# Examen de Algorítmica. Septiembre de 2012.
# Tiempo de duración: 2 horas.

1. Considerar el siguiente algoritmo recursivo basado en Divide y Venceras:

```
float DV(int n)
{
int m,k;
float d;
 d = n + 1;
 if (n > 1){
  m=n/2;
  d = d + DV(m);
  if(m > 1){
   k=m/2;
   d=d+4DV(k);
   d=d+8DV(k);
  }
 }
 return d;
}
```

   - Estimar el orden de complejidad del algoritmo.
   - Explicar de forma clara y concisa los cambios que habría que realizar para mejorar la eficiencia de este algoritmo.

2. Escribir el algoritmo para la ordenación por mezcla Mergesort basado en la técnica Divide y Vencerás, incluyendo la descripción del código para la operación Combinar. Obtener el tiempo de ejecución de la ordenación por mezcla.

3. Sobre un rio navegable hay $n$ embarcaderos. En cada uno de ellos se puede alquilar un bote que permite ir a cualquier otro embarcadero rio abajo (ya que es imposible ir rio arriba). Existe una tabla de tarifas $T[i, j]$ que indica el coste del viaje del embarcadero $i$ al $j$ para cualquier embarcadero de partida $i$ y cualquier embarcadero de llegada $j$ mas abajo en el rio ($i < j$).

   Puede suceder que un viaje de $i$ a $j$ sea mas caro que una sucesión de viajes mas cortos, en cuyo caso se tomaria un primer bote hasta un embarcadero $k$ y un segundo bote para continuar a partir de $k$. No hay coste adicional por cambiar de bote. Aquí el problema consiste en diseñar un algoritmo eficiente usando Programación Dinámica que determine el coste mínimo para cada par de puntos $i, j$ con $i < j$.

   Para el problema anterior resuelto usando Programación Dinámica, construir la ecuación de recurrencia con casos base, y definir la estrategia de aplicación de la fórmula (tablas utilizadas por el algoritmo, orden y forma de rellenarlas). Indicar también el coste de complejidad (en función de $n$) del algoritmo.

4. En el problema de la mochila no 0/1 tenemos $n$ objetos, cada uno con un peso $p_i$ y un beneficio $b_i$. También disponemos de una mochila en la que podemos meter objetos, con una capacidad máxima $M$. El objetivo es llenar la mochila, maximizando el beneficio de los objetos transportados, y respetando la limitación de capacidad máxima $M$. Los objetos se pueden partir en trozos.

   Para este problema de la mochila no 0/1 se os pide determinar el mejor criterio posible para seleccionar el mejor objeto de los restantes (función de selección) usando la técnica de construcción de algoritmos voraces. Demostrar (por reducción al absurdo) que dicho criterio garantiza la solución óptima. Finalmente escribir el algoritmo voraz resultante.

5. Describir los tipos comunes de árboles de backtracking, especificando los tipos de problemas mas adecuados a cada uno de ellos. Determinar para cada caso un ejemplo concreto. Se os pide claridad y concisión en las respuestas.

   Además escribir el esquema general (no recursivo) de backtracking, especificando las variables y las funciones usadas en el esquema general.