**Implementasi VoG (Vocabulary based summarization of Graph) Pada Web Graph**

***Implementation of VoG (Vocabulary based summarization of Graph) on the Web Graph***

**Tugas Akhir**

**Kelompok Keahlian : SIDE**

**Satrio Adityo Hartomo**

**1103120029**

****

**Program Studi Teknik Informatika**

**Fakultas Informatika**

**Universitas Telkom**

**Bandung**

**2015**

# LEMBAR PERNYATAAN

Dengan ini saya menyatakan bahwa tugas akhir dengan judul **“Implementasi VoG (Vocabulary based summarization of Graph) pada Web Graph”** dan seluruh isinya benar-benar merupakan karya saya sendiri dan saya tidak melakukan penjiplakan atau pengutipan dengan cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan. Atas pernyataan ini, saya siap menanggung resiko/sanksi yang dijatuhkan kepada saya apabila kemudian ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya ini, atau ada klaim dari pihak lain terhadap karya keaslian saya ini.

|  |
| --- |
| Bandung, Desember 2015  Penulis, |
|  |
| Satrio Adityo Hartomo |

# LEMBAR PENGESAHAN

**Implementasi VoG (Vocabulary based summarization of Graph) pada Web Graph**

***Implementation of VoG (Vocabulary based summarization of Graph) on the Web Graph***

**Satrio Adityo Hartomo**

**1103120029**

Telah disetujui dan disahkan sebagai tugas akhir

Program Studi Teknik Informatika

Fakultas Informatika Universitas Telkom

Bandung, Desember 2015

Menyetujui

|  |  |  |
| --- | --- | --- |
| Pembimbing I |  | Pembimbing II |
|  |  |  |
|  |  |  |
| Kemas Rahmat Saleh W, S.T., M.Eng. NIP. 06830335-1 |  | Siti Sa'adah, S.T., M.T.  NIP. 13861143-1 |

# ABSTRAK

Pada tahun 2000 terdapat dua milyar web dan terus bertambah jumlahnya hingga empat milyar web pada tahun 2001. Web tersebut dapat direpresentasikan sebagai web graph yang mana node merepresentasikan url dan edge merepresentasikan link. Dalam tugas akhir ini dilakukan penelitian mengenai web graph. Dataset yang digunakan adalah data user yang diperoleh dari crawling web quora.com. Dataset yang sudah diperoleh direpresentasikan sebagai graph yang mana node merepresentasikan user dan edge merepresentasikan dua user menjawab pertanyaan dalam satu url yang sama. VoG (Vocabulary based summarization of Graph) adalah metode yang dapat meringkas graph yang besar. Dalam VoG dilakukan graph decomposition menggunakan algoritma Slashburn untuk mengurai graph yang besar menjadi sekumpulan subgraph. Subgraph yang terbentuk dipengaruhi oleh jumlah node maksimum dalam GCC. Semakin banyak jumlah node dalam GCC menghasilkan struktur subgraph clique dan star yang semakin sedikit namun semakin banyak menghasilkan struktur subgraph near chain. Dari struktur subgraph yang sudah berhasil diidentifikasi dapat dilakukan analisis menggunakan metode manual inspection untuk memperoleh informasi. Dalam tugas akhir ini diperoleh informasi dalam struktur clique yang menggambarkan bahwa terdapat sekelompok user yang aktif melakukan jajak pendapat dalam suatu pertanyaan. Untuk struktur star menggambarkan bahwa terdapat satu user yang aktif berkontribusi menjawab beberapa pertanyaan dalam web quora.

# ABSTRACT

In 2000 there were two billion web and the number of web increase up to four billion web in 2001. That web can be represented as a web graph which node represent url and edge represent link. In this final task research is conducted on web graph. The used dataset is user data that obtained from web crawling quora.com. The dataset that has been obtained is represented as a web graph which the node represent user and edge represent two users answered question in the same url. VoG (Vocabulary based summarization of Graph) is a method that can summarize a large graph. In VoG, graph decomposition is conducted using Slashburn algorithm for decompose large graph into a set of subgraph. Generated subgraph is affected by the maximum number of node in GCC. The increasing number of node in GCC generating decreasing number of clique and star subgraph structure but increasing number of near chain subgraph structure. Subgraph structure that have been identified can be analized using manual inspection method to obtain information. In this final task, the information obtained in clique subgraph structure represent that there is a group of active user polled in a question. For star subgraph structure represent that there is one user actively contribute to answer some question in web Quora.

# LEMBAR PERSEMBAHAN

Asdfasdf

Asdf

Asdf

Asdf

Asdf

# KATA PENGANTAR

Asdfasdf

Asdf

Asdf

asdf

# DAFTAR ISI

[LEMBAR PERNYATAAN ii](#_Toc438970541)

[LEMBAR PENGESAHAN iii](#_Toc438970542)

[ABSTRAK iv](#_Toc438970543)

[ABSTRACT v](#_Toc438970544)

[LEMBAR PERSEMBAHAN vi](#_Toc438970545)

[KATA PENGANTAR vii](#_Toc438970546)

[DAFTAR ISI viii](#_Toc438970547)

[DAFTAR GAMBAR x](#_Toc438970548)

[DAFTAR TABEL xii](#_Toc438970549)

[DAFTAR ISTILAH xiii](#_Toc438970550)

[1. PENDAHULUAN 1](#_Toc438970551)

[1.1 Latar Belakang 1](#_Toc438970552)

[1.2 Perumusan Masalah 1](#_Toc438970553)

[1.3 Tujuan 1](#_Toc438970554)

[1.4 Batasan Masalah 2](#_Toc438970555)

[1.5 Metodologi Penyelesaian Masalah 2](#_Toc438970556)

[1.6 Sistematika Penulisan 3](#_Toc438970557)

[2. DASAR TEORI 4](#_Toc438970558)

[2.1 Teori Graph 4](#_Toc438970559)

[2.2 Graph Database 5](#_Toc438970560)

[2.3 Web Graph 6](#_Toc438970561)

[2.4 Graph Compression 7](#_Toc438970562)

[2.4.1 Graph Shattering 7](#_Toc438970563)

[2.4.2 Slash and Burn 7](#_Toc438970564)

[2.5 VoG 8](#_Toc438970565)

[2.5.1 Perfect Clique 9](#_Toc438970566)

[2.5.2 Near Clique 9](#_Toc438970567)

[2.5.3 Perfect Bipartite 9](#_Toc438970568)

[2.5.4 Near Bipartite 10](#_Toc438970569)

[2.5.5 Perfect Star 10](#_Toc438970570)

[2.5.6 Near Star 10](#_Toc438970571)

[2.5.7 Perfect Chain 11](#_Toc438970572)

[2.5.8 Near Chain 11](#_Toc438970573)

[2.5.9 Algoritma VoG 11](#_Toc438970574)

[3. PERANCANGAN SISTEM 13](#_Toc438970575)

[3.1 Deskripsi Sistem 13](#_Toc438970576)

[3.2 Perancangan Sistem 16](#_Toc438970577)

[3.2.1 Web Crawling 16](#_Toc438970578)

[3.2.2 Representasi Graph dari Dataset 17](#_Toc438970579)

[3.2.3 Implementasi VoG 17](#_Toc438970580)

[4. PENGUJIAN DAN ANALISIS 20](#_Toc438970581)

[4.1 Tujuan Pengujian 20](#_Toc438970585)

[4.2 Skenario Pengujian 20](#_Toc438970586)

[4.2.1 Skenario pengujian algoritma graph decomposition (Slashburn) 20](#_Toc438970587)

[4.2.2 Skenario pengujian pengaruh jumlah node maksimum dalam GCC terhadap subgraph yang dihasilkan oleh Algoritma Slashburn 20](#_Toc438970588)

[4.2.3 Skenario pengujian pengaruh jumlah node maksimum dalam GCC terhadap waktu eksekusi VoG 21](#_Toc438970589)

[4.3 Analisis Hasil Pengujian 21](#_Toc438970590)

[4.3.1 Analisis hasil pengujian algoritma graph decomposition 21](#_Toc438970591)

[4.3.2 Analisis pengaruh jumlah node maksimum dalam GCC terhadap subgraph yang dihasilkan oleh algoritma graph decomposition (Slashburn) 26](#_Toc438970592)

[4.3.3 Analisis pengaruh jumlah node maksimum dalam GCC terhadap waktu eksekusi VoG 31](#_Toc438970593)

[4.3.4 Analisis informasi struktur subgraph 33](#_Toc438970594)

[5. KESIMPULAN DAN SARAN 34](#_Toc438970595)

[5.1 Kesimpulan 34](#_Toc438970596)

[5.2 Saran 34](#_Toc438970597)

[DAFTAR PUSTAKA 35](#_Toc438970598)

[LAMPIRAN 1 : HASIL PENGUJIAN SUBGRAPH YANG DIHASILKAN OLEH VoG 36](#_Toc438970599)

[LAMPIRAN 2 : HASIL PENGUJIAN WAKTU EKSEKUSI VoG 39](#_Toc438970600)

# DAFTAR GAMBAR

[Gambar 2‑1: Directed Graph 4](#_Toc438970601)

[Gambar 2‑2: Undirected Graph 4](#_Toc438970602)

[Gambar 2‑3: Simple Graph 4](#_Toc438970603)

[Gambar 2‑4: Multigraph 4](#_Toc438970604)

[Gambar 2‑5: Pseudograph 4](#_Toc438970605)

[Gambar 2‑6: Graph Berlabel 4](#_Toc438970606)

[Gambar 2‑7: Graph Tak Berlabel 4](#_Toc438970607)

[Gambar 2‑8: Graph G 5](#_Toc438970608)

[Gambar 2‑9: Graph H (subgraph G) 5](#_Toc438970609)

[Gambar 2‑10 Representasi graph - adjacency list [8] 5](#_Toc438970610)

[Gambar 2‑11 Model graph database 6](#_Toc438970611)

[Gambar 2‑12: Contoh Web Graph [11] 7](#_Toc438970612)

[Gambar 2‑13: Algoritma Slashburn 8](#_Toc438970613)

[Gambar 2‑14: (a) adalah graph G, (b) adalah graph G' setelah satu kali eksekusi slashburn [6] 8](#_Toc438970614)

[Gambar 2‑15: Contoh full-clique subgraph 9](#_Toc438970615)

[Gambar 2‑16: Beberapa contoh near-clique subgraph 9](#_Toc438970616)

[Gambar 2‑17: full-bipartite subgraph 10](#_Toc438970617)

[Gambar 2‑18: near-bipartite subgraph 10](#_Toc438970618)

[Gambar 2‑19: Star subgraph 10](#_Toc438970619)

[Gambar 2‑20: Near Star 11](#_Toc438970620)

[Gambar 2‑21: Chain subgraph 11](#_Toc438970621)

[Gambar 2‑22: Near Chain subgraph 11](#_Toc438970622)

[Gambar 2‑23: Pseudocode Algoritma VoG [6]. 12](#_Toc438970623)

[Gambar 3‑1:Flowchart sistem secara umum 13](#_Toc438970624)

[Gambar 3‑2: Flowchart proses slashburn untuk decomposing graph menjadi sekumpulan subgraph 14](#_Toc438970625)

[Gambar 3‑3: Flowchart subgraph labeling untuk mengidentifikasi struktur subgraph 15](#_Toc438970626)

[Gambar 3‑4: link question dari quora.com 16](#_Toc438970627)

[Gambar 3‑5: satu link question hasil crawl untuk dijadikan dataset 16](file:///E:\VoG\Buku%20Implementasi%20VoG.docx#_Toc438970628)

[Gambar 3‑6: Visualisasi graph P dari dataset 17](#_Toc438970629)

[Gambar 3‑7: Visualisasi subgraph dari Graph P setelah proses slashburn selesai 18](#_Toc438970630)

[Gambar 4‑1: Gambar dataset-1 yang digunakan untuk pengujian algoritma graph decomposition 21](#_Toc438970631)

[Gambar 4‑2: Gambar keluaran algoritma slashburn untuk graph decomposition 22](#_Toc438970632)

[Gambar 4‑3: Gambar keluaran proses SUBGRAPH LABELING 22](#_Toc438970633)

[Gambar 4‑4: Subgraph-subgraph dengan struktur perfect clique dari gambar 4-2 22](#_Toc438970634)

[Gambar 4‑5: Subgraph-subgraph dengan struktur perfect star dari gambar 4-2 23](#_Toc438970635)

[Gambar 4‑6: Subgraph-subgraph dengan struktur perfect bipartite dari gambar 4-2 23](#_Toc438970636)

[Gambar 4‑7: Subgraph-subgraph dengan struktur perfect chain dari gambar 4-2 23](#_Toc438970637)

[Gambar 4‑8: Gambar dataset-2 yang digunakan untuk pengujian algoritma graph decomposition 24](#_Toc438970638)

[Gambar 4‑9: Gambar keluaran algoritma slashburn untuk graph decomposition. Subgraph yang diberi tanda oval adalah subgraph non-perfect structure. 24](#_Toc438970639)

[Gambar 4‑10: Gambar keluaran proses SUBGRAPH LABELING, non-perfect structure teridentifikasi. 25](#_Toc438970640)

[Gambar 4‑11: Subgraph dengan struktur near clique dari gambar 4-9 25](#_Toc438970641)

[Gambar 4‑12: Subgraph dengan struktur near star dari gambar 4-9 25](#_Toc438970642)

[Gambar 4‑13: Subgraph dengan struktur near bipartite dari gambar 4-9 26](#_Toc438970643)

[Gambar 4‑14: Subgraph dengan struktur near chain dari gambar 4-9 26](#_Toc438970644)

[Gambar 4‑15: Dataset hasil crawling quora.com dengan jumlah node 2250 dan edge 5029 27](#_Toc438970645)

[Gambar 4‑16: Graph quora yang sudah dilakukan graph decomposition, terdapat 1742 node dan 1321 edge 27](#_Toc438970646)

[Gambar 4‑17: Zoom Graph dari gambar 4-8 28](#_Toc438970647)

[Gambar 4‑18: Hasil identifikasi struktur graph quora dengan jumlah node maksimum dalam GCC = 3 28](#_Toc438970648)

[Gambar 4‑19: Grafik hasil pengujian pola pertama 30](#_Toc438970649)

[Gambar 4‑20: Grafik hasil pengujian pola kedua 30](#_Toc438970650)

[Gambar 4‑21: Grafik hasil pengujian waktu eksekusi 33](#_Toc438970651)

# DAFTAR TABEL

[Tabel 4‑1: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 3 28](#_Toc438970652)

[Tabel 4‑10: Tabel summary hasil pengujian pola pertama 29](#_Toc438970653)

[Tabel 4‑11: Tabel summary hasil pengujian pola kedua 29](#_Toc438970654)

[Tabel 4‑12: Hasil pengujian waktu eksekusi VoG dengan jumlah node maksimum dalam GCC = 5 31](#_Toc438970655)

[Tabel 4‑13: Hasil pengujian waktu rata-rata eksekusi VoG 32](#_Toc438970656)

[Tabel 4‑2: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 5 36](#_Toc438970657)

[Tabel 4‑3: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 7 36](#_Toc438970658)

[Tabel 4‑4: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 9 36](#_Toc438970659)

[Tabel 4‑5: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 11 36](#_Toc438970660)

[Tabel 4‑3: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 5 37](#_Toc438970661)

[Tabel 4‑4: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 10 37](#_Toc438970662)

[Tabel 4‑8: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 15 37](#_Toc438970663)

[Tabel 4‑9: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 20 38](#_Toc438970664)

[Tabel 0‑1: Hasil pengujian waktu eksekusi VoG dengan jumlah node maksimum dalam GCC = 5 39](#_Toc438970665)

[Tabel 0‑2: Hasil pengujian waktu eksekusi VoG dengan jumlah node maksimum dalam GCC = 10 40](#_Toc438970666)

[Tabel 0‑3: Hasil pengujian waktu eksekusi VoG dengan jumlah node maksimum dalam GCC = 15 41](#_Toc438970667)

[Tabel 0‑4: Hasil pengujian waktu eksekusi VoG dengan jumlah node maksimum dalam GCC = 20 42](#_Toc438970668)

# DAFTAR ISTILAH

# PENDAHULUAN

## Latar Belakang

Dengan adanya teknik penyimpanan data yang baru yaitu *graph database* [1], cara untuk menyimpan data tidak lagi terbatas pada *relational database* yang struktural. Data yang mempunyai perbedaan atribut (semi terstruktur) dapat disimpan ke *database* dengan representasi *graph* [1], seperti Facebook dan Twitter yang juga menggunakan *graph database* untuk menyimpan data-datanya [2] [3]. *Website* yang lain pun memiliki data yang unik, berjumlah banyak dan semakin bertambah hingga saat ini [4]. *Website* yang dulunya hanya menampilkan data statis kini telah berevolusi menjadi *website* yang dapat menampilkan data secara dinamis dan dapat direpresentasikan sebagai *graph* (*web graph*) [1].

Pada referensi [4] disebutkan bahwa pada Juli 2000 terdapat dua milyar *web*, dan terus meningkat jumlahnya hingga empat milyar *web* pada tahun 2001. *Web graph* yang besar dengan ribuan *node* dan *edge* [5], akan sulit untuk diidentifikasi bagaimana bentuk strukturnya, apakah *random* atau tidak. Dengan ukuran yang besar tersebut akan sulit juga untuk memperoleh informasi. Oleh karena itu diperlukan metode untuk meringkas *web graph* agar ukurannya menjadi lebih kecil dari *web graph* aslinya, sehingga dapat dilakukan pengidentifikasian struktur dan analisis untuk mendapatkan informasi pada hasil ringkasan *web graph*.

VoG (Vocabulary based summarization of Graph) adalah metode untuk meringkas graph yang besar secara efisien [6]. Dalam metode VoG akan dilakukan graph decomposition menggunakan algoritma Slashburn. Algoritma Slashburn akan menghasilkan sekumpulan subgraph yang mana subgraph yang dihasilkan akan dilakukan proses identifikasi struktur. Dalam tugas akhir ini akan dilakukan implementasi web graph *summarization* menggunakan metode VoG untuk meringkas web graph menjadi subgraph-subgraph yang lebih kecil ukuran atau jumlah nodenya dan mendapatkan informasi dari subgraph-subgraph yang dihasilkan.

## Perumusan Masalah

Dari latar belakang yang telah diuraikan, permasalahan yang akan dibahas dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana mengimplementasikan VoG untuk *summarize* web graph?
2. Bagaimana pengaruh jumlah *node* maksimum dalam *Giant Connected Component (GCC)* terhadap struktur yang dihasilkan oleh VoG?
3. Bagaimana pengaruh jumlah node maksimum dalam GCC terhadap waktu eksekusi VoG?
4. Bagaimana menganalisis struktur subgraph yang dihasilkan oleh VoG untuk memperoleh minimum satu kalimat informasi?

## Tujuan

Adapun tujuan dalam tugas akhir ini adalah sebagai berikut:

1. Mengimplementasikan metode VoG untuk men-*summarize* web graph.
2. Menganalisis pengaruh jumlah node maksimum dalam *Giant Connected Component (GCC)* terhadap struktur yang dihasilkan oleh VoG.
3. Menganalisis pengaruh jumlah node maksimum dalam GCC terhadap waktu eksekusi VoG.
4. Menganalisis struktur subgraph yang dihasilkan oleh VoG untuk memperoleh minimum satu kalimat informasi.

## Batasan Masalah

Mengenai batasan permasalahan dalam tugas akhir ini adalah sebagai berikut:

1. Struktur yang dapat diidentifikasi oleh metode VoG dalam tugas akhir ini terdiri dari *Perfect Clique, Perfect Star, Perfect Bipartite, Perfect Chain, Near Clique, Near Star, Near Bipartite,* dan *Near Chain.*
2. Metode VoG hanya dapat melakukan identifikasi struktur setelah memiliki keluaran dari algoritma *graph decomposition* yang dalam tugas akhir ini adalah algoritma Slashburn.
3. Dataset yang digunakan didapat dari hasil crawling web menggunakan aplikasi buatan sendiri.
4. Dua *node* yang terhubung hanya memiliki satu *edge*.
5. Untuk pengolahan graph menggunakan bantuan library GraphStream.
6. Untuk menganalisis informasi dalam subgraph digunakan metode manual inspection.

## Metodologi Penyelesaian Masalah

Metode penyelesaian masalah yang diterapkan dalam tugas akhir ini adalah sebagai berikut:

1. Identifikasi permasalahan

Pada tahap ini dilakukan studi literatur dan diskusi untuk mengidentifikasi permasalahan yang akan dibahas pada tugas akhir ini. Identifikasi meliputi observasi terkait fenomena yang terjadi di bidang graph database, metode yang digunakan untuk solusi permasalahan, dan batasan masalah yang ada.

1. Studi literatur

Pada tahap ini dilakukan pengumpulan dan pemahaman terkait materi-materi yang dibutuhkan untuk menyelesaikan permasalahan yang tertera di perumusan masalah. Referensi yang dicari antara lain materi mengenai metode VoG, metode *graph decomposition* untuk menghasilkan graph summarization.

1. Perancangan dan pembuatan sistem

Pada tahap ini dilakukan pemodelan dataset ke dalam bentuk graph. Kemudian membuat rancangan sistem untuk memberikan solusi dari permasalahan. Dari rancangan sistem yang sudah dibuat dilakukan pembuatan sistem dengan menggunakan bahasa pemrograman Java.

1. Pengujian dan analisis

Pada tahap ini dilakukan pengujian terhadap dataset yang sudah dimodelkan dalam bentuk graph. Pengujian dilakukan sesuai dengan skenario yang ada pada bagian strategi pengujian. Kemudian dilakukan analisis terhadap subgraph-subgraph hasil ringkasan graph.

1. Penyusunan laporan

Pada tahap ini dilakukan dokumentasi dan pelaporan hasil sesuai dengan aturan dan sistematika penulisan yang ditetapkan institusi.

## Sistematika Penulisan

Sistematika penulisan buku tugas akhir ini adalah sebagai berikut:

* 1. Bab I Pendahuluan

Bab ini berisi latar belakang, perumusan masalah, tujuan, batasan masalah, metodologi penyelesaian masalah dan sistematika penulisan buku tugas akhir.

* 1. Bab II Landasan Teori

Bab ini berisi landasan teori yang menjadi dasar dilakukannya penelitian pada tugas akhir ini.

* 1. Bab III Perancangan Sistem

Bab ini berisi perancangan sistem yang akan diimplementasikan pada tugas akhir ini.

* 1. Bab IV Pengujian dan Analisis

Bab ini berisi pengujian dan analisis hasil dari sistem yang telah dibuat.

* 1. Bab V Kesimpulan dan Saran

Bab ini berisi kesimpulan yang didapat dari permasalahan, pengujian dan analisis hasil, serta saran untuk penelitian selanjutnya.

# DASAR TEORI

## 2.1 Teori Graph

*Graph*, secara definisi adalah sekumpulan *nodes* () dan *edges* (). Sekumpulan *nodes* bisa saja tidak terbatas atau bisa disebut ***infinite graph***, sedangkan yang sekumpulan *nodes*nya terbatas disebut ***finite graph***. Dalam tugas akhir ini hanya akan fokus pada *finite graph*. *Graph* dapat dinotasikan sebagai . adalah sekumpulan *nodes* dan adalah sekumpulan *edges* yang menghubungkan *nodes* [7].

|  |  |  |  |
| --- | --- | --- | --- |
| Gambar 2‑1: Directed Graph | | Gambar 2‑2: Undirected Graph | |
| Gambar 2‑3: Simple Graph | Gambar 2‑4: Multigraph | | Gambar 2‑5: Pseudograph |
| Gambar 2‑6: Graph Berlabel | | Gambar 2‑7: Graph Tak Berlabel | |

Perbedaan jenis *graph* dapat dilihat berdasarkan ada atau tidaknya arah *edge*, *graph* jenis ini terdiri dari ***directed graph*** (*graph* yang mempunyai *edge* berarah) dan ***undirected graph* (***graph* yang mempunyai *edge* tidak berarah). Berdasarkan *edge* yang menghubungkan *nodes* terdiri dari ***simple graph*** (tidak memiliki *loop* dan *multiple edge*), ***multigraph*** (tidak memiliki *loop edge* tetapi memiliki *multiple edge*) dan ***pseudograph*** (memiliki *loop edge*). Berdasarkan ada atau tidaknya label terdiri dari ***graph* berlabel** dan ***graph* tak berlabel**. Contoh jenis-jenis *graph* dapat dilihat pada gambar 2-1 sampai 2-7 di bawah ini [7].

Sekumpulan *nodes* dan *edges* yang ada di *graph* dapat digunakan untuk memodelkan permasalahan yang ada pada dunia nyata, sebagai contoh *graph* jaringan komputer. Namun, dalam kondisi tertentu untuk menyelesaikan permasalahan yang ada, tidak seluruh *nodes* digunakan. Sehingga yang digunakan untuk menyelesaikan masalah adalah bagian kecil dari *graph*, atau bisa disebut dengan *subgraph*. Secara definisi, *subgraph* dari *graph* adalah *graph* dimana dan dan [7].

|  |  |
| --- | --- |
| Gambar 2‑8: Graph G | Gambar 2‑9: Graph H (subgraph G) |

Cara merepresentasikan *graph* ada beberapa cara, diantaranya yaitu menggunakan *Adjacency Matrix* dan *Adjacency List*. Berikut contoh cara merepresentasikan graph [7]:

Dari gambar 2-8 dapat direpresentasikan *adjacency list* seperti gambar 2-10 di bawah ini :



Gambar 2‑10 Representasi graph - adjacency list [8]

Atau sebagai *adjacency matrix* seperti di bawah ini :

Angka 1 merepresentasikan adanya *edge* antar*nodes* [7].

## 2.2 Graph Database

*Graph Database Management System* adalah *database management system* yang memiliki metode CRUD (*Create*, *Read*, *Update*, dan *Delete*) untuk memaparkan model data *graph*. Properti model *graph* dapat dideskripsikan sebagai berikut :

* Terdiri dari *nodes* dan *relationship/edges*.
* Setiap *nodes* terdiri dari pasangan *key-value*.
* *Edge* mempunyai nama dan arah, jika tidak memiliki arah disebut *undirected graph.*
* *Edge* dapat juga mempunyai properti.

Dengan abstraksi sederhana dari *nodes* dan *edges* kedalam struktur yang saling terhubung, *graph database* memungkinkan untuk dibangunnya model yang mirip dengan permasalahan yang ada di dunia nyata [1].



Gambar 2‑11 Model graph database

## 2.3 Web Graph

*Webpages* dan *hyperlinks* di *World Wide Web* dapat direpresentasikan sebagai *nodes* dan *edge* di *directed graph*. Saat ini, *graph* tersebut memiliki milyaran *nodes* dan *edge* dan bertambah banyak seiring berjalannya waktu. Ada beberapa alasan untuk mempelajari *web graph*, diantaranya adalah bagaimana algoritma yang bekerja pada *web graph*, bagaimana *web search*nya, dan bagaimana pengklasifikasiannya [9].

Tidak hanya itu, ada alasan lain untuk mengembangkan pemahaman dari *web graph* ini, diantaranya dapat :

1. Mendesain strategi *crawling* pada web.
2. Memahami sosiologi pembuatan konten pada web.
3. Menganalisis *behavior* dari algoritma web.
4. Memprediksi evolusi struktur web.
5. Memprediksi munculnya fenomena baru di web [10].

Berikut contoh *web graph*:



Gambar 2‑12: Contoh Web Graph [11]

## 2.4 Graph Compression

Jika dalam suatu *graph* dapat ditemukan *good community*, maka *graph* tersebut dapat dikompres yang mana dapat membantu untuk menyusutkan ukuran data [12]. Rincian proses *graph compression* dalam tugas akhir ini dijelaskan pada bagian 2.4.1 dan 2.4.2.

### Graph Shattering

Untuk menyelesaikan permasalahan dalam *graph compression*, diperlukan proses untuk mengurai *graph* dengan cara mengeksploitasi *hubs* (*node* dengan degree paling tinggi). Dengan hasil eksploitasi *hubs* didapatkan *alternative community* yang berbeda dari *clique community* [12].

Dari referensi [12] disebutkan bahwa berdasarkan observasi, *real world graph* dapat dengan mudah dipecah dengan menghapus *hub node* dari *graph*. Dengan penghapusan *hub node*, *graph* terurai menjadi banyak *subgraph-subgraph*.

### 2.4.2 Slash and Burn

Baris 1 dan 2 dari algoritma 1 menghapus *nodes* dengan *degree* paling tinggi dan *edge* yang berasosiasi, dengan demikian *graph* terurai dalam tiga kelompok:

* -hubset: *nodes* dengan *degree* paling tinggi.
* GCC: *node* yang dimiliki oleh *Giant Conncected Component* dari .
* Spokes: *node* yang dimiliki oleh *non-giant connected component* dari [12].



Gambar 2‑13: Algoritma Slashburn



Gambar 2‑14: (a) adalah graph G, (b) adalah graph G' setelah satu kali eksekusi slashburn [6]

## 2.5 VoG

*Vocabulary based summarization of Graph* (VoG) merupakan metode yang efektif dan efisien untuk meringkas graph yang besar. *Vocabulary* (Ω) adalah sekumpulan struktur *graph* yang terdiri dari struktur *full-clique, near-clique, full-bipartite, near-bipartite, perfect-star, near-star, perfect-chains* dan *near-chains*. Masih ada struktur *graph* yang lain, akan tetapi dari referensi [6] disebutkan bahwa struktur *graph* yang sering muncul dalam *real world* *graph* adalah struktur yang ada pada *Vocabulary* tersebut. Oleh karena itu, dalam tugas akhir ini dibangun sistem yang dapat mengidentifikasi struktur dalam *vocabulary*. Penjelasan mengenai struktur *graph* dapat dilihat di bagian 2.5.1 - 2.5.4 [6].

*Graph* yang ada di dunia sangatlah banyak dan besar, orang-orang sulit untuk memahami dengan mudah *graph* yang besar dan mungkin berantakan strukturnya. Dengan adanya VoG, *graph* dapat diringkas dan orang-orang dapat lebih mudah memahaminya [6].

### 2.5.1 Perfect Clique

*Perfect Clique* adalah *complete subgraph* dari suatu *graph* atau semua *node* didalam *subgraph* saling terhubung dengan *node* yang lain di dalam *subgraph* yang sama. Berikut contoh struktur *perfect clique* *graph* seperti gambar 2-15 [13]:



Gambar 2‑15: Contoh full-clique subgraph

### 2.5.2 Near Clique

*Near Clique* adalah *subgraph* yang hampir *complete* dari suatu *graph*. Misalnya diberikan *graph* seperti gambar 2-15. *Near clique* yang terbentuk adalah [6]:



Gambar 2‑16: Beberapa contoh near-clique subgraph

### 2.5.3 Perfect Bipartite

*Perfect bipartite* adalah sekumpulan *nodes* dan yang mana semua *node* di terhubung ke semua *node* di , namun tidak ada *edge* dalam kumpulan *nodes* atau . Contoh *full bipartite* [6]:



Gambar 2‑17: full-bipartite subgraph

### 2.5.4 Near Bipartite

*Near bipartite* adalah *nodes* dan yang mana **hampir** semua *node* di terhubung ke semua *node* di , **ada** *node* di yang tidak terhubung ke salah satu *node* di , namun tidak ada *edge* dalam kumpulan *nodes* atau . Contoh *near bipartite* [6]:



Gambar 2‑18: near-bipartite subgraph

### 2.5.5 Perfect Star

*Star* adalah kasus spesifik dari *bipartite graph*, yang mana hanya ada *single node* terhubung ke semua *node* yang mana paling tidak terdiri minimal 2 *nodes* [6].



Gambar 2‑19: Star subgraph

### 2.5.6 Near Star

*Near Star* adalah subgraph yang mempunyai struktur nyaris perfect star [6]. Berikut contoh near star dari star pada gambar 2-19:



Gambar 2‑20: Near Star

### 2.5.7 Perfect Chain

*Perfect Chain* adalah *graph* yang setiap *node*nya memiliki edge ke *node* selanjutnya [6].



Gambar 2‑21: Chain subgraph

### 2.5.8 Near Chain

*Near Chain* adalah *subgraph* yang mempunyai struktur hampir perfect chain [6].



Gambar 2‑22: Near Chain subgraph

### 2.5.9 Algoritma VoG

Pseudocode dari VoG adalah sebagai berikut :



Gambar 2‑23: Pseudocode Algoritma VoG [6].

# PERANCANGAN SISTEM

## Deskripsi Sistem

Sistem yang dibangun dalam tugas akhir ini adalah sistem yang dapat meringkas *graph* yang besar menjadi sekumpulan *subgraph* yang berukuran lebih kecil. Dari masing-masing *subgraph* akan diidentifikasi struktur dan dianalisis informasinya. Dataset yang digunakan dalam pengujian sistem adalah dataset yang didapat melalui *crawling* *web* quora.com. Data yang di-*crawl* adalah data *username* dalam suatu *link question*. Dari data yang sudah diperoleh kemudian direpresentasikan sebagai *graph*, yang mana *node* merepresentasikan *username*, dan *edge* merepresentasikan dua *username* menjawab pertanyaan di *link* yang sama. Berikut ini adalah *flow chart* dari sistem yang dibangun:



Gambar 3‑1:Flowchart sistem secara umum

Gambar 3-1 adalah gambaran secara umum proses-proses yang ada dalam sistem yang dibangun. Proses yang lebih rinci mengenai proses *slashburn* dan *subgraph labeling*, dapat dilihat dalam gambar 3-2 dan 3-3 di bawah ini:



Gambar 3‑2: Flowchart proses slashburn untuk decomposing graph menjadi sekumpulan subgraph



Gambar 3‑3: Flowchart subgraph labeling untuk mengidentifikasi struktur subgraph

## Perancangan Sistem

Alur dari sistem yang dibangun dalam tugas akhir ini terdiri dari tiga tahap:

* 1. *Crawling* dataset dari *web* quora.com, dataset disimpan dalam bentuk file dengan format .txt.
  2. Representasi dataset ke bentuk *graph* dengan bantuan *library* GraphStream Java.
  3. Penerapan VoG, yaitu *graph compression* untuk menghasilkan *subgraph-subgraph* dari *graph input*, lalu *subgraph labeling* untuk mengidentifikasi struktur masing-masing *subgraph*.

### Web Crawling

Untuk mendapatkan dataset yang digunakan dalam tugas akhir ini, dilakukan *web crawling* pada web quora.com. Data yang diambil adalah *username* dan *link question*. Berikut ini contoh *web* yang di-*crawl*:



Gambar 3‑4: link question dari quora.com

Setelah *web crawler* selesai melakukan *crawling*, akan diperoleh dataset. Berikut ini adalah contoh dataset yang didapatkan:



Gambar 3‑5: satu link question hasil crawl untuk dijadikan dataset

### Representasi Graph dari Dataset

Dari dataset yang sudah didapatkan, data setiap baris akan dipecah untuk mendapatkan *url*, dan *username*. Dari *url* dan *username* tersebut akan dibuat representasi *graph* menggunakan bantuan *library* GraphStream Java. Berikut ini adalah contoh visualisasi *graph* di aplikasi:



Gambar 3‑6: Visualisasi graph P dari dataset

### Implementasi VoG

Untuk proses implementasi VoG, harus ditentukan terlebih dahulu berapa jumlah *node* maksimal dalam GCC (). Jumlah *node* maksimal ini akan menjadi batas untuk iterasi proses *graph compression* menggunakan algoritma Slashburn.

#### 3.2.3.1 Proses Graph Compression menggunakan Algoritma Slashburn

Algoritma Slashburn akan berhenti dieksekusi jika . Ketika proses slashburn berjalan, *graph* diproses secara iteratif dengan batas . Graph akan dicari *hub node* di dalam GCC untuk dihapus. Penghapusan *hub node* ini disertai dengan *edge*-nya dan menghasilkan *subgraph*. Jika proses penghapusan *node* ini menghasilkan satelit *node* yang *isolated*, maka *hub node* akan dibuat kembali dan satelit *node* akan dihubungkan dengan *hub node*. Hal ini dilakukan agar tidak terbentuk *subgraph* dengan satu *node*. Setelah proses *graph compression* selesai, akan dihasilkan *subgraph-subgraph*. Berikut ini adalah *subgraph* yang terbentuk dari pada gambar 3-6 dengan :



Gambar 3‑7: Visualisasi subgraph dari Graph P setelah proses slashburn selesai

#### 3.2.3.2 Proses Subgraph Labeling

Setelah proses *graph compression* menggunakan algoritma slashburn selesai, didapatkan *subgraph*. Dari *subgraph-subgraph* tersebut akan diidentifikasi strukturnya. Untuk proses identifikasi struktur, yang pertama dilakukan adalah mendapatkan jumlah *node* dalam *subgraph* dan *adjacency matrix*-nya. Setelah jumlah *node* didapat, maka dibuatlah empat model yaitu *clique, star, chain,* dan *bipartite* model. Model-model inilah yang akan digunakan untuk proses mendapatkan *error matrix*. Berikut ini adalah contoh proses untuk mendapatkan *error matrix*:

(1)

Contoh operasi matematika (1) diatas adalah operasi *adjacency matrix subgraph* XOR dengan *matrix* model clique, yang menghasilkan *error matrix* untuk model *clique*.

(2)

Contoh operasi matematika (2) diatas adalah operasi *adjacency matrix subgraph* XOR dengan *matrix* model star, yang menghasilkan *error matrix* untuk model *star*.

Dari dua contoh *error matrix* diatas, *error matrix* kemudian di-*encode* menjadi *error score* untuk masing-masing model. Dalam kasus di atas, *error score* untuk *error matrix* model *clique* = 2. Sedangkan *error score* untuk *error matrix star* adalah 0. Setelah didapatkan *error score* untuk masing-masing model, akan dipilih *error score* yang paling kecil nilainya, sehingga *subgraph* akan diidentifikasi sebagai **Subgraph dengan Struktur Star**.

# PENGUJIAN DAN ANALISIS

Sistem dalam tugas akhir ini dibuat menggunakan bahasa pemrograman Java dengan bantuan library GraphStream untuk mengelola graph. Dataset yang digunakan dapat berbentuk file .txt yang dalam hal ini adalah dataset hasil crawling web quora.com. Dapat juga berbentuk file .dgs hasil penyimpanan graph yang mana file .dgs tersebut dapat dimuat kembali ke sistem.



## 4.1 Tujuan Pengujian

Dari pengujian yang dilakukan, tujuan yang ingin dicapai adalah sebagai berikut:

1. Menganalisis subgraph yang dihasilkan oleh algoritma graph decomposition apakah subgraph yang dihasilkan sesuai dengan vocabulary dalam VoG.
2. Menganalisis pengaruh jumlah nodes maksimum dalam GCC terhadap subgraph yang dihasilkan.
3. Menganalisis pengaruh jumlah nodes maksimum dalam GCC terhadap waktu eksekusi VoG.
4. Menganalisis informasi yang terkandung dalam struktur subgraph.

## 4.2 Skenario Pengujian

Dalam pengujian sistem ini, terdapat tiga skenario pengujian. Skenario yang pertama adalah pengujian algoritma graph decomposition (Slashburn) untuk menghasilkan subgraph yang ada pada vocabulary. Skenario kedua adalah pengujian pengaruh jumlah node maksimum dalam GCC terhadap subgraph yang dihasilkan oleh algoritma graph decomposition (Slashburn). Skenario ketiga adalah pengujian pengaruh jumlah node maksimum dalam GCC terhadap waktu eksekusi VoG.

### 4.2.1 Skenario pengujian algoritma graph decomposition (Slashburn)

Pada skenario pengujian ini, akan dibuktikan apakah algoritma untuk mengurai graph menjadi subgraph sudah sesuai dengan vocabulary yang ada pada VoG. Untuk membuktikannya, dibuatlah dataset yang memiliki seluruh struktur yang ada pada vocabulary. Kemudian dataset tersebut direpresentasikan sebagai web graph dalam sistem. Graph yang terbentuk kemudian dilakukan graph decomposition menggunakan algoritma Slashburn. Setelah proses graph decomposition selesai, subgraph yang dihasilkan akan diidentifikasi strukturnya.

### 4.2.2 Skenario pengujian pengaruh jumlah node maksimum dalam GCC terhadap subgraph yang dihasilkan oleh Algoritma Slashburn

Pada skenario pengujian ini, akan dilakukan pengujian terhadap dataset hasil crawling web quora.com. Pengujian dilakukan delapan kali dengan jumlah node maksimum dalam GCC yang diubah-ubah. Karena setiap kali jumlah node maksimum dalam GCC diubah akan menghasilkan sekumpulan subgraph yang berbeda. Oleh karena itu, subgraph yang dihasilkan oleh algoritma Slashburn dapat dianalisis polanya. Dalam pengujian ini akan dilakukan dengan dua pola. Pola pertama yaitu dengan jumlah node maksimum dalam GCC sama dengan 5, 7, 9, dan 11. Pola yang kedua yaitu dengan jumlah node maksimum dalam GCC sama dengan 5, 10, 15, dan 20.

### 4.2.3 Skenario pengujian pengaruh jumlah node maksimum dalam GCC terhadap waktu eksekusi VoG

Pada skenario pengujian ini, dilakukan pengujian terhadap dataset untuk mengukur waktu eksekusi VoG. Karena tujuannya adalah menganalisis pengaruh jumlah node maksimum dalam GCC terhadap waktu eksekusi VoG, maka pengujian dilakukan dengan cara mengubah-ubah jumlah node maksimum dalam GCC. Kemudian dilakukan pengukuran terhadap waktu eksekusi VoG. Pengukuran waktu eksekusi VoG akan dilakukan sebanyak 30 kali untuk setiap jumlah node maksimum dalam GCC. Hal ini ditujukan untuk mendapatkan waktu rata-rata pengeksekusian VoG. Kemudian dari waktu eksekusi yang telah didapatkan akan dilakukan analisis pengaruh jumlah node dalam GCC terhadap waktu eksekusi VoG.

## 4.3 Analisis Hasil Pengujian

Dari skenario pengujian yang telah diuraikan pada bagian sebelumnya, dalam bagian ini akan diuraikan analisis hasil pengujian.

### 4.3.1 Analisis hasil pengujian algoritma graph decomposition

Pengujian graph decomposition menggunakan sample dataset yang dibuat sendiri. Terdapat dua dataset yang digunakan. Berikut adalah tampilan graph dari dataset pertama yang digunakan untuk menguji algoritma graph decomposition:



Gambar 4‑1: Gambar dataset-1 yang digunakan untuk pengujian algoritma graph decomposition

Setelah graph pada gambar 4-1 dilakukan proses graph decomposition menggunakan algoritma Slashburn, didapatlah keluaran sekumpulan subgraph. Berikut adalah gambar keluaran algoritma Slashburn:



Gambar 4‑2: Gambar keluaran algoritma slashburn untuk graph decomposition

Setelah didapatkan sekumpulan subgraph hasil algoritma Slashburn, akan diidentifikasi masing-masing struktur subgraphnya. Untuk proses pengidentifikasian struktur, dilakukan proses SUBGRAPH LABELING seperti yang tertera pada Perancangan Sistem. Berikut ini adalah gambar keluaran proses subgraph labeling:



Gambar 4‑3: Gambar keluaran proses SUBGRAPH LABELING

Dari gambar 4-3 diperoleh subgraph dengan struktur perfect clique berjumlah 3. Subgraph yang mempunyai struktur perfect clique dalam gambar 4-2 diantaranya adalah:



Gambar 4‑4: Subgraph-subgraph dengan struktur perfect clique dari gambar 4-2

Dari gambar 4-3 diperoleh subgraph dengan struktur perfect star berjumlah 4, diantaranya adalah:



Gambar 4‑5: Subgraph-subgraph dengan struktur perfect star dari gambar 4-2

Dari gambar 4-3 diperoleh subgraph dengan struktur perfect bipartite berjumlah 3, diantaranya adalah:



Gambar 4‑6: Subgraph-subgraph dengan struktur perfect bipartite dari gambar 4-2

Dari gambar 4-3 diperoleh subgraph dengan struktur perfect chain berjumlah 2, diantaranya adalah:



Gambar 4‑7: Subgraph-subgraph dengan struktur perfect chain dari gambar 4-2

Berikut adalah gambar dataset kedua yang digunakan:



Gambar 4‑8: Gambar dataset-2 yang digunakan untuk pengujian algoritma graph decomposition

Setelah graph dataset-2 pada gambar 4-8 dilakukan proses graph decomposition menggunakan algoritma Slashburn, didapatlah keluaran sekumpulan subgraph. Berikut adalah gambar keluaran algoritma Slashburn:



Gambar 4‑9: Gambar keluaran algoritma slashburn untuk graph decomposition. Subgraph yang diberi tanda oval adalah subgraph non-perfect structure.

Setelah didapatkan sekumpulan subgraph hasil algoritma Slashburn di gambar 4-9, akan diidentifikasi masing-masing strukturnya. Untuk proses pengidentifikasian struktur, dilakukan proses SUBGRAPH LABELING seperti yang tertera pada Perancangan Sistem. Berikut ini adalah gambar keluaran proses subgraph labeling:



Gambar 4‑10: Gambar keluaran proses SUBGRAPH LABELING, non-perfect structure teridentifikasi.

Dari gambar 4-10 diperoleh struktur subgraph near clique berjumlah 1. Struktur tersebut dari gambar 4-9 adalah:



Gambar 4‑11: Subgraph dengan struktur near clique dari gambar 4-9

Dari gambar 4-10 diperoleh subgraph dengan struktur near star berjumlah 1, struktur tersebut dari gambar 4-9 adalah:



Gambar 4‑12: Subgraph dengan struktur near star dari gambar 4-9

Dari gambar 4-10 diperoleh subgraph dengan struktur near bipartite berjumlah 2, struktur tersebut diantaranya adalah:



Gambar 4‑13: Subgraph dengan struktur near bipartite dari gambar 4-9

Dari gambar 4-10 diperoleh subgraph dengan struktur near chain berjumlah 1, struktur tersebut adalah:



Gambar 4‑14: Subgraph dengan struktur near chain dari gambar 4-9

Dari gambar 4-2, algoritma Slashburn berhasil mengurai graph pada gambar 4-1 menjadi sekumpulan subgraph yang mana masing-masing subgraph berhasil diidentifikasi strukturnya. Struktur yang dapat teridentifikasi adalah perfect clique, perfect bipartite, perfect star, dan perfect clique.

Dari gambar 4-8, dataset kedua yang digunakan berhasil diurai oleh algoritma Slashburn menghasilkan sekumpulan subgraph seperti pada gambar 4-9. Terdapat beberapa subgraph yang kurang untuk menjadi perfect structure. Karena struktur subgraph tidak sempurna, maka akan diidentifikasi sebagai near structure. VoG dapat mengidentifikasi struktur near clique, near star, near chain, dan near bipartite seperti tertera pada gambar 4-10.

Dari kedua sample dataset yang digunakan, graph dapat diidentifikasi strukturnya sesuai dengan vocabulary dalam VoG.

### 4.3.2 Analisis pengaruh jumlah node maksimum dalam GCC terhadap subgraph yang dihasilkan oleh algoritma graph decomposition (Slashburn)

Pada pengujian VoG ini digunakan dataset yang didapat dari hasil crawl web quora.com. Hasil VoG akan ditampilkan dalam bentuk tabel. Berikut ini adalah gambar dataset dari quora.com yang digunakan:



Gambar 4‑15: Dataset hasil crawling quora.com dengan jumlah node 2250 dan edge 5029

Dari graph seperti pada gambar 4-15 yang sudah dimuat kedalam sistem, akan dilakukan graph decomposition untuk menghasilkan subgraph dengan variable jumlah node dalam GCC yang diubah-ubah. Berikut ini adalah gambar graph setelah graph decomposition dengan jumlah node maksimum dalam GCC sama dengan 3:



Gambar 4‑16: Graph quora yang sudah dilakukan graph decomposition, terdapat 1742 node dan 1321 edge

Untuk lebih jelas, dapat dilakukan zoom di sistem. Berikut ini adalah graph gambar 4-16 yang di-zoom:



Gambar 4‑17: Zoom Graph dari gambar 4-8

Dari hasil graph docomposition, masing-masing subgraph akan diidentifikasi strukturnya sesuai dengan vocabulary. Berikut ini adalah hasil identifikasi struktur graph gambar 4-16:



Gambar 4‑18: Hasil identifikasi struktur graph quora dengan jumlah node maksimum dalam GCC = 3

Untuk selanjutnya, hasil identifikasi struktur akan dibuat dalam bentuk tabel seperti tabel dibawah ini:

Tabel 4‑1: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 3

|  |  |
| --- | --- |
| Struktur | Jumlah |
| Perfect Clique | 349 |
| Perfect Star | 15 |
| Perfect Bipartite | 0 |
| Perfect Chain | 0 |
| Near Clique | 0 |
| Near Star | 0 |
| Near Bipartite | 0 |
| Near Chain | 0 |

Dari graph quora seperti pada gambar 4-15 dilakukan 2 pola graph decomposition dengan jumlah node maksimum dalam GCC sama dengan 5, 7, dan 9 untuk pola pertama. Untuk pola kedua akan dilakukan graph decomposition dengan node maksimum dalam GCC sama dengan 5, 10, 15 dan 20. Berikut ini adalah hasil pengujian untuk pola pertama:

Tabel 4‑10: Tabel summary hasil pengujian pola pertama

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Struktur | Jumlah node maksimum dalam GCC | | | |
| **5** | **7** | **9** | **11** |
| Perfect Clique | 337 | 293 | 273 | 253 |
| Perfect Star | 13 | 9 | 5 | 5 |
| Perfect Bipartite | 0 | 0 | 0 | 0 |
| Perfect Chain | 2 | 2 | 1 | 1 |
| Near Clique | 12 | 21 | 21 | 18 |
| Near Star | 21 | 47 | 47 | 50 |
| Near Bipartite | 0 | 0 | 0 | 0 |
| Near Chain | 2 | 9 | 19 | 27 |

Berikut ini adalah hasil pengujian untuk pola kedua:

Tabel 4‑11: Tabel summary hasil pengujian pola kedua

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Struktur | Jumlah node maksimum dalam GCC | | | |
| **5** | **10** | **15** | **20** |
| Perfect Clique | 337 | 264 | 230 | 225 |
| Perfect Star | 13 | 5 | 5 | 5 |
| Perfect Bipartite | 0 | 0 | 0 | 0 |
| Perfect Chain | 2 | 1 | 1 | 1 |
| Near Clique | 12 | 20 | 11 | 10 |
| Near Star | 21 | 49 | 38 | 34 |
| Near Bipartite | 0 | 0 | 0 | 0 |
| Near Chain | 2 | 22 | 43 | 43 |

Untuk hasil pengujian yang lebih rinci terlampir pada lampiran 1 dalam buku tugas akhir ini.

Gambar 4‑19: Grafik hasil pengujian pola pertama

Gambar 4‑20: Grafik hasil pengujian pola kedua

Dari grafik pada gambar 4-19 dan gambar 4-20 terlihat bahwa struktur perfect clique semakin berkurang jumlahnya seiring bertambahnya jumlah node maksimum dalam GCC. Struktur near clique dan near star juga semakin menurun pada jumlah node maksimum dalam GCC diatas 10. Penurunan ini dikarenakan bentuk GCC yang banyak tersusun dari subgraph clique dan star lalu ketika algoritma Slashburn beroperasi, Slashburn menghapus hub node dalam GCC, memotong edgenya dan membentuk subgraph-subgraph near chain. Terlihat dari grafik 4-11 dan 4-12 subgraph near chain bertambah jumlahnya pada jumlah node maksimum dalam GCC diatas 10.

### 4.3.3 Analisis pengaruh jumlah node maksimum dalam GCC terhadap waktu eksekusi VoG

Pada pengujian VoG ini digunakan dataset yang didapat dari hasil crawl web quora.com. Dari satu dataset tersebut akan dilakukan pengujian dengan mengubah-ubah jumlah node maksimum dalam GCC lalu mencatat waktu eksekusi VoG. Berikut ini adalah tabel hasil pengujian:

Tabel 4‑12: Hasil pengujian waktu eksekusi VoG dengan jumlah node maksimum dalam GCC = 5

|  |  |  |  |
| --- | --- | --- | --- |
| **Percobaan ke** | **Eksekusi Slashburn (ms)** | **Eksekusi Subgraph Labeling (ms)** | **Eksekusi VoG (ms)** |
| **1** | **10513** | **3498** | **14012** |
| **2** | **10850** | **2932** | **13782** |
| **3** | **10751** | **3166** | **13918** |
| **4** | **11254** | **3131** | **14386** |
| **5** | **11549** | **2988** | **14538** |
| **6** | **11192** | **2741** | **13934** |
| **7** | **11543** | **3402** | **14946** |
| **8** | **10665** | **3067** | **13733** |
| **9** | **12474** | **4199** | **16674** |
| **10** | **11422** | **3195** | **14617** |
| **11** | **12892** | **3741** | **16633** |
| **12** | **11973** | **3738** | **15712** |
| **13** | **11182** | **3388** | **14571** |
| **14** | **11579** | **3265** | **14844** |
| **15** | **12151** | **3218** | **15370** |
| **16** | **11773** | **3198** | **14972** |
| **17** | **11209** | **3708** | **14918** |
| **18** | **11494** | **2714** | **14208** |
| **19** | **11288** | **4006** | **15294** |
| **20** | **11265** | **3159** | **14424** |
| **21** | **12422** | **4916** | **17339** |
| **22** | **11861** | **3885** | **15747** |
| **23** | **12024** | **3335** | **15360** |
| **24** | **11738** | **3464** | **15202** |
| **25** | **11204** | **3128** | **14333** |
| **26** | **11400** | **3381** | **14782** |
| **27** | **11605** | **3649** | **15254** |
| **28** | **11169** | **3488** | **14658** |
| **29** | **11515** | **3201** | **14717** |
| **30** | **11938** | **3211** | **15149** |
| **Average** | **11529.83333** | **3403.733333** | **14934.23333** |

Untuk lebih rincinya, hasil pengujian terkait waktu eksekusi VoG terlampir pada lampiran 2 dalam buku tugas akhir ini. Berikut ini adalah tabel hasil pengujian yang diperoleh mengenai waktu rata-rata eksekusi VoG:

Tabel 4‑13: Hasil pengujian waktu rata-rata eksekusi VoG

|  |  |  |  |
| --- | --- | --- | --- |
| **Jumlah node maksimum dalam GCC** | **Eksekusi Slashburn (ms)** | **Eksekusi Subgraph Labeling (ms)** | **Eksekusi VoG (ms)** |
| **5** | **11529.83333** | **3403.733333** | **14934.23333** |
| **10** | **5085.966667** | **3755.133333** | **8841.6** |
| **15** | **3257.6** | **3323.433333** | **6581.466667** |
| **20** | **2375.033333** | **3472.633333** | **5848** |

Gambar 4‑21: Grafik hasil pengujian waktu eksekusi

Dari grafik pada gambar 4-21 terlihat bahwa semakin banyak jumlah node maksimum dalam GCC semakin cepat juga waktu eksekusi VoG. Hal ini juga dipengaruhi oleh waktu eksekusi algoritma Slashburn. Semakin banyak jumlah node maksimum dalam GCC semakin cepat juga waktu eksekusi Slashburn. Hal ini dikarenakan semakin banyak jumlah node dalam GCC subgraph yang dihasilkan semakin sedikit, oleh karena itu proses untuk mengurai graph menjadi subgraph yang sedikit jumlahnya membutuhkan waktu yang lebih singkat.

Untuk waktu eksekusi proses subgraph labeling dari subgraph-subgraph yang dihasilkan oleh Slashburn relatif stabil pada waktu 3 detik. Hal ini dikarenakan oleh jumlah banyaknya subgraph dan jumlah node yang ada pada subgraph. Pada jumlah node maksimum dalam GCC sama dengan 5, subgraph yang dihasilkan berjumlah lebih banyak daripada pada jumlah node maksimum dalam GCC sama dengan 20. Untuk memproses banyak subgraph dengan node yang sedikit waktu eksekusinya akan relatif sama dengan memproses sedikit subgraph dengan node yang banyak. Hal ini terbukti pada hasil pengujian dalam grafik di atas yang mana waktu eksekusi subgraph labeling relatif stabil pada waktu 3 detik.

### 4.3.4 Analisis informasi struktur subgraph

Dari hasil pengujian graph quora yang telah dilakukan sebelumnya, dapat diperoleh struktur clique, star, dan chain. Karena graph yang digunakan mempunyai representasi node sebagai user dan edgenya adalah relasi bahwa dua user menjawab pertanyaan yang sama, maka dengan metode manual inspection pada masing-masing subgraph yang dihasilkan, struktur clique dapat dianalisis informasinya sebagai subgraph yang banyak usernya melakukan jajak pendapat pada suatu pertanyaan tertentu. Sedangkan untuk struktur star dapat dianalisis informasinya bahwa terdapat satu user yang banyak berkontribusi menjawab banyak pertanyaan dalam web quora.

# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Berdasarkan hasil pengujian dapat diambil kesimpulan sebagai berikut:

1. Metode VoG dapat digunakan untuk men-summarize graph dan mengidentifikasi struktur subgraph yang dihasilkan. Subgraph-subgraph yang dihasilkan juga sesuai dengan vocabulary dalam VoG.
2. Jumlah node maksimum dalam GCC berbanding terbalik dengan jumlah struktur perfect clique, near clique, dan near star, namun berbanding lurus dengan jumlah struktur near chain. Semakin banyak node dalam GCC menghasilkan struktur clique dan star yang semakin sedikit namun semakin banyak menghasilkan struktur chain.
3. Jumlah node maksimum dalam GCC berbanding terbalik dengan waktu eksekusi VoG. Semakin banyak node dalam GCC semakin cepat waktu eksekusi VoG.
4. Struktur subgraph yang dihasilkan oleh VoG dapat dianalisis dengan metode manual inspection untuk mendapatkan informasi. Dalam tugas akhir ini, struktur clique menggambarkan bahwa banyak user di quora yang melakukan jajak pendapat. Struktur star menggambarkan bahwa ada satu user yang berkontribusi menjawab banyak pertanyaan.

## 5.2 Saran

Berikut ini adalah saran yang untuk penelitian selanjutnya:

1. Sistem yang dikembangkan dalam tugas akhir ini tidak terbatas pada web graph. Representasi graph apapun dapat digunakan menjadi dataset dalam sistem ini. Jadi untuk selanjutnya dapat digunakan dataset Email Graph yang mungkin dapat terdapat struktur bipartite atau chain yang menggambarkan terdapat jaringan broadcast email dan email forwarding.
2. Metode Slashburn yang digunakan untuk mengurai graph menjadi subgraph masih dapat dioptimasi menggunakan algoritma Greedy untuk memilih hub yang terbaik untuk dihapus.
3. Metode Subgraph Labeling dapat ditingkatkan akurasi pemilihan subgraphnya dengan encoding subgraph menjadi nilai bits. Nilai hasil encode yang rendah dapat dipilih untuk menentukan struktur subgraph.
4. Untuk menganalisis informasi dalam subgraph dapat menggunakan metode lain yang lebih akurat dan objektif.

# DAFTAR PUSTAKA

|  |  |
| --- | --- |
| [1] | I. Robinson, J. Webber dan E. Eifrem, Graph Databases, Sebastopol: O'Reilly Media, Inc, 2013. |
| [2] | P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang dan R. Zadeh, “WTF: The Who to Follow Service at Twitter,” dalam *International World Wide Web Conference Committee (IW3C2)*, Rio de Janeiro, 2013. |
| [3] | M. Curtiss, I. Becker, T. Bosman, S. Doroshenko, L. Grijincu, T. Jackson, S. Kunnatur, S. Lassen, P. Pronin, S. Sankar, G. Shen, G. Woss, C. Yang dan N. Zhang, “Unicorn: A System for Searching the Social Graph,” dalam *Proceedings of the VLDB Endowment*, Trento, 2013. |
| [4] | B. H. Murray dan A. Moore, “Sizing The Internet,” Cyveillance, 2000. |
| [5] | N. Shervashidze, S. Vishwanathan, T. H. Petri, K. Mehlhorn dan K. M. Borgwardt, “Efficient graphlet kernels for large graph comparison,” dalam *International Conference on Artificial Intelligence and Statistics (AISTATS)*, Florida, 2009. |
| [6] | D. Koutra, U. Kang, J. Vreeken dan C. Faloutsos, “VoG : Summarizing and Understanding Large Graphs,” dalam *Proceedings of the 2014 SIAM International Conference on Data Mining*, 2014. |
| [7] | K. H. Rosen, Discrete Mathematics and Its Applications, New York: The McGraw-Hill Companies, Inc., 2012. |
| [8] | M. A. Kolosovskiy, Data structure for representing a graph:, Altai State Technical University, Russia. |
| [9] | R. Kumar, P. Raghavan, S. Rajagopalan, D. Shivakumar, A. S. Thomkins dan E. Upfal, “The Web as a Graph,” dalam *ACM*, Texas, 2000. |
| [10] | A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Thomkins dan J. Wiener, “Graph Structure in the Web,” *Computer Nerworks Elsevier,* vol. 33, pp. 309-320, 2000. |
| [11] | Dhia, “Finding the Patterns in a Mysterious New DGA,” [Online]. Available: https://labs.opendns.com/2013/10/24/mysterious-dga-lets-investigate-sgraph/. [Diakses 18 March 2015]. |
| [12] | Y. Lim, U. Kang dan C. Faloutsos, “SlashBurn: Graph Compression and Mining beyond Caveman Communities,” dalam *IEEE Transaction On Knowledge and Data Engineering*, 2014. |
| [13] | R. Balakhisnan dan K. Ranganathan, A Textbook of Graph Theory, Springer Science & Business Media, 2012. |

# LAMPIRAN 1 : HASIL PENGUJIAN SUBGRAPH YANG DIHASILKAN OLEH VoG

Tabel 4‑2: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 5

|  |  |
| --- | --- |
| Struktur | Jumlah |
| Perfect Clique | 337 |
| Perfect Star | 13 |
| Perfect Bipartite | 0 |
| Perfect Chain | 2 |
| Near Clique | 12 |
| Near Star | 21 |
| Near Bipartite | 0 |
| Near Chain | 2 |

Tabel 4‑3: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 7

|  |  |
| --- | --- |
| Struktur | Jumlah |
| Perfect Clique | 293 |
| Perfect Star | 9 |
| Perfect Bipartite | 0 |
| Perfect Chain | 2 |
| Near Clique | 21 |
| Near Star | 47 |
| Near Bipartite | 0 |
| Near Chain | 9 |

Tabel 4‑4: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 9

|  |  |
| --- | --- |
| Struktur | Jumlah |
| Perfect Clique | 273 |
| Perfect Star | 5 |
| Perfect Bipartite | 0 |
| Perfect Chain | 1 |
| Near Clique | 21 |
| Near Star | 47 |
| Near Bipartite | 0 |
| Near Chain | 19 |

Tabel 4‑5: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 11

|  |  |
| --- | --- |
| Struktur | Jumlah |
| Perfect Clique | 253 |
| Perfect Star | 5 |
| Perfect Bipartite | 0 |
| Perfect Chain | 1 |
| Near Clique | 18 |
| Near Star | 50 |
| Near Bipartite | 0 |
| Near Chain | 27 |

Tabel 4‑3: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 5

|  |  |
| --- | --- |
| Struktur | Jumlah |
| Perfect Clique | 337 |
| Perfect Star | 13 |
| Perfect Bipartite | 0 |
| Perfect Chain | 2 |
| Near Clique | 12 |
| Near Star | 21 |
| Near Bipartite | 0 |
| Near Chain | 2 |

Tabel 4‑4: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 10

|  |  |
| --- | --- |
| Struktur | Jumlah |
| Perfect Clique | 264 |
| Perfect Star | 5 |
| Perfect Bipartite | 0 |
| Perfect Chain | 1 |
| Near Clique | 20 |
| Near Star | 49 |
| Near Bipartite | 0 |
| Near Chain | 22 |

Tabel 4‑8: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 15

|  |  |
| --- | --- |
| Struktur | Jumlah |
| Perfect Clique | 230 |
| Perfect Star | 5 |
| Perfect Bipartite | 0 |
| Perfect Chain | 1 |
| Near Clique | 11 |
| Near Star | 38 |
| Near Bipartite | 0 |
| Near Chain | 43 |

Tabel 4‑9: Tabel Hasil Identifikasi Struktur dengan jumlah node maksimum dalam GCC sama dengan 20

|  |  |
| --- | --- |
| Struktur | Jumlah |
| Perfect Clique | 225 |
| Perfect Star | 5 |
| Perfect Bipartite | 0 |
| Perfect Chain | 1 |
| Near Clique | 10 |
| Near Star | 34 |
| Near Bipartite | 0 |
| Near Chain | 43 |

# LAMPIRAN 2 : HASIL PENGUJIAN WAKTU EKSEKUSI VoG

Tabel 0‑1: Hasil pengujian waktu eksekusi VoG dengan jumlah node maksimum dalam GCC = 5

|  |  |  |  |
| --- | --- | --- | --- |
| **Percobaan ke** | **Eksekusi Slashburn (ms)** | **Eksekusi Subgraph Labeling (ms)** | **Eksekusi VoG (ms)** |
| **1** | **10513** | **3498** | **14012** |
| **2** | **10850** | **2932** | **13782** |
| **3** | **10751** | **3166** | **13918** |
| **4** | **11254** | **3131** | **14386** |
| **5** | **11549** | **2988** | **14538** |
| **6** | **11192** | **2741** | **13934** |
| **7** | **11543** | **3402** | **14946** |
| **8** | **10665** | **3067** | **13733** |
| **9** | **12474** | **4199** | **16674** |
| **10** | **11422** | **3195** | **14617** |
| **11** | **12892** | **3741** | **16633** |
| **12** | **11973** | **3738** | **15712** |
| **13** | **11182** | **3388** | **14571** |
| **14** | **11579** | **3265** | **14844** |
| **15** | **12151** | **3218** | **15370** |
| **16** | **11773** | **3198** | **14972** |
| **17** | **11209** | **3708** | **14918** |
| **18** | **11494** | **2714** | **14208** |
| **19** | **11288** | **4006** | **15294** |
| **20** | **11265** | **3159** | **14424** |
| **21** | **12422** | **4916** | **17339** |
| **22** | **11861** | **3885** | **15747** |
| **23** | **12024** | **3335** | **15360** |
| **24** | **11738** | **3464** | **15202** |
| **25** | **11204** | **3128** | **14333** |
| **26** | **11400** | **3381** | **14782** |
| **27** | **11605** | **3649** | **15254** |
| **28** | **11169** | **3488** | **14658** |
| **29** | **11515** | **3201** | **14717** |
| **30** | **11938** | **3211** | **15149** |
| **Average** | **11529.83333** | **3403.733333** | **14934.23333** |

Tabel 0‑2: Hasil pengujian waktu eksekusi VoG dengan jumlah node maksimum dalam GCC = 10

|  |  |  |  |
| --- | --- | --- | --- |
| **Percobaan ke** | **Eksekusi Slashburn (ms)** | **Eksekusi Subgraph Labeling (ms)** | **Eksekusi VoG (ms)** |
| **1** | **4535** | **3332** | **7868** |
| **2** | **4434** | **3715** | **8150** |
| **3** | **4614** | **3747** | **8362** |
| **4** | **4665** | **3061** | **7727** |
| **5** | **5230** | **5128** | **10358** |
| **6** | **4469** | **3121** | **7591** |
| **7** | **5017** | **3128** | **8146** |
| **8** | **4967** | **4364** | **9332** |
| **9** | **5036** | **5142** | **10178** |
| **10** | **4912** | **3249** | **8162** |
| **11** | **5111** | **2855** | **7967** |
| **12** | **4931** | **5010** | **9941** |
| **13** | **4707** | **3400** | **8108** |
| **14** | **4959** | **3981** | **8940** |
| **15** | **6209** | **3800** | **10010** |
| **16** | **4945** | **3913** | **8859** |
| **17** | **4976** | **3907** | **8883** |
| **18** | **5233** | **3180** | **8414** |
| **19** | **5651** | **3976** | **9627** |
| **20** | **5708** | **3488** | **9196** |
| **21** | **5289** | **3259** | **8548** |
| **22** | **5129** | **3868** | **8997** |
| **23** | **5474** | **3644** | **9118** |
| **24** | **4299** | **3915** | **8214** |
| **25** | **5205** | **3661** | **8867** |
| **26** | **5690** | **3784** | **9474** |
| **27** | **5537** | **4021** | **9558** |
| **28** | **4461** | **3655** | **8116** |
| **29** | **5495** | **4096** | **9592** |
| **30** | **5691** | **3254** | **8945** |
| **Average** | **5085.966667** | **3755.133333** | **8841.6** |

Tabel 0‑3: Hasil pengujian waktu eksekusi VoG dengan jumlah node maksimum dalam GCC = 15

|  |  |  |  |
| --- | --- | --- | --- |
| **Percobaan ke** | **Eksekusi Slashburn (ms)** | **Eksekusi Subgraph Labeling (ms)** | **Eksekusi VoG (ms)** |
| **1** | **3161** | **3852** | **7013** |
| **2** | **3013** | **3217** | **6230** |
| **3** | **3078** | **3446** | **6525** |
| **4** | **2477** | **2938** | **5415** |
| **5** | **3341** | **3542** | **6883** |
| **6** | **3328** | **3151** | **6480** |
| **7** | **3254** | **2895** | **6149** |
| **8** | **3431** | **2973** | **6405** |
| **9** | **3172** | **3252** | **6424** |
| **10** | **3261** | **3334** | **6595** |
| **11** | **3108** | **4558** | **7667** |
| **12** | **3544** | **2993** | **6538** |
| **13** | **3192** | **3578** | **6771** |
| **14** | **3486** | **3677** | **7163** |
| **15** | **3403** | **3030** | **6433** |
| **16** | **3004** | **3312** | **6317** |
| **17** | **2879** | **2724** | **5603** |
| **18** | **3264** | **2868** | **6132** |
| **19** | **3156** | **3630** | **6786** |
| **20** | **3552** | **2854** | **6406** |
| **21** | **3013** | **3148** | **6161** |
| **22** | **2813** | **3146** | **5960** |
| **23** | **3307** | **3379** | **6687** |
| **24** | **3208** | **2945** | **6154** |
| **25** | **3587** | **3244** | **6831** |
| **26** | **3406** | **3254** | **6660** |
| **27** | **3327** | **3600** | **6928** |
| **28** | **3537** | **3437** | **6975** |
| **29** | **3706** | **3542** | **7249** |
| **30** | **3720** | **4184** | **7904** |
| **Average** | **3257.6** | **3323.433333** | **6581.466667** |

Tabel 0‑4: Hasil pengujian waktu eksekusi VoG dengan jumlah node maksimum dalam GCC = 20

|  |  |  |  |
| --- | --- | --- | --- |
| **Percobaan ke** | **Eksekusi Slashburn (ms)** | **Eksekusi Subgraph Labeling (ms)** | **Eksekusi VoG (ms)** |
| **1** | **2946** | **2867** | **5813** |
| **2** | **2703** | **3410** | **6113** |
| **3** | **3213** | **3370** | **6583** |
| **4** | **3061** | **3055** | **6116** |
| **5** | **1915** | **3354** | **5269** |
| **6** | **1871** | **2695** | **4566** |
| **7** | **1908** | **3147** | **5055** |
| **8** | **2519** | **3543** | **6063** |
| **9** | **2149** | **2850** | **4999** |
| **10** | **2311** | **2970** | **5282** |
| **11** | **2391** | **3403** | **5795** |
| **12** | **2113** | **3365** | **5478** |
| **13** | **2133** | **3534** | **5667** |
| **14** | **3245** | **3550** | **6796** |
| **15** | **2161** | **3587** | **5748** |
| **16** | **2289** | **4455** | **6745** |
| **17** | **2570** | **3561** | **6131** |
| **18** | **2083** | **3015** | **5098** |
| **19** | **2055** | **3363** | **5419** |
| **20** | **2156** | **3143** | **5299** |
| **21** | **2064** | **3345** | **5410** |
| **22** | **2220** | **3122** | **5342** |
| **23** | **2112** | **3507** | **5619** |
| **24** | **2256** | **3927** | **6184** |
| **25** | **2503** | **3999** | **6503** |
| **26** | **2512** | **3998** | **6510** |
| **27** | **2260** | **4538** | **6798** |
| **28** | **2046** | **3407** | **5454** |
| **29** | **2236** | **3798** | **6034** |
| **30** | **3250** | **4301** | **7551** |
| **Average** | **2375.033333** | **3472.633333** | **5848** |