Penerapan Topological Sort Menggunakan DFS dan Brute Force untuk Menentukan Prosedur Langkah Terbaik

Angelica Winasta Sinsuka - 13520097
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (13520097@std.stei.itb.ac.id):

***Abstract*—Kertas ini dibuat dengan tujuan untuk mendapatkan urutan langkah pengerjaan terbaik dengan menggunakan pembobotan melalui topological sort dfs. Sortir ini dilakukan pada diagram asiklik. Untuk mencapai suatu tujuan, dibutuhkan langkah-langkah yang dikerjakan secara berturut-turut untuk membuahkan hasil. Terdapat berbagai urutan langkah pengerjaan untuk mencapai tujuan. Setiap langkah, bergantung pada langkah lain, sehingga pembobotan yang digunakan adalah jumlah jalur dan simpul yang dilewati. Eksperimen yang dihasilkan tidak sepenuhnya benar karena asumsi yang digunakan mempengaruhi output dari program (dibahas lebih lanjut pada bagian implementasi). Algoritma brute force digunakan untuk mendapatkan langkah-langkah yang paling efisien.**

Keywords— dfs;diagram asiklus, topological sort, brute force

I. PENDAHULUAN

Setiap kegiatan memiliki langkah-langkah tertentu untuk menghasilkan sesuatu yang diinginkan. Namun, ketika kegiatan tersebut diterapkan dalam suatu kelompok, pengerjaan menjadi lebih sulit karena pembagian tugas dan koordinasi yang banyak. Tentunya. ketika setiap langkah tidak independen, dipierlukan untuk mempertimbangkan hal tersebut ketika lanjut ke langkah berikutnya. Kegiatan-kegiatan tersebut dapat digunakan untuk penjadwalan, pembuatan makanan, pengerjaan proyek.

Penerapan dari langkah yang tidak independent adalah pemilihan mata kuliah yang diinginkan. Tidak semua mata kuliah prasyarat yang perlu diambil, tetapi ada juga yang perlu. Sehingga apabila seorang mahasiswa tahap pertama ingin memilih mata kuliah tahun 4, diperlukan rencana untuk memilih matakuliah yang perlu diambil agar mempelajari mata kuliah yang diinginkan.

Pelaksanaan resep makanan memenuhi persyaratann seperti atas, yaitu kegiatan pasti selesai (tidak ada looping) dan terdapat ketergantungan antar langkah untuk membuat makanan atau minumannya. Misalnya penghidupan pemanggang dilakukan setelah membuat dasar kue. Namun hal itu akan menyita waktu karena menunggu sampai suhu pemanggang sudah sesuai. Alangkah cepatnya kalau pemanggang sudah dihidupkan sebelum pembuatan dasar kue selesai agar waktu masak tidak lama.

Ketika ada proyek programming yang perlu diselesaikan secara berkelompok, pembagian tugas, koordinasi, dan pengerjaan menjadi aspek penting agar proyek dapat dikerjakan secara parallel atau konkuren untuk setiap fitur. Ketika membangun suatu program, diperlukan rencana-rencana yang matang sebelum dikode dan menentukan hal-hal yang berkaitan antar komponen. Misalnya untuk membangun fitur 1, fungsi 2 diperlukan dan fitur 3 memerlukan fungsi 2 untuk membangunnya. Jika suatu proyek sudah menentukan hal-hal yang bakal ada dalam program dan hal yang berkaitan, alangkah baiknya dapat mengupdate progress dengan menggunakan langkah-langkah yang sudah ditentukan sebelumnya. Langkah-langkah tersebut menjadi bentuk *timeline* dan perhitungan *progress* proyek.

Oleh karena itu, alangkah baiknya kalau dalam mengerjakan kegiatan diatas, dapat dipastikan langkah-langkah yang kita buat benar dan optimum. Pemilihan metode brute force didasarkan alasana persoalan ini merupakan ukuran yang memiliki masukan yang kecil.

II. LANDASAN TEORI

A. *Traversal Graf*

Traversal Graf merupakan pengunjungan setiap simpul secara sistematik. Penelusuran graf memiliki beberapa algoritma, yaitu pencarian melebar atau breadthe first search, pencarian mendalam atau depth first search. Penelusuran graf dilakukan untuk mencari solusi dari setiap persoalan. Persoalan tersebut direpresentasikan sebagai graf.

B. *Algoritma Pencarian Solusi*

Algoritma pencarian solusi dipisahkan menjadi 2 aspek, yaitu pencarian solusi tanpa informasi atau blind search dan pencarian solusi dengan informasi atau *informed search. Blind search* merupakan pencarian solusi yang tidak memiliki info keterangan dari solusi sedangkan *informed search* merupakan pencarian solusi yang memiliki informasi keterangan dari solusi sehingga penelusuran tidak perlu mengunjungi semua

simpul. Optimalisasi dapat menggunakan informasi tambahan dari solusi yang dapat digunakan untuk membuat fungsi heuristik bagi *informed search*. Bagi *uninformed search* pendekatan dengan heuristik tidak dapat dilakukan. Algoritma yang diimplementasikan pada *blind search* adalah DFS, BFS, Depth Limited Search, Iterative Deepening Search, dan Unform Cost Search. Sedangkan pada *informed search*, algoritma yang diterapkan adalah Best First Search, dan A*.

C. *Representasi Graf pada Proses Pencarian*

Proses pencarian solusi memilki dua pendekatan yaitu graf statis dan graf dinamis. Graf statis merupakan graf yang sudah terbentuk sebelum penelusuran biasanya graf tersebut direprensentasikan sebagai struktur data dan graf tersebut tidak akan berubah. Graf dinamis merupakan graf yang terbentuk selama penelusuran sehingga sebelum penulusuran, graf belum ada. Oleh karena itu, hasil dari graf dinamis bisa berbeda-beda tergantung dari jenis penelusuran yang dipakai.

D. *DFS*

DFS dapat digunakan untuk menelusuri struktur data pohon atau graf. Algoritma ini dimulai dari node akar atau *root node.* Bagi kasus graf, node akar dapat dipilih kemudian penelusuran dilakukan sejauh mungkin atau mendalam sebelum melakukan backtracking. DFS menggunakan informasi jika node sudah dikunjungi atau belum agar tidak menelusuri node yang sama setelah backtracking. Penelusuran dilakukan sampai tidak ada node yang belum dikunjungi. DFS tidak dapat diiterapkan pada graf yang memilik siklus. Pseudocode adalah sebagai berikut:

```
DFS(G, u)
   u.visited = true
   for each v ∈ G.Adj[u]
      if v.visited == false
         DFS(G,v)

init() {
   For each u ∈ G
      u.visited = false
    For each u ∈ G
      DFS(G, u)
}
```

sumber: https://www.programiz.com/dsa/graph-dfs

E. *Graf Asiklik Berarah*

Graf asiklik merupakan grafik yang tidak mempunyai siklik. Apabila graf tersebut tersambung semua, itu disebut pohon, sedangkan asiklik yang tidak tersambung dikenal juga dengan hutan (kumpulan dari pohon). Grafik asiklik berarah dikenal juga dengan sebutan *Directed Acylic Graph* yang merujuk ke graf berarah yang tidak memiliki siklus. Aplikasi dari DAG adalah routing di jaringan komputer, penjadwalan kerja, pemrosesan data, genealogy, dan grafik sitasi.

Graf memiliki simpul/*vertices* yang dihubungkan oleh garis. Garis tersebut disebut dengan *edges*. Pada graf berarah, setiap edge mengarah dari vertex pertama ke yang kedua. DAG selalul terurut secara *toplogical*. Setiap edge di graf, vertex awal dimulai dahulu daripada vertex edge ujung.

F. *Brute Force*

Brute Force tidak memiliki algoritma yang spesifik, tetapi digunakan untuk memecahkan persoalan secara langsung. Bruteforce bersifat sederhana, langsung, dan jelas. Algoritma bruteforce disebut juga dengan algortima yang naif. Terdapat beberapa kelemahan dan kelebihan dari algoritma bruteforce. Beberapa kelebihan dari algoritma ini adalah sebagai berikut:

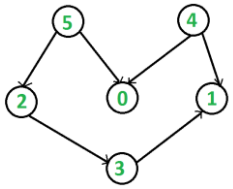
1. Algoritma brute force bisa diterapkan untuk memecahkan semua persoalan
2. Algoritma brute force mudah dipahami dan sederhana
3. Algoritma ini dapat menghasilkan algoritma yang layak seperti searching, pencocokan string.
4. Algoritma brute force menjadi standar untuk perbandingan komputasi algoritma

Beberapa kelemehan dari algoritma ini adalah sebagai berikut:

1. Algoritma brute force tidak selelau menghasilkan algoritma yang efisien
2. Algoritma brute force kebanyak lama untuk menyelesaikan masalah dengan masukan yang berukuran besar.
3. Algoritma brute force tidak kreatif untuk pemecahan masalah lain.

G. *Topological Sort*

Topological sort merupakan pengurutan linear untuk DAG atau Graf Asiklik Berarah. Pengurutan berdasarkan simpul untuk semua edge berarah u ke v (vertex u lebih dahulu sebelum v). Topological sort tidak dapat digunakan untuk graf lain kecuali DAG. Misalkan mempunyai graf sebagai berikut



sumber: https://www.geeksforgeeks.org/topological-sorting/

Hasil yang dapat dihasilkan dari hasil pengurutan seperti berikut 5→4→2→3→1→0, 5→2→3→4→0→1, dan yang lain-lain.

H. *Topological Sort Menggunakan DFS*

Topological sort metode ini mengimpelementasikan DFS untuk memproses semua node tetapi hasil berbentuk

backtracking dari DFS. Langkah-langkah dari DFS adalah sebagai berikut:

1. Membuat stack sementara
2. Topological sort dipanggil secara rekursif kemudian dipush ke dalam stack dan ditandai *visited* ketika tujuan simpul semua dari simpul asal sudah masuk stack.
3. Isi stack diprint

```
function Sort(G)
        visit ← []
        stack ← []
        in ← indegree setiap vertex jadi 0
        for each v ∈ V do
                for each u adjacent to v do
                        increment in[v] {menambah
jumlah adjacent atau indegree v}
        for each v ∈ V do
                if in[v] = 0 then
                        add v to Z
while S is not empty do
         v ← stack.remove
        append v to visit
        for each u adjacent to v do
                decrement in[u]
                if in[u] = 0 then
                        add u to visit
                         return T
```

sumber: http://ozark.hendrix.edu/~yorgey/382/static/topsort.pdf

III. IMPLEMENTASI

Solusi dari persoalan ini dilakukan berdasarkan pembobotan setiap vertex. Semakin banyak vertex yang bergantung pada suatu vertex, maka nilai pembobotan vertex tersebut semakin tinggi. Asumsi yang digunakan dari eksperimen ini bahwa vertex tujuan memiliki nilai bobot 10. Apabila terdapat vertex dependent terhadap vertex tersebut, nilai bobot bertambah 10. Jika sudah mencapai vertex independen, maka hasil perjalanan nilai bobot akan dikali dengan jumlah vertex yang dilewati. Hal tersebut dilakukan agar faktor jumlah jalur dan vertex yang dilewati dipertimbangkan.

```java
public int recursiveScore(int[] degree, int access) {
    int count = 1;
    if (access == degree.length-1) {
        hash.get(access).setScore((10*count));
        return 10;
    }
    else {
        int x = 0;
        for (int adjecent :this.adjListArray[access]) {
            x = recursiveScore(degree, adjecent);
            count++;
            hash.get(access).setScore((x+10)*count);
        }
        return 10 + x;
    }
}
```

Gambar 1.1 Prosedur perhitungan pembobotan

Setelah setiap vertex memiliki pembobotannya, maka hasil semua sortir topological akan diproses dengan mengambil nilai selisih pembobotan setiap langkah terkecil dengan nilai langkah prosedur awal terbesar.

```java
// Fungsi untuk menentukan langkah efisiensi tertinggi berdasarkan scorenya...
public void efisiensiLangkah() {
    ArrayList<Integer> result = new ArrayList<>();
    int next = 0;
    int count = 0;
    // Hasil result diisi dengan 0
    for (int i = 0 ; i < solution.size(); i++) {
        result.add(e: 0);
    }
    // jika score terurut menurun dengan selisih kecil, efisiensi semakin tinggi
    for (ArrayList<Integer> isi : solution) {
        for (int isian: isi) {
            int x = hash.get(isian).getScore();
            if (next > x) {
                result.set(count, result.get(count)+ (next - x));
            }
            next = hash.get(isian).getScore();
        }
        count++;
        next = 0;
    }
    // Tentukan indeks rangking langkah terbaik yaitu yang terkecil
    int resultPerbandingan = result.get(index: 0);
    for (int i = 1; i < result.size(); i++) {
        if (resultPerbandingan > result.get(i)) {
            resultPerbandingan = result.get(i);
        }
    }
    // Keluarkan hasil solutionnya yang paling efisien
    System.out.println(x: "Hasil Langkah Terbaik");
    int numbering = 1;
    for (int i = 0; i < result.size(); i++ ) {
        if (resultPerbandingan == result.get(i)) {
            System.out.print(numbering + ". ");
            solution.get(i).forEach(j -> System.out.print(j + " "));
            System.out.println();
            numbering+=1;
        }
    }
}
```

Gambar 2.1 Prosedur Pengambilan Langkah Terbaik

A. Deskripsi Data

Data yang akan digunakan untuk eksperimen ini ada 3, yaitu:

1. Langkah-langkah pembuatan *cheese cake*
2. Memilih mata kuliah ITB IF4050 Pengembangan Sistem IoT
3. Proses pembuatan prototype perangkat lunak yang memiliki 2 fitur

Berikut merupakan Siklus Asiklik Berarah Data diatas:

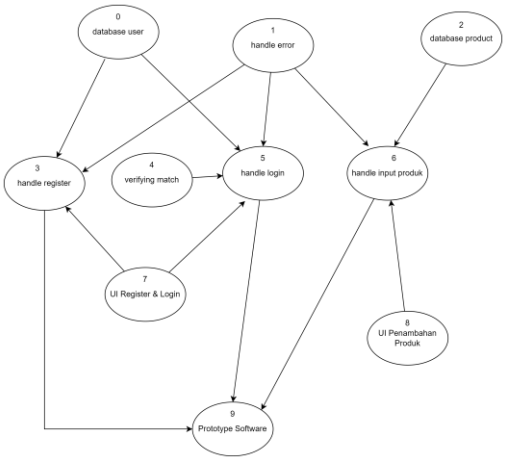
1. Langkah-langkah pembuatan cheese cake

Gambar 3.1 Graf Asiklik Berarah Pembuatan Cheesecake

2. Prasyarat memilih mata kuliah ITB IF4050 Pengembangan system IoT

Gambar 3.2 Graf Asiklik Berarah Pengambilan Mata kuliah IF4050

3. proyek suatu rekayasa perangkat lunak yang memiliki 3 fitur



Gambar 3.3 Graf Asiklik Berarah Perencanaan Pembuatan Prototype Software

B. Hasil Eksperimen dan Disksusi

1. Eksperimen data 1 dan pembahasan hasil output

Setelah memasukkan data pengujian satu pada program utama didapatkan hasil sebagai berikut

```
0 score: 1960
1 score: 1690
2 score: 1440
3 score: 1210
4 score: 1000
5 score: 810
6 score: 1920
7 score: 810
8 score: 1440
9 score: 1210
10 score: 1000
11 score: 810
12 score: 1470
13 score: 360
14 score: 1440
15 score: 490
16 score: 1000
17 score: 640
18 score: 160
19 score: 40
Results: 32340
```

```
Hasil Langkah Terbaik
1. 0 1 2 8 3 9 4 10 5 7 11 15 6 12 14 16 17 13 18 19
2. 0 1 2 8 3 9 4 10 5 7 15 11 6 12 14 16 17 13 18 19
3. 0 1 2 8 3 9 4 10 5 11 7 15 6 12 14 16 17 13 18 19
4. 0 1 2 8 3 9 4 10 5 11 15 7 6 12 14 16 17 13 18 19
5. 0 1 2 8 3 9 4 10 5 15 7 11 6 12 14 16 17 13 18 19
6. 0 1 2 8 3 9 4 10 5 15 11 7 6 12 14 16 17 13 18 19
7. 0 1 2 8 3 9 4 10 7 5 11 15 6 12 14 16 17 13 18 19
8. 0 1 2 8 3 9 4 10 7 5 15 11 6 12 14 16 17 13 18 19
9. 0 1 2 8 3 9 4 10 7 11 5 15 6 12 14 16 17 13 18 19
10. 0 1 2 8 3 9 4 10 7 11 15 5 6 12 14 16 17 13 18 19
11. 0 1 2 8 3 9 4 10 7 15 5 11 6 12 14 16 17 13 18 19
12. 0 1 2 8 3 9 4 10 7 15 11 5 6 12 14 16 17 13 18 19
13. 0 1 2 8 3 9 4 10 11 5 7 15 6 12 14 16 17 13 18 19
14. 0 1 2 8 3 9 4 10 11 5 15 7 6 12 14 16 17 13 18 19
15. 0 1 2 8 3 9 4 10 11 7 5 15 6 12 14 16 17 13 18 19
16. 0 1 2 8 3 9 4 10 11 7 15 5 6 12 14 16 17 13 18 19
```

Gambar 3.4 Hasil Masukkan Data Graf Cheesecake

Berdasarkan hasil yang didapat, Gambar kiri dari gambar 3.4 menunjuk jumlah nilai bobot yang dimiliki setiap node. Gambar kanan menunjuk semua langkah-langkah yang memiliki bobot yang sama. Dari 32340 jenis langkah hanya 192 langkah yang paling efektif dengan bobot selisih paling kecil jika kedua nilai bobot menurun. Langkah 1 sampai 192 merupakan posisi hasil sorting yang didapat bukan ranking keefektifan. Jika dilihat nomor 1, didapatkan langkah prosedur sebagai berikut

1. *Preheat Oven 350F* (0)
2. Hancurkan *graham crackers* (1)
3. Gabungan *Graham Craker* dengan gula dan *butter* (2)
4. Aduk *cream cheese* dan gula dengan *speed medium-high* selama 2 menit (8)
5. Hasil gabungan graham craker ditekan ke panci bawah dengan ketebalan 9 inchi (3)
6. Menambahkan *sour cream*, vanila, jus lemon ke dalam *cream cheese mix* (9)
7. *Prebake crust* selama 8 menit (4)
8. Menambahkan telur satu persatu dalam kecepatan medium pada *cream cheese* (10)
9. *Crust* terbentuk (5)
10. Panci kue dibungkus dengan aluminium foil (7)
11. *Batch* terbentuk (11)
12. Membuat air mendidih(15)
13. Tuangkan batch ke dalam panci *crust* (6)
14. Meletakkan panci kue ke dalam panci luar kue(12)

15. Menuangkan air mendidih ke dalam panci luar kue(14)
16. Memasak cheesecake selam 55-70 menit (16) dan seterusnya.

    langkah 1 dan langkah 13 sampai 20 tidak ada perubahan posisi langkah. Namun untuk menilai suatu prosedur sudah efisien dapat dilihat dari jumlah score dari setiap vertex. vertex 0 mendapatkan bobot sebesar 280.Vertex 19 mendapatkan 40 karena vertex 19 memiliki dibutuhkan oleh 4 vertex independen. Setiap perjalanan akan bertambah 10. Salah satu jalan yang dilewati adalah 19-18-13-17-16-14-12-6-5-4-3-2-1-0 sehingga bobot semakin tinggi apabila semakin banyak vertex yang berjalan menunju vertex independen.

    Sebelumnya, prosedur langkah adalah sebagai berikut.

    1. *Preheat Oven* 350F

    2. Hancurkan *graham crackers*

    3. Gabungan *Graham Craker* dengan gula dan butter

    4. Hasil gabungan graham craker ditekan ke panci bawah dengan ketebalan 9 inchi

    5. *Prebake crust* selama 8 menit

    6. *Crust* terbentuk

    7. Aduk *cream cheese* dan gula dengan speed medium-high selama 2 menit

    8. Menambahkan sour cream, vanilla, jus lemon ke dalam cream cheese mix Menambahkan sour cream, vanilla, jus lemon ke dalam cream cheese mix

    9. Menambahkan telur satu persatu dalam kecepatan medium pada *cream cheese*

    10. *Batch* terbentuk

    11. Membuat air mendidih

    12. Tuangkan *batch* ke dalam panci crust

    13. Panci kue dibungkus dengan aluminium foil

    14. Tuangkan *batch* ke dalam panci crust

2. Eksperimen data 2 dan pembahasan hasil output

```
0 score: 250
1 score: 250
2 score: 250
3 score: 270
4 score: 270
5 score: 240
6 score: 960
7 score: 60
Results: 12
```

```
Hasil Langkah Terbaik
1. 0 1 2 6 3 4 5 7
2. 0 1 2 6 4 3 5 7
3. 0 2 1 6 3 4 5 7
4. 0 2 1 6 4 3 5 7
5. 1 0 2 6 3 4 5 7
6. 1 0 2 6 4 3 5 7
7. 1 2 0 6 3 4 5 7
8. 1 2 0 6 4 3 5 7
9. 2 0 1 6 3 4 5 7
10. 2 0 1 6 4 3 5 7
11. 2 1 0 6 3 4 5 7
12. 2 1 0 6 4 3 5 7
```

Gambar 3.5 Hasil Masukkan Data Graf Pemgambilan Mata Kuliah IF4050

Diasumsikan mata kuliah prasyarat bukan merupakan mata kuliah wajib, tetapi hanya bisa diambil pada semester tertentu. Berdasarkan hasil yang didapat, Dari 12 jenis langkah tidak ada pengurangan, sehingga ke-12 langkah tersebut adalah prosedur yang paling efektif menurut program. Dari hasil sortir diantara 1-0-2 ketika berganti posisi tidak memiliki efek apa-apa karena sama-sama memiliki bobot yang sama yaitu 100. vertex no 6,7,5 tidak berpindah.Kemudian vertex 4, 3 berpindah.

Bobot nilai 0 didapatkan dari penelusuran node 7-4-5-6-0, bobot nilai 3 didapat dari penelusuran node 7-5-3 sebanyak 3 kali. Bobot nilai 6 didapat dairi penelusuran 7-5-3-6 dan 7-5-4-5 sebanyak 3 kali

3. Eksperimen data 3 dan pembahasan hasil output

```
0 score: 180
1 score: 270
2 score: 90
3 score: 120
4 score: 90
5 score: 160
6 score: 120
7 score: 180
8 score: 90
9 score: 100
Results: 8172
```

```
Hasil Langkah Terbaik
1. 1 0 2 4 8 6 7 5 3 9
2. 1 0 2 4 8 7 5 3 6 9
3. 1 0 2 4 8 7 5 6 3 9
4. 1 0 2 8 4 6 7 5 3 9
5. 1 0 2 8 4 7 5 3 6 9
6. 1 0 2 8 4 7 5 6 3 9
7. 1 0 4 2 8 6 7 5 3 9
8. 1 0 4 2 8 7 5 3 6 9
9. 1 0 4 2 8 7 5 6 3 9
10. 1 0 4 8 2 6 7 5 3 9
11. 1 0 4 8 2 7 5 3 6 9
12. 1 0 4 8 2 7 5 6 3 9
13. 1 0 7 2 4 8 3 5 6 9
14. 1 0 7 2 4 8 3 6 5 9
15. 1 0 7 2 4 8 5 3 6 9
```

Gambar 3.6 Hasil Masukkan Data Prototype PL

```
91. 1 7 0 2 4 8 3 5 6 9
92. 1 7 0 2 4 8 3 6 5 9
93. 1 7 0 2 4 8 5 3 6 9
94. 1 7 0 2 4 8 5 6 3 9
95. 1 7 0 2 4 8 6 3 5 9
96. 1 7 0 2 4 8 6 5 3 9
97. 1 7 0 2 8 4 3 5 6 9
98. 1 7 0 2 8 4 3 6 5 9
99. 1 7 0 2 8 4 5 3 6 9
100. 1 7 0 2 8 4 5 6 3 9
101. 1 7 0 2 8 4 6 3 5 9
```

```
158. 1 8 2 4 0 7 5 6 3 9
159. 1 8 2 4 6 0 7 5 3 9
160. 1 8 2 4 6 7 0 5 3 9
161. 1 8 2 4 7 0 5 3 6 9
162. 1 8 2 4 7 0 5 6 3 9
163. 1 8 4 2 0 7 5 3 6 9
164. 1 8 4 2 0 7 5 6 3 9
165. 1 8 4 2 6 0 7 5 3 9
166. 1 8 4 2 6 7 0 5 3 9
167. 1 8 4 2 7 0 5 3 6 9
168. 1 8 4 2 7 0 5 6 3 9
```

Gambar 3.7 Hasil Gambar lanjutan Gambar 3.6

Berdasarkan gambar diatas, terdapat masing-masing nilai setiap vertiex adalah sebagai berikut. Hasil yang keluar merupakan node dengan score nilai tertinggi, yaitu vertex 1 dengan nilai bobot 270. Hal tersebut dapat terjadi karena terdapat 3 jalur yang melewati vertex 1 kemudian dikali dengan jumlah langkah yang dibutuhkan untuk sampai vertex 1. Berikut merupakan gambaran langkah prosedur no 1:

1. handle error
2. database user
3. database product
4. verifying match
5. UI Penambahan Produk
6. handle input produk
7. UI Register dan Login
8. handle login
9. handle register
10. Prototypye Software

Berdasarkan hasil di atas, langkah yang dikerjakan terlebih dahulu adalah node independen. kemudian bertahap-tahap mengabungkannya. Pembobotan vertex 0 didapat dari 9-5-0 dan 9-5-3. Sedangkan pembobotan vertex 1 didapat dari 9-5-1 dikali 3.

IV. KESIMPULAN

Solusi dari persoalan ini dilakukan berdasarkan pembobotan setiap vertex. Semakin independent suatu vertex maka semakin tinggi nilainya, semakin banyak yang vertex yang bergantung pada suatu vertex akan bertambah nilainya. Asumsi dari experiment ini, vertex paling akhir/ tujuan memiliki nilai bobot 10, apabila terdapat vertex yang dependent vertex tersebut, nilai bobot bertambah sebanyak 10. Hasil yang didapatkan oleh program belum tentu benar karena pembuktian secara matematis untuk menghitung prioritas belum terbukti. Prioritas yang digunakan pada program ini adalah prosedur yang diasumsikan terurut menurun dengan jumlah seluruh selisih antara kedua angka dalam satu solusi paling minimal.

Hasil yang didapatkan dari percobaan 1 belum tepat karena faktor eksternal seperti membuat air mendidih tidak dipertimbangkan. Kemudian dari percobaan 2, tidak ada urutan langkah baru yang dihasilkan sehingga semua langkah yang dihasilkan dari sort dianggap efisien. Dari percobaan 1, didapatkan

Berdasarkan 3 data yang digunakan, semuanya memilki kesamaan, yaitu untuk memenuhi satu tujuan, sehingga prosedur yang menghasilkan lebih dari 1 tujuan tidak dapat digunakan untuk program ini.

Asumsi yang digunakan yaitu hasil pembobotan dari topological sort menurun. Hal itu mempengaruhi hasil keluaran karena hasil pembobotan tidak terurut menurun. Sebagai alternatif, maka diambillah selisih yang paling kecil antara elemen-elemennya dengan nilai awal paling terbesar yang mengakibatkan langkah yang dihasilkan tidaklah sepenuhnya efisien. Penggunaan brute force dalam algoritma ini mengakibatkan data yang diproses sangat banyak. Terlihat dari *result* yang dihasilkan oleh program. Oleh sebab itu, pengembangan lebih lanjut agar program berjalan lebih cepat.

VIDEO LINK AT YOUTUBE

https://youtu.be/j3FkqtRbVm8

UCAPAN TERIMA KASIH

Penulis mengucapkan puji dan syukur karena sudah dapat menyelesaikan tugas ini degan tepat waktu. Penulis ucapkan terima kasih kepada Dosen mata kuliah IF 2211 terutama Dra. Masayu lelia Khodra, S.T, M.T, Dr. Ir. Rinaldi Munir, dan Dr. Nur Ulfa Maulidevi, S. T, M. Sc. yang telah memberikan bimbingannya selama satu semester ini.

REFERENCES

[1] sallysbakingaddiction.com (2018, 2 Mei) Classic Cheesecake Recipe. Diakses pada 20 Mei 2022,darihttps://sallysbakingaddiction.com/classic-cheesecake/

[2] informatika.stei.itb.ac.id Algoritma Brute Force. Diakses pada 20 Mei 2022, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

[3] informatika.stei.itb.ac.id Breadth/DepthFirst Search(BFS.DFS). Diakses pada 20 Mei 2022, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf

[4] iq.opengenus.org Topological Sorting. Diakses pada 20 Mei 2022, dari https://iq.opengenus.org/topological-sorting-dfs/

[5] ozark.hendrix.edu Topological sorting: pseudocode and analysis. Diakses pada 21 Mei 2022, dari http://ozark.hendrix.edu/~yorgey/382/static/topsort.pdf

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 Mei 2022

Angelica Winasta Sinisuka/13520097