******REPÚBLICA BOLIVARIANA DE VENEZUELA**

**MINISTERIO DEL PODER PUPULAR PARA LA DEFENSA**

**UNIVERSIDAD NACIONAL EXPERIMENTAL POLITÉCNICA**

**DE LA FUERZA ARMADA NACIONAL BOLIVARIANA**

**NÚCLEO FALCON - EXTENSIÓN PUNTO FIJO**

Enunciado 13

Circuitos y Caminos Eulerianos.

**Integrantes**

* Chirinos, Enmanuel. 26.058.259
* Petit, Anyelo. 26.057.580
* Vargas, Paola. 25.723.540

Ingeniería de Sistemas,

5to Semestre, Sección “A”.

Punto Fijo, Mayo del 2017

**TEORÍA APLICADA PARA LA RESOLUCIÓN DE LOS PROBLEMAS**

### Para la resolución de los problemas planteados en el enunciado nro. 13, realizamos algoritmos elaborados por nosotros mismos, basándonos en la lógica de métodos de ordenamiento (como el método de burbuja, por ejemplo) y en algoritmos de recorrido, como el DFS (Depth First Search) o Búsqueda de Profundidad, para analizar el grafo, permitiendo conocer cuántos vértices posee y la valencia de cada uno de ellos, y así determinar si dicho grafo posee un circuito (cuando la valencia de todos sus vértices son pares) o un recorrido (cuando posee exactamente dos vértices de valencia impar) y de ser así, lograr recorrer el grafo por el camino más óptimo.

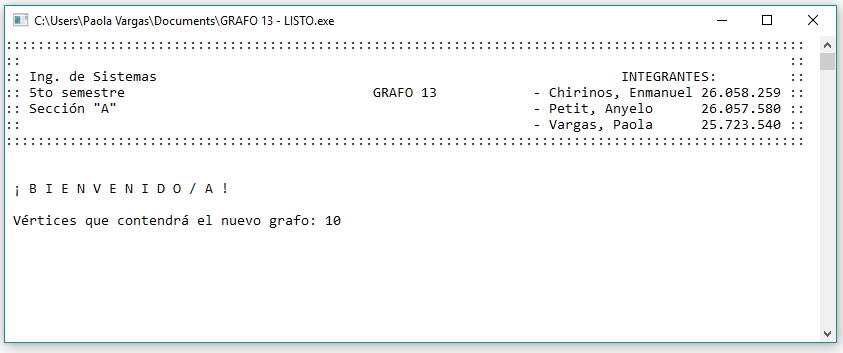
### Un circuito, siendo un grafo con todos sus vértices de valencia par, puede comenzar a ser recorrido desde cualquiera de estos, ya que de esta manera cada uno de ellos tiene entradas y salidas por igual. Si se comienza en un vértice, ubicado en un grafo no dirigido donde existe un circuito euleriano, y este posee seis (6) aristas conectadas directamente a él, por ejemplo, esto significa que tendrá tres (3) aristas por donde podrá salir y tres (3) aristas por donde podrá entrar de igual manera, terminando el circuito en este mismo vértice.

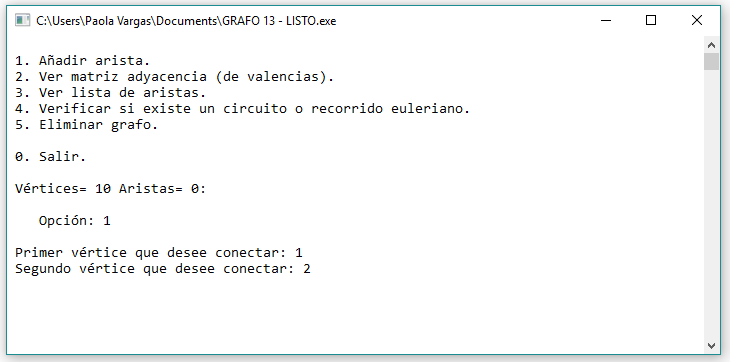
### Por otro lado, un recorrido euleriano, es un grafo que posee exactamente dos (2) vértices de valencia impar, de los cuales uno de ellos será desde donde se comience el recorrido y el otro donde se termine. En un recorrido euleriano, si se parte desde un vértice de valencia N, este poseerá N aristas por donde podrá entrar al vértice y N+1 aristas por donde podrá salir y continuar con el recorrido, hasta llegar al otro vértice de valencia impar, que posea M salidas y M+1 entradas.

### Dada la verificación de dichas condiciones, se realiza el recorrido guardando dinámicamente en variables, el vértice inicial (desde donde se parte) y el vértice final (hacia dónde se irá) a través de una arista, descartándola cuando se pasa por ella y por ende disminuyendo las valencias de los vértices implicados, pasando el vértice inicial a tomar el valor del vértice final y así seguir recorriendo el grafo hasta que termine el circuito o recorrido.

**ESPECIFICACIONES DE ENTRADA**

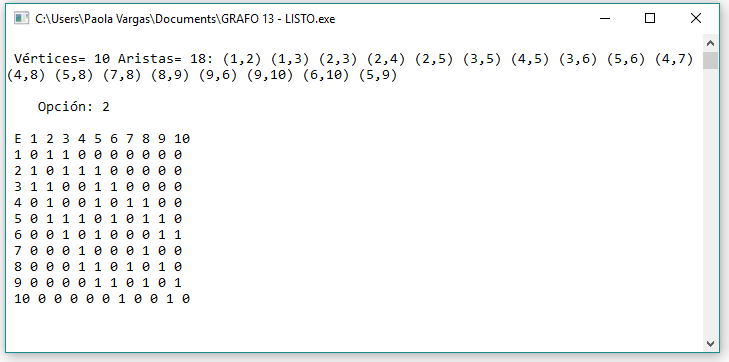
Todos y cada uno de los datos que podrán ser ingresados en el programa serán números enteros, ya sea para elegir las opciones del menú de opciones, para determinar la cantidad de vértices del grafo o para ingresar el primer y el segundo vértice de una arista. Las aristas serán ingresadas dinámicamente mientras se quiera con la **opción 1** del menú. Al ingresar los vértices implicados en la arista, deberá elegir dos (2) números desde 1 hasta la cantidad de vértices que posea el grafo.



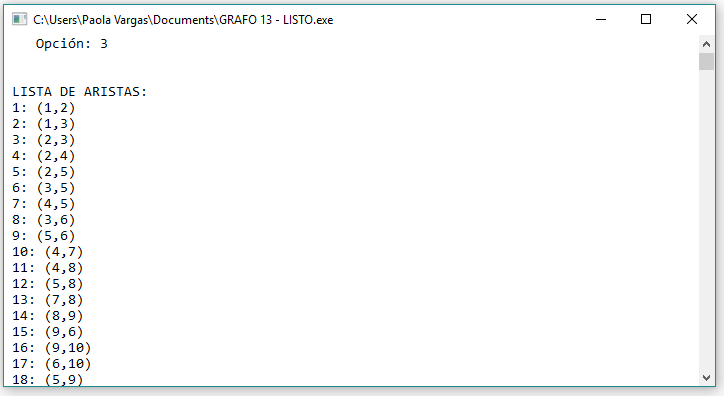


**ESPECIFICACIONES DE SALIDA**

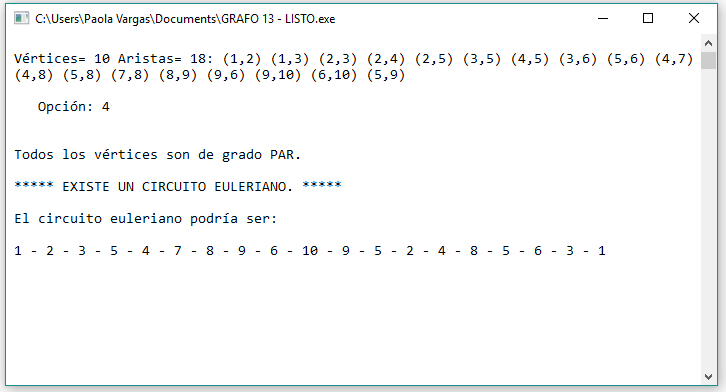
Dados los vértices ingresados para determinar las aristas, con la **opción 2** se mostrará una matriz adyacencia (simétrica por ser un grafo no dirigido) donde se reflejarán las incidencias que tiene un vértice con otro, sumándose 1 por cada arista entre vértices diferentes y 2 por cada lazo (o bucle) en un mismo vértice, tomando en cuenta que se le sumará una entrada y una salida.



Con la **opción 3** se mostrará una lista de aristas, con su índice, el orden en el que fue ingresada y sus vértices enlazados mostrados entre paréntesis y separados por comas.



Con la **opción 4** se verificará si existe un circuito o recorrido euleriano en el grafo. Dada la verificación, se mostrará un mensaje afirmativo o negativo dependiendo del caso y de tal manera de ser positivo, se mostrará un posible camino entre vértices separados por guiones (-). De existir un circuito, se mostrarán un mismo vértice al principio y al final, para determinar que es cerrado. Por otra parte, si existe un recorrido, se mostrará comenzando en un vértice y terminando en otro para determinar que es abierto.



**ESTRUCTURAS DE DATOS UTILIZADAS**

**Variables de tipo ENTERO:**

**op**: utilizada para capturar la opción elegida por el usuario.

**v**: determina la cantidad de vértices del grafo.

**q**:determina la cantidad de aristas del grafo.

**impares**: cantidad de vértices de valencia impar del grafo.

**aislados**: cantidad de vértices aislados del grafo.

**impar1**: captura el primero de los vértices impares del recorrido.

**impar2**: captura el segundo y último de los vértices impares del recorrido.

**aux**: determina el vértice auxiliar o actual en el recorrido del grafo.

**aux2**: determina el siguiente vértice a recorrer en el grafo.

**Variables de tipo ARREGLO (lista de números enteros):**

**av[v]**: arreglo de vértices utilizado para manipular individualmente cada uno.

**val[v]**: arreglo de valencia de cada uno de los vértices del grafo.

**vi[q]**: arreglo del vértice inicial de cada una de las aristas del grafo.

**vf[q]**: arreglo del vértice final de cada una de las aristas del grafo.

**visited[q]**: arreglo de valores booleanos para determinar si cada arista está visitada.

**Variables de tipo MATRIZ (tabla de números enteros):**

**e[v][v]**: matriz adyacencia de tamaño n\*n vértices.

**Variables de tipo VECTOR (lista dinámica de números enteros):**

**inc**: utilizado para guardar las incidencias que tiene cada vértice vecino con el vértice auxiliar.

**Variables de tipo SET (lista ordenada y sin repetición de números enteros):**

**k**: determina si existe más de un componente en el grafo.

**vr**: guarda dinámicamente los vértices recorridos en el grafo.

**Variables de tipo BOOLEANO (retornan verdadero o falso):**

**allvisited**: verifica si ya todas las aristas han sido recorridas y así terminar el ciclo.

**ALGORITMO DISEÑADO PARA RESOLVER EL PROBLEMA**

**Mientras** que *todas las aristas visitadas* sea **igual a** falso

**Por** cada arista en el grafo

**Si** *aux* es el vértice inicial de una arista no visitada

Se ingresa a *inc* el vértice final de la arista

**Si** *aux* es el vértice final de una arista no visitada

Se ingresa a *inc* el vértice inicial de la arista

**Para** el tamaño de *inc*

**Si** vértice vecino se repite más veces en *inc*

**Y** vértice no existe en *vr* (vértices recorridos)

**O** de *aux* a vértice existe al menos un camino válido

**Entonces** *aux2* es igual al vértice

**Para** cada arista en el grafo

**Si** *aux* es el vértice inicial y final de la arista

**Entonces** la arista es un lazo o bucle

**Imprime** *aux*

**Resta** 2 a la valencia de *aux*

**Marca** arista como visitada

**Inserta** *aux* a *vr*

**Si** *aux* y*aux2* son los vértices de una arista

**Y** esta arista no está recorrida

**Si** *aux* posee sólo una arista incidente

**O** *aux* y *aux2* tienen valencia **mayor a** 1

**Entonces** es una arista válida

**Resta** 1 a la valencia de *aux*

**Resta** 1 a la valencia de *aux2*

*aux* ahora toma el valor de *aux2*

**Imprime** *aux*

**Marca** arista como visitada

**Inserta** *aux* a *vr*

*visitadas* **es igual** **a** la suma de todas las aristas visitadas

*grado* es igual a la suma de la valencia de todos los vértices

**Si** *visitadas* **es igual a** la cantidad de aristas en el grafo

**Y** *grado* **es igual a** cero

**Entonces** recorrido terminado

*todas las aristas visitadas* **es igual a** verdadero