ANÁLISIS Y DISEÑO DE ALGORITMOS

Ramificación y poda

Práctica final

Entrega: hasta el 31 de mayo, 23:55h.
A través de Moodle

Suma máxima limitada (y IV)

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 nueva práctica se pide aplicar el método ramificación y poda para obtener la mejor selección posible de objetos.

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

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

   `maxsum-bb -f` *fichero_entrada*

   El fichero con el problema a a resolver se suministra a través de la opción `-f`. En el caso de que no se suministre el fichero o se suministre uno inexistente se advertirá con un mensaje de error. No es necesario controlar posibles errores en el contenido del fichero de entrada ya que siempre se ajustará fielmente al formato establecido (véase siguiente apartado).

   Si se hace 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

[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.

correcta.

2. **Entrada de datos al programa. Formato del fichero de entrada.**

   De manera idéntica a la práctica anterior, 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.

3. **Salida del programa, formato de salida y ejemplo de ejecución.**

   El programa mostrará los siguientes resultados, uno por cada línea y sin incorporar texto alguno salvo el propio valor numérico (véase los ejemplos de ejecución para obtener información acerca de la sintaxis de salida):

   a) En primer lugar, la suma de los valores de los objetos seleccionados (en esta práctica no debe mostrarse la selección de objetos).

   b) En segundo lugar, una cuantificación de los distintos tipos de nodos encontrados durante el proceso de búsqueda de la solución óptima. Se mostrará los siguientes valores:

      1) Número de nodos que han sido expandidos (es decir, aquellos de los que se exploran sus hijos),
      2) Número de nodos añadidos a la lista de nodos vivos,
      3) Número de nodos descartados por no ser factibles,
      4) Número de nodos descartados por no ser prometedores,
      5) Número de nodos que fueron prometedores pero que finalmente se descartaron,
      6) Número de nodos completados (se entiende que un nodo está completado cuando se trata de una hoja del árbol de estados posibles),
      7) Número de veces que la mejor solución hasta el momento se ha actualizado a partir de un nodo completado,
      8) Número de veces que la mejor solución hasta el momento se ha actualizado a partir de la cota pesimista de un nodo sin completar.

   c) En tercer y último lugar, el tiempo de CPU, expresado en milisegundos, que ha consumido el algoritmo de ramificación y poda para resolver el problema.[2]

   Cada uno de los tres tipos de resultados enumerados se mostrará en una línea distinta que terminará con un único salto de línea y nada más. Por

[2]Para obtener el tiempo CPU debe utilizarse la función *clock()* —véase la práctica 1— .

lo tanto, los valores que corresponden a la estadística sobre los nodos se mostrarán en una única línea y separados mediante un espacio en blanco. Debe tenerse en cuenta que esos valores dependen de cada implementación particular.

De esta manera, un ejemplo de salida del programa es el siguiente:

```
$./maxsum-bb -f 1k_bb.problem
7183807
1709 2192 1195 32 1 5 4 0
39.368
```

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 sólo uno). **Debe respetarse el formato mostrado y en ningún caso debe añadirse texto o valores adicionales.**

Para comprobar que tu programa emite la salida con un formato correcto, puedes utilizar un filtro desarrollado a tal efecto. El archivo fuente (`filter.cc`) se ha publicado en la sección de entregas de prácticas de Moodle. Para usarlo debes primero compilarlo utilizando elprocedimiento habitual (se deja como ejercicio la creación de un fichero *makefile*) y después suministrarle la salida de tu programa mediante un *pipe*, como por ejemplo:

```
./maxsum-bb -f 1k_bb.problem | filter
```

Deberá tratarse también posibles errores en los argumentos de la orden, no suministrar el fichero de entrada o su inexistencia, tal y como se hizo en las prácticas anteriores.

4. **Documentación del trabajo realizado.**
Se debe entregar una **memoria**, en formato PDF, con nombre `memoria.pdf`, que describa las estructuras de datos empleadas así como la/s estrategia/s de búsqueda, los mecanismos de poda y las cotas optimistas y pesimistas utilizadas.
Incluirá también un análisis del comportamiento de distintas estrategias de búsqueda en conjunción con distintos mecanismos de poda. Para ello, se realizará un cuadro comparativo que contenga, para cada caso estudiado, el número de nodos que se añaden a la lista de nodos vivos y el tiempo de respuesta del algoritmo (según se ha explicado en clase de teoría).

   La primera línea de la memoria contendrá el nombre del autor y su DNI. El resto se estructurará en seis apartados de la siguiente manera:

   1. **Estructuras de datos**

      1.1 **Nodo**
      En este apartado se describirá brevemente el contenido del nodo, es decir, las variables que lo componen y su cometido.

1.2 **Lista de nodos vivos**
Breve descripción de la estructura de datos utilizada y si procede, qué criterios se han analizado para extraer el nodo más prometedor (El programa debe utilizar el que se considere más rápido pero todos los demás se mencionarán también en este apartado).

2. **Mecanismos de poda**

2.1. **Poda de nodos no factibles**
Se deberá indicar cómo se descartan los nodos que no son factibles; si no procede este tipo de poda debe indicarse el motivo.

2.2. **Poda de nodos no prometedores**
Se deberá indicar cómo se decide que un nodo no es prometedor; si no procede debe indicarse el motivo.

3. **Cotas pesimistas y optimistas**
Se describirá en qué consisten dichas cotas, distinguiéndose, en el caso de la pesimista, entre el nodo inicial y los demás nodos.

3.1 **Cota pesimista inicial (inicialización).**

3.2 **Cota pesimista del resto de nodos.**

3.3 **Cota optimista.**

4. **Otros medios empleados para acelerar la búsqueda**
En este apartado se citará cualquier otro mecanismo utilizado con el objetivo de acelerar la búsqueda o cualquier otro aspecto que se quiera reflejar en la memoria y que no se adapta a ninguno de sus apartados.

5. **Estudio comparativo de distintas estrategias de búsqueda**
En este apartado se incluirá el cuadro comparativo mencionado anteriormente junto con una breve discusión de los resultados obtenidos.

6. **Tiempos de ejecución**.
Se mostrará en este apartado el tiempo de proceso requerido para resolver cada uno de los ficheros de test publicados para esta práctica. Si alguno de los ficheros no ha podido ser resuelto por el programa realizado se escribirá '¿?'. Por ejemplo:

- Fichero 1_bb.problem: 0.044 ms.
- Fichero 10_bb.problem: 0.026 ms.
- Fichero 20_bb.problem: 1.187 ms.
- Fichero 30_bb.problem: 1673.07 ms.
- Fichero 40_bb.problem: ¿?
- Fichero 45_bb.problem: ¿?
- Fichero 50_bb.problem: 0.042 ms.
- Fichero 100_bb.problem: 0.757 ms.
- Fichero 200_bb.problem: 0.474 ms.
- Fichero 1k_bb.problem: 1.584 ms.
- Fichero 2k_bb.problem: 2.902 ms.
- Fichero 3k_bb.problem: 1.865 ms.

IMPORTANTE: Todos los apartados (salvo el último) se complementarán con el correspondiente fragmento de código que corrobore el trabajo descrito. Si algunos de los apartados no ha sido realizado se pondrá únicamente la frase "NO IMPLEMENTADO".

De los distintos elementos que puede contener un algoritmo de ramificación y poda, sólo se evaluarán los que estén reflejados en esta memoria. Es decir, todo lo que no esté en dicho documento se entenderá que no ha sido realizado.

5. **Sobre la evaluación de esta práctica.**

   Además del uso apropiado de la estrategia, de la obtención del resultado correcto y de los mecanismos implementados para minorar la cantidad de nodos explorados, en esta práctica se tendrá en cuenta también el tiempo de CPU que se requiere para obtener la solución; por lo tanto, aunque se hayan implementado distintas estrategias de búsqueda, funciones de cota, etc, el programa que se entrega debe funcionar con la configuración que se entienda más rápida. No obstante, en la memoria se deberá hacer mención a cualquier otro trabajo realizado que, por ese motivo, no se haya incluido en la versión final del programa.

Normas para la entrega.

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

a) Se debe entregar únicamente el código fuente, con nombre `maxsum-bb.cc`, el fichero `makefile` y el fichero `memoria.pdf` . No hay que entregar nada más, en ningún caso se entregarán ficheros de test.

b) Es imprescindible que no presente errores ni de compilación ni de interpretación (según corresponda). Para su corrección, la práctica se compilará con el estándar 11 de *g++* y con sistema operativo *GNU/Linux*. Se tratará de evitar también cualquier tipo de aviso (*warning*).

c) 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).

d) 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.**

e) 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.

f) La práctica hay que subirla a *Moodle* respetando las fechas expuestas en el encabezado de este enunciado.