Jurnal Natural
Vol. 13, No. 1, 2013

IDENTIFY CONNECTIVITY GRAPH USING A MODIFIED PRÜFER'S ALGORITHM LABELLING TREES

Al Aiyub, Mahyus Ihsan, Rahma Zuhra

Multimedia Research Group
Jurusan Matematika
FMIPA Universitas Syiah Kuala
Darussalam - Banda Aceh

Abstract. Connectivity of graph easily can be given when we see it with the bare of eyes, but needs an algorithm that can assure the connectivity in computerization. Some graphs require the connectivity in their definitions, such as Hamiltonian graph, Eulerian graph and tree. The connectivity become important in order to solve any of graphs' problems by using computer technology. Most of the algorithm cannot run perfectly, if the graph which is given was not connected. It is happened to Djikstra and Prufer graphs for example. However, this study will provide the solution by modifying Prufer algorithm to show the connectivity in the graph.

Keywords: graph, connectivity, algorithm, Djikstra's algorithm, Prufer's algorithm.

I. PENDAHULUAN

Penggunaan graf dalam berbagai bidang untuk menyelesaikan berbagai macam persoalan telah menjadi hal yang lazim sekarang ini. Contohnya pada kasus pencarian jarak terpendek suatu rute perjalanan angkutan kota di suatu daerah. Sebuah graf G memuat 2 himpunan berhingga yang dinyatakan dengan V dan E. Tiap elemen dari himpunan V disebut sebagai vertek dan setiap elemen dari himpunan E disebut sebagai sisi yang merupakan pasangan tak terurut dari 2 vertek. Vertek yang berderajat tepat 1 pada graf disebut dengan *leaf* dan vertek yang berderajat 0 disebut sebagai *isolate* vertek [4]. derajat untuk sebarang vertek v dari graf merupakan jumlah berapa kali vertek v menjadi titik ujung dari suatu sisi dan dinotasikan dengan d(v) [8].

Sebuah graf G yang terkoneksi dan tidak memuat *cycle* di dalamnya maka graf G dikatakan sebagai *tree* [6]. *Tree* merupakan jenis graf yang sering dipakai untuk memodelkan rute perjalanan, pengeluaran biaya dalam suatu proyek, rute pemasangan kabel dengan tujuan dapat meminimumkan waktu dan biaya dikarenakan *tree* tidak memuat *cycle*. Untuk memudahkan penulisannya, t*ree* dapat dibuat dalam bentuk *label* dan *label* tersebut dapat di-rekonstruksi kembali menjadi bentuk *tree* yang isomorfis [4].

Dalam menyelesaikan masalah pada *tree* seperti masalah pembentukan *label* atau pencarian jarak terpendek untuk *tree* berukuran kecil dapat dilakukan secara manual, sedangkan untuk *tree* yang berukuran besar dibutuhkan media komputer untuk memudahkan dalam penyelesaiannya. Algoritma yang tepat dibutuhkan agar komputer dapat menyelesaikannya dengan cepat dan efisien. Algoritma dapat diartikan sebagai prosedur komputasi yang terdefinisi dengan baik yang mengambil beberapa nilai sebagai masukan dan menghasilkan beberapa nilai sebagai output. Sebuah algoritma juga dapat dikatakan sebagai urutan langkah komputasi yang mengubah input menjadi output [1]. Setiap algoritma memiliki tingkat kompleksitas yang berbeda-beda. Kompleksitas suatu algoritma diukur dari waktu (*time*) dan ruang (*space*) yang dibutuhkan untuk menyelesaikan suatu proses semakin cepat waktu penyelesaian dan semakin kecil ruang yang dibutuhkan maka semakin bagus algoritmanya dan sebaliknya [2].

Algoritma akan berjalan dengan baik jika input yang diberikan sesuai dengan apa yang dibutuhkan oleh algoritma. Misalkan pada proses pembentukan *label* dengan menggunakan algoritma Prüfer dan proses penentuan jarak terpendek dengan menggunakan algoritma Djikstra. Secara komputasi, kedua algoritma ini tidak akan berkerja sempurna untuk suatu graf yang tidak terhubung. Suatu graf dikatanan

terhubung jika dan hanya jika ada jalan (*path*) yang menghubungankan setiap pasangan vertek dari graf [7]. Untuk menjamin keterhubungan graf ini diperlukan suatu validasi input yang dapat mengetahui graf atau *tree* yang diberikan terhubung atau tidak. Dalam penelitain ini akan ditunjukkan bahwa bagaimana memodifikasi algroitma Prüfer dalam pembentukan *label* sehingga dapat menentukan suatu graf terhubung atau tidak terhubung.

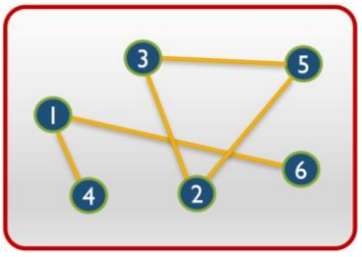
II. METODOLOGI

Analisa proses kerja algoritma Prüfer dalam pembentukan *label*

Algoritma Prüfer dalam pembentukan *label* terdiri dari 5 langkah yaitu:

1. Inisialisasi i = 0, dan $T_i = T$
2. Temukan *leaf* T_i dengan *label* paling kecil sebagai v
3. Simpan label vertek yang berdekatan dengan v
4. Hapus v dari T_i untuk membuat *tree* baru T_{i+1}
5. Jika $T_{i+1} = K_2$ maka berhenti. Jika $T_{i+1} > K_2$ maka tambahkan i dengan 1 dan kembali ke langkah 2.

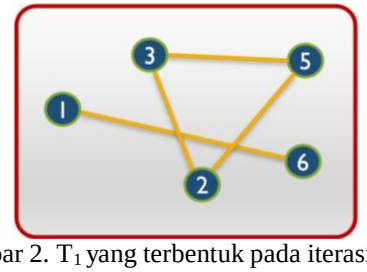
Dimana i menunjukkan indeks iterasi, T merupakan *tree*, v vertek terkecil yang terdapat dalam T, dan K_2 merupakan graf komplit dengan 2 vertek.



Gambar 1. Bentuk awal graf T.

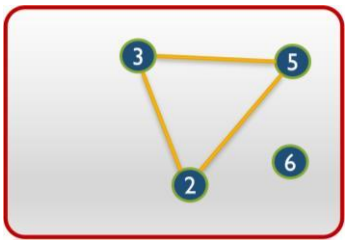
Bila diberikan suatu input graf T yang diperlihatkan pada Gambar 1, proses kerja algoritma Prüfer dalam menangani graf tersebut adalah sebagai berikut:

- Iterasi i = 0, $T_0 = T$, v = 4 karena 4 adalah *leaf* terkecil dari T_0, 1 disimpan sebagai label, dan vertek v beserta sisi yang berdekatan dihapus dari T_0 untuk membentuk T_1 seperti pada Gambar 2.



Gambar 2. T_1 yang terbentuk pada iterasi I = 0.

- Iterasi I = 1, v = 1 karena 1 adalah *leaf* terkecil dari T_1, 6 disimpan sebagai *label*, dan vertek v beserta sisi yang berdekatan dihapus dari T_1 untuk membentuk T_2 seperti pada Gambar 3



Gambar 3. T_2 yang terbentuk pada iterasi i = 1.

- Iterasi i = 2 tidak bisa dilanjutkan lagi dikarenakan tidak ada lagi *leaf* yang dapat dipilih sebagai v padahal hasil akhir dari algoritma yaitu sederatan *label* dan terbentuknya graf komplit dengan 2 vertek.

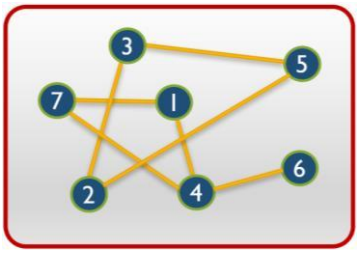
III. HASIL DAN PEMBAHASAN

Hasil analisa dari proses kerja algoritma Prüfer memperlihatkan bahwa algoritma tersebut tidak akan bekerja untuk sebuah graf yang tidak terhubung sehingga dengan memodifikasi algoritma tersebut dapat menentukan suatu graf terhubung atau tidak. Adapun modifikasinya adalah sebagai berikut:

1. Inisialisasi i = jumlah sisi, and $T_i = T$
2. Selama $i \geq 0$ dan tidak ada *isolate* vertek di dalam T_i
   - jika tidak terdapat vertek berderajat 1 maka hapus satu sisi dari vertek v yang paling kecil indeks dan derajatnya serta berdekatan dengan suatu vertek dengan indeks paling kecil dari vertek lain yang berdekatan dengan vertek v untuk membentuk T_{i+1} serta kurangkan i dengan 1
   - Jika terdapat vertek berderajat 1 maka hapus satu vertek yang berderajat 1 dengan indeks terkecil beserta sisi yang berdekatan dengannya untuk membentuk T_{i+1} dan kurangkan i dengan 1
3. Jika i = 0 maka terhubung = *true*. Jika i > 0 maka terhubung = *false*.

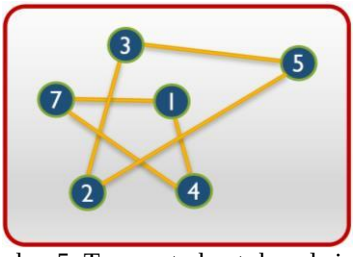
Algoritma hasil modifikasi menunjukkan bahwa bila suatu graf memuat *isolate* vertek sudah dapat dipastikan bahwa graf tersebut tidak terhubung. Kompleksitas waktu pengerjaan algoritma hasil modifikasi tersebut adalah linier terhadap jumlah sisi dari graf yaitu O(n), dimana n menunjukkan jumlah sisi dari graf. Skenario terbaik algoritma ini adalah untuk *input* graf yang memuat *isolate* vertek dan skenario terjelek adalah untuk *input* graf yang semua verteknya terhubung.

Bila diberikan suatu graf seperti pada diperlihatkan pada Gambar 4 maka langkah kerja algoritma hasil modifikasi untuk menentukan keterhubungan graf tersebut yaitu:



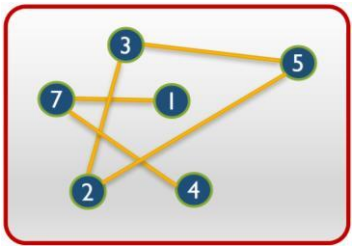
Gambar 4. Bentuk Awal dari graf T.

- i = 7 (karena jumlah sisi graf adalah 7), vertek 6 beserta sisi yang berdekatan dengannya dihapus dari graf karena derajatnya 1 untuk membentuk graf T_1 seperti diperlihatkan pada Gambar 5 dan kurangkan i dengan 1.



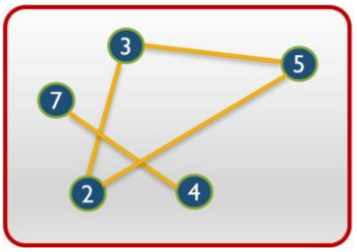
Gambar 5. T_1 yang terbentuk pada i = 7.

- i = 6, karena tidak terdapat vertek yang berderajat satu maka dihapus sebuah sisi yang menghubungkan vertek 1 dan vertek 4 untuk membentuk graf T_2 seperti diperlihatkan pada Gambar 6 dan kurangkan i dengan 1.



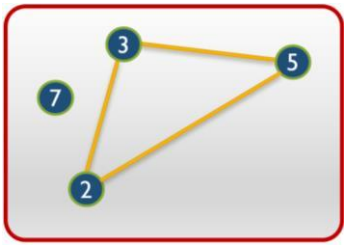
Gambar 6. T_2 yang terbentuk pada i = 6.

- i = 5, 1 merupakan vertek yang berderajat 1 terkecil dari graf. Kemudian vertek 1 serta sisi yang berdekatan dengannya dihapus untuk membentuk graf T_3 seperti diperlihatkan pada Gambar 7 dan kurangkan i dengan 1.



Gambar 7. T_3 yang terbentuk pada i = 5.

- i = 4, 4 merupakan vertek berderajat 1 terkecil dari graf. Kemudian vertek 1 serta sisi yang berdekatan dengannya dihapus untuk membentuk graf T_4 seperti diperlihatkan pada Gambar 8 dan kurangkan i dengan 1.



Gambar 8. T_4 yang terbentuk pada i = 4.

- Karena telah terdapat *isolate* vertek yaitu vertek 7 dan i > 0 maka terhubung = *false*.

Hasil akhir dari proses algoritma yang dimodifikasi menunjukkan bahwa graf pada Gambar 4 adalah graf yang tidak terhubung.

KESIMPULAN

Algoritma Prüfer yang telah dimodifikasi dapat menentukan keterhubungan dalam graf sehingga algoritma tersebut dapat digunakan untuk mem-validasi *input* graf dalam menyelesaikan permasalahan graf dengan bantuan media komputer. Kompleksitas waktu pengerjaan dari algoritma ini linier terhadap jumlah sisi dari graf dan dapat dinyatakan dengan O(n), dimana n menyatakan jumlah sisi dari graf.

UCAPAN TERIMA KASIH

Ucapan terima kasih Penulis kepada anggota Multimedia Research Group Jurusan Matematika FMIPA Unsyiah yang telah turut membantu hingga penelitian ini dapat terlaksana dengan baik.

REFERENSI

1. T.H. Chormen, C.E. Leiserson, R.L. Rivest, Clifford Stein, 2001, *Introduction of Algorithm*, McGraw-Hill, San Francisco.

2. Oded Goldreich, 2008, *Computational Complexity a Conceptual Perspective*, Cambridge University, New York.

3. Goodaire, G.E., and Parmenter, M.M. 2002. *Discrete Mathematics with Graph Theory*, Prentice Hall, Upper Saddle River.

4. Harris, M.J., Hirst, L.J., and Mossinghoff, J.M. 2008. *Combinatorics and Graph Theory*, Springer, New York.

5. J.W. Moon, 1970, *Labelled Trees*, University of Alberta.

6. König, D, 1936, *Theorie der endlichen und unendlichen Graphen*, Akademische Verlagsgesellschaft, New York.

7. R.J. Wilson, 1996, *Introduction to Graph Theory*, Addison Wesley, London.

8. S. Evan, 1979, *Graph Algorithm*, Computer Science, Maryland.