International Journal of Engineering Research and General Science Volume 3, Issue 2, March-April, 2015
ISSN 2091-2730

Design And Implementation Of Interactive Graph Simulator

Sachin Gajbhiye

Student, Dept. of Computer Science & Engg
Rajiv Gandhi College of Engg. & Research
Hingna Road, Nagpur ,India
e-mail: sach.ocean13@gmail.com

Nikita Godghate

Student, Dept. of Computer Science & Engg
Rajiv Gandhi College of Engg. & Research
Hingna Road, Nagpur,India

Swapnil Mahalle

Student, Dept. of Computer Science & Engg.
Rajiv Gandhi College of Engg. & Research
Hingna Road, Nagpur,India

Kalyani Bhoyar

Student, Dept. of Computer Science & Engg.
Rajiv Gandhi College of Engg. & Research
Hingna Road, Nagpur,India

Abstract— The field of mathematics plays important role in most of fields. The most important areas in mathematics is graph theory which is used in structural models. This structural arrangement of various objects or technologies leads to new inventions and modifications in the existing environment for enhancement in most fields. Graph simulator is concern with output of algorithms which can be applied on graph. There are many different algorithms which can be applied on graph and studying those algorithms theoretically is quite difficult. So, the graph simulator is the simulator, which helps us to learn different graph algorithms easy and get the output for the given input easily. These mechanisms which can easily solve graph algorithm according to the user requirements. The graph simulator is mainly dealing with the shortest path algorithms which will be sufficient to generate results as required. Those results will have efficiency in an execution as well as understanding.

Keywords— Simulator, TSP(Travelling Salesman Problem),BFD(Bellman-Ford),Dijkstra,GC(Graph Coloring),G=(V,E)-Graph having nodes as V and edges as E,dist-distance from particular ,cost is the weight on that edge.

INTRODUCTION

The term graph is an ordered pair G = (V, E) comprising a set V of vertices or nodes together with a set E of edges or lines, which are 2-element subsets of V (i.e., an edge is related with two vertices, and the relation is represented as an unordered pair of the vertices with respect to the particular edge). To avoid ambiguity, this type of graphs may be

described precisely as undirected and simple [11]. In one more generalized notion E is a set together with a relation of incidence that associates with each edge of two vertices. In another generalized notion, E is a multi-set of unordered pairs of vertices. The vertices belonging to an edge are called the ends, endpoints, or end vertices of the edge. A vertex may exist in a graph and not belong to an edge. V and E are usually taken to be finite, as many of the well-known results are not true for infinite graphs because many of the arguments fail in the infinite case. In computer science, graphs are used to represent networks of communication, data organization, computational devices, the flow of computation, etc. For instance, the link structure of a website can be represented by a directed graph, in which the vertices represent web pages and directed edges represent links from one page to another. A similar approach can be taken to problems in travel, biology, computer chip design, and many of the other fields. The development of algorithms to handle graphs has been therefore it is a major interest in computer science. The transformation of graphs is often formalized and represented by graph rewrite systems. Graph theory is also widely used in sociology as a way, for example, to measure actors' prestige or to explore rumor spreading, notably through the use of social network analysis software. Under the umbrella of social networks, there are many different types of graphs they are Acquaintanceship and friendship graph [9]. As there are very less a tool which simulates graph algorithms, the project aims to create such an interface that accepts number of edges & vertices from user along with their weights as input and produce the graph according to give input. It will be extended towards simulation of graph based algorithms such as TSP, Graph coloring, Bellman-ford algorithm, Dijkstra algorithm. The system will also be applicable to solve the graph related problems like finding shortest path, graph coloring. Graph coloring is utilized in resource allocation, scheduling the paths, walks and circuits in graph theory are used in tremendous applications such as traveling salesman problem, database design concepts, resource networking [12]. This leads to the development of new algorithms, which can be used in many applications.

REMAINING CONTENTS

I. PROPOSED SYSTEM

The system consists of three sections whichare as follows:

A. Input section

B. Graph section

C. Algorithm section.

A. Input section

This section is used for user input. User has to provide number of nodes and edges between the nodes along with their weights. The edges between the nodes are user defined so user has to provide edges according to requirement

B. Graph Section

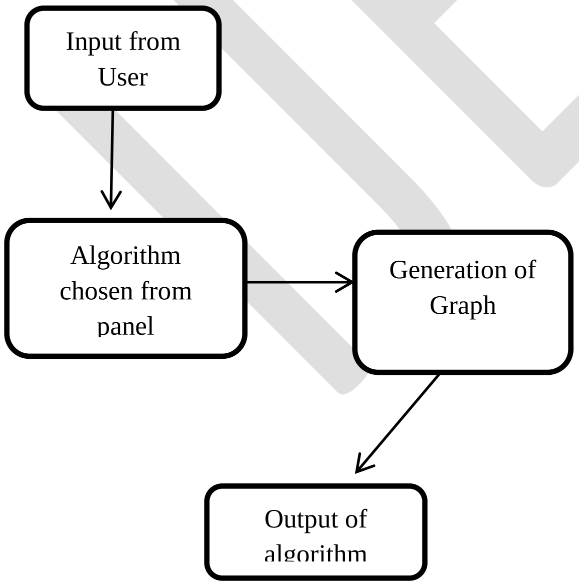
After the entire input gather from user there is a provision in the system that it automatically generate the graph.

C. AlgorithmSection

In this section system provides graph algorithms in which User has to select algorithm for generated graph and the output of the selected algorithm will be displayed.

II. FLOW DIAGRAM

Fig(1) shows how the Graph simulator will works:



Fig(1): Process flow of proposed system

III. PROBLEM ALGORITHMS

The major role of Graph in Computer Science is to develop the graph Algorithms. the algorithms are used to solve problems that are models in form of graph. These algorithms can solve the graph theoretical concepts which are used in computer science applications [9].

Some Algorithms are as follows:

A. Graph coloring algorithm
B. Travelling salesman problem
C. Bellman-ford Algorithm
D. Dijkstra Algorithm

A. Graph coloring algorithm

Given a graph G=(V,E) where V={V1,V2,…Vn} is the set of vertices or the nodes is the set of edges and then assign colors 'm' to the nodes according to following constraints:

The first node V1 is assign the color number 1.Once the node V1 has assigned color number 1,Vi+1 cannot assigned same color number. Now the second constraint is number of color should be less than equal to the 'm' where m is integer value of colors such as 1 for red,2 for green and so on[8].

Algorithm for Graph coloring

```
Graph_ coloring (index i)
{
Enter  Number of colors;
If (processing (i))
 If (i= =n)                    \\ plainer graph logic
Display  vcolor[i] through vcolor[n]
 Else
For(color=1;color<=m;coor++)
{
        Assign color to next immediate node;
m.coloring (i+1);
}
}
```

B. Travelling salesman Problem

Consider G<V, E> graph and Cij be cost of edge (i, j) ,Cij>0, if edge exists otherwise edge does not exist.

Let n be number of vertices.

The objective of this algorithm is to carry a tour starting with a vertex 'I' and ending with same vertex (cycle), visiting each vertex only once; such that cost of tour should be minimum[3].

Algorithm for Travelling Salesman Problem

Consider the matrix of Figure 1. The entry at position (i,j), say c(i,j), represents the cost (distance) for going from city i to city j. A tour is a set of city pairs, e.g.,

t = [(1,3) (3,2) (2,5) (5,6) (6,4) (4,1)] which spell out a trip that goes to each city once and only once. Let z be the cost of a. tour. From Figure 1 it may be seen that the above tour would cost:

z = 43 + 13 t 30 + 5 + 9 + 21 = 121.

If a constant is subtracted from each element of the first row of Figure 2, that constant is subtracted from the cost of every tour. This is because every tour must include one and only one element from the first row. The relative costs of all tours, however, are unchanged and so the tour which would be optimal is unchanged. The same argument can be applied to the columns. The process of subtracting the smallest element of a row from each element of a row will be called reducing the row. Thus the first row in Figure 2 can be reduced by 16. Note that, in terms of the unreduced matrix, every trip out of city 1 (and therefore every tour) will have a cost of at least 16. Thus, the amount of the reduction constitutes a lower bound on the length of all tours in the original matrix.

-	27	43	13	30	20
7	-	16	1	30	25
20	13	-	35	5	0
21	16	25	-	18	18
12	48	27	48	-	5
23	5	5	9	5	-

Figure(2): Example of Travelling Salesman Problem

Evaluation

Step 1: Reduce the rows and columns of the cost matrix. Save the sum of the reductions as a lower bound on the cost of a tour.

Step 2: Given the node, say X, from which to branch next, and given the cost matrix associated with X, find the city pair (k,l) which maximizes and extend the tree from X to a node k,.{. Add Q(k.X) to the lower bound of X to set the lower bound of the new node.

Step 3: If the number of city pairs committed to the tours of X is n-2, go to Step 6. Otherwise continue.

Step 4; Finish the branching based on (k,^) by extending the tree from X to a node k,^m,p. Here (m,p) is the city pair which would join the ends of the longest connected path involving (k,Q) in the set of committed city pairs of the new node. Delete row k and column \ in the cost matrix and set c(m,p) = c/j . Reduce rows and columns if possible and add the amount of the reduction to the lower bound of X to set the lower bound for the new node.

Step 5: If no node has been found which contains only a single tour, find the terminal node with the smallest lower bound and return to Step 2 for branching. Otherwise continue.

Step 6: If entry to this step is from Step 3, the next node is k,l, m ,P where (m,p) is the only city pair left after crossing out row k and column X. The node contains a single tour. Calculate its cost. Entry to the step.

C. Bellman Ford Algorithm

The Bellman Ford algorithm solves single source short path problem in the case in which edge may have negative weights. If the graph has a negative weight cycle, the algorithm will tell us otherwise it gives us a shortest path from source to every node.

If there is a negative weight cycle on some path from 'u' to 'v' in a graph G=<V,E>,then shortest path may not exists between 'u' to 'v' because algorithm may not terminate due to negative weight cycle[13].

Algorithm

```
Bel_ford (v, cost, dist, n)

{

     For (i=1 to n)

Dist[i] =cost [v,i];

For (k=2 to n-1)
For each u such that u! =v and u has at least one incoming edge
  For each (I, u) in the graph
If (dist[u]>dist[i]+cost[I,u])
Dist[u] =dist[i] +cost [I,u];
}
```

Overall complexity of this algorithm isO (n^3).

Evaluation
Step1. Initialize distance array for all vertices. i.e. dist[i]=cost[v,i];

.Step2. Note there is single source and multiple destinations for each destination 'u' such that u!=v ('v' is source) and 'u' has at least one incoming edge.
Assumptions- there are intermediate nodes present between source and destination (intermediate is 'I' which is from 1 to n)

```
For each (i,u) in graph

  If dist[u]>dist[i]+cost[I,u] then

Add the dist[i] and cost[I,u] to get Dist[u];
```

D. Dijsktra Algorithm

The Dijkstra algorithm is a greedy class of algorithm. Which works as finding optimum value in each step[10].The Dijkstra algorithm finds the path with lowest cost (i.e. the shortest path) between that vertex and every other vertex (although Dijkstra originally only considered the shortest path between a given pair of nodes). It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined [2].

Algorithm

```
Dijkstra_algorithm
{
distance to source vertex s  is zero then
for all      v ∈ V–{s}
do
{
       Set all other  distanceof v as  infinity;
   The set of visited vertices S is initially empty;
 Q the queue  initially contain set all vertices;
}while
      Queue ’Q’ is not empty;
do
     Select the element of Q with the  min. distance;
        Add up to list  of visited vertices;
 for all v∈ neighbors[u]
 do
         if new shortest path foundthen
  set new value  of shortest  path;
Return   Dist}
```

CONCLUSION

The aim of the system is to provide interactive environment to study graph algorithms which can be easy to understand for user. This system can be used in educational institutes for practical implementation of graph algorithms which they study theoretically in classrooms.

REFERENCES:

Huberto Ayanegui and Alberto Chavez-Aragon, “A complete algorithm for solving graph coloring problem”, Tlaxcala, Mexico, 2011.

Melissa Yan, Dijstra algorithm,2002.

John D. C. Little, Dura W. Sweeney, , “An algorithm for travelling salesman problem”, Massachusetts Institute of Technology, March 1, 1963.

Corman H. Thomas, Leiserson E. Charles, Rivest L. Ronald, Stein Clifford, “Introduction to Algorithms”, Second Edition McGrawHill Book Company.

Bondy, J.A.; Murty, “Graph Theory”, Springer U.S.R. (2008).

AnindyaJ.Pal, Samar S.Sarma, Biman Ray, “CCTP, Graph Coloring algorithms – Soft computing Solutions”, IEEE, 2007.

Sven Dickinson, Pelillo, RaminZabih, “Introduction to the special section on graph algorithms in computer vision”, IEEE on patternanalysis, Vol 23 No. 10, September 2001.

Frank Thomson Leighton,” A Graph ColoringAlgorithm for large scheduling problems”, JOURNAL OF RESEARCH of the Notional Bureau of Standards,vol. 84, no. 6, November-December 1979.

S.G.Shirinivas,S.Vetrivel,Dr.N.M.Elango, ”Applications of Graph Theory in Computer Science an Overview”, International Journal of Engineering Science and Technology, Vol. 2(9), 2010, 4610-4621.

Amir Kamil,” Graph Algorithms”, CS61B, Spring , UCBerkeley,2003.

http://en.wikipedia.org/wiki/Graph_Theory

http://en.wikipedia.org/wiki/Applicationsof Graph_coloring

http://en.wikipedia.org/wiki/Bellmen Ford