An approach to query for large complex knowledge base graph database

Developed By

ARGHA SARKAR

ARKAPRABHA GUIN

DEBSANKAR SINHA ROY

Under the Supervision of

Prof. SAFIKURESHI MONDAL

Assistant Professor

Department of Computer Science & Engineering.

Narula Institute of Technology

02.05.2018

An approach to query for large complex knowledge base graph database

A Dissertation Submitted in partial fulfillment for the Degree of Bachelor of Technology (B.TECH), 8th Semester in Computer Science & Engineering

Submitted By

Name	University Registration Number	University Roll Number
ARGHA SARKAR	141270110024	12700114024
ARKAPRABHA GUIN	141270110027	12700114027
DEBSANKAR SINHA ROY	141270110037	12700114037

Under the Supervision of

Prof. SAFIKURESHI MONDAL

Assistant Professor

Department of Computer Science & Engineering.

Narula Institute of Technology

Maulana Abul Kalam Azad University of Technology

(May, 2018)

CERTIFICATE OF ORIGINALITY

The project entitled **“An approach to query for large complex knowledge base graph database”** has been carried out by ourselves in partial fulfillment of the degree of Bachelor of Technology in Computer Science & Engineering of Narula Institute of Technology, Agarpara, Kolkata under Maulana Abul Kalam Azad University of Technology during the academic year……2014-2018…..

While developing this project no unfair means or illegal copies of software etc. have been used and neither any part of this project nor any documentation have been submitted elsewhere or copied as far in our knowledge.

Signature:

Name: ARGHA SARKAR

University Roll No.:12700114024

University Registration No.:141270110024

Signature:

Name: ARKAPRABHA GUIN

University Roll No.:12700114027

University Registration No.:141270110027

Signature:

Name: DEBSANKAR SINHA ROY

University Roll No.:12700114037

University Registration No.:141270110037

CERTIFICATE OF APPROVAL

This is to certify that the project entitled **“An approach to query for large complex knowledge base graph database”** has been carried out by ARGHA SARKAR, ARKAPRABHA GUIN, DEBSANKAR SINHA ROY, under my supervision in partial fulfillment for the degree of Bachelor of Technology (B.TECH) in Computer Science & Engineering of Narula Institute of Technology, Agarpara affiliated to Maulana Abul Kalam Azad University of Technology during the academic year…2014-2018....

It is understood that by this approval the undersigned do not necessarily endorse any of the statements made or opinion expressed therein but approves it only for the purpose for which it is submitted.

Submitted By:

Name: ARGHA SARKAR	Name: ARKAPRABHA GUIN
University Roll No.: 12700114024	University Roll No.:12700114027
University Registration No.:141270110024	University Registrations No.:141270110027

Name: DEBSANKAR SINHA ROY

University Roll No.: 12700114037

University Registration No.: 141270110037

-- ------------------------------------

(Mr. SAFIKURESHI MONDAL) (External Examiner)

Assistant Professor, CSE Dept.

(Mr. Jayanta Pal)

HOD, CSE Dept.

Acknowledgement

We have taken efforts in this project. However, it would not have been possible without the kind support and help of Mr. Safikureshi Mondal and my team members. We would like to extend our sincere thanks to all of them.

We are highly indebted to our teachers for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express our gratitude towards our parents for their kind co-operation and encouragement which helped us in completion of this project.

Our thanks and appreciations also go to our friends in developing in the project and who have willingly helped us out whit their abilities.

Table of Contents:

Abstract

Depth first search (DFS) is one of the most fundamental algorithm for searching a graph. In our previous work, a mobile-assisted diagnosis scheme has been proposed and we have designed a Disease Symptom data model as a large Knowledge base. In this work, we present the design and implementation of a pruning algorithm to capture a part of the large graph based data model. Generally, to search a large graph, most of the graph queries are too long and very cumbersome to write and sometimes very difficult to implement. Pruning algorithm is one of the prominent solutions of this problem. It results a sub graph or forest for the desired input. Here, our proposed pruning algorithm is multi source sequential DFS algorithm and it is demonstrated on Disease-Symptom graph database.

I. Introduction

The uses of knowledge graph databases are increasing due to the increase in real life data. Most of the real life data are stored as graph databases because of their dynamic, irregular and heterogeneous nature. Compared to other traditional relational databases and XML models, graph databases capture the semantics of the application efficiently, have more power and play important role in the application development and data access. It is used to store structured and unstructured data. Protein in molecules, linkage between web pages, internet and social network are the examples of applications where real life graph databases are used. Therefore, searching and querying are the basic operations in any graph database most of the time. Although, these are trivial tasks, there are some general problems which emerge while accessing and searching large-sized knowledge graph databases. These are as follows:

I. It is very difficult to understand the graph queries while accessing the data due to the complex structure of queries.

II. In case of cloud-based graph databases, repeated queries made to the cloud require a large number of packet transmissions. Hence, it may be very expensive in terms of communication and might delay the process.

III. Sometimes for smart devices like mobile phones and PDAs, it is quite impossible task to bring down the entire knowledge graph on a mobile device due to limited storage capacity of the devices.

IV. Errors may creep in during transmission from cloud to mobile devices. It is therefore necessary to evolve techniques to reduce error in the output and response time for querying.

In our previous work [1], we have designed and developed a Knowledge graph database for Disease-Symptom data model. We propose a mobile-assisted healthcare scheme, where we use the Disease-Symptom graph and store that in the cloud. The graph knowledge base is searched to make the preliminary diagnosis of a disease based on the symptoms reported by a patient. In such cases, instead of extracting the whole database it is efficient to offload a part of the database onto the mobile and graph pruning algorithms are needed to be developed for this purpose. Breath First Search (DFS) and Depth First Search (DFS) are the two basic traversal algorithms for any graph. According to nature of DFS algorithm, it is ruled out for creating any pruning algorithm for any directed graph. It is used to reach a decision node always, whereas DFS can visit every reachable node during searching. According to our requirement we need to prune a sub-graph or part of the graph from the KB graph database. In this paper, we propose a multi-source DFS based pruning algorithm as the best suitable choice for pruning. The remainder of the paper is organized as follows. In Section II, a brief overview of the Disease-Symptom data model is given. Following that in Section III, a rigorous review and state-of-art of pruning algorithms are presented. In Section IV, DFS based pruning algorithm is described. Implementation and analysis of the proposed algorithm are presented in Section V. In Section VI, we finally conclude the paper.

1. Background

Mobile-assisted diagnosis scheme was proposed which was based on Integrated Management of Neonatal and Childhood Illness (IMNCI) [2]. In rural areas, health assistants follow the IMNCI chart booklet for the treatment of a sick child. This guideline tells health assistants how to take care of a sick child through the assessment, classification, identification of treatment, treatment of the sick child and follow-up care processes. The knowledge graph is the common platform to represent structured and unstructured data. Therefore, to implement the proposed system, we have first created a knowledge graph database and its data model. The data model contains the Diseases, Symptoms and some guidelines or Information which help to classify the appropriate symptoms for preliminary diagnosis of the disease. A knowledge graph is an appropriate representation here for the following reasons: (i) a particular disease is diagnosed on the basis of the combination of multiple symptoms or multiple symptoms may be presented against multiple diseases; and (ii) a disease is identified with stored symptoms through a series of key questions or information. Thus, due to the above requirements in remote health care system, a knowledge graph based on the Disease-Symptom data model has been built with some guidelines and help from the medical practitioners. In Figure 1, the Disease-Symptom data model is shown where S1, S2, S3…Sn represents the symptoms. I1, I2, I3 ...In are the information or questions and D1, D2 ...Dn are the set of diseases which are identified by the symptoms based on information. Information nodes are actually based on clinical signs or clinical assessment made by the health assistants through observations or querying the patients. Multiple diseases can have common symptoms; e.g. for diseases D1 and D2, S1 is the common symptom in this figure. The particular symptom with specific information leads to a particular disease.

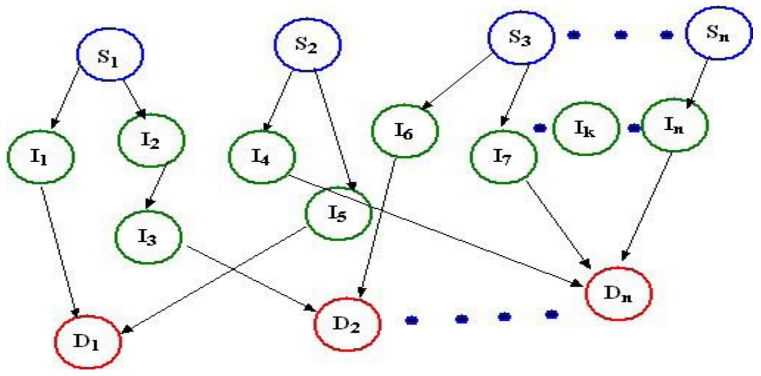


Fig.1 The Data model for Disease-Symptom relationship

Therefore the Disease-Symptom data model can be represented as a directed acyclic graph G=(V, E) where V=(S, I, D), where S, I and D are the sets of symptom nodes, information nodes and disease nodes respectively and E is the set of edges between S to I, I to I and I to D. For the mobile-assisted diagnosis scheme, this knowledge graph is stored in cloud or server. Whenever, a patient visits a health center or a health assistant at home, the health assistant notes the symptoms of diseases from patient's complaint and then uses the relevant parts of the knowledge graph to ask questions to the patient and makes relevant entries in our system based on the answers provided by the patient and finally arrives at a diagnosis. The entire process is accomplished through traversal of the graph-based knowledge base. As mentioned earlier, this Disease-Symptom knowledge graph is stored in cloud. Hence repeated queries made to the cloud require a large number of packet transmissions for the data. Further, the graph can be very large and complex when it stores large number of diseases. Thus, it is impossible task to bring down the entire knowledge graph on a mobile device due to the limited storage capacity of the device. Hence, pruning algorithms are necessary for accessing this Disease-Symptom database.

2. Related Work

Searching a complex heterogeneous graph database is difficult because of nonprofessional users and non-availability of any standard schema. Further, in order to retrieve a sub-graph from large graphs, efficient pruning technique is needed that reduces the size of graph or trees. A pruning algorithm can be exact and in-exact. Exact pruning algorithms are the techniques in which sub graph is produced by exact matches of a query graph or given set of inputs. But if sub graph is produced by best possible match like calculating cost function or using probability function, then it is called in-exact pruning algorithm. Traversal based or tree based, keyword search techniques, reachability query based techniques and partitioning of graph which is based on given input, are examples of exact pruning techniques. According to our requirement in the previous framework [1], we propose exact sub graph pruning algorithm in which a sub graph consisting of disease, symptom, and information will be generated by entering the input symptoms. The large Disease-Symptom graph database is basically a Directed Acyclic Graph (DAG) and the disease diagnosis is one kind of decision making procedure where primary symptoms lead to the exact sign of diseases. Therefore, reachability query based pruning algorithm is appropriate for our proposed work.

2.1 Reachability Query Based Pruning Algorithm:

A common problem in several graph based applications is to verify whether a vertex is reachable from another vertex or not. A reachability query r(u,v) asks whether a vertex v is reachable from a vertex u, i. e., whether there is a path from u to v in G. By the above property in a DAG, sub graph can be pruned. The required pruning algorithm should have low memory requirement and should be fast. It is also necessary that the algorithm should be accurate, and efficient. There are lots of existing well-known algorithms for sub graph generation which can be applicable to our Disease Symptom database. These algorithms have different worst case and index size complexities. The worst case and index size complexities for some of the well-known algorithms are as follows: for Transitive Closure [3] O(nm) and O(n2), Tree Cover [4] O(nm) and O(n2), for 2-Hop Cover [5] O(n3 ·|TC|) and O(nm1/2), and 3-Hop Cover [6] the values are O(kn2·|Con(G)|) and O(nk), Chain Cover [7] O(n2 + kn√k) and O(nk), for Path-Tree Cover [8] they are O(mk0) or O(nm) and O(nk) respectively.

2.2 DFS Based Pruning Algorithm:

The disease-symptom knowledge graph forms a large-scale complex graph. Querying the large graph database has lot of problems which are discussed earlier and inefficient too. Further, it is a difficult task for users to inspect a large number of matches produced from querying the graph database. In our proposed system, pruning algorithm plays a vital role to generate the sub graph which is needed to help the diagnosis of diseases. In this algorithm inputs are the Disease-Symptom database and the patient's symptoms. The output will be a sub graph which is related to this particular set of symptoms and also it will be classified after questioning the patient according to our proposed scheme [1]. It helps to improve response time by avoiding repeated queries to the cloud which requires transmission of a large number of packets and may be expensive and may delay the process. For diagnosis of a patient's disease, it is also an impossible task to offload the entire knowledge graph on a mobile device as it has limited storage and processing capacity. Hence, part of the knowledge graph is needed to be extracted onto the mobile device. The pruning algorithm proposed for our system is shown in Figure 2. In the first step, the symptoms(S) are entered and checked whether they are present in the graph G or not. If they exist, then the symptoms are added to the sub graph H. Also, all the vertices which have edges from symptoms S, are added to V_{list}. Next, again another vertex is chosen from V_{list} and the same procedure is repeated until the list V_{list} becomes empty. When a vertex is chosen from this list, it is checked whether the vertex is in H sub-graph. If the vertex is not in the sub graph, it is added to the sub graph H. All the vertices are added to V_{list}, if there are edges from this vertex.

3. Analysis and Implementation of Pruning Algorithm

Initially, in the Direct Acyclic Graph (DAG), three diseases, such as Diarrhea, Influenza and Tuberculosis are taken. For the implementation of this DFS based pruning algorithm, this graph database is represented as an adjacency list.

3.1 Analysis of Pruning Algorithm

For the proposed pruning algorithm, we use depth-first search technique. It is designed as multi source sequence DFS algorithm.

The best case time complexity of this algorithm is O(n) but worst-case time complexity is O(n2/64). In this algorithm, complexity of the lines 3-6 lines is O(n2/64) and lines 9-12 is O(n2/64). Lines 7-13 are executed inside while loop and worst-case time complexity is O(n2/64). Lines 2-13 are executed for the (ith) input symptom. Therefore, overall worst-time complexity is O(n2/64). However the complexity of simple DFS technique is O(n2/64) for only one source, but for multiple sources it is O(n2/64).

3.1.1 Algorithm

```
Start Loop 1:
Initialize variable i=0
for(int i=0 to n; i=i+64)  //divide the branches of the graph into 64 sub branches
{
Start Loop 2:
     Initialize variable j=0
     for(int j=0 to n-1)
     {
     int v=ts[j];  [initially ts=0, size of ts=n]
     cur[v]=0;    ////[current[vertices]=0]

           Start loop 3:
           if((i<=v and v<(i+64))
           {
           cur[v]=1ull<<(v-i);
           }
           End Loop 3.
     }
End Loop 2.
           start Loop 4:
               for(int j=0 to m-1)
               {
                    int u=a[egdesTs[j]], v=b[edges[j]];
                    cur[u] |= cur[v];  ////[ "|=" bitwise or operation]
               }
           End Loop 4.
               Start Loop 5:
                    for(int j=0 to n)
                    {
                    ans[j] +=popcount(cur[j]);
                    }
               End Loop 5.
}
End Loop 1.
```

II. Motivation

There are so many examples of real life graphs which are use to represent the real life research problems. Graphs provide an intuitive way to model of data in particular application. Compare with traditional relation model and XML model, graph has more power and plays an important role. Such as protein in molecules, linkage between web pages and internet and social network.

Our work based on mobile-assisted remote healthcare service delivery in which disease-symptom database stored in cloud and it is accessed by mobile devices. The knowledge base (KB) disease-symptom database is the combination of semi-structured or unstructured data. Disease-symptom database is represented by complex graphs which is formed big data.

The wide uses of graphs are mainly, retrieving information from complex, big graph in a timely and efficient manner. Many researchers have focused for unstructured or semi-structured data storage and analysis also.

Due to sheer size and complexity of knowledge base (unstructured) big data, it is a daunting task to query for this data. Searching from complex knowledge base data is not an easy task and impossible task especially for non-professional users. Either no standard schema is available or schemas become too complicated for users. Thus, in order to retrieve required knowledge base data or sub-graph from large graphs, efficient query algorithm or searching is important and writing efficient, proper query algorithms is another big challenge.

III. Survey

In graph research, particularly in sub-graph generation or pruning a sub-graph, a lot of research work has been done as application requirement and it is shown in figure 2.

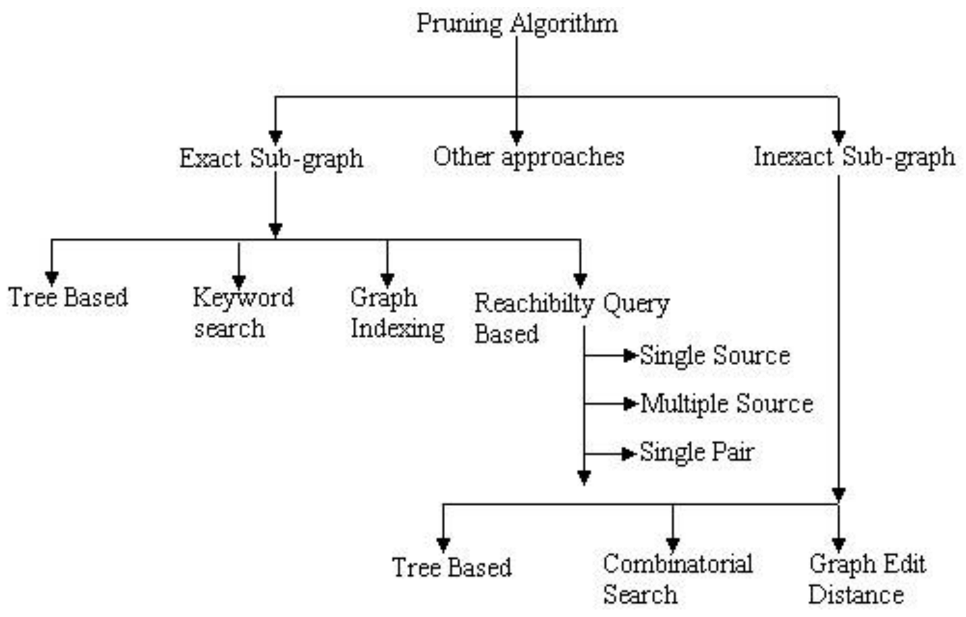


Fig.2 The variations of Sub-graph Pruning Algorithm.

There are two general categories of pruning algorithms: exact and inexact. There another category, other approaches which is not fall direct sub-graph pruning categories. Other approaches include decomposition methods, neural networks, genetic algorithms, methods based on bipartite matching and methods based on local properties. In sub-graph isomorphism problem according to query graphs, sub-graph is pruned in exact matching procedure. Exact matching seeks to find all embedding of a query graph in a data graph with the same structure and labeling. Exact Sub-graph also be pruned by the tree based, indexing based, keyword search based and reachability query. Inexact sub-graph is also be pruned by the sub-graph isomorphism are used that allow more latency when matching. Some use a cost function like edit distance to obtain a real value indicating how different two graphs are (Query graph and Data graph). Sub-graph also is pruned by tree based and also by combinatorial search.

IV. Broad observations

Reachability query based pruning algorithm:

A common problem in several graph-based applications is to verify whether a vertex is reachable from another. A reachability query r (u, v) asks whether a vertex v reachable from a vertex u, i. e., whether there is a path from u to v in G. By the above property in a directed graph or in DAG, sub-graph can be pruned.

In our proposed work, the sub-graph is to be pruned into mobile devices by input some set of symptoms and the pruned sub-graph consists of given symptoms, information and diseases.
The required pruning algorithm should be low memory based, fast and reducing search space of graph search also algorithm should be friendly, accurate, and efficient.

V. Objectives

In our proposed work, the sub-graph is to be pruned into mobile devices due to various constraints.

I. Frequent search or matching of sign or symptom cannot be possible on big, complex graph which is stored in cloud. Our requirement for proposed work, sub-graph has to be pruned by giving multiple symptoms.

II. Sub-graph pruning in mobile devices minimizes a lot of transmission delay or no. of message exchanges. So that it required minimum accessing time for further search or classifying appropriate symptoms.

III. Much more processing and transmission in mobile devices, lot of battery power wasted.

IV. Sub-graph solves all problem of memory requirement of mobile devices.

V. There are another challenges regarding pruning algorithm which is to be required minimum response time.

VI. Working Procedures

In any depth-first search of a (directed or undirected) graph *G = (V, E)*, for any two vertices *u* and *v*, exactly one of the following three conditions holds:

I. The intervals [d[u], f [u]] and [d[v], f [v]] are entirely disjoint, and neither unor v is a descendant of the other in the depth-first forest,

II. The interval [d[u], f [u]] is contained entirely within the interval [d[v], f [v]],and u is a descendant of v in a depth-first tree, or

III. The interval [d[v], f [v]] is contained entirely within the interval [d[u], f [u]],and v is a descendant of u in a depth-first tree.

We begin with the case in which d[u] <d[v]. There are two subcases to consider, according to whether d[v] < f [u] or not. The first subcase occurs when d[v] < f [u], so v was discovered while u was still grey. This implies that v is a descendant of u. Moreover, since v was discovered more recently than u, all of its outgoing edges are explored, and v is finished, before the search returns to and finishes u. In this case, therefore, the interval [d[v], f [v]] is entirely contained within the interval [d[u], f [u]]. In the other subcase, f [u] < d[v], and inequality implies that the intervals [d[u], f [u]] and [d[v], f [v]] are disjoint. Because the intervals are disjoint, neither vertex was discovered while the other was grey, and so neither vertex is a descendant of the other. The case in which d[v] < d[u] is similar, with the roles of u and v reversed in the above argument.

VII. Result Evaluation

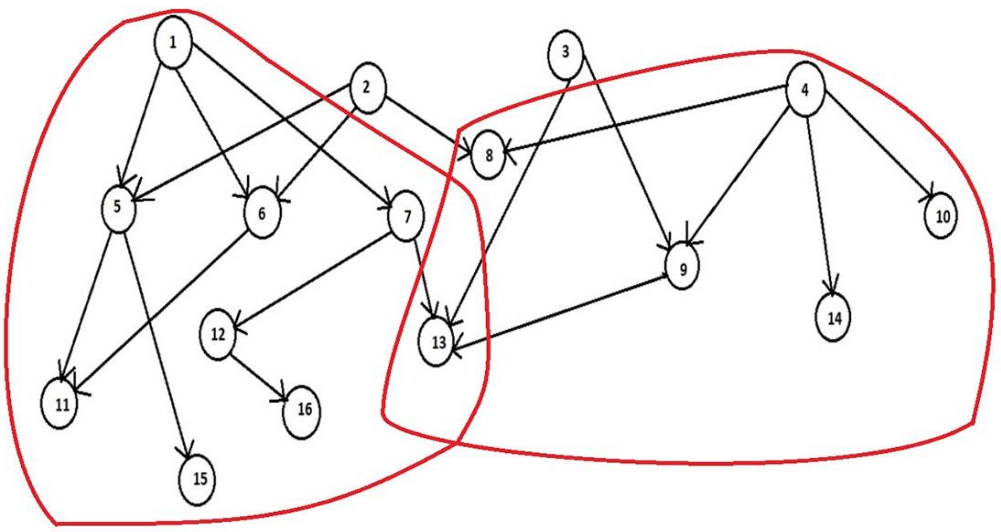
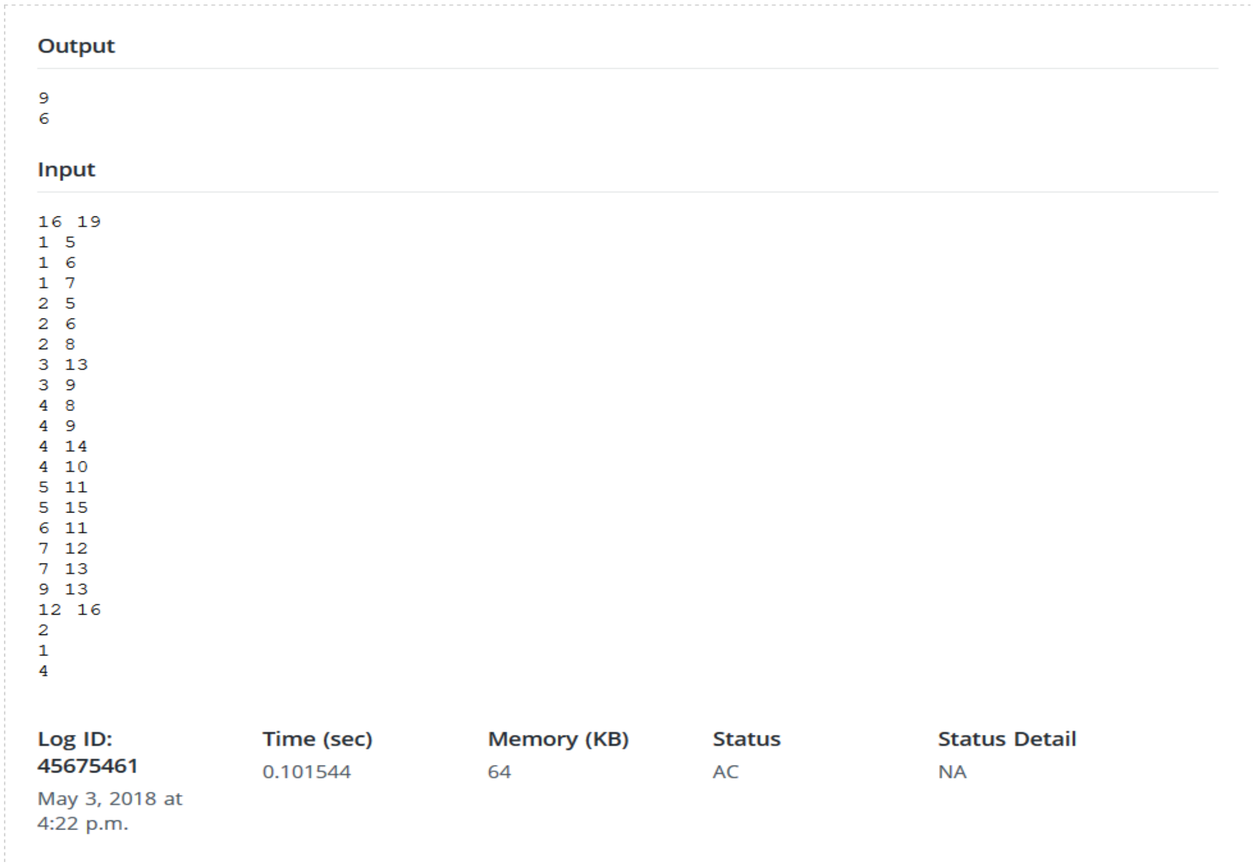


Fig.3 Nine and six nodes are pruned as shown in the above figure.



Output

```
9
6
```

Input

```
16 19
1 5
1 6
1 7
2 5
2 6
2 8
3 13
3 9
4 8
4 9
4 14
4 10
5 11
5 15
6 11
7 12
7 13
9 13
12 16
2
1
4
```

```
Log ID:         Time (sec)   Memory (KB)   Status   Status Detail
45675461        0.101544     64            AC       NA
May 3, 2018 at
4:22 p.m.
```

Fig.4 The output is shown above.

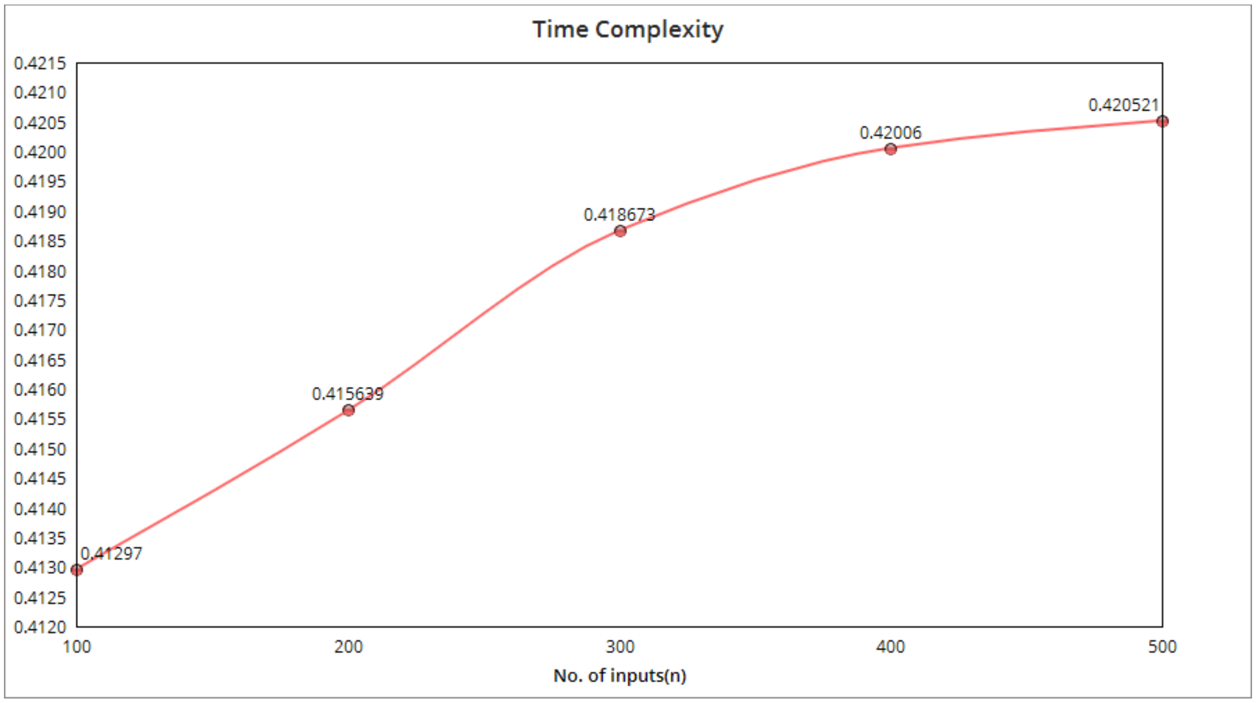
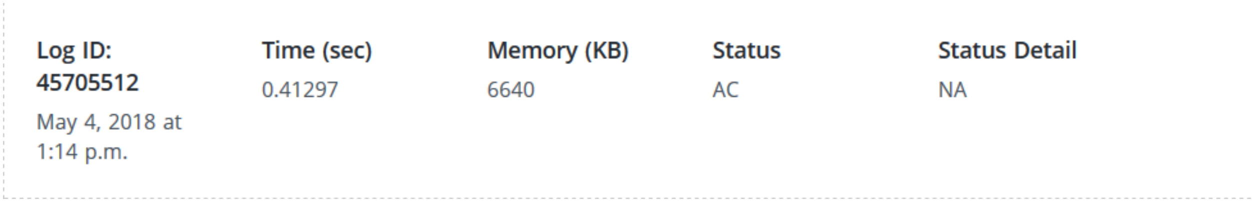
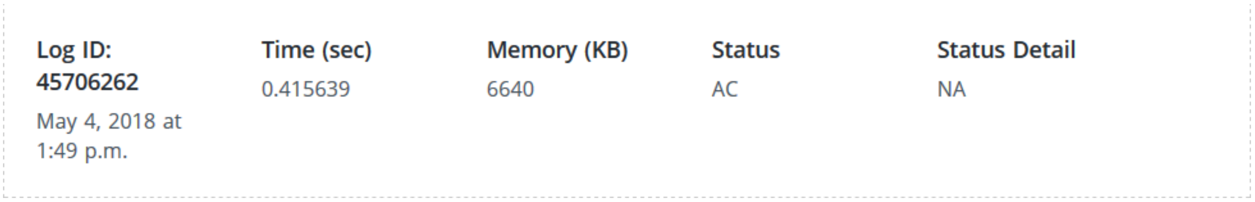


Fig.5 Time required to execute 100 to 500 inputs on a large data set of 50000 nodes.



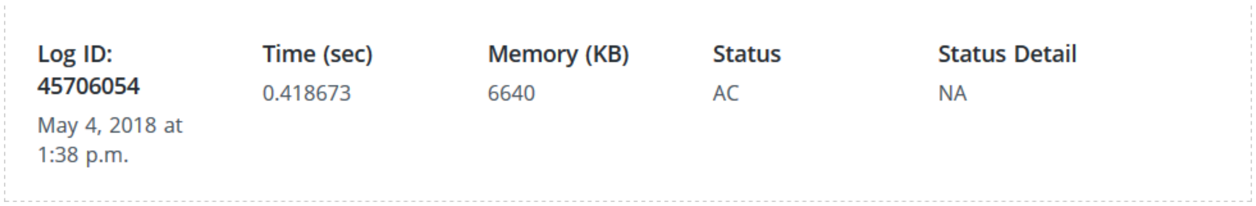
Log ID: 45705512	Time (sec)	Memory (KB)	Status	Status Detail
May 4, 2018 at 1:14 p.m.	0.41297	6640	AC	NA

Fig.5.a Execution time for 100 inputs.



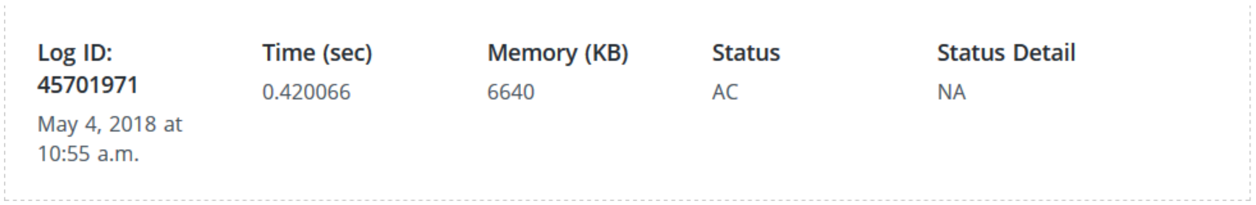
Log ID: 45706262	Time (sec)	Memory (KB)	Status	Status Detail
May 4, 2018 at 1:49 p.m.	0.415639	6640	AC	NA

Fig.5.b Execution time for 200 inputs.



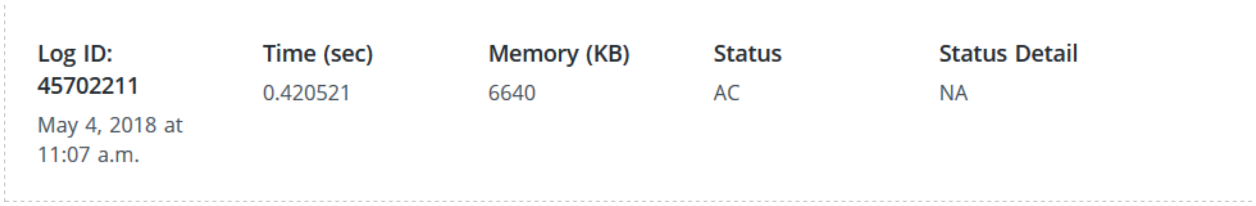
Log ID: 45706054	Time (sec)	Memory (KB)	Status	Status Detail
May 4, 2018 at 1:38 p.m.	0.418673	6640	AC	NA

Fig.5.c Execution time for 300 inputs.



Log ID: 45701971	Time (sec)	Memory (KB)	Status	Status Detail
May 4, 2018 at 10:55 a.m.	0.420066	6640	AC	NA

Fig.5.d Execution time for 400 inputs.



Log ID: 45702211	Time (sec)	Memory (KB)	Status	Status Detail
May 4, 2018 at 11:07 a.m.	0.420521	6640	AC	NA

Fig5.e Execution time for 500 inputs.

VIII. Scope for future work

There are two main areas in which future research can seek to improve the performance of our DAG algorithm. The simpler of these two metrics involves changing the thresh-hold parameters for merging in the model. In particular, relaxing the thresh-holds to allow for false classification, and using a metric that is not a simple integer threshold but instead one that is dynamically adjusted based upon the projected size of the node's theoretical example space could provide useful improvements. While, the idea of using a percentage-based thresh-hold was experimented with early in the design of the model, it was opted against in an effort to limit the scope of investigation, and due to the integer thresh-hold yielding better results on early test sweeps.

The other area that is ripe with possibility for improvement lies in bringing our method more in line with the established methodology of the Decision Graph field. In particular, the lack of a global view when selecting the next move, with the focus instead resting on the potential moves of a randomly selected node lead to less than optimal results. However, our method demonstrated gains over the traditional methods of Trees and Pruned Trees similar to early work on Graphs. Therefore a merging of the two different approaches could lead to even better results. The main downside to any research in this area lies in the added layer of computational complexity that this adds to the equation. By performing the calculations required for each step in computation for all open nodes, the complexity increases in magnitude by the number of open nodes at any point in time. Therefore, improving the

IX. Conclusion

In this paper, a multi-source DFS based pruning algorithm is proposed and implemented according to our previous data model discussed in [1]. For querying a complex knowledge graph, this DFS based pruning algorithm is investigated and validated according to our framework. Also it is compared with simple DFS based algorithm and concluded that multi-source DFS based pruning algorithm is better than the simple DFS based algorithm. Here, we are considering only few numbers of diseases. The overall graph is also very simple. Further, the DFSbased algorithm is also not efficient. Therefore, an efficient pruning algorithm is needed and should be compared with other pruning algorithms. The algorithms will be investigated with large set of diseases as a future direction of this work.

References:

1. Mondal, S., Mukherjee, N.: Mobile-assisted Remote Healthcare Delivery. In: Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC), IEEE, India (2016).

2. Integrated Management of Neonatal and Childhood Illness. http://www.who.int

3. Simon, K.:An Improved Algorithm for Transitive Closure on Acyclic Digraphs. Theor.Comput. Sci., vol. 58, pp. 325–346, Elsevier (1988).

4. Agrawal, R., Borgida, A., Jagadish, H. V.: Efficient Management of Transitive Relationships in Large Data and Knowledge Bases. SIGMOD Rec., v-18, pp. 253– 262, (1989).

5. Cohen, E., Halperin, E., Kaplan, H., Zwick, U.: Reachability and Distance Queries via 2-hop Labels. In: 13th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA,(2002).

6. Jin, R., Xiang, Y., Ruan, N., Fuhry, D.: 3-HOP: A High-compression Indexing Scheme for Reachability Query. In: International Conference on Management of Data, pp. 813–826, ACM SIGMOD (2009).

7. Bramandia, R., Choi, B., Ng, W. K.: On Incremental Maintenance of 2-hop Labeling of Graphs, In: IEEE Transactions on Knowledge and Data Engineering .Vol. 22, pp. 682-698, (2010).

8. Jin, R., Xiang, Y., Ruan, N., Wang, H.: On Efficiently Answering Reachability Queries on Very Large Directed Graphs. In: International Conference on Management of Data, pp. 595-608. ACM SIGMOD,(2008).