Metodología y tecnología de la programación

Ingeniería Técnica en Informática de Sistemas (UCM)

Hoja de ejercicios 7 Curso 2008/2009

Ejercicios de algoritmos probabilistas

Ejercicio 1 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 2 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 es superior al 99% de que ningún número compuesto mayor que 100 aparecerá alguna vez, independientemente de cuánto tiempo el programa se ejecute.

Ejercicio 3 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 el 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 4 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 5 Transformar el algoritmo de *quicksort* en un algoritmo de tipo Las Vegas. Razonar que se ejecuta en tiempo esperado en el peor caso en $\Theta(n \lg n)$.

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 $LV(x, y, éxito)$ que resuelva el mismo problema. ¿Con qué probabilidad?