Implementasi dan Analisis Visualisasi Graph Pada Graph Statis Menggunakan Representasi Visual Treemap

Lukman Arie Susanto [1], Kemas Rahmat Saleh W., S.T., M.Eng.[2], Shinta Yulia P, S.T., M.T.[3]
[1]Program Studi S1 Teknik Informatika, School of Computing, Telkom University
[2,3]School of Computing, Telkom University
[1,2,3]Gedung E-F, Jalan Telekomunikasi 1, Terusan Buah Batu, Bandung 40257
[1]lukmanarie@students.telkomuniversity.ac.id, [2]bagindokemas@telkomuniversity.ac.id,
[3]shintayulia@telkomuniversity.ac.id

Abstrak

Implementasi graph dapat diterapkan di bidang *sosial network*, transportasi, biokimia, dan lain-lain. Visualisasi graph pada media sosial menghubungkan antar orang untuk saling bertukar informasi. Pada bidang transportasi menggambarkan jalur atau jalan dengan tempat tertentu sebagai node atau titik. Permasalahan yang timbul jika data tersebut sangat besar maka akan menimbulkan masalah dalam hal visualisasi graph tersebut. Tampilan graph yang besar mempengaruhi pengguna menjadi tidak dapat membaca dan memahami data-data pada visualisasi graph. Selain itu karena data yang besar membuat node-node dan sisi saling *overlap* dan menggunakan ruang tampilan yang besar. Implementasi treemap pada tugas akhir ini adalah menggunakan data graph berarah dengan konten dataset *social network* wiki vote. Selain itu juga output yang ditampilkan yaitu berupa visualisasi treemap 2D. Implementasi treemap ini juga dapat berinteraksi dengan user untuk melihat hubungan antar node. Hasil yang diperoleh dari penelitian ini adalah bahwa dengan menggunakan data graph berarah, maka data yang tampil pada visualisasi lebih banyak daripada jumlah dari dataset. Selain itu juga visualisasi treemap ini efektif karena user dapat melihat node yang dipilih terhubung ke node mana saja. Tetapi kekurangan yang didapat adalah bahwa user terlebih dahulu harus mencari node yang ingin dipilih sehingga hal ini menjadi tidak efisien karena membutuhkan waktu yang lebih banyak. Penelitian yang akan datang diharapkan dapat menggunakan visualisasi graph dengan representasi visual yang lain selain treemap dan juga menggunakan graph yang dinamis.

Kata kunci : *graph berarah, treemap, visualisasi*

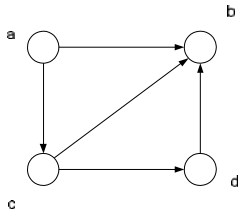
1. Pendahuluan

Visualisasi gambar dengan bentuk graph sering digunakan dalam bidang biokimia, *social network*, transportasi dan lain-lain. Misalkan pada *social network*, orang diidentifikasi sebagai titik dan hubungan yang mewakili antar orang sebagai sisi, maka antar titik terhubung oleh sisi. Jika dalam masalah di atas setiap orang mempunyai teman yang tidak terbatas dan sampai puluhan ribu atau jutaan maka akan menghasilkan titik berjumlah puluhan ribu atau jutaan dengan sisi-sisi yang menghubungi antar titik tersebut. Permasalahan tampilan graph dalam kasus *social network* yang sangat besar mempengaruhi pengguna menjadi tidak dapat membaca data-data pada graph karena tampilan graph yang sangat padat dan kompleks sehingga sulit mencari titik atau node yang sangat penting atau hubungan antar node tersebut dan juga menggunakan ruang tampilan yang besar. Oleh karena itu diperlukan visualisasi yang efektif dan juga efisien supaya dapat memecahkan masalah di atas. Pada tulisan ini penulis membahas tugas akhir dengan judul *Implementasi dan Analisis Visualisasi Graph Pada Graph Statis Menggunakan Representasi Visual Treemap.* Dalam penulisan tugas akhir ini diharapkan dapat membuat visualisasi yang efektif dan efisien serta interaktif agar mudah dibaca dan dipahami oleh pengguna selain itu penggunaan ruang tampilan yang seminimal mungkin.

2. Dasar Teori

2.1 Graph Berarah

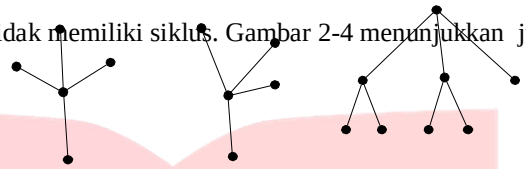
Graph berarah atau digraph terdiri dari kumpulan obyek v terbatas disebut simpul atau titik bersama-sama dengan kumpulan sisi terbatas terarah atau busur, yang memerintahkan pasang simpul. Hal ini seperti graph kecuali bahwa setiap sisi dialokasikan arah, satu titik yang ditunjuk awal dan yang lainnya adalah akhir. Busur diarahkan dari awal s sampai akhir t dinotasikan dengan (s, t), atau hanya st. Hal ini penting untuk mengamati bahwa, tidak seperti graph, digraph dapat memiliki dua busur dengan titik akhir yang sama, asalkan diarahkan dengan cara berlawanan [2].



Gambar 2-1 Jenis Digraph

2.2 Tree

Tree adalah graph terhubung yang tidak memiliki siklus. Gambar 2-4 menunjukkan jumlah *trees* [4].



Gambar 2-2 Beberapa Trees[4]

Trees adalah graph terhubung paling kecil; menghapus sisi dari tree dan itu menjadi terputus. Selain sebagai sebuah kelas penting dari graph, tree yang penting dalam ilmu komputer sebagai struktur data, dan sebagai obyek yang dibangun oleh algoritma pencarian. Sebuah properti fundamental dari tree adalah bahwa semua tree di *n* titik memiliki jumlah yang sama dari sisi [4].

2.3 Representasi Visual Graph Statis

2.3.1 Teknik Space Filling

Teknik *space filling* dapat dikategorikan oleh strategi penempatan bekerja dalam *enclosure*, *adjacency*, dan *crossing* [6].

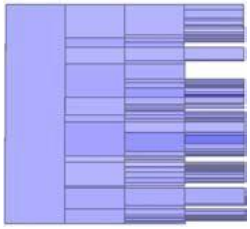
- *Enclosure*

  Teknik ini secara rekursif layout node anak dalam area node induknya. Contoh yang paling menonjol adalah treemaps bentuk persegi panjang secara rekursif pengelompokan ruang tampilan persegi panjang sesuai hierarki yang mendasari, yang diperkenalkan oleh Shneiderman (disebut algoritma *slice* dan *dice*). Teknik tersebut dapat ditampikan baik dalam 2D dan 3D.

Gambar 2-3 Treemap [6]

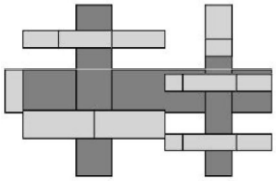
- *Adjacency*

  Berbeda dengan treemaps, teknik berbasis *adjacency* tidak *overlap* node induk oleh node anak dan sebaliknya, mewakili hubungan titik dengan menempatkan node anak di sebelah node induknya. Penempatan dapat di lapisan melingkar seperti dalam metode Sunburst (variasi 2D atau 3D), atau pada lapisan linier, yang menghasilkan disebut *"Icicle plots"*.



Gambar 2-4 *Icicle Plot*[6]

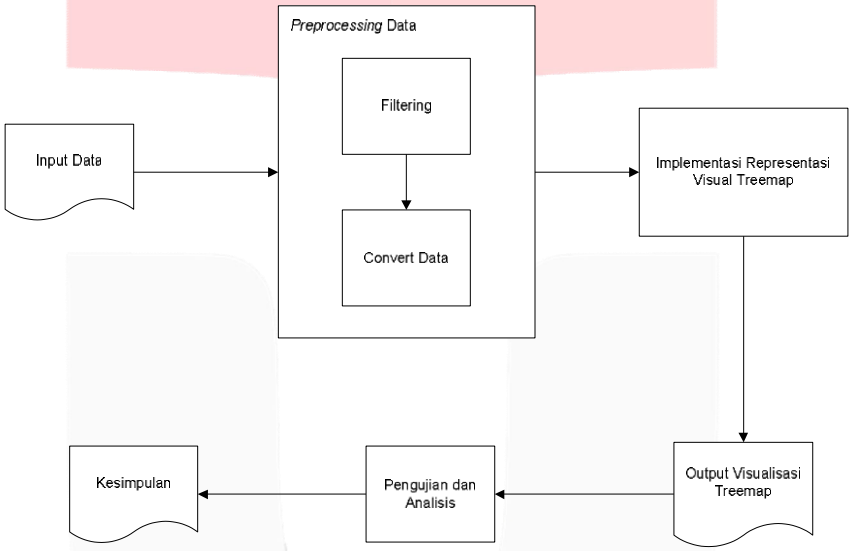
- *Crossings*
  Metode *crossing* tempat node anak di node induk, sehingga hanya sebagian *overlap* orangtua. Metode "Beamtree" meningkatkan lebih masalah treemap klasik dimana struktur hierarkis mungkin sulit untuk menilai secara visual, sementara masih menjadi lebih banyak ruang efisien daripada teknik *adjacency*.



Gambar 2-5 *Beamtree* [6]

3. Pembahasan

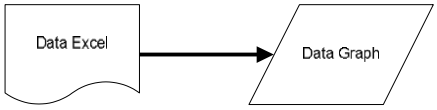
Pada bagian ini menjelaskan arsitektur atau gambaran umum sistem yang akan dibangun dan menganalisis kebutuhan apa saja yang digunakan untuk membangun sistem visualisasi graph dengan representasi visual treemap. Berikut adalah alur sistem visualisasi yang akan dibuat pada gambar 3-1.



Gambar 3-1 Alur Sistem Visualisasi

Tahapan untuk membuat sistem visualisasi graph dengan representasi visual treemap ini adalah sebagai berikut :

1. Input Data
   Untuk input data dalam penelitian ini digunakan dataset dari web *https://snap.stanford.edu/data/* [14]. Pada web ini diberikan beberapa koleksi data dengan *large network* atau jaringan yang sangat besar. Untuk tugas akhir ini menggunakan dataset *social networks* yaitu wiki vote dengan jenis graph berarah. Dari web tersebut data diberikan dalam bentuk *text* atau dengan file ekstensi *text*. Untuk memudahkan dalam membaca data maka file dari *text* diubah menjadi file excel. Dataset yang sudah tersedia kemudian diinputkan ke dalam program untuk proses selanjutnya yaitu *preprocessing* data.
2. *Preprocessing* Data
   Berdasarkan dataset yang diberikan yaitu jenis graph berarah, maka untuk visualisasi graph dengan visual treemap dataset tersebut harus diubah ke dalam bentuk tree. Pada proses ini dataset yang diinputkan dalam bentuk excel diubah terlebih dahulu ke dalam bentuk graph.

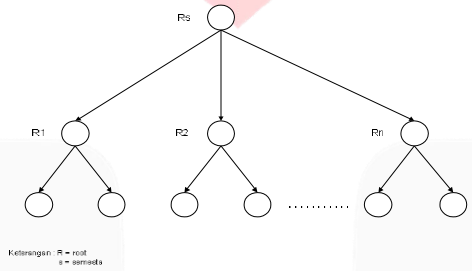


Gambar 3-2 Mengubah Data Excel ke Data Graph

Data yang sudah dalam bentuk graph akan memudahkan proses filtering dan convert data selanjutnya. Hasil akhir dari preprocessing data ini adalah dataset akan berbentuk struktur pohon atau tree dan memiliki struktur seperti dokumen JSON.

a. Filtering

Pada tahapan ini dataset yang sudah ada dipilih atau diklasifikasikan berdasarkan jumlah derajat masuk atau *indegree* yang bernilai nol dari suatu node. Tujuan dari proses ini adalah untuk mencari dan menentukan root. Root ini akan dijadikan elemen pertama pada sebuah tree atau juga dapat disebut sebagai *parent* pertama. Node dengan jumlah derajat masuk bernilai nol, maka node tersebut akan dijadikan root atau akar dalam sebuah tree. Dataset diinputkan menggunakan bahasa pemrograman java untuk mencari node dengan jumlah *indegree* atau derajat masuk bernilai nol. Dari hasil program didapatkan bahwa node yang *indegree* bernilai nol memiliki jumlah lebih dari satu node. Karena jumlah node dengan *indegree* bernilai nol lebih dari satu maka harus menggunakan root semesta untuk menampung semua root yang *indegree* bernilai nol. Selain itu tidak dapat memilih dari salah satu node tersebut untuk dijadikan root, karena akan berakibat pada node-node tersebut tidak akan dikunjungi. Root semesta ini digunakan supaya node-node yang *indegree* nol tidak terpisah satu sama lain dan mengakibatkan visualisasi treemap yang terpisah. Oleh karena itu diperlukan root semesta untuk menampung semua root dari node yang *indegree* nol. Setelah node-node tersebut didapatkan maka tahapan selanjutnya adalah mengkonversi dari data excel diubah menjadi file JSON atau dokumen JSON.



Gambar 3-3 Root Semesta yang Terhubung ke Beberapa Root

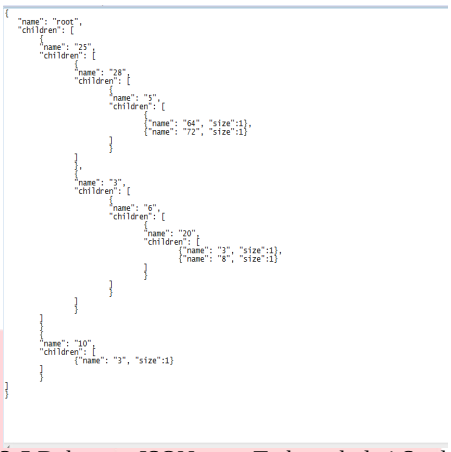
b. Convert data

Setelah node-node dari dataset telah dipilih berdasarkan jumlah derajat masuk atau indegree yang bernilai nol, kemudian dataset tersebut dikonversi ke dalam bentuk struktur pohon. Pengubahan bentuk dataset ini dikarenakan visualisasi treemap harus menggunakan data dengan struktur tree untuk memudahkan membaca dataset.

Gambar 3-4 Mengubah Data Graph ke Dokumen JSON

Dokumen JSON terdapat aturan cara penulisan yang benar untuk dapat dibaca oleh suatu program. Setelah proses filtering sebelumnya, sudah didapatkan hasil node-node dengan *indegree* bernilai nol. Hasil dari proses filtering membuktikan bahwa lebih dari satu node dengan *indegree* nol dan solusi untuk mengatasi masalah tersebut adalah dengan menggunakan node baru yang dijadikan root semesta yang dapat menghubungkan ke semua node dengan *indegree* nol. Setelah root semesta beserta node-node dengan indegree bernilai nol ditampilkan, maka *rule* kedua adalah menampilkan semua node berdasarkan sisi yang menghubungkan dari node dengan *indegree* nol. Semua node yang sudah ditampilkan tersebut dilakukan lagi proses yang sama secara berulang hingga kondisi dimana ada node yang memiliki *outdegree* atau derajat keluar bernilai nol. Selain kondisi itu perlu juga suatu kondisi dimana jika ada sisi yang sudah dikunjungi tidak boleh lagi dikunjungi agar tidak menyebabkan perulangan yang tak terhingga. Sehingga ketika sisi yang sudah dikunjungi diberikan sebuah status *true*, perulangan akan

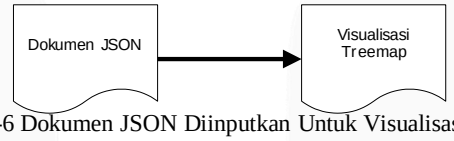
berhenti dan menampilkan node yang terhubung dengan sisi tersebut. Hasil dalam dokumen JSON yang terbentuk adalah seperti gambar di bawah ini.



```
{
  "name": "root",
  "children": [
      {
      "name": "25",
      "children": [
            {
            "name": "28",
            "children": [
                  {
                  "name": "5",
                  "children": [
                        {"name": "64", "size":1},
                        {"name": "72", "size":1}
                  ]
                  }
            ],
            },
            {
            "name": "3",
            "children": [
                  {
                  "name": "6",
                  "children": [
                        {
                        "name": "20",
                        "children": [
                              {"name": "3", "size":1},
                              {"name": "8", "size":1}
                        ]
                        }
                  ]
                  }
            ]
            }
      ],
      },
      {
      "name": "10",
      "children": [
            {"name": "3", "size":1}
      ]
      }
}
```

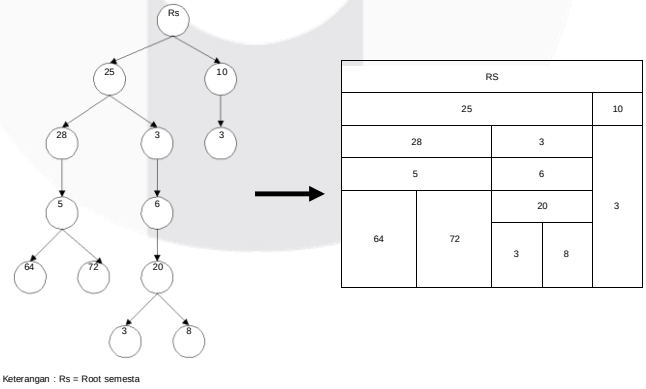

Gambar 3-5 Dokumen JSON yang Terbentuk dari Struktur Tree

3. Implementasi Representasi Visual Treemap
   Tahapan selanjutnya adalah melakukan implementasi representasi visual treemap menggunakan algoritma squarified. Penerapan algoritma squarified treemap pada program menggunakan *library*. Setelah proses *preprocessing* data selesai dilakukan dan dari proses tersebut didapatkan hasil konversi data berupa dokumen JSON. Dari dokumen JSON kemudian diinputkan ke dalam program untuk visualisasi treemap.



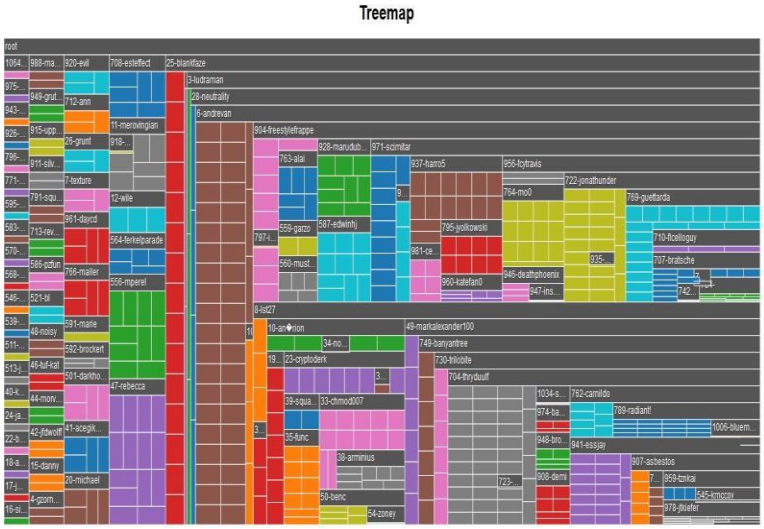
Gambar 3-6 Dokumen JSON Diinputkan Untuk Visualisasi Treemap

Dalam visualisasi treemap, dokumen JSON yang sudah ada dan struktur tree yang terbentuk diimplementasikan ke bentuk treemap. Di bawah ini adalah gambar dari struktur tree dari dokumen JSON yang diubah ke dalam visual treemap.



Gambar 3-7 Transformasi Tree ke Bentuk Treemap

4. Output Visualisasi Treemap
   Hasil dari proses implementasi representasi visual treemap digambarkan seperti gambar di bawah ini.



Gambar 3-8 Implementasi Visual Treemap

Pada gambar di atas merupakan hasil implementasi visual treemap dari program. Data yang ditampilkan pada gambar tersebut dipilih pendekatan *top down*. Dengan pendekatan *top down*, semua node yang ada ditampilkan mulai dari root sampai pada leaf node. Secara umum struktur graph dapat dilihat dan dapat melihat bagian yang menarik dari graph tersebut [6]. Semakin besar ukuran data maka ukuran persegi yang ada pada treemap juga semakin kecil tetapi untuk ukuran treemap itu tetap, dan itu menyebabkan sulitnya membaca isi data tersebut. Tetapi dengan implementasi treemap pada tugas akhir ini, node yang sulit dibaca bisa dipilih atau diklik untuk berganti layer dan melihat struktur yang lebih jelas.

4. Pengujian dan Analisis

4.1 Skenario Pengujian

Dalam melakukan pengujian diperlukan beberapa skenario pengujian yaitu sebagai berikut :

4.1.1 Pengujian Kesesuaian Jumlah Data Node yang Tampil Terhadap Dataset

Pada pengujian ini akan dilakukan pengujian kesesuaian jumlah data node yang tampil terhadap jumlah data dari dataset. Pengujian ini dilakukan dengan cara menghitung jumlah node yang tampil pada sebuah dokumen JSON yang terbentuk hasil dari konversi data.

4.1.2 Pengujian Interaksi User Terhadap Kesesuaian dari Struktur Graph

Pada pengujian ini akan dilakukan pengujian interaksi user terhadap kesesuaian dari struktur graph. Pengujian ini dilakukan dengan cara user berinteraksi secara langsung dengan sistem visualisasi treemap. Dari interaksi user ini maka dapat dilihat apakah treemap yang sudah terbentuk sudah sesuai dengan struktur graph.

4.1.3 Pengujian Perbandingan Struktur Antara Graph Biasa Dengan Treemap

Pada pengujian ini akan dilakukan pengujian perbandingan struktur antara graph biasa dengan struktur dari treemap. Pengujian ini dilakukan dengan cara menampilkan struktur dari graph biasa dan juga menampilkan struktur dari treemap. Setelah ditampilkan maka dilakukan perbandingan antara kedua struktur tersebut. Untuk pengujian ini juga menggunakan 4 node, 5 node, 6 node dan 7 node yang sudah dalam bentuk struktur.

4.1.4 Pengujian Interaksi User Terhadap Informasi yang ada Pada Treemap

Pada pengujian ini akan dilakukan pengujian dari interaksi user terhadap informasi yang ada pada treemap. Selain itu pada pengujian ini user juga dapat berinteraksi secara langsung untuk mencari informasi yang dibutuhkan. Dari pengujian interaksi user untuk mencari sebuah informasi maka akan dapat diketahui apakah user dapat mencari sebuah informasi dengan cepat pada visual treemap ini.

4.2 Analisis Hasil Pengujian

Setelah skenario pengujian selesai dilakukan maka, selanjutnya adalah melakukan analisis terhadap semua skenario pengujian. Berikut ini adalah analisis dari hasil pengujian.

4.2.1 Analisis Hasil Jumlah Data Node yang Tampil Terhadap Dataset

Berdasarkan pengujian kesesuaian jumlah data node yang tampil pada visualisasi treemap terhadap dataset, maka hasil yang didapatkan adalah sebagai berikut :

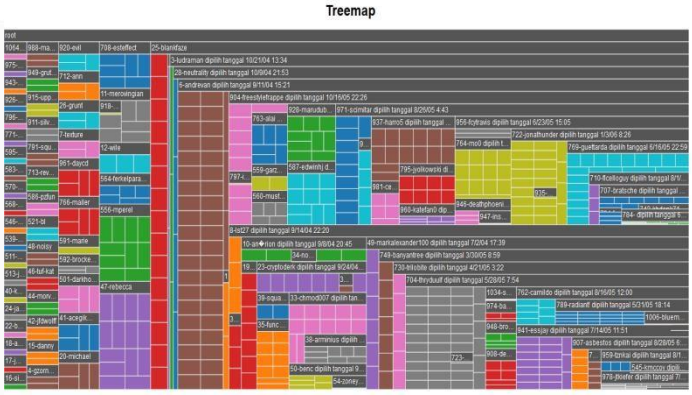
No.	Node	Jumlah
1.	Dataset	643
2.	Data yang tampil	720

Table 4-1 Hasil Penghitungan Node

Dari data tabel diatas dapat disimpulkan bahwa data node yang tampil pada visualisasi treemap lebih banyak daripada data node dari dataset. Hal ini disebabkan bahwa ada node yang melakukan perulangan yang kembali lagi ke node tersebut dan juga ada node lain yang terhubung ke node yang sudah dikunjungi. Jadi ada 77 data node yang ditampilkan lebih dari satu. Dan juga hal tersebut menandakan bahwa graph memiliki siklus dan jika diubah ke dalam bentuk tree maka node yang sudah dikunjungi ditampilkan lagi dalam visual treemap dan tidak perlu menjabarkan lagi hubungan antar node yang sudah dikunjungi tersebut karena akan mengakibatkan perulangan selamanya.

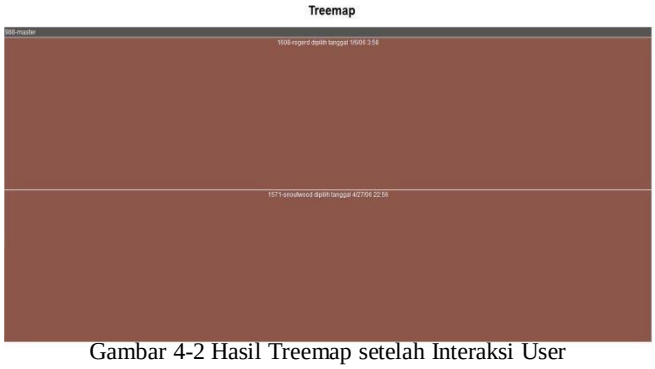
4.2.2 Analisis Hasil Interaksi User Terhadap Kesesuaian dari Struktur Graph

Berikut ini adalah pengujian untuk melakukan interaksi user terhadap kesesuaian dari struktur graph dari visualisasi treemap, maka gambar di bawah ini adalah visual treemap secara umum setelah dijalankan.



Gambar 4-1 Hasil Visual Treemap

Dari gambar di atas maka dilakukan interaksi user dengan memilih salah satu node misalnya node 988. Maka hasil dari interaksi user tersebut digambarkan seperti gambar di bawah ini.



Gambar 4-2 Hasil Treemap setelah Interaksi User

Pada gambar di atas terdapat dua kotak warna coklat dan juga satu warna abu-abu tua. Untuk warna abu-abu tua merupakan *parent* node dan dua warna coklat merupakan *child* node. Hasil gambar di atas juga menampilkan semua atribut atau informasi yang ada pada dataset. Dan untuk memastikan bahwa hasil visual treemap yang terbentuk sesuai dengan struktur graph, maka dilihat dari keterhubungan node dari program. Berikut ini adalah potongan program yang menunjukkan bahwa node 988 terhubung ke node apa saja.

```
988,master : 4/27/06 22:56 1571-snoutwood, 1/6/06 3:58 1608-rogerd,
Indegree : 0
Outdegree : 2
```

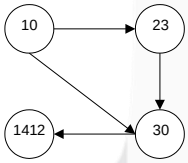
Gambar 4-3 Hasil Potongan Program Suatu Node

Dari gambar hasil potongan program di atas dapat dijelaskan bahwa node 988 dengan username master memiliki keterhubungan dengan dua node yaitu node 1571 dan node 1608. Dan dari hasil visual treemap dengan interaksi user memilih node 988 dan menampilkan hasil seperti gambar 4.4-2 dengan muncul dua persegi berwarna coklat maka dapat disimpulkan bahwa node dari visual treemap sudah sesuai dengan struktur graph beserta dengan atribut yang ada. Berdasarkan pengujian interaksi user terhadap kesesuaian dari struktur graph maka dapat disimpulkan bahwa hasil dari treemap yang terbentuk sudah sesuai dengan struktur graph dari dataset. Hal ini menunjukkan bahwa treemap dengan implementasi dapat pindah layer saat node dipilih menghasilkan visual treemap yang efektif karena user dapat mengetahui node yang dipilih tersebut terhubung ke node mana saja. Tetapi dari hasil pengujian tersebut dapat diketahui bahwa ada kekurangan dari visual treemap. Kekurangan tersebut adalah bahwa user harus mencari terlebih dahulu node yang ingin dipilih pada visual treemap. Dan hal tersebut menunjukkan bahwa membutuhkan waktu yang lebih banyak untuk mencari node yang ingin dilihat. Sehingga visual treemap tersebut tidak efisien dalam hal waktu karena membutuhkan waktu yang lebih banyak untuk melihat node yang diinginkan. Selain itu juga ada beberapa node yang atribut dari data tidak dapat dilihat dengan jelas.

4.2.3 Analisis Hasil Perbandingan Struktur Antara Graph Biasa Dengan Treemap

a. Struktur Graph 4 Node

Dibawah ini adalah gambar struktur graph 4 node dengan membaca data dari dataset.



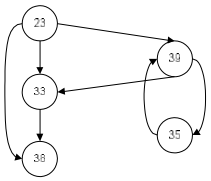
Gambar 4-4 Hasil Struktur Graph dengan 4 Node

Dan berikut ini adalah gambar struktur tree dari dokumen JSON pada treemap.

Gambar 4-5 Hasil Struktur Tree yang Terbentuk dari 4 Node

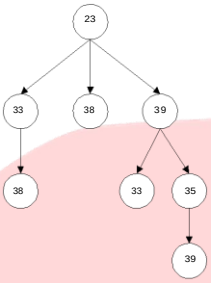
b. Struktur Graph 5 Node

Dibawah ini adalah gambar struktur graph 5 node dengan membaca data dari dataset.



Gambar 4-6 Hasil Struktur Graph dengan 5 Node

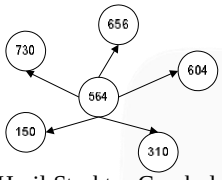
Dan berikut ini adalah gambar struktur tree dari dokumen JSON pada treemap.



Gambar 4-7 Hasil Struktur Tree yang Terbentuk dari 5 Node

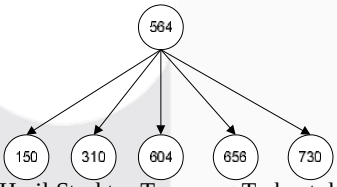
c. Struktur Graph 6 Node

Dibawah ini adalah gambar struktur graph 6 node dengan membaca data dari dataset.



Gambar 4-8 Hasil Struktur Graph dengan 6 Node

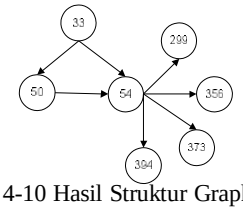
Dan berikut ini adalah gambar struktur tree dari dokumen JSON pada treemap.



Gambar 4-9 Hasil Struktur Tree yang Terbentuk dari 6 Node

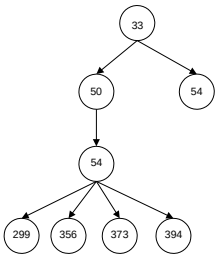
d. Struktur Graph 7 Node

Dibawah ini adalah gambar struktur graph 7 node dengan membaca data dari dataset.



Gambar 4-10 Hasil Struktur Graph dengan 7 Node

Dan berikut ini adalah gambar struktur tree dari dokumen JSON pada treemap.



Gambar 4-11 Hasil Struktur Tree yang Terbentuk dari 7 Node

Berdasarkan hasil pengujian perbandingan struktur graph dengan struktur tree pada treemap maka dapat disimpulkan bahwa dengan pengujian struktur 4 node, 5 node dan 7 node terdapat perbedaan jumlah node dari kedua struktur tersebut. Dari struktur tree yang terbentuk memiliki jumlah node yang lebih banyak dibandingkan dengan struktur graph biasa. Hal ini disebabkan oleh node tersebut memiliki *indegree* atau derajat masuk yang lebih dari satu atau juga disebabkan adanya perulangan yang menuju node tersebut sehingga ketika ada suatu node a yang sudah mendeklarasikan node b tersebut hingga ke *child* node, maka node c yang terhubung ke node b hanya menampilkan node b saja tanpa harus menampilkan lagi *child* node dari node b. Sedangkan untuk pengujian struktur 6 node dapat diketahui bahwa struktur graph biasa sama struktur tree pada treemap memiliki jumlah node yang karena pada struktur graph biasa tersebut tidak memiliki perulangan node atau ada node yang memiliki *indegree* lebih dari satu. Selain itu juga dapat dilihat dari perbandingan struktur graph dengan struktur tree bahwa meskipun berbeda jumlah node, tetapi masing-masing struktur graph atau struktur tree memiliki jumlah edge yang sama.

4.2.4 Analisis Hasil Interaksi User Terhadap Informasi yang ada Pada Treemap

Berdasarkan hasil pengujian interaksi user terhadap informasi yang ada pada treemap atau user ingin mencari suatu informasi dari treemap maka dapat disimpulkan bahwa informasi yang ingin dicari oleh user misalkan mengambil atribut tanggal atau *username* tertentu dari dataset memungkinkan user harus menelusuri satu per satu dari treemap tersebut. Karena informasi yang diinginkan oleh user tidak langsung dapat dilihat begitu saja, melainkan harus mengamati satu per satu node untuk informasi tanggal yang diinginkan. Hal ini menyulitkan user karena mencari suatu informasi menjadi tidak efisien dalam hal waktu pencarian informasi tersebut. Informasi yang didapatkan dari perancangan dan implementasi treemap ini adalah user dapat mengetahui bahwa node yang dengan *indegree* nol merupakan node dengan *username* yang tidak dipilih oleh siapapun atau tidak ada yang *vote* berdasarkan studi kasus wiki *vote*. Hal ini dapat dilihat dari treemp awal pada bagian persegi panjang yang berwarna abu-abu tua dan juga tidak menampilkan atribut tanggal di persegi tersebut.

Gambar 4-12 Hasil Interaksi Treemap

5. Kesimpulan

Dari hasil pengujian dan analisis dalam penelitian tugas akhir ini dapat disimpulkan sebagai berikut.

1. Data graph yang digunakan menunjukkan data node yang tampil pada visualisasi treemap lebih banyak dibandingkan dengan jumlah data node dari dataset. Data node yang tampil lebih banyak ini dikarenakan adanya siklus atau *looping* dan juga ada node yang memiliki *indegree* lebih dari satu. Untuk jumlah edge atau sisi sama dengan jumlah edge dari dataset. Hal ini menunjukkan bahwa data graph berarah yang divisualisasikan dengan visual treemap memiliki jumlah data node yang tampil berbeda dengan jumlah node dari dataset.

2. Visualisasi treemap dengan pindah layer jika salah satu node dipilih maka akan menghasilkan visualisasi treemap yang efektif karena dapat dibaca dan dipahami dengan mengetahui node yang dipilih terhubung ke node mana saja. Tetapi dari visualisasi treemap tersebut juga terdapat kekurangan yaitu user harus mencari node yang ingin dipilih atau mencari informasi yang ingin didapatkan dan hal ini membutuhkan waktu yang lebih untuk melakukan pencarian. Sehingga visual treemap tersebut tidak efisien dalam hal waktu untuk melihat informasi yang diinginkan.

6. Saran

Saran untuk pengembangan penelitian dari tugas akhir ini adalah sebagai berikut.

1. Untuk visualisasi graph statis dapat menggunakan representasi visual yang lain seperti *icicle plot*, *beamtree*, dan juga melakukan kombinasi visual.
2. Dapat juga melakukan penelitian terhadap visualisasi graph dinamis dengan representasi visual treemap, *icicle plot*, dan node link diagram.

Daftar Pustaka

[1] Bruls, M., Huizing, K., & Wijk, J. J. *Squarified Treemaps.* Eindhoven: Eindhoven University of Technology.

[2] Bunke, H., Dickinson, P. J., Kraetzl, M., & Wallis, W. D. (2007). *A Graph-Theoretic Approach to Enterprise Network Dynamics.* Boston, Basel, Berlin: Birkhauser.

[3] Deo, N. (1989). *Graph Theory with Applications to Engineering and Computer Science.* New Delhi: Prentice-Hall.

[4] Kocay, W., & Kreher, D. L. (2005). *Graphs, Algorithms, And Optimization.* Boca Raton: Chapman & Hall/CRC Press.

[5] Kusmayadi, H., & Darwiyanto, E. (2009). *XML dan WEB SERVICES.* Bandung: Politeknik Telkom.

[6] Landesberger, T., Kujiper, A., Schreck, T., Kohlhammer, J., Wijk, J. v., Fekete, J.-D., et al. (200x). Visual Analysis of Large Graphs. *The Eurographics Association* .

[7] O., A. D., & J., O. O. (2010). HierarchyMap: A Novel Approach to Treemap Visualization of Hierarchical Data. *Global Journal of Computer Science and Technology , 9* (5), 77.

[8] Padioleau, Y. (2010, September 24). Treemaps in OCaml.

[9] *Publications : High-Resolution Measurements of Face-to-Face Contact Patterns in a Primary School.* (n.d.). Retrieved November 1, 2014, from SocioPatterns: www.sociopatterns.org

[10] Sud, A., Fisher, D., & Lee, H.-P. (n.d.). Fast Dynamic Voronoi Treemaps. *Microsoft Research* , 2.

[11] Tu, Y., & Shen, H.-W. (2007). Visualizing Changes of Hierarchical Data using Treemaps. *IEEE Transactions on Visualization and Computer Graphics , 13* (6), 1286-1293.

[12] *Tutorial Point: JSON Tutorial.* (n.d.). Retrieved October 2014, from Simply Easy Learning: www.tutorialspoint.com

[13] Vliegen, R., Wijk, J. J., Member, IEEE, & Linden, E.-J. v. (2006). Visualizing Business Data with Generalized Treemaps. *IEEE Transactions on Visualization and Computer Graphics* , 789-796.

[14] *Wikipedia vote network*. (n.d.). Retrieved May 12, 2015, from Stanford Large Network Dataset Collection: https://snap.stanford.edu/data/wiki-Vote.html