Penerapan Algoritma Pencarian Graf pada Pouring Water Problem

Irfan Haris Widyadhana - 13517041
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517041@std.stei.itb.ac.id

***Abstract*—Algotitma Pencarian Graf banyak dimanfaatkan untuk menyelesaikan sejumlah permasalahan dengan merepresentasikan permasalahan tersebut menjadi sebuah pohon ruang status. Salah satunya dapat digunakan untuk menyelesaikan game puzzle yang cukup terkenal yakni pouring water problem, dimana kita disediakan sejumlah gelas yang berbeda ukuran dan disediakan sejumlah air, tujuan kita adalah mendapatkan air dalam jumlah tertentu hanya dengan memindahkan air antar gelas. Pada makalah ini akan dicoba penyelesaian permasalahan ini menggunakan dua algoritma penelusuran graf yakni Breadth First Search dan Depth First Search.**

***Keywords*—BFS, DFS, Graf, State**

I. PENDAHULUAN

Water Pouring Problem juga dikenal dengan water jug problem sering merupakan suatu permasalahan yang sering dijadikan sebagai soal dalam game maupun kuis atau teka teki yang mengasah logika berpikir dan sudah sangat umum bagi orang-orang yang gemar akan tantangan terkait logika. Hingga sekarang masalah ini masih sering dijadikan bahan pembelajaran dalam bidang informatika terkait pada bidang artificial inteligence.

Pada permasalahan ini terdapat sejumlah gelas dengan volume yang berbeda-beda dalam bilangan bulat, untuk gelas yang berukuran paling besar kita isi air hingga penuh dan untuk gelas lainnya tetap kita biarkan kosong, tujuan dari permasalahan ini adalah kita harus dapat menuangkan air berpindah-pindah antar gelas sehingga salah satu gelas dapat terisi jumlah volume air yang kita inginkan.

Banyak permasalahan dalam sehari-hari yang mirip dengan permasalahan tersebut, dan untuk menyelesaikannya kita cukup menggunakan logika ataupun bisa kita coba saja seluruh kemungkinan hingga ketemu solusinya, tetapi jika permasalahan tersebut besar dan banyak tentunya dengan perhitungan sendiri akan memakan waktu yang lama dan karena perhitungan dilakukan oleh otak manusia yang jika dipakai terus-menerus akan menjadi lelah dan menyebabkan perhitungan kita menjadi rawan terjadi kesalahan, berbeda dengan komputer yang terus-menerus bekerja dengan performa yang sama.

Permasalahan ini cocok untuk diselesaikan menggunakan penelusuran graf dengan terlebih dahulu kita representasikan menggunakan graf. Kondisi yang menggambarkan kondisi volume air dari gelas-gelas saat ini dapat kita jadikan sebagai simpul atau node dan pemilihan gelas untuk penuangan air merupakan sisi yang menghubungkan kondisi gelas sebelum dan sesudah pemindahan air.

Dalam proses penyelesaian masalah ini dapat kita coba menggunakan algoritma DFS (Depth First-Search) dan BFS (Breadth First-Search). DFS dan BFS merupakan algoritma penulusuran graf yang sangat populer di dunia informatika, algoritma ini banyak dipakai untuk memecahkan berbagai permasalahan yang dapat diimplementasikan kedalam sebuah graf dengan membagi masalah kedalam beberapa state yang memungkinkan hingga ditemukannya state finish atau state solusi.

II. DASAR TEORI

A. Graf dan Algoritma Penelusuran Graf

Graf adalah sekumpulan simpul yang saling dihubungkan dengan sisi, simpul yang dihubungkan dengan sisi dikatakan bertetangga dengan simpul yang dihubungkan sisi tersebut. Banyak permasalahan di bidang computer science yang menerapkan penggunaan graf diantaranya permasalahan labirin yang bisa kita modelkan sebagai graf dimana setiap persimpangan dinyatakan sebagai sebuah simpul dan jalur di antara kedua persimpangan dinyatakan sebagai sisi graf.

Walaupun sejumlah permasalahan cukup sulit untuk membuat pemodelan graf-nya, tetapi setelah graf berhasil kita buat selanjutnya metode untuk kita menelusuri graf tersebut menjadi lebih mudah dengan menggunakan algoritma traversal graf yang sudah ada.

Algoritma traversal graf adalah teknik berfikir untuk mencari satu atau lebih simpul yang merupakan anggota dari graf dengan melakukan penelusuran yang sistematis. Perbedaan dalam algoritma traversal graf merupakan perbedaan dalam urutan penelusuran, perbedaan perhitungan bobot menuju tujuan dan keputusan pengambilan jalur. Secara umum terdapat dua jenis algoritma traversal graf yakni pencarian dengan informasi tambahan dan tidak (informed search & uninformed search).

Ada banyak pendekatan untuk menelusuri graf, sehingga sampai pada simpul yang diinginkan (node goal), diantaranya :

Breadth First Search, Depth First Search,Iterative deepening search, Uniform Cost Search,A*, Greedy Best First Search, dan banyak lagi variasi pencariannya. Tiap metode pencarian graf memiliki karakternya masing-masing, tapi untuk permasalahan ini hanya akan dibahas mengenai Breadth First Search dan Depth First Search.

B. Breadth First Search

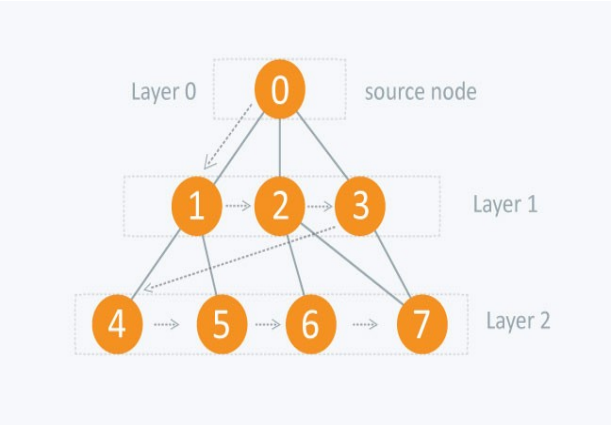
Breadth first search (BFS) ditemukan oleh E.F Moore, seorang profesor di bidang matematika dan computer science. Pada mulanya algoritma BFS digunakan pada tahun 1950 untuk menyelesaikan persoalan labirin. Pada 1961, C. Y. Lee juga menemukan algoritma BFS secara independen untuk menemukan solusi jalur kabel listrik.

Sesuai namanya, breadth first search memiliki prioritas untuk melakukan penelusuran graf secara meluas terlebih dahulu, maksudnya penelusuran dilakukan dengan memeriksa semua simpul tetangga dari suatu simpul terlebih dahulu. Misalnya saat eksekusi dimulai dari akar dari graf, maka selanjutnya seluruh simpul yang bertetangga dengan simpul tersebut akan kita periksa.

Saat kita melakukan penelusuran maka semua tetangga dari suatu simpul akan kita masukkan ke dalam queue dimulai dari simpul yang memiliki penomoran indeks paling kecil, setelah semua simpul tetangga dimasukkan ke dalam queue maka simpul yang kita periksa sekarang akan berpindah ke elemen pertama dari queue atau dengan kata lain merupakan simpul tetangga pertama.

Berikut adalah langkah-langkah BFS :

1. Kunjungi simpul v.
2. Kunjungi semua simpul yang bertetangga dengan simpul v terlebih dahulu, dengan urutan pemilihan simpul yang konsisten.
3. Kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang telah dikunjungi dimulai dari urutan pertama, demikian seterusnya.



Gambar 1. Algoritma penelusuran breadth first search

sumber :
https://www.hackerearth.com/practice/algorithms/graphs/breadth-first-search/tutorial/

Pencarian terus dilakukan dengan cara meng-expand simpul yang diperiksa sekarang dan memasukkan semua simpul tetangga yang belum ke dalam queue kita secara urut sesuai index. Langkah tersebut dilakukan berulang-ulang dengan memeriksa simpul yang belum dikunjungi dan memeriksa simpul yang bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya.

Pada implementasinya BFS memerlukan struktur data queue atau antrian. Antrian tersebut berisi simpul-simpul mana saja yang harus dikunjungi selanjutnya. Pada awalnya antrian tersebut berisi simpul akar atau awal. Setelah mengunjungi suatu simpul, semua tetangga dari simpul tersebut yang belum pernah dikunjungi akan dimasukkan ke antrian (enqueue). Kemudian simpul tersebut akan dikeluarkan dari antrian (dequeue). Jika antrian kosong, maka artinya graf sudah ditelusuri sepenuhnya.Kompleksitas waktu yang diperlukan oleh BFS adalah O(b^d). Di mana b adalah jumlah simpul anak dari tiap simpul pada pohon penelusuran dan d adalah kedalaman dari graf. Memori yang diperlukan BFS juga Θ(|b^d) , sehingga dapat disimpulkan algoritma penelusuran BFS ini kurang baik dalam hal penggunaan memori.

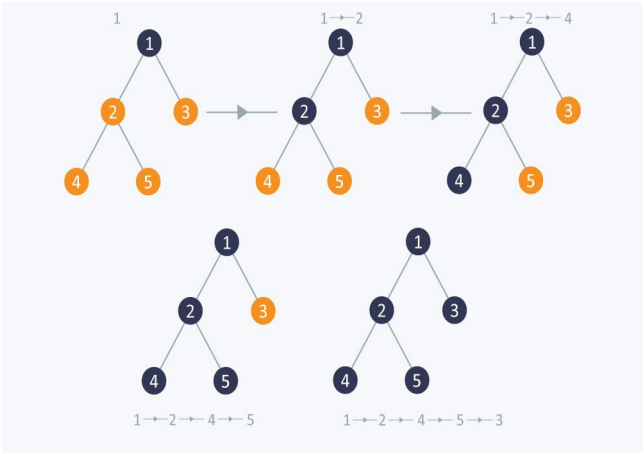
C. Depth First Search

Algoritma Depth-First Search (DFS) adalah algoritma traversal untuk melakukan pencarian di dalam suatu graf dengan cara melakukan pencarian dengan memilih salah satu simpul yang ada dan menelusuri anak dari simpul tersebut hingga salah satu solusi ditemukan atau telah mencapai simpul daun sehingga pencarian akan dilanjutkan ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai satu atau lebih simpul yang belum dikunjungi.

Tidak diketahui secara pasti siapa yang menemukan algoritma penelusuran DFS, tetapi algoritma yang serupa dengan DFS pertama kali diteliti oleh seorang matematikawan prancis Charles Pierre Tremaux, dia melakukannya dalam sebuah model Tremaux tree.

Berikut adalah langkah-langkah DFS :

1. Kunjungi simpul v.
2. Kunjungi simpul w yang bertetangga dengan simpul v.
3. Ulangi DFS mulai dari simpul w.
4. Ketika mencapai simpul u sedemikian sehingga semua simpul yang bertetangga dengannya telah dikunjungi, pencarian dirunut-balik (backtrack) ke simpul terakhir yang dikunjungi sebelunmnya dan mempunyai simpul w yang belum dikunjungi.
5. Pencarian berakhir bila tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi.



Gambar 2. Algoritma penelusuran depth first search

sumber : https://www.hackerearth.com/practice/algorithms/graphs/depth-first-search/tutorial/

Pada implementasinya DFS memerlukan struktur data stack atau tumpukan yang digunakan untuk menyimpan simpul-simpul yang akan dikunjungi pada jalur yang sedang ditelusuri. Pada awalnya tumpukan tersebut kosong. Pada saat simpul baru dikunjungi maka simpul tersebut akan ditambakahkan (push) ke tumpukan. Sedangkan pada saat dilakukan runut balik, simpul yang dirunut balik akan dikeluarkan dari tumpukan (pop).

Tumpukan pada DFS dapat bersifat explisit maupun implisit. Secara eksplisit artinya terdapat implementasi struktur data tumpukan pada program. Sedangkan implisit artinya DFS dilakukan secara rekursif yang belaku seperti tumpukan. DFS yang menggunakan tumpukan implisit sering juga disebut DFS rekursif. Kompleksita waktu yang diperlukan oleh DFS untuk menelusuri graf secara menyeluruh adalah $\Theta(b^m)$. Di mana b adalah jumlah simpul anak dari tiap simpul pada pohon penelusuran dan m adalah kedalaman maksimum dari graf. Sementara memori yang diperlukan DFS adalah $\Theta(bm)$, sehingga dapat disimpulkan algoritma penelusuran DFS ini lebih baik dalam hal penggunaan memori tetapi kurang dalam hal kompleksitas waktu.

IV. PEMBAHASAN

Gambar 3. Ilustrasi Water Jug Problem

sumber : https://www.transum.org/Software/Investigations/jugs.asp

Permasalahan pouring water problem ini merupakan salah satu permasalahan yang cukup mengasah otak, berdasarkan dari penjelasan sebelumnya, permasalahan ini akan dijelaskan kembali sebagai berikut.

Kita diberikan sejumlah gelas dengan ukuran yang berbeda-beda, dari gelas-gelas tersebut kita hanya mengetahui volume maksimum air yang dapat ditampung gelas tersebut dan di gelas tersebut tidak terdapat penanda takaran air. Untuk gelas yang memiliki volume terbesar akan diisi air hingga penuh.

Tujuan yang ingin kita capai adalah kita harus bisa mendapatkan air sebanyak x liter di salah satu gelas, dimana x merupakan voume jumlah air yang kita inginkan.Untuk mencapai tujuan tersebut kita harus memindahkan air dari salah satu gelas ke gelas lainnya.

Aturannya yakni jika kita memindahkan air menuju gelas yang tidak dapat menampung seluruh air dari gelas asal, maka gelas tujuan hanya akan diisi sebagian air hingga gelas tersebut penuh dan sisanya tetap akan berada di gelas awal. Sementara jika gelas yang dituju dapat menampung air yang lebih besar dari gelas awal maka seluruh air akan dipindahkan dari gelas awal ke gelas tujuan mengakibatkan gelas awal menjadi kosong, kita tidak diperbolehkan untuk mencoba memindahkan air ke gelas yang sudah penuh. Dengan aturan tersebut dapat dipastikan tidak akan ada air yang hilang selama proses pemindahan, sehingga jumlah total air yang ada di seluruh gelas pasti sama dengan volume gelas terbesar.

Untuk permasalahan ini tujuan kita hanyalah mencari langkah-langkah yang membawa kita ke solusi yakni mendapatkan jumlah air yang kita inginkan di salah satu gelas, tetapi kita akan mengusahakan agar langkah solusi kita merupakan langkah tercepat yakni langkah-langkah yang membutuhkan jumlah pemindahan air paling sedikit.

Permasalahan penuangan air ini dapat kita representasikan menjadi pohon ruang status, dimana simpulnya merupakan state atau kondisi seluruh gelas saat ini dan sisi-nya melambangkan proses penuangan dari salah satu gelas ke gelas lainnya.

Sehingga untuk permasalahan ini dapat diterapkan pencarian pohon ruang status menggunakan Depth First Search, algoritma DFS ini bisa lebih berguna jika dilakukan bersamaan dengan membangun pohon DFS. Simpul awal akan berlaku sebagai akar dari pohon ini. Setiap mengunjungi simpul untuk pertama kali, simpul tersebut diberlakukan sebagai simpul anak yang dibangkitkan oleh simpul sebelumnya.

Penyelesaian permasalahan ini menerapkan konsep Artificial Intelligence yaitu dengan bantuan pohon ruang-status dan menerapkan metode pencarian melebar pertama (breadth-first search / BFS). Pencarian solusi dimulai dari kondisi dimana gelas kosong dan gelas terbesar terisi penuh (akar dari pohon pencarian). Proses dilanjutkan dengan menggambarkan kondisi (state) berikutnya (dengan melakukan aksi terhadap state sebelumnya) hingga semua state diperiksa dan mendapatkan tujuan (goal state).

Karena kita menginginkan untuk mendapatkan solusi dengan jumlah pemindahan paling sedikit, dari masing-masing sifat algoritma penelusuran pada graf penelusuran pohon menggunakan Depth First Search (DFS) tidak memberikan hasil yang optimal, hal ini dikarenakan kedalaman pohon yang

belum diketahui dan jika kita temukan solusi menggunakan DFS solusinya bisa saja terletak sangat dalam sementara terdapat solusi lain yang membutuhkan langkah yang lebih sedikit.

Sementara jika permasalahan ini kita coba selesaikan dengan menggunakan Breadth First Search, BFS mengunjungi semua simpul yang dibangkitkan dari simpul akarnya terlebih dahulu, dan jika terdapat lebih dari satu solusi maka BFS selalu menemukan solusi pertama pada kedalaman pohon yang paling rendah atau paling dekat dengan akar.

Untuk membuktikan hasil penelusuran yang lebih baik, akan kita coba dengan melakukan pengujian menggunakan sebuah program BFS dan DFS yang di-test dengan menggunakan bahasa python. Pada program ini kita menggunakan class State untuk merepresentasikan simpul pada pohon, class State didefinisikan sebagai suatu class yang menyimpan atribut dari kondisi volume tiap gelas saat ini.

Algoritma DFS dan BFS pada program yang digunakan sekilas terlihat mirip, namun tentunya masih terdapat perbedaan di antara keduanya. Algoritma DFS yang digunakan memanfaatkan stack untuk menyimpan state, Sementara pada algoritma BFS kita memanfaatkan queue. Selain itu perlu diperhatikan algoritma DFS, fungsi yang digunakan untuk meng-expand simpul/state anak menghasilkan urutan child yang berlawanan dengan prioritas yang kita terapkan, sementara pada algoritma BFS fungsi menghasilkan urutan child sesuai prioritas yang kita terapkan.

Program DFS dan BFS tersebut bekerja dengan cara meng-expand simpul/state yang ditelusuri dan memasukkannya ke dalam stack atau queue, proses ini dilakukan berulang-ulang hingga stack atau queue kosong, jika ditemukan simpul/state yang merupakan solusi maka stack atau queue akan dikosongkan dan program berhenti.

Berikut pseudo code algoritma DFS dan BFS yang digunakan untuk pengujian :

```
procedure DFS(state : start_state)
  visited : array of state
  stack : array of state

  insert start_state into stack

  { loop selama stack belum kosong }
  while stack:
     state ← top of stack

     {cek apakah state sudah pernah ditelusuri }
     if ( not hasVisited(visited,state)):

        { child yang di-expand dalam urutan terbalik }
         nextState traversal getChild(state):
 { masukkan state ke dalam stack }
              insert nextState into stack
         {end of traversal }

     insert state into visited
     if (Finish(state)):{ Simpul tujuan }
        Stack ← []  { set stack empty }

  { end of while }
```

```
procedure BFS(state : start_state):
  visited : array of state
  queue : array of state

  insert start_state into queue

 { loop selama stack belum kosong }
 while queue:
     state ←  head of queue

     {cek apakah state sudah pernah ditelusuri }
     if ( not hasVisited(visited,state)):

     { child yang di-expand sesuai urutan prioritas }
     nextState traversal getChild(state):
 { masukkan state ke dalam queue}
              insert nextState into queue
          {end of traversal }

     insert state into visited
     if (Finish(state)):{ Simpul tujuan }
       queue ← []{ set stack empty }

 { end of while }
```

Kita akan mengambil contoh kasus yang sudah sering dipakai di kebanyakan soal teka-teki, misalkan dalam contoh permasalahan ini kita memiliki 3 buah gelas masing-masing dengan volume 8 L , 5L dan 3L dan kita menginginkan air sebanyak 4 L, sehingga jumlah air sebanyak 4 L yang kita inginkan dapat berada di gelas 8 L atau gelas 5 L tetapi tidak di gelas 3 L karena volumenya lebih kecil dan tidak muat untuk menampungnya. Kondisi state awal dari permasalahan ini adalah kondisi saat gelas dengan volume terbesar yakni gelas 8 L diisi penuh air dan gelas lainnya kosong.

Untuk pohon ruang status yang kita bangkitkan, kita membuat notasi simpul berupa state kondisi volume air dalam gelas saat ini, kita menuliskannya dalam 3 buah angka dimana setiap angka merepresentasikan volume air dalam gelas dimulai dari gelas dengan kapasitas terbesar terurut menurun. Simpul yang serupa dengan simpul yang pernah dibangkitkan sebelumnya tidak akan dibangkitkan lagi.

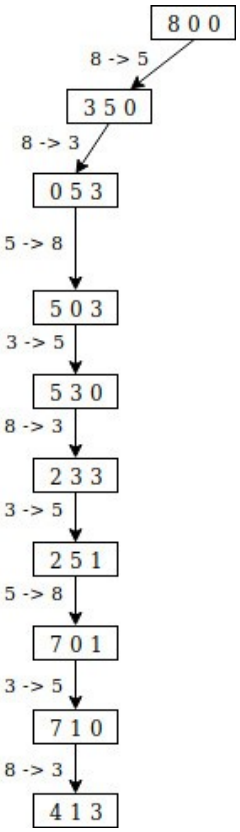
Gambar 4. Representasi state
sumber : Dokumen pribadi

State tersebut diartikan sebagai kondisi saat gelas 8 L berisi 8 L air, gelas 5 L kosong dan gelas 3 L kosong.
Urutan pemilihan penuangan air dalam contoh ini yakni:

8 → 5
8 → 3
5 → 8
5 → 3
3 → 8
3 → 5

Dengan menggunakan algoritma DFS, penelusuran dimulai dari simpul awal yakni state dimana gelas 8 L terisi penuh dan gelas lainnya kosong, dengan mengikuti urutan expand simpul kita akan menuangkan air dari gelas 8 L ke gelas 5 L hingga penuh, dan state dari gelas-gelas sekarang adalah gelas 8 L berisi 3 L, gelas 5L terisi penuh dan gelas 3 L kosong. Selanjutnya dengan memperhatikan urutan expand simpul kita tidak bisa lagi menuangkan air dari gelas 8 L ke gelas 5 L karena gelas 5 L sudah terisi penuh, sehingga sesuai urutan kita akan menuangkan air dari gelas 8 L ke gelas 3 L dan kita sekarang mencapai state gelas 8 L kosong, gelas 5 L penuh dan gelas 3 L penuh. Langkah tersebut terus diulangi hingga kita mencapai simpul goal yakni dalam kasus ini salah satu gelas berisi 4 L air.

Berikut hasil penelusuran yang kita peroleh dengan menggunakan algoritma Depth First Search :



Gambar 5. Hasil penelusuran dengan DFS
sumber : Dokumen pribadi

Terlihat bahwa langkah penelusuran dengan DFS seperti kita menyelesaikan permasalahan ini secara biasa yakni kita terus-menerus memindahkan air dari gelas ke gelas lain hingga tercapai kondisi gelas yang kita inginkan.

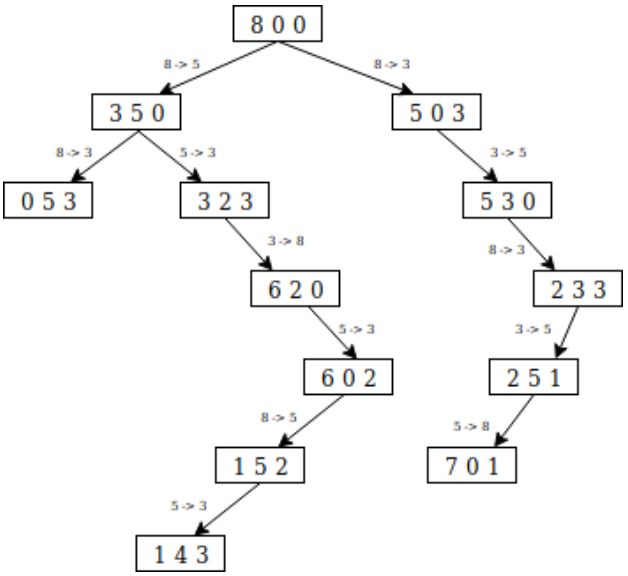
Dengan menggunakan algoritma penelusuran BFS kita memulai penelusuran dari state awal dimana gelas 8 L terisi penuh dan gelas lainnya kosong, dengan mengikuti urutan expand simpul kita akan mengambil langkah menuangkan air dari gelas 8 L ke gelas 5 L hingga penuh, sehingga state dari gelas-gelas sekarang adalah gelas 8 L berisi 3 L, gelas 5L terisi penuh dan gelas 3 L kosong (3,5,0). Selanjutnya kita coba kemungkinan lain dari state awal sebelumnya, selain menuangkan air dari gelas 8 L ke gelas 5 L kita juga bisa mencoba menuangkan air dari gelas 8 L ke gelas 3 L sehingga tercapai state gelas 8 L berisi 5 L, gelas 5 L kosong dan gelas 3 L terisi penuh (5,0,3). Selanjutnya tidak ada lagi kemungkinan penuangan lagi yang mungkin dari state awal karena pada kondisi awal hanya gelas 8 L yang berisi air sehingga penuangan hanya dapat berasal dari gelas yang berukuran 8 L.

Karena tidak ada lagi penuangan air yang mungkin dari state awal, selanjutnya kita akan meng-expand simpul anak pertama dari simpul/state awal yakni state saat gelas 8 L berisi 3 L, gelas 5L terisi penuh dan gelas 3 L kosong (3,5,0). Dari state ini kita bisa mencoba sesuai urutan prioritas memindahkan air dari dari gelas 8 L ke gelas 3 L hingga tercapai state gelas 8 L kosong dan gelas 3 L dan 5 L terisi penuh (0,5,3), proses selanjutnya adalah meng-expand state sebelumnya yaitu state gelas gelas 8 L berisi 3 L, gelas 5L terisi penuh dan gelas 3 L kosong (3,5,0).

Terdapat kemungkinan kita mencoba menuangkan air dari gelas 5 L ke gelas 8 L, tetapi itu hanya membuat kita kembali ke state sebelumnya lagi yakni saat kondisi gelas masih seperti semula, sehingga langkah ini tidak perlu kita ambil, langkah selanjutnya yang mungkin kita bisa mencoba untuk memindahkan air dari gelas 5 L ke gelas 3 L sehingga kondisi gelas menjadi gelas 8 L terisi 3 L, gelas 5 L terisi 2 L dan gelas 3 L terisi penuh (3,2,3), karena seluruh simpul/state anak dari state (3,5,0) telah di-expand semua maka selanjutnya kita akan meng-expand simpul anak selanjutnya dari simpul awal (8,0,0) yakni kita akan meng-expand simpul (5,0,3).

Langkah tersebut kita ulangi dengan meng-expand suatu simpul terlebih dahulu hingga habis, barulah kita meng-expand anak pertama simpul tersebut hingga habis dan dilanjutkan meng-expand simpul anak kedua dan seterusnya hingga simpul anak ter-expand semua.

Berikut hasil penelusuran yang kita peroleh dengan menggunakan algoritma Breadth First Search :



Gambar 6. Hasil penelusuran dengan BFS
sumber : Dokumen pribadi

Langkah penelusuran dengan BFS ini terlihat lebih rumit daripada penelusuran dengan DFS sebelumnya, hal ini dikarenakan DFS hanya akan meng-expand simpul anak pertamanya terus menerus sehingga langkah penelusuran itu terlihat seperti kita terus menerus memindahkan air antar-gelas hingga mencapai solusi, sementara langkah penelusuran BFS ini tidak seperti itu karena setelah kita melakukan penuangan air dari satu gelas ke gelas lain kita harus memikirkan juga langkah penuangan lain yang mungkin dan itu harus kita coba terlebih dahulu, sehingga jika langkah penelusuran BFS ini coba kita terapkan langsung akan sangat sulit dilakukan .

Dari hasil pencarian contoh yang kita gunakan menggunakan kedua algoritma DFS dan BFS, dapat terlihat karakteristik dari masing-masing pencarian. Dengan menggunakan algoritma DFS dilakukan penelusuran sebanyak 9 kali pengecekan simpul, sementara dengan menggunakan algoritma BFS dilakukan pengecekan simpul sebanyak 12 kali.

Hasil tersebut menunjukkan untuk contoh kali ini yang menggunakan 3 gelas berukuran 8 L, 5 L dan 3 L algoritma DFS memberikan penelusuran yang lebih cepat, tetapi jika kita lihat dari banyaknya langkah pemindahan air hingga tercapai solusi kita mendapatkan algoritma BFS memberikan solusi jumlah langkah pemindahan yang lebih sedikit yakni pada contoh ini sebanyak 6 kali pemindahan air, sementara DFS membutuhkan 9 kali pemindahan air.

Dari hasil tersebut, terlihat ciri masing-masing dari kedua algoritma ini, pada pencarian menggunakan DFS jika terdapat lebih dari satu solusi tetapi berada pada level yang berbeda, maka pada Depth First Search tidak ada jaminan untuk menemukan solusi dengan langkah pemindahan air paling sedikit, sementara algoritma BFS selalu menjamin solusi tersebut tercapai karena pencarian dengan BFS berhenti pada solusi dengan kedalaman terendah dari akar node, sehingga untuk memperoleh solusi dengan jumlah pemindahan paling sedikit lebih baik menggunakan algoritma penelusuran BFS.

V. KESIMPULAN

Dari percobaan program, dapat terlihat kedua algorima yakni breadth first search dan depth first search sama-sama dapat memberikan solusi untuk permasalahan ini, dan dari hasil yang diperoleh dapat terlihat bahwa algoritma pencarian BFS dapat memberikan solusi dengan jumlah pemindahan atau penuangan air paling sedikit dibandingkan menggunakan algoritma pencarian dengan DFS dikarenakan sifat dari algoritma BFS itu sendiri.

Tetapi kita juga harus perhatikan metode penelusuran dengan DFS lebih mudah untuk dimengerti karena penelusuran dengan DFS sama seperti kita mencoba dengan cara naive yakni memindahkan air antar gelas terus-menerus hingga kita temukan kondisi yang kita inginkan, sementara metode pencarian BFS mengharuskan kita mengecek semua kemungkinan pemindahan air dari kondisi saat ini baru kita lanjut ke pengecekan kondisi berikutnya dan itu akan sulit untuk langsung kita terapkan penelusurannya secara langsung.

REFERENCES

[1] Munir, Rinaldi, “Diktat Kuliah IF2211 Strategi Algoritma”,Program Studi Teknik Informatika STEI ITB, 2017

[2] E. F. Moore (1959), The shortest path through a maze. In *Proceedings of the International Symposium on the Theory of Switching*, Harvard University Press, pp. 285–292.

[3] https://www.hackerearth.com/practice/algorithms/graphs/depth-first-search/tutorial/ diakses Jumat, 19 April 2019 pukul 21.03

[4] https://www.hackerearth.com/practice/algorithms/graphs/breadth-first-search/tutorial/ diakses Jumat, 19 April 2019 pukul 21.05

[5] https://www.transum.org/Software/Investigations/jugs.asp diakses Senin, 22 April 2019 pukul 22.12

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, 26 April 2019

Irfan Haris Widyadhana
13517041