CC4102 - Diseño y Análisis de Algoritmos
Auxiliar 9

Prof. Gonzalo Navarro; Aux. Mauricio Quezada

29 de Diciembre, 2011

1 Rank, revisited

Recuerde que $rank_b(B, i)$ se define como la cantidad de repeticiones del bit b en $B[1, i]$, con B de n bits.

1. Diseñe una estructura que permita soportar la operación $rank$ en tiempo constante usando $2n + o(n)$ bits adicionales.
2. Haga lo mismo, pero sólo usando $o(n)$ bits adicionales.

1.1 Solución

1.1.1 Parte I

- Cortar B en bloques de $b = (\log n)/2$ bits
- Almacenar un arreglo $R[1, n/b]$ con el valor de rank al comienzo de los bloques
- R ocupa $\frac{n}{b} \cdot \log n = 2n$ bits

$rank(B, i)$ se calcula como

- Descomponer $i = qb + r$, con $0 \leq r < b$
- La cantidad de 1's hasta qb es $R[q]$
- Por lo que hay que contar los 1's en $B[qb + 1, qb + r]$

Para esto, construyamos una tabla $T[0, 2^b - 1][0, b.1]$, tal que $T[x, r] =$ total de 1's en $x[1, r]$

- T necesita $2^b \cdot b \cdot \log b = 2^{\frac{\log n}{2}} \cdot \frac{\log n}{2} \cdot (\log \log n - 1)$
- lo cual es $\leq \frac{1}{2}\sqrt{n} \log n \log \log n = o(n)$ bits
- Por ejemplo, si $n = 2^{32}$, entonces T sólo necesita 512KB

Por lo tanto, la respuesta se calcula con

$$R[q] + T[B[qb + 1, qb + r], r]$$

1.1.2 Parte II

- Para ello, cortamos B también en superbloques de $s = \frac{(\log n)^2}{2} = b \cdot \log n$ bits
- En $S[1, n/s]$ almacenamos los valores de $rank$ al comienzo de los superbloques
- Ahora R sólo almacena el valor desde el comienzo del superbloque correspondiente

$$R[q] = rank(B, qb) - S[\lfloor q/\log n \rfloor]$$

$$R[q] = rank(B, qb) - rank(B, \lfloor q/\log n \rfloor \cdot \log n)$$

- S ocupa $\frac{n}{s} \cdot \log n = \frac{n}{(\log n)^2/2} \cdot \log n = \frac{2n}{\log n} = o(n)$ bits.
- Como los valores de R se almacenan relativos al superbloque, sólo pueden llegar a valer $s = O(\log n)^2$, y por ello sólo necesitan $2 \log\log n$ bits, por lo tanto, R ocupa

$$\frac{n}{b} \cdot 2 \log\log n = \frac{n}{(\log n)/2} \cdot 2 \log\log n$$

$$= \frac{4n \cdot \log\log n}{\log n} = o(n)$$

Por lo tanto, $rank(B, i)$ se calcula como

- Descomponer $i = qb + r$
- Descomponer $i = q's + r'$
- $rank(B, i) = S[q'] + R[q] + T[B[qb + 1, qb + b], r]$

2 k-servers en una línea

Dé una función potencial Φ que demuestre un radio competitivo de k para el algoritmo Double Coverage (visto en la Auxiliar 7, parte 1.2).

3 Tom y Jerry

Jerry lanza platos desde lo alto de una repisa y Tom intenta recogerlos antes de que se estrellen contra el suelo del pasillo. El pasillo tiene $2k + 1$ baldosas y Tom recorre una baldosa por segundo. Cada plato tarda k segundos antes de llegar al suelo desde que Jerry lo lanza. Tom sabe en qué baldosa caerá el plato en el instante en que Jerry lo lanza, de manera que puede recorrer hasta k baldosas para salvarlo. Jerry lanza un plato por segundo, y lo puede lanzar hacia la baldosa que quiera. Nos interesa diseñar una estrategia *competitiva* para Tom, en términos de la cantidad de platos salvados.

1. Muestre que la estrategia de ir a buscar el siguiente plato lanzado en caso de que sea posible alcanzarlo (e ignorar todo el resto hasta recogerlo), y sino seguir en el mismo lugar esperando el próximo plato, no es competitiva.

2. Muestre que la estrategia de ir a buscar siempre el plato más cercano al suelo tampoco es competitiva

3. Diseñe una estrategia $2k$-competitiva para Tom. Demuestre su competitividad y encuentre un caso donde se salve sólo 1 de cada $2k$ platos que podría salvar el algoritmo óptimo.

3.1 Solucion

3.1.1 Parte I

Tirar un plato en una esquina, esperar a que llegue, y luego tirar $n-1$ en la otra, donde no irá a buscarlos (o infinitos platos en la otra esquina); el óptimo ignoraría el primero y salvaría todo el resto.

3.1.2 Parte II

Tirar uno en -1, y los demás en $1, 2, 3, \ldots, k, k-1, k-2, \ldots$, etc. Los pierde todos. El óptimo ignoraría el primero y salvaría el resto.

3.1.3 Parte III

Pararse al medio e ir a buscar el próximo plato que salga (siempre llegará), y volver al medio. En el peor caso tarda $2k$ y salva 1 plato. El óptimo puede salvar $2k$ platos, por lo que la estrategia es $2k$-competitiva. Se llega a esta tasa si, por ejemplo, tiran un plato en $+k$ y $2k-1$ platos en $-k$.