D_4_180014_180022

Abhay 180014 Abhishek Mittal 180022

21 October 2020

1 Problem Statement

Program analysis tools are extremely important for understanding program behavior. Computer architects need such tools to evaluate how well programs will perform on new architectures. Software writers need such tools to analyze their programs and identify critical pieces of code. Compiler writers often use such tools to find out how well their instruction scheduling or branch prediction algorithm is performing or to provide input for profile-driven optimizations. Nearly 25 years ago, a program analysis tool was designed. At the core, it solves an algorithmic problem on a directed acyclic graph (DAG). This result appeared in one of the most prestigious conference in systems area of computer science, and it has been a milestone in the area of program analysis. You will be solving the algorithmic problem lying at the core of this result, which is neatly described as follows. Given a directed acyclic graph G = (V, E) with the vertex s as the root vertex and the vertex t as the exit vertex, pathid of a path p from s to t is defined as the sum of weights of edges along p. The problem is to assign integral weights to the edges in G such that all paths from s to t get unique pathids from 0 to N - 1, where N is the total number of paths from s to t. Design the most efficient algorithm to accomplish this task. Full marks will be given only if your algorithm achieves the best time complexity known for this problem. You must also provide a concise and formal proof of correctness of the algorithm.

2 Solution

2.1 Algorithm

In the algorithm given below, Num_paths[v] denotes the number of paths from v to t. Also, weights[v, u] denotes the weight of edge (v, u) assigned to it.

Procedure : Unique_Pathids(G, s, t)

1. Do topological ordering of the given graph G. (G is a DAG)
2. Num_paths[t] $\leftarrow$ 1
3. For each v $\in$ (V -{t}) in decreasing order of topological numbering of vertices :
4. Num_paths[v] $\leftarrow$ 0
5. For each edge (v, u) $\in$ E and u $\in$ V :
6. weights[v, u] $\leftarrow$ Num_paths[v]
7. Num_paths[v] $\leftarrow$ Num_paths[v] + Num_paths[u]
8. **return** weights

2.2 Proof of Correctness

Lemma 1: The Num_paths[v] in the algorithm correctly stores the number of paths from v to t.
Proof : Let us prove the lemma by using induction on the vertices in decreasing order of topological order τ. Here, τ denotes the topological numbering from 0 to $|V|$-1.
Base Case :
Since, number of paths from t to t in just 1. Hence, Num_paths[t] = 1 in line 2.
Induction Hypothesis :
Let us assume the lemma holds true $\forall$ u $\in$ V such that i $\leq \tau_u \leq |V|$ -1 for some i from 0 to $|V|$ - 1.
Induction Step :
Now, we shall prove the lemma to be true for vertex v $\in$ V such that τ_v = i-1.
It is clear that the number of paths from v to t is equal to the sum of number of paths from u to t such that u $\in$ V and (v, u) $\in$ E.
Number of paths from v to t = $\sum_{(v,u)\in E}$ Number of paths from u to t
In topological order of the vertices, v must come before vertex u where (v, u) $\in$ E. Thus, by using the induction hypothesis, we can say :
Num_paths[v] = $\sum_{(v,u)\in E}$ Num_paths[u]
Hence, the lemma is proved.

Theorem : The algorithm assigns weights to the edges in such a way that the pathids(sum of weights of edges along a path) of all the paths from vertex v to t are unique and range from 0 to Num_paths[v] - 1.
Proof : Let us prove the theorem by using induction on the vertices in decreasing order of topological order τ of G. Again, here numbering is given from 0 to $|V|$ - 1 as done in proof of lemma.
Base Case :
When the vertex v is the vertex t, then essentially there is single path from v to t and Num_paths[t] = 1. Thus, the base case is trivial.
Induction Hypothesis :
Let us assume the theorem holds true $\forall$ u $\in$ V such that $i \leq \tau_u \leq |V| - 1$.
Induction Step :
Now, we shall prove the theorem to be true for vertex v such that τ_v = i - 1.
Now, let us suppose the edges emanating from vertex v end up at vertex $u_1, u_2, ..., u_l$.
Considering an edge (v, u_j) $\in$ E for $1 \leq j \leq l$.
By induction hypothesis, the pathids from vertex u_j to t are unique and range from 0 to Num_paths[u_j] - 1.
Also, in line 6 of our algorithm, weight of edge (v, u_j) = $\sum_{k=1}^{j-1} Num\_paths[u_k]$ since Num_paths[v] initialises with 0. For j = 1, weight of edge (v, u_j) = 0.
Hence, the pathids of any path from v to t whose initial edge is (v, u_j) are **unique** and range from $\sum_{k=1}^{j-1} Num\_paths[u_k]$ to $\sum_{k=1}^{j} Num\_paths[u_k]$ - 1.
Uniqueness follows from uniqueness of all pathids from u_j and range follows from above.
$\rightarrow$ Now, let us consider all edges emanating from v. Now, clearly the range of values of pathids starting from edge (v, u_j) for different values of j are different since, Num_paths[u_j] is greater than equal to zero for all u_j.
$\rightarrow$ Let us denote a range of values of pathids from a vertex v starting with edge u_j by R_{u_j}. Then, clearly the range of values of pathids starting from vertex v is the union of all ranges since all the ranges are unique and are disjoint. Thus,
Range of pathids from v = $R_{u_1} \cup R_{u_2} \cup ... \cup R_{u_l}$
Hence, range of pathids from v is from 0 to Num_paths[v] - 1. (Because, Num_paths[v] = $\sum_{k=1}^{l} Num\_paths[u_k]$).
Hence, proved.

By using the above theorem, we get all the pathids from s to t which are unique and range from 0 to N-1 where N is the total number of paths from vertex s to t. Thus, the algorithm correctness has been proved.

2.3 Time Complexity :

We applied topological ordering to the graph G in line 1 which takes O(m+n) time, where m = no. of edges and n = no. of vertices. In the for loop in line 3, we iterate over all the vertices of the graph (O(n) time) and in each loop we iterate over all the edges emanating from that vertex. Here, $\sum O(m_i)$ = O(m)
Hence, total time complexity is O(m + n).