FEBRERO 2012 – SEMANA 2 (MODELO A)

1. Identifique cuál de las siguientes secuencias de código implementa un recorrido en anchura de una componente conexa de un grafo:

a.

```
fun RecAnchura(v: nodo, visitado: Vector)
var
   u,w: nodo
   Q: TCola
fvar
Q ← ColaVacía
Encolar(v,Q)
mientras ¬ vacia(Q) hacer
    visitado[v] ← cierto
    u ← Primero(Q)
    Desencolar(u,Q)
    para cada w adyacente a u hacer
      si ¬ visitado[w]  entonces
         visitado[w] ← cierto
      fsi
    Encolar(w,Q)
    fpara
mientras
ffun
```

b.

```
fun RecAnchura(v: nodo, visitado: Vector)
var
   u,w: nodo
   Q: TCola
fvar
Q ← ColaVacía
visitado[v] ← cierto
Encolar(v,Q)
mientras ¬ vacia(Q) hacer
    u ← Primero(Q)
    Desencolar(u,Q)
    para cada w adyacente a u hacer
      si ¬ visitado[w]  entonces
         visitado[w] ← cierto
         Encolar(w,Q)
      fsi
    fpara
mientras
ffun
```

c.

```
fun RecAnchura(v: nodo, visitado: Vector)
var
   u,w: nodo
   Q: TCola
fvar
Q ← ColaVacía
visitado[v] ← cierto
Encolar(v,Q)
u ← Primero(Q)
mientras ¬ vacia(Q) hacer
    Desencolar(u,Q)
    para cada w adyacente a u hacer
      si ¬ visitado[w]  entonces
         Encolar(w,Q)
      fsi
     visitado[w] ← cierto
  fpara
mientras
ffun
```

d. Ninguna de las secuencias anteriores es correcta.

Respuesta B

2. En una carrera de coches por el desierto uno de los principales problemas es el abastecimiento de gasolina. Uno de los participantes está decidido a ganar y ha calculado que con el tanque de gasolina lleno puede recorrer M km. sin parar. El participante dispone de un mapa que le indica las distancias entre las gasolineras que

hay en la ruta y cree que, parándose a repostar el menor número de veces posible, podrá ganar. Para ayudarle hay que diseñar un algoritmo que le sugiera en qué gasolineras debe hacerlo. Hay que tener en cuenta que hay una única ruta posible. Indique de los esquemas siguientes cuál es el más adecuado:

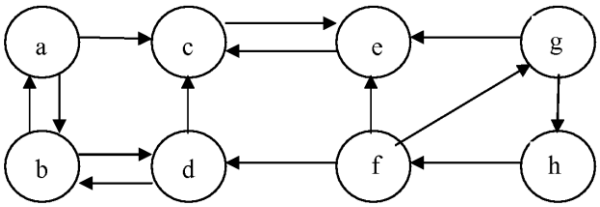
a. El esquema voraz.

b. El esquema programación dinámica.

c. El esquema de vuelta atrás.

d. El esquema de ramificación y poda.

3. ¿Cuántos componentes fuertemente conexos existen en el grafo de la siguiente figura?



a. Uno solo, formado por todo el grafo.

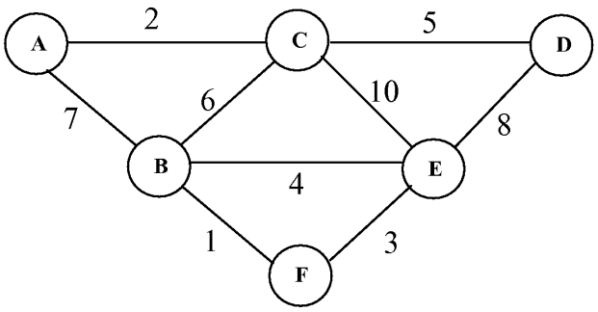
b. Dos.

c. Tres.

d. Más de tres.

Explicación: a-b-d; c-e; f-g-h

4. Dado el grafo no dirigido de la figura,



indique cuál sería el orden en que se seleccionarían (pasan a pertenecer al árbol) los nodos al aplicar el algoritmo de Prim comenzando por el nodo A:

a. A, F, E, C, B, D

b. A, B, C, E, D, F

c. A, C, D, B, F, E

d. Ninguno de los anteriores

5. Dado el vector v=[5,6,9,3,4, 1] y el algoritmo Quicksort, los vectores argumento de la primera invocación recursiva, con pivote v[1] = 5, son:

a. [3,4, 1] y [6,9]

b. [5,6,9] y [5,3,4,1]

c. [5] y [6,9,3,4,1]

d. [1] y [5,6,9,3,4]

HAY UNA ERRATA - En la primera invocación, solo separa el pivote. Después, va ordenando el resto para finalmente separarlo en 2 mitades y poner el pivote en medio. La errata existe porque la respuesta C) correspondería a la elección del pivote, la respuesta A) corresponde a las 2 mitades, pero falta el pivote. Cuando se va a invocar la primera recursividad hay 3 conjuntos, el pivote [5], los elementos menores que el pivote [3,4,1] y los mayores [6,9].

6. Dado el problema de la mochila donde tenemos una mochila de capacidad M, n objetos con beneficios b_1, b_2, b_3, ... b_n y pesos $_p1$, $_p2$, $_p3$, ... $_pn$. y siendo el objetivo maximizar el valor de los objetos transportados, respetando la limitación de la capacidad impuesta M, en el caso de que los objetos sean fraccionables, la resolución del mismo:

a. Debe resolverse mediante un esquema voraz ya que no es eficiente realizarlo mediante ramificación y poda.

b. Debe hacerse mediante ramificación y poda, ya que no es posible resolverlo utilizando un esquema voraz.

c. Sólo puede resolverse mediante un esquema de programación dinámica.

d. Puede hacerse mediante un esquema voraz pero no es posible resolverlo mediante ramificación y poda.

PROBLEMA (4 Puntos)

Se quiere realizar en un soporte secuencial (cinta de dos caras) una recopilación de canciones preferidas. Se dispone de una lista de n canciones favoritas, junto con la duración individual de cada una. Lamentablemente, la cinta de T minutos no tiene capacidad para contener todas las canciones, por lo que se les ha asignado una puntuación (cuanto más favorita es, mayor es la puntuación). Se pretende obtener la mejor cinta posible según la puntuación, teniendo en cuenta que las canciones deben caber enteras y no es admisible que una canción se corte al final de una de las caras.

La resolución de este problema debe incluir, por este orden:
- Elección del esquema más apropiado, el esquema general y explicación de su aplicación al problema (0,5 puntos).
- Descripción de las estructuras de datos necesarias (0.5 puntos).
- Algoritmo completo a partir del refinamiento del esquema general (2,5 puntos).
- Estudio del coste del algoritmo desarrollado (0.5 puntos).

SOLUCIÓN

1. Elección del esquema más apropiado, el esquema general y explicación de su aplicación al problema.

El esquema elegido es el de Ramificación y poda. Se trata de una optimización y no existe una función de selección de soluciones parciales que nos permita garantizar que dichas soluciones parciales construidas lleven a la óptima final, por lo que no se trata de un algoritmo voraz. Tampoco existe una forma de dividir el problema en subproblemas que se puedan resolver independientemente, por lo que tampoco es posible un esquema divide y vencerás. Ramificación y poda tiene la ventaja de podar la exploración de muchos nodos mediante la estimación de las cotas.

```
funcion ramificacion_poda (ensayo) dev ensayo {ensayo es un nodo}
        m ← monticulo_vacio();
        cota_superior ← cota_superior_inicial;
        solucion ← primera_solucion;
        añadir_nodo(m, ensayo);
        mientras ¬ vacio(m) hacer
                nodo ← extraer_raiz(m);
                si valido(nodo) entonces {solución completa}
                        si coste_asig(nodo) < cota_superior entonces
                                solucion ← nodo;
                                cota_superior ← coste_asig(nodo)
                fsi
                si no
                        si cota_inferior(nodo) ≥ cota_superior entonces dev solucion
                        si no
                                para cada hijo en compleciones(nodo) hacer
                                        si cota_inferior(hijo) < cota_superior
                                                añadir_nodo(m, hijo)
                                        fsi
                                fpara
                        fsi
                fsi
        fmientras
ffuncion
```

Se considera en cada etapa una canción y se considera grabarla en cada una de las dos caras, si hay espacio suficiente, así como la posibilidad de no grabarla en la cinta. La posibilidad de grabarla en la segunda cara sólo se plantea cuando la ocupación de las dos caras es distinta.

Una cota superior podría ser la puntuación estimada considerando como espacio libre total la suma del espacio libre en cada cara y grabar el máximo posible de canciones, aunque alguna quede cortada.

2. Descripción de las estructuras de datos necesarias

Se utiliza un montículo en el que cada nodo será necesario almacenar: Solución parcial Etapa Puntuación-estimada Puntuación acumulada Ocupación de cada cara.

Además, en un array tendremos las canciones con su puntuación y su duración. Por lo que la estructura de cada nodo sería un registro con los siguientes campos:

Sol[1..n] de 0..2
K: 0..n
Puntuación_acumulada: real
Puntuación-estimada: real {prioridad}
Ocupada[1..2] de real

3) Algoritmo completo a partir del refinamiento del esquema general

Una descripción detallada de la solución puede encontrarse en el libro de “Estructuras de datos y métodos algorítmicos” en la Sección 15.3, página 513.

4. Estudio del coste del algoritmo desarrollado

Una estimación del coste es el tamaño del árbol de búsqueda, que en el peor caso crece como O(n!), ya que cada nodo del nivel k puede expandirse con las n − k canciones que quedan por considerar.