Proyecto de Matemática Discreta II-2022 3ra parte

Contents

1 Main

Deberán entregar un solo archivo .c con un main (y funciones auxiliares que necesiten para el main, pero para esta etapa, un solo .c) el cual haga lo siguiente:

1. Para empezar deberá requerir 3 parametros enteros α, β, ρ. Esto debe ser hecho de forma tal que si compilamos a un ejecutable ejec podamos hacer

   ./ejec α, β, ρ <enter>

   y luego cargar a mano el grafo, o bien poder usar el operador de redirección y hacer pej

   ./ejec α, β, ρ <q10.txt

   donde q10.txt sea algun archivo con un grafo.

2. Luego de cargar el grafo, imprime el número de vértices, el número de lados y Delta.

3. Luego de eso, corre Bipartito() para ver si el grafo es bipartito o no. Si lo es, declara eso y en caso que la cantidad de vértices sea menor a 101, imprime las dos partes del grafo y termina la ejecución. Si no lo es, se declara eso y se continua con el resto.

4. Para el resto de la corrida se usarán procesos pseudoaleatorios, pej en AleatorizarVertices() uno de los parametros es R que funciona como semilla de aleatoriedad. La fuente "inicial"de aleatoriedad a partir de la cual todas las demas fuentes se vayan obteniendo será el parámetro ρ.

5. Construye $\alpha + 2$ ordenamientos iniciales como se explica a continuación:

   (a) Un orden se obtiene simplemente declarando Orden[i]=i, es decir, es el orden natural.

   (b) Otro orden será el orden "Welsh-Powell"que consiste en ordenar los vertices de mayor a menor grado. Este orden se obtiene usando OrdenFromKey a partir de definir key[i]=Grado(i,G);

   (c) Los otros α ordenes se obtienen usando OrdenFromKey a partir de definir key usando AleatorizarKeys con distintos Rs, generados a partir de ρ.

6. Para cada uno de esos $\alpha + 2$ ordenamientos iniciales debe correr Greedy e imprimir cuantos colores se obtienen.

   Luego de esto, a partir de ese coloreo inicial obtenido, debe hacer $3 \times \beta$ Greedys con ordenamientos que se obtendran a partir del coloreo anterior. Como OrdenFromKey a partir de las key que se definen en esta etapa va a crear un orden que cumple las hipotesis del VIT, la cantidad de colores no puede ser mayor que la obtenida anteriormente, en estos $3 \times \beta$ ordenamientos, asi que el main debe chequear esto y detener la corrida si no se cumple. Especificamente, debe hacer β veces los tres siguientes recoloreos:

   (a) Crear un nuevo orden a partir del coloreo anterior, en donde se ordenan los indices de forma tal que los indices con el mayor color queden primero, luego los del segundo mayor color, etc, con el color 0 al final. Esto se hace usando OrdenFromKey pasandole el coloreo obtenido anteriormente como key.

   (b) Corre Greedy con ese orden e imprime cuantos colores se obtienen, y chequea que no hayan aumentado los colores.

   (c) Luego crea un nuevo orden a partir del Coloreo obtenido por Greedy en el item anterior, usando OrdenFromKey a partir de definir key usando la salida de PermutarColores con alguna permutacion de los colores pseudoaleatoria creada a partir de la semilla de aleatoriedad.

   (d) Corre Greedy con ese orden e imprime cuantos colores se obtienen, y chequea que no hayan aumentado los colores.

   (e) Luego crea un nuevo orden a partir del Coloreo obtenido por Greedy en el item anterior, usando OrdenFromKey a partir de definir key usando RecoloreoCardinalidadDecrecienteBC.

(f) Corre Greedy con ese orden e imprime cuantos colores se obtieneny chequea que no hayan aumentado los colores.

7. Al termino de esa secuencia para cada uno de los $\alpha + 2$ ordenamientos iniciales, se verifica si la cantidad de colores obtenidas al final es la mejor de las corridas con los ordenamientos iniciales procesados hasta ese momento. Si lo es, se guardan los colores obtenidos en el Coloreo final en un array que indique esto. En resumen, al final de los $\alpha + 2$ ordenamientos iniciales se debe tener un array extra que haya guardado los colores del mejor coloreo logrado hasta ese momento, comenzando desde $\alpha + 2$ lugares distintos y haciendo $3 \times \beta$ reordenamientos "VIT" para cada uno.

8. Comenzando con ese coloreo optimo logrado, se hace un total de $(\alpha + 1) \times 3 \times \beta$ reordenes por colores con la misma estructura de la parte inicial de hacer $(\alpha + 1) \times \beta$ ordenamientos revirtiendo los colores, permutando colores al azar y permutando colores de acuerdo con la cardinalidad del bloque de colores.

9. Al final de todo se imprime cual es la cantidad de colores obtenidas y cuantos Greedys se hicieron en total.

10. Se termina la ejecucion, liberando toda la memoria ocupada en los distintos arrays usados y destruyendo el Grafo.