Demostración complejidad Wave

Nicolás Cagliero

24 de junio de 2024

Theorem 1. *El algoritmo Wave tiene complejidad* $O(n^3)$

Demostración. Esta será una idea de la demostración y sus partes, los detalles y ciertas cuentas no serán escritas.

Sabemos que la distancia en Networks Auxiliares sucesivos aumenta, por ende sabes que vamos a tener $O(n)$ NA en una corrida de Wave. Luego sabemos que $O(Wave) = O(n) \cdot O($compl(hallar un flujo bloqueante en un NA)+compl(construir un NA))

La complejidad de armar un NA podemos ver que está dada por la cantidad de lados que puede llegar a tener $\therefore$ compl(construir un NA) $= O(m)$

Si logramos probar que la compl(hallar un flujo bloqueante en un NA) $= O(n^2)$ entonces queda demostrado el teorema.

Vamos a dividir esta complejidad en varias partes, primero diferenciaremos entre olas hacia adelante y olas hacia atrás:

1. Olas hacia adelante:

   Cuando un vértice x le manda flujo a un vértice z, puede pasar que $\overrightarrow{xz}$ se sature o no. Definimos entonces

   a) $S =$ todos los casos donde se satura un lado.

   b) $P =$ todos los casos donde no se satura un lado.

   Veamos en detalle a S, supongamos que $\overrightarrow{xz}$ se saturó, ¿Puede volver a saturarse? Para que esto suceda, primero tiene que desaturarse $\therefore$ z debe devolverle flujo a $x \Rightarrow z$ debe estar bloqueado. Pero en las olas hacia adelante, para mandar flujo de un vértice, me fijo en sus vecinos no bloqueados, por lo tanto no puedo mandarle flujo a z (bloqueado) $\Rightarrow$ no puede volverse a saturar $\overrightarrow{xz}$

   A P lo analizaremos más adelante.

2. Olas hacia atrás:

   Cuando un vértice x le devuelve flujo a un vértice u, puede pasar que $\overrightarrow{ux}$ quede vacío o no. Definimos entonces

   a) $V =$ todos los casos donde se vacía un lado.

   b) $Q =$ todos los casos donde no se vacía un lado.

Veamos en detalle a V, supongamos que $\overrightarrow{ux}$ se vació, ¿Puede volver a vaciarse? Para que esto suceda, primero tiene que llenarse, es decir, necesita que u le envíe flujo a x pero esto no puede pasar porque x está bloqueado (le devolvió flujo a u).

Veamos ahora P y Q. Al intentar balancear hacia adelante, en cada vértice vamos a lograr saturar todos los lados, salvo quizá el último, que no se satura y cuenta para la complejidad de P. Enviar flujo por ese lado sabemos que es $O(1)$ $\therefore$ la complejidad de $P = O(n)\cdot$ (#olas hacia adelante)

Similar con Q, al balancear hacia atrás. Podremos vaciar todos los lados, salvo quizá el último, que no se vacía y cuenta para Q. Luego la complejidad de $Q = O(n)\cdot$(#olas hacia atrás).

Sabemos que #(olas hacia atrás) = #(olas hacia adelante). Veamos qué pasa en cada ola hacia adelante. En cada ola, salvo quizá en la última, algún vértice quedará desbalanceado y la ola lo bloquea $\therefore$ en cada ola, salvo quizá la última, se bloquea al menos un vértice y este nunca se desbloquea. $\Rightarrow O(n) =$ #olas.

Luego, $O(S) = O(n)$, $O(V) = O(n)$, $O(Q) = O(n^2)$, $O(P) = O(n^2)$

$\therefore$ la complejidad de encontrar un flujo bloqueante es $O(m)+O(m)+O(n^2)+O(n^2) = O(m)+O(n^2) = O(n^2)$

$\square$