NAVPREET KAUR

Student ID V0082334

Description :

G[NMAX][MMAX] is 2d array to store read graph.

dominated[MMAX] is working as my tabu list .It is storing all the dominated vertices which are dominated by a red vertex. So that Next red vertex will not be same vertex and one which is already dominated by red vertex.

min_dom_Set[MMAX] storing minimum dominated set found so far the graph.

num_dominated[NMAX] keep track of how many times a vertex is dominated.

dom_set[MMAX] stores partial dominating set for my graph.

graph_number keep track of index of graph.

deleted[MMAX] keep track of all the red vertices which are deleted while finding for better dominating set by making changes in current solution.

size_del stores the size of deleted array.

size_dom is the size of my partial dominating set.

min_size is the size of minimum dominating set.

The evaluation function that this heuristic is using depends upon total number of vertices of graph. Each time if a vertex have num_dominated >1 than that vertex is considered to make changes in current solution.

1. Declare n, m, min_size, graph_number, G[NMAX][MMAX], dominated[MMAX], dom_set[MMAX], min_dom_set[MMAX],num_dominated[NMAX]
2. Intialise graph_number ← 0
3. while read_graph
4. Increment graph_number
5. Display graph_number,n
6. find_dom_set(n, m, G, dominated, dom_set, min_dom_set, num_dominated, min_size)
7. print_ver(1, &min_size,n,min_dom_set)
8. End while

```
9.  Procedure INTEGER read_graph(INTEGER n,INTEGER m,INTEGER G[NMAX][MMAX])
10.         Declare i, j, u, d
11.         if read n ≠1
12.                 return 0
13.         End if
14.         if read n > NMAX
15.                 Display Increase NMAX and recomplie
16.                 return 0
17.         End if
18.         m ← (n+31)/32
19.         For i=0 to n-1
20.                 For j=0 to m-1
21.                         G[i][j] ← 0
22.                 End For
23.         End For
24.         For i=0 to n-1
25.                 if read d ≠ 1
26.                         return 0
27.                 End if
28.                 For j=0 to d-1
29.                         if read u ≠ 1
30.                                 return 0
31.                         End if
32.                         ADD_ELEMENT(G[i], u);
33.                         ADD_ELEMENT(G[u], i);
34.                 End For
35.         End For
36. End Procedure read_graph


37. Procedure INTEGER find_dom_set(INTEGER n,INTEGER m,INTEGER
    G[NMAX][MMAX],INTEGER dominated[MMAX],INTEGER dom_set[MMAX],INTEGER
    min_dom_set[MMAX],INTEGER num_dominated[NMAX],INTEGER min_size)
38.         Declare u
39.         Intialise u<-0
40.         random_dom_set(n,m,G,dominated,dom_set,min_dom_set,num_dominated,mi
                                n_size)
41.         while u < n
42.                 better_dom(n, m, u, G, dominated, dom_set, min_dom_set,
                                num_dominated, min_size)
43.                 Increment u
```

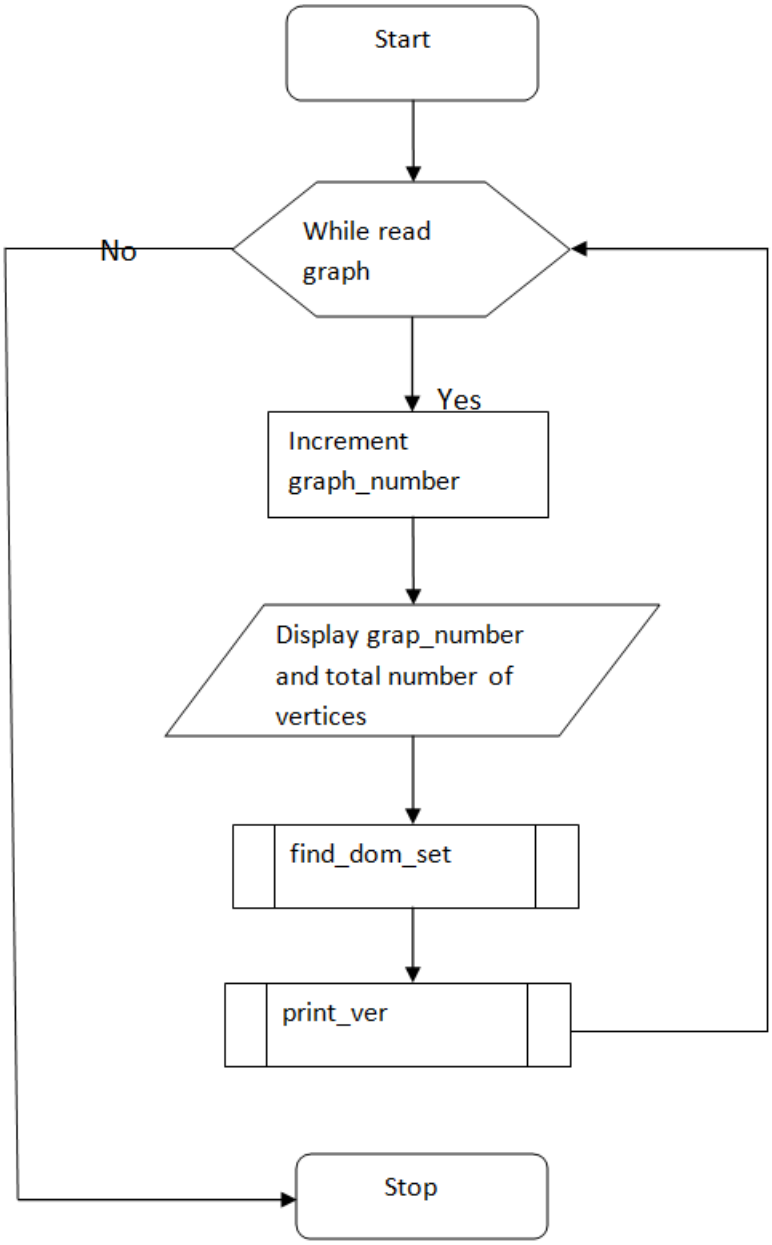
```
44.         End while
45. End Procedure find_dom_set
46. Procedure INTEGER random_dom_set(INTEGER n,INTEGER m,INTEGER
    G[NMAX][MMAX],INTEGER dominated[MMAX],INTEGER dom_set[MMAX],INTEGER
    min_dom_set[MMAX],INTEGER num_dominated[NMAX],INTEGER min_size)
47.         Declare i,j,u,size,d_size
48.         Intialise size <-0
49.         For i=0 to m-1
50.                 dominated[i]<-0
51.                 dom_set[i]<-0
52.                 min_dom_set[i]<-0
53.         End For
54.         For i=0 to n-1
55.                 ADD_ELEMENT(G[i], i)
56.                 num_dominated[i]=0;
57.         End For
58.         For i=0 to n-1
59.                 u<-i
60.                 if not IS_ELEMENT(dominated,u)
61.                         ADD_ELEMENT(dom_set,u)
62.                         Increment size
63.                         For j=0 to m-1
64.                                 dominated[j] <- dominated[j] or G[u][j]
65.                         End For
66.                         For j=0 to n-1
67.                                 if IS_ELEMENT(G[u],j)
68.                                         num_dominated[j] <- num_dominated[j]+1
69.                                 End if
70.                         End For
71.                 End if
72.         End For
73.         For j=0 to m-1
74.                 min_dom_set[j] <- dom_set[j]
75.         End For
76.         min_size <- size
77.         print_ver(0, min_size,n,min_dom_set);
78. End Procedure random_dom_set
79. Procedure INTEGER better_dom(INTEGER n,INTEGER m,INTEGER u,INTEGER
    G[NMAX][MMAX],INTEGER dominated[MMAX],INTEGER dom_set[MMAX],INTEGER
    min_dom_set[MMAX],INTEGER num_dominated[NMAX],INTEGER min_size)
80.         Declare i, j, n_choice, choice, size_del, size_dom, n_domi, store, deleted[MMAX]
```

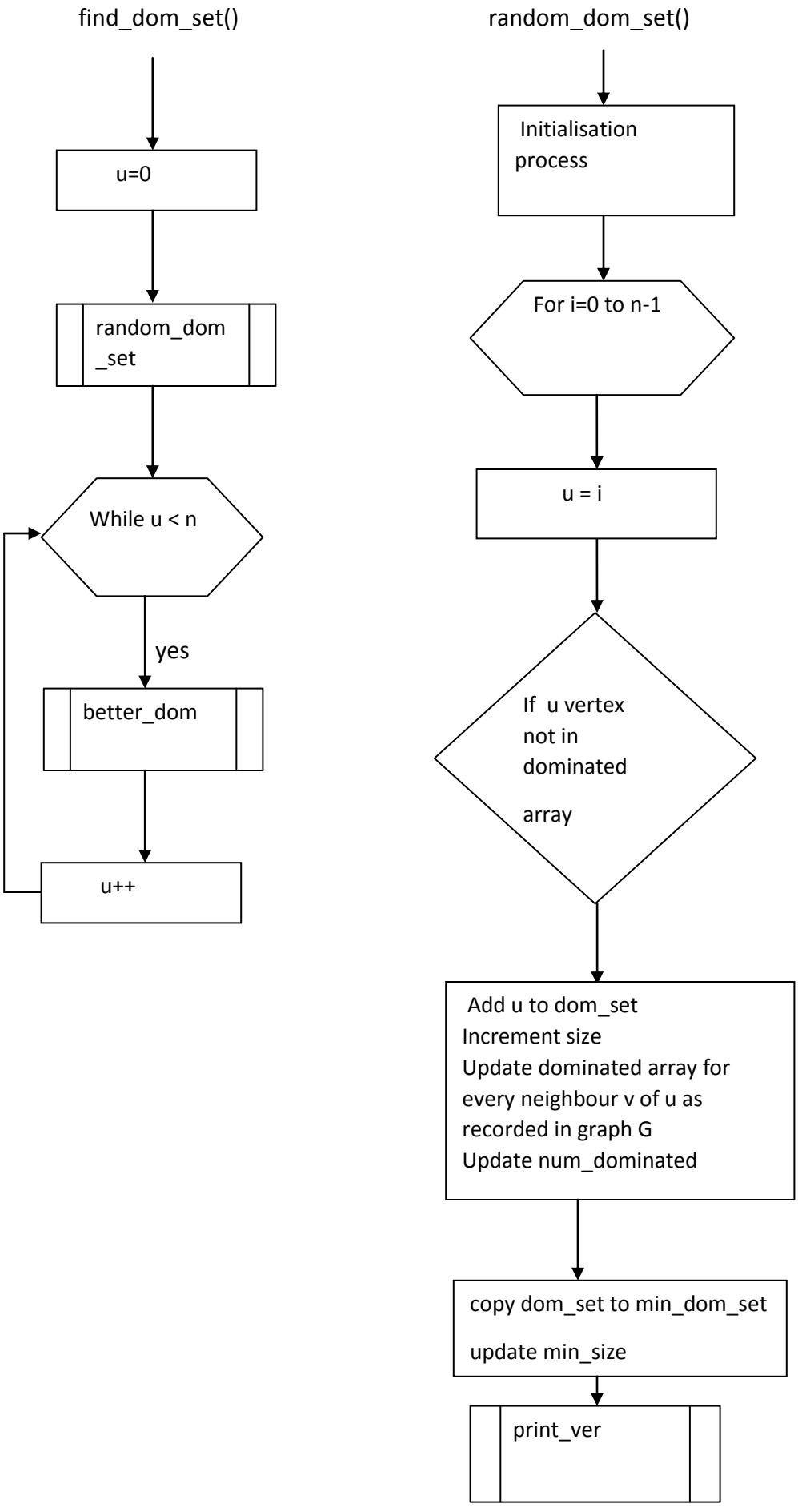
```
81.     Intialise  n_choice <- 0, choice <- 0, size_del <- 0, n_domi<-0
82.     For j=0 to m-1
83.         deleted[j]<-0
84.     End For
85.     if num_dominated > 1
86.         For i=0 to n-1
87.             if IS_ELEMENT(G[u],i)
88.                 num_dominated[i]=num_dominated[i]+1
89.             End if
90.         End For
91.         For i=0 to n-1
92.             if (IS_ELEMENT(G[u],i) and i != u)
93.                 if (IS_ELEMENT(dom_set,i))
94.                     n_choice=0
95.                     For j=0 to n-1
96.                         if IS_ELEMENT(G[i],j)
97.                         num_dominated[j]=num_dominated[j]-1
98.                         End if
99.                     End For
100.                    For j=0 to n-1
101.                        if num_dominated[j] equals 0
102.                            n_choice=1;
103.                        End if
104.                    End For
105.                    if n_choice equals 1
106.                    For j=0 to n-1
107.                        if IS_ELEMENT(G[i],j)
108.                        num_dominated[j]=num_dominated[j]+1
109.                        n_choice <- 0
110.                        End if
111.                    End For
112.                    else
113.                        ADD_ELEMENT(deleted,i);
114.                        DEL_ELEMENT(dom_set,i);
115.                    End if
116.                End if
117.            End if
118.        End For
119.        size_del=set_size(n,deleted);
120.        if size_del equals 1
121.            For j=0 to n-1
```

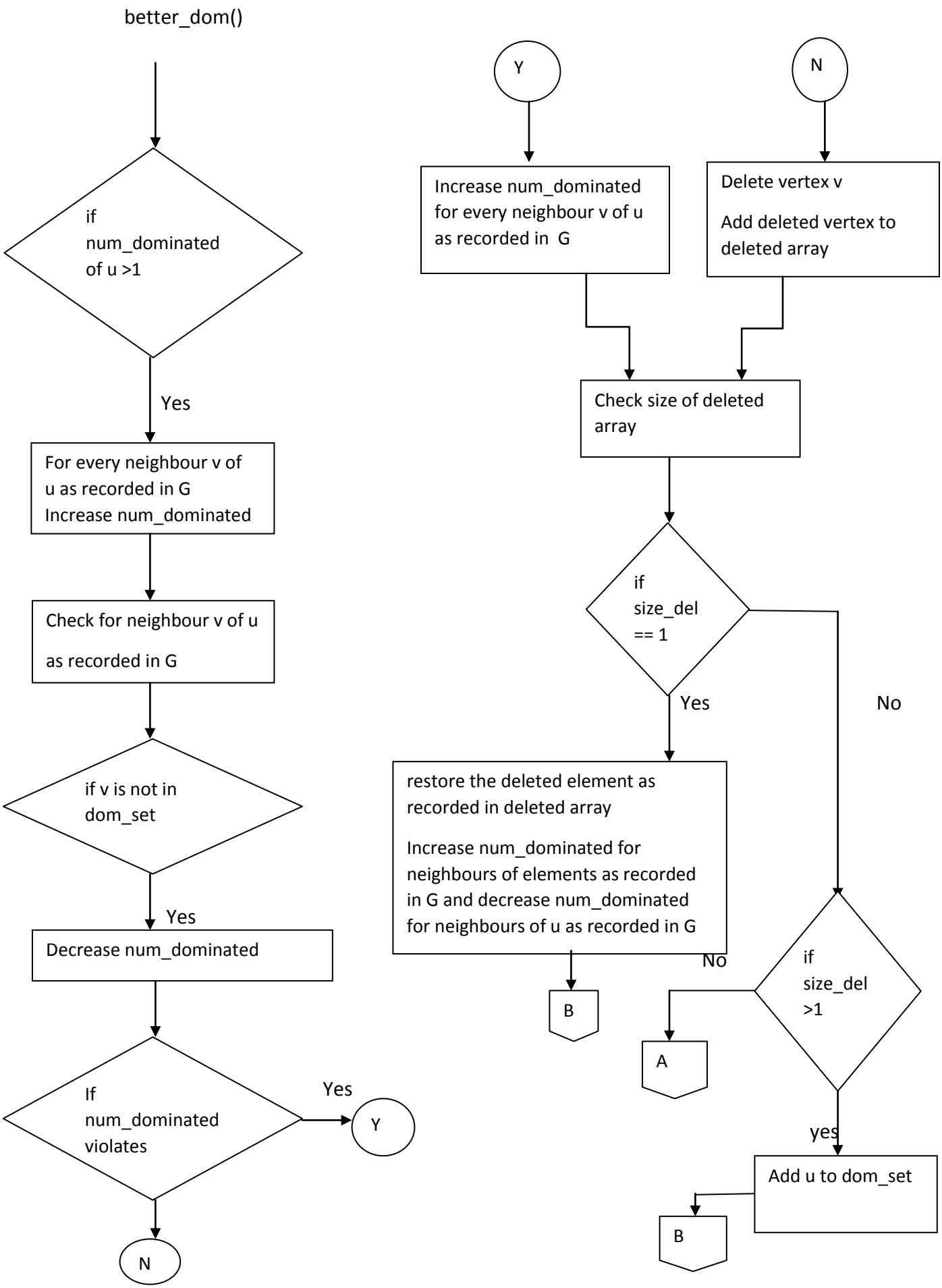
```
122.                                    if IS_ELEMENT(deleted,j)
123.                                            store=j;
124.                                            break;
125.                                    End if
126.                            End For
127.                            DEL_ELEMENT(deleted,store)
128.                            ADD_ELEMENT(dom_set,store)
129.                            For j=0 to n-1
130.                                    if IS_ELEMENT(G[store],j)
131.                                            num_dominated[j]=num_dominated[j]+1
132.                                    End if
133.                            End For
134.                            For j=0 to n-1
135.                                    if IS_ELEMENT(G[u],j)
136.                                            num_dominated[j]=num_dominated[j]-1
137.                                    End if
138.                            End For
139.
140.                    else if size_del > 1
141.                            ADD_ELEMENT(dom_set,u)
142.                    else if size_del equals 0
143.                            For j=0 to n-1
144.                                    if IS_ELEMENT(G[u],j)
145.                                            num_dominated[j]=num_dominated[j]-1
146.                                    End if
147.                            End For
148.                    End if
149.                    For j=0 to n-1
150.                            if num_dominated[j] equals 0
151.                                    n_domi=1
152.                            End if
153.                    End For
154.                    size_dom=set_size(n,dom_set)
155.                    if size_dom < min_size
156.                            if n_domi equals 0
157.                                    For j=0 to m-1
158.                                            min_dom_set[j]=dom_set[j]
159.                                    End For
160.                                    min_size=size_dom
161.                                    print_ver(0, min_size,n,min_dom_set)
162.                            End if
```

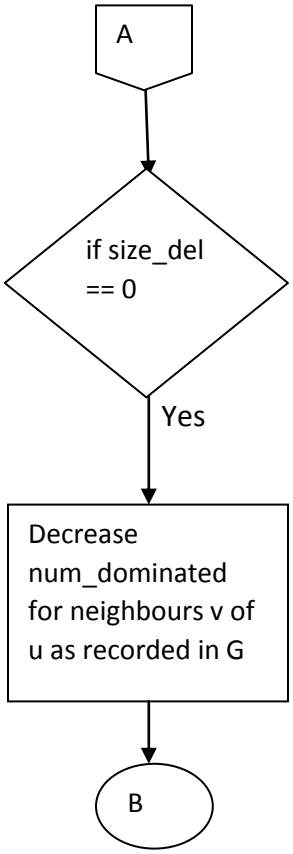
```
163.                End if
164.        End if

165.  End Procedure better_dom

166. Procedure INTEGER set_size(INTEGER n,INTEGER set[])
167.        Declare j,m,d
168.        Intialise d<-0
169.        m <- (n+31)/32
170.        For j=0 to m-1
171.                d <- d +  POP_COUNT(set[j])
172.        End For
173.        return d
174.   End Procedure set_size
175.  Procedure VOID print_set(INTEGER n, INTEGER set[])
176.        Declare i
177.        For i=0 to n-1
178.                if IS_ELEMENT(set, i)
179.                        Display i
180.                End if
181.        End For
182. End Procedure print_set
183.  Procedure INTEGER print_ver(INTEGER status,INTEGER min_size,INTEGER n,INTEGER
                        min_dom_set[])

184.        Display status,min_size
185.        print_set(n,min_dom_set)
186.  End Procedure print_ver
```

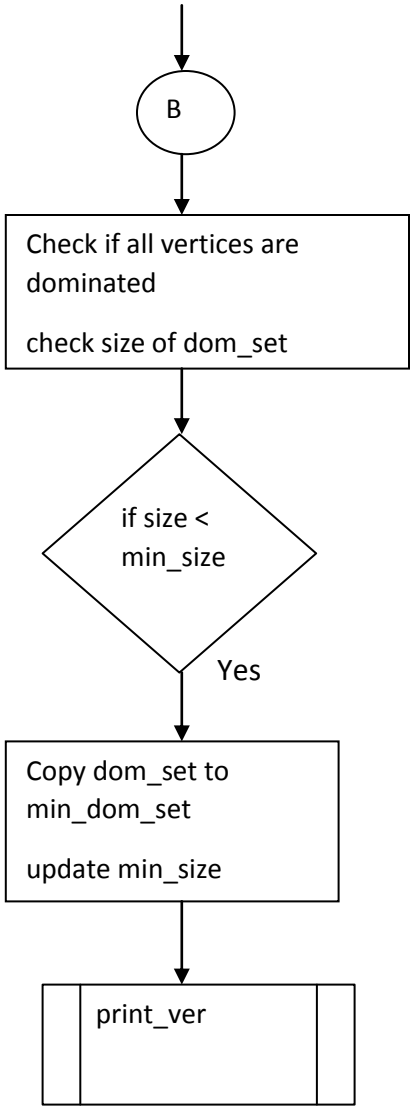
Design Decisions :




find_dom_set()
u=0
random_dom
_set
While u < n
yes
better_dom
u++
random_dom_set()
Initialisation
process
For i=0 to n-1
u = i
If u vertex
not in
dominated
array
Add u to dom_set
Increment size
Update dominated array for
every neighbour v of u as
recorded in graph G
Update num_dominated
copy dom_set to min_dom_set
update min_size
print_ver


better_dom()
if num_dominated of u >1
Yes
For every neighbour v of u as recorded in G Increase num_dominated
Check for neighbour v of u as recorded in G
if v is not in dom_set
Yes
Decrease num_dominated
If num_dominated violates
Yes
Y
N
Y
Increase num_dominated for every neighbour v of u as recorded in G
N
Delete vertex v
Add deleted vertex to deleted array
Check size of deleted array
if size_del == 1
Yes
No
restore the deleted element as recorded in deleted array
Increase num_dominated for neighbours of elements as recorded in G and decrease num_dominated for neighbours of u as recorded in G
B
No
if size_del >1
A
yes
Add u to dom_set
B


A
if size_del == 0
Yes
Decrease num_dominated for neighbours v of u as recorded in G
B


B
Check if all vertices are dominated
check size of dom_set
if size < min_size
Yes
Copy dom_set to min_dom_set
update min_size
print_ver