Penerapan *Topological Sort* dalam Mengurutkan Proses Manufaktur secara Linier

Bintang Dwi Marthen - 13521144[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13521144@std.stei.itb.ac.id

***Abstract*—Saat ini manufaktur telah menjadi suatu kegiatan yang melekat pada manusia. Hampir setiap lapisan masyarakat telah melakukan manufaktur meskipun sederhana. Dalam proses manufaktur sering kali suatu tahapan menjadi prasyarat dari tahapan selanjutnya. Makalah ini membahas mengenai penerapan algoritma *topological sort* dalam mengurutkan secara linier proses manufaktur untuk mendapatkan urutan linier proses yang harus dilakukan dalam proses manufaktur tersebut.**

***Kata Kunci*—Manufaktur, Tahapan, *Topological Sort*, Urutan Linier.**

I. PENDAHULUAN

Menurut Kamus Besar Bahasa Indonesia, manufaktur adalah suatu proses mengubah bahan mentah menjadi barang untuk dapat digunakan atau dikonsumsi oleh manusia. Proses manufaktur merupakan sebuah proses yang sering dilakukan oleh masyarakat dari berbagai lapisan sosial. Salah satu contoh manufaktur yang paling sederhana adalah memasak: dalam memasak perlu disiapkan berbagai bahan masakan dalam proses yang berbeda yang kemudian baru diproses secara bersama pada tahapan tertentu menjadi makanan jadi.

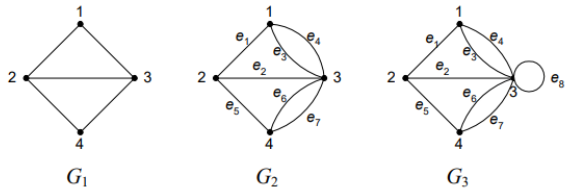
Melalui contoh memasak, diketahui bahwa dalam suatu proses manufaktur terdapat urutan pemrosesan yang perlu diperhatikan bahwa ada suatu proses yang harus dilakukan sebelum proses lainnya dapat dilaksanakan. Dengan ada persyaratan tersebut, urutan dalam pemrosesan menjadi penting sehingga bila suatu proses manufaktur yang kompleks dilaksanakan menjadi sulit bagi manusia untuk mengetahui urutannya.

Proses dalam manufaktur dapat direpresentasikan dengan suatu *operation process chart*. *Operation process chart* sendiri merupakan salah satu contoh graf berarah. Pada *operation process chart*, simpul merepresentasikan suatu proses yang dilaksanakan dalam proses manufaktur dan sisi merepresentasikan bahwa suatu proses harus dilaksanakan sebelum proses lainnya dapat dilaksanakan.

Topological sort merupakan sebuah algoritma yang digunakan untuk mengurutkan suatu proses secara linier sehingga suatu proses yang mengawali proses sebelumnya akan muncul terlebih dahulu. *Topological sort* sendiri memanfaatkan fungsi rekursif untuk membangun urutan linier dari suatu proses.

II. TEORI DASAR

A. Graf



Gambar 2.1 Graf
Sumber: Diktat Kuliah *Graf (Bag 1)*

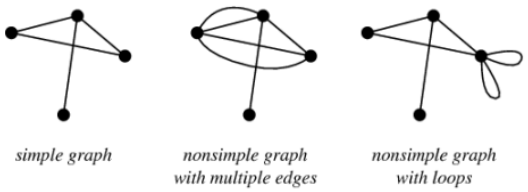
Graf dapat didefinisikan sebagai sebuah pasangan himpunan (V,E), yang dalam hal ini:

V = Himpunan tidak-kosong dari simpul-simpul (*vertices*)

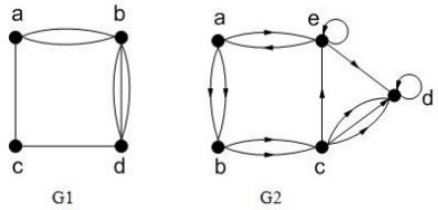
$$V = \{v_1, v_2, v_3, \ldots, v_n\}$$

E = Himpunan sisi (*edges*) yang menghubungkan sepasang simpul

$$E = \{e_1, e_2, e_3, \ldots, e_n\}$$



Gambar 2.2 Graf Sederhana, Graf Ganda, dan Graf Semu
Sumber: Diktat Kuliah *Graf (Bag 1)*

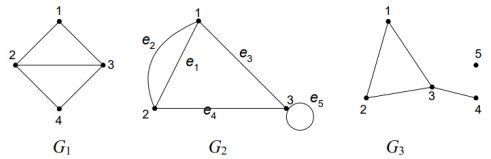


Gambar 2.3 Graf Tak-Berarah dan Graf Berarah
Sumber: Diktat Kuliah *Graf (Bag 1)*

Graf dapat dikategorikan berdasarkan ada tidaknya gelang atau sisi ganda dan orientasi arah pada sisi graf. Pengkategorian graf berdasarkan ada tidaknya gelang atau sisi ganda dibagi menjadi dua: graf sederhana dan graf tak-sederhana. Graf tak-sederhana sendiri dibagi lagi menjadi dua: graf ganda (graf yang mengandung sisi ganda) dan graf semu (graf yang mengandung

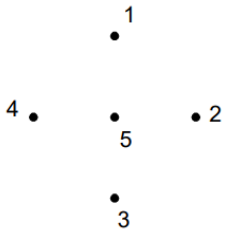
sisi gelang). Berdasarkan orientasi arah pada sisi, graf dibedakan menjadi dua: graf tak-berarah dan graf berarah.

B. Terminologi Graf



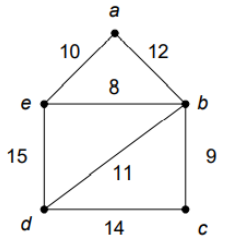
Gambar 2.4 Graf untuk Penjelasan Terminologi
Sumber: Diktat Kuliah *Graf (Bag 1)*

1. Ketetanggaan (*Adjacent*)
   Dua buah simpul dikatakan bertetangga bila keduanya terhubung secara langsung. Sebagai contoh, pada graf G_1 pada gambar 2.4, simpul 2 bertetangga dengan simpul 1, 3, dan 4.
2. Bersisian (*Incidency*)
   Sebuah sisi yang menghubungkan simpul v_i dan v_j dikatakan bersisian dengan simpul v_i dan v_j. Sebagai contoh, pada graf G_2 pada gambar 2.4, simpul e_3 bersisian dengan simpul 1 dan 3.
3. Simpul Terpencil (*Isolated Vertex*)
   Simpul terpencil adalah simpul yang tidak memiliki sisi yang bersisian dengannya. Sebagai contoh, pada graf G_3 pada gambar 2.4 terdapat sebuah simpul terpencil yaitu simpul 5.
4. Graf Kosong (*null graph* atau *empty graph*)



Gambar 2.5 Graf Kosong
Sumber: Diktat Kuliah *Graf (Bag 1)*

   Graf kosong merupakan graf yang himpunan sisinya merupakan himpunan kosong.
5. Derajat (*Degree*)
   Derajat suatu simpul adalah jumlah sisi yang bersisian dengan suatu simpul tersebut. Sebagai contoh adalah pada graf G_2 pada gambar 2.4, simpul 1 memiliki derajat 3 dan simpul 3 memiliki derajat 4.
6. Lintasan (*Path*)
   Lintasan adalah suatu barisan yang terdiri atas sisi-sisi yang menjadi jalan dari suatu simpul asal v_o ke simpul tujuan v_n. Panjang dari suatu lintasan adalah jumlah sisi dalam lintasan tersebut. Sebagai contoh, lintasan 1,3,2 pada graf G_2 pada gambar 2.4 adalah lintasan dengan barisan sisi e_3, e_4 dan memiliki panjang lintasan 2.
7. Siklus (*Cycle*) atau Sirkuit (*Circuit*)
   Siklus atau sirkuit adalah lintasan yang berawal dan berakhir pada simpul yang sama. Sebagai contoh, pada graf G_2, pada gambar 2.4, 1,3,2,1 adalah sebuah sirkuit dengan panjang 3.
8. Keterhubungan (*Connected*)
   Dua buah simpul dikatakan terhubung bila terdapat lintasan antara kedua simpul tersebut. Sebuah graf G dikatakan terhubung bila setiap pasang simpul pada graf tersebut terhubung. Bila pada suatu graf G, terdapat setidaknya satu pasang simpul yang tidak terhubung maka graf tersebut disebut sebagai graf tak-terhubung.
9. Upagraf (*Subgraph*) dan Komplemen Upagraf
   Misalkan terdapat sebuah graf $G = (V,E)$, $G_1 = (V_1, E_1)$ adalah upagraf dari G jika $V_1 \subseteq V$ dan $E_1 \subseteq E$. Komplemen dari upagraf G_1 terhadap graf G adalah graf $G_2 = (V_2, E_2)$ sehingga $E_2 = E - E_1$ dan V_2 adalah simpul-simpul yang bersisian dengan simpul-simpul pada E_2.
10. Upagraf Merentang (*Spanning Subgraph*)
    Upagraf $G_1 = (V_1, E_1)$ dari $G = (V, E)$ disebut sebagai upagraf merentang bila $V_1 = V$, yaitu graf G_1 mengandung semua simpul yang ada pada graf G.
11. *Cut-Set*
    Cut-set dari suatu graf terhubung G adalah himpunan sisi yang bila dibuang dari G akan menyebabkan G menjadi graf tidak terhubung.
12. Graf Berbobot



Gambar 2.6 Graf Berbobot
Sumber: Diktat Kuliah *Graf (Bag 1)*

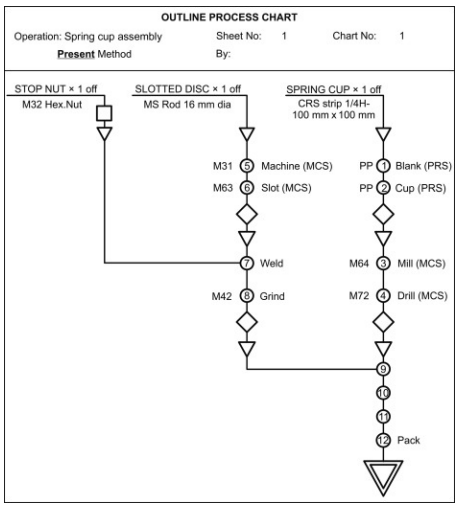
    Graf berbobot adalah graf yang setiap sisinya memiliki suatu bobot atau nilai.

C. Manufaktur

Menurut KBBI, manufaktur merupakan suatu proses mengubah bahan menjadi barang untuk dapat digunakan atau dikonsumsi oleh manusia. Manufaktur sudah ada sejak 4000 tahun sebelum Masehi, pada zaman itu manusia memanufaktur berbagai alat-alat dengan cara memalu.

Produk hasil manufaktur bisa terdiri atas satu komponen saja (contoh: paku, sekrup, dan gantungan baju), tetapi tidak jarang produk hasil manufaktur terdiri atas lebih dari satu komponen (contoh: pulpen, mesin cuci, dan mobil). Komponen-komponen dari produk tersebut dapat dimanufaktur secara paralel bila tidak memiliki ketergantungan satu sama lain.

Diagram proses manufaktur sering kali divisualisasikan dengan sebuah bagan alur. Bagan alur yang digunakan tepatnya adalah *operation process chart* yang menggambarkan rentetan tahapan yang dilakukan terhadap suatu komponen. *Operation process chart* memberikan sebuah gambaran secara umum mengenai berbagai operasi, inspeksi, maupun penyimpanan yang dilakukan kepada sebuah komponen dalam proses manufaktur sebuah produk.



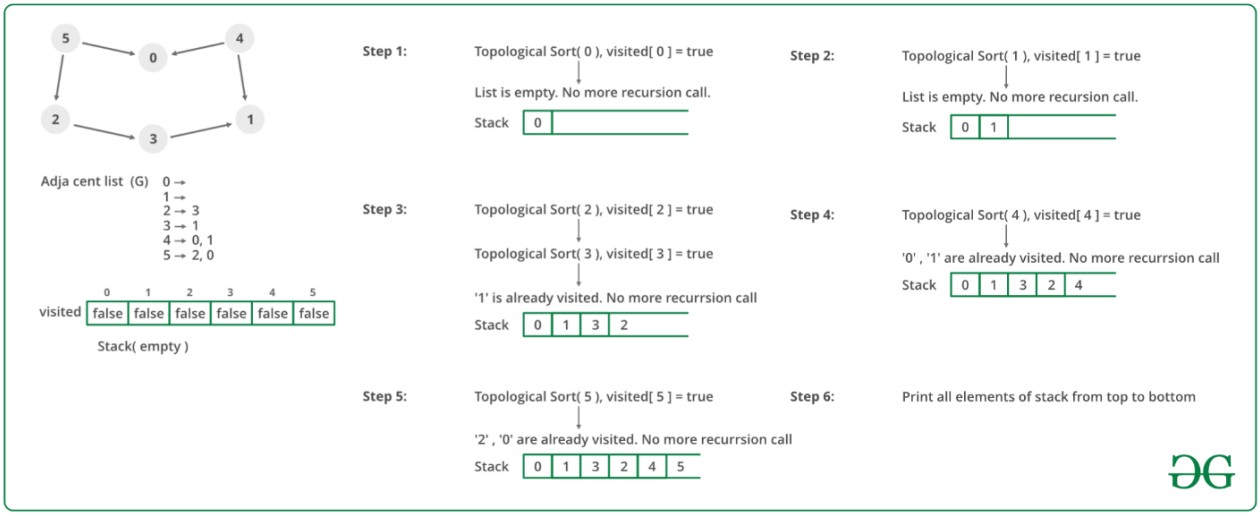
Gambar 2.7 *Operation Process Chart*
Sumber: https://www.sciencedirect.com/topics/engineering/process-chart

D. Topological Sort

Topological sort merupakan suatu pengurutan linier simpul-simpul dari suatu graf berarah asiklik untuk setiap sisi berarah *(u, v)* sehingga simpul *u* akan diposisi lebih awal dari simpul *v* dalam pengurutannya. Untuk sebuah graf, hasil pengurutan liniernya menggunakan *topological sort* tidaklah unik. Algoritma ini sering kali digunakan untuk menyelesaikan persoalan *scheduling* dari ketergantungan antar pekerjaan, sebagai contoh: pengurutan mata kuliah dalam suatu urutan linier tertentu karena terdapat mata kuliah yang menjadi prasyarat mata kuliah lain.

Pendekatan dari algoritma *topological sort* adalah sebagai berikut:

1. Sebuah *stack* kosong yang akan berisikan simpul-simpul dari graf dibuat
2. Sebuah *array of boolean* yang menandakan status keterkunjungan dari suatu simpul diinisialisasi dengan nilai *false*



Gambar 2.8 Algoritma *Topological Sort*
Sumber: *GeeksforGeeks (https://www.geeksforgeeks.org/topological-sorting/)*

3. Sebuah iterasi dari 0 hingga *V* (jumlah simpul pada graf) dilakukan untuk mengunjungi simpul-simpul pada graf
4. Bila simpul yang dikunjungi belum pernah dikunjungi, sebuah fungsi rekursif *topological sort* akan dipanggil untuk memproses simpul tersebut
5. Status keterkunjungan simpul tersebut ditandai sebagai sudah pernah dikunjungi melalui fungsi rekursi tersebut
6. Sebuah iterasi terhadap simpul-simpul yang mengarah pada simpul yang sedang diproses dilakukan, apabila simpul tersebut belum pernah dikunjungi maka panggil fungsi rekursif *topological sort* terhadap simpul tersebut
7. Simpul yang sedang diproses sekarang di-*push* ke *stack*
8. *Stack* yang dihasilkan akan menghasilkan suatu urutan linier sehingga suatu simpul *u* akan muncul sebelum simpul *v* bila terdapat sebuah sisi berarah dari *u* ke *v*.

Algoritma *topological sort* memiliki kompleksitas waktu O(V+E) dan kompleksitas memori O(V) dengan *V* adalah jumlah simpul dan *E* adalah jumlah sisi dari graf. Algoritma *topological sort* dalam bahasa pemrograman *C++* adalah sebagai berikut:

```
#include <bits/stdc++.h>
using namespace std;

// Class yang Digunakan untuk Merepresentasikan Graph
class Graph {
    // Jumlah Simpul
    int V;

    // Pointer ke Array yang Berisi Adjacency List
    list<int>* adj;

    void topologicalSortRecursive(int v, bool visited[], stack<int>& Stack);

    public:
    //Constructor dari Class Graph
    Graph(int V);

    // Fungsi untuk Menambahkan Edge ke Graph
    void addEdge(int v, int w);

    // Fungsi untuk Melakukan Topological Sort
    void topologicalSort();
};

Graph::Graph(int V){
    this->V = V;
```

```
    adj = new list<int>[V];
}

void Graph::addEdge(int v, int w){
    adj[v].push_back(w);
}

void Graph::topologicalSortRecursive(int v, bool visited[], stack<int>& Stack){
    // Menandai Node yang Sedang Dikunjungi
    visited[v] = true;

    // Melakukan Iterasi pada Adjacency List
    list<int>::iterator i;
    for (i = adj[v].begin(); i != adj[v].end(); ++i)
        if (!visited[*i])
            topologicalSortRecursive(*i, visited, Stack);

    // Memasukkan Node ke Stack
    Stack.push(v);
}

void Graph::topologicalSort(){
    stack<int> Stack;

    //Tandai semua titik sebagai belum dikunjungi
    bool* visited = new bool[V];
    for (int i = 0; i < V; i++)
        visited[i] = false;

    // Memanggil fungsi rekursif untuk semua simpul yang belum dikunjungi
    for (int i = 0; i < V; i++)
        if (visited[i] == false)
            topologicalSortRecursive(i, visited, Stack);

    // Print isi Stack
    while (Stack.empty() == false){
        cout << Stack.top() << " ";
        Stack.pop();
    }
    delete [] visited;
}
```

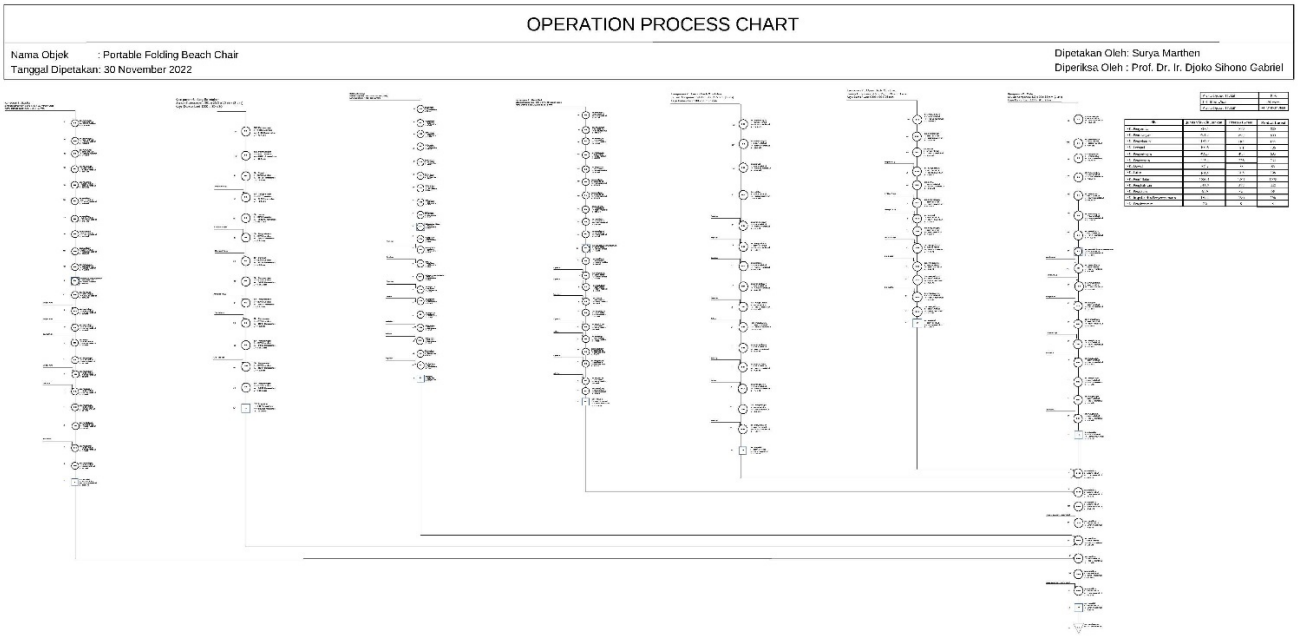
III. PENERAPAN *TOPOLOGICAL SORT* DALAM PROSES MANUFAKTUR

A. Penyusunan Graf

Operation process chart setiap proses manufaktur berbeda-beda. Makalah ini membahas menggunakan *operation process chart* untuk proses manufaktur suatu kursi pantai lipat yang terbuat dari kayu. Kursi pantai lipat tersebut dapat dilihat pada gambar di bawah ini.

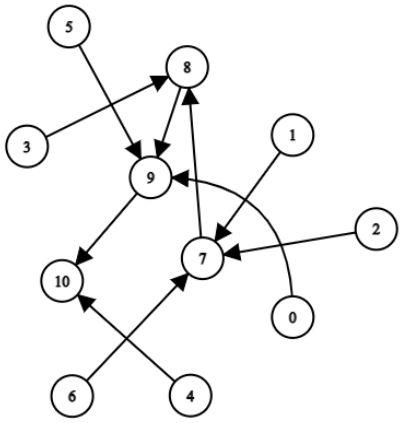
Gambar 3.1 Kursi Pantai Lipat
Sumber: Dokumentasi Surya Marthen

Dengan *operation process chart*, sebagai berikut.



Gambar 3.2 *Operation Process Chart* Kursi Pantai Lipat
Sumber: Dokumentasi Surya Marthen

Melalui *operation process chart* tersebut, dapat terlihat bahwa kursi terdiri atas tujuh komponen: *leg, lower back stretcher, stile, back rail, seat rail, leg stretcher,* dan *upper back stretcher.* Karena terdapat berbagai proses yang berjalan secara linier, *operation process chart* tersebut akan disederhanakan menjadi graf yang lebih sederhana. Graf yang telah disederhanakan tersebut adalah sebagai berikut.



Gambar 3.3 Graf Penyederhanaan *Operation Process Chart*
Sumber: Dokumentasi Penulis

Pada graf yang terdapat pada gambar 3.3, setiap simpul dienumerasi demi kemudahan dalam menjalakan algoritma *topological sort* yang telah dibuat dalam bahasa pemrograman C++. Angka-angka tersebut melambangkan berikut:

0. Pembuatan komponen *leg*
1. Pembuatan komponen *lower back stretcher*
2. Pembuatan komponen *stile*
3. Pembuatan komponen *back rail*
4. Pembuatan komponen *seat rail*
5. Pembuatan komponen *leg stretcher*
6. Pembuatan komponen *upper back stretcher*
7. Perakitan komponen *stile, lower back stretcher*, dan *upper back stretcher*
8. Perakitan komponen *back rail* dan komponen pada simpul tujuh
9. Perakitan komponen *leg, leg stretcher,* dan komponen pada simpul delapan
10. Perakitan terakhir menjadi kursi pantai lipat

Graf tersebut bila direpresentasikan dalam himpunan akan menjadi sebagai berikut.

$$G = (V,E)$$
$$V = \{0,1,2,3,4,5,6,7,8,9,10\}$$
$$E = \{(1,7),(2,7),(6,7),(3,8),(7,8),(0,9),(5,9),(8,9),(4,10),(9,10)\}$$

Selain representasi melalui himpunan, graf *G* tersebut dapat direpresentasikan dalam matriks ketetanggaan seperti di bawah.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	1	0	0	0
2	0	0	0	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	1	0
6	0	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	1	0	0	0
8	0	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0	0	0	0

Tabel 3.1 Matriks Ketetanggaan
Sumber: Dokumentasi Penulis

Selain itu, pada program digunakan representasi senarai ketetanggaan sebagai berikut.

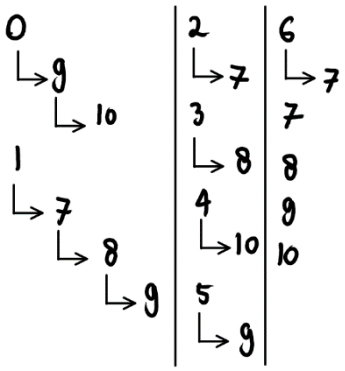
Simpul	**Simpul Tetangga**
0	9
1	7
2	7
3	8
4	10
5	9
6	7
7	8
8	9
9	10

Tabel 3.2 Senarai Ketetanggaan
Sumber: Dokumentasi Penulis

B. Membangun Urutan Proses dengan Algoritma Topological Sort

Sebelum membangun urutan linier, sebuah *stack* kosong perlu diinisialisasi sehingga isi dari *stack* di awal masihlah kosong []. Selain itu, sebuah *array* untuk menyimpan status keterkunjungan diinisialisasi juga dengan *false* (dilambangkan dengan nol).

Pada proses iterasi pertama, simpul nol akan dikunjungi pertama kali karena iterasi dilakukan terurut dari nol hingga simpul terakhir. Berikut merupakan urutan pemrosesan simpul menggunakan algoritma *topological sort*.



Gambar 3.4 Urutan Pemrosesan Simpul
Sumber: Dokumentasi Penulis

Perlu diingat bahwa penambahan simpul ke dalam *stack* dilakukan dari fungsi rekursif yang terdalam hingga ke fungsi rekursif yang terluar. Sehingga berdasarkan gambar 3.4, proses pembangunan *stack* akan menjadi sebagai berikut.

Simpul yang Sedang Dicek	**Apakah Sudah Dikunjungi**	**Isi *Stack***
10	Belum	[10]
9	Belum	[10,9]
0	Belum	[10,9,0]
9	Sudah	[10,9,0]
8	Belum	[10,9,0,8]
7	Belum	[10,9,0,8,7]
1	Belum	[10,9,0,8,7,1]
7	Sudah	[10,9,0,8,7,1]
2	Belum	[10,9,0,8,7,1,2]
8	Sudah	[10,9,0,8,7,1,2]

3	Belum	[10,9,0,8,7,1,2,3]
10	Sudah	[10,9,0,8,7,1,2,3]
4	Belum	[10,9,0,8,7,1,2,3,4]
9	Sudah	[10,9,0,8,7,1,2,3,4]
5	Belum	[10,9,0,8,7,1,2,3,4,5]
7	Sudah	[10,9,0,8,7,1,2,3,4,5]
6	Belum	[10,9,0,8,7,1,2,3,4,5,6]
7	Sudah	[10,9,0,8,7,1,2,3,4,5,6]
8	Sudah	[10,9,0,8,7,1,2,3,4,5,6]
9	Sudah	[10,9,0,8,7,1,2,3,4,5,6]
10	Sudah	[10,9,0,8,7,1,2,3,4,5,6]

Tabel 3.3 Proses Pembangunan Isi *Stack*
Sumber: Dokumentasi Penulis

Berdasarkan tabel 3.3, didapatkan sebuah *stack* dengan isi [10,9,0,8,7,1,2,3,4,5,6]. Mengingat *stack* menggunakan prinsip *last in first out*, maka hasil urutan yang dihasilkan adalah kebalikan dari isi *stack*. Oleh karena itu, urutan linier dari pemrosesan graf pada gambar 3.3 dengan algoritma *topological sort* adalah 6, 5, 4, 3, 2, 1, 7, 8, 0, 9, 10.

Melalui gambar 3.3, terlihat bahwa simpul 9 dan 4 harus muncul sebelum simpul 10, simpul 5, 7, dan 8 harus muncul sebelum simpul 9, simpul 3 dan 7 harus muncul sebelum simpul 8, dan simpul 1, 2, dan 6 harus muncul sebelum simpul 7. Dari urutan linier yang didapat, syarat-syarat tersebut terpenuhi sehingga algoritma *topological sort* memenuhi syarat bahwa suatu simpul *u* yang memiliki sisi berarah ke simpul *v* harus muncul dahulu pada urutan linier yang dihasilkan.

Berdasarkan urutan linier yang didapatkan, proses manufaktur kursi pantai lipat yang terbuat dari kayu akan menjadi:

1. Pembuatan komponen *upper back stretcher*
2. Pembuatan komponen *leg stretcher*
3. Pembuatan komponen *seat rail*
4. Pembuatan komponen *back rail*
5. Pembuatan komponen *stile*
6. Pembuatan komponen *lower back stretcher*
7. Perakitan komponen *stile, lower back stretcher,* dan *upper back stretcher*
8. Perakitan komponen *back rail* dan komponen pada simpul tujuh (*stile, lower back stretcher,* dan *upper back stretcher*)
9. Pembuatan komponen *leg*
10. Perakitan komponen *leg, leg stretcher*, dan komponen pada simpul delapan (*back rail, stile, lower back stretcher,* dan *upper back stretcher*)
11. Perakitan terakhir menjadi kursi pantai lipat

IV. Kesimpulan

Dalam suatu proses manufaktur, tahapan prosesnya sering kali dimodelkan dalam suatu *operation process chart* yang merupakan suatu graf berarah. *Operation process chart* tersebut dapat direpresentasikan dengan berbagai variasi: matriks ketetanggaan, senarai ketetanggan, himpunan simpul dan sisi, serta gambar. Proses-proses dalam manufaktur tersebut dapat diurutkan secara linier menggunakan algoritma *topological sort*. Berdasarkan *operation process chart* pada gambar 3.2 yang disederhanakan menjadi graf pada gambar 3.3, hasil dari *topological sort* yang digunakan menghasilkan urutan 6, 5, 4, 3, 2, 1, 7, 8, 0, 9, 10.

V. Lampiran

Source code dalam bahasa pemrograman C++ dapat diperoleh melalui pranala berikut: https://github.com/Marthenn/topological-sort/tree/main

VI. Ucapan Terima Kasih

Penulis mengucapkan puji syukur kepada Tuhan Yang Maha Esa karena rahmat dan karunia-Nya makalah ini dapat diselesaikan dengan baik dan tepat waktu. Penulis juga mengucapkan terima kasih kepada seluruh dosen mata kuliah IF2120 Matematika Diskrit terutama Ibu Fariska Zakhralativa Ruskanda, S.T. M.T. selaku dosen mata kuliah IF2120 Matematika Diskrit kelas K02 dan dan Pak Dr. techn. Wikan Danar Sunindyo, S.T., M.Sc. selaku dosen mata kuliah IF2110 Algoritma dan Struktur Data kelas K02 karena telah memberikan penulis ilmu untuk menyelesaikan makalah ini. Selain itu, penulis juga berterima kasih kepada Surya Marthen sebagai sumber *operation process chart* yang digunakan pada makalah ini. Tidak lupa, penulis juga mengucapkan terima kasih kepada kedua orang tua dan pihak-pihak yang membantu dalam pengerjaan makalah ini.

Referensi

[1] GeeksforGeeks. 2022. *Topological Sorting*. Diakses pada laman https://www.geeksforgeeks.org/topological-sorting/ pada tanggal 9 Desember 2022.

[2] Kalpakjian, Serope. Schmid, Steven R. 2006. *General Introduction*. Diakses pada laman https://drive.google.com/file/d/1d16FRwFr4I3ne4faW-t7u5MLpGwX6uYJ/view pada tanggal 9 Desember 2022.

[3] Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi Republik Indonesia. 2016. *Manufaktur*. Diakses pada laman https://kbbi.kemdikbud.go.id/entri/manufaktur pada tanggal 9 Desember 2022.

[4] Kiran, D.R. 2020. *Method Study – record*. Diakses pada laman https://www.sciencedirect.com/science/article/pii/B9780128199565000078 pada tanggal 9 Desember 2022.

[5] Munir, Rinaldi. 2022. *Graf (Bag 1)*. Diakses pada laman https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf pada tanggal 9 Desember 2022.

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, 10 Desember 2022

Bintang Dwi Marthen
13521144