CC40A: Diseño y Análisis de Algoritmos
Pauta Control 2 - Semestre Otoño 2002

Profesor: Ricardo Baeza Yates
Prof. Auxiliar: Rodrigo Paredes

P1 (1 pto.)

Indique y justifique si la siguiente afirmación (o parte de ella) es verdadera o falsa: *Hashing y compresión de archivos binarios son ejemplos de uso de características de los datos que son finitas.*

Respuesta: Hashing supone que la función de hashing se puede calcular en forma eficiente. Esto implica que los datos deben tener largo finito, ya sean números o strings (el tamaño de la tabla generalmente NO depende de los datos). Por otra parte, compresión es un ejemplo válido si el método usa finititud. Hay métodos que si lo usan, otros no, así que en general es falso (si se justifica vale también decir que es verdadero).

P2 (2 ptos.)

Considere el siguiente algoritmo:

```
Misterio( A, n, x, i, j, k ) {   // A es un arreglo de n enteros
   if( i >= j ) return( x == A[i] );
   elif( k==1 ) return( x == A[random( i..j )] );
   else {
      m = (i+j)/2; k--;
      if( A[m] < x ) Misterio( A, n, x, m+1, j, k );
      else           Misterio( A, n, x, i, m, k );
   }
}
```

- ¿Qué hace este algoritmo?

  Respuesta: es un test de pertenencia de x al arreglo que usa una búsqueda binaria que divide el arreglo k veces y luego si le queda más de un elemento compara uno de ellos al azar.

- ¿La respuesta es siempre correcta? ¿El algoritmo es de tipo Las Vegas o Montecarlo?

  Sí, si $k \geq \lceil \log_2 n \rceil$ y entonces es Las Vegas. No en caso contrario y entonces es Montecarlo.

- ¿El algoritmo es 1-sesgado o 2-sesgado?

  Cuando $k < \lceil \log_2 n \rceil$ es 1-sesgado pues sólo se equivoca cuando dice que no está y en realidad está.

- ¿Qué valor de k tiene que usar para estar seguro que la probabilidad de una respuesta incorrecta es menor o igual a 1/2 si considera una distribución uniforme para los elementos?

  Para que la probabilidad sea menor o igual a 1/2 deben quedar sólo dos elementos como candidatos. Es decir, $k = \lceil \log_2(n/2) \rceil = \lceil \log_2(n) \rceil - 1$.

P3 (2 ptos.)

Considere el problema, visto en clases, del piso más alto desde el cual se podía lanzar un objeto sin que se quebrara considerando un edificio infinito. Diseñe e indique la complejidad de un algoritmo eficiente cuando se tienen k objetos, $k > 0$.

Respuesta: Hay que buscar un límite para el tamaño del edificio como en el problema de adivinar un número y luego buscar en lo que resta usando la solución vista en clases. El límite se puede buscar usando A_1 o A_2. No vale la pena usar algo mejor pues la parte dominante del algoritmo es la segunda parte. Si usamos A_1 tenemos $\lceil \log_2 n \rceil$ comparaciones para encontrar el límite (donde n es la respuesta que estamos buscando) y luego nos quedan $k-1$ objetos para buscar en un edificio de altura $2^{\lceil \log_2 n \rceil}$ que toma tiempo $(k-1)2^{\lceil \log_2 n \rceil/(k-1)}$. Es decir, en total $O((k-1)n^{1/(k-1)} + \log n)$ (1 punto).

Esta solución sin embargo no funciona si $k = 1$ pues nos quedamos sin objetos buscando el límite. Por lo tanto para $k = 1$ debemos usar búsqueda secuencial (0.5 ptos).

Para el caso $k = 2$ tampoco tiene mucho sentido buscar el límite pues el segundo objeto tendrá que hacer búsqueda secuencial. Por lo tanto es mejor comenzar con búsqueda secuencial desde el principio (0.5 ptos). Así que para $k < 3$ el mejor algoritmo es $O(n)$.

P4 (1 pto.)

Elija uno de los siguientes problemas:

a) Suponga que tiene un algoritmo aleatorio A que tarda siempre un tiempo $4T(n)$ y da una respuesta incorrecta con probabilidad $1/8$; y un algoritmo aleatorio B que toma tiempo promedio $T(n)$ pero que en el peor caso no tiene tiempo acotado, pero siempre da una respuesta correcta.

   Use B para encontrar un algoritmo más rápido que A en el peor caso, con la misma o mejor probabilidad de fracaso.

   Respuesta: Ejecutamos B hasta un tiempo $T(n)$. Con probabilidad $1/2$ dará una respuesta correcta. Como es un algoritmo aleatorio cada ejecución es independiente y entonces si lo ejecutamos 3 veces, la probabilidad de no tener una respuesta es $1/8$ y en ese caso respondemos al azar. Este algoritmo toma tiempo $3T(n)$ y siempre termina.

b) Demuestre que una secuencia aleatoria de n números enteros positivos siempre tiene en promedio una subsecuencia (no necesariamente contigua) de números crecientes de largo a lo menos $\log n$ (Indicación: $\sum_{i=1}^{n} 1/i > \log n$).

   Respuesta: Diseñamos un algoritmo aleatorio que encuentre una subsecuencia creciente con largo promedio al menos $\log n$. Usamos el algoritmo para buscar el máximo de un arreglo y suponemos que la subsecuencia comienza en x_1. El número x_i será más grande que cualquiera de los i anteriores con probabilidad $1/i$. Así que el largo promedio de la subsecuencia es $\sum_{i=1}^{n} 1/i > \log n$. En realidad se puede demostrar que la subsecuencia es mucho más larga que esto con otros métodos.