Universidad Rey Juan Carlos

Grado en Ingeniería de la Ciberseguridad

Criptografía

Sesión 1. Teoría de Complejidad.

---

1. Escribe (en pseudocódigo, o en cualquier lenguaje que te sea familiar) un algoritmo que multiplique dos matrices de enteros (cuadradas, de dimensión m). Si las entradas de las matrices son números enteros, cada uno representado con l bits, calcula su complejidad en función de l y m.

2. Escribe un algoritmo ("naive") para la exponenciación modular y calcula su complejidad con respecto al tamaño en bits del módulo (eso marca el tamaño de la entrada). Puedes expresar esta complejidad sencillamente contando operaciones elementales (productos, sumas y reduciones modulares). Haz el mismo cálculo para el algoritmo "square and multiply".

3. Estudia y compara la complejidad computacional de dos algoritmos de búsqueda: una búsqueda secuencial (naive) y la búsqueda binaria (con el pseudocódigo de abajo, visto en clase de MDA). Supón que en ambos casos la entrada es una lista ordenada de manera creciente de números enteros (cuya longitud, n, es 2^s para algún s.), y un entero al que buscar. ¿Qué conviene utilizar si la lista de entrada no está ordenada?

   procedure BINARYSEARCH$(a_1, \ldots, a_n; x)$
   $i := 1, j := n$
   while $i < j$ do **do**
   $m := \frac{i+j}{2}$
   if $a_m < a$ **then** $i := m + 1$
   else $j := m$
   end if
   end while
   if $x == a_i$ **then** then $r := i$
   else $r := 0$
   end if
   return r.
   end procedure

4. Escribe (en pseudocódigo, o en cualquier lenguaje que te sea familiar) un algoritmo que reciba dos números enteros de entrada x, y. El algoritmo calcula un valor entero α (que suponemos que existe) tal que $y = x^\alpha$.