Laboratorio A.E.D. Laboratorio 7

Guillermo Román
guillermo.roman@upm.es

Marina Álvarez
marina.alvarez@upm.es

Lars-Åke Fredlund
larsake.fredlund@upm.es

Julio García
juliomanuel.garcia@upm.es

Manuel Carro
manuel.carro@upm.es

Tonghong Li
tonghong.li@upm.es

Normas

- Fechas de entrega y penalización:
  Hasta el jueves 22 de diciembre, 23:59 horas 0 %
  Hasta el viernes 23 de diciembre, 23:59 horas 20 %
  Después la puntuación máxima será 0
- Se comprobará plagio y se actuará sobre los detectados.
- Usad las horas de tutoría para preguntar sobre programación – son oportunidades excelentes para aprender.

Entrega

- Todos los ejercicios de laboratorio se deben entregar a través de

  `http://deliverit.fi.upm.es`

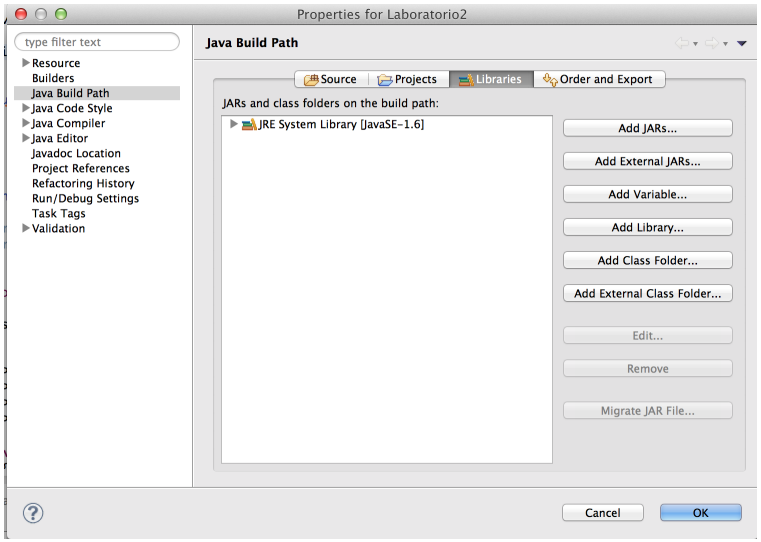
- Los fichero/s que hay que subir es/son `Delivery.java`.

Configuración previa

- Arrancad Eclipse
- Si trabajáis en portátil, podéis utilizar cualquier versión reciente de Eclipse. Es suficiente con que instaléis la *Eclipse IDE for Java Developers*.
- Cambiad a "Java Perspective".
- Debéis tener instalado al menos Java JDK 8.
- Cread un proyecto Java llamado `aed`:
  - Seleccionad separación de directorios de fuentes y binarios.
  - **No debéis elegir la opción de crear el fichero `module-info.java`**
- Cread un *package* `aed.delivery` en el proyecto `aed`, dentro de `src`
- Aula Virtual → AED → Laboratorios y Entregas Individuales → Laboratorio 7 → `Laboratorio7.zip`; descomprimidlo
- Contenido de Laboratorio7.zip:
  - `TesterLab7.java`, `Delivery.java`

Configuración previa

- Importad al paquete `aed.delivery` los fuentes que habéis descargado (`TesterLab7.java`, `Delivery.java`)
- Añadid al proyecto `aed` la librería `aedlib.jar` que tenéis en Moodle (en Laboratorios y Entregas Individuales).

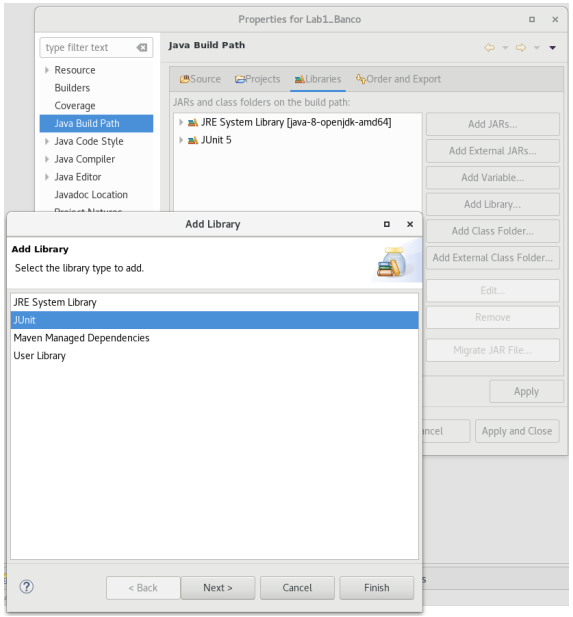


Para ello:

- Project → Properties → Java Build Path. Se abrirá una ventana como la de la izquierda
- Usad la opción “Add External JARs...”.
- Si vuestra instalacion distingue `ModulePath` y `ClassPath`, instalad en `ClassPath`

Configuración previa

- Añadid al proyecto `aed` la librería `JUnit 5`



Para ello:

- Project → Properties → Java Build Path. Se abrirá una ventana como la de la izquierda;
- Usad la opción "Add Library..." → Seleccionad "Junit" → Seleccionad "JUnit 5"
- Si vuestra instalacion distingue `ModulePath` y `ClassPath`, instalad en `ClassPath`

- En la clase `TesterLab7` tenéis las pruebas, para ejecutarlas, abrid el fichero `TesterLab7`, pulsando el botón derecho sobre el editor, seleccionar "Run as..." → "JUnit Test"
- NOTA: Si al ejecutar, no aparece la vista "JUnit", podéis incluirla en "Window" → "Show View" → "Java" → "JUnit"

Documentación de la librería aedlib.jar

- La documentación de la API de `aedlib.jar` está disponible en `https://costa.ls.fi.upm.es/teaching/aed/docs/aedlib/`
- Tambien se puede añadir la documentación de la librería a Eclipse (*no es obligatorio*):
  - En el "Package Explorer": "Referenced Libraries" → aedlib.jar y elige la opción "Properties". Se abre una ventana donde se puede elegir "Javadoc Location" y ahí se pone como "javadoc location path:"

  `https://costa.ls.fi.upm.es/teaching/aed/docs/aedlib/`

  y presionar el buton "Apply and Close"

Tarea: *SS. MM. de Oriente* necesitan ayuda

Los Reyes Magos recorren todos los pueblos cada año pero tienen un problema: si al ir de un pueblo a otro pasan por uno donde ya habían estado, entonces los niños están descontentos con el plazo de entrega sus regalos.
Éstos, organizados mediante grupos de Telegram, les están esperan a la entrada de sus casas para pedirles explicaciones por el retraso.
Vamos a ayudar a los Reyes a encontrar un camino que les lleve por todos los pueblos pero que no pase dos veces por el mismo pueblo, aunque ese camino no sea el recorrido más corto.

Condiciones y modelo

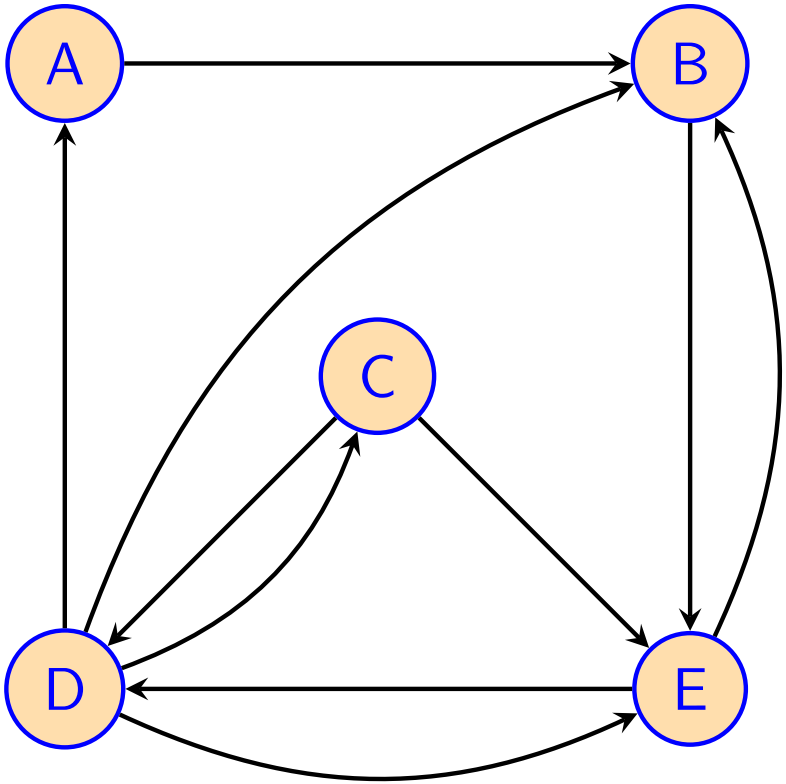
- Los lugares / pueblos pueden estar conectados o no por caminos.
- Si están conectados, cada camino tiene una longitud/coste.
  - Asumiremos que este coste es un entero.
  - Puede ser diferente en cada dirección.
- Podría haber camino de A a B, pero no de B a A.

Objeto	**Formalización**
Lugar	Nodo de grafo
Nombre lugar	Etiqueta nodo
Camino entre lugares	Arista
Distancia entre lugares	Etiqueta de arista
Camino	Lista de nodos conectados por aristas
Longitud de camino	Suma de longitudes de las aristas entre nodos sucesivos en el camino

¿Existe un camino que pase una y solo una vez por cada nodo del grafo?

Ejemplos

Sin coste



Caminos válidos

A - B - E - D - C

C - E - D - A - B

C - D - A - B - E

...

Caminos no válidos

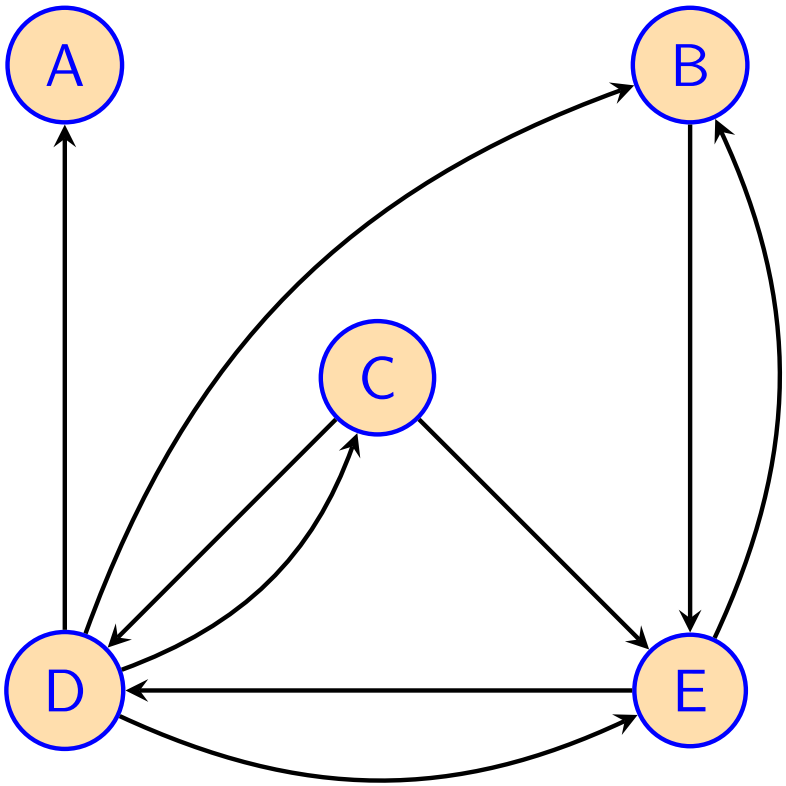
D - A - B - E - D - C

C - D - E - B

...

Ejemplos (cont.)

Sin coste

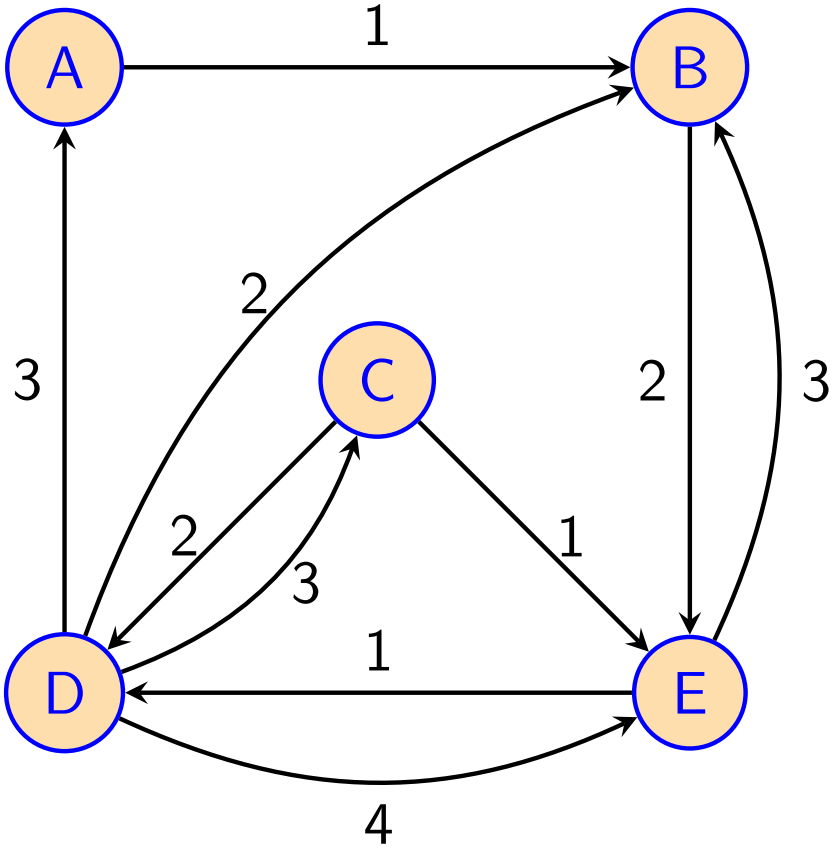


Caminos válidos

Ninguno

Ejemplos (cont.)

Con coste



Caminos válidos y su longitud

A - B - E - D - C $\rightsquigarrow$ 7

C - E - D - A - B $\rightsquigarrow$ 6

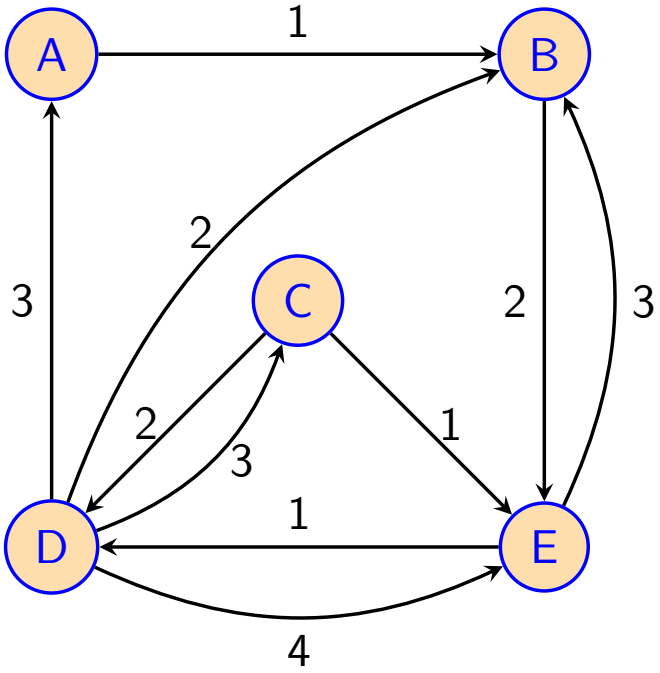
C - D - A - B - E $\rightsquigarrow$ 8

...

Código a realizar: completar la clase Delivery

```
public class Delivery<V> {

  // Construct a graph from a series of vertices
  // with labels of type V and an adjacency matrix.
  public Delivery(V[] places, Integer[][] gmat) {...}

  // Return the constructed graph
  public DirectedGraph<V, Integer> getGraph() {...}

  // Return a path (a list of vertices) in the constructed
  // graph that contains every node in the graph
  // exactly once. Return null if there is no such path.
  public PositionList <Vertex<V>> tour() {...}

  // Return the length of a path in the graph.
  // The path has to have at least one node.
  public Integer length(PositionList<Vertex<V>> path) {...}

}
```

Datos para el constructor



Número de lugares: 5

`gmat[i][j]` = **null**: no hay camino desde el nodo `places[i]` al nodo `places[j]`

`gmat[i][j]` = `k`: hay un camino de longitud `k` desde el nodo `places[i]` al nodo `places[j]`

`places` =

0	A
1	B
2	C
3	D
4	E

`gmat` =

	0	1	2	3	4
0	**null**	1	**null**	**null**	**null**
1	**null**	**null**	**null**	**null**	2
2	**null**	**null**	**null**	2	1
3	3	2	3	**null**	4
4	**null**	3	**null**	1	**null**

Ejemplos de ejecución

```
// Delivery<String> t;
// PositionList<Vertex<String>> trip;
// places = {"A","B"}, gmat = {{null,null},{null,null}}
t = new Delivery<String>(places, gmat);
trip = t.tour();  // ⇝ null

// places = {"A","B"}, gmat = {{null,3},{null,null}}
t = new Delivery<String>(places, gmat);
trip = t.tour();    // ⇝ [v("A"),v("B")]
l = t.length(trip);  // ⇝ 3
```

Ejemplos de ejecución (cont.)

```
// places = {"A","B","C","D"},
// gmat = { {null,    3, null,    1},
//          {null, null,    2, null},
//          {null, null, null, null},
//          {null, null,    2, null}}
t = new Delivery<String>(places, gmat);
trip = t.tour();  // ⇝ null

// gmat = { {null,    3, null,    1},
//          {null, null,    2, null},
//          {null, null, null, null},
//          {null,    4,    2, null}}
t = new Delivery<String>(places, gmat);
trip = t.tour();    // ⇝ [v("A"),v("D"),v("B"),v("C")]
l = t.length(trip);  // ⇝ 7
```

Recomendaciones

- Hay muchas maneras de buscar los caminos que queremos. Es válido explorar todas las posibilidades hasta encontrar uno: recursión + *backtracking*.
- Puede ser computacionalmente muy costoso (p.e., $O(n!)$). Un código descuidado rebasaría tiempos de ejecución razonables.
- No hay restricciones en cuanto a variables internas, variables, métodos privados. . . (pero ver la siguiente transparencia).
- La clase `DirectedGraph<V>` implementa `toString()`. Puede usarse para examinar el grafo generado.
- Si hay varios caminos válidos, algunos serán mínimos. **No** pedimos buscar caminos mínimos — sólo uno que pase una vez por cada nodo.

Notas

- Aparte de las estructuras de datos disponibles en la biblioteca `aedlib`, está permitido utilizar `java.util.Iterator`, pero no otras clases de `java.util` .
- Debe ejecutar `TesterLab7` correctamente y sin mensajes de error
  - Nota: una ejecución sin mensajes de error no significa que el método sea correcto (es decir, que funcione bien para cada posible entrada)
- Todos los ejercicios se comprueban manualmente antes de dar la nota final