EI1022/MT1022 - Algoritmia

Entregable 1 - Esconder el tesoro en el laberinto

Fecha de entrega: miércoles 15 de octubre de 2025

Índice

1 El problema: Esconder el tesoro en el laberinto **1**
- 1.1 Representación de laberintos como grafos no dirigidos . . . 2
- 1.2 Formato de la entrada . . . 3
- 1.3 Formato de la salida . . . 3

2 Archivos de partida **4**
- 2.1 Archivos de prueba: `public.zip` . . . 4
- 2.2 Archivo principal: `e1.py` . . . 4
- 2.3 Visor de laberintos y soluciones: `e1_viewer.py` . . . 6

3 Entrega en el aula virtual **6**

4 Calificación del entregable **7**
- 4.1 Medición de tiempos de ejecución . . . 7
- 4.2 Errores graves en el entregable . . . 8
- 4.3 Penalización por copia . . . 8

1. El problema: Esconder el tesoro en el laberinto

Disponemos de un laberinto conexo de R filas y C columnas en el que puede haber más de un camino entre un par de celdas. Consideraremos que la entrada del laberinto está en la la celda superior izquierda, $(0, 0)$, y la salida en la celda inferior derecha, $(R - 1, C - 1)$.

Vamos a esconder un tesoro en el laberinto. Para ponérselo lo más difícil posible a los cazatesoros queremos averiguar en qué celda colocar el tesoro de forma que el cazatesoros tenga que gastar la mayor cantidad de calorías posible. Cada vez que el cazatesoros se mueve de una celda a otra vecina gasta X calorías. Sin embargo, cuando encuentra el tesoro tiene que cargarlo hasta la salida y, en ese estado, cada cambio de celda le suponen un gasto de Y calorías, con $Y \geq X$. Sabemos que el cazatesoros utilizará el camino más corto para ir de la entrada hasta el tesoro y del tesoro a la salida.

Implementad un programa de línea de órdenes, `e1.py`, que reciba por la entrada estándar un laberinto en el formato que se especifica en el apartado 1.2. Como resultado de su ejecución el programa mostrará sus resultados por la salida estándar en el formato que se detalla en el apartado 1.3.

Los resultados que debéis obtener son:

- Las coordenadas de la celda en la que colocar el tesoro.
- La cantidad de calorías que deberá gastar el cazatesoros para llegar hasta el tesoro y llevárselo a la salida. El cazatesoros siempre utiliza el camino más corto.
- Los caminos desde la entrada al tesoro y desde el tesoro hasta la salida.

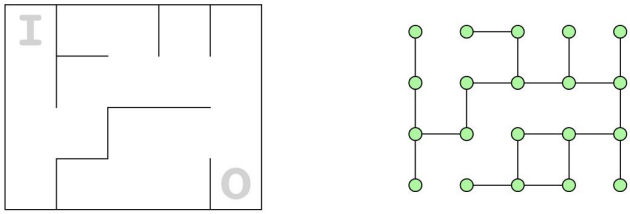
Es posible que existan varias soluciones que maximicen el total de calorías (es decir, empates) y todas se considerarán válidas en la calificación del entregable.

© Universitat Jaume I 2025

1.1. Representación de laberintos como grafos no dirigidos

Para representar un laberinto de R filas y C columnas utilizaremos un *grafo no dirigido*, donde los vértices son las celdas del laberinto y las aristas son las conexiones entre celdas vecinas. La entrada al laberinto será el vértice $(0, 0)$, que se corresponde con la celda superior izquierda y la salida será el vértice $(R-1, C-1)$, que se corresponde con la celda inferior derecha.

En la figura 1 podemos ver la instancia de prueba `e1-01-4x5-220.i` representada de dos formas, como plano y como grafo no dirigido. La entrada y la salida del laberinto están marcadas con una `I` y una `O`, respectivamente.



(a) Laberinto **(b)** Grafo subyacente

Figura 1: Instancia de prueba del archivo `e1-01-4x5-220.i`.

Los valores de gasto de calorías de la instancia anterior son $X = 10$ e $Y = 20$. Debemos utilizar estos valores para encontrar la celda en la que colocar el tesoro.

En la figura 2 podemos ver la solución de esta instancia. La celda amarilla indica la posición del tesoro, la línea roja muestra el camino desde la entrada al tesoro (12 pasos) y la línea azul el camino del tesoro a la salida (5 pasos). Las calorías totales de la solución son $12X + 5Y = 12{\cdot}10 + 5{\cdot}20 = 220$.

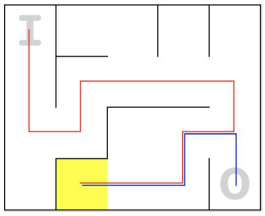


Figura 2: Solución de la instancia `e1-01-4x5-220.i`.

Se garantiza que los laberintos de todas las instancias del problema serán conexos.

Tenéis que implementar un programa de línea de órdenes, `e1.py`, que lea la instancia desde la entrada estándar y muestre el resultado por la salida estándar. El formato de las instancias se especifica en el apartado 1.2 y el de la salida en el apartado 1.3.

Para poder optar a la máxima nota, el coste temporal de las funciones `read_data` y `process` deberá ser lineal con el número de vértices del grafo.

Para representar el laberinto debéis utilizar la clase `UndirectedGraph` de la biblioteca `algoritmia`.

© Universitat Jaume I 2025

1.2. Formato de la entrada

Una instancia es un archivo de texto con el siguiente contenido:

- La primera línea contendrá dos enteros separados por un blanco. El primer entero, X, son las calorías que gasta el cazatesoros cada vez que cambia de celda cuando no está cargando con el tesoro y el segundo entero, Y, son las que gasta cuando sí está cargando con el tesoro:

  ```
  10 15
  ```

- La segunda línea contendrá el tamaño del laberinto como dos enteros separados por un blanco, el número de filas y el de columnas, respectivamente:

  ```
  4 5
  ```

- El resto de líneas serán los pasillos del laberinto (aristas del grafo), con un pasillo por línea, representado mediante cuatro enteros. Por ejemplo, el pasillo `((1, 2), (1, 3))` se representa con la línea:

  ```
  1 2 1 3
  ```

Para obtener los números de una cadena de texto te resultará muy útil el método `split`:

```python
line = '1 2 3 4'
nums = [int(n) for n in line.split()]
print(nums)  # Resultado: [1, 2, 3, 4]
```

1.3. Formato de la salida

El programa deberá mostrar, por la salida estándar, cuatro líneas de texto, en orden:

- Una línea con dos enteros, la fila y la columna, respectivamente, de la posición el tesoro:

  ```
  3 1
  ```

- Una línea con un entero que indica las calorías que necesitará el cazatesoros para ir desde la entrada hasta el tesoro y luego para cargarlo hasta la salida, recorriendo en ambos casos el camino más corto:

  ```
  220
  ```

- Una línea con la lista de los vértices del camino desde la entrada hasta el tesoro. Cada vértice se representa con dos enteros, `row` y `col`, separados por un blanco, y se separa del siguiente vértice con la cadena ‘ `#` ’:

  ```
  0 0 # 1 0 # 2 0 # 2 1 # 1 1 # 1 2 # 1 3 # 1 4 # 2 4 # 2 3 # 3 3 # 3 2 # 3 1
  ```

- Una línea con la lista de los vértices del camino desde el tesoro hasta la salida. Utiliza el mismo formato que la línea anterior:

  ```
  3 1 # 3 2 # 3 3 # 2 3 # 2 4 # 3 4
  ```

Los archivos `e1-*.o`, con las soluciones de las instancias de prueba, tienen este formato. Las cuatro líneas de texto anteriores que se han utilizado como ejemplo, son el contenido del archivo `e1-01-4x5-220.o`.

© Universitat Jaume I 2025

2. Archivos de partida

En el aula virtual disponéis de una carpeta con el siguiente contenido:

- `public.zip`: un archivo comprimido que contiene la carpeta `public` con un conjunto de diez instancias de prueba y sus soluciones.
- `e1.py`: el programa que debéis de utilizar como punto de partida y que hay que entregar.
- `e1_viewer.py`: un visor de laberintos y sus soluciones.

Los siguientes apartados explican con detalle estos contenidos, pero ya podéis crear un nuevo proyecto PyCharm y añadirle los archivos `e1.py` y `e1_viewer.py` y la carpeta que contiene el fichero `public.zip`.

2.1. Archivos de prueba: `public.zip`

Al descomprimir el fichero `public.zip` obtendréis la carpeta `public`, que contiene diez instancias y sus soluciones. Sus nombres siguen el patrón:

```
e1-<num>-<size>-<cal>.<ext>
```

donde:

- `<num>`: El número de instancia, un número entre 1 y 10.
- `<size>`: El tamaño de la instancia en el formato $rows \times cols$, p.ej. '4x5'.
- `<cal>`: Las calorías de la solución óptima.
- `<ext>`: '`i`' si el archivo se corresponde con una instancia, '`o`' si es una solución.

Es posible que algunas instancias tengan más de una solución válida, por lo que es posible que algunas de vuestras soluciones no coincidan con las que contiene la carpeta. En cualquier caso, las soluciones deben coincidir en el total de calorías.

2.2. Archivo principal: `e1.py`

Contiene la estructura del programa. Debéis modificarlo teniendo en cuenta lo siguiente:

- Podéis añadir funciones o clases adicionales, pero no debéis modificar nada del programa principal. Modificar el programa principal supondrá un cero en la calificación del entregable. Tampoco podéis modificar la firma (el tipo) de las tres funciones que se utilizan en el programa principal.
- Podéis importar módulos de la biblioteca estándar de Python, pero la única biblioteca externa permitida es `algoritmia`. Por ejemplo, `NumPy` es una biblioteca externa y, por lo tanto, no está permitida.

Antes de continuar debemos señalar que, para detectar errores en el entorno de trabajo, el programa incluye una función, `_check_environment`, que se ejecuta automáticamente al lanzar el programa y comprueba que estén instaladas las versiones adecuadas de Python y de la biblioteca `algoritmia`.

Veamos la estructura de `e1.py` y las funciones que debéis implementar.

Por comodidad, el programa define estos tres tipos:

```
type Vertex = tuple[int, int]          # (row, col)
type Edge   = tuple[Vertex, Vertex]    # ((row1, col1), (row2, col2))
type Path   = list[Vertex]             # [(row1, col1), (row2, col2), ...]
```

© Universitat Jaume I 2025

También se define el tipo `Data` para las instancias y el tipo `Result` para los resultados:

```
type Data = tuple[int, int, int, int, UndirectedGraph[Vertex]]
type Result = tuple[Vertex, int, Path, Path]
```

Estos dos tipos se detallan más adelante, en la implementación de las funciones `read_data` y `process`.

El programa principal utiliza la estructura de tres funciones vista en clase:

```
data0 = read_data(sys.stdin)
result0 = process(data0)
show_result(result0)
```

La función `show_result` ya está implementada y no debéis modificarla. Debéis implementar las funciones `read_data` y `process`:

- `read_data(f: TextIO) -> Data:`
  - Entrada: El archivo con la instancia (cuidado, `f` no es un nombre de archivo, es un archivo).
  - Salida: Una tupla `(X, Y, n_rows, n_cols, lab_graph)`, donde:
    - `X`: calorías consumidas al cambiar de celda sin estar cargando con el tesoro.
    - `Y`: calorías consumidas al cambiar de celda mientras se está cargando con el tesoro.
    - `n_rows`: número de filas del laberinto.
    - `n_cols`: número de columnas del laberinto.
    - `lab_graph`: grafo que representa el laberinto.
- `process(data: Data) -> Result:`
  - Entrada: La tupla que devuelve `read_data`.
  - Salida: Una tupla `(treasure_pos, calories, path1, path2)`, donde:
    - `treasure_pos`: tupla `(row, col)` con la posición del tesoro.
    - `calories`: calorías que gastará el cazatesoros.
    - `path1`: lista de celdas del camino de la entrada al tesoro.
    - `path2`: lista de celdas del camino del tesoro a la salida.

Cómo ejecutar vuestro programa `e1.py`

Dado que el programa lee de la entrada estándar, se recomienda ejecutarlo desde el terminal que incorpora el propio PyCharm (el botón está en la parte inferior izquierda). Una vez abierto, utilizad el comando `cd` para ir al directorio donde están `e1.py` y `public`.

Una vez en el directorio, podéis ejecutar el programa con el siguiente comando:

- Linux y MacOS: `python3 e1.py < public/e1-01-4x5-220.i`
- Windows (CMD): `python e1.py < public\e1-01-4x5-220.i`
- Windows (PowerShell): `Get-Content public\e1-01-4x5-220.i | python e1.py`

De ahora en adelante sólo pondremos los comandos en formato compatible con los terminales de Linux y MacOS. Si usáis Windows tenéis que tener en cuenta que el ejecutable se llama `python`, no `python3`, y que el separador de directorios es '\' en lugar de '/'.

© Universitat Jaume I 2025

2.3. Visor de laberintos y soluciones: `e1_viewer.py`

El visor de laberintos utiliza la función `read_data` de vuestro archivo `e1.py`, por lo que no funcionará hasta que la implementéis.

Para ejecutar el visor abrid el terminal de PyCharm e id al directorio donde están `e1_viewer.py`, `e1.py` y la carpeta `public`.

Este programa tiene dos modos de funcionamiento:

- Si le pasáis dos parámetros (el archivo de instancia y el de solución), mostrará la solución sobre el laberinto:

  ```
  python3 e1_viewer.py public/e1-01-4x5-220.i public/e1-01-4x5-220.o
  ```

- Si sólo le pasáis un parámetro (que debe ser el archivo de instancia) ejecutará vuestra función `process` para obtener la solución y la mostrará sobre el laberinto:

  ```
  python3 e1_viewer.py public/e1-01-4x5-220.i
  ```

Una vez se abra la ventana con el laberinto, podéis utilizar las teclas '`g`' y '`l`' para ocultar/mostrar el grafo subyacente y el laberinto que representa. Pulsad la tecla '`Escape`' o '`Enter`' para cerrar la ventana y terminar el programa.

Por último, en la barra de título de la ventana se muestra el número de pasos de los dos caminos, sus calorías y las calorías totales.

3. Entrega en el aula virtual

La entrega consistirá en subir dos ficheros a la tarea correspondiente del aula virtual, sólo debe subirlos uno de los miembros de la pareja. Estos son los dos ficheros:

- `e1.py`: El fichero con el código del entregable.
- `alxxxxxx_alyyyyyy.txt`: Un fichero de texto que deberá contener el nombre de cada miembro de la pareja (un nombre por línea). Evidentemente, en el nombre del fichero tenéis que reemplazar `alxxxxxx` y `alyyyyyy` por los correspondientes a los dos miembros de la pareja. Si la entrega es individual, el fichero de texto deberá llamarse `alxxxxxx.txt`, reemplazando `alxxxxxx` por el que corresponda.

Vuestra entrega debe cumplir las siguientes restricciones:

- Sólo debéis subir los dos ficheros solicitados, nada más.
- Los nombres de los dos ficheros deben utilizar únicamente minúsculas.
- El carácter '`_`' que aparece en `alxxxxxx_alyyyyyy.txt` es el guion bajo, no utilicéis el carácter '`-`' ni ningún otro carácter similar como separador.

Información importante si utilizáis Windows para subir el archivo al aula virtual

Si usáis Windows, es probable que el administrador de archivos os oculte algunas extensiones de archivo, pues es la configuración por defecto de Windows. En ese caso, es posible que subáis algún archivo con doble extensión porque la extensión que ya tiene no la veréis y le añadiréis otra, (p. ej., `e1.py.py`). Como consecuencia, el nombre no será correcto y se penalizará la nota con un punto. Para evitar este error, configurad el administrador de archivos de Windows para que muestre las extensiones de todos los archivos.

© Universitat Jaume I 2025

4. Calificación del entregable

El entregable se calificará sobre 10 utilizando diez nuevas instancias, de tamaño similar al de las instancias públicas. Las nuevas instancias se publicarán junto con las calificaciones para que podáis probarlas con vuestro programa.

Una instancia se considerará superada si se cumplen estas dos condiciones:

- La solución obtenida es óptima.
- El tiempo acumulado de ejecución de vuestras funciones `read_data` y `process` está por debajo del tiempo máximo de ejecución: un segundo.

Con los laberintos más pequeños es posible obtener la solución óptima incluso con un algoritmo muy ineficiente, con los medianos se necesita un algoritmo más eficiente. Con los laberintos más grandes necesitaréis un algoritmo con un coste lineal con el número de celdas.

Importante: podéis detectar y corregir muchos errores en vuestro programa utilizando las instancias públicas, aunque superar las instancias públicas no garantiza superar las privadas, sobre todo si la implementación se ha "ajustado" específicamente para superar las públicas.

4.1. Medición de tiempos de ejecución

Para poder medir tiempos de forma consistente necesitamos utilizar un ordenador de referencia. El equipo elegido es `rei.dlsi.uji.es`. Las instancias se han generado para que cada una pueda resolverse en este equipo en menos de un segundo, con un algoritmo lineal con el número de vértices del grafo.

Para obtener el tiempo de ejecución sobre el ordenador de referencia, podéis subir vuestro programa al sitio web:

`https://rei.dlsi.uji.es/algoritmia`

donde un servicio lo ejecutará para todas las instancias y mostrará el tiempo total de ejecución de vuestras funciones `read_data` y `process` para cada una de ellas.

Algunas consideraciones sobre el sitio web:

- Para conectaros desde fuera de s la UJI, necesitaréis utilizar la VPN de la universidad. Si no sabéis cómo configurarla, podéis encontrar toda la información en

  `https://connectaxarxa.uji.es/es/index.html#vpn`

  Importante: Tened en cuenta que al activar la VPN recibiréis un correo con un enlace que debéis pulsar para que la VPN no se cierre automáticamente a los 60 segundos (es el doble factor).
- El equipo no tiene IP pública por lo que su certificado no tiene una firma válida. Tendréis que aceptar una excepción de seguridad para poder acceder al sitio web.
- El servicio sólo muestra el tiempo de ejecución conjunto de vuestras funciones `read_data` y `process`, no comprueba que la solución que obtiene vuestro programa sea correcta.

© Universitat Jaume I 2025

4.2. Errores graves en el entregable

Se calificarán con un cero los entregables:

- Que no lean las instancias desde la entrada estándar, tal y como se indica en el apartado 1.2. Si no sabéis hacerlo, mirad el código que escribisteis en la primera sesión de prácticas. Si no lo entendéis, preguntad al profesor.
- Que hayan modificado el programa principal de `e1.py` o los tipos de cualquiera de las funciones que utiliza.

También se penalizará que el contenido de la salida estándar no respete el formato que se especifica en el apartado 1.3. Una salida errónea puede tener dos causas:

- Una implementación de `show_result` que no respete el formato especificado. Se penalizará la nota con dos puntos. Dado que en este entregable se os proporciona la función `show_result` ya implementada, este error no debería suceder.
- Algún `print` de depuración olvidado en el código. Se penalizará la nota con un punto. Para que no os pase, podéis mostrarlos por la salida de error estándar, utilizando el parámetro `file=sys.stderr` de la función `print`:

```
print("Esto se muestra en la salida de error estándar", file=sys.stderr)
```

Por último, también se penalizará con un punto cada restricción incumplida en la entrega al aula virtual (ver el apartado 3). Revisad la entrega con detenimiento antes de subirla al aula virtual.

4.3. Penalización por copia

En caso de detectarse una copia, se aplicará la normativa de la universidad: la calificación de la evaluación continua (60 % de la nota final) será de cero en la primera convocatoria para todos los involucrados.

© Universitat Jaume I 2025