Tema 7 Algoritmos voraces.pdf

raulcb98

Fundamentos de análisis de algoritmos

1º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería
Universidad de Huelva

KEEP CALM AND ESTUDIA UN POQUITO

WUOLAH + #QuédateEnCasa

INGENIERÍA INFORMÁTICA

FUNDAMENTOS DE ANÁLISIS DE ALGORITMOS.
TEMA 7
ALGORITMOS VORACES

RAÚL CASTILLA BRAVO
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ETSI – HUELVA)
Curso: 1º

ÍNDICE

Tema 7. Algoritmos voraces

1. ESQUEMA Y FUNCIONAMIENTO

También conocidos como técnicas ávidas, es uno de los esquemas más simples para la resolución de problemas de optimización. En un algoritmo voraz se distinguen tres conjuntos de elementos:

C: Conjunto de candidatos pendientes de seleccionar (inicialmente todos).

S: Candidatos seleccionados para la solución.

R: Candidatos rechazados después de haber sido seleccionados.

1.1 Ejecución

1. Parte de una solución vacía.

2. Se escoge el mejor elemento del conjunto candidatos.

3. Se aplica un criterio de selección para comprobar que el elemento es válido.

   3.1 Si es válido, se añade al conjunto solución y ya no podemos sacarlo.

   3.2 Si no es válido se rechaza.

4. Se repite el proceso hasta tener una solución o no tener más candidatos.

1.2 Ejemplo en pseudocódigo

```
funcion voraz( C: CjtoCandidatos  var S: CjtoSolución )
    S ← Ø
    mientras (C ≠ Ø) Y NO solución(S) hacer
        x ← seleccionar(C)
        C ← C - {x}
        si  factible(S, x)  entonces
              insertar(S, x)
        finsi
    finmientras
    si NO solución(S) entonces
        devolver "No se puede encontrar solución"
    sino devolver S
    finsi
```

Funciones genéricas

1.- solución (S): comprueba si un conjunto de candidatos es solución independientemente de que sea óptima o no.

2.- seleccionar (C): escoge el elemento más "prometedor" del conjunto de candidatos.

3.- factible (S, x): indica si introduciendo x en el conjunto S es posible construir una solución.

4.- Insertar (S, x): añade el elemento x al conjunto solución.

5.- Función objetivo (S): dada una solución, devuelve el coste asociado a la misma (resultado del problema de optimización).

2. ANÁLISIS DE LOS TIEMPOS DE EJECUCIÓN

El orden de complejidad depende de:

-Número de candidatos (n).

-Número de elementos de la solución (m).

-Funciones básicas a realizar.

Como máximo, el bucle se ejecuta n veces y como mínimo m. Vamos a ver ahora el coste que tienen cada una de las funciones básicas:

Función básica	Orden	Nombre de la función
Solución (S)	Entre O(1) y O(m)	f(m)
Selección (C)	Entre O(1) y O(n)	g(n)
Factible(S,x)	Parecido a Solución(S) pero con una solución parcial.	h(n)
Insertar (S,x)	Depende de las operaciones de inserción.	j(n,m)
Tiempo de ejecución genérico	$t(n,m) \in O(n * \big(f(m) + g(n) + h(m)\big) + m * j(n,m))$	

Suelen ser rápidos y se encuentran en los órdenes polinomiales.

3. APLICACIONES DE LOS ALGORITMOS VORACES

3.1 Selección de actividades

Consiste en seleccionar un subconjunto de las actividades propuestas de manera que sean compatibles entre sí y sea un subconjunto de cardinal máximo de entre los posibles. Lo que conocemos en este tipo de problemas es:

N: número de actividades.

C_i: hora de comienzo.

F_i: hora de finalización.

(C_i , F_i): intervalo en el que se debe realizar la actividad.

Dos actividades son compatibles si los intervalos de tiempo no se superponen. El criterio de selección para estos problemas consiste en escoger la actividad que termine más pronto. De esta forma se obtiene la solución óptima.

3.2 Mochila fraccionaria

Consiste en llenar una mochila con fragmentos de objetos (n), de los que conocemos su peso (p_i) y su beneficio (b_i), sabiendo además que la mochila tiene un peso máximo M. El diseño del algoritmo voraz sería el siguiente:

- Candidatos: cada uno de los n objetos de partida.
- Función solución: tendremos solución se introducimos el peso máximo M.
- Función seleccionar: escoge el elemento con mayor relación beneficio/peso.
- Función factible: será siempre cierta porque podemos añadir trozos.
- Función añadir: se añade el objeto entero si cabe o una gracción.
- Función objetivo: maximizar $\sum_{i=1}^{n} x_i b_i$ bajo la restricción $\sum_{i=1}^{n} x_i p_i \leq M \; y \; 0 \leq x_i \leq 1$.