Project Report

on

Vertex split using directed acyclic graph

Design and Analysis of Algorithms

By

VIYYAPU V S D SAI VAMSI (2010030529)

under the supervision of

# P. Sree Lakshmi



Department of Computer Science and Engineering

K L University Hyderabad,

Aziz Nagar, Moinabad Road, Hyderabad – 500 075, Telangana, India.

March, 2022

**Declaration**

The Project Report entitled “vertex split using directed acyclic graph”is a record of bonafide work of VIYYAPU V S D SAI VAMSI (2010030529), submitted as a requirement for the completion of the course **Design and Analysis of Algorithms** in the Department of Computer Science and Engineering to the K L University, Hyderabad. The results embodied in this report have not been copied from any other Departments/University/Institute.

<Signature of the Students >

## Certificate

This is to certify that the Project Report entitled “Vertex split using directed acyclic graph” is being submitted by VIYYAPU V S D SAI VAMSI (2010030529) as a requirement for the completion of the course **Design and Analysis of Algorithms** in the Department of Computer Science and Engineering, K L University, Hyderabad is a record of bonafide work carried out under our guidance and supervision.

The results embodied in this report have not been copied from any other departments/ University/Institute.

## Signature of the Supervisor

Name and Designation

## Signature of the HOD Signature of the Examiner

**ACKNOWLEDGEMENT (Sample - write it on your own)**

First and foremost, we thank the lord almighty for all his grace & mercy showered upon us, for completing this project successfully.

We take grateful opportunity to thank our beloved Founder and Chairman who has given constant encouragement during our course and motivated us to do this project. We are grateful to our Principal **Dr. L. Koteswara Rao** who has been constantly bearing the torch for all the curricular activities undertaken by us.

We pay our grateful acknowledgement & sincere thanks to our Head of the Department **Dr. Chiranjeevi Manike** for his exemplary guidance, monitoring and constant encouragement throughout the course of the project. We thank **Course instructor name** of our department who has supported throughout this project holding a position of supervisor.

We whole heartedly thank all the teaching and non-teaching staff of our department without whom we won’t have made this project a reality. We would like to extend our sincere thanks especially to our parents, our family members and friends who have supported us to make this project a grand success.

# ABSTRACT

**Directed acyclic graphs (dags) are often used to model circuits. Path lengths in such dags represent circuit delays. In the vertex splitting problem, the objective is to determine a minimum number of vertices to split so that the resulting dag has no path of length δ. This problem has application to the placement of flip-flops in partial scan designs, placement of latches in pipelined circuits, placement of signal boosters in lossy circuits and networks, etc. Several simplified versions of this problem are shown to be NP-hard. A linear time algorithm is obtained for the case when the dag is a tree. A backtracking algorithm and heuristics are developed for general dags and experimental results using dags obtained from ISCAS benchmark circuits are obtained**

# INDEX

|  |  |
| --- | --- |
| **S.N** | **TITLE** |
| 1 | Introduction |
| 2 | Literature Survey |
| 3 | Hardware & Software requirements |
| 4 | Functional & Non-functional requirements |
| 5 | Proposed System |
| 6 | Implementation |
| 7 | Results Discussion |
| 8  9  10 | Conclusion and Future Work  References  Appendices (if any) |

Introduction

In order to achieve high fault coverage in sequential circuits they are often designed to be easily testable. The current method of choice is the scan-design. In test mode all flip-flops in a sequen tial circuit, using scan-design, are connected into one or more shift registers. This allows one to set the contents of the flip-flops to the desired state as well as to observe the states of the flip flops. As the complexity of logic circuits grows, the overhead for full scan-designs may become unacceptable. For such situations, partial-scan designs have been proposed. In partial-scan designs only a selected subset of the flip-flops in a sequential circuit are included in the scan path. Several methods to choose the flip-flops to be included in the scan-path have been proposed . One of these proposals gives a method to use the structural information in a sequential circuit to determine the flip-flops to be placed in a scan-path [CHEN90]. We briefly discuss this method. A sequential circuit is represented by a directed graph (digraph) called S-graph. Each flip flop in a sequential circuit is represented by a node in the S-graph. A directed edge exists in the S-graph from node i to node j if the state of the flip-flop represented by node j depends on the state of the flip-flop represented by node i (that is ,there is a path, through combinational logic, in the circuit from the output of flip-flip i to the input of flip-flop j). Figure 1 is an example of a S graph. Empirical evidence suggests that the existence of cycles and the maximum path length between nodes of the S-graph increase the complexity of deriving tests for sequential circuits. It was therefore suggested in to include a minimum subset of flip-flops into a scan-path such that the resulting S-graph is cycle-free and the maximum distance between a pair of nodes is small.

distance between any pair of nodes is less than or equal to a specified number δ. This second step can be modeled as a vertx splitting problem on directed acyclic graphs (dags). In this paper we study solutions to the problem of finding a minimum number of nodes, in a dag, to be split such that the maximum distance between any two nodes in the resulting digraph is less than or equal to a pre-specified value δ. The dags we consider are more general than the ones that arise from S-graphs. We permit each edge in the dag to have a positive integral weight instead of requiring all edges to have unit weight. This generalization can be shown to have application in the placement of latches in pipelined circuits and in the placement of signal boosters in lossy circuits. In Section 2, we introduce the terminology we shall use in the remainder of this paper. The NP-hard results are developed in Section 3 and the linear time algorithm for tree dags is given in Section 4. A backtracking algorithm and heuristics for the dag vertex splitting problem are proposed in Section 5 and 6, respectively. Section 7 reports on experiments with the ISCAS benchmark circuits. It should be noted that a linear time algorithm for series-parallel dags is easily derived from the linear time dag vertex deletion algorithm of

Literature Survey

sequential circuit is represented by a directed graph (digraph) called S-graph. Each flip flop in a sequential circuit is represented by a node in the S-graph. A directed edge exists in the S-graph from node i to node j if the state of the flip-flop represented by node j depends on the state of the flip-flop represented by node i (that is ,there is a path, through combinational logic, in the circuit from the output of flip-flip i to the input of flip-flop j).

 Path lengths in such dags represent circuit delays. In the vertex splitting problem, the objective is to determine a minimum number of vertices to split so that the resulting dag has no path of length > δ. This problem has application to the placement of flip-flops in partial scan designs, placement of latches in pipelined circuits, placement of signal boosters in lossy circuits and networks, etc. Several simplified versions of this problem are shown to be NP-hard.

I want to split a graph into its components (like in the example DAG below. Note the colored identifiers of each node as they represent the components). After I've found the components in the picture I want to find the root and last child of that component

Hardware & Software requirements

* Python
* PyCharm

Functional & Non-functional requirements

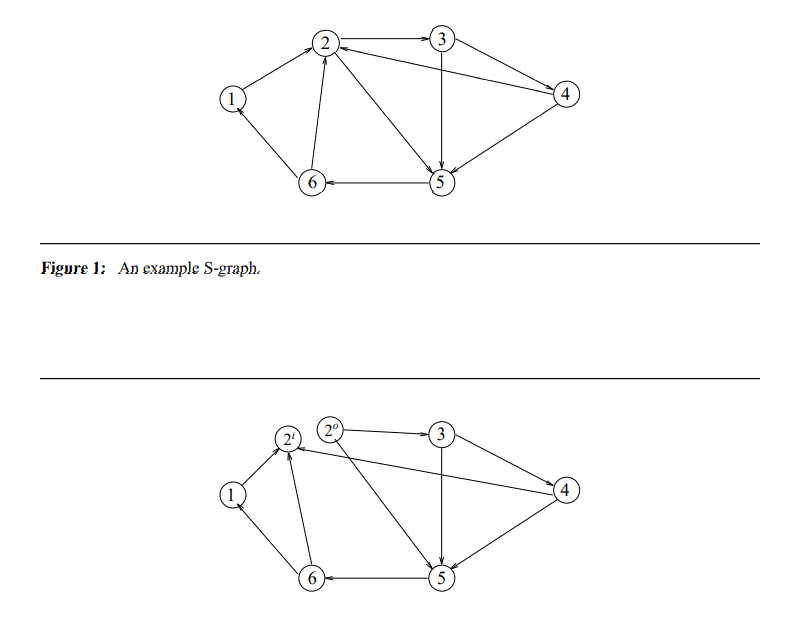
To do this project we used classes to create tree

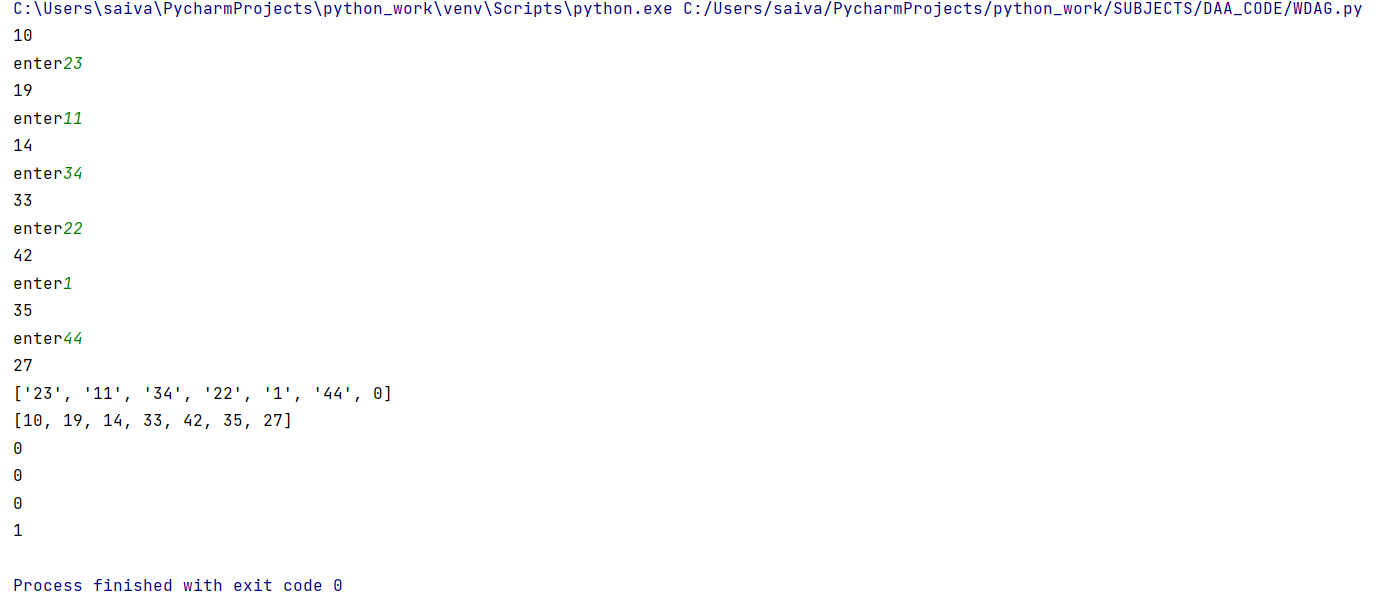
We used trees because tree is also a acyclic graph

Proposed System

In the proposed system they used some directed cyclic graph to show but as we are working on directed acyclic graph we have to take a graph with no cycles in it

By using directed acyclic graph we gent more cycles because of that when we run the code the cycle will be executed again and again because of this reason only we used directed acyclic graph

****



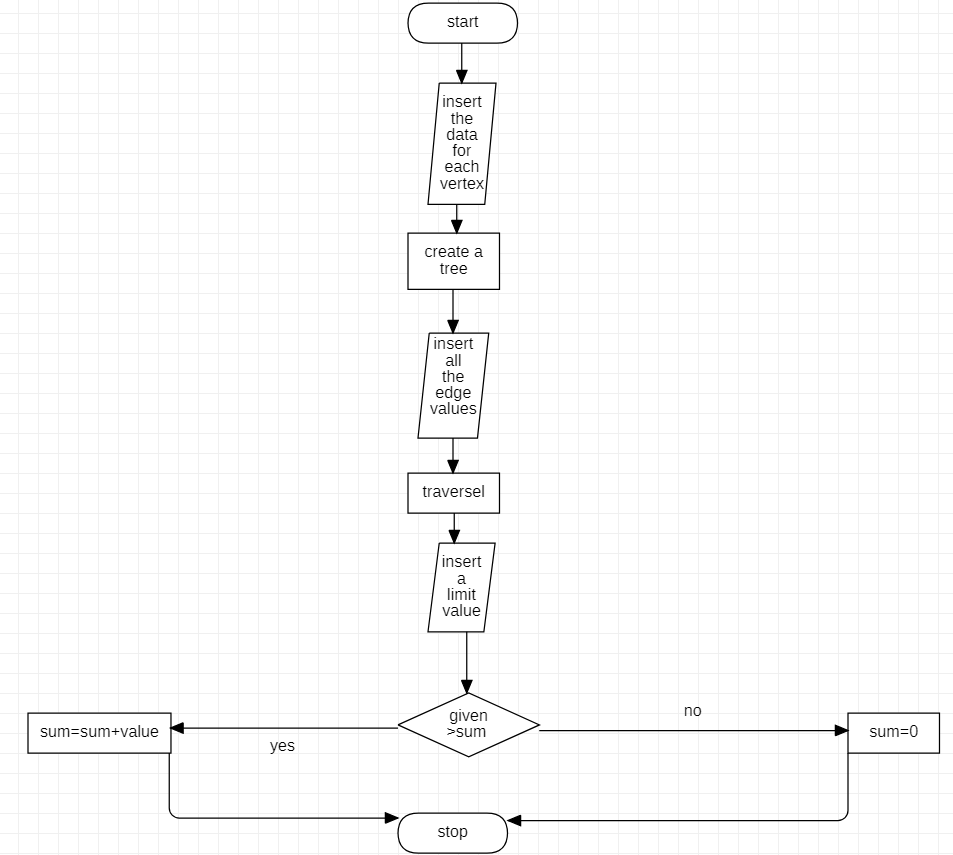
Here we see I created tree in the diagram because we are mainly concentrating on vertex split here I gave some input this input is edge lengths

Implementation

My project is to work on vertex split in directed acyclic graph means I have to implement vertex split on direct acyclic graph so we are taking a tree first I created a tree after that I gave some edge values and started traversal from one node to another in this program I gave some value and my sum of edge values should not be more than the given value is the value is more than the given value the value of the traversal will become zero and it will continue from the current position

The main condition used in this is I used in this is

**if** sumofterms<given:  
 print(sumofterms)  
 sumofterms = sumofterms +int(hhh[i])  
  
**else**:  
 # print(sumofterms-int(hhh[i]))  
 sumofterms=0



Results Discussion

Directed Acyclic Graphs are often used to model circuits and networks. The path length in such Directed Acyclic Graphs represents circuit or network delays. I has many applications in the fields of computer science and electrical engineering.

Conclusion and Future Work

Conclusion is I used in trees but we can do this in any acyclic graph this Teck nick will be used in networking we can reduce the distance from points the data delivery will we limited

References

[splitc.pdf (ufl.edu)](https://www.cise.ufl.edu/~sahni/papers/splitc.pdf)

<https://stackoverflow.com/questions/36888554/splitting-a-directed-acyclic-graph-dag-into-components-then-finding-root-and>

[Splitting a Directed Acyclic Graph (DAG) into components, then finding root and last child in those components - Stack Overflow](https://stackoverflow.com/questions/36888554/splitting-a-directed-acyclic-graph-dag-into-components-then-finding-root-and)

arranged in IEEE/ASME/ASCE format.