TEORIA BLOQUE 3 DE ALGORITMICA

Jaime Lorenzo Sánchez

18 de diciembre de 2021

Capítulo 1

Cuestiones de Programación Dinámica

1. Describe el método general de la programación dinámica, escribiendo la fórmula recursiva que representa el método general.

El método general de la programación dinámica consiste en resolver un problema combinando las soluciones para subproblemas más pequeños.

La fórmula recursiva que representa el método general es: $Dinamico(A, B) = Optimo((A, C_i) + Dinamico(C_i, B))$

2. Enuncia el principio de optimalidad de Belman. ¿Se cumple siempre este principio?. Indica un ejemplo que justifique tu respuesta.

Principio de optimalidad de Belman: Una política óptima tiene la propiedad de que cualquiera que sea el estado inicial y la decisión inicial, las decisiones restantes deben constituir una política óptima en relación con el estado resultante de la primera decisión.

Sin embargo, este principio no siempre se cumple. Por ejemplo, en el caso de que los subcaminos para ir desde una ciudad A a una ciudad B no sean independientes, la selección óptima de un subcamino puede impedir que sea óptima la selección del resto de subcaminos.

3. Describe el algoritmo de la competición internacional. Escribe la fórmula general para obtener cualquiera de los posibles estados que se producen a lo largo de la competición.

El algoritmo de la competición general consiste en calcular la probabilidad de que dados dos equipos (A y B), el equipo A gane la competición. Para ello, se utiliza una matriz de modo que un elemento P(i,j) reflejará la probabilidad de que A gane la competición (si P(i,j) = 1 gana el equipo A; si P(i,j) = 0 gana el equipo B), de modo que para obtener un elemento P(i,j) cualquiera se utiliza la siguiente fórmula recursiva:

$P(i,j) = p * P(i-1,j) + (1-p) * P(i,j-1)$

4. ¿Cual es el orden de complejidad del algoritmo de la competición internacional en sus versiones recursiva e iterativa?. Justifica tu respuesta.

El orden de complejidad del algoritmo de la competición internacional recursivo es exponencial, pues se generan dos llamadas recursivas en la ejecución del algoritmo. De este modo, se realizan como máximo 2^n llamadas recursivas.

El orden de complejidad del algoritmo de la competición internacional iterativa es $O(n^2)$, pues se resuelve el algoritmo iterativo recorriendo la matriz resultado por filas y rellenándola durante el recorrido. De este modo, se realizarán como máximo (n*n) operaciones.

5. ¿Como sería la fórmula general recursiva del algoritmo de la competición internacional si se enfrentasen tres jugadores (A, B y C) de forma tal que gana la competición aquel que gane n veces, sabiendo que la probabilidad de que gane A en un enfrentamiento es p, la de que gane B es q y la de que gane C es 1-p-q? Escribe también los casos particulares.

Casos particulares: $P(0,j,q) = 1,\ P(i,0,q) = 0,\ P(i,j,0) = 0$

Caso general: $P(i,j,q) = p*P(i-1,j,q) + q*P(i,j-1,q) + (1-p-q)*P(i,j,q-1)$

6. Indica como se resuelve el problema del cambio usando programación dinámica, describiendo la expresión recursiva que lo resuelve.

El problema del cambio usando programación dinámica se resuelve generando una tabla donde fuesen apareciendo los resultados de los subproblemas intermedios útiles y que se combinarán para obtener la solución del problema inicial.

Expresión recursiva: $c(i,j) = min(c(i-1,j), 1 + c(i, j - v_i)$

7. Resuelve el problema del cambio usando programación dinámica para el caso siguiente: Cambio = 6, valores de las monedas = 1, 3, 4. Indica paso a paso las monedas que se obtienen de cada tipo en la solución final.

Paso 0: Se crea la tabla de 4x8, de modo que las filas 1,2,3 almacenan el valor de las monedas 1,3 y 4 respectivamente, y las columnas 2-7 representan el cambio (0 a 6 respectivamente).

	Cantidad:	0	1	2	3	4	5	6
	$v1 = 1$	0						
	$v2 = 3$	0						
	$v3 = 4$	0						

Cuadro 1.1: Tabla paso 0

Paso 1: Se obtiene el número de monedas de valor 1. Para ello, se rellenan los elementos de la fila 1 ($v1 = 1$) eligiendo el mínimo entre el valor que hay encima de cada elemento y uno más el valor que queda una posición a su izquierda (pues el valor de la moneda es 1).

	Cantidad:	0	1	2	3	4	5	6
	$v1 = 1$	0	1	2	3	4	5	6
	$v2 = 3$	0						
	$v3 = 4$	0						

Cuadro 1.2: Tabla paso 1

Paso 2: Se aplica el paso 1 para la fila 2, de modo que se elige el mínimo entre el valor que hay encima de cada elemento y uno más el valor que queda 3 posiciones a su izquierda (el valor de la moneda es 3).

	Cantidad:	0	1	2	3	4	5	6
	$v1 = 1$	0	1	2	3	4	5	6
	$v2 = 3$	0	1	2	1	2	3	2
	$v3 = 4$	0						

Cuadro 1.3: Tabla paso 2

Paso 3: Se aplica el paso 1 para la fila 3, de modo que se elige el mínimo entre el valor que hay encima de cada elemento y uno más el valor que queda 4 posiciones a su izquierda (el valor de la moneda es 4).

	Cantidad:	0	1	2	3	4	5	6
	$v1 = 1$	0	1	2	3	4	5	6
	$v2 = 3$	0	1	2	1	2	3	2
	$v3 = 4$	0	1	2	3	1	2	3

Cuadro 1.4: Tabla paso 3

Paso 4: Para determinar que cantidad de cada moneda forma parte de la solución, debemos tener en cuenta lo siguiente:

1. Para un elemento cualquiera c(i,j), si c(i,j) = c(i-1,j), indica que no se va a necesitar ninguna moneda de tipo i en el cambio, y pasaríamos a ver el elemento c(i-1,j).

2. Si c(i,j) = 1 + c(i, j-v(i)), entonces se contabiliza la moneda de tipo i que vale v(i) y pasaríamos a ver el elemento c(i,j-v(i)).

3. Si c(i,j) = c(i-1,j) y c(i,j) = 1 + c(i,j-v(i)), podemos seguir cualquiera de los pasos anteriores.

Aplicando este método tenemos lo siguiente:

1. (c(3,7) = 3) = (1+c(3,3) = 1+2=3) ->monedas (v3=4) = 1. Pasamos a ver el elemento c(3,3)

2. (c(3,3) = 2) = (c(2,3) = 2) ->Pasamos a ver el elemento c(2,3)

3. (c(2,3) =2) = (c(1,3) = 2) ->Pasamos a ver el elemento c(1,3)

4. (c(1,3) = 2) = (1+c(1,2) = 1 + 1 = 2) ->Monedas(v1=1) = 1. Procesamos el elemento c(1,2).

5. (c(1,2) = 1) = (1+c(1,1) = 1+ 0 = 1) ->Monedas (v1=1) = 2. Procesamos el elemento c(1,1).

6. (c(1,1) = 0) ->Finaliza el calculo del cambio.

Por tanto, la solución es: 1 moneda de valor 4 y 2 monedas de valor 1.

8. ¿Podría tener más de una solución el problema del cambio?. Justifica tu respuesta.

No, porque la solución del problema del cambio es la solución óptima.

9. Resuelve el problema de la mochila si se tienen 4 materiales de volúmenes 1, 2, 3, 4 y de precios unitarios (por unidad de volumen) de 1, 3, 4, 5 sabiendo que el volumen de la mochila es de 6.

Para calcular los elementos de cada fila, debemos aplicar la siguiente fórmula:

$C(i,j) = max(C(i-1,j), p_i * v_i + C(i-1, j-v_i))$

Volumen Limite:	0	1	2	3	4	5	6
v1=1,p1=1	0	1	2	3	4	5	6
v2=2,p2=3	0	1	6	7	8	9	10
v3=3,p3=4	0	1	6	12	13	18	19
v4=4,p4=5	0	1	6	12	20	21	26

Cuadro 1.5: Problema mochila (ejercicio 9)

Para comprobar si se incluye el material, debemos aplicar lo siguiente:

1. Si $C(i,j) = C(i-1,j)$ pero $C(i,j)! = C(i,j-v_i) + p_i * v_i$: No se incluye el material i y analizamos el material $C(i-1,j)$

2. Si $C(i,j)! = C(i-1,j)$ pero $C(i,j) = C(i,j-v_i) + p_i * v_i$: Se incluye el material y analizamos el material $C(i-1,j-v_i)$

3. Si $C(i,j) = 0$: No se incluye el material i y analizamos el material $C(i-1,j)$

Aplicando dichas condiciones, tenemos lo siguiente:

1. $C(4,6)! = C(3,6)$, pero $C(4,6) = C(4,2)+20)$: Entra el material v4. Comprobamos el material $C(3,2)$

2. $C(3,2) = C(2,2)$, pero $C(3,2)! = C(3,-1) + 12$: No entra el material v3. Comprobamos el material $C(2,2)$

3. $C(2,2)! = C(1,2)$,pero $C(2,2) = C(2,0) + 6$: Entra el material v2. Comprobamos el material $C(1,0)$

4. Como $C(1,0) = 0$, no entra el material v1.

Como el precio final de la mochila viene dado por $C(4,6)$. Por tanto, se han utilizado los materiales v4 y v2, siendo el precio final de la mochila 26.

10. Indica como se resuelve el problema de la mochila usando programación dinámica, describiendo la expresión recursiva que lo resuelve.

El problema de la mochila usando programación dinámica se resuelve creando una tabla que representase los distintos problemas, donde hay una fila para cada material disponible y una columna para cada volumen desde 0 hasta V. De este modo, la solución al problema planteado sería el elemento C(n,V).

Expresión recursiva: $C(i,j) = max(C(i-1,j), p_i * v_i + C(i-1,j-v_i))$

11. ¿Podría tener más de una solución el problema de la mochila?. Justifica tu respuesta.

No, porque el problema de la mochila usando programación dinámica obtiene la solución óptima.

12. ¿Qué pequeña diferencia existe entre las fórmulas recursivas para obtener la solución al problema del cambio y al problema de la mochila?

En el problema de la mochila el material i sólo se selecciona una vez, mientras que en el problema del cambio una moneda se puede seleccionar varias veces.

13. ¿En qué consiste el problema del camino mínimo en un grafo polietápico?. ¿Se puede obtener más de una solución?. Justifica tu respuesta.

Consiste en obtener el camino mínimo entre el nodo origen y un nodo Y, siendo Y un nodo cualquiera del grafo que podría corresponder a cualquier etapa.

Si, porque todos los caminos que van desde el nodo A al nodo Y tienen el mismo número de lados.

14. ¿Cuales son las fórmulas recursiva a partir de la cual obtenemos la distancia y caminos mínimos en un grafo polietápico?. Indica lo que significa cada uno de los elementos de las fórmulas.

$$c(A,Y) = \min_{X \in E_i} \{c(A,X) + d(X,Y)\}$$

,siendo $c(A,Y)$ el camino mínimo entre el nodo origen A y un nodo independiente Y perteneciente a una etapa cualquiera E_{i+1}, c(A,X) el camino mínimo entre el nodo X y el nodo X, y d(X,Y) el coste del lado (X,Y).

15. Describe como se puede resolver el problema del viajante de comercio mediante programación dinámica. Usa para ello la fórmula recursiva general que lo resuelve.

Construyendo la solución al problema a partir de las soluciones óptimas para problemas más pequeños.

La solución al problema se puede representar mediante la siguiente fórmula recursiva:

$d(n_1, N-(n_1) = min_{2 \leq k \leq n}(L_{n_1,n_k} + d(n_k, N-(n_1,n_k)))$

16. ¿Cuantos posibles caminos hay que evaluar en el problema del viajante de comercio?: ¿De qué orden de complejidad es el algoritmo del viajante de comercio basado en programación dinámica?. Justifica tu respuesta.

El algoritmo del viajante de comercio basado en programación dinámica tiene un orden de complejidad exponencial, pues el número de conjuntos posibles que se podría obtener de un conjunto de n elementos es de 2^n.

17. Escribe la fórmula recursiva general para obtener la aproximación poligonal óptima, usando programación dinámica, indicando lo que significa cada uno de sus elementos.

$$E(n,m) = \min_{m-1 \leq j \leq n-1} \{E(j, m-1) + e(P_j, P_n)\}$$

E(n,m) es el error acumulado ISE cuando se aproximan los n primeros puntos de la curva con m puntos.

E(j,m-1) es el error acumulado ISE cuando se aproximan los j primeros puntos de la curva con m-1 puntos.

e(Pj,Pn) es el error acumulado ISE del segmento que une el punto j con el n.

18. ¿Cómo se pueden obtener los puntos de la aproximación poligonal óptima a partir de la fórmula general cuando se usa programación dinámica?.

Usando una matriz M donde el elemento M[i,j] contendrá al punto que precede al punto i, cuando éste ocupa la posición j en la aproximación poligonal, de modo que el valor almacenado es el índice que minimiza la expresión recursiva.

19. ¿Cual es el orden de complejidad del algoritmo para obtener la aproximación poligonal óptima en el caso de una curva cerrada y en el caso de una curva abierta?. ¿A qué se debe la diferencia entre ambas complejidades?.

Cuando la curva es abierta, el orden de complejidad del algoritmo es $O(M * N^2)$; mientras que cuando la curva es cerrada, el orden de complejidad del algoritmo es $O(M * N^3)$.

Esta diferencia de complejidades se debe a que para obtener el óptimo en una curva

cerrada, debemos aplicar el algoritmo tomando los N puntos de la curva como puntos de comienzo.

20. ¿Podríamos mejorar la complejidad del algoritmo de la aproximación poligonal óptima usando alguna estrategia de poda?.

La complejidad computacion del algoritmo de la aproximación poligonal óptima no puede ser mejorada.

Capítulo 2

Cuestiones Backtracking

1. Describe el método general del bactracking.

El método general del bactracking consiste en obtener una solución, que se puede expresar como una tupla, que satisfaga unas restricciones y tal vez que optimice cierta función objetivo.

2. ¿Cuales son los dos aspectos más importantes que influyen en la eficiencia de un algoritmo que se resuelva usando el método del Backtracking?.

1. El conjunto de partida, con las posibles soluciones del problema, debe ser del menor tamaño posible.

2. La complejidad de la prueba o condición que se utilizará para detectar nodos fracaso.

3. Describe el algoritmo de las n-reinas usando un tablero de 4x4.

El algoritmo de las n-reinas consiste en colocar en el tablero de 4*4 un número de 4 reinas (el tablero es de n*n, siendo n el número de reinas a colocar), de forma que ninguna reina amenace a las demás.

De este modo, si se han colocado todas las reinas en el tablero y ninguna de ellas es amenazada por otra reina, hemos encontrado la solución al algoritmo. En caso contrario, el algoritmo no tiene solución.

4. ¿Cómo modificarías el algoritmo de las n reinas si en vez de colocar reinas colocásemos torres (las torres mueven en horizontal y vertical)?. ¿Cuantas soluciones tendría?

Modificando la condición de lugar, de modo que una posición del tablero solo es válida cuando no hay otra reina en la misma columna o fila.

El número de soluciones que tendría sería: n!

5. ¿Cómo modificarías el algoritmo de las n-reinas si la reina también tuviese el movimiento del caballo?. En ese caso, el número de soluciones ¿sería mayor o menor que el número de soluciones del problema original?. Justifica esta última respuesta.

Para hacer que la reina también tuviese el moviente de caballo, añadiría la condición de que una reina no puede estar posicionada, con respecto a otra reina ya colocada, en las siguientes posiciones: (i-2,j-1) o (i-2,j+1) o (i+2,j-1) o (i+2,j+1).

En este caso, el número de soluciones con respecto al problema original es menor, pues el número de posiciones válidas para colocar una reina disminuye.

6. ¿Cual es el orden de complejidad del algoritmo de las n reinas?. Justifica tu respuesta.

El orden de complejidad del algoritmo de las n-reinas es de complejidad polinómica determinista, pues se comprueban en total n! posibles soluciones.

7. Deducir la condición para que las reinas no se apunten en diagonal

Suponiendo que 2 reinas se colocan en escaques (i,j),(k,l), no ocupan la misma diagonal si i-j != k-l o i+j != k+l, de las cuales podemos deducir que j-l != i-k y j-l != k-i, de donde se desprende que |j-l| != |k-i|

8. ¿Cuántas posibles soluciones explora el algoritmo de las n reinas para un tablero de 6x6?

Sólo se ensayan 6! posibles soluciones.

9. Describe brevemente como funciona el algoritmo que resuelve el problema de la suma de subconjuntos usando Backtracking.

El algoritmo backtracking para resolver el problema de la suma de subconjuntos consiste en construir un árbol (genera las soluciones), de modo que para un nodo X(i) cualquiera de nivel i, el hijo izquierdo se corresponderá con X(i) = 1 y el derecho con X(i) = 0.

10. En el problema de la suma de los subconjuntos, ¿qué condiciones se han de dar inicialmente para que el problema pueda tener solución?. Nota: Si estas condiciones no se diesen no habrá que explorar nada

1. La suma de los candidatos ya seleccionados más los que quedan por seleccionar ha de ser como mínimo M.

2. La suma de los valores de los candidatos ya seleccionados con el valor del más pequeño de los candidatos que queda por seleccionar no puede ser mayor a M.

11. En el problema de la suma de los subconjuntos, ¿qué condiciones se han de cumplir para que a los k primeros candidatos se les pueda añadir al candidato k+1 para formar parte de la solución?.

1. La suma de los candidatos ya seleccionados más los que quedan por seleccionar ha de ser como mínimo M.

2. La suma de los valores de los candidatos ya seleccionados con el valor del más pequeño de los candidatos que queda por seleccionar no puede ser mayor a M.

12. ¿Cual es el orden de complejidad del algoritmo de la suma de subconjuntos?. Justifica tu respuesta.

En el algoritmo de la suma de subconjuntos, se generan como máximo $2^n - 1$ nodos. Por tanto, el orden de complejidad del algoritmo es $O(2^n)$

13. ¿Se puede reducir el orden de complejidad del algoritmo de la suma de los subconjuntos si mejoramos la condición de poda?. Justifica tu respuesta.

Si, porque al mejorar la condición de poda se reduce el número de nodos generados para la construcción del árbol.

14. Describe brevemente como funciona el algoritmo que resuelve el problema de los ciclos hamiltonianos usando Backtracking.

Comenzando en el nodo inicial, se comprueba si el siguiente nodo al nodo actual es un nodo candidato de la solución. De este modo, se aplica un algoritmo recursivo que aplica este método hasta que no se encuentre nodos candidatos o se haya encontrado un ciclo hamiltoniano.

15. En el problema de los ciclos hamiltonianos para un grafo de n nodos. Si sabemos que existen soluciones, y consideramos soluciones distintas si hay una alteración en el orden de visita de los nodos ¿cual sería el número mínimo de soluciones que se pueden obtener?.

16. ¿Cómo adaptarías el algoritmo para obtener los ciclos hamiltonianos para resolver el problema del viajante de comercio?.

Guardando el coste mínimo de entre todos los ciclos calculados.

17. Indica alguna idea que sirva para reducir el número de caminos a explorar en la solución del problema del viajante de comercio basada en la obtención de los ciclos hamiltonianos.

Aplicando un algoritmo de poda que guardase el ciclo de menor coste encontrado hasta el momento, y en caso de que un camino en exploración tuviera un coste peor que el mejor encontrado hasta dicho momento, se interrumpiría la exploración de ese camino.

18. ¿Se puede reducir el orden de complejidad del algoritmo del viajante de comercio usando ciclos hamiltonianos si usamos un algoritmo de poda?. Justifica tu respuesta.

No, porque en ningún caso la poda podría reducir la complejidad computacional del algoritmo.

19. Describe brevemente como funciona el algoritmo de la mochila en su versión del Bactracking

El algoritmo de la mochila usando bactracking funciona insertando materiales en la mochila, con la condición de que el volumen del material al insertarlo en la mochila, no se supere el volumen total de la mochila.

20. ¿Cual es la condición de poda que se utiliza para resolver el problema de la mochila por el método del backtracking?. ¿Cómo hay que organizar los datos para poder aplicar dicha condición?.

El límite superior a la mejor solución que se pueda obtener no supere al valor de la mejor solución determinada hasta ese momento.

Para aplicar dicha condición, los datos deben estar ordenados en orden descendiente de precios.

21. ¿Cual es el orden de complejidad del algoritmo de la mochila resuelto por backtracking?. Justifica tu respuesta.

Debido a que el árbol de búsqueda de soluciones nos daría 2^n posibles soluciones, el orden de complejidad del algoritmo es exponencial.

22. ¿Se puede reducir el orden de complejidad del algoritmo de la mochila resuelto por backtracking si mejoramos la condición de poda?. Justifica tu respuesta.

No, porque la poda no reduce el orden de complejidad computacional del algoritmo.

23. ¿Cómo se obtiene el valor del límite para la poda en el problema de la mochila usando el método del backtracking?.

Se insertan los materiales restantes más caros hasta llegar al volumen de la mochila, pudiendo introducir el material por completo (si el volumen del material no supera el volumen restante de la mochila) o el material i parcialmente (si el volumen del material supera el volumen restante de la mochila).

24. De los tres algoritmos usados para resolver el problema de la mochila (voraz, programación dinámica y bactracking), ¿cual de los tres proporciona una solucion de más valor para la mochila?. Justifica tu respuesta

El algoritmo bactracking porque permite seleccionar los materiales parcialmente, permitiendo obtener mejores soluciones.

Capítulo 3

Cuestiones algoritmos probabilistas

1. ¿Por qué los algoritmos probabilistas se pueden comportar de forma distinta cuando se aplican más de una vez con los mismos datos de entrada?

Porque puede ocurrir que al afrontar una decisión posible entre varias posibilidades, sea preferible seleccionar aleatoriamente alguna de dichas posibilidades.

2. ¿Cuáles son los algoritmos probabilistas que no garantizan la corrección del resultado y cuáles son los que garantizan la corrección?

No garantizan la corrección los algoritmos numéricos y los algoritmos de Montecarlo.

Garantizan la corrección los algoritmos Las Vegas.

3. Enumera dos ventajas que pueda tener un algoritmo probabilista frente a un algoritmo determinista

1. Los algoritmos probabilistas son, en general, más eficientes.
2. Al calcular integrales múltiples, en los algoritmos probabilistas el coste crece exponencialmente con la dimensión del espacio, permitiendo mantener la precisión.

4. ¿Qué diferencia existe a la hora de medir el tiempo medio empleado en un algoritmo determinista y uno probabilista?

En un algoritmo determinista el tiempo medio promedio puede incurrir a error si todos los casos no son equiprobables.

5. Supongamos que se implementan tres algoritmos probabilistas para determinar el valor del número pi con 4 decimales de aproximación (3.1416). El primer de ellos es numérico, el segundo es de Montecarlo y el tercero es de Las Vegas. Si cada uno de ellos se ejecuta 5 veces, escribe 5 soluciones posibles que se pudiesen obtener con dichos algoritmos

Algoritmo numérico: Entre 3.1410 y 3.1419,entre 3.1411 y 3.1420,entre 3.1413 y 3.1421, entre 3.1414 y 3.1423, entre 3.1415 y 3.1424

Algoritmo Montecarlo: 3.1416,3.1415,3.1416,3.0000,3.1416

Algoritmo Las Vegas: 3.1416,1.1416,error,3.1416,error

6. ¿Cómo varía el error de un algoritmo probabilista como el de integración numérica, frente al tiempo empleado por el mismo?

El error suele ser inversamente proporcional a la raíz cuadrada de la cantidad de tiempo empleado.

7. ¿Cómo se puede hacer uso del experimento de la aguja de Buffon para obtener un valor aproximado del número pi?

Si lanzamos N agujas y, suponiendo que n agujas han caído sobre 2 planchas, tendremos que $\pi = \frac{N}{n}$

8. Describe el algoritmo probabilista para resolver el problema de la integración numérica. ¿Qué ventajas tendría frente a un método numérico como el de los trapecios?

Sea y=f(x) una función continua en el intervalo [a,b] y de valor positivo, se realiza un número de N veces la suma de f(x) para x un valor aleatorio entre a y b. una vez calculada la suma total, se calcula la superficie generada por la función.

Ventajas frente a un método numérico:

1. Es más eficiente a la hora de obtener una integral múltiple.

9. ¿Cuando puede ser ventajoso el algoritmo probabilista para la integración numérica?. Justifica tu respuesta.

Al evaluar integrales de 4 o más dimensiones.

10. ¿Cuánto habría que aumentar el número de simulaciones para aumentar la precisión de un algoritmo probabilista en una décima?

Habría que aumentar 100 veces más el número de simulaciones.

11. ¿Proporcionan los algoritmos de Montecarlo una respuesta correcta?. Justifica tu respuesta.

Pueden proporcionar una respuesta correcta o no, porque pueden cometer ocasionalmente un error.

12. ¿Cuándo se dice que un algoritmo de Montecarlo es p-correcto?

Si devuelve una respuesta correcta con una probabilidad no menor que p, sea cual sea el caso considerado.

13. ¿En qué consiste la amplificación de la ventaja estocástica?. ¿En qué tipo de algoritmos se emplea?

Consiste en reducir arbitrariamente la probabilidad de error a costa de un ligero aumento del tiempo de cálculo.

Se emplea en algoritmos probabilistas de Montecarlo.

14. ¿Porqué hay que realizar muchas pruebas en el algoritmo de Montecarlo de la multiplicación de matrices?

Porque permite mejorar la fiabilidad del algoritmo.

15. Supongamos que en el algoritmo de Montecarlo de la multiplicación de matrices se realizan 50 pruebas y en todas ellas se verifica el producto. ¿Qué conclusión se puede sacar?

Se ha obtenido una respuesta correcta con probabilidad superior a 0.999

16. En el algoritmo para verificar el producto de matrices basado en el método de Montecarlo, si se realizan n pruebas y la respuesta es siempre correcta ¿podemos afirmar que se verifica el producto de matrices?. Justifica tu respuesta.

Sí, porque el decrecimiento del error es proporcional al número de llamadas del algoritmo.

17. ¿Porqué los algoritmos de las Vegas nunca proporcionan una respuesta errónea?

Porque los algoritmos de Las Vegas exigen que la probabilidad de éxito para un caso cualquiera sea mayor que 0.

18. Describe el algoritmo probabilista para resolver el problema de las n-reinas.

Se colocan aleatoriamente reinas en filas sucesivas, teniendo en cuenta que no se puede poner una que sea amenazada por una reina colocada previamente, de modo que el algoritmo tendrá éxito cuando se ubican las n reinas o fracasará cuando llegue a un callejón sin salida.

19. ¿Cómo implementarías un algoritmo las Vegas para obtener un ciclo Hamiltoniano?

Dado un nodo actual, se indica dicho nodo como visitado y se calcula de los nodos conectados al nodo actual, aquel cuya distancia entre el nodo actual y el siguiente nodo sea menor. Obtenido dicho nodo, se marca dicho nodo como visitado y se repite el proceso de búsqueda del siguiente nodo.

De este modo, el algoritmo tendrá éxito si se obtiene un camino hamiltoniano, o fracasará

si encuentra un camino sin salida.

20. ¿Son mejores en tiempo promedio los algoritmos Las Vegas que los deterministas?. Justifica tu respuesta

No, porque aquellos casos que requieren un tiempo mínimo con el algoritmo determinista se enlentecen hasta un valor medio cuando se usa un algoritmo de Las Vegas.