Métodos Algorítmicos en Resolución de Problemas II

Grado en Ingeniería Informática

Doble Grado en Ingeniería Informática y Matemáticas

Hoja de ejercicios 5 Curso 2025–2026

Ejercicios de algoritmos probabilistas

Ejercicio 1 Demostrar que los testigos falsos fuertes de primalidad son también testigos falsos con respecto al test de Fermat.

Ejercicio 2 Diseñar un algoritmo de Monte Carlo que compruebe en tiempo constante si un elemento dado pertenece a un vector $V[1..n]$. ¿Cuál es la probabilidad de que la respuesta sea correcta? Para $n = 10$, ¿cuántas veces es necesario ejecutar el algoritmo para poder garantizar que devuelve la respuesta correcta con una probabilidad mayor o igual que 0,9?

Ejercicio 3 Considérese el siguiente programa que no termina.

```
proc imprimirPrimos
  imprimir 2, 3
  n := 5
  mientras verdadero hacer
    si repetirMillRab(n, ⌊lg n⌋) entonces imprimir n fsi
    n := n + 2
  fmientras
fproc
```

Claramente, todo número primo terminará siendo imprimido por este programa, pero uno podría también esperar algún número compuesto de vez en cuando: probar que es improbable que esto ocurra. Más precisamente, demostrar que la probabilidad de que ningún número compuesto mayor que 400 aparezca alguna vez es superior al 99 %, independientemente de cuánto tiempo el programa se ejecute.

Ejercicio 4 Investigar la amplificación de la ventaja estocástica en problemas con más de dos potenciales respuestas. En general, incluso podría ocurrir que exista más de una respuesta correcta para algunas instancias (por ejemplo, en el problema de encontrar un divisor de un número compuesto). La dificultad potencial es que aunque un algoritmo p-correcto devuelve una respuesta correcta con alta probabilidad cuando p es grande, podría ocurrir que una respuesta errónea sea devuelta sistemáticamente más a menudo que cualquier respuesta correcta. En este caso, ¡la amplificación de la ventaja estocástica por mayoría de votos disminuiría la probabilidad de ser correcto! Mostrar que si un algoritmo *MC* es 75 %-correcto, podría ocurrir que *MC3* no sea ni siquiera 71 %-correcto, donde *MC3* devuelve la respuesta más frecuente en tres llamadas a *MC*. (Los empates se rompen arbitrariamente.)

Ejercicio 5 Dar un algoritmo de Monte Carlo 1/2-correcto y sesgado para decidir si un vector $T[1..n]$ contiene un elemento mayoritario. El algoritmo debería ejecutarse en tiempo lineal y las únicas comparaciones permitidas entre los elementos de T son tests de igualdad. Nótese que el único mérito de este algoritmo es su sencillez ya que el algoritmo determinista visto en clase resuelve el problema en tiempo lineal con una constante oculta muy pequeña.

Ejercicio 6 Sean A y B dos algoritmos de Monte Carlo sesgados que resuelven el mismo problema de decisión. El algoritmo A es p-correcto y su respuesta es correcta cuando devuelve *cierto*; el algoritmo B es q-correcto y su respuesta es correcta cuando devuelve *falso*. Mostrar cómo combinar A y B en un algoritmo de tipo Las Vegas $LV(x, y, \textit{éxito})$ que resuelva el mismo problema. ¿Con qué probabilidad?

Ejercicio 7 Tenemos una colección de n tuercas de distintos diámetros y sus correspondientes n tornillos. Es necesario emparejar cada tuerca con su tornillo pero las diferencias en tamaño son demasiado pequeñas como para ser apreciables a simple vista, con lo que la única comparación posible es intentar enroscar una tuerca con un tornillo para ver si es demasiado grande, demasiado pequeña o si encaja perfectamente. Escribir un algoritmo que resuelva el problema en tiempo $O(n^2)$. Escribir un algoritmo probabilista que resuelva el problema en tiempo esperado $O(n \lg n)$.

Ejercicio 8 La ONCE acaba de sacar un nuevo juego *Rasca y gana* en el que todos los números entre 1 y n^2 aparecen ocultos en un orden desconocido bajo los correspondientes n^2 círculos plateados, dispuestos en un cuadrado $n \times n$. Cuando rascamos un círculo, vemos el correspondiente número oculto i, junto con la ubicación precisa donde se esconde su sucesor $i+1$. Además, el boleto incluye un círculo adicional con el número premiado del boleto, que podemos rascar al principio sin ningún coste. Según las reglas del juego, el número premiado será siempre un múltiplo de n. Se pide un algoritmo de Las Vegas de tipo (a) que permita encontrar la ubicación del premio de un boleto cualquiera, tras rascar un número esperado de casillas en $O(n)$. Razonar por qué dicha complejidad se cumple y dar con la mayor precisión posible la constante multiplicativa.