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.

Examen de Algorítmica. Septiembre de 2013. Tiempo disponible: Una hora y media.

1. Calcular $t(n)$ y el orden de complejidad para

$$t(n) = 2t(n-1) + 3^n(n+1)$$

2. En el problema de la mochila 0/1 disponemos de tres mochilas, con capacidades $M1$, $M2$ y $M3$. El objetivo es maximizar la suma de beneficios de los objetos transportados en las tres mochilas, respetando las capacidades de cada una. Resolver el problema mediante programación dinámica, definiendo la ecuación de recurrencia, las tablas usadas y el algoritmo para rellenarlas.

3. Escribir un algoritmo de Backtraking para el problema de encontrar un subconjunto del conjunto $T = \{t_1, t_2, \cdots, t_n\}$, que sume exactamente una cantidad P. Es necesario describir la representación de la solución, el esquema general del algoritmo, y las funciones de dicho esquema general.

4. En el problema de asignación, existen n trabajos y n personas. Cada persona i puede realizar un trabajo j con más o menos rendimiento: $B[i, j]$. El objetivo es asignar una tarea a cada trabajador (asignación uno-a-uno), de manera que se maximice la suma de rendimientos.

   Para el problema de asignación de n tareas a n personas resuelto usando Ramificación y Poda, describir en primer lugar la representación de la solución, y en segundo lugar el método de obtención de estimaciones precisas para las funciones dadas por la Cota Inferior, la Cota Superior, y el Beneficio Estimado que emplearemos en las estrategias de poda y de ramificación. Es necesario mostrar el código de las funciones.

Examen de Algorítmica. Julio de 2014. Tiempo disponible: 2 horas.

1. Función de selección para el algoritmo VORAZ que resuelve de forma óptima el problema de la mochila no(0/1), donde 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 y existe un total de n objetos distintos cada uno con un peso p_i y un beneficio b_i.
2. Obtener la ecuación de recurrencia para el problema de la mochila 0/1 resuelto mediante **Programación Dinámica**. En este caso los objetos no se pueden partir en trozos, y existe un total de n objetos distintos cada uno con un peso p_i y un beneficio b_i.
3. Consideremos de nuevo el problema de la mochila (0/1). Si lo resolvemos mediante un algoritmo de **Backtracking**, identificar como tendremos que definir la función **Criterio** haciendo uso de una cota superior del beneficio que se puede obtener a partir del nodo actual.
4. Para el problema de la mochila (0/1) usando **Ramificación y Poda**, describir (mostrando exclusivamente el código de las funciones) estimaciones precisas para la Cota Inferior, la Cota Superior, y el Beneficio Estimado que emplearemos en las estrategias de poda y de ramificación.
5. *Evaluación Continua (Valor: 1 punto). Entrega de los ejercicios resueltos de evaluación continua.*

Universidad de Granada
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL
ETSI Informática y Telecomunicación, C/Periodista Daniel Saucedo Aranda s/n - 18071 - Granada (España)

Examen de Algorítmica.

Curso 2014–2015. Convocatoria ordinaria de Junio. Duración: 3 horas.

Grado en Ingeniería Informática.

1. (2.5 pt) **Eficiencia**

   - Usando la notación O, determinar la eficiencia de los siguientes segmentos de código:

```
int n,j; int i=1; int x=0;
 do{
    j=1;
    while (j <= n){
       j=j*2;
       x++;
    }
    i++;
  }while (i<=n);
```

```
int n,j; int i=2; int x=0;
 do{
     j=1;
     while (j <= i){
         j=j*2;
         x++;
     }
     i++;
   }while (i<=n);
```

   - Resolver la siguiente recurrencia:

$$T(n) = 3T(n/2) - 2T(n/4) + n, \qquad \text{con } T(1) = T(2) = 1$$

   ¿Cuál es su orden de eficiencia?

2. (2 pt.) **Divide y Vencerás**

   - Dado un vector de enteros (positivos y negativos) de tamaño n, encontrar el subvector de suma máxima, esto es, se debe cumplir que la suma de todos los elementos del subvector sea la mayor posible.

3. (1.5 pt.) **Greedy**

   - Encontrar (usando el algorítmo de Dijkstra) el camino más corto entre el vértice 1 y todos los demás en el grafo con la siguiente matriz de adyacencias:

	1	2	3	4
1	0	4	3	9
2	2	0	7	3
3	7	9	0	8
4	5	8	5	0

FAX: +34.958.243317 TLF.: +34.958.244019

4. (1.5 pt.) **Programación Dinámica**

   - Obtener la ecuación de recurrencia para el problema de las 3 mochilas $0/1$ resuelto mediante **Programación Dinámica**. En este caso los objetos no se pueden partir en trozos, y hay un total de n objetos distintos cada uno con un peso p_i y un beneficio b_i. Suponemos que hay un total de tres mochilas disponibles, cuyas capacidades máximas respectivas son $M1$, $M2$ y $M3$. El objetivo es llenar las tres mochilas maximizando el beneficio de los objetos transportados (Indicación: una ecuación de recurrencia permite definir una sola función mochila, aunque esta puede tener más parámetros de entrada).

5. (2.5 pt.) **Branch and bound**

   - Tenemos un conjunto de barcos (b1,b2,b3,b4) y un conjunto de pasajeros (p1,p2,p3.p4). El coste de que un determinado conjunto de pasajeros p_i viaje en el barco b_j viene dado por la siguiente tabla:

	b1	b2	b3	b4
p1	95	2	55	69
p2	75	11	89	83
p3	63	89	9	77
p4	12	75	82	22

   Diseñar un algorítmo para asignar los pasajeros a los barcos con el mínimo coste global.

Universidad de Granada
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL
ETSI Informática y Telecomunicación, C/Periodista Daniel Saucedo Aranda s/n - 18071 - Granada (España)

Examen de Algorítmica.

Curso 2014–2015. Convocatoria de Septiembre. Duración: 3 horas.

Grado en Ingeniería Informática.

1. (2.5 pt) **Eficiencia**

   - Usando la notación O, determinar la eficiencia de los siguientes segmentos de código:

```
int sum1=0; int k,j,n;
                                    |
                                    |
    for(k=1; k<=n; k*=2)            |
        for(j=1; j<=n; j++)         |
                sum1++;             |
```

```
int sum2=0; int k,j,n;

  for(k=1; k<=n; k*=2)
      for(j=1; j<=k; j++)
              sum2++;
```

   - Resolver la siguiente recurrencia:

$$T(n) = 3T(n/2) - 2T(n/4) + log_2 n, \qquad \text{con } T(1) = T(2) = 3$$

   ¿Cuál es su orden de eficiencia? ¿Proviene realmente T(n) de analizar la eficiencia de un algorítmo?

2. (2 pt.) **Divide y Vencerás**

   - Dado un vector de enteros (positivos y negativos) de tamaño n **no ordenado**, encontrar mediante Divide y Vencerás los valores máximo y mínimo presentes en el vector .

3. (1.5 pt.) **Greedy**

   - Una compañia eléctrica debe elegir la ruta que deben seguir sus técnicos para arreglar una serie de averías que se han producido en una ciudad. las averías están situadas en 5 puntos B,C,D,E,F situándose la central en un punto A. la distancia entre los distintos puntos viene dada por la tabla:

A						
B	6					
C	8.07	6				
D	17.55	12.70	15			
E	16.52	12.05	15.32	4		
F	19	15.32	19.38	8.6	5	
Dist	A	B	C	D	E	F

FAX: +34.958.243317 TLF.: +34.958.244019

Encontrar (usando una técnica greedy) la mejor ruta posible para minimizar la distancia recorrida por los técnicos.

4. (1.5 pt.) **Programación Dinámica**

   - Plantear la ecuación recurrente solución al siguiente problema: Sobre un rio hay n embarcaderos. En cada uno de ellos, se puede alquilar un bote que permite ir a cualquier otro embarcadero rio abajo (se supone 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 más abajo en el rio ($i < j$). Puede suceder que un viaje de i a j sea más caro que una sucesión de viajes más cortos, en cuyo caso se tomaría 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. Nuestro problema consiste en diseñar un algorítmo eficiente que determine el coste mínimo para cada par de puntos i,j ($i < j$). Solo se pide la ecuación de recurrencia base para resolver el problema.

5. (2.5 pt.) **Branch and bound**

   - Una empresa constructora de motores tiene distintas empresa proveedoras de piezas. Para construir un motor se necesitan un conjunto de piezas (p1,p2,p3,p4) que pueden ser servidas por un conjunto de empresas (e1,e2,e3.e4). El coste de que una determinada pieza p_i si se compra a una determinada empresa e_j viene dado por la siguiente tabla:

	p1	p2	p3	p4
e1	96	3	56	70
e2	76	12	90	84
e3	64	90	10	78
e4	13	76	83	23

   Diseñar un algorítmo para determinar la compra óptima de piezas para la empresa constructora de motores (a qué empresa se compra cada pieza) con el mínimo coste global.

Universidad de Granada
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL
ETSI Informática y Telecomunicación, C/Periodista Daniel Saucedo Aranda s/n - 18071 - Granada (España)

Examen de Algorítmica

Curso 2015–2016. Convocatoria ordinaria de Junio. Duración: 3 horas.

Grado en Ingeniería Informática.

1. (2.5 pt) **Eficiencia**

   - Calcular el tiempo de ejecución del siguiente algoritmo:

```
EnteroLargo Algjunio2016 (EnteroLargo u, v; int n)
    {
        if (n == 1)
           return  MultBasica(u,v);
        else {
                 w = primeros(n/2, u);
                 x = ultimos(n/2, u);
                 y = primeros(n/2, v);
                 z = ultimos(n/2, v);
                m1 = Algjunio2016(w, y, n/2);
                m2 = Algjunio2016(x, z, n/2);
                m3 = Algjunio2016(w-x, z-y, n/2);
                return (10^n * m1 + 10^{n/2} * (m1 + m2 + m3) + m2);
              }
     }
```

     donde u y v son enteros de longitud n; $MultBasica(u, v)$ es la multiplicación escalar de u por v (eficiencia constante); $primeros(n/2, u)$ devuelve los primeros $n/2$ dígitos de u; $ultimos(n/2, u)$ devuelve los últimos $n/2$ dígitos de u (eficiencia lineal).

   - Resolver la recurrencia siguiente:

     $$T(n) = 2 \times T(\sqrt{(n)}) + log_2 n \quad n \geq 4 \quad T(2)=1$$

2. (2 pt.) **Divide y Vencerás**

   - Un vector A[1 ... 2n+1] está peinado si:

     $A[1] \leq A[2] \geq A[3] \leq A[4] \geq A[5] \ldots \leq A[2n] \geq A[2n+1]$

     Dado un vector (no ordenado) de números reales, diseñar un algoritmo divide y vencerás para peinarlo.

FAX: +34.958.243317 TLF.: +34.958.244019

3. (1.5 pt.) **Greedy**

   - Un estudiante está haciendo su examen de algorítmica. Sabe que el profesor ha asignado puntos a cada problema de acuerdo a su dificultad. El estudiante a su vez estima para cada problema, el tiempo que le llevará solucionarlo. El objetivo es hacer las preguntas en un cierto orden que maximice la nota final de acuerdo a los parámetros (nota de la pregunta y tiempo estimado para resolverla). Diseñar un algorítmo Greedy que resuelva el problema. Los datos son:
     - Notas: $p_1, p_2, \ldots, p_n$
     - Tiempo estimado: $t_1, t_2, \ldots, t_n$
     - Tiempo total para hacer el examen: T

4. (1.5 pt.) **Programación Dinámica**

   - Obtener la ecuación de recurrencia para el problema de las 2 mochilas $0/1$ resuelto mediante **Programación Dinámica**. En este caso los objetos no se pueden partir en trozos, y hay un total de n objetos distintos cada uno con un peso p_i y un beneficio b_i. Suponemos que hay un total de dos mochilas disponibles, cuyas capacidades máximas respectivas son $M1$ y $M2$. El objetivo es llenar las dos mochilas maximizando el beneficio de los objetos transportados. Poner un ejemplo de su funcionamiento.

5. (2.5 pt.) **Branch and Bound**

   - Un operador en el desierto tiene que llenar n recipientes de agua (cada uno con un volumen diferente) para su distribución y dispone para ello de diferentes tanques de agua cada uno con una capacidad diferente. Por el sistema de llenado de que dispone, tiene que asignar cada tanque a un recipiente de forma que se minimice la suma de los valores absolutos de las diferencias entre los volúmenes de los recipientes y los tanques a los que han sido asignados para su llenado. Resolver el problema usando un algorítmo Branch and Bound y aplicarlo al siguiente ejemplo:
     - Volumen: 193 183 205 185
     - Capacidad: 198 203 183 190

     Como ayuda, indicar que el vector de mejor diferencia estimada es
     - 3, 0, 2, 2

Universidad de Granada
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL
ETSI Informática y Telecomunicación, C/Periodista Daniel Saucedo Aranda s/n - 18071 - Granada (España)

Examen de Algorítmica

Curso 2015–2016. Convocatoria ordinaria de Junio. Duración: 3 horas.

Grado en Ingeniería Informática.

1. (2.5 pt) **Eficiencia**

   - Calcular el tiempo de ejecución del siguiente algoritmo:

```
EnteroLargo Algjunio2016 (EnteroLargo u, v; int n)
    {
        if (n == 1)
           return  MultBasica(u,v);
        else {
                 w = primeros(n/2, u);
                 x = ultimos(n/2, u);
                 y = primeros(n/2, v);
                 z = ultimos(n/2, v);
                m1 = Algjunio2016(w, y, n/2);
                m2 = Algjunio2016(x, z, n/2);
                m3 = Algjunio2016(w-x, z-y, n/2);
                return (10^n * m1 + 10^{n/2} * (m1 + m2 + m3) + m2);
              }
     }
```

     donde u y v son enteros de longitud n; $MultBasica(u, v)$ es la multiplicación escalar de u por v (eficiencia constante); $primeros(n/2, u)$ devuelve los primeros $n/2$ dígitos de u; $ultimos(n/2, u)$ devuelve los últimos $n/2$ dígitos de u (eficiencia lineal).

   - Resolver la recurrencia siguiente:

     $$T(n) = 2 \times T(\sqrt{(n)}) + log_2 n \quad n \geq 4 \quad T(2)=1$$

2. (2 pt.) **Divide y Vencerás**

   - Un vector A[1 ... 2n+1] está peinado si:

     $A[1] \leq A[2] \geq A[3] \leq A[4] \geq A[5] \ldots \leq A[2n] \geq A[2n+1]$

     Dado un vector (no ordenado) de números reales, diseñar un algoritmo divide y vencerás para peinarlo.

FAX: +34.958.243317 TLF.: +34.958.244019

3. (1.5 pt.) **Greedy**

   - Un estudiante está haciendo su examen de algorítmica. Sabe que el profesor ha asignado puntos a cada problema de acuerdo a su dificultad. El estudiante a su vez estima para cada problema, el tiempo que le llevará solucionarlo. El objetivo es hacer las preguntas en un cierto orden que maximice la nota final de acuerdo a los parámetros (nota de la pregunta y tiempo estimado para resolverla). Diseñar un algorítmo Greedy que resuelva el problema. Los datos son:
     - Notas: $p_1, p_2, \ldots, p_n$
     - Tiempo estimado: $t_1, t_2, \ldots, t_n$
     - Tiempo total para hacer el examen: T

4. (1.5 pt.) **Programación Dinámica**

   - Obtener la ecuación de recurrencia para el problema de las 2 mochilas $0/1$ resuelto mediante **Programación Dinámica**. En este caso los objetos no se pueden partir en trozos, y hay un total de n objetos distintos cada uno con un peso p_i y un beneficio b_i. Suponemos que hay un total de dos mochilas disponibles, cuyas capacidades máximas respectivas son $M1$ y $M2$. El objetivo es llenar las dos mochilas maximizando el beneficio de los objetos transportados. Poner un ejemplo de su funcionamiento.

5. (2.5 pt.) **Branch and Bound**

   - Un operador en el desierto tiene que llenar n recipientes de agua (cada uno con un volumen diferente) para su distribución y dispone para ello de diferentes tanques de agua cada uno con una capacidad diferente. Por el sistema de llenado de que dispone, tiene que asignar cada tanque a un recipiente de forma que se minimice la suma de los valores absolutos de las diferencias entre los volúmenes de los recipientes y los tanques a los que han sido asignados para su llenado. Resolver el problema usando un algorítmo Branch and Bound y aplicarlo al siguiente ejemplo:
     - Volumen: 193 183 205 185
     - Capacidad: 198 203 183 190

     Como ayuda, indicar que el vector de mejor diferencia estimada es
     - 3, 0, 2, 2

Universidad de Granada
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL
ETSI Informática y Telecomunicación, C/Periodista Daniel Saucedo Aranda s/n - 18071 - Granada (España)

Examen de Algorítmica

Curso 2015–2016. Convocatoria extraordinaria Septiembre. Duración: 3 horas.

Grado en Ingeniería Informática.

1. (2.5 pt) **Eficiencia**

   - Calcular el tiempo de ejecución del siguiente algoritmo:

```
int Algsep2016 (int u, v; int n)
    {
       int x,y, m1,m2;

        if (n == 1)
           return  OpBasica(u,v);
        else {
                x = funcion1(n/2, u);
                y = funcion2(n/2, v);

               m1 = Algsep2016(x, y, n/2);
               m2 = Algsep2016(x+1, y+1, n/2);

               return (m1 + m2);
             }
     }
```

     donde $OpBasica(u, v)$ tiene eficiencia constante y $funcion1(n/2, u)$ $funcion2(n/2, v)$ tienen eficiencia cuadrática.

   - Resolver la siguiente recurrencia:

     $$T(n) = T(n/2) + 2T(n/4) + n, \qquad \text{con } T(1), T(2) = 1$$

     y determinar su orden de eficiencia

2. (2 pt.) **Divide y Vencerás**

   - Dado un vector con n elementos donde se sabe que algunos están repetidos. Diseña un algoritmo DyV para eliminar todos los elementos repetidos (los elementos deben mantener el mismo orden relativo).

3. (1.5 pt.) **Greedy**

   - Un club deportivo quiere completar su plantilla con un máximo de k jugadores. Tras explorar el mercado, el ojeador preselecciona n posibles candidatos. Para cada candidato dicho ojeador ha acordado una ficha a pagar a su club de origen, f_i. De igual forma, el ojeador proporciona al club el rendimiento esperado de cada jugador, r_i.

     Se pide diseñar un algoritmo que permita al equipo realizar la selección del mejor grupo de jugadores (el rendimiento del grupo es la suma del rendimiento de los distintos componentes), teniendo en cuenta que no se puede pasar del presupuesto global P.

     Resolver el problema con la siguiente instancias: Presupuesto global $P = 570$; número de jugadore $n = 5$; tamaño del grupo $k = 3$, fichas $F = (20, 180, 210, 220, 350)$; rendimiento de jugadores $R = (15, 19, 28, 35, 37)$.

4. (1.5 pt.) **Programación Dinámica**

   - Mediante Programación Dinámica encontrar la parentización óptima para realizar la multiplicación de matrices siguiente:

$$A_{10x20} \cdot B_{20x50} \cdot C_{50x1} \cdot D_{1x100}$$

5. (2.5 pt.) **Branch and Bound**

   - Un supervisor de intendencia tiene que asignar n botas (cada una con una talla diferente) para su distribución a un conjunto de soldados cada uno con una medida diferente de pie. Por el sistema de asignación de que dispone, tiene que asignar cada bota a un soldado de forma que se minimice la suma de los valores absolutos de las diferencias entre las tallas de las botas y las medidas de los pies a las que han sido asignadas. Resolver el problema usando un algorítmo Branch and Bound y aplicarlo al siguiente ejemplo:

     - Talla: 19.0 18.0 20.2 18.2
     - Tamaño de pie: 19.5 20.0 18.0 18.7

     Como ayuda, indicar que el vector de mejor diferencia estimada es:

     - 0.3, 0, 0.2, 0.2