Algorithms for Problem Solving – 11650

Álgebra, Aritmética y Números Grandes

Jon Ander Gómez Adrián
jon@dsic.upv.es

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

11 de febrero de 2014

Enteros de alta precisión

- Los ordenadores tienen capacidad finita para representar los números
- La mayoría de compiladores disponen de enteros de 64 bits
- Para algunos problemas será necesario trabajar con números números cuya codificación en base 2 excede de 64 bits
- Para trabajar con números grandes podemos adoptar, básicamente, dos estrategias: un array de dígitos o una lista enlazada de dígitos
- El código para trabajar con números grandes utilizando un array de dígitos lo podéis encontrar en el siguiente enlace:

  `http://www.cs.sunysb.edu/~skiena/392/programs/bignum.c`
- Conviene tener presente relaciones como la siguiente para reducir la cantidad cómputo

$$a^n = a^{\frac{n}{2}} \times a^{\frac{n}{2}} \times a^{n \bmod 2}$$

- Por la capacidad finita de los ordenadores para representar los números, cuando se trata de reales debemos tener presente la falta de precisión en los cálculos por el redondeo que sufre el resultado de cada operación

- No siempre se cumple

$$(a + b) + c = a + (b + c)$$

- Cuando sepamos que vamos a trabajar con racionales podemos manipular dos enteros: numerador y denominador

- Mantener la fracción $\frac{m}{n}$ por separado es mucho más preciso

- Las funciones $log(x)$ y $exp(x)$, disponibles en todos los lenguajes de programación, son de gran ayuda para realizar operaciones que requieren calcular potencias

- Algunas igualdades que conviene tener presentes según para qué cálculos

$$a^b = e^{ln(a^b)} = e^{b \cdot ln(a)}$$

$$\log_a n^b = b \cdot \log_a n$$

$$\log_a b = \frac{\log_c b}{\log_c a}$$

110501/10035

- La entrada estándar se procesa con simples `scanf()`. No entraña ningún problema
- Este problema es tan fácil que no digo nada más

110502/10018

- La entrada estándar se procesa con simples `scanf()`.

  No entraña ningún problema

- Tened en cuenta el valor máximo, así que será mejor trabajar con enteros de 32 bits sin signo

- Este problema es tan fácil que no digo nada más

110504/10127

- Si tomamos el 3, podemos comprobar si $1 \bmod 3 = 0$, después si $11 \bmod 3 = 0$, y finalmente si $111 \bmod 3 = 0$, que en este último caso es que sí
- Pero esta no es la mejor manera de atacar este problema, para ciertos números de entrada su múltiplo con todo unos puede ser un número demasiado grande
- Este problema puede resolverse sin utilizar números grandes
- Por ejemplo, sabemos que 111 = 7*3 + 30*3 = 21 + 90
- O mejor: 111 = (21 % 10) + (21/10 + 3*3) * 10 (aprovechando la división entera)

110507/10077

- No necesitamos representar todo el árbol, simplemente necesitamos mantener tres fracciones
- Inicialmente serán $\frac{0}{1}$, $\frac{1}{1}$ y $\frac{1}{0}$
- Cuando la central coincida con la de entrada ya hemos acabado
- Cuando no coincida decidiremos hacia donde ir (L o R) y actualizaremos dos de las tres fracciones según interese
- Resultará cómodo implementar funciones para *copiar*, *comparar* y *crear* fracciones

110508/10202

- En primer lugar debemos ordenar los números de la entrada en orden ascendente
- Para descubrir el primer número fijémonos en lo siguiente:
  $b_1 = a_1 + a_2$
  $b_i = a_1 + a_3$
  $b_j = a_2 + a_3$
  donde las b's representan los números que nos dan, y las a's los números que hemos de averiguar
- Luego $b_1 + b_i - b_j = 2 * a_1$
  si no encontramos ningún par $\rightarrow$ Impossible
- A partir de aquí podemos comenzar a despejar

Resumen problemas

- 110501/10035 “Primary Arithmetic”
- 110502/10018 “Reverse and Add”
- 110504/10127 “Ones”
- 110507/10077 “The Stern-Brocot Number System”
- 110508/10202 “Pairsumonious Numbers”