Dos términos resaltantes, en lo que grafos se refiere, fueron estudiados con detalle para la resolución de la problemática planteada. La primera determinante que nos presenta el enunciado es que toda la resolución será girará en torno a lo que se denomina como un grafo simple. La teoría referente a este término nos indica que un grafo simple no es más que un grafo el cual no posee ni lazos (bucles) ni aristas múltiples, esta última característica no dice que no debe existir más de una arista que conecte dos vértices cualquieras, por ende, del vértice A hacia el vértice B solo debe existir una arista que los conecte.

Siendo este el primer condicional, del cual derivan otros, que el ejercicio nos plantea proseguimos a lo que en realidad éste nos pide, siendo el objetivo verificar si es o no bipartito un grafo dado. Esto nos da otra teoría en la cual fijarnos, la cual nos habla de un grafo bipartito se denomina al grafo cuyos vértices se pueden separar en dos subconjuntos disjuntos V1(G) y V2(G) y las líneas siempre unen vértices de un subconjunto con vértices de otro subconjunto. El problema de verificar si es un grafo bipartito o no es equivalente al problema de colorabilidad con dos colores, pero con diferente nomenclatura. Cualquier grafo con un ciclo de longitud impar es claramente no colorable para dos colores, por ende, podemos decir que un grafo que no es colorable con dos colores posee ciclos impares en él y de esta manera de ninguna manera puede ser un grafo bipartito.

Para colorear un grafo con dos colores, dado un color asignado a un vértice v, debe colorearse el resto del grafo asignando el otro color a cada vértice adyacente a v. Este proceso es equivalente a alternar los dos colores para los nodos en cada nivel de un recorrido DFS (**D**epth **F**irst **S**earch), y durante el retorno de las llamadas recursivas verificar si no hay arista que conecte dos nodos del mismo color, ya que tal arista es evidencia de un ciclo de longitud impar.

Recorrer un grafo consiste en “visitar” cada uno de los nodos a través de las aristas del mismo. Se trata de realizar recorridos de grafos de manera eficiente. Para ello, se pondrá una marca en un nodo en el momento en que es visitado, de tal manera que, inicialmente, no está marcado ningún nodo del grafo.

El tipo de entrada establecido en la solución del problema es dictada por el enunciado, diciendo éste que el grafo deber ser introducido por lista de aristas, entonces se puede asumir que la entrada presente en el algoritmo serán de pares ordenados, recibiendo como primer término del par ordenado (x) el vértice inicial de la arista, y como segundo término (y) el vértice final de ésta.

La primera entrada que pide el archivo de extensión .exe en ejecución con la solución del ejercicio será la cantidad de vértices que tendrá el grafo, almacenando dicho dato en una variable. El siguiente dato que será solicitado será la cantidad de aristas poseerá. Ya por consiguiente comenzará a solicitar los términos de cada arista por lista de pares ordenados, enumerados los vértices automáticamente del numero 1 hasta la cantidad establecida en la primera variable, la cual lo especifica la cantidad de vértices que insertó el usuario.

La salida que es generada por el algoritmo de acuerdo al problema en general muestra la secuencia de pasos que sigue para la resolución de éste, donde primeramente al terminar la introducción de todos los datos, se genera una lista de aristas por pares ordenados “(1,2) (3,4) (4,5)”. Siguiente a esa impresión de datos se muestra la lista de todos los vértices presentes dentro del grafo cada uno dentro de sus respectivos corchetes “[1] [2] [3] [4] [5]”. Comienza con el proceso de recorrido DFS, el cual colorea de manera lógica cada vértice en base a dos colores, y dependiendo del color con el que se coloreó el nodo anterior. Imprimiendo el valor de cada variable mostrando el proceso del algoritmo paso por paso.

El valor AUX impreso es el nodo en el que está situado actualmente el algoritmo, ya debajo muestra el tamaño del vector antes de agregar los vértices incidentes con el nodo instanciado en AUX y se muestra los nodos que se encuentran dentro del vector mismo. Luego ya de haber agregado los vértices incidentes al vector, se imprime nuevamente el tamaño del mismo y los nodos que en ahora residen. Ya para finalizar el bloque de código se imprime por última vez los vértices dentro del vector para verificar que el primer nodo del vector, que este caso y en los siguientes será AUX, sea eliminado y la variable AUX tome como valor el nodo siguiente.

Ya realizado el proceso de coloreo lógico de vértices