ANÁLISIS Y DISEÑO DE ALGORITMOS

Programación Dinámica

Práctica 6 de laboratorio

Esta práctica ocupa dos entregas:

Entrega 1: Versiones recursivas (ingenua y memoización) y la gestión de argumentos.
Entrega 2: Práctica completa.

Fechas de entrega: Respectivamente, hasta los días 29 de marzo y 5 de abril, 23:55h.

Suma máxima limitada

Un comercio ha agraciado a uno de sus clientes con un premio. Consiste en el obsequio de todos los artículos que elija de su local comercial siempre que no repita ninguno y que el precio de venta de la selección no supere una cierta cantidad $T \in \mathbb{N}$. Si no cumple estas dos condiciones, pierde el premio. El cliente dispondrá de una relación con los n productos que podría seleccionar junto con sus precios de venta ($p_i \in \mathbb{N} - \{0\},\ \forall\ i$). Se desea conocer cuál es la mejor selección teniendo en cuenta que lo único que interesa es su valor total.

Por ejemplo, si se trata de decidir sobre $n = 8$ artículos cuyos precios son {11, 13, 19, 23, 29, 31, 41, 43} y el umbral es $T = 99$, la mejor selección posible arroja un valor acumulado igual a 98 (corresponde, por ejemplo, a la selección de los cuatro objetos de valores 13, 19, 23 y 43[1]).

En esta práctica se pide aplicar el método *divide y vencerás* y su extensión a *programación dinámica* para obtener el precio total de los artículos que componen la mejor selección posible, y, a partir de este dato, los artículos que han sido seleccionados.

Para resolver este ejercicio se debe implementar los siguientes algoritmos:[2]

1. Recursivo sin almacén (ineficiente —también llamada *versión ingenua*—)
2. Recursivo con almacén (memoización)
3. Iterativo con almacén que hace uso de una tabla para almacenar los resultados intermedios.
4. Iterativo con almacén con complejidad espacial mejorada.

Por último deberá mostrarse los artículos que han sido seleccionados.

[1] Podría ocurrir, como de hecho sucede en este ejemplo, que la solución no sea única, ni en los objetos seleccionados ni en el número de estos. En tal caso bastará con mostrar una alternativa cualquiera de entre todas las posibles.

[2] Como es lógico, la solución de los algoritmos que muestran el precio total deben coincidir.

- **Nombre del programa, opciones y sintaxis de la orden.**

  El programa a realizar se debe llamar **maxsum**. La orden tendrá la siguiente sintaxis:

  maxsum [-t] [--ignore_naive] -f `fichero_entrada`

  En el siguiente apartado se describe el significado de las opciones.

- **Salida del programa y descripción de las opciones:**

  En todas las formas posibles de utilizar la mencionada orden, la salida del programa (véase los ejemplos a continuación) será, en primer lugar, la solución obtenida mediante los cuatro algoritmos anteriormente citados: recursivo sin almacén (versión ingenua), memoización, iterativo con tabla e iterativo con complejidad espacial mejorada. (debe seguirse este orden). No obstante, si se incorpora a la orden la opción [--ignore_naive] se deberá excluir de la salida la solución recursiva sin almacén, por su elevado coste computacional (evidentemente el programa tampoco deberá hacer la llamada a esta función); en este caso la solución de los otros tres algoritmos sí que deberá mostrarse. En segundo lugar, deberá mostrarse la relación de objetos seleccionados y a continuación el valor total calculado a partir de estos objetos.[3] En cuanto al resto de opciones:

  - Si se hace uso de la opción [-t] se mostrará además la matriz obtenida en la versión iterativa (donde se almacenan los resultados intermedios).
  - Las opciones no son excluyentes entre sí, ninguna de ellas. Además pueden aparecer en cualquier orden.
  - En cualquiera de las formas posibles de utilizar la orden, si se incorpora la opción [--ignore_naive] se debe excluir (únicamente) la solución recursiva sin almacén.
  - La opción -f, la única de uso obligado, se utiliza para suministrar el nombre del fichero donde está el problema a resolver (véase siguiente apartado)
  - En el caso de que se haga uso de la orden con una sintaxis distinta a la descrita se emitirá un mensaje de error advirtiéndolo y a continuación se mostrará la sintaxis correcta. Considerar en este caso únicamente las opciones inexistentes; la duplicidad de opciones puede ser ignorada y tratada por tanto como si se hubiera escrito solo una vez. No es necesario indicar todos los errores sintácticos que pueda contener la orden, basta con hacerlo solo con el primero que se detecte.

- **Entrada al programa**

  El problema a resolver se suministrará codificado en un fichero de texto cuyo nombre se recogerá a través de la opción -f. Su formato y contenido será:

  - Línea 1 del fichero: Valores n y T, en este orden.
  - Línea 2: Valor de los n objetos: Una lista de n números naturales.

  Por tanto, el fichero contendrá 2 líneas que finalizarán con un salto de línea, salvo en todo caso, la última línea. El carácter separador entre números siempre será el espacio en blanco.

  A través de *Moodle* se puede descargar un archivo comprimido con varios ejemplos y sus soluciones; entre ellos está `0.problem` y `1.problem` utilizados a continuación para describir el formato de la salida.

[3] Este valor coincidirá con el obtenido en cualquiera de los algoritmos pedidos pero servirá como comprobación de que la selección es compatible con la solución mostrada con dichos algoritmos.

- **Formato de salida. Ejemplos de ejecución.**

  Sea el fichero 1.problem; se trata del problema utilizado como ejemplo anteriormente.

  A continuación se muestra posibles formas de utilizar en la terminal la orden descrita y la salida que el programa debe mostrar. Es imprescindible ceñirse al formato y texto de salida mostrado, incluso en lo que se refiere a los saltos de línea o carácter separador, que en todos los casos es el espacio en blanco (aunque no hay problema si hay más de uno). La última línea debe terminar con un salto de línea (y solo uno). **Debe respetarse los textos y formatos mostrados y en ningún caso debe añadirse texto o valores adicionales.**

```
 $maxsum -f 1.problem
Naive: 98
Memoization: 98
Iterative (table): 98
Iterative (vector): 98
Selection: 11 13 31 43
Selection value: 98
$
$maxsum -f 1.problem --ignore_naive
Memoization: 98
Iterative (table): 98
Iterative (vector): 98
Selection: 11 13 31 43
Selection value: 98
$
$maxsum -t --ignore_naive -f 1.problem
Memoization: 98
Iterative (table): 98
Iterative (vector): 98
Selection: 11 13 31 43
Selection value: 98
Iterative table:
"VER LA TABLA EN LAS SOLUCIONES
PUBLICADAS A TRAVÉS de MOODLE"
$
```

```
$maxsum -f -t
ERROR: can't open file: -t.
Usage:
$maxsum  [-t] [--ignore_naive] -f file
$
$maxsum -f 0.problem -t -a
ERROR: unknown option -a.
Usage:
$maxsum  [-t] [--ignore_naive] -f file
$
$maxsum -f mifichero -t
ERROR: can't open file: mifichero.
Usage:
$maxsum  [-t] [--ignore_naive] -f file
$
```

Normas para la entrega.

> **ATENCIÓN: Estas normas son de obligado cumplimiento para que esta práctica sea evaluada.**

1. Se debe entregar únicamente el código fuente, con nombre `maxsum.cc`, y el fichero `makefile`. No hay que entregar nada más; en ningún caso se entregarán ficheros de test.
2. Es imprescindible que no presente errores ni de compilación ni de interpretación (según corresponda), en los ordenadores del laboratorio asignado.[4] Se tratará de evitar también cualquier tipo de *warning*.
3. Todos los ficheros que se entregan deben contener el nombre del autor y su DNI (o NIE) en su primera línea (entre comentarios apropiados según el tipo de archivo).
4. Se comprimirán en un archivo `.tar.gz` cuyo nombre será el DNI del alumno, compuesto de 8 dígitos y una letra (o NIE, compuesto de una letra seguida de 7 dígitos y otra letra). Por ejemplo: 12345678A.tar.gz o X1234567A.tar.gz. **Solo se admite este formato de compresión y solo es válido esta forma de nombrar el archivo.**
5. En el archivo comprimido **no debe existir subcarpetas**, es decir, al extraer sus archivos estos deben quedar guardados en la misma carpeta donde está el archivo que los contiene.
6. La práctica hay que subirla a *Moodle* respetando las fechas expuestas en el encabezado de este enunciado.
7. En la primera entrega solo es necesario que estén implementadas ambas versiones recursivas (ingenua y memoización). Sin embargo, la gestión de opciones del programa debe estar hecha en su totalidad. El resultado que se debe mostrar para los casos aún sin hacer es "¿?". Por ejemplo:

```
$maxsum -t -f 1.problem
Naive: 98
Memoization: 98
Iterative (table): ¿?
Iterative (vector): ¿?
Selection: ¿?
Selection value: ¿?
Iterative table:
¿?
```

[4] Si trabajas con tu propio ordenador o con otro sistema operativo asegúrate de que este requisito se cumple.