

Algoritmos de Vuelta Atrás

- Planteamiento General
- Ejemplos de Aplicación
 - Problema de las 8 reinas
 - Problema de la suma de subconjuntos
 - Problema del coloreado de grafos
 - Problema de la mochila 0/1
 - Problema de los ciclos hamiltonianos

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Planteamiento General

- Es uno de los métodos más clásicos (50's) y aplicables
- Aplicable a problemas que cumplen
 - La solución se puede expresar como una n-tupla $X=(x_1, \dots, x_n)$ donde cada x_i se escoge de un *conjunto finito* S_i
 - La solución(es) es(son) el vector(es) que optimiza(n) o *satisface(n) una función criterio* $C(x_1, \dots, x_n)$

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Planteamiento General

- La Vuelta Atrás es una combinación de:
 - Fuerza bruta
 - Generar todos los candidatos a solución
 - Uso de heurísticas
 - Introducir información que reduzca el número de candidatos que pueden dar lugar a una solución
- Se consigue reducir en un buen número la cantidad de candidatos a solución generados

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Planteamiento General

- Se generan los vectores candidatos componente a componente
- En cada paso, se aplica una función de limitación para decidir si el candidato que se está construyendo puede ser solución
- Ejemplos de Aplicación
 - Problema de la mochila 0/1
 - Problema de las 8 reinas

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Problema de la mochila 0/1

- Estructura de la solución
- $x(i) \in \{0, 1\}$ vs. $\in \{1..n\}-\{x(1)..x(i-1)\}$
- Función criterio

$$C(X) \equiv (\sum x(i)w(i) \leq M) \wedge \\ (\forall Y / \sum y(i)w(i) \leq M \rightarrow \\ \sum y(i)p(i) \leq \sum x(i)p(i))$$

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Problema de la mochila 0/1

- Se obtiene una estructura de árbol de posibilidades => estados candidatos y solución
- Se corresponde de manera natural con una exploración en profundidad => estructura recursiva
- Conviene descartar candidatos antes de generarlos => funciones de limitación

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Problema de las 8 reinas

- Problema
 - Colocar 8 reinas en un tablero de ajedrez de manera que no se amenacen entre sí
 - Dos reinas se amenazan si se hallan en la misma fila (restricción 1), columna (2) o diagonal (3)
 - El problema se generaliza a colocar n reinas en un tablero de $n \times n$ casillas

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Problema de las 8 reinas

- La colocación de una reina consiste en la selección de un par $X_k=(i,j)$
 - Representa la disposición de la reina k en la casilla de fila i y columna j , $1 \leq i,j,k \leq n$
- El conjunto de candidatos S está formado por todas las tuplas de n pares distintos dos a dos
 - Dos reinas no pueden estar en la misma casilla

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Problema de las 8 reinas

- La función criterio determina que las filas, columnas y diagonales no pueden coincidir en distintos pares x_k y x_k'
 - Las tres condiciones se comprueban a posteriori, generándose las n^2 posibilidades para x_1 , n^2-1 para x_2 , etc
 - Inicialmente, para el caso de $n=8$, el cardinal de S es de aproximadamente 22 billones de tuplas

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Problema de las 8 reinas

- Como dos reinas no se pueden colocar en la misma fila, se puede decidir que la reina k se dispone en la fila k (restricción 1 a priori)
 - x_i es la columna en la que se coloca la reina i
 - Cada x_i puede tomar un valor entre 1 y n , con lo cual existen n^n posibles combinaciones (16 millones de tuplas para $n=8$).
- Si además se impone que las columnas no coincidan, se están generando las permutaciones de $\{1, \dots, n\}$ (restricciones 1 y 2 a priori)
 - El número de tuplas o candidatos generados es de $n!$ (40320 tuplas para $n=8$)

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Problema de las 8 reinas

- Existen 2 tipos de restricciones
 - Consideradas a priori, o explícitas
 - Definen los S_i
 - Consideradas a posteriori, o implícitas
 - Definen la función criterio (y las funciones de limitación)
- Las tuplas que cumplen las restricciones explícitas constituyen el espacio de soluciones

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Generación del conjunto de soluciones

- El método de vuelta atrás se basa en generar sistemáticamente el conjunto de candidatos a soluciones
- Los candidatos a soluciones se construyen eligiendo valores para los X_i sucesivamente
- El espacio de soluciones tiene la estructura natural de un árbol de búsqueda

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Generación del conjunto de soluciones

- A medida que se eligen los valores de las X_i , se pasa a un nuevo estado de construcción de un candidato a solución
 - Los estados se pueden representar por medio de nodos
 - Los arcos entre los nodos representan elecciones de valores para algún X_i
- Ejemplo n reinas => *árbol permutacional*
- Ejemplo mochila 0/1 => *árbol combinatorial*

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Definiciones

- Cada nodo define un *estado del problema*
- El conjunto de candidatos S se organiza en forma de árbol en el que los caminos desde la raíz a los nodos representan tuplas de S
- El conjunto de todos los caminos desde la raíz del árbol de estados constituye el *espacio de estados*
- Los *estados solución* son aquellos tales que el camino de la raíz al estado representa una tupla del espacio de soluciones (un *candidato a solución*)
- Los *estados respuesta* son aquellos estados solución que cumplen las restricciones implícitas del problema y definen por tanto una *solución* del problema

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Elementos de la exploración

- Exploración en anchura o en profundidad
 - El método de vuelta atrás se corresponde de manera intuitiva más con el recorrido en profundidad
- Exploración en anchura o en profundidad
 - El método de vuelta atrás se corresponde con *una exploración organizada del espacio de estados con ayuda de funciones de limitación*
 - La diferencia principal entre la vuelta atrás y la fuerza bruta es el uso de las funciones de limitación

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Elementos de la exploración

- Se puede realizar una exploración en anchura
- Ventajas
 - Los candidatos a solución pueden tener infinitos elementos siempre que la solución sea finita
 - Si interesa minimizar el número de elementos de la solución, es más eficientes (se exploran menos estados)
- Desventaja
 - Más memoria y complicación al programar
 - Recursión infinita si $|S_i| = \infty$

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Algoritmo recursivo

- Sólo puede corresponder a una búsqueda en profundidad

```
procedimiento Vuelta_atrás1(i, X : matriz[1..n] de valor_x)
{Convención: T(X[1],...,X[n]) = ∅ para toda tupla X, es decir, no se
pueden elegir valores para X[n+1]}
para cada X[i] / (X[i] ∈ T(X[1],...,X[i-1])) y Bi(X[1],...,X[i])
  {es decir, se toma X[i] cada valor posible según los valores de
X[1],...,X[i-1] que puede dar lugar a un candidato a solución}
  hacer si (X[1],...,X[i]) ∈ EstadosSolución
    entonces escribir(X[1],...,X[i]) {se encuentra una solución}
  Vuelta_atrás1(i + 1, X) {búsqueda en profundidad}
```

- Si se quiere obtener sólo una solución, se añade un parámetro a la cabecera

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Algoritmo iterativo

```
procedimiento Vuelta_atrás2(n)
{Convención: T(X[1],...,X[n]) = ∅ para toda tupla X, es decir, no se pueden
elegir valores para X[n+1]}
var X: matriz [1..n] de valor_x
i = 1
mientras i > 0
  hacer si ∃ X[i] / (X[i] ∈ T(X[1],...,X[i-1])) y (no(marcado(X[i])) y Bi(X[1],...,X[i])
  {es decir, se toma si existe X[i] un valor posible según los valores de
X[1],...,X[i-1], no explorado previamente y que puede dar lugar a un
candidato a solución}
  entonces
    marcar(X[i])
    si (X[1],...,X[i]) ∈ EstadosSolución
      entonces escribir(X[1],...,X[i]) {se encuentra una solución}
      i = i + 1 {búsqueda en profundidad}
    sino i = i - 1 {vuelta atrás}
```

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Complejidad

- La eficiencia general del esquema de vuelta atrás depende de cuatro factores principales
 - El tiempo necesario para la generación de un x_i (ejecutar $T(x_1, \dots, x_{i-1})$)
 - El número de x_i que se pueden generar (es decir, $|S_i|$)
 - El coste de ejecución de las funciones de limitación B_i
 - El número de x_i que cumplen las restricciones implícitas, esto es, las B_i
- Se obtiene siempre un tiempo $O(p(n) \cdot a^n)$ o superior (con $p(n)$ un polinomio)

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Complejidad

- La eficiencia depende en realidad del número de estados explorados realmente => función de limitación
- Compromiso entre nodos generados y coste de generación de los nodos!!
- Mediremos la eficiencia usando un método teórico/empírico

Planteamiento General

Complejidad

- El número de nodos generados depende en general del ejemplar de problema considerado
- Para lograr una aproximación del coste de un algoritmo de tipo vuelta atrás, se suele emplear el método de *Monte Carlo*
 - Obtener un camino aleatorio en el árbol de estados
 - Sumar todas las posibles alternativas no tomadas como estimación del máximo número de nodos explorados

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Complejidad

- Para aplicar el método de Monte Carlo, se deben realizar dos suposiciones
 - Si sólo se desea obtener una solución, la estimación obtenida es una cota superior que puede estar muy alejada del coste real
 - Se emplea la misma función de limitación B_i para todos los nodos del mismo nivel
- El método de Monte Carlo genera un camino aleatorio, eligiendo al azar uno de los posibles hijos de cada nivel

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Complejidad

- Algoritmo que estime le número m de nodos que se generan.

función Estimar()

var S : conjunto de valor_x

var X : matriz [1..n] de valor_x

r = 1 {Estados en este nivel}

m = 1 {Estados acumulados hasta este nivel}

i = 1

repetir

S = {X[i] ∈ T(X[1],...,T[i-1]) : B_i(X[1],...,X[i])}

r = r * |S|

m = m + r

X[i] = azar(S) {Elige un valor del conjunto C al azar}

i = i + 1

hasta S = ∅

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Planteamiento General

Complejidad

- Resulta más acertado ejecutar este algoritmo varias veces y calcular la media de los valores obtenidos
 - Es de esperar que cada ejecución del algoritmo genere por un camino distinto

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Problema de las 8 reinas

- Problema
 - Colocar 8 reinas en un tablero de ajedrez de manera que no se amenacen entre sí
 - Dos reinas se amenazan si se hallan en la misma fila, columna o diagonal
 - Las casillas de una diagonal “\” cumplen que el valor de la diferencia fila-columna es idéntico
 - Las casillas de una diagonal “/” cumplen que el valor de la suma fila+columna es el mismo
 - Las reinas de las casillas (i,j) y (k,l) están en la misma diagonal si y sólo si $|j-i|=|k-l|$

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Problema de las 8 reinas

- Pseudocódigo
 - Se desarrolla una función booleana *colocar* que devuelve cierto si se puede colocar la reina késima en la posición *pos*

función colocar(pos, k, X : matriz[1..n] de entero)

 i = 1

mientras i < k

hacer si (X[i] = pos) o (|X[i] - pos| = |i - k|)

entonces devolver falso

 i = i + 1

devolver cierto

Problema de las 8 reinas

- El procedimiento *reinas* obtiene *todas* las posibles colocaciones de las n reinas en un tablero de $n \times n$

```
procedimiento reinas(n)
  var X : matriz[1..n] de entero
  X[1] = 0
  k = 1
  mientras k > 0
    hacer repetir {dar valores a X[k]}
      X[k] = X[k] + 1
    hasta (X[k] > n) o (colocar(X[k], k, X))
    si (X[k] ≤ n) {se puede colocar la reina késima}
      entonces
        si (k = n) {solución completa}
          entonces escribir(X)
        si no k = k + 1
          X[k] = 0
        si no k = k - 1 {vuelta atrás}
```

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Problema de las 8 reinas

- Complejidad
 - Función Colocar: $O(k-1)$
 - Función Reinas: n^n posibilidades con coste $k-1 \Rightarrow n^{n+1}$
 - Estimación del porcentaje del n° de nodos generados
 - Máximo n° de nodos sin f. de limitación

$$m = 1 + \sum_{j=0}^7 \left(\prod_{i=0}^j (8-i) \right) = 69.281$$

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Problema de las 8 reinas

- Supongamos que se ejecuta el algoritmo estimar

Camino	Vector de posibilidades no limitadas	Coste total
(2,4,1,3,5)	(8,5,4,3,2)	1649
(4,6,3,5,7,1)	(8,5,3,1,2,1)	769
(1,8,6,3,7,2,4)	(8,6,4,2,1,1,1)	1401
(1,3,5,2,4)	(8,6,4,3,2)	1977
(3,6,2,7,1,4,8,5)	(8,5,3,2,2,1,1,1)	2329

- Media de las 5 ejecuciones: 1625 (2,34%)

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Problema de la suma de subconjuntos

- Problema
 - Dados n números positivos en un conjunto W , encontrar todos los subconjuntos cuya suma es el número positivo M
 - Se parte de la formulación en la que $x_i=0$ si no se toma el i ésimo elemento de W , y $x_i=1$ si se toma
 - Asumimos que los elementos de W vienen dados por orden creciente

Problema de la suma de subconjuntos

– La función de limitación se evalúa como

$$B_k^1(X(1), \dots, X(k)) \Leftrightarrow \sum_{i=1}^k W(i) \cdot X(i) + \sum_{i=k+1}^n W(i) \geq M$$
$$B_k^2(X(1), \dots, X(k)) \Leftrightarrow \sum_{i=1}^k W(i) \cdot X(i) + W(k+1) \leq M$$
$$B_k(X(1), \dots, X(k)) \Leftrightarrow B_k^1(X(1), \dots, X(k)) \wedge B_k^2(X(1), \dots, X(k))$$

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Problema de la suma de subconjuntos

- Pseudocódigo (Esquema recursivo)

```
procedimiento sumasub(W : matriz[1..n] de valor, X: matriz[1..n] de 0..1,
                    k, Suma_Parcial, Resto_Disponible)
X[k] = 1 {Primera posibilidad: se toma el elemento}
si ((Suma_Parcial + W[k]) = M) {Encontrada solución}
  entonces escribir(X, k) {Escribir hasta la variable késima}
si no si ((Suma_Parcial + W[k] + W[k+1]) ≤ M) {Se cumple Bk}
  entonces sumasub(W, X, k+1, Suma_Parcial + W[k],
                  Resto_Disponible – W[k])
{Segunda posibilidad: no se toma el elemento}
si ((Suma_Parcial + Resto_Disponible – W[k]) ≥ M) y
  ((Suma_Parcial + W[k+1]) ≤ M) {Se cumple Bk}
  entonces X[k] = 0
sumasub(W, X, k+1, Suma_Parcial, Resto_Disponible - W[k])
```

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Problema de la suma de subconjuntos

- Complejidad

- Complejidad teórica (caso peor: f. de limitación no eliminan ningún nodo de S)

$$t(k) = \begin{cases} 2 & \text{si } k = 0 \\ 2t(k-1) + 3 & \text{si } k > 0 \end{cases}$$

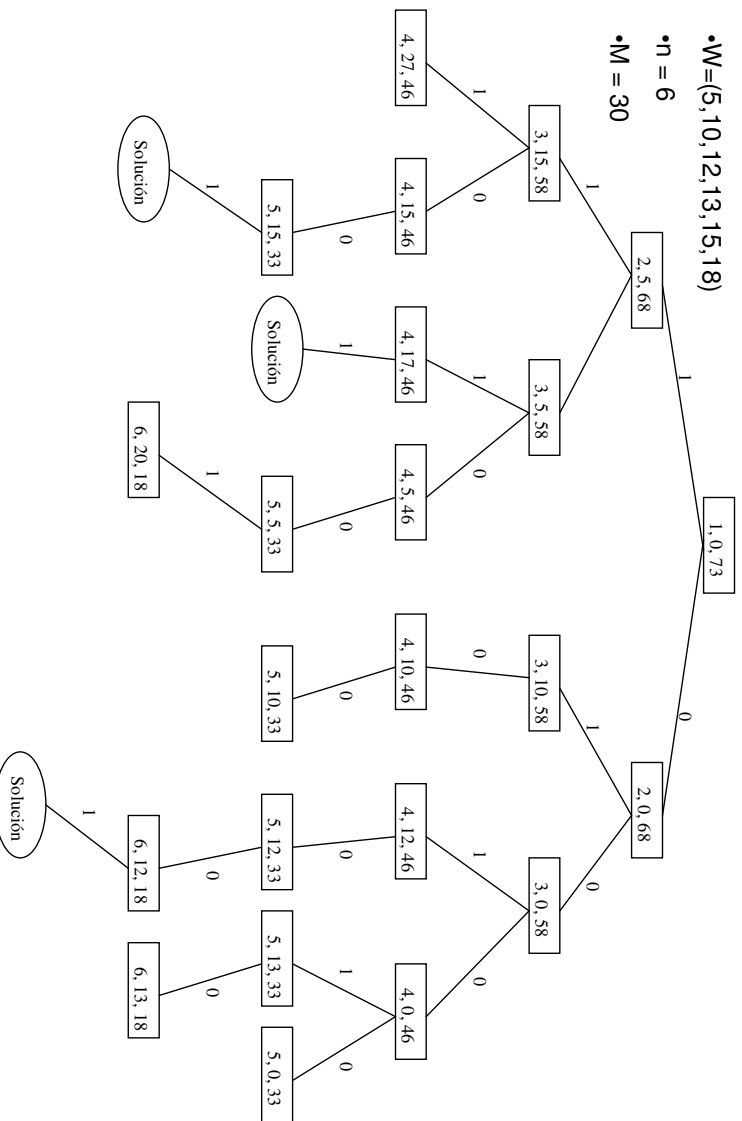
- Obtenemos $t(n) \in O(2^n)$

- Estudio empírico

- Ejecución sobre un ejemplar del problema
- Cálculo de los nodos explorados frente al número potencial de nodos a explorar

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Problema de la suma de subconjuntos



Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Problema del coloreado de grafos

- Problema
 - Dado un grafo G y un número m de colores, asignar dichos colores a los vértices del grafo de modo que dos vértices adyacentes no tengan el mismo color
 - Coloreado de mapas \Rightarrow grafos planos
 - Colores: enteros de 1 a m . Vértices: enteros de 1 a n
Definimos x_i como color del vértice i
 - El espacio de soluciones es $S = \{(x_1, \dots, x_n) : x_i \in \{1, \dots, m\}\}$
 - F. de limitación: Se pueden etiquetar dos vértices con el mismo color si no son adyacentes

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Problema del coloreado de grafos

- Pseudocódigo (Procedimiento Siguiente)

procedimiento siguiente(G: grafo, X: matriz[1..n] de 0..m, k, m)

{Se supone que a $X(k)$ ya se le ha asignado algún color previamente}

repetir

$X[k] = (X[k] + 1) \bmod (m + 1)$

{Se asigna el siguiente valor}

si $X[k] \neq 0$ {No se han agotado los colores}

entonces coincide = falso

para $i = 1$ hasta $k-1$

hacer coincide = coincide o $(X[i] = X[k])$ y $G.A[i, k]$

{Si coincide = cierto, es que algún vértice adyacente

(con $G.A[i, k]$) posee el mismo color (es decir, $X[i] = X[k]$)}

hasta $(X[k] = 0)$ o no(coincide)

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Problema del coloreado de grafos

- Pseudocódigo (Esquema recursivo)

```
procedimiento colorear(G: grafo, X: matriz[1..n] de 0..m, k, m)
{ Se inicializa X(k) a 0 y se le van asignando valores}
  X[k] = 0
  repetir
    siguiente(G, X, k, m) {Se asigna el siguiente valor}
    si X[k] ≠ 0 {No se han agotado los colores}
      entonces si (k = n)
        entonces escribir(X) {Solución encontrada}
        si no colorear(G, X, k+1, m)
      hasta (X[k] = 0)
```

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Problema del coloreado de grafos

- Complejidad
 - Complejidad teórica (caso peor: f. de limitación no eliminan ningún nodo de S)
 - En cada nivel i hay un número m^{i-1} de nodos, y en cada nivel se realizan i-1 comparaciones por nodo.

K	Nodos	Comparaciones totales	Comparaciones por nodo
1	m	0	0
2	m^2	1	m^2
3	m^3	2	$2m^3$

$$t(n) = m^2 + 2m^3 + \dots + (n-1)m^n = \sum_{k=1}^n (k-1)m^k \leq \sum_{k=1}^n nm^k = n \frac{m^{n+1} - 1}{m - 1}$$

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Problema del coloreado de grafos

- Obtenemos $t(n) \in O(n \cdot m^n)$
- Ejemplo:

Camino	Vector de posibilidades no limitadas	Coste total
(2,3,2,1)	(3,2,2,2)	46
(1,2,3,2)	(3,2,2,1)	34
(3,1,3,1)	(3,2,2,2)	46
(2,1,3,1)	(3,2,2,1)	34

- Se obtiene una estimación de 40 nodos

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Problema de los ciclos hamiltonianos

- Problema

- Un ciclo hamiltoniano es un camino en G (grafo conexo con n vértices) que pasa por cada vértice exactamente una vez y vuelve a la posición inicial.
- Se define x_i como el i ésimo vértice en un ciclo hamiltoniano.
- Asumimos que el primer vértice, x_1 , es siempre 1.
- El espacio de soluciones es $S = \{(x_1, \dots, x_n) : x_i \in \{1, \dots, n\}\}$
- F. de limitación:

$$B_k(X(1), \dots, X(k)) \Leftrightarrow G.A(X(k-1), X(k)) \wedge (X(k) \neq X(i), i = 1, \dots, k-1)$$

Algorítmica - José María Gómez Hidalgo, Francisco Carrero García

Problema de los ciclos hamiltonianos

- Pseudocódigo (Procedimiento Siguiente).

```
procedimiento siguiente(G: grafo, X: matriz[1..n] de 0..n, k)
{Se supone que a X[k] ya se le ha asignado algún vértice
previamente}
  repetir
    X[k] ← (X[k] + 1) mod (n + 1) {Se asigna el siguiente valor}
    si X[k] ≠ 0 {No se han agotado los vértices}
      entonces si G.A(X[k-1],X[k]) {Si están conectados}
        entonces válido ← falso
          para i ← 1 hasta k-1
            hacer válido ← válido y (X[i] ≠ X[k])
          {Si válido = cierto, es que X[k] es distinto de X[i]}
          para todo i = 1, ..., k-1
            hasta (X[k] = 0) o válido
```

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Problema de los ciclos hamiltonianos

- Pseudocódigo (Esquema recursivo).

```
procedimiento cicloshamilton(G: grafo, X: matriz[1..n] de 0..n, k)
{ Se inicializa X(k) a 0 y se le dan valores sucesivamente}
  X[k] ← 0
  repetir
    siguiente(G, X, k) {Se asigna el siguiente valor}
    si X[k] ≠ 0
      entonces si (k = n)
        entonces si G.A(X[n],1)
          entonces escribir(X) {Solución encontrada}
          si no cicloshamilton(G, X, k+1)
            hasta (X[k] = 0)
```

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Problema de los ciclos hamiltonianos

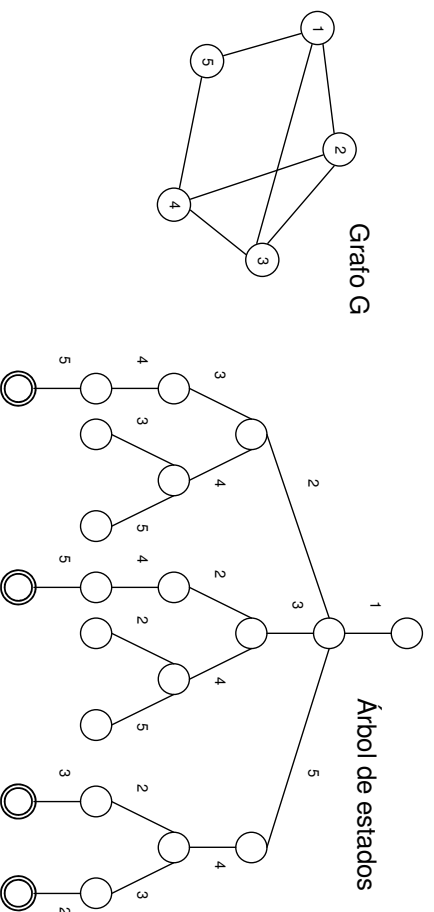
- Complejidad
 - Complejidad teórica (caso peor: grafo completamente conectado)
 - Se generan las $(n-1)!$ permutaciones posibles que comienzan en 1 .

K	Nodos	Comparaciones totales	Comparaciones por nodo
2	n	1	n
3	n ²	2	2n ²
4	n ³	3	3n ³

$$t(n) = n + 2n^2 + 3n^3 + \dots = \sum_{i=1}^n i \cdot n^i \leq \sum_{i=1}^n n^{i+1} = \frac{n^{n+2} - 1}{n - 1} \in O(n^{n+1})$$

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García

Problema de los ciclos hamiltonianos



Camino	Vector de posibilidades no limitadas	Coste total
(1,2,4,3)	(1,3,2,2)	23
(1,3,2,4,5)	(1,3,2,1,1)	23
(1,5,4,3,2)	(1,3,1,2,1)	20

Algorítmica - José maría Gómez Hidalgo, Francisco Carrero García