Flujo máximo y corte mínimo

Agustín Santiago Gutiérrez

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Campamento Caribeño ACM-ICPC 2016

Contenidos

“Todo fluye, nada permanece.”

Heráclito

“Si sus fuerzas están unidas, sepáralas.”

Sun Tzu, El Arte de la Guerra.

Contenidos

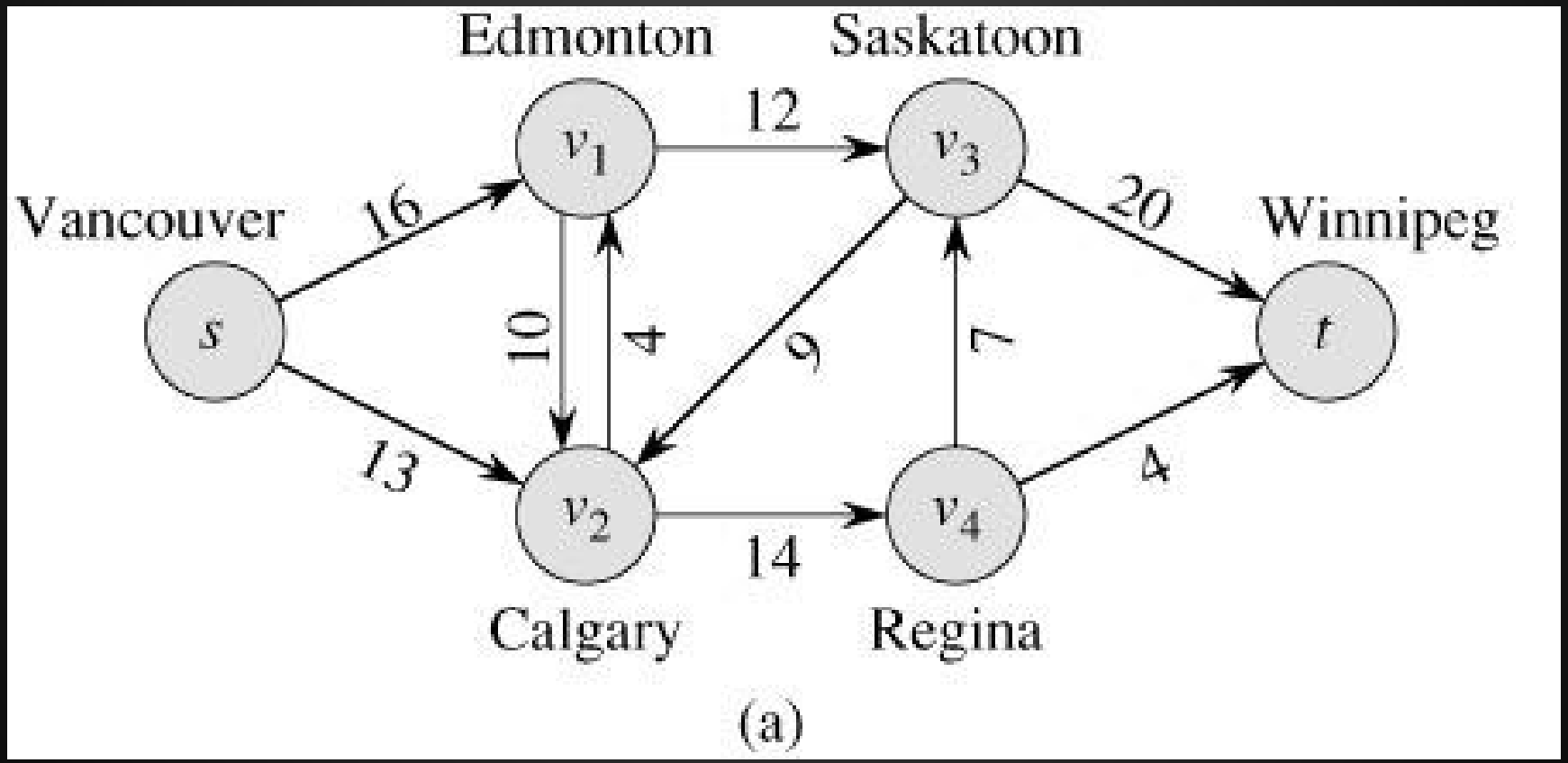
Flujo Máximo

Training Camp Argentina 2014

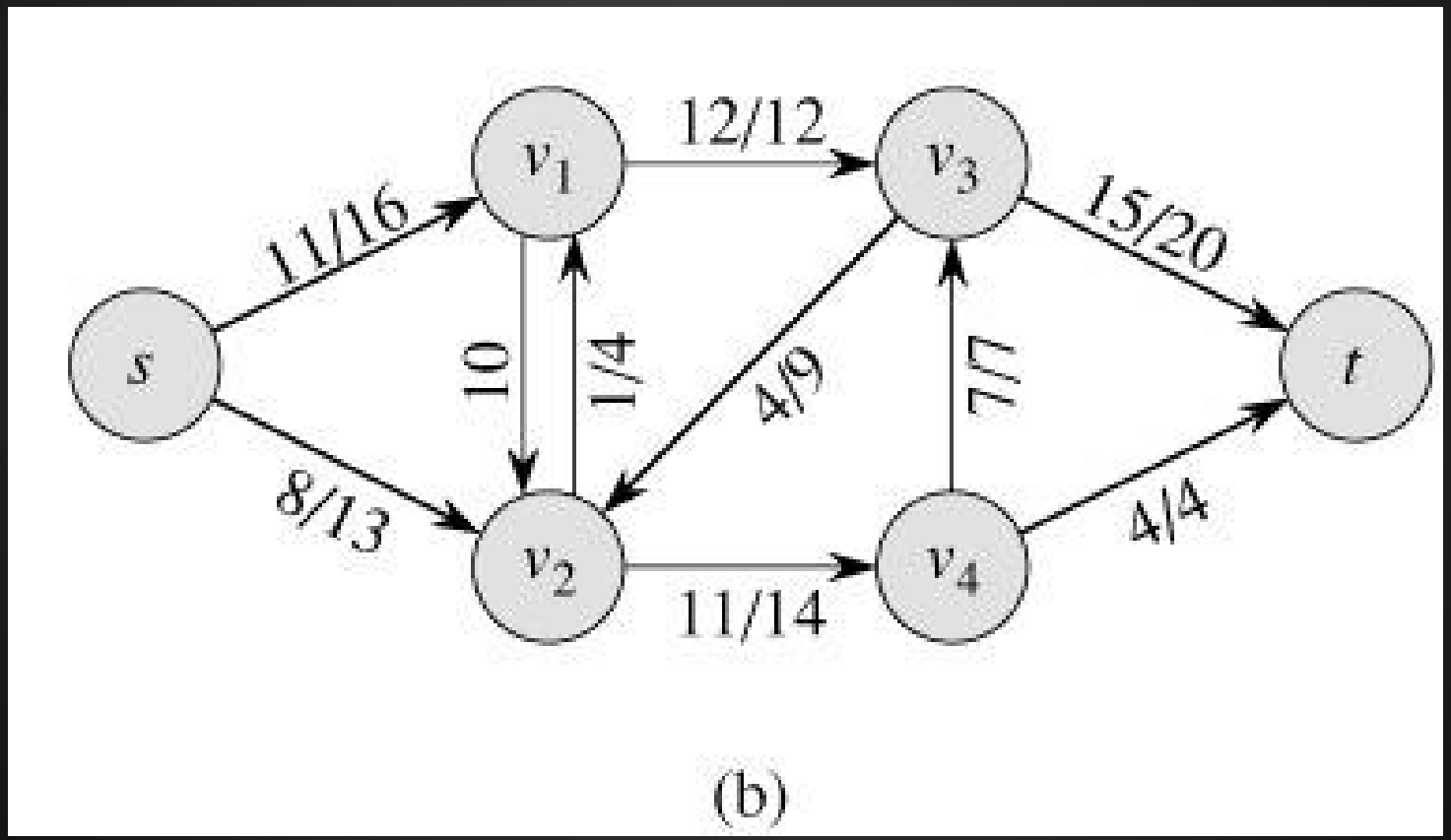
Nicolás Álvarez
naa [at] cs uns edu ar

(Con algunos ~~cambios que arruinan todo~~ pequeños retoques cosméticos)

Idea


Edmonton
Saskatoon
v_1
v_3
12
Vancouver
16
20
Winnipeg
s
t
10
4
9
7
13
v_2
v_4
4
14
Calgary
Regina
(a)

Idea



Idea

- Un *source* s
- Un *sink* t
- Nodos intermedios
- Arcos dirigidos con una capacidad dada
- Objetivo: Enviar el máximo flujo posible de s a t, cumpliendo las restricciones de capacidad.

Definición: Red de Flujo

- Una red de flujo es un grafo dirigido G = <N,A> donde cada arco e $\in$ A tiene asociado una capacidad c(e) >= 0.

- Se distinguen dos nodos s y t llamados fuente y destino

Definición: Flujo

Un flujo en G es una función f : A -> R que satisface:

- *restricción de capacidad*:
  - f(e) <= c(e) para todo e $\in$ A
- *conservación de flujo*:
  - Para todo u $\in$ N - {s,t}

$$\sum_{\substack{e \in A \\ e \text{ sale de } u}} f(e) = \sum_{\substack{e \in A \\ e \text{ llega a } u}} f(e)$$

Definición: Problema Flujo Máximo

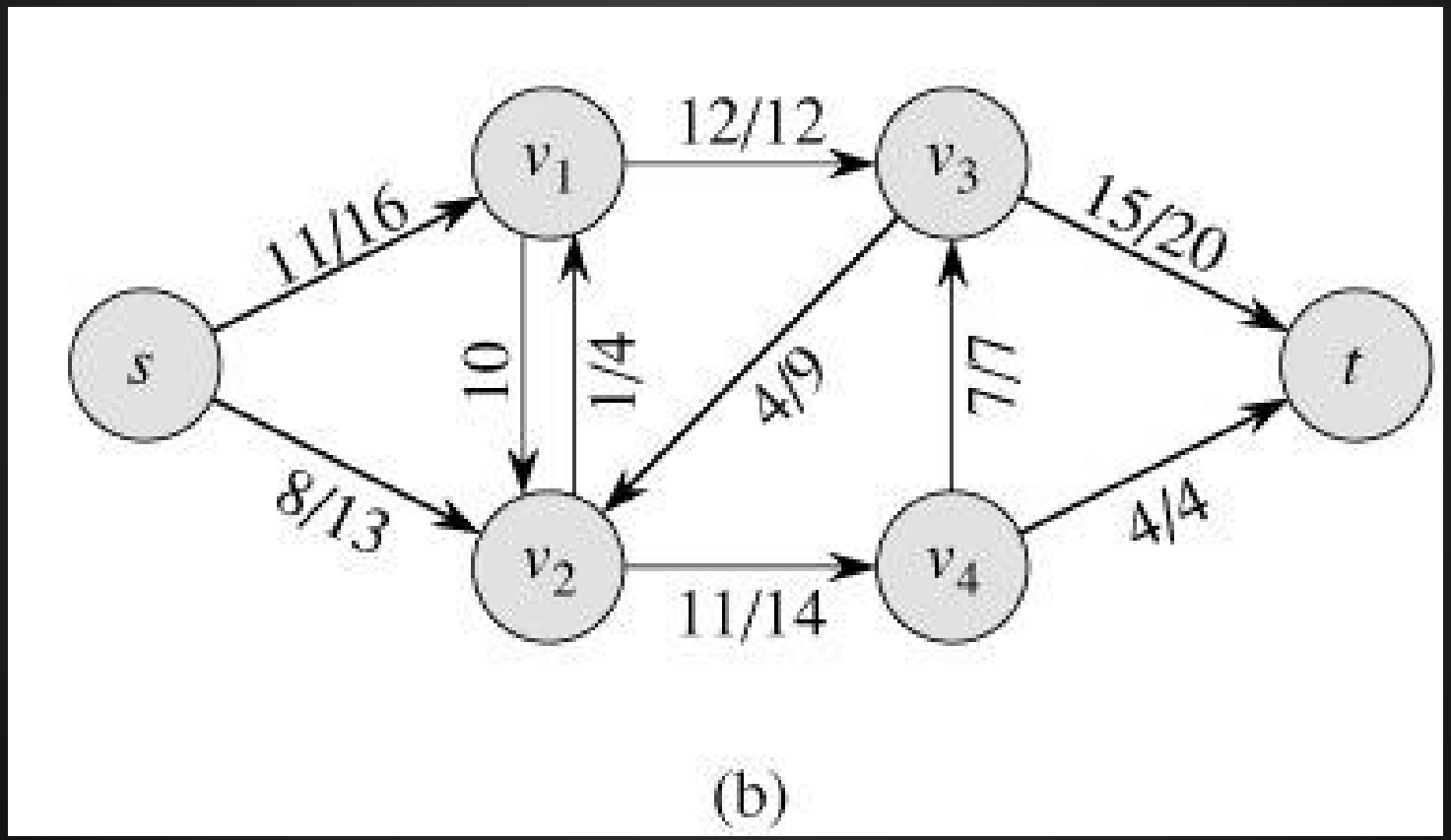
- El valor de un flujo | f | se define como

$$|f| = \sum_{\substack{e \in A \\ e \text{ sale de } s}} f(e) - \sum_{\substack{e \in A \\ e \text{ llega a } s}} f(e)$$

- Dada una red de flujo G, el Problema de Flujo Máximo consiste en encontrar el flujo de máximo valor que admite G

Repaso

¿Es un flujo máximo?



Método de Ford-Fulkerson

Es un método general, no un algoritmo específico.

Se basa en 3 conceptos claves:

- redes residuales
- caminos de aumento
- cortes

Método de Ford-Fulkerson

El método comienza con una red de flujo vacía.

En cada iteración se busca un camino de aumento en la red residual para incrementar el valor del flujo.

Se puede demostrar mediante el teorema de max-flow min-cut que, cuando termina, el flujo obtenido es máximo

Redes residuales

Intuitivamente, la red residual es una nueva red de flujo que representa cómo podemos cambiar el flujo.

Formalmente, la red residual de G = <N,A> es G_f = <N,A_f>, donde por cada arco e : u --> v en A, se tiene en A_f :

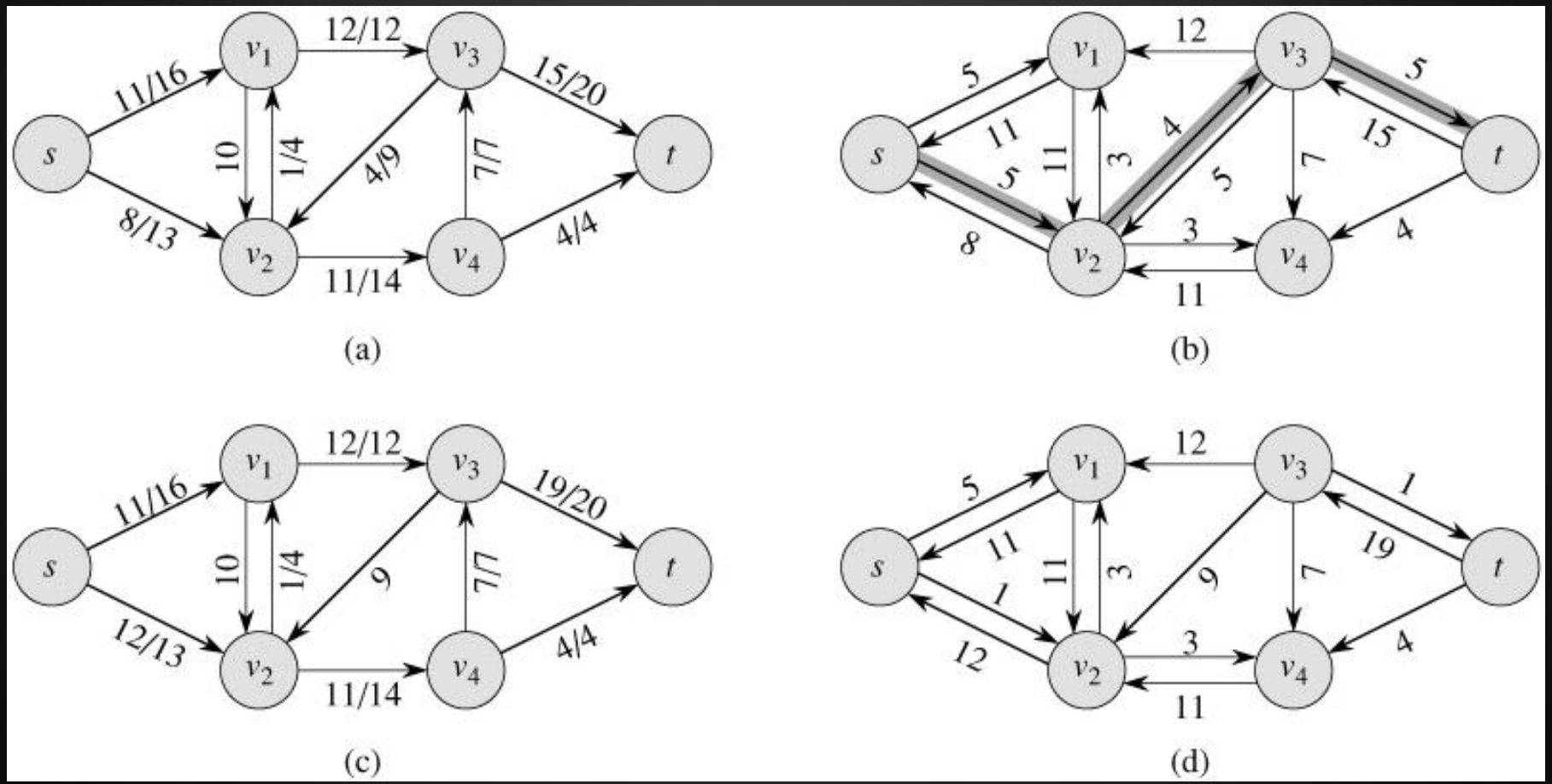
- Un arco de u a v con capacidad c(e) - f(e)
- Un arco de v a u con capacidad f(e)

Caminos de aumento

Como vimos, dada una red de flujo G = <N,A> y un flujo f, podemos obtener una red residual G_f
Un camino de aumento es, sencillamente, un camino simple de s a t en la red residual.

Si existe un camino de aumento en la red residual, el flujo puede aumentarse a lo largo de dicho camino con un valor igual a la menor capacidad de los arcos del camino.

Aumento de flujo



Cortes

Dada una red de flujo G = <N,A>. Un corte es una partición de N en S y T = N-S tal que $s \in S$ y $t \in T$.

Dado un corte (S,T).

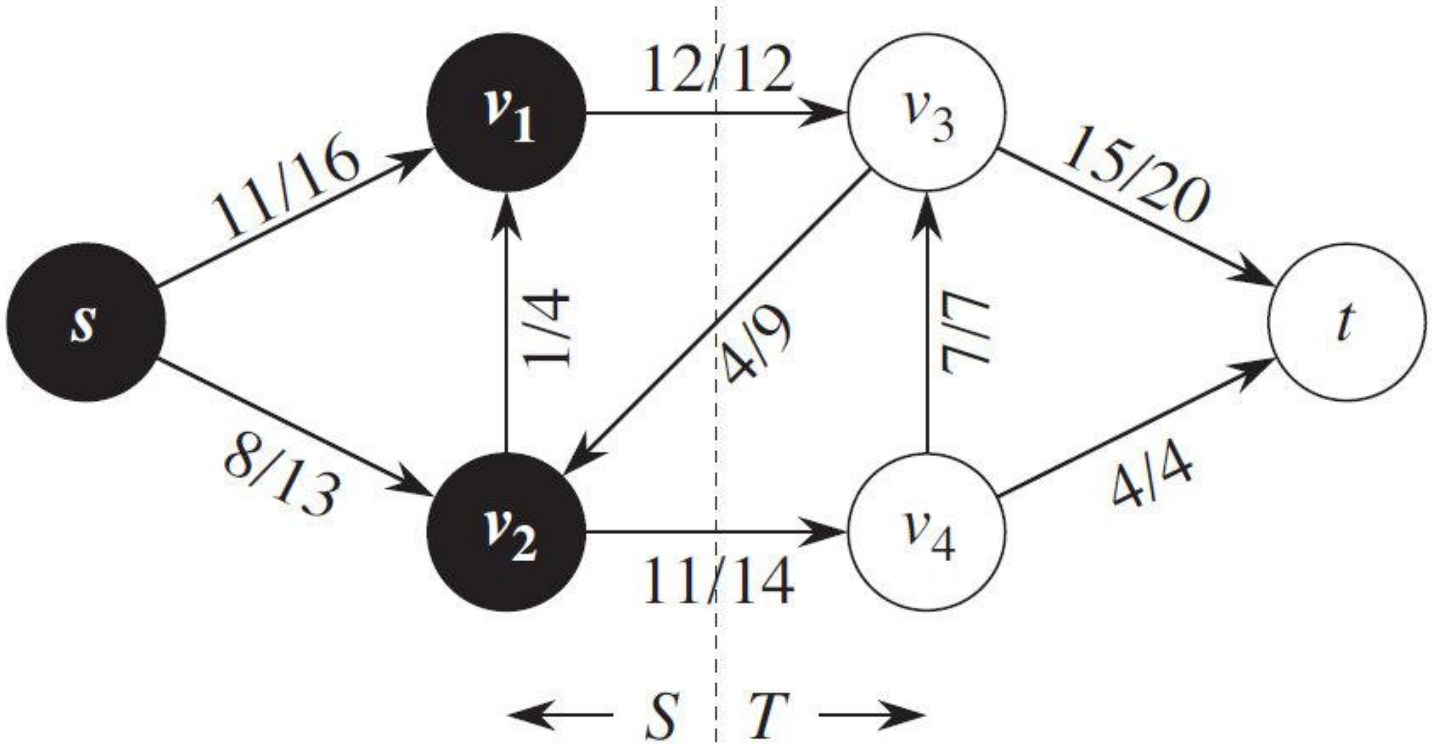
El flujo neto a través del corte es:

$$f(S,T) = \sum_{\substack{e \in A \\ e: u \to v \\ u \in S, v \in T}} f(e) - \sum_{\substack{e \in A \\ e: u \to v \\ u \in T, v \in S}} f(e)$$

Y la capacidad del corte es:

$$c(S,T) = \sum_{\substack{e \in A \\ e: u \to v \\ u \in S, v \in T}} c(e)$$

Cortes



Teorema max-flow min-cut

Los 3 siguientes condiciones son equivalentes:

1. f es un flujo máximo
2. La red residual G_f no contiene caminos de aumentos
3. $| f | = c(S,T)$ para algún corte (S,T) de G

Este teorema prueba que el método de Ford-Fulkerson es correcto.

Max-flow min-cut: Demostración

1 => 2

Si el flujo es máximo entonces no pueden existir caminos de aumento. De otro modo, sería posible aumentar el valor del flujo

2 => 3

Si no existen caminos de aumento en la red residual, s y t están desconectados. Por lo tanto si S es el conjunto de los nodos alcanzables desde s y T = N - S. El corte (S,T) está saturado

Max-flow min-cut: Demostración

3 => 1

Se puede observar que todo flujo tiene un valor menor o igual que la capacidad de cualquier corte. Es decir, todo corte implica una cota superior al valor de un flujo válido.

Si el flujo satura algún corte (S,T), esto quiere decir que el flujo tiene el máximo valor posible. Además, el corte (S,T) es el corte de capacidad mínima (o *min-cut*) de la red de flujo.

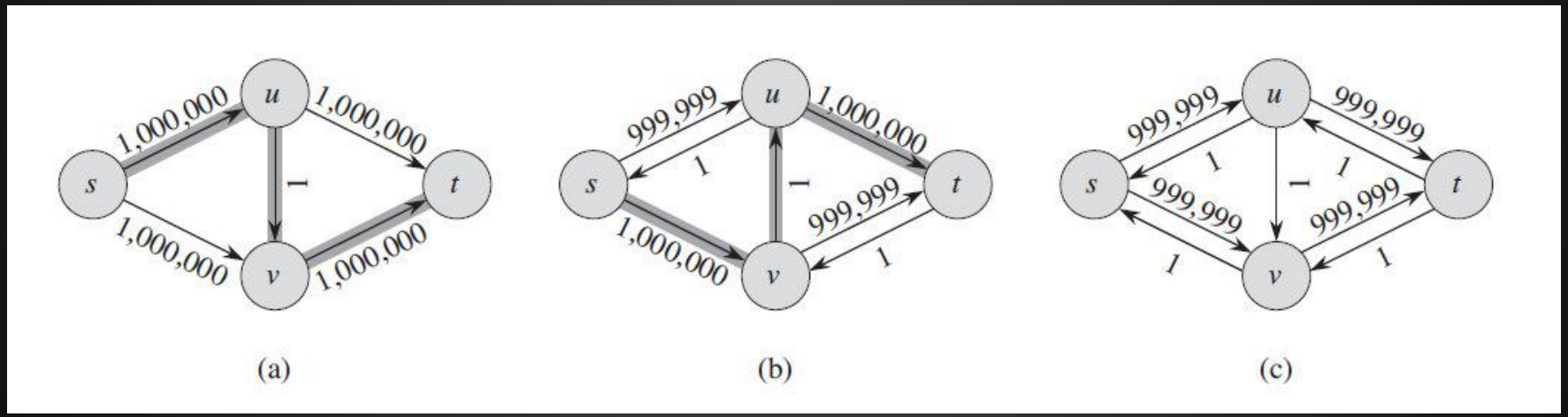
Ford-Fulkerson: Pseudo-código

```
FORD-FULKERSON(G, s, t):
   para cada arco (u,v) ∈ G.E:
      (u,v).f = 0

   mientras exista camino de aumento p en G_f:
      c_f(p) = min {c_f(u,v) : (u,v) ∈ P}
      para cada arco (u,v) ∈ p:
         (u,v).f += c_f(p)
         (v,u).f -= c_f(p)
```

Algoritmo de Edmonds-Karp

Motivación:



Algoritmo de Edmonds-Karp

Como se vió en la figura anterior, si no elegimos los caminos de aumento de manera inteligente el tiempo de ejecución puede no estar acotado polinomialmente sobre el tamaño de la entrada.

Afortunadamente, si buscamos el camino de aumento más corto con un BFS, se puede demostrar una cota polinomial: $O(m^2n)$

Esta implementación de Ford-Fulkerson recibe el nombre de algoritmo de Edmonds-Karp

Problema de tarea

Necklace - Regional China 2008 (Hefei)
http://goo.gl/xIFUBV

Dado un grafo no dirigido ($n <= 10^4$, $m <= 10^5$) y dos nodos S y T. Determinar si existe un collar (necklace) entre S y T.

Generalización de Conectividad

- Conocemos el concepto de conectividad en un grafo.
- Se puede generalizar a k-conectividad

Generalización de Conectividad

- Un grafo es k-vertex connected si no existe ningún conjunto de k-1 vértices tal que al eliminarlos nos deja el grafo desconectado
- A partir de lo anterior, podemos definir componentes k-vertex connected de un grafo como los subconjuntos maximales que cumplen la propiedad
- Existe una definición similar para k-edge connectivity

Como calcular k-connectivity?

- A alguien se le ocurre una idea?
- Con el teorema de Max Flow - Min Cut podemos obtener la k-edge connectivity
- Y cuando queremos calcular k-vertex. QUE HACEMOS???

Problema de tarea

Optimal Marks - SPOJ
goo.gl/X4fX5Q

Dado un grafo no dirigido donde algunos nodos tienen asignado un número de 31 bits. La tarea es asignar números al resto de los nodos de manera que la suma de los xor de nodos adyacentes sea mínima. O sea, para todo (u,v) $\in$ G, minimizar $\sum_{(u,v)}$ marca(u) ^ marca(v)

Bipartite Matching

Existen problemas combinatorios que, en principio, parecen no guardar relación con el problema de Flujo Máximo pero que pueden ser reducidos a éste.

Uno de los ejemplos más conocidos es el problema del matching bipartito.

Bipartite Matching

Dado un grafo G = <N,A>, un *matching* es un subconjunto $M \subseteq A$ tal que para todo vértice $v \in M$ existe a lo sumo un arco de M incidente a v.

Se puede pensar como un "apareamiento" de los nodos, en los que cada nodo se aparea con a lo sumo un nodo, y la relación es simétrica.

Bipartite Matching

El problema de *Maximum Matching* consiste en encontrar un *matching* válido de máxima cardinalidad en un grafo.

Este problema resulta más sencillo para una clase restringida de grafos llamados grafos bipartitos.

Grafo bipartito

Un grafo bipartito es un grafo G = <N,A> donde el conjunto de nodos N puede ser particionado en 2 conjuntos N = L $\cup$ R donde L y R son disjuntos y los arcos unen nodos entre L y R.

El problema de hallar un matching máximo en un grafo bipartito se conoce como *maximum bipartite matching*.

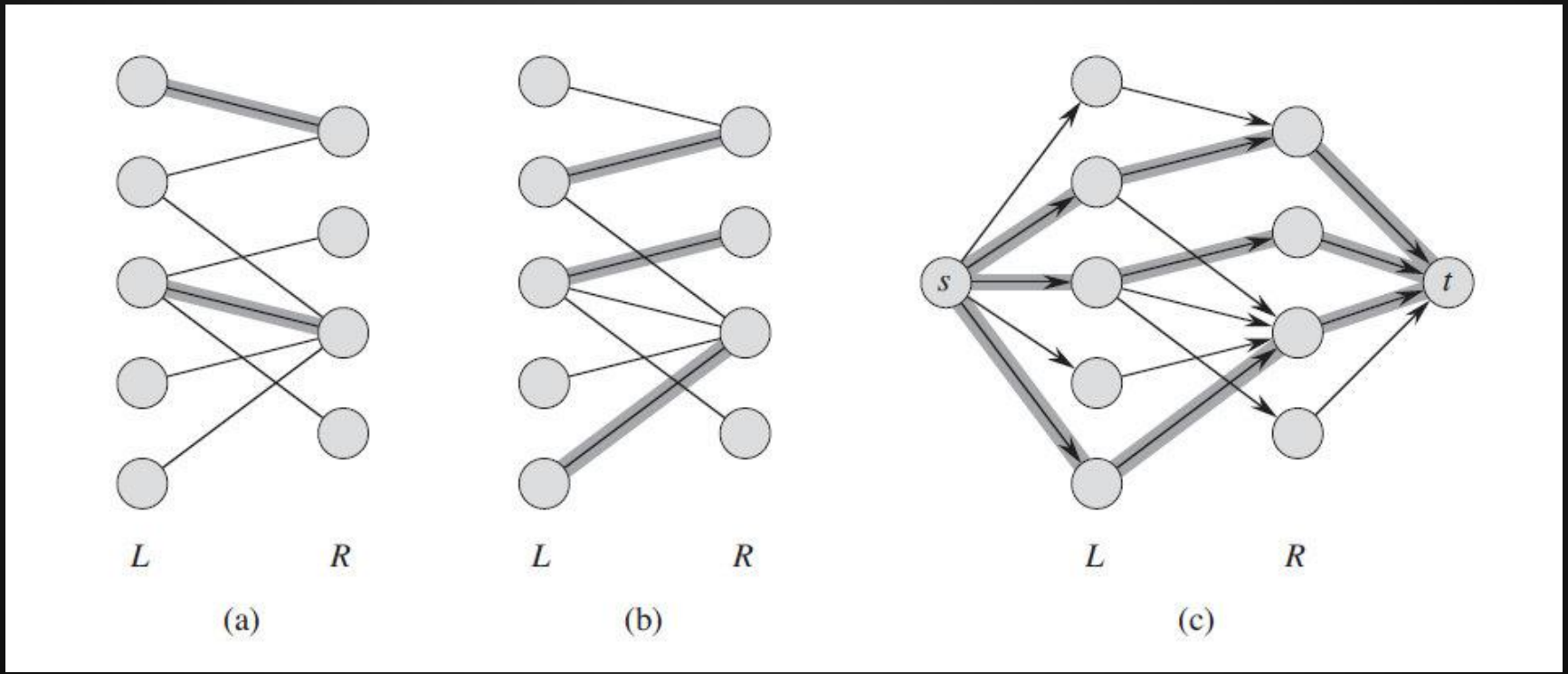
Maximum bipartite matching

Este problema puede reducirse a un problema de flujo máximo.

Para ello, construimos una red de flujo donde se agregan 2 nodos: un source s y un sink t.

Se agrega un arco de capacidad 1 entre el *source* y cada nodo de L y se agrega un arco de capacidad 1 entre cada nodo de R y el *sink*.

Maximum bipartite matching



Problema de tarea

Attacking Rooks - Regional Latam 2013
http://goo.gl/oph4Q8

Dado un tablero de ajedrez de m x n (1 <= m,n <= 100) donde algunas casillas fueron bloqueadas. Determinar cuál es la máxima cantidad de torres que se pueden colocar de manera que no existan 2 que se amenacen.

Más problemas!

- Inspection: goo.gl/iYho0F
- Super Watch: goo.gl/Y1h5zY
- Oil Company: goo.gl/XW3jSp
- Shogui Tournament: goo.gl/KRz3Yi
- Graduation: goo.gl/hnCwOU
- Parking: goo.gl/21ZJqC
- "Shortest" pair of paths: goo.gl/ardYaY
- Angels and Devils: goo.gl/sCkd30

Bibliografía

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. *Introduction to Algorithms 3ed. Capítulo 26*

- Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*

- Topcoder Tutorials:
  - Max flow: goo.gl/luDODH
  - Min-cost max-flow: goo.gl/XfRWjS

Preguntas?

Contenidos

Idea

- La idea de Dinitz es muy similar a la del algoritmo de Edmonds y Karp, pero mejor.
- Los caminos de aumento a lo largo de la ejecución del algoritmo de Edmonds y Karp son cada vez más largos, lo cual permite acotar la cantidad de veces que se satura cada eje a $O(N)$ veces.

Idea

- La idea de Dinitz es muy similar a la del algoritmo de Edmonds y Karp, pero mejor.
- Los caminos de aumento a lo largo de la ejecución del algoritmo de Edmonds y Karp son cada vez más largos, lo cual permite acotar la cantidad de veces que se satura cada eje a $O(N)$ veces.
- Pero un BFS recorre **todo** el grafo...
- Y a partir de sus valores de distancia, nos resume en un DAG **todos** los caminos mínimos desde/hacia el nodo inicial...

Idea

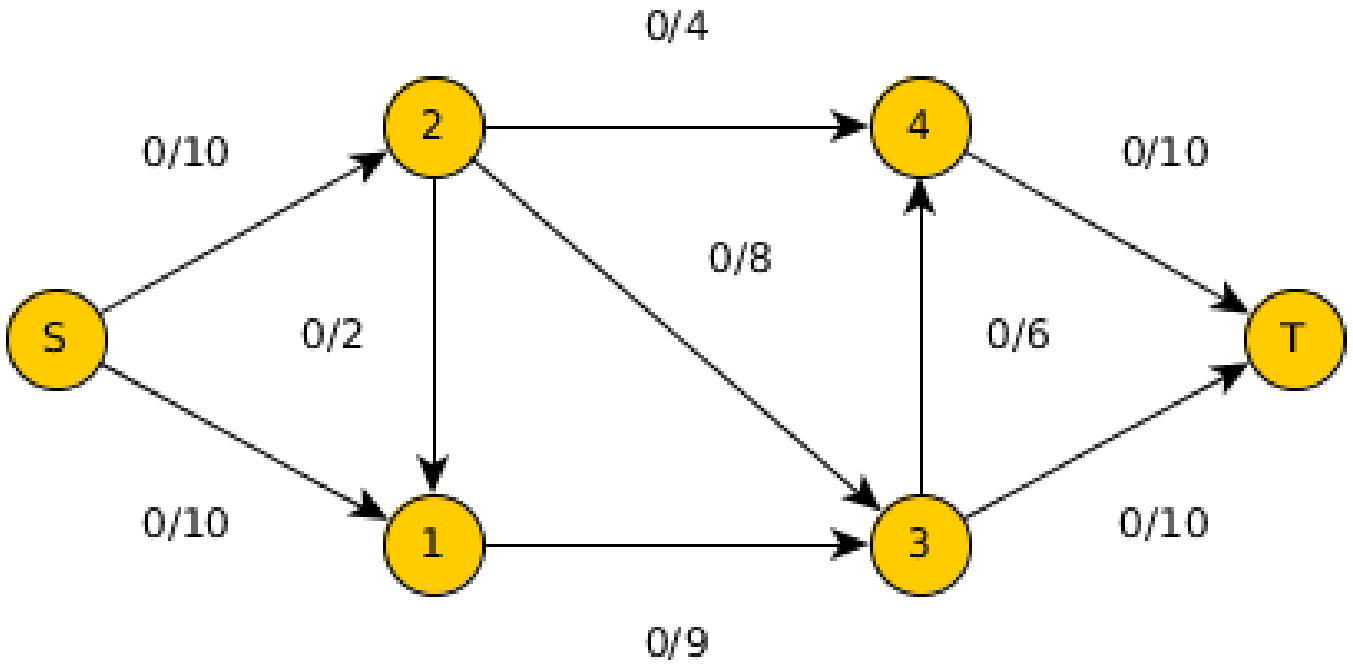
- La idea de Dinitz es muy similar a la del algoritmo de Edmonds y Karp, pero mejor.
- Los caminos de aumento a lo largo de la ejecución del algoritmo de Edmonds y Karp son cada vez más largos, lo cual permite acotar la cantidad de veces que se satura cada eje a $O(N)$ veces.
- Pero un BFS recorre **todo** el grafo...
- Y a partir de sus valores de distancia, nos resume en un DAG **todos** los caminos mínimos desde/hacia el nodo inicial...
- ¿Entonces, por qué usamos **solamente un** camino de aumento mínimo?
- ¡Usemos **todos** los caminos de aumento mínimos en cada iteración!

Idea (cont)

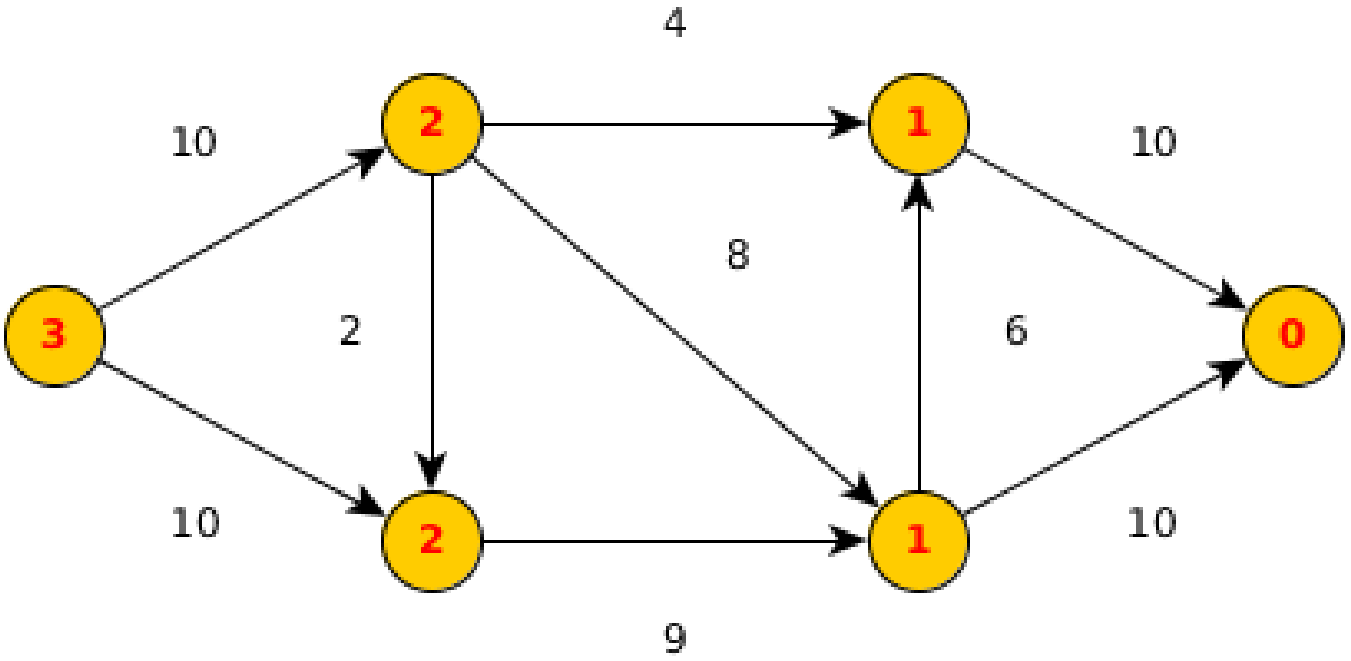
- En cada iteración, Dinitz calcula con BFS el **rank** de cada nodo: la longitud de un camino mínimo en la red residual desde ese nodo hasta t.
- Notar que t será el nodo inicial, y las aristas se miran al revés.
- Si solo miramos las aristas de la red residual que van de un nodo con rank r a uno con rank $r-1$, queda un DAG, que contiene **todos** los caminos de aumento mínimos.
- La idea será extraer ahora caminos de aumento igual que Edmonds Karp, pero varios en esta misma pasada para que no quede ninguno de esta longitud mínima.
- De esta forma habrá en total $O(V)$ pasadas.

Ejecución

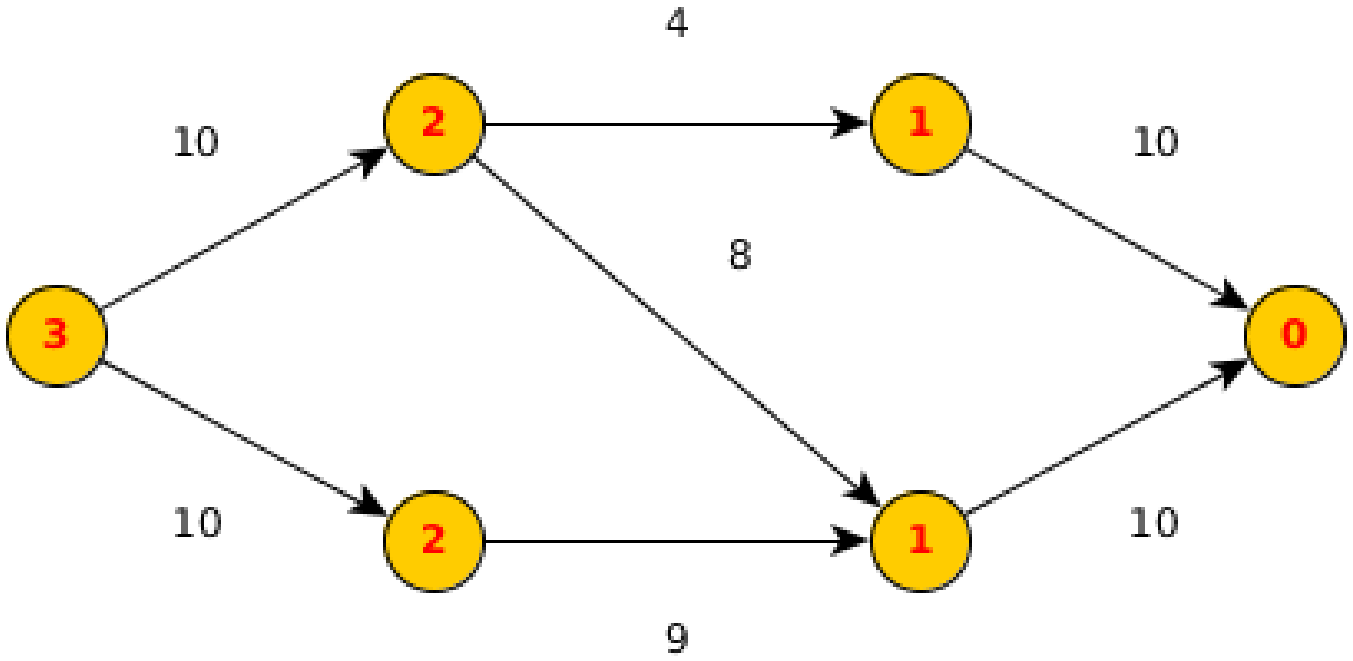
Grafo inicial:



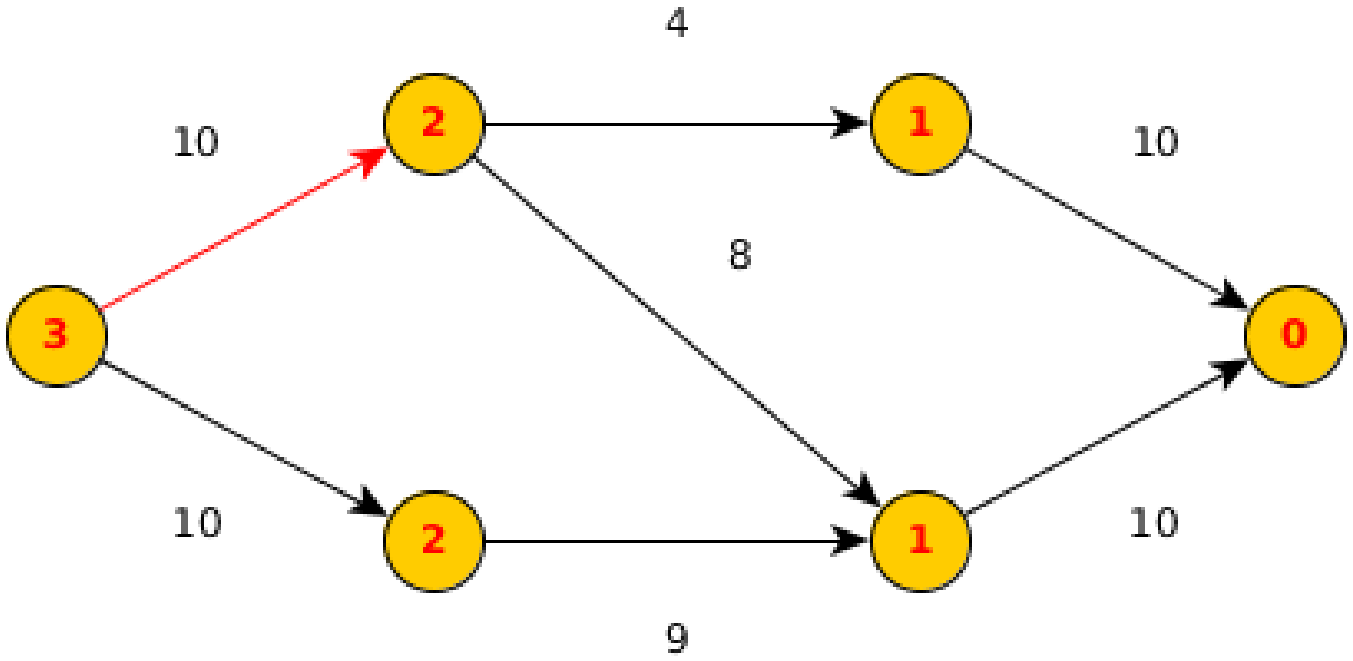
Ejecución



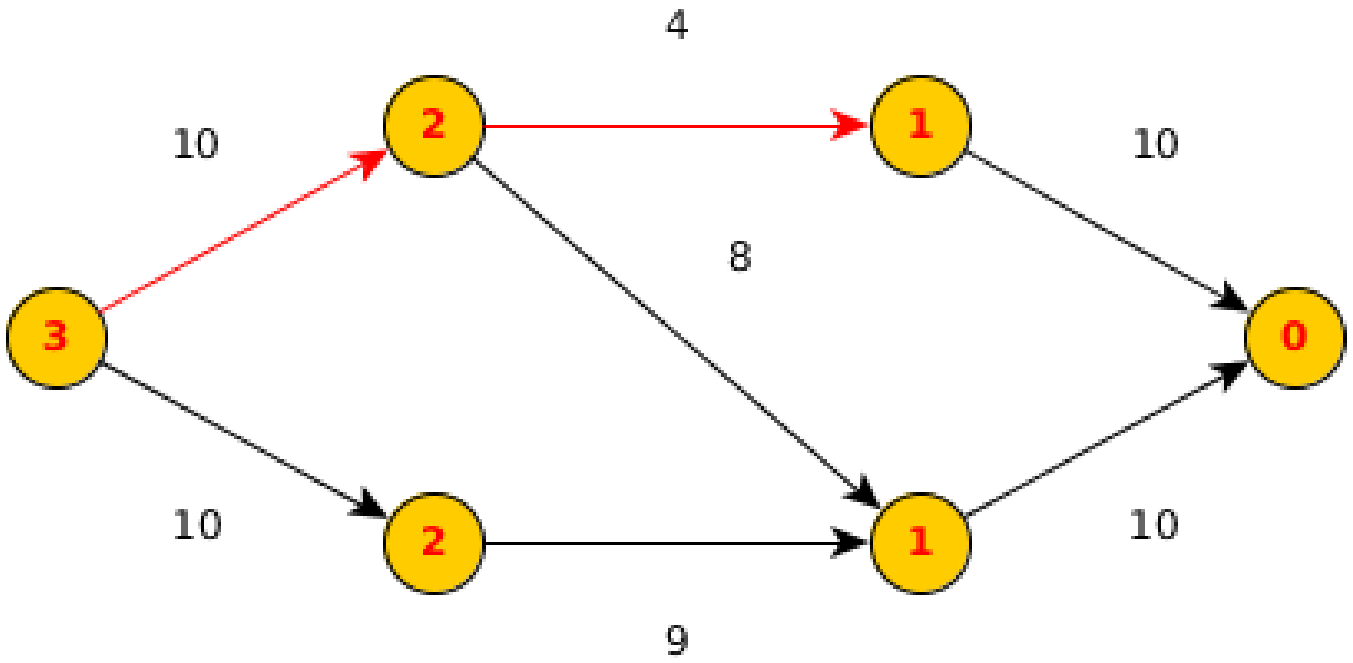
Ejecución



Ejecución

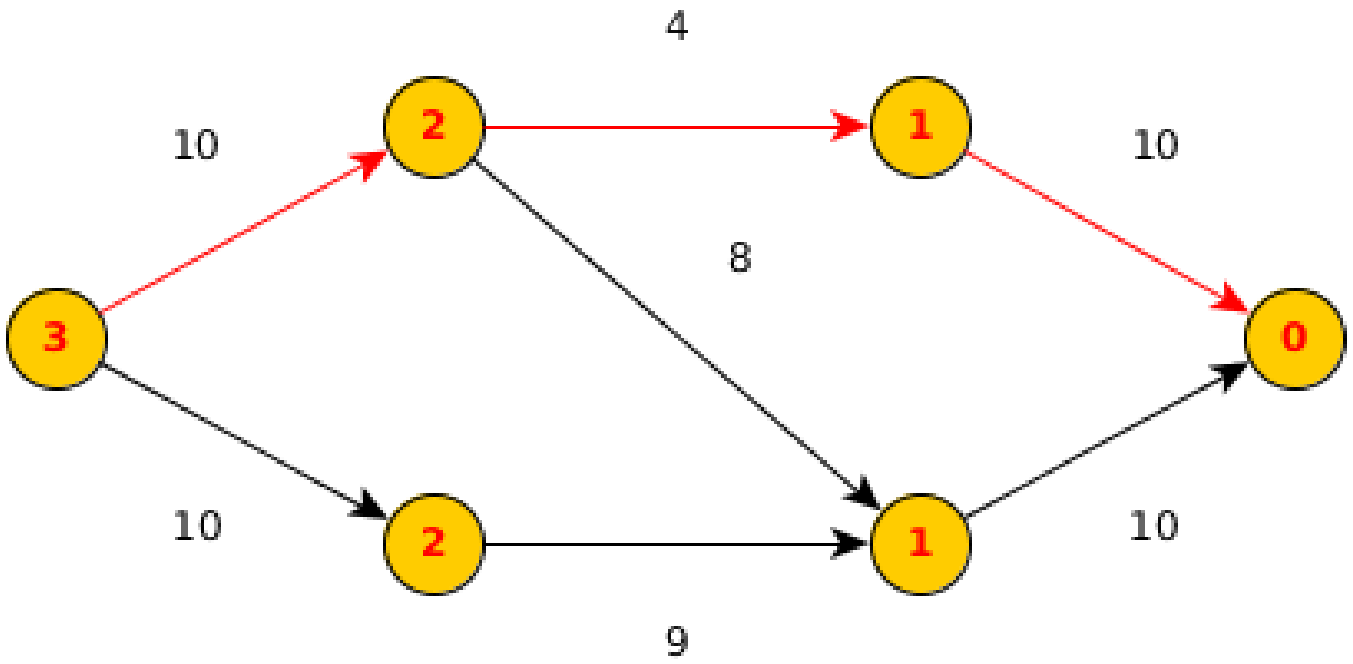


Ejecución

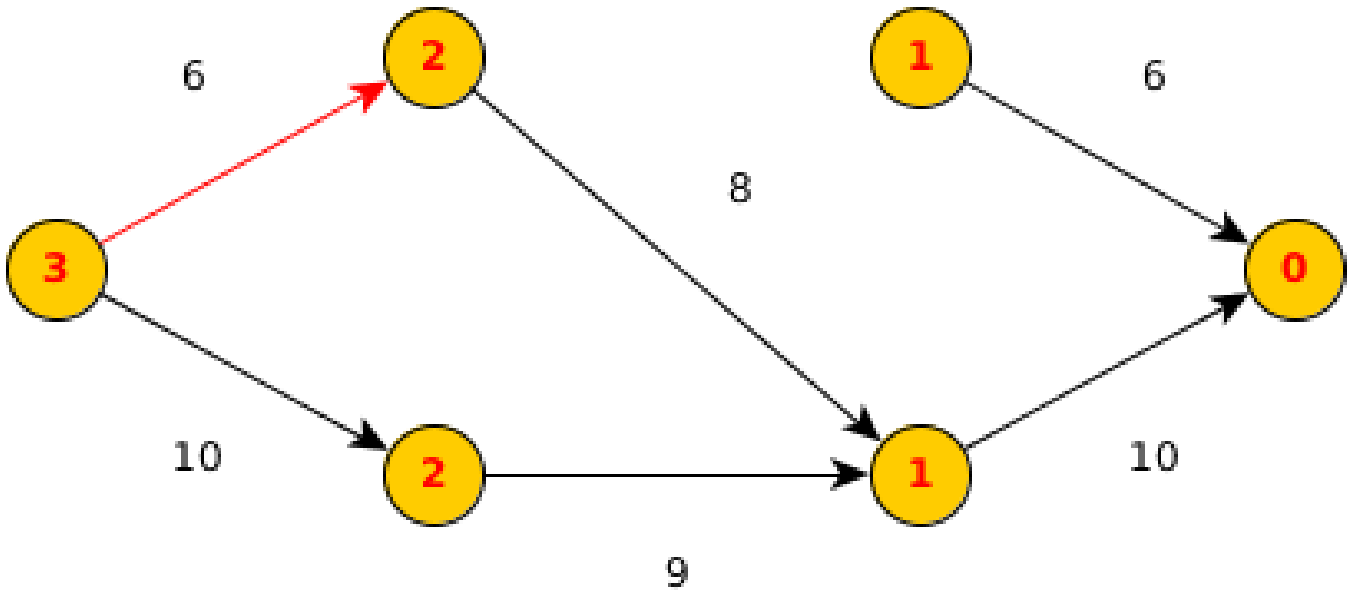


Ejecución

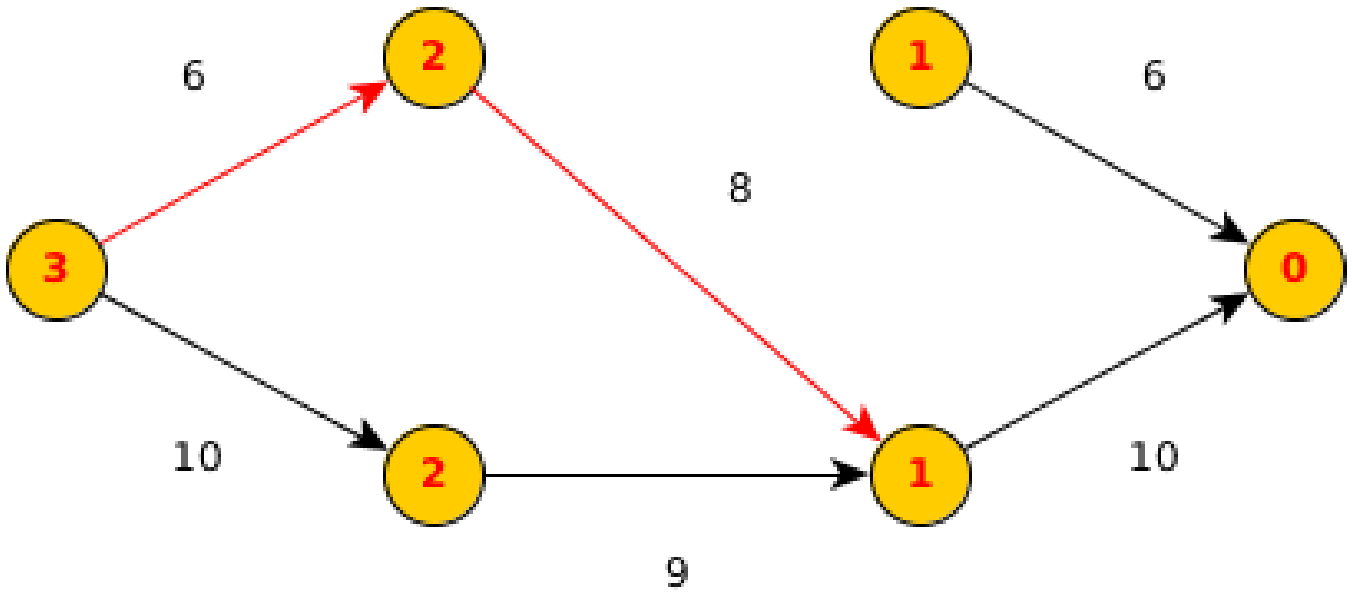
Recorremos el camino encontrado, pasando 4 unidades de flujo por cada arista.



Ejecución

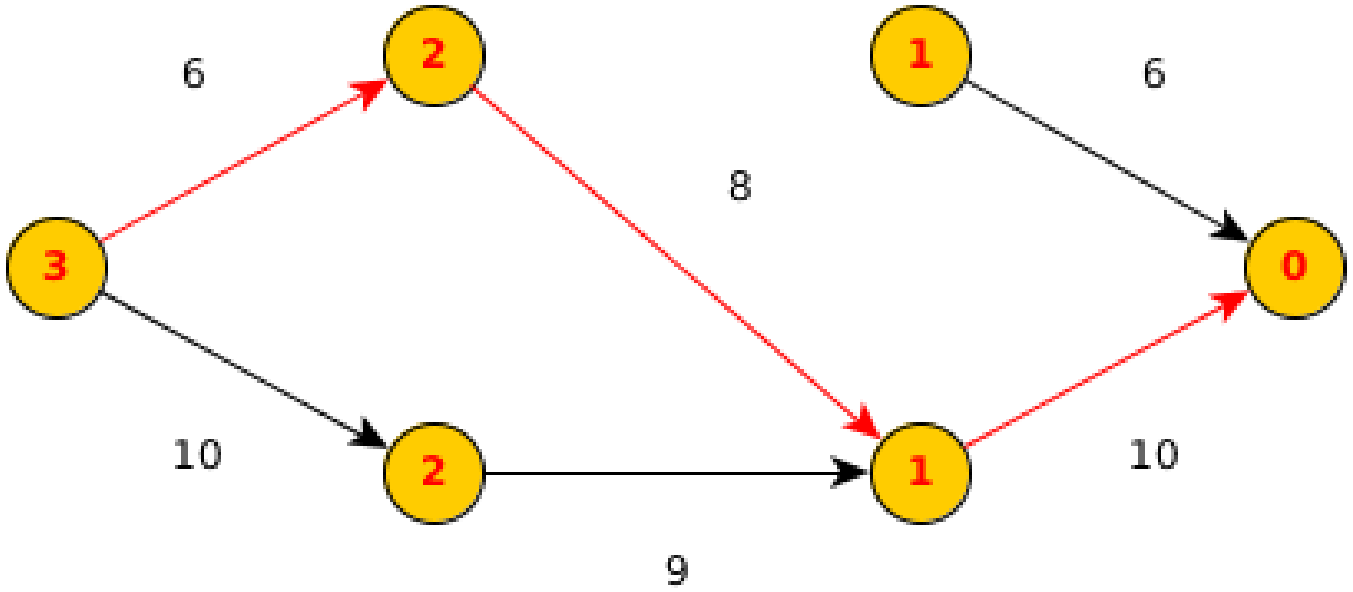


Ejecución

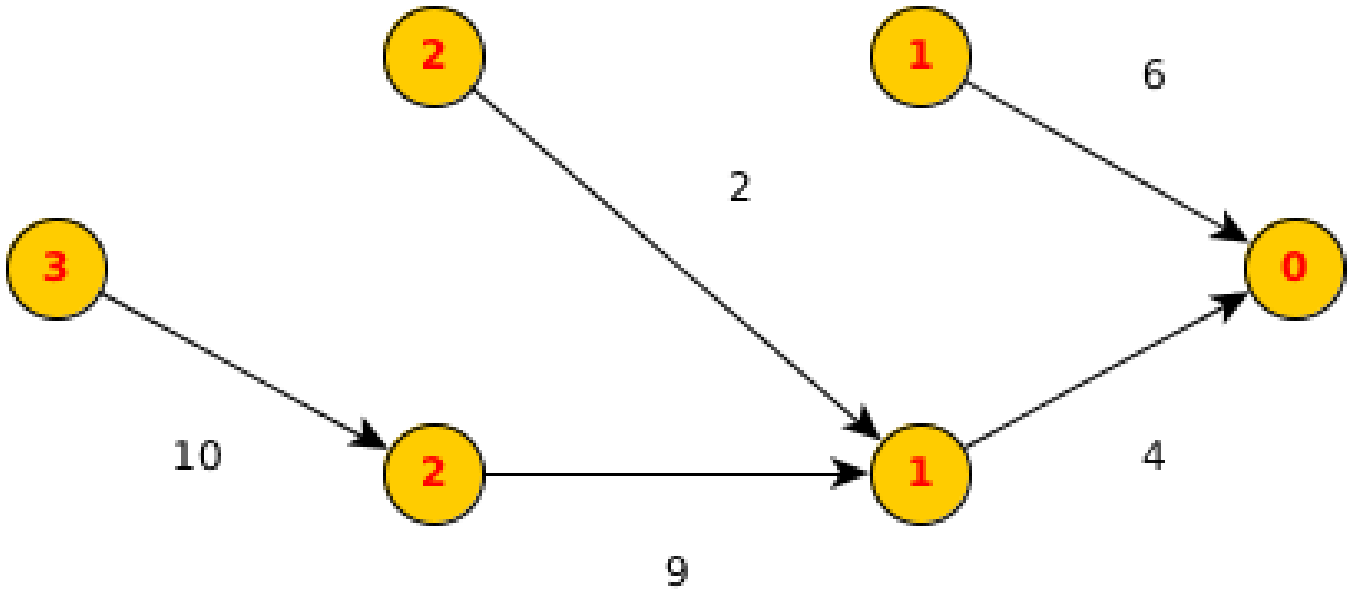


Ejecución

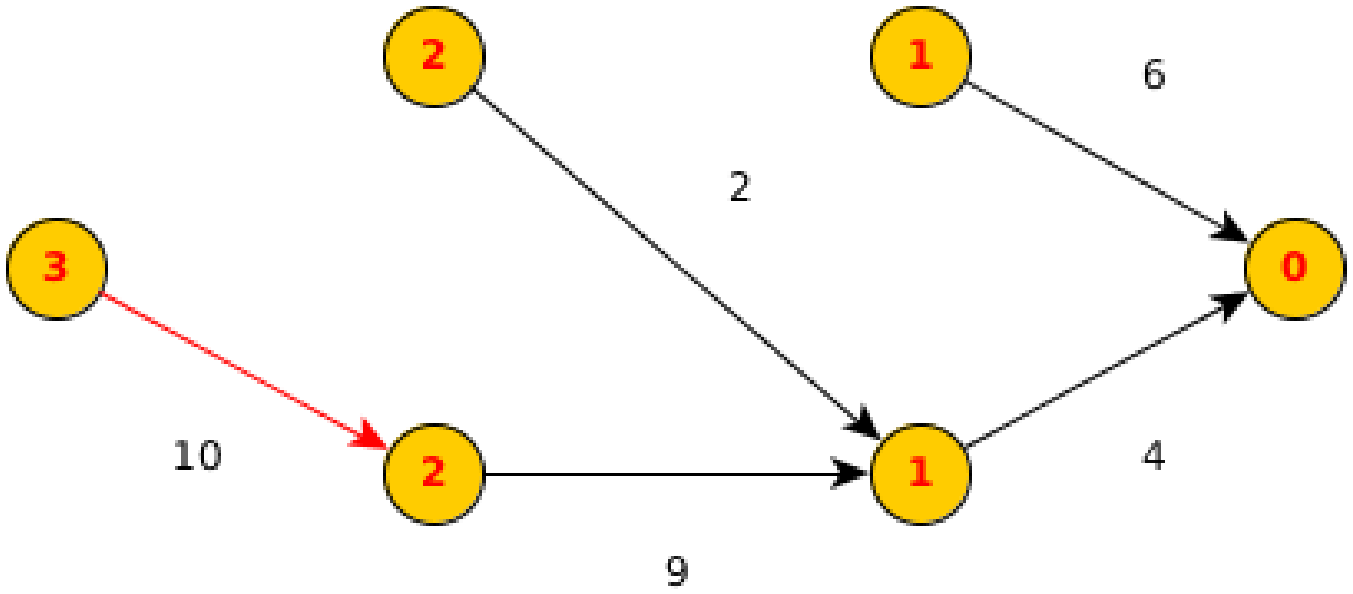
Recorremos el camino encontrado, pasando 6 unidades de flujo por cada arista.



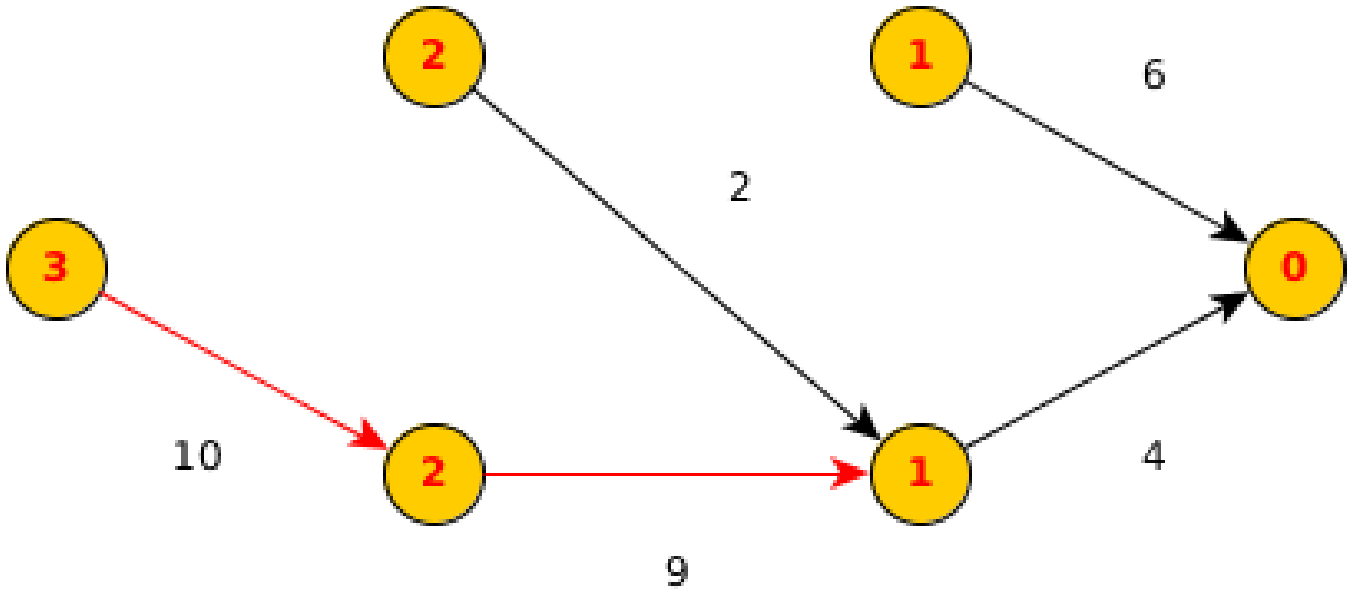
Ejecución



Ejecución

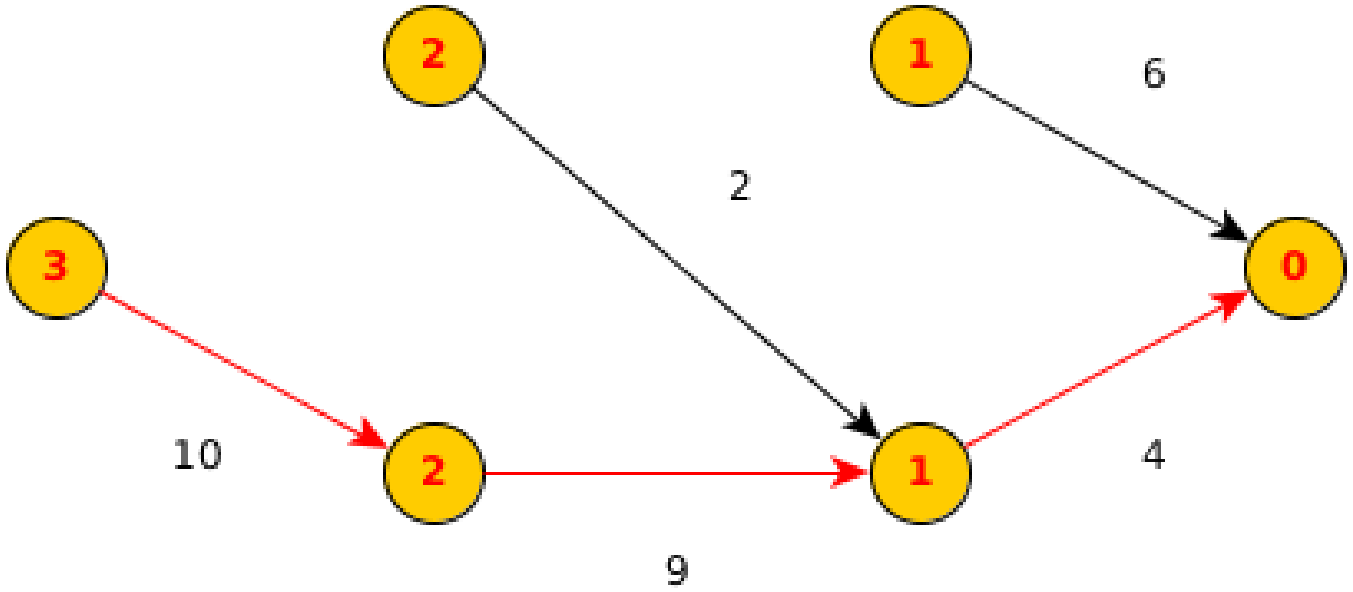


Ejecución

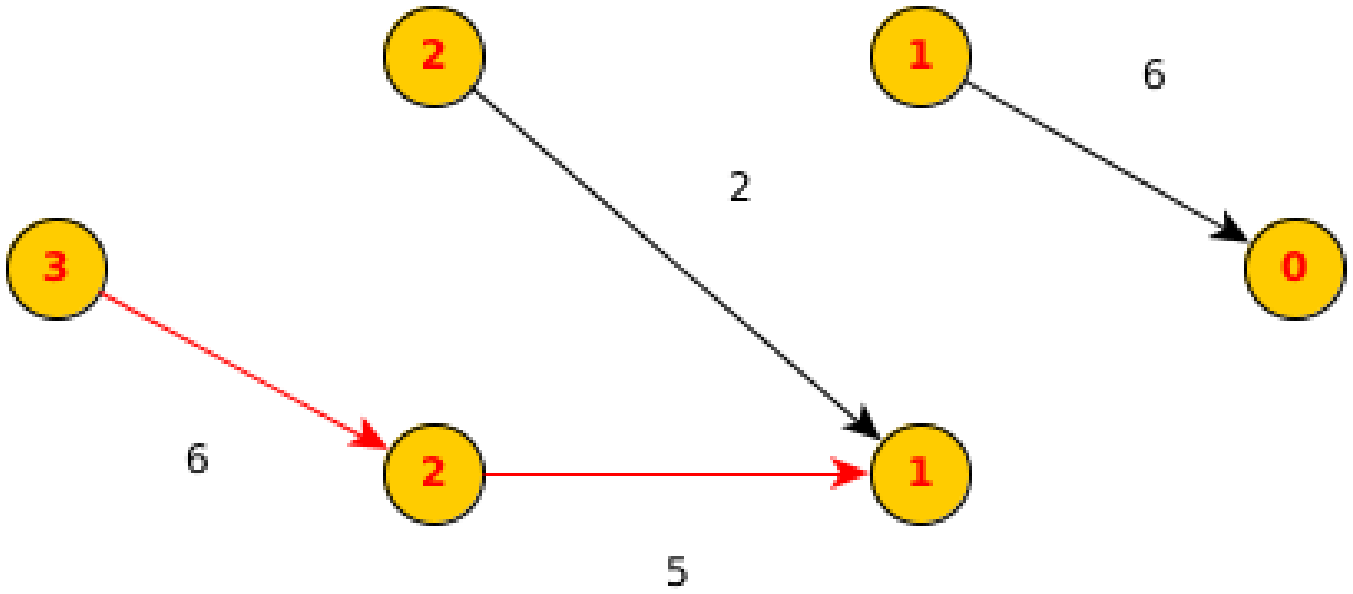


Ejecución

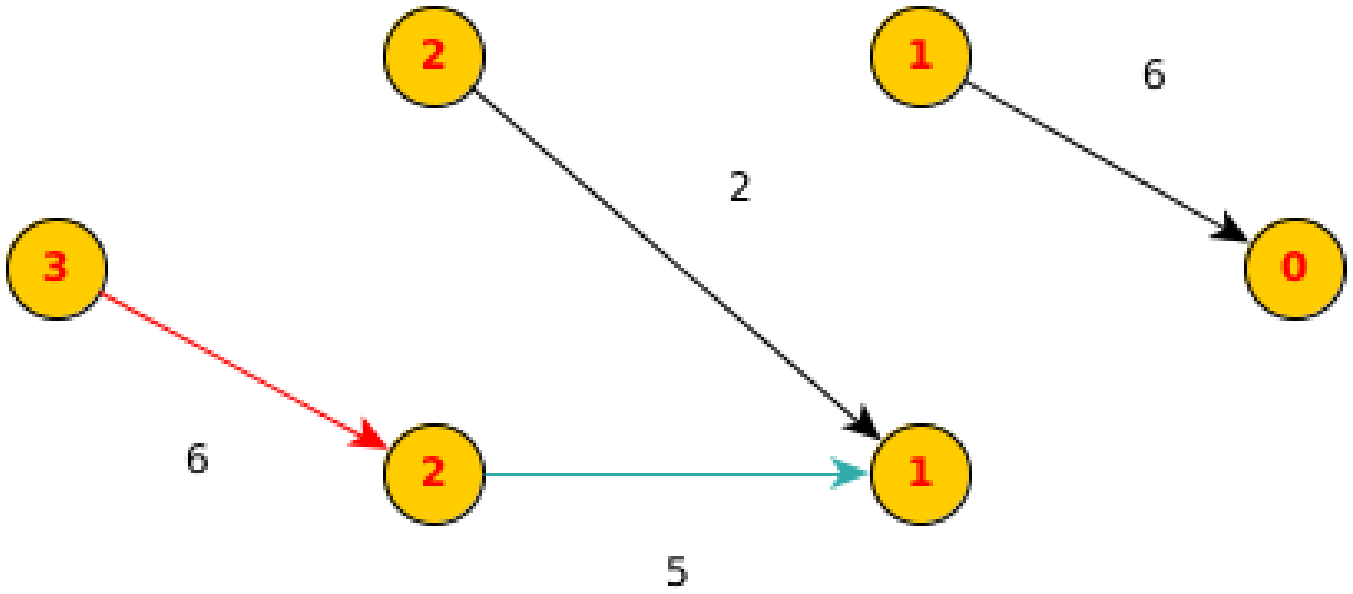
Recorremos el camino encontrado, pasando 4 unidades de flujo por cada arista.



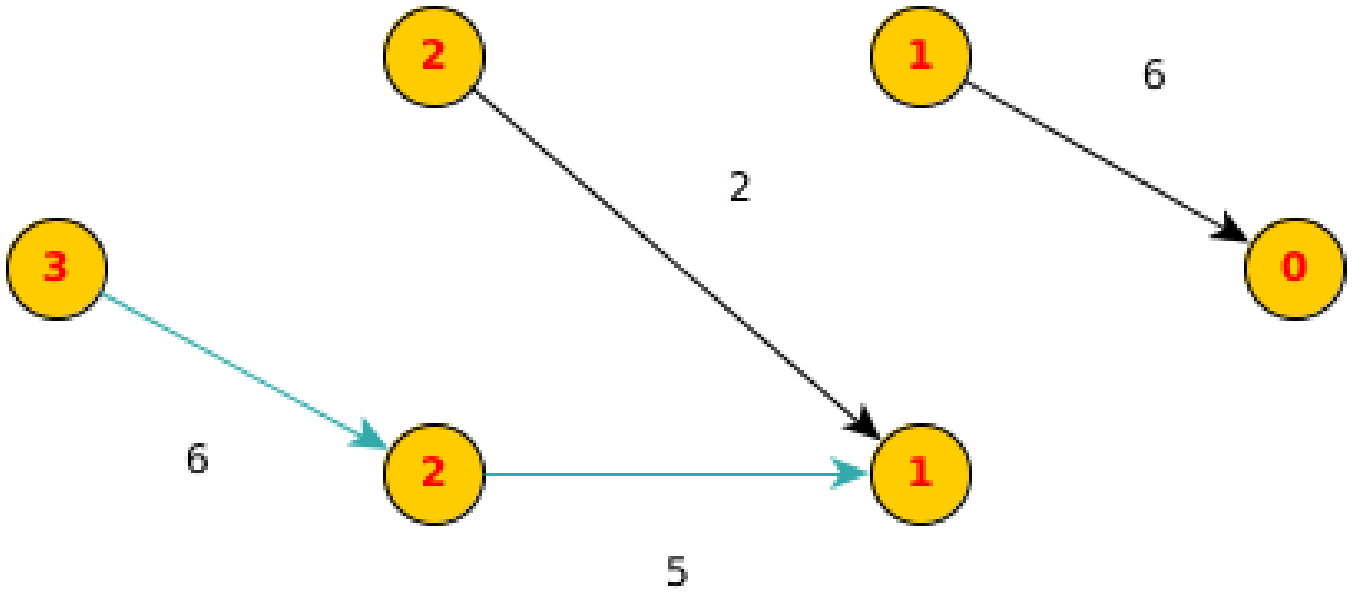
Ejecución



Ejecución

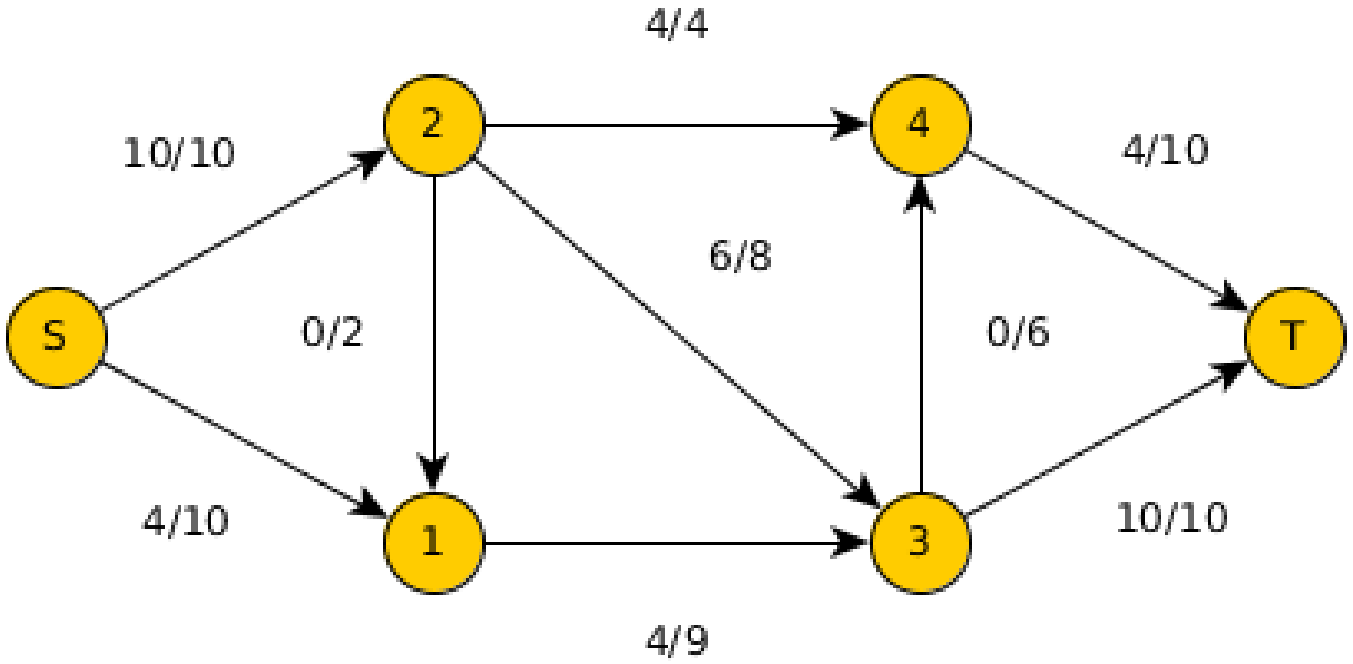


Ejecución

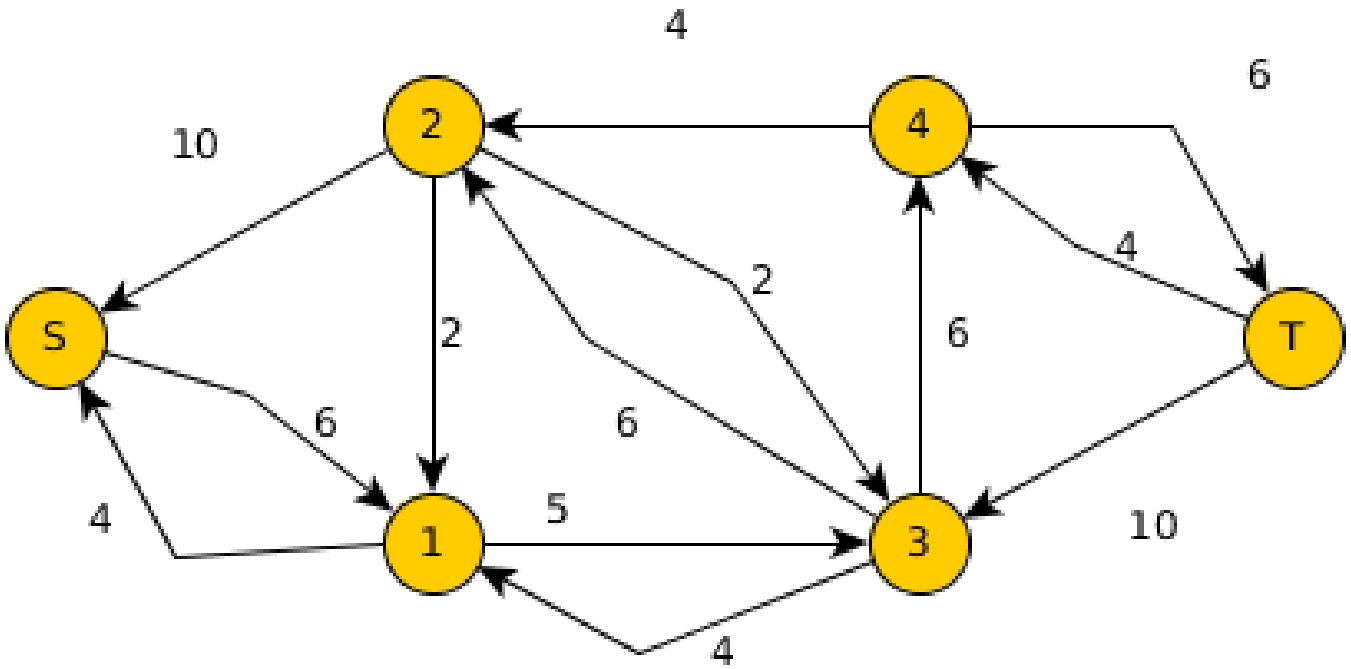


Ejecución

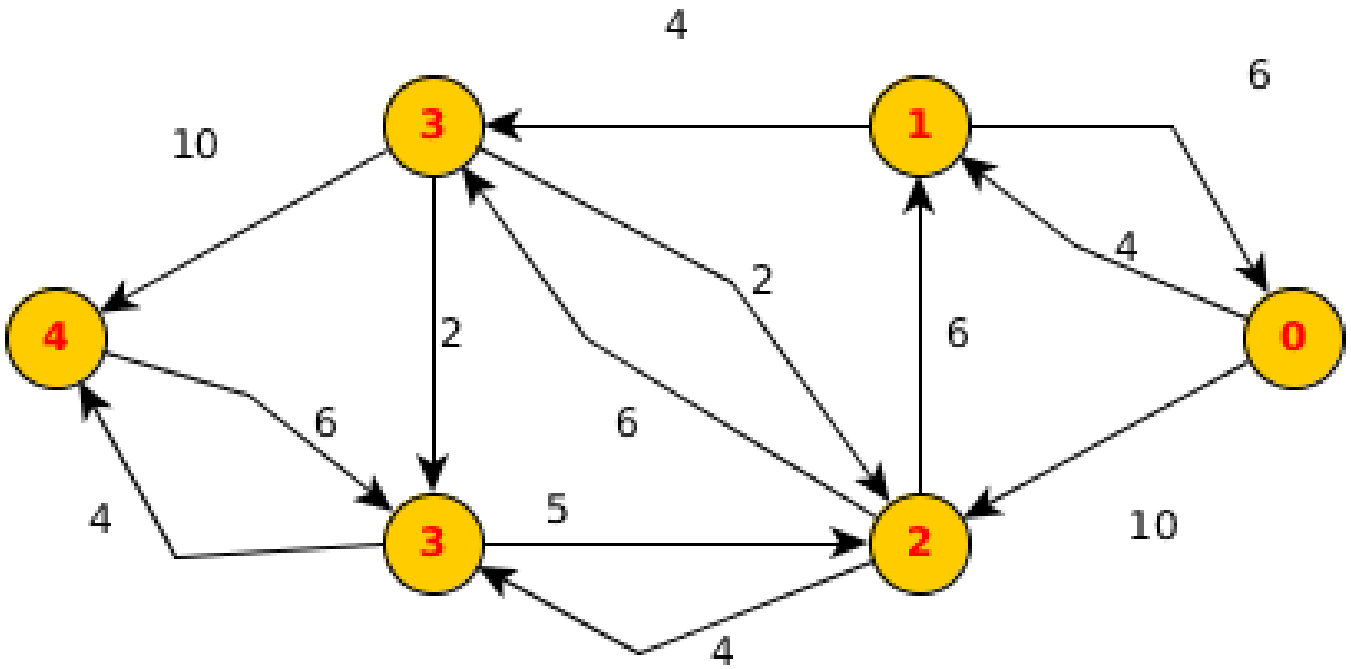
Flujo actual = 14



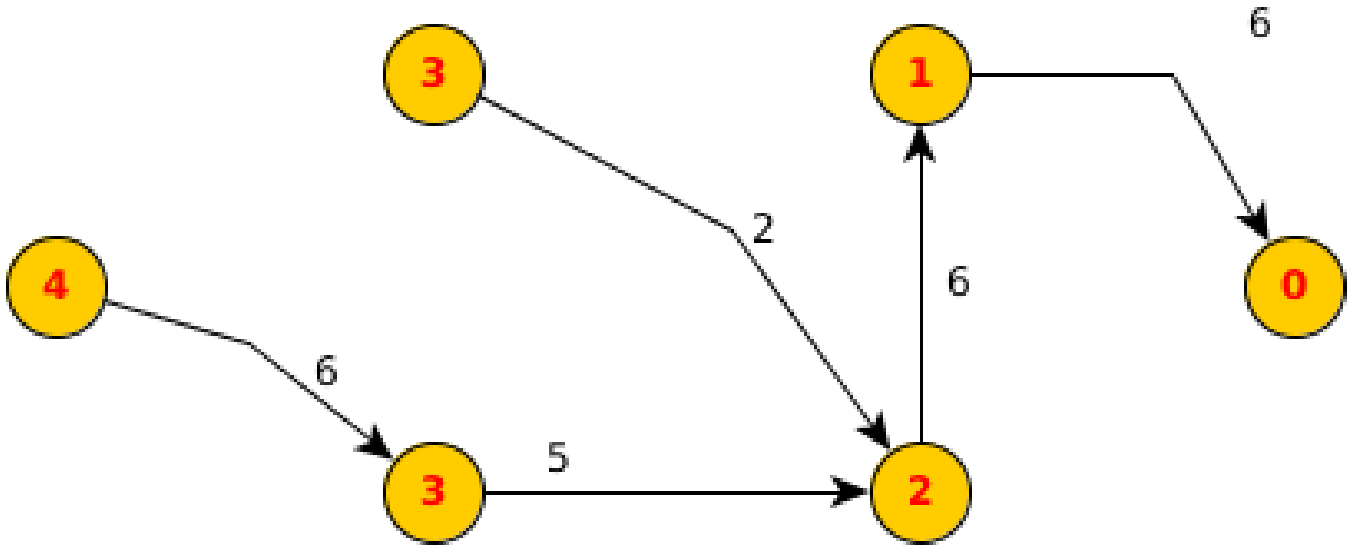
Ejecución



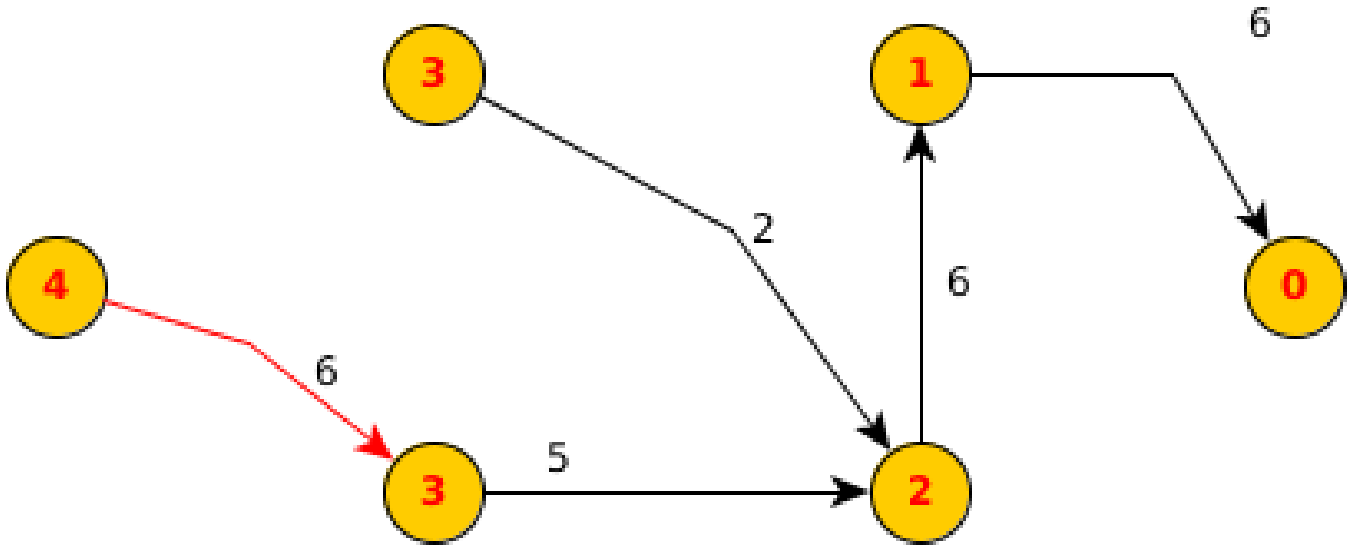
Ejecución



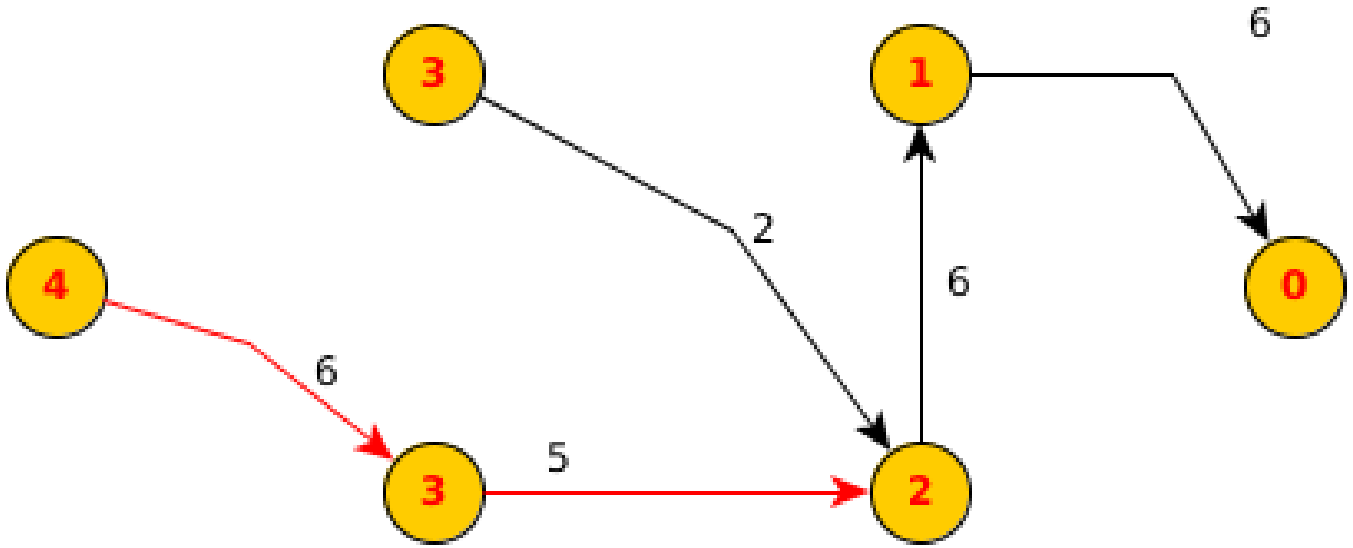
Ejecución



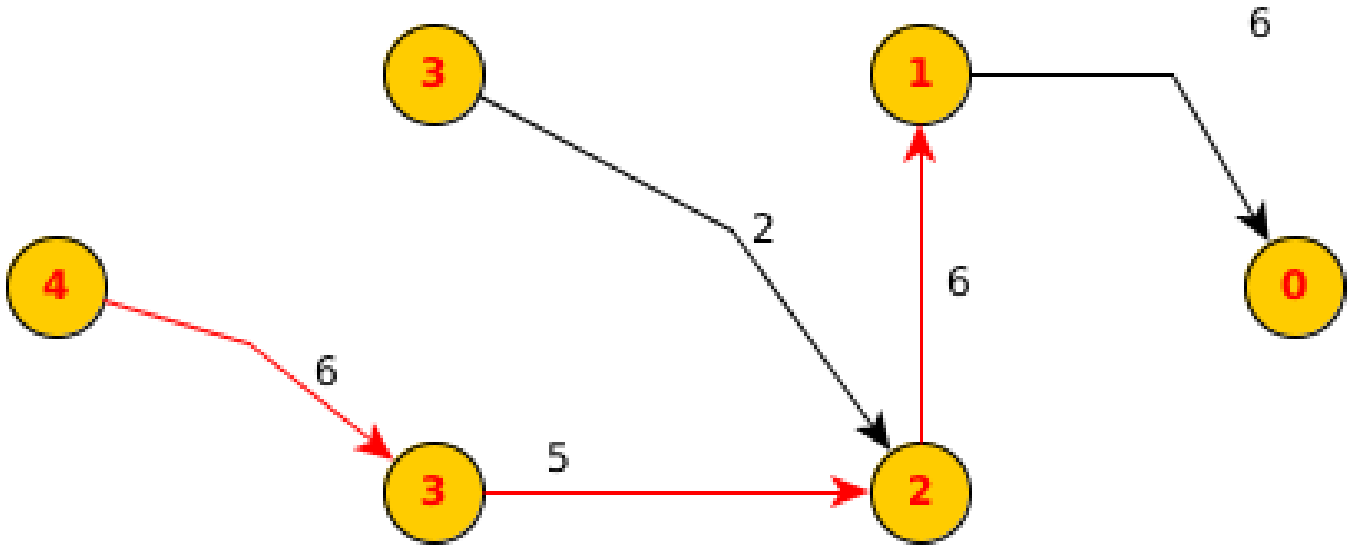
Ejecución



Ejecución

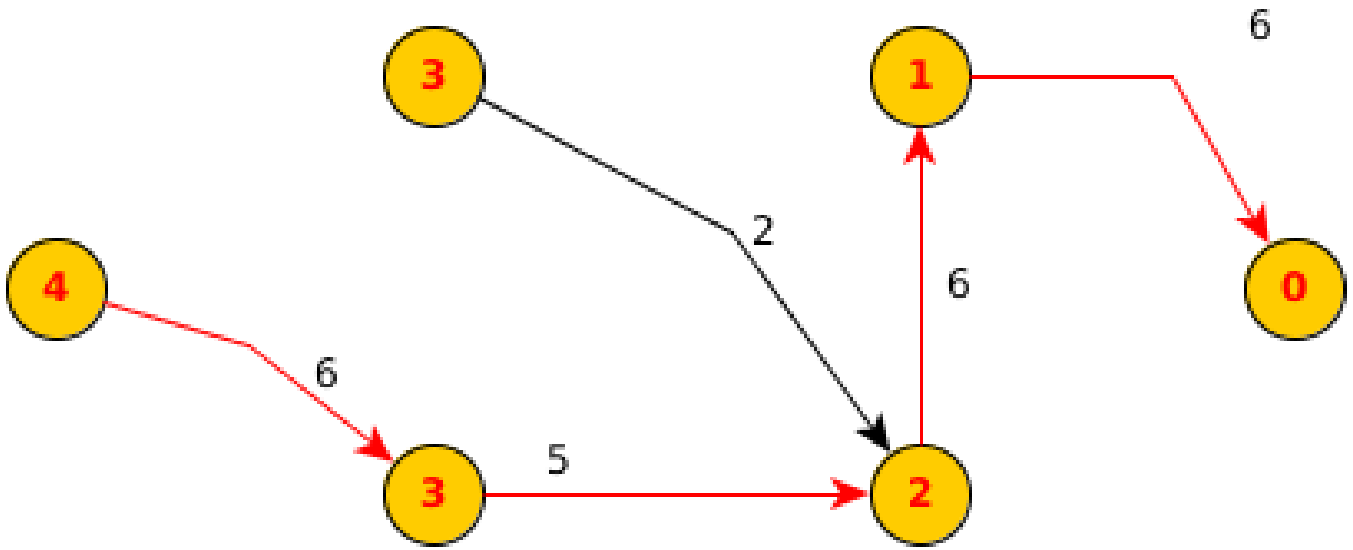


Ejecución

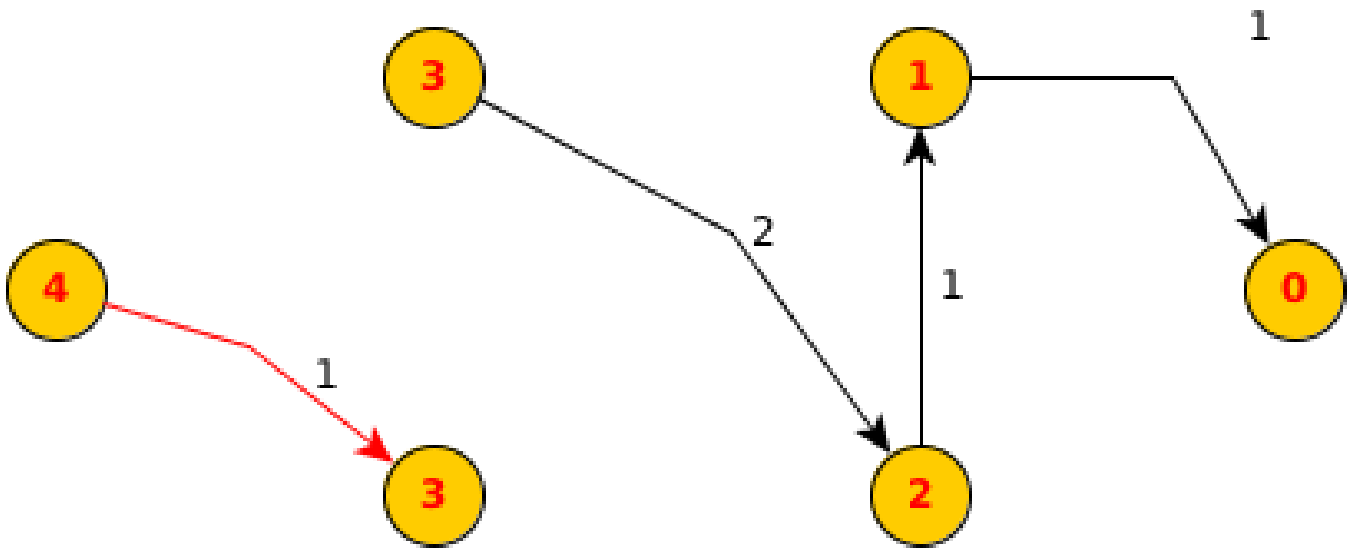


Ejecución

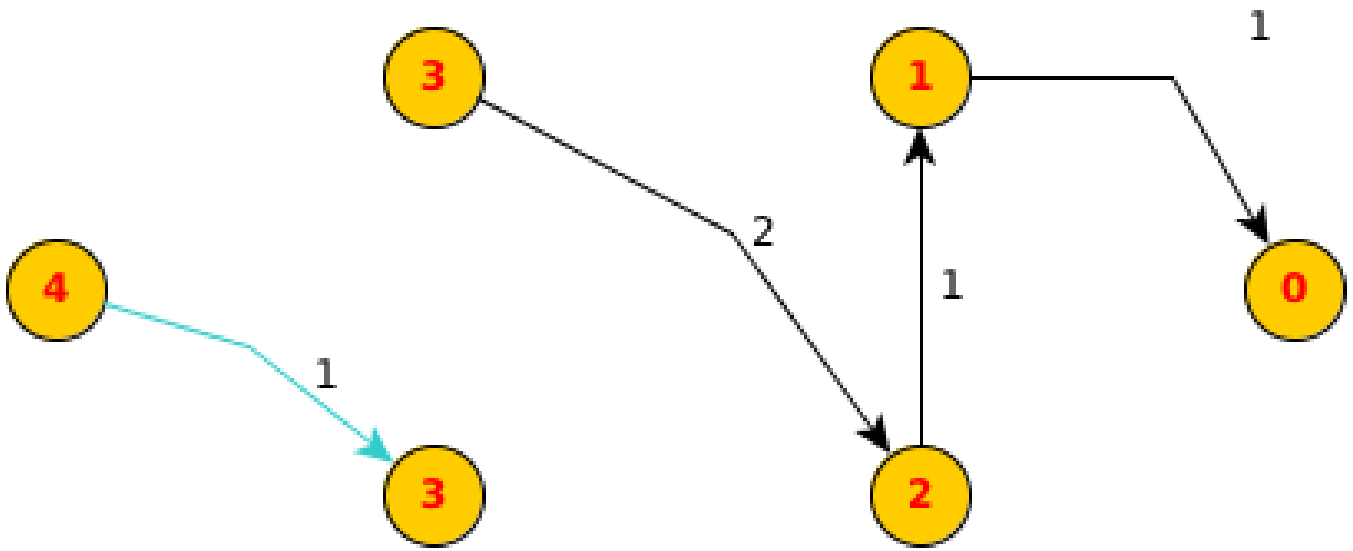
Recorremos el camino encontrado, pasando 5 unidades de flujo por cada arista.



Ejecución

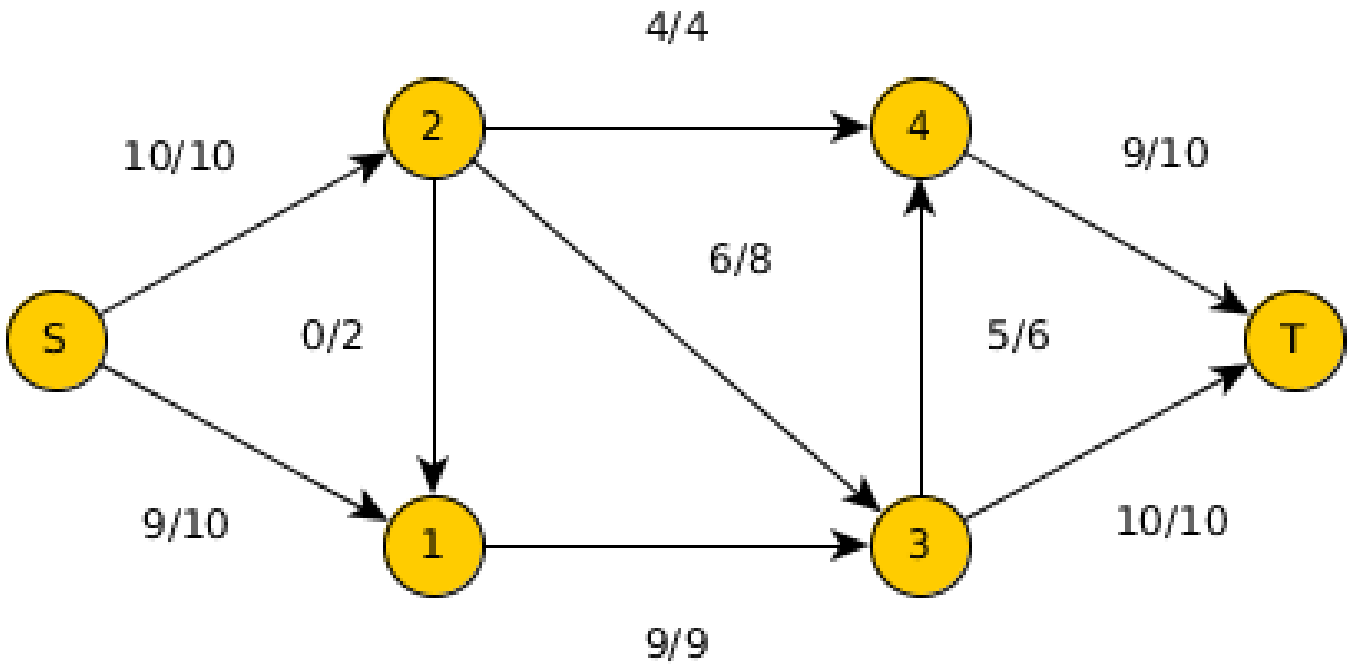


Ejecución



Ejecución

Flujo actual = 19.
Es máximo, así que el BFS no llegará a s desde t ($rank(s) = \infty$).



Complejidad

- El algoritmo realiza $O(V)$ iteraciones, y en cada una realiza un BFS y un DFS.
- La complejidad **NO ES** $O(VE)$ (Esto pensó Dinitz inicialmente, antes de ir la clase de flujo máximo). ¿Por qué?
- En peor caso la complejidad es $O(V^2E)$, y existen casos donde el algoritmo realiza esa cantidad de pasos.

Casos especiales:

- Si todas las capacidades son 1, la complejidad es $O(min(E^{\frac{1}{2}}, V^{\frac{2}{3}})E)$
- Si se usa Dinitz para resolver máximo matching bipartito, la complejidad resultante es $O(\sqrt{V}E)$

Complejidad

- El algoritmo realiza $O(V)$ iteraciones, y en cada una realiza un BFS y un DFS.
- La complejidad **NO ES** $O(VE)$ (Esto pensó Dinitz inicialmente, antes de ir la clase de flujo máximo). ¿Por qué?
- En peor caso la complejidad es $O(V^2E)$, y existen casos donde el algoritmo realiza esa cantidad de pasos.

Casos especiales:

- Si todas las capacidades son 1, la complejidad es $O(min(E^{\frac{1}{2}}, V^{\frac{2}{3}})E)$
- Si se usa Dinitz para resolver máximo matching bipartito, la complejidad resultante es $O(\sqrt{V}E)$

Sugerencias de implementación

- Nunca borrar ni agregar aristas explícitamente: Con solo calcular el rank y saber el flujo (o la capacidad residual) actual, es posible saber si una arista hay que recorrerla o ignorarla.
- Se recorren solo las aristas con capacidad residual > 0, que van a un vecino con menor rank.
- Notar que DFS *vuelve para atrás* al saturar un camino, con lo cual puede *revisitar nodos*.
- Para esto es cómodo implementarlo no-recursivamente, con una pila de nodos para el camino en construcción, y un indicador de "siguiente arista por considerar" por cada nodo.
- Para eficiencia y practicidad, es conveniente guardar solamente la capacidad residual de las aristas. El flujo final puede reconstruirse de ser necesario restando de las capacidades iniciales.

Referencias

- *Introduction to Algorithms, 2nd Edition*. MIT Press. Thomas H. Cormen, y otros.
- "*Dinitz' Algorithm: The Original Version and Even's Version*" Yefim Dinitz