FUNDAMENTOS DE ANÁLISIS DE ALGORITMOS

GRADO EN INGENIERÍA INFORMÁTICA. La Rábida 8 de junio de 2015. **Tiempo máximo: 120 minutos.**

ALUMNO/A		Nº HOJAS		NOTA	

EJERCICIO 1 — PUNTOS: 2

- Escribir un algoritmo que dados un vector de **n** enteros y un entero **X**, determine si existen en el vector dos números cuya suma sea X. (1 punto).
- El tiempo del algoritmo debe ser de O(n**log* n). Analiza el algoritmo y demuestra que es así. (1 punto).

EJERCICIO 2 — PUNTOS: 1

Usar las relaciones $\subset$ y **=** para ordenar los órdenes de complejidad, **O**, **Ω**, y **Θ**, de las siguientes funciones:

$n\log n$, $n^2\log n$, n^8, n^{1+a}, $(1+a)^n$, $(n^2+8n+\log^3 n)^4$, $n^2/\log n$, 2^n, siendo **a** una constante real, $0 < \mathbf{a} < 1$.

EJERCICIO 3 — PUNTOS: 1

Usando las definiciones de notación asintótica, demostrar si son verdaderas o falsas las afirmaciones siguientes:

1. $1024n^2 + 5n \in \mathbf{\Theta}(n^2)$. (0,5 puntos)
2. $(n+1)! \in \mathbf{O}\ (3(n!))$. (0,25 puntos)
3. $n^2 \in \mathbf{\Omega}\ ((n+1)^2)$. (0,25 puntos)

EJERCICIO 4 — PUNTOS: 2

Para resolver cierto problema se dispone de dos algoritmos, $\mathbf{A_1}$ y $\mathbf{A_2}$, de divide y vencerás:

- $\mathbf{A_1}$ descompone el problema de tamaño **n** en tres subproblemas de tamaño **n/2** y cuatro subproblemas de tamaño **n/4**. La división y combinación requieren $\mathbf{3n^2}$, y el caso base, con **n** menor que 5, es **n!**
- $\mathbf{A_2}$ descompone el problema de tamaño **n** en un subproblema de tamaño **n-3** y dos de tamaño **n-2**. El tiempo de la división y combinación es despreciable, y el caso base, con **n** menor que 5, es de orden constante.

1. Calcular el orden de complejidad de los dos algoritmos. (1,5 puntos)
2. Estudiar cuál de los dos algoritmos es más eficiente. (0,5 puntos)

EJERCICIO 5 — PUNTOS: 1

Para resolver cierto problema se dispone de un algoritmo trivial cuyo tiempo de ejecución t(n) (para problemas de tamaño n) es cuadrático ($t(n)\in\Theta(n^2)$). Se ha encontrado una estrategia Divide y Vencerás (DyV)para resolver el mismo problema; dicha estrategia realiza ***D(n)= n log n*** operaciones para dividir el problema en dos subproblemas de tamaño mitad y ***C(n)= n log n*** operaciones para componer una solución del original con la solución de dichos subproblemas.

1. Calcular la eficiencia para el algoritmo DyV por el método de la **ecuación característica.** (0,5 puntos)
2. Corroborar el resultado anterior aplicando el teorema maestro. (0,25 puntos)
3. Estudiar cuál de los dos algoritmos es más eficiente. (0,25 puntos)

NOTA:

Teorema: La solución a la ecuación $T(n)= aT(n/b) + \Theta(n^k \log^p n)$, con $a \geq 1$, $b>1$ y $p \geq 0$, es:

$$\mathbf{T(n)=}\begin{cases} \mathbf{O}(n^{\log_b a}) & \text{si } a>b^k \\ \mathbf{O}(n^k \log^{p+1} n) & \text{si } a=b^k \\ \mathbf{O}(n^k \log^p n) & \text{si } a<b^k \end{cases}$$

FUNDAMENTOS DE ANÁLISIS DE ALGORITMOS

GRADO EN INGENIERÍA INFORMÁTICA**.** La Rábida 8 de junio de 2015. **Tiempo máximo: 120 minutos.**

EJERCICIO 6 PUNTOS: 2

Dado el siguiente algoritmo de ordenación por Selección modificado (de forma que se intercambien los elementos únicamente si son distintos) para el caso medio,

1. Estudiar la complejidad del algoritmo. (1,5 puntos)
2. Decidir si es rentable o no la modificación. (0,5 puntos)

- El procedimiento Selección_Modificado puede ser implementado como sigue:

```
procedimiento  SelectionModif (a:vector; primero,ultimo: int);
    para i=primero hasta ultimo-1 hacer
           posmin = PosMinimo(a,i,ultimo);
           si a[i] ≠ a[posmin] entonces  /* nueva instrucción para Selección_Modificado */
               Intercambia(a, i, posmin);
           fsi;
    fpara
fprocedimiento SelectionModif
```

➢ En el algoritmo anterior se utiliza una función que calcula la posición del elemento mínimo de un subvector :

```
Int función PosMinimo (a:vector;primero,ultimo:int);
/* devuelve la posición del mínimo elemento de a[primero..ultimo] */
    pmin=primero;
    para i=primero+1 hasta ultimo hacer
           si a[i] < a[pmin] entonces
                  pmin = i
           fsi;
    fpara
    return pmin;
ffuncion PosMinimo;
```

➢ También se utiliza el procedimiento ***Intercambia*** para intercambiar dos elementos de un vector:

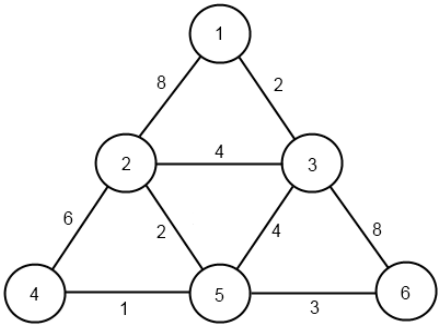
```
función Intercambia (a:vector ; i , j :int );
/* intercambia a[i]  con a[j]  */
    aux = a[i] ;
    a[i] = a[j] ;
    a[j] = aux;
ffuncion Intercambia;
```

EJERCICIO 7 PUNTOS: 1

Consideremos el mapa del casco antiguo o centro de una ciudad donde están todas las intersecciones de las calles y todas sus longitudes. Supongamos que en cada intersección de calles existe una plaza. El ayuntamiento ha decidido peatonalizar las calles del centro. Sin embargo, como el presupuesto es reducido ha decidido peatonalizar aquellos tramos de calle de tal forma que todas las plazas queden unidas por tramos de calle peatonalizadas.

➢ Determinar que tramos de calle son necesarios peatonalizar para resolver el problema con un coste mínimo utilizando **el algoritmo de Prim**. Detallar :

1. Las estructuras y/o variables necesarias para representar la información del problema y el método voraz utilizado (El procedimiento o función que implemente el algoritmo). (0,75 puntos)
2. Realizar la traza para el mapa ejemplo de la figura. (0,25 puntos)



NOTAS:

- El problema consiste en obtener el árbol de recubrimiento mínimo del grafo formado por todas las calles (aristas) y plazas (nodos) del centro.
- Suponer que el gasto por unidad de longitud es g.
- Como todas las calles tienen el mismo coste de peatonalización (g por cada unidad de longitud), se puede considerar directamente su longitud como el peso de cada arista.
- El algoritmo de Prim se basa en la estrategia voraz: parte de un vértice cualquiera y va extendiendo el árbol de recubrimiento, incorporando un nuevo vértice en cada iteración (y la arista correspondiente), hasta cubrir todos los vértices del grafo.