MATEMATICA DISCRETA II-2007, Práctico 2

I): Probar que si $\Delta(G) \geq 3$ y $\chi(G) = \Delta(G) + 1$, entonces G contiene un subgrafo isomorfo a $K_{\Delta(G)+1}$.

II): Se tiene un rectangulo en el plano al cual se le ha realizado una triangulación : una subdivisión de su superficie en triangulos. Se deben colorear los triangulos que constituyen la triangulación de forma tal que si dos triangulos tienen un lado en común entonces se coloreen con colores distintos. Se crea un grafo de la siguiente forma: hay un vértice por cada triangulo, y dos vertices son vecinos si los triángulos correspondientes tienen un lado en comun.

Se desea un programa que coloree el grafo con la menor cantidad de colores posibles. (el grafo le sera dado como input, no tiene que construirlo)

Dar un programa en pseudocodigo que haga esto, que sea correcto (es decir, que devuelva un coloreo con la menor cantidad de colores posibles) y de complejidad razonable. (no solo debe dar el algoritmo, sino probar que es correcto y calcular la complejidad).

III): Sea G un grafo tal que $\chi(G) = k$, pero ningún subgrafo de G tenga $\chi = k$. (un grafo asi se llama k-critico). Probar que $\delta \geq k - 1$.

IV): Sabemos que un grafo sin ciclos impares tiene $\chi(G) \leq 2$. Probar que un grafo G con a lo sumo dos ciclos impares tiene $\chi(G) \leq 3$.

V): Correr Greedy en los grafos del ejercicio 3) del practico 1, ordenando los vertices al azar. Compare el resultdo que obtuvo con el numero cromatico del grafo.

VI): Correr Greedy en el grafo de Petersen (ej. 4 del pr.1) en el orden afcegdbhij y luego en el orden alfabetico.

VII): Considere el siguiente algoritmo: Dado un orden de los vertices, se le asigna el color 1 al primer vertice, y luego se recorre la lista de los vertices, asignando el color 1 a todo vertice que no sea vecino con algun vertice al cual ya se le asigno el color 1. Una vez que se recorrio toda la lista, se la recorre nuevamente, pero ahora con el color 2: se le asigna el color 2 al primer vertice no coloreado de la lista, y luego se le va asignando el color 2 a todos los vertices no coloreados que no sean vecinos con ningun vertice ya colroeado con el color 2. Luego de esto se sigue con 3, etc, hasta colorear todos los vertices.

Rehacer el ejercicio VI) con este algoritmo.

VIII): Basado en el ejercicio anterior, formule una conjetura y pruebela.

IX): Un algoritmo muy usado es el algoritmo de Welsh-Powell, que es simplemente Greedy, con el orden dado por el ejercicio (11) del practico 1, es decir, en el orden no creciente de los grados.

Repetir el ejercicio V) usando el algoritmo de Welsh-Powell.

X): El algoritmo RLF (Recursive largest first) de Leighton (1979) es bastante rapido y en la practica da resultados bastante buenos. Es como sigue:

```
color:= 1;
while (existan vertices sin colorear)
    Hacer una lista L de los vertices sin colorear;
    while (L no vacio)
       Hallar v en L con la mayor cantidad de vecinos sin colorear;
       color[v] := color;
       Borrar a  v y a sus vecinos de L;
    endwhile;
    color++;
endwhile;
endprogram;
```

Repetir el ejercicio V) con RLF.

XI): Otro algoritmo muy usado es DSATUR de Brelaz (1979). Dado un coloreo parcial de un grafo, el grado de saturación de un vertice x es igual al numero de colores de los vecinos coloreados de x. DSATUR(degree of saturation) es basicamente Greedy, pero el orden de los vertices no es creado de antemando sino dinamicamente: el primer vertice es el vertice de mayor grado. Se calculan los grados de saturacion, se elije algun vertice con el mayor grado de saturación y se colorea ese vertice. Se recalculan los grados de saturación, y se sigue de esta forma. Repetir el ejercicio V) con DSATUR.

XII): El siguiente algoritmo, basado en otro algoritmo similar de Petford-Welsh, es aleatorio:

```
Correr Greedy;
while (No se produzca la condicion de parar)
   if (coloreo es propio)  then recolorear vertices del color mas grande;
                           else elejir vertice ‘‘malo’’ y recolorearlo;
   endif;
endwhile;
endprogram;
```

Aqui un vertice “malo” es algun vertice que tiene algun vecino del mismo color.

El recoloreo de los vertices es al azar, no necesariamente se colorea con un color que no produsca conflictos. Por ejemplo, se elije algun color al azar, y se decide si se colorea al vertice con ese color o no de acuerdo a una distribucione de probabilidad que depende de cuantos vecinos ese vertice tiene de ese color. La distribución es tal que si no tiene vecinos el color se asigna con probabilidad 1. Por ejemplo, se usa la distribución $e^{N_i/T}$ donde N_i es el numero de vecinos del color i, y T es un parametro que hay que elejir cuidadosamente para que el algoritmo funcione bien. La condicion de parar es una condicion del tipo “si ya coloreamos con menos de tantos colores o ya paso tanto tiempo”

Para hacer este ejercicio necesitara una calculadora y alguna forma de seleccionar numeros al azar (algunas calculadoras lo tienen incorporado). Repetir el ejercicio V) usando este algoritmo, con $T = 0,3$, y alguna condicion razonable de parar. Alternativamente al uso de calculadora, programe este algoritmo en algun lenguaje de sus elección y corralo.