Ejercicios Tema 2

Algoritmia para problemas difíciles

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura – Universidad de Zaragoza

20 de octubre de 2023

Ejercicios sobre algoritmos probabilistas

Ejercicio 1 Implementar el algoritmo probabilista numérico para integrar funciones y probarlo con funciones de 4 variables comparando su error y eficiencia con el método determinista para distintas funciones. El objetivo es por un lado probar casos de funciones muy regulares y por otro encontrar casos de funciones en los que el método probabilista mejore sustancialmente el error para tiempos de cálculo similares.

Ejercicio 2 Dado un conjunto no ordenado S y un natural k, queremos seleccionar el k-ésimo elemento de S en orden de menor a mayor. Implementar un algoritmo probabilista que resuelva el problema utilizando pivotes aleatorios de forma similar a la ordenación vista en clase (transparencias tema 2). ¿Qué k da el peor coste medio? Plantear la ecuación en recurrencias del tiempo medio para cada entrada (no es necesario resolverla). Estimar experimentalmente el tiempo medio para cada entrada. **Nota:** Con este ejercicio hay que entregar el código realizado y la lista de pruebas ejecutadas, además de una explicación completa de los resultados.

Ejercicio 3 Una lista compacta de longitud n se implementa con dos vectores, $val(1..n)$ y $ptr(1..n)$, y un entero, cabeza. El primer elemento de la lista está en $val(cabeza)$, el siguiente en $val(ptr(cabeza))$, etcétera. En general, si $val(i)$ no es el último elemento de la lista, $ptr(i)$ guarda el índice en val del siguiente elemento de la lista. Si $val(i)$ es el último elemento de la lista, entonces $ptr(i) = 0$.
Considerar una lista compacta cuyos elementos están en orden creciente, es decir, $val(i) < val(ptr(i))$ para todo $i = 1, 2, \ldots, n$ tal que $ptr(i) \neq 0$. Sea x un elemento. El problema es localizar x en la lista, es decir, dar su posición en val. El método de búsqueda dicotómica no es aplicable porque no hay forma directa de localizar el punto intermedio de la lista. Diseñar un algoritmo probabilista más rápido que la búsqueda secuencial para resolver el problema, este algoritmo debe estar basado en hacer un número $s(n)$ de elecciones probabilistas y quedarse con la mejor de ellas. Plantear la ecuación en recurrencias del tiempo medio para cada entrada según $s(n)$ (no es necesario resolverla). Probar experimentalmente distintos valores de $s(n)$. **Nota:** Con este ejercicio hay que entregar el código realizado y la lista de pruebas ejecutadas, además de una explicación completa de los resultados.

Ejercicio 4 Para una compra de material para tu empresa recibes una lista de n ofertas. Quieres seleccionar la mejor oferta dentro de las que cumplen unos requisitos dados. Cuando examinas una oferta compruebas si cumple los requisitos además de hacer todas las preguntas que desees, adjudicando una puntuación a la oferta. Una vez examinada la oferta debes aceptarla inmediatamente o se pierde, debido a la condiciones de compra establecidas.
Suponiendo que las ofertas se presentan en un orden aleatorio, diseñar un algoritmo DETERMINISTA que seleccione con al menos el 25 % de probabilidad la mejor oferta. Justificar la respuesta.

Ejercicio 5 Sea A un algoritmo de Montecarlo eficiente. El algoritmo A es p-correcto para un $p > 1/2$.

- ¿Cómo podemos repetir el algoritmo para mejorar el error?
- Escribir un algoritmo eficiente para el mismo problema con probabilidad de error como mucho ϵ.

- ¿Cuánto tiempo tarda el nuevo algoritmo?
- ¿Para qué q se trata de un algoritmo q-correcto?

Ejercicio 6 Sea A un algoritmo Las Vegas con probabilidad de respuesta al menos p y B un algoritmo Montecarlo que sea q-correcto (con p y q constantes en $(0, 1)$), ambos eficientes y para un mismo problema decisional. Plantear alguna manera de combinar A y B en un algoritmo probabilista eficiente para resolver el mismo problema intentando mejorar la probabilidad de error y respuesta para algún caso. ¿Cuánto tiempo tarda el nuevo algoritmo? ¿Qué probabilidad de error y de respuesta tiene?

Ejercicio 7 Sea Max-3SAT el siguiente problema de optimización:

Entrada: Un circuito booleano en CNF C con una única salida y con exactamente 3 literales por cláusula, además en cada cláusula no hay variables repetidas.

Salida: Una asignación de las variables que hace ciertas el máximo número posible de cláusulas.

Dar un algoritmo probabilista eficiente que calcule una asignación que se acerque a la óptima y razonar para cada C cuántas cláusulas satisface **en media** la asignación calculada. **Notación:** Un *literal* es una variable booleana afirmada o negada (p. ej. x, $\neg y$, etc). Una *cláusula* es una disyunción de literales (p. ej. $(x \vee \neg y)$). Un *circuito en CNF* es una conjunción de cláusulas (p. ej. $(x \vee \neg y \vee z) \wedge (\neg z \vee y) \wedge (\neg x)$). Una asignación de las variables satisface una cláusula cuando la hace cierta, una asignación de las variables satisface un circuito cuando hace ciertas todas sus cláusulas.

Ejercicio 8 Nos dan un conjunto de n variables $x_1, x_2, \ldots, x_n$, cada una de las cuales puede tomar valores en $\{0, 1\}$. Nos dan también un conjunto de k ecuaciones, la r-ésima ecuación tiene la forma

$$(x_{i_1} + \ldots + x_{i_{m_r}}) \text{ mód } 2 = b_r$$

para $m_r \geq 2$ variables distintas $x_{i_1}, \ldots, x_{i_{m_r}}$ y para un valor b_r que es 0 ó 1. Por tanto cada ecuación especifica si la suma de varias variables es par o impar.

Por ejemplo, supón que nos dan las ecuaciones

$$\begin{aligned} (x_1 + x_2) \text{ mód } 2 &= 0 \\ (x_1 + x_3) \text{ mód } 2 &= 0 \\ (x_2 + x_4) \text{ mód } 2 &= 1 \\ (x_3 + x_4) \text{ mód } 2 &= 0 \end{aligned}$$

sobre las cuatro variables $x_1, \ldots, x_4$. Entonces es posible demostrar que ninguna asignación de valores a estas variables satisface todas las ecuaciones simultáneamente, pero asignando a todas las variables 0 satisfacemos tres de las cuatro ecuaciones.

Sea c el número máximo de ecuaciones que pueden ser satisfechas por una asignación de valores a las variables. Dar un algoritmo probabilista eficiente que cumpla que el número medio de ecuaciones satisfechas es al menos $1/2 \cdot c$.

Ejercicio 9 Sean A y B dos algoritmos probabilistas eficientes para un mismo problema decisional. El algoritmo A es p-correcto y además su respuesta está garantizada cuando la respuesta correcta es *verdadero*; el algoritmo B es q-correcto y además su respuesta está garantizada cuando la respuesta correcta es *falso*. Mostrar la manera de combinar A y B en un algoritmo probabilista eficiente para resolver el mismo problema con menor probabilidad de error. ¿Cuánto tiempo tarda el nuevo algoritmo? ¿Qué probabilidad de error tiene?

Ejercicio 10 *3-Coloreado* es un problema decisional (decidir si un grafo G se puede colorear con 3 colores), pero podemos considerar el siguiente problema de optimización asociado a *3-Coloreado*:

Supongamos que nos dan un grafo G y queremos colorear cada nodo con uno de tres posibles colores, aunque no podamos dar colores distintos a vértices adyacentes. En lugar de eso, decimos que la arista (u, v) se satisface si los colores asignados a u y v son diferentes.
Considerar un 3-coloreado que maximiza el número de aristas satisfechas y llamemos c a este número. Dar un algoritmo en tiempo polinómico que calcule un 3-coloreado que satisfaga al menos $2/3 \cdot c$ aristas. Si se desea puede ser un algoritmo probabilista, en ese caso el número *medio* de aristas satisfechas debe ser al menos $2/3 \cdot c$.

Ejercicio 11 Implementar y experimentar un posible test de primalidad (que hemos visto en clase que no es correcto) `miTest` basado en el pequeño teorema de Fermat y los falsos testigos de primalidad. Comparar esta implementación con un test de primalidad correcto (para ello se puede utilizar código de otras fuentes citándolas) ¿Cuántos números clasifica mal `miTest` con alta probabilidad? ¿Cuáles? **Nota:** Con este ejercicio hay que entregar el código realizado y la lista de pruebas ejecutadas, además de una explicación completa de los resultados.

Ejercicio 12 Sea $T(1..n)$ un vector de n elementos. Se dice que x es el dato mayoritario en T si $|\{i|T(i) = x\}| > n/2$. Se dice que el vector T es mayoritario si contiene un elemento mayoritario.
Diseñar un algoritmo de Monte Carlo para determinar si un vector es mayoritario basado en la selección aleatoria de un elemento y la comprobación de si ese elemento es mayoritario o no. Argumentar sobre la probabilidad de error del algoritmo. Basado en el algoritmo anterior, diseñar otro con tiempo de ejecución $O(n \log(1/e))$, donde n es el número de elementos del vector y e es la probabilidad de error.
Nota: Este ejercicio es sólo interesante como ilustración de los algoritmos de Monte Carlo pues se conoce un algoritmo determinista lineal (¿cuál es dicho algoritmo determinista? Cuidado: es un poco más delicado que el recursivo cuasi-lineal).

Ejercicio 13 Sea A el test de primalidad de Miller-Rabin y B un algoritmo Las Vegas para Primalidad, con probabilidad de respuesta de al menos 30 % y que funciona en tiempo cúbico en el tamaño de la entrada. Plantear alguna manera de combinar A y B en un algoritmo probabilista eficiente para resolver Primalidad intentando mejorar la probabilidad de error y respuesta de A y B. ¿Cuánto tiempo tarda el nuevo algoritmo? ¿Qué probabilidades de error y de respuesta tiene? **Nota:** En este ejercicio es importante recordar cuál es el tamaño de la entrada cuando la entrada es un número entero y qué quiere decir eficiente.

Ejercicio 14 Recordad el algoritmo de quicksort con pivote aleatorio (transparencias tema 2). Dado un vector A de n elementos a ordenar, llamamos *buen pivote* a aquel que queda en una posición entre $n/4$ y $3n/4$ una vez ordenado el vector. Diseñar una variante de quicksort con pivote aleatorio basada en buscar un pivote aleatorio que sea bueno (con un número fijo de repeticiones o intentos hasta encontrarlo o conformarse con uno malo). Escribir una ecuación en recurrencias que acote el tiempo medio del algoritmo diseñado.

En caso de entregar alguno de estos ejercicios, la fecha límite es el lunes 30 de octubre.

Antes de realizar cualquiera de estos ejercicios el alumno debe seleccionarlo en moodle.

Cualquier fuente utilizada en la resolución de estos ejercicios debe ser indicada claramente en la solución.