UNIVERSIDAD TECNICA ESTATAL DE QUEVEDO

FACULTAD CIENCIAS DE LA INGENIERIA

INGENIERIA EN SOFTWARE

Práctica 1:

Problemas e instancias

Materia:

Modelos de decisión y optimización

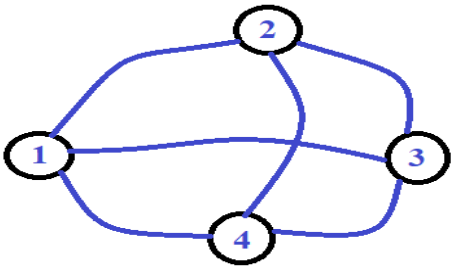
Nombre:

Vera Macias John kleiner

Cutwidth Problem

Introducción

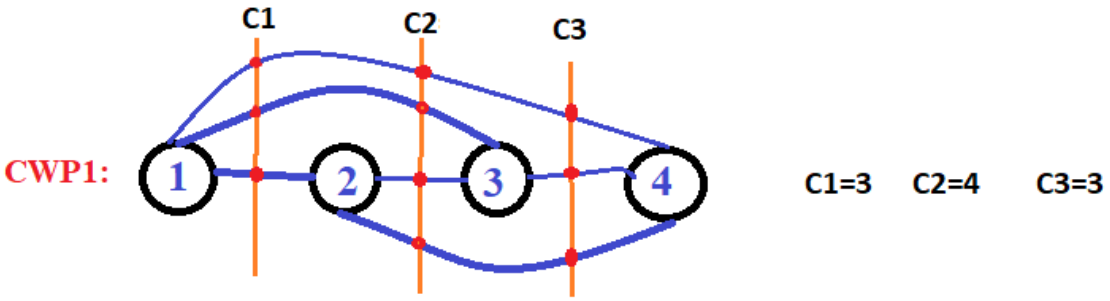
El Problema del Minimizado del CutWidth (CutWidth Problem – CWP) es un problema de optimización aplicado a grafos, cuyo objetivo es encontrar una ordenación lineal de los nodos de manera que el máximo número de aristas cortadas entre dos vértices consecutivos sea mínima. Este tipo de problemas tiene aplicaciones en áreas como la optimización de circuitos eléctricos y la minimización de conexiones cruzadas en redes. Ejemplo tenemos el siguiente grafo:



Descripción del Problema (Cutwidth Problem)

El problema se formula de la siguiente manera: dado un grafo, se debe encontrar una secuencia de los nodos tal que el número máximo de aristas que se "cortan" entre dos nodos adyacentes en la secuencia sea lo más pequeño posible. Primeramente, se realiza permutación con la cantidad de los nodos y se obtiene todas las posibles combinaciones con ordenación lineal de los nodos, para después realizar el respectivo corte entre dos nodos adyacentes de cada combinación.

Ejemplo tenemos la siguiente combinación, aplicando el Cutwidth se ordena de manera lineal con sus respectivos cortes:



Después de realizar los cortes obtenemos el número de arista que se “cortan”, de esto debemos sacar el máximo, en este caso: **C2=4**

Así se repite el proceso con las demás combinaciones. Para finalmente encontrar el valor mínimo de los máximos obtenidos de todas las combinaciones.

De forma matemática el problema de cutwidth se formula de la siguiente manera:

$$\text{Minimizar } z(x) = \max_{i \in V} c_i$$

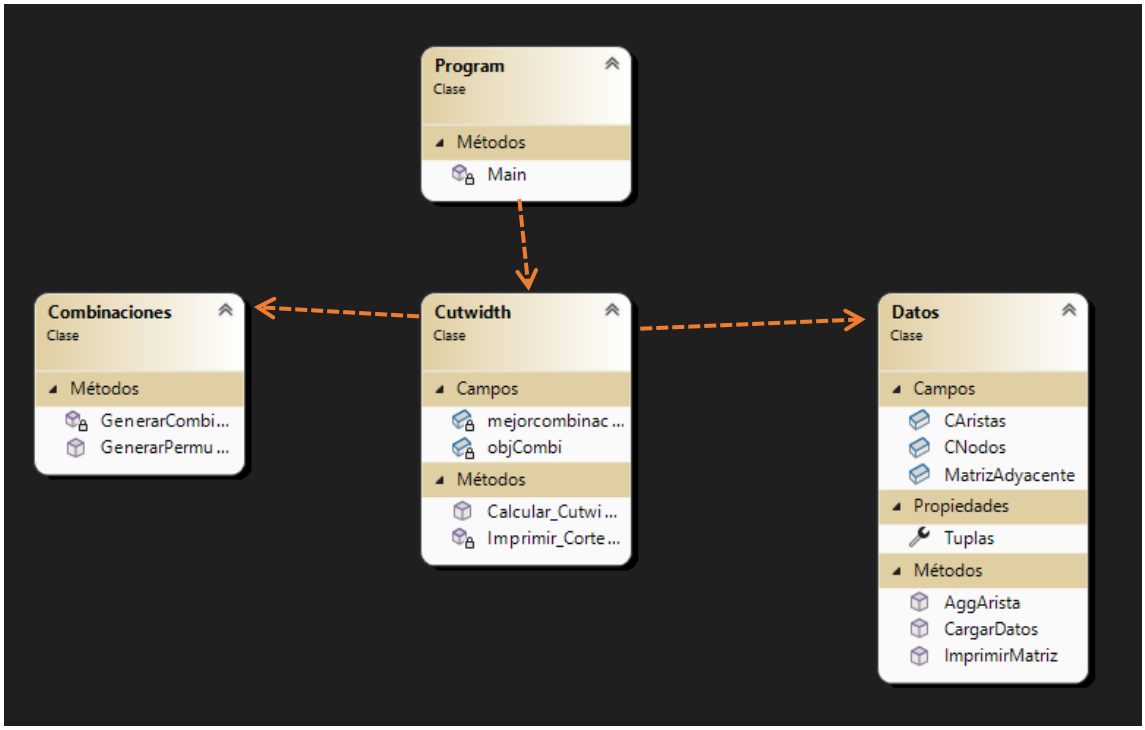
$$c_i = \{(j,k) \in E : x_j \leq x_i < x_k\}$$

Planteamiento para resolver Cutwidth Problem

Para abordar el problema de Cutwidth, se utilizó el lenguaje de programación C# y el entorno de desarrollo integrado (IDE) Visual Studio 2022.

1. **Estructuras de datos para almacenar las instancias en memoria.**

La estructura de datos se encarga de representar y almacenar la información de las instancias del problema. En este caso, el objetivo principal es describir cómo están organizados estos datos para resolver el problema de Cutwidth de manera eficiente. En la siguiente imagen se muestra la distribución de las clases.



❖ Datos

La clase llamada Datos que se encarga de representar y almacenar las instancias del grafo. Esta clase contiene las siguientes estructuras y propiedades:

- **Campos:**
  - **CNodos:** Un entero que almacena la cantidad de nodos en el grafo.
  - **CAristas:** Un entero que almacena la cantidad de aristas en el grafo.
  - **MatrizAdyacente:** Una matriz de enteros de dimensiones CNodos x CNodos que representa las conexiones entre nodos, donde un 1 indica que hay una arista entre dos nodos y un 0 indica que no la hay.

```
Matriz de Adyacencia:
0 1 1 1
1 0 1 1
1 1 0 1
1 1 1 0
```

- **Métodos:**
  - **AggArista(int node1, int node2):** Añade una arista entre dos nodos en la matriz de adyacencia y en la lista de tuplas.

- **CargarDatos(string nomArchivo):** Carga las instancias del grafo desde un archivo en formato Harwell-Boeing (de un archivo .txt).
- **ImprimirMatriz():** Imprime la matriz de adyacencia en la consola.

La clase Datos permite cargar, almacenar y manipular la información de los nodos y aristas del grafo, lo que es fundamental para resolver el problema de CutWidth.

❖ Combinaciones

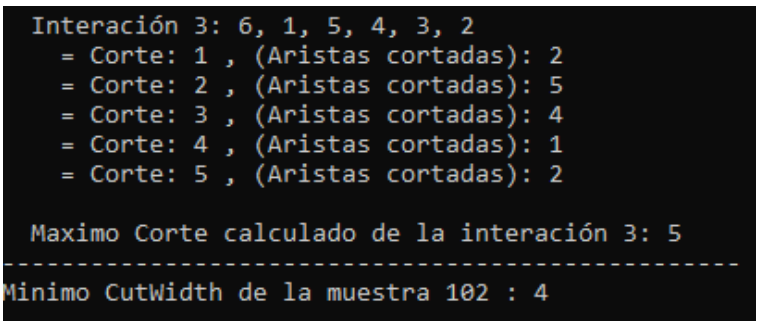
La clase combinaciones se encargar de generar las respectivas permutaciones del grafo de manera aleatoria que se deseen para resolver el problema de Cutwidth. Asi, esta clase tiene un método GenerarPermuAle: Este método recibe una lista de enteros y un número determinado de permutaciones a generar. Utiliza el método GenerarCombiAle para crear cada permutación de manera aleatoria y las almacena en una lista de listas, que luego devuelve.

❖ Cutwidth

Esta clase se encarga de resolver el problema de CutWidth, aquí se plantea:

Función Objetivo: Minimizar el valor máximo de aristas cortadas en una ordenación lineal de los nodos.

Esta clase se utiliza para calcular el máximo corte de las x permutaciones establecidas a evaluar, con el objetivo de minimizar el valor de CutWidth. A través de un enfoque iterativo y aleatorio, evalúa diferentes disposiciones de los nodos y selecciona la que produce el menor corte de aristas. También, imprime por consola los resultados obtenidos, en este caso la combinación que obtuvo el mínimo del máximo de corte.



❖ Program

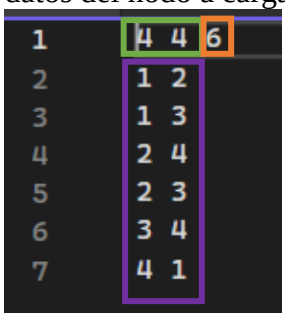
Esta es la clase principal que ejecutar el programa, esta sería la clase principal que une a las demás, es decir, está asociada con las demás clases. También en ella se detalla el archivo .txt a usar y la cantidad de muestras y permutación por muestra que se desean evaluar.

2. Procedimiento para cargar las instancias del problema

Carga de Datos

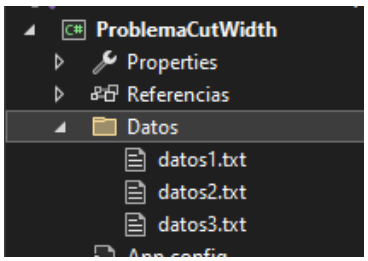
Para comenzar, es necesario cargar los datos en un formato específico. Es decir, los datos deben seguir un único conjunto de instancias denominado Harwell-Boeing.

Siguiendo el formato para cargar los datos, a continuación, tenemos un ejemplo de los datos del nodo a cargar:



- **Primera línea:** Indica el **número de nodos** (repetido dos veces) y el **número de aristas**.
- **Líneas siguientes:** Describen las **aristas entre los nodos (Adyacencias)**, utilizando dos índices que corresponden al número de

Estos se guardan en un archivo .txt, este archivo se guarda dentro de la solución del programa.



El procedimiento para cargar las instancias del grafo se realiza a través del método CargarDatos de la clase Datos. Este método lee un archivo de texto en formato Harwell-Boeing, que contiene la descripción del grafo (nodos y aristas), y lo carga en las estructuras de datos adecuadas. A continuación, se muestra el código para cargar las instancias:

```
public void CargarDatos(string nomArchivo)
        {
            // Obtener la ruta completa al archivo en la carpeta 'datos'
            string Ruta = Path.Combine(AppDomain.CurrentDomain.BaseDirectory,
"Datos", nomArchivo);
            //Verifica si existe el archivo
            if (!File.Exists(Ruta))
            {
                Console.WriteLine("Archivo no encontrado: " + Ruta);
                return;
            }
            //Leer los datos con el separador de " "
            var lines = File.ReadAllLines(Ruta);
            var header = lines[0].Split(' ');

            //Leer la cantidad de nodos y aristas
            CNodos = int.Parse(header[0]);
            CAristas = int.Parse(header[2]);

            MatrizAdyacente = new int[CNodos, CNodos];
            Tuplas = new List<(int, int)>();
            //Lee las aristas
            for (int i = 1; i <= CAristas; i++)
            {
```

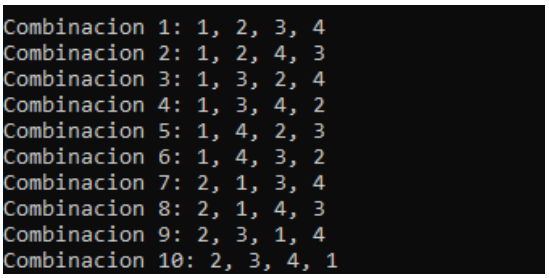
```
            var arista = lines[i].Split(' ');
            int node1 = int.Parse(arista[0]) - 1;
            int node2 = int.Parse(arista[1]) - 1;
            AggArista(node1, node2);
        }
    }
```

Este método lee la cantidad de nodos y aristas del grafo, y luego almacena las conexiones en la matriz de adyacencia y en la lista de tuplas.

3. Estructura de una solución

Para resolver el problema de CutWidth, se definió cómo se evaluará una solución. Así, la estructura de la solución y su evaluación se manejan mediante la clase Program y la clase Cutwidth que a su vez usa la clase Combinaciones. La clase Combinaciones genera combinaciones (permutaciones) de manera aleatoria de ordenaciones de nodos y después le devuelve esa lista generada a la clase Cutwidth la cual calcula el CutWidth para cada una.

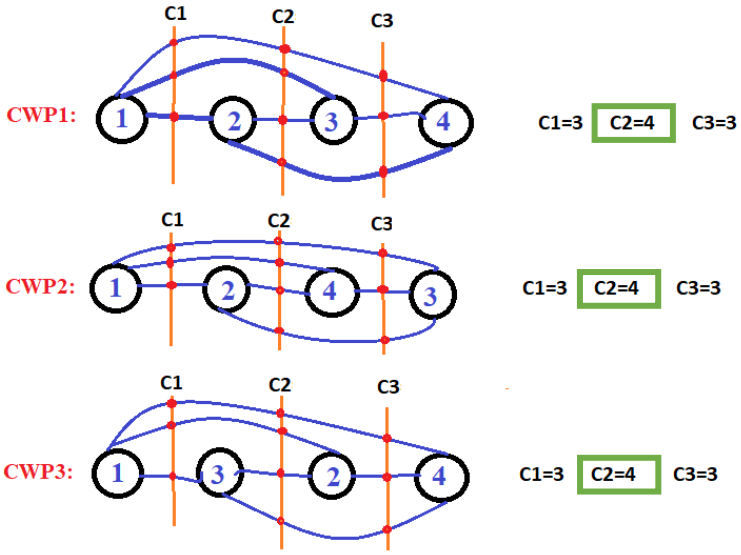
La solución al problema se basa en:
Del conjunto de permutaciones totales (n!) se sacan x muestras con x interacciones definidas previamente las cuales son permutaciones (combinaciones) posibles de los nodos, estas combinaciones son generadas de manera aleatoria. Hay que decir que la población en este caso sería la población total de combinaciones. Para ellos se creó una clase llamada **Combinaciones** en donde se hiso uso de una función para generar x cantidad de permutaciones de manera aleatoria. A continuación, se muestra un ejemplo de posibles combinaciones de un grafo de ejemplo:



Para realizar el proceso del CutWidth, este proceso se realiza en la clase CutWidth donde se realiza el proceso del corte sacando el máximo corte de cada combinación, sacar el mínimo de por muestra y el mínimo final de todas las muestras.

Así, cada muestra va a tener x combinaciones, en donde para cada permutación, se calcula el CutWidth, primero realizando el corte y calculando el número máximo de aristas cortadas entre nodos adyacentes en la secuencia. Para ellos se hizo uso de un bucle for que recorra la lista de combinaciones sacada anteriormente y mediante otros bucles for se sacó la cantidad de cortes:

Muestra 1:



Después se saca el valor máximo de los cortes de cada combinación.

Máximo corte del CWP1 = **4**

Máximo corte del CWP2 = **4**

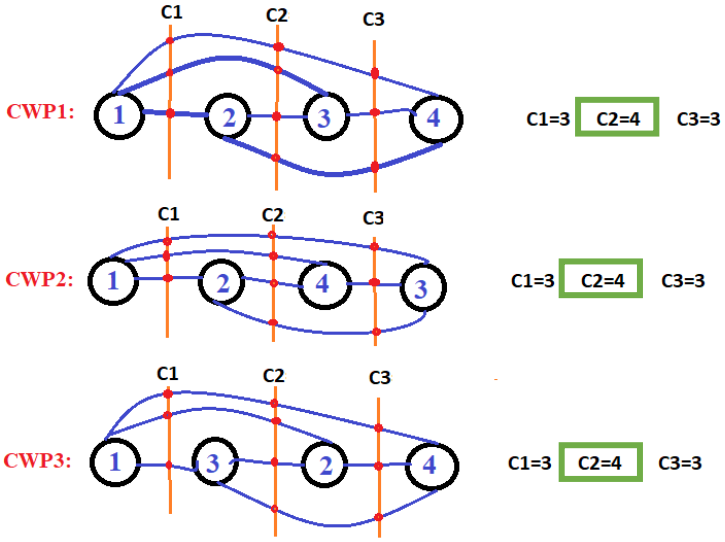
Máximo corte del CWP3 = **4**

A continuación se saca del valor mínimo de las interacciones (combinaciones) de la muestra 1, en este caso:

Mínimo de la muestra 1 = **4**

Se repite el proceso anterior con la siguiente muestra:

Muestra 2:



Máximo corte del CWP1 = **4**

Máximo corte del CWP2 = **4**

Máximo corte del CWP3 = **4**

Mínimo de la muestra 2 = **4**

Después de realizar el mismo proceso se saca al final el valor mínimo de la muestra 2, al final se saca el valor minino de los mínimos de las muestras calculadas, en este caso tenemos de ejemplo dos muestras:

Mínimo de la muestra 1 = **4**

Mínimo de la muestra 2 = **4**

Finalmente, se saca el valor mínimo de los mínimos de las muestras calculadas.

Mínimo = **4**

4. Procedimiento para calcular la función objetivo

Para calcular la función objetivo, primeramente tenemos que tener claro cuál es, en este caso para el problema de CutWidth sería la siguiente:

Función Objetivo: Minimizar el valor máximo de aristas cortadas en una ordenación lineal de los nodos.

Población: Cantidad de permutaciones conocidas de ese grafo.

Solución: Una ordenación de los nodos que minimiza la función objetivo.

Evaluación del CutWidth: Para realizar el cálculo del CutWidth, este se realiza mediante una función definida en la clase CutWidth llamada Calcular_Cutwidth en donde se saca las respectivas permutaciones a usarse y se realiza el corte, se evalúa el número de aristas cortadas en cada corte entre nodos consecutivos, almacenando el valor máximo de cortes para esa permutación.

Ejemplo:

Se muestra un ejemplo de cómo se realiza el proceso:

Se calcula el **CutWidth** para un grafo de 6 nodos con 100 muestras y 3 interacciones por cada una.

```
6 6 7
1 2
1 4
1 5
2 3
4 5
4 6
5 6
```

Muestra 1:

```
--------------------------------Muestra 1-------------
  Interación 1: 5, 2, 6, 1, 3, 4
    = Corte: 1 , (Aristas cortadas): 3
    = Corte: 2 , (Aristas cortadas): 5
    = Corte: 3 , (Aristas cortadas): 5
    = Corte: 4 , (Aristas cortadas): 4
    = Corte: 5 , (Aristas cortadas): 3

  Maximo Corte calculado de la interación 1: 5
-------------------------------------------------
  Interación 2: 5, 4, 3, 1, 2, 6
    = Corte: 1 , (Aristas cortadas): 3
    = Corte: 2 , (Aristas cortadas): 4
    = Corte: 3 , (Aristas cortadas): 5
    = Corte: 4 , (Aristas cortadas): 4
    = Corte: 5 , (Aristas cortadas): 2

  Maximo Corte calculado de la interación 2: 5
-------------------------------------------------
  Interación 3: 6, 5, 4, 2, 3, 1
    = Corte: 1 , (Aristas cortadas): 2
    = Corte: 2 , (Aristas cortadas): 3
    = Corte: 3 , (Aristas cortadas): 2
    = Corte: 4 , (Aristas cortadas): 4
    = Corte: 5 , (Aristas cortadas): 3

  Maximo Corte calculado de la interación 3: 4
-------------------------------------------------
Minimo CutWidth de la muestra 1 : 4
```

Muestra 2:

```
--------------------------------Muestra 2-------------
  Interación 1: 5, 2, 6, 1, 3, 4
    = Corte: 1 , (Aristas cortadas): 3
    = Corte: 2 , (Aristas cortadas): 5
    = Corte: 3 , (Aristas cortadas): 5
    = Corte: 4 , (Aristas cortadas): 4
    = Corte: 5 , (Aristas cortadas): 3

  Maximo Corte calculado de la interación 1: 5
-------------------------------------------------
  Interación 2: 5, 4, 3, 1, 2, 6
    = Corte: 1 , (Aristas cortadas): 3
    = Corte: 2 , (Aristas cortadas): 4
    = Corte: 3 , (Aristas cortadas): 5
    = Corte: 4 , (Aristas cortadas): 4
    = Corte: 5 , (Aristas cortadas): 2

  Maximo Corte calculado de la interación 2: 5
-------------------------------------------------
  Interación 3: 6, 5, 4, 2, 3, 1
    = Corte: 1 , (Aristas cortadas): 2
    = Corte: 2 , (Aristas cortadas): 3
    = Corte: 3 , (Aristas cortadas): 2
    = Corte: 4 , (Aristas cortadas): 4
    = Corte: 5 , (Aristas cortadas): 3

  Maximo Corte calculado de la interación 3: 4
-------------------------------------------------
Minimo CutWidth de la muestra 2 : 4
```

La demás muestras…

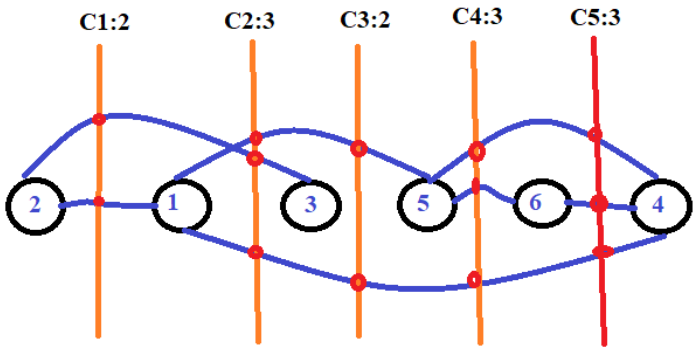
Muestra 100:

```
--------------------------------Muestra 100-----------
  Interación 1: 6, 4, 1, 5, 3, 2
    = Corte: 1 , (Aristas cortadas): 2
    = Corte: 2 , (Aristas cortadas): 3
    = Corte: 3 , (Aristas cortadas): 4
    = Corte: 4 , (Aristas cortadas): 1
    = Corte: 5 , (Aristas cortadas): 2

  Maximo Corte calculado de la interación 1: 4
-------------------------------------------------
  Interación 2: 1, 3, 4, 5, 2, 6
    = Corte: 1 , (Aristas cortadas): 3
    = Corte: 2 , (Aristas cortadas): 4
    = Corte: 3 , (Aristas cortadas): 5
    = Corte: 4 , (Aristas cortadas): 4
    = Corte: 5 , (Aristas cortadas): 2

  Maximo Corte calculado de la interación 2: 5
-------------------------------------------------
  Interación 3: 6, 3, 2, 5, 4, 1
    = Corte: 1 , (Aristas cortadas): 2
    = Corte: 2 , (Aristas cortadas): 3
    = Corte: 3 , (Aristas cortadas): 3
    = Corte: 4 , (Aristas cortadas): 4
    = Corte: 5 , (Aristas cortadas): 3

  Maximo Corte calculado de la interación 3: 4
-------------------------------------------------
Minimo CutWidth de la muestra 100 : 4
```

Resultado final me da el valor mínimo de los mínimos de las muestras encontrado con la combinación que lo dio:

```
Mejor CutWidth encontrado:

Minimo CutWidth encontrado es de : 3 de la muestra: 17

Mejor combinacion encontrada es:  2, 1, 3, 5, 6, 4

Tiempo: 1,2446938 segundo(s).
```

Representado de manera gráfica:



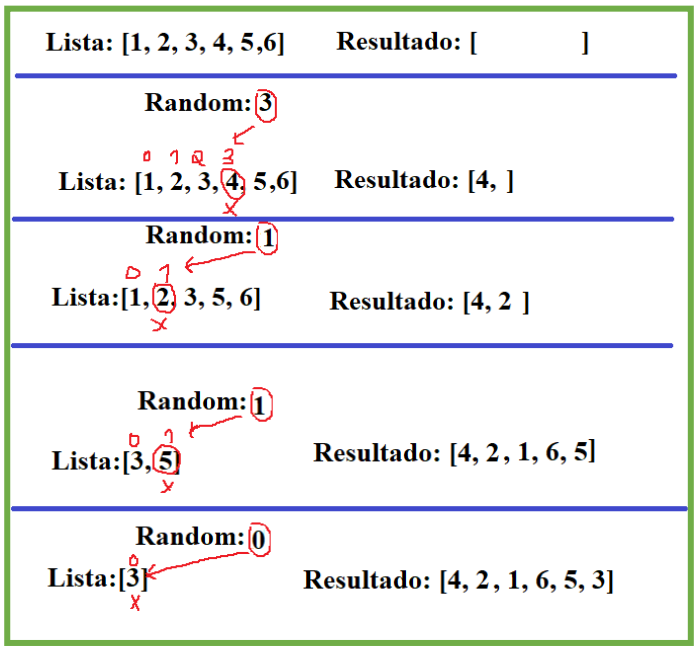
Max= 3

A continuación se muestra el método:

```
        public void Calcular_Cutwidth(int numMuestras, int combinacionesPorMuestra, int CNodos, int[,] MatrizAdyacente)
{

    // Iniciar el cronómetro
    Stopwatch tiempo = new Stopwatch();
    tiempo.Start();

    int minCutWidth = int.MaxValue;
    int mejorMuestra = 0;

    mejorcombinacion = new List<int>();

    for (int muestra = 1; muestra <= numMuestras; muestra++)
    {
        Console.WriteLine($"\n-------------------------------Muestra {muestra}----------------------------");

        var combinaciones = objCombi.GenerarPermuAle(Enumerable.Range(0, CNodos).ToList(), combinacionesPorMuestra);

        int minimoCutWidthMuestra = int.MaxValue;
        var mejorcombinacionMuestra = new List<int>(); // Almacena la mejor combinación de esta muestra

        for (int numeroCombinacion = 1; numeroCombinacion <= combinaciones.Count; numeroCombinacion++)
        {
            var combinacion = combinaciones[numeroCombinacion - 1];
            Console.WriteLine($"  Interación {numeroCombinacion}: {string.Join(", ", combinacion.Select(x => x + 1))}");
            int maxCutWidth = 0;

            for (int i = 1; i < combinacion.Count; i++)
            {
                int corte = 0;
                for (int j = 0; j < i; j++)
                {
                    for (int k = i; k < combinacion.Count; k++)
                    {
                        if (MatrizAdyacente[combinacion[j], combinacion[k]] == 1)
                        {
                            corte++;
                        }
                    }
                }
                Console.WriteLine($"    = Corte: {i} , (Aristas cortadas): {corte}");
                if (corte > maxCutWidth)
                {
                    maxCutWidth = corte;

                }
            }

            Console.WriteLine();
            Console.WriteLine($"  Maximo Corte calculado de la interación {numeroCombinacion}: {maxCutWidth}");
            Console.WriteLine(new string('-', 50));

            if (maxCutWidth < minimoCutWidthMuestra)
            {
                minimoCutWidthMuestra = maxCutWidth;
                mejorcombinacionMuestra = new List<int>(combinacion); // Guardar la combinación actual
            }
        }

        Console.WriteLine($"Minimo CutWidth de la muestra {muestra} : {minimoCutWidthMuestra}\n");
        //Console.WriteLine(new string('-', 100));

        if (minimoCutWidthMuestra < minCutWidth)
        {
            minCutWidth = minimoCutWidthMuestra;
            mejorMuestra = muestra;
            mejorcombinacion = new List<int>(mejorcombinacionMuestra);
        }
    }

    // Detener el cronómetro
    tiempo.Stop();
    Console.ForegroundColor = ConsoleColor.Red;
    Console.WriteLine(new string('-', 100));
    Console.WriteLine("\n----------------------Mejor CutWidth encontrado:----------------------");
    Console.ForegroundColor = ConsoleColor.Green;
    Console.WriteLine($"\nMinimo CutWidth encontrado es : {minCutWidth} de la muestra: {mejorMuestra}");
    Console.WriteLine($"De la Combinacion:  {string.Join(", ", mejorcombinacion.Select(x => x + 1))}");

    Imprimir_Cortes_Mejor(MatrizAdyacente);

    Console.WriteLine($"\nTiempo: {tiempo.Elapsed.TotalSeconds} segundo(s).\n");

    Console.ReadLine();
}
```

5. Método constructivo de una instancia de manera aleatoria

La clase combinaciones se encargar de generar las respectivas permutaciones del grafo de manera aleatoria que se deseen para resolver el problema de Cutwidth. Asi, esta clase tiene un método GenerarPermuAle: Este método recibe una lista de enteros y un número determinado de permutaciones a generar. Utiliza el método GenerarCombiAle para crear cada permutación de manera aleatoria y las almacena en una lista de listas, que luego devuelve a la clase Cutwidth para realizar el respectivo proceso.



Para sacar las permutaciones de manera aleatoria se realiza le siguiente proceso que se muestra en la imagen, en donde de la lista original se saca de manera aleatoria los elementos a otra lista, formando la nueva combinación.

A continuación se muestra parte del código:

```
//Genera una lista con las permutaciones aleatorias
public List<List<int>> GenerarPermuAle(List<int> lista, int cantidad)
{
    var combinaciones = new List<List<int>>();
    var random = new Random();

    // Generar X(cantidad) permutaciones aleatorias
    for (int i = 0; i < cantidad; i++)
    {
        combinaciones.Add(GenerarCombiAle(new List<int>(lista), random));
    }

    return combinaciones;
}

//Genera la combinacion
private List<int> GenerarCombiAle(List<int> lista, Random random)
{
    var resultado = new List<int>();

    while (lista.Count > 0)
    {
        //Genera una valor random de la cantidad de elementos de la lista.
        int index = random.Next(lista.Count);
        resultado.Add(lista[index]);
        lista.RemoveAt(index);
    }

    return resultado;
}
```

6. **Implementar un programa que genere 1000 soluciones aleatorias por cada instancia, reportando el valor de la función objetivo y la estructura de la solución para la mejor solución encontrada para cada instancia**

El programa desarrollado implementa una solución para el problema de minimización del CutWidth (CWP). Este problema consiste en encontrar una ordenación lineal de los nodos de un grafo de tal manera que el máximo número de aristas cortadas entre dos vértices consecutivos sea mínimo.

Función Objetivo: Minimizar el valor máximo de aristas cortadas en una ordenación lineal de los nodos.

Población: Cantidad de permutaciones conocidas de ese grafo.

Solución: Una ordenación de los nodos que minimiza la función objetivo.

- **Generación de Permutaciones Aleatorias:**

Se ha implementado una clase Combinaciones que genera combinaciones aleatorias de nodos del grafo. Esta clase utiliza un generador de números aleatorios para crear permutaciones diferentes en cada ejecución, lo que permite explorar diversas soluciones potenciales.

- **Evaluación de Soluciones:**

La clase Cutwidth contiene la lógica principal para calcular el valor de la función objetivo para cada permutación generada. Este valor se determina evaluando cuántas aristas se cortan en cada posición de la permutación, lo que permite identificar la calidad de cada solución.

- **Selección de la Mejor Solución:**

El programa genera un número configurable de muestras y combinaciones por muestra para evaluar el CutWidth. De entre todas las combinaciones generadas, se selecciona aquella que minimiza el número máximo de aristas cortadas, identificando así la mejor solución.

- **Reporte del Resultado:**

Finalmente, el programa reporta la mejor solución encontrada, mostrando tanto la permutación de nodos que minimiza el CutWidth como el tiempo total de ejecución.

Ejemplo

Dado el siguiente grafo:

```
6 6 7
1 2
1 4
1 5
2 3
4 5
4 6
5 6
```

Se ejecutó con un valor de 1000 muestras con 3 interacciones (combinaciones) cada una

```
// Establecer la cantidad de muestras y combinaciones por muestra
int numMuestras = 1000;
int combinacionesPorMuestra = 3;
```

Se imprime los datos leídos del archivo .txt y la respectiva matriz de adyacencia.

```
Cantidad Nodos: 6 , Cantidad Aristas: 7

Adyacencias:
(0, 1) (0, 3) (0, 4) (1, 2) (3, 4) (3, 5) (4, 5)

Matriz de Adyacencia:
0 1 0 1 1 0
1 0 1 0 0 0
0 1 0 0 0 0
1 0 0 0 1 1
1 0 0 1 0 1
0 0 0 1 1 0
```

```
-------------------------------Muestra 1000------------
 Interación 1: 4, 6, 2, 1, 5, 3
   = Corte: 1 , (Aristas cortadas): 3
   = Corte: 2 , (Aristas cortadas): 3
   = Corte: 3 , (Aristas cortadas): 5
   = Corte: 4 , (Aristas cortadas): 4
   = Corte: 5 , (Aristas cortadas): 1

 Maximo Corte calculado de la interación 1: 5
-------------------------------------------------
 Interación 2: 4, 1, 5, 6, 2, 3
   = Corte: 1 , (Aristas cortadas): 3
   = Corte: 2 , (Aristas cortadas): 4
   = Corte: 3 , (Aristas cortadas): 3
   = Corte: 4 , (Aristas cortadas): 1
   = Corte: 5 , (Aristas cortadas): 1

 Maximo Corte calculado de la interación 2: 4
-------------------------------------------------
 Interación 3: 4, 5, 1, 2, 6, 3
   = Corte: 1 , (Aristas cortadas): 3
   = Corte: 2 , (Aristas cortadas): 4
   = Corte: 3 , (Aristas cortadas): 3
   = Corte: 4 , (Aristas cortadas): 3
   = Corte: 5 , (Aristas cortadas): 1

 Maximo Corte calculado de la interación 3: 4
-------------------------------------------------
Minimo CutWidth de la muestra 1000 : 4
```

Se realiza el respectivo cálculo hasta la muestra 1000:

Finalmente se imprime el resultado de la **Función Objetivo,** obteniendo el mínimo de los máximos de corte.

```
----------------------Mejor CutWidth encontrado:-----

Minimo CutWidth encontrado es : 3 de la muestra: 15
De la Combinacion:  1, 2, 3, 5, 4, 6
Cortes de la combinacion
  Corte 1 , (Aristas cortadas): 3
  Corte 2 , (Aristas cortadas): 3
  Corte 3 , (Aristas cortadas): 2
  Corte 4 , (Aristas cortadas): 3
  Corte 5 , (Aristas cortadas): 2

Tiempo: 4,0819753 segundo(s).
```

Enlace del código:

https://github.com/JohnVeraXD/CutWidth-Problem.git