EI1022/MT1022 - Algoritmia

Entregable 3 - Un reparto justo

Fecha de entrega: miércoles 19 de noviembre de 2025

Índice

1. El problema

Un grupo de tres mineros ha estado trabajando toda la temporada en su explotación de esmeraldas. Al terminar la temporada han encontrado N esmeraldas, que han decidido repartirse antes de volver a sus casas. El valor de cada esmeralda está almacenado en un vector $V = (v_0, v_1, \ldots, v_{N-1})$. Podemos identificar cada esmeralda por su índice en el vector.

Los mineros quieren repartírselas de forma que el valor total de las esmeraldas que se lleve cada uno sea exactamente el mismo. Eso implica que puede que queden esmeraldas sin repartir (no se pierden, se guardan para la siguiente temporada). Debéis encontrar un reparto óptimo que minimice el valor de las esmeraldas que se quedan sin repartir.

Tened en cuenta que puede haber instancias en las que sea posible repartir todas las esmeraldas y otras en las que no se pueda repartir ninguna, un ejemplo trivial de este último caso es cuando hay tres esmeraldas y cada una tiene un valor distinto.

2. Implementación

Implementad un programa de línea de órdenes, `e3.py`, que reciba por la entrada estándar el vector V con los valores de las N esmeraldas en el formato que se especifica en el apartado 2.1. Como resultado de su ejecución el programa mostrará por la salida estándar el reparto obtenido en el formato que se especifica en el apartado 2.2.

Obligatoriamente, vuestra implementación debe resolver el problema utilizando explícitamente el esquema de búsqueda con retroceso de la biblioteca `algoritmia` (`bt_scheme.py`).

© Universitat Jaume I 2025

2.1. Formato de la entrada

Una instancia es un archivo de texto con N líneas de texto, cada una con un entero que representa el valor de una esmeralda y que sigue el orden de los elementos en el vector V.

Como ejemplo, el archivo `e3-01-15.i` contiene la instancia $V = (15, 13, 5, 22, 15, 10)$ y este es su contenido:

```
15
13
5
22
15
10
```

2.2. Formato de la salida

El programa deberá mostrar cinco líneas de texto por la salida estándar:

- Una primera línea con un entero indicando el valor total de las esmeraldas que se lleva cada minero. Por ejemplo, si en el reparto cada minero se lleva esmeraldas con un valor total de 15, la línea sería:

  ```
  15
  ```

- Una línea por cada minero, con los índices de las esmeraldas que se le asignan en el reparto, separados por un espacio en blanco. Supongamos que un minero recibe la esmeralda de índice 0, otro las esmeraldas de índice 2 y 5, y el tercero la esmeralda de índice 4, las tres líneas de la salida serían:

  ```
  0
  2 5
  4
  ```

- Una última línea con los índices de las esmeraldas que quedan sin repartir, separados por un espacio en blanco. Supongamos que las esmeraldas de índice 1 y 3 se quedan sin repartir, el contenido de la línea sería:

  ```
  1 3
  ```

Los archivos `e3-*.o`, con los resultados de las instancias de prueba, tienen este formato. Las cinco líneas de texto que se han utilizado como ejemplo, son el contenido del archivo `e3-01-15.o`.

Como ejemplo de una instancia en la que no se puede repartir nada, tenemos el archivo `e3-02-0.i` que contiene la instancia $V = (90, 10, 20, 200, 100, 50)$. Su resultado se encuentra en el archivo `e3-02-0.o` que tiene este contenido:

```
0



0 1 2 3 4 5
```

donde las tres líneas con los índices de las esmeraldas de cada minero están en blanco y la línea con los índices de las esmeraldas no repartidas los contiene todos.

© Universitat Jaume I 2025

Es posible que existan varios repartos óptimos. También es posible que, con el mismo reparto, el orden de los mineros sea diferente o que lo sea el orden de los índices de las esmeraldas. Todos estos casos se considerarán válidos en la calificación del entregable.

3. 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 resultados.
- `e3.py`: el programa que debéis utilizar como punto de partida y que hay que entregar.

Los siguientes apartados explican con detalle estos contenidos, pero ya podéis crear un nuevo proyecto PyCharm y añadirle el archivo `e3.py` y la carpeta que contiene el archivo `public.zip`.

3.1. Archivos de prueba: `public.zip`

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

`e3-<num>-<value>.<ext>`

donde:

- `<num>`: El número de instancia, entre 1 y 10.
- `<value>`: El valor total de las esmeraldas que se lleva cada minero en el reparto óptimo.
- `<ext>`: La extensión del archivo. Indica el contenido: '`i`' si contiene una instancia u '`o`' si contiene un resultado.

Es posible que algunas instancias tengan más de una solución válida, por lo que puede que algunas de vuestras soluciones no coincidan con las que contiene la carpeta. En cualquier caso, las soluciones deben coincidir en el valor total de las esmeraldas que se lleva cada minero.

3.2. Archivo principal: `e3.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 `e3.py` y las funciones que debéis implementar.

© Universitat Jaume I 2025

El programa define el tipo `Data` para las instancias y el tipo `Result` para los resultados:

```
type Data = tuple[int, ...]
type Result = tuple[int, tuple[int, ...]]
```

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 de enteros con los valores de las esmeraldas. El índice en el vector identifica cada esmeralda.
- `process(data: Data) -> Result:`
  - Entrada: La tupla que devuelve `read_data`.
  - Salida: Una tupla `(valor, tupla_reparto)`, donde:
    - `valor`: El valor total de las esmeraldas que recibe cada minero.
    - `tupla_reparto`: una tupla en la que el elemento en la posición i-ésima indica qué hacemos con la esmeralda i-ésima. Cada elemento puede tomar un valor del conjunto $\{-1, 0, 1, 2\}$, donde el valor -1 indica que la esmeralda no participa en el reparto y los otros tres valores representan la asignación de la esmeralda a uno de los tres mineros.

Cómo ejecutar vuestro programa `e3.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 `e3.py` y `public`.

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

- Linux y macOS: `python3 e3.py < public/e3-01-15.i`
- Windows (CMD): `python e3.py < public\e3-01-15.i`
- Windows (PowerShell): `Get-Content public\e3-01-15.i | python e3.py`

4. Entrega en el aula virtual

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

- `e3.py`: El archivo con el código del entregable.
- `alxxxxxx_alyyyyyy.txt`: Un archivo de texto que deberá contener el nombre de cada miembro de la pareja (un nombre por línea). Evidentemente, en el nombre del archivo tenéis que reemplazar `alxxxxxx` y `alyyyyyy` por los correspondientes a los dos miembros de la pareja. Si la entrega es individual, el archivo de texto deberá llamarse `alxxxxxx.txt`, reemplazando `alxxxxxx` por el que corresponda.

© Universitat Jaume I 2025

Vuestra entrega debe cumplir las siguientes restricciones:

- Sólo debéis subir los dos archivos solicitados, nada más.
- Los nombres de los dos archivos 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 los archivos 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., `e3.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.

5. 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 por instancia.

Con las instancias más pequeñas es posible obtener la solución óptima incluso con un algoritmo muy ineficiente, con las medianas se necesita un algoritmo más eficiente. Para las más grandes necesitaréis reducir el tamaño del espacio de estados sin disparar el coste.

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.

5.1. 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 una implementación de `state` adecuada.

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.

© Universitat Jaume I 2025

Algunas consideraciones sobre el sitio web:

- Para conectaros desde fuera de 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 de la UJI con un enlace que debéis pulsar para que la VPN no se cierre automáticamente a los 60 segundos (es el doble factor).
- Dado que el equipo no está conectado directamente a Internet, su certificado es autofirmado. Tendréis que aceptar una excepción de seguridad en el navegador para poder acceder al sitio web.
- El servicio sólo mide el tiempo de ejecución conjunto de vuestras funciones `read_data` y `process`, no comprueba que la solución que obtiene vuestro programa sea correcta.

5.2. 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 2.1. 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 `e3.py` o los tipos de cualquiera de las funciones que utiliza.
- Que no resuelvan el problema utilizando explícitamente el esquema de búsqueda con retroceso de la biblioteca `algoritmia` (`bt_scheme.py`).

También se penalizará que el contenido de la salida estándar no respete el formato que se especifica en el apartado 2.2. 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 4). Revisad la entrega con detenimiento antes de subirla al aula virtual.

5.3. 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