EI1022/MT1022 - Algoritmia

Entregable 2 - La inversión en bolsa

Fecha de entrega: miércoles 5 de noviembre de 2025

Índice

1. El problema

Un inversor quiere comprar y vender acciones de una empresa. Tiene un modelo muy sofisticado que le permite prever el precio de las acciones a lo largo de un periodo de tiempo. Por distintas razones, solo puede hacer una compra y una venta y además tiene un máximo de días que pueden transcurrir entre la compra y la venta. El objetivo de tu programa será averiguar las fechas de compra y venta que maximicen el beneficio del inversor.

Para ello, deberás utilizar la estrategia de divide y vencerás. Además, para que tu solución alcance la máxima nota, su coste temporal deberá ser $\mathcal{O}(n \log n)$, donde n es el número de días.

Más formalmente, tenemos un vector C que contiene n cotizaciones, que son números enteros positivos, indexadas de 0 a $n-1$ y por otro lado la máxima distancia entre la fecha de venta y la de compra, que es un entero positivo m. El programa debe encontrar dos índices i y j, correspondientes a las fechas de compra y venta, respectivamente, de modo que se cumplan tres condiciones. Primera, el valor de j para un i dado es un valor tal que $i < j \leq \text{mín}(i+m, n-1)$ y que maximiza la diferencia $C[j] - C[i]$. Segunda, i es el valor entre 0 y $n-2$ que maximiza la diferencia con el correspondiente j. Tercera, el valor de la diferencia debe ser estrictamente positivo. Si encuentra esos valores, el programa escribirá en una línea tres enteros: el beneficio máximo, el índice del día de compra y el del día de venta. Si no es posible obtener un beneficio mayor que cero, el programa escribirá "`NO PROFIT`".

Veamos un ejemplo. El fichero `e2-01-10-8.i` contiene una instancia en la que $m = 4$ y las cotizaciones son:

0	1	2	3	4	5	6	7	8	9
1	5	6	3	2	10	5	4	8	6

El máximo beneficio se obtiene al comprar el día 4 (cotización 2) y vender el día 5 (cotización 10). Observa que el beneficio comprando el día 0 (cotización 1) y vendiendo el día 5 (cotización 10) es mayor, pero no es posible porque serían cinco días de diferencia, por encima del máximo de cuatro.

Por otro lado, el fichero `e2-02-10-X.i` contiene una instancia en la que $m = 5$ y las cotizaciones son:

0	1	2	3	4	5	6	7	8	9
20	19	18	17	16	15	14	13	12	11

En este caso, no es posible obtener ningún beneficio.

2. Implementación

Tenéis que implementar un programa de línea de órdenes, `e2.py`, que lea la instancia desde la entrada estándar y muestre el resultado por la salida estándar. Los formatos de entrada y salida se especifican en los dos apartados siguientes.

Para poder optar a la máxima nota, el coste temporal del programa deberá ser $\mathcal{O}(n \log n)$, donde n es el número de cotizaciones. Para ello, deberéis utilizar la estrategia de divide y vencerás. La solución del problema "subvector de suma máxima" os puede servir de guía para diseñar el algoritmo.

2.1. Formato de la entrada

Una instancia es un archivo de texto que contiene en su primera línea la máxima distancia entre el día de compra y el de venta y en las líneas siguientes las cotizaciones de cada día, una por línea.

2.2. Formato de la salida

La salida del programa se mostrará por la salida estándar y tiene dos posibilidades:

- Si la instancia no tiene solución, se mostrará la cadena "`NO SOLUTION`", todo en mayúsculas.
- Si la instancia tiene solución, se mostrará una línea con tres enteros separados por blancos:
  - El beneficio máximo.
  - El índice del día de compra.
  - El índice del día de venta.

Los índices de los días comienzan en cero. En caso de haber más de una solución, se aceptará cualquiera de ellas.

3. Ficheros 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 instancias de prueba y sus soluciones.
- `e2.py`: el programa que debéis de utilizar como punto de partida y que hay que entregar.

Los siguientes apartados explican con detalle estos ficheros.

3.1. Ficheros de prueba (public.zip)

Al descomprimir el fichero public.zip obtendréis la carpeta public, que contiene las instancias de prueba y sus soluciones, los ficheros e2-*.i y e2-*.o, respectivamente. El nombre de las instancias sigue el patrón e2-<num_ins>-<tamaño>-<beneficio>.i, donde:

- <num_ins>: el número de instancia, un número entre 1 y 10.
- <tamaño>: el tamaño de la instancia, esto es, el número de días.
- <beneficio>: el beneficio máximo de la instancia o X si no tiene solución.

3.2. Fichero principal (e2.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 los parámetros y el tipo devuelto) 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.

Veamos la estructura de e2.py y las funciones que debéis implementar.
En el programa se definen los tipos Data:

```
type Data = tuple[int, list[int]]
```

y Result:

```
type Result = tuple[int, int, int] | None
```

En el caso de Data, el primer elemento de la tupla es la máxima distancia entre el día de compra y el de venta y el segundo elemento es la lista de cotizaciones. En el caso de Result, si la instancia tiene solución, la tupla contiene el beneficio máximo, el día de compra y el día de venta. Si la instancia no tiene solución, Result es None.
El programa principal utiliza la estructura de tres funciones vista en clase:

```
if __name__ == "__main__":
    data0 = read_data(sys.stdin)
    result0 = process(data0)
    show_result(result0)
```

Por lo tanto, el programa contiene las funciones read_data, process y show_result:

- read_data: recibe f, un fichero (no su nombre), y devuelve la distancia máxima y la lista de cotizaciones. Su firma es:

  ```
  def read_data(f: TextIO) -> Data:
  ```

- process: recibe el data leído por read_data y devuelve el máximo beneficio y los índices de compra y venta o None si no hay solución. Su firma es:

```
def process(data: Data) -> Result:
```

- `show_result`: recibe `result`, la tupla devuelta por `process` o el valor `None`. Si ha recibido una tupla, la muestra por la salida estándar como tres enteros separados por blancos. Si ha recibido `None`, muestra la cadena "`NO SOLUTION`". Su firma es:

  ```
  def show_result(result: Result):
  ```

La función `show_result` ya está implementada.

Cómo ejecutar vuestro programa `e2.py`

Dado que el programa lee de la entrada estándar, no resulta cómodo lanzarlo desde PyCharm. Se recomienda ejecutarlo desde el terminal que incorpora el propio PyCharm (el botón está en la parte inferior izquierda). Una vez abierto, utilizad la orden `cd` para ir al directorio donde estén `e2.py` y `public`.

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

- Linux y MacOS: `python3 e2.py < public/e2-01-10.i`
- Windows (CMD): `python e2.py < public\e2-01-10.i`
- Windows (PowerShell): `GetContent public\e2-01-10.i | python e2.py`

4. 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:

- `e2.py`: El fichero con el código del entregable.
- El fichero con la identificación del grupo. Para parejas, el fichero se llamará `alxxxxxx_alyyyyyy.txt` y contendrá dos líneas, cada una con el nombre completo de un miembro de la pareja. En las entregas individuales, el fichero se llamará `alxxxxxx.txt` y contendrá una línea con el nombre completo de la persona que realiza la entrega. En ambos casos, `alxxxxxx` y `alyyyyyy` son los `al` de los miembros de la pareja. Por ejemplo, si la pareja está formada por Elena Nito (al000000) y Aitor Menta (al111111), el fichero se llamará `al000000_al111111.txt` y su contenido será:

  ```
  Elena Nito
  Aitor Menta
  ```

  Si Elena entrega individualemente, el fichero se llamará `al000000.txt` y su contenido será:

  ```
  Elena Nito
  ```

Vuestra entrega debe cumplir las siguientes restricciones:

- Solo debéis subir los dos ficheros solicitados.
- 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 los ficheros al aula virtual

Si usáis Windows, es probable que el administrador de archivos os oculte algunas extensiones de archivo, pues, incomprensiblemente, es la configuración por defecto de Windows. En ese caso, es posible que subáis los archivos con doble extensión (`e2.py.py` y `alxxxxxx_alyyyyyy.txt.txt`) porque la extensión que tiene el archivo no la veréis y le añadiréis otra. Como consecuencia, los nombres de los archivos no serán los correctos 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 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 o se anuncia correctamente que no hay solución.
- El tiempo acumulado de las funciones `read_data` y `process` no ha superado el tiempo máximo de ejecución: un segundo.

Con los problemas 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 problemas más grandes necesitaréis un algoritmo con un coste $\mathcal{O}(n \log n)$, siendo n el número de días.

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 un ordenador de referencia. Para este entregable será `rei.dlsi.uji.es`.

Para obtener el tiempo de ejecución sobre el ordenador de referencia, podéis subir vuestro programa a

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

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

Algunas consideraciones sobre la página web:

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

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

  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 tras unos pocos minutos (es el doble factor).
- El servidor `rei.dlsi.uji.es` no tiene IP pública por lo que su certificado es autofirmado. Tendréis que aceptar una excepción de seguridad para poder acceder a la página.
- El servicio sólo muestra el tiempo de ejecución de vuestra función `process`, no comprueba que la solución que obtiene vuestro programa sea correcta, eso debéis comprobarlo vosotros.

5.2. Errores graves en el entregable

Se calificarán con un cero:

- Los entregables que hayan modificado el programa principal de `e2.py` o los tipos de cualquiera de las funciones que utiliza.
- Los entregables que no utilicen la estrategia de divide y vencerás para resolver el problema.

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 modificación de `show_result` que haga que no se respete el formato especificado. Se penalizará la nota con dos puntos.
- Algún `print` de depuración olvidado en el código. Se penalizará la nota con un punto. Para que no os pase, podéis utilizar la salida de error estándar, por ejemplo:

  ```
  print("Este mensaje se mostrará 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 (los dos miembros de la pareja).

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.