Computation of Pseudocoloring of Graphs through Python

V. Yegnanarayanan, S. B. Pravallika, Mokkala Mounika

***Abstract:** In the task of coloring the vertices of a simple graph G one come across innumerable number of challenges. There are various graph coloring parameters available in the literature. The concept of pseudo coloring is quite interesting. In this type of coloring, we can allot same color to the adjacent vertices. The maximum number of colors used in a pseudocoloring where for any two distinct colors, one can always find at least one edge between them is called pseudo achromatic number, $\psi(G)$ of G. In the paper we determine this coloring parameter for some classes of graphs through python code.*

***Keywords:** Graphs, pseudocoloring, pseudo achromatic number.*

I. INTRODUCTION

The graphs we dealt with here are all finite, simple and undirected. Given a graph G = (V,E) with vertex set V and edge set E , a function f : $V(G) \rightarrow \{1,\ldots,k\}$ is called a proper k-coloring if$\forall$ (u,v) ϵ E(G), $f(u)\neq f(v)$. If k is least , then we call k, the chromatic number $\chi(G)$ of G By a k-pseudocoloring of the vertices of G we mean a coloring using k-colors in which adjacent vertices can be allotted the same color.If we impose a further restriction that for any two distinct colors used in such a k-pseudocoloring there must be at least one edge in the graph with its end vertices colored with these two colors. The greatest number of colors used such a type of coloring is called pseudoachromatic number $\Psi(G)$ of G. It is trivial to note that $\chi(G)\leq\Psi(G)$.For the complete graph K_n the two parameters coincide. For a complete bipartite graph $K_{n,n}$ these two parameters differ. That is $\chi(K_{n.n}) = 2$ whereas $\Psi(K_{n,n}) = n+1$.

For a given graph G=(V,E) the middle graph M(G) possess $V(G)\cup E(G)$ as its vertex set and the edge set E(M(G)) ={(u, v) : either u, v $\in$E(G) and u is adjacent with v in G or u $\in$ V(G) and v $\in$ E(G) and v is incident with u in G. The total graph T(G) also possess $V(G)\cup E(G)$ as its vertex set and edge set E(T(G))={(u,v) : u, v $\in$ V(G) and u is adjacent to v in G or u, v $\in$E(G) and u, v are adjacent in G or u $\in$ V(G) and v$\in$ E(G) and v is incident with u in G}. The central graph C(G) of G is derived by subdividing every edge of G only once introducing an edge between all non-adjacent vertices of G

We now determine the $\Psi(C(S_n))$, $\Psi(M(S_n))$ and $\Psi(T(S_n))$ where $S_n = K_1\ \mathrm{v} nK_1$. Note that K_1 is a graph on one vertex and nK_1 is n copies of K_1. By join operation v we mean the only vertex of K_1 is adjacent with every vertex of nK_1.

II. RESULTS

Theorem 2.1$\Psi(C(S_n))$ = n+1.

Proof : Let $V(S_n) = \{u,u_1,u_2,\ldots,u_n\}$ and $E(S_n)=\{(u,u_i) : 1\leq i\leq n\}$.Note that each vertex and u_i of nK_1, has degree n. Now sub divide each edge uu_i with the vertex v_i for $1\leq i\leq n$. Then $V(C(S_n)) = \{u\}\cup\{u_1,u_2,...,u_n\}\cup\{v_1,v_2,\ldots,v_n\}$ and$E(C(S_n))=\{(u,v_i):i\leq i\leq n\}\cup\{(v_i,u_i):1\leq i\leq n\}\cup\{(u_1,u_2),\ldots,(u_{i,} u_n),(u_2,u_3),\ldots,(u_{2,}u_n),\ldots,(u_{n-1},u_n)\}$. So$|E(C(S_n))|$ = n+n+(n(n-1)/2) = n+n(n+1)/2) = $(n^2+3n)/2$. Observe that $\Psi(C(S_n))\leq$ n+1. It is easy to allot a(n+1)-pseudocoloring for the vertices of $C(S_n)$ as follows: Allot the color e_i for u_i,$1\leq i\leq n$; the color e_{n+1} for every v_i , $1\leq i\leq n$; the color e_1 to u. Hence $\Psi(C(S_n))$ = n+1.

Theorem2.2 $\Psi(M(S_n))$ = n+1.

Proof:: Let $V(S_n)=\{u;u_1,u_2,\ldots,u_n\}$ and $E(S_n)= \{(u, u_i) : 1\leq i\leq n\}$. By the definition of $M(S_n)$ we see that $V(M(S_n) = \{u\}\cup\{u_i : 1\leq i\leq n\}\cup\{v_i : 1\leq i\leq n\}$ where each v_i lies on the edge $(u,u_i) \in E(S_n)$and thereby subdividing each (u, u_i) for $1\leq i\leq n$. Observe that the subgraph induced by $\{u,v_1,v_2,\ldots,v_n\}$ namely, $<\{u,v_1,v_2,\ldots,v_n\}>\cong K_{n+1}$and hence we see that $|E(M(S_n))| = n(n+1)/2+n = (n^2+3n)/2$. So $\Psi(M(S_n)) \leq$ n+1 as n(n+1)/2 + n< (n+1)(n+2)/2. It is easy to allot a(n+1)-pseudo coloring for the vertices of $M(S_n)$ as follows: For each u_i, $2\leq i\leq n$ allot the color e_1 ; allot to color e_n to u_1 ; allot the color e_i, $1\leq i\leq n$ to each v_i; allot the color e_{n+1} to u. So $\Psi(M(S_n))$=n+1.

Theorem-2.3 $\Psi(T(S_n))$ = n+2

Proof: Let $V(S_n) = \{u,u_1,u_2,\ldots,u_n\}$ and $E(S_n)=\{(u, u_i) : 1\leq i\leq n\}$. By the definition of $T(S_n)$ we see that $V(T(S_n))=\{u\}\cup\{v_i:1\leq i\leq n\}\cup\{u_i:1\leq i\leq n\}$. Observe that $<\{v,\ldots,v_n\}>\tilde{=}K_{n+1}$. Moreover $|E(T(S_n)| = (n^2+5n)/2 <$ (n+2)(n+3)/2.So $\Psi(T(S_n))\leq$n+2. Also it is easy to allot a (n+2)-pseudocoloring to $T(S_n)$. Allot the color e_i, $1 \leq i\leq n$ to each v_i allot the colore_{n+1} to u ; $e_{n+2,}$ $1\leq i\leq n$ to each u_i So $\Psi(T(S_n))$ = n+2

III. ALGORITHM

In this section we give a pseudocode to determine the Ψ for each of $C(S_n)$, $M(S_n)$ and $T(S_n)$

STAR GRAPH

Algorithm StarGraph(n)

Pre : n is the last subscript of u and v vertices type

Post : Edges , Vertices and number of minimum colors required are printed

Revised Manuscript Received on November 02, 2019.

* Correspondence Author

V. Yegnanarayanan*, School of Arts, Science and Humanities, SASTRA Deemed to be University, Thanjavur-613401 TN,
Email: prof.yegna@gmail.com

S.B. Pravallika, School of Computing, SASTRA Deemed to be University, Thanjavur-613401 TN, India.
E-mail: pravallika0123456789@gmail.com

Mokkala Mounika, School of Computing, SASTRA Deemed to be University, Thanjavur-613401 TN, India.

//PRINTING VERTICES
1.print “Vertices=”
2. print v and set i to 1
3. loop(i less than n+1)
    1. print v i
    2. incrementi
4. end loop
5. seti to 1
6. loop(i less than n+1)
    1. print u i
    2. incrementi
7. end loop
8. print number of vertices which is (2*n+1)
//PRINTING EDGES
9. create empty list a
10. seti to 1
11. loop(i less than n+1)
    1. create tuple with (0,i)
    2. append tuple to a
    3. incrementi
12. end loop
13. seti to 1

14. loop(i less than n+1)
    1. create tuple with (i,i)
    2. append tuple to a
    3. incrementi
15. end loop
16. set i to 1 and j to (i+1)
17. loop(i less than n)
    1. loop(j less than n+1)
        1. create tuple with (i,j)
        2. append tuple to a
        3. increment j
    2. end loop
    3. incrementi
18. end loop
19. print length of a as the number of edges
20. seti to 0
21. loop(i less than 2*n)
    1. print (u,v) with a[i]
    2. incrementi
22. end loop
23. loop(i less than length of a)
    1. print (u,v) with a[i]
    2. incrementi
24. end loop
//ALLOCATION OF COLOUR TO EACH VERTEX
25. create two empty lists u and v
26. append 1 to v and o to u
27. set j to 2
28. seti to 1
29. loop(i less than n+1)
    1. append j to u
    2. increment both j and i
30. end loop
31. seti to 1
32. loop(i less than n+1)
    1. if(u[i] = = n+1)
        1. append 2 to v
    2. else
        1. append 1 to u[i]
    3. end if
    4. incrementi
33. end loop
//PRINTING OF VERTEX COLOURS
34. seti to 1
35. loop(i less than n+1)
    1. print u i = C u[i]
    2. incrementi
36. end loop
37. seti to 0
    1. print v i = C v[i]
2.incrementi
38.end loop
//CHECKING THE MINIMUM NUMBER OF COLOURS
CONDITION
39. set count to 0
40. set i to 1 and j to i+1
41. loop(i less than n+2)
    1. loop(j less than n+2)
        1. if(i is in v and j is in u)
            1. Make tuple1 with (index of (i) in v , index of (j) in u)
        2. end if
        3. if(i is in u and j is in v)
            1. Make tuple2 with (index of (i) in u , index of (j) in v)
        4. end if
        5. if(i is in v and j is in v)
            1. Make tuple3 with (index of (i) in v , index of (j) in v)
        6. end if
        7. if(tuple1 or tuple2 or tuple3 is in a)
            1. increment count
        8. increment j
    2. end loop
    3. incrementi
42. end loop
43. if(count equals to (n+1)*(n/2))
    1. print maximum number of colours as n+1
44. end if
End StarGraph

///

MIDDLE GRAPH
Algorithm MiddleGraph(n)
Pre : n is the last subscript of u and e vertices type
Post : Edges , Vertices and number of minimum colours required are printed
//PRINTING VERTICES

1.print “Vertices=”
2. print u and set i to 1
3. loop(i less than n+1)
    1. print u i
    2. incrementi
4. end loop
5. seti to 1
6. loop(i less than n+1)
    1. print e i
    2. incrementi
7. end loop
8. print number of vertices which is (2*n+1)

//PRINTING EDGES
9. create empty list a
10. seti to 1
11. loop(i less than n+1)
    1. create tuple with (0,i)
    2. append tuple to a
    3. incrementi
12. end loop
13. seti to 1
14. loop(i less than n+1)
    1. create tuple with (i,i)
    2. append tuple to a
    3. incrementi
15. end loop
16. set i to 1 and j to (i+1)
17. loop(i less than n)
    1. loop(j less than n+1)
        1. create tuple with (i,j)
        2. append tuple to a
        3. increment j
    2. end loop
    3. incrementi
18. end loop
19. print length of a as the number of edges
20. seti to 0
21. loop(i less than 2*n)
    1. print (u,e) with a[i]
    2. incrementi
22. end loop
23. loop(i less than length of a)
    1. print (e,e) with a[i]
    2. incrementi
24. end loop

//ALLOCATION OF COLOUR TO EACH VERTEX
25. create two empty lists u and e
26. append 0 to e
27. seti to 0
28. loop(i less than n+1)
    1. append i+1 to u
    2. increment i
29. end loop
30. seti to 1
31. loop(i less than n+1)
1.append i+1 to e
2.incrementi
32. end loop

//PRINTING OF VERTEX COLOURS
33. seti to 0
34. loop(i less than n+1)
    1. print u i = C u[i]
    2. incrementi
35. end loop
36. seti to 1
37.loop(I less than n+1)
    1. print E i = C v[i]
2.incrementi
38.end loop

//CHECKING THE MINIMUM NUMBER OF COLOURSCONDITION

39. set count to 0
40. set i to 1 and j to i+1

41. loop(i less than n+2)
    1. loop(j less than n+2)
        1. if(i is in u and j is in e)
            1. Make tuple1 with (index of (i) in u , index of (j) in e)
        2. end if
        3. if(i is in e and j is in u)
            1. Make tuple2 with (index of (i) in e , index of (j) in u)
        4. end if
5. if(i is in e and j is in e)
1. Make tuple3 with (index of (i) in v , index of (j) in v)
        6. end if
        7. if(tuple1 or tuple2 or tuple3 is in a)
            1. increment count
        8. increment j
    2. end loop
    3. incrementi
42. end loop
43. if(count equals to (n+1)*(n/2))
    1. print maximum number of colours as n+1
44. end if
End MiddleGraph

//

T-GRAPH

Algorithm TGraph(n)
Pre : n is the last subscript of u and e vertices type
Post : Edges , Vertices and number of minimum colours required are printed

//PRINTING VERTICES
1.print “Vertices=”
2. print u and set i to 1
3. loop(i less than n+1)
    1. print u i
    2. incrementi
4. end loop
5. seti to 1
6. loop(i less than n+1)
    1. print e i
    2. incrementi
7. end loop
8. print number of vertices which is (2*n+1)

//PRINTING EDGES
9. create empty list a
10. seti to 1
11. loop(i less than n+1)
    1. create tuple with (0,i)
    2. append tuple to a
    3. incrementi
12. end loop
13. seti to 1
14. loop(i less than n+1)
    1. create tuple with (i,i)
    2. append tuple to a
    3. incrementi
15. end loop
16. set i to 1 and j to (i+1)
17. loop(i less than n)
    1. loop(j less than n+1)
        1. create tuple with (i,j)
        2. append tuple to a
        3. increment j
    2. end loop
    3. incrementi

18. end loop
19.set b to length of a
20. loop(i less than n)
    1. loop(j less than n+1)
        1. create tuple with (i,j)
        2. append tuple to a
        3. increment j
    2. end loop
    3. incrementi
21. end loop
22. print length of a as the number of edges
23. seti to 0
24. loop(i less than 2*n)
    1. print (u,e) with a[i]
    2. incrementi
25. end loop
26. loop(i less than b+1)
    1. print (e,e) with a[i]
    2. incrementi
27. end loop
28. loop(i less than length of a)
    1. print (u,u) with a[i]
    2. incrementi
29. end loop

//ALLOCATION OF COLOUR TO EACH VERTEX

30. create two empty lists u and e
31. append o to e
32. seti to 0
33. loop(i less than n+1)
1.if(i is equal to 1)
    1. append 1 to u
    2. else
1.append n+2 to u
3.end if
4.incrementi
34. end loop
35. seti to 1
36. loop(i less than n+1)
1.append i+1 to e
2.incrementi
37. end loop

//PRINTING OF VERTEX COLOURS

38. seti to 0
39. loop(i less than n+1)
    1. print u i = C u[i]
    2. incrementi
40. end loop
41. seti to 1
    1. print E i = C v[i]
2.incrementi
42.end loop

//CHECKING THE MINIMUM NUMBER OF COLOURSCONDITION

43. set count to 0
44. set i to 1 and j to i+1

45. loop(i less than n+3)
    1. loop(j less than n+3)
        1. if(i is in u and j is in e)
            1. Make tuple1 with (index of (i) in u , index of (j) in e)
        2. end if
        3. if(i is in e and j is in u)
            1. Make tuple2 with (index of (i) in e , index of (j) in u)
        4. end if
        5. if(i is in e and j is in e)
            1. Make tuple3 with (index of (i) in e , index of (j) in e)
        6. end if
        7. if(i is in u and j is in u)
            1. Make tuple4 with (index of (i) in u , index of (j) in u)
        8. end if
        9. if(tuple1 or tuple2 or tuple3 or tuple4 is in a)
            1. increment count
        10. increment j
    2. end loop
    3. incrementi
46. end loop
47. if(count equals to ((n+2)*(n+1)/2))
    1. print maximum number of colours as n+2
48. end if
End TGraph

IV. PHYTHON CODE

In this section we give a Python code for each of the pseudo code of $\Psi(C(S_n))$, $\Psi(M(S_n))$ and $\Psi(T(S_n))$.

```
def check(i,v):
ifi in v:
return True
else:
return False
defcolorcheck(a,v,u,n):
count=0;
fori in range(1,n+2):
for j in range(i+1,n+2):
if(check(i,v)==True and check(j,u)==True):
        t1=(v.index(i),u.index(j))
if(check(i,u)==True and check(j,v)==True):
        t2=(u.index(i),v.index(j))
if(check(i,v)==True and check(j,v)==True):
        t3=(v.index(i),v.index(j))
if t1 in a or t2 in a or t3 in a:
count+=1
if(count==(n+1)*n/2):
print("Maximum no of colors required",n+1)
else:
print("Error occured")
def main():
print("Star graph")
   n=int(input("Enter n value\n"))
   a=[];a1=[];a2=[]
   a=[(0,i) for i in range(1,n+1)]
   a1=[(i,i) for i in range(1,n+1)]
   a2=[(i,j) for i in range(1,n) for j in range(i+1,n+1)]
   a=a+a1+a2
print("NO OF EDGES IN STAR GRAPH=",len(a));
fori in range(2*n):
print('(u,v):',a[i])
fori in range(2*n,len(a)):
print('(v,v):',a[i])
   v=[];u=[];v.append(1);u.append(0);j=2;
```

```
fori in range(1,n+1):
u.append(j)
      j+=1
fori in range(1,n+1):
if(not(u[i]==n+1)):
v.append(u[i]+1)
else:
v.append(2)
fori in range(1,n+1):
print("U",i,":C",u[i])
fori in range(0,n+1):
print("V",i,":C",v[i])
colorcheck(a,v,u,n);
print("Middle graph");a=[];a1=[];a2=[];
   a=[(0,i) for i in range(1,n+1)]
   a1=[(i,i) for i in range(1,n+1)]
   a2=[(i,j) for i in range(1,n) for j in range(i+1,n+1)]
   a=a+a1+a2;
print("NO OF EDGES IN TNE MIDDLE GRAPH=",len(a));
fori in range(2*n):
print('(u,e):',a[i])
fori in range(2*n,len(a)):
print('(e,e):',a[i])
   e=[];u=[];e1=[0];
   u=[i+1 for i in range(0,n+1)]
   e=e1+[i for i in range(1,n+1)]
fori in range(0,n+1):
print("U",i,":C",u[i])
fori in range(1,n+1):
print("E",i,":C",e[i])
count=0;
colorcheck(a,e,u,n);
   print("T graph");a=[];a1=[];a2=[];a3=[];
   a=[(0,i) for i in range(1,n+1)]
   a1=[(i,i) for i in range(1,n+1)]
   a2=[(i,j) for i in range(1,n) for j in range(i+1,n+1)]
   a=a+a1+a2
   b=len(a)
   a3=[(i,j) for i in range(1,n) for j in range(i+1,n+1)]
   a=a+a3;
print("NO OF EDGES IN THE T GRAPH=",len(a));
fori in range(2*n):
print('(u,e):',a[i])
fori in range(2*n,b+1):
print('(e,e):',a[i])
fori in range(b,len(a)):
print('(u,u):',a[i])
   e=[];u=[];e1=[0];
fori in range(0,n+1):
if(i==0):
u.append(1)
else:
u.append(n+2)
   e=e1+[i+1 for i in range(1,n+1)]
fori in range(0,n+1):
print("U",i,":C",u[i])
fori in range(1,n+1):
print("E",i,":C",e[i])
count=0;
fori in range(1,n+3):
for j in range(i+1,n+3):
if(check(i,u)==True and check(j,e)==True):
            t1=(u.index(i),e.index(j))
if(check(i,e)==True and check(j,u)==True):
            t2=(e.index(i),u.index(j))
if(check(i,e)==True and check(j,e)==True):
            t3=(e.index(i),e.index(j))
if(check(i,u)==True and check(j,u)==True):
            t4=(u.index(i),u.index(j))
if t1 in a or t2 in a or t3 in a or t4 in a:
count+=1
if(count==(n+2)*(n+1)/2):
print("No of colors in the t graph=",n+2)
else:
print("Error occured")
main()
```

V. CONCLUSION

In this paper we have determined the exact value of $\Psi(E(S_n))$, $\Psi(M(S_n))$ and $\Psi(T(S_n))$. We have also given the pseudo code and Python code to determine the exact values. We propose to determine the parameter Ψ for several other classes of graphs elsewhere.

REFERENCES

1. V. Yegnanarayan and B. Logeshwary, “On vertex coloring of graphs”, *International journal of Mathematical Analysis*, vol. 9, no. 17, 2015, pp. 857–868.
2. V. Yegnanarayan and P.K. Thirupurasundari, “On some graph operations and related applications”, *Electronic notes in Discrete Mathematics*, vol. 33, 2009, pp. 123–130.

AUTHORS PROFILE

V. Yegnanarayanan obtained his Full Time PhD in Mathematics from Annamalai University in Nov 1996. He has 31 years of total experience in which 16 years as Professor & Dean/HOD. He has also worked as a Visiting Scientist on lien in TIFR and IMSC. He has authored 164 research papers and guided 6 students to Ph.Ddegree . He has delivered a number of invited talks, organized funded conferences, FDP etc, did a lot of review work for MR, zbMATH and reputed journals, completed research projects funded by NBHM-DAE, GOI, won Sentinel of Science Award from Publons. TN State (Periyar) University has recognized his qualifications and experience for the post of Principal in 2009.

S.B. Pravallika is doing her B.Tech in Computer Science and Engineering@ School of Computing, SASTRA Deemed to be University, Thanjavur, Tamilnadu. My current research interests are cyber security, ethical hacking and big data analytics.

MokkalaMounikais doing her B.Tech in Computer Science and Engineering@ School of Computing, SASTRA Deemed to be University, Thanjavur, Tamilnadu. My current research interests are web development, app development, cyber security and machine learning.