Control Flow Graph Based Notasi Algoritmik Autograder

Muhammad Hasan
School of Electrical Engineering and Informatics, Institut Teknologi Bandung
Bandung, Indonesia
Email: 13518012@std.stei.itb.ac.id

Satrio Adi Rukmono
School of Electrical Engineering and Informatics, Institut Teknologi Bandung
Bandung, Indonesia
Email: sar@itb.ac.id

Muhammad Zuhri Catur Chandra
School of Electrical Engineering and Informatics, Institut Teknologi Bandung
Bandung, Indonesia
Email: m.candra@itb.ac.id

***Abstract*—Institut Teknologi Bandung (ITB) Informatics Engineering students are required to learn programming by designing and writing algorithms. The notation used in writing the algorithm is called Notasi Algoritmik which is also used as a standard for writing algorithms in various exams and assessments. Problems occur when educators carry out manual assessments of the Notasi Algoritmik exam. This assessment usually takes a long time and can lead to a decrease in the objectivity of the assessment. One way to solve this problem is to create a Notasi Algoritmik autograder system. Currently, the autograder system is not yet available, but there have been several studies on autograding system that can assist in making the autograder system. This research is the result of the Final Project of several ITB students, but the existing research system is still disconnected and can be further developed to create a Notasi Algoritmik autograder system. On the basis of this, in this project, the result of those researches will be used to develop Control Flow Graph Based Notasi Algoritmik Autograder system. The autograder system will then be measured for its accuracy in regards to educator's manual assessment. Based on the experiment carried out, the results of the assessment from the Notasi Algoritmik autograder system have better accuracy for test data of simpler problems and the results also tend to have a greater value than the results of the educator's manual assessment. In addition, the results of autograder assessments in general have a fairly positive correlation with the manual scores carried out by educators, but the correlation between the two results is still weak.**

***Index Terms*--Notasi Algoritmik, Autograder, Control Flow Graph (CFG)**

I. INTRODUCTION

Notasi Algoritmik in the Informatics Engineering Institut Teknologi Bandung (ITB) environment is a special notation for writing algorithmic texts and is widely used as a standard for writing algorithms in various exams and assessments such as quizzes, midterm exams, and final exams.

Currently, there are still problems when educators carry out assessments of Notasi Algoritmik tests manually, they need to carry out long and repetitive assessment activities which can reduce the objectivity of the assessment. One way to overcome this problem is to create an autograder or automatic assessment system so that educators can use the autograder to automatically assess the assignments. Currently, there are researches regarding automatic assessment which is also the result of the final projects of several ITB students conducted in the year of 2021. However, these researches are made separate and are still disconnected.

In regards to those researches, a larger project called Advances in Automated Grading for Programming Exercise aims to develop and integrate the results of the previous researches so that it can be used as a complete autograder system. This project of creating Control Flow Graph Based Notasi Algoritmik Autograder is part of the larger project. In this large project, there are two main roles, namely the role in creating Autograder Integration with Moodle and GitLab, and the role in the creation of several autograder systems. The two roles are performed separately, but can eventually be merged using an agreed web service contract. The role of this project is to create an autograder for Notasi Algoritmik.

There are two research systems that will be carried out in this project, namely the research system of Sofyana et al. [1] which can convert Notasi Algoritmik into a Control Flow Graph (CFG) and the research system of Sendjaja et al. [2] which can assess programming task based on CFG similarity. These two researches can be integrated into one system named Control Flow Based Notasi Algoritmik Autograder. However, until now the integration has not been carried out, due to several problems, among which are caused by the researches being done separately and the differences in the CFG structure used by each research. Therefore, in this project, problems related to the integration will be resolved.

In addition, the results of the assessment of the autograder made can be compared with the educator's manual assessment to measure its grading accuracy. Finally, the results of this project then need to be made into a web service so that it can be integrated with the parts of the larger project.

II. FOUNDATIONS

A. Notasi Algoritmik

Notasi Algoritmik is simply a notation for writing algorithmic text. This notation used in Institut Teknologi Bandung environment is considered necessary to bridge the diversity and complexity of languages so that students are able to do "abstraction", this notation is also more oriented to detail design than coding [3].

Text in Notasi Algoritmik always consists of three parts, namely the title (header), dictionary, and algorithm. This title is in the form of a program naming declaration, a dictionary containing the variables used, and an algorithm containing the logic and core content of the algorithm from a text Notasi Algoritmik. An example of Notasi Algoritmik is shown in Fig. 1 as follows.

```
PROGRAM odd_even
{program to check whether a number is odd or
even}

KAMUS
   n: integer

ALGORITMA
   input(n)
   if (n mod 2 = 1) then
       output("odd")
   else
       output("even")
```

Fig. 1: Example of Notasi Algoritmik

B. *Autograder System*

An autograder system or an automatic assessment system is a system provided to assess work results automatically and can provide a quick response so that it can be used interactively. The use of automatic assessment will provide convenience in the assessment and can also improve the quality of learning.

Based on what was described by Sherman et al. [4] in his paper, there are indications of positive impact experienced by students in programming subjects by the application of the use of automatic assessment. Automated assessment will enhance the learner's experience in hands-on programming learning. To evaluate a particular program code, it is necessary to test the program. In general, there are two ways of testing, namely black box testing and white box testing.

Black box testing is a testing technique used to determine whether a program is working as intended. Black box testing technique, also known as functional testing, performs testing using several test cases in the form of input and configuration of the program. This technique will then evaluate the program code based on the output given by the program.

In contrast to black box testing, the white box testing method performs testing based on information obtained from the program code. White box testing, also known as structural testing, is a method that can be applied to test the mechanism, structure, and flow of a program. The main focus of white box testing is testing the control flow and data flow of a program. Testing with white box testing can be done by analyzing the program structure alone or by making test cases according to the known program structure.

C. *Control Flow Graph*

A Control Flow Graph is a representation, using graphical notation, of all the possible paths that may be traversed during a program's execution. The Control Flow Graph was invented by Frances E. Allen in 1970, who noted that Reese T. Prosser used a boolean connectivity matrix for flow analysis earlier [5].

In a control flow graph, each vertex in the graph represents a basic block, that is, a piece of code in a line without jumps or jumps targets. Directed edges are used to represent jumps in the control flow. In most presentations, there are two specially designated blocks: the entry block, where control enters through the flow graph, and the exit block, where all control flow flows out [6].

Fig. 2 shows the control flow graph of a sums program. In the figure, each line is represented by a node, then the presence of directed edges shows the flow of control in the program.

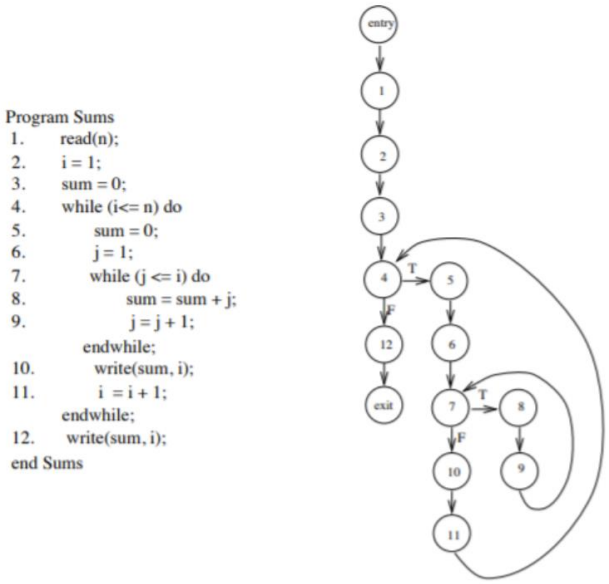

Fig. 2: Control Flow Graph of The Program Sums

III. NOTASI ALGORITMIK AUTOGRADER SYSTEM

The Notasi Algoritmik Autograder System is the result of the project and can be used to grade the writing of Notasi Algoritmik.

The general diagram block of this system is shown in Fig. 3. and has the following flow description when used to grade a writing of Notasi Algoritmik:

1. The system accepts input in the form of educators' Notasi Algoritmik writing files and students' Notasi Algoritmik writing files.
2. The files will then be generated into CFG by the CFG Generator Module.
3. The results of the CFG generation are then converted into an Object Graph by the Intermediate Module

4. The Object Graphs will then be graded by the Graph Grader Module.
5. The results of the grading are then returned to the output of the system.

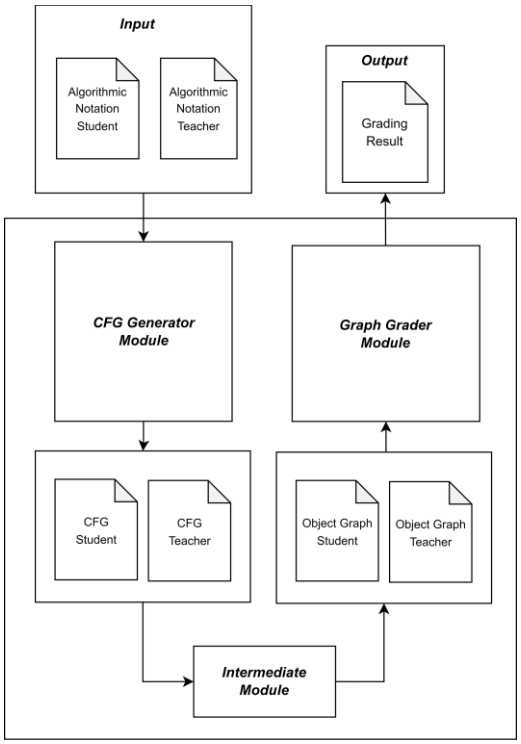

Fig. 3: Diagram Block of Control Flow Graph Based Control Flow Graph Notasi Algoritmik Autograder

As shown from the Fig. 3, This system consists of various modules namely, CFG Generator Module, CFG Grader Module and Intermediate Module.

CFG Generator Module is a module to generate Notasi Algoritmik writing into Control Flow Graph (CFG). In essence, this module is an extension development of the research system by Sofyana et al. [1]. In this module, minor improvements in Grammar and Parsing of Notasi Algoritmik were made.

The process for this module can be broken down into three stages, the descriptions are as follows:

1. Lexical Analysis

In Lexical Analysis, lexing or tokenization is done by converting a sequence of characters into a sequence of lexical tokens. These tokens are referred by the predefined token and is the smallest unit of Notasi Algoritmik text.

2. Syntax Analysis

In Syntax Analysis, the syntax of Notasi Algoritmik will be checked. The token set generated by the lexical analyzer is received by the syntax analyzer. The syntax analyzer then analyzes the token sequence against the predefined grammar in Notasi Algoritmik. After this stage, an AST (Abstract Syntax Tree) will be made

3. CFG Generation

The resulting AST is then used in the CFG generation process by the CFG generator. AST is used as a data structure whose role is to represent Notasi Algoritmik. When generating CFG, the CFG generator component is assisted by the AST Parser component.

The diagram design for this module is shown in Fig. 4 as follows:

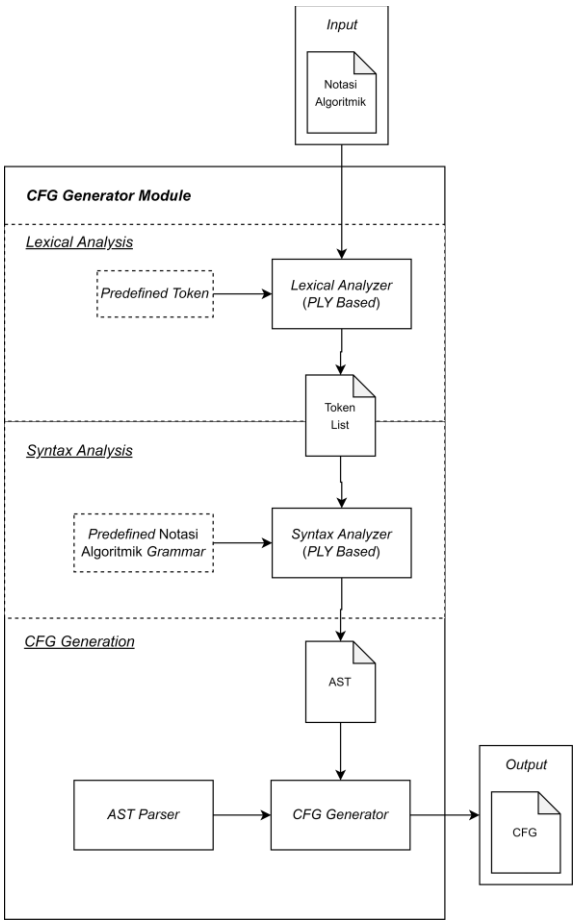

Fig. 4: CFG Generator Module Diagram Design

The Graph Grader Module is a module for Class Graphs by making comparisons between Object Graphs. In essence, this module is an extension development of the research system by Sendjaja et al. [2]. The diagram design for this module is shown in Fig. 5, there are two important parts about this module:

1. Graph Collapser

Graph Collapser is a function that is used to simplify existing graph shapes. The simpler graph structure can result into more accurate grading. In essence, two or more node with the same flows are merged into one node.

2. Graph Comparator

Graph Comparator is a function that can assess the similarity between two graphs. The graph representation that has been created will be used to create a cost matrix that describes the cost of mapping a node into another node between two graphs, as well as the cost of removing a node. After the cost matrix is completed, the total cost will be calculated using the Hungarian algorithm according to the algorithm used by Hu et al. [7].

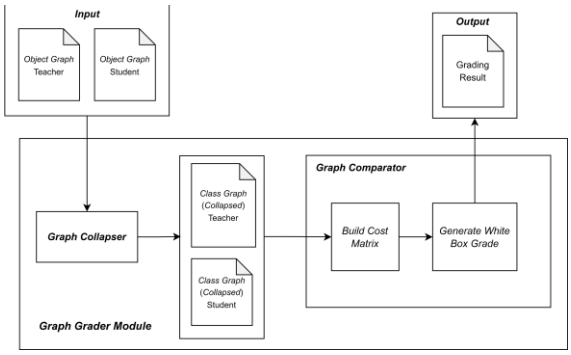

Fig. 5: Graph Grader Module Diagram Design

The Intermediate Module is a unifier between systems. Sofyana et al. [1] with the system of Sendjaja et al. [2]. The main function of this module is to convert the CFG generated by the CFG Generator Module into an Object Graph that can be used by the Graph Grader Module. The Diagram Class for this module is shown in Fig. 6.

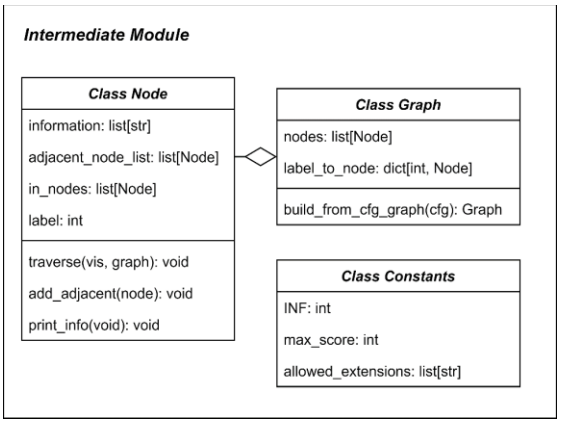

Fig. 6: Diagram Class of Intermediate Module

A brief description for the diagram class is described as follows:

1. Class Node

Class Node is a class that serves to represent nodes in a graph. It has attributes including information that is used to store a part of text in Notasi Algoritmik.

2. Class Graph

Class Graph is a class that serves to represent a graph. In this system, because the graph used is a control flow graph, the implementation of the required graph would be a Directed Graph.

3. Class Constants

Class Constants is a class that contains several constant values that can be used throughout the system, an example of this constant is the maximum score that can be obtained when autograding.

This system was also made into a web service, and has some endpoints that could be accessed shown in Fig. 7.

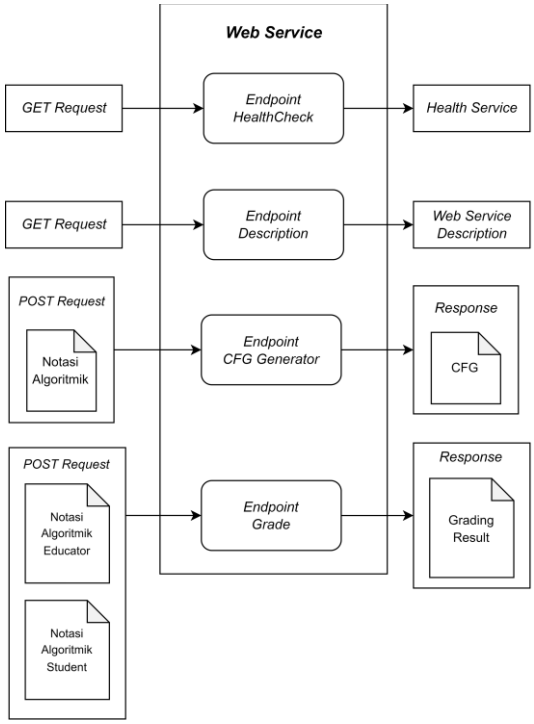

Fig. 7: Web Service Endpoints of The Autograder System

IV. EXPERIMENTS

A. Purpose

The purpose of this experiment is to measure the accuracy of the assessment of the Control Flow Graph based Notasi Algoritmik autograder system against the results of the educator's manual assessment.

B. Step Process

The experiment step process carried out for the experiment is as follows:

1. Collect students' Notasi Algoritmik submission from various exams that can be generated into a Control Flow Graph by the system.
2. The system will run the control flow graph comparison algorithm on a collection of answers from student

submissions and reference answer provided by educators.

3. The system will automatically create a CSV file for the result of the grading.
4. The CSV file of the autograding result will be processed to another sheet and will be compared to manual assessment from educators.
5. The comparison of autograding with manual assessment of autograder then will be analyzed with various measurements.

C. *Result*

The Notasi Algoritmik exam students' submission will be taken from Quiz 2 and Midterms Answers in the IF2110 Data Structure Algorithm Course Year 2020. TABLE I. describes the name of the test data and the number of submissions to be taken for this experiment.

TABLE I. Test Data Name and Number of Submissions

Test Data Name	Number of Submissions
Quiz 2A	117
Quiz 2B	58
Quiz 2C	61
Midterms 2A	14
Midterms 2B	14
Overall	264

The complexity and difficulty of the test problem is increasing for every type of exam, so Quiz 2C is more complex & difficult than Quiz 2B and Quiz 2A. and Midterms 2B is more complex & difficult than Midterm 2A.

The overall test data is a combined test data of Quiz 2A, Quiz 2B, Quiz 2C, Midterms 2A, and Midterms 2B. To see the data spread easily the regression linear visualization for overall test data was made and is shown in Fig. 8.

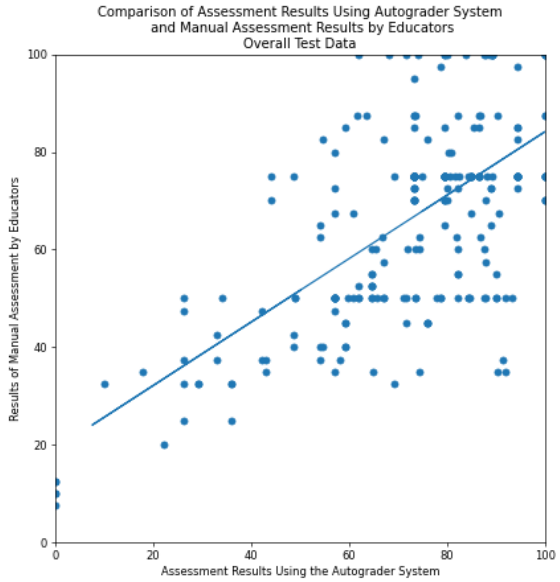

Fig. 8. Regression Linear Visualization for Overall Test Data

To see the data difference visually, a histogram visualization difference was made and is shown in Fig. 9.

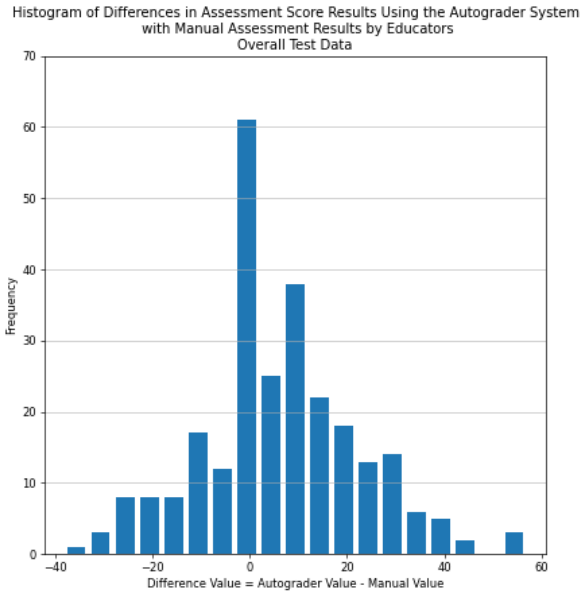

Fig. 9: Histogram Visualization of Differences in Autograder Assessment Results with Manual Assessment Results

The MAE (Mean Absolute Error) for the test data was measured. This MAE value measures the average error between the results of the autograder assessment and the results of the educator's manual assessment. In particular, this helps measure the accuracy of the autograder system in regards to educator's manual assessment. TABLE II. shows the MAE measurement result.

TABLE II. Mean Absolute Error Result for The Test Data

Test Data Name	Mean Absolute Error
Quiz 2A	8.544
Quiz 2B	18.296
Quiz 2C	15.852
Midterms 2A	13.733
Midterms 2B	17.184
Overall	13.108

The Pearson Correlation Coefficient for the test data was also measured. This value measures the strength of the relationship between two variables. In this case, if the value obtained is positive, the greater the value of the autograder, the greater the value of the educator's manual obtained, but if the value is negative, the greater the value of the autograder, the smaller the value of the educator's manual obtained, this can help in making value prediction. TABLE III. shows the Pearson Correlation Coefficient result.

TABLE III. Pearson Correlation Coefficient for The Test Data

Test Data Name	Pearson Correlation Coefficient
Quiz 2A	0.144728
Quiz 2B	0.317872
Quiz 2C	0.634688
Midterms 2A	0.510713
Midterms 2B	0.404776
Overall	0.65628

V. DISCUSSION

In the linear regression visualization shown in Fig. 8 we can see that many points data are still far from the regression line, this shows that there are still a lot of errors in accuracy.

The result of Pearson Correlation Coefficient as well as the linear regression visualization shows that there is a positive correlation between autograder and the manual assessment done by educator, although not quite strong. Interestingly enough, the value of correlation tends to increase for more complex and difficult problem.

From The MAE table shown in TABLE II., it can be seen that the smallest MAE value was obtained by Quiz 2A followed by Midterm IA. The two problems are compared with the same test and different numbers do have easier difficulty this indicates that simpler problems have better accuracy for the autograder.

From The histogram difference visualization shown in Fig. 9, it can be seen that the value of the difference tends to have a positive value, meaning that the result of the autograder value tends to increase more than the educator's manual assessment value, with the farthest value reaching a difference of 56.8367. However, there are also some cases when the result of the autograder score is less than the teacher's manual score, with the farthest value reaching a difference of -38,095.

In the results of the difference data, the average value is measured to be 5.89 and the standard deviation is measured to be 16,398, from these values, it shows that the data is quite spread out from the average value obtained.

VI. CONCLUSION

The Control Flow Based Notasi Algoritmik Autograder was made by Integrating the research results of Sofyana et al. [1] with the research system of Sendjaja et al. [2].

This integration was carried out by looking at the relationship between the two research systems, which can then be integrated with the help of Intermediate Module that functions as an intermediary as well as unifying the two systems.

The results of the assessment of the Notasi Algoritmik autograder system have better accuracy for test cases of simpler problems. The results of the autograder assessment also tend to have a greater value than the results of the educator's manual assessment.

In addition, the results of the autograder assessment in general have a fairly positive correlation with the manual scores carried out by educators, but the correlation between the two results is still weak with a correlation value ranging between 0.1 to 0.6.

Future work

In regards to generating CFG from Notasi Algoritmik text, there are still improvements that could be made by doing trial and error experiments with more students' answer submissions from more various exams.

The comparison algorithm for control flow graph is still limited to the comparison of the graph structure, so a better algorithm can be used to compare the control flow graph.

In addition, more varied Notasi Algoritmik test data can be tested to perform a better analysis of accurate scores and the correlation between the results of the autograder system assessment and the results of the educator's manual assessment

VII. ACKNOWLEDGMENT

The research covered in this paper is a part of a more extensive project named Advances in Automated Grading for Programming Exercise involving our peers Dimas Wahyu Langkawi, Morgen Sudyanto, and Muhammad Kamal Shafi.

REFERENCES

[1] Sofyana, Irfan., Rukmono, S. A., Perdana, R. S. (2021). Pembangkitan Abstract Syntax Tree (AST) dan Control Flow Graph (CFG) Notasi Algoritmik. Skripsi. Bandung: Institut Teknologi Bandung.

[2] Sendjaja, Kevin., Rukmono, S. A., Perdana, R. S. (2021). Evaluasi Tugas Pemrograman Dengan Pendekatan Control Flow Graph. Skripsi. Bandung: Institut Teknologi Bandung.

[3] Liem, I. (2007). Draft Diktat Kuliah Dasar Pemrograman (Bagian Pemrograman Prosedural). Bandung: Kelompok Keahlian Rekayasa Perangkat Lunak dan Data STEI ITB.

[4] Sherman, M., Bassil, S., Lipman, D., Tuck, N., & Martin, F. (2013). Impact of Auto-Grading on an Introductory Computing Course. J. Comput. Sci. Coll. 28, 6, 69–75.

[5] Reese T. Prosser (1959). Applications of Boolean matrices to the analysis of flow diagrams. Papers presented at the December 1–3, 1959, eastern joint IRE-AIEE-ACM computer conference. pp. 133–138.

[6] Yousefi, Javad (2015). *Masking wrong-successor Control Flow Errors employing data redundancy*. IEEE. pp. 201–205.

[7] Xin Hu. dkk. 2009. Large-scale malware indexing using function-call graphs. Proceedings of the 16th ACM conference on Computer and communications security (CCS'09), 611–62