

# Teoremas para el final de matemática discreta II 2024

Tomás Maraschio

18 de junio de 2024

## Índice

1. ¿Cuál es la complejidad del algoritmo de Wave? No hace falta probar que la distancia en NAs sucesivos aumenta 1
  2. Probar que el valor de todo flujo es menor o igual a la capacidad de todo corte. También que si  $f$  es flujo, es maximal si y solo si existe  $S$  corte con  $V(f) = \text{Cap}(S)$  y  $S$  es minimal. No hace falta probar que  $V(f) = f(S, \bar{S}) - f(\bar{S}, S)$  3
  3. Probar que si  $G$  es un grafo conexo no regular entonces  $\chi(G) \leq \Delta(G)$  4
- 
1. ¿Cuál es la complejidad del algoritmo de Wave? No hace falta probar que la distancia en NAs sucesivos aumenta

Probaré que  $\text{Compl}(\text{Wave}) = O(n^3)$ .

Al igual que en Dinic, como la distancia entre  $s$  y  $t$  aumenta en NAs sucesivos, tengo  $O(n)$  NAs, pues  $t$  no puede estar a una distancia mayor a  $n$  de  $s$ . Ahora, como

$$\text{Compl}(\text{Wave}) = (\text{Compl}(\text{hallar flujo bloqueante}) + \text{Compl}(\text{construir NA})) \times O(n)$$

y  $\text{Compl}(\text{construir NA}) = O(m)$  pues uso BFS, bastaría probar que

$$\text{Compl}(\text{hallar flujo bloqueante}) = O(n^2).$$

Notar que  $O(n^2) + O(m) = O(n^2)$  pues  $m \leq \binom{n}{2} = O(n^2)$ .

Voy a dividir el proceso de hallar un flujo bloqueante en casos.

Cuando estoy balanceando vértices hacia adelante:

- V: los pasos donde sature un lado
- P: los pasos donde no sature un lado

Cuando estoy balanceando los vértices hacia atrás:

- S: los pasos donde vació un lado
- Q: los pasos donde no vació un lado

V: Sup. que  $x$  le manda flujo a  $z$  y  $\overrightarrow{xz}$  se satura. Para que  $\overrightarrow{xz}$  vuelva a saturarse, primero tiene que vaciarse un poco, pero si se vacía es porque  $z$  se bloqueó y entonces le devolvió flujo a  $x$ . Pero como  $z$  está bloqueado (y nunca se desbloqueará),  $x$  no le va a volver a mandar flujo a  $z$ . Entonces, cada lado puede saturarse a lo sumo una vez.  $\therefore Compl(V) = O(m)$ .

S: Sup. que  $x$  le devuelve flujo a  $z$  y  $\overrightarrow{zx}$  se vacía. Como  $x$  devolvió flujo significa que está bloqueado, entonces  $z$  no volverá a mandarle flujo, entonces  $\overrightarrow{zx}$  no se podrá llenar para volver a vaciarse. Esto implica que un lado se puede vaciar a lo sumo una vez.  $\therefore Compl(S) = O(m)$ .

P: Cuando un vértice manda flujo a sus vecinos para balancearse, satura todos los lados, excepto quizás uno. Esto significa que para cada vértice en cada ola hacia adelante se satura parcialmente a lo sumo un lado. Entonces  $Compl(P) = O(n) \times \#olas \text{ hacia adelante}$ .

Ahora, en cada ola hacia adelante (excepto la última) queda al menos un vértice desbalanceado, es decir que cada ola hacia adelante hace que se bloquee al menos un vértice. Y como no se desbloquean una vez bloqueados, tengo que  $\#olas \text{ hacia adelante} = O(n)$ .  $\therefore Compl(P) = O(n^2)$ .

Q: Cuando un vértice devuelve flujo a los vecinos, vacía todos los lados excepto quizás uno. Esto significa que para cada vértice en cada ola hacia atrás se vacía parcialmente a lo sumo un lado. Entonces  $Compl(Q) = O(n) \times \#olas \text{ hacia atrás} = O(n) \times \#olas \text{ hacia adelante} = O(n^2)$ .

**Finalmente,**  $Compl(\text{hallar flujo bloqueante}) = O(m) + O(n^2) + O(m) + O(n^2) = O(n^2)$ .

**2. Probar que el valor de todo flujo es menor o igual a la capacidad de todo corte. También que si  $f$  es flujo, es maximal si y solo si existe  $S$  corte con  $V(f) = \text{Cap}(S)$  y  $S$  es minimal. No hace falta probar que  $V(f) = f(S, \bar{S}) - f(\bar{S}, S)$**

**2.1. Valor de todo flujo menor o igual a capacidad de todo corte**

$$f(\bar{S}, S) = \sum_{\substack{x \notin S \\ y \in S \\ \vec{xy} \in E}} f(\vec{xy}) \geq 0 \text{ pues } f(\vec{xy}), \forall \vec{xy} \in E$$

Entonces tengo  $V(f) = f(S, \bar{S}) - f(\bar{S}, S) \leq f(S, \bar{S}) \leq c(S, \bar{S}) = \text{Cap}(S)$

**2.2.  $\Leftarrow$  vuelta**

Sea  $f$  flujo y  $S$  corte con  $V(f) = \text{Cap}(S)$ . Para todo flujo  $g$ , por lo que probé arriba tengo  $V(g) \leq \text{Cap}(S) = V(f) \implies f$  es maximal. Además, para todo corte  $T$  tengo  $\text{Cap}(T) \geq V(f) = \text{Cap}(S) \implies S$  es minimal.

**2.3.  $\implies$  ida**

Sea  $f$  flujo maximal. Voy a construir un  $S$  y probaré que es un corte minimal con  $V(f) = \text{Cap}(S)$ .

$$S = \{s\} \cup \{x : \text{existe } f\text{-camino aumentante de } s \text{ a } x\}$$

**$S$  es corte**

Sup. que no lo es. La única forma que pase esto es porque  $t \notin S$ , pues  $s \in S$ . Pero esto significa que existe un  $f$ -camino aumentante de  $s$  a  $t$ , por lo que puedo aumentar  $f$  en ese camino. Absurdo porque  $f$  es maximal, luego  $S$  es corte.

**$V(f) = \text{Cap}(S)$**

Sean  $x \in S, y \notin S, \vec{xy} \in E$ .

$$x \in S \implies \text{existe } f\text{-camino aumentante de } s \text{ a } x$$

$y \notin S \implies$  no existe f-camino aumentante de  $s$  a  $y$

Entonces el camino  $s \cdots xy$  podría ser aumentante, pero no lo es, y esto solo puede ser si  $f(\vec{xy}) = c(\vec{xy})$ . Entonces

$$f(S, \bar{S}) = \sum_{\substack{x \in S \\ y \notin S \\ \vec{xy} \in E}} f(\vec{xy}) = \sum_{\substack{x \in S \\ y \notin S \\ \vec{xy} \in E}} c(\vec{xy}) = c(S, \bar{S}) = \text{Cap}(S)$$

Ahora sean  $x \notin S, y \in S, \vec{xy} \in E$ .

$x \notin S \implies$  no existe f-camino aumentante de  $s$  a  $x$

$y \in S \implies$  existe f-camino aumentante de  $s$  a  $y$

Entonces el camino  $s \cdots \overleftarrow{yx}$  podría ser aumentante, pero no lo es, y esto solo puede ser si  $f(\vec{xy}) = 0$ . Entonces

$$f(\bar{S}, S) = \sum_{\substack{x \notin S \\ y \in S \\ \vec{xy} \in E}} f(\vec{xy}) = \sum_{\substack{x \notin S \\ y \in S \\ \vec{xy} \in E}} 0 = 0$$

Finalmente:  $V(f) = f(S, \bar{S}) - f(\bar{S}, S) = \text{Cap}(S) - 0 = \text{Cap}(S)$

### 3. Probar que si $G$ es un grafo conexo no regular entonces $\chi(G) \leq \Delta(G)$

Sea  $x$  tal que  $d(x) = \delta(G)$ . Corro BFS empezando por  $x$  y guardo el orden inverso en que los vértices se visitaron. Ahora voy a correr greedy en este orden que acabo de guardar. Notar que en BFS todo vértice es incluido por un vértice que ya ha sido visitado, entonces en el orden inverso todo vértice tiene al menos un vecino por delante (excepto el  $x$ ). Esto significa que para cada vértice  $y$ , en el peor caso va a tener  $d(y) - 1 < \Delta$  vecinos coloreados todos con un color distinto, entonces voy a poder elegir un color en  $\{1, \dots, \Delta\}$ . Finalmente, cuando llego a  $x$ , como  $d(x) = \delta < \Delta$ , podré elegir algún color de  $\{1, \dots, \Delta\}$  que no está usado por ningún vecino de  $x$  para colorearlo. De esta forma tengo un coloreo propio de  $G$  que usa (a lo sumo)  $\Delta$  colores, entonces  $\chi(G) \leq \Delta(G)$ .