Penerapan Algoritma BFS, DFS, dan A* untuk P2P File Sharing dengan Torrenting

Fabian Zhafransyah / 13518022
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): fabianzhafransyahharahap@gmail.com

***Abstrak*—Di era digital ini, tentunya kita sudah tidak asing lagi dengan istilah internet. Cara berkomunikasi di internet ada macam-macam, salah satunya adalah dengan men-transfer data. Salah satu teknologi transfer data adalah *peer-to-peer communication* lewat torrent. Pada makalah ini, akan dieksplorasi salah satu algoritma klasik untuk pencarian, yaitu BFS dan DFS untuk diimplementasikan pada salah satu metode komunikasi *client* pada internet, yaitu *peer-to-peer connection*.**

Keywords—Graf Traversal, Breadth First Search, Depth First Search, Networking, Peer-to-peer, Torrent

I. Pendahuluan

Internet adalah tempat yang sangat luas. Setiap hari kita menggunakan internet untuk keperluan sehari-hari. Hampir tidak ada dari kita yang tidak menggunakan internet. Mulai dari hiburan, berita, sampai ke pendidikan formal. Dalam penggunaan internet, tentunya kita juga berinteraksi dengan banyak sekali bentuk file, dan tak jarang ketika kita harus mengunduh file untuk mengaksesnya. Contohnya adalah ketika kita ingin mendapatkan akses terhadap suatu file. Ada banyak teknologi networking yang mendukung *file sharing*. Bentuk *file sharing* yang paling umum digunakan adalah dengan bentuk *host*, dimana suatu server terpusat menyediakan layanan bagi siapapun yang punya akses kepadanya dapat mengunduh *file* yang terdapat pada server tersebut. Ada juga yang tidak banyak menaruh *file* pada server, melainkan pada komputer *client* masing-masing. Teknologi ini disebut dengan *peer-to-peer file sharing*.

Peer-to-peer file sharing bahkan menggunakan sekitar 30% dari *traffic* internet di seluruh benua Asia. Salah satu alasannya adalah teknologi *torrent* yang memanfaatkan teknologi *peer-to-peer*. Teknologi ini sendiri mulai ramai pada saat aplikasi Napster di tahun 1999 dirilis. Kepopuleran teknologi ini mengakibatkan banyak perkembangan yang telah dilakukan agar keberjalanannya di dunia luar semakin baik. Maka dari itu, kita sebagai masyarakat yang berpotensi menjadi *software engineer* harus mengetahui apa saja yang terjadi dibalik teknologi ini. Salah satu yang penulis temukan adalah pencarian *peer* tentunya menggunakan algoritma pencarian. Sampai sekarang, algoritma yang digunakan merupakan hasil perkembangan yang lama. Tetapi tidak bisa dilupakan bahwa algoritma yang digunakan sekarang adalah algoritma yang bermulai dari algoritma sederhana yang dikembangkan. Maka dari itu, penting bagi kita untuk memulai dari dasarnya terlebih dahulu.

Makalah ini akan membahas tahap dari pencarian *peer* pada saat mengunduh file menggunakan teknologi torrent yang membutuhkan algoritma pencarian. Pencarian yang digunakan adalah dengan menggunakan algoritma Breadth First Search dan Depth First Search. Selain itu juga digunakan algoritma A* yang harapannya lebih mengoptimasi hasil pencarian *peer*.

II. Dasar Teori

A. Graf

Sebuah graf adalah pasangan himpunan V dan E, dimana V adalah himpunan *vertex* / simpul dari graf tersebut, dan E sebagai himpunan *edge* / sisi dari simpul pada V yang menyatakan keterhubungan dari dua simpul di V.

Berdasarkan himpunan sisinya, sebuah graf dapat dikategorikan menjadi dua tipe, yaitu:

1. Graf berarah, yaitu sebuah graf dengan sisi yang memiliki arah. Artinya, jika sebuah sisi *e* menghubungkan simpul *a* dan *b*, maka dapat dikatakan ada hubungan antara *a* ke *b*, tapi belum tentu ada hubungan dari *b* ke *a*.
2. Graf tak berarah, yaitu kebalikan dari graf berarah. Artinya, untuk setiap sisi pada himpunan E, jika ia menghubungkan simpul *a* dan *b*, maka diimplikasikan bahwa *b* ke *a* juga terhubung.

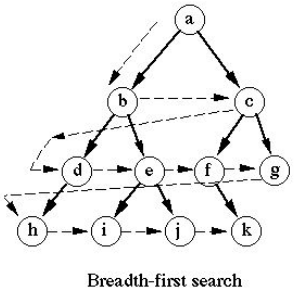
Pada graf tak berarah, jika *a* dan *b* terhubung, artinya mereka saling bertetangga.

B. Traversal Graf

Traversal Graf adalah pengunjungan tiap vertex / node / simpul yang dapat dikunjungi dari graf G(V, E) dimana V adalah himpunan vertex dan E adalah himpunan edge dari graf G. Pada makalah ini, algoritma graf Traversal digunakan untuk melakukan pencarian simpul yang tepat.

C. Breadth First Search

Breadth First Search adalah algoritma yang melakukan pengunjungan *node* pada suatu graf G dengan graf Traversal secara "melebar". Artinya, pencarian dilakukan dengan berdasarkan kedalaman pencarian *node*.



Gambar 1. Ilustrasi Breadth First Search pada sebuah graf. (Sumber: https://vivadifferences.com/difference-between-dfs-and-bfs-in-artificial-intelligence/ diakses pada 1 Mei 2020)

Dengan Breadth First Search, pencarian dilakukan dengan menggunakan sebuah struktur data *queue*, yaitu dengan teknik FIFO (First In First Out). Selain itu, penyimpanan dari tipa sisi yang bertetangga disimpan pada sebuah larik dua dimensi. Pada makalah ini, hanya akan dibahas graf yang memiliki sisi yang tidak berarah (*undirected edge*). Dengan data penyimpan simpul yang sedang dikunjungi berupa *queue*, misal *q*, langkah pencarian dilakukan dengan *step-step* berikut:

1. Kunjungi simpul pertama, misal *v*. Masukkan *v* ke antrian *q*.
2. Pop antrian *q*. Misal hasil *pop* dari *q* adalah *v'*, maka kunjungi semua simpul yang *adjacent* / bertetangga dengan simpul *v'* dan belum pernah dikunjungi sebelumnya dengan memasukkan semuanya ke antrian *q*.
3. Jika antrian *q* habis, artinya pencarian tidak ditemukan. Jika belum habis, lanjutkan ke langkah 4.
4. Jika simpul ditemukan, hentikan pencarian. Jika belum ditemukan, kembali ke langkah 2.

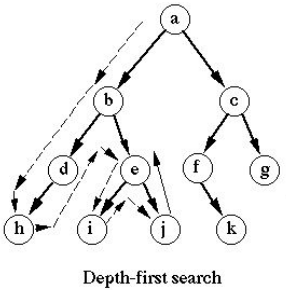
Pada Gambar 1, jika kita mencari simpul *j*, maka dengan Breadth First Search, urutan pencariannya adalah *a, b, c, d, e, f, g, h, i, j*.

Misalnya b adalah banyak simpul dan d adalah kedalaman traversal, Breadth First Search memiliki beberapa properti, yaitu:

- Completeness?
  Ya (selama banyaknya simpul terbatas)
- Optimality?
  Ya, jika langkah proporsional dengan biaya. Dalam kasus pencarian *peer*, tidak.
- Kompleksitas waktu:
  $O(b^d)$
- Kompleksitas ruang:
  $O(b^d)$

D. Depth First Search

Depth First Search juga merupakan algoritma graf Traversal. Bedanya dengan Breadth First Search terletak pada runtutan langkah yang diambil ketika mengambil keputusan untuk mengunjungi sebuah simpul. Seperti namanya, Depth First Search mencari simpul dengan cara mengunjungi simpul sedalam mungkin. Ketika sudah kehabisan arah, ia baru berbalik dan mengunjungi simpul sebelumnya, dan mencari simpul lain yang belum dikunjungi. Kemampuan ini disebut juga *backtracking*.



Gambar 2. Ilustrasi Depth First Search pada sebuah graf. (Sumber: https://vivadifferences.com/difference-between-dfs-and-bfs-in-artificial-intelligence/ diakses pada 1 Mei 2020)

Sama seperti bagian Breadth First Search, algoritma Depth First Search pada makalah ini akan menyelesaikan pencarian pada graf tak berarah. Untuk Depth First Search, struktur data yang digunakan untuk menyimpan simpul adalah *stack*, misal s. Selain itu, penyimpanan dari tipa sisi yang bertetangga disimpan pada sebuah larik dua dimensi. Langkah pencarian dilakukan dengan *step-step* berikut:

1. Kunjungi simpul pertama, misal *v*. Masukkan *v* ke tumpukan / *stack s*.
2. Cek nilai *top* dari *s*. Misal *top* dari *s* adalah *v'*. Pilih satu simpul saja dari semua simpul yang bertetangga dengan *v'* dan belum dikunjungi. Pemilihan simpul tidak berdasarkan apapun, namun sebaiknya heuristik pemilihan simpul pada setiap langkah konsisten. Misal hasil pemilihan simpul yang bertetangga dengan *v'* adalah *p*, *push p* ke *s*.
3. Jika *top* dari s, yaitu simpul *p*, adalah simpul yang dicari, maka hentikan pencarian. Jika tidak, lanjutkan ke langkah 4.
4. Jika *top* dari *s*, yaitu simpul *p,* masih mempunyai tetangga, ulangi lagi langkah 2.
5. Jika *top* dari *s*, yaitu simpul *p,* tidak mempunyai tetangga, maka *pop s* dan ulangi langkah 4. Langkah inilah yang disebut *backtracking*, yaitu kembali sebanyak 1 level secara kedalaman ke simpul sebelumnya.
6. JIka *stack* sudah habis, maka simpul tidak ditemukan. Hentikan pencarian.

Pada Gambar 2, jika kita mencari simpul *j*, maka dengan Depth First Search, urutan pencariannya adalah *a, b, d, h, c, i, j*.

Misalnya b adalah banyak simpul dan m adalah kedalaman terjauh, maka Depth First Search memiliki beberapa properti, yaitu:

- Completeness?
  Ya (selama banyaknya simpul terbatas)
- Optimality?
  Ya, jika langkah proporsional dengan biaya. Dalam kasus pencarian *peer*, tidak.
- Kompleksitas waktu:
  $O(b^m)$
- Kompleksitas ruang:
  O(bm)

E. Algoritma A*

Algoritma A* adalah algoritma yang termasuk *informed search*, yaitu menggunakan heuristik terhadap simpul yang akan dicari. Algoritma A* juga terkadang disebut sebagai gabungan dari *Uniform Cost Search* dan *Greedy Best-First*. Berbeda dengan Breadth First Search dan Depth First Search, algoritma A* memerlukan graf yang berbobot, karena ada fungsi heuristik yang berjalan terhadap bobot tiap sisi graf tersebut.

Algoritma A* memiliki sifat pencarian yang sama seperti Breadth First Search yaitu dengan pencarian yang "melebar", namun perbedaannya terletak pada pemilihan cabang yang akan di-ekspan. Pada Breadth First Search, cabang yang di-ekspan pertama tidak berdasarkan apapun, bisa dibebaskan. Pada algoritma A*, simpul yang di-ekspan memenuhi peraturan fungsi evaluasi, yang secara general adalah:

$$f(n) = g(n) + h(n)$$

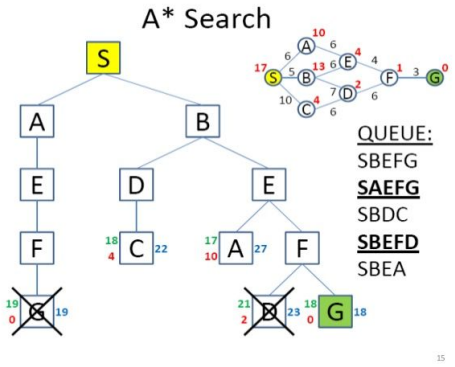
f(n) : fungsi evaluasi.

g(n) : estimasi biaya dari simpul awal ke simpul *n*.

h(n) : estimasi biaya dari simpul *n* ke simpul target.

Dalam tiap kedalaman, akan dipilih simpul dengan nilai *f(n)* yang terkecil. Dengan begitu, harapannya adalah biaya dari simpul awal ke simpul tujuan menghasilkan biaya paling kecil.

Karena *h(n)* ini sebuah estimasi, maka ada aturan yang harus dipenuhi *h(n)* agar algoritma ini berjalan seperti seharusnya, yaitu heuristik yang dipilih haruslah *admissible*. Artinya, *h(n)* tidak pernah membesar-besarkan jarak, sehingga jika dibandingkan dengan jarak sebenarnya, misalnya *h'(n)*, akan lebih kecil. Pada makalah ini, heuristik yang digunakan adalah jarak *euclidean*.



Gambar 4.Ilustrasi algoritma A*.
(Sumber:https://www.slideshare.net/hemak15/lecture-14-heuristic-searcha-star-algorithm, diakses pada 2 Mei 2020)

Untuk mengeksekusi algoritma A*, diperlukan struktur data *priority queue* agar simpul dengan biaya terkecil dimasukkan pada antrian paling depan. Selain itu, dibutuhkan data nilai *h(n)* dari setiap simpul. Langkah pencarian yang dilakukan adalah dengan *step-step* berikut:

1. Kunjungi simpul awal *v* dan masukan ke antrian *q*.
2. Pop antrian *q*. Misal hasil pop adalah simpul *v,* maka hitung nilai *f(v)* dan simpan. Jika *v* adalah simpul target, cek apakah nilai *f(v)* dengan jalur yang telah disimpan lebih kecil atau tidak dengan nilai *f(v)* yang baru. Jika ya, maka jadikan simpul target sebagai solusi sementara, dan perbaharui nilai *f(v)*. Jika tidak, maka cari simpul lain dengan melanjutkan ke langkah 4. Jika *v* bukan simpul target, lanjutkan ke langkah 3.
3. Cari semua simpul yang bertetangga dengan simpul *v* pada langkah 2, dan hitung masing-masing *f(n)* - nya dan masukan ke antrian *q*. Simpul dengan *f(n)* paling kecil akan masuk ke antrian paling depan. Lanjutkan ke langkah 4.
4. Jika antrian *q* habis, maka pencarian sudah selesai. Biaya minimum telah tersimpan. Jika belum, maka lanjutkan ke langkah 2.

Pada Gambar 4, jika kita mencari simpul G, maka urutan simpul yang di ekspan adalah S, C, A, E, F, D, B, D, E, F, lalu ditemukan G.

Algoritma A* memiliki beberapa properti, yaitu:

- Completeness?

  Ya asalkan simpul terbatas
- Optimal?

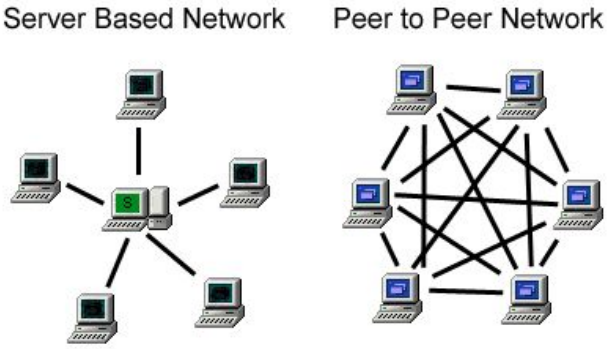
  Ya
- Time?

  $O(b^m)$
- Space?

  $O(b^m)$

F. Jaringan Peer-To-Peer (P2P)

Jaringan peer-to-peer adalah bentuk jaringan komputer yang merupakan bentuk jaringan yang memungkinkan pekerjaan dilakukan oleh masing-masing peer / komputer yang saling terkoneksi. P2P Berbeda dengan yang biasanya digunakan oleh server kini, karena P2P bersifat terdesentralisasi, yaitu tidak ada satu komputer khusus yang bertugas untuk menyediakan servis, dan semua komputer yang menginginkan akses pada servis tersebut harus terkoneksi pada komputer khusus tersebut. Pada jaringan P2P, tiap peer pada sebuah jaringan bertindak sebagai *client* dan *server*, yang berarti *file* terdapat di banyak komputer *client* sebagai *server*, dan diminta oleh banyak komputer *client*.



Gambar 3.Ilustrasi jaringan berbasis server dibanding peer-to-peer.
(Sumber: https://mind42.com/public/df20efe1-d240-461f-8ba4-a78ee9172002 diakses pada 1 Mei 2020)

Jaringan peer-to-peer didesain dengan menggunakan *overlay network*, yaitu sebuah jaringan yang dibuat diatas jaringan lain. Jaringan ini berjalan diatas jaringan internet yang menggunakan TCP/IP. *Overlay network* peer-to-peer membuat subset sendiri *node* dari jaringan dibawahnya. Himpunan *node* inilah yang digunakan sebagai *peer* yang dapat saling mencari satu sama lain.

Jaringan peer-to-peer dapat dibagi menjadi dua, yaitu :

1. *Unstructured*, yang berarti jaringannya tidak mempunyai struktur yang jelas. *Overlay network* tidak mengharuskan jaringan untuk mematuhi suatu peraturan tertentu.
2. *Structured*, yang berarti jaringannya mematuhi suatu topologi jaringan tertentu.

G. Torrenting

Torrenting adalah tipe *file-sharing* yang menggunakan teknologi peer-to-peer, dimana user-user dapat saling berhubungan dan berbagi file. Ada beberapa istilah yang harus diketahui, yaitu:

1. *Peers*, yaitu semua pengguna yang berinteraksi dalam proses *file-sharing* yang berlangsung.
2. *Seeders*, yaitu pengguna yang mengunggah *file* yang akan dibagikan agar *user* lain dapat mengunduhnya.
3. *Leechers*, yaitu pengguna yang hanya mengunduh *file* saja tanpa mengunggahnya kembali, sehingga *file* yang diunduh "berkurang" kuotanya.
4. *Tracker*, yaitu list yang berisi alamat IP dari pengguna torrent sebagai sumber untuk mengunduh *file*.

Ketika sebuah pengguna internet ingin membagikan sebuah *file*, misal file X, maka ia akan menggunakan aplikasi torrent untuk membuat file torrent dari file yang ingin ia bagikan. *File* torrent ini hanya mengandung informasi *metadata*, yaitu informasi *file* X. Didalamnya terdapat informasi nama *file*, *tracker,* dan detail lain seperti ukuran *file* dan *file* apa saja yang dapat diunduh. Lalu, file ini diunggah ke suatu *server* agar pengguna lain dapat mengunduhnya.

Ketika sebuah pengguna internet mengunduh torrent, ia harus menggunakan aplikasi *torrent client* seperti BitTorrent atau uTorrent. Saat terbuka, informasi *tracker* pada *file torrent* akan ditambah dengan IP pengguna internet yang mengunduh torrent tersebut. Lalu, aplikasi *torrent client* akan membaca isi detail *tracker* untuk melihat alamat-alamat IP yang ada, lalu mengunduh file dari alamat IP tersebut.

III. Aplikasi Algoritma BFS dan DFS Pada Pencarian Peer Torrent

A. Struktur Data dan Istilah

Pada saat pengguna aplikasi client torrent meminta untuk mengunduh, ia akan mendapatkan list alamat IP yang dapat digunakan sebagai sumber mengunduh file dari *tracker*. Setelah itu, semuanya akan disimpan pada sebuah larik, misalnya *l*. Semua IP di *l* akan dicari dalam suatu jaringan komputer. Semua data ketetanggaan pada jaringan komputer disimpan pada matriks ketetanggaan *m*, yaitu larik dua dimensi yang menyatakan apakah dua alamat IP saling bertetangga atau tidak. Matriks ketetanggaan berukuran *n x n* dimana *n* adalah banyaknya alamat IP di jaringan komputer. Elemen matriks *e(i, j)* dan *e(j, i)* bernilai sama, yaitu *true* jika alamat IP pada *i* bertetangga dengan alamat IP pada *j*, atau *false* jika tidak bertetangga. Selain itu juga diperlukan larik *t*, yang menyimpan jalur alamat IP yang pada akhirnya menuntun kepada *tracker* yang ditemukan di jaringan komputer agar dapat menginisiasi *file sharing* dengan alamat IP tersebut..

B. Penggunaan Breadth First Search Pada Pencarian Peer

Penggunaan Breadth First Search adalah dengan mencari *peer* dengan cara melebar. Untuk struktur data antrian / *queue*, digunakan instans *q*. Langkah yang diambil adalah:

1. Ambil alamat IP pengguna sebagai simpul pertama untuk dikunjungi. Masukkan pada antrian *q*.
2. Pop antrian *q*, simpan hasil pop pada suatu variabel, misalnya *v*. Jika *q* bukan simpul awal dan *q* adalah alamat IP yang dicari, simpan pada larik *l*.
3. Pop antrian *q*, dan simpan hasil pop pada suatu variabel, misalnya *v*. Cek semua IP yang bertetangga secara langsung dengan *v* pada matriks ketetanggaan *m*, dan masukan pada antrian *q*.
4. Jika antrian *q* habis, maka pencarian sudah selesai. Jika larik *l* kosong, tidak dapat dimulai *file sharing*

karena tidak ada alamat yang ditemukan. Jika larik *l* berisi, inisiasi *file sharing*.

C. Penggunaan Depth First Search Pada Pencarian Peer

Penggunaan Depth First Search adalah dengan mencari *peer* dengan cara mendalam. Struktur data *stack* digunakan instans *s*. Langkah yang diambil adalah:

1. Ambil alamat IP pengguna sebagai simpul pertama untuk dikunjungi. Masukkan pada *stack* s.
2. Cek nilai *top* dari *s*. Misal *top* dari *s* adalah *v*. Pilih satu alamat IP yang bertetanggan dengan *v*, dan belum pernah dikunjungi, misal *v'*. Jika *v'* adalah alamat IP yang dicari, maka masukkan ke larik *l* dan hentikan pencarian. Jika bukan, *push v'* ke *s*.
3. Jika *top* dari *s* punya tetangga, ulangi langkah 2.
4. Jika *top* dari s tidak punya tetangga, *pop s*. Dengan begini, kita kembali ke alamat IP sebelumnya. Lalu ulangi langkah 3.
5. Jika *stack* sudah habis, artinya tidak ditemukan alamat IP yang dicari.

D. Penggunaan A* Pada Pencarian peer

Penggunaan Breadth First Search dan Depth First Search dapat dilakukan sekali saja. Untuk algoritma A*, harus dilakukan pencarian pada setiap alamat IP pada tracker secara terpisah, agar dapat mengeksekusi fungsi heuristik yang tepat. Langkah yang diambil berikut dilakukan pada setiap alamat IP:

1. Kunjungi alamat IP pengguna dan masukan ke antrian *q*.
2. Pop antrian *q*. Misal hasil pop adalah alamat IP *v,* maka hitung nilai *f(v)* dan simpan. Jika *v* adalah alamat IP target, cek apakah nilai *f(v)* dengan jalur yang telah disimpan lebih kecil atau tidak dengan nilai *f(v)* yang baru. Jika ya, maka jadikan rute yang dikalkulasikan sebagai solusi sementara, dan perbaharui nilai *f(v)*. Jika tidak, maka cari rute lain dengan melanjutkan ke langkah 4. Jika *v* bukan simpul target, lanjutkan ke langkah 3.
3. Cari semua simpul yang bertetangga dengan alamat IP *v* pada langkah 2, dan hitung masing-masing *f(n)* -nya dan masukan ke antrian *q*. Simpul dengan *f(n)* paling kecil akan masuk ke antrian paling depan. Lanjutkan ke langkah 4.
4. Jika antrian *q* habis, maka pencarian sudah selesai. Biaya minimum telah tersimpan. Jika belum, maka lanjutkan ke langkah 2.

IV. Kesimpulan

Pada makalah ini, hanya dibahas beberapa algoritma yang dapat dijadikan pilihan saat mencari sebuah alamat IP yang akan digunakan sebagai sumber *seed* ketika sedang menggunakan layanan torrent.

Dengan algoritma Breadth First Search dan Depth First Search, ia akan menemukan semua alamat IP yang dituju jika ada pada jaringan komputer. Akan tetapi, kekurangannya adalah algoritma ini tidak akan memilih jalur yang paling dekat. Asalkan jalurnya sudah ditemukan, pencarian untuk sebuah simpul akan diberhentikan. Tetapi ada juga keuntungannya, yaitu algoritma yang diimplementasikan lebih mudah, dan cepat berjalan ketika ada di jaringan komputer yang kecil.

Sedangkan untuk algoritma A*, karena sifat algoritmanya adalah optimisasi, maka hasil dari setiap jalur untuk simpul yang dicari akan optimal. Akan tetapi, algoritma ini mempunyai beberapa kekurangan. Contohnya adalah ia memerlukan komputasi yang lebih besar, apalagi ketika jaringan sudah sangat besar. Selain itu, algoritma ini harus dieksekusi untuk setiap target yang ada pada *tracker*, karena heuristik yang berjalan hanya bisa satu simpul saja. Adapun keuntungannya adalah karena jalur yang dipilih optimal, waktu yang dibutuhkan untuk berhubungan dengan *seed* target akan lebih cepat, juga koneksi yang lebih kuat.

Akan tetapi, bahasan tentang *trade-off* dan performa yang lengkap terhadap masing-masing algoritma belum dibahas, sehingga harus dibahas diluar makalah ini agar dapat memilih dengan pasti algoritma mana yang menjadi pertimbangan ketika mencari *peer*.

V. Video Link at Youtube

https://youtu.be/TvhZK8lub6s

VII. Ucapan Terima Kasih

Pertama-tama penulis menyatakan syukur pada Allah SWT. yang atas berkah dan rahmat-Nya lah penulis dapat diberi kesempatan untuk belajar materi strategi algoritma dan menyelesaikan masalah ini. Tanpa bantuan-Nya, tidak ada dari semua ini yang dimungkinkan. Lalu penulis mengucapkan terimakasih pada orang tua penulis yang selalu menyempatkan waktu untuk memberi dukungan dalam bentuk apapun, baik yang besar maupun yang tidak, dalam proses pembelajaran strategi algoritma sehingga penulis mempunyai kemampuan untuk menulis dan menyelesaikan makalah ini. Terakhir, penulis ucapkan terima kasih sebesar-besarnya untuk para dosen pengajar IF2211 Strategi Algoritma tahun ajaran 2019/2020, yaitu pada Bapak Dr. Ir. Rinaldi Munir, M.T., Ibu Dr. Masayu Leylia Khodra, ST., MT., dan Ibu Dr. Nur Ulfa Maulidevi, ST., MT. karena telah membimbing dengan tulus dan memberi materi yang penulis kira adalah salah satu mata kuliah yang krusial dalam kehidupan sebagai mahasiswa teknik informatika pada jangka pendek, dan kehidupan sebagai lulusan teknik informatika pada jangka panjang.

VIII. Referensi

[1] Munir, Rinaldi. 2004. “Diktat Strategi Algoritma”, Bandung.

[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/BFS-dan-DFS-(2019).pdf diakses pada 1 Mei 2020 pukul 18.30

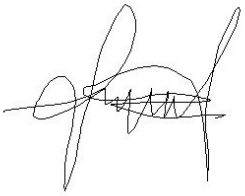
[3] https://www.howtogeek.com/141257/htg-explains-how-does-bittorrent-work/ diakses pada 1 Mei 2020 pukul 18.30

[4] https://www.rcrwireless.com/20180418/overlay-networks-explained-tag27-tag99 diakses pada 1 Mei 2020 pukul 18.30

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, 29 April 2020



Fabian Zhafransyah, 13518022