# Menerapkan Algoritma Dsatur dalam Meminimalisasi Potensi Kecurangan Seleksi Nasional Berbasis Tes (SNBT)

Maheswara Bayu Kaindra - 13523015[1]
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
[1]kaindramaheswara11@gmail.com, 13523015@std.stei.itb.ac.id

***Abstract*—Seleksi Nasional Berbasis Tes (SNBT) is one of main selection routes for highschool graduates in Indonesia to continue their studies at state universities. This selection process ensures a fair exam by minimizing the potential for cheating. In this research, an algorithm – the Dsatur Algorithm – is implemented to model the exam classroom as a graph that will be colored to ensure that exam participants sitting next to each other do not get the same question package. This research will use Python programming language for implementing Dsatur algorithm by using packages such as *NetworkX*, *Matplotlib*, and *NumPy* for modelling graphs. This research successfully shows that graph coloring is an effective approach to minimize the potential for cheating and Dsatur is an effective algorithm for graph coloring.**

***Keywords*—Dsatur Algorithm, graph coloring, Python, Seleksi Nasional Berbasis Tes**

## I. Pendahuluan

Seleksi Nasional Berbasis Tes (SNBT) menjadi jalur utama yang dipilih sebagian besar pelajar yang akan melanjutkan pendidikan ke perguruan tinggi negeri di Indonesia. Proses ini menjadi ajang persaingan bagi ribuan peserta dari seluruh penjuru Indonesia untuk mendapatkan kursi di perguruan tinggi impian. Untuk itu, diperlukan pengelolaan yang baik untuk memastikan integritas dan keadilan ujian. Salah satu hal yang penting untuk menjaga keadilan tersebut adalah dengan memastikan tidak ada kecurangan oleh peserta.

Salah satu bentuk kecurangan yang sering dilakukan saat ujian adalah penyebaran jawaban, terutama oleh peserta yang duduk berdekatan. Dalam hal ini, distribusi paket soal yang tidak optimal dapat meningkatkan risiko kecurangan. Dengan ini, penelitian ini dilakukan untuk memberi solusi yang dapat dibuktikan secara matematis untuk memastikan bahwa peserta yang duduk berdekatan tidak menerima paket soal yang sama.

Untuk memodelkan ruang ujian, digunakan teori Graf, sebuah cabang ilmu Matematika Diskrit. Teori graf mencakup sebuah konsep yang dinamakan pewarnaan graf (salah satunya pewarnaan vertex) yang dapat memastikan setiap simpul berdekatan memiliki warna berbeda. Dalam hal ini, simpul merepresentasikan peserta ujian, sisi merepresentasikan kedekatan tempat duduk peserta ujian, dan warna merepresentasikan paket-paket soal ujian. Algoritma Dsatur menjadi algoritma pewarnaan graf yang digunakan dalam penelitian ini.

Penelitian bertujuan untuk mengimplementasikan Algoritma Dsatur dalam pendistribusian paket soal ujian pada SNBT dengan memanfaatkan bahasa pemrograman Python. Modul seperti *NetworkX* akan digunakan untuk mempermudah pemodelan graf. Dengan menggabungkan konsep Matematika Diskrit dan teknologi, penelitian ini diharapkan dapat memberi solusi efektif untuk Seleksi Nasional Berbasis Tes yang lebih adil bagi semua peserta.

## II. Dasar Teori

### A. Teori Graf

#### 1. Pengertian dan Jenis-jenis Graf

Menurut Munir (2024), graf merupakan "diagram" yang merepresentasikan objek-objek dan hubungan antara objek-objek tersebut. Graf terdiri atas simpul (*vertex*) dan sisi (*edge*). Graf $G$ didefinisikan sebagai $G = (V, E)$ di mana

$V$ = himpunan tidak kosong dari simpul-simpul

$$V = \{v_1, v_2, ..., v_n\} \quad (1)$$

$E$ = himpunan sisi yang menghubungkan dua buah simpul

$$E = \{e_1, e_2, ..., e_n\} \quad (2)$$

Graf terdiri dari graf sederhana dan graf tidak sederhana. Graf sederhana merupakan graf yang tidak memiliki sisi ganda dan sisi gelang, sehingga setiap pasang simpul dihubungkan oleh satu buah sisi.

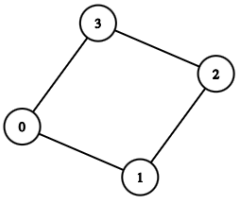


*Fig 2.1 Graf Sederhana*
*Dibuat dengan csacademy (csacademy.com)*

Graf tidak sederhana terdiri dari graf ganda (*multi-graph*) dan graf semu (*pseudo-graph*). Graf ganda merupakan graf yang mengandung sisi ganda, atau lebih dari satu sisi yang menghubungkan pasangan simpul yang sama. Graf semu

mengandung sisi gelang, yaitu sisi yang menghubungkan satu simpul ke simpul itu sendiri.

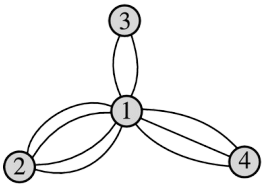


*Fig 2.2 Graf Berganda*
*Sumber:*
*https://www.researchgate.net/figure/A-multigraph-and-its-underlying-simple-graph_fig2_355441273*

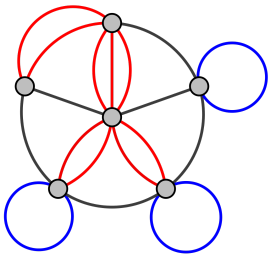

*Fig 2.3 Graf Semu*
*Sumber:*
*https://upload.wikimedia.org/wikipedia/commons/thumb/c/c9/Multi-pseudograph.svg/1200px-Multi-pseudograph.svg.png*

Berdasarkan orientasi arah sisi, graf dibedakan menjadi graf berarah dan graf tidak berarah. Pada graf tak-berarah, sisi-sisinya tidak mempunyai orientasi arah, sehingga apabila simpul $A$ terhubung dengan simpul $B$ oleh sisi $AB$, maka hubungan tersebut juga berlaku setara pada simpul $B$ (tidak ada yang menuju dan dituju).

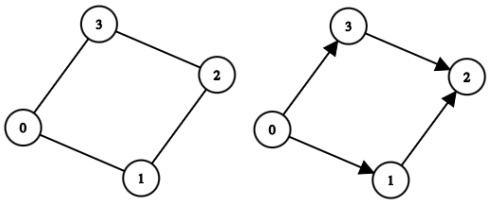


*Fig 2.4 Graf Tak-Berarah dan Graf Berarah*
*Dibuat dengan csacademy (csacademy.com)*

### *2. Terminologi (istilah penting) Graf*

Terdapat beberapa istilah penting yang perlu diketahui untuk memahami graf, diantaranya ketetanggaan, bersisian, simpul terpencil, graf kosong, dan derajat (Munir, 2024).

*a. Ketetanggan*
Dua buah simpul bertetangga jika terhubung langsung oleh sebuah sisi. Contohnya pada *Fig 2.4*, simpul 0 bertetangga dengan simpul 1.

*b. Bersisian*
Suatu sisi dikatakan bersisian dengan sepasang simpul apabila sisi tersebut secara langsung menghubungkan kedua simpul tersebut.

*c. Simpul Terpencil*
Merupakan istilah untuk menyebut simpul yang tidak dihubungkan oleh sisi manapun.

*d. Graf Kosong*
Suatu graf disebut kosong jika $E = \{\}$ (himpunan kosong).

*e. Derajat*
Derajat suatu simpul merepresentasikan jumlah sisi yang bersisian dengan simpul tersebut. Contohnya, simpul 1 pada *Fig 2.2* berderajat 9.

### *3. Pewarnaan Graf*

Pewarnaan graf merupakan konsep mewarnai graf sedemikian rupa sehingga simpul atau sisi yang berdekatan (berhubungan langsung) memiliki warna yang berbeda. Pada penelitian ini, pewarnaan graf terfokus pada pewarnaan simpul.

Pada pewarnaan graf, terdapat istilah bilangan kromatik. Bilangan kromatik merupakan jumlah minimum warna yang diperlukan untuk mewarnai graf.

$\chi(G)$, simbol kromatik ()

- Graf kosong memiliki $\chi(G) = 1$
- Graf lengkap memiliki $\chi(G) = n$, di mana $n$ merupakan jumlah simpul
- Graf bipartit mempunyai $\chi(G) = 2$
- Graf lingkaran bersimpul ganjil memiliki $\chi(G) = 3$ dan graf lingkaran bersimpul genap memiliki $\chi(G) = 2$
- Graf pohon memiliki $\chi(G) = 2$.

Contoh aplikasi pewarnaan graf adalah mewarnai peta agar lebih mudah dilihat dan distribusi soal ujian seperti pada penelitian ini.

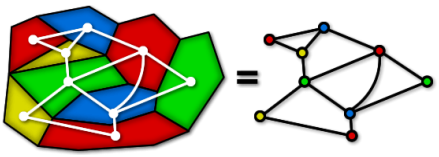


*Fig 2.5 Mewarnai Peta untuk Membuatnya Mudah Dilihat*
*Sumber:*
*https://www.networkpages.nl/wp-content/uploads/2015/07/fourcolour-1.png*

## *B. Algoritma Dsatur*

Dsatur (singkatan dari *degree of saturation*) merupakan algoritma salah satu algoritma pewarnaan graf yang mengutamakan simpul dengan derajat saturasi tertinggi. Di sini, derajat saturasi suatu simpul didefinisikan sebagai jumlah warna berbeda yang telah diberikan kepada simpul-simpul tetangganya. Berikut merupakan tahapan-tahapan algoritma Dsatur menurut Brelaz (1979).

1. Urutkan simpul-simpul dari derajat paling tinggi ke derajat paling rendah,
2. Warnai simpul dengan derajat tertinggi dengan warna pertama,
3. Pilihlah salah satu simpul yang belum diwarnai dengan derajat saturasi tertinggi,
4. Warnai simpul tersebut dengan warna paling mendekati warna pertama, namun bukan warna sebelumnya atau warna tetangga,
5. Lakukan langkah 3 sampai seluruh simpul diberi warna.

Secara kompleksitas waktu, algoritma Dsatur cukup baik karena memiliki nilai big-O $O(n^2)$ dengan $n$ merupakan jumlah simpul pada graf.

## *C. NetworkX Sebagai Salah Satu Pustaka Python*

*NetworkX* merupakan salah satu pustaka Python untuk membuat, menganalisis, dan memanipulasi struktur data yang berkaitan dengan jaringan kompleks. *NetworkX* mencakup data

struktur, fungsi primitif, dan alat visualisasi untuk graf. Dalam penelitian ini, *NetworkX* digunakan untuk merepresentasikan ruang ujian sebagai sebuah graf, di mana simpul mewakili peserta ujian, dan sisi menggambarkan kedekatan atau ketetanggaan antara peserta ujian.

### *D. Seleksi Nasional Berbasis Tes*

Seleksi Nasional Berbasis Tes (SNBT) merupakan ujian tahunan yang dilaksanakan dalam rangka penerimaan mahasiswa baru perguruan tinggi Negeri di Indonesia. SNBT mencakup pengujian terhadap Penalaran Umum, Pemahaman Bacaan dan Menulis, Pengetahuan Kuantitatif, Pengetahuan dan Pemahaman Umum, Penalaran Kuantitatif, Penalaran Verbal, dan Pengetahuan Bahasa Inggris (Cerebrum, 2024).

Berdasarkan informasi dari salah satu peserta SNBT 2024, pelaksanaan ujian dilakukan dengan komputer (*computerized*) dengan setiap ruangan terdiri atas 20 sampai 30 peserta. Setiap peserta duduk bersebelahan (tanpa jarak satu kursi kosong) dengan sekat pembatas di kiri, kanan, dan depan setiap kursi. Hal tersebut sesuai dengan spesifikasi ruangan SNBT di Universitas Negeri Malang, yaitu 20 orang per ruangan.

## III. Pembahasan

### *A. Analisis dan Pendekatan Tipe Ruang Ujian*

Pada penelitian ini, ruang ujian (tempat pelaksanaan SNBT) dimodelkan sebagai sebuah graf tidak berarah yang terdiri atas peserta ujian sebagai simpul dan kedekatan dua peserta ujian sebagai sisi. Dalam memodelkan sisi, dipilih dua tingkat keamanan (disebut sebagai tipe ruangan 1 dan tipe ruangan 2) dengan spesifikasi sebagai berikut.

*1. Tipe Ruangan 1*

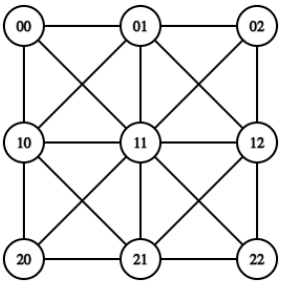


*Fig 3.1 Tipe Ruangan 1 untuk m = n = 3*
*Dibuat dengan csacademy (csacademy.com) dengan bantuan NetworkX (Implementasi tercantum).*

Ruangan ujian tipe 1 berbentuk persegi panjang dan terdiri dari $m \times n$ kursi. Peserta ujian dibagi menjadi 3 jenis, yaitu peserta ujung (4 orang), pinggir ($2 \times ((m-2)+(n-2))$ orang), dan tengah ($(m-2)(n-2)$ orang).

a. Peserta ujung berderajat 3, dengan tetangga sebagai berikut.
   kiri $\oplus$ kanan,
   atas $\oplus$ bawah, dan
   1 diagonal.
b. Peserta pinggir berderajat 5, dengan tetangga sebagai berikut.
   (kiri + kanan) $\oplus$ (atas + bawah),
   ((kiri $\oplus$ kanan) jika sudah ada (atas + bawah)) $\oplus$
   ((atas $\oplus$ bawah) jika sudah ada (kiri + kanan)), dan
   2 diagonal.
c. Peserta tengah berderajat 8, dengan tetangga sebagai berikut.
   kiri + kanan,
   atas + bawah,
   4 diagonal.

*2. Tipe Ruangan 2*

Ruangan ujian tipe 2 memiliki bentuk yang sama dengan ruangan tipe 1, namun memiliki sifat ketetanggan yang berbeda sebagai berikut.

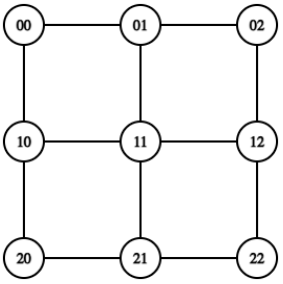


*Fig 3.2 Tipe Ruangan 2 untuk m = n = 3*
*Dibuat dengan csacademy (csacademy.com) dengan bantuan NetworkX (Implementasi tercantum).*

a. Peserta tujung berderajat 2, dengan tetangga sebagai berikut.
   (kiri $\oplus$ kanan) dan
   (atas $\oplus$ bawah).
b. Peserta pinggir berderajat 3, dengan tetangga sebagai berikut.
   (kiri + kanan) $\oplus$ (atas + bawah) dan
   ((kiri $\oplus$ kanan) jika sudah ada (atas + bawah)) $\oplus$
   ((atas $\oplus$ bawah) jika sudah ada (kiri + kanan)).
c. Peserta tengah berderajat 4, dengan tetangga sebagai berikut.
   (kiri + kanan) dan
   (atas + bawah).

### *B. Implementasi Tipe Ruang Ujian*

Kedua tipe ruang ujian diimplementasikan dalam fungsi *create_room_graph.py* sesuai spesifikasi sebelumnya, yaitu sebagai berikut.

```
def create_room_graph(m, n, room_type):

                    Inisialisasi Graf
    G = nx.Graph()

                  Tipe 1 (prosedur dalam fungsi)
    def add_edges_tipe1(x,y):
        neighbors = []
        # Tetangga Kiri
        if x>0: neighbors.append((x-1, y))
        # Tetangga Kanan
        if x<m-1: neighbors.append((x+1, y))
        # Tetangga Atas
        if y>0: neighbors.append((x,y-1))
```

```
        # Tetangga Bawah
        if y<n-1: neighbors.append((x,y+1))
        # Tetangga Diagonal kiri atas
        if x>0 and y>0:
           neighbors.append((x-1,y-1))
        # Tetangga Diagonal kanan atas
        if x<m-1 and y>0:
           neighbors.append((x+1,y-1))
        # Tetangga Diagonal kiri bawah
        if x>0 and y<n-1:
            neighbors.append((x - 1, y + 1))
        # Tetangga Diagonal kanan bawah
        if x<m-1 and y<n-1:
           neighbors.append((x + 1, y + 1))
        # Peserta Ujung
        if (x in [0, m-1] and y in [0, n-1]):
            neighbors = neighbors[:3]
        # Peserta Pinggir
        elif x in [0, m-1] or y in [0, n-1]:
             neighbors = neighbors[:5]
        for nx, ny in neighbors:
             G.add_edge((x,y), (nx,ny))
```

*Tipe 2 (prosedur dalam fungsi)*

```
def add_edges_tipe2(x, y):
    neighbors = []
    # Tetangga Kiri
    if x>0: neighbors.append((x-1, y))
    # Tetangga Kanan
    if x<m-1: neighbors.append((x+1, y))
    # Tetangga Atas
    if y>0: neighbors.append((x, y-1))
    # Tetangga Bawah
    if y<n-1: neighbors.append((x, y+1))
    # Peserta Ujung
    if (x in [0, m-1] and y in [0, n-1]):
       neighbors = neighbors[:2]
    # Peserta Pinggir
    elif x in [0, m-1] or y in [0, n-1]:
       neighbors = neighbors[:3]
    for nx, ny in neighbors:
       G.add_edge((x, y), (nx, ny))
```

*Proses Penambahan Sisi dan Mengembalikan Graf*

```
for i in range(m):
    for j in range(n):
       if room_type == 1:
          add_edges_tipe1(i, j)
       elif room_type == 2:
          add_edges_tipe2(i, j)
return G
```

## C. Pendekatan dan Implementasi Algoritma Dsatur

Algoritma Dsatur diimplementasi dalam fungsi *dsatur_algorithm* pada *dsatur_algorithm.py*. Tahapan metode implementasi direferensikan dari algoritma Dsatur menurut Brelaz (sesuai dasar teori) sebagai berikut.

```
def dsatur_algorithm(graph):
```

*Inisialisasi warna, derajat, dan saturasi*

```
    colors = {}
    degrees = {node:
len(list(graph.neighbors(node))) for node in
graph.nodes}
    saturation = {node: 0 for node in
graph.nodes}
```

*Urutkan simpul-simpul dari derajat paling tinggi ke derajat paling rendah*

```
    uncolored_nodes = sorted(graph.nodes, key =
lambda x: degrees[x], reverse = True)
```

*Tahap 2 dan iterasi berulang tahap 3-4-5 sesuai dasar teori*

```
    while uncolored_nodes:
        node = max(uncolored_nodes, key = lambda
x: (saturation[x], degrees[x]))
        used_colors = {colors[neighbor] for
neighbor in graph.neighbors(node) if neighbor
in colors}
        # Warna pertama
        color = 1
        # Pilih warna paling mendekati warna
pertama yang belum digunakan
        while color in used_colors:
            color += 1
        colors[node] = color
        uncolored_nodes.remove(node)
        # Update saturation dan derajat simpul
        for neighbor in graph.neighbors(node):
            if neighbor not in colors:
                saturation[neighbor] =
len({colors[n] for n in
graph.neighbors(neighbor) if n in colors})
```

*Mengembalikan list warna*

```
    return colors
```

Secara kompleksitas waktu (Big-O), algoritma ini memiliki kompleksitas $O(n^2)$ yang berasal dari *while-loop* dan *for-loop* di dalam *while-loop*.

## D. Implementasi Algoritma Visualisasi

Algoritma Visualisasi diimplementasi dalam fungsi *visualize_colored_graph*. Berikut merupakan implementasi kode tersebut menggunakan *matplotlib* dan *networkx*.

```
def visualize_colored_graph(graph,colors,m,n):
```

*Format posisi simpul dalam bentuk grid persegi panjang*

```
        pos = {(x,y): (y,-x) for x,y in
graph.nodes()}
```

*Format warna untuk node*

```
    node_colors = [colors[node] for node in
graph.nodes()]
```

*Menggambar Graf*

```
    plt.figure(figsize=(10,8))
    nx.draw(
        graph,
        pos,
        with_labels=True,
        node_color=node_colors,
        cmap=plt.cm.Set3,
        node_size=500,
        font_color='black',
        font_weight='bold',
    )
```

*Memastikan graf berbentuk persegi panjang*

```
    plt.axis("equal")
    plt.show()
```

## *E. Eksekusi Program*

Pada *Github Repository* (tercantum), sudah disediakan file-file program yang dapat dieksekusi untuk mengimplementasikan pewarnaan graf dengan algoritma Dsatur. Secara umum, terdapat beberapa tahapan utama untuk mengeksekusi program:

1. Mengunduh keperluan modul-modul Python
2. Menjalankan *main.py*

Secara umum, program akan meminta pengguna memasukkan tipe ruangan dan ukuran ruangan. Program akan mengeluarkan hasil berupa *list of nodes*, *list of edges*, dan hasil visualisasi pewarnaan dengan Algoritma Dsatur.

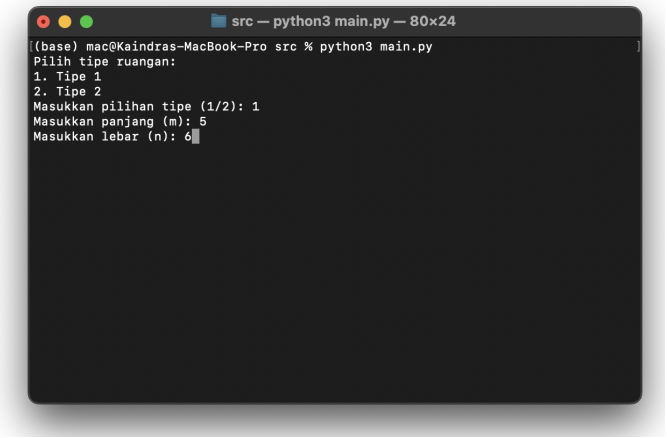


*Fig 3.3 Program meminta input pengguna*

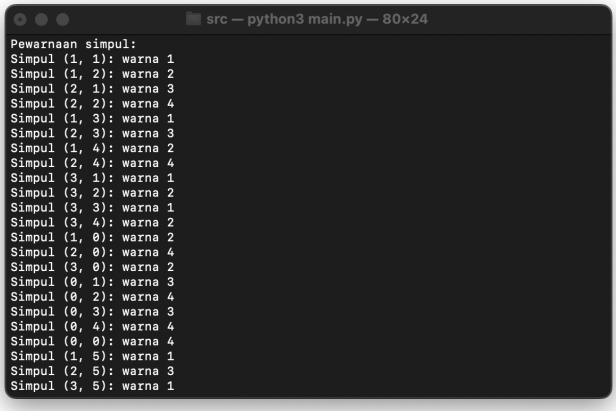


*Fig 3.4 Output hasil pewarnaan*

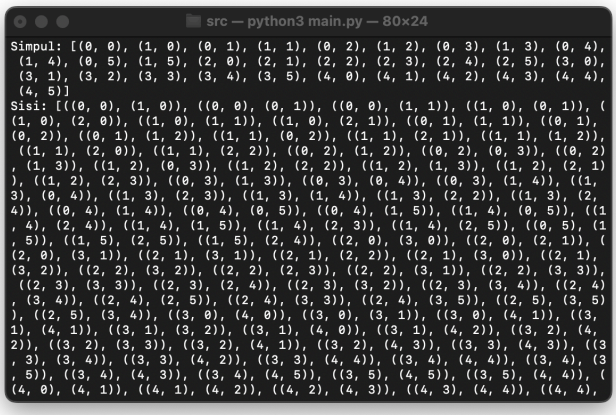


*Fig 3.5 Output list of nodes dan list of edges*

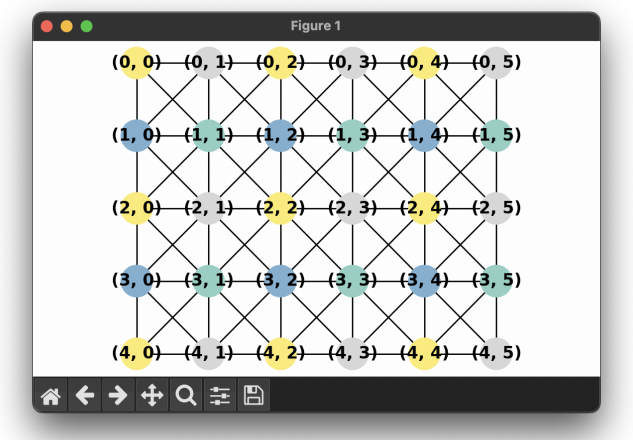


*Fig 3.6 Visualisasi*

## *F. Analisis jumlah tipe soal ujian minimum pada ruangan SNBT berdasarkan algoritma Dsatur*

Sesuai informasi pada dasar teori, setiap ruangan SNBT menampung 20–30 peserta. Oleh karena itu, digunakan model ruangan berukuran $5 \times 6$ dalam pemodelan ini.

### *1. Tipe 1*

Dengan menggunakan algoritma yang sudah dibuat, didapatkan sebuah graf yang setiap titiknya direpresentasikan dengan koordinat elemen sebuah matriks $m \times n$ di mana $m = 5$ dan $n = 6$.

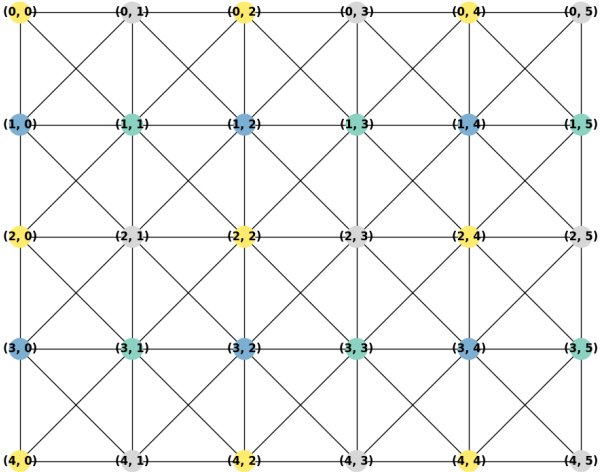


*Fig 3.3 Penerapan Algoritma Dsatur pada ruangan ujian tipe 1 yang menampung 30 peserta.*
*Dibuat dengan visualisasi NetworkX*

Dengan pendekatan tipe ruangan uji 1, dibutuhkan minimal 4 tipe soal ujian dalam satu sesi untuk memastikan peserta yang duduk berdekatan tidak mendapatkan tipe soal ujian yang sama.

*2. Tipe 2*

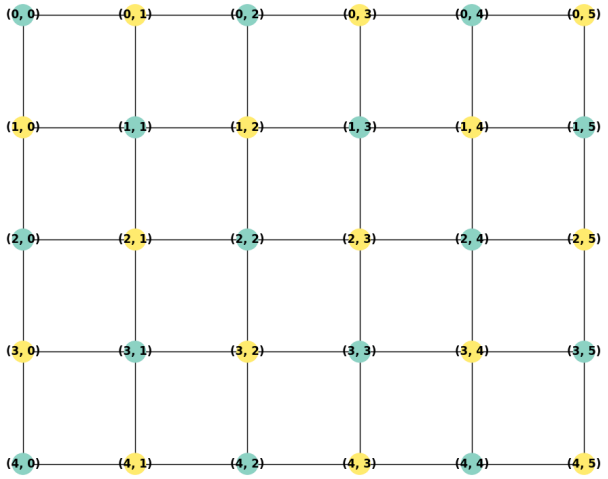


*Fig 3.3 Penerapan Algoritma Dsatur pada ruangan ujian tipe 2 yang menampung 30 peserta.*
*Dibuat dengan visualisasi NetworkX*

Dengan pendekatan tipe ruangan uji 1, dibutuhkan minimal 2 tipe soal ujian dalam satu sesi untuk memastikan peserta yang duduk berdekatan tidak mendapatkan tipe soal ujian yang sama.

Diketahui bahwa terdapat total 14 hari pelaksanaan SNBT, dengan masing-masing 2 sesi per hari. Ujian yang adil adalah ujian yang soalnya tidak dapat disebarkan oleh peserta yang sudah melaksanakan. Oleh karena itu, idealnya soal setiap sesi harus berbeda dan tidak boleh digunakan kembali pada sesi-sesi berikutnya. Oleh karena itu, jumlah paket minimum yang ideal untuk menjaga keadilan SNBT berdasarkan pendekatan penelitian ini adalah sebagai berikut.

$$\text{jumlah soal minimum} = jumlah\ sesi \times tipe\ soal\ per\ sesi$$

Dengan pendekatan tersebut, dibutuhkan 56 tipe soal untuk tipe ruangan 1 dan 112 tipe soal untuk tipe ruangan 2.

## IV. Kesimpulan

Penelitian ini berhasil membuktikan bahwa algoritma Dsatur efektif dalam pewarnaan graf, yang dicontohkan dengan implementasi dunia nyata pada Seleksi Nasional Berbasis Tes. Algoritma ini memiliki kompleksitas waktu cukup baik, yaitu $O(n^2)$ dan berhasil menentukan kondisi ideal ujian dengan memastikan peserta yang duduk berdekatan memiliki paket soal yang berbeda.

Dalam model ruangan SNBT yang digunakan, tipe ruangan 1 membutuhkan 4 paket soal untuk setiap sesi ujian. Tipe ruangan 2 hanya membutuhkan 2 paket soal per sesi. Komposisi ini menjadi cara mencegah penyebaran jawaban antar peserta dalam ruangan yang sama. Dengan pendekatan tersebut, dapat disimpulkan bahwa diperlukan 56 sampai 112 tipe soal untuk keamanan maksimal dalam penyelenggaraan 28 sesi SNBT (sesuai SNBT 2024). Dengan tipe soal saat ini yang masih berjumlah 28, masih ada ruang peningkatan bagi sistem SNBT untuk mewujudkan ujian yang lebih aman dan adil.

Penelitian ini membuktikan konsep pewarnaan graf dapat diimplementasikan dalam dunia nyata dan pada konteks ini dapat disesuaikan dengan berbagai konfigurasi kasus ruangan SNBT maupun ujian lain.

## V. Lampiran

Seluruh *source code*, data gambar, maupun sumber-sumber materi yang digunakan dalam penelitian ini disimpan di dalam sebuah *Github Repository* dengan tautan sebagai berikut.
https://github.com/MaheswaraKaindra/Dsatur-Algorithm-SNBT.git

## VI. Kata-kata Penutup

Penulis mengucapkan terima kasih kepada semua pihak yang telah membantu, baik secara langsung maupun tidak langsung, selama proses penulisan makalah ini, terutama kepada para Dosen Matematika Diskrit 2024/2025, teman-teman yang memberi inspirasi, dan kedua orang tua yang selalu mendukung.

Penulis menyadari bahwa makalah ini masih memiliki kekurangan karena ditulis dalam keadaan tidak prima. Kritik dan saran sangat diharapkan melalui repository yang telah dibuka untuk umum, sehingga penelitian ini bersifat *open source*. Semoga penelitian ini dapat menjadi inspirasi bagi adik-adik tingkat untuk terus berusaha menyelesaikan tugas dengan baik dalam segala kondisi.

## Referensi

[1] Daniel, Brelaz. 1979. *New Methods to Color The Vertices of a Graph*. Lausanne, Ecole Polytechnique de Lausanne.
[2] Munir, Rinaldi. 2024. *Graf (Bagian 1)*. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf. Diakses 1 Januari 2025.
[3] Munir, Rinaldi. 2024. *Graf (Bagian 2)*. *https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf*. Diakses 1 Januari 2025.
[4] Munir, Rinaldi. 2024. *Graf (Bagian 3)*. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/22-Graf-Bagian3-2024.pdf. Diakses 1 Januari 2025.
[5] Cerebrum. 2024. *Apa Saja 7 Subtes SNBT – Yuk Kenali Semuanya Biar Siap!* https://cerebrum.id/apa-saja-7-subtes-snbt/. Diakses 4 Januari 2024.
[6] Kompas. 2024. *Jadwal Sesi Pagi dan Siang UTBK SNBT 2024, Catat jam mulainya*. https://www.kompas.com/edu/read/2024/03/21/134747971/jadwal-sesi-pagi-dan-siang-utbk-snbt-2024-catat-jam-mulainya?page=all&utm_source=chatgpt.com. Diakses 4 Januari 2025.

## 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, 26 Desember 2024

Maheswara Bayu Kaindra - 13523015