Algorithms for Problem Solving – 11650

Combinatoria

Jon Ander Gómez Adrián
jon@dsic.upv.es

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

16 de febrero de 2014

- La combinatoria resulta de gran utilidad en problemas de conteo
- En algunos casos, debido al coste computacional de la solución, debemos calcularla previamente para luego construir el programa que enviaremos al juez
- Este programa dispondrá de una tabla con los valores de salida precalculados
- También nos encontraremos con problemas en los que deberemos utilizar aritmética de números grandes

- Regla del producto. Cuando necesitamos combinar cada uno de los elementos de un conjunto A con cada uno de los elementos de otro conjunto B

$$|A| \times |B|$$

- Regla de la suma. Queremos contar las posibilidades de tomar un elemento de un conjunto A o de un conjunto B

$$|A| + |B|$$

- Fórmula de Inclusión-Exclusión. La anterior regla es un caso especial de ésta. Por ejemplo, si juntamos dos conjuntos de lápices de colores en los que no había repetidos, los colores repetidos por unirlos están en la intersección, por tanto, el número de colores diferentes es como sigue

$$|A \cup B| = |A| + |B| - |A \cap B|$$

Técnicas de Conteo Básicas

- Permutaciones. Número de combinaciones diferentes de n elementos

$$n! = \prod_{i=1}^{n} i \qquad \{123, 132, 213, 231, 312, 321\}$$

- Subconjuntos. Número de subconjuntos que podemos formar con n elementos. **Incluyendo el conjunto vacío**

$$2^n \qquad \{\phi, 1, 2, 3, 12, 13, 23, 123\}$$

- Combinaciones de m elementos tomados de n en n, con repetición

$$m^n \qquad \{111, 112, 113, 121, 122, 123, \ \ldots \ 331, 332, 333\}$$

- Conviene conocer la rapidez con que crece el número de objetos para saber en qué momento la técnica de búsqueda exhaustiva ya no es viable.

- Nos permiten contar fácilmente los componentes de estructuras definidas recursivamente
- Muchas funciones se expresan de manera natural con relaciones de recurrencia
- Incluso la solución de muchos problemas se expresa mejor y más escuetamente mediante funciones recursivas
- Una función recursiva se define en función de sí misma

$$a_n = a_{n-1} + 1, \quad a_1 = 1 \quad \Rightarrow \quad a_n = n$$

$$a_n = 2 \cdot a_{n-1}, \quad a_1 = 2 \quad \Rightarrow \quad a_n = 2^n$$

$$a_n = n \cdot a_{n-1}, \quad a_1 = 1 \quad \Rightarrow \quad a_n = n!$$

- $\binom{n}{k} = \frac{n!}{(n-k)!\cdot k!} = \binom{n-1}{k-1} + \binom{n-1}{k}$

- Fáciles de calcular todos los necesarios de manera iterativa utilizando principios de *Programación Dinámica*

  El algoritmo está en el apartado 4 del capítulo 6 del libro *"Programming-Challenges"*

- Se utilizan para obtener los coeficientes de fórmulas tipo $(a+b)^n$, p.e. el triángulo de Pascal

- Nos dan el número de combinaciones de n elementos tomados de k en k, sin repetición. Lo que resulta de gran utilidad en muchos problemas de conteo:

  ¿cuántos comités de k miembros podemos formar con n personas?

Otras secuencias de conteo

Existen otras secuencias de conteo, algunas de las cuales citamos aquí, que resultan de utilidad en algunos problemas:

- Números de Fibonacci

$$
\begin{aligned}
F_0 &= 0 \\
F_1 &= 1 \\
F_n &= F_{n-1} + F_{n-2}
\end{aligned}
$$

- Los números de Catalan

$$C_n = \sum_{k=0}^{n-1} C_k \cdot C_{n-1-k}$$

- Los números de Euler y de Stirling [1]

[1] Ver formulación en el libro

110601/10183

- Este problema tiene fácil solución utilizando números grandes. Cuyo código podéis encontrar en

  `http://www.cs.sunysb.edu/~skiena/392/programs/bignum.c`

- En el libro se sugiere que puede haber una manera de solucionarlo sin números grandes
- La solución con números grandes acaba en décimas de segundo
- La estrategia consiste en precalcular todos los números de Fibonacci hasta aquel cuyo número de dígitos supera los 100. Después contando entre los límites es suficiente

110602/10213

- Para este problema es necesario utilizar números grandes
- La secuencia de números sigue una fórmula que en base a los primeros podemos encontrar en

  `http://oeis.org`

- Se aconseja intentar deducirla antes de buscarla
- Existen dos relaciones de recurrencia que aplicadas cuando $n > 4$ obtienen la secuencia, pero ¿son viables?

110603/10198

- Para resolver este problema es necesario detectar la relación de recurrencia que permite obtener el número de combinaciones que sumen n a partir del número de combinaciones que sumaban valores inferiores a n
- ¿Pero hasta cuantos anteriores?
- Una estrategia, que no resulta muy útil en las competiciones, es obtener manualmente, o mediante un programa, el número de combinaciones para los primeros valores de n, y con ellos buscar si existe una secuencia calculable en

  `http://oeis.org`
- Provad ``PreCalculo.java'' para obtener el número de combinaciones para valores pequeños de n

110607/10049

- Para este problema debemos cambiar el punto de vista
- Como calcular en el momento el valor 'f(n)' es inviable en cuanto a tiempo, y almacenar toda la secuencia es inviable en cuanto a espacio en memoria central
- ¿Entonces? Podemos guardar la secuencia a la inversa
- Una tabla donde 'f(n)' es el índice, y el máximo 'n' de 'f(n)' es el valor contenido en la tabla
- Una vez precalculada la tabla a la inversa tan solo queda buscar con la búsqueda dicotómica

110608/846

- Este problema es bastante sencillo, pero para percatarse de la secuencia implícita conviene observar el número de saltos o pasos que hay que dar para diferencias desde 1 hasta 30 por ejemplo
- ¿Qué función se debe aplicar a la distancia?
- ¿Qué sencilla operación se debe realizar sobre el valor obtenido según qué condición?

Resumen problemas

- 110601/10183 “How Many Fibs?”
- 110602/10213 “How Many Pieces of Land?”
- 110603/10198 “Counting”
- 110607/10049 “Self-describing Sequence”
- 110608/846 “Steps”