En MKM, en cada PP hay como una "ola" forward que arrastra flujo hacia adelante en los PUSH, y una "ola" hacia atras en los PULL. Como se comienza en el vertice de menor capacidad, esas olas siempre llegan a destino y nunca se produce un cuello de botella. Pero, luego hay que largar una ola en otro vertice, y como siempre se manda por el de menos capacidad, MKM tiende a mandar por vez menos flujo del que se podria y aparentemente se demora en la practica.

Wave, como su nombre lo indica, opera por "olas", pero en forma distinta: Primero se manda una ola hacia adelante, partiendo de s, que mas que una ola es un tsunami, pues lleva todo lo que se puede llevar desde s. Por supuesto que si s es el vertice de menor capacidad, en realidad esto seria como PUSH de MKM, y listo. PEro que pasa si s no es el vertice de menor capacidad? Entonces la ola se topara con cuellos de botellas, es decir, habra vertices x en los cuales hay entrando mas flujo del que sale, aun luego de que la ola llegara a t. Entonces, lo que se le ocurrio a Tarjan es usar una ola que devuelva flujo: lanza una ola retrograda desde t, que va devolviendo flujo desde los vertices hacia sus vecinos hacia atras. Pero ¿cuanto flujo devolver? Simplemente, el desbalanceo del vertice. Luego que esta ola llega a t, se puede re-intentar mandar flujo por otro lado, asi que se vuelve a lanzar una ola forward, y asi sucesivamente. Para evitar re-visitar vertices que ya sabemos que en ellos se produciran cuellos de botella, una vez que una ola descubre un cuello de botella, se marca el vertice como "bloqueado", y las sucesivas olas forwards lo ignoraran.

Un ultimo detalle que queda: supongo que se podria en la ola backward simplemente devolver flujo desde todos los vertices desbalanceados, pero Tarjan no hace esto: solamente devuelve flujo desde aquellos vertices que esten bloqueados. Lo que podria pasar es que un vertice puede haber quedado balanceado en la ola forward, pero alguno de sus vecinos forward puede haberse bloqueado, cuando este vecino le devuelve flujo, el vertice queda desbalanceado, pero en una de esas podria mandar flujo a otro vertice forward, asi que Tarjan no lo hace devolver flujo, sino que lo deja "en espera" de la proxima ola forward. Si esta no es exitosa, entonces recien ahi lo bloquea, y entonces si la ola backward lo balanceará. Esta forma de implementarlo hace que sea facil saber cuando parar: simplemente se para cuando todos los vertices estan balanceados: i.e., el invariante aca es que el pre-flujo ES BLOQUEANTE de entrada. Lo que hace Wave es ir ajustando el pre-flujo hasta que se balancean todos los vertices, i.e., hasta que se vuelve flujo. Como el invariante de ser bloqueante nunca cambia, al final el flujo que se obtiene es bloqueante.

De la otra forma, i.e., devolviendo desde todos los balanceados, y luego volviendo a mandar, si bien creo que se podria probar que es mas o menos equivalente a Wave, no tiene esto como invariante, asi que supongo que habria que poner otra condicion de STOP, y no se bien como seria el analisis. Supongo que uno seguiria mandando flujo hasta que s se vuelva bloqueado.

En la implementacion de Tarjan, Tarjan bloquea s en el comienzo.

Una observacion: Este detalle de Wave hace que si hay una serie de vertices que solo tienen vecinos a lo largo de un camino muy largo con solo un vertice en el extremo bloqueado, el metodo Wave hara una ola forward y una backward por cada vertice en el camino, hasta finalmente bloquearlos a todos. De todos modos, la variación de Wave de devolver todo hasta s no soluciona este problema.

Usaremos un registro $B(x)$ que nos dira si x esta o no , y uno $D(x)$ que nos dira cuanto es el desbalanceo de x.

Una aclaracion: en Wave no hace falta saber quienes son los vertices hacia atras, excepto para devolver flujo. Asi, supondremos que solo estan definidos los $\Gamma^+(x)$, y que se han definido por default $\Gamma^-(x) = \emptyset$ para todos los x. (i.e., esto no corresponde a una definicion normal de network, puesto que si $y \in \Gamma^+(x)$, entonces $x \in \Gamma^-(y)$, pero en este caso consideraremos estos como conjuntos que no necesariamente mantienen esa relación. Lo que en realidad haremos es construir los Γ^- solo si efectivamente mandamos flujo. (Alternativamente, se puede pensar que los $\Gamma^\pm$ son en realidad unos $\tilde{\Gamma}^\pm$ si se quiere)

Wave:

g:=0
FORALL $x \neq s$ DO:
$B(x) := 0$//No bloqueado
$D(x) := 0$//desbalanceo cero
ENDFOR
$N := \{s\}$//conjunto de no balanceados
FORALL $x \in \Gamma^+(s)$ DO:
$g(\overrightarrow{sx}) := c(\overrightarrow{sx})$//mandamos todo lo que podemos sin importarnos el futuro
$D(x) := c(\overrightarrow{sx})$//$x$ quedó desbalanceado
$N := N \cup \{x\}$//se agrega al conjunto de no balanceados
$\Gamma^-(x) := \{s\}$//mande flujo de s a x.
ENDFOR
WHILE $(N \neq \{s, t\})$ DO:
FOR $x =$"$s+1$" TO "$t-1$" DO: //BFS orden. (Begin INCREASEFLOW)
IF $((B(x) = 0))$ AND $(D(x) > 0)$) THEN FORWARDBALANCE(x)
//solo balanceamos los desbalanceados que no esten bloqueados.
ENDFOR//end INCREASEFLOW
FOR $x =$"$t-1$" TO "$s+1$" DO: //reverse BFS orden. (Begin DECREASEFLOW)
IF $((B(x) = 1))$ AND $(D(x) > 0)$) THEN BACKWARDBALANCE(x)
//solo balanceamos los desbalanceados que SI esten bloqueados.
ENDFOR//end DECREASEFLOW
ENDWHILE
OUTPUT(g)

FORWARDBALANCE(x):

WHILE $((D(x) > 0)$ AND $(\Gamma^+(x) \neq \emptyset))$ DO:
Tomar $y \in \Gamma^+(x)$
IF $B(y) = 1$ THEN $\Gamma^+(x) := \Gamma^+(x) - y$//si esta bloqueado no podemos mandar flujo
ELSE $A := Min\{D(x), c(\overrightarrow{xy}) - g(\overrightarrow{xy})\}$
$g(\overrightarrow{xy}) := g(\overrightarrow{xy}) + A$
$D(x) := D(x) - A$
$D(y) := D(y) + A$
$N := N \cup \{y\}$//como conjunto, si y ya estaba, no hace nada.
$\Gamma^-(y) := \Gamma^-(y) \cup \{x\}$
IF $g(\overrightarrow{xy}) = c(\overrightarrow{xy})$ THEN $\Gamma^+(x) := \Gamma^+(x) - y$ ENDIF
ENDIF
ENDWHILE
IF $D(x) > 0$ THEN $B(x) := 1$//si luego de todo quedo desbalanceado, se bloquea
ELSE $N := N - x$//si $D(x) = 0$, debemos sacarlo de los no balanceados
ENDIF

BACKWARDBALANCE(x):

WHILE $D(x) > 0$ DO://backward balance siempre tiene exito
Tomar $y \in \Gamma^-(x)$
$A := Min\{D(x), g(\overrightarrow{yx})\}$//lo maximo que puedo devolver
$g(\overrightarrow{yx}) := g(\overrightarrow{yx}) - A$
$D(x) := D(x) - A$
$D(y) := D(y) + A$
$N := N \cup \{y\}$
IF $g(\overrightarrow{yx}) = 0$ THEN $\Gamma^-(x) := \Gamma^-(x) - y$ ENDIF
ENDWHILE
$N := N - x$