Clase 2 de Flujos: Greedy.

Who?

Daniel Penazzi

When?

9 de abril de 2021

Tabla de Contenidos

Criterio simple para maximalidad

- Si queremos ver que f es máximal, usando directamente la definición, tenemos un problema.
- Requeriria probar que $v(g) \leq v(f)$ PARA TODO flujo g.
- Dado que hay infinitos flujos esto en general podria no ser simple.
- Asi que en principio deberiamos hacer un análisis delicado para probar eso.
- Pero en algunos casos podemos dar una prueba corta.

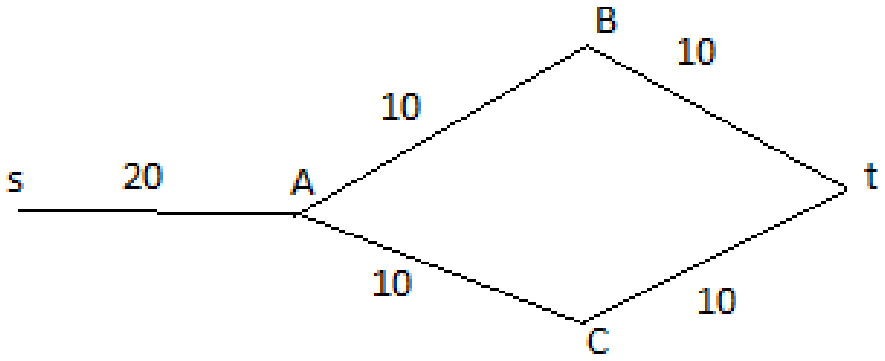
Ejemplo simple

- En el siguiente ejemplo: (nota: en los dibujos siempre asumiremos que los lados van de izquierda a derecha)

- Es obvio que el flujo maximal es $f(\overrightarrow{st}) = 10$.

Ejemplo simple

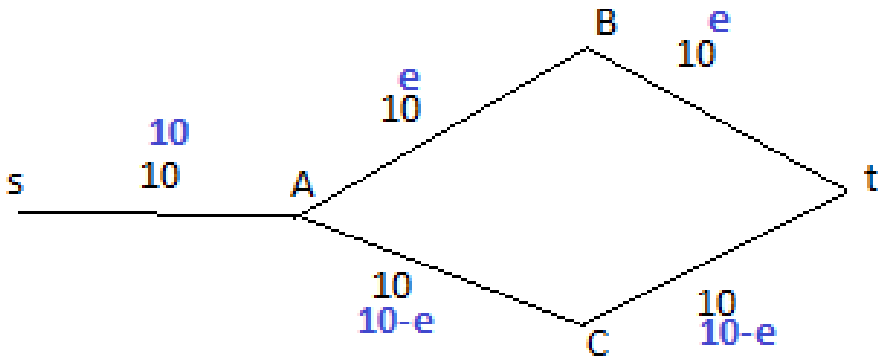
- O bien



- Es obvio que el flujo maximal es $f(\overrightarrow{xy}) = 10 \forall \overrightarrow{xy} \in E$.
- Lo mismo en este otro ejemplo, excepto el lado $\overrightarrow{sA}$ con $f(\overrightarrow{sA}) = 20$
- En esos ejemplos tenemos que $f(\overrightarrow{xy}) = c(\overrightarrow{xy}) \forall \overrightarrow{xy}$ asi que claramente son maximales porque ningún flujo puede mandar mas que la capacidad de un lado.

Ejemplo no tan simple

- Pero no siempre podremos hacer eso. Pej:



- Si mandamos $f(\overrightarrow{xy}) = 10 = c(\overrightarrow{xy}) \forall \overrightarrow{xy}$ no nos quedaria un flujo pues tendriamos $in_f(A) = 10$ pero $out_f(A) = 20$.
- Esta claro que si queremos un flujo maximal tenemos que mandar $f(\overrightarrow{sA}) = 10$, pero no queda claro qué hacer despues.

- O incluso algo como:

Ejemplo no tan simple

- Intuitivamente todos esos flujos son maximales pues estan saturando la salida por s.
- Demostraremos enseguida que esta intuición es correcta.
- Este ejemplo muestra que puede haber infinitos flujos maximales.
- En este ejemplo el valor de todos ellos es 10.
- En networks mas complicados quizás no sea tan fácil darse cuenta.
- Necesitamos algunas herramientas mas, pero empezaremos primero por las obvias.

Una propiedad obvia de la notación $g(A, B)$

- Recordemos que definimos en la clase anterior $g(A, B)$ cuando:
- g es una función sobre los lados y $A, B \subseteq V$
- como $g(A, B) = \sum_{x,y}[x \in A][y \in B][\overrightarrow{xy} \in E]g(\overrightarrow{xy})$
- Una propiedad obvia es la siguiente:

Propiedad:

Sean f, g funciones sobre los lados tales que

$$g(\overrightarrow{xy}) \leq f(\overrightarrow{xy}) \quad \forall \overrightarrow{xy} \in E$$

Entonces

$$g(A, B) \leq f(A, B) \quad \forall A, B \subseteq V$$

Prueba obvia de la propiedad obvia

$$\begin{aligned} g(A,B) &= \sum_{x,y}[x \in A][y \in B][\overrightarrow{xy} \in E]g(\overrightarrow{xy}) \\ &\leq \sum_{x,y}[x \in A][y \in B][\overrightarrow{xy} \in E]f(\overrightarrow{xy}) \\ &= f(A,B) \end{aligned}$$

Como ven, bastante obvio.

Criterio simple para maximalidad

Propiedad:

Sea f flujo en un network N tal que $v(f) = c(\{s\}, V)$. Entonces f es maximal.

- Prueba: Sea g un flujo cualquiera.
- Por la primera propiedad de un flujo, vale $g(\overrightarrow{xy}) \leq c(\overrightarrow{xy}) \; \forall \overrightarrow{xy} \in E$.
- Por la propiedad obvia anterior, entonces $g(A, B) \leq c(A, B) \; \forall A, B \subseteq V$.
- En particular $g(\{s\}, V) \leq c(\{s\}, V)$.

Continuación prueba

Pero:

$$\begin{aligned} v(g) &= out_g(s) - in_g(s) \\ &\leq out_g(s) \\ &= g(\{s\}, V) \qquad \text{(definición de out)} \\ &\leq c(\{s\}, V) \qquad \text{(lo que acabamos de probar)} \\ &= v(f) \qquad \text{(por la hipotésis del teorema)} \end{aligned}$$

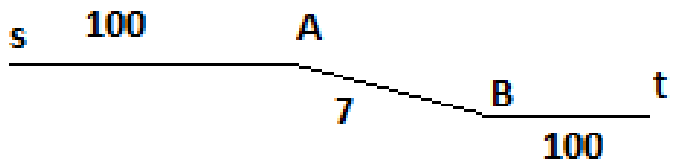
Por lo tanto f es maximal. Fin

Nota:Con una prueba similar, se ve que si $v(f) = c(V, \{t\})$, entonces f es maximal.

Ejemplo no tan simple

- Entonces en el ejemplo anterior, como en todos los flujos que definimos teniamos que $v(f) = 10 = c(\{s\}, V)$, todos ellos eran maximales.
- Esta condición es suficiente para probar que un flujo es maximal, pero no es necesaria.
- Pej, si en esos ejemplos reemplazamos t por D y luego agregamos un lado de D a t de capacidad 5, entonces los flujos maximales tendrán valor 5 y por lo tanto no será cierto que $v(f) = c(\{s\}, V)$.
- En ese ejemplo seria cierto que $v(f) = c(V, \{t\})$.
- Pero tampoco es necesariamente cierto que deba valer $v(f) = c(V, \{t\})$ o $v(f) = c(\{s\}, V)$.

Condiciones suficientes pero no necesarias



- Es claro que cualquier flujo de s a t debe pasar por $\overrightarrow{AB}$ y por lo tanto, no puede tener valor mas de 7, por feasability.
- y el flujo f que manda 7 por los tres lados tiene ese valor, asi que es maximal.
- Pero $v(f) = 7 < 100 = c(\{s\}, V) = c(V, \{t\})$

Existencia

- La clase pasada mencione que de la definición no es claro que EXISTA un flujo maximal.
- Si agregamos la condición de que el flujo sea "entero", es decir que las capacidades y el flujo en cada lado deben ser números enteros,
- entonces, como hay una cantidad finita de flujos enteros, es claro que existe un flujo entero maximal.
- Es decir, un flujo que es maximal entre todos los flujos enteros.
- Lo que no es claro es que el flujo entero maximal sea maximal entre TODOS los flujos.
- Por no hablar del caso en el que las capacidades no son enteros.

Existencia

- En algunos libros (p.ej. el Biggs) esto se ignora, porque facilita las pruebas
- Es decir, se asume que las capacidades y los flujos serán enteros.
- En el caso general se puede probar “fácilmente” que existe un flujo maximal.
- “Fácilmente” si se sienten cómodos manejando sucesiones, subsucesiones, subsubsucesiones, etc.
- Y usando muchas veces el teorema de Bolzano-Weiertrass.
- Hace unos años descubrí que han dejado de dar ese teorema a los alumnos de Computación.
- Asi que no me queda mas remedio que postergar la prueba de la existencia para mas adelante.

Encontrando flujos maximales

- Asumiendo que existe un flujo maximal ¿como hallarlo? (con un algoritmo polinomial).
- Incluso, mas básico: ¿como construir un flujo? (aun si no es máximal).
- Si colocamos numeros al azar en los lados, lo mas probable es que la función resultante NO sea un flujo.
- Poder construir flujos aunque no sean maximales resultará bastante central:
- CASI todos los algoritmos que veremos encuentran un flujo maximal construyendo flujos no maximales, cada uno con valor mas grande que el anterior, hasta llegar a un flujo maximal.

Encontrando flujos

- Idea principal: supongamos que ya tenemos un flujo f. ¿Cómo "cambiar" f pero de tal forma que lo que quede siga siendo un flujo?
- Consideremos un "cámino dirigido" entre s y t.
- Es decir, una sucesión de vértices que empiezan en s, terminan en t y que esta compuesta de lados en la dirección "correcta".
- $x_0, x_1, ..., x_r$ con:
  - $x_0 = s$
  - $x_r = t$
  - $\overrightarrow{x_i x_{i+1}} \in E \quad \forall i = 0, ..., r-1.$

Encontrando flujos

- Dado ese camino dirigido, podemos cambiar f a lo largo del camino, incrementando el valor de f en cada lado por una constante:
- $nuevaf(x_i\overrightarrow{x_{i+1}}) = f(x_i\overrightarrow{x_{i+1}}) + cte$.
- y manteniendo el valor de f en los otros lados como estaba.
- Cada vértice que no sea un x_i mantiene sus *in* y *out* como estaban.
- Y en los x_i ambos se incrementan en *cte*, por lo que su diferencia es la misma.
- Asi que se mantiene la regla de conservación de la definición de flujo.

Encontrando flujos

- Claro que hay que tener cuidado con la primera regla:
- No podemos hacer “explotar” un caño.
- Asi que la *cte* no puede ser tan grande como para que el nuevo flujo supere la capacidad de ninguno de los lados $\overrightarrow{x_i x_{i+1}}$.
- Pero, provisto que tengamos ese cuidado, la nueva función será un flujo.
- Como en todo network hay un flujo desde el cual empezar (el flujo nulo, que es igual a cero en todos los lados), ya tenemos la base de un algoritmo:

Encontrando flujos

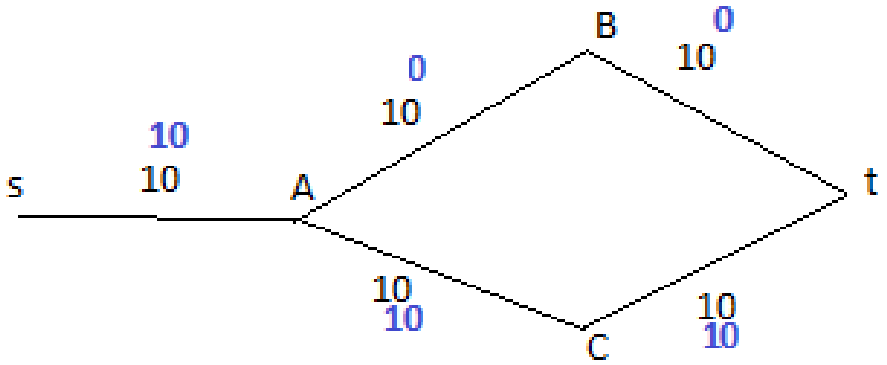
- Comenzando desde algún flujo (pej el nulo) ir encontrando caminos dirigidos desde s a t
- aumentando el flujo a lo largo de ese camino teniendo en cuenta de no mandar mas flujo por el mismo que lo que pueden soportar los lados.
- Una pregunta es, una vez detectado un camino y obtenido una cota superior para cuanto podemos mandar por ese camino: ¿cuanto mandamos?
- Lo mas obvio sería ser greedy y mandar todo lo que se pueda, y eso hace el algoritmo Greedy

Greedy

Algoritmo Greedy para hallar flujo maximal

1. Comenzar con $f = 0$ (es decir, $f(\overrightarrow{xy}) = 0 \forall \overrightarrow{xy} \in E$).
2. Buscar un camino dirigido $s = x_0, x_1, ..., x_r = t$, con $\overrightarrow{x_i x_{i+1}} \in E$ tal que $f(\overrightarrow{x_i x_{i+1}}) < c(\overrightarrow{x_i x_{i+1}})$ para todo $i = 0, ..., r-1$.
3. (llamaremos a un tal camino un camino dirigido “no saturado” .)
4. Calcular $\varepsilon = \min\{c(\overrightarrow{x_i x_{i+1}}) - f(x_i x_{i+1})\}$.
5. Aumentar f a lo largo del camino de 2. en ε, como se explicó antes.
6. Repetir 2 hasta que no se puedan hallar mas caminos con esas condiciones.

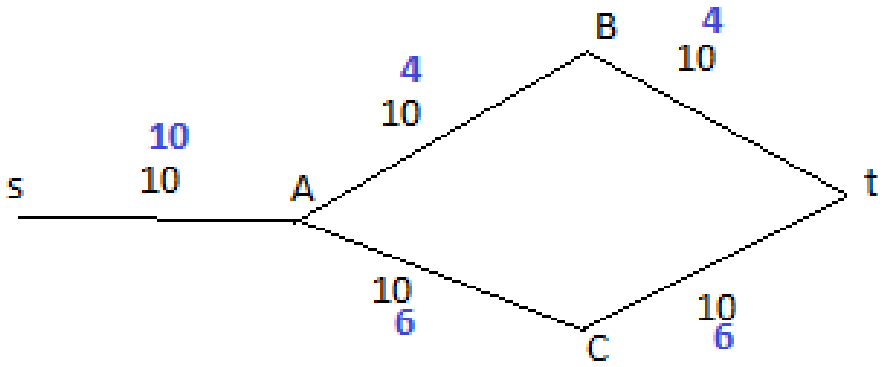
Ejemplo de Greedy



- Greedy, comenzando con $f = 0$, encuentra el camino s, A, B, t y aumenta f en 10 a lo largo de ese camino.
- No existen mas caminos dirigidos no saturados, Greedy termina
- Otra posibilidad seria que comenzando con $f = 0$, se encontrara el camino s, A, C, t y aumentara f en 10 a lo largo de ese camino.

Ejemplo de Greedy

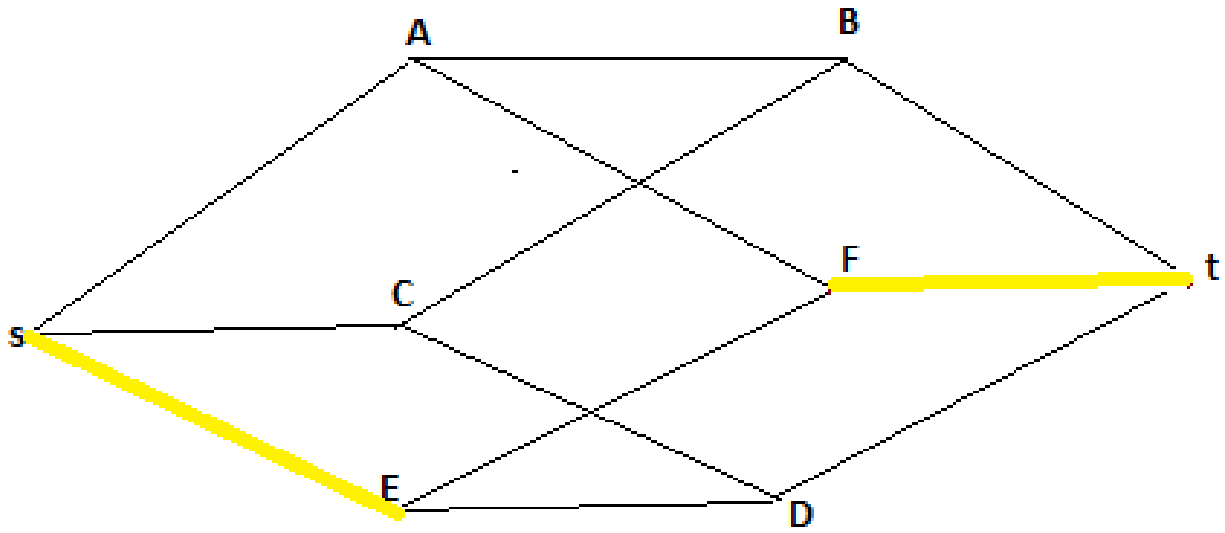
- Vimos en el ejemplo de no unicidad que efectivamente cualquiera de esos f obtenidos es maximal.
- Lo que Greedy NO PUEDE obtener es un flujo maximal del tipo:



- Pues la característica de Greedy (de ahi su nombre) es que una vez que detecta un camino dirigido no saturado entre s y t, manda todo el flujo que puede por ese camino.

Ejemplo de Greedy

- Ese ejemplo era muy trivial. Veamos otro:



- con capacidades todas 10, excepto por los lados $\overrightarrow{sE}$ y $\overrightarrow{Ft}$ de capacidad 20

Ejemplo de Greedy

- Para indicar que estamos aumentando f a lo largo de un camino dirigido no saturado $s, x_1, ..., x_{r-1}, t$ en ε unidades, usaremos la notación: $sx_1...x_{r-1}t : \varepsilon$.
- Greedy aumenta $f = 0$ por medio de los siguientes caminos, en sucesión:
  - sABt:10
  - sCDt:10
  - sEFt:10
- y luego no puede hacer mas, porque si busca un camino:
  - solo puede salir desde s a E
  - luego ir a D y ahi no puede seguir mas.

Ejemplo de Greedy

- Por lo tanto, Greedy obtiene un flujo de valor igual a 30.
- Observemos que:

$$\begin{aligned} c(\{s\}, V) &= \sum_x [x \in V][\overrightarrow{sx} \in E] c(\overrightarrow{sx}) \\ &= c(\overrightarrow{sA}) + c(\overrightarrow{sC}) + c(\overrightarrow{sE}) \\ &= 10 + 10 + 20 = 40 \end{aligned}$$

- y que:

$$\begin{aligned} c(V, \{t\}) &= \sum_x [x \in V][\overrightarrow{xt} \in E] c(\overrightarrow{xt}) \\ &= c(\overrightarrow{Bt}) + c(\overrightarrow{Dt}) + c(\overrightarrow{Ft}) \\ &= 10 + 10 + 20 = 40 \end{aligned}$$

Ejemplo de Greedy

- Por lo tanto $v(f) = 30 < 40 = c(\{s\}, V) = c(V, \{t\})$
- Asi que no podemos usar la propiedad que probamos para ver que f es maximal
- Pero recordemos que habiamos visto un ejemplo simple con $v(f) < c(\{s\}, V) = c(V, \{t\})$ pero con f maximal, asi que tampoco podemos concluir que f no es maximal.
- Asi que la pregunta es si f es maximal o no.

Ejemplo de Greedy

- Supongamos que en vez de elegir esos caminos, Greedy elija estos:
  - sAFt:10
  - sCBt:10
  - sEFt:10
  - sEDt:10
- terminando con un flujo de valor 40.
- Como el valor de este flujo es igual a $c(\{s\}, V)$, vemos que este flujo es maximal y el flujo que habiamos obtenido antes no.

Conclusiones sobre Greedy

- Por lo tanto vemos que al igual que el Greedy de coloreo, este Greedy no necesariamente va a encontrar un flujo maximal.
- En este caso eligiendo inteligentemente los caminos encontramos un flujo maximal.
- No siempre se puede anticipar cual seria la forma correcta de elegir caminos de entre todas las sucesiones posibles de caminos.
- Esto es similar con el problema del Greedy de coloreo, que no sabemos a priori en que orden elegir los vértices.
- Y sin embargo, al contrario del Greedy de coloreo, el Greedy de caminos puede ser modificado para encontrar un flujo maximal en tiempo polinomial (el algoritmo resultante no se llama Greedy)

Not Greedy

- El truco no está en tratar de “adivinar” la secuencia correcta.
- En el caso de flujos, se puede construir un algoritmo que corre Greedy y cuando llega a un cierto punto, “SE DA CUENTA” que se equivocó en la elección de los caminos
- y CORREGIR los errores.
- Esta corrección no se hace con un camino dirigido no saturado, sino con algo mas general.
- Para poder dar el algoritmo necesitaremos dos cosas:
  1. Algo que nos haga darnos cuenta que nos hemos equivocado.
  2. Una generalización del concepto de camino dirigido no saturado, para poder corregir los errores.
- Veamos primero lo primero

Definición de Corte

Definición:

Un Corte es un subconjunto de los vertices que tiene a s pero no tiene a t.

- Por ejemplo, $S = \{s\}$ es un corte.
- $S = V - \{t\}$ tambien es un corte.
- Otro ejemplo: si los vértices son $s, A, B, C, D, E, F, G, t$
- Entonces $S = \{s, A, C, D, E, G\}$ también es un corte.

Capacidad de un Corte

Definición:

La capacidad de un corte es $cap(S) = c(S, \overline{S})$, donde $\overline{S} = V - S$

Definición:

Un corte es MINIMAL si su capacidad es la menor de las capacidades de todos los cortes.

Es decir, S es un corte minimal si $cap(S) \leq cap(T)$ para todo corte T.

Sobre cortes y flujos

- Cortes minimales existen porque hay una cantidad finita de cortes: es fácil ver que hay 2^{n-2} cortes, donde n, como siempre, es la cantidad de vértices.
- ¿De que sirven los cortes?
- Veremos luego que para todo flujo f y para todo corte S, se tiene que $v(f) \leq cap(S)$.
- Asi que ya con eso tenemos una cota superior para el valor de cualquier flujo, en particular si S_μ es un corte minimal, nos da la cota superior mas baja posible.
- En particular sale de inmediato que si encontramos un flujo f con $v(f) = cap(S_\mu)$, entonces f es maximal.

Sobre cortes y flujos

- Entonces si pej usamos Greedy y terminamos con un flujo f con $v(f) = cap(S_\mu)$ sabremos que la elección de caminos fue correcta y no nos equivocamos: f es maximal.
- Pero ¿qué pasa si el f con el que terminamos tiene $v(f) < cap(S_\mu)$?
- Recordemos que en el caso de los cortes $S = \{s\}$ y $S = V - \{t\}$ podiamos tener $v(f) < cap(S)$ y aun asi ser f maximal en algunos ejemplos.
- Esos ejemplos funcionaban asi porque en esos ejemplos ni $\{s\}$ ni $V - \{t\}$ eran cortes minimales.
- Tambien demostraremos (pero la prueba es larga) que si f es maximal entonces necesariamente debe ser $v(f) = cap(S_\mu)$, donde S_μ es un corte minimal.

Sobre cortes y flujos

- De todos modos podrian decir que esto no sirve para mucho, pues dado que hay 2^{n-2} cortes, encontrar un corte minimal es exponencial.
- Pero hay una magia maravillosa en lo que veremos.
- Asociaremos a cada flujo f un corte correspondiente $S(f)$.
- Mientras el flujo no sea maximal, $v(f)$ será menor que $cap(S(f))$.
- y si $f = f_{max}$ es un flujo maximal, entonces $S(f_{max})$ será un corte minimal y $v(f_{max})$ será igual a $cap(S(f_{max}))$.

Sobre cortes y flujos

- Eso haremos en las proximas clases, pero luego tendremos un problema adicional: ver que efectivamente ALGUNA VEZ llegamos a un flujo maximal.
- Recuerden que dije que por el momento todavia ni siquiera sabemos que exista un flujo maximal.
- Si bien como dije esto se puede demostrar puramente con herramientas de Análisis Matemático, lo que haremos es que una vez que demostremos que nuestro algoritmo siempre termina con un flujo maximal, entonces obviamente habremos demostrado la existencia de un flujo maximal.
- Pero la primera parte de lo que sigue estará concentrada en demostrar que si(if) existe un flujo maximal, entonces su valor es igual a la capacidad de un corte minimal.