Tarea 5: Backtracking

Programación 3
Instituto de Computación
Facultad de Ingeniería de la Universidad de la República
Curso 2010
Versión 1.0

1. Objetivos

- Diseñar e implementar un algoritmo basado en la técnica de Backtracking.

2. Conocimientos previos

- Metodología de programación estructurada.
- Conocimientos básicos de C^*.
- Backtracking

3. Descripción del problema

Un equipo de Fórmula 1 desea minimizar el tiempo que le lleva completar una carrera. Para ello debe tener en cuenta que:

- La carrera consta de una cierta cantidad de vueltas L.
- El automóvil a utilizar tiene un tanque de combustible de capacidad C (litros).
- El auto tiene S posibles velocidades.
- Para cada velocidad i, t_i es el tiempo en segundos que requiere dar una vuelta a dicha velocidad.
- Para cada velocidad i, el automóvil realiza un consumo de combustible de c_i (litros/vuelta). Cuanto mayor sea la velocidad, mayor será el consumo de combustible.
- Antes de comenzar cada vuelta, el piloto puede decidir parar en boxes, lo que requiere un tiempo de B segundos. La parada en boxes implica siempre llenar el tanque de combustible.

- El piloto, al comenzar una vuelta, le indica al mecanismo electrónico de su auto, una de las S velocidades disponibles. Ésta se mantiene constante por el resto de la vuelta. Si para en boxes antes de dicha vuelta, se suma B al tiempo que le lleva dar la vuelta a la velocidad elegida.
- El automóvil comienza la carrera con el tanque lleno, y no puede quedarse sin combustible en ningún momento.

La solución que se busca, es una secuencia de largo L que indique el índice de la velocidad de cada una de las vueltas de la carrera, y si en esa vuelta paró o no en boxes.

3.1. Se pide

Resolver el problema planteado, implementando la siguiente función especificada en el archivo carrera.h:

carrera.h

```
1  /*
2   * Tarea 5 - Backtracking
3   *
4   * Programacion 3
5   * Instituto de Computacion
6   * Facultad de Ingenieria
7   * Universidad de la Republica
8   *
9   * Version $Rev: 313 $
10  */
11  #ifndef CARRERA_H
12  #define CARRERA_H
13  struct Par{
14          int velocidad;
15          bool boxes;
16  };
17  typedef struct Par Par;
18  /**
19  * Descripcion de parametros:
20  * L = cantidad de vueltas.
21  * C = capacidad en litros.
22  * B = tiempo que demora en parar en boxes y cargar combustible.
23  * S = cantidad de velocidades.
24  *
25  * t = Vector de tiempos/velocidades. t[i] es el tiempo que demora en dar
26          una vuelta yendo a velocidad i.
27  * c = Vector donde c[i] representa el consumo por vuelta de la velocidad i.
28  *
29  * sol = Vector solucion donde cada elemento sol[i], representa la velocidad
30  *       en cada vuelta i, y si el auto paro o no en la vuelta i.
31  * Precondiciones y postcondiciones:
32  * @Pre: L es mayor a 0.
33  * @Pre: C es mayor a 0.
34  * @Pre: B es mayor a 0.
35  * @Pre: S es mayor a 0.
36  * @Pre: t fue previamente creado y tiene largo L.
37          t[i] es mayor a 0 para todo i,  0 <= i < L
38  * @Pre: c fue previamente creado y tiene largo L.
39          c[i] es mayor a 0 para todo i, 0 <= i < L
40  * @Pre: el arreglo sol tiene largo L. La memoria para dicho arreglo fue
41  *  previamente reservada.
42  * @Post: Retorna el tiempo minimo de la carrera. Si no hay solucion retorna -1.
43  * @Post: Retorna en sol[i].velocidad la velocidad a la que debe ir en la vuelta i
44  y en sol[i].boxes retorna true si se detiene en boxes antes de la vuelta i.
45  **/
46  int minimizar_tiempo_carrera (int L, int C, int B, int S,
47                                              int *t, int *c, Par* &sol);
48  #endif /* CARRERA_H */
```

El valor de retorno de la función es el tiempo en segundos que le toma al auto completar la carrera. Si no hay ninguna solución posible, deberá retornar el valor -1.

3.2. Ejemplos

3.2.1. Ejemplo 1

$L = 7$ vueltas
$C = 37$ litros
$B = 10$ segundos
$S = 10$ velocidades
$t = \{120, 70, 34, 27, 25, 20, 19, 17, 15, 10\}$
$c = \{1, 2, 3, 4, 5, 6, 7, 9, 14, 38\}$

La solución óptima para esta instancia del problema es:
sol $= \{(7, 0), (9, 0), (9, 0), (8, 1), (8, 0), (8, 0), (8, 0)\}$
duración $= 127$ segundos
Que representa que el automóvil corrió a las velocidades:
v_7 (vuelta 1)
v_9 (vueltas 2 y 3)
v_8 (vuelta 4, 5, 6 y 7)

Además, se detuvo 10 segundos en boxes al comenzar la vuelta 4 (cuando le quedaban 2 litros de combustible).

3.2.2. Ejemplo 2

$L = 10$ vueltas
$C = 10$ litros
$B = 5$ segundos
$S = 10$ velocidades
$t = \{40, 35, 27, 24, 20, 14, 13, 12, 11, 10\}$
$c = \{15, 16, 17, 18, 19, 20, 21, 23, 25, 27\}$

En este caso no hay solución posible (el auto no puede dar ni una vuelta ya que la capacidad de combustible es menor que lo que gastaría para cualquier velocidad).

4. Lenguaje a utilizar

El lenguaje a utilizar en este trabajo será C con las siguientes extensiones:

- Operadores new y delete.
- Pasaje de parámetros por referencia (uso de &).
- Declaración de tipos como en C++ para registros y enumerados.
- Sobrecarga de funciones.
- Uso de cin y cout.
- Uso del tipo bool predefinido en C++.

5. Qué se espera

Para cada módulo de cabecera (.h) con los prototipos de las operaciones solicitadas, debe entregarse un módulo (.cpp) con la implementación de dichas operaciones. Debe respetarse estrictamente los prototipos especificados, esto es: nombre de la operación, tipo, orden y forma de pasaje de los parámetros y tipo de retorno.

Los **módulos de cabecera pueden** bajarse de la página web del curso. Estos módulos no forman parte de la entrega, y por lo tanto, **no deben ser modificados**. **Los módulos deben funcionar en el ambiente MinGW instalado en facultad**. Se espera que todos los módulos compilen **sin errores** utilizando las flags "-Wall" , "-Werror" y "-O1", se ejecuten **sin colgarse** y den los **resultados correctos**.

6. Forma de la entrega

Se deberá entregar únicamente los siguientes archivos (respetando las mayúsculas en los nombres):

- carrera.cpp

correspondiente a la implementación del sistema especificado. No se podrá entregar otros archivos que no sean estos.

La primera línea del archivo debe contener un comentario (/* ... */) con la cédula del autor, sin puntos ni dígito de verificación. Por ejemplo, si la cédula es 1.234.567-8, la primera línea de cada archivo deberá ser exactamente:

```
/* 1234567 */
```

7. Advertencia sobre el manejo de la memoria

Cuando un programa contiene errores en el manejo de la memoria, su comportamiento puede ser inestable. Esto implica que algunas veces funciona correctamente y otras no. En ciertos casos esto puede inducir a creer (erróneamente) que ciertos programas, que en realidad son incorrectos, funcionan correctamente. Este aspecto es influenciado, entre otras cosas, por el sistema operativo en el que se ejecuten los programas. Recomendamos tener sumo cuidado con este punto y testear los módulos en sistemas operativos Windows XP. CON LA VERSIÓN DE MINGW INSTALADA EN LAS SALAS DE INFORMÁTICA DE LA FACULTAD.

8. Sobre la individualidad del trabajo

El laboratorio es INDIVIDUAL. Los estudiantes pueden estudiar en grupo y consultarse mutuamente, pero NO pueden realizar en grupo las tareas de codificación, escritura, compilación y depuración del programa.

Los trabajos de laboratorio que a juicio de los docentes no sean individuales serán eliminados, con la consiguiente pérdida del curso, para todos los involucrados. Además todos los casos serán enviados a los órganos competentes de la Facultad, lo cual puede acarrear sanciones de otro carácter y gravedad para los estudiantes involucrados.

No existirán instancias en el curso para reclamos frente a la detección por parte de los docentes de trabajos de laboratorio no individuales, independientemente de las causas que pudiesen originar la no individualidad. A modo de ejemplo y sin ser exhaustivos: utilizar código realizado en cursos anteriores u otros cursos, perder el código, olvidarse del código en lugares accesibles a otros estudiantes, prestar el código o dejar que el mismo sea copiado por otros estudiantes, dejar la terminal con el usuario abierto al retirarse, enviarse código por mail, etc. Asimismo, se prohíbe el envío de código al foro del curso, dado que el mismo será considerado como una forma de compartir código y será sancionada de la manera más severa posible.

Es decir que se considera a cada estudiante RESPONSABLE DE SU TRABAJO DE LABORATORIO Y DE QUE EL MISMO SEA INDIVIDUAL. NO CONFIAR en el borrado automático del directorio pub de las salas Windows. Es decir antes de cerrar sesión borrar todos los archivos del directorio.

9. Fecha de entrega

El trabajo debe entregarse el día **martes 9 de Noviembre de 2010 antes de las 22:00 horas**. La entrega se realizará mediante un formulario que se habilitará oportunamente en la página web del curso.