Segundo Parcial de Programación 3 (28/11/2011)

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. (22 puntos)

Un repostero conoce P postres navideños diferentes. Cada unidad del postre p, con $1 <= p <= P$, requiere ingredientes por un costo c_p pesos. El repostero cuenta con C pesos para comprar ingredientes. Además, cada unidad del postre p lleva un tiempo de preparación t_p horas y genera al momento de la venta v_p pesos. Por un tema de exclusividad, no pueden prepararse mas de E unidades de un postre p cualquiera.
El repostero confía tanto en su destreza, que asume puede vender todas las unidades de los postres que prepare. Bajo esta suposición, desea planificar la cantidad de unidades de cada postre a preparar en las H horas antes de navidad de manera de maximizar la ganancia total. La ganancia de una unidad de un postre p es $v_p - c_p$, se puede asumir que $c_p < v_p$ para todo postre p.
Se pide:

1) **(10 puntos)** 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).

2) **(6 puntos)** Argumente de forma breve si las siguientes condiciones podan el espacio de soluciones y además si son o no predicados de poda (considere p como el postre actual, es decir el siguiente a planificar):

   a) La ganancia acumulada hasta la planificación del postre p inclusive, es menor que la ganancia de la mejor solución encontrada hasta el momento.
   b) La ganancia acumulada hasta la planificación del postre p inclusive, sumado al valor mas optimista para el resto de la planificación, es menor o igual que la ganancia de la mejor solución encontrada hasta el momento. ¿Cual es el valor más optimista?
   c) No considerar como opción a preparar una determinada cantidad de unidades del postre p si, al incluirlas, el costo acumulado de la planificación es mayor que C.

3) **(6 puntos)** Responda las siguientes preguntas de forma breve y concisa.

   a) Si suponemos que el tiempo de preparación de los postres es 0 y que C es infinito, ¿cuál es la mejor solución?
   b) Si suponemos que el tiempo de preparación es igual para todos los postres y el costo de los ingredientes es 0 para todos los postres, ¿puede hallarse la solución óptima mediante la técnica Greedy? En caso afirmativo explique cómo e indique la estrategia a utilizar.

c) Si suponemos que el costo de los ingredientes es 0 para todos los postres y que E es 1, ¿que otra técnica de resolución vista en el curso puede utilizarse y cómo? (Sugerencia: puede resultar útil reducir a un problema conocido que se resuelva por la técnica en cuestión).

Ejercicio 2. (18 puntos)

Sea un grafo $G = (V, A)$ con $|V| = N$, dirigido, completo, sin lazos y con costos positivos en las aristas, donde los vértices representan ciudades y las aristas carreteras que las unen.
Los costos de las aristas representan el peso máximo que está permitido que circule por las carreteras correspondientes.
Una empresa de transporte desea obtener, a partir de la matriz de costos W (que representa el grafo), el camino que le permita transportar la mayor cantidad de carga (peso) para cada par de ciudades.

a) **(10 puntos)** Formalizar el problema aplicando **Programación Dinámica** (sugerencia: puede resultar útil basarse en alguna de las recurrencias vistas en el curso).
Llame a la fórmula recursiva $\boldsymbol{f}$. Explicar qué representa el/los pasos base y cada paso recursivo de la solución, así como también qué representa la función f indicando sus índices.

Nota: La función $\boldsymbol{f}$ solo debe devolver cuál es la máxima carga que se puede transportar entre cada par de ciudades.

b) **(4 puntos)** Implemente en forma **iterativa** la recurrencia de la parte anterior. El procedimiento debe tener el siguiente cabezal:

```
void cargaMax(int** W, int N, int** Carga, int** Camino)
```

donde
- *Carga*[*i*][*j*] es la carga máxima que se puede transportar por la carretera que une la ciudad *i* a la ciudad *j*.
- *Camino* indica el camino a tomar para transportar esa carga.

c) **(4 puntos)** Escriba un algoritmo (**pseudocódigo**) para reconstruir el camino de carga máxima de *i* a *j* a partir de la matriz *Camino*. **No es necesario reformular la recurrencia**.

Ejercicio 3. (20 puntos)

Dado el problema de ordenar un arreglo A de n elementos mediante comparaciones de elementos 2 a 2.
Considere lo visto en el teórico para la resolución de los siguientes ítems:

a) En este contexto, ¿por qué se utiliza Árbol de Decisión como herramienta para el estudio del problema? Indique brevemente cómo se usa un Árbol de decisión para el estudio del peor caso y del caso medio (no se pide ninguna demostración, sólo mostrar cómo se razona sobre el AD)

b) ¿Que representan los nodos internos y externos de un árbol de decisión?

c) Para un problema de ordenamiento como se indica arriba ¿Cuál es la mínima cantidad de nodos externos que puede tener el árbol de decisión asociado a cualquier algoritmo que resuelva el problema? Justifique su respuesta.

d) Implementar (en C*) QuickSort, inclusive PARTITION, tomando como pivote el último elemento. La posición del pivote no se puede modificar hasta el final de PARTITION. Comentar cada línea de forma de mostrar la estrategia que sigue el algoritmo.

e) Dado un arreglo A=[a,b,c] construya el árbol de decisión del algoritmo de la parte anterior. En cada nodo indique cómo se encuentra el arreglo y la posición del pivote.