Un algoritmo recursivo basado en el esquema *divide y vencerás* ...

Seleccione una:

a. ... será más eficiente cuanto más equitativa sea la división en subproblemas.

b. Las demás opciones son verdaderas.

c. ... nunca tendrá una complejidad exponencial.

La respuesta correcta es: ... será más eficiente cuanto más equitativa sea la división en subproblemas.

Indicad cuál de estas tres expresiones es falsa:

Seleccione una:

a. $\Theta(n/2) = \Theta(n)$

b. $\Theta(n) \subseteq \Theta(n^2)$

c. $\Theta(n) \subseteq O(n)$

La respuesta correcta es: $\Theta(n) \subseteq \Theta(n^2)$

¿Cuál de estas tres expresiones es falsa?

Seleccione una:

a. $3n^2 + 1 \in O(n^3)$

b. $n + n\log(n) \in \Omega(n)$

c. $n + n\log(n) \in \Theta(n)$

Indica cuál es la complejidad en el peor caso de la función replace:

```
unsigned bound( const vector<int> &v ) {
    for( unsigned i = 0; i < v.size(); i++ )
        if( v[i] == '0')
            return i;
    return v.size();
}

void replace( vector<int>& v, int c ) {
    for( insigned i = 0; i < bound(v); i++)
        v[i] = c;
}
```

Seleccione una:

- a. $O(n \log n)$
- b. $O(n^2)$ ✓
- c. $O(n)$

¿Cuál es la complejidad temporal de la siguiente función recursiva?

```
unsigned desperdicio (unsigned n){
if (n<=1)
   return 0;
unsigned sum = desperdicio (n/2) + desperdicio (n/2);
for (unsigned i=1; i<n-1; i++)
    for (unsigned j=1; j<=i; j++)
        sum+=i*j;
return sum;
}
```

Seleccione una:

- a. $\Theta(n^2)$ ✓
- b. $\Theta(2^n)$
- c. $\Theta(n^2 \log n)$

Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n/2) + 1$; $f(1) = 1$. Indicad cuál de estas tres expresiones es cierta:

Seleccione una:

- a. $f(n) \in \Theta(n)$
- b. $f(n) \in \Theta(n^2)$
- c. $f(n) \in \Theta(n \log(n))$

La respuesta correcta es: $f(n) \in \Theta(n)$

Considerad estos dos fragmentos:

```
s=0;for(i=0;i<n;i++) s+=i;
```

y

```
s=0;for(i=0;i<n;i++) if (a[i]!=0) s+=i;
```

y un array `a[i]` de números enteros. Indicad cuál de estas tres afirmaciones es cierta:

Seleccione una:

- a. El coste temporal asintótico del primer programa en el caso peor es más alto que en el segundo.
- b. El coste temporal asintótico, tanto en el caso mejor como en el caso peor, de los dos programas es el mismo.
- c. El coste temporal asintótico del segundo programa en el caso peor es más alto que en el primero.

La respuesta correcta es: El coste temporal asintótico, tanto en el caso mejor como en el caso peor, de los dos programas es el mismo.

Indica cuál es la complejidad, en función de n, del fragmento siguiente:

```
int a = 0;
for( int i = 0; i < n; i++ )
    for(  int j = i; j > 0; j /=2 )
        a += A[i][j];
```

Seleccione una:

- a. $O(n \log n)$
- b. $O(n)$
- c. $O(n^2)$

La respuesta correcta es: $O(n \log n)$

Indica cuál es la complejidad en función de n, donde k es una constante (no depende de n), del fragmento siguiente :

```
for( int i = k; i < n - k; i++){
    A[i] = 0;
    for( int j = i - k; j < i + k; j++ )
        A[i] += B[j];
}
```

Seleccione una:

- a. $O(n)$
- b. $O(n \log n)$
- c. $O(n^2)$

La respuesta correcta es: $O(n)$

Pertenece $3n^2+3$ a $O(n^3)$?

Seleccione una:

- a. Sólo para $c=1$ y $n_0=5$.
- b. No.
- c. Sí.

La respuesta correcta es: Sí.

La complejidad temporal en el mejor de los casos...

Seleccione una:

- a. Las demás opciones son verdaderas.
- b. ... es el tiempo que tarda el algoritmo en resolver la talla más pequeña que se le puede presentar.
- c. ... es una función de la talla que tiene que estar definida para todos los posibles valores de ésta.

La respuesta correcta es: ... es una función de la talla que tiene que estar definida para todos los posibles valores de ésta.

La versión de *Quicksort* que utiliza como pivote la mediana del vector ...

Seleccione una:

- a. ... se comporta mejor cuando el vector ya está ordenado.
- b. ... se comporta peor cuando el vector ya está ordenado.
- c. ... El hecho de que el vector estuviera previamente ordenado o no, no influye en la complejidad temporal de este algoritmo.

La respuesta correcta es: ... El hecho de que el vector estuviera previamente ordenado o no, no influye en la complejidad temporal de este algoritmo.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n = 1 \\ \sqrt{n} + 3f(n/3) & n > 1 \end{cases}$$

Seleccione una:

- a. $f(n) \in \Theta(n)$
- b. $f(n) \in \Theta(n^3)$
- c. $f(n) \in \Theta(\sqrt{n} \log n)$

La respuesta correcta es: $f(n) \in \Theta(n)$

Un problema de tamaño n puede transformarse en tiempo $O(n^2)$ en nueve de tamaño $n/3$; por otro lado, la solución al problema cuando la talla es 1 requiere un tiempo constante.
¿cual de estas clases de coste temporal asintótico es la más ajustada?

Seleccione una:

- a. $O(n^2)$
- b. $O(n^2 \log n)$
- c. $O(n \log n)$

La respuesta correcta es: $O(n^2 \log n)$

Indica cuál es la complejidad, en función de n, del siguiente fragmento de código:

```
s=0; for(i=0;i<n;i++) for(j=i;j<n;j++) s+=i*j;
```

Seleccione una:

- a. $\Theta(n^2)$ ✓
- b. $O(n^2)$ pero no $\Omega(n^2)$.
- c. $\Theta(n)$

Sea $f(n)$ la solución de la relación de recurrencia $f(n)=2f(n-1)+1$. $f(1)=1$. Indicad cuál de estas tres expresiones es cierta:

Seleccione una:

- a. $f(n)\in\Theta(n)$
- b. $f(n)\in\Theta(2^n)$ ✓
- c. $f(n)\in\Theta(n^2)$

Un programa con dos bucles anidados uno dentro del otro, El primero hace n iteraciones aproximadamente y el segundo la mitad, tarda un tiempo

Seleccione una:

- a. $O(n\log n)$
- b. $O(n^2)$ ✓
- c. $O(n\sqrt{n})$

¿Cuál es la complejidad temporal de la siguiente función recursiva?

```
unsigned desperdicio (unsigned n){
if (n<=1)
    return 0;
unsigned sum = desperdicio (n/2) + desperdicio (n/2);
for (unsigned i=1; i<n-1; i++)
    for (unsigned j=1; j<=i; j++)
        for (unsigned k=1; k<=j; k++)
            sum+=i*j*k;
return sum;
}
```

Seleccione una:

- a. $\Theta(2^n)$
- b. $\Theta(n^3\log n)$
- c. $\Theta(n^3)$ ✓

Los algoritmos de ordenación *Quicksort* y *Mergesort* tienen en común ...

Seleccione una:

a. ... que se ejecutan en tiempo $O(n)$.

b. ... que ordenan el vector sin usar espacio adicional.

c. ... que aplican la estrategia de *divide y vencerás*.

Indica cuál es la complejidad de la función siguiente:

```
unsigned sum( const mat &A ) {   // A es una matriz cuadrada
unsigned d = A.n_rows();
unsigned a = 0;
    for( unsigned i = 0; i < d; i++ )
        for( unsigned j = 0; j < d; j++ )
            a += A(i,j);
return a;
}
```

Seleccione una:

a. $O(n^2)$

b. $O(n)$

c. $O(n \log n)$

Indicad cuál de estas tres expresiones es falsa:

Seleccione una:

a. $\Theta(n/2) = \Theta(n)$

b. $\Theta(n) \subseteq O(n)$

c. $\Theta(n) \subseteq \Theta(n^2)$

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

Seleccione una:

a. El árbol de cobertura de coste mínimo de un grafo conexo.

b. El problema de la mochila discreta o sin fraccionamiento.

c. El problema de la mochila continua o con fraccionamiento.

Los algoritmos de programación dinámica hacen uso ...

Seleccione una:

- a. ... de que la solución óptima se puede construir añadiendo a la solución el elemento óptimo de los elementos restantes, uno a uno.
- b. ... de que se puede ahorrar cálculos guardando resultados anteriores en un almacén. ✓
- c. ... de una estrategia trivial consistente en examinar todas las soluciones posibles.

Cuando se calculan los coeficientes binomiales usando la recursión $\binom{n}{r}=\binom{n-1}{r}+\binom{n-1}{r-1}$, con $\binom{n}{0}=\binom{n}{n}=1$, qué problema se da y cómo se puede resolver?

Seleccione una:

- a. La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica.
- b. Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz.
- c. Se repiten muchos cálculos y ello se puede evitar usando programación dinámica. ✓

Sea $f(n)$ la solución de la relación de recurrencia $f(n)=2f(n/2)+n$; $f(1)=1$. Indicad cuál de estas

Seleccione una:

- a. $f(n)\in\Theta(n^2)$
- b. $f(n)\in\Theta(n)$
- c. $f(n)\in\Theta(n\log(n))$ ✓

Para que la complejidad de un algoritmo presente caso mejor y peor distintos ...

Seleccione una:

- a. ... es condición necesaria y suficiente que existan instancias distintas del problema con el mismo tamaño.
- b. ... es condición necesaria que existan instancias distintas del problema con el mismo tamaño. ✓
- c. ... es condición suficiente que existan instancias distintas del problema con el mismo tamaño.

Un problema de tamaño n puede transformarse en tiempo $O(n)$ en siete de tamaño $n/7$; por otro lado, la solución al problema cuando la talla es 1 requiere un tiempo constante.

¿cual de estas clases de coste temporal asintótico es la más ajustada?

Seleccione una:

- a. $O(n^2)$
- b. $O(n)$
- c. $O(n\log n)$ ✓

Indicad cuál de estas tres expresiones es cierta:

Seleccione una:

- a. $O(n^2) \subset O(2^{\log(n)}) \subset O(2^n)$
- b. $O(n^2) \subset O(2^{\log(n)}) \subseteq O(2^n)$
- c. $O(2^{\log(n)}) \subset O(n^2) \subset O(2^n)$ ✔

La complejidad temporal en el mejor de los casos de un algoritmo recursivo...

Seleccione una:

- a. ... coincide con el valor del caso base de la ecuación de recurrencia que expresa la complejidad temporal del algoritmo.
- b. Las demás opciones son falsas. ✔
- c. ... siempre coincidirá con la complejidad temporal de las instancias que están en el caso base del algoritmo recursiva.

Considerad la función siguiente:

```
int M( int i, int f) {
    if ( i == f )
        return i;
    else {
        e = v[ M( i, (i+f)/2 ) ];
        f = v[ M( (i+f)/2+1, f ) ];
        if (e<f)
             return e;
        else
             return f;
    }
}
```

Si la talla del problema viene dada por $n = f - i + 1$, ¿cuál es el coste temporal asintótico en el supuesto de que n sea una potencia de 2?

Seleccione una:

- a. $O(n)$. ✔
- b. $O(n^2)$.
- c. $O(n\,\log(n))$.

El coste temporal asintótico del fragmento

```
s=0; for(i=0;i<n;i++) for(j=i;j<n;j++) s+=i*j;
```

y el del fragmento

```
s=0; for(i=0;i<n;i++) for(j=0;j<n;j++) s+=i*i*j;
```

son ...

Seleccione una:

- a. ... iguales. ✓
- b. ... el del segundo, menor que el del primero.
- c. ... el del primero, menor que el del segundo.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n = 1 \\ n^2 + 3f(n/3) & n > 1 \end{cases}$$

Seleccione una:

- a. $f(n) \in \Theta(n^2)$
- b. $f(n) \in \Theta(n^2 \log n)$
- c. $f(n) \in \Theta(n)$

La versión de *Quicksort* que utiliza como pivote el elemento del vector que ocupa la primera posición ...

Seleccione una:

- a. ... se comporta mejor cuando el vector ya está ordenado.
- b. ... se comporta peor cuando el vector ya está ordenado. ✓
- c. ... El hecho de que el vector estuviera previamente ordenado o no, no influye en la complejidad temporal de este algoritmo.

La versión de *Quicksort* que utiliza como pivote el elemento del vector que ocupa la posición central ...

Seleccione una:

- a. ... se comporta mejor cuando el vector ya está ordenado. ✓
- b. ... se comporta peor cuando el vector ya está ordenado.
- c. ... no presenta casos mejor y peor distintos para instancias del mismo tamaño.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n)=\begin{cases}1 & n=1\\ n+3f(n/3) & n>1\end{cases}$$

Seleccione una:

- a. $f(n)\in\Theta(n\log n)$ ✓
- b. $f(n)\in\Theta(n^3)$
- c. $f(n)\in\Theta(n)$