ARTIFICIAL INTELLIGENCE

CSE 3013

J-COMPONENT REPORT

TOPIC: VERTEX COVER

TEAM MEMBERS:

ANANYA BAL 16BCE0063

PRERNA JOTWANI 16BCE2299

CHAITANYA BHOJWANI 16BCE0082

FACULTY: Prof. RISHIN HALDAR

Papers Chosen and Implemented

Paper - 1

Dahiya, Sonika. “A New Approximation Algorithm for Vertex Cover Problem.” *2013 International Conference on Machine Intelligence and Research Advancement*, 2013, doi:10.1109/icmira.2013.100.

Abstract: The vertex cover problem is an NP-Complete optimisation problem, so there is no proof of existence of an optimal algorithm. Therefore, a lot of research has been done in this area yet there is no optimal solution. Like all NP-hard solutions, this paper attempts an optimal solution to the Vertex Cover problem. A brief introduction to existing algorithms is given. This is followed by a proposed variation to the Alom’s algorithm with heuristics relating to pendant vertices. The proposed algorithm first gather the adjacent vertices of pendant vertices into the solution set and then proceeds to the Alom’s algorithm. The algorithm provides a near optimal solution for unweighted graphs and outperforms the existing approximation algorithm.

Paper - 2

Gajurel, Sanjaya, and Roger Bielefeld. “A Simple NOVCA: Near Optimal Vertex Cover Algorithm.” *Procedia Computer Science*, vol. 9, 2012, pp. 747–753., doi:10.1016/j.procs.2012.04.080.

Abstract: This paper proposes a Near Optimal Vertex Cover Algorithm (NOVCA) which produces an optimal or near optimal vertex cover for any known undirected graph G (V, E) in polynomial time. At each step of the algorithm, the vertex with the minimum degree is identified, all vertices adjacent to the vertex of minimal degree are repeatedly added to the vertex cover VC. In case of a tie, it selects the adjacent vertex having the maximum sum of degrees of its neighbours. The bounds on the size of the minimum vertex cover as well as polynomial complexity are experimentally verified.

1. **Approximate Vertex Cover Algorithm**

   This algorithm works as follows:

   1. Set C $\leftarrow \phi$
   2. E’ $\leftarrow$ E[G]
   3. While E’ $\neq \phi$
   4. Begin
   5. let (u, v) be an arbitrary edge of E’
   6. C $\leftarrow$ C U {u, v}
   7. every edge incident on either u or v, remove it from E’
   8. End while
   9. Return C

2. Aloms Algorithm

This algorithm works as follows:

1. C’ $\leftarrow \Phi$
2. E’ $\leftarrow$ E[G]
3. While E’ $\neq \Phi$
4. Begin
5. M $\leftarrow$ vertex which as maximum degree
6. If (more than one vertex has maximum degree)
7. Begin
8. M $\leftarrow$ choose that maximum degree vertex which covers at least one such edge that is not covered by other maximum degree vertices.
9. End if
10. C’ $\leftarrow$ C’ U M
11. Delete all the edges incident on M.
12. Compute degree of each vertex in this new graph
13. End while
14. Return C’

3. Proposed Algorithm in Paper 1 (New Algorithm)

1. Set VC $\leftarrow \phi$
2. Set E’ $\leftarrow$ E[G]
3. Compute degree of each vertex
4. P $\leftarrow$ set of pendant vertices in given graph
5. While P $\neq \phi$
6. Begin
7. Select any vertex u P
8. v $\leftarrow$adj(u)
9. VC $\leftarrow$ VC U v
10. E’ $\leftarrow$ E’ /v
11. P$\leftarrow$P/u
12. End while
13. While E’ $\neq \phi$
14. Begin
15. M $\leftarrow$ maximum degree vertex

16. If (more than one maximum degree vertex)
17. Begin
18. v ← choose that maximum degree vertex which covers at least one edge that is not covered by other maximum degree vertices
19. End if
20. E' ← E' /v
21. Compute degree of each vertex
22. End while
23. Return VC

4. Near Optimal Vertex Cover Algorithm (NOVCA)

```
Declarations:
    V is the set of vertices
    of G E is the set of edges
    of G
    deg[V] is an integer array indexed by V for
           a set of vertices V
    sum_adj_deg[V] is an integer array indexed by
                  V for a set of vertices V
    VC is the set of vertices comprising a vertex
    cover Q_sum_adj__deg is the set of vertices having min
    deg[V]
            (local variable in GetMinVertex())

Functions:
    Degree(v) is the degree of the vertex v ε V
    Adj(v) gives the set of vertices that are
           adjacent to v ε V
    GetMinVertex() identifies the next adjacent
                  vertices to include in the cover

    Heap_MIN(deg) returns the value of min.
    deg[V] HEAP_MAX(Q_sum_adj_deg) returns the vertex
    having max
                          Q_sum_adj_deg


   for each vϵ V {
      deg[v] =
      Degree(v)
   }

   for each vϵ V {

     sum_adj_deg[v] = Σ_{v' ε Adj(v)} deg[v']
  }
```

```
E′  = E VC = ϕ

 while  (E′   ϕ){
v_c = GetMinVertex(deg,
sum adj deg) VC = VC + {
Adj(v_c) }
for       each       v
Adj(Adj(v_c)){ E' = E −
{   (adj(v_c),    v)    }
deg[v] = deg[v] − 1
  }
V = V − { Adj(v_c) }
for each v  V{
  If (Adj(v) ==   ) con-
  tinue sum_adj_deg[v] =
                v′ Adj(v)deg[v′]
  }
 } //end while
/// Magic Function GetMinVer-
tex() Declarations /// Vertex
GetMinVertex(deg,
sum_adj_deg){
   Q_sum_adj_deg =


vmin_deg =
HEAP_MIN(deg) for
each v    V{
 If (deg[v] ==
  vmin deg) Q_sum_adj_deg =
  Q_sum_adj_deg + {v}
  }
  return Heap_MAX(Q_sum_adj_deg)
 }
```

5. Our proposed algorithm (Modification of New Algorithm in Paper 1)

Step 1: Check for pendant vertices in the graph (vertices with degree = 1).

Step 2: If yes, go to Step 4, else got Step 8

Step 3: Make a set of all the pendent vertices.

Step 4: Calculate the set of adjacent vertices for each of the pendent vertices in the set.

Step 5: Add the set of adjacent vertices made in step 4 to vertex cover.

Step 6: Remove all the edges connected to vertices which have been added to vertex cover.

Step 7: With the updated graph, if all edges are covered then stop, else Go to Step 1

Step 8: Find the maximum degree vertex from the graph.

Step 9: In case of tie, choose that maximum degree vertex which covers at least one such edge that is not covered by other maximum degree vertices together.

Step 10: Add maximum degree vertex to the vertex cover.

Step 11: Remove all the edges connected to vertices which have been added to vertex cover.

Step 12: With the updated graph, if all edges are covered then stop, Go to Step 1.

Results

Dataset	Algorithm	Number of Vertices	Solution Set Size	Solution Set
Figure 2	APPROX-VERTEX-COVER algorithm	7	6	{ e, f, c, d, b, g}
Figure 2	Alom's algorithm	7	3	{ f, d, b}
Figure 2	New proposed algorithm	7	3	{ b, f, d}
Figure 2	Our proposed algorithm	7	3	{f, b, d}
Figure 2	NOVCA	7	3	{f, d, b}
Figure 3	APPROX-VERTEX-COVER algorithm	9	7	{ e, f, a, b, c, d, h, i}
Figure 3	Alom's algorithm	9	5	{ e, a, c, f, h}
Figure 3	New proposed algorithm	9	4	{ b, d, f, h}
Figure 3	Our proposed algorithm	9	4	{b, d, f, h}
Figure 3	NOVCA	9	4	{b, d, f, h}
Figure 4	APPROX-VERTEX-COVER algorithm	17	14	{ a, b, d, j, e, k, f, l, g, o, h, p, i, q}
Figure 4	Alom's algorithm	17	8	{ b, f, c, d, e, g, h, i}
Figure 4	New proposed algorithm	17	7	{ d, e, f, g, h, i, a}
Figure 4	Our proposed algorithm	17	7	{d, e, f, g ,h, i, a}
Figure 4	NOVCA	17	7	{f, d, e, g, a, h, i}
Figure 5	APPROX-VERTEX-COVER algorithm	25	16	{ a, e, b, c, g, h, i, j, k, n, m, q, p, t, r, s }
Figure 5	Alom's algorithm	25	13	{ s, n, f, g, q, i, j, k, a, c, p, y, d}
Figure 5	New proposed algorithm	25	13	{ q, s, f, g, n, i, k, p, a, c, d, j, t}
Figure 5	Our proposed algorithm	25	13	{q, s, n, t, f, g, d, b, e, j, m, k, p}
Figure 5	NOVCA	25	13	{s, q, n, t, d, g, b, i, j, e, f, l, o}
Figure 6	APPROX-VERTEX-COVER algorithm	25	24	{0,6,1,7,2,8,3,9,4,10,5,11,12,18,13,19, 14,20,15,21,16,22,17,23}
Figure 6	Alom's algorithm	25	13	{24,6,8,10,18,20,22,7,9,11,13,15,17}
Figure 6	New proposed algorithm	25	13	{6,7,8,9,10,11,24,18,20,22,13,15,17}
Figure 6	Our proposed algorithm	25	12	{6,7,8,9,10,11,18,19,20,21,22,23}
Figure 6	NOVCA	25	12	{6,8,10,19,21,23,7,9,11,18,20,22}

We can see that in Figure 6, our proposed algorithm works better than the new algorithm in paper 1 as it gives a more optimal set of 12 vertices compared to 13 vertices by the latter.

User Manual

This user manuals guides us through the five possible algorithms we have executed in our project, Namely

1. Approximation Algorithm
2. Aloms Algorithm
3. New Algorithm
4. The Algorithm we have proposed (Our Algorithm)
5. Near Optimal Vertex Cover Algorithm

- For All the above-mentioned algorithms, separate folders have been made.

- In each algorithm folder, there is a file named “Algorithm_name”.java

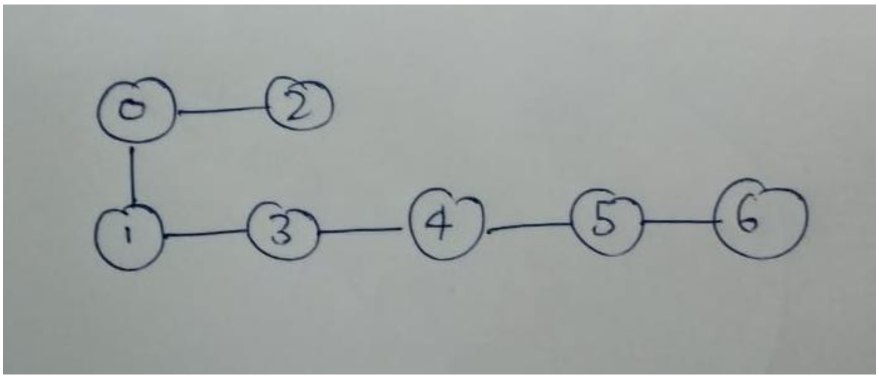
- On compiling and running the java file, we get a menu driven program to select a graph from 6 input graphs.

- The code prints out the vertex cover for the selected graph by the chosen algorithm.

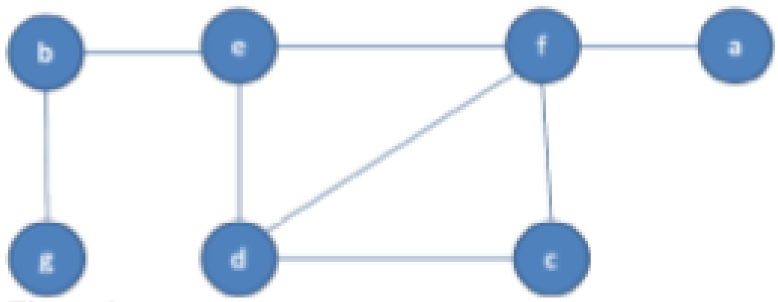
- Choose the graph from the given choices.

The graphs and their corresponding numbers are given below:

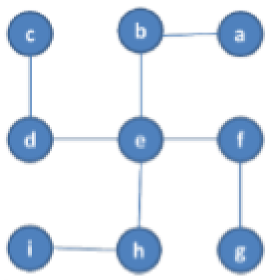
Graph 1:



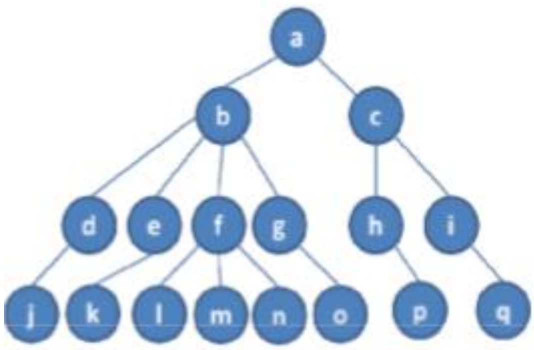
Graph 2:



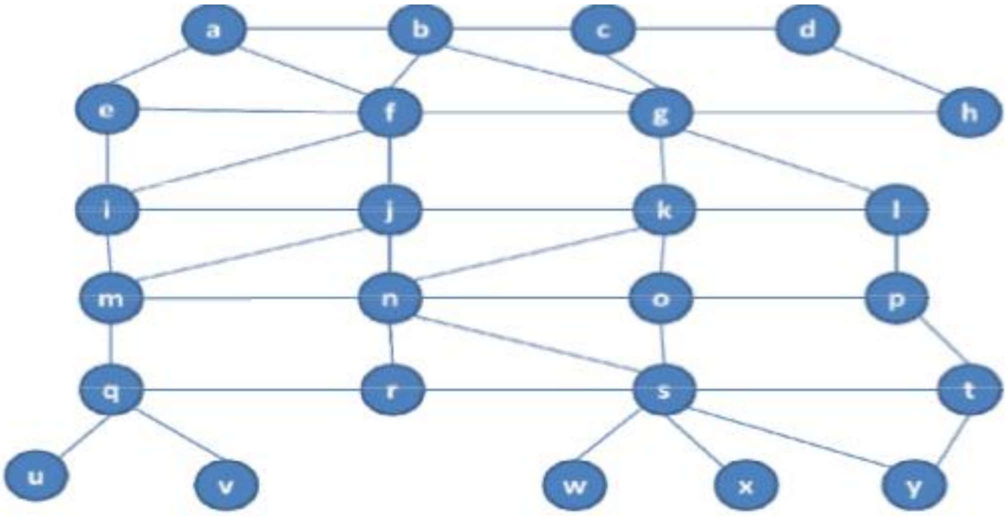
Graph 3:



Graph 4:



Graph 5:



Graph 6: