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, Ψ(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 γ : V(G) → {1,...,k} is called a proper k-coloring ifV (u,v) ∈ E(G), f(u)≠f(v). If k is least , then we call k, the chromatic number χ(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 Ψ(G) of G. It is trivial to note that χ(G)≥Ψ(G). For the complete graph Kn, the two parameters coincide. For a complete bipartite graph Kn,m these two parameters differ. That is Ψ(Kn,m)=2 whereas Ψ(Kn,n)=n+1.

For a given graph G=(V,E) the middle graph M(G) possess V(G)∪E(G) as its vertex set and the edge set E(M(G))={(u,v) : u, v ∈ V(G) or u, v ∈ E(G) or u is incident with v in G}. The total graph T(G) also posses V(G)∪E(G) as its vertex set and the edge set E(T(G))={(u,v) : either u, v ∈ V(G) and v is incident with u in G or u, v ∈ E(G) and u is incident with v in G}.

We now determine the Ψ(C(Sn)), Ψ(M(Sn)) and Ψ(T(Sn)) where Sn = K1 vnK1. Note that Kn is a graph on one vertex and nK1 is n copies of K1. By join operation we mean the only vertex of K1 is adjacent with every vertex of nK1.

II. RESULTS

Theorem 2.1 Ψ(C(Sn))=n+1.

Proof : Let V(Sn)={u1,u2,u3,..,un} and E(Sn)={(ui,ui+1) : 1≤i≤n}. Note that each vertex and ui of nK1, has degree n. Now sub divide each edge uu, with the vertex v1 for 1≤i≤n. Then V(C(Sn)) = {u|U{ui,ui+1}|U{v1,v2,..,vn} andE(C(Sn))={{(ui,ui+1): 1≤i≤n}|U{(ui,ui+1)|U{ui,ui+1}:1≤i≤n}∪{(ui,ui+1)|U{ui,ui+1}:1≤i≤n}}. So |E(C(Sn))| = \(\frac{n(n+1)}{2}\) = n(n+1)/2. Observe that Ψ(C(Sn))≤n+1. It is easy to allot \(a(n+1)\)-pseudocoloring for the vertices of C(Sn) as follows: Color the vertex ei for \(u_i, 1≤i≤n\; \text{the color}\; e_{n+1}\; \text{for every}\; v_i, 1≤i≤n; \text{the color} \; e_{2i} \; \text{to} \; u_i, \text{Hence} \; Ψ(C(S_n))=n+1.

Theorem 2.2 Ψ(M(Sn))=n+1.

Proof: Let V(Sn)={u1,u2,u3,..,un} and E(Sn)={(ui,ui+1) : 1≤i≤n}. By the definition of M(Sn) we see that V(M(Sn)) = {u|U{ui,ui+1}|U{v1,v2,..,vn} andE(M(Sn))={{(ui,ui+1): 1≤i≤n}|U{(ui,ui+1)|U{ui,ui+1}:1≤i≤n}∪{(ui,ui+1):1≤i≤n}}. So |E(M(Sn))| = \(\frac{n(n+1)}{2}\) + n(n+1)/2 = n(n+3)/2. Observe that Ψ(M(Sn))≤n+1 as \(\frac{n(n+1)}{2}+n(n+1)/2\) ≤ n(n+3)/2. It is easy to allot \(a(n+1)\)-pseudo coloring for the vertices of M(Sn) as follows: For each wi, 2≤i≤n allot the color ei to ui , allot the color ei+1 to ui+1 for each ei , 1≤i≤n to each vi; allot the color e_{n+1} to u_i. So Ψ(M(Sn))=n+1.

Theorem 2.3 Ψ(T(Sn))=n+2

Proof: Let V(T(Sn))={u1,u2,u3,..,un} and E(T(Sn))={(ui,ui+1) : 1≤i≤n}. By the definition of T(Sn) we see that V(T(Sn)) = {u|U{ui,ui+1}|U{v1,v2,..,vn} andE(T(Sn))={{(ui,ui+1): 1≤i≤n}|U{(ui,ui+1)|U{ui,ui+1}:1≤i≤n}}. Observe that T(Sn) is \(n(n+2)/2\) < \(n(n+3)/2\). So Ψ(T(Sn))=n+2. Also it is easy to allot a \(n+2\)-pseudocoloring to T(Sn). Allot the color ei, 1≤i≤n to each vi, the color e_{n+1} to u_i, allot the color e_{n+2}, 1≤i≤n to each u_i. So Ψ(T(S_n))=n+2.

III. ALGORITHM

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

STARC 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
Computation of Pseudocoloring of Graphs through Python

//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. increment i
4. end loop
5. set i to 1
6. loop(i less than n+1)
   1. print u i
   2. increment i
7. end loop
8. print number of vertices which is (2*n+1)

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

//PRINTING OF VERTEX COLOURS
34. set i to 1
35. loop(i less than n+1)
   1. print u i = C u[i]
   2. increment i
36. end loop
37. set i to 0
   1. print v i = C v[i]
   2. increment i
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
      2. if(i is in u and j is in v)
         1. Make tuple2 with (index of (i) in u ,
            index of (j) in v)
         2. end if
      3. if(tuple1 or tuple2 is in a)
         1. increment count
      4. end if
   2. end loop
   3. increment i
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 v and set i to 1
3. loop(i less than n+1)
   1. print u i
   2. increment i
4. end loop
5. set i to 1
6. loop(i less than n+1)
   1. print e i
   2. increment i
7. end loop
8. print number of vertices which is (2*n+1)
//PRINTING EDGES
9. create empty list a
10. set i to 1
11. loop (i less than n+1)
  1. create tuple with (0, i)
  2. append tuple to a
  3. increment i
12. end loop
13. set i to 1
14. loop (i less than n+1)
  1. create tuple with (i, i)
  2. append tuple to a
  3. increment i
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
  end loop
  2. end loop
  3. increment i
18. end loop
19. print length of a as the number of edges
20. set i to 0
21. loop (i less than 2*n)
  1. print (u, e) with a[i]
  2. increment i
22. end loop
23. loop (i less than length of a)
  1. print (e, e) with a[i]
  2. increment i
24. end loop

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

//PRINTING VERTEX COLOURS
33. set i to 0
34. loop (i less than n+1)
  1. print u[i] = C
  2. increment i
35. end loop
36. set i to 1
37. loop (i less than n+1)
  1. print e[i] = C
  2. increment i
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 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)
        2. end if
    end loop
  end loop
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. increment i
4. end loop
5. set i to 1
6. loop (i less than n+1)
  1. print e[i]
  2. increment i
7. end loop
8. print number of vertices which is (2*n+1)

//PRINTING EDGES
9. create empty list a
10. set i to 1
11. loop (i less than n+1)
  1. create tuple with (0, i)
  2. append tuple to a
  3. increment i
12. end loop
13. set i to 1
14. loop (i less than n+1)
  1. create tuple with (i, i)
  2. append tuple to a
  3. increment i
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
  end loop
  2. end loop
  3. increment i
Computation of Pseudocoloring of Graphs through Python

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

//ALLOCATION OF COLOUR TO EACH VERTEX
30. create two empty lists u and e
31. append 0 to e
32. set i 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. increment i
34. end loop
35. set i to 1
36. loop(i less than n+1)
   1. append i+1 to e
   2. increment i
37. end loop

//PRINTING OF VERTEX COLOURS
38. set i to 0
39. loop(i less than n+1)
   1. print u[i] = C v[i]
   2. increment i
40. end loop
41. set i to 1
42. end loop

//CHECKING THE MINIMUM NUMBER OF COLOURS
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. end loop
   2. end loop
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. PYTHON 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 \( \Psi(T(S_n)) \).

```python
def check(i,v):
    if i in v:
        return True
    else:
        return False

def colorcheck(a,v,u,n):
count=0;
for i 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=[(i,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));
for i in range(2*n):
    print('(u,v):',a[i])
for i in range(2*n,len(a)):
    print('(v,v):',a[i])
v=[];u=[];
v.append(1);u.append(0);j=2;
```
for i in range(1,n+1):
    u.append(i)
for i in range(1,n+1):
    j+=1
for i in range(1,n+1):
    if(not(u[i]==n+1)):
        v.append(u[i]+1)
else:
    v.append(2)
for i in range(1,n+1):
    print("U",i,":C",u[i])
for i 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 THE MIDDLE GRAPH=",len(a));
for i in range(2*n):
    print('(u,e):',a[i])
for i 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)]
e1+=[i for i in range(1,n+1)]
    for i in range(0,n+1):
        print("U",i,":C",u[i])
        for i 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+=a1+a2
b=len(a)
a3=[(i,j) for i in range(1,n) for j in range(i+1,n+1)]
a+=a3;
print("NO OF EDGES IN THE T GRAPH=",len(a));
for i in range(2*n):
    print('(u,e):',a[i])
for i in range(2*n,b+1):
    print('(e,e):',a[i])
for i in range(b,len(a)):
    print('(u,u):',a[i])
e1=();u1=[];e1=[];e1=[]
    for i in range(0,n+1):
        if(i==0):
            u.append(1)
        else:
            u1.append((i,j) for i in range(1,n+1)]
    e1+=[i+1 for i in range(0,n+1)]
    print("U",i,":C",u[i])
    print("E",i,":C",e[i])
count=0;
for i in range(1,n+3):
    if(check(i,u)==True and check(j,e)==True):
        t1=(e.index(i),u.index(j))
    else:
        print("Error occurred")
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 $\Psi$ for several other classes of graphs elsewhere.

REFERENCES


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.D degree. 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.

Mokkala Mounika 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 web development, app development, cyber security and machine learning.