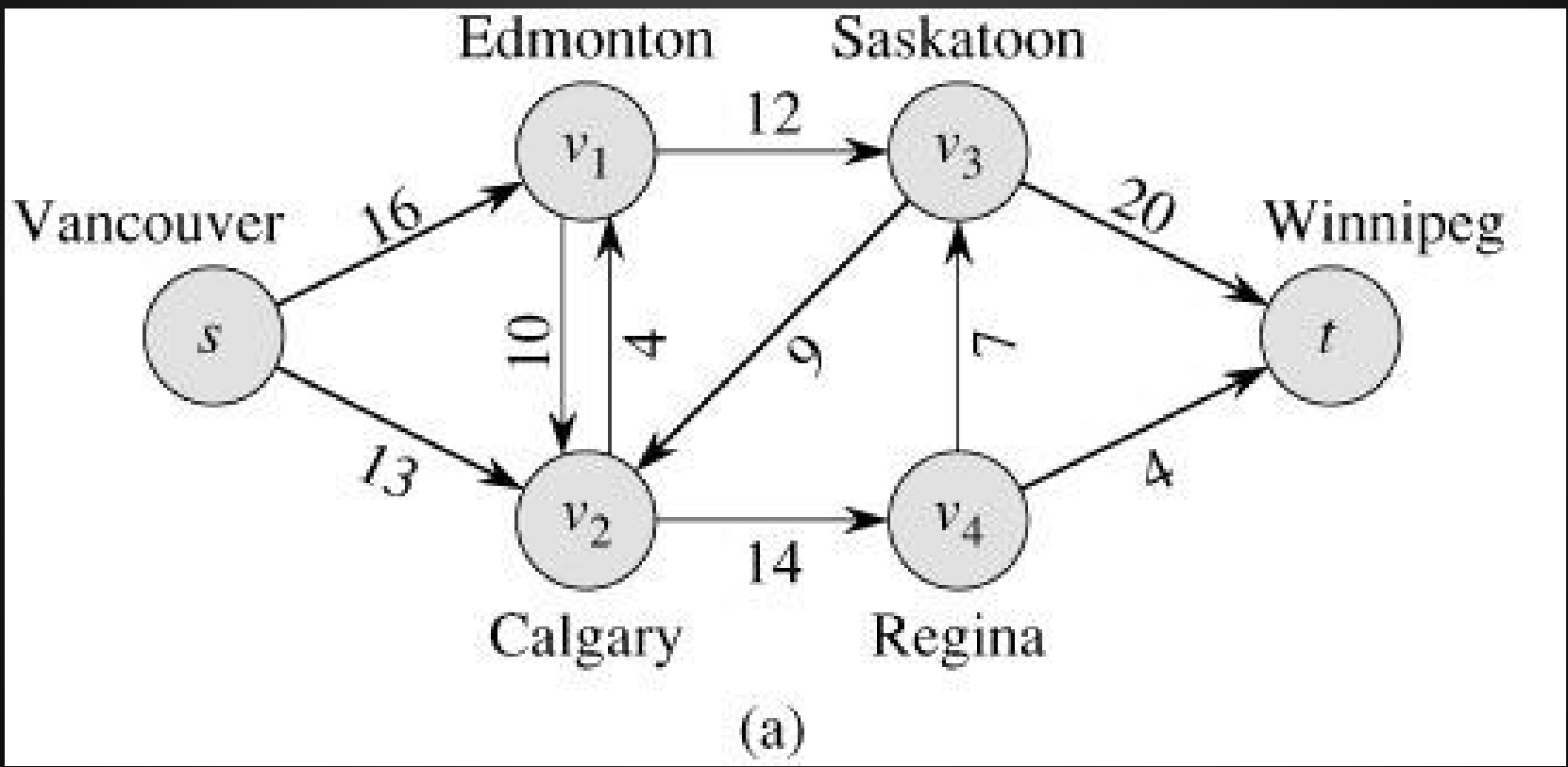


Flujo Máximo

Training Camp Argentina 2013

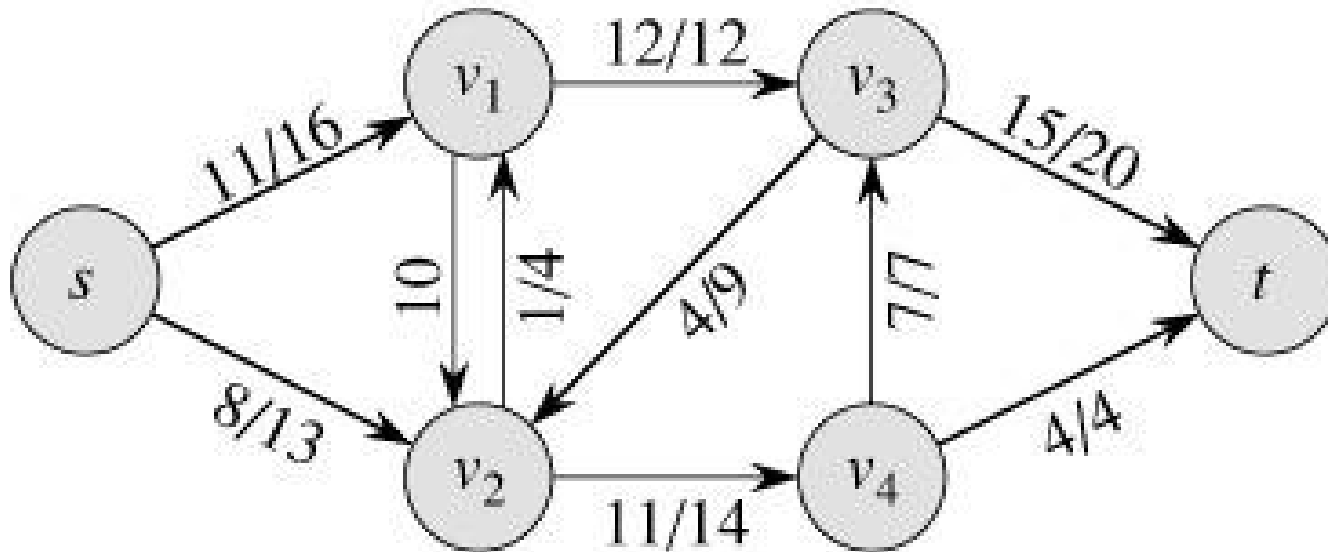
Nicolás Álvarez
naa [at] cs uns edu ar

Idea



Idea

Idea



(b)

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 $G = \langle N, A \rangle$ donde cada arco $(u, v) \in A$ tiene asociado una capacidad $c(u, v) \geq 0$.
- Se distinguen dos nodos **s** y **t** llamados **fuente y destino**, tal que ningún arco llega a la fuente o sale del destino.
- Además, por conveniencia, si $(u, v) \notin A$ entonces se supone $c(u, v) = 0$
- También que para todo u , existe un camino $S \rightarrow u \rightarrow T$

Definición: Flujo

Un **flujo** en G es una función $f : N \times N \rightarrow R$ que satisface:

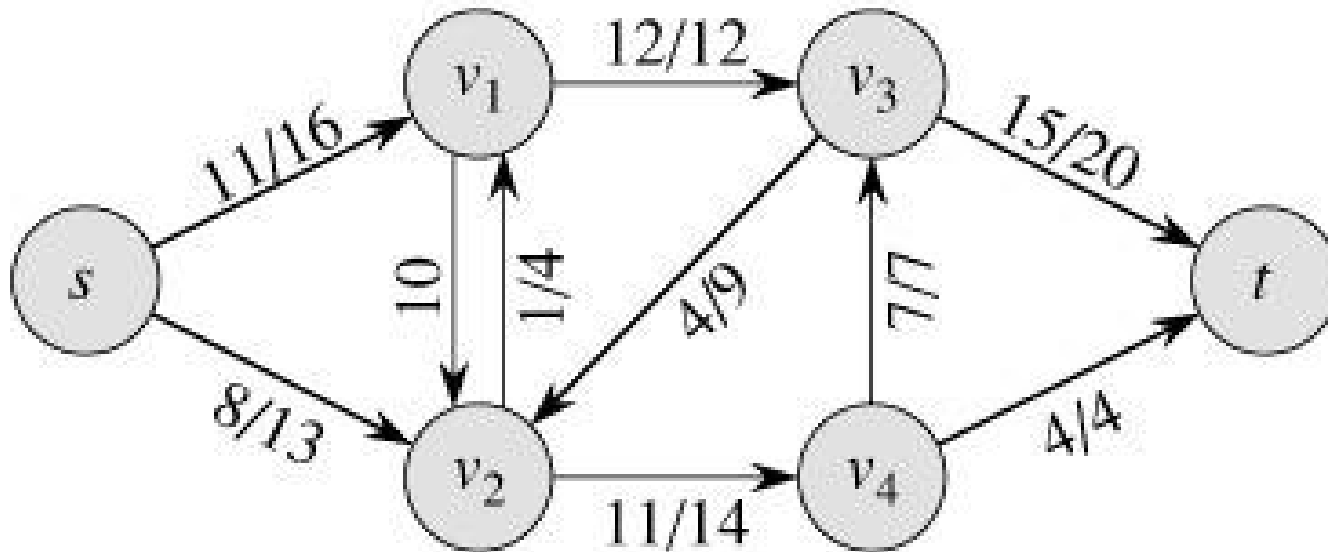
- **restricción de capacidad:**
 - $f(u,v) \leq c(u,v)$ para todo $u,v \in N$.
- **simetría oblicua:**
 - $f(u,v) = -f(v,u)$ para todo $u,v \in N$
- **conservación de flujo:**
 - Para todo $u \in N - \{s,t\}$

$$\sum_{v \in N} f(u,v) = 0$$

Definición: Problema Flujo Máximo

- El **valor de un flujo** $|f|$ se define como
$$|f| = \sum_{v \in N} f(s,v) = \sum_{v \in N} f(v,t)$$
- Dada una red de flujo G , el **Problema de Flujo Máximo** consiste en encontrar el flujo de máximo valor que admite G

Repaso



(b)

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 $G = \langle N, A \rangle$ es $G_f = \langle N, A_f \rangle$, donde las capacidades de los arcos están dadas por:

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

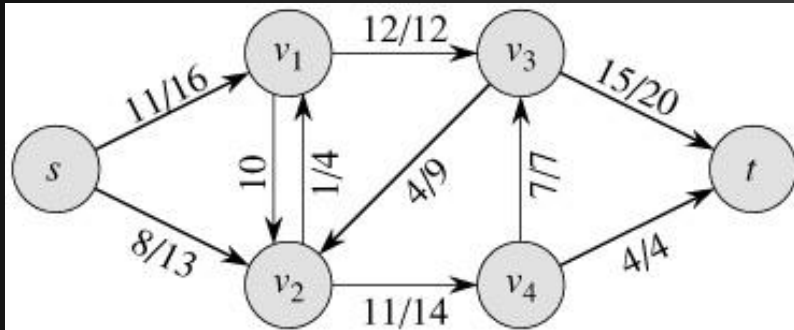
Caminos de aumento

Como vimos, dada una **red de flujo** $G = \langle N, A \rangle$ y un **flujo** f , podemos obtener una **red residual** G_f

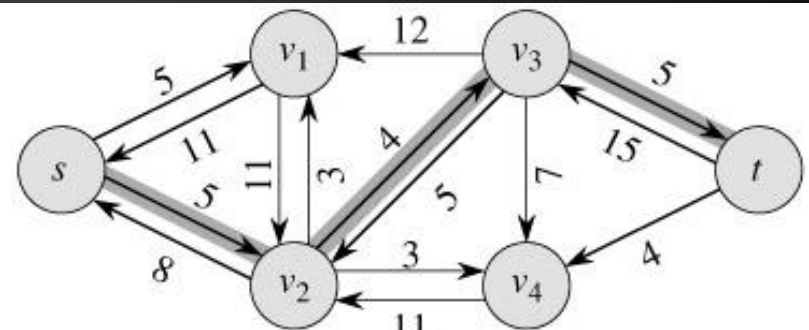
Un camino de aumento es, sencillamente, un camino simple de **s** a **t** en la red residual.

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.

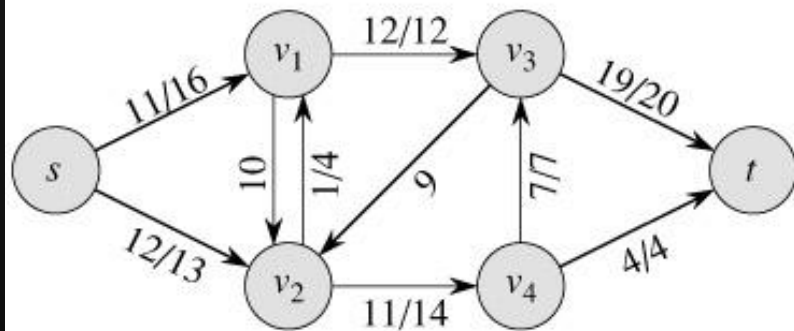
Aumento de flujo



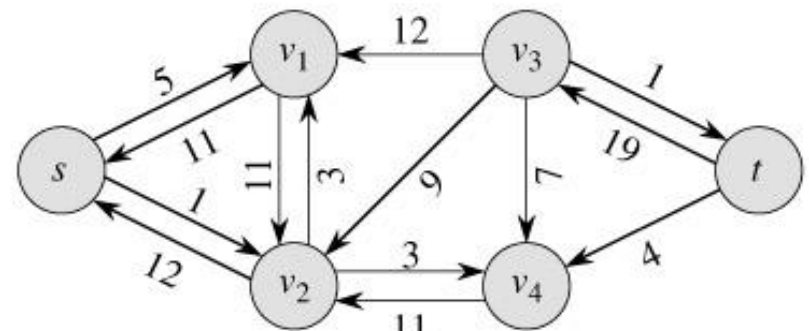
(a)



(b)



(c)



(d)

Cortes

Dada una red de flujo $G = \langle N, A \rangle$. 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) .

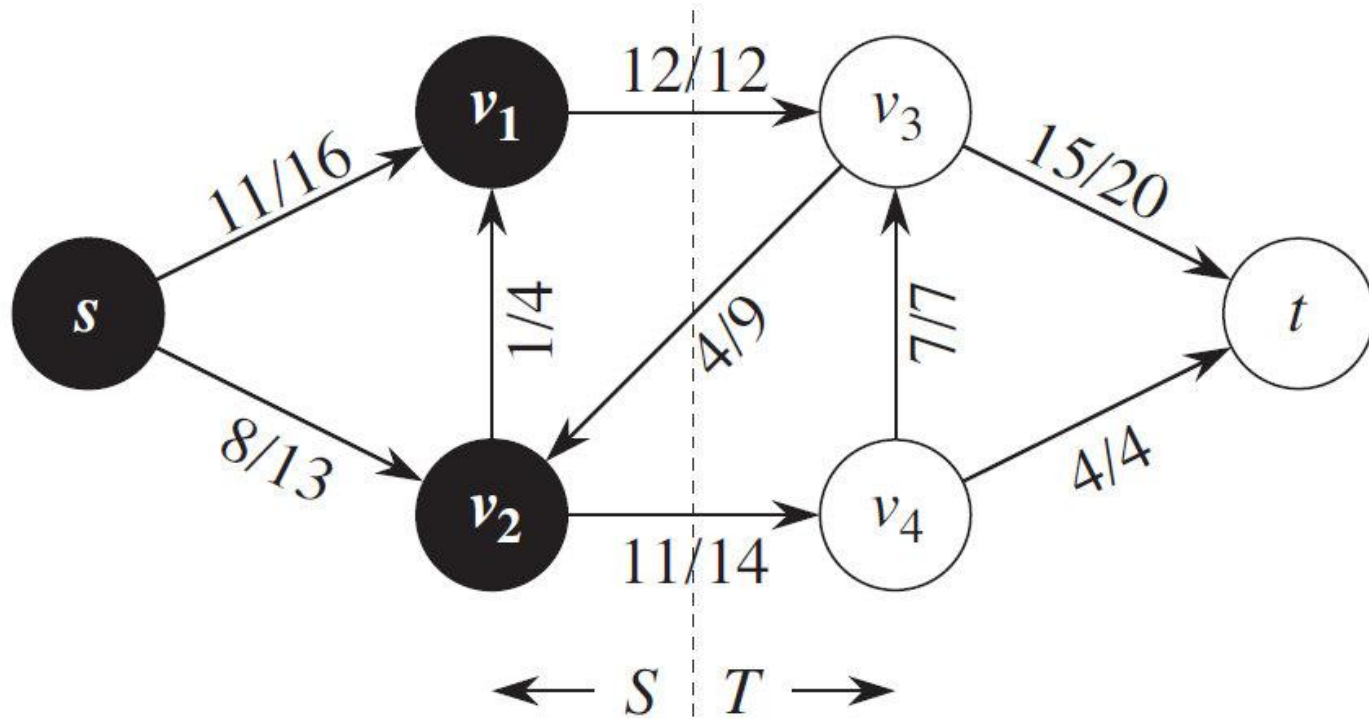
El **flujo neto** a través del corte es:

$$f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u)$$

Y la **capacidad** del corte es:

$$c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v)$$

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 \Rightarrow 2

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

2 \Rightarrow 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 \Rightarrow 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 $(u, v) \in G.E$:

$(u, v).f = 0$

mientras exista camino de aumento p en G_f :

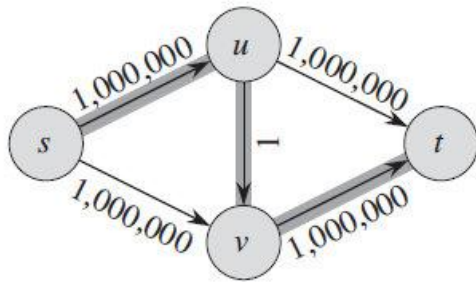
$c_f(p) = \min \{c_f(u, v) : (u, v) \in p\}$

para cada arco $(u, v) \in p$:

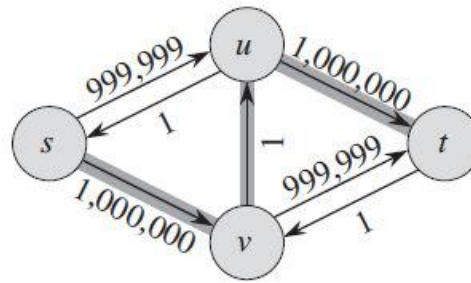
$(u, v).f += c_f(p)$

$(v, u).f -= c_f(p)$

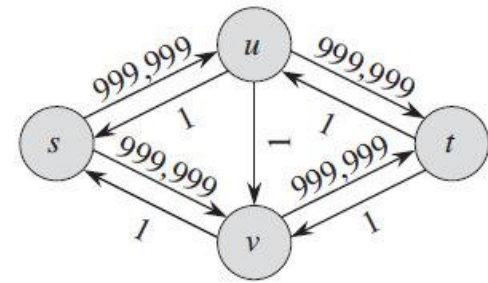
Algoritmo de Edmonds-Karp



(a)



(b)



(c)

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

Problemas

Hooligans - Regional Latinoamérica 2009

goo.gl/3EoOPO

Dado un torneo entre $N \leq 40$ equipos, donde juegan todos contra todos durante $M \leq 4$ rondas y algunos partidos ya jugados. Donde en cada partido se reparten 2 puntos (2 para el ganador, 1 si hay empate y 0 para el perdedor). Determinar si un cierto equipo puede salir campeón.

Problemas

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 $\sum_{(u,v)} \text{marca}(u) \wedge \text{marca}(v)$

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 $G = \langle N, A \rangle$, 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 $G = \langle N, A \rangle$ 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*.

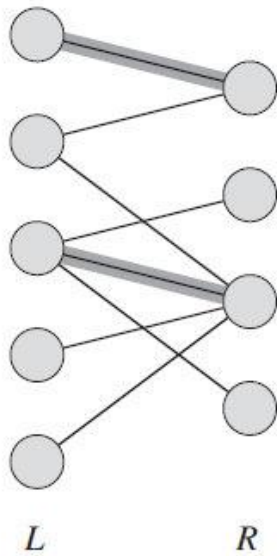
Maximum bipartite matching

Este problema puede reducirse a un problema de flujo máximo.

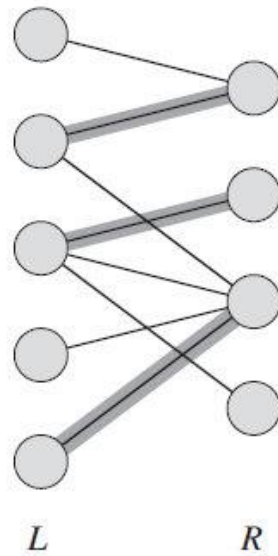
Para ello, construimos una red de flujo donde se agregan 2 nodos: un source s y un sink t .

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*.

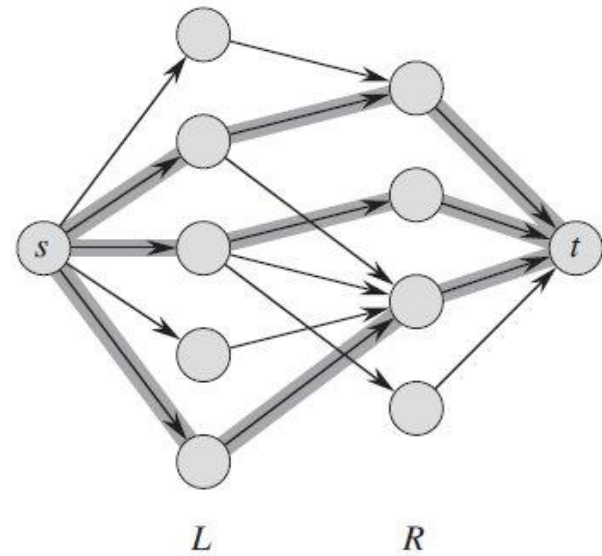
Maximum bipartite matching



(a)



(b)



(c)

Problemas

Rook Attack - Topcoder. TCO Semifinal R4

<http://goo.gl/Aoe4gW>

Dado un tablero de ajedrez de $m \times n$ ($1 \leq m, n \leq 300$) donde algunas casillas fueron eliminadas (o prohibidas). Determinar cuál es la máxima cantidad de torres que se pueden colocar de manera que no existan 2 que se amenacen.

Problemas

Stock Charts - Google Code Jam R2 (2009)

goo.gl/lpVuZN

Dados varios charts, representados como polilíneas. Dos charts se pueden combinar en una sola hoja si no se intersectan. Determinar la mínima cantidad de hojas necesarias para dibujar todos los charts.

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

Código fuente

Para los problemas dados pueden utilizar el código fuente en el siguiente repositorio:

<https://github.com/nicalvarez/flujo>

Hay implementaciones de:

- **Bipartite Matching (DFS)**
- **Edmonds-Karp**
- **Dinic**
- **Preflow-push**

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?