Aux #12

2015年12月3日 (木)

P2 n enteros $\in \{0, .., k\}$
$\rightarrow$ $n(a,b) \rightarrow$ # en t. cte.
Tiempo $\theta(n+k)$ de preproc.

Sol: Si construyo la tabla de conteos y luego hago sumas parciales en una tabla T.

$$T[i] = \sum_{j=0}^{i} x_j$$

elementos $\leq i$

↑ # de apariciones de j, suma de frecuencias acumuladas

Dado $A[1,n]$

$\theta(n+k)$
- $\theta(k)$ { for $i = 0..k$: $C[i] \leftarrow 0$
- $\theta(n)$ { for $i = 1..n$: $C[A[i]]++$ // Aquí $C[i] = x_i$, uso Counting Sort
- $\theta(k)$ { for $i = 0..k$: $C[i] = C[i] + C[i-1]$ // considerando $C[-1] = 0$

Luego $n(a,b) = C[b] - C[a-1]$
Esta respuesta es la que toma tiempo cte.

P3 m procs
lista de tareas $t_1 .. t_n$
con tiempos de procesamiento $p_1 .. p_n > 0$

- Algoritmo online conoce p_i solo cuando t_i llega.
- Carga $\sum_{t_i \in proc} p_i$

Algoritmo: enviar la tarea al proc. con menor carga.

1.- Dem que es $\left(2-\frac{1}{m}\right)$-competitivo.

- "Hint". Si s es el procesador con mayor carga, ¿qué ocurre con el tiempo de ocio de los demás?
- Si no soy s, mi tiempo de ocio es $\leq$ a la última tarea asignada a s.
  → pues de lo contrario esa tarea me habría sido asignada
  $\Rightarrow t_{ocio} \leq \max_{1 \leq i \leq n} p_i$

Por otro lado

$m \cdot C(I) = \sum \text{cargas} + \sum t_{ocio}$

↑ costo total del alg.

$\Rightarrow m\,C(I) \overset{(*)}{\leq} \sum p_i + (m-1) \max_{1 \leq i \leq n} p_i$

Por otro lado,

$OPT(I) \geq \frac{1}{m} \sum_{i=1}^{n} p_i$ (no puedo demorar menos que un caso ideal con $t_{ocio} = 0$)

Además, $OPT(I) \geq \max_{1 \leq i \leq n} p_i$ (no puedo demorar menos que la tarea + larga)

$\Rightarrow C(I) \leq OPT(I) + \left(\frac{m-1}{m}\right) \cdot OPT(I) = \left(2-\frac{1}{m}\right) OPT(I)$

$\Rightarrow$ el algoritmo es $\left(2-\frac{1}{m}\right)$-competitivo.

2.- Demostrar que la cota es óptima para el algoritmo.
O sea, que $\exists$ un caso I^* en que $C(I^*) = \left(2 - \frac{1}{m}\right) OPT(I^*)$

Consideraremos $m \cdot (m-1)$ tareas de costo 1 y luego otra de costo por determinar.
El algoritmo online va a dejar los m procs con carga $(m-1)$
Ahora llega una tarea de costo x
$\Rightarrow$ costo $= (m-1+x)$ (uno de ellos tendrá carga máx $m-1+x$)

¿Qué haría OPT? OPT conoce todas las tareas.
- Por ejemplo, sabiendo que x puede ser grande, reservas un proc para x.

$\Rightarrow$ Uso $(m-1)$ procs para las de tamaño 1 $\longrightarrow$ $m-1$ procs con carga m
$\Rightarrow$ costo $= \max(m-1, x)$

$\Rightarrow \frac{C(I)}{OPT(I)} = \frac{m-1+x}{\max(m-1,x)} \overset{\text{buscando } 2-\frac{1}{m}}{\Longrightarrow} x = m \Rightarrow \frac{2m-1}{m} = 2 - \frac{1}{m}$

P4 X, $|X| = n$, $\circ$, $x \circ y \ldots ?$
$\forall x, y, z \in X$, $(x \circ y) \circ z = x \circ (y \circ z)$

- Naive: chequear todas las tuplas (x, y, z) $\Rightarrow$ tiempo $\Theta(n^2)$
- Como con otros casos (mult. de matrices, etc) podría buscar un testigo (x^*, y^*, z^*) o similar. El problema es que los testigos pueden no ser densos. Existe una familia de ops binaria tq el número de testigos es cte.

Escogemos conjuntos de tuplas como testigos

Sea $P = P(X)$

Cada $R \in P$ puede verse como un vector binario de largo n, donde el i-ésimo bit denota si el i-ésimo elemento de X está o no en P

$$\left(R = \sum_{i \in X} r_i \, i\right)$$

Definimos: $R + S = \sum_{i \in X} (r_i \mid s_i) \cdot i$

$$R \circ S = \sum_{i,j \in X} (r_i \, s_j)(i \circ j) \in P$$

$$\alpha R = \sum_{i} (\alpha r_i)\, i \quad \text{para } \alpha \text{ cte.}$$

- Algoritmo:
  - elegir R, S, T de P (uniforme, indep)
  - si $(R \circ S) \circ T \neq R \circ (S \circ T)$ } $\Theta(n^2)$
    - responder no ← seguro
  - else responder sí ← puede equivocarse.

- $\circ$ es asociativa en X $\iff$ $\circ$ es asociativa en P (fácil de mostrar (?))

Prop: Si "$\circ$" NO es asociativa, al menos $\frac{1}{8}$ de las tuplas R, S, T son testigos. ($\frac{1}{8} = \frac{7}{8}$)

O sea, $\mathbb{P}[(R \circ S) \circ T = R \circ (S \circ T)] \leq \frac{7}{8}$

- Vamos a particionar P^3 en grupos de 8, tq cada grupo tenga al menos 1 testigo.

Si "$\circ$" no es asociativa, $\exists (i^*, j^*, k^*)$ tq $(i^* \circ j^*) \circ k^* \neq i^* \circ (j^* \circ k^*)$

- Sea R_0 tq $i^* \notin R_0$, S_0 tq $j^* \notin S_0$, T_0 tq $k^* \notin T_0$

Luego llamamos $R_1 = R_0 \cup \{i\}$
$S_1 = S_0 \cup \{j\}$
$T_1 = T_0 \cup \{k\}$

Sea $f(R,S,T) = \sum_{\substack{i \in R \\ j \in S \\ k \in T}} f(i,j,k)$ con $f(i,j,k$