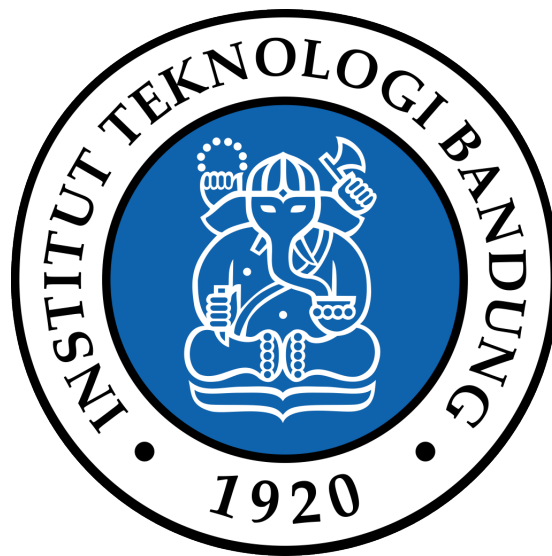


Laporan Tugas Kecil II

Implementasi *Topological Sort* dengan Menggunakan Pendekatan *Decrease and Conquer*



Nama : Josep Marcello
NIM : 13519164
Kelas : K-03
Dosen : Prof. Dwi Hendratmo Widyantoro
Bahasa : Java

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2021

1 Algoritma *Decrease and Conquer*

Proses *decrease and conquer* untuk topological sort dilakukan dengan *decrease by variable size*. Langkah-langkah algoritma *decrease and conquer*-nya adalah sebagai berikut:

1. Lakukan partisi pada graf, misal $G_1 = (V_1, E_1)$, dengan pertama-tama mencari sudut-sudut di graf yang memiliki derajat masuk 0 atau $d_{in}(V_i) = 0$.
2. Partisikan G_1 menjadi sebuah senarai dan graf baru, misal graf $G'_1 = (V'_1, E'_1)$. Senarai berisi sudut dari graf yang memiliki derajat masuk 0, sedangkan graf baru berisi sudut-sudut pada graf awal dikurangi sudut yang memiliki derajat masuk 0.
3. Lakukan kembali langkah-langkah sebelumnya untuk G'_1 sampai dengan $|V'_1| = 0$.
4. Masukkan senarai-senarai dari langkah 1–3 ke sebuah senarai dari senarai dari simpul secara berurutan dari iterasi pertama.
5. Senarai baru adalah hasil *topological sort* yang sudah diurutkan.

2 Source Code Program

Listing 1: App.java

```
1 package Uranaishi;
2
3 import Uranaishi.lib.*;
4 import java.io.FileNotFoundException;
5 import java.io.IOException;
6 import java.util.ArrayList;
7 import java.util.Scanner;
8
9 /**
10  * Berkas java utama untuk aplikasi Uranishi
11  * @author Josep Marcello
12  * 25 Februari 2021
13  */
14 public class App {
15     private static void printResult(ArrayList<ArrayList<Node>> nodes) {
16         int i = 1;
17         for (ArrayList<Node> vertList : nodes) {
18             System.out.print("Semester " + i + ": ");
19             int j = 0;
20             for (Node node : vertList) {
21                 if (j != vertList.size()-1) {
22                     System.out.print(node.getInfo() + ", ");
23                 } else {
24                     System.out.print(node.getInfo());
25                 }
26                 ++j;
27             }
28             System.out.println();
29             ++i;
30         }
31     }
32     public static void main(String[] args) throws IOException, FileNotFoundException {
33         //System.out.println("current working directory: " + System.getProperty("user.dir"));
34         Scanner scan = new Scanner(System.in);
35         Parser fp = null;
36         Graph g1 = new Graph();
37
38         // buat ngebaca argument dari user
39         if (args.length != 0) {
40             for (int i = 0; i < args.length; ++i) {
41                 if (args[i].equals("-f") || args[i].equals("--file")) {
42                     fp = new Parser(args[++i]);
43                 } else if (args[i].equals("--help") || args[i].equals("-h")) {
44                     System.out.println("uranaishi [nama jar] [--file|-f file-name]
45                     [--help|-h]");
46                     System.out.println("\t-f\t--file\tArgumen ini diikuti path ke file yang
47                     berisi data graf.");
48                     System.out.println("\t-h\t--help\tMenuiskan bantuan ini.");
49                     System.exit(0);
50                 }
51             }
52         }
```

```

50     }
51
52     // NgeBaca file kalo belum dibaca dari argumen
53     if (fp == null) {
54         System.out.print("Tuliskan path ke file yang berisi data graf: ");
55         String input = scan.nextLine();
56         fp = new Parser(input);
57     }
58
59     fp.openFile();
60     g1 = fp.parse();
61     scan.close();
62     fp.close();
63
64     System.out.println("Graf masukan:\n");
65     g1.print();
66
67     System.out.println("\n-----\n");
68     System.out.println("Hasil topological sort:\n");
69
70     long start = System.nanoTime();
71     ArrayList<ArrayList<Node>> nodes = g1.topoSort(0);
72     long elapsedTime = System.nanoTime() - start;
73     printResult(nodes);
74     System.out.println("\nWaktu untuk memproses graf: " + elapsedTime + " nanodetik.");
75     g1.print();
76 }
77 }

```

Listing 2: lib/Node.java

```

1 package Uranaishi.lib;
2
3 /**
4  * Kelas yang merepresentasikan node (sudut) pada graf
5  * @author Josep Marcello
6  * 25 Februari 2021
7  */
8 public class Node {
9     // *** attribute ***
10    private String info;
11
12    // *** Getters and setters ***
13    /**
14     * Getter info attribute
15     * @return info (nama) node
16     */
17    public String getInfo() {
18        return this.info;
19    }
20
21    // *** Methods **
22    /**
23     * Konstruktor untuk class Node
24     * @param info isi (nama) node
25     */
26    public Node(String info) {
27        this.info = info;
28    }
29
30    /**
31     * Fungsi untuk membandingkan node "this" dengan node lain
32     * @param n2 node lain yang ingin dibandingkan dengan node "this"
33     * @return true jika kedua node sama, false jika tidak
34     */
35    @Override
36    public boolean equals(Object o) {
37        if (o == null) {
38            return false;
39        }
40        if (o.getClass() != this.getClass()) {
41            return false;
42        }
43
44        Node n2 = (Node) o;
45        return info.equals(n2.info);

```

```

46 }
47
48 /**
49  * Fungsi untuk meng-override fungsi hashCode sehingga Node dapat digunakan
50  * untuk key pada collection Map
51  */
52 @Override
53 public int hashCode() {
54     return this.info.hashCode();
55 }
56 }

```

Listing 3: lib/Graph.java

```

1  package Uranaishi.lib;
2
3  import java.util.HashMap;
4  import java.util.Iterator;
5  import java.util.ArrayList;
6
7  /**
8   * Class yang merepresentasikan DAG yg memanfaatkan adjacency list
9   * ({@link ArrayList}), {@link HashMap}, dan {@link Node}
10  * @author Josep Marcelo
11  * 25 Februari 2021
12  */
13 public class Graph {
14     // *** attribute ***
15     /// adjacency list untuk graf
16     private HashMap<Node, ArrayList<Node>> inEdges;
17
18     // *** Getters and setters ***
19
20     // *** Methods **
21     /**
22      * Konstruktor graf kosong
23      */
24     public Graph() {
25         inEdges = new HashMap<>();
26     }
27
28     /**
29      * Konstruktor graf berisi
30      * Graf yang dibentuk adalah DAG dengan sisi e dan sudut v
31      * @param v sisi-sisi pada DAG
32      * @param e informasi sudut pada DAG [src vertex, dest vertex], pasti
33      * berukuran [n][2], n sebuah integer
34      */
35     public Graph(Node[] v, Node[][] e) {
36         inEdges = new HashMap<>();
37         for (Node[] adjNodes : e) {
38             // add edge otomatis nambahin vertex kalo vertex-nya belum ada
39             addEdge(adjNodes[0], adjNodes[1]);
40         }
41     }
42
43     /**
44      * Fungsi untuk menambahkan sebuah sisi berarah antara 2 sudut. Jika sudut
45      * asal dan sudut tujuan sama, maka tidak akan dilakukan apa-apa. Selain
46      * itu, jika sudut asal tidak ada di graf, maka akan ditambahkan secara
47      * otomatis ke graf
48      * @param src sudut asal
49      * @param dest sudut tujuan
50      */
51     public void addEdge(Node src, Node dest) {
52         if (src != dest) {
53             ArrayList<Node> adjList = inEdges.get(src);
54             if (adjList == null) {
55                 adjList = new ArrayList<Node>();
56                 inEdges.put(src, adjList);
57             }
58             adjList.add(dest);
59         }
60     }
61 }
62

```

```

63  /**
64   * Fungsi untuk menambahkan sebuah sudut tanpa sisi ke graf. Jika sudut
65   * sudah ada, tidak akan dilakukan apa-apa
66   * @param n1 sudut yang akan ditambahkan ke graf
67   */
68  public void addVertex(Node n1) {
69      if (!inEdges.containsKey(n1)) {
70          inEdges.put(n1, new ArrayList<Node>());
71      }
72  }
73
74  /**
75   * Fungsi untuk menuliskan isi DAG (ditunjukkan sebagai adjacency list)
76   */
77  public void print() {
78      for (Node vert: inEdges.keySet()) {
79          ArrayList<Node> adjacentVertexes = inEdges.get(vert);
80          int adjacentVertexCount = adjacentVertexes.size();
81
82          // tulis vertex
83          if (adjacentVertexCount != 0) {
84              System.out.print(vert.getInfo() + "<-");
85          } else {
86              System.out.print(vert.getInfo());
87          }
88
89          // tulis sudut-sudut yang bertetanggaan dengan vertex
90          int i = 0;
91          for (Node adjacentVertex : adjacentVertexes) {
92              if (i++ != adjacentVertexCount-1) {
93                  System.out.print(adjacentVertex.getInfo() + "<-");
94              } else {
95                  System.out.print(adjacentVertex.getInfo());
96              }
97          }
98          System.out.println();
99      }
100  }
101
102  /**
103   * Fungsi untuk menghapus sebuah vertex dari graf
104   * @param n vertex yang ingin dihapus
105   */
106  private void removeVertex(Node n) {
107      // iterasi key-nya
108      Iterator<Node> itV = inEdges.keySet().iterator();
109      while (itV.hasNext()) {
110          Node vert = itV.next();
111          ArrayList<Node> adjVerts = inEdges.get(vert);
112
113          // kalau key-nya adalah elemen yang mau di-remove, remove vertex
114          if (vert.equals(n)) {
115              itV.remove();
116          }
117
118          // iterasi adjacency list setiap sudut
119          Iterator<Node> itN = adjVerts.iterator();
120          while(itN.hasNext()) {
121              Node adjVert = itN.next();
122              if (adjVert.equals(n)) {
123                  // hapus kalo ada vertex n di dalam adjacency list
124                  itN.remove();
125              }
126          }
127      }
128  }
129
130  /**
131   * Fungsi untuk mengurutkan graf dengan topological sort. PERHATIAN: fungsi
132   * ini akan menghapus isi graf.
133   * @return Urutan vertexes berdasarkan requirements yang sudah
134   * dipisah-pisah
135   */
136  public ArrayList<ArrayList<Node>> topoSort(int iteration) {
137      if (inEdges.isEmpty() || iteration == 10) {
138          return new ArrayList<ArrayList<Node>>();

```

```

139     }
140
141     ArrayList<Node> takenNow = new ArrayList<>();
142     ArrayList<ArrayList<Node>> ret = new ArrayList<>();
143
144     // iterasiin vertices-nya
145     for (Node vert: inEdges.keySet()) {
146         ArrayList<Node> adjVert = inEdges.get(vert);
147
148         // kalo ga ada adjacent vertex, tambahkan vertex tadi ke list
149         if (adjVert.isEmpty()) {
150             takenNow.add(vert);
151         }
152     }
153
154     // hapus vertex yang sudah "diambil"
155     // bagian decrease
156     for (Node vert : takenNow) {
157         removeVertex(vert);
158     }
159
160     ret.add(takenNow);
161     // ulangi toposort
162     // bagian conquer
163     ret.addAll(topoSort(++iteration));
164
165     return ret;
166 }
167 }

```

Listing 4: lib/Parser.java

```

1 package Uranaishi.lib;
2
3 import java.io.BufferedReader;
4 import java.io.FileNotFoundException;
5 import java.io.FileReader;
6 import java.io.IOException;
7
8 /**
9  * Class yang digunakan untuk parsing file yang mengandung data {@link Graph}
10  * menjadi {@link Graph} sungguhan
11  * @author Josep Marcello
12  * 25 Februari 2021
13  */
14 public class Parser {
15     // *** attribute ***
16     String pathToFile;
17     FileReader fileReader;
18     BufferedReader bufRead;
19
20     // *** Methods ***
21     /**
22      * Konstruktor parser
23      * @param path path ke file yang ingin di parse
24      */
25     public Parser(String path) {
26         pathToFile = path;
27     }
28
29     /**
30      * Fungsi untuk menge-parse 1 baris dari file menjadi {@link Node} untuk
31      * {@link Graph}
32      * @param line baris yang ingin di-parse menjadi {@link Node}
33      * @return array of array of {@link Node} yang mengandung sudut "utama" di
34      * indeks 0 dan sudut yang bertetangga di indeks 1. Ukurannya adalah
35      * [2][n], n = 1 jika indeks 0, n = bilangan bulat jika indeks 1
36      */
37     private Node[][] parseLine(String line) {
38         Node vertex;
39
40         // Pisahin pada ", "
41         String[] vertexesStrings = line.split(", ");
42
43         // ngehapus titik dari vertex kalo ada
44         String[] tmp = vertexesStrings[0].split("\\.");

```

```

45     if (tmp.length == 1) {
46         vertex = new Node(tmp[0]);
47     } else {
48         vertex = new Node(vertexesStrings[0]);
49     }
50
51     // ngehapus titik dari adjacent vertex terakhir
52     int i = 1;
53     int len = vertexesStrings.length;
54     Node[] adjVert = new Node[vertexesStrings.length-1];
55     for (; i < len; ++i) {
56         // ngehapus titik dari adjacent vertex terakhir
57         String[] vs = vertexesStrings[i].split("\\.");
58         if (vs.length != 0) {
59             adjVert[i-1] = new Node(vs[0]);
60         } else {
61             adjVert[i-1] = new Node(vertexesStrings[i]);
62         }
63     }
64
65     Node[][] ret = { {vertex}, adjVert };
66
67     return ret;
68 }
69
70 /**
71  * Fungsi untuk membuka file dan memasukkanya ke {@link BufferedReader}
72  * @throws FileNotFoundException {@link FileNotFoundException}
73  * @throws IOException {@link IOException}
74  */
75 public void openFile() throws FileNotFoundException, IOException {
76     FileReader = new FileReader(pathToFile);
77     bufRead = new BufferedReader(fileReader);
78 }
79
80 /**
81  * Fungsi untuk parsing file menjadi {@link Graph}
82  * @return {@link Graph} dari hasil bacaan file
83  * @throws IOException {@link IOException}
84  */
85 public Graph parse() throws IOException {
86     Graph ret = new Graph();
87
88     String line = bufRead.readLine();
89     while (line != null) {
90         if (line.length() != 0) {
91             Node[][] lineParsed = parseLine(line);
92             Node vertex = lineParsed[0][0];
93             Node[] adjVert = lineParsed[1];
94
95             ret.addVertex(vertex);
96             for (Node adj : adjVert) {
97                 ret.addEdge(vertex, adj);
98             }
99         }
100         line = bufRead.readLine();
101     }
102
103     return ret;
104 }
105
106 /**
107  * Fungsi untuk menutup {@link BufferedReader} dan {@link FileReader} pada
108  * Parser
109  * @throws IOException {@link IOException}
110  */
111 public void close() throws IOException {
112     bufRead.close();
113     fileReader.close();
114 }
115 }

```

3 Hasil Pengujian

3.1 Pengujian Pertama

```
Hasil topological sort:

Semester 1: C3
Semester 2: C1
Semester 3: C4
Semester 4: C2
Semester 5: C5

Waktu untuk memproses graf: 186600 nanodetik.
```

Figure 1: *input*

Listing 5: *output*

```
1 Hasil topological sort:
2
3 Semester 1: C3
4 Semester 2: C1
5 Semester 3: C4
6 Semester 4: C2
7 Semester 5: C5
8
9 Waktu untuk memproses graf: 115700 nanodetik.
```

3.2 Pengujian Kedua

Listing 6: *input*

```
1 Kriptografi, Matdis.
2 Kalkulus.
3 TBF0, Matdis.
4 Fisika.
5 Stima, Matdis, Kalkulus.
6 Matdis, Kalkulus.
```

```
Hasil topological sort:

Semester 1: Kalkulus, Fisika
Semester 2: Matdis
Semester 3: TBF0, Kriptografi, Stima

Waktu untuk memproses graf: 163100 nanodetik.
```

Figure 2: *input*

3.3 Pengujian Ketiga

Listing 7: *input*

```
1 MA1201, MA1101.
2 FI1201, FI1101.
3 IF1210, KU1102.
4 KU1202, KU1102.
5 KI1002, KU1011.
6 EL1200, FI1101.
```



```

7 KU1102.
8 MA1101.
9 FI1101.
10 KU1011.

```

```

Hasil topological sort:

Semester 1: MA1101, KU1011, FI1101, KU1102
Semester 2: MA1201, KI1002, FI1201, EL1200, KU1202, IF1210

Waktu untuk memproses graf: 259100 nanodetik.

```

Figure 3: *input*

3.4 Pengujian Keempat

Listing 8: *input*

```

1 MA1101.
2 FI1101.
3 KU1001.
4 KU1102.
5 KU1011.
6 KU1024.
7 MA1201, MA1101.
8 FI1201, FI1101.
9 IF1210.
10 KU1202.
11 EL1200, MA1101.
12 IF2121.
13 IF2110.
14 IF2120.
15 IF2124.
16 IF2123, MA1101.
17 IF2130.
18 IF2210, IF2110.
19 IF2211.
20 IF2220, MA1101, MA1201, IF2120.
21 IF2230.
22 IF2240.
23 IF2250.
24 IF3170, IF2121, IF2124, IF2220, IF2211.
25 IF3110, IF2210, IF2110.
26 IF3130, IF2230.
27 IF3141, IF2240, IF2250.
28 IF3150, IF2250.
29 IF3140.
30 IF3151, IF2250.
31 IF3210, IF2130, IF2110.
32 IF3270, IF3170, IF2110.
33 IF3230, IF3130.
34 IF3250, IF3150, IF2250.
35 IF3260, IF2130, IF2110, IF2123.
36 IF3280.
37 IF4090, IF3280.
38 IF4091.
39 KU2071.
40 IF4092, IF4091.
41 KU206X.
42 AS2005.

```

```

Hasil topological sort:

Semester 1: KU2071, KU1001, KU206X, KU1024, KU1102, KU1202, IF2121, IF2120, IF2240, IF2124, IF2120, AS2005, MA1101, KU1011, FI1101, IF3280, IF3140, IF4091, IF2110, IF2230, IF2130, IF2250, IF2211
Semester 2: EL1200, IF3150, IF3151, IF3130, IF2123, IF3210, MA1201, FI1201, IF4090, IF4092, IF3141, IF2210
Semester 3: IF3250, IF2220, IF3230, IF3110, IF3260
Semester 4: IF3170
Semester 5: IF3270

Waktu untuk memproses graf: 1651700 nanodetik.

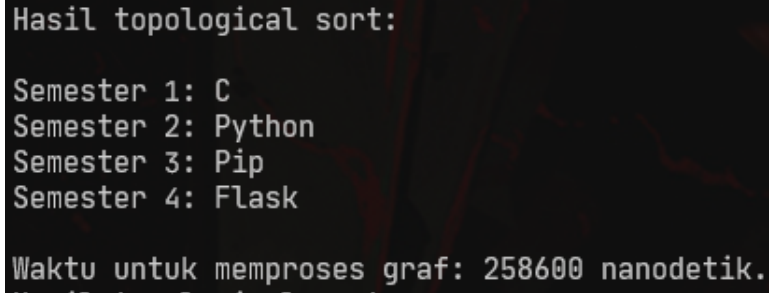
```

Figure 4: *input*

3.5 Pengujian Kelima

Listing 9: *input*

```
1 Flask, Python, Pip.  
2 Pip, Python.  
3 Python, C.  
4 C.
```



```
Hasil topological sort:  
  
Semester 1: C  
Semester 2: Python  
Semester 3: Pip  
Semester 4: Flask  
  
Waktu untuk memproses graf: 258600 nanodetik.
```

Figure 5: *input*

3.6 Pengujian Keenam

Listing 10: *input*

```
1 MA1101.  
2 FI1101.  
3 KU1001.  
4 KU1102.  
5 KU1011.  
6 KU1024.  
7  
8 MA1201, MA1101.  
9 FI1201, FI1101.  
10 IF1210, KU1102.  
11 KU1202, KU1102.  
12 KI1002, KU1011.  
13 EL1200, FI1101.  
14  
15 IF2121, IF1210, MA1101, MA1201.  
16 IF2110, KU1102, IF1210.  
17 IF2120, MA1201, MA1101.  
18 IF2124, EL1200.  
19 IF2123, MA1201.  
20 IF2130, KU1202.  
21  
22 IF2210, IF2110.  
23 IF2211, IF2110.  
24 IF2220, MA1101, MA1201, IF2120.  
25 IF2230, IF2130.  
26 IF2240, IF2121, IF2120.  
27 IF2250, KU1202, IF2110.  
28  
29 IF3170, IF2121, IF2124, IF2220, IF2211.  
30 IF3110, IF2210, IF2110.  
31 IF3130, IF2230.  
32 IF3141, IF2240, IF2250.  
33 IF3150, IF2250.  
34 IF3140, IF2240.  
35 IF3151, IF2250.  
36  
37 IF3210, IF2110, IF2130, IF3110.  
38 IF3270, IF2210, IF3170.  
39 IF3230, IF3130.  
40 IF3250, IF2250, IF3150.  
41 IF3260, IF2123, IF2110, IF2130, IF3151.  
42 IF3280, IF3151, IF3150.  
43  
44 IF4090, IF3280.  
45 IF4091, IF3280.  
46
```

```

Hasil topological sort:

Semester 1: KU1001, KU1024, KU1102, MA1101, KU1011, FI1101
Semester 2: EL1200, KU1202, IF1210, MA1201, KI1002, FI1201
Semester 3: IF2121, IF2120, IF2124, IF2123, IF2110, IF2130
Semester 4: IF2220, IF2240, IF2230, IF2250, IF2211, IF2210
Semester 5: IF3170, IF3150, IF3151, IF3130, IF3110, IF3140, IF3141
Semester 6: IF3270, IF3250, IF3230, IF3210, IF3280, IF3260
Semester 7: IF4090, IF4091
Semester 8: IF4092

Waktu untuk memproses graf: 1443100 nanodetik.

```

Figure 6: *input*

3.7 Pengujian Ketujuh

Listing 11: *input*

```

1 A.
2 full.
3 commitments.
4 what.
5 i'm.
6 thinking.
7 of.
8 You, full.
9 wouldn't, what.
10 get, A.
11 this, thinking.
12 from, what.
13 any, thinking.
14 other, of.
15 guy, A.
16 I, wouldn't.
17 just, thinking, this.
18 wanna, A, full, guy.
19 tell, You.
20 you, wouldn't, get.
21 how, wouldn't.
22 I'm, thinking, of, You.
23 feeling, You, wouldn't, commitments.
24 Gotta, get, this, feeling.
25 make, you, thinking.
26 u, wanna, tell, feeling.
27 understand, you.
28 Never, thinking, of, u.
29 gonna, make, u, understand.
30 give, u, feeling.
31 U, Gotta, understand.
32 up, of, u.
33 never, make, You, thinking, Never.
34 Gonna, make, u, understand, U, Never, give, up.
35 let, feeling, make, you, give, up.
36 yOu, wanna, tell, feeling, give, up.
37 down, Never, make.

```

```

Hasil topological sort:

Semester 1: A, thinking, i'm, of, commitments, what, full
Semester 2: other, guy, get, from, wouldn't, this, any, You
Semester 3: tell, I, how, just, you, feeling, wanna, I'm
Semester 4: Gotta, make, understand, u
Semester 5: U, Never, up, give, gonna
Semester 6: Gonna, down, let, never, yOu

Waktu untuk memproses graf: 1607700 nanodetik.

```

Figure 7: *input*

3.8 Pengujian Kedelapan

Listing 12: *input*

```

1  nodemon, chokidar, glob-parent, debug, ignore-by-default, minimatch, pstree, semver,
   supports-color, touch, undefsafe, update-notifier.
2  update-notifier, boxen, chalk, configstore, has-yarn, import-lazy, is-ci,
   is-installed-globally, is-npm, is-yarn-global, latest-version, pupa, semver-diff,
   xdg-basedir.
3  chokidar, anymatch, braces, fsevents, glob-parent, is-binary-path, is-glob, normalize-path,
   readdirp.
4  fsevents.
5  anymatch, normalize-path, picomatch.
6  normalize-path.
7  picomatch.
8  braces, fill-range.
9  fill-range, to-regex-range.
10 to-regex-range, is-number.
11 is-number.
12 glob-parent, is-glob.
13 is-glob, is-extglob.
14 is-extglob
15 is-binary-path, binary-extensions.
16 binary-extensions.
17 readdirp, picomatch.
18 debug, ms.
19 ms.
20 minimatch, brace-expansion.
21 brace-expansion, balanced-match, concat-map.
22 balanced-match.
23 concat-map.
24 ignore-by-default.
25 pstree.
26 semver.
27 supports-color, has-flag.
28 has-flag.
29 touch, nopt.
30 nopt, abbrev.
31 abbrev.
32 undefsafe, debug.
33 boxen, ansi-align, camelcase, chalk, cli-boxes, string-width, term-size, type-fest,
   widest-line.
34 camelcase.
35 ansi-align, string-width.
36 string-width, emoji-regex, is-fullwidth-code-point, strip-ansi.
37 emoji-regex.
38 is-fullwidth-code-point.
39 strip-ansi, ansi-regex.
40 ansi-regex.
41 chalk, ansi-styles, supports-color.
42 ansi-styles, @types/color-name, color-convert.
43 @types/color-name.
44 color-convert, color-name.
45 color-name.
46 has-flag.
47 cli-boxes.
48 term-size.
49 type-fest.

```

```

50 widest-line, string-width.
51 configstore, dot-prop, graceful-fs, make-dir, unique-string, write-file-atomic, xdg-basedir.
52 dot-prop, is-obj.
53 is-obj.
54 graceful-fs.
55 make-dir, semver.
56 unique-string, crypto-random-string.
57 crypto-random-string.
58 write-file-atomic, imurmurhash, is-typedarray, signal-exit, typedarray-to-buffer.
59 imurmurhash.
60 is-typedarray.
61 signal-exit.
62 typedarray-to-buffer, is-typedarray.
63 xdg-basedir.
64 has-yarn.
65 import-lazy.
66 is-ci, ci-info.
67 ci-info.
68 is-installed-globally, global-dirs, is-path-inside.
69 global-dirs, ini.
70 ini.
71 is-path-inside.
72 is-npm.
73 is-yarn-global.
74 pupa, escape-goat.
75 escape-goat.
76 semver-diff, semver.
77 latest-version, package-json.
78 package-json, got, registry-auth-token, registry-url, semver.
79 registry-url, rc.
80 rc, deep-extend, ini, minimist, strip-json-comments.
81 deep-extend.
82 minimist.
83 strip-json-comments.
84 registry-auth-token, rc.
85 got, @sindresorhus/is, @szmarczak/http-timer, cacheable-request, decompress-response,
    duplex3, get-stream, lowercase-keys, mimic-response, p-cancelable, to-readable-stream.
86 @sindresorhus/is.
87 duplex3.
88 lowercase-keys.
89 mimic-response.
90 p-cancelable.
91 to-readable-stream.
92 url-parse-lax, prepend-http.
93 prepend-http.
94 get-stream, pump.
95 pump, end-of-stream, once.
96 end-of-stream, once.
97 once, wrappy.
98 wrappy.
99 decompress-response, mimic-response.
100 cacheable-request, clone-response, get-stream, http-cache-semantics, keyv, lowercase-keys,
    normalize-url, responselike.
101 responselike, lowercase-keys.
102 normalize-url.
103 keyv, json-buffer.
104 json-buffer.
105 http-cache-semantics.
106 get-stream, pump.
107 clone-response, mimic-response.
108 mimic-response.
109 @szmarczak/http-timer, defer-to-connect.
110 defer-to-connect.

```

```

Hasil topological sort:
Semester 1: fsevents, normalize-url, emoji-regex, duplex3, has-flag, crypto-random-string, balanced-match, ignore-by-default, minimist, picomatch, ansi-regex, has-yarn, imurmurhash, type-fest, is-obj, is-number, @indresorhus/is, ci-info, import-lazy, wrappy, escape-goat, is-typedarray, is-npm, mimic-response, xdg-basedir, deep-extend, signal-exit, is-fullwidth-code-point, normalize-path, pstree, cli-boxes, lowercase-keys, graceful-fs, senver, fs, camelcase, color-name, @types/color-name, json-buffer, binary-extensions, strip-json-comments, abbrev, is-yarn-global, to-readable-stream, term-size, p-cancelable, is-path-inside, is-extglob, http-cache-semantics, ini, prepend-http, defer-to-connect, concat-map
Semester 2: to-regex-range, dot-prop, is-ci, rc, readdirp, unique-string, supports-color, is-binary-path, is-glob, make-dir, global-dirs, pupa, decompress-response, color-convert, responselike, typedarray-to-buffer, @szmarczak/http-timer, strip-ansi, url-parse-lax, once, npt, anymatch, clone-response, keyv, semver-diff, debug, brace-expansion
Semester 3: minimatch, ansi-styles, string-width, registry-auth-token, end-of-stream, registry-url, write-file-atomic, undefsafe, glob-parent, touch, fill-range, is-installed-globally
Semester 4: braces, pump, widest-line, chalk, ansi-align, configstore
Semester 5: chokidar, boxen, get-stream
Semester 6: cacheable-request
Semester 7: got
Semester 8: package-json
Semester 9: latest-version
Semester 10: update-notifier
Semester 11: nodemon
Waktu untuk memproses graf: 3788180 nanodetik.

```

Figure 8: *input*

3.9 Tabel Penilaian

Poin	Ya	Tidak
1. Program berhasil dikompilasi	✓	
2. Program berhasil <i>running</i>	✓	
3. Program dapat menerima berkas input dan menuliskan output	✓	
4. Luaran sudah benar untuk semua kasus input	✓	

Link ke repository Github

Link ke repository: <https://github.com/jspmarc/Uranaishi>.