Test

1. Pregunta

Un algoritmo óptimo que comprueba si un vector de n elementos es estrictamente decreciente tiene complejidad en el caso mejor:

1.

1.a.O(logn)

1.b.O(1)

1.c.O(nlogn)

1.d.Ninguna de las anteriores

Solución

Falso. En el caso mejor, el vector no cumple la propiedad y se detiene inmediatamente.

Cierto. En el caso mejor, el vector no cumple la propiedad y se detiene inmediatamente.

Falso. En el caso mejor el vector no cumple la propiedad y se detiene inmediatamente.

Falso. La respuesta correcta es O(1).

1. Pregunta

El coste en el caso peor del algoritmo de ordenación por inserción sobre un vector de n elementos está en:

Algoritmo	In situ	Estable	Peor	Medio	Mejor	Comentarios
Burbuja	si	si	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n)$	no usar nunca
Selección	si	no	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	n intercambios
Inserción	si	si	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n)$	utilizar para vectores pequeños
Shellsort	si	no	$\Theta(n^{1+1/k})$	$\Theta(n^{1+1/k})$	$\Theta(n)$	se puede mejorar
Quicksort	si	no	$\Theta(n^2)$	$\Theta(n \log n)$	$\Theta(n \log n)$	el mas ràpido en la práctica
Mergesort	no	si	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	$n \log n$ garantizado, estable
Heapsort	si	no	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	$n \log n$ garantizado, in situ

1.

1.a.Θ(n)

1.b.Θ(2n)

1.c.Θ(logn)

1.d.Θ(n2)

Solución

El caso peor del algoritmo de ordenación por inserción se da cuando el vector a ordenar está ordenado de forma inversa a la deseada. En tal caso para insertar cada elemento en su posición definitiva ha de desplazar todos los anteriores. Por tanto, el coste está en Θ(n2).

1. Pregunta

Un algoritmo óptimo para insertar un elemento en un vector no necesariamente ordenado sin elementos repetidos (de forma que siga sin tener elementos repetidos) tiene complejidad en el caso peor (la más ajustada):

1.

1.a.O(n)

1.b.O(n2)

1.c.O(logn)

1.d.O(nlogn)

Solución

Para poder insertar primero hemos de saber si el elemento ya está en el vector, lo que tiene coste O(n) en el caso peor ya que por no estar ordenado puede ser necesario mirar todos los elementos. Una vez comprobado que no está ya en el vector basta con colocarlo al final.

1. Pregunta

Un algoritmo óptimo que busca el mínimo en un vector ordenado de n elementos tiene complejidad en el caso peor:

1.

1.a.Θ(n2)

1.b.Θ(nlogn)

1.c.Θ(n)

1.d.Ninguna de las anteriores.

Solución

Falso. Basta con consultar el primer o último elemento del vector dependiendo de si está ordenado creciente o decrecientemente.

Falso. Basta con consultar el primer o último elemento del vector dependiendo de si está ordenado creciente o decrecientemente.

Falso. Basta con consultar el primer o último elemento del vector dependiendo de si está ordenado creciente o decrecientemente.

Cierto. La respuesta correcta es Θ(1).

1. Pregunta

Indica la complejidad del siguiente algoritmo:

```
int a = 0;
for (int i = 0; i < n+20; ++i)
  for (int j = m+30; j >= 0; --j)
    --a;
```

1.

1.a.Θ(n^m)

1.b.Θ(max(n,m))

1.c.Θ(1)

1.d.Ninguna de las anteriores.

Solución

Falso. En los bucles anidados independientes la complejidad se multiplica.

Falso. En los bucles anidados independientes la complejidad se multiplica.

Falso. En los bucles anidados independientes la complejidad se multiplica.

Cierto. La respuesta correcta es Θ(n*m).

1. Pregunta

Indica la complejidad del siguiente algoritmo:

```
int a = 0;
for (int i = 1; i < n; i *= 3)
   --a;
```

1.

1.a.Θ(nlogn)

1.b.Θ(n2)

1.c.Θ(n)

1.d.Ninguna de las anteriores.

Solución

Falso. El bucle no da un número de vueltas proporcional a nlogn.

Falso. El bucle no da un número de vueltas proporcional a n2.

Falso. El bucle no da un número de vueltas proporcional a n.

Cierto. La respuesta correcta es Θ(logn)

1. Pregunta

Dada la especificación

{ 0 <= n <= long(v) }

fun xxx (int v[], int n, int x) dev int r

{ r = # u : 0 <= u < n : v[u] < x }

y teniendo en cuenta que estamos considerando los n primeros elementos del vector, indica qué afirmación es correcta con respecto a ella.

1.

1.a.La postcondición está mal definida cuando n=0.

1.b.El valor de r es la posición en el vector del último elemento menor que x.

1.c.El valor de r es la posición en el vector del primer elemento menor que x.

1.d.Ninguna de las anteriores.

Solución

Falso. Cuando n=0, el operador de conteo está bien definido y r vale 0.

Falso. El operador # es el operador de conteo.

Falso. El operador # es el operador de conteo.

Cierto. La respuesta correcta es: El valor de r es el número de posiciones del vector que contienen elementos menores que x.

1. Pregunta

Dada la especificación

{ 0 < n <= longitud(v) }

fun xxx (int v[], int n) dev int r

{ r = max u : 0 <= u < n Λ u mod 2 = 0 : v[u] }

y teniendo en cuenta que estamos considerando los n primeros elementos del vector, indica qué afirmación es correcta con respecto a ella.

1.

1.a.El valor de r es la posición más a la derecha que contiene un elemento par.

1.b.El valor de r es el máximo de los elementos pares del vector.

1.c.El valor de r es el máximo de los elementos del vector que se encuentran en posiciones pares.

1.d.Ninguna de las anteriores.

Solución

Falso. El valor de r es un elemento del vector, no una posición.

Falso. Son las posiciones del vector las pares, no los elementos.

Cierto. El valor de r es un elemento del vector, el mayor de los que están situados en posiciones pares.

Falso. La respuesta correcta es: El valor de r es el máximo de los elementos del vector que se encuentran en posiciones pares.

1. Pregunta

Dada la siguiente especificación y valores de los parámetros de entrada, indica cuál es el valor del resultado:

{ 0 <= n <= longitud(v) }

fun xxx (int v[], int n) dev int r

{ r = # u : 0 <= u < n : suma(v,0,u) < suma(v,u+1,n) }

siendo

suma(v,c,f) = sum z : c <= z < f : v[z]

Datos de entrada: n=5, v=[1,-1,1,-1,1]

1.

1.a.r=3.

1.b.r=0.

1.c.r=2.

1.d.Ninguna de las anteriores.

Solución

1.

1.a.Falso. Para las posiciones 0, 1 y 2 se cumple que suma(v,0,0) = 0 = suma(v,1,5), suma(v,0,1) = 1 =suma(v,2,5) y suma(v,0,2) = 0 = suma(v,3,5).

1.b.Cierto. Las posiciones i∈{0,2,4} cumplen suma(v,0,i) = 0 = suma(v,i+1,5) y las posiciones i∈{1,3} cumplen suma(v,0,i) = 1 = suma(v,i+1,5).

1.c.Falso. Las posiciones i∈{0,2,4} cumplen suma(v,0,i) = 0 = suma(v,i+1,5) y las posiciones i∈{1,3} cumplen suma(v,0,i) = 1 = suma(v,i+1,5).

1.d.Falso. La respuesta correcta es: r=0.

1. Pregunta

El siguiente predicado es más débil que cualquier otro predicado:

$x>0 \vee x<0$

1.

1.a.Verdadero

1.b.Falso

Solución

Falso. El valor x=0 no satisface el predicado.

1. Pregunta

El siguiente predicado es más débil que cualquier otro predicado:

$false \rightarrow true$

1.

1.a.Falso

1.b.Verdadero

Solución

Verdadero. El predicado equivale a true y este es el predicado más débil.

1. Pregunta

Dado un vector v con N elementos, ¿qué representa la siguiente expresión?

Max p, q: 0 ≤ p ≤ q ≤ N ∧ P (v, p, q): q − p

siendo

P (v, p, q) = ∀ l : p ≤ l < q : v[l] ≥ 0 ∧ v[l] mod 2 = 0

1.

1.a.El número de posiciones del vector cuyos elementos cumplen la propiedad de ser pares.

1.b.La resta de las dos mayores posiciones del vector cuyos elementos cumplen la propiedad de ser pares.

1.c.El máximo número de posiciones consecutivas cuyos elementos cumplen la propiedad de ser pares.

1.d.Ninguna de las anteriores.

Solución

Falso. La propiedad P describe una propiedad sobre segmentos [p,q).

Falso. El máximo se refiere a la longitud q−p del segmento más largo [p,q) que cumple la propiedad indicada, no a las posiciones.

Cierto. Expresa la longitud q−p del segmento más largo [p,q) que cumple la propiedad indicada.

Falso. La respuesta correcta es: El máximo número de posiciones consecutivas cuyos elementos cumplen la propiedad de ser pares. Expresa la longitud q−p del segmento más largo [p,q) que cumple la propiedad indicada.

1. Pregunta

Indica cuál es la precondición más débil:

{??}

x = y;

y = x;

{x=3 ∧ y=4}

1.

1.a.y≠x

1.b.y=3 ∧ x=4

1.c.true

1.d.Ninguna de las anteriores.

Solución

Falso. Al sustituir en la postcondición y por x y luego x por y se obtiene y=3 ∧ y=4, así que la precondición más débil es false.

Falso. Al sustituir en la postcondición y por x y luego x por y se obtiene y=3 ∧ y=4, así que la precondición más débil es false.

Falso. Al sustituir en la postcondición y por x y luego x por y se obtiene y=3 ∧ y=4, así que la precondición más débil es false.

Cierto. La respuesta correcta es: false.

1. Pregunta

¿Cuál de estas afirmaciones es correcta? (x e y variables de tipo entero).

1.

1.a.[X > 0 ∧ y < 0] nada x+y > 0

1.b.[x+y > 0] nada x > 0

1.c.[x > 0 ∧ y > 0] nada x/y > 0

1.d.Ninguna de las anteriores.

Solución

Falso. El estado x = 3, y = -4, satisface el predicado x>0∧y<0 pero no x+y>0

Falso. El estado x = -2, y = 3, satisface el predicado x+y>0, pero no x>0.

Falso. El estado x = 2, y = 4, satisface el predicado x>0∧y>0, pero no x/y>0

Cierto. La respuesta correcta es: {x > 0 ∧ y > -x} nada {x + y > 0}.

1. Pregunta

Indica cuál es la afirmación correcta acerca de este algoritmo de búsqueda binaria en el que la precondición es 0 ≤ c ≤ f+1 ≤ longitud(v)

```
int f (int v[], int x, int c, int f)

{
```

```
 int resultado;
 if (c>f)
  {resultado = c-1;}
 else
  {
   int m = (c+f)/2;
   if (v[m] > x)
    {resultado = f(v,x,c,m);}
   else
    {resultado = f(v,x,m,f);}
  }
 return resultado;
}
```

1.

1.a.Solo termina si c<f.

1.b.Nunca termina.

1.c.Siempre termina.

1.d.Ninguna de las anteriores.

Solución

Falso. El algoritmo no termina en todos aquellos casos en los que c≤f. El punto medio cumple c≤m≤f y por tanto las llamadas recursivas (con c,m y m,f) de nuevo cumplen que c≤f. Luego, si inicialmente c≤f se llegará a un caso base en el que c=m=f, que repetidamente se invocará a sí mismo, generando una recursión infinita.

Falso. En los casos en que c>f el algoritmo termina.

Falso. El algoritmo no termina en todos aquellos casos en los que c≤f. El punto medio cumple c≤m≤f y por tanto las llamadas recursivas (con c,m y m,f) de nuevo cumplen que c≤f. Luego, si inicialmente c≤f se llegará a un caso base en el que c=m=f, que repetidamente se invocará a sí mismo, generando una recursión infinita.

Cierto. La respuesta correcta es: Solo termina si c>f.

1. Pregunta

Indica cuál de las siguientes es una función de cota para este bucle:

```
int i=0; bool encontrado=false;
while (i<n && !encontrado){
if (v[i]>0)
  {encontrado=true;}
else
  {i=i+1;}
}
```

1.

1.a.n−i

1.b.i

1.c.n−i+1

1.d.Ninguna de ellas.

Solución

Incorrecta. La variable i no siempre se incrementa, por lo que n−i no decrece en todas las vueltas del bucle.

Incorrecta. La variable i no decrece en cada vuelta.

Incorrecta. La variable i no siempre se incrementa, por lo que n−i+1 no decrece en todas las vueltas del bucle.

Correcta. Puesto que la variable i no se incrementa cuando la variable booleana se hace cierta, una función de cota podría ser: n−i+(encontrado?0:1). Dicha función decrece estrictamente en aquellas vueltas en las que i se incrementa en 1. También decrece estrictamente cuando i no crece, ya que en ese caso encontrado se hace cierto y la función pasa de valer n−i+1 a n−i.

1. Pregunta

En un problema de maximización resuelto con vuelta atrás la cota optimista o beneficio estimado

1.

1.a.siempre puede ser el valor de la solución parcial.

1.b.puede ser 0 si los beneficios son positivos.

1.c.debe ser cota inferior de la mejor solución alcanzable.

1.d.Ninguna de las anteriores.

Solución

Falso. En un problema de maximización del beneficio la cota optimista ha de ser una cota superior de la mejor solución alcanzable. De esa forma, si la cota optimista es menor que el mejor beneficio obtenido hasta el momento podemos podar esa rama puesto que en ella no hay ninguna solución que mejore a la actual. El valor de la solución parcial no es en general una cota superior de la mejor solución alcanzable.

Falso. En un problema de maximización del beneficio la cota optimista ha de ser una cota superior de la mejor solución alcanzable. De esa forma, si la cota optimista es menor que el mejor beneficio obtenido hasta el momento podemos podar esa rama puesto que en ella no hay ninguna solución que mejore a la actual. El valor 0 no es en general una cota superior de la mejor solución alcanzable.

Falso. En un problema de maximización del beneficio la cota optimista ha de ser una cota superior, no inferior, de la mejor solución alcanzable. De esa forma, si la cota optimista es menor que el mejor beneficio obtenido hasta el momento podemos podar esa rama puesto que en ella no hay ninguna solución que mejore a la actual.

Cierto. La respuesta correcta es: debe ser cota superior de la mejor solución alcanzable.

1. Pregunta

El coste de la poda de optimalidad puede ser tan elevado que no compense la poda conseguida.

1.

1.a.Verdadero

1.b.Falso

Solución

Verdadero. Si para determinar si el subárbol correspondiente a una solución parcial puede conducir o no a una solución mejor que la encontrada hasta el momento se invierte demasiado tiempo, el coste total del algoritmo en el que se realiza dicha poda puede ser superior a aquel en que no se utiliza incluso aunque el número de nodos podados sea mayor utilizándola