Algorithms

Hw4: Subgraph Matching: DAF

June 19, 2019

Choonghyun Han 자유전공학부 2015-14403

Hyojin Park 자유전공학부 2016-12184

Hyejung Park 자유전공학부 2016-13206

1. Summary of the Assignment

As subgraph matching is an NP-hard problem, there have been countless attempts to find a better performing algorithm. One of the characteristics of the state-of-the-art algorithm DAF is that it chooses a rooted DAG(directed acyclic graph) of the query graph. And our objective is to write a program that with inputs of data graph G and query graph q, outputs query DAG which is optimized for our data sets so that it will perform superior to the original algorithm. DAG will be written in a form of linear order of query vertices.

2. Development Environment

Ubuntu 18.04 LTS java openjdk 10.0.2

3. Comparison between our algorithm and orignal algorithm

Our Algorithm is presented as result 1, and original as result 2

result1

Average elapsed time: 0.289047585366

Average #recursive calls: 24.0

#Solved queries: 99

result2

Average elapsed time: 73.7185745122

Average #recursive calls: 30885.7682927

#Solved queries: 82

Above graph is presented in log scale.

If t is an average elapsed time for original algorithm, ours take only 0.039t.

Our algorithm out performs the original by 1287 times for the number of recursive calls, and solves 17 more queries out of 100.

4. Description of our algorithm and reason upon selection

Our algorithm selects the root of DAG by comparing the numQueryEdge instead of original algorithm’s approach of comparing the number of candidates and degree. This method will check more extensively for similarity of the root in query graph to the vertex comparing in data graph than just using degree. The number of edges in data graph is saved in numEdge, and is used to calculate the number of edges in query graph which is saved in numQueryEdge. Then from the root, DFS traversal is used to build the DAG instead of original algorithm’s BFS approach. Our hypothesis is that the given data sets of data graph and query graphs are formed in a way that will be easily traversed through with DFS method rather than BFS, which means a deep tree with little branches. The performance evaluations, as shown in above, strengths this idea as with the new DAG, DAF is able to perform at much higher speed.