Análisis de un Algoritmo en un Ambiente Distribuido

Resumen

Estudiamos y presentamos los elementos del Modelo de Computo Distribuido, en el cual se definió de manera formal sus elementos, asi como se formalizo lo que significa la ejecución del algoritmo en un modelo de computo Distribuido
Se presentaran y definirán las nociones del análisis de un Algoritmo Distribuido, como son:

- Complejidad del Algoritmo,i.e Complejidad temporal.]
- Correctud del Algoritmo]

Definicion de complejidad de un Algoritmo ejecutado en un modelo de computo: $LOCAL$**.**

Como en un modelo secuencial la medida de complejidad en el sentido temporal es el numero de pasos que toma del inicio al final, entonces inspirado en esa noción de complejidad secuencial definimos el siguiente concepto

Complejidad Temporal de un Algoritmo

Definición:
Sea G una gráfica y π un algoritmo entonces definimos la complejidad de π de manera *Sincrona* como el numero de rondas que se generaron durante la ejecución de π en G

De igual manera se puede definir la complejidad de una manera *Asincrona* como las unidades de tiempo que se generan desde el inicio del algoritmo π en G hasta el final de su ejecución.

Noción de Correctud del Algoritmo.
Definición

Diremos que un Algoritmo es correcto si:

- Resuelve el problema computacional por el cual fue diseñado
- Para cada entrada, produce la salida deseada
- Termina en un tiempo de ejecución finito

Definicion(Notacion Big-Oh): Si f y g son funciones de $\mathbb{Z}$ en $\mathbb{Z}$ entonces decimos que:

- $f = O(g)$ si existe una constante c tal que: $f(n) < \cdot g(n)$ para todo n suficientemente grande.
- Diremos que $f = \theta(g)$ si $f = O(g) y g = O(g)$

Entonces con base a esa definición podemos encapsular la complejidad(Temporal) del Algoritmo en términos de las definiciones anteriores, entonces podemos decir que la complejidad esta dada de la siguiente manera: *como la complejidad temporal es dada con sintaxis y semántica*:

$Time(\pi, G) = O(g(n)$ donde g dependerá de la naturaleza del algoritmo en cuestión.

Algorithm 1 First-dis

loop

$A_{first}(T_{r_0}, Id, n \in \mathbb{N})$

if p_i es una hoja **then**

return $M =< (x_i, 1 >$

else

Esperar los $M_{1j}, ..., M_{kj}$ mensajes de sus hijos

$$M_{ik} \leftarrow < (A_{ik}, m \in \mathbb{N})$$

$$A_i \leftarrow \cup_{j=1}^{k} A_{ij} \cup \{x_i\}$$

for all $x_l \in A_i$ **do**

$$B_{x_l} \leftarrow \{x_p \in A_i : x_p = x_l\}$$

end for

$$M \leftarrow \{B_{x_l} : x_l \in A_i\}$$

$$N = \{|B_{x_l}| : B_{x_l} \in M\}$$

$$n - A_i = d$$

while

$$d + |B_{i \neq x_p}| < max(N) = |B_{x_p}| \ \forall B_{i \neq x_p} \in M$$

do

return

$$< (B_{x_p}, max(N) = |B_{x_p}|) >$$

end while

end if

end loop

Teorema: El Algoritmo $\pi = A_{first}$ produce el elemento mas repetido del Arbol T_{r_0} i.e en terminos estadisticos el modo.

***Demostracion*:** Si en una iteracion del algoritmo, v_j es tal que

$$\exists B_{x_m}$$

con

$$|B_{x_m}| = max(N) \; y \; |B_{x_m}| > (n - |A_{x_i}|) + |B_{x_i}|$$

donde $|B_{x_i} \in M$ es arbitrario.
Afirmacion: x_m es el numero mas repetido, i.e es el modo

Observamos que

$$n = \sum_{x_p} |B_{x_p}|$$

y que

$$|A_j| = \sum_{x_j} |B_{x_j}|, x_j \in A_j$$

entonces podemos escribir

$$n - |A_j| = \sum_{x} p|B_{x_p}| - \sum_{x_j} |B_{x_j}|$$

$$= \sum_{x_p \not\in A_j} |B_{x_p}|$$

como $n - |A_j| < max(N) \therefore$

$$\sum_{x_p \not\in A_j} |B_{x_p}| + |B_{x_i} < max(N)$$

$\therefore$

$$|B_{x_p}| < max(N) = |B_{x_m}| \forall x_p \neq x_m$$

$\therefore x_m$ es el elemento mas repetido en T_{r_0}
Para ver como es la complejidad temporal del algoritmo A_{first} observemos lo siguiente:

Si tomamos el algoritmo *Convergecast* y aplicamos el problema en dicho algoritmo, entonces

$$CONVERGE(MODO) \; en \; T_{r_0}$$

Entonces $Time(CONVERGE) = O(Depth(T_{r_0}))$ con respecto a r_0

Entonces en $\pi = A_{first}$ estamos optimizando con base a los inputs el Algoritmo *CONVERGE* pues en general x_p no es necesariamente la raiz del Arbol T_{r_0} por lo tanto:

$$Time(A_{first}) = O(depth(T_{r_0}))$$

con respecto al x_m para el cual $max(N) = |B_{x_m}|$ y cumple la condicion **WHILE**