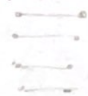
Matchings

→ (definicion): Un matching es un grafo G con un subgrafo M con $d_M(x) = 1$ $\forall x \in V(M)$



• problema a trabajar: Dado G, hallar un matching en G con la mayor cant. de lados posibles. Lo reduciremos a un problema de flujo maximal.

Estudiaremos el problema para grafos bipartitos.

Tenemos que transformar un grafo bipartito G con partes $\overline{X}$ e $\overline{Y}$ en un network.

- vertices del network: $\{s,t\} \cup \overline{X} \cup \overline{Y}$
- lados: $\{\overrightarrow{xy} : x \in \overline{X}, y \in \overline{Y}, xy \in E\} \cup \{\overrightarrow{sx} : x \in \overline{X}\} \cup \{\overrightarrow{yt} : y \in \overline{Y}\}$
- capasidades: 1 para todos los vertices

→ (propiedad): Flujos maximales en este network, corresponden con matching maximal en G.

$|\overline{X}| = |\overline{Y}|$

→ (definicion): si $W \subseteq V$ entonces: $\Gamma(W) = \{z : \exists w \in W : \overrightarrow{zw} \in E\} = \bigcup_{w \in W} \Gamma(w)$

→ (definicion): un matching es perfecto si $V_M = V(G)$ use todos los vertices de G

→ (definicion): un matching es completo si $V_M \cap \overline{X} = \overline{X}$ use todos los vertices de $\overline{X}$ o de $\overline{Y}$

→ (Teorema de Hall): si $G = (\overline{X} \cup \overline{Y}, E)$ es bipartito con partes $\overline{X}$ e $\overline{Y}$ entonces existe un matching completo de $\overline{X}$ a $\overline{Y}$ si y solo si

$$|S| \leq |\Gamma(S)| \quad \forall S \subseteq \overline{X}$$

→ (Teorema del matrimonio): todo grafo bipartito regular tiene un matching perfecto. König

→ (corolario): Si G es bipartito $\Rightarrow \chi'(G) = \Delta$, donde χ' es el índice cromático, es decir la menor cantidad de colores necesarios para colorear los lados de un grafo de forma tal que los lados con vértices en común tengan colores distintos.

→ (propiedad): $\chi'(G) \geq \Delta$

→ (lema): G bipartito $\Rightarrow \exists\, H$ bipartito regular tq $G \subseteq H$, $\Delta(G) = \Delta(H)$

máximo

Grafo bipartito con pesos

- Vamos a tener dos criterios para elegir los matchings:
  - (binario) minimizar el costo (I)
  - (húngaro) minimizar la suma de los costos. (II)
- Asumimos de ahora en más:
  - $|X| = |Y|$
  - $\exists$ al menos 1 matching perfecto

→ (lema): Sea A una matriz de pesos $(n \times n)$, sea $\bar{A}$ la matriz que se obtiene de restar una constante a cada entrada de una sola fila o columna de A. Entonces un matching minimiza la suma relativa de A si y solo si la minimiza a la suma relativa a $\bar{A}$.

Sirve para demo de complejidad

→ (corolario): puede haber a lo sumo $O(n)$ "cambios de matriz" antes de extender el matching en un lado, pues S puede crecer a lo sumo $O(n)$ veces

→ (teorema): la complejidad del húngaro es $O(n^4)$ (de la forma que lo hizo el profe)

→ (teorema): el húngaro se puede codificar en $O(n^3)$

en ambos alg. puedo usar el teorema de Hall,

$S \rightarrow$ filas marcadas

$\Gamma(S) \rightarrow$ col marcadas

A1

B2

C3 → (I) minimiza el costo mayor.

total → húngaro minimiza este

ejemplo algoritmo (I) : minimiza el costo

		1	2	3	4	5	6	7	8
	A	1	2	5	5	3	8	2	9
	B	9	8	8	9	8	8	1	3
	C	3	1	5	8	9	6	5	8
A =	D	9	1	7	9	3	8	8	5
	E	8	9	2	4	8	5	9	9
	F	9	8	3	9	8	9	8	1
	G	5	4	8	9	1	8	9	8
	H	8	8	9	1	3	3	1	1

(1) hacer una lista de los n° dentro de la matriz, para hacer busqueda binaria.
la idea es empezar con el n° del medio
i) si hay matching buscar si hay uno menor (muevo izq)
ii) si no hay ver si hay con uno mayor (muevo der)

n° a ver
$\left[\frac{9}{2}\right] = \frac{4}{2}$

1 2 3 4 5 6 7 8 9

(2) me creo una nueva matriz B de la sig forma

$$B_{ij} = \begin{cases} 0 & \text{si } A_{ij} > 4 \\ 1 & \text{si } A_{ij} \leq 4 \end{cases}$$

en realidad es de la variable elegida en binary search

(1)

	1	2	3	4	5	6	7	8	
A	1	1	0	0	1	0	1	0	1
B	0	0	0	0	0	0	1	1	7
C	1	1	0	0	0	0	0	0	2
D	0	1	0	0	1	0	0	0	5
E	0	0	1	1	0	0	0	0	3
F	0	0	1	0	0	0	0	1	8
G	0	1	0	0	1	0	0	0	5
H	0	0	0	1	0	1	1	1	4
	C	G	F	E	G		A	B	

pude encontrar un matching ✓ :)
veamos si hay uno menor. ⟶ 1 2 3 4 (binary search)

repito desde paso (2)

(2)

	1	2	3	4	5	6	7	8	
A	1	1	0	0	0	0	1	0	
B	0	0	0	0	0	0	1	0	
C	0	1	0	0	0	0	0	0	
D	0	1	0	0	0	0	0	0	S
E	0	0	1	0	0	0	0	0	
F	0	0	0	0	0	0	0	1	
G	0	0	0	0	1	0	0	0	
H	0	0	0	1	0	0	1	1	

es facil ver que no puede haber un matching pues la fila D solo tiene 1 uno y no lo puedo usar por C que tambien tiene un solo uno.
Solo nos queda ver en (3) si hay o no matching repitiendo los pasos desde (2).
Luego de hacer todo vemos que no hay matching. por lo cual, el primero que encontramos minimiza el costo. Basta dar el matching con los pesos de A

A7 : 2
B8 : 3
C1 : 3
D5 : 3
E4 : 4
F3 : 3
G2 : 4
H6 : 3

$\boxed{\text{Suma} = 25}$

(3) dar un matching inicial, itero fila tras fila buscando el primer 1 que salga de izq a der y que no haya sido marcado otro 1 en la misma col.
A las que no puedo marcar, etiquetar con una S.

(4) etiquetar filas y col de la sig forma:

↑ 1s marcados ← 1s marcados o no

y:
- si estoy viendo una fila etiqueto la col con la letra de la fila
- si estoy viendo una col etiqueto la fila con el n° de la col.

por ultimo tacho las etiquetas que ya use.
(si ya tenia etiqueta no le pongo una nueva)

(5) itero en (4) hasta llegar a la col no marcada o darme cuenta que no puedo hacer un matching.

(6) usar las etiquetas de la der (filas) para desmarcar y la de abajo (col) para posicionarte.

ejemplo algoritmo II : minimiza la suma de los costos

↳ pensar en perlie [illegible]

$$A = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline A & \infty & 5 & 9 & 7 \\ B & 9 & 5 & 5 & 8 \\ C & 7 & 5 & 9 & 5 \\ D & \infty & 5 & 4 & 7 \end{array} \xrightarrow{(1)} \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline A & \infty & 0 & 4 & 2 \\ B & 4 & 0 & 0 & 3 \\ C & 2 & 0 & 4 & 0 \\ D & \infty & 1 & 0 & 3 \end{array} \xrightarrow{(2)} [illegible]$$

no puedo extender

(1) restar el minimo a cada fila
(2) restar el minimo a cada col
(3) buscar un matching de ceros
(4) tratar de extenderlo igual que el algo anterior ← os ↑ os marcados

$S = \{A, B, D\}$ $\Gamma(S) = \{2, 3\}$ $|S| = 3 > 2 = |\Gamma(S)|$ no hay matching por hall.

compruebo

opuesto

(5) tomamos $m = \min(S \times \overline{\Gamma(S)}) = 2$

(6) tacho : • las col ~~si~~ con etiqueta { tacho $\overline{S} \times \Gamma(S)$
• las filas sin etiquetas

(7) creo una nueva matriz de la sig forma:

$$\begin{array}{c|cccc|c} & 1 & 2 & 3 & 4 & \\ \hline A & \infty & \boxed{0} & 4 & \boxed{0} & \cancel{2} \\ B & 0 & \boxed{0} & \boxed{0} & 1 & \cancel{3} \\ C & \boxed{0} & 2 & 6 & 0 & \\ D & \infty & 1 & \boxed{0} & 1 & 8 \\ \hline & & B & D & A & \end{array}$$

• ~~k~~ si esta tachado 1 vez dejo el mismo numero
• si no esta tachado le resto m
• si esta tachado dos veces le sumo m

(y mantengo las etiquetas)

(8) trato de extender el matching

puede extender el matching :)

(9) dar el matching con los valores de la matriz A

A4	7
B2	5
C1	7
D3	4
	23

(10) si despues de hacer (8) no puedo extender el matching entonces tengo que repetir desde paso (5).

Codigo de correxion de errores

→ (definicion): Un codigo de bloque binario es un subconjunto de $\{0,1\}^n$ para algun n fijo. ↳ misma longitud

Una suposicion es que el medio de transmicion puede cambiar bits pero no puede eliminarlos o añadirlos. Tambien suponemos que la probabilidad de cambiar de 0 a 1 es la misma de cambiar de 1 a 0, y es para todos los bits independientemente. Si esta probabilidad es p, suponemos $0 < p < \frac{1}{2}$. La prob. de t errores es p^t.

→ (definicion): la distancia de hamming entre 2 palabras $x, y \in \{0,1\}^n$, es el numero de bits de diferencia entre x e y.

→ (propiedad): d_H es una distancia, es decir:

A) $d_H(x,y) = d_H(y,x)$

B) $d_H(x,y) \geq 0$

C) $d_H(x,y) = 0 \iff x = y$

D) $d_H(x,y) \leq d_H(x,z) + d_H(z,y)$ Desigualdad triangular.

→ (definicion): Sea $S = S(C) = \min\{d_H(x,y) : x,y \in C \mid x \neq y\}$ minima distancia entre 2 palabras de C

→ (definicion). Un codigo C "detecta" r errores si $D_r(x) \cap C = \{x\}$ $\forall x \in C$ donde $D_r(x) = \{y : d_H(x,y) \leq r\}$

C corrige t errores si $D_t(x) \cap D_t(y) = \emptyset$ $\forall x,y \in C \mid x \neq y$

→ (Teorema): Sea C un codigo tal que R es la mayor capacidad de detectar errores de C, es decir, C detecta R errores pero no R+1. Y a la mayor capacidad de correcciones, es decir, C corrige Q errores pero no corrige Q+1. Entonces $\underset{\text{detecta}}{R} = S-1$ y $\underset{\text{corrige}}{Q} = \frac{S-1}{2}$