagai alat untuk menunjukkan simpul yang telah dibuka dan siap untuk dieksplorasi.

2.2. Stratergi-strategi Pencarian

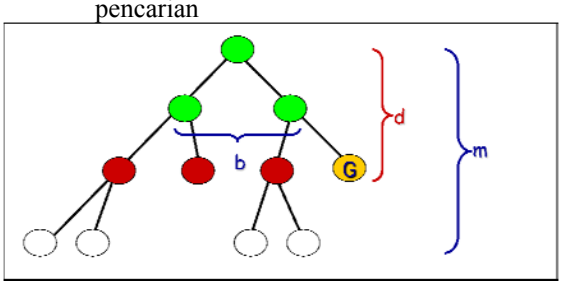
Strategi pencarian didefinisikan berdasarkan penentuan urutan pemrosesan node. Strategi ini dievaluasi berdasarkan beberapa factor, yaitu :

1. Kelengkapan : apakah dapat diberikan garansi bahwa solusi terdapat dalam ruang pencarian

2. Optimasi : apakah solusi yang akan didapatkan memerlukan biaya minimal
3. Kompleksitas waktu : jumlah nodes yang diproses
4. Kompleksitas memori : jumlah maksimum nodes di memori

Terminologi untuk kompleksitas waktu dan memori adalah

1. b: pencabangan maksimal dari pohon pencarian
2. d : kedalaman dari solusi terendah
3. m : kedalaman maksimum dari ruang pencarian



Strategi pencarian yang akan dibahas pada makalah ini hanya menggunakan informasi yang tersedia pada definisi masalah atau yang disebut dengan pencarian buta (uninformed). Berbagai strategi pencarian uninformed adalah:

1. Breadth-first search (BFS)
2. Uniform-cost search
3. Depth-first search (DFS)
4. Depth-limited search
5. Depth First Iriterative Deepening (DFID)

Pada makalah ini, kita hanya membahas algoritma BFS yang memiliki optimasi sehingga dapat kita gunakan untuk menyelesaikan permasalahan penentuan jalur kota. Algoritma BFS ini sendiri akan sedikit dimodifikasi karena bobot dari setiap jalur yang berbeda.

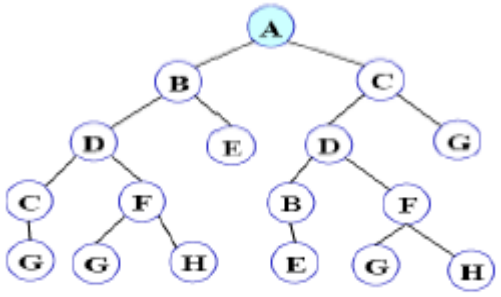
2.3. Breadth-first search (BFS)

Algoritma dasar dari BFS adalah

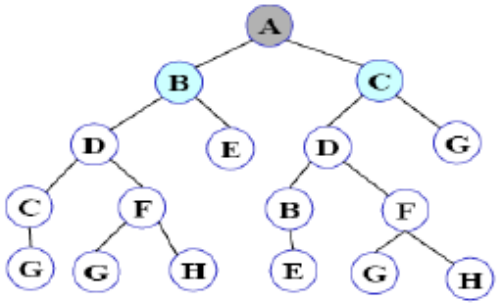
Ambil fringe sebagai list berisi keadaan awal
Loop
    If fringe kosong return failure
    Node <- remove first (fringe)
    If Node is a goal
    then return jalur dari keadaan awal sampai Node
    else buka semua suksesor Node dan (satukan Nodes baru ke dalam fringe) tambahkan Nodes baru ke akhir fringe
End Loop

Ciri utama dari algoritma BFS ini adalah memprioritaskan node se level. Pencarian akan melebar, mengikur pembukaan semua suksesor dari sebuah simpul. Simpul baru dalam BFS diletakkan pada akhir fringe. Dengan demikian, Fringe memiliki model antrian FIFO (first in first out) yang tidak lain berupa queue.

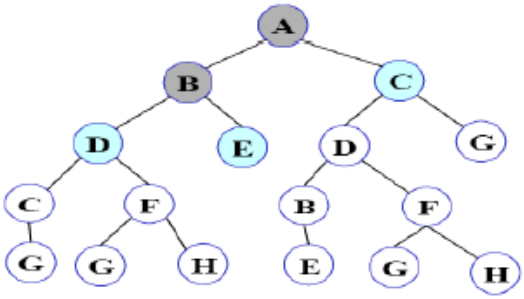
Ilustrasi dari BFS dapat kita lihat sebagai berikut dengan Goal state adalah G.



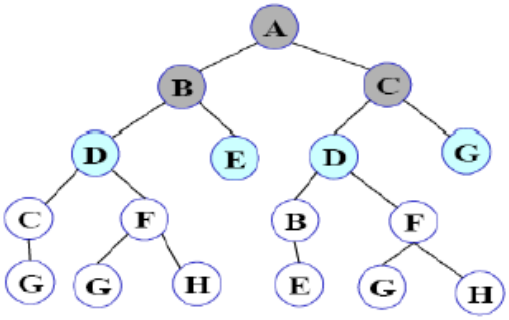
Fringe : A
Pada awalnya hanya ada satu simpul awal dalam fringe yaitu A.



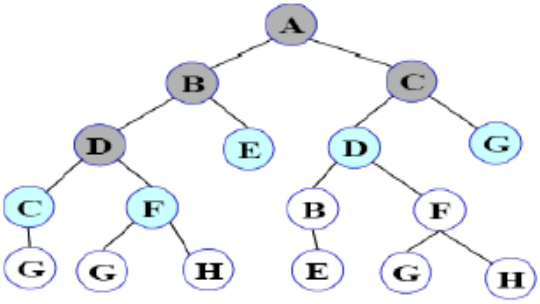
Fringe : B C
Simpul A dikeluarkan dari fringe dan membuka semua suksesornya, yaitu B dan C sehingga B dan C masuk dalam antrian fringe.



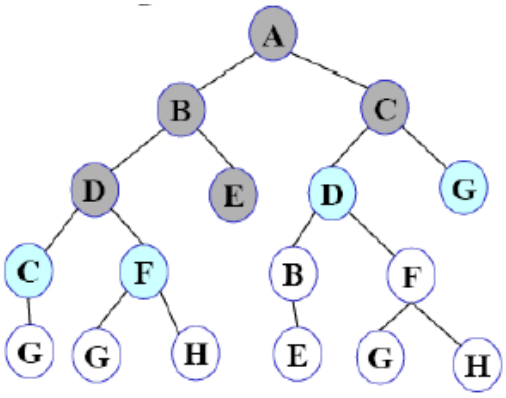
Fringe : C D E
Simpul B dibuka dan dikeluarkan dari fringe, dan membuka suksesor-suksesornya , yaitu D dan E. D dan E kemudian dimasukkan pada antrian fringe. Proses ini berlanjut hingga simpul Goal (G) berada pada antrian di awal dan siap untuk dibuka.



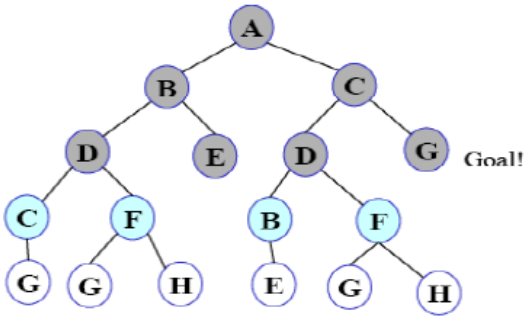
Fringe : D E D G



Fringe : E D G C F



Fringe : D G C F



Fringe : **G** C F B F
Simpul G dibuka dan ternyata merupakan Goal, algoritma BFS berhenti.

Evaluasi dari BFS :

1. Kelengkapan :
   Ya, karena semua kemungkinan dalam satu tingkat kedalaman akan diekspan semuanya.
2. Optimasi :
   BFS akan menemukan solusi dengan jalur tersingkat.
3. Kompleksitas waktu : $1+b+b^2+b^3+\ldots+b^d=O(b^d)$
4. Kompleksitas ruang :
   $O(b^d)$ karena menyimpan semua node di memori

3. BRANCH AND BOUND

3.1. Konsep BFS yang disesuaikan untuk masalah penentuan jalur kota terpendek

Seperti yang telah disebutkan sebelumnya, BFS memiliki solusi tersingkat tetapi dalam masalah penentuan jalur kota yang biaya (jarak) menuju state selanjutnya berbeda-beda, solusi tersingkat belum tentu menjadi solusi yang memiliki biaya terendah (jarak terpendek). Oleh karena itu, konsep BFS ini akan sedikit dimodifikasi untuk mencapai tujuan tersebut. Modifikasi BFS ini dinamakan Branch and Bound.

Inti penerapan algoritma Branch and Bound pada penentuan jalur kota ini adalah adanya batas atas biaya, yang memperbolehkan pencarian BFS pada node selanjutnya selama biaya total dari node tersebut kurang dari batas atas yang ditentukan, meskipun node Goal sudah ditemukan. Batas atas yang dimaksudkan adalah jumlah biaya dari node goal yang sudah ditemukan terlebih dahulu. Algoritma ini akan berhenti jika semua node dalam fringe memiliki biaya di atas batas atas yang ditentukan.

Untuk lebih jelasnya, ada baiknya jika kita melihat algoritma ini per langkah.

Langkah 1. Inisiasi yang sama dengan inisiasi BFS, yaitu penentuan simpul awal. Selain itu, sebagai inisiasi, batas atas biaya ditentukan tak hingga (∞).

Langkah 2. Melakukan BFS sampai ditemukan simpul Goal. Goal tersebut disimpan kemudian batas atas diubah menjadi biaya dari node Goal tersebut.

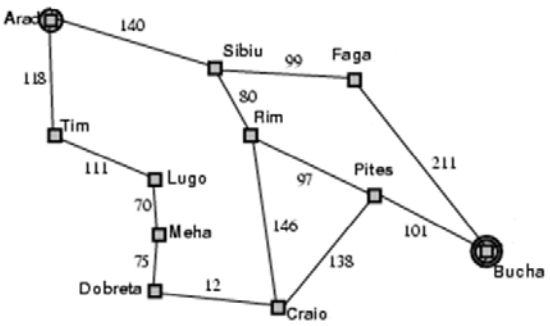
Langkah 3. Melakukan pengecekan pada setiap node pada fringe. Jika biaya dari node tersebut melebihi atau sama dengan batas atas, maka node tersebut diacuhkan (dihentikan pencariannya) dengan dibuang dari fringe. Node lain yang memiliki biaya di bawah batas atas yang sudah ditentukan, tetap dilakukan pencarian selanjutnya.

Langkah 4. Pencarian dilakukan hingga ditemukan node Goal baru yang memiliki biaya di bawah batas atas atau ada node dalam fringe yang memiliki biaya di atas batas atas. Langkah 4.1 Jika saat melakukan pencarian ditemukan node goal lain yang memiliki biaya di bawah batas atas sebelumnya, batas atas diubah menjadi biaya dari goal yang baru. Kemudian kembali ke langkah 3.

Langkah 4.2 Jika saat melakukan pencarian selanjutnya diketemukan ada node dalam fringe yang memliki biaya diatas batas atas, node tersebut dihilangkan dari fringe. Pencarian dilanjutkan hingga fringe menjadi kosong dan node yang daripadanya didapati batas atas biaya terakhir merupakan goal terpendek. Pencarian dihentikan.

3.2. Pengujian algoritma Branch and Bound pada penentuan jalur kota

Kita mulai dengan mendefinisikan masalah seperti berikut

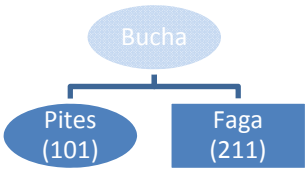


Ini adalah peta Romania, Anda disuruh menuju kota Arad dari kota Bucha untuk mengantarkan Pizza. Karena Pizza harus sampai pada konsumen hangat-hangat, jadi sebisa mungkin gunakanlah rute terpendek menuju kota Arad.

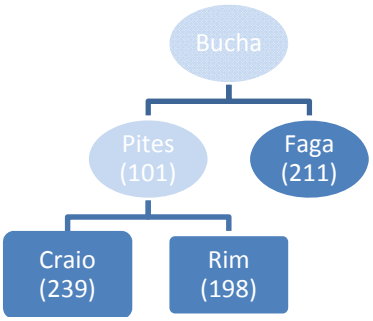
Langkah 1
Kita ambil state awal adalah Bucha. Bucha memiliki pilihan jalur yaitu menuju Pites dan Faga. Sehingga kita representasi pohon pencarian sebagai berikut

Fringe : Bucha(0)
Batas atas : ∞

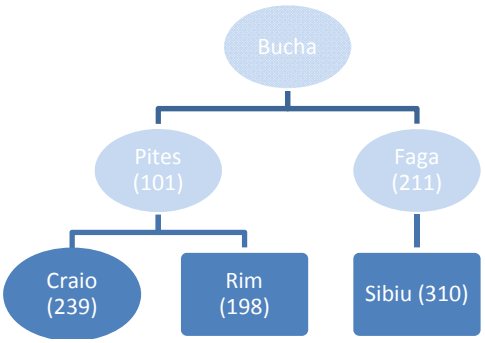
Langkah 2
Kita melakukan BFS seperti yang telah di jelaskan sebelumnya.



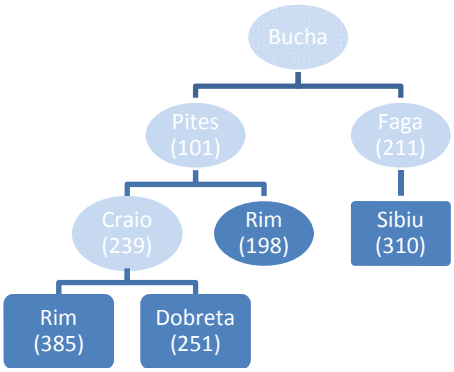
Fringe : Pites(101) Faga(211)



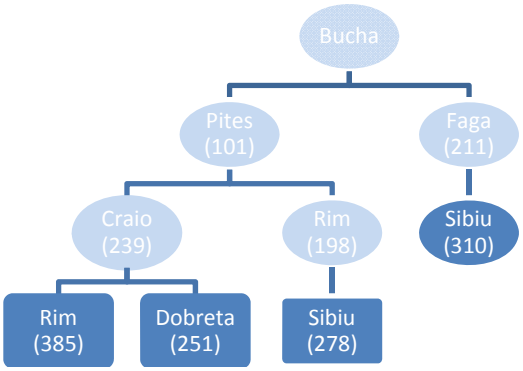
Fringe : Faga(211) Craio(239) Rim(198)



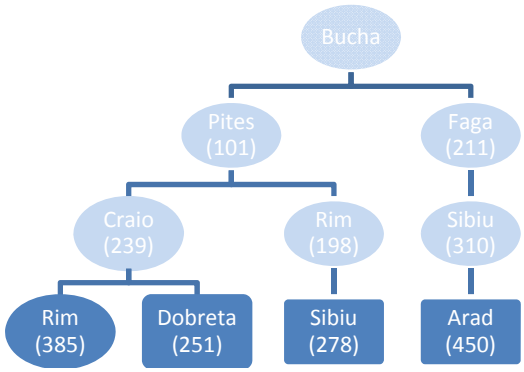
Fringe : Craio(239) Rim(198) Sibiu(310)



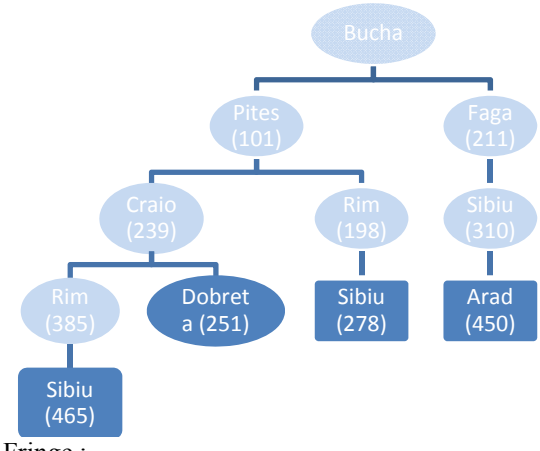
Fringe :
Rim(198) Sibiu(310) Rim(385) Dobreta(251)



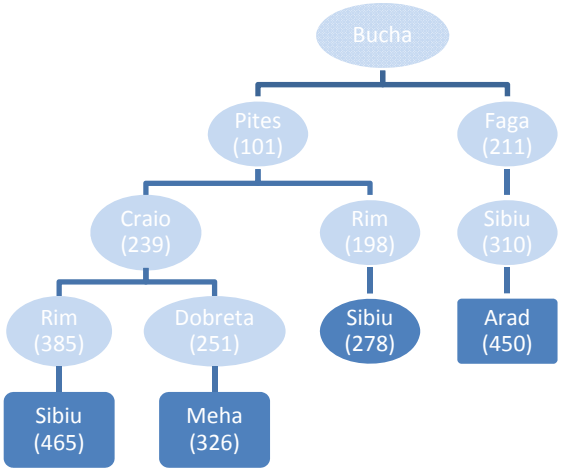
Fringe :
Sibiu (310) Rim(385) Dobreta(251) Sibiu(278)



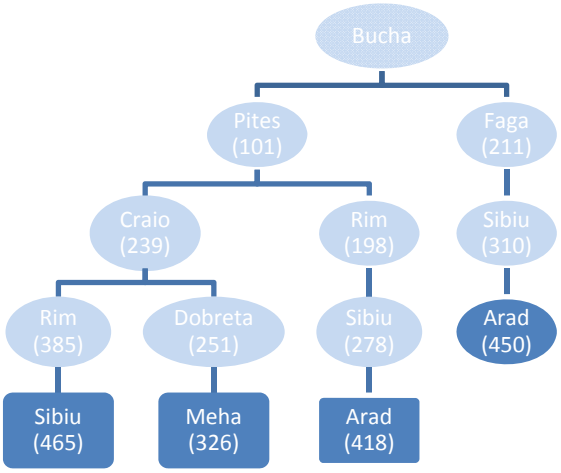
Fringe :
Rim(385) Dobreta(251) Sibiu(278) Arad(450)



Fringe :
Dobreta(251) Sibiu(278) Arad(450) Sibiu(465)

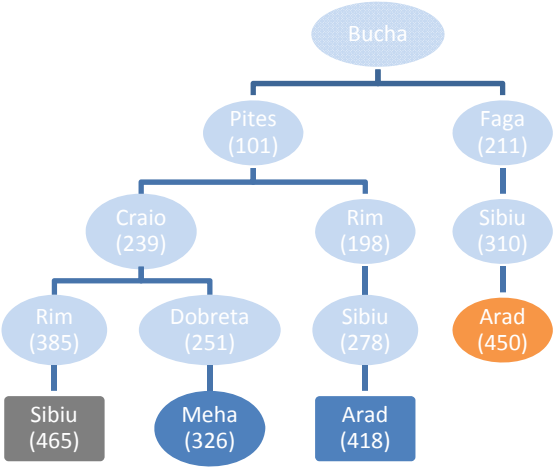


Fringe :
Sibiu(278) Arad(450) Sibiu(465) Meha(326)

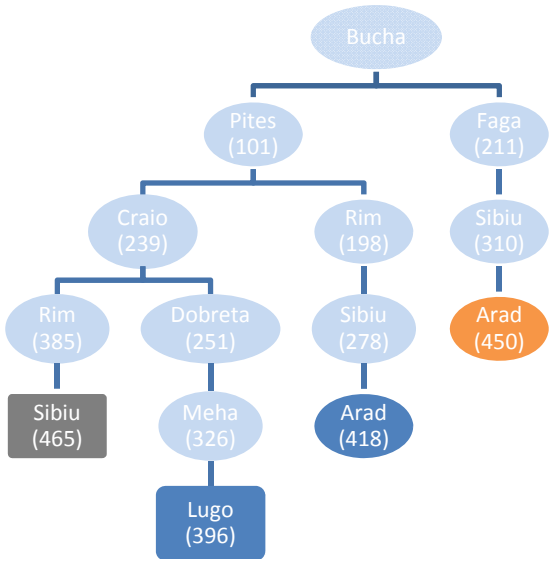


Fringe :
Arad(450) Sibiu(465) Meha(326) Arad(418)
Diketemukan Goal awal yaitu Arad dengan jarak sebesar 450. Goal ini disimpan dan batas atas menjadi 450.

Langkah 3
Fringe dicek untuk melihat node mana yang tidak bisa lagi dilanjutkan pencariannya karena melampaui atau sama dengan batas atas dan kemudian dibuang
Fringe awal:
Arad(450) Sibiu(465) Meha(326) Arad(418)
Fringe baru menjadi:
Meha(326) Arad(418)
Batas Atas : 450

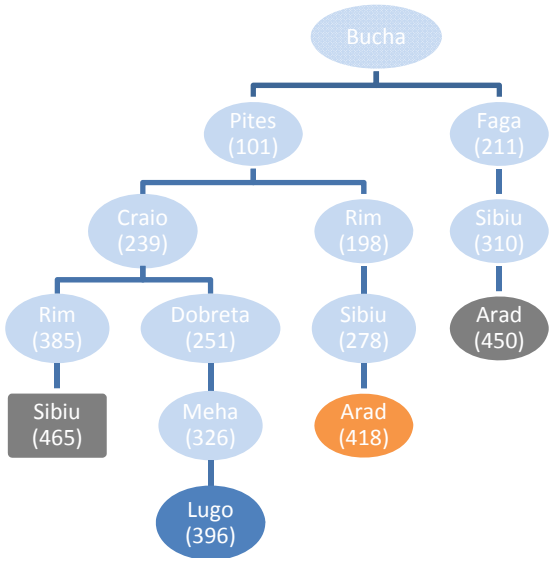


Langkah 4.
Pencarian dilanjutkan berdasarkan fringe yang baru yaitu pencarian pada node Meha.

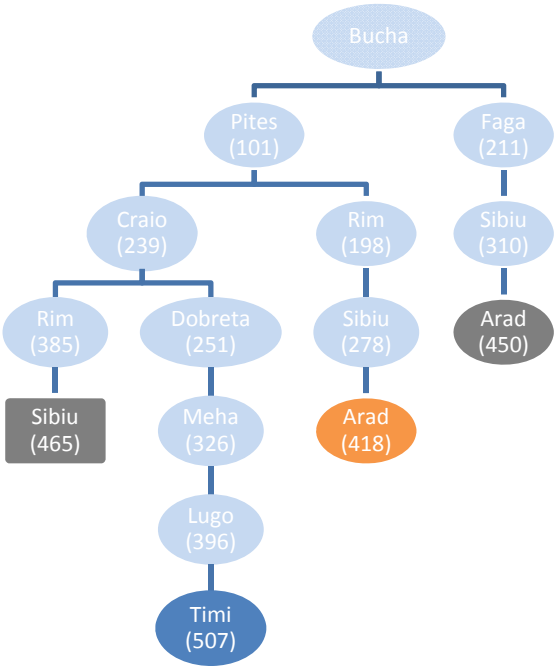


Fringe:
Arad(418) Lugo(396)
Goal Arad ditemukan lagi dengan biaya jarak 418 sehingga kita maju ke langkah 4.1

Langkah 4.1
Goal ini disimpan menggantikan goal lama kemudian batas atas diubah menjadi 418. Setelah itu, diulangi lagi langkah 3 sehingga :
Fringe awal:
Arad(418) Lugo(396)
Fringe baru menjadi:
Lugo(396)
Batas atas : 418

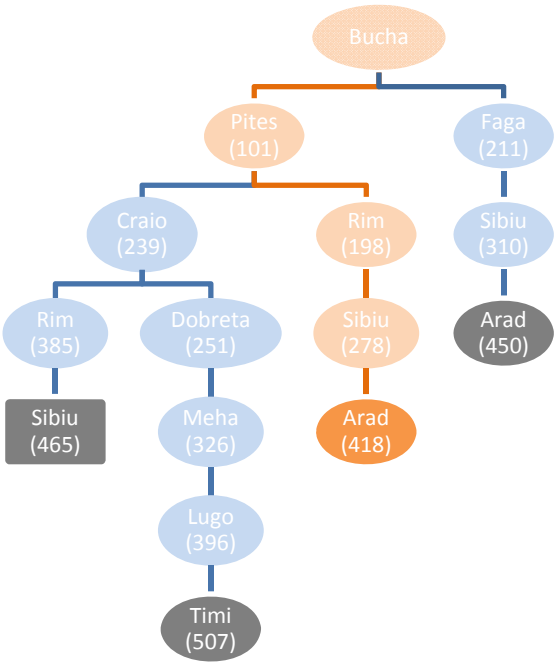


Setelah langkah 3 selesai dilakukan, kita maju kembali lagi ke langkah 4. Pencarian dilanjutkan berdasarkan fringe yang baru yang menunjuk pada antrian pertama yaitu Lugo (396).



Fringe :
Timi (507)
Ditemukan node dalam fringe yang memiliki biaya di atas batas atas yang ditentukan yaitu 418, maka kita maju ke langkah 4.2.

Langkah 4.2
Node Timi(507) dihilangkan dari fringe, sehingga fringe menjadi kosong. Akibatnya, algoritma pencarian ini dihentikan dan didapatkan jalur terpendek yang diinginkan yaitu Bucha-Pites-Rim-Sibiu-Arad, dengan jarak yang ditempuh adalah 418.



Dengan demikian, penetuan jalur kota terpendek dapat diselesaikan dengan algoritma ini.

3.3. Evaluasi Algoritma Branch and Bound dalam Pencarian Jalur Terpendek

1. Kelengkapan :
   Ya, karena semua kemungkinan dalam satu tingkat kedalaman akan diekspan semuanya sama seperti BSF.
2. Optimasi :
   Algoritma ini akan menemukan jalur dengan biaya/bobot terkecil
3. Kompleksitas waktu :
   Pada kasus terburuk, memiliki kompleksitas yang lebih besar dari BFS walaupun memiliki nilai O besar yang sama, yaitu.
   $1+b+b^2+b^3+\ldots+b^d=O(b^d)$
   Tetapi, tentu saja variable d(kedalaman) pada Branch and Bound lebih besar dari BSF.
4. Kompleksitas ruang :
   Pada kasus terburuk memiliki kompleksitas ruang $O(b^d)$ karena menyimpan semua node di memori.

4. KESIMPULAN

Permasalahan penentuan Jalur kota terpendek membentuk suatu ruang solusi yang dapat direpresentasikan dengan menggunakan pohon pencarian. Pohon pencarian ini dapat dibentuk dengan algoritma Branch and Bound yang berbasis pada BFS. Algoritma ini digunakan untuk mendapatkan solusi yang lebih mangkus yang dalam artian optimal dalam biayanya. Hal ini dikarenakan basis penerapan Branch and Bound itu sendiri adalah untuk persoalan optimasi yang berdasarkan biaya antar state yang berbeda sehingga cocok diterapkan untuk penentuan jalur kota terpendek .

Walaupun memiliki kompleksitas nilai O besar yang bersifat eksponensial, Algoritma Branch and Bound dapat digolongkan sebagai algoritma yang memiliki kompleksitas yang kecil jika dibandingkan dengan algoritma yang juga berbasis optimasi lainnya.

DAFTAR PUSTAKA

[1]. Munir, Rinaldi. (2006). "Matematika Diskrit", InformatikaBandung, 2005, hal.IX-1 –X-32.

[2]. *http://www.informatika.org/~rinaldi/Stmik/2006-2007/ Penerapan%20BFS%20dan%20 DFS% 20pada%20Pencarian%20Solusi.pdf* Tanggal akses: 29 Desember 2008 pukul 14:00

[3]. *http://elearning.uin-suka.ac.id/attachment/8_penerapan_bfs_dan_dfs_p.ppt* Tanggal akses:29 Desember 2008 pukul 15:00

[4]. *http://www.stttelkom.ac.id/staf/atw/pages/download.php* Tanggal akses:29 Desember 2008 pukul 15:00

[5]. *http://www.cs.ui.ac.id/WebKuliah/IKI30320/materi/iki30320_20070905_handout_ rev.pdf* Tanggal akses: 29 Desember 2008 pukul 16:00

[6]. *http://en.wikipedia.org/wiki/Branch_and_bound* Tanggal akses:30 Desember 2008 pukul 11:00

[7]. *http://mufidnilmada.staff.gunadarma.ac.id/Downloads/files/9705/Algoritma+**Branch**+and+**Bound**+(bagian+1).ppt* Tanggal akses: 1 Januari 2009 pukul 14:00