

# Semillero de Programación

## Algoritmo de máximo flujo

Ana Echavarría    Juan Francisco Cardona

Universidad EAFIT

26 de abril de 2013

# Contenido

- 1 Problemas semana anterior
  - Problema A - I love strings!
  - Problema B - Power strings
- 2 Motivación
- 3 Problema de máximo flujo
  - Redes de flujos
  - Red residual
  - Camino de aumentación
  - Teorema de máximo flujo mínimo corte
- 4 Algoritmo
- 5 Algoritmo de Ford-Fulkerson
- 6 Algoritmo de Edmonds-Karp
- 7 Casos especiales
- 8 Tarea

# Contenido

- 1 Problemas semana anterior
  - Problema A - I love strings!
  - Problema B - Power strings

## Problema A - I love strings!

- Hay que hallar rápidamente si un string  $s$  aparece en un string  $t$ .
- Utilizar kmp y retornar verdadero o falso dependiendo de si se encontró el string o no.

# Implementación I

```
1  bool kmp(const string &needle, const string &haystack){
2      int m = needle.size();
3      vector<int> border(m);
4      border[0] = 0;
5
6      for (int i = 1; i < m; ++i) {
7          border[i] = border[i - 1];
8          while (border[i] > 0 and needle[i] != needle[border[i]])
9              border[i] = border[border[i] - 1];
10         if (needle[i] == needle[border[i]]) border[i]++;
11     }
12
13     int n = haystack.size();
14     int seen = 0;
15     for (int i = 0; i < n; ++i){
16         while (seen > 0 and haystack[i] != needle[seen])
17             seen = border[seen - 1];
18         if (haystack[i] == needle[seen]) seen++;
```

# Implementación II

```
19         if (seen == m) return true;
20     }
21     return false;
22 }
23
24 int main(){
25     int cases; cin >> cases;
26     while(cases--){
27         string haystack; int q;
28         cin >> haystack >> q;
29         while (q--){
30             string needle; cin >> needle;
31             if (kmp(needle, haystack)) puts("y");
32             else puts("n");
33         }
34     }
35     return 0;
36 }
```

# Problema B - Power strings

- Un string  $s$  de tamaño  $n$  se dice que tiene periodo  $k$  si  $s(x+k) = s(x) \forall x : x+k < n$
- Notemos que el mínimo  $k$  que cumpla esa propiedad es  $n - \text{border}[n-1]$
- Este  $k$  es mínimo porque  $\text{border}[n-1]$  es máximo

---

- En el problema nos interesa que el mínimo periodo (prefijo de tamaño  $k$ ) aparezca un número entero de veces en la cadena
- Para comprobar esto último basta con ver que  $n$  sea divisible por  $k$
- Si  $n$  no es divisible por  $k$  entonces el mínimo periodo no aparece un número entero de veces y la respuesta es 1.



# Implementación I

```
1  const int MAXN = 1000005;
2  int border[MAXN];
3
4  int main(){
5      ios::sync_with_stdio(false); // Para hacer la I/O mas rapido
6      // Con esa linea ya slo se puede usar cin y cout si no el
        programa se enloquece con la entrada y la salida
7      string s;
8      while (cin >> s){
9          if (s == ".") break;
10         int n = s.size();
11
12         border[0] = 0;
13         for (int i = 1; i < n; ++i) {
14             border[i] = border[i - 1];
15             while (border[i] > 0 and s[i] != s[border[i]]) {
16                 border[i] = border[border[i] - 1];
17             }
```

## Implementación II

```

18         if (s[i] == s[border[i]]) border[i]++;
19     }
20
21     int base = n - border[n-1];
22     if (n % base == 0) cout << n / base << endl;
23     else cout << 1 << endl;
24 }
25 return 0;
26 }

```

# Contenido

## 2 Motivación

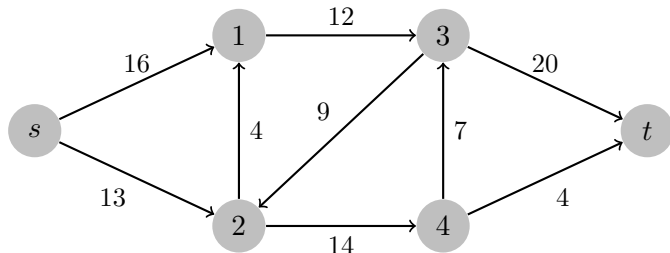
# Red de flujos

- De igual manera como se puede crear un grafo para modelar rutas y hallar la mínima distancia entre dos lugares, se pueden usar grafos para representar una **red de flujos**.
- Acá los nodos son lugares a los que hay que transportar material y los pesos de cada arista es la **tasa máxima** a la cual ese material fluye entre los nodos.
- Algunos ejemplos en las que las redes de flujos se usan para modelar problemas son: líquido que fluye por tuberías, corriente eléctrica que fluye por un cableado, flujo de producción en una línea de ensamble, etc.

# Ejemplo

## Ejemplo

Se tiene un sistema de tuberías unidireccionales que salen desde la planta de tratamientos en la ciudad  $s$  y pasan por varias ciudades incluida la ciudad  $t$ . La capacidad máxima de las tuberías (litros/hora) entre cada par de ciudades es conocida. ¿Cuál es la máxima cantidad de agua que puede llegar a la ciudad  $t$  en una hora?



# Contenido

- 3 Problema de máximo flujo
  - Redes de flujos
  - Red residual
  - Camino de aumentación
  - Teorema de máximo flujo mínimo corte

# Red de flujos

## Red de flujos

Una red de flujos  $G = (V, E)$  es un grafo conexo y dirigido en donde cada arista  $(u, v) \in E$  tiene una capacidad **no negativa**  $c(u, v) \geq 0$ .

Si  $(u, v) \notin E$  entonces  $c(u, v) = 0$

Se distinguen dos vértices: la fuente  $s$  y el sumidero  $t$ .

# Flujo

Sea  $G = (V, E)$  una red de flujos con fuente  $s$  y sumidero  $t$ .

## Flujos

Un flujo es una función  $f : V \times V \rightarrow \mathbb{R}$  que cumple que:

- **Restricción de capacidad:**

$$f(u, v) \leq c(u, v) \quad \forall u, v \in V$$

- **Simetría:**

$$f(u, v) = -f(v, u) \quad \forall u, v \in V$$

- **Conservación de flujo:** El flujo que sale de un nodo diferente de  $s$  y  $t$  es igual al que entra al nodo.

$$\forall u \in V - \{s, t\}$$

$$\sum_{v \in V} f(u, v) = \sum_{\substack{v \in V \\ f(u, v) < 0}} f(u, v) + \sum_{\substack{v \in V \\ f(u, v) > 0}} f(u, v) = 0$$



# Problema de máximo flujo

Sea  $G = (V, E)$  una red de flujos con fuente  $s$ , sumidero  $t$ , función de flujos  $f$ .

## Definiciones

- Al valor  $f(u, v)$  se le llama flujo del nodo  $u$  al nodo  $v$ .
- El valor del flujo de  $G$  se denota por  $|f|$  y corresponde al flujo que sale de  $s$  menos el flujo que entra a  $s$

$$|f| = \sum_{v \in V} f(s, v)$$

## Problema de máximo flujo

Dados  $G$ ,  $s$  y  $t$  hallar un flujo de  $G$  que tenga valor máximo.

# Red residual

Dados una red e flujos  $G = (V, E)$  y un flujo  $f$ .

La red residual de  $G$  dado que se ha “bombeado” un flujo  $f$  es un grafo  $G_f = (V, E_f)$  con una función de costos  $c_f$ .

El peso  $c_f$  de una arista  $(u, v) \in E_f$  es el valor de flujo que todavía se puede “bombear” desde  $u$  hasta  $v$  sin violar la capacidad de  $(u, v)$ .

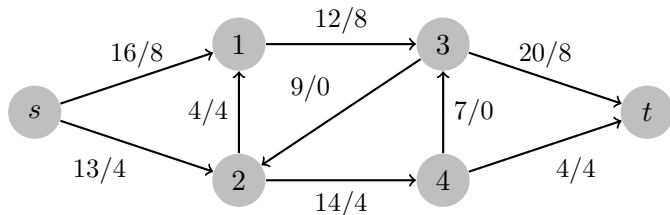
$$c_f(u, v) = c(u, v) - f(u, v)$$

# Ejemplos

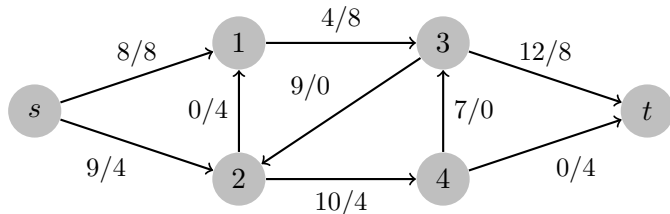
- Si  $c(u, v) = 16$  y  $f(u, v) = 11$  entonces  $c_f(u, v) = 16 - 11 = 5$ . Esto significa que por la arista  $(u, v)$  todavía se pueden bombear 5 unidades de flujo sin violar la capacidad.
- Si  $c(u, v) = 16$  y  $f(u, v) = -4$  (es decir que  $f(v, u) = 4$ ) entonces  $c_f(u, v) = 16 - (-4) = 20$ . Esto significa que por la arista  $(u, v)$  no sólo se pueden “bombear” las 16 unidades que tiene de capacidad esa arista, sino que también se pueden “desbombear” las 4 unidades que se habían bombeado de  $v$  a  $u$ .

# Ejemplo

Red de flujos: Peso de arista  $(a, b)$  es  $c(a, b)/f(a, b)$



Red residual: Peso de arista  $(a, b)$  es  $c_f(a, b)/c_f(b, a)$



# Camino de aumentación

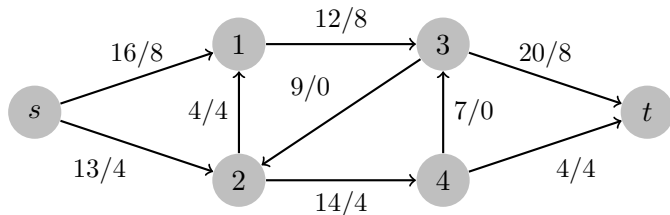
## Camino de aumentación

Dada una red residual  $G_f$  un camino de aumentación es un camino simple (sin ciclos)  $p$  que va de  $s$  a  $t$  en  $G_f$  tomando solo aristas de peso mayor que 0.

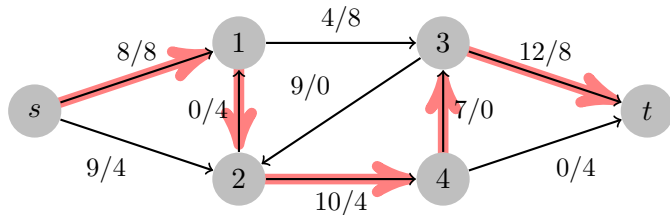
En otras palabras un camino de aumentación es un camino por el cual todavía se puede bombear flujo de  $s$  a  $t$  en  $G$  sin violar las restricciones de capacidad.

# Ejemplo

Red de flujos: Valor escrito en arista  $(a, b)$  es  $c(a, b)/f(a, b)$



Red residual: Valor escrito en arista  $(a, b)$  es  $c_f(a, b)/c_f(b, a)$



# Pregunta

## Pregunta

En el camino anterior, ¿cuál era la máxima cantidad de flujo que se podía “bombear” sin violar las restricciones de capacidad?

## Pregunta

## Pregunta

En el camino anterior, ¿cuál era la máxima cantidad de flujo que se podía “bombear” sin violar las restricciones de capacidad?

## Respuesta

El máximo valor que se puede “bombear” es el mínimo de los valores de las aristas del camino  $\min\{8, 4, 10, 7, 12\} = 4$ .

Este valor tiene el nombre de cuello de botella o bottleneck.



## Cuello de botella

La máxima cantidad de flujo que se puede enviar por un camino de aumentación  $p$  corresponde al mínimo valor de las aristas de  $p$  en la red residual.

Este valor se conoce como cuello de botella

$$\text{bottleneck}(p) = \min\{c_f(u, v) : (u, v) \in p\}$$

# Aumentando el flujo

Se tienen una red de flujos  $G = (V, E)$ , un flujo  $f$  y un camino  $p$  en la red residual.

Si se aumenta el flujo  $f(u, v)$  en el valor  $\text{bottleneck}(p)$  para cada  $(u, v) \in p$  y se reduce en  $\text{bottleneck}(p)$  para  $(v, u)$  entonces:

- El valor resultante para  $f(u, v)$  cumple con las propiedades de flujo.
- El valor del nuevo flujo es el valor del antiguo flujo más  $\text{bottleneck}(p)$ .

## Teorema de máximo flujo mínimo corte

## Max-flow min-cut

Si  $f$  es un flujo en una red de flujos  $G = (V, E)$  con fuente  $s$  y sumidero  $t$  entonces las siguientes condiciones son equivalentes:

- $f$  es un flujo máximo en  $G$ .
- La red residual  $G_f$  no contiene caminos de aumentación
- $|f|$  es el mínimo costo de cortar aristas de  $G$  de manera que  $s$  y  $t$  queden separados cuando costo de cortar una arista es el peso de la arista.

# Contenido

## 4 Algoritmo

# Algoritmo de máximo flujo

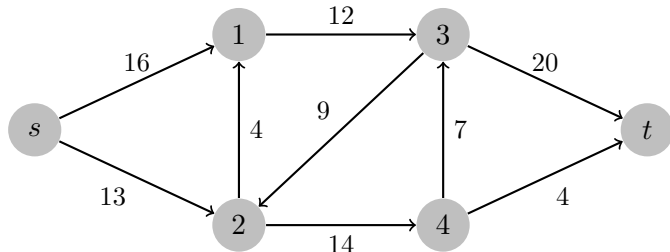
- 1 Inicializar el flujo en 0.
- 2 Mientras que haya caminos de aumentación  $p$  en  $G_f$ 
  - 3 Aumente el flujo  $f$  a lo largo de  $p$  en el valor  $\text{bottleneck}(p)$
- 4 retornar  $|f|$

## ¿Por qué funciona?

- Cuando  $f$  es 0 se cumplen las restricciones de flujo.
- En cada iteración  $f$  sigue cumpliendo las restricciones de flujo porque ya vimos que al aumentar  $f$  en el valor del cuello de botella de  $p$  (para las aristas de  $p$ ) no se violan las restricciones necesarias para ser un flujo.
- Cuando se termina la ejecución del algoritmo, se tiene un flujo para la red  $G$  y no hay más caminos de aumentación en  $G_f$ .
- El teorema de máximo flujo mínimo corte dice que el flujo  $f$  es máximo cuando no hay más caminos de aumentación

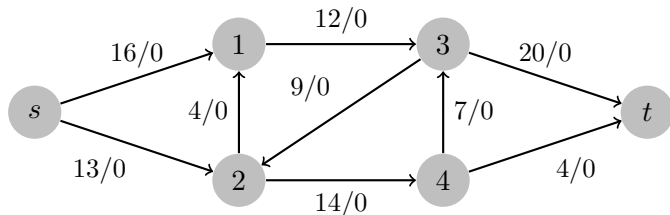
# Ejemplo

Hallar el máximo flujo entre el nodo  $s$  y el nodo  $t$

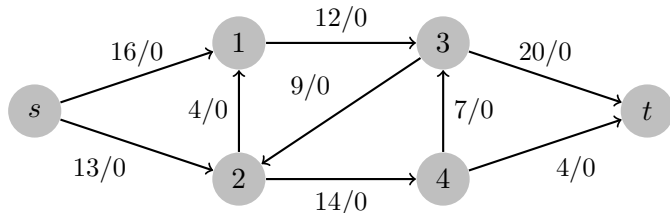


# Ejemplo: Iteración 1

Red de flujos: Peso de arista  $(a, b)$  es  $c(a, b)/f(a, b)$



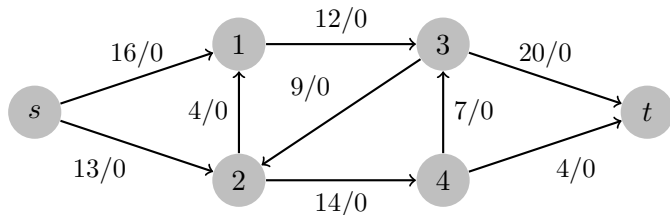
Red residual: Peso de arista  $(a, b)$  es  $c_f(a, b)/c_f(b, a)$



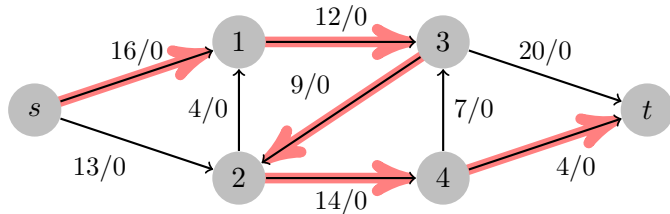


# Ejemplo: Iteración 1

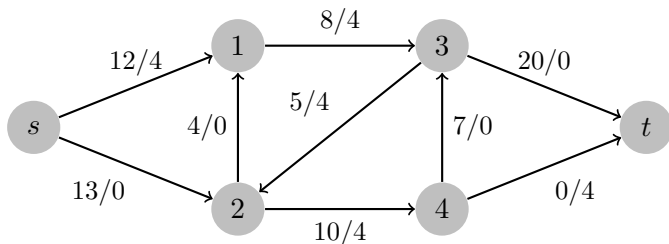
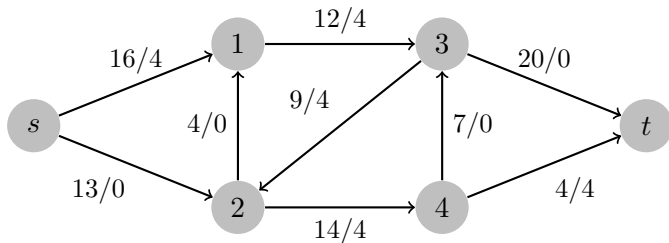
Red de flujos: Peso de arista  $(a, b)$  es  $c(a, b)/f(a, b)$



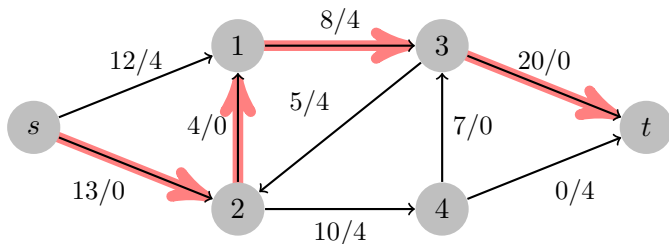
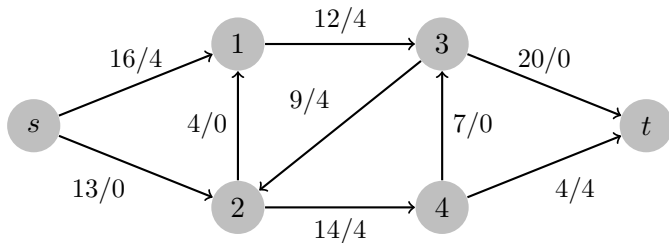
Red residual: Peso de arista  $(a, b)$  es  $c_f(a, b)/c_f(b, a)$



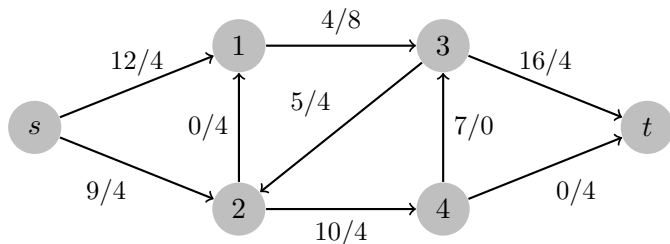
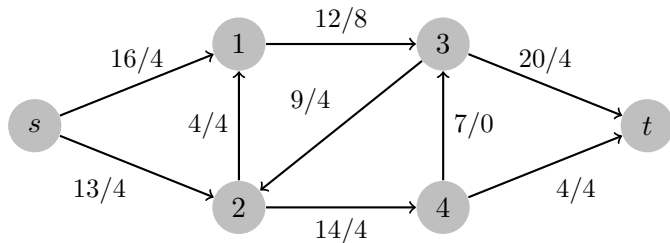
## Ejemplo: Iteración 2



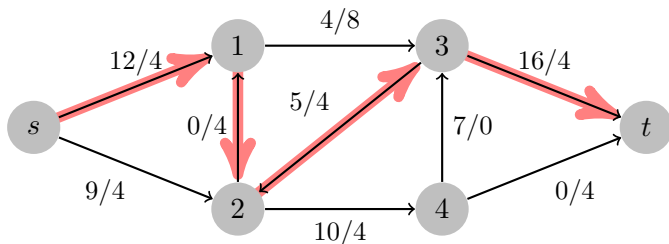
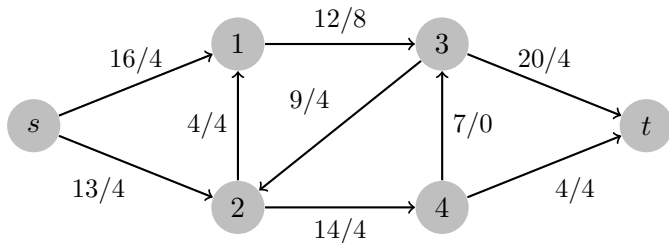
## Ejemplo: Iteración 2



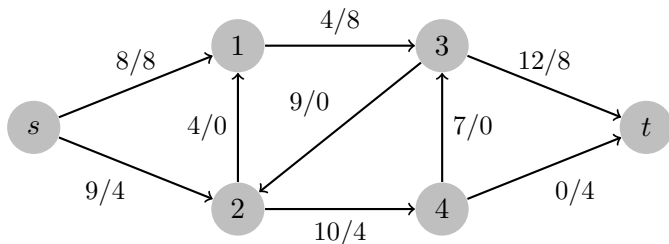
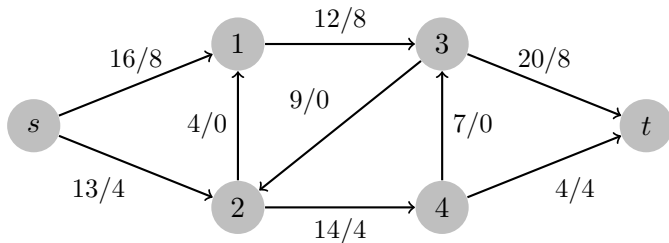
## Ejemplo: Iteración 3



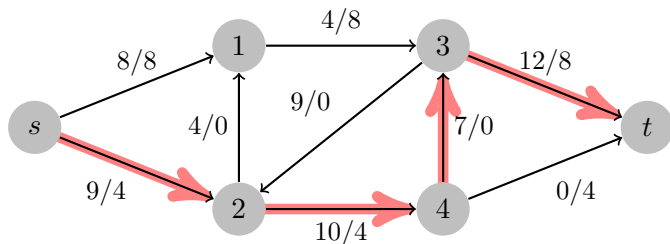
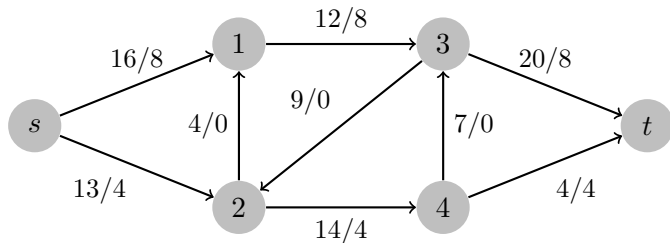
## Ejemplo: Iteración 3



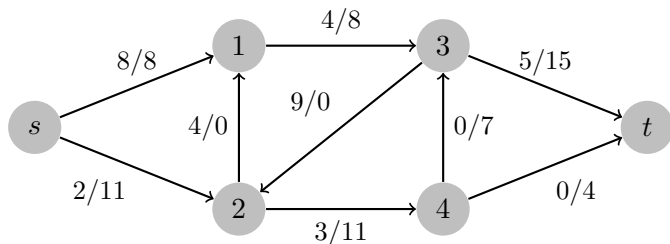
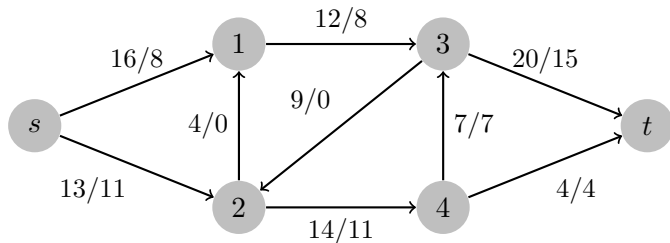
# Ejemplo: Iteración 4



# Ejemplo: Iteración 4

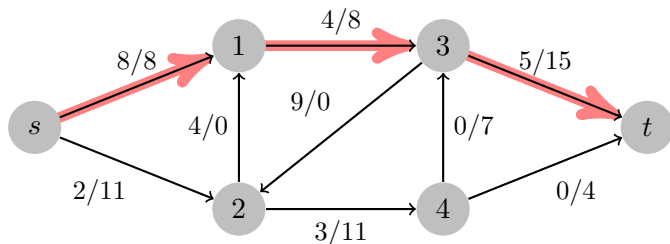
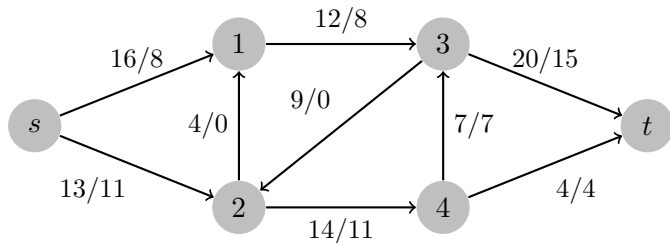


# Ejemplo: Iteración 5

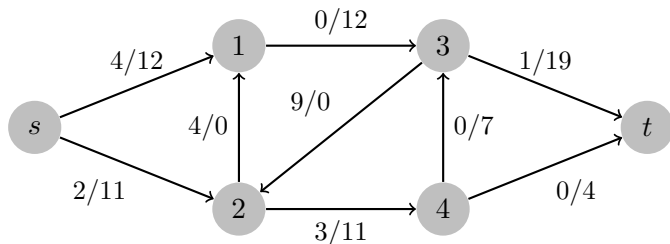
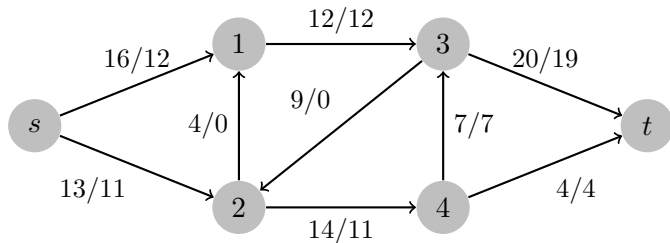




# Ejemplo: Iteración 5

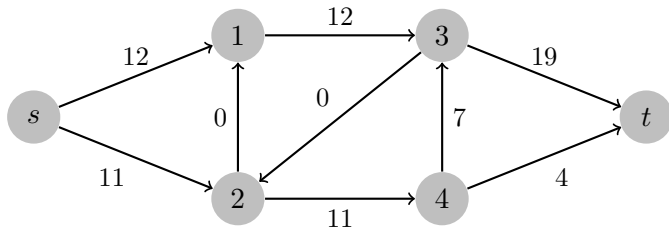


# Ejemplo: Iteración 6



## Ejemplo: Solución

El máximo flujo entre el nodo  $s$  y el nodo  $t$  es (se omiten las aristas de peso negativ)



El valor del flujo es 23.

# Contenido

## 5 Algoritmo de Ford-Fulkerson

# Ford-Fulkerson

El algoritmo de Ford-Fulkerson utiliza DFS para hallar el camino de aumentación.

- ❶ Para todo  $u$  y  $v$   $f(u, v) = 0$ .
- ❷  $flow = 0$
- ❸ Mientras que se encuentre un caminos de aumentación  $p$  en  $G_f$  usando DFS
  - ❹  $bottleneck = \min\{c_f(u, v) : (u, v) \in p\}$
  - ❺ Para cada  $(u, v) \in p$ 
    - ❻  $f(u, v) += bottleneck$
    - ❼  $f(v, u) = -f(u, v)$
  - ❽  $flow += bottleneck$
- ❾ retornar  $flow$

# Complejidad

## Complejidad

Si  $f^*$  es el valor del máximo flujo para el grafo  $G = (V, E)$  el algoritmo de Ford-Fulkerson tiene una complejidad de  $O(|E| \cdot f^*)$

# Ejemplo

Figura: Red de flujos

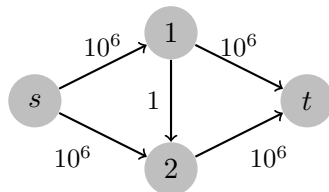


Figura: Red residual iteración 1

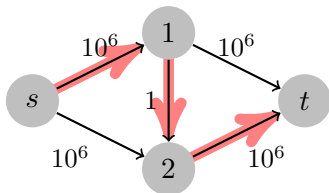
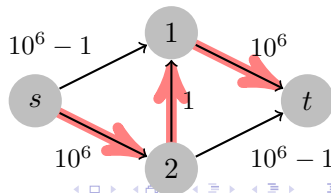


Figura: Red residual iteración 2



# Contenido

## 6 Algoritmo de Edmonds-Karp



# Edmonds-Karp

Se puede mejorar la complejidad del algoritmo usando BFS en lugar de DFS para hallar el camino de aumentación.  
El BFS permite que el camino hallado tenga el mínimo número de aristas posible, lo que cambia la complejidad del algoritmo.

# Ejemplo

Figura: Red de flujos

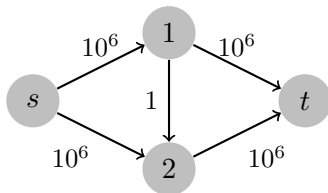


Figura: Red residual iteración 1

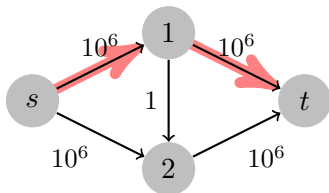
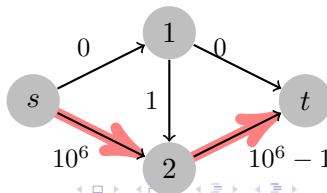


Figura: Red residual iteración 2



# Implementación I



# Complejidad

# Complejidad

El algoritmo de Edmonds-Karp tiene una complejidad de  $O(|V| \cdot |E|^2)$

# Contenido

## 7 Casos especiales

# Problema con múltiples fuentes o múltiples sumideros

# Problema donde los nodos también tienen capacidad

# Contenido

## 8 Tarea



# Tarea

## Tarea

Resolver los problemas de  
<http://contests.factorcomun.org/contests/59>