TUGAS AKHIR – IF184802

# DESAIN DAN ANALISIS ALGORITMA PEMROGRAMAN DINAMIS MODEL TREE PADA PENYELESAIAN PERMASALAHAN URI ONLINE JUDGE 2911 INK COLORS

MUHAMMAD FARRAS RAHMATULLAH
05111640000050

Dosen Pembimbing I
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing II
Yudhi Purwananto, S.Kom., M.Kom

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya
2020

**[*Halaman ini sengaja dikosongkan*]**

TUGAS AKHIR – IF184802

# DESAIN DAN ANALISIS ALGORITMA PEMROGRAMAN DINAMIS MODEL TREE PADA PENYELESAIAN PERMASALAHAN URI ONLINE JUDGE 2911 INK COLORS

MUHAMMAD FARRAS RAHMATULLAH
05111640000050

Dosen Pembimbing I
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing II
Yudhi Purwananto, S.Kom., M.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya
2020

**[*Halaman ini sengaja dikosongkan*]**

**UNDERGRADUATE THESIS – IF184802**

# DESIGN AND ANALYSIS OF DYNAMIC PROGRAMMING ALGORITHM FOR SOLVING URI ONLINE JUDGE'S 2911 INK COLORS PROBLEM

MUHAMMAD FARRAS RAHMATULLAH
05111640000050

Supervisor I
Rully Soelaiman, S.Kom., M.Kom.

Supervisor II
Yudhi Purwananto, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya
2020

**[*Halaman ini sengaja dikosongkan*]**

LEMBAR PENGESAHAN

**DESAIN DAN ANALISIS ALGORITMA PEMROGRAMAN DINAMIS MODEL TREE PADA PENYELESAIAN PERMASALAHAN URI ONLINE JUDGE 2911 INK COLORS**

**TUGAS AKHIR**

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Algoritma dan Pemrograman
Program Studi S-1
Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:
**Muhammad Farras Rahmatullah**
**NRP: 051116 40000 050**

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Rully Soelaiman, S.Kom., M.Kom.
NIP. 197002131994021001

(Pembimbing 1)

Yudhi Purwananto, S.Kom., M.Kom.
NIP. 197007141997031002

(Pembimbing 2)

**SURABAYA**
**MEI 2020**

**[*Halaman ini sengaja dikosongkan*]**

**DESAIN DAN ANALISIS ALGORITMA PEMROGRAMAN DINAMIS MODEL TREE PADA PENYELESAIAN PERMASALAHAN URI ONLINE JUDGE 2911 INK COLORS**


| | | |
|---|---|---|
| Nama Mahasiswa | : | Muhammad Farras Rahmatullah |
| NRP | : | 051116 40000 050 |
| Departemen | : | Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, ITS |
| Dosen Pembimbing 1 | : | Rully Soelaiman, S.Kom., M.Kom. |
| Dosen Pembimbing 2 | : | Yudhi Purwananto, S.Kom., M.Kom. |


## *Abstrak*


*Permasalahan tugas akhir ini bermula dari adanya permasalahan Ink Colors pada situs URI Online Judge. Permasalahan tersebut menggambarkan sebuah tree yang akan dicari berapa jumlah tree berbentuk Stick Man maksimal yang ada pada tree tersebut.*

*Pemrograman dinamis adalah sebuah paradigma untuk mendapatkan nilai optimal dari beberapa kemungkinan jawaban, dimana permasalahan tersebut memiliki submasalah tumpang tindih dan struktur optimal.*

*Algoritma pemrograman dinamis pada struktur data tree dapat diimplementasikan dengan beberapa cara, salah satunya adalah dengan menggunakan pemrograman dinamis dalam proses penggabungan nilai submasalah pada chidren suatu vertex.*

*Pada tugas akhir ini, permasalahan Ink Colors ini akan didesain dan diimplementasikan dengan pendekatan pemrograman dinamis dalam proses penggabungan submasalah pada children suatu vertex. Solusi yang dikembangkan berjalan dengan kompleksitas waktu O(N) dimana N adalah jumlah vertex pada tree. Solusi yang dibuat cukup efisien dengan rata-rata waktu penyelesaian 0,0624 detik dan 0,0476 detik bila tree direpresentasikan dalam bentuk left child right sibling.*

***Kata kunci: Pemrograman Dinamis , Tree***

**DESIGN AND ANALYSIS OF DYNAMIC PROGRAMMING ALGORITHM FOR SOLVING URI ONLINE JUDGE'S 2911 INK COLORS PROBLEM**


| | |
|---|---|
| Student Name | : Muhammad Farras Rahmatullah |
| NRP | : 051116 40000 050 |
| Department | : Informatics, Faculty of Intelligent Electrical and Informatics Technology, ITS |
| First Supervisor | : Rully Soelaiman, S.Kom., M.Kom. |
| Second Supervisor | : Yudhi Purwananto, S.Kom., M.Kom. |


## *Abstract*


*This thesis starts with an Ink Colors problems on URI Online Judge. The problem describes a tree that we will try to find a maximum Stick Man shaped tree on that tree.*

*Dynamic programming is a paradigm to gain optimal value from several possible answer, where the problem has overlapping subproblem and optimal structure.*

*Dynamic programming on a tree data structure algorithm can be implemented with several ways, one of them is using dynamic programming in the process of merging subproblems value in children of each vertex.*

*In this thesis, Ink Colors problem will be designed and implemented with a dynamic programming approach in the process of merging subproblems value on each children. The time complexity of the solution is O(N), where N is the number of vertex in the tree. The solution made is quite efficient with an average completion time of 0.0624 seconds and 0.0476 seconds if the tree represented in the form of left child right sibling.*



***Keywords: Dynamic Programming, Tree***

**[*Halaman ini sengaja dikosongkan*]**

# KATA PENGANTAR

Puji syukur penulis ucapkan kepada Tuhan Yang Maha Esa atas pimpinan, penyertaan, dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

**DESAIN DAN ANALISIS ALGORITMA PEMROGRAMAN DINAMIS MODEL TREE PADA PENYELESAIAN PERMASALAHAN URI ONLINE JUDGE 2911 INK COLORS**

Pengerjaan Tugas Akhir ini dilakukan untuk memenuhi salah satu syarat meraih gelar Sarjana di Departemen Teknik Informatika Fakultas Teknologi Elektro dan Informatika Cerdas Institut Teknologi Sepuluh Nopember.

Dengan selesainya Tugas Akhir ini diharapkan apa yang telah dikerjakan penulis dapat memeberikan manfaat bagi perkembangan ilmu pengetahuan terutama di bidang teknologi informasi serta bagi diri penulis sendiri selaku peneliti.

Penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan baik secara langsung maupun tidak langsung selama penulis mengerjakan Tugas Akhir maupun selama menempuh masa studi antara lain:

1. Terimakasih kepada Tuhan Yang Maha Esa, di mana penulis masih diberi kesempatan, kesehatan, dan umur untuk menempuh kuliah di sini dan menjalani hidup dengan baik.
2. Mama dan Papa penulis yang selalu memberikan perhatian, do’a, dorongan, dan juga kasih sayang agar lebih semangat menempuh kuliah dan segera menyelesaikan Tugas Akhir ini.
3. Bapak Rully Soelaiman, S.Kom., M.Kom. selaku Dosen Pembimbing yang telah membimbing saya selama masa kuliah maupun selama penyelesaian Tugas Akhir ini, Dosen yang paling perhatian kepada saya dalam memberi ilmu, nasihat, dan motivasi selama

menempuh kuliah di Departemen Teknik Informatika ITS.

4. Bapak Yudhi Purwananto, S.Kom., M.Kom.. Selaku dosen pembimbing yang telah memberikan ilmu, dan masukan kepada penulis.
5. Yoshima Syach Putri yang telah membantu penulis dalam pembuatan buku tugas akhir.
6. Teman-teman grup “Bugar” yang telah menghibur dan memberi semangat untuk menyelesaikan penulisan ini.
7. Teman-teman angkatan 2016 Departemen Teknik Informatika ITS yang telah menemani perjuangan penulis selama masa perkuliahan.
8. Serta pihak-pihak lain yang tidak dapat disebutkan di sini yang telah banyak membantu penulis dalam penyusunan Tugas Akhir ini.

Penulis mohon maaf apabila masih ada kekurangan pada Tugas Akhir ini. Penulis juga mengharapkan kritik dan saran yang membangun untuk pembelajaran dan perbaikan di kemudian hari. Semoga melalui Tugas Akhir ini penulis dapat memberikan kontribusi dan manfaat yang sebaik-baiknya.

Surabaya, Mei 2020

Muhammad Farras Rahmatullah

# DAFTAR ISI

**LEMBAR PENGESAHAN** ..... **v**
***Abstrak*** ..... **vii**
***Abstract*** ..... **ix**
**KATA PENGANTAR** ..... **xi**
**DAFTAR ISI** ..... **xiii**
**DAFTAR GAMBAR** ..... **xvii**
**DAFTAR TABEL** ..... **xix**
**DAFTAR KODE SUMBER** ..... **xxi**
**1 BAB I PENDAHULUAN** ..... **1**
1.1 Latar Belakang ..... 1
1.2 Rumusan Masalah ..... 1
1.3 Batasan Masalah ..... 2
1.4 Tujuan ..... 2
1.5 Manfaat Tugas Akhir ..... 2
1.6 Metodologi ..... 3
1.6.1 Penyusunan Proposal Tugas Akhir ..... 3
1.6.2 Studi Literatur ..... 3
1.6.3 Desain dan Analisis ..... 3
1.6.4 Implementasi Algoritma ..... 3
1.6.5 Pengujian dan Evaluasi ..... 4
1.6.6 Penyusunan Buku Tugas Akhir ..... 4
1.7 Sistematika Penulisan ..... 4
**2 BAB II DASAR TEORI** ..... **7**
2.1 Deskripsi Umum Permasalahan ..... 7
2.2 Penjelasan Permasalahan *Ink Colors* ..... 9
2.3 Deskripsi Umum Teori ..... 12
2.3.1 Tree ..... 12
2.3.2 Dynamic Programming ..... 12
2.3.3 *Left Child Right Sibling* ..... 13
**3 BAB III DESAIN DAN ANALISIS** ..... **15**
3.1 Analisa Submasalah Optimal pada Permasalahan *Ink Colors* ..... 15

3.1.1 Analisa Level Transisi 1 ..... 16
3.1.2 Analisa Level Transisi 2 ..... 17
3.1.3 Analisa Level Transisi 3 ..... 18
3.2 Definisi dan Notasi ..... 18
3.3 Pemodelan Relasi Rekuren pada Permasalahan *Ink Colors* 19
3.3.1 Pemodelan Relasi Rekuren Level Transisi 1 ..... 19
3.3.2 Pemodelan Relasi Rekuren Level Transisi 2 ..... 20
3.3.3 Pemodelan Relasi Rekuren Level Transisi 3 ..... 23
3.4 Pemilihan Kaki dan Tangan *Stick Man* ..... 27
3.5 Pembuatan *Data Generator* untuk Uji Coba ..... 27
3.6 Deskripsi Umum Program ..... 28
3.7 Desain Algoritma ..... 28
3.7.1 Desain Fungsi *Choosebest* ..... 29
3.7.2 Desain Fungsi *Solve* ..... 30
**4 BAB IV IMPLEMENTASI ..... 35**
4.1 Lingkungan Implementasi ..... 35
4.2 Rancangan Data ..... 35
4.2.1 Data Masukan ..... 35
4.2.2 Data Keluaran ..... 37
4.3 Penggunaan *Library*, Konstanta, dan Variabel Global ..... 37
4.4 Implementasi Fungsi Main ..... 38
4.5 Implementasi Algoritma ..... 39
4.5.1 Implementasi Fungsi *ChooseBest* ..... 39
4.5.2 Implementasi Fungsi *Solve* ..... 39
**5 BAB V UJI COBA DAN EVALUASI ..... 43**
5.1 Lingkungan Uji Coba ..... 43
5.2 Skenario Uji Coba ..... 43
5.2.1 Uji Coba Kebenaran ..... 43
5.2.2 Uji Coba Kinerja ..... 44
5.3 Analisis Hasil Uji Coba ..... 47
**6 BAB VI KESIMPULAN DAN SARAN ..... 49**
6.1 Kesimpulan ..... 49

6.2 Saran.....49
**DAFTAR PUSTAKA .....51**
**BIODATA PENULIS .....53**

**[*Halaman ini sengaja dikosongkan*]**

# DAFTAR GAMBAR

Gambar 2.1 Deskripsi Permasalahan *Ink Colors* .....7
Gambar 2.2 Deskripsi Sebuah *Stick Man* .....8
Gambar 2.3 *Tree* dengan Satu *Stick Man* .....8
Gambar 2.4 Ilustrasi Penggambaran Tree .....10
Gambar 2.5 Ilustrasi Penggambaran Tree dengan satu *Stick Man* .. 10
Gambar 2.6 Ilustrasi Penggambaran Tree dengan dua *Stick Man* ... 11
Gambar 3.1 Ilustrasi Level Transisi pada *Stick Man* .....15
Gambar 3.2 Ilustrasi Level Transisi 1 dengan Banyak Badan.....16
Gambar 3.3 Ilustrasi Level Transisi 1 dengan Salah Satu Anaknya Adalah Level Transisi 1.....17
Gambar 3.4 Ilustrasi Tangan *Stick Man* Sebagai Kepala *Stick Man* Lain.....17
Gambar 3.5 Ilustrasi Kaki *Stick Man* Sebagai Kepala *Stick Man* Lain.....18
Gambar 3.6 Ilustrasi Submaslaah Level Transisi 1 .....19
Gambar 3.7 Ilustrasi Submasalah Level Transisi 2 .....21
Gambar 3.8 Ilustrasi Badan *Stick Man* yang Menjadi Kepala *Stick Man* Lain .....22
Gambar 3.9 Ilustrasi Submasalah Level Transisi 3 .....24
Gambar 3.10 Ilustrasi Perut *Stick Man* yang Menjadi Badan *Stick Man* Lain .....25
Gambar 3.11 Ilustrasi *Stick Man* Level Transisi 3 untuk *Vertex* yang Memiliki Satu *Child Vertex* atau Kurang .....26
Gambar 3.12 *Pseudocode* Pembuatan Test Case.....28
Gambar 3.13 *Pseudocode* Fungsi Main.....28
Gambar 3.14 *Pseudocode* Fungsi *ChooseBest* .....30
Gambar 3.15 *Pseudocode* Fungsi *Solve* .....30
Gambar 3.16 *Pseudocode* Fungsi Solve untuk Level Transisi 1 31
Gambar 3.17 *Pseudocode* Fungsi Solve untuk Level Transisi 2 32
Gambar 3.18 *Pseudocode* Fungsi Solve untuk Level Transisi 3 34
Gambar 4.1 Format Masukan Permasalahan *Ink Colors* .....36
Gambar 4.2 Representasi Format Masukan.....36
Gambar 4.3 Format Keluaran Permasalahan *Ink Colors* .....37
Gambar 5.1 Hasil Uji Coba Kebenaran Program .....44

Gambar 5.2 Grafik Uji Coba Pengaruh Banyaknya *Vertex* Terhadap Waktu ..... 44
Gambar 5.3 Hasil Uji Coba Kinerja Program ..... 45
Gambar 5.4 Variasi Waktu Program ..... 45
Gambar 5.5 Hasil Submit Paling Cepat..... 46
Gambar 5.6 Hasil Uji Coba Kinerja Representasi LCRS ..... 47
Gambar 5.7 Variasi Waktu Program dengan Representasi LCRS... 47
Gambar 5.8 *Tree* Uji Coba ..... 48

# DAFTAR TABEL

Tabel 4.1 Spesifikasi Lingkungan Implementasi .....35
Tabel 4.2 Daftar Variabel Global Permasalahan *Ink Colors* .....38
Tabel 5.1 Spesifikasi Lingkungan Uji Coba.....43
Tabel 5.2 Nilai Level Transisi Pada setiap *vertex* .....48

**[*Halaman ini sengaja dikosongkan*]**

# DAFTAR KODE SUMBER

Kode Sumber 4.1 Potongan Kode Implementasi *Library*, Konstanta, dan Variabel Global .....37
Kode Sumber 4.2 Potongan Kode Implementasi Fungsi *Main* Permasalahan *Ink Colors*.....38
Kode Sumber 4.3 Potongan Kode Implementasi Fungsi *ChooseBest* Permasalahan *Ink Colors*.....39
Kode Sumber 4.4 Implementasi Fungsi *Solve* Permasalahan *Ink Colors*.....41

**[*Halaman ini sengaja dikosongkan*]**

# BAB I
# PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan masalah, batasan permasalahan, metodologi pembuatan tugas akhir, dan sistematika penulisan buku tugas akhir ini.

## 1.1 Latar Belakang

Mencari sebuah maksimal *tree* spesifik berbentuk pada struktur data *tree* merupakan sebuah permasalahan yang dapat diselesaikan dengan algoritma yang berdasarkan metode pemrograman dinamis. Pemrograman dinamis merupakan salah satu algoritma yang dapat memecah sebuah permasalahan menjadi submasalah yang lebih sederhana, dan menyimpan setiap solusi submasalah sehingga setiap submasalah cuma diselesaikan sekali.

Topik penelitian ini mengangkat permasalahan pada URI *Online Judge* dengan kode soal 2911 yang berjudul *Ink Colors*. Pada permasalahan ini diberikan sebuah *tree*. Tujuannya adalah untuk mencari maksimal *tree* berbentuk *Stick Man.* Untuk menyelesaikan permasalahan ini dibutuhkan pemrograman dinamis pada struktur data *tree.*

Hasil dari tugas akhir ini diharapkan dapat menentukan implementasi algoritma yang tepat dalam memecahkan permasalahan *Ink Colors* secara optimal dan diharapkan dapat memberikan kontribusi pada perkembangan ilmu pengetahuan dan teknologi informasi.

## 1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana menganalisis serta menentukan desain dan algoritma yang tepat dan optimal untuk menyelesaikan permasalahan *Ink Colors?*

2. Bagaimana mengimplementasikan algoritma yang tepat untuk menyelesaikan permasalahan *Ink Colors*?
3. Bagaimana hasil dari kinerja algoritma pemrograman dinamis pada struktur data *tree* dalam menyelesaikan permasalahan *Ink Colors*?

## 1.3 Batasan Masalah

Permasalahan yang dibahas pada Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

- Implementasi algoritma menggunakan bahasa pemrograman C++.
- Batas nilai $N$ di antara 1 sampai $10^5$, dimana $N$ merupakan jumlah *vertex* pada *tree.*
- Batas waktu yang diberikan adalah 1 detik.
- Data masukan dan data keluaran yang digunakan untuk menguji algoritma yang telah dirancang adalah *dataset* pada situs *Online Judge* URI untuk permasalahan *Ink Colors*.

## 1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah sebagai berikut:

1. Memahami desain metode pemrograman dinamis pada struktur data *tree* untuk menyelesaikan permasalahan *Ink Colors.*
2. Melakukan implementasi metode pemrograman dinamis pada struktur data *tree* dalam permasalahan permasalahan *Ink Colors.*
3. Melakukan uji coba untuk mengevaluasi implementasi metode pemrograman dinamis pada permasalahan *Ink Colors.*

## 1.5 Manfaat Tugas Akhir

Tugas Akhir ini diharapkan dapat membantu untuk memahami dan mengimplementasikan pemrograman dinamis dengan struktur data *tree* pada permasalahan *Ink Colors* serta

dapat mengimplementasikan solusi yang ada pada permasalahan lain yang ada pada dunia nyata sehingga dapat memberikan kontribusi pada pengembangan ilmu pengetahuan dan teknologi informasi.

## 1.6 Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu:

### 1.6.1 Penyusunan Proposal Tugas Akhir

Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Pada proposal ini, penulis mengajukan gagasan untuk menyelesaikan permasalahan dalam menentukan perkiraan waktu minimal dan maksimal pada studi kasus *Ink Colors*.

### 1.6.2 Studi Literatur

Pada tahap ini dilakukan pencarian informasi dan studi literatur yang relevan untuk dijadikan referensi dalam melakukan pengerjaan Tugas Akhir. Informasi didapatkan dari materi-materi yang berhubungan dengan optimasi linier dan algoritma-algoritma yang ada dalam optimasi linier. Informasi tersebut didapatkan dari buku, internet, dan materi kuliah yang berhubungan dengan metode yang akan digunakan.

### 1.6.3 Desain dan Analisis

Pada tahap ini dilakukan desain rancangan algoritma untuk memecahkan permasalahan *Ink Colors*. Luaran dari tahap ini adalah algoritma yang digunakan sebagai solusi pada permasalahan tersebut.

### 1.6.4 Implementasi Algoritma

Tahapan ini merupakan tahapan untuk membangun algoritma yang akan digunakan. Pada tahap ini dilakukan implementasi dari rancangan struktur data yang akan dimodelkan

yang sesuai dengan permasalahan. Implementasi ini dilakukan dengan menggunakan bahasa pemrograman C++.

### 1.6.5 Pengujian dan Evaluasi

Pada tahap ini dilakukan uji coba kebenaran dan kinerja solusi yang telah diimplementasikan dengan melakukan pengiriman sumber kode program ke situs penilaian *URI Online Judge* pada permasalahan yang terkait dan melihat hasil umpan baliknya.

### 1.6.6 Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam Tugas Akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

## 1.7 Sistematika Penulisan

Buku Tugas Akhir ini merupakan laporan secara lengkap mengenai Tugas Akhir yang telah dikerjakan baik dari sisi teori, rancangan, maupun implementasi sehingga memudahkan bagi pembaca dan juga pihak yang ingin mengembangkannya lebih lanjut. Sistematika penulisan buku Tugas Akhir secara garis besar antara lain:

**Bab I Pendahuluan**

Bab ini berisi penjelasan latar belakang, rumusan masalah, batasan masalah dan tujuan pembuatan Tugas Akhir. Selain itu, metodologi pengerjaan dan sistematika penulisan laporan Tugas Akhir juga dijelaskan di dalamnya.

**Bab II Dasar Teori**

Bab ini berisi penjelasan secara detil mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

**Bab III Desain dan Analisis**

Bab ini berisi penjelasan tentang rancangan dari sistem yang akan dibangun beserta *pseudocode* dari algoritma.

**Bab IV Implementasi**

Bab ini berisi implementasi dari rancangan yang telah dibuat pada bab sebelumnya. Implementasi disajikan dalam bentuk kode sumber disertai dengan penjelasannya.

**Bab V Pengujian dan Evaluasi**

Bab ini berisi penjelasan mengenai data hasil percobaan dan pembahasan mengenai hasil percobaan yang telah dilakukan.

**Bab VI Kesimpulan dan Saran**

Bab ini merupakan bab terakhir yang menjelaskan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan algoritma ke depannya.

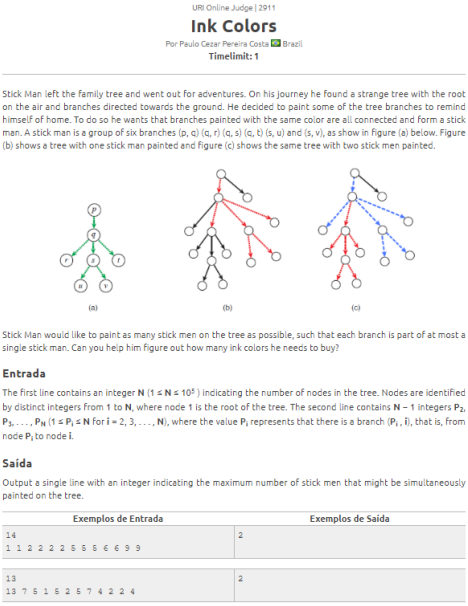
**[*Halaman ini sengaja dikosongkan*]**

# BAB II
# DASAR TEORI

Pada bab ini akan dijelaskan mengenai dasar teori yang menjadi dasar pengerjaan Tugas Akhir ini.

## 2.1 Deskripsi Umum Permasalahan

*Ink Colors* merupakan suatu permasalahan yang terdapat pada situs penilaian URI Online Judge, lihat gambar 2.1.



URI Online Judge | 2911

**Ink Colors**

Por Paulo Cezar Pereira Costa Brazil

**Timelimit: 1**

Stick Man left the family tree and went out for adventures. On his journey he found a strange tree with the root on the air and branches directed towards the ground. He decided to paint some of the tree branches to remind himself of home. To do so he wants that branches painted with the same color are all connected and form a stick man. A stick man is a group of six branches (p, q) (q, r) (q, s) (q, t) (s, u) and (s, v), as show in figure (a) below. Figure (b) shows a tree with one stick man painted and figure (c) shows the same tree with two stick men painted.

(a) (b) (c)

Stick Man would like to paint as many stick men on the tree as possible, such that each branch is part of at most a single stick man. Can you help him figure out how many ink colors he needs to buy?

**Entrada**

The first line contains an integer **N** ($1 \le N \le 10^5$) indicating the number of nodes in the tree. Nodes are identified by distinct integers from **1** to **N**, where node **1** is the root of the tree. The second line contains **N − 1** integers $P_2, P_3, \ldots, P_N$ ($1 \le P_i \le N$ for $i = 2, 3, \ldots, N$), where the value $P_i$ represents that there is a branch $(P_i, i)$, that is, from node $P_i$ to node **i**.

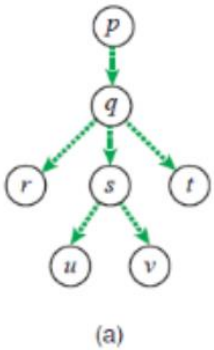
**Saída**

Output a single line with an integer indicating the maximum number of stick men that might be simultaneously painted on the tree.

| Exemplos de Entrada | Exemplos de Saída |
|---|---|
| 14<br>1 1 2 2 2 2 5 5 5 6 6 9 9 | 2 |
| 13<br>13 7 5 1 5 2 5 7 4 2 2 4 | 2 |

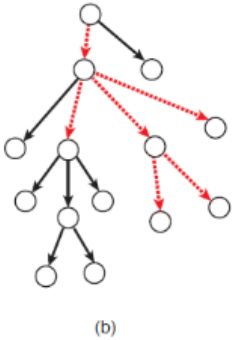
Gambar 2.1 Deskripsi Permasalahan *Ink Colors*

Permasalahan *Ink Colors* menceritakan tentang seorang *Stick Man* yang menemukan sebuah pohon aneh dengan akar yang berada di udara dan ranting yang menghadap ke tanah. Dia memutuskan untuk mewarnai beberapa batang. Dia ingin mewarnai batang-batang yang terhubung dan membentuk *Stick Man* warna yang sama. Sebuah *Stick Man* adalah kelompok 6 *edge* dan 7 *vertex* (p,q)(q,r)(q,s)(q,t)(s,u)(s,v) seperti Gambar 2.2.




Gambar 2.2 Deskripsi Sebuah *Stick Man*

Sebagai contoh, berikut adalah *tree* dengan satu *Stick Man* diwarnai.




Gambar 2.3 *Tree* dengan Satu *Stick Man*

Permasalahan yang akan diselesaikan pada kasus ini adalah mencari jumlah maksimum *Stick Man* yang mungkin secara bersamaan diwarnai pada sebuah *tree*. Pada permasalahan ini, terdapat batasan permasalahan sebagai berikut:

1. *N*, yang menunjukkan *vertex* di *tree* memiliki jumlah maksimum $10^5$. Semua *vertex* bisa diidentifikasi dengan

bilangan bulat berbeda dari satu hingga *N*, dengan *vertex* satu merupakan *root* dari *tree* tersebut.

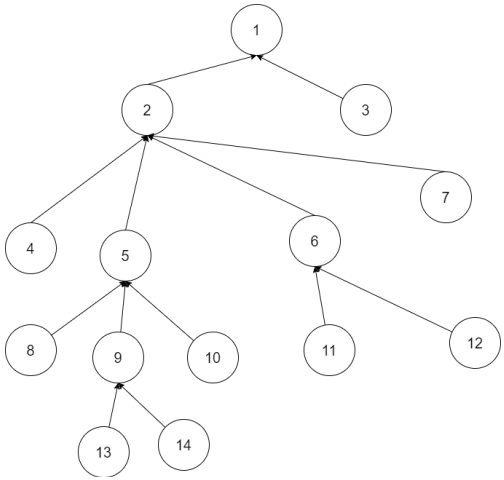
2. Dari nomor satu dan karena setiap *Stick Man* memerlukan 6 *edges*, sehingga tepat maksimal ada 16666 *Stick Man* untuk setiap *tree* yang diberikan.

Masukkan yang diberikan adalah *N* yang merepresentasikan *edge* yang ada pada *tree*. Kemudian diikuti dengan masukan *N-1* bilangan bulat $P_2, P_3, \ldots, P_N$ $(1 \leq P_i \leq N$ *for* $i = 2, 3, \ldots, N)$, dimana $P_i$ merepresentasikan adanya *edge*$(P_i, i)$ dari *vertex* $P_i$ ke *vertex i*. Keluaran dari soal ini adalah *M* banyak *Stick Man* maksimum yang bisa digambarkan

## 2.2 Penjelasan Permasalahan *Ink Colors*

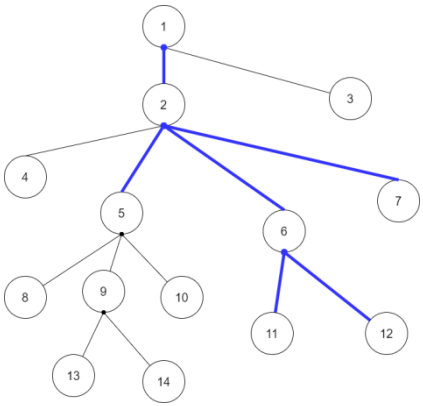
Permasalahan *Ink Colors* meminta kita untuk mewarnai sebanyak mungkin *Stick Man* pada *tree* yang diberikan. Untuk setiap *edge* yang ada pada *tree* tersebut, *edge* tersebut cuma bisa menjadi bagian dari maksimal satu *Stick Man*. Untuk Memperjelas permasalahan, kita dapat mengasumsikan bahwa *Stick Man* cuma bisa digambar dari atas ke bawah.

Misalkan diberikan *tree* dengan 14 *vertex* sesuai dengan ilustrasi pada Gambar 2.4.



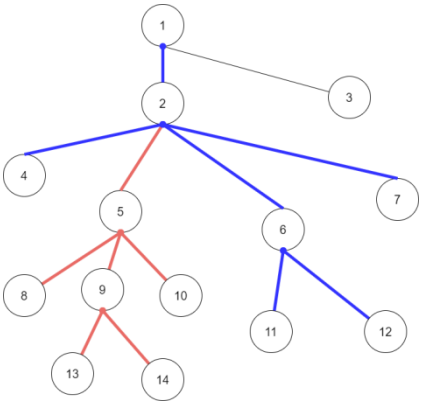

Gambar 2.4 Ilustrasi Penggambaran Tree

Selanjutnya, akan diberikan ilustrasi dimana *tree* ini akan dibuat sebuah *Stick Man* dengan gabungan *vertex* (1,2)(2,5)(2,6)(2,7)(6,11)(6,12).




Gambar 2.5 Ilustrasi Penggambaran Tree dengan satu *Stick Man*

Kemudian apabila *Stick Man* ini kita ubah gabungan *vertex*nya menjadi (1,2)(2,4)(2,6)(2,7)(6,11)(6,12), kita dapat menyisipkan *Stick Man*(2,5)(5,8)(5,9)(5,10)(9,13)(9,14) seperti Gambar 2.6 berikut.




Gambar 2.6 Ilustrasi Penggambaran Tree dengan dua *Stick Man*

Pada Gambar 2.6 adalah pewarnaan *tree* maksimum sehingga *Stick Man* yang diwarnai juga maksimum. Keluaran yang dihasilkan untuk permasalahan diatas adalah $M = 2$.

Permasalahan ini adalah suatu permasalahan optimasi, dimana tujuan dari permasalahan ini adalah mencari nilai terbaik dari semua kemungkinan yang ada. Pendekatan pemrograman dinamis dapat menyelesaikan permasalahan ini karena permasalahan ini memiliki kriteria submasalah optimal dan submasalah tumpang tindih.

Kriteria submasalah optimal terdapat pada pewarnaan *Stick Man* yang membutuhkan pemilihan *vertex Stick Man* yang tepat agar pewarnaan Stick Man maksimal. Kriteria tumpang tindih pada permasalahan ini terdapat pada nilai level transisi tiap vertex yang nantinya akan disimpan dalam memori sehingga tidak perlu dihitung ulang.

Dengan pendekatan ini, penghitungan akan dimulai dari setiap *leaf* dan naik hingga mencapai *root*. Pada setiap *vertex*

akan disimpan nilai-nilai submasalah pada *vertex* tersebut yang akan digunakan pada *vertex*-*vertex ancestor*-nya.

## 2.3 Deskripsi Umum Teori

Pada subbab ini, akan dijelaskan berbagai landasan teori secara umum yang digunakan untuk melakukan pendekatan terhadap penyelesaian permasalahan.

### 2.3.1 Tree

Dalam teori graf, sebuah *tree* adalah sebuah graf tidak terarah dimana dua *vertex* dihubungkan dengan tepat satu *edge*, atau graf asiklik tidak terarah.

Sebuah tree adalah graf *G* yang memenuhi salah satu dari kondisi berikut:

1. *G* terhubung dan asiklik.
2. *G* asiklik dan sebuah sirkuit akan terbentuk bila ditambahkan *edge* apapun.
3. *G* terhubung, tapi menjadi tidak terhubung bila *edge* manapun dihapus.
4. Dua *vertex* manapun pada *G* dapat dihubungkan dengan jalan sederhana yang unik.

### 2.3.2 Dynamic Programming

*Dynamic Programming* adalah metode optimasi yang berbeda dari rekursi (perulangan) biasa. Sebagian besar permasalahan yang membutuhkan penyelesaian dengan *Dynamic Programming* dapat dikategorikan dalam dua jenis:

*1. Optimization problems*
*2. Combinatorial problems*

*Optimization problems* mengharapkan kita untuk memilih solusi yang memungkinkan, sehingga nilai dari fungsi yang dibutuhkan dapat diminimalkan atau dimaksimalkan. *Combinatorial problems* mengharapkan kita untuk mencari tahu

berbagai macam cara untuk melakukan sesuatu atau dapat dikatakan kemungkinan dari beberapa proses yang bisa terjadi.

Biasanya implementasi dari *Dynamic Programming* adalah untuk *optimization problems*. Masalah seperti itu dapat memiliki banyak solusi yang memungkinkan, dan setiap solusi memiliki nilai. Tujuannya adalah untuk menemukan solusi dengan nilai optimal (minimal atau maksimal). Saat pengembangan algoritma *Dynamic Programming*, terdapat empat langkah proses berdasarkan definisi Cormen:

1. Melakukan identifikasi karakteristik struktur sebuah solusi yang optimal.
2. Menentukan nilai solusi optimal secara rekursif (berulang).
3. Menghitung nilai solusi optimal, biasanya dengan cara bottom-up.
4. Membangun solusi optimal dari informasi yang telah dihitung.

Apabila dikaitkan dengan permasalahan *Ink Colors*, *Dynamic Programming* ini dapat membantu proses perulangan dalam pengimplementasian fungsi *Solve*. Fungsi ini akan dijalankan pada setiap *vertex* dan pendekatan ini nantinya dapat membantu pencarian nilai yang optimal untuk masing-masing submasalah.

### 2.3.3 *Left Child Right Sibling*

*Left child right sibling* adalah representasi tree dimana setiap *vertex* memiliki maksimal dua *child*. Dimana *edge* keluar pertama menghubungkan *vertex* tersebut dengan *child* pertamanya dan *edge* keluar kedua menghubungkan *vertex* tersebut dengan *sibling* berikutnya.

Proses pembangunan hubungan *edge* keluar dari setiap *vertex* adalah untuk setiap *vertex*, bila memiliki *child*, maka hubungkan *edge* keluar pertama dari *vertex* tersebut dengan *child* pertama, lalu hubungkan *edge* keluar kedua dari *child* ke-i

menuju *child* ke-i+1, dari *child* pertama hingga *child* sebelum terakhir vertex tersebut.

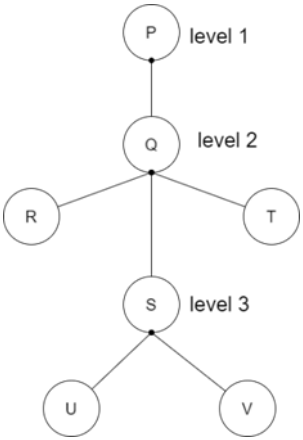
Representasi *left child right sibling* ini lebih efisien dalam penggunaan memori daripada representasi *tree* biasa, tetapi waktu yang diperlukan untuk mencari sebuah *child* dari *vertex* spesifik lebih lambat.

# BAB III
# DESAIN DAN ANALISIS

Pada bagian ini akan dijelaskan analisis dan desain program yang digunakan untuk menyelesaikan permasalahan pada Tugas Akhir ini.

## 3.1 Analisa Submasalah Optimal pada Permasalahan *Ink Colors*

Pada subbab ini akan dijelaskan mengenai submasalah-submasalah yang jawabannya dapat membangun jawaban akhir. Secara garis besar, semua *vertex* mempunyai tiga level transisi yang dapat menggambarkan posisinya di bagan *Stick Man* seperti yang bisa dilihat di Gambar 3.1.

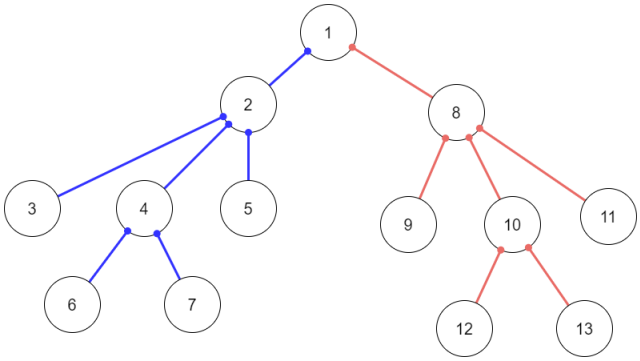



Gambar 3.1 Ilustrasi Level Transisi pada *Stick Man*

Untuk memperjelas bagian-bagian berikutnya, penulis akan mereferensikan *P* di Gambar 3.1 sebagai kepala *Stick Man*, *Q* sebagai badan *Stick Man*, *S* sebagai perut *Stick Man*, *R* dan *T* sebagai tangan *Stick Man*, terakhir *U* dan *V* sebagai kaki *Stick Man*.

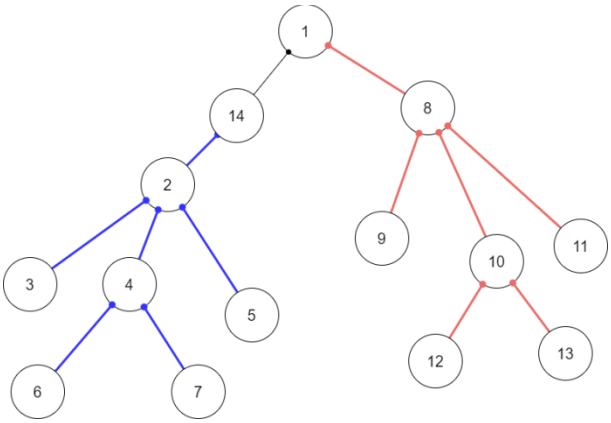
### 3.1.1 Analisa Level Transisi 1

Level transisi 1 menggambarkan “kepala” dari *Stick Man* tersebut. Karena setiap kepala *Stick Man* ini bisa mempunyai badan lebih dari satu seperti Gambar 3.2, Untuk gambar ini, nilai level transisi 1 untuk *vertex* 1 adalah 2.




Gambar 3.2 Ilustrasi Level Transisi 1 dengan Banyak Badan

Bisa juga untuk level transisi 1 ini, memiliki anak yang juga merupakan level transisi 1 untuk *Stick Man*-nya sendiri seperti Gambar 3.3, untuk gambar ini, nilai level transisi 1 untuk *vertex* 1 adalah 2.

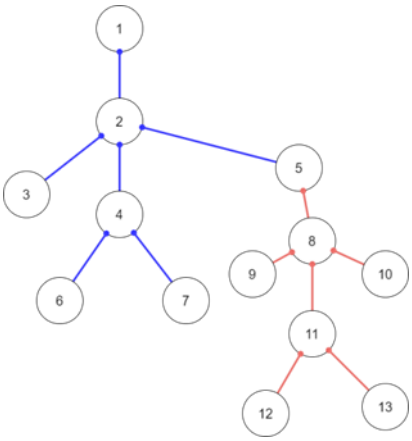


Gambar 3.3 Ilustrasi Level Transisi 1 dengan Salah Satu Anaknya Adalah Level Transisi 1.

### 3.1.2 Analisa Level Transisi 2

Level transisi 2 menggambarkan “badan” dari *Stick Man*. Cara mencari nilai dari level transisi ini adalah dengan mencari semua anaknya yang akan dijadikan perut atau level transisi 3. untuk setiap calon level transisi 3 milik *vertex* ini, akan dicari 2 *vertex* tangan lainnya yang akan membuat nilainya maksimum. Alternatifnya, *vertex* level transisi 2 ini menjadi level transisi 1, akan dipilih dari semua opsi ini yang menjadikan nilai *vertex* ini maksimal.

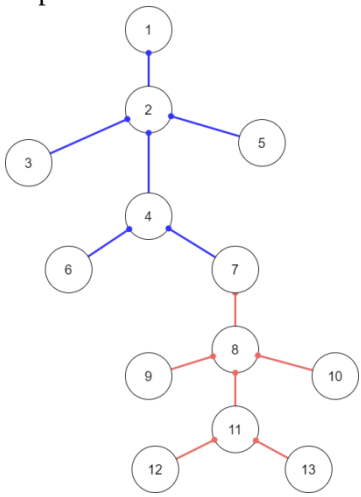
Untuk tangan, akan dicari nilai level transisi 1 untuk setiap tangan karena cuma *vertex* ini cuma bisa menjadi kepala *Stick Man* lain, seperti Gambar 3.4, *vertex* 5 menjadi kepala *Stick Man* lain. Karena hal ini, nilai maksimum untuk tangan *Stick Man* ini adalah nilai Level Transisi 1.




Gambar 3.4 Ilustrasi Tangan *Stick Man* Sebagai Kepala *Stick Man* Lain

### 3.1.3 Analisa Level Transisi 3

Level transisi 3 menggambarkan "perut" dari *Stick Man*. Pada level ini kita cari kaki *Stick Man*. Bila anak dari *vertex* ini kurang dari 3 maka cukup mengeluarkan hasil level transisi 1 di *vertex* ini karena masih mungkin *vertex* ini menjadi kepala *Stick Man* lain. Untuk kaki di *vertex* ini, sama seperti mencari tangan di level transisi 2, akan dicari nilai level transisi 1 untuk setiap kaki karena cuma *vertex* ini cuma bisa menjadi kepala *Stick Man* lain seperti digambarkan pada Gambar 3.5.




Gambar 3.5 Ilustrasi Kaki *Stick Man* Sebagai Kepala *Stick Man* Lain

## 3.2 Definisi dan Notasi

Berikut adalah notasi yang akan digunakan pada penelitian ini.

1. $DP1_{(i)}$ adalah notasi untuk nilai dari *vertex i* pada level transisi 1. Dimana *vertex i* disini berperan sebagai kepala *Stick Man*.

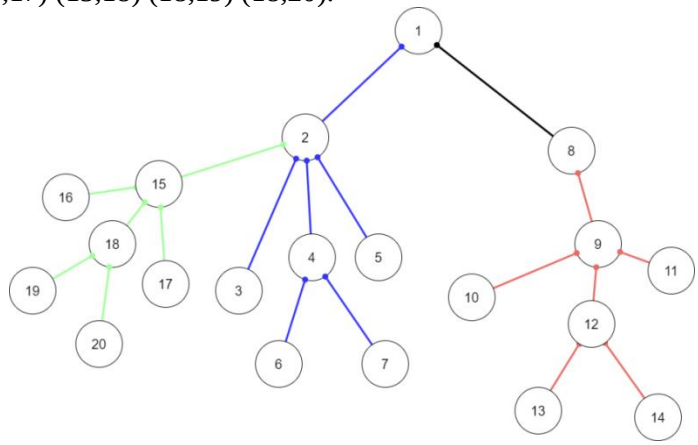
2. $DP2_{(i)}$adalah notasi untuk nilai dari *vertex i* pada level transisi 2. Dimana *vertex i* disini berperan sebagai badan *Stick Man*.
3. $DP3_{(i)}$ adalah notasi untuk nilai dari *vertex i* pada level transisi 3. Dimana *vertex i* disini berperan sebagai perut *Stick Man*.

## 3.3 Pemodelan Relasi Rekuren pada Permasalahan *Ink Colors*

Pada subbab ini akan dijelaskan pemodelan relasi rekuren untuk menyelesaikan permasalahan *Ink Colors*.

### 3.3.1 Pemodelan Relasi Rekuren Level Transisi 1

Untuk *tree* seperti yang terlihat pada Gambar 3.6, total maksimal *Stick Man* yang bisa diwarnai adalah tiga, yaitu *Stick Man* (1,2) (2,3) (2,4) (2,5) (4,6) (4,7), *Stick Man* (8,9) (9,10) (9,11) (9,12) (12,13) (12,14), dan *Stick Man* (2,15) (15,16) (15,17) (15,18) (18,19) (18,20).




Gambar 3.6 Ilustrasi Submaslaah Level Transisi 1

Disini *vertex* 1 tidak bisa berperan sebagai level transisi 2 dan level transisi 3 karena *vertex* 1 adalah *vertex root*. Semua *child* dari *vertex* satu bisa berperan sebagai level transisi 1, level transisi 2, atau keduanya secara bersamaan. Apabila *vertex* yang dipertanyakan tidak punya *child* maka nilai dari $DP1_{(i)}$adalah 0 karena tidak ada *Stick Man* yang bisa dibuat. Dapat disimpulkan bahwa secara umum relasi rekuren untuk submasalah *vertex* 1 seperti yang terlihat pada persamaan (1). Disini $DP2_{(i)}$ diasumsikan juga mengandung unsur $DP1_{(i)}$ seperti yang terjadi pada *vertex* 2 dimana *vertex* tersebut merupakan badan *Stick Man* dan kepala *Stick Man*.

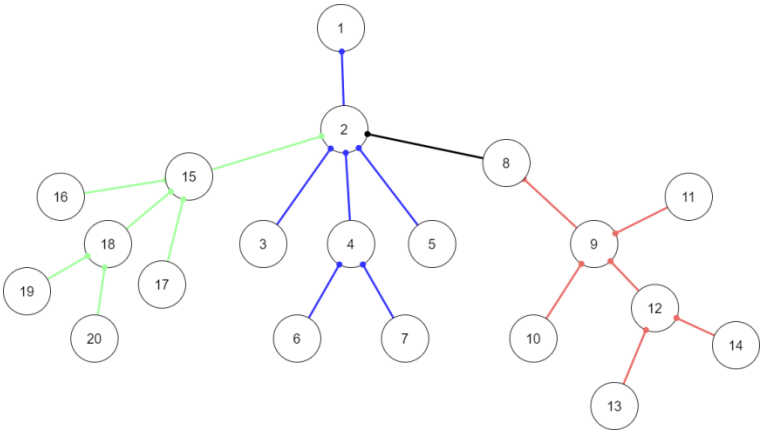
$$DP1_{(i)} = \begin{cases} 0, ChildCount_{(i)} = 0 \\ \sum DP2_{(child(i))} \end{cases} \quad (1)$$

Karena pertanyaan utamanya adalah berapa *Stick Man* maksimal yang bisa diwarnai, dan $DP1_{(i)}$ adalah total dari *Stick Man* yang bisa diwarnai oleh dirinya dan anak-anaknya maka jawaban keseluruhan dari *Ink Colors* adalah $DP1_{(root)}$.

### 3.3.2 Pemodelan Relasi Rekuren Level Transisi 2

Untuk *tree* seperti yang terlihat pada Gambar 2.13, total maksimal *Stick Man* yang bisa diwarnai adalah tiga, yaitu *Stick Man* (1,2) (2,3) (2,4) (2,5) (4,6) (4,7), *Stick Man* (8,9) (9,10) (9,11) (9,12) (12,13) (12,14), dan *Stick Man* (2,15) (15,16) (15,17) (15,18) (18,19) (18,20). Untuk *case* normal dimana kita mencari tangannya, kita cari maksimal $DP3_{(tangan(i))}$ untuk kedua tangan, $DP3_{(perut(i))}$ ,dan $DP2_{(i)}$ untuk *vertex child* lainnya. Dapat disimpulkan secara umum relasi rekuren untuk *case* ini adalah seperti persamaan (2). Untuk mempermudah pencarian nilai $DP2_{(i)}$ pencarian tangan di *case* ini akan disederhanakan dengan bantuan persamaan (3). Sehingga

persamaannya menjadi (4). Disini kita akan mencoba untuk semua kemungkinan level transisi 3 anak dari *vertex* yang akan dicari, dan karena nilai total $\sum DP2_{(child(i) \neq perut(i)))}$ dan $DP3_{(perut(i))}$ selalu sama untuk setiap *vertex* perut yang dicari maka (3) harus minimum pada kedua tangan agar $DP2_{(i)}$ maksimum.



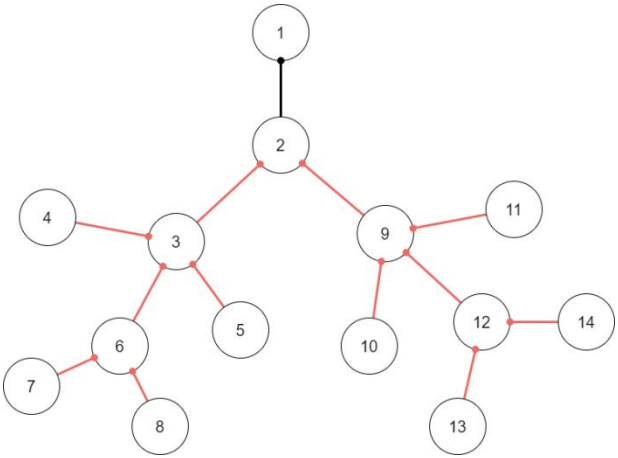

Gambar 3.7 Ilustrasi Submasalah Level Transisi 2

$$DP2_{(i)} = DP3_{(perut(i))} + DP1_{(tangan1(i))} + DP1_{(tangan2(i))} + \sum DP2_{(child(i) \neq \{perut(i), tangan1(i), tangan2(i)\})} \quad (2)$$

$$Choose_{(i)} = DP2_{(i)} - DP1_{(i)} \quad (3)$$

$$DP2_{(i)} = DP3_{(perut(i))} + \sum DP2_{(child(i) \neq perut(i))} - (Choose_{(tangan1)} + Choose_{(tangan2)}) \quad (4)$$

Untuk Level Transisi 2 ini, diperlukan level transisi 1 pada relasi rekurennya untuk mengatasi *case* seperti *vertex* 15 pada Gambar 3.7 yang merupakan badan dari *Stick Man* yang kepalanya merupakan badan *Stick Man* lain. Seperti di Gambar 3.8 di *vertex* 2 yang menjadi kepala dua *Stick Man* berbeda. Untuk *case* ini, relasi rekuren yang menghasilkan nilai maksimal adalah seperti di persamaan (4). Perbedaan persamaan (4) dan (5) adalah di persamaan (4) kita mengasumsikan bahwa ada tangan yang bisa membuat *Stick Man* sedangkan di persamaan (5) kita mengasumsikan bahwa tidak ada tangan yang cocok untuk membuat *Stick Man*.




Gambar 3.8 Ilustrasi Badan *Stick Man* yang Menjadi Kepala *Stick Man* Lain

$$DP2_{(i)} = DP1_{(i)} \tag{5}$$

Dari dua persamaan ini dicari yang maksimal sehingga $DP2_{(I)}$ maksimal seperti di persamaan (6).

$$DP2_{(i)} = \max(DP3_{(perut(i))} + \sum DP2_{(child(i) \neq perut(i))} - (Choose_{(tangan1)} + Choose_{(tangan2)}), DP1_{(i)}) \quad (6)$$

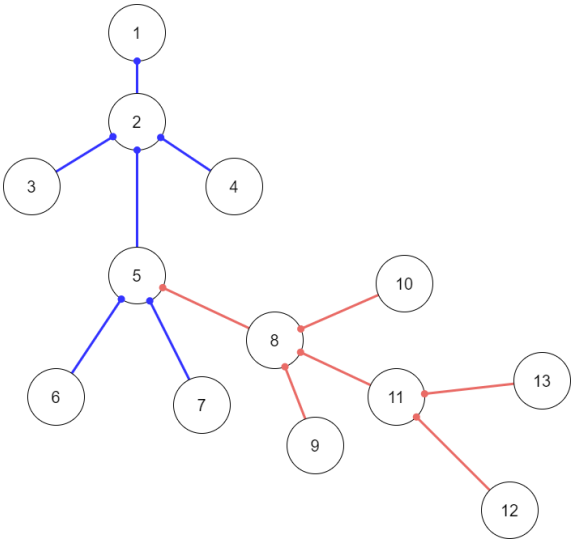
`

### 3.3.3 Pemodelan Relasi Rekuren Level Transisi 3

Pada subbab ini akan dijelaskan pemodelan relasi rekuren untuk level transisi 3.

#### 3.3.3.1 Kasus Relasi Rekuren Level Transisi 3 untuk *Vertex* yang Memiliki *Child Vertex* lebih dari Satu

Relasi Rekuren untuk level transisi 3 secara garis besar sama dengan Relasi Rekuren Level Transisi 2. Level transisi 3 memiliki pemilihan kaki yang harus dimaksimalkan yang mirip dengan pemilihan tangan level transisi 2 yaitu mencari $DP1_{(kaki(i))}$ untuk kedua kaki dan $DP2_{(i)}$ untuk *vertex child* lainnya seperti di Gambar 3.9 dimana pemilihan kaki maksimal adalah *vertex* 6 dan 7. Sehingga kita dapat menyimpulkan salah satu relasi rekuren untuk level transisi 3 adalah seperti di persamaan (7). Dengan bantuan persamaan (3) kita mendapatkan persamaan (8).

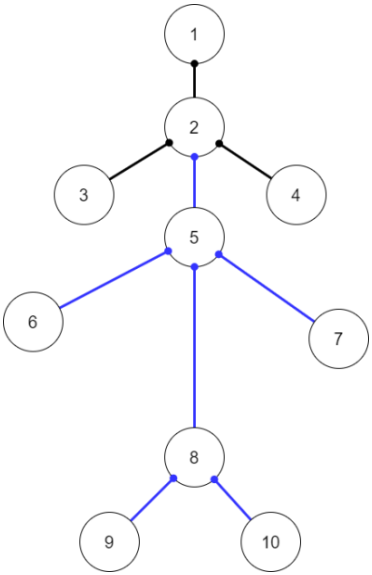



Gambar 3.9 Ilustrasi Submasalah Level Transisi 3

$$DP3_{(i)} = 1 + DP1_{(Kaki1(i))} + DP1_{(kaki2(i))} + \sum DP2_{(child(i) \neq \{kaki1(i), kaki2(i)\})} \quad (7)$$

$$DP3_{(i)} = 1 + \sum DP2_{(child(i))} - (Choose_{(kaki1)} + Choose_{(kaki2)}) \quad (8)$$

*Case* lain relasi rekuren 3 dimana perut sebuah *Stick Man* dapat menjadi badan *Stick Man* lain yang kepalanya adalah *parent vertex* seperti Gambar 3.10 pada *vertex* 5 yang menjadi badan untuk *Stick Man* berkepala *vertex* 2, untuk kasus ini mirip dengan persamaan (5) sehingga menjadi seperti persamaan (9). Diantara persamaan (8) dan (9) akan kita cari maksimal sehingga kita dapat persamaan (10).



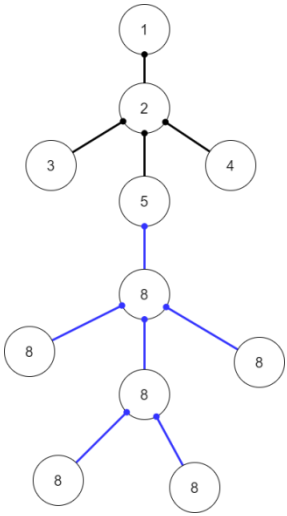

Gambar 3.10 Ilustrasi Perut *Stick Man* yang Menjadi Badan *Stick Man* Lain

$$DP3_{(i)} = DP2_{(i)} \tag{9}$$

$$DP3_{(i)} = \max(\,1 + \sum DP2_{(child(i))} - \left(Choose_{(kaki1)} + Choose_{(kaki2)}\right), DP2_{(i)}) \tag{10}$$

### 3.3.3.2 Kasus Relasi Rekuren Level Transisi 3 untuk *Vertex* yang Memiliki *Child Vertex* satu atau kurang

Bila sebuah *vertex* yang dicari level transisi 3 nya cuma memiliki satu atau kurang *child*, kita tidak bisa membuat sebuah *Stick Man* seperti yang dicontohkan Gambar 3.11, Tetapi *vertex* ini masih bisa menjadi kepala *Stick Man* lain, untuk *case* ini, digunakan level transisi 1 seperti pada persamaan (11).




Gambar 3.11 Ilustrasi *Stick Man* Level Transisi 3 untuk *Vertex* yang Memiliki Satu *Child Vertex* atau Kurang

$$DP3_{(i)} = DP1_{(i)}, ChildCount_{(i)} \leq 1 \tag{11}$$

### 3.3.3.3 Kasus Relasi Rekuren Level Transisi 3 Secara Umum

Dari persamaan (10) dan (11) dapat disimpulkan persamaan relasi rekuren untuk level transisi 3 adalah persamaan (12)

$$DP3_{(i)} = \begin{cases} DP1_{(i)}, ChildCount_{(i)} \leq 1 \\ \max\left(1 + \sum DP2_{(child(i))} - \left(Choose_{(kaki1)} + Choose_{(kaki2)}\right), DP2_{(i)}\right) \end{cases} \tag{12}$$

## 3.4 Pemilihan Kaki dan Tangan *Stick Man*

Seperti yang sudah dijelaskan pada subbab 3.1.2, $(Choose_{(tangan1)} + Choose_{(tangan2)})$ haruslah minimal sehingga persamaan (4) dan (8) menghasilkan nilai maksimal. Untuk mempermudah hal ini, akan ada persamaan seperti persamaan (13) dimana *k* dan *l* adalah *child* dari *vertex i*, dan *j* adalah *vertex* yang akan dijadikan perut *Stick Man* dan *k* adalah level transisi yang dicari.

Untuk kasus level transisi 2, bila child dari vertex *i* kurang dari 3, tidak ada tangan yang cocok sehingga nilai *ChooseBest* akan dijadikan *infinite* agar nilai dari persamaan (4) maksimal.

Untuk kasus level transisi 3, *j* bernilai -1 karena tidak membutuhkan bagian lebih bawah dari perut *Stick Man*. Sama dengan level transisi 2, bila child dari *vertex* i kurang dari 2 tidak akan ada kaki yang cocok sehingga nilai *ChooseBest* akan dijadikan *infinite* agar nilai dari persamaan (8) maksimal.

$$ChooseBest_{(i,j,k)} = \begin{cases} infinite, k = 2 \text{ and } ChildCount_{(i)} < 3 \\ infinite, k = 3 \text{ and } ChildCount_{(i)} < 2 \\ min\left(Choose_{(l)} + Choose_{(m)}\right), l \text{ and } m \in Child_{(i)} \end{cases} \quad (13)$$

## 3.5 Pembuatan *Data Generator* untuk Uji Coba

Pembuatan data generator bertujuan untuk membuat kasus uji dalam rangka melihat performa dari algoritma yang dibuat pada penelitian ini.

Data Generator memberikan kasus uji coba yang disesuaikan dengan data masukan yang dijelaskan pada subbab 4.2.1. Akan terbentuk sebuah *tree* sesuai dengan masukan yang diberikan pada data generator .

Proses pembuatan kasus uji Coba dilakukan oleh data generator seperti yang terlihat pada Gambar 3.12. Baris pertama meminta masukan jumlah vertex yang ada dan baris 2-4

menuliskan *tree* sesuai dengan format data masukan yang dijelaskan pada subbab 4.2.1.

```
CreateTestCase ()
n ← input
print n
for i ← 1 to N-1
    print (randomnumber mod i) + 1
```

Gambar 3.12 *Pseudocode* Pembuatan Test Case

## 3.6 Deskripsi Umum Program

Dalam Menyelesaikan Permasalahan *Ink Colors*, program akan menerima masukan berupa jumlah *vertex*, pada bagian kedua berisi *edge* yang menghubungkan *vertex-vertex* yang membentuk *tree*. Setelah menerima masukan, maka masukan tersebut diolah dan hasilnya akan ditampilkan di layar. *Pseudocode* untuk fungsi *main* ditunjukan pada Gambar 3.13

```
Main ()
n ← input
for  i ← 2 to n
        m ← input
        Adj[m].push_back(i)
solve(1,1) → output
```

Gambar 3.13 *Pseudocode* Fungsi Main

## 3.7 Desain Algoritma

Program akan menggunakan algoritma yang dijelaskan pada subbab 3.3 dan 3.4. Hanya saja untuk diimplementasikan ke dalam sebuah kode harus dilakukan modifikasi. Sehingga program akan terdiri dari fungsi-fungsi yang dijelaskan pada subbab berikut.

### 3.7.1 Desain Fungsi *Choosebest*

Fungsi *ChooseBest* adalah fungsi untuk mencari $ChooseBest_{(k,j)}$ Minimal dimana *k* adalah adalah *vertex parent* dan *j* adalah *vertex* perut *Stick Man* yang akan dicoba. Fungsi ini merupakan hasil implementasi dari persamaan (14). Gambar 3.2 memperlihatkan proses perhitungan seperti berikut. Input berisi *vector* yang sudah di sort, *vector* ini berisi dengan kumpulan $Choose_{(i)}$ & *vertex i* yang merupakan *child-child* dari *vertex* parent yang ingin kita cari dan *vertex* j yaitu *vertex* yang akan menjadi perut di kasus level transisi 2.

Pada baris 1-2 akan di cek berapa *child* yang dimiliki oleh *vertex i*, bila kurang dari dua akan di-*return infinite* karena tidak bisa dibuat kaki atau tangan. Pada baris 3 dan 4 dimasukkan calon $Choose_{(i)}$ yang akan menjadi kaki atau tangan yang paling minimal. Baris 5-8 melakukan pengecekan apakah calon yang akan menjadi salah satu tangan adalah *vertex* yang akan dijadikan perut *Stick Man* bila iya maka tangan tersebut kita ganti menjadi *vertex* yang memiliki $Choose_{(i)}$ paling minimal ketiga. Baris 9-12 melakukan hal yang sama dengan baris 5-8 tetapi untuk *vertex* tangan yang berbeda dari baris 5-8. Baris 13 akan mengembalikan nilai $ChooseBest_{(k,j)}$ yang minimal. *Pseudocode* dari fungsi *ChooseBest* ditunjukan pada Gambar 3.14.

```
ChooseBest()
 Input : sorted vector x of Choose(i) and
 vertex i, vertex j
1. if size(x) < 2
2.     return INFINITE
3. l ← x[0].first
4. r ← x[1].first
5. if x[0].second = i
6.     if size(x) < 3
7.         return INFINITE
8.     else l=x[2].first
9. if x[1].second = i
```

```
    if size(x) < 3
        return INFINITE
    else r=x[2].first
return l+r
```

Gambar 3.14 *Pseudocode* Fungsi *ChooseBest*

### 3.7.2 Desain Fungsi *Solve*

Fungsi *Solve* ini merupakan inti dari algoritma yang diterapkan pada program. Fungsi ini mencakup tahapan yang ada pada subbab 3.3. Karena fungsi *Solve* ini mencakup semua level transisi, maka input yang diberikan juga membutuhkan level mana yang dicari. Detail untuk setiap level transisi akan dijelaskan di subbab 3.7.2.1 – 3.7.2.3. Untuk mempercepat proses akan digunakan variabel global yang berisi hasil $DP1_{(i)}, DP2_{(i)}, DP3_{(i)}$ untuk setiap *vertex*. Hasil ini akan digunakan ketika level transisi *vertex* tersebut sudah pernah diproses. *Pseudocode* dari fungsi *Solve* ditunjukkan pada Gambar 3.15.

```
Solve()
 Input : vertex (i), level transisi (j)
if j = 1
    return DP1(i)
if j = 2
    return DP2(i)
if j = 3
    return DP3(i)
```

Gambar 3.15 *Pseudocode* Fungsi *Solve*

#### 3.7.2.1 Desain Fungsi Solve untuk Level Transisi 1

Fungsi ini merupakan salah satu bagian dari fungsi *Solve* untuk kasus level transisi 1. Fungsi ini mencakup tahapan yang dibahas pada subbab 3.3.1.

Pada baris 1 dan 2 akan dicek apakah level transisi 1 pada *vertex* ini sudah pernah diproses dan akan dikembalikan bila sudah pernah diproses. Pada baris 3 di inisialisasi sebuah variabel *ret* yang mencakup semua level transisi 2 dari *vertex* ini. Pada baris 4 dan 5 ditambahkan hasil level transisi 2 dari semua anak *vertex i*. Pada baris 6 dikembalikan nilai *ret* tersebut. *Pseudocode* dari fungsi ini ditunjukkan pada Gambar 3.16.

```
DP1()
 Input : vertex (i)
1.  if dp[1][i]!=-1
2.        return dp[1][i]
3.  ret ← 0
4.  for every child of i as j
5.        ret += solve(j,2)
6.  return ret
```

Gambar 3.16 *Pseudocode* Fungsi Solve untuk Level Transisi 1

### 3.7.2.2 Desain Fungsi Solve untuk Level Transisi 2

Fungsi ini merupakan salah satu bagian dari fungsi *Solve* untuk kasus level transisi 2. Fungsi ini mencakup tahapan yang dibahas pada subbab 3.3.2.

Pada baris 1 dan 2 akan dicek apakah level transisi 1 pada *vertex* ini sudah pernah diproses dan akan dikembalikan bila sudah pernah diproses. Pada baris 3 kita menginisialisasi sebuah *vector of pair of integer and integer* yang akan digunakan untuk mencari perut dan dua tangan *Stick Man*. Baris 3-5 menambah semua kemungkinan $Choose_{(i)}$ kedalam *vector tmp* Dan menambah semua hasil $DP2_{(i)}$ kedalam variabel *tot*. Baris 6 mensortir *vector tmp* untuk fungsi *ChooseBest*. Baris 7 menginisialisasi variabel *ret* yang akan dikembalikan nantinya. Baris 8-12 mencoba semua kemungkinan level transisi 3 seperti yang dijelaskan pada persamaan (7). Pada baris 13 di kembalikan hasil $DP2_{(i)}$ yang maksimal. *Pseudocode* dari fungsi ini ditunjukkan pada Gambar 3.17

```
DP2()
 Input : vertex (i)
1.  if dp[2][i]!=-1
2.      return dp[2][i]
3.  vector of integer and integer tmp
4.  tot ← 0
5.  for every child of i as j
6.      tmp.add(solve(j,2)-solve(j,1), j)
7.      tot += solve(j,2)
8.  sort tmp
9.  ret ← solve(i,1)
10. for every child of i as j
11.     tot -= solve(j,2)
12.     ret2 = solve(j,3)+tot-choosebest(tmp,j)
13.     ret= max(ret2,ret)
14.     tot += solve(j,2)
15. return ret
```

Gambar 3.17 *Pseudocode* Fungsi Solve untuk Level Transisi 2

### 3.7.2.3 Desain Fungsi Solve untuk Level Transisi 3

Fungsi ini merupakan salah satu bagian dari fungsi *Solve* untuk kasus level transisi 3. Fungsi ini mencakup tahapan yang dibahas pada subbab 3.3.3. Pada baris 1 dan 2 akan dicek apakah level transisi 1 pada *vertex* ini sudah pernah diproses dan akan dikembalikan bila sudah pernah diproses.

Baris 1-3 mengecek apakah *child vertex* yang dicari kurang dari dua, dan bila *child* kurang dari dua akan dikembalikan $DP1_{(i)}$ seperti yang sudah dijelaskan pada persamaan (12). Pada baris 5 kita menginisialisasi sebuah *vector of pair of integer and integer* yang akan digunakan untuk mencari dua kaki *Stick Man*. Baris 7-9 menambah semua kemungkinan $Choose_{(i)}$ kedalam *vector tmp* Dan menambah semua hasil $DP2_{(i)}$ kedalam variabel *tot*. Baris 10 mensortir *vector tmp* untuk fungsi *ChooseBest*. Baris 11-12 melakukan operasi yang sudah didefinisikan pada persamaan (11). Fungsi *ChooseBest* disini menggunakan parameter -1 karena kita tidak butuh bagian lebih bawah dari perut selain kedua kaki *Stick Man*. Baris 11 mengembalikan hasil $DP3_{(i)}$ yang maksimal. *Pseudocode* dari fungsi ini ditunjukkan pada Gambar 3.18.

***DP3***()

```
 Input : vertex (i)
1.  if dp[3][i] != -1
2.       return dp[3][i]
3.  if childofverte(i) < 2
4.       return solve(i,1)
5.  vector of integer and integer tmp
6.  tot ← 0
7.  for every child of i as j
8.       tmp.add(solve(j,2)-solve(j,1), j)
9.       tot += solve(j,2)
10. sort tmp
11. ret ← solve(i,2)
12. ret = max(ret, 1+tot-choosebest(tmp,-1))
13. return ret
```

Gambar 3.18 *Pseudocode* Fungsi Solve untuk Level Transisi 3

# BAB IV
# IMPLEMENTASI

Pada bab ini akan dijelaskan tentang implementasi yang dilakukan berdasarkan algoritma yang telah dirancang pada bab 3.

## 4.1 Lingkungan Implementasi

Digunakan lingkungan implementasi dengan spesifikasi perangkat lunak dan perangkat keras seperti terlihat pada Tabel 4.1.

Tabel 4.1 Spesifikasi Lingkungan Implementasi

| No. | Jenis Perangkat | Spesifikasi |
|---|---|---|
| **1** | Perangkat Keras | • *Processor* Intel Core i5-6600HQ CPU @ 3.30GHz<br>• *Memory* 8GB DDR4 |
| **2** | Perangkat Lunak | • Sistem operasi Windows 10 Pro 64-bit<br>• Bahasa pemrograman C++<br>• *Integrated Development Environment* Code::Blocks 16.01 |

## 4.2 Rancangan Data

Pada subbab ini dijelaskan mengenai desain data masukan yang diperlukan untuk melakukan proses algoritma serta data keluaran yang dihasilkan oleh program.

### 4.2.1 Data Masukan

Data masukan adalah data yang akan diproses oleh program sebagai masukan dengan menggunakan algoritma dan

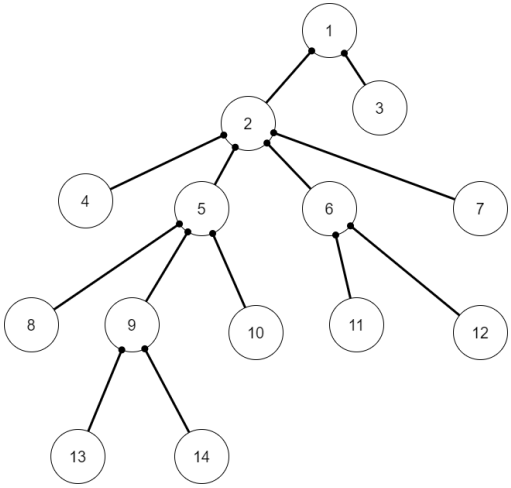
struktur data yang telah dirancang dalam penyelesaian permasalahan *Ink Colors* ini. Data masukan dari persoalan ini dapat dilihat di bawah ini.

1. Masukan berupa $n$ dengan rentang $1 \leq n \leq 10^5$yang merepresentasikan jumlah *vertex*, dengan penomoran *vertex* dimulai dengan 1 hingga $n$.
2. Baris berikutnya terdapat $n - 1$ bilangan bulat bilangan bulat $P_2$, $P_3$, … , $P_N$ ($1 \leq P_i \leq N$ for i = 2, 3, . . . , N) dimana $P_i$ merepresentasikan adanya *edge*($P_i$ , i) dari *vertex* $P_i$ ke *vertex* i.

Berikut contoh masukan pada permasalahan (Gambar 4.1) beserta *Tree* yang direpresentasikan (Gambar 4.2).

```
14
1 1 2 2 2 2 5 5 5 6 6 9 9
```

Gambar 4.1 Format Masukan Permasalahan *Ink Colors*




Gambar 4.2 Representasi Format Masukan

### 4.2.2 Data Keluaran

Data keluaran yang dihasilkan oleh program berupa satu baris yang berisi maksimal *Stick Man* yang bisa diwarnai oleh *tree* yang disediakan. Berikut contoh keluaran pada permasalahan (Gambar 4.3) dengan masukan pada Gambar 4.2.

```
2
```

Gambar 4.3 Format Keluaran Permasalahan *Ink Colors*

## 4.3 Penggunaan *Library*, Konstanta, dan Variabel Global

Pada subbab ini dijelaskan mengenai *library*, *template*, konstanta, dan variabel global yang digunakan dalam program. Pada Kode Sumber 4.1, terdapat 1 *library* yang digunakan yaitu *bits/stdc++.h*. Didefinisikan $pii$ sebagai *vector<int, int>*. Selanjutnya didefinisikan konstanta MAXN yang bernilai $1e^5 + 5$. Sedangkan konstanta INF bernilai $1e^9$. Tabel 4.2 menampilkan variabel-variabel global yang akan digunakan dalam implementasi program.

```
1.  #include<bits/stdc++.h>
2.  using namespace std;
3.  typedef pair<int,int> pii;
4.  const int MAXN=1e5+5;
5.  const int INF=1e9;
6.  int dp[4][MAXN];
7.  vector<int> adj[MAXN];
```

Kode Sumber 4.1 Potongan Kode Implementasi *Library*, Konstanta, dan Variabel Global

Tabel 4.2 Daftar Variabel Global Permasalahan *Ink Colors*

| No | Nama Variabel | Tipe | Penjelasam |
|---|---|---|---|
| **1** | *dp* | int[][] | Digunakan untuk menyimpan hasil dynamic programming untuk setiap level transisi. |
| **2** | *adj* | vector<Int[]> | Digunakan untuk menyimpan jumlah kota yang memiliki jalan penghubung |

## 4.4 Implementasi Fungsi Main

Dalam menyelesaikan permasalahan *Ink Colors* akan diimplementasikan fungsi *Main*.

```
int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(0);cout.tie(0);
   memset(dp,-1,sizeof(dp));
   int n,P;
   cin >> n;
   for(int i=2;i<=n;i++){
         cin >> P;
         adj[P].push_back(i);
   }
   cout << solve(1,1) << '\n';
   return 0;
}
```

Kode Sumber 4.2 Potongan Kode Implementasi Fungsi *Main* Permasalahan *Ink Colors*

## 4.5 Implementasi Algoritma

Pada bagian ini akan dijelaskan mengenai implementasi dari subbab 3.2 yang dibagi menjadi beberapa subbab sebagai berikut.

### 4.5.1 Implementasi Fungsi *ChooseBest*

Fungsi *ChooseBest* ini digunakan untuk membantu memilih kedua kaki atau kedua tangan agar pewarnaan *Stick Man* optimal. *pseudocode* dari fungsi ini sudah dijelaskan pada subbab 2.6. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 4.3.

```
1.  int  Choosebest(vector<pii> tmp,int
    vertex){
2.    if(tmp.size()<2)return INF;
3.     int L=tmp[0].first,R=tmp[1].first;
4.     if(tmp[0].second==vertex){
5.           if(tmp.size()<3)return INF;
6.           else L=tmp[2].first;
7.     }
8.     if(tmp[1].second==vertex){
9.           if(tmp.size()<3)return INF;
10.          else R=tmp[2].first;
11.    }
12.    return L+R;
13. };
```

Kode Sumber 4.3 Potongan Kode Implementasi Fungsi *ChooseBest* Permasalahan *Ink Colors*

### 4.5.2 Implementasi Fungsi *Solve*

Fungsi *Solve* ini akan mencari hasil dari semua level transisi seperti yang dijelaskan pada subbab 2.5. Untuk hasil pewarnaan maksimal *Stick Man* di *tree* akan dipanggil *Solve*(1,1) dimana *vertex* 1 adalah *root* dari *tree* tersebut seperti yang sudah dijelaskan pada subbab 2.5.1. Implementasi dari fungsi ini dapat dilihat pada Kode Sumber 4.4.

```
int solve(int now,int level){
   if(level==1){
         int &ret=dp[1][now];
         if(ret!=-1)return ret;
         ret=0;
         for(auto nxt : adj[now])ret+=solve(nxt,2);
         return ret;
   }
   if(level==2){
         int &ret=dp[2][now];
         if(ret!=-1)return ret;
         vector<pii> tmp;
         int tot=0;
         for(auto nxt : adj[now]){
               tmp.push_back({solve(nxt,2)-solve(nxt,1),nxt});
               tot+=solve(nxt,2);
         }
         sort(tmp.begin(),tmp.end());
         ret=solve(now,1);
         for(auto nxt : adj[now]){
               tot-=solve(nxt,2);

   ret=max(ret,solve(nxt,3)+tot-Choosebest(tmp,nxt));
               tot+=solve(nxt,2);
         }
         return ret;
   }
   if(level==3){
         int &ret=dp[3][now];
         if(ret!=-1)return ret;
         if(adj[now].size()<2)return ret=solve(now,1);
         vector<pii> tmp;
         int tot=0;
```

```
33.         for(auto nxt : adj[now]){
34.               tmp.push_back({solve(nxt,2)-
    solve(nxt,1),nxt});
35.               tot+=solve(nxt,2);
36.         }
37.         sort(tmp.begin(),tmp.end());
38.         ret=solve(now,2);
39.         ret=max(ret,1+tot-Choosebest(tmp,-
    1));
40.         return ret;
41.   }
42. }
```

Kode Sumber 4.4 Implementasi Fungsi *Solve* Permasalahan *Ink Colors*

Pada baris 3-4 dibuat sebuah variabel yang mereferensikan variabel global *dp*, variabel ini akan digunakan untuk menyimpan nilai hasil dari level transisi tersebut. Hal ini dapat dilihat lagi pada baris 10-11 dan 28-29.

**[*Halaman ini sengaja dikosongkan*]**

# BAB V
# UJI COBA DAN EVALUASI

Pada bab ini dijelaskan tentang uji coba dan evaluasi dari implementasi yang telah dilakukan pada tugas akhir ini.

## 5.1 Lingkungan Uji Coba

Digunakan lingkungan uji coba dengan spesifikasi perangkat lunak dan perangkat keras seperti terlihat pada Tabel 5.1.

Tabel 5.1 Spesifikasi Lingkungan Uji Coba

| No. | Jenis Perangkat | Spesifikasi |
|---|---|---|
| **1** | Perangkat Keras | • *Processor* Intel Core i5-6600HQ CPU @ 3.30GHz<br>• *Memory* 8GB DDR4 |
| **2** | Perangkat Lunak | • Sistem operasi Windows 10 Pro 64-bit<br>• Bahasa pemrograman C++<br>• *Integrated Development Environment* Code::Blocks 16.01 |

## 5.2 Skenario Uji Coba

Pada subbab ini akan dijelaskan skenario yang akan digunakan untuk melakukan pengujian terhadap implementasi yang dibuat untuk menyelesaikan permasalahan *Ink Colors*. Skenario uji coba terdiri dari uji coba kebenaran dan uji coba kinerja.

### 5.2.1 Uji Coba Kebenaran

Uji coba kebenaran dilakukan dengan mengirimkan kode sumber ke dalam situs penilaian daring URI Online Judge. Dari

pengiriman tersebut akan didapatkan umpan balik. Umpan balik *Accepted* menyatakan bahwa algoritma pada kode sumber yang diberikan berhasil menyelesaikan permasalahan tersebut. Umpan balik untuk kode sumber yang dijelaskan pada bab 4 dapat dilihat pada Gambar 5.1.

| | | | | | | |
|---|---|---|---|---|---|---|
| 17121751 | 2911 | Ink Colors | **Accepted** | C++ | 0.068 | 2/24/20, 3:44:15 PM |

Gambar 5.1 Hasil Uji Coba Kebenaran Program

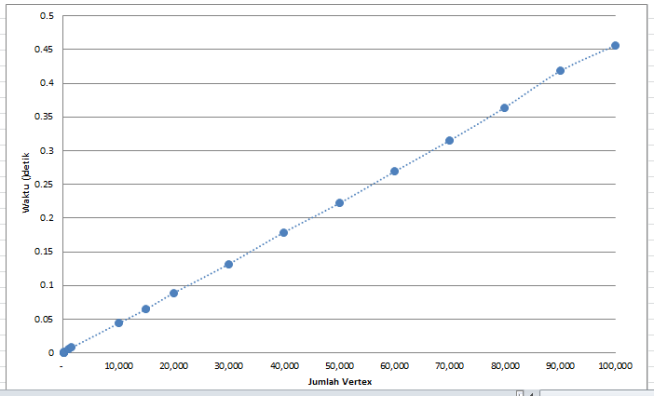
### 5.2.2 Uji Coba Kinerja

Pada subbab ini dijelaskan mengenai pengaruh beberapa variabel terhadap waktu.

#### 5.2.2.1 Pengaruh Jumlah Vertex Terhadap Waktu

Pada uji coba ini digunakan data generator yang sudah dijelaskan pada subbab 3.5. Jumlah *vertex* dibuat bertambah. Hasil uji coba ini seperti yang terlihat pada Gambar 5.2

Terlihat dari grafik bahwa pengaruh jumlah *vertex* terhadap waktu mendekati kurva linier.



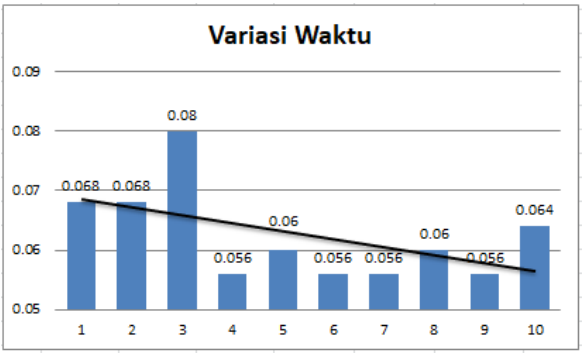

Gambar 5.2 Grafik Uji Coba Pengaruh Banyaknya *Vertex* Terhadap Waktu

#### 5.2.2.2 Uji Coba pada Situs Penilaian Daring URI Online Judge

Uji coba ini dilakukan dengan cara mengirimkan kode sumber ke dalam situs penilaian daring URI Online Judge sebanyak 10 kali. Hal tersebut bertujuan untuk melihat variasi waktu yang dibutuhkan program. Hasil uji coba sebanyak 10 kali pada URI Online Judge dapat dilihat pada Gambar 5.3. Sedangkan untuk diagram variasi waktu dapat dilihat pada Gambar 5.4. Hasil tercepat yang didapatkan dengan metode ini dapat dilihat pada Gambar 5.5.

| | | | | | | |
|---|---|---|---|---|---|---|
| 18133342 | 2911 | Ink Colors | Accepted | C++ | 0.068 | 5/11/20, 2:01:03 AM |
| 18133340 | 2911 | Ink Colors | Accepted | C++ | 0.068 | 5/11/20, 2:00:50 AM |
| 18133313 | 2911 | Ink Colors | Accepted | C++ | 0.080 | 5/11/20, 1:54:57 AM |
| 18133311 | 2911 | Ink Colors | Accepted | C++ | 0.056 | 5/11/20, 1:54:32 AM |
| 18133310 | 2911 | Ink Colors | Accepted | C++ | 0.060 | 5/11/20, 1:54:23 AM |
| 18133308 | 2911 | Ink Colors | Accepted | C++ | 0.056 | 5/11/20, 1:54:05 AM |
| 18133304 | 2911 | Ink Colors | Accepted | C++ | 0.056 | 5/11/20, 1:52:37 AM |
| 18133299 | 2911 | Ink Colors | Accepted | C++ | 0.060 | 5/11/20, 1:51:16 AM |
| 18133295 | 2911 | Ink Colors | Accepted | C++ | 0.056 | 5/11/20, 1:50:36 AM |
| 18133288 | 2911 | Ink Colors | Accepted | C++ | 0.064 | 5/11/20, 1:50:11 AM |

Gambar 5.3 Hasil Uji Coba Kinerja Program




Gambar 5.4 Variasi Waktu Program

| | | | | | | |
|---|---|---|---|---|---|---|
| 18470598 | 2911 | Ink Colors | **Accepted** | C++ | 0.036 | 6/9/20, 10:36:32 AM |

Gambar 5.5 Hasil Submit Paling Cepat

Gambar 5.3 memperlihatkan 10 kali percobaan pengiriman kode sumber ke dalam situs penilaian daring URI Online Judge. Semua mendapatkan hasil *Accepted* dengan waktu bervariasi. Variasi waktu dan memori kemudian dirangkum dalam sebuah diagram.

Dari diagram variasi waktu pada Gambar 5.4, dapat dilakukan pengambilan nilai rata-rata dari 10 percobaan untuk menarik kesimpulan berapa waktu yang dibutuhkan oleh program untuk menyelesaikan permasalahan *Ink Colors*. Rata-rata waktu yang diperlukan program adalah 0,0624 detik dengan standar deviasi 0,007419.

Gambar 5.5 menunjukkan hasil tercepat yang didapatkan saat uji coba ini, hal ini mungkin dikarenakan kecepatan server saat men-*submit* kode sumber pada waktu tersebut lebih cepat dari biasanya.

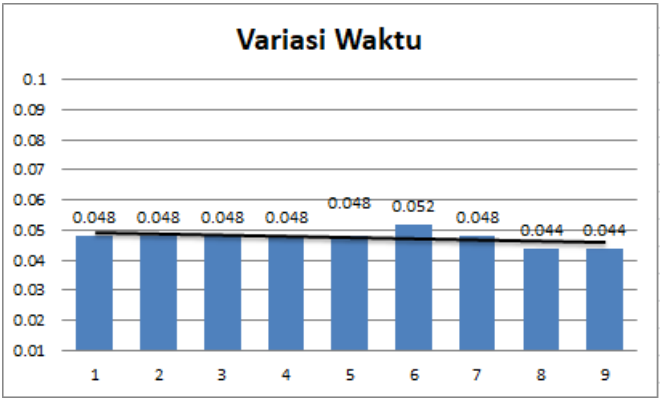
#### 5.2.2.3 Uji Coba Perbandingan dengan Representasi *Tree Left Child Right Sibling*

Pada uji coba ini, representasi *arbitrary tree* masukan diubah menjadi *left child right sibling tree*. Untuk algoritma dilakukan algoritma yang sama dengan uji coba - uji coba sebelumnya. Gambar 5.6 memperlihatkan 10 kali percobaan pengiriman kode sumber ke dalam situs penilaian daring URI Online Judge dengan representasi *tree left child right sibling*.

Pada Gambar 5.7 dapat dilakukan pengambilan nilai rata-rata dari 10 pengumpulan tersebut. Rata-rata waktu yang diperlukan oleh representasi ini adalah 0,0476 detik.

| | | | | | | |
|---|---|---|---|---|---|---|
| 18700306 | 2911 | Ink Colors | Accepted | C++ | 0.048 | 6/28/20, 10:26:22 AM |
| 18700305 | 2911 | Ink Colors | Accepted | C++ | 0.048 | 6/28/20, 10:26:14 AM |
| 18700295 | 2911 | Ink Colors | Accepted | C++ | 0.048 | 6/28/20, 10:23:34 AM |
| 18700293 | 2911 | Ink Colors | Accepted | C++ | 0.048 | 6/28/20, 10:22:58 AM |
| 18700291 | 2911 | Ink Colors | Accepted | C++ | 0.048 | 6/28/20, 10:22:51 AM |
| 18700269 | 2911 | Ink Colors | Accepted | C++ | 0.052 | 6/28/20, 10:18:14 AM |
| 18700267 | 2911 | Ink Colors | Accepted | C++ | 0.048 | 6/28/20, 10:17:48 AM |
| 18700243 | 2911 | Ink Colors | Accepted | C++ | 0.044 | 6/28/20, 10:12:07 AM |
| 18700109 | 2911 | Ink Colors | Accepted | C++ | 0.044 | 6/28/20, 9:47:09 AM |
| 18700098 | 2911 | Ink Colors | Accepted | C++ | 0.048 | 6/28/20, 9:45:24 AM |

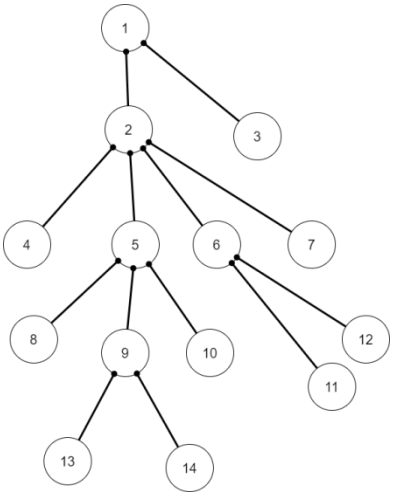
Gambar 5.6 Hasil Uji Coba Kinerja Representasi LCRS




Gambar 5.7 Variasi Waktu Program dengan Representasi LCRS

## 5.3 Analisis Hasil Uji Coba

Dari Subbab 5.2 terlihat bahwa program berjalan dengan benar. Untuk lebih jelas bisa dilihat pada simulasi bagaimana program dapat menyelesaikan permasalahan dengan *tree* yang terlihat pada Gambar 5.8.




Gambar 5.8 *Tree* Uji Coba

Proses penyelesaian dimulai dengan memanggil fungsi Solve untuk level transisi 1 pada *root vertex* pada kasus ini adalah *vertex* 1. Dari situ akan dipanggil sesuai urutan yang sudah dijelaskan pada subbab 3.3. Hasil dari semua nilai transisi dapat dilihat pada tabel 5.2.

Tabel 5.2 Nilai Level Transisi Pada setiap *vertex*

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | -1 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | -1 | -1 | -1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

Program memiliki kompleksitas waktu $O(N)$ dan memiliki kebutuhan memori $O(N)$ untuk menyimpan nilai level transisi.

# BAB VI
# KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan kesimpulan dari hasil uji coba yang telah dilakukan serta saran-saran tentang pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang,

## 6.1 Kesimpulan

Kesimpulan yang dapat diambil berdasar hasil uji coba dan evaluasi terhadap implementasi algoritma pemrograman dinamis untuk permasalahan *Ink Colors* adalah sebagai berikut:

1. Permasalahan untuk mencari pewarnaan maksimal *Stick Man* telah berhasil diselesaikan dengan teori pemrograman dinamis dalam struktur data *tree*.
2. Implementasi algoritma menggunakan pemrograman dinamis dalam menyelesaikan permasalahan *Ink Colors* dengan mendapat umpan balik *Accepted* mendapatkan waktu rata-rata 0,0624 detik dengan standar deviasi 0,007419 dari 10 kali pengumpulan dan rata-rata 0,0476 detik bila menggunakan representasi *left child right sibling*.
3. Pemrograman dinamis pada permasalahan ini mempunyai kompleksitas waktu $O(N)$ dengan N adalah jumlah *vertex* ini sejalan dengan uji coba kinerja yang didapat dimana jumlah *vertex* mempengaruhi secara linier.

## 6.2 Saran

Pada tugas akhir ini tentunya terdapat kekurangan serta nilai-nilai yang dapat penulis ambil. Berikut adalah saran-saran yang dapat diambil melalui tugas akhir ini:

1. Untuk kedepannya, materi yang ada pada tugas akhir ini dapat menjadi bahan riset untuk mencari optimasi lebih lanjut.

2. Optimasi penggunaan memori yang mungkin dilakukan adalah dengan merepresentasikan bentuk *tree* ke dalam *left-child right-sibling binary tree*

# DAFTAR PUSTAKA

[1] "2911-Ink Colors URI Online Judge" *URI Online Judge* Available: *https://www.urionlinejudge.com.br/judge/en/problems/view/2911 [Accessed 07 03 2020]*.

[2] Halim, S., & Halim, F. (2013). Competitive Programming (3th ed.). Singapore.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to Algorithms 3rd Edition, London, England: The MIT Press, 2009.

[4] Bender, Edward A.; Williamson, S. Gill (2010), Lists, Decisions and Graphs. With an Introduction to Probability

**[*Halaman ini sengaja dikosongkan*]**

# BIODATA PENULIS

**Muhammad Farras Rahmatullah**, lahir di Surabaya tanggal 16 Desember 1997. Penulis merupakan anak pertama dari dua bersaudara. Penulis telah menempuh pendidikan formal di SD Muhammadiyah 1 Sidoarjo (2004-2010), SMP Negeri 1 Surabaya (2010-2013), SMA Negeri 5 Surabaya (2013-2016). Penulis melanjutkan studi dengan berkuliah pada program sarjana (S1) di Departemen Teknik Informatika ITS. Selama kuliah di Teknik Informatika ITS, penulis mengambil bidang minat Algoritma dan Pemrograman (AP). Selama menempuh perkuliahan, penulis berpartisipasi dalam kegiatan kepanitiaan Schematics, yaitu sebagai staff teknis National Logic Competition pada tahun 2017 dan menjadi Koordinator Teknis National Logic Competition pada tahun 2018. Penulis dapat dihubungi melalui surel di `farrasrahmatullah@gmail.com`.