ESTRUCTURA DE DATOS 2
Código ST0247

Laboratorio Nro. 3
Vuelta atrás (*Backtracking*)

Objetivo: 1. Diseñar algoritmos usando la técnica de diseño de vuelta atrás. **2.** Resolver problemas fundamentales de grafos, incluyendo búsqueda DFS y BFS

Consideraciones: Lean y verifiquen las consideraciones de entrega,

Leer la Guía

Trabajar en Parejas

Si tienen reclamos, regístrenlos en http://bit.ly/2g4TTKf

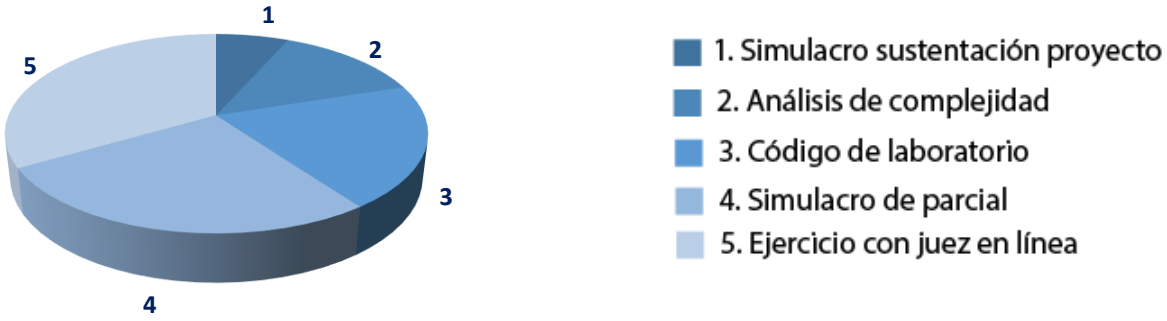
Ver calificaciones en Eafit Interactiva

Subir el informe pdf en la carpeta `informe`, el código del ejercicio 1 en la carpeta `codigo` y el código del 2 en la carpeta `ejercicioEnLinea`

Hoy, plazo de entrega

Si toman la respuesta de **alguna fuente**, deben referenciar según el **tipo de cita**.
Vean *Guía* ***numerales 4.16 y 4.17***

Porcentajes y criterios de evaluación



PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

1. Simulacro de Proyecto

Códigos para entregar en GitHub en la carpeta `codigo`

1 Simulacro de Proyecto

Códigos para entregar en GitHub en la carpeta `codigo`:

Vean Guía numeral 3.4

*Código de laboratorio en **GitHub.*** Vean Guía en **numeral 4.24**

Documentación opcional. Si lo hacen, utilicen **Javadoc** o equivalente. No suban el HTML a GitHub.

No se reciben** archivos en **.RAR** ni en **.ZIP

Utilicen Java, C++ o Python

En la vida real, la documentación del software hace parte de muchos estándares de calidad como CMMI e ISO/IEC 9126

1.1 En la mayoría de las grandes ciudades del mundo, el tráfico es uno de los principales problemas. Existen varias alternativas para solucionar este problema: aumentar la capacidad del transporte masivo, el uso de la bicicleta, horarios diferentes para ingresar a trabajar, el pico y placa, entre otros.

Para nosotros como individuos, una alternativa de mucha ayuda a solventar este problema son las aplicaciones que permiten calcular la ruta que tomará menos tiempo.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ejemplos de estas aplicaciones son Waze y Google Maps. Estas aplicaciones funcionan estimando el tráfico que hay en cada vía a partir de los datos de GPS que suministran los mismos usuarios.

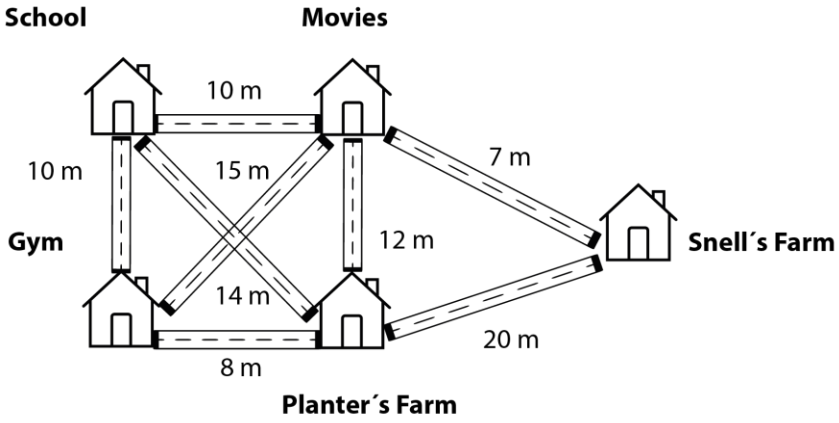
Con las velocidades promedio de cada vía, construyen un grafo dirigido y con ese grafo calculan el camino más corto entre un punto y otro.

Extraída de www.google.com

Calculen la ruta más corta entre dos puntos en un grafo dirigido. Utilicen un algoritmo de *backtracking* para encontrar la solución. NO utilice algoritmos ya desarrollados para este problema como *Dijkstra o A*.*

Nota:. Para realizar una prueba, en Github encontrará el archivo *puentes_colgantes.txt* que contiene el ejemplo a continuación. También encontrará un mapa de la ciudad de Medellín

Ejemplo 1, para el siguiente mapa, el archivo de entrada es el siguiente:



Vertices. Formato: ID, coordenada x, coordenada y, nombre

10000 2.00000 0.00000 School
1 4.00000 1.00000 Movies
2 5.00000 2.00000 Snell
3 2.00000 5.00000 Planters
4 0.00000 2.00000 Gym

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Arcos. Formato: ID, ID, distancia, nombre

10000 1 10.0 Calle 1
10000 3 14.0 desconocido
10000 4 10.0 desconocido
1 10000 10.0 Calle 2ª
1 2 7.0 desconocido
1 3 12.0 desconocido
1 4 15.0 desconocido
2 1 7.0 desconocido
2 3 20.0 desconocido
3 10000 14.0 desconocido
3 1 12.0 desconocido
3 2 20.0 desconocido
3 4 8.0 desconocido
4 10000 10.0 desconocido
4 1 15.0 desconocido
4 3 8.0 desconocido

1.2

En 1996, el computador Deep Blue, desarrollado por IBM, venció al campeón mundial de ajedrez Garry Kasparov. Posteriormente, en 2016, el programa AlphaGo, desarrollado por Google, venció a uno de los mejores jugadores de Go en el mundo.

Extraída de https://bit.ly/2RoG1tH

Programas que sean capaz de jugar ajedrez o Go, son de mucho interés para las grandes multinacionales porque dan los cimientos para sistemas de computación cognitiva como Watson de IBM. Un primer paso para construir sistemas que jueguen ajedrez, es resolver el acertijo de las *n* reinas.

- Implementen el algoritmo de *backtracking* para encontrar UNA solución de las N Reinas.

- **[Opc]** Construyan ejemplos usando JUnit para probar su implementación de las N Reinas usando *backtracking.* Como muestra, usen los ejemplos que ya conocen para el problema de las 4 reinas.

Nota: Si utilizan Python o C++, utilicen una librería equivalente para pruebas unitarias en dichos lenguajes.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

En la vida real, la búsqueda en amplitud es usada para encontrar patrones en redes sociales, como por ejemplo Facebook, para recomendar nuevos amigos a sus usuarios.

1.3 **[Opc]** Implementen en un método el algoritmo de *BFS*, de tal forma de que funcione tanto para la implementación de grafos que utiliza matrices de adyacencia como para la implementación que usa listas de adyacencia, es decir, que reciba un objeto de la clase *Graph*, así como se hizo para *DFS*. El algoritmo debe retornar un *ArrayList* con los vértices en el orden en que los recorrió.

En la vida real, el trabajo de *testing* es uno de los mejor remunerados y corresponde a un Ingeniero de Sistemas

1.4 **[Opc]** Construyan ejemplos usando *JUnit* para probar su implementación de *BFS*

En la vida real, es importante saber si un grafo tiene o no ciclos porque muchos algoritmos sólo funcionan o sólo son eficientes cuando no hay ciclos

1.5 **[Opc]** Implementen un método para un grafo que diga si un grafo tiene ciclos o no

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

2. Simulacro de Maratón de Programación sin documentación HTML, en GitHub,

en la carpeta `ejercicioEnLinea`

2 Simulacro de Maratón de Programación sin documentación HTML, en GitHub, en la carpeta `ejercicioEnLinea`

Vean Guía numeral 3.3

No entregar documentación **HTML**

Utilicen Java, C++ o Python

No se reciben *archivos en* ***.PDF***

No se reciben *archivos en* ***.RAR*** *ni en* ***.ZIP***

Código del ejercicio en línea en ***GitHub.*** Vean Guía en **numeral 4.24**

En la vida real, el camino más corto entre dos puntos en un grafo se aplica en sistemas de información de geográfica como *Google Maps* y en enrutadores de red como el *Cisco ISR* 4000

En la vida real, los algoritmos para encontrar caminos se utilizan en los videojuegos como parte de la inteligencia artificial de la máquina en juegos como League of Legends. Más información en http://bit.ly/2D5Yo3z

2.1 Resuelvan el siguiente problema usando *backtracking y SIN usar el algoritmo de Dijkstra ni otros algoritmos voraces*

Nota: Esta técnica no es la más eficiente para resolver este problema, pero es la que usaremos en este ejercicio

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

A ustedes le entregan un grafo no dirigido con pesos. Los vértices están enumerados del 1 al *n*. Su tarea es encontrar la ruta más corta entre el vértice 1 y el vértice *n*.

Entrada

La primera línea contiene 2 enteros n y m ($2 \le n \le 105$, $0 \le m \le 105$), donde *n* es el número de vértices y m es el número de arcos. Después hay m líneas, donde cada uno contiene un arco de la forma *ai*, *bi* and *wi* ($1 \le ai, bi \le n$, $1 \le wi \le 106$), donde *ai, bi* son los vértices del arco y *wi* es el peso del arco.

Es posible que en el grafo haya ciclos y que haya varios vértices entre el mismo par de vértices.

Salida

Escriban -1 en caso de que no haya camino. Escriban el camino más corto de lo contrario. Si hay varias soluciones, impriman cualquiera de ellas.

Ejemplos de las entradas

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

Ejemplos de la salida

```
1 4 3 5
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ejemplos de las entradas

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

Ejemplos de la salida

```
1 4 3 5
```

[Opc] Para los numerales 2.2 al 2.5, resuelvan los siguientes problemas

2.2 **http://bit.ly/2gTLZ53**

2.3 **http://bit.ly/2hGqJPB**

2.4 **http://bit.ly/2hrrCfS**

2.5 **http://bit.ly/2k8CGSG**

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

3. Simulacro de preguntas de sustentación de Proyecto en la carpeta informe

3 Simulacro de preguntas de sustentación de Proyecto en la carpeta informe

Vean **Guía** numeral 3.4

Exportar y entregar informe de laboratorio en **PDF, en español o Inglés**

Si hacen el ***informe en español***, usen la **plantilla en español**

Si hacen **el informe en inglés,** usen a **plantilla en inglés**

No apliquen **Normas Icontec** para esto

En la vida real, las técnicas usadas para resolver muchos problemas en Ingeniería de Sistemas son las mismas que las existentes para las N reinas

Sobre el Simulacro del Proyecto

3.1 

Para resolver el problema del camino más corto en un grafo, fuera de fuerza bruta y backtracking, ¿Qué otras técnicas computacionales existen?

3.2 

Si fuéramos a enumerar todos los caminos que hay en un grafo para elegir el más corto, en un grafo dirigido completo, ¿Cuántos caminos hay?

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

En la vida real, grandes compañías como Google, valoran más los conocimientos en complejidad computacional que un título de *X* o *Y* universidad. Tomado de http://bit.ly/2hQAZHP

3.3 Tomen los tiempos de ejecución del programa realizado en el numeral 1.2 y en el laboratorio anterior con la solución de fuerza bruta de las *n* reinas. Completen la siguiente tabla.

Si se demora más de 50 minutos, coloque *"se demora más de 50 minutos",* no sigan esperando, podría tomar siglos en dar la respuesta, literalmente.

Valor de N	Tiempo de ejecución
4	
5	
6	
…	
32	
N	**O ( )**

3.4  Para recorrer grafos, ¿en qué problemas conviene usar *DFS*? ¿En qué problemas *BFS*?

Sobre el simulacro de maratón de programación

3.5 Expliquen con sus propias palabras la estructura de datos que utiliza para resolver el problema del numeral 2.1 y 2.2 **[Ejercicio Opcional]** y digan cómo funciona el programa.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Nota: Recuerden que deben explicar su implementación en el informe PDF

3.6 Calculen la complejidad de los ejercicios en línea del numeral 2.1 y 2.2 **[Ejercicio Opcional]** y agréguenla al informe PDF

3.7 Expliquen con sus palabras las variables (*qué es 'n', qué es 'm', etc.*) del cálculo de complejidad del numeral 3.6

Ejemplo de esta respuesta:

"n es el número de elementos del arreglo",
"V es el número de vértices del grafo",
"n es el número de filas de la matriz y m el número de columnas'.

3.8 Escriban una explicación entre 3 y 6 líneas de texto del código del ejercicio en línea del numeral 1.1. Digan cómo funciona, cómo está implementado y destaquen las estructuras de datos y algoritmos usados

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

4. Simulacro de parcial en informe PDF

4 Simulacro de Parcial en el informe PDF

Resuelvan los ejercicios

Para este simulacro, agreguen ***sus respuestas*** **en el informe PDF.**

El ***día del Parcial no tendrán computador, JAVA o acceso a internet.***

Si hacen el ***informe en español***, usen la **plantilla en español**

Exportar y entregar informe de laboratorio en **PDF, en español o inglés**

Si hacen **el informe en inglés,** usen a **plantilla en inglés**

No apliquen **Normas Icontec** para esto

4.1 Wilkenson y Sofronio están aquí de nuevo. En esta vez han traído un juego muy interesante, en el cual Sofronio, en primer lugar, escoge un numero n ($1 \leq n \leq 20$) y, en segundo lugar, escoge tres números a,b y c ($1 \leq a \leq 9, 1 \leq b \leq 9, 1 \leq c \leq 9$).

Después, Sofronio le entrega estos números a Wilkenson y Wilkenson le tiene que decir a Sofronio **la cantidad máxima de números, usando a, b y c (se puede tomar un número más de una vez), que al sumarlos den el valor n**.

Como un ejemplo, si Sofronio escoge n=14 y a=3, b=2, c=7. ¿Qué posibilidades hay de sumar 14 con a,b y c?

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

7+7=14	cantidad es 2
7+3+2+2=14	cantidad es 4
3+3+3+3+2=14	cantidad es 5
…	
2+2+2+2+2+2+2=14	cantidad es 7

La cantidad máxima de números es 7. Esta sería la respuesta que da Wilkenson a Sofronio.

Como Wilkenson es muy astuto, ha diseñado un algoritmo para determinar la cantidad máxima de números y quiere que le ayudes a terminar su código. Asuma que hay al menos una forma de sumar *n* usando los números *a*, *b* y *c* en diferentes cantidades, incluso si algunos de los números se suman 0 veces como sucede en el ejemplo anterior.

```
1 int solucionar (int n, int a, int b, int c)
2   if (n == 0 )
3      return 0;
4   int res = solucionar(_____________) + 1;
5   res = Math.max(_______,___________);
6   res = Math.max(_______,___________);
7   return res;
```

De acuerdo a lo anterior, resuelvan lo siguiente:

4.1.1 Completen el espacio vacío en la línea 4

_____

4.1.2 Completen los espacios vacíos en la línea 5

_____ , _____

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

4.1.3 Completen los espacios vacíos en la línea 6

_____ , _____

4.2 ***Un camino hamiltoniano*** en un grafo dirigido es un camino que visita cada vértice exactamente una vez. Un ***ciclo hamiltoniano*** es un camino hamiltoniano para el cual existe un arco (en el grafo) que conecta el último vértice del camino hamiltoniano con el primer vértice del camino hamiltoniano.

Su tarea **es determinar si dado un grafo, este grafo contiene un ciclo hamiltoniano o no. Si lo contiene, retorne verdadero; de lo contrario, retorne falso.**

Parte de su tarea ya está hecha. La función `sePuede` verifica si un vértice `v` se puede agregar al ciclo hamiltoniano que está almacenado en el arreglo `path` en la posición `pos`, dado un grafo representado con matrices de adyancencia `graph`.

Por simplicidad, sólo se busca si existe un camino que empieza y termina en el primer vértice (es decir, en el vértice 0).

Por esta razón, en el arreglo `path` se entrega con todas sus posiciones en −1, excepto la posición 0, como se muestra en la función `cicloHamil`. También, por esta razón, en el ciclo de la línea 08, `v` inicia se con 1.

```
boolean cicloHamil(int graph[][]) {
  path = new int[g.length];
  for (int i = 0; i < g.length; i++)
      path[i] = -1;
  path[0] = 0;
  return cicloHamilAux(graph, path, 1);
}


boolean sePuede(int v, int graph[][],
                int path[], int pos) {
 if (graph[path[pos - 1]][v] == 0)
    return false;
 for (int i = 0; i < pos; i++)
    if (path[i] == v)
      return false;
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

```
 return true;
}

01 boolean cicloHamilAux(int graph[][],
                  int path[], int pos) {
02   if (pos == ____________){
03    if (graph[path[pos-1]][path[0]] == 1)
04           return true;
05    else
06           return false;
07   }
08   for (int v = 1; v < graph.length; v++) {
09       if (sePuede(__,__,__,__)) {
10           path[pos] = v;
11           if (cicloHamilAux(__,__,__))
12               return true;
13           path[pos] = -1;
14       }
15   }
16   return false;
17 }
```

De acuerdo a lo anterior, resuelvan lo siguiente:

4.2.1 Completen el espacio en línea 02 que corresponde a la condición de parada

_____

4.2.2 Completen los espacios en línea 09 que corresponden al llamado de la función *sePuede*

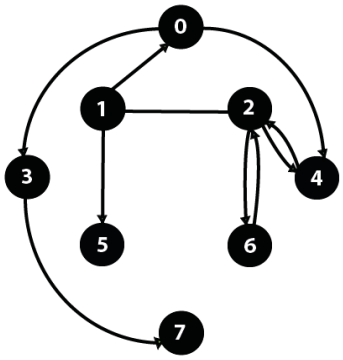
_____, _____, _____, _____

4.2.3 Completen los espacios en la línea 11 que corresponden al llamado recursivo de la función *cicloHamil*

_____, _____, _____

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

4.3 **[Opc]** Para el grafo siguiente, completen la salida que darían los siguientes algoritmos:



De acuerdo a lo anterior, resuelvan lo siguiente:

4.3.1 Completen el orden en que se recorren los nodos usando **búsqueda en profundidad** (en Inglés DFS) a partir de cada nodo. Si hay varias opciones de recorrer el grafo con DFS, elijan siempre el vértice más pequeño.

4.3.2 Completen el orden en que se recorren los nodos usando **búsqueda en amplitud** (en Inglés BFS) a partir de cada nodo. Si hay varias opciones de recorrer el grafo con BFS, elijan siempre el vértice más pequeño.

4.4 **[Opc]** La empresa *Gugol Mas* creó un nuevo sistema de mapas georeferenciados. Sus tecnólogos implementaron fácilmente las funcionalidades de GPS, y la interfaz gráfica web y móvil para el sistema.

Desafortunadamente, Gugol Mas no tiene ingenieros en su nómina y nadie ha podido escribir un método que calcule un camino entre 2 vértices en un grafo dirigido.

Su misión es escribir un método que reciba un digrafo y el identificador de dos vértices, y que retorne un camino entre los dos vértices representado como una lista de enteros. Si no hay camino, retorne una lista vacía.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Pista: En un mapa, las intersecciones se representan como vértices y las vías como arcos.

```
public class EjemplosGrafo {
    public static LinkedList<Integer>
     unCamino(Graph g, int p, int q) {
       ...
    }
}
```

Tengan en cuenta que la clase Graph está definida de la siguiente forma:

```
public class Graph { // Todos los mtodos
   Graph(int vertices); // son públicos
   ArrayList<Integer> getSuccessors(int vertice);
   int size(); // Nmero de vrtices
   int getWeight(int p, int q); // peso del arco
}
```

4.5 El problema de la **subsecuencia común más larga** es el siguiente. Dadas dos secuencias, encontrar la longitud de la secuencia más larga presente en ambas.

Una subsecuencia es una secuencia que aparece en el mismo orden relativo, pero no necesariamente de forma contigua.

Como un ejemplo, “abc”, “abg”, “bdf”, “aeg” y “acefg” son subsecuencias de “abcdefg”. Entonces, para una cadena de longitud *n* existen posibles subsecuencias.

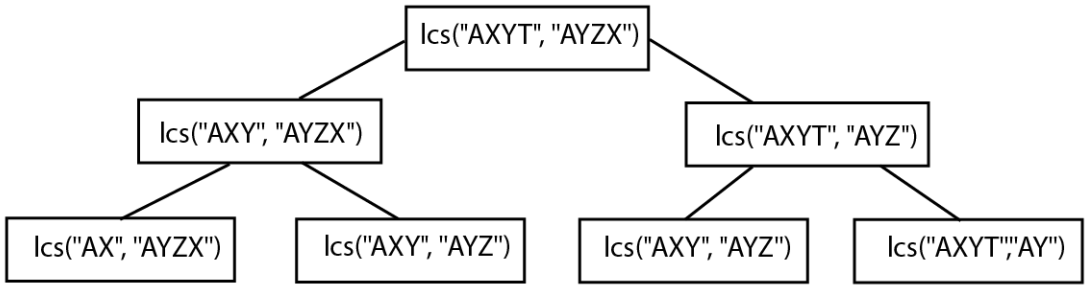
Este problema es utilizado en la implementación del comando `diff`, para comparación de archivos, disponible en sistemas Unix. También tiene muchas aplicaciones en bioinformática.

Consideren los siguientes ejemplos para el problema:

- Para “ABCDGH” y “AEDFHR” es “ADH” y su longitud es 3.
- Para “AGGTAB” y “GXTXAYB” es “GTAB” y su longitud es 4.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Una forma de resolver este problema es usando *backtracking*, como un ejemplo, para las cadenas “AXYT” y “AYZX”, dada una función recursiva `lcs` que resuelve el problema, se obtendría el siguiente árbol (parcial) de recursión:



Al siguiente código le faltan algunas líneas, complétenlas por favor:

```
01 private int lcs(int i, int j, String s1, String s2){
02     if(i == 0 || j == 0){
03         return 0;
04     }
05     boolean prev = i < s1.length() && j < s2.length();
06     if(prev && s1.charAt(i) == s2.charAt(j)){
07         return ______+ lcs(i - 1, j - 1, s1, s2);
08     }
09     int ni = lcs(i - 1, j, s1, s2);
10     int nj = lcs(i, j - 1, s1, s2);
11     return Math.max(_____, _____);
12 }
13 public int lcs(String s1, String s2){
14     return lcs(s1.length(), s2.length(), s1, s2);
15 }
```

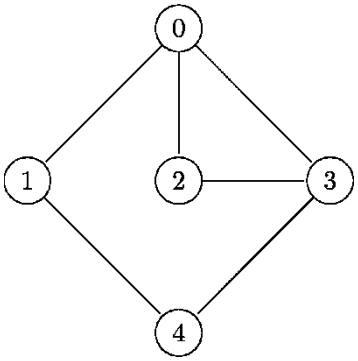
4.5.1 Línea 7 _____

4. 5.2 Línea 11 _____, _____ .Y completen la complejidad por favor

4. 5.3 Supongan que *n* es la suma de la longitud de las dos cadenas. El algoritmo `lcs` ejecuta, en el peor de los casos, *T*(*n*) = _____ instrucciones.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

4.6 **[Opc]** Los ejercicios de esta sección se deberán resolver de acuerdo al siguiente grafo



4.6.1 DFS. Un posible recorrido ***DFS*** del grafo anterior, al ejecutarlo desde el vértice 0, es:

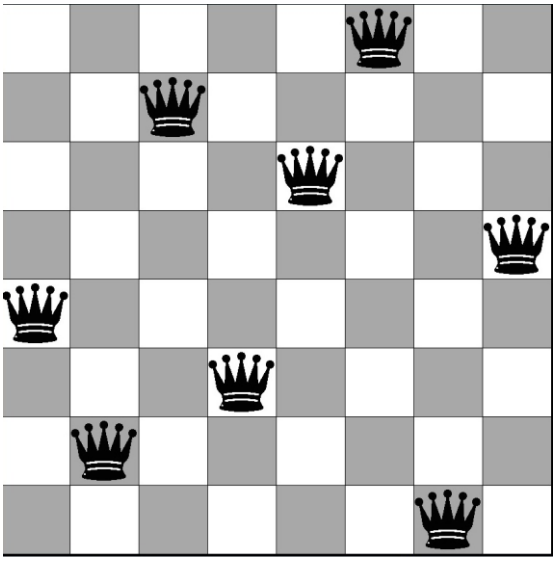
a) 0,4,1,2,3
b) 0,2,4,3,1
c) 0,1,4,3,2
d) 0,4,2,3,1

4. 6.2 BFS. Un posible recorrido ***BFS*** del grafo anterior, al ejecutarlo desde el vértice 0, es:

a) 0,1,2,3,4
b) 0,1,4,2,3
c) 0,4,3,2,1
d) 0,4,2,1,3

4.7 El problema de las *N* reinas consiste en tomar un tablero de ajedrez de *N*×*N* y ubicar *N* reinas de tal manera que ninguna reina quede amenazada. Una posible solución para *N*=8 sería:

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473



Solucion = {4,6,1,5,2,0,7,3} (*columnas*)

Ayúdanos a resolver el problema anterior. La solución consiste en entregar un arreglo *a* donde es un entero que indica la columna de la fila *i* donde hay una reina.

```
void sol(int[] a, int r) {
  int N = a.length;
  if (...........) { //Línea 3
    print(Arrays.toString(a));
    return;
  }
  for (int i = 0; i < N; ++i) {
    a[r]= ........; //Línea 8
    if(place(a,r)) sol(a, .......); //Línea 9
  }
}
boolean place(int[]a,int r){
  for(int i=0;i<r;++i){
    if(a[i]==a[r]) return false;
    if((a[i]-a[r]) == (r-i)) return false;
    if((a[r]-a[i]) == (r-i)) return false;
  }
  return true;
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

```
}
void print(int[]a){
  int n=a.length;
  System.out.print("[");
  for(int i=0;i<n-1;i++){
    System.out.print(a[i]+",");
  }
  System.out.println(a[n-1] +"]");
}
```

De acuerdo a lo anterior, resuelvan lo siguiente:

4.7.1. Completen la línea 3:

_____

4.7.2. Completen la línea 8 .....

_____

4.7.3. Completen la línea 9 .....

_____

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

5. [Opcional]
Lecturas Recomendadas

5 [Opc] Lecturas recomendadas

Vean Guía en ***numeral 3.5 y 4.20***

Exportar y entregar informe de laboratorio en **PDF, en español o Inglés**

Si hacen el ***informe en español***, usen la **plantilla en español**

Si hacen **el informe en inglés,** usen a **plantilla en inglés**

No apliquen **Normas Icontec** para esto

"El ejercicio de una profesión requiere la adquisición de competencias que se sustentan en procesos comunicativos. Así cuando se entrevista a un ingeniero recién egresado para un empleo, una buena parte de sus posibilidades radica en su capacidad de comunicación; pero se ha observado que esta es una de sus principales debilidades…" *Tomado de* http://bit.ly/2gJKzJD

Lean a ***"R.C.T Lee et al., Introducción al análisis y diseño de Algoritmos. Capítulo 5. Páginas 157 – 181.",*** y sumen puntos adicionales, así:

Hagan un mapa conceptual con los principales elementos teóricos.

Si desean una lectura adicional en español, consideren la siguiente ***"John Hopcroft et al., Estructuras de Datos y Algoritmos, Sección 10.4. 1983",*** que encuentran en biblioteca

Nota: Estas respuestas también deben incluirlas en el informe PDF

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

6. [Opcional] Trabajo en Equipo y Progreso Gradual

6 [Opc] Trabajo en Equipo y Progreso Gradual

Vean Guía en ***numeral 3.6, 4.21, 4.22*** *y* ***4.23***

Exportar y entregar informe de laboratorio en **PDF, en español o Inglés**

Si hacen el ***informe en español***, usen la **plantilla en español**

Si hacen **el informe en inglés,** usen a **plantilla en inglés**

No apliquen **Normas Icontec** para esto

El trabajo en equipo es imprescindible. "Algunos medios retratan la programación como un trabajo solitario, la realidad es que requiere de mucha comunicación y trabajo con otros. En la vida laboral serás parte de un equipo de trabajo y deberás comunicarte con otras personas". *Tomado de* http://bit.ly/2qJKzJD

6.1 Entreguen copia de todas las actas de reunión usando el *tablero Kanban*, con fecha, hora e integrantes que participaron

6.2 Entreguen el reporte de *git* con los cambios en el código y quién hizo cada cambio, con fecha, hora e integrantes que participaron

6.3 Entreguen el reporte de cambios del informe de laboratorio que se genera *Google docs* o herramientas similares

NOTA: Estas respuestas también deben incluirlas en el informe PDF

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

7. [Opcional] Laboratorio en Inglés con plantilla en Inglés

7 [Opc] Laboratorio en inglés

*Vean Guía en **numeral 3.6, 4.21, 4.22** y **4.23***

Exportar y entregar informe de laboratorio en **PDF, en español o Inglés**

Si hacen el ***informe en español***, usen la **plantilla en español**

Si hacen **el informe en inglés,** usen a **plantilla en inglés**

No apliquen **Normas Icontec** para esto

El inglés es un idioma importante en la Ingeniería de Sistemas porque la mayoría de los avances en tecnología se publican en este idioma y la traducción, usualmente se demora un tiempo.

Adicionalmente, dominar el inglés permite conseguir trabajos en el exterior que son muy bien remunerados. *Tomado de [goo.gl/4s3LmZ](goo.gl/4s3LmZ)*

Entreguen el código y el informe en inglés.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Resumen de ejercicios a resolver

1.1. Calculen la ruta más corta entre dos puntos en un grafo dirigido. Utilicen un algoritmo de *backtracking* para encontrar la solución. NO utilice algoritmos ya desarrollados para este problema como *Dijkstra o A*.*

1.2. Implementen el algoritmo de *backtracking* para encontrar UNA solución de las N Reinas

[Opcional] Construyan ejemplos usando JUnit para probar su implementación de las N Reinas usando *backtracking.* Como muestra, usen los ejemplos que ya conocen para el problema de las 4 reinas.

1.3. Implementen en un método el algoritmo de *BFS*, de tal forma de que funcione tanto para la implementación de grafos que utiliza matrices de adyacencia como para la implementación que usa listas de adyacencia, es decir, que reciba un objeto de la clase *Graph*, así como se hizo para *DFS*. El algoritmo debe retornar un *ArrayList* con los vértices en el orden en que los recorrió.

1.4 [Ejercicio Opcional] Construyan ejemplos usando *JUnit* para probar su implementación de *BFS*

1.5 [Ejercicio Opcional] Implementen un método para un grafo que diga si un grafo tiene ciclos o no

2.1 Resuelvan el siguiente problema usando *backtracking y SIN usar el algoritmo de Dijkstra ni otros algoritmos voraces*

2.2 http://bit.ly/2gTLZ53

2.3 http://bit.ly/2hGqJPB

2.4 http://bit.ly/2hrrCfS

2.5 http://bit.ly/2k8CGSG

3.1 Para resolver el problema del camino más corto en un grafo, fuera de fuerza bruta y backtracking, ¿Qué otras técnicas computacionales existen?

3.2 Si fuéramos a enumerar todos los caminos que hay en un grafo para elegir el más corto, en un grafo dirigido completo, ¿Cuántos caminos hay?

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

3.3 Tomen los tiempos de ejecución del programa realizado en el numeral 1.2 y en el laboratorio anterior con la solución de fuerza bruta de las *n* reinas. Completen la siguiente tabla

3.4 Para recorrer grafos, ¿en qué problemas conviene usar *DFS*? ¿En qué problemas *BFS*?

3.5 Expliquen con sus propias palabras la estructura de datos que utiliza para resolver el problema del numeral 2.1 y 2.2 **[Ejercicio Opcional]** y digan cómo funciona el programa

3.6 Calculen la complejidad de los ejercicios en línea del numeral 2.1 y 2.2 **[Ejercicio Opcional]** y agréguenla al informe PDF

3.7 Expliquen con sus palabras las variables (*qué es 'n', qué es 'm', etc.*) del cálculo de complejidad del numeral 3.6

3.8 Escriban una explicación entre 3 y 6 líneas de texto del código del ejercicio en línea del numeral 1.1. Digan cómo funciona, cómo está implementado y destaquen las estructuras de datos y algoritmos usados

4. Simulacro de Parcial

5. [Ejercicio Opcional] Lectura recomendada

6. [Ejercicio Opcional]Trabajo en Equipo y Progreso Gradual

7. [Ejercicio Opcional] Entreguen el código y el informe traducido al inglés.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ayudas para resolver los Ejercicios

Ayudas para el Ejercicio 1.2

Pista 1: Vean Guía en numeral 4.14

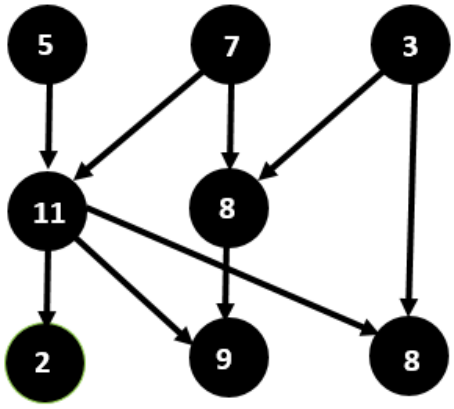
Pista 2: Usen el método *AssertArrayEquals* de *JUnit* que encuentra en **http://junit.sourceforge.net/javadoc/org/junit/Assert.html**

Ayudas para el Ejercicio 1.4

Pista 1: Si deciden hacer la documentación, consulten la Guía en numeral 4.14 y 4.15

Pista 2: Como un ejemplo, construyan el grafo que hicimos en el taller en clase y comprueben que su algoritmo sí arroja la respuesta correcta. Si utilizan *Python* o *C++,* usen una librería para test unitarios disponible en esos lenguajes.

Pista 3: Como un ejemplo, si se llama el método *BFS* con este grafo y con el vértice 7, el *ArrayList* que retorna debe ser así: [7, 8, 11, 2, 9, 10]



PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Pista 4: Usen el método *AssertArrayEquals* de *JUnit* que encuentran en **http://junit.sourceforge.net/javadoc/org/junit/Assert.html**

Ayudas para el Ejercicio 1.5

Pista: Vean Guía en numeral 4.1

Ayudas para el Ejercicio 1.6

Pista 1: Solución en pseudocódigo

```
llenar el arreglo de distancias con infinito
  marcar la distancia al nodo inicial como 0
  llamar dfs con el nodo raiz
  dfs {
    Para (cada hijo) {
       si (puedo mejorar distancias hasta este) {
 marque nueva distancia (mejorada)
   llamese recursivamente para el hijo
      }
    }
  }
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Pista 2: Vean **Problema** y **Solución**

Ayudas para el Ejercicio 1.8

Pista: Vean Guía en numeral 4.1

Ayudas para el Ejercicio 2.1

Pista 1: Construyan un grafo. Utilicen el recorrido DFS.

Pista 2: Retornen una pareja que contiene el camino y el peso total

Ayudas para el Ejercicio 2.4

Pista: Usen un algoritmo para corroborar si es un grafo bipartito. Léase qué es bipartito en **http://bit.ly/2hGwAo2**

Ayudas para el Ejercicio 2.5

Pista 1: Algoritmos para hallar componentes fuertemente conexos. Ordenamiento topológico. DFS. Lean **http://bit.ly/2qTeJKh**

Ayudas para el Ejercicio 3.1

Pista: Lean **http://bit.ly/2hPomyn**

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ayudas para el Ejercicio 3.2

Pista 1: Vean Guía en numeral 4.6

Ayudas para el Ejercicio 3.3

Error Común:

Ayudas para el Ejercicio 3.4

Pista : Vean **http://kevanahlquist.com/osm_pathfinding/**

Ayudas para el Ejercicio 3.6

Pista: Vea Guía en numeral 4.11

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ayudas para el Ejercicio 4.0

Pista: Vean Guía en numeral 4.18

Ayudas para el Ejercicio 5

Pista 1: Para que hagan el mapa conceptual se recomiendan herramientas como las que encuentran en ***https://cacoo.com/*** o ***https://bit.ly/2QM9D3M***

Ayudas para el Ejercicio 6.1

Pista : *Vean* ***Guía en numeral 4.21***

Ayudas para el Ejercicio 6.2

Pista : Vean *Guía en numeral 4.23*

Ayudas para el Ejercicio 6.3

Pista: Vean *Guía en numeral 4.22*

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

¿Alguna inquietud?

CONTACTO

Docente Mauricio Toro Bermúdez
Teléfono: (+57) (4) 261 95 00 Ext. 9473
Correo: mtorobe@eafit.edu.co
Oficina: 19- 627

Agenden una cita dando clic en la pestaña -*Semana*- de http://bit.ly/2gzVg10