**<http://cse.iitkgp.ac.in/~abhij/course/theory/Algo1/Autumn13/lab/A4.html>**

**Greedy Graph Coloring**

Let *G* = (*V*,*E*) be a (simple undirected) graph. Two types of coloring problems are of importance in graph theory and the study of algorithms.

**Vertex coloring:** We assign colors to the vertices of *G* in such a manner that no two adjacent vertices receive the same color. The objective is to accomplish this task using as few colors as possible.

**Edge coloring:** We assign colors to the edges of *G* in such a manner that no two edges sharing an endpoint receive the same color. Again, the objective is to accomplish this task using as few colors as possible.

Both these problems appear to be difficult. Under certain complexity-theoretic assumptions, no polynomial-time algorithms exist to solve these problems. However, given the importance of these problems, we need to solve them. Given that polynomial-time algorithms fail to guarantee optimal solutions, we have to keep ourselves happy with suboptimal solutions. But then, how bad can these suboptimal solutions be? Can they be guaranteed to be within a constant multiple of the optimal solutions? The answer is again *no*. Indeed, if *G* contains *n* vertices, no polynomial-time algorithm is expected to exist whose solutions are poorer than the optimal solutions by a factor of O(*n*1−ε) for any ε > 0. Nevertheless, for random graphs, these suboptimal solutions are often not as bad as these pessimistic estimates suggest.

In this assignment, you implement a greedy algorithm to solve the vertex-coloring problem. Later, you solve the edge-coloring problem by reducing the problem to the vertex-coloring problem. The assignment has four important parts.

**Generate a random graph**

The user supplies the number *n* of vertices and a probability *p*. Write a function that generates a random graph with *n* vertices. For each pair (*u*,*v*) of distinct vertices, include the edge (*u*,*v*) in the edge set *E* with probability *p*. This means that for each vertex pair, you generate a random floating-point number in the interval [0,1]. If that random number is ≤ *p*, add the edge joining the vertices, else do not include the edge. Use the *adjacency-list representation* of the graph. Print the graph (vertices and the neighbors of each vertex).

**Constructing maximal independent sets in a graph**

All vertices that receive the same color must be non-adjacent to one another. Such a set of vertices is called an *independent set*. Write a function that, given a graph *G*, generates a maximal independent set in the graph. A maximal independent set is one with the property that the addition of any remaining vertex to the set destroys the "independent-set" property of the set. Ideally, we like to locate a maximum independent set in *G* (that is, an independent set of size as large as possible), but this is again a difficult computational problem. What you instead implement is the following greedy algorithm. This algorithm always outputs maximal independent sets (but they need not be maximum).

|  |
| --- |
| Initialize *S* to the empty set. Repeat the following three steps so long as the graph *G* is not empty :       Locate a vertex *u* of (remaining) minimum degree in *G*.       Add *u* to *S*.       Delete *u* and all its neighbors from *G*. Return *S*. |

By the deletion of a vertex *v* from a graph *G* = (*V*,*E*), we imply the deletion of *v* from the vertex set *V* and the deletion, from the edge set *E*, of all the edges in which *v* is an endpoint. The loop in the above algorithm terminates when all vertices are deleted from *G*. In the process, *S* accumulates some vertices. By construction, these vertices must form an independent set. Moreover, it must be maximal. The greedy choice of *u* (one of minimum degree) is justified because we desire to make *S* large.

**Greedy vertex coloring**

Write a function to compute a coloring of the vertices of a graph *G*. Implement the following greedy algorithm that invokes the function for constructing maximal independent sets multiple times.

|  |
| --- |
| Set *c* = 0. Repeat the following four lines until *G* is empty :       Compute a maximal independent set *S* in *G*.       Increment *c* (use the next color).       Color all the vertices in *S* with the color *c*.       Delete all the vertices of *S* from *G*. Return this coloring and the number of colors used. |

Here, the colors are numbered as 1,2,3,…. In this and the previous parts, do not destroy the original graph (while deleting vertices), because you are going to need it in the next part. You may alter a copy of *G*. Alternatively, you can effect deletion by using special markers against vertices.

**Greedy edge coloring**

In order to color the edges of *G* = (*V*,*E*), you contruct the *line graph* *L*(*G*) = (*V* ′,*E* ′) of *G* defined as follows. The vertex set *V* ′ of *L*(*G*) is the same as the edge set *E* of *G*. Two vertices in *L*(*G*) (that is, two edges in *E*) are adjacent (in *E* ′) if and only if the edges share a common endpoint. Evidently, an edge coloring of *G* is the same as a vertex coloring of *L*(*G*). Write a function that, upon the input of *G*, returns the line graph *L*(*G*). Run the above greedy vertex-coloring algorithm on the returned graph. Report the coloring produced and the number of colors used. Notice that both *G* and *L*(*G*) are undirected graphs and can be stored in the same format. They differ only by the formats of the vertex names. For *G*, the vertices can be named as 0,1,2,…,*n*−1, whereas for *L*(*G*), the vertices are named by a pair of such integers.

**Sample Output**

n = 20

p = 0.1

+++ The graph before vertex coloring

0 [0] : 3 [0] 19 [0]

1 [0] : 12 [0]

2 [0] : 9 [0] 12 [0] 18 [0]

3 [0] : 0 [0] 9 [0] 11 [0] 16 [0] 19 [0]

4 [0] : 5 [0]

5 [0] : 4 [0] 12 [0] 13 [0]

6 [0] :

7 [0] : 11 [0] 15 [0]

8 [0] :

9 [0] : 2 [0] 3 [0]

10 [0] : 18 [0]

11 [0] : 3 [0] 7 [0] 16 [0]

12 [0] : 1 [0] 2 [0] 5 [0] 17 [0]

13 [0] : 5 [0] 16 [0]

14 [0] : 19 [0]

15 [0] : 7 [0] 19 [0]

16 [0] : 3 [0] 11 [0] 13 [0]

17 [0] : 12 [0] 19 [0]

18 [0] : 2 [0] 10 [0]

19 [0] : 0 [0] 3 [0] 14 [0] 15 [0] 17 [0]

Vertex coloring done

Number of colors used = 3

+++ The graph after vertex coloring

0 [1] : 3 [3] 19 [2]

1 [1] : 12 [3]

2 [1] : 9 [2] 12 [3] 18 [2]

3 [3] : 0 [1] 9 [2] 11 [1] 16 [2] 19 [2]

4 [1] : 5 [2]

5 [2] : 4 [1] 12 [3] 13 [1]

6 [1] :

7 [2] : 11 [1] 15 [1]

8 [1] :

9 [2] : 2 [1] 3 [3]

10 [1] : 18 [2]

11 [1] : 3 [3] 7 [2] 16 [2]

12 [3] : 1 [1] 2 [1] 5 [2] 17 [1]

13 [1] : 5 [2] 16 [2]

14 [1] : 19 [2]

15 [1] : 7 [2] 19 [2]

16 [2] : 3 [3] 11 [1] 13 [1]

17 [1] : 12 [3] 19 [2]

18 [2] : 2 [1] 10 [1]

19 [2] : 0 [1] 3 [3] 14 [1] 15 [1] 17 [1]

+++ The line graph before coloring

(0,3) [0] : (0,19) [0] (3,9) [0] (3,11) [0] (3,16) [0] (3,19) [0]

(0,19) [0] : (0,3) [0] (3,19) [0] (14,19) [0] (15,19) [0] (17,19) [0]

(1,12) [0] : (2,12) [0] (5,12) [0] (12,17) [0]

(2,9) [0] : (2,12) [0] (2,18) [0] (3,9) [0]

(2,12) [0] : (1,12) [0] (2,9) [0] (2,18) [0] (5,12) [0] (12,17) [0]

(2,18) [0] : (2,9) [0] (2,12) [0] (10,18) [0]

(3,9) [0] : (0,3) [0] (2,9) [0] (3,11) [0] (3,16) [0] (3,19) [0]

(3,11) [0] : (0,3) [0] (3,9) [0] (3,16) [0] (3,19) [0] (7,11) [0] (11,16) [0]

(3,16) [0] : (0,3) [0] (3,9) [0] (3,11) [0] (3,19) [0] (11,16) [0] (13,16) [0]

(3,19) [0] : (0,3) [0] (0,19) [0] (3,9) [0] (3,11) [0] (3,16) [0] (14,19) [0] (15,19) [0] (17,19) [0]

(4,5) [0] : (5,12) [0] (5,13) [0]

(5,12) [0] : (1,12) [0] (2,12) [0] (4,5) [0] (5,13) [0] (12,17) [0]

(5,13) [0] : (4,5) [0] (5,12) [0] (13,16) [0]

(7,11) [0] : (3,11) [0] (7,15) [0] (11,16) [0]

(7,15) [0] : (7,11) [0] (15,19) [0]

(10,18) [0] : (2,18) [0]

(11,16) [0] : (3,11) [0] (3,16) [0] (7,11) [0] (13,16) [0]

(12,17) [0] : (1,12) [0] (2,12) [0] (5,12) [0] (17,19) [0]

(13,16) [0] : (3,16) [0] (5,13) [0] (11,16) [0]

(14,19) [0] : (0,19) [0] (3,19) [0] (15,19) [0] (17,19) [0]

(15,19) [0] : (0,19) [0] (3,19) [0] (7,15) [0] (14,19) [0] (17,19) [0]

(17,19) [0] : (0,19) [0] (3,19) [0] (12,17) [0] (14,19) [0] (15,19) [0]

Edge coloring done

Number of colors used = 5

+++ The graph after edge coloring

(0,3) [2] : (0,19) [1] (3,9) [3] (3,11) [1] (3,16) [4] (3,19) [5]

(0,19) [1] : (0,3) [2] (3,19) [5] (14,19) [2] (15,19) [3] (17,19) [4]

(1,12) [1] : (2,12) [3] (5,12) [4] (12,17) [2]

(2,9) [1] : (2,12) [3] (2,18) [2] (3,9) [3]

(2,12) [3] : (1,12) [1] (2,9) [1] (2,18) [2] (5,12) [4] (12,17) [2]

(2,18) [2] : (2,9) [1] (2,12) [3] (10,18) [1]

(3,9) [3] : (0,3) [2] (2,9) [1] (3,11) [1] (3,16) [4] (3,19) [5]

(3,11) [1] : (0,3) [2] (3,9) [3] (3,16) [4] (3,19) [5] (7,11) [2] (11,16) [3]

(3,16) [4] : (0,3) [2] (3,9) [3] (3,11) [1] (3,19) [5] (11,16) [3] (13,16) [1]

(3,19) [5] : (0,3) [2] (0,19) [1] (3,9) [3] (3,11) [1] (3,16) [4] (14,19) [2] (15,19) [3] (17,19) [4]

(4,5) [1] : (5,12) [4] (5,13) [2]

(5,12) [4] : (1,12) [1] (2,12) [3] (4,5) [1] (5,13) [2] (12,17) [2]

(5,13) [2] : (4,5) [1] (5,12) [4] (13,16) [1]

(7,11) [2] : (3,11) [1] (7,15) [1] (11,16) [3]

(7,15) [1] : (7,11) [2] (15,19) [3]

(10,18) [1] : (2,18) [2]

(11,16) [3] : (3,11) [1] (3,16) [4] (7,11) [2] (13,16) [1]

(12,17) [2] : (1,12) [1] (2,12) [3] (5,12) [4] (17,19) [4]

(13,16) [1] : (3,16) [4] (5,13) [2] (11,16) [3]

(14,19) [2] : (0,19) [1] (3,19) [5] (15,19) [3] (17,19) [4]

(15,19) [3] : (0,19) [1] (3,19) [5] (7,15) [1] (14,19) [2] (17,19) [4]

(17,19) [4] : (0,19) [1] (3,19) [5] (12,17) [2] (14,19) [2] (15,19) [3]