

# Aux # 10

2015年11月23日 (月)

Algoritmos aleatorizados → tira monedas  
y probabilíticos → hay P asociado al tiempo / correctitud.

Algoritmos tipo:

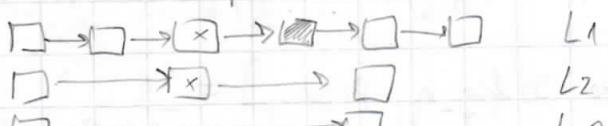
- MonteCarlo: se detienen siempre → si el algoritmo tiene 2 pero pueden equivocarse respuestas (si / no)
  - one-sided error
- Las Vegas - siempre es correcto pero pueden no detenerse
  - two-sided error



Skiplists: "aplicar búsqueda binaria a lista enlazada"



- idea 0: ordenar
- queremos los "saltos" de BB
- idea: metro expreso



$$L_1, |L_1| = l_1 = n$$

$$L_2, |L_2| = l_2$$

$$L_3, |L_3| = l_3$$

:

Si hay 2 niveles, buscar cresta:

$$\leq l_2 + \frac{l_1}{l_2} = l_2 + \frac{n}{l_2} \Rightarrow \text{minimizo c/r a } l_2 \\ (\text{¿Cuántos saltos pongo en } l_2?)$$

$$\Rightarrow C_{\min} = 2\sqrt{n}$$

Si tenemos cualquier cantidad de niveles, lo ideal sería simular un árbol binario  $\Rightarrow$  links de largo 1, 2, 4, ... ~~longitud~~

¿Cómo logro que se vea así si hay inserciones?

Idea desde árboles平衡: cada piso tiene ~~2^k+1~~ nodos  
 $\frac{1}{2}$  de los nodos del piso de abajo

$\Rightarrow$  insertar(x)

- Buscar en  $L_1$  la posición que le corresponde a x
- Tiro una moneda hasta que salga sello
- Si tire la moneda  $K$  veces, x estará en las listas  $L_1, L_2, \dots, L_K$   
 $P[\text{un elem. esté en la lista } K] = \frac{1}{2^{K-1}} \rightarrow$  siempre tiro la moneda 1 vez.

Hay que analizar:

- Tamaño de la estructura
- Tiempo de ejecución

1) Con alta probabilidad, si hay  $n$  elementos hay  $\Theta(\log n)$  pisos.

¿Por qué?

$$P[x \text{ esté en más de } c \cdot \log_2 n \text{ pisos}] = \frac{1}{2^{c \log_2 n}}$$

$$= \frac{1}{n^c}, \text{ pero esto es para 1 } x!$$

Queremos acotar la  $P.$  de que ~~c log n~~ sea más alto que  $c \log_2 m$

$$\begin{aligned} P[\text{alguno de } x_1, \dots, x_n] &= P[p(x_1) \cup p(x_2) \cup \dots \cup p(x_n)] \\ &\leq \sum_i P[p(x_i)] = \frac{n}{n^c} = \frac{1}{n^{c-1}} \end{aligned}$$

el número de pisos es  $> c \cdot \log_2 n$  con prob  $\leq \frac{1}{n^{c-1}}$

Otra forma es ver la esperanza.

2) ¿cuántos nodos tiene la estructura? (en promedio)

• Nos gustaría saber  $E(l_i)$

$$E(l_i) = \sum_{x=0}^{\infty} x \cdot P[l_i = x] = ?$$

VARS INDICADORAS: v.a con valor 0 ó 1

Definimos  $X_{ij} = \text{"el elemento } x_i \in L_j"$

$$\Rightarrow E(l_i) = E\left(\sum_{j=1}^{\infty} X_{ij}\right)$$

$$= E\left(\sum_i^n \frac{1}{2^{i-1}}\right) = \frac{n}{2^{i-1}} = ?$$

$$E(n^{\circ} \text{ nodos}) = E\left(\sum_{j=1}^{\infty} l_j\right)$$

$$= \sum_{j=1}^{\infty} \frac{n}{2^{j-1}} = \sum_{i=0}^{\infty} \frac{n}{2^i} = 2n$$

$$E(n^{\circ} \text{ de pisos}) = ? \quad H_i = \begin{cases} 0 & \text{si } L_i \text{ vacía} \\ 1 & \text{si no} \end{cases}$$

$$\Rightarrow E(n^{\circ} \text{ de pisos}) = E\left(\sum_i H_i\right)$$

$$\bullet H_i \leq l_i \Rightarrow E(H_i) \leq \frac{n}{2^{i-1}}, \quad H_i \leq 1$$

$$E\left(\sum_i H_i\right) = \sum_{i=1}^{\log n} H_i + \sum_{i=\log n+1}^{\infty} H_i$$

$$\leq \sum_{i=1}^{\log n} 1 + \sum_{i=\log n+1}^{\infty} \frac{n}{2^{i-1}} \leq \log n + \sum_{i=\log n+1}^{\infty} \frac{n}{2^k} \cdot \frac{1}{2^i} = \log n + \sum_{i=0}^{\infty} \left(\frac{n}{2^k}\right) \cdot \frac{1}{2^i} \geq \log n + 1$$

$$= \log n + \sum_{i=0}^{\infty} \frac{1}{2^i} = \log n + 2$$

\* Para ver el costo de la búsqueda, conviene ver el camino al revés



Partimos en  $L_1$  y nos movemos a la izq.

→ A penas encuentra un nodo que permite bajar, lo sigo  
¿Cuándo pasa esto? Cuando la moneda correspondiente  
salio cara.

La idea del camino inverso es llegar al último piso

⇒ "en cuántas monedas saco h caras?"

$$\Rightarrow E[S] = E[h + \sum_{r=0}^{\infty} S_r] = E(h) + \sum_{r=0}^{\infty} E[S_r]$$

cuantos monedas antes de la 1<sup>a</sup> cara

$S_r$  es tal que  $E(S_r) \leq 1$  y  $S_r \leq \frac{n}{2^r}$  (largo esperado de

$$\Rightarrow E(S) \leq \sum_{i=0}^{\lfloor \log_2 n \rfloor} 1 + \sum_{i=\lfloor \log_2 n \rfloor + 1}^{\infty} \frac{n}{2^{i-1}} + E(h)$$

la lista r.

$$= E(h) + \log n + 3$$

$$= 2 \log n + 5.$$

P2

$$A \ B \ C \quad AB = ?$$

$\Rightarrow$  Naire: mult y comparar  $\rightarrow O(n^3)$

- Elegir al azar un vector  $r$
  - Calcular  $A \cdot (B \cdot r)$  y comparar.
- $C \cdot r$
- Si  $A(Br) = Cr \Rightarrow$  respondo si }  
                  ~                no }

$O(n^2)$

- Si  $AB = C \Rightarrow A(Br) = Cr \quad \forall r, r$  es "testigo"

El algoritmo NO se va a equivocar cuando dice "No"  
 $\Rightarrow$  el error es one-sided

El problema es tratar de estimar la "densidad" de buenos testigos  
( $r = \vec{0}$  es el mal testigo)

elegimos el mal testigo

Mostraremos que si  $AB \neq C$ ,  $P[(AB)r = Cr] \leq \frac{1}{2}$

Hay que especificar la elección de  $r$ :

$\rightarrow$  se tiene un conjunto  $S$ , con  $|S| \geq 2$

$\rightarrow$  las componentes  $r_i$  se eligen de  $S$ , con prob. i.i.d. uniforme

Sea  $D = AB - C$ , si  $D \neq 0$ , s.p.g.  $d_{11} \neq 0$  (si no, permuto columnas y filas)

Si  $Dr = 0 \Rightarrow (Dr) = \sum_{i=1}^n d_{1i} \cdot r_i = 0$

$$\Rightarrow r_1 = -\frac{1}{d_{11}} \left( \sum_{i=2}^n d_{1i} \cdot r_i \right)$$

único dist.  
i.i.d.

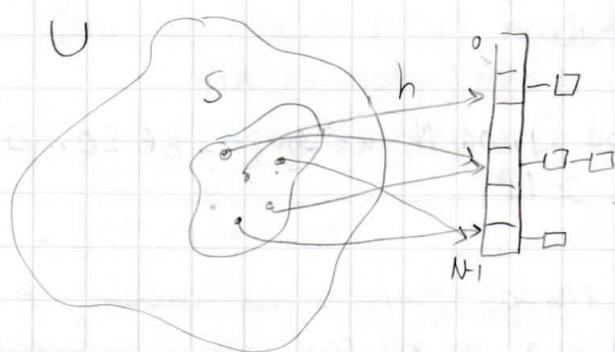
- Para cada valor de  $r_2 \dots r_n$  hay sólo 1 valor posible para  $r_1$  (si fijo  $r_2 \dots r_n$ , hay un solo valor para  $r_1$  para que sea mal testigo)
- Como  $r_1$  se elige de forma uniforme, dado indep.

$$P[r_1 \text{ sea mal testigo}] = P[r_1 = r_1^* \mid r_2 \dots r_n = \dots] \\ = \frac{1}{|S|} \leq \frac{1}{Z}$$

Si repito  $t$  veces, la prob. de equivocarse es  $\frac{1}{Z^t}$   
 si alguna vez dice no  $\Rightarrow$  no  
 $\sim \Rightarrow$  sí

# Hashing Universal y Perfecto

2015年11月24日 (X)



S conjunto de elementos  
N tamaño de la tabla  
 $\frac{|S|}{N}$

Def: Una familia  $H$  de funciones de hashing es 2-universal si:  

$$\forall x \neq y, \Pr_{h \in H} (h(x) = h(y)) \leq \frac{1}{N}$$
 "para todo par de ellos"

Def: Variable indicadora

3-universal es más fuerte,  
sería para todo triple.

$$C_{xy} = \begin{cases} 1 & \text{si } h(x) = h(y) \\ 0 & \text{no} \end{cases}$$

$$C_{xs} = |\{y \in S, C_{xy} = 1\}|$$

OBS:  $C_{xs}$  es el costo de buscar la clave  $x$ , o insertarla

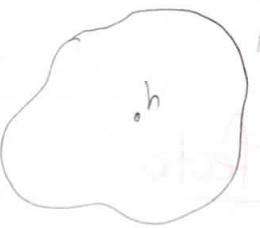
$$C_{xs} = \sum_{y \in S} C_{xy} = 1 + \sum_{\substack{y \in S \\ y \neq x}} C_{xy}$$

∴ Usando  
 $N = \Theta(|S|)$   
 calculas, el  
 costo esperado  
 es  $\Theta(1)$

$$E(C_{xs}) = 1 + \sum_{\substack{y \in S \\ y \neq x}} E(C_{xy})$$

$$= 1 + \sum_{\substack{y \in S \\ y \neq x}} \Pr_{h \in H} (h(x) = h(y)) < 1 + \frac{|S|}{N}$$

2-universal



H

La prob de que  $h$  aleatoria,  
para 2 ( $x$  e  $y$ ) de mi conjunto fijo  
de claves, cualesquiera,  $h$  se  
porta bien con alta prob.

Veamos algunas familias 2-universal

$$H_p = \{ h_{ab}(x) = ((ax + b) \bmod p) \bmod N, a \in [1..p-1], b \in [0..p] \}$$

dónde  $p$  es un primo  $> N$ .

para cada  $a$  y  $b$  en sus resp. rangos tenemos una función  $h$ ,  
Al crear la estructura, elegimos  $a$  y  $b$  con prob. uniforme.  
(el mod  $N$  es para que queda en la tabla)  
Debo hacerlo aleatorio para poder hablar de probabilidades. Dentro  
de  $H$  puede haber una  $h$  muy mala, pero con cierta probabilidad  
Lo bacán de 2-universal es que hay alta densidad de funciones  
que se portan bien, que tienen pocas colisiones entre  $x$  e  $y$ .

Teorema:  $H_p$  es 2-universal.

Dem: Fijemos  $r \neq s$  y calculemos, para  $x \neq y$

$$\Pr(ax+b=r \bmod p \wedge ay+b=s \bmod p)$$

Si eso ocurre  $ax+b=r \bmod p$

$$\underline{ay+b=s}$$

$$\frac{a(x-y)}{a(x-y)} = r-s \bmod p$$

$$a = \frac{(r-s)}{(x-y)} \bmod p$$

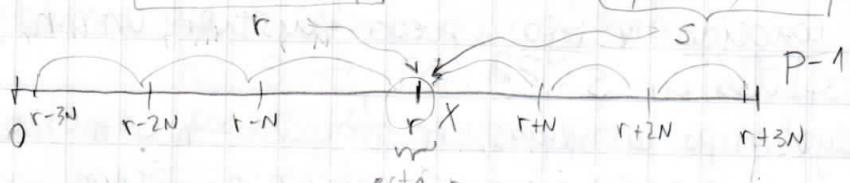
y además  $b = r - ax (= s - ay)$

$\therefore$  hay exactamente un valor de  $a$ , y un valor de  $b$  que cumplen eso

$$\Pr = \frac{1}{p(p-1)}$$

Para que  $h(x) = h(y)$  necesitamos que

$$(ax + b) \bmod p \bmod N = (ay + b) \bmod p \bmod N$$



este r  
no se cuenta  
pues no se puede dar  $r=s$   
(si no  $a=0$ )

Hay a lo más  $\lceil p/N \rceil - 1$  valores de  $s$  que colisionan con ese valor de  $r$ ,  $r \neq s$

→ tenemos  $p$  opciones para  $r$

→ para cada  $r$  tenemos  $\lceil p/N \rceil - 1$  opciones para  $s$ .

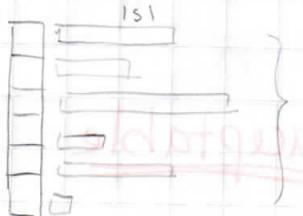
→ la prob. de que se dé cada par  $(r,s)$  específica es  $\frac{1}{p(p-1)}$

$$\text{Entonces, } \Pr(h(x) = h(y)) \leq \frac{1}{p(p-1)} \cdot \underbrace{p(\lceil p/N \rceil - 1)}_{< p(p-1)} < \frac{1}{N}$$

$$\begin{aligned} \lceil p/N \rceil - 1 &= \left\lfloor \frac{p+N-1}{N} \right\rfloor - 1 \\ &\leq \frac{p+N-1}{N} - 1 \\ &= \frac{p-1}{N} \end{aligned}$$

\* hay que escoger algún primo más grande que  $N$ .

Si  $U$  es más grande que  $|S| \cdot N$  siempre hay un  $h$  que hace caer muchos en una misma celda.



siempre hay una celda  
con  $|S|$  o más elementos  
colisionados

## Hashing perfecto

Dado un  $S$  conocido y fijo, puedo construir un  $h$  que no genere colisiones en  $S$ ??

Puedo hacer algo tipo las Vegas, ir probando  $h \in H$  hasta encontrar una función que no produzca colisiones. Pero qué pasa si  $H$  es muy grande?

Elijo  $h \in H$  2-universal

$$\Pr(h(x) = h(y)) \leq \frac{1}{N}$$

$$\Pr(\exists x \neq y, h(x) = h(y)) \leq \sum_{x \neq y \in S} \Pr(h(x) = h(y))$$

$$= |S|(|S|-1) \cdot \frac{1}{N}$$

cota para elegir  
 $h$  no perfecta

∴ si elijo  $N = |S|^2$  (tabla grande)  
entonces  $\Pr(h \text{ no es perfecta}) < \frac{1}{2}$

Luego es poco probable

que un algoritmo Las Vegas demore mucho en probar y encontrar

$h$  que no produzca colisiones.

(El trabajo de probar cada  $h$  toma  $\Theta(|S|)$ , e inicializar  
puede hacerse en  $\Theta(1)$ , si tabla es grande y uso paos ellos)

⇒ es un alg. Las Vegas con tiempo esperado  $\Theta(|S|)$

↓  
número "cte" → proba  
de intentos

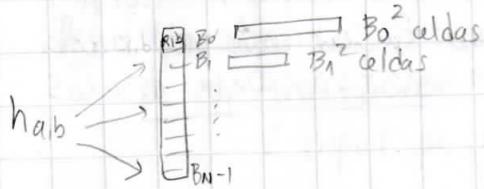
Pero  $N = |S|^2$  es inaceptable

Queremos tiempo de construcción lineal y espacio lineal  $\mathcal{O}(|S|)$

Probamos una función distribuidora.

$$h: S \rightarrow [0..N-1] \quad N = |S|$$

Contamos cuántos elementos pone en cada celda,  $B_i^2$



Elegí una  $h \neq$  para cada celda.

a cada celda le construyo otra tabla de tamaño cuadrático para buscar rápidamente ahí con alta probabilidad de no colisión. Los  $B_i^2$  no son muy grandes pues ya está distribuidos

$$\text{Cuánto es el valor esperado de } \sum_{i=0}^{N-1} B_i^2 = \sum_{x,y} C_{xy}$$

$\rightarrow x, y, z \quad B^2 = 9 \quad x, y, z \text{ colisionan}$

$$\begin{array}{lll} C_{xx} = 1 & C_{yx} = 1 & C_{zx} = 1 \\ C_{xy} = 1 & C_{yy} = 1 & C_{zy} = 1 \\ C_{xz} = 1 & C_{yz} = 1 & C_{zz} = 1 \end{array}$$

$$\sum_{i=0}^{N-1} B_i^2 = \sum_{x,y} C_{xy} = N + \sum_{x \neq y} C_{xy}$$

$$\begin{aligned} E\left(\sum_{i=0}^{N-1} B_i^2\right) &= N + \sum_{x \neq y} E(C_{xy}) \\ &\leq N + \overbrace{N(N-1)}^{\text{cantidad de pares } (x,y)} \frac{1}{N} \end{aligned}$$

$$< 2N$$

(12) Puede ser que haya una función que tire todos los eltos a una misma celda  $i$ , luego  $B_i^2 = N^2$  y  $B_j^2 = 0$   
 $\forall j \neq i$ .

Pero si es un conjunto  $\Sigma$  universal, la esperanza de la suma de  $B_i^2$  es lineal. Pero no es suficiente% fuerte  
Puedo que hay una función muy muy buena  
y el resto son relativamente malas.



todas las buenas deben producir  $\sum B_i^2 < 4N$ .  
porque si  $\exists$  buena  $\geq 4N \Rightarrow$  todas las malas  $\geq 4N$   
 $\Rightarrow E(\cdot) \geq 2N$

Si hay una buena que es mala, luego todas las malas son peores.  
(y al más hay  $\frac{1}{2}$  buenas,  $\frac{1}{2}$  malas.)

Entonces debo usar Las Vegas para encontrar distribuidora que produzca  $\sum B_i^2 < 4N$

(Recordar que en Las Vegas estoy escogiendo  $h$  al azar con repetición,  
por eso pude demorar  $\infty$ )

Encontrar  $h$  distribuidora toma  $\Theta(|S|)$  y pasar celda por celda asignando las otras  $h$  es  $\Theta\left(\frac{\sum B_i^2}{4N}\right) = \Theta(|S|)$   
(pues  $N = |S|$ )

Una vez que la estructura se construyó, es determinista.

2015年11月26日(木)

# ALGORITMOS APROXIMADOS

Problemas de decisión NP-completos (clique tamaño K)

☰ Problemas de optimización también difíciles. (máx clique?)  
(minimizar/maximizar)

Todo probl. de opt., tiene un probl. de decisión asociado. Podría usar uno para responder el otro.

Def: Un algoritmo es un  $\rho(n)$ -aproximación a un problema de optimización si,

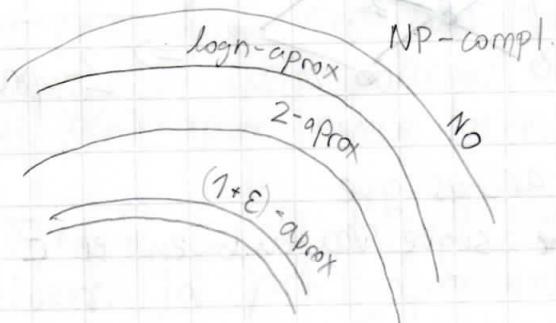
• input de tamaño  $n$ ,

$$\max \left( \frac{C}{C^*}, \frac{C^*}{C} \right) \leq \rho(n)$$

alg de  
minimizac  
> 1      alg de  
máx

donde  $C$  = valor que encuentra el algoritmo  
 $C^*$  = valor óptimo.

probl. de decisión NP-compl,  
se dejan aproximar con  
un probl. de opt. 2-aprox,  
logn-aprox,  $1+\epsilon$  aprox



más deseable

$$\mathcal{O}(n^{2/\epsilon}), \mathcal{O}\left(\frac{1}{\epsilon^2} n^3\right)$$

el costo crece polinomialmente cuando  $\epsilon$  disminuye

Def: Un esquema de aproximación polinomial es un algoritmo de aproximación que recibe un parámetro extra  $\epsilon$  y produce una  $(1+\epsilon)$ -aproximación.  
Su costo es una función de  $n$  y de  $\epsilon$ , polinomial en  $n$  para todo  $\epsilon$  fijo

Si la función también es un polinomio en  $1/\epsilon$ , se llama esquema de aproximación completamente polinomial.

Nota: los problemas de decisión se dejan "trasladar" de uno NP a otro NP.  
 Pero los prob. de optimización quizás no.

VERTEX COVER: (prob. de decisión: existe vertex cover de tamaño  $k$ ?)

Dado un grafo  $G(V, E)$  elegir un subconjunto mínimo  $V' \subseteq V$  que cubra todas las aristas.

Vamos a hacer una 2-aprox. muy simple.

VC:

$$V' \leftarrow \emptyset$$

mientras  $E \neq \emptyset$

elegir  $(u, v) \in E$

$$V' \leftarrow V' \cup \{u, v\}$$

sacar de  $E$  toda arista incidente en  $u \circ v$ .

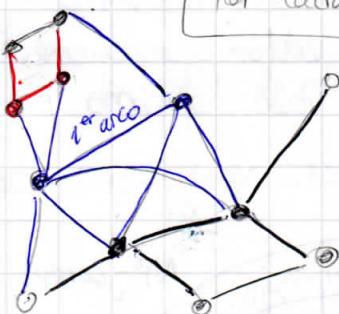
puedo usar esta  
aproximación para resolver  
(de manera aprox también)  
el problema de decisión.

→ incluyo ambos nodos, en vez  
de elegir uno solo.

Por cada par de nodos

que meto, el  
conjunto óptimo  
tiene al menos  
1 de esos 2.

⇒ 2-aprox



Es una 2-aprox pues cada vez que mete los nodos  $u$  y  $v$  en  $V'$ , un VC óptimo debe contener  $u \circ v$  (o ambos).

Al eliminar las aristas que  $u \circ v$  cubren, el invarianta sigue valiendo en el  $E$  resultante.

Recordemos que:  $VC \ k \Leftrightarrow$  clique de  $n-k$  en  $\bar{G}$

ej: 100 nodos ( $n$ )

clique de tam 10 →  $\bar{G}$  VC de tam 90

clique de  $\emptyset$  ← la aprox encuentra  $\leq 180 / 100$

X

No puedo reducir algoritmos de opt. aproximados.

## Camino Hamiltoniano

Dado  $G(V, E)$ , existe un circuito que pase por cada nodo exactamente una vez? Sí? No?

Problema del viajante de comercio (prob. de opt) ("vendedor viajero")

Además las aristas tienen un costo, y quiero un circuito que minimice la suma de los costos.

(Supondremos que existe camino Hamiltoniano)

Probaremos que este problema NO se deja aproximar.

Supongamos que existe una  $p(n)$ -aproximación. La uso para resolver un problema de circ. Hamiltoniano en  $G(V, E)$

Creo un grafo completo  $G'(V, V \times V)$

con costo  $C(u, v) = \begin{cases} 1 & \text{si } (u, v) \in E \\ |V|p(|V|) + 1 & \text{si } (u, v) \notin E. \end{cases}$

La idea es que si tengo una  $p(n)$ -aprox cualquiera, puedo responder el problema de circ. hamiltoniano SIEMPRE en tiempo polinomial escogiendo costos correctos. (Aristas muy costosas, de manera que el  $p(n)$ -aprox nunca los escoge) → que no están en  $G$ , y si en  $G'$

Si existe un circuito en  $G$ , el costo del mejor camino en  $G'$  es  $|V|$

∴ la  $p(n)$ -aproximación me encontrará un camino de costo  $\leq |V|p(|V|)$

Si no existe un circuito Hamiltoniano en  $G$ , entonces toda solución en  $G'$  necesita usar al menos una arista que NO está en  $G$ , la cual tiene costo  $|V|p(|V|) + 1$

∴ el costo total es siempre, usando cualquier solución aprox o no,  $> |V|p(|V|)$ .

Ahora ejecuto solv $\phi$  aprox, veo si es  $> \leq |V|p(|V|)$  y siempre puedo resolver prob. de camino Hamiltoniano.



Perna

## Desigualdad Triangular

$$C(u, v) \leq C(u, w) + C(w, v)$$

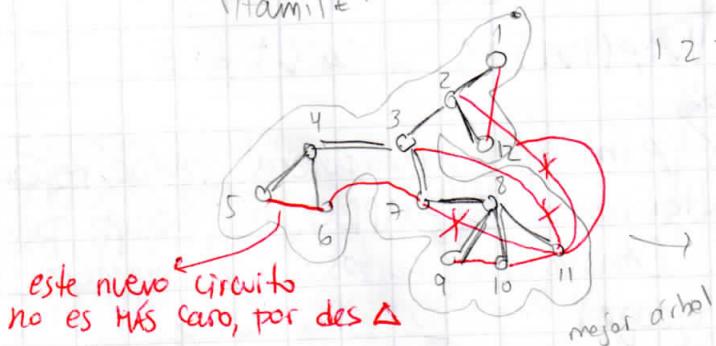
Si los costos del problema del viajante de comercio satisfacen la desigualdad triangular, con  $E = V \times V$ , entonces existe una **2-aprox.** (y una  $\frac{3}{2}$ , Christofides)

Un circuito es un camino + 1 arista.

$$C(\text{circ.}_{\text{Hamilt}}) \geq C(\text{camino Hamilt}) \geq C(\text{MST})$$

mín spanning tree

1 2 3 4 5 6 7  
8 9 10 11 12 13  
14 15 16 17 18 19



→ no hay problema pues existen todas las aristas.

$$C'(\text{Euler}) \leq C(\text{Euler}) = 2C(\text{MST}) \leq 2C(\text{circ.}_{\text{H.}})$$

↳ pasa 2 veces por cada arista de MST

2-aprox:

- 1) Construir un MST de G
- 2) Hago un circuito Euleriano del MST.

## Vertex Cover con costos

Cada nodo pesa  $w(v)$ . Quiero minimizar la suma de los pesos.

$$\text{Sea } x(v) = \begin{cases} 1 & \text{si elijo } v \\ 0 & \sim \end{cases}$$

Quiero encontrar  $x(\cdot)$  tal que

$$\forall v \quad x(v) \geq 0$$

$$x(v) \leq 1$$

$$\forall u, v \in E, \quad x(u) + x(v) \geq 1$$

Mínimizar  $\sum_{v \in V} x(v) w(v)$

Es un problema de programación lineal que se resuelve en tiempo polinomial. El óptimo lineal,  $C_L$ , es  $C_L \leq C^*$

De esa solución  $x(v) \in [0, 1]$ , obtenemos la de VC como:

elegir  $v$  sii  $x(v) \geq 0.5$ .

- 1) es  $v$  en VC? si, pues  $(u, v) \in E \Rightarrow x(u) + x(v) \geq 1$   
 $\Rightarrow x(u) \geq 0.5$  ó  $x(v) \geq 0.5$   
 $\Rightarrow$  elijo  $u$  o  $v$ .

2) qué aproximación resulta?

$$\sum_{v \in V'} w(v) = \sum_{v \in V'} y(v) w(v)$$

$$y(v) = \begin{cases} 1 & \text{si } v \in V' \\ 0 & \sim \end{cases}$$

$$\text{si } v \in V', \quad y(v) = 0 \leq x(v)$$

$$\text{si } v \notin V', \quad y(v) = 1 \leq 2 \cdot x(v) \quad x(v) \geq 0.5 \text{ porque lo elegí.}$$

∴ La suma de los pesos es

$$\sum_{v \in V'} w(v) \leq 2 \cdot C_L \leq 2 \cdot C^*$$