Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 3, No. 1, Januari 2019, hlm. 515-520 http://j-ptiik.ub.ac.id

Analisis Performansi Algoritma *Greedy Best First Search* dan *Dijkstra* Pada Aplikasi Pencarian Jalur Pendonor Darah Terdekat

Andro Subagio[1], Bayu Rahayudi[2], Mochammad Ali Fauzi[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]androsubagio@student.ub.ac.id, [2]ubay1@ub.ac.id, [3]moch.ali.fauzi@ub.ac.id

Abstrak

Palang Merah Indonesia (PMI) memegang peranan yang sangat penting dalam memberikan informasi yang dibutuhkan masyarakat. Salah satu informasi utama yang dibutuhkan oleh masyarakat adalah persediaan darah. Untuk mendapatkan informasi yang berkaitan dengan persediaan darah di PMI, terdapat prosedur yang harus dilakukan terlebih dahulu oleh keluarga pasien yaitu mengirim formulir dan contoh darah ke Bank darah di rumah sakit atau ke laboratorium Unit Transfusi dan Donor Darah (UTDD) PMI di Kota tersebut. Namun, ketersediaan stok darah di UTDD PMI seringkali tidak mencukupi kebutuhan darah dan terpaksa pihak keluarga pasien yang harus mencari pendonor darah yang cocok dengan golongan darah pasien. Hal ini tentunya tidak memberikan kemudahan bagi pihak keluarga pasien. Selain itu keluarga pasien yang sedang membutuhkan darah juga tidak mudah untuk mencari pendonor darah yang lokasinya terdekat dari rumah sakit. Pencarian rute terpendek (*shortest path)* memiliki beberapa permasalahan, tetapi permasalahan utama pencarian rute terpendek tentu saja mencari rute atau jalur terpendek yang memungkinkan. Pada penelitian ini akan dianalisis performansi perangkat lunak untuk mencari informasi pendonor darah terdekat dengan dua metode berbeda, yaitu metode *Greedy Best First Search* dan *Dijkstra*, selain menganalisis performansi dari algoritma tersebut penelitian ini juga dapat mengetahui persentase kesamaan hasil keluaran dari pencarian jarak terdekat kedua algortima tersebut. Hasil nilai order of growth algoritma *Dijkstra* lebih baik daripada metode *Greedy Best First Search* dengan nilai O(n^2) dan kesamaan hasil dari kedua algoritma tersebut adalah 75%.

Kata kunci : *darah, Dijkstra, Greedy Best First Search*

Abstract

The Indonesian Red Cross (PMI) plays a very important role in providing information needed by the community. One of the main information needed by the community is blood supply. To get information related to blood supply at PMI, there is a procedure that must be done first by the patient's family, namely sending the form and blood sample to the blood bank in the hospital or to the laboratory of PMI Blood Donation Unit (UTDD) in the City. However, the availability of blood stock in the PMI UTDD is often insufficient for blood and forced by the families of patients who have to find blood donors that match the blood type of the patient. This certainly does not make it easy for the patient's family. In addition, the family of patients who are in need of blood is also not easy to find blood donors whose location is closest to the hospital. The shortest path search has several problems, but the main problem is finding the shortest route, of course, looking for the shortest possible route or path. In this study, the performance of the software will be analyzed to find information on the nearest blood donor with two different methods, namely the Greedy Best First Search method and Dijkstra, in addition to analyzing the performance of the algorithm. . The result of the Dijkstra algorithm's order of growth value is better than the Greedy Best First Search method with a value of $O(n^2)$ and the similarity of the results of the two algorithms is 75%.

Keywords : *blood, Dijkstra, Greedy Best First Search*

1. PENDAHULUAN

Palang Merah Indonesia (PMI) memegang peranan yang sangat penting dalam memberikan informasi yang dibutuhkan masyarakat. Salah satu informasi utama yang dibutuhkan oleh masyarakat adalah persediaan darah. Informasi ini sangat penting mengingat kebutuhan akan tranfusi darah dapat terjadi kapan saja seperti untuk korban kecelakaan yang dalam kondisi gawat darurat, pasien dengan kebutuhan mendesak seperti operasi jantung, bedah perut atau para penderita penyakit darah seperti thalassemia.

Untuk mendapatkan informasi yang berkaitan dengan persediaan darah di PMI, terdapat prosedur yang harus dilakukan terlebih dahulu oleh keluarga pasien yaitu mengirim formulir dan contoh darah ke Bank darah di rumah sakit atau ke laboratorium Unit Transfusi dan Donor Darah (UTDD) PMI di Kota tersebut. Namun, ketersediaan stok darah di UTDD PMI seringkali tidak mencukupi kebutuhan darah dan terpaksa pihak keluarga pasien yang harus mencari pendonor darah yang cocok dengan golongan darah pasien. Hal ini tentunya tidak memberikan kemudahan bagi pihak keluarga pasien. Selain itu keluarga pasien yang sedang membutuhkan darah juga tidak mudah untuk mencari pendonor darah yang lokasinya terdekat dari rumah sakit.

Pencarian rute terpendek (shortest path) memiliki beberapa permasalahan, tetapi permasalahan utama pencarian rute terpendek tentu saja mencari rute atau jalur terpendek yang memungkinkan. Namun untuk implementasinya, persoalan ini dapat dikembangkan lebih luas lagi diantaranya untuk mencari biaya minimum, dan lain-lain. Intinya adalah mencari solusi yang paling efektif yang dapat diterapkan dalam persoalan yang dihadapi (Prahasta, 2009). Pada penelitian ini akan menggunakan algoritma *Dijkstra* dan *Greedy Best First Search* untuk dianalisis besar ruang waktu dan ruang memori yang digunakan oleh algoritma tersebut pada aplikasi pencarian lokasi pendonor darah terdekat. Analisis algortima dilakukan agar dapat mengetahui efisiensi waktu dan ruang dari algoritma tersebut dan dapat mengetahui performa algoritma tersebut lebih baik daripada algoritma yang lain. Untuk menganalisis sebuah algoritma dapat dilakukan dengan cara mencari nilai Order of Growth (OOG). Nilai OOG dapat digunakan agar dapat mengetahui kecepatan waktu eksekusi dari algoritma teserbut apabila diberi nilai masukan yang semakin lama semakin besar.

Pada penelitian ini akan dianalisis performansi perangkat lunak untuk mencari informasi pendonor darah terdekat dengan dua metode berbeda, yaitu metode *Greedy Best First Search* dan *Dijkstra*, selain menganalisis performansi dari algoritma tersebut penelitian ini juga dapat mengetahui persentase kesamaan hasil keluaran dari pencarian jarak terdekat kedua algortima tersebut, sehingga dapat mengetahui berapa persen tingkat kesamaan kedua algoritma tersebut.

Pada aplikasi pencarian pendonor darah dengan jarak terdekat ini akan dilakukan perbandingan implementasi metode *Greedy Best First Search* dan *Dijkstra*. Dengan perbandingan dua metode tersebut diharapkan dapat memberikan hasil performansi metode yang lebih baik dan efektif dalam menentukan jarak terdekat pendonor darah

2. LANDASAN KEPUSTAKAAN

2.1 *Dijkstra*

Algoritma *Dijkstra* adalah algortima yang sering digunakan untuk mencari rute terpendek, penggunaan algortima dijsktra yaitu menggunakan simpul-simpul pada jalan yang tidak rumit (Chamero, 2006). Algortima *Dijkstra* berasal dari nama penemunya sendiri yaitu Edsger Dikstra.

Dalam mencari jarak terpendek, algoritma *Dijkstra* menggunakan prinsip Greedy, yaitu mencari solusi optimum pada setiap langkah yang dilalui dengan tujuan mendapatkan hasil optimum pada langkah-langkah selanjutnya untuk mendapatkan solusi terbaik dalam mencari jarak terpendek.

Algortima dijsktra memiliki cara kerja yang hampir sama dengan algoritma BFS yaitu menggunakan prinsip antrian (queue), tetapi antrian pada algortima *Dijkstra* merupakan antrian yang berprioritas (priority queue). Antrian berprioritas ini digunakan pada algoritma *Dijkstra* untuk menentukan simpul yang berprioritas tertinggi saja yang ditelusuri. Dalam menentukan simpul berprioritas, algortima ini membandingkan setiap nilai bobot dari simpul yang berada sama satu level dengan simpul tersebut. Selanjutnya nilai bobot simpul tersebut akan disimpan untuk dibandingkan dengan nilai yang ditemukkan dari rute yang

baru ditemukan kemudian, dan begitu seterusnya hingga ditemukan simpul yang dicari. Asalkan nilai bobot simpul tidak bernilai negatif, jika nilai bobot simpul negatif maka akan memberikan lebih dari satu solusi jarak terpendek (Khantanapoka, 2009).

2.2 *Greedy Best First Search*

Greedy Best First Search termasuk dalam kategori algoritma informed search (pencarian heuristik). Prinsip Greedy adalah mengambil keputusan terbaik pada saat terjadi masalah yang diharapkan keputusan tersebut menjadi solusi terbaik. Oleh karena itu, keputusan yang dibuat harus keputusan terbaik, karena keputusan yang telah diambil tidak dapat diubah lagi.

Greedy Best First Search sama seperti algoritma Best First Search yang memiliki sebuah fungsi evaluasi f(n). Nilai fungsi evaluasi pada *Greedy Best First Search* bergantung pada nilai fungsi heuristik h(n) itu sendiri. Fungsi heuristik h(n) akan memberikan estimasi arah yang benar, sehingga pencarian jalur terpendek dapat sangat cepat. Secara matematis fungsi evaluasi pada *Greedy Best First Search* dapat ditulis :

$$f(n)=h(n) \quad (2.1)$$

Dengan :

f(n) = fungsi evaluasi

h(n) = estimasi biaya dari simpul n ke simpul tujuan

3. METODOLOGI

3.1 Alur *Dijkstra*

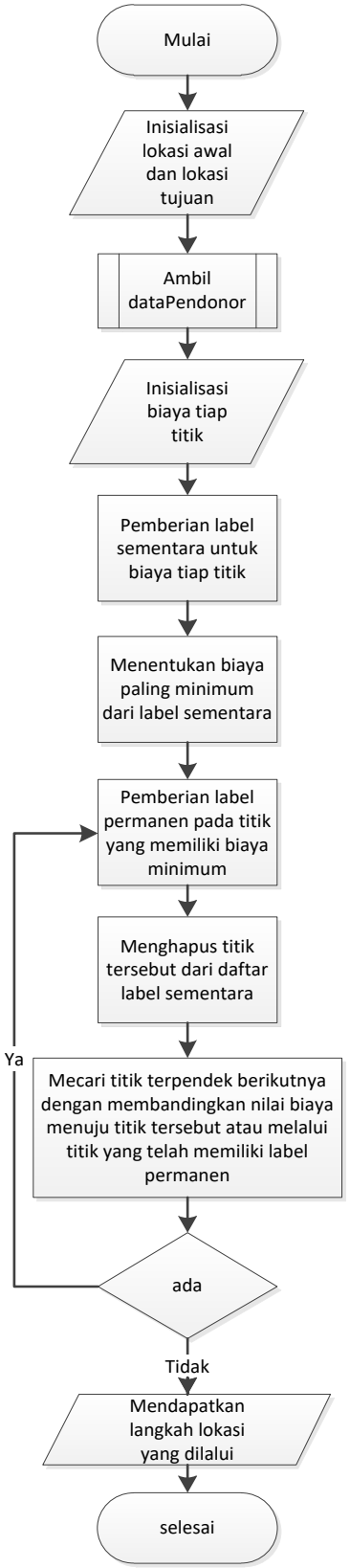
Dalam memahami proses detail pada sistem pencarian pendonor darah terdekat dengan metode *Dijkstra* dan *Greedy Best First Search* pada aplikasi ini, secara garis besar diagram alir pencarian pendonor darah terdekat dengan metode *Dijkstra* yang ditunjukkan pada Gambar 1. Dimulai dari proses inisialisasi lokasi awal pengguna, lokasi awal pengguna langsung diambil dari posisi pengguna saat menggunakan aplikasi tersebut dengan fungsi API Google Map. Proses dilanjutkan dengan pengguna menentukan golongan darah dan rhesus yang ingin dicari oleh pengguna, setelah pengguna menentukan golongan darah dan rhesus yang akan dicari dilanjutkan dengan mengambil dataPendonor pada server. Setelah mendapatkan inisialisasi awal lokasi pengguna dan lokasi tujuan pengguna sebagai inputan dilanjutkan menghitung jarak terpendek menggunakan metode *Dijkstra*.

Proses metode *Dijkstra* diawali dengan inisialisasi biaya tiap titik yang akan dilalui, titik yang telah di inisialisasi akan diberi label sementara. Titik yang memiliki label sementara akan ditentukan titik yang memiliki biaya paling minimum atau biaya termurah, dan titik yang memiliki biaya termurah akan diberi label permanen dan dihapus dari daftar titik label sementara. Selanjutnya akan mencari titik berikutnya, pencarian titik berikutnya dilakukan dengan membandingkan biaya menuju semua titik berikutnya atau melalui titik yang telah memiliki label permanen. Apabila tidak ada titik berikutnya yang akan dilalui maka pencarian jarak terpendek telah selesai, dan apabila masih ada titik yang akan dilalui akan terjadi pengulangan untuk pemberian label permanen pada titik tersebut sampai selesai.

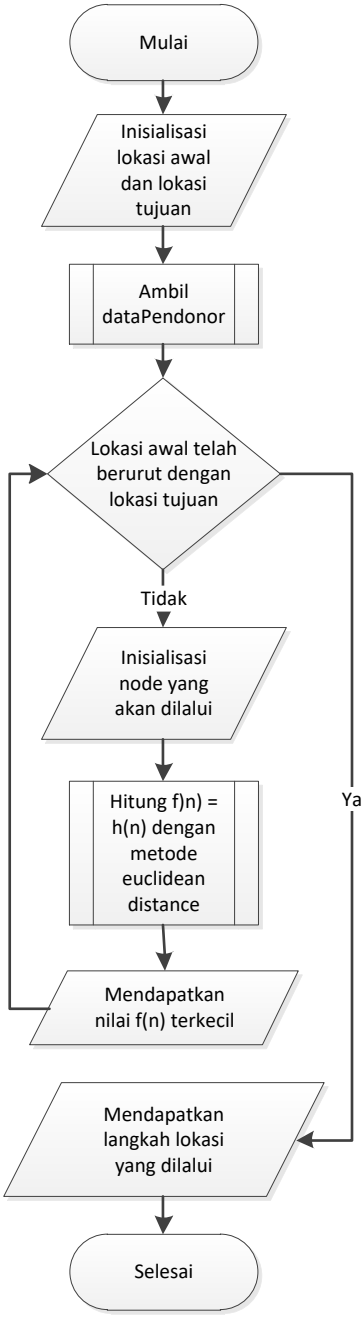
3.2 Alur *Greedy Best First Search*

Untuk proses pencarian pendonor darah terdekat dengan metode *Greedy Best First Search* ditunjukkan pada gambar 2. Proses pencarian pendonor darah dimulai dari inisialisasi lokasi awal pengguna, inisialisasi lokasi awal pengguna diambil saat pengguna menggunakan aplikasi tersebut dengan fungsi API Google Map. Selanjutnya pengguna menentukan golongan darah dan rhesus yang ingin dicari. Saat telah menentukan golongan darah dan rhesus yang ingin dicari, maka diambil dataPendonor dari server untuk dilanjutkan ke proses pencarian pendonor terdekat dengan metode *Greedy Best First Search*.

Proses pencarian dengan metode *Greedy Best First Search* dimulai dari pengecekan lokasi awal berurut atau terhubung dengan lokasi tujuan, jika lokasi awal berurut atau terhubung dengan lokasi tujuan maka pencarian jarak terdekat dengan metode *Greedy Best First Search* telah selesai dan apabila lokasi awal tidak berurut atau tidak terhubung dengan lokasi tujuan maka akan dilakukan proses inisialisasi semua *node* atau titik yang akan dilalui. Setelah menginisialisasi semua titik yang akan dilalui dilanjutkan dengan proses menghitung nilai h(n) dengan metode euclidean distance. Penghitungan nilai h(n) dilakukan untuk mendapatkan nilai f(n) terkecil. Semua proses perulangan akan dilakukan sampai lokasi awal telah berurut atau terhubung ke lokasi tujuan.



Gambar 1 Diagram Alir *Dijkstra*



Gambar 2 Diagram Alir *Greedy Best First Search*

4. PENGUJIAN DAN ANALISIS

Pengujian akan dilakukan dengan memperhatikan pertumbuhan jumlah nilai n dengan 10 kali pengujian. Nilai n merupakan hasil dari nilai fungsi f(n). Peningkatan jumlah langkah pengujian seperti inilah yang menyebabkan kita mengukur efisiensi algoritma dengan ukuran pertumbuhan jumlah langkah pengujian relatif terhadap jumlah data.

Selain itu juga ada pengujian hasil keluaran, Hasil output dari 8 kali pengujian dengan dua metode yang berbeda yaitu metode

```
1  def Dijkstra(graph, start):
2  S = set()
3  delta = dict.fromkeys(list(graph.vertices), math.inf)
4  previous = dict.fromkeys(list(graph.vertices), None)
5  delta[start] = 0
6  while S != graph.vertices:
7  v = min((set(delta.keys()) - S), key=delta.get)
8  for neighbor in set(graph.edges[v]) - S:
9  new_path = delta[v] + graph.weights[v, neighbor]
10 if new_path < delta[neighbor]:
11 delta[neighbor] = new_path
12 previous[neighbor] = v
13 S.add(v)
14 return (delta, previous)
```

Dijkstra dan *Greedy Best First Search* akan dibandingkan kesamaan jumlah rute untuk mendapatkan algortima yang lebih efisien dalam pencarian pendonor darah terdekat di Kota Malang.

Pada proses algoritma *Dijkstra* terdapat perulangan bersarang yang ditunjukkan pada baris kode program ke 6 dan 8, sehingga nilai fungsi f(n) dinotasikan pada Persamaan 4-1. Pada Persamaan 4-1 menghasilkan nilai fungsi n memiliki nilai n^2 yang ditunjukkan pada Persamaan 4-2. Dari nilai fungsi n menunjukkan bahwa nilai waktu pertumbuhan kode program adalah O(n^2).

f(n)=c x n x n (4.1)

f(n)=c x n^2 (4.2)

```
1  def Dijkstra(graph, start):
2  S = set()
3  delta = dict.fromkeys(list(graph.vertices), math.inf)
4  previous = dict.fromkeys(list(graph.vertices), None)
5  delta[start] = 0
6  while S != graph.vertices:
7  v = min((set(delta.keys()) - S), key=delta.get)
8  for neighbor in set(graph.edges[v]) - S:
9  new_path = delta[v] + graph.weights[v, neighbor]
10 if new_path < delta[neighbor]:
11 delta[neighbor] = new_path
12 previous[neighbor] = v
13 S.add(v)
14 return (delta, previous)
```

Pada proses algoritma *Greedy Best First Search* terdapat dua kali perulangan yang ditunjukkan pada baris kode 3 dan 8. Sehingga nilai fungsi n dinotasikan pada Persamaan 6-3. Dari Persamaan 4-3 dapat di sederhanakan menjadi Persamaan 4-4, sehingga nilai fungsi n memiliki nilai n . Dari nilai fungsi n menunjukkan nilai waktu pertumbuhan kode program adalah O(n).

f(n)=c x n+c x n (Persamaan 4-3)

f(n)=n (Persamaan 4-4)

```
1  def greedyBFS(G, source, dest, heuristics, pos):
2  visited = {}
3  for node in G.nodes():
4  visited[node] = False
5  final_path = []
6  goal = greedyBFSUtil(G, source, visited, final_path, dest, 0)
7  prev = -1
8  for var in final_path:
9  if prev != -1:
10 curr = var
11 nx.draw_networkx_edges(G, pos, edgelist = [(prev,curr)], width = 2.5, alpha = 0.8, edge_color = 'black')
12 prev = curr
13 else:
14 prev = var
15 return
```

Tabel 1 Pengujian Hasil Keluaran

Awal - Tujuan	Rute (Dijkstra)	Rute (Greedy)	Jarak (Dijkstra)	Jarak (Greedy)
Anisa – Andre (A-)	Anisa, Umam, Rani, Deny, Karid, Andre	Anisa, Pratama, Sari, Fadel, Vineka, Ganindar, Finish, Aang, Hafidzul, Ilham, Nur, Andre	9.700	19.600
Aang – Subagio (A+)	Aang, Sari, Fadel, Vineka, Bonita,	Aang, Hafidzul, Ilham, Nur, Andre,	18.650	10.050

	Arius, Anisa, Umam, Rani, Deny, Karid, Subagio	Subagio		
Pratama – Ilham (B-)	Pratama, Sari, Aang, Hafidzul, Ilham	Pratama, Sari, Aang, Hafidzul, Ilham	8.150	8.150
Putra – Rani (B+)	Putra, Karid, Deny, Rani	Putra, Karid, Deny, Rani	6.200	6.200
Umam – Evita (O-)	Umam, Rani, Deny, Adzhana, Evita	Umam, Rani, Deny, Adzhana, Evita	5.900	5.900
Karid – Arius (O+)	Karid, Deny, Rani, Umam, Anisa, Arius	Karid, Deny, Rani, Umam, Anisa, Arius	7.400	7.400
Kurnia – Chandra (AB-)	Kurnia, Marmi, Putra, Karid, Chandra	Kurnia, Marmi, Putra, Karid, Chandra	4.600	4.600
Vineka – Jason (AB+)	Vineka, Bonita, Jason	Vineka, Bonita, Jason	3.800	3.800

Uji coba dilakukan berdasarkan hasil keluaran jalur dari masing algoritma yang digunakan yaitu *Dijkstra* dan *Greedy Best First Search.* Data yang digunakan untuk pengujian ini adalah data pendonor darah di Kota Malang. Jalur dan jarak yang tercatat pada hasil percobaan di Tabel 1 merupakan hasil dari 8 (delapan) kali percobaan. Algoritma *Dijkstra* pada percobaan yang dilakukan memiliki jalur terpendek dengan jarak tempuh terpendek pada lokasi awal Anisa dan lokasi tujuan Andre dengan hanya mengunjungi 5 node tetangga dan jarak tempuh 9.700, Sedangkan untuk algoritma *Greedy Best First Search* mengunjungi 11 node tetangga dan jarak tempuh 19.600. Tetapi untuk percobaan dengan lokasi awal Aang dan lokasi tujuan Subagio, algoritma *Greedy Best First Search* memiliki jalur terpendek dengan jarak tempuh terpendek yaitu mengunjungi 5 node tetangga dengan jarak tempuh 10.050 dan untuk algoritma *Dijkstra* mengunjungi 11 node tetangga dengan jarak tempuh 18.650. Dari 8 (delapan) kali percobaan yang dilakukan memiliki kesamaan hasil keluaran pada 6 (enam) dari 8 (delapan) percobaan sehingga untuk persentase kesamaan hasil keluarannya yaitu 75%.

5. KESIMPULAN

Berdasarkan hasil pengujian pertumbuhan waktu kode program di dapatkan bahwa nilai fungsi n dari *Dijkstra* lebih besar daripada *Greedy Best First Search*, Sehingga waktu eksekusi dari algoritma *Dijkstra* lebih cepat dibandingkan dengan *Greedy Best First Search*.

Berdasarkan hasil pengujian pencarian jalur pendonor darah terdekat dengan 8 (delapan) kali pengujian di dapatkan persentase kesamaan hasil 75%. Hasil tidak sama 25% karena terdapat dua pencarian yang menghasilkan rute berbeda.

6. DAFTAR PUSTAKA

Prahasta, Eddy (2009). Sistem Informasi Geografis Konsep-konsep Dasar : Informatika.