Algorithms for Problem Solving – 11650

Backtraking

Jon Ander Gómez Adrián
jon@dsic.upv.es

Departament de Sistemes Informàtics i Computació
Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

2 de marzo de 2014

Estudio del código base

- Generación de todas las permutaciones posibles
  - Ver código `AllPermutations.java`
- Generación de todos los subconjuntos posibles de un conjunto
  - Ver código `AllSubsets.java` y `AllSubsets2.java`
- Poda: ejemplo de las n-reinas en un tablero de $n \times n$
  - Ver código `NReinas.java`

110801/861

- Para este problema existe una solución aplicando una fórmula de conteo basada en combinatoria.
- El problema 10237 de la UVa contiene casos de prueba de este mismo problema que implican utilizar la solución combinatoria.
- Mi solución por rastreo exhaustivo no entra en tiempo.
- Pero sirve para generar la tabla de todas las posibles combinaciones de ancho de tablero (n) y número de alfiles (k).
- Matríz de $n \times n$ inicializada a ceros.
- Cada vez que se coloca un alfil se pone un -1 y se marcan las posiciones que bloquea incrementando su valor en 1.

  Cuando se quita para el *backtraking* se resta 1.

110802/10181

- Comprobar si es solucionable con una fórmula.
- Necesitamos definir una distancia desde una configuración dada a la configuración objetivo.
- La distancia se obtiene acumulando la distancia de *Manhattan* de cada pieza a la posición que debería tener.
- En cada descenso del *backtraking* expandir la configuración actual a las cuatro posibles con los movimientos: $\{\texttt{U},\texttt{D},\texttt{L},\texttt{R}\}$
- Se debe comprobar que no se visita una configuración dos veces, y que la heurística no sobrepasa un umbral que sirve para podar.

$$x \times \text{distancia recorrida} + y \times \text{estimación hasta el objetivo}$$

110803/10128

- Por *backtraking* da TLE.
- Por combinatoria es más rápido:

$N\ P\ R$		k	
7 4 1	4 5 6 1 2 3 7	3	$Q(3,3,1) * \binom{5}{2} * 3!$
	4 5 1 6 2 3 7	4	$Q(4,3,1) * \binom{5}{3} * 2!$
	4 1 2 5 6 3 7	5	$Q(5,3,1) * \binom{5}{4} * 1!$
	4 1 2 3 5 6 7	6	$Q(6,3,1) * \binom{5}{5} * 0!$
7 4 3	4 5 6 7 1 3 2	4	$Q(4,4,1) * \binom{6}{3} * Q(4,3,1)$
	4 5 6 1 7 3 2	5	$Q(5,4,1) * \binom{6}{4} * Q(3,3,1)$

110803/10128

- El problema es simétrico, asumiremos siempre $P \geq R$
- El máximo puede ir moviéndose desde P hasta $N - R + 1$
- Debemos precalcular todas las posibles combinaciones de N, P y R porque puede haber hasta 10000 casos de prueba.
- Para los casos tipo $N\ P\ 1$ (con $P \geq 1$)

$$Q(N, P, 1) = \sum_{k=P-1}^{N-1} Q(k, P-1, 1) * \binom{N-2}{k-1} * (N-k-1)!$$

- Para los casos tipo $N\ P\ R$ (con $P \geq R > 1$)

$$Q(N, P, R) = \sum_{k=P}^{N-R+1} Q(k, P, 1) * \binom{N-1}{k-1} * Q((N-k+1, R, 1)$$

110804/10160

- Programación dinámica con objetos de una clase que refleja una situación concreta con cuantas estaciones se han utilizado y cuantas ciudades faltan por tener servicio.
- Se utiliza una array de estos objetos de tamaño $N + 1$ y se comienza por el final (posición N), situación en la cual todavía no se ha instalado ninguna estación de servicio.
- Cada vez que se añade una estación en una ciudad se contabiliza a cuantas ciudades da servicio.
- La nueva situación alcanzará una posición en el array según el número de ciudades pendientes de tener servicio. La solución es la situación en la que todas las ciudades tienen servicio y con el mínimo número de estaciones.

110806/10001

- *Backtraking* puro con poda, es decir, *branch and bound*.
- Debemos buscar hacia atrás un estado del autómata celular que nos pueda llevar a la configuración objetivo.
- Debemos intentar componer una secuencia de ternas (tres bits) que sean encadenables.
- En cuanto encontremos una ya podemos parar y decir que es *REACHABLE*.

objetivo	1	0	1	1	0	1	0	1
ternas	abc	bcd	cde	def	efg	fgh	gha	hab
predecessor	b	c	d	e	f	g	h	a

- La tabla de transiciones que define el autómata celular nos permite saber qué ternas corresponden a un bit del objetivo.

Resumen problemas

Del tema 8 hemos abordado los problemas:

- 110801/861 “Little Bishops”
- 110802/10181 “15-Puzzle Problem”
- 110803/10128 “Queue”
- 110804/10160 “Servicing Stations”
- 110806/10001 “Garden of Eden”