Segundo Parcial de Programación 3 (27/11/2010)

Instituto de Computación, Facultad de Ingeniería

- Este parcial dura **4** horas y contiene 2 carillas. El total de puntos es **60**.
- En los enunciados llamamos C^* a la extensión de C al que se agrega el operador de pasaje por referencia &, y las sentencias *new*, *delete* y el uso de *cout* y *cin*.
- NO se puede utilizar ningún tipo de material de consulta. Salvo que se indique lo contrario podrá usarse todo lo visto en el teórico, práctico y laboratorio sin demostrarlo, indicando claramente lo que se está usando.

Se requiere:

i. Numerar todas las hojas e incluir en cada una el nombre y la cédula de identidad.
ii. Utilizar las hojas de un sólo lado y escribir con lápiz.
iii. Iniciar cada ejercicio en hoja nueva.
iv. Poner en la carátula la cantidad de hojas entregadas, y un índice indicando en qué hojas respondió cada problema.

Ejercicio 1. (25 puntos)

Un equipo de ciclismo de pista desea ***minimizar el tiempo*** que le lleva completar una carrera a un ciclista. La carrera consta de una cierta cantidad de vueltas ***L***. El ciclista puede ir a ***S*** posibles velocidades (discretas y predefinidas) identificadas por su índice. Para cada velocidad *i*, t_i es el tiempo en segundos que requiere dar una vuelta a dicha velocidad. El ciclista consume c_i (calorías/vuelta) para cada velocidad *i*. Cuanto mayor sea la velocidad, mayor será el consumo de calorías.
En el punto de comienzo de cada vuelta el ciclista puede decidir parar y alimentarse, lo que requiere un tiempo de ***B*** segundos, recargando así toda su energía que equivale a ***C*** calorías. Luego de haberse detenido o no en dicho punto, el director técnico del equipo le indica al ciclista a qué velocidad debe realizar la vuelta teniendo en cuenta la energía que éste dispone. Luego de esto el ciclista comenzará la vuelta y mantendrá la velocidad indicada de forma constante por el resto de la misma.
El ciclista comienza la carrera con toda la energía, y **no puede quedarse sin calorías** en ningún momento.
Para ninguna velocidad se consume más calorías que C.
La solución que se busca es la forma de minimizar el tiempo total. Dicha solución **será expresada** como una secuencia que indique para cada vuelta: el índice de la velocidad a la que se realiza la vuelta, y cuanta energía se tiene **al comenzar** ésta.

a) Formalizar el problema para la estrategia de **Backtracking** indicando: forma de la tupla, restricciones implícitas y explícitas, y función objetivo si corresponde (además explique todos los ítems en lenguaje natural).
b) Argumente de forma breve si las siguientes condiciones son o no predicados de poda:
(vuelta actual = vuelta por comenzar)
    i. El tiempo que lleva el ciclista hasta la vuelta actual, es mayor o igual que el tiempo de la mejor solución encontrada hasta el momento.
    ii. El tiempo que lleva el ciclista hasta la vuelta actual más el tiempo que le llevaría realizar el resto de las vueltas a la mayor velocidad posible sin tener en cuenta el gasto de energía, es mayor o igual que el tiempo de la mejor solución encontrada hasta el momento.
    iii. No considerar para la vuelta actual las velocidades cuyo tiempo sea mayor que la suma del tiempo que lleva dar la vuelta a la máxima velocidad más el tiempo de detención **B**.
    iv. No considerar para la vuelta actual, las velocidades que consumen más energía que la que posee para esa vuelta el ciclista.
c) Independientemente de sus respuestas a la parte (b) :
    i. Describa (en forma breve, correcta y concisa, no escriba código) cómo implementaría cada uno de los ítems **b-i** a **b-iv** en su algoritmo de **Backtracking** durante la construcción de la tupla. Explique brevemente la función que cumple cada ítem en la implementación general.
    ii. En particular, con respecto a (**b-iii**): ¿varía el conjunto de velocidades que deja de considerar por el ítem **b-iii** según cual sea la vuelta actual? ¿esto le permitiría cambiar la implementación sugerida en **c-i**? justifique.

Ejercicio 2. (15 puntos)

Ana y Luis formaron un equipo para participar en una carrera, la cual tiene las siguientes reglas:

- Los equipos deben pasar y marcar los *lugares* $l_0, l_1, \ldots, l_n$, **n>1.**
- La marca de un *lugar* la lleva a cabo uno sólo de los integrantes; el otro queda en el último lugar que marcó.
- Los integrantes del equipo determinan cual de ellos marcará cada *lugar*.
- Se debe respetar el orden de los *lugares* (dado por su numeración), es decir que se debe marcar el *lugar* l_{i+1} inmediatamente después de marcar el *lugar* l_i.
- El integrante que marca el lugar l_{i+1} puede ser cualquiera de los dos integrantes: el que marcó el lugar l_i o el otro (que está en algún lugar anterior al l_i).
- La carrera comienza con un integrante marcando l_0 y el otro l_1 .
- La carrera termina para un equipo cuando alguno de sus integrantes llegue a l_n.

Ana y Luis quieren calcular la **distancia total mínima** que deben recorrer entre **ambos**.

Se dispone de la función ***distancia*** tal que dados dos *lugares* retorna la distancia entre los mismos.

Se pide:
Formalizar el problema aplicando Programación Dinámica. Llame a la fórmula recursiva f. Explicar qué representa el/los pasos base y el/los pasos recursivos de la solución, así como también qué representa la función f indicando sus índices. Se debe indicar la invocación de la función para resolver el problema.

Ejercicio 3. (20 puntos)

a) (10 puntos)
   i. Escriba el algoritmo de *MergeSort* que ordene un arreglo de n enteros sobre sí mismo. Deberá implementar toda función auxiliar que necesite. El código en C* y/o seudocódigo que utilice deberá tener al menos el nivel de detalle visto en el curso teórico.

   ii. Calcule la cantidad de comparaciones realizadas en su algoritmo *MergeSort* en el peor caso. Suponga n potencia de 2.

   iii. ¿Mergesort es óptimo en el peor caso? justifique

b) (10 puntos)
   i. Defina el concepto de árbol de decisión.

   ii. Dibuje el árbol de decisión para un arreglo de 3 elementos para el algoritmo de *MergeSort* escrito en la parte **a -i**.

   iii. Marque la representación de la ejecución de un peor caso en el árbol de decisión de la parte **b-ii**. Muestre que la elección cumple la parte **a-ii**.