Algoritma Pembangkit Graf Planar Terhubung Sederhana Acak dengan Jumlah Simpul yang Ditentukan

Muhammad Nafis Habibi - 13524018
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: nafiskreatif@gmail.com , 13524018@std.stei.itb.ac.id

***Abstrak*—Pembangkit graf planar acak berguna untuk berbagai hal di bidang informatika. Makalah ini membahas tentang algoritma yang dapat membuat suatu graf planar terhubung sederhana. Algoritma pertama diambil dari milik Eric Fusy yang secara teori dapat menghasilkan graf dalam kompleksitas waktu kuadratik untuk jumlah simpul tetap dan waktu linear untuk jumlah simpul di antara sebuah interval. Algoritma kedua menghasilkan graf planar dengan cara menambahkan simpul satu per satu yang dihubungkan dengan beberapa simpul terluar acak dari graf. Algoritma ketiga menghasilkan graf planar dengan cara menghapus beberapa sisi secara acak dari graf planar maksimal sambil memastikan keterhubungan graf.**

***Kata Kunci*—graf planar, algoritma, pembangkit acak**

I. Pendahuluan

Berbagai masalah dapat direpresentasikan sebagai sebuah graf. Transformasi menjadi graf dapat memudahkan penyelesaian masalah dan pendapatan wawasan baru. Contohnya yaitu peta daerah dapat diubah menjadi graf planar yang berisi hubungan daerah yang berbatasan.

Graf acak telah menjadi kebutuhan, utamanya dalam bidang informatika. Graf khusus, seperti graf planar, memiliki kegunaannya tersendiri. Karena dapat digambar tanpa adanya sisi yang bersilangan, graf planar berguna untuk visualisasi dan jaringan atau sirkuit yang membutuhkan planaritas. Dalam pembuatan *game* pula, graf planar dapat dijadikan peta dan gambar yang menarik.

Pada makalah ini, graf planar yang akan dibahas adalah graf planar terhubung sederhana (tanpa gelang atau sisi ganda). Hasil yang merata secara distribusi semua graf planar yang mungkin tidak terlalu ditekankan di sini. Tujuan makalah ini adalah:

- Bereksperimen dalam membuat algoritma yang dapat membuat graf planar terhubung sederhana acak.
- Mencoba algoritma yang telah dibuat di masa lampau.

II. Landasan Teori

A. Graf

Suatu graf (tidak berarah) G adalah pasangan himpunan (V, E) dengan V adalah himpunan tak kosong yang berisi simpul dan E adalah himpunan sisi berupa pasangan antara dua simpul pada graf. Setiap sisi di E menghubungkan dua simpul dari V [1].

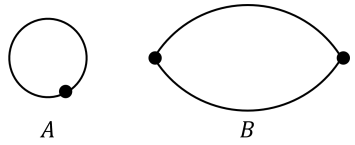
Berikut beberapa terminologi yang berhubungan dengan graf [2]:

- Bertetangga (*adjacent*): Dua buah simpul dikatakan bertetangga jika terhubung oleh suatu sisi secara langsung.
- Bersisian (*incident*): Untuk sebuah sisi e yang menghubungkan simpul v_1 dan v_2, dikatakan e bersisian dengan v_1 dan e bersisian dengan v_2.
- Derajat (*degree*): Derajat dari sebuah simpul adalah banyak sisi yang bersisian dengan simpul tersebut. Berdasarkan teorema jabat tangan, jumlah derajat pada suatu graf adalah dua kali jumlah sisinya.
- Lintasan (*path*): Lintasan dari simpul awal v_0 ke simpul tujuan v_n berisi barisan selang-seling antara simpul dan sisi, misalnya $\{v_0, e_1, v_1, e_2, v_2, ..., e_n, v_n\}$. Setiap sisi e_i bersisian dengan simpul v_{i-1} dan v_i.
- Siklus (*cycle*) atau sirkuit (*circuit*): Sirkuit adalah lintasan yang berawal dan berakhir pada simpul yang sama.
- Terhubung (*connected*): Dua simpul v_1 dan v_2 dikatakan terhubung jika terdapat lintasan dari v_1 ke v_2. Sebuah graf G dikatakan terhubung jika setiap pasang simpul (v_i, v_j) dalam graf terdapat lintasannya. Selain itu, G dikatakan tidak terhubung (*disconnected*).
- Ordo (*order*): Ordo suatu graf adalah jumlah dari simpulnya.

B. Jenis-Jenis Graf

1) Berdasarkan ada tidaknya gelang atau sisi ganda

Gelang (*loop*) adalah sisi yang menghubungkan sebuah simpul dengan simpul itu sendiri. Sisi ganda (*multi-edges*) adalah dua atau lebih sisi yang menghubungkan dua simpul yang sama.



Gambar 1. A adalah graf dengan gelang dan B adalah graf dengan sisi ganda

Berdasarkan ada tidaknya gelang atau sisi ganda di dalam graf, graf digolongkan menjadi dua macam:

a) Graf sederhana (simple graph)

Graf sederhana tidak memiliki gelang ataupun sisi ganda di dalamnya [2].

b) Graf tak sederhana (unsimple graph)

Graf tak sederhana memiliki gelang ataupun sisi ganda di dalamnya [2].

2) Berdasarkan orientasi arah pada graf

a) Graf tak berarah (undirected graph)

Setiap sisi pada graf berarah tidak diberikan orientasi arah. Pada graf tak berarah, sisi (v_1, v_2) dianggap sama dengan sisi (v_2, v_1). Sisi (v_1, v_2) pada graf tak berarah dapat direpresentasikan dengan dua sisi (v_1, v_2) dan (v_2, v_1) pada graf berarah.

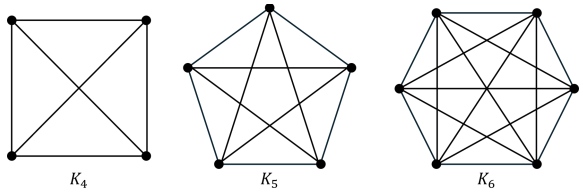
b) Graf berarah (directed graph)

Setiap sisi pada graf berarah diberikan orientasi arah. Sisi pada graf berarah menunjukkan hubungan satu arah antar simpulnya.

3) Beberapa Graf Khusus

a) Graf Lengkap (Complete Graph)

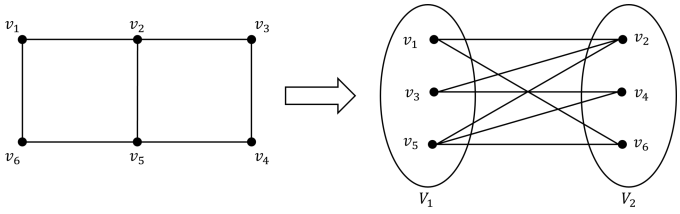
Graf lengkap adalah graf yang setiap simpulnya bertetangga dengan simpul lain dalam graf tersebut. Graf lengkap dengan jumlah simpul n dinotasikan dengan K_n.



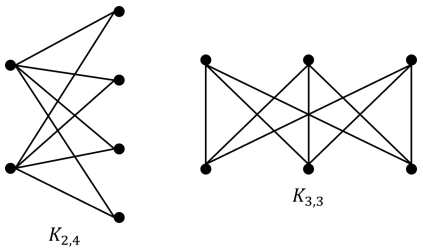
Gambar 2. Graf lengkap dengan jumlah simpul 4, 5, dan 6.

b) Graf Bipartit (Bipartite Graph)

Graf bipartit adalah graf yang simpul-simpulnya dapat dibagi menjadi dua himpunan V_1 dan V_2 semedikian sehingga setiap sisi pada graf menghubungkan simpul dari V_1 dengan V_2. Graf bipartit dinyatakan dengan $G(V_1, V_2)$ [2]. Jika terdapat semua sisi yang mungkin untuk menhubungkan setiap simpul dari V_1 dengan V_2 pada graf, graf tersebut disebut graf bipartit lengkap dan dapat dinotasikan dengan $K_{n, m}$ (n adalah jumlah simpul di V_1 dan m adalah jumlah simpul di V_2).



Gambar 3. Contoh graf bipartit



Gambar 4. Contoh graf bipartit lengkap

c) Pohon (Tree)

Pohon adalah graf tak berarah yang terhubung dan tidak memiliki siklus [4]. Pohon G dengan jumlah simpul n memiliki sifat-sifat berikut:

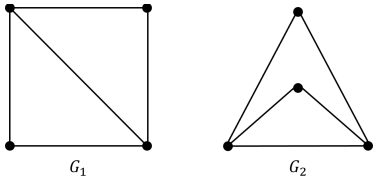
- Jumlah sisinya adalah e = n + 1.
- Setiap pasang simpul dalam G memiliki lintasan tunggal.
- G tidak mengandung sirkuit dan penambahan satu sisi pada G akan mengakibatkan adanya satu sirkuit.
- Semua sisi dalam G adalah jembatan.

Pohon merentang adalah subgraf dari G yang memiliki semua simpul pada G. Setiap graf terhubung memiliki pohon merentang sebagai subgrafnya.

C. Hubungan Antar Dua Graf

1) Isomorfik

Dua graf dikatakan isomorfik jika terdapat korespondensi satu-satu antara simpul dan sisi sedemikian sehingga hubungan kesisian tetap terjaga [3]. Misalkan sisi e bersisian dengan simpul v_1 dan v_2 di graf G_1, maka e' harus bersisian juga dengan simpul v_1' dan v_2' di graf G_2. Dengan kata lain, graf G_1 dapat digambar sedimikian rupa menjadi G_2, baik dengan penamaan simpul dan sisi yang sama ataupun berbeda.



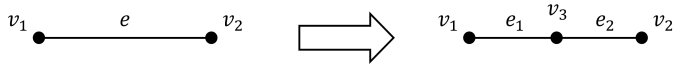
Gambar 5. Dua graf yang isomorfik

Dua graf yang isomorfik memiliki kesamaan berikut [3]:

- Mempunyai jumlah simpul yang sama
- Mempunyai jumlah sisi yang sama
- Mempunyai jumlah simpul berderajat tertentu yang sama

2) Homeomorfik

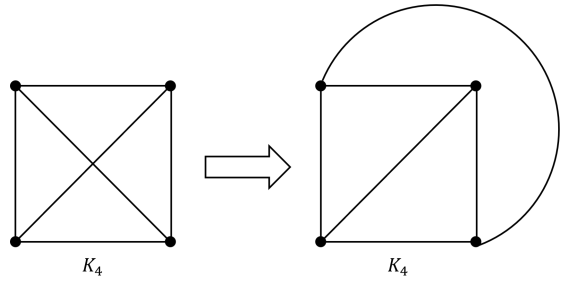
Dua graf G_1 dan G_2 dikatakan homeomorfik jika terdapat salah satu subdivisi dari G_1 yang isomorfik dengan salah satu subdivisi dari G_2. Subdivisi dari graf G adalah hasil dari subdivisi dari beberapa sisi di dalam G. Misalkan terdapat sisi e = (v_1, v_2), sisi tersebut dapat dipisahkan menjadi dua sisi dengan cara menambah simpul baru v_3 di tengah-tengah sisi e.



Gambar 6. Subdivisi dari sisi e menjadi e_1 dan e_2.

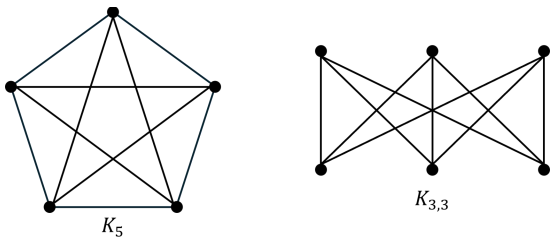
D. Graf Planar

Graf planar adalah graf yang dapat digambarkan pada bidang datar tanpa adanya sisi yang bersilangan [3]. Contoh dari graf planar adalah graf lengkap dengan empat simpul.

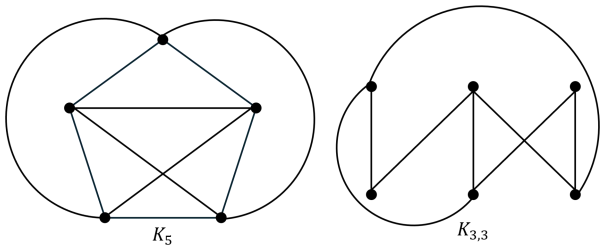


Gambar 7. Contoh graf planar berupa graf lengkap dengan empat simpul.

Menurut teorema Kuratowski, suatu graf adalah planar jika dan hanya jika tidak mengandung subgraf yang homeomorfik terhadap graf K_5 (graf lengkap dengan lima simpul) dan graf $K_{3,3}$ (graf bipartit lengkap dengan masing-masing tiga simpul).



Gambar 8. Kedua graf Kuratowski.

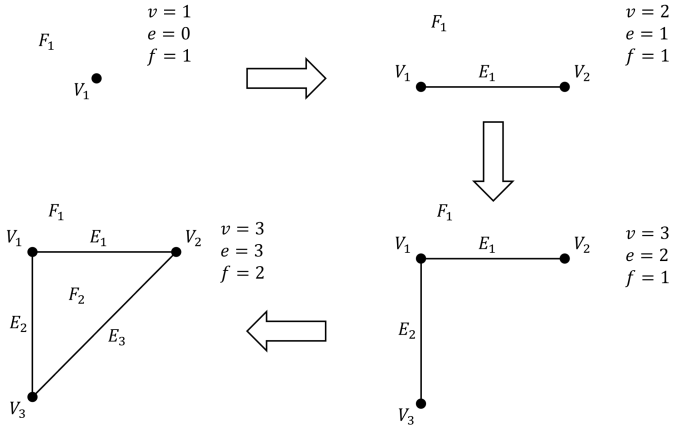


Gambar 9. Graf Kuratowski selalu ada sisi yang bersilangan

Rumus Euler (*Euler's formula*) menyatakan bahwa jumlah simpul (v), jumlah sisi (e), dan jumlah daerah (f) pada graf planar terhubung adalah:

$$v - e + f = 2 \quad (1)$$

Daerah di luar graf planar termasuk dalam himpunan daerah. Rumus Euler dapat dibuktikan secara induksi. Dimulai dengan graf dengan satu simpul tanpa sisi dan satu daerah bagian luar (v = 1, e = 0, f = 1). Saat menambahkan satu sisi, jika sisi menghubungkan dua simpul yang ada pada graf, satu daerah baru akan terbentuk (e bertambah satu dan f bertambah satu). Jika sisi menghubungkan satu simpul dalam graf dan satu simpul baru, tidak ada daerah baru (e bertambah satu dan v bertambah satu). Penambahan satu sisi membuat adanya pertambahan salah satu dari simpul atau daerah. Oleh karena itu, persamaan (1) tetap terpenuhi.



Gambar 10. Pembuktian rumus Euler secara induksi dengan penambahan sisi

Pada graf planar terhubung sederhana, setiap daerah, kecuali daerah luar, dibatasi oleh setidaknya tiga sisi dan setiap sisi paling banyak bersentuhan dengan dua daerah sehingga $3f \leq 2e$. Diturunkan dari rumus Euler, didapatkan ketidaksamaan Euler.

$$e \leq 3v - 6 \quad (2)$$

Setiap graf planar terhubung sederhana memenuhi (2), tetapi ada graf tidak planar yang juga memenuhi (2), contohnya graf $K_{3,3}$. Jika diasumsikan bahwa setiap daerah dibatasi oleh setidaknya empat sisi, dari penurunan rumus Euler diperoleh:

$$e \leq 2v - 4 \quad (3)$$

Sebuah graf planar G dikatakan planar maksimal jika untuk setiap pasangan simpul tak berdekatan u dan v dalam G maka G + uv adalah tidak planar. Batas setiap daerah terdiri dari tiga sisi pada G dengan jumlah simpul lebih dari 2. Karena itu, graf planar maksimal disebut juga graf planar segitiga. [6].

III. ALGORITMA

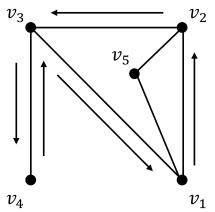
A. *Éric Fusy (2005)*

Algoritma oleh Eric didasarkan pada prinsip dari *Boltzmann sampler*, sebuah algoritma untuk mengambil sampel dari *combinatorial class* (himpunan yang berisi semua objek dari kombinasi yang mungkin). Dari kombinasi yang ada, *Boltzmann sampler* mengambil sebuah objek dengan besar n dengan peluang yang proporsional dengan x^n untuk objek tak berlabel dan $x^n/n!$ untuk objek berlabel. Nilai x berupa bilangan *real* yang ditentukan. Peluang sebuah objek diambil adalah sama untuk sesama objek dengan besar yang sama [5].

Algortima ini dapat membuat sebuah graf planar dalam kompleksitas waktu $O(n^2)$ untuk jumlah simpul n dan $O(n/\varepsilon)$ untuk jumlah simpul pada interval $[n - \varepsilon, n + \varepsilon]$. Untuk sebuah pembangkit graf planar acak yang merata distribusinya (*uniformly distributed*), algoritma ini lebih baik dari algoritma yang telah ada (sebelum milik Eric Fusy).

B. *Menambah Simpul yang Dihubungkan ke Beberapa Simpul Terluar secara Langsung*

Misalkan terdapat suatu graf planar terhubung sederhana dengan himpunan simpul V dan himpunan sisi E. Definisikan list berkait siklis A yang berisi simpul yang bersentuhan dengan daerah luar graf. Simpul-simpul pada A diurutkan dengan urutan sesuai arah jarum jam atau kebalikannya. Simpul pertama dapat dipilih secara acak. Untuk setiap $a_i \in A$, terdapat sisi yang menghubungkan a_i dengan a_{i+1}, kecuali kecuali untuk simpul terakhir a_n, terdapat sisi yang menghubungkan a_n dengan a_1. Terkhusus untuk a_i yang berderajat satu, arah iterasi dikembalikan ke a_{i-1}. Dengan kata lain $a_{i+1} = a_{i-1}$ untuk a_i berderajat satu.



Gambar 11. Contoh urutan simpul dalam list berkait A

Saat menambahkan simpul pada graf, ambil salah satu simpul dari A dan tetapkan sebagai a_1 sambil mempertahankan urutan iterasi searah jarum jam. Lalu, buat satu simpul baru p yang terhubung dengan a_1. Simpul p memiliki peluang sebesar 1/2 untuk terhubung pada masing-masing simpul pada A selain a_1. Karena bisa terdapat lebih dari satu simpul yang sama pada A, simpul yang sudah terhubung harus dipastikan tidak terhubung dua kali. Simpul yang sudah terhubung dapat dimasukkan ke dalam sebuah himpunan B.

Simpul yang ada di dalam A perlu diubah menyesuaikan penambahan sisi. Sisi yang menghubungkan p dengan A_k akan menutupi semua simpul a_i dengan $2 \leq i \leq k - 1$. Jadi, simpul yang dihilangkan dari A adalah a_i dengan $2 \leq i \leq k_{maks} - 1$ dan k_{maks} adalah urutan terakhir dari simpul yang bertetangga dengan p. Lalu, p disisipkan di antara a_1 dan a_k.

Berikut adalah *pseudocode* dari prosedur penambahan simpul:

```
procedure tambahSimpul(V: HimpunanSimpul, E: HimpunanSisi, A: ListBerkaitSimpulTerluar)
  p ← buatSimpul()
  V ← V ∪ {p}
  randomize(A)
  E ← E ∪ {(p, a1)}
  B ← {}
  k ← 1
  i traversal [2..length(A)]
    if (ai ∉ B and random() < 0.5) then
      E ← E ∪ {(p, ai)}
      B ← B ∪ {ai}
      k ← i
  A ← {a1, p, ak, ak+1, ..., an}
```

Kode 1. *Pseudocode* tambahSimpul untuk algoritma B

Algoritma dapat dimulai dengan graf berisi satu simpul tanpa sisi. Himpunan simpul terluar juga berisi satu simpul yang sama (A = V). Maka, untuk membuat sebuah graf planar terhubung sederhana dengan simpul sebanyak n, dapat dilakukan iterasi penambahan simpul sebanyak n – 1 kali.

Berikut adalah *pseudocode* dari prosedur pembuatan graf planar terhubung sederhana:

```
function buatGrafPlanar(n: integer) → Graf
  V ← {buatSimpul()}
  E ← {}
  A ← {v1}
  i traversal [2..n]
    tambahSimpul(V, E, A)
  return (V, E)
```

Kode 2. *Pseudocode* buatGrafPlanar untuk algoritma B

Waktu yang dibutuhkan untuk menjalankan "tambahSimpul" didapat dari waktu traversal list berkait A. Besar maksimal dari A sama dengan besar maksimal V, yaitu n, sehingga kompleksitas waktu yang dibutuhkan yaitu O(n). Prosedur "tambahSimpul" dilakukan n – 1 kali pada fungsi "buatGrafPlanar". Maka dari itu, kompleksitas waktu dari algoritma di atas adalah $O(n^2)$.

C. *Membuat Graf Planar Maksimal, Lalu Menghapus Sisi-Sisinya*

Graf planar maksimal memiliki sisi sebanyak 3v – 6 dan daerah sebanyak 2v – 4. Sisi-sisi pada graf planar maksimal G dapat dihapus secara acak. untuk mendapatkan subgraf dari G

dengan jumlah sisi acak. Pilihan lainnya yaitu dengan menghapus beberapa sisi hingga tersisa m sisi.

Sama seperti Bagian III.B, definisikan list berkait siklis A yang berisi simpul yang bersentuhan dengan daerah luar graf. Simpul-simpul pada A diurutkan dengan urutan sesuai arah jarum jam atau kebalikannya. Untuk setiap $a_i \in A$, terdapat sisi yang menghubungkan a_i dengan a_{i+1}, kecuali untuk simpul terakhir a_n, terdapat sisi yang menghubungkan a_n dengan a_1.

Karena semua daerah pada graf planar maksimal berbentuk segitiga (kecuali daerah luar), penambahan simpul baru p bisa dengan cara menambahkan sisi pada dua simpul berurutan. Misalkan a_i adalah simpul acak yang diambil dari A, maka sisi yang ditambahkan yaitu (a_i, p) dan (p, a_{i+1}). Daerah baru berbentuk segitiga terbentuk dari tiga sisi (a_i, a_{i+1}), (a_i, p) dan (p, a_{i+1}). Berikut *pseudocode* dari penambahan simpul:

```
procedure tambahSimpul(V: HimpunanSimpul, E: HimpunanSisi, A: ListBerkaitSimpulTerluar)
  p ← buatSimpul()
  V ← V ∪ {p}
  ai ← randomNode(A)
  E ← E ∪ {(p, ai)}
  E ← E ∪ {(p, ai+1)}
  A ← {a1, a2, ..., ai, p, ai+1, ..., an}
```

Kode 3. *Pseudocode* tambahSimpul untuk algoritma C

Penambahan sisi tanpa menambah simpul juga dapat dilakukan. Pilih a_i dari A, lalu tambahkan sisi (a_i, a_{i+2}), maka daerah baru berbentuk segitiga terbentuk dari tiga sisi (a_i, a_{i+1}), (a_{i+1}, a_{i+2}) dan (a, a_{i+2}). Syarat dapat dilakukannya hal ini adalah panjang dari list A lebih dari dua. Berikut *pseudocode* dari penambahan sisi:

```
procedure tambahSisi(E: HimpunanSisi, A: ListBerkaitSimpulTerluar)
  ai ← randomNode(A)
  E ← E ∪ {(ai, ai+2)}
  A ← A ¥ {ai+1}
```

Kode 4. *Pseudocode* tambahSisi untuk algoritma C

Penambahan simpul dan sisi dilakukan secara acak hingga jumlah simpul mencapai n dan jumlah sisi mencapai 3n – 6. Setelah didapat graf planar maksimal G, dilakukan penghapusan sisi secara acak sambil mempertahankan keterhubungan graf. Salah satu caranya adalah dengan mencari pohon menjalar acak T pada G. Sisi-sisi yang termasuk dalam T tidak boleh dihapus untuk menjaga keterhubungan graf. Sisi-sisi lainnya dapat dipilih secara acak untuk dihapus. Jumlah sisi yang dihapus dapat acak ataupun ditentukan.

Berikut adalah *pseudocode* dari prosedur pembuatan graf planar terhubung sederhana:

```
function buatGrafPlanar(n: integer) → Graf
  V ← {buatSimpul(), buatSimpul()}
  E ← {(v1, v2)}
  A ← {v1, v2}
  while(|V| < n and |E| < 3n - 6)
    if (|V| < n and (random() < 0.5 or length(A) = 2)) then
      tambahSimpul(V, E, A)
    else
      tambahSisi(E, A)
  T ← getSpanningTree((V, E))
  for each ei ∈ E and ei ∉ T.E
    if (random() < 0.5) then
      E ← E ¥ {ei}
  return (V, E)
```

Kode 5. *Pseudocode* buatGrafPlanar untuk algoritma C

Waktu yang dibutuhkan untuk menjalankan "tambahSimpul" dan "tambahSisi" didapat dari waktu traversal list berkait A karena "randomNode" memerlukan traversal untuk mendapatkan simpul acak. Besar maksimal dari A sama dengan besar maksimal V, yaitu n, sehingga kompleksitas waktu yang dibutuhkan yaitu O(n). Pada fungsi "buatGrafPlanar", Prosedur "tambahSimpul" dilakukan n – 1 kali dan "tambahSisi" dilakukan maksimal 3n – 6 kali. Maka dari itu, kompleksitas waktu dari algoritma di atas adalah $O(n^2)$.

IV. Implementasi dan Hasil

A. *Éric Fusy (2005)*

Fusy, E. telah membuat implementasi dari algoritmanya dalam bahasa pemrograman Java (lihat lampiran 1). Program yang diberikan hanya menerima masukan 1000, 10000, dan 100000 sebagai target jumlah simpul dari graf yang diberikan. Jumlah simpul yang diberikan dapat menyimpang hingga ribuan, tetapi tetap di sekitar jumlah simpul yang diinginkan. Namun, programnya cukup sering error untuk graf besar. Kemungkinan karena limitasi komputer untuk menyimpan bilangan real.

Program dijalankan sebanyak 500 kali untuk tiap masukan. Berikut beberapa statistik terkait hasil dari algoritma di atas:

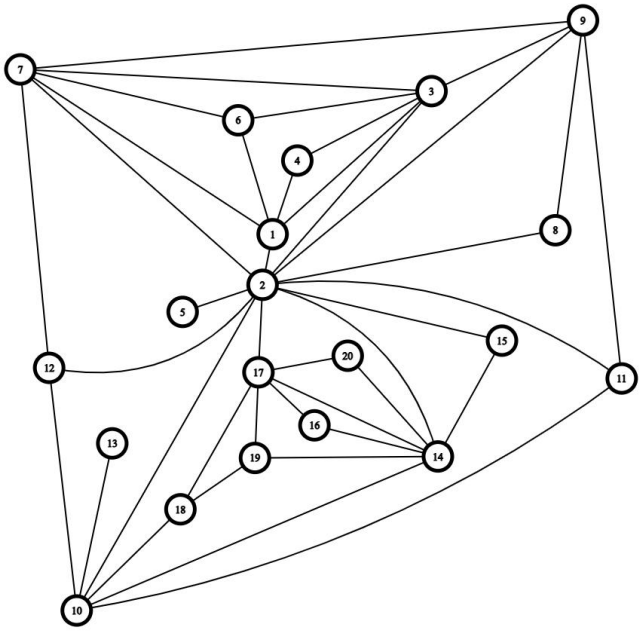
TABLE I. Statistik Algoritma Eric Fusy

Masukan (n)	Jumlah Program Berhasil	Rata-Rata Jumlah Simpul (v)	Rata-Rata Jumlah Sisi (e)	Rata-Rata Rasio Sisi/Simpul	Rata-Rata Waktu Eksekusi (ms)
1000	0.304	1877.40	631671.0	2.21063	8.416
10000	0.404	17678.55	7911376	2.21566	137.548
100000	0	-	-	-	-

Sayangnya, program didapati eror saat masukan n = 100000 dan tidak sanggup dijalankan. Melihat rata-rata waktu eksekusi, kompleksitas waktu yang secara teori dapat linear justru terlihat seperti kuadratik. Bisa jadi, ada proses yang menyebabkan waktu eksekusi dapat lebih besar. Hasil implementasi yang linear belum terlihat.

B. Menambah Simpul yang Dihubungkan ke Beberapa Simpul Terluar secara Langsung

Implementasi dari Bagian III.B dibuat dalam bahasa pemrograman c++ (lihat lampiran 2). Algoritma tersebut berhasil membuat graf planar terhubung sederhana.



Gambar 12. Salah satu graf planar yang dihasilkan dengan jumlah simpul = 20

Rata-rata dari jumlah sisi, derajat, dan waktu eksekusi berdasarkan jumlah simpul dihitung dengan menjalankan program 1000 kali. Berikut detail statistiknya:

TABLE II. STATISTIK ALGORITMA BAGIAN III.B

Jumlah Simpul (n)	Rata-Rata Jumlah Sisi (e)	Rata-Rata Derajat	Rata-Rata Waktu Eksekusi (ms)
100	225.434	4.50868	0.098
1000	2298.45	4.59690	0.909
10000	23028.2	4.60565	9.880
100000	230326	4.60652	124.21

Rata-rata waktu eksekusi terlihat memiliki pertumbuhan linear berdasarkan jumlah simpulnya. Hal ini tidak seperti analisis kompleksitas waktu yang menghasilkan batas atas berupa $O(n^2)$. Kompleksitas waktu kuadratik berasal dari asumsi bahwa besar dari A sama dengan V yang memiliki besar akhir n, sedangkan kompleksitas waktu untuk kasus rata-ratanya adalah linear. Misalkan besar dari A adalah m, maka saat menambah satu simpul, pertumbuhan m adalah:

$$m_{i+1} = m_i + 1 - (k - 2) \quad (4)$$

Konstanta k berada di interval $[1..m_i]$ sesuai dengan simpul terakhir dalam A yang terhubung dengan simpul baru. Berdasarkan banyak kombinasi dari A_i yang terpilih:

- Peluang $k = m_i$ adalah 1/2
- Peluang $k = m_i - 1$ adalah 1/4
- Pelauag $k = m_i - c$ adalah setengah dari peluang $k = m_i - c + 1$

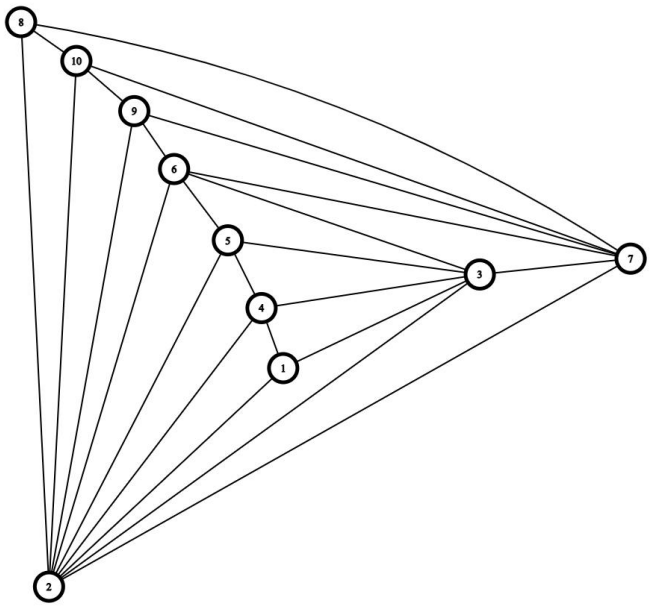
Dikarenakan k cenderung mendekati m_i, m_{i+1} akan cenderung mengecil. Nilai harapan dari k dapat dihitung dengan persamaan:

$$E(k) = \frac{\sum_{j=1}^{m_i} \left(\frac{1}{2}\right)^j \times (m_i - j + 1))}{m_i} \quad (5)$$

Dengan menjalankan program sebanyak 1000 kali dengan jumlah simpul sebanyak 10000, didapat rata-rata dari besar A adalah 4,58697. Angka tersebut cukup dekat dengan rata-rata derajat yang didapatkan.

C. Membuat Graf Planar Maksimal, Lalu Menghapus Sisi-Sisinya

Implementasi dari Bagian III.C dibuat dalam bahasa pemrograman c++ (lihat lampiran 2). Algoritma tersebut berhasil membuat graf planar terhubung sederhana dengan membuat graf planar maksimal terlebih dahulu. Setelah itu, dicari salah satu pohon menjalar dari graf yang didapat dengan cara *loop-erased random walk* [7]. Lalu, sisi-sisi selain pada pohon menjalar dihapus secara acak hingga menghasilkan graf planar akhir.



Gambar 13. Graf planar makimum yang dihasilkan dengan jumlah simpul = 10

Rata-rata dari jumlah sisi, derajat, dan waktu eksekusi berdasarkan jumlah simpul dihitung dengan menjalankan program 1000 kali, kecuali untuk jumlah simpul 100000, program dijalankan 10 kali. Berikut detail statistiknya:

TABLE III. STATISTIK ALGORITMA BAGIAN III.C

Jumlah Simpul (n)	Rata-Rata Jumlah Sisi (e)	Rata-Rata Derajat	Rata-Rata Waktu Eksekusi (ms)
100	201.102	4.02204	0.306
1000	2033.14	4.06628	4.605
10000	20353.1	4.07062	104.3
100000	203589	4.07179	15068

a.

Terlihat bahwa waktu eksekusi tumbuh secara kuadratik paling minimal. Dikarenakan penggunaan *std::set*, kompleksitas akhir dari algoritma tersebut adalah $O(n^2 \log n)$. Masih ada hal yang dapat diimplementasikan dengan lebih baik untuk memangkas waktu yang dibutuhkan saat eksekusi program.

V. KESIMPULAN

Pembuatan graf planar dapat dilakukan paling cepat dalam waktu linear. Untuk distribusi peluang yang rata, gunakan algoritma oleh Eric Fusy dan sebaiknya implementasikan sendiri karena implementasi yang diberikan kurang fleksibel dan mudah error untuk graf besar. Untuk algorima yang sederhana dan cepat tanpa memerhatikan kerataan distribusi peluang, gunakan algoritma yang dibahas di Bagian III.B, yaitu dengan penambahan simpul satu per satu. Algoritma di Bagian III.C masih perlu ditingkatkan lagi implementasinya. Masih ada algoritma lain selain pada makalah ini yang dapat diimplementasikan. Percobaan dalam implementasi diperlukan untuk mencari algoritma yang sesuai dengan kebutuhan.

LAMPIRAN

1. Implementasi oleh Eric Fusy (Java). http://igm.univ-mlv.fr/~fusy/Programs/BoltzmannPlanarGraphs.tar.gz
2. Implementasi algoritma B dan C (c++). https://github.com/NafisKreatif/Random-Planar-Graph

UCAPAN TERIMA KASIH

Puji syukur kepada Tuhan Yang Maha Esa yang telah memberikan kesempatan dan karunia-Nya sehingga penulis dapat menyelesaikan makalah ini**.** Terima kasih juga bagi orang tua yang telah membimbing hingga titik ini. Terima kasih kepada Pak Arrival Dwi Sentosa, S.Kom., M.T. sebagai dosen kelas K-02 pada mata kuliah IF1221 Matematika Diskrit. Terima kasih juga Pak Ir. Rinaldi Munir, M.T. yang telah menyediakan referensi dan materi. Penulis juga berterima kasih kepada teman-teman dan pihak lainnya yang terus mendukung penulis.

REFERENCES

[1] Marsudi, 2016. Teori Graf. Universitas Briwijaya Press. ISBN 978-602-432-015-7.

[2] Munir, R., 2024. "Graf (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf (diakses 16 Juni 2025)

[3] Munir, R., 2024. "Graf (Bagian 2)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf (diakses 16 Juni 2025)

[4] Munir, R., 2024. "Pohon (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/23-Pohon-Bag1-2024.pdf (diakses 16 Juni 2025)

[5] Fusy, É., 2005. Quadratic exact-size and linear approximate-size random generation of planar graphs. *Discrete Mathematics & Theoretical Computer Science* (Proceedings).

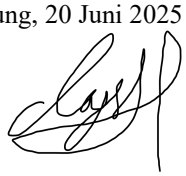
[6] Azizah, A. N., 2024. *PLANARITAS SUATU GRAF*. Undergraduate thesis, Universitas Muhammadiyah Malang.

[7] Wilson, D.B., 1996, July. Generating random spanning trees more quickly than the cover time. In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing (pp. 296-303).

PERNYATAAN

Dengan ini, saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Juni 2025



Muhammad Nafis Habibi
13524018