ALGORITMOS PROBABILISTAS

ALGORITMOS PROBABILISTAS

QUE SON?

Algoritmos deterministas:

Son aquellos que bajo las mismas condiciones producen la misma salida. En un algoritmo las condiciones del problema suelen ser los datos de entrada.

Algoritmos probabilistas:

Son aquellos en los que en algún punto del algoritmo donde hay que tomar una decisión, esta se elige basándose en el azar.

Por tanto el mismo algoritmo puede comportarse de forma distinta aplicado a los mismos datos.

Esto es interesante por ejemplo en casos en los que es muy costoso elegir la alternativa más adecuada.

El interés radica en que el algoritmo probabilista funcione mejor que el determinista en promedio.

El algoritmo probabilistico se ejecuta más eficientemente en tiempo y/o memoria, a costa de no dar la respuesta correcta el 100% de las ejecuciones.

ALGORITMOS PROBABILISTAS vs DETERMINISTAS

Más diferencias entre deterministas y probabilistas:

− A un algorítmo determinista **nunca** se le debe permitir que no termine (división por cero, bucle infinito,...), o que dé una solución incorrecta.

− En cambio a uno probabilista esto si que se le permite siempre y cuando sea con una probabilidad muy pequeña para datos cualesquiera, y que de esta forma funcione mejor en promedio.

En caso de fallar, se repite la ejecución del algoritmo hasta obtener el grado de confianza deseado.

− Si existe más de una solución para unos datos dados, un algoritmo determinista **siempre** encuentra la misma solución (a no ser que se programo para que encuentre todas las soluciones).

− Sin embargo uno probabilista puede encontrar soluciones diferentes ejecutándose varias veces con los mismos datos.

− El análisis de un algoritmo determinista es a veces difícil.

− El análisis de uno probabilista casi siempre es muy difícil.

Interes práctico:

Con algoritmos probabilistcos se puede conseguir resolver problemas en un tiempo razonable, que de otra

forma tendrian un coste en tiempo inviable.

Esto ocurre por ejemplo en la factorización de números de muchas cifras, lo que supuso un fuerte impacto en el campo de la *criptografía*.

Ejemplo:

En 1977 *M. Gardner* planteo un sistema para descifrar mensajes encriptados mediante RSA que exigía factorizar un número de 129 cifras. Gardner estimó en 2 millones de veces la edad del Universo el tiempo necesario para resolverlo.

En 1994 se resolvio el problema en 8 mese de cálculo utilizando 600 computadores en paralelo mediante un algoritmo de Las Vegas.

ALGORITMOS PROBABILISTAS.

CLASIFICACIÓN.

Algoritmos que no garantizan la corrección de la solución:

− Algoritmos numéricos:

− dan una solución aproximada

− dan un intervalo de confianza

− a mayor tiempo de ejecución, mejor es la aproximación.

− Algoritmos de Monte Carlo:

− dan la respuesta exacta con una alta probabilidad

− en algunas ocasiones dan una respuesta incorrecta

− no se puede saber con certeza si la respuesta es la correcta

− se reduce la probabilidad de error alargando la ejecución

Algoritmos que nunca dan una solución incorrecta:

− Algoritmos de Las Vegas:

− toman decisiones al azar

− si no encuentran la solución correcta lo admiten

− es posible volver a intentarlo con los mismos datos hasta obtener la repuesta correcta

ALGORITMOS DE MONTE CARLO.

Sea p un número real tal que $0<p<1$.

Un algoritmo de Monte Carlo es *p*−correcto si devuelve una solución correcta con probabilidad mayor o igual que *p*, cualesquiera que sean los datos de entrada.

A veces *p* dependerá del tamaño de la entrada, pero nunca de los datos de la entrada en si.

Interes práctico:

Normalmente en problemas que han de devolver un valor booleano, se consigue un algoritmo de Monte Carlo que si devuelve uno de los dos valores (verdad por ejemplo) estemos seguros de que ha acertado, mientras que si devuelve el otro (falso) no sepamos nada sobre la respuesta más que la probabilidad de fallo.

Si da la respuesta que no nos asegura nada, el decrecimiento de la probabilidad de error es espectacular repitiendo la prueba varias veces.

Ejemplo:

Supongamos un algoritmo 1/2−correcto, es decir, nos da la solución correcta con probabilidad 1/2. Si ejecutamos el algoritmo k veces, la probabilidad de que falle todas ellas es 1/2k. Por tanto, si nos da la respuesta que sabemos es cierta, estamos seguros de ella. Si por el contrario nos devuelve la respuesta que no es segura en k ejecuciones, podemos darla por valida con una probabilidad de (1−1/2k), lo que para k=10 supone mejor que 0,999−correcto.

ALGORITMOS DE LAS VEGAS.

Un algoritmo de las vegas **nunca** da una solución falsa.

Toma decisiones al azar para encontrar una solución **antes** que un algoritmo determinista, y si no la encuentra, lo admite.

Hay dos tipos de algoritmos de Las Vegas:

a) Los que **siempre** encuentran una solución correcta, aunque las decisiones al azar no sean afortunadas y la eficiencia disminuya.

Existen algoritmos deterministas que funcionan mucho mejor en media que en el peor de los casos.

Ej. quicksort: coste peor ð (n2)

coste promedio. O(n log n)

Puede haber aplicaciones en los que las entradas no sean equiprobables, y se repitan entradas catastróficas (en el ej. vectores casi ordenados).

Los algoritmos de Sherwood (Las Vegas) uniformizan el tiempo de ejecución para cualquiera que sea la entrada.

b) Los que **a veces**, debido a decisiones desafortunadas, no encuentran una solución.

Son aceptables si fallan con probabilidad baja.

Si fallan, se vuelven a ejecutar con la misma entrada.

Resuelven problemas para los que no se conocen algoritmos deterministas eficientes, como por ejemplo la factorización de enteros grandes.

El tiempo de ejecución no está acotado pero sí es razonable con la probabilidad deseada para toda entrada.

En ocasiones se combinan algoritmos deterministas con probabilistas, evaluando en cada caso hasta donde se utiliza uno y hasta donde el otro.