Penerapan Algoritma BFS & DFS untuk Routing PCB

Hisham Lazuardi Yusuf 13515069
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
hishamlazuardi@gmail.com
13515069@std.stei.itb.ac.id

***Abstrak*— Perkembangan dunia teknologi dalam satu dekade terakhir telah berkembang secara pesat, terutama perkembangan di dalam bidang elektro atau elektronika. Banyak hal-hal yang telah tercipta atau berkembang seiring dengan perkembangan di dalam dunia elektronika. Untuk mempermudah proses pendesainan PCB terutama dalam melakukan *routing* PCB maka dapat diimplementasikan algoritma seperti DFS dan BFS untuk menghubungkan komponen-komponen di dalam PCB yang dianalogikan seperti simpul di dalam graf dan jalur konduktor yang dianalogikan seperti sisi di dalam graf. Pada makalah ini akan diuraikan dan diimplementasikan bagaimana penerapan algoritma DFS dan BFS untuk mengatasi permasalahan di dalam *routing* PCB.**

Keywords—BFS, DFS, PCB, routing, simpul

I. PENDAHULUAN

Perkembangan dunia teknologi dalam satu dekade terakhir telah berkembang secara pesat, terutama perkembangan di dalam bidang elektro atau elektronika. Banyak hal-hal yang telah tercipta atau berkembang seiring dengan perkembangan di dalam dunia elektronika. Salah satunya adalah perkembangan di dalam rangkaian elektrik seperti penggunaan PCB atau *Printed Circuit Board* untuk alat-alat atau rangkaian elektrik seperti mikrokontroller contohnya Arduino, radio, kalkulator, controller, dan alat-alat lainnya.

PCB atau Printed Circuit Board adalah sebuah circuit board yang terdiri dari komponen-komponen elektronika yang tersusun membentuk suatu rangkaian elektronik tanpa dihubungkan dengan kabel. Alat-alat atau rangkaian elektrik yang menggunakan PCB biasanya lebih *compact* dan lebih tersusun rapi dibandingkan dengan tidak menggunakan PCB.

Dalam proses pembuatan PCB atau *Printed Circuit Board* ada beberapa hal yang perlu diperhatikan agar PCB dapat bekerja secara baik dan benar. Salah satunya adalah dalam hal pendesainan PCB, dalam mendesain PCB ada beberapa peraturan yang harus diterapkan agar PCB dapat bekerja dengan benar yaitu adalah tidak boleh ada *routing* jalur konduktor pada PCB yang bersilangan di dalam satu layer yang satu sisi. Oleh karena itu, untuk menghindari terjadinya jalur konduktor yang bersilangan diantara komponen-komponen maka dapat digunakan penerapan algoritma DFS dan BFS untuk menangani permasalahan routing PCB ini.

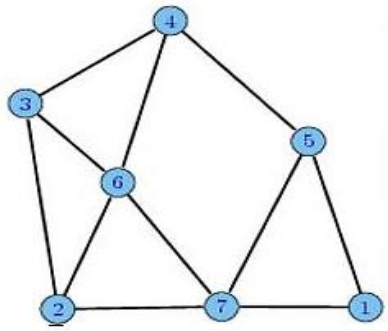
Dalam melakukan penerapan algoritma DFS dan BFS untuk masalah *routing* PCB atau *Printed Circuit Board* ini ada dua komponen yang dianalogikan sebagai titik mulai atau *start* dan titik tujuan atau *goal* dan komponen-komponen lainnya dinomori dengan nomor 1, 2, 3, dan seterusnya sesuai dengan jumlah komponen. Kemudian akan dicari jalur konduktor yang dapat ditempuh dari komponen-komponen yang berada di PCB dan tidak bersilangan satu sama lainnya.

Gambar 1.1 *Printed Circuit Board*
Sumber : *malfurqan.wordpress.com*

II. TEORI DASAR

2.1 Definisi Graf

Graf adalah suatu struktur diskrit yang didefinisikan sebagai suatu pasangan himpunan (V, E) yang ditulis dalam notasi G = (V, E) dengan V merupakan suatu himpunan yang tidak kosong dari simpul (*vertices* atau *node*), V = {v_1, v_2, ….., v_n} dan E adalah suatu himpunan sisi (*edge* atau *arcs*) yang menghubungkan sepasang simpul, E = {e_1, e_2, ……, e_n}. Sebuah graf mungkin tidak mempunyai satu buah sisi namun minimal harus ada satu simpul.



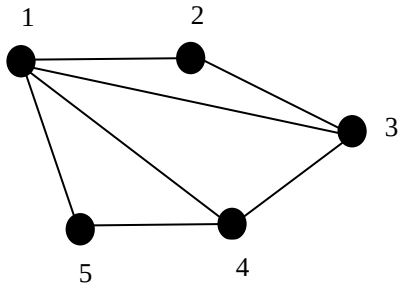
Gambar 2.1 Graf
Sumber : *math.stackexchange.com*

Graf dapat dikelompokkan menjadi beberapa kategori tergantung pada sudut pandang pengelompokkannya. Graf dapat dikelompokkan berdasarkan sisi graf mempunyai orientasi arah atau tidak, jumlah sisi pada graf, ada tidaknya sisi kalang (*loop*) atau sisi ganda pada graf.

2.1.1 Graf Sederhana

Graf sederhana adalah graf yang tidak mengandung sisi ganda ataupun sisi kalang (*loop*). Pada graf sederhana, sisi merupakan suatu pasangan tak-terurut. Sehingga kita bias menuliskan sisi (u,v) sama dengan sisi (v,u).

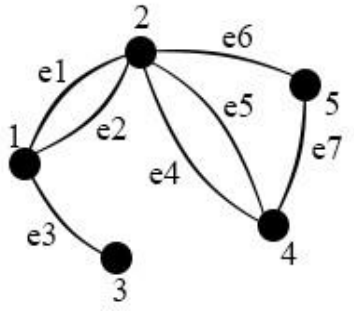
Kita bias mendefinisikan graf sederhana G = (V,E), yang terdiri dari himpunan simpul yang tidak kosong dan pasangan tak-terurut berbeda yang merupakan sisi.



Gambar 2.2 Graf Sederhana

2.1.2 Graf Ganda

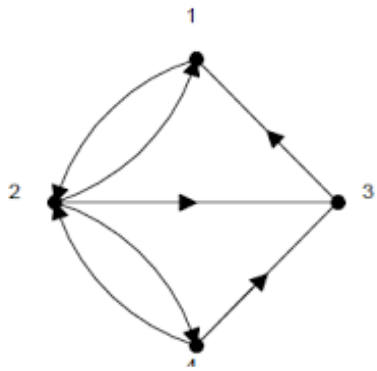
Graf ganda adalah graf tak sederhana yang dapat mengandung sisi ganda, namun tidak ada sisi kalang (*loop*). Sisi ganda dapat didefinisikan sebagai sebuah pasangan tak-terurut yang sama. Definisi dari graf ganda G = (V, E) adalah sebuah himpunan simpul yang tidak kosong dan himpunan-ganda yang mengandung sisi ganda.



Gambar 2.3 Graf Sederhana

2.1.3 Graf Berarah

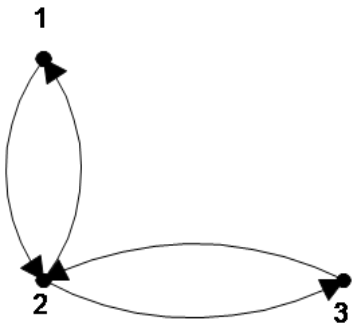
Graf berarah adalah graf yang setiap sisinya diberikan orientasi arah dan tidak dapat mengandung sisi ganda. Graf berarah G = (V, E) adalah sebuah graf dengan himpunan simpul yang tidak kosong dan himpunan sisi-sisi yang memiliki bentuk (u,v) dan (v,u), dimana setiap (u,v) ≠ (v,u). Untuk busur (u,v), dikatakan bahwa u merupakan simpul asal (*initial vertex*) dan v merupakan simpul terminal (*terminal vertex*).



Gambar 2.4 Graf Berarah

2.1.4 Graf Ganda Berarah

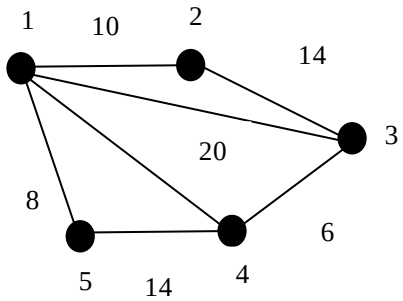
Graf ganda berarah adalah graf yang setiap sisinya diberikan orientasi arah dan setiap sisi nya boleh mengandung sisi ganda. Graf berarah G = (V, E) adalah sebuah graf dengan himpunan simpul yang tidak kosong dan himpunan-ganda dari sisi yang memiliki bentuk (u,v) dan (v,u), dimana setiap (u,v) ≠ (v,u). Untuk busur (u,v), dikatakan bahwa u merupakan simpul asal (*initial vertex*) dan v merupakan simpul terminal (*terminal vertex*).



Gambar 2.5 Graf Ganda Berarah

2.1.5 Graf Berbobot

Graf berbobot adalah suatu graf yang setiap sisinya diberi sebuah nilai atau bobot. Nilai atau bobot pada setiap sisinya dapat merepresentasikan suatu masalah yang ingin dipecahkan, misalnya pada graf dibawah nilai atau bobot pada sisi menunjukkan jarak antara dua simpul yang terhubung.



Gambar 2.6 Graf Berbobot

2.2 Depth-First Search

Algoritma Depth-First Search atau DFS adalah algoritma traversal untuk melakukan pencarian di dalam suatu graf tanpa informasi tambahan (*Uninformed Search*) 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.

```
procedure DFS(input v:integer)
{Mengunjungi seluruh simpul graf dengan algoritma pencarian DFS

Masukan: v adalah simpul awal kunjungan
Keluaran: semua simpulyang dikunjungi ditulis ke layar
}
Deklarasi
   w : integer

Algoritma:
   write(v)
   dikunjungi[v]←true
   for w←1 to n do
      if A[v,w]=1 then {simpul v dan simpul w bertetangga }
          if not dikunjungi[w] then
             DFS(w)
          endif
      endif
   endfor
```

Gambar 2.7 Pseudocode Algoritma DFS
Sumber : Diktat Rinaldi Munir

Kelebihan dari penggunaan algoritma DFS adalah pemakaian memori yang hanya sedikit dibandingkan dengan algoritma BFS dan jika solusi yang dicari berada pada simpul yang paling kiri maka solusi dapat ditemukan dengan cepat.

Kelemahan dari penggunaan algoritma BFS adalah algoritma ini tidak mempunyai jaminan untuk menemukan solusi atau bisa dibilang pencarian dapat hilang ketika suatu graf atau pohon yang dibangkitkan mempunyai level yang terlalu dalam dan DFS belum tentu menemukan solusi yang paling optimal di dalam suatu permasalahan yang berhubungan dengan graf

2.3 Breadth-First Search

Algoritma Breadth-First Search atau BFS adalah algoritma traversal untuk melakukan pencarian di dalam graf tanpa informasi tambahan (*Uninformed Search*) dengan cara memilih simpul-simpul yang bertetangga dengan simpul asal dan menelusuri semua simpul tersebut sesuai dengan aturan urutan pembangkitan simpul hingga semua simpul telah dibangkitkan atau solusi telah ditemukan.

```
procedure BFS(input v:integer)
{ Traversal graf dengan algoritma pencarian BFS.

  Masukan: v adalah simpul awal kunjungan
  Keluaran: semua simpul yang dikunjungi dicetak ke layar
}
Deklarasi
   w : integer
   q : antrian;

   procedure BuatAntrian(input/output q : antrian)
   { membuat antrian kosong, kepala(q) diisi 0 }

   procedure MasukAntrian(input/output q:antrian, input v:integer)
   { memasukkan v ke dalam antrian q pada posisi belakang }

   procedure HapusAntrian(input/output q:antrian,output v:integer)
   { menghapus v dari kepala antrian q }

   function AntrianKosong(input q:antrian) → boolean
   { true jika antrian q kosong, false jika sebaliknya }

Algoritma:
   BuatAntrian(q)          { buat antrian kosong }

   write(v)                { cetak simpul awal yang dikunjungi }
   dikunjungi[v]←true      { simpul v telah dikunjungi, tandai dengan
                             true}
   MasukAntrian(q,v)       { masukkan simpul awal kunjungan ke dalam
                             antrian}

  { kunjungi semua simpul graf selama antrian belum kosong }
   while not AntrianKosong(q) do
       HapusAntrian(q,v)   { simpul v telah dikunjungi, hapus dari
                             antrian }
       for tiap simpul w yang bertetangga dengan simpul v do
             if not dikunjungi[w] then
                write(w)     {cetak simpul yang dikunjungi}
                MasukAntrian(q,w)
                dikunjungi[w]←true
             endif
       endfor
   endwhile
  { AntrianKosong(q) }
```

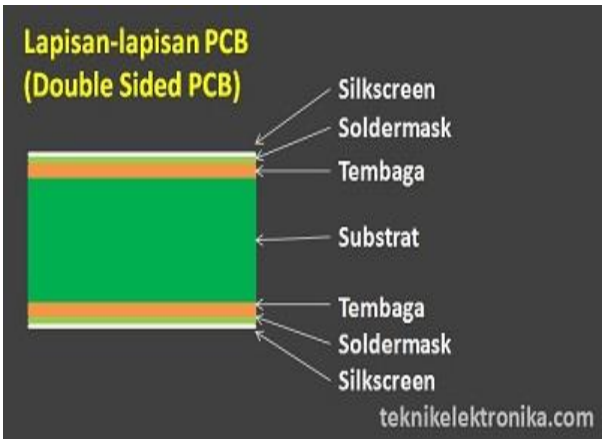
Gambar 2.8 Pseudocode Algoritma BFS
Sumber : Diktat Rinaldi Munir

Kelebihan dari penggunaan algoritma BFS adalah algoritma ini pasti selalu menemukan solusi yang optimal dengan berbagai macam bentuk graf atau pohon di dalam suatu permasalahan dan pasti menemukan solusi optimal.

Kelemahan dari penggunaan algoritma BFS adalah kompleksitas memori yang dibutuhkan oleh algoritma BFS sangat besar dibandingkan dengan DFS karena menyimpan semua simpul yang berada di dalam graf atau pohon.

2.4 PCB (Printed Circuit Board)

PCB atau *Printed Circuit Board* adalah sebuah *circuit board* yang terdiri dari komponen-komponen elektronika yang tersusun membentuk suatu rangkaian elektronik tanpa dihubungkan dengan kabel. PCB ditemukan oleh seorang ilmuwan Austria yang bernama Paul Eisler pada tahun 1936. Paul Eisler menggunakan PCB pertama kalinya di sebuah rangkaian Radio. Kemudian pada tahun 1943, Amerika Serikat mulai memanfaatkan teknologi PCB ini pada Radio Militer dalam skala yang lebih besar. Tiga tahun setelah perang dunia kedua yaitu pada tahun 1948, PCB mulai digunakan untuk produk-produk komersil oleh perusahaan-perusahaan Amerika Serikat.



Gambar 2.9 Struktur PCB
Sumber : *teknikelektronika.com*

Secara struktur PCB terdiri seperti kue lapis yang terdiri dari beberapa lapisan yang digabung menjadi satu. PCB terdiri dari 4 lapisan dasar yaitu lapisan substrat, tembaga, soldermask, dan silkscreen.

Lapisan dasar (landasan) PCB biasanya disebut dengan Substrat. Bahan Substrat yang paling umum digunakan adalah FR2 dan FR4. FR2 atau Flame Resistant 2 adalah kertas bonding resin sintetis (synthetic resin bonded paper) yaitu bahan komposit yang terbuat dari kertas yang diresapi dengan resin plastik formaldehida fenol (*plasticized phenol formaldehyde resin*). Sedangkan FR4 atau Flame Resistant 4 adalah anyaman Fiberglas yang dilapisi dengan resin epoksi (*epoxy resin*). FR4 memiliki daya serap air yang rendah, properti isolasi yang bagus serta tahan suhu panas hingga 140 derajat celcius. Namun, PCB yang berbahan FR4 lebih mahal jika dibandingkan dengan PCB yang berbahan FR2.

Lapisan PCB berikutnya adalah lapisan tembaga tipis yang dilaminasi ke lapisan substrat dengan suhu tinggi tertentu dan perekat. Tergantung pada jenis PCB-nya, lapisan tembaga tipis ini hanya akan dilapisi pada satu sisi substrat untuk jenis Single Sided PCB. Sedangkan Double Sided PCB terdapat lapisan tembaga tipis di dua sisi Substrat. Seiring dengan perkembangan Teknologi manufakturing PCB saat ini, PCB telah dapat dibuat hingga 16 lapisan atau bahkan lebih dari 16 lapisan tergantung pada perancangan PCB dan rangkaian yang diinginkan.

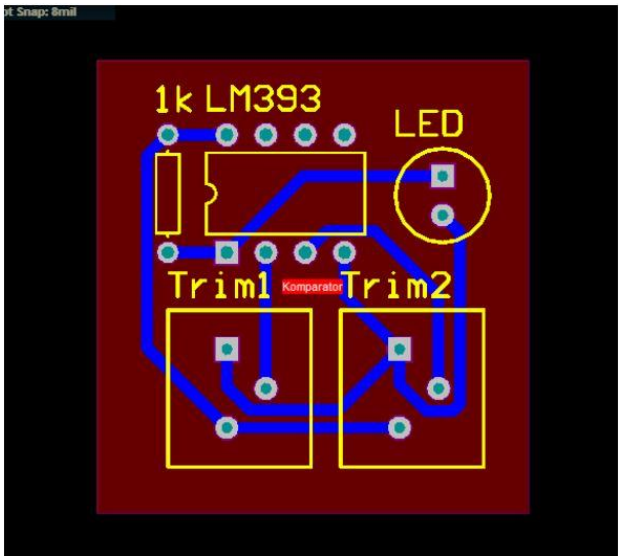
Soldermask adalah lapisan diatas lapisan tembaga yang berfungsi melindungi tembaga atau jalur konduktor dari hubungan atau kontak yang tidak disengaja. Lapisan soldermask ini hanya terdapat pada bagian-bagian PCB yang tidak disolder, sedangkan bagian yang akan disolder tidak ditutupi oleh lapisan soldermask. Lapisan soldermask ini juga dapat membantu para pengguna PCB untuk menyolder tepat pada tempatnya sehingga mencegah solder short (hubung singkat solder). Lapisan soldermask ini biasanya berwarna hijau, namun ada juga yang berwarna lain seperti warna biru dan merah.

Lapisan setelah soldermask adalah lapisan silkscreen yang biasanya berwarna putih atau hitam. Namun ada juga silkscreen yang berwarna lain seperti warna abu-abu, warna

merah dan bahkan ada berwarna kuning keemasan. Silkscreen merupakan cetakan huruf, angka dan simbol pada PCB. Silkscreen ini berfungsi sebagai tanda atau indikator untuk komponen-komponen elektronika pada PCB sehingga mempermudah orang dalam merakitnya.

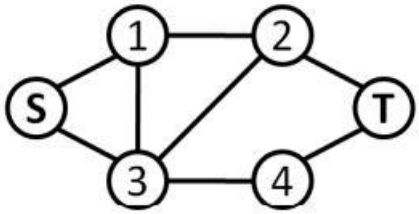
III. Implementasi Algoritma untuk Routing PCB

Dalam melakukan *routing* PCB atau *Printed Circuit Board* terdapat beberapa algoritma yang dapat digunakan untuk melakukan penghubungan jalur konduktor antara komponen yang satu dengan komponen lainnya. Jenis algoritma yang *trivial* dan sering untuk digunakan adalah algoritma BFS (*Breadth-First Search*) dan algoritma DFS (*Depth-First Search*) untuk melakukan *routing* jalur konduktor pada PCB.



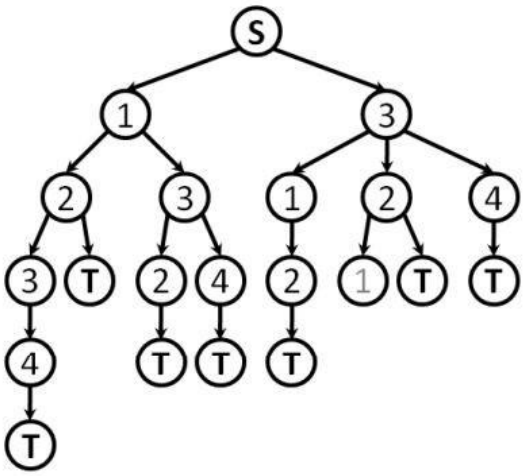
Gambar 3.1 Jalur Konduktor pada PCB

Implementasi algoritma DFS dan BFS yang digunakan hampir sama dengan algoritma DFS dan BFS pada umumnya namun ada beberapa hal yang perlu diperhatikan untuk melakukan *routing* jalur konduktor pada PCB. Dalam implementasi kita akan menentukan salah satu komponen di dalam PCB sebagai titik awal atau sebagai simpul *start* yang dimisalkan sebagai simpul S dan salah satu komponen lainnya sebagai titik tujuan atau sebagai simpul *goal* yang dimisalkan sebagai simpul T.



Gambar 3.2 Graf PCB

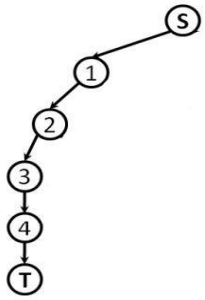
Untuk permasalahan *routing*, iterasi di dalam graf akan dimulai dari simpul S, kemudian untuk algoritma DFS pada *routing* PCB akan melakukan traversal dari simpul yang sekarang ke simpul-simpul yang dikunjungi. Kemudian melakukan *backtracking* melalui simpul-simpul yang telah dikunjungi sebelumnya sampai ke simpul awal lagi. Pencarian di dalam graf akan dihentikan jika semua simpul di dalam graf telah dikunjungi ataupun simpul tujuan telah dicapai.



Gambar 3.2 Depth-First Routing Method

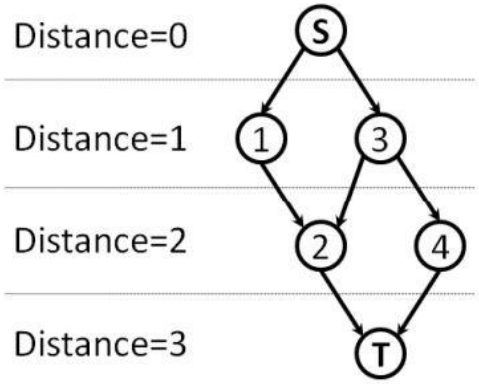
Langkah-langkah yang digunakan untuk memecahkan persoalan *routing* dengan menggunakan algoritma DFS adalah :

1. Pilih salah satu komponen (simpul) sebagai simpul awal (*start*) dan pilih satu komponen (simpul) yang memiliki hubungan dengan simpul awal sebagai simpul tujuan (*goal*)
2. Kemudian kunjungi simpul atau komponen yang bertetangga dengan simpul saat ini dan simpul dengan nomor terkecil dikunjungi lebih dahulu.
3. Jika simpul yang dikunjungi tidak bersilangan sesuai dengan posisi komponen di dalam PCB, maka kunjungi simpul tersebut, jika tidak maka simpul tersebut tidak dapat dikunjungi.
4. Lalu kunjungi simpul-simpul yang bertetangga dengan simpul saat ini dan memenuhi kondisi nomor 3 serta ulangi terus hingga semua simpul telah di kunjungi di dalam graf atau salah satu solusi telah ditemukan.



Gambar 3.3 Solusi Pencarian Menggunakan Algoritma DFS

Dengan menggunakan algoritma BFS, dengan aturan simpul awal dan simpul akhir yang mirip dengan algoritma DFS namun proses pencarian di dalam graf yang mengunjungi simpul bertetangga yang belum dikunjungi di dalam satu level yang sama, dimana level ini ditentukan oleh jarak dari simpul yang dikunjungi saat ini dengan simpul awal (*start node*).



Gambar 3.4 Breadth-First Routing Method

Langkah-langkah yang digunakan untuk memecahkan persoalan *routing* dengan menggunakan algoritma BFS adalah :

1. Pilih salah satu komponen (simpul) sebagai simpul awal (*start*) dan pilih satu komponen (simpul) yang memiliki hubungan dengan simpul awal sebagai simpul tujuan (*goal*)
2. Kemudian kunjungi semua simpul atau komponen yang bertetangga pada level yang sama dan simpul dengan nomor terkecil dikunjungi lebih dahulu.
3. Jika simpul yang dikunjungi tidak bersilangan sesuai dengan posisi komponen di dalam PCB, maka kunjungi simpul tersebut, jika tidak maka simpul tersebut tidak dapat dikunjungi.
4. Lalu kunjungi simpul-simpul pada level selanjutnya dengan simpul saat ini dan memenuhi kondisi nomor 3 serta ulangi terus hingga semua simpul telah di kunjungi di dalam graf atau salah satu solusi telah ditemukan.

Setelah menerapkan langkah-langkah di dalam algoritma BFS dan DFS untuk permasalahan *routing*, selanjutnya ulangi secara terus-menerus algoritma BFS dan DFS hingga semua komponen di dalam PCB menjadi terhubung satu sama lain sesuai dengan rancangan atau *schematic* yang telah dibuat sebelumnya..

IV. KESIMPULAN

Penggunaan algoritma DFS dan BFS dapat diimplementasikan pada permasalahan *routing* jalur konduktor di dalam PCB. Sehingga jalur konduktor yang berada di PCB memenuhi kondisi-kondisi tertentu dan PCB dapat bekerja dengan baik dan benar.

Dengan pemanfaatan algoritma DFS dan BFS ini maka permasalahan *routing* dapat ditangani secara efisien karena kita akan menelusuri komponen-komponen yang diibaratkan seperti simpul-simpul yang saling bertetangga atau berhubungan dan tidak akan memiliki jalur konduktor yang bersilangan diantara komponen-komponen yang berada di atas PCB tersebut. Selanjutnya algoritma DFS dan BFS ini dapat dimanfaatkan untuk pengembangan algoritma *auto-route* yang telah ada di dalam software pendesainan PCB sehingga ketika dilakukan *auto-route* maka jalur-jalur konduktor terhubung antara satu sama lain dan semua komponen terhubung dengan komponen-komponen yang dirancang untuk saling berhubungan antara satu sama lain. Sehingga jalur konduktor yang dihasilkan dengan menggunakan fitur *auto-route* akan efisien dan optimal.

REFERENCES

[1] Munir, Rinaldi, *Diktat Kuliah IF 2251 : Strategi Algoritmik*, Bandung, 2010.

[2] *http://belajarelektronika.net/pengertian-pcb-dan-fungsinya/* diakses pada tanggal 18 April 2017, jam 20.23.

[3] *http://anthonilockheart.blogspot.co.id/2013/04/depth-first-search-dfs-breath-first.html* diakses pada tanggal 18 April 2019, jam 22.05.

[4] Zhang, Ran, *A Study of Algorithms for PCB Design*, Waseda University Doctoral Disertation.

[5] Munir, Rinaldi, *Matematika Diksrit Edisi Ketiga*. Bandung : Informatika. 2010.

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, 19 April 2017

Hisham Lazuardi Yusuf
13515069