

→ Un grafo G es un par (V, E) donde:

- V es un conjunto (finito) → "vertices o nodos"

- $E \subseteq \{A \subseteq V : |A| = 2\}$ → "aristas o lados"

→ (notación): $\{x, y\} \rightarrow xy = yx$

→ (vecindario de x): $N(x) = \{y \in V : xy \in E\}$ (los vecinos de x)

→ (grado de valencia): el grado de valencia de un vértice x es $d(x) = |N(x)|$

$$S = S(G) = \min \{d(x) : x \in V\}$$

$$\Delta = \Delta(G) = \max \{d(x), x \in V\}$$

→ Un grafo es planar si se puede dibujar sin que ningún lado se cruce con otro.

→ (Subgrafo): Un subgrafo H de un grafo G ($H = (W, F)$) es un grafo tq $W \subseteq V$ y $F \subseteq E$.

→ Un grafo es cíclico si $V = \{1, 2, \dots, n\}$ y $E = \{12, 23, \dots, n1\}$



→ Un grafo es completo si $V = \{1, 2, \dots, n\}$, $E = \{A \subseteq V : |A| = 2\}$



→ (convención): $n = |V|$ y $m = |E|$ todas las conexiones posibles

Coloreo

→ (definición): Un coloreo con k colores de un grafo $G = (V, E)$ es cualquier función c tq $c: V \rightarrow S$ con $|S| = k$.

El coloreo es propio si $x, y \in E \Rightarrow c(x) \neq c(y)$ (ningún lado tiene distintos colores)

El número cromático $\chi(G) = \min \{k : \exists$ coloreo propio con k colores en $G\}$

(1) Una propiedad obvia es que $\chi(G) \leq n$

Para probar un coloreo propio en un grafo G .

aprovechar simetrías: recorrer todos los colores a la vez (por simetría) y luego probar que no se puede con $k-1$

(2) Probar que existe un coloreo propio (deberá ser) de k colores

(3) Demostrar que es el mínimo (por absurdo), es decir \nexists colores con $k-1$ colores

(2)

→ (propiedad): Si H es un subgrafo de $G \Rightarrow \chi(H) \leq \chi(G)$
 $\bullet \chi(K_n) = n$

→ (corolario): Si G tiene un KR $\Rightarrow r \leq \chi(G)$

Algoritmo de greedy para coloreo

→ (greedy para coloreo): repite un orden de los vértices x_1, \dots, x_n

$$c(x_1) = 0$$

para $i \geq 2$ $\underbrace{\min_{K \geq 0} \text{mínimo color posible}}_{\substack{\text{Vemos adyacentes}}} \quad |$

$$c(x_i) = \min \{ K \geq 0; c(x_i) \neq K : \forall j < i : x_i \in N(x_j) \}$$

Invariante: los colores parciales son propios y por ende el final lo es.

→ (complejidad de greedy): $\boxed{O(d(x_i))} = O(\boxed{d(x_i)}) = O(z_n) = O(n)$

Colorear $\begin{cases} \text{ciclos pares} & \text{necesito 2 colores} \\ \text{ciclos impares} & \text{necesito 3 colores} \end{cases}$ T. spetrum de manos

→ (camino): Un camino entre dos vértices x e y es una sucesión de vértices

x_1, \dots, x_n tq $x_1 = x, x_r = y, x_i x_{i+1} \in E \quad \forall i \in \{1, 2, \dots, r-1\}$ 

→ Un grafo es conexo si tiene una sola componente conexa puede llegar de cualquier ptó a cualquier ptó mediante un camino.

→ (Teorema): $\chi(G) \leq \Delta + 1$ | $\Delta(K_n) = n-1$

→ (Teorema de Brooks): Si G es conexo $\Rightarrow \chi(G) \leq \Delta$, al menos pue G sea un ciclo impar o un grafo completo.

→ (propiedad): Si G es conexo, entonces existe un ordenamiento de los vértices tq greedy colorea a todos los vértices, salvo a uno, con Δ colores o menos.

Líquidamente decir que greedy con un mal ordenamiento puede dar mal resultado

(3)

→ (Teorema VIT): Sea $G = (V, E)$ un grafo cuyos vértices están coloreados con un colorido propio C con r colores $\{0, \dots, r-1\}$. Sea π una permutación de los números $0, \dots, r-1$. Sea $V_i = \{x \in V : C(x) = i\}$ $i=0, \dots, r-1$.

Ordenamos los vértices poniendo primero los vértices de $V_{\pi(0)}$, luego $V_{\pi(1)}$, hasta $V_{\pi(r-1)}$. Entonces greedy en ese orden colorea a G con r colores.

→ (corolario): Existe un ordenamiento de los vértices de G tq greedy colorea G con $\chi(G)$ colores.

→ (grafo bipartito): Se dice que un grafo es bipartito si $\chi(G) = 2$, o sea que $\exists x, y \subseteq V$ tq: $\begin{cases} V = x \cup y \\ x \cap y = \emptyset \end{cases}$ "partir el grafo en 2"

→ (Teorema): Si G es bipartito \Rightarrow es polinomial (revisar)

Flujos y Networks

importa orden

→ (grafo dirigido): un grafo dirigido es un par (V, E) con $E \subseteq V \times V$

→ (network): un network es una tripleta (V, E, g) donde (V, E) es un grafo dirigido y $g: E \rightarrow \mathbb{R}_{>0}$

costo/capacidad

→ (notación): Si E está definido en E y $A, B \subseteq V$, entonces

$$g(A, B) = \sum_{\substack{x \in A \\ y \in B \\ (x, y) \in E}} g(x, y) \quad \left. \begin{array}{l} \text{(separada} \\ \text{salida)} \end{array} \right\} \text{suma ale m subgrafo}$$

$$\circ \text{out}_g(x) = g(\{x\}, V) = \sum_{\substack{y \in V \\ (x, y) \in E}} g(x, y) \quad \text{todo lo que sale de x por medio de } g$$

$$\circ \text{in}_g(x) = g(V, \{x\}) = \sum_{\substack{y \in V \\ (y, x) \in E}} g(y, x) \quad \text{todo lo que entra a x por medio de } g$$

$$\bullet \overrightarrow{xy} = (x, y)$$

(4)

→ (definición):

- $\Gamma^+(x) = \{y : (x, y) \in E\}$ $\text{out}_g(x) = \bigcup_{y \in \Gamma^+(x)} g(\vec{xy})$
- $\Gamma^-(x) = \{y : (y, x) \in E\}$ $\text{ing}(x) = \bigcup_{y \in \Gamma^-(x)} g(\vec{yx})$

veces hacia adelante

→ (definición): Dado un network (V, E, c) y vértices $s, t \in V$. Un flujo de s a t es una función f en los lados con:

- $s \leftarrow \rightarrow t$
- $0 \leq f(\vec{xy}) \leq c(\vec{xy}) \quad \forall \vec{xy} \in E$
 - $\inf(x) = \text{out}_f(x) \quad \forall x \neq s, t$ *todo lo que entra sale*
 - $\inf(s) = \text{out}_f(t) = 0$ *nada entra a s y nada sale de t*

→ (def): el valor de un flujo es $v(f) = \text{out}_f(s) - \inf(s) = \text{out}_f(s)$

→ (flujo maximal): Un flujo es maximal si $v(g) \leq v(f)$ *g flujo de sat*

→ (prop): $v(f) = \inf(t)$

→ puede no ser nulo pero si es mayor.

→ (camino dirigido): Un camino dirigido de x a y es una sucesión de vértices x_0, \dots, x_r con $x_0 = x$, $x_r = y$, $\overrightarrow{x_i x_{i+1}} \in E$.

Computar caminos pre y o

recibir más

(5)

→(definición): Dado un flujo f en un network, un "camino aumentante" es una sucesión de vértices x_0, \dots, x_r tq $x_0 = s, x_r = t$ y $\forall i < r$ pasa una de las dos cosas siguientes:

I) $\overrightarrow{x_i x_{i+1}} \in E \wedge f(\overrightarrow{x_i x_{i+1}}) < c(\overrightarrow{x_i x_{i+1}})$ puedes mandar más (forward)

II) $\overrightarrow{x_{i+1} x_i} \in E \wedge f(\overrightarrow{x_{i+1} x_i}) > 0$ que hay más manda al flujo para pillar (backward) devolviendo

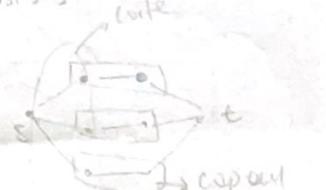
Copiar Algoritmo del FF y ious

Por eso t nunca pertenece a S

los elementos del corte son los vértices alcanzados por la última it.
la capacidad de un corte son los pesos de los lados tq \vec{xy} con $x \in S$ e $y \notin S$

→(def): Un corte es m $S \subseteq V$ tq $s \in S, t \notin S$ hay 2^{n-2} wnes posibles

La capacidad de un corte es: $\text{cap}(S) = c(S, \bar{S}) = \sum_{\substack{x \in S \\ y \notin S \\ \vec{xy} \in E}} c(\vec{xy})$



→(def): Un corte es minimal si: $\text{cap}(S) < \text{cap}(T)$ $\forall T$ corte posible.

la Σ de los caminos que no se cortan con una sola arista

→(Tema): Al cambiar el flujo f como en el algoritmo del F-F a un f' , lo que se queda es flujo y $V(f') = V(f) + e$

(6)

→ (Teorema de FF):

A) f flujo, S corte $\Rightarrow v(f) = f(S, \bar{S}) - f(\bar{S}, S)$

B) si f flujo, S corte $\Rightarrow v(f) \leq \text{Cap}(S)$

C) si f flujo las siguientes son equivalentes:

(1). $\exists S$ corte tq $v(f) = \text{Cap}(S)$

(2). f es maximal

(3). $\nexists f$ -caminos aumentantes entre s y t

Si ocurre alguna de las 3 $\Rightarrow S$ es minimal

→ (corolario): Si FF termina, termina con flujo maximal



→ (Teorema de Integridad): Si las capacidades son enteras, FF termina y el flujo maximal que obtiene es entero.

Todos y Algo (≥ 0) de EK

$$EK = F \cdot F + DFS$$

C. o de menor long

(7)

→ Dinit mejora EK si hay muchos caminos del mismo tamaño:

- 1) $F=0$
- 2) A partir de f me construyo un network auxiliar (NA)
- 3) Correr preedy en el NA. (modificar f)
- 4) Repetir 2 hasta que $t \neq n_a$.

→ (Flujo bloqueante): un flujo de s a t es un flujo bloqueante si todo camino siniestro entre s y t tiene al menos un lado saturado. \exists que no pueda aumentar el flujo más.

→ (def): una network pos niveles, es un network (V, E, C) donde existen $V_i \subseteq V$ $i=0, \dots, r$ (niveles) tq:

$$1) V = \bigcup_{i=0}^r V_i$$

$$2) E \subseteq \bigcup_{i=0}^{r-1} (V_i \times V_{i+1}) \quad (\vec{xy} \in E \Rightarrow \exists i, x \in V_i, y \in V_{i+1})$$

→ Dado un network N y un flujo f de s a t en N , el "network auxiliar"

relativo a f es $V = \bigcup_{i=0}^r V_i$ con $r = |d_f(s, t)|$ largo/distancia del camino aumentante entre s y t .

$$\left\{ \begin{array}{l} V_j = \{s\} \\ V_i = \{x : d_f(s, x) = i\} \text{ oír} \\ V_r = \{t\} \end{array} \right.$$



→ (lazos y capacidades del n.a.): ~~sin~~

- Si \vec{xy} es en lazo de N , $f(\vec{xy}) < c(\vec{xy})$, $x \in V_i$, $y \in V_{i+1}$ entonces \vec{yx} es un lazo del n.a. con capacidad = $c(\vec{xy}) - f(\vec{xy})$
- Si \vec{xy} es en lazo de N con $f(\vec{xy}) > 0$, $y \in V_i$, $x \in V_{i+1}$ entonces \vec{yx} es un lazo del n.a. con capacidad igual a $f(\vec{xy})$ (representa backward del network original)

→ (Teorema): la cantidad de networks auxiliares usados son $O(n)$

→ (corolario): la complejidad de calcular el algoritmo para usar network aux. es: $O(n) \cdot O(m) + \text{complejidad de hallar flujo bloqueante.}$

↓ ↓
Cant n.a. complejidad de
construir n.a.

→ (Teorema): La complejidad del Dinit original es $O(n^2m)$

(8)

→ (Teorema): La complejidad de Dinic-even es $O(n^2m)$

→ (diferencia entre Diniz y Diniz-even): even no va depurando a medida que los caminos se saturan, por lo tanto un dfs demora mas que $O(n)$, pero va depurando a medida que se corre dfs, borrando los lados por los cuales tenemos que retroceder. A dif. de diniz que come un alvo aparte su suspencion.

Algo del Dinic-even.

Matchings

(1)

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



• Problema a trabajar: Dado G , hallar un matching en G con la mayor cantidad de lazos posibles. Lo reduciremos a un problema de flujo maximal. Estudiaremos el problema para grafos bipartidos.

Tenemos que transformar un grafo bipartido G con partes X e Y en un network.



• vértices del network: $\{s, t\} \cup X \cup Y$

• lazos: $\{xy : x \in X, y \in Y, xy \in E\} \cup \{sx : x \in X \text{ y } y \in Y\}$

• capacidades: 1 para todos los vértices

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

$$|X| = |Y|$$

→ (definición): si $w \subseteq V$ entonces: $\Pi(w) = \{z : \exists w \in W : z \rightarrow w \in E\} = \bigcup_{w \in W} \Pi(w)$

→ (definición): un matching es perfecto si $V_m = V(G)$ usa todos los vértices de G

→ (definición): un matching es completo si $V_m \cap X = X$ usa todos los vértices de X

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

$$|S| \leq |\Pi(S)| \quad \forall S \subseteq X$$

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

König

(2)

→ (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): $\chi'(G) \geq \Delta$

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

Grafo bipartito con pesos

- Vamos a tener dos criterios para elegir los matchings
 - Asumimos de ahora en más: $|X| = |Z|$
 - \exists al menos 1 matching perfecto

(bímano) minimizar el costo I

(hungrero) minimizar la suma de los costos. II

→ (tema): 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 lo minimiza a la suma relativa a \bar{A} .

Sirve para demostrar 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 hungrero es $O(n^4)$ (a la prima fue lucido el profe)

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

en ambos algoritmos puedes usar el teorema de Hall,

$S \rightarrow$ filas marcadas

$\pi(S) \rightarrow$ col. marcadas

A1

B2

C3

Total

→ Hungrero
minimiza
este

→ Hungrero
minimiza
este

→ Hungrero
minimiza
este

→ Hungrero
minimiza
este

ejemplo algoritmo (1) : minimiza el costo

(3)

	1	2	3	4	5	6	7	8
A	1	2	5	5	3	8	2	9
B	9	8	3	9	8	8	1	3
C	3	1	5	8	9	6	5	8
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	4	1

	1	2	3	4	5	6	7	8
A	1	2	5	5	3	8	2	9
B	9	8	3	9	8	8	1	3
C	1	2	5	8	9	6	5	8
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	4	1

pueden encontrar un matching ✓ veamos si hay uno menor.

	1	2	3	4	5	6	7	8
A	1	2	5	5	3	8	2	9
B	9	8	3	9	8	8	1	3
C	1	2	5	8	9	6	5	8
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	4	1

es facil ver que no pueden haber un matching porque la fila D solo tiene 1 uno y no lo puedes usar por C que tambien tiene un solo uno.

Solo nos queda ver en (3) si hay otro matching repitiendo los pasos desde (2).

Luego de hacer todo vemos que no hay matching. Por lo cual, el primer pue encontramos minimiza el costo. Basa el matching con los pesos de A

A7:2 F3:3
B8:3 G2:1
C1:3 H6:3
D5:3
E4:4

Suma = 25

(1) hacer una lista de los n° dentro de la matriz, para hacer busquedas binaria. La idea es empezar con el n° del medio

i) si hay matching buscar si hay uno menor (menorizq)

ii) si no hay ver si hay con uno mayor (mayorizq)

$\frac{n^2}{2}$ veces

1 2 3 ④ 5 6 7 8 9

(2) me creo una nueva matriz B de la siguiente 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

(3) dar un matching inicial, iterativo

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 pues no puedo marcar, etiquetar con una S.

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

↑ 1s marcados

← 1s marcados

y:

- Si estoy viendo una fila etiquetar la col con la letra de la fila
- Si estoy viendo una col etiquetar la fila con el n° de la col.

por ultimo tacho las etiquetas que ya use.

(si ya tenia etiqueta no lo pongo una nueva)

(5) iterar 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

(4)

	1	2	3	4	
A	10	5	9	7	→ (1) → A → 0 0 0 2
B	9	5	5	8	→ B → 4 0 0 3
C	7	5	9	5	→ C → 2 0 4 0
D	10	5	4	7	→ D → 0 1 0 3

	1	2	3	4	
A	0	0	1	2	→ 0 0 1 2
B	4	0	0	3	→ 4 0 0 3
C	2	0	4	0	→ 2 0 4 0
D	0	1	0	3	→ 0 1 0 3

(1) restar el minimo a cada fila

(2) restar el minimo a cada columna

(3) buscar un matching de ceros

(4) tratar de extenderlo igual que el algo anterior

$$S = \{A, B, D\} \quad \pi(S) = \{2, 3\} \quad |S| = 3 > 2 = |\pi(S)| \quad \text{no hay matching por hoy.}$$

comprobamos →

$$(5) \text{ tomaremos } m = \min(S \times \overline{\pi(S)}) = 2$$

- (6) tachado:
 - las col ~~sin~~ con etiqueta
 - las filas sin etiquetas

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

- si esta tachado 1 vez dejo el mismo numero
- si no esta tachado le resto m
- si esta tachado obs veces le sumo m

(y monto las etiquetas)

(8) trato de extender el matching

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

(10) si despues de hacer (9) no puedo extender el

matching entonces tengo que repetir desde paso (5).

← os ↑ os marcados

opuesto

$$\left(\begin{array}{c} \text{tachado} \\ S \times \overline{\pi(S)} \end{array} \right)$$

	1	2	3	4	
A	0	0	4	0	1 2
B	0	0	0	1	3
C	0	2	6	0	.
D	0	1	0	1	8

B, D, A

pu de extender el matching

$$\begin{array}{r}
 A_4 \quad 7 \\
 B_2 \quad 5 \\
 C_1 \quad 7 \\
 D_3 \quad 4 \\
 \hline
 23 //
 \end{array}$$

(1)

Código de corrección de errores

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

Una suposición es que el medio de transmisión puede cambiar bits pero no puede eliminarlos o añadirlos. También 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 $0 < p \leq \frac{1}{2}$. La prob. de t errores es p^t .

→ (definición): la distancia de Hamming entre 2 palabras $x, y \in \{0,1\}^n$, es el número de bits de diferencia entre x y y .

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

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

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

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

$$D) d_H(x, z) \leq d_H(x, y) + d_H(y, z) \quad \text{Desigualdad triangular.}$$

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

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

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

→ (Teorema): Sea C un código 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 $R = S-1$ y $Q = \frac{S-1}{2}$

detecta corrige

Ejemplo: $C = \{00, 11, 10, 01\}$

(2)

$$\begin{aligned} d(00, 11) &= 2 \\ d(00, 01) &= 1 \\ d(10, 11) &= 1 \end{aligned} \quad \left. \begin{array}{l} \min = 1 = S \Rightarrow R = S - 1 = 0 \text{ (diferencia)} \\ Q = \frac{S-1}{2} = \frac{1-1}{2} = 0 \text{ (complemento de } S \text{ en } n=2) \end{array} \right.$$

→ En general si C es un código y $\text{repr}_r(C) = \{\underbrace{vv \dots v}_r : v \in C\}$ entonces es trivial ver que $S(\text{repr}_r(C)) = r \cdot S(C)$

→ (Teorema): (Cota de Hamming): Sea C un código de longitud n , $t = \frac{S-1}{2}$ entonces $|C| \leq \frac{z^n}{\binom{n}{t}}$

$$\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \dots + \binom{n}{t}$$

↑
Cont de palabras long

Ejemplo: $|C_4| = 4$ $S = 3$ $n = 4$, ¿Existe algún código tq $|C| = 4$, $n = 4$, $S = 3$?

$$\text{Si lo obviara tendríamos } t = \frac{S-1}{2} = 1 \text{ y } |C| = 4 \leq \frac{z^4}{\binom{4}{1}} = 3 \Rightarrow \text{abs} \Rightarrow \emptyset$$

→ (definición): Un código es perfecto si $|C| = \frac{z^n}{\sum_{r=0}^t \binom{n}{r}}$ con $t = \frac{S-1}{2}$

Códigos Lineales

→ (definición): Un código es lineal si es un sub-espacio vectorial de $\{0,1\}^n$

→ (subespacio vectorial): C es subespacio vectorial de $\{0,1\}^n$ si

el cuerpo es $\{0,1\}$ {

- $x, y \in C \Rightarrow x+y \in C$ Cerrado por suma
- $k \in \text{cuerpo}, x \in C \Rightarrow kx \in C$ Cerrado por mult
- $C \neq \emptyset$ No sea vacío.

↳ basta con ver 2 cosas:

- $x, y \in C \Rightarrow x+y \in C$
- $0 \in C$

imp

→ (definición): El peso de Hamming es $\|x\| = d_H(x, 0) = \# \text{cont de } 1 \text{ de } x$

→ (propiedad): C es lineal $\Rightarrow S(C) = \min \{ \|x\| : x \neq 0, x \in C \}$

- Todo espacio vectorial tiene dimensión k y al menos una base
- Una base cumple
 - genera el espacio
 - linealmente independiente $c_1x_1 + \dots + c_nx_n = 0 \Rightarrow c_1 = \dots = c_n = 0$

→ (definición): una matriz generadora de un código lineal C es una matriz cuyas filas son base de C . Como las filas son base cualquier matriz generadora debe ser $k \times n$.

→ cant de filas
cant de columnas

Obs: Si $k = \dim(C) \Rightarrow C$ es isomorfo a $\{0, 1\}^k \Rightarrow |C| = 2^k$

→ (definición): Una matriz H es una matriz de chequeo de un código C si $C = N(H) = \{y \in \{0, 1\}^n : Hy^t = 0\}$

→ (Teorema): Si H es una matriz de chequeo de C , entonces $S(C) = \min$ numero de columnas de H linealmente dependientes $= \min \{r : \exists r$ columnas L.D. de $H\}$
ej: $H^2 = H^1 + H^3 + H^4 \Rightarrow S=4$ y cant de terminos,

→ (corolario): Si H no tiene columnas 0 ni columnas repetidas y $C = N(H)$ entonces $S(C) \geq 3$ y comete al menos un error.

$$\frac{3-1}{2} = 1$$

Sirve para calcular $\dim(C)$ a partir de H , suponiendo que las filas de H son LI.

→ (Teorema): Si $T: V \rightarrow W$ es una transformación lineal entonces

$$\dim(V) = \dim(N(T)) + \dim(\text{Img}(T))$$

$$T(x) = Hx^t \Rightarrow n = k + \dim(\text{Img}(T))$$

$$k = n - \# \text{filas}$$

falta el razonamiento que lleva de C

→ (propiedad): la matriz de chequeo H va a ser una matriz de $(n-k) \times n$

si tengo k filas y n columnas la cantidad de combinaciones posibles sin repetir col es de 2^{K-1}

para sacar el vector nulo

(2)

- (propiedad): Si H es de la forma $[A \mid I]$ y $C = N(H)$, entonces $G = [I \mid A^t]$ es generadora de C . Viceversa si $G = [I \mid A^t]$ es generadora, $[A \mid I]$ es de cheques.

Ejemplo: $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$

$$\begin{cases} n=7 \\ k=7-3=4 \\ 2^k=2^4=16=|C| \end{cases}$$

Supongamos que llega mensaje $z = 1001010$

$$Az^t = H \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = H^{(1)} + H^{(2)} + H^{(3)} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

el x mas probable es $x = z + e_3 = 1011010$

Sino estaba todo en error

es una col $H = H^{(3)}$

tenemos que sumar a la palabra que me llevo el 3 para tener la mas probable. O sea puse numero que cambiar el bit i fui me dio la col $H^{(3)}$

- (definición): Si H tiene todas las columnas no nulas, el código se llama "código de hamming".

- (propiedad): Los códigos de hamming son perfectos.

Una palabra está en el código si cuando la multiplico por H da 0

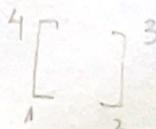
Códigos cíclicos.

los códigos de hamming son lineales

- Cambio de notación: a la palabra w_1, \dots, w_n la denotaremos w_0, \dots, w_{n-1} pues identificaremos la palabra $w = w_0, \dots, w_{n-1}$ con el polinomio $w(x) = w_0 + w_1x + \dots + w_{n-1}x^{n-1}$ en $\mathbb{Z}_2[x]$. No confundir con funciones, $1+x \neq 1+x^2$ como pol., pero como funciones como estos en $\mathbb{Z}_2[x]$ son iguales.

- Queremos multiplicar palabras de longitud n , podemos multiplicar los polinomios asociados, pero queremos que el resultado tenga grado $< n$. Lo que se hace es tomar la multiplicación modulo algun polinomio de grado n . Tomaremos el pol. $1+x^n$.

para hacerlo basta con mirar así



(3)

→ (definición): $v(x) \odot w(x) = v(x) \cdot w(x) \bmod (1+x^n)$

$$\text{ej: } (1011) \odot (0110) = (1+x^2+x^3) \cdot (x+x^2) \bmod 1+x^4 = x^2+x^3 = 0111$$

$$\begin{aligned} \text{obs: } (1+x^n) \bmod (1+x^n) &= 0 \\ x^n \bmod (1+x^n) &= 1 \end{aligned}$$

→ (definición): $\text{rot}(w) = \text{rot}(w_0, \dots, w_{n-1}) = w_{n-1}, w_0, \dots, w_{n-2}$

→ (propiedad): $\text{rot}(w) = x \odot w(x)$
"Obvia"

→ (definición): Un código C es cíclico si es lineal e invariante por rotaciones:

$\text{rot}(w) \in C : \forall w \in C$ lo rota y vuelve a caer en C .

→ (propiedad): Si C es cíclico entonces existe un único polinomio no nulo en C de grado mínimo, (que tiene la menor cantidad de x 's)

→ (definición): El único pol. no nulo de grado mínimo de un código cíclico C , se lo llama polinomio generador y se denota $g(x)$.

→ (corolario de "propiedad"): Sea C cíclico, $w(x)$ cualquier palabra en C y $v(x)$ cualquier polinomio de cualquier grado. Entonces $w(x) \odot v(x) \in C$
Toda palabra de igual longitud a la longitud de $w(x)$.

→ (Teorema fundamental de los códigos cíclicos): Sea C un código cíclico de longitud n , con generador $g(x)$. Entonces:

$$(1) \quad C = \{ p(x) \in \mathbb{Z}_2[x] : \text{gr}(p(x)) < n \wedge g(x) \mid p(x) \}$$

$$(2) \quad C = \{ v(x) \odot g(x) : v \in \mathbb{Z}_2[x] \}$$

$$(3) \quad \text{Si } k = \dim(C) \Rightarrow \text{gr}(g(x)) = n-k \rightarrow [k = n - \text{gr}(g(x))]$$

$$(4) \quad g(x) \mid (1+x^n)$$

$$(5) \quad \text{Si } g(x) = g_0 + g_1 x + \dots + g_{n-1} x^{n-1} \Rightarrow g_0 = 1$$

Ejemplo método 1:

Método 1 de codificación (método 2)

(6)

$$g(x) = 1 + x^2 + x^3 \quad (n=7) \rightarrow \text{longitud}$$

$$k = \dim(c) = 7 - \text{gr}(g(x)) = 7 - 3 = 4 \quad (\text{dimension})$$

$$|c| = 2^4 = 16 \quad (\text{cant de palabras})$$

Si quiero codificar $\{0,1\}^4 \rightarrow \{0,1\}^7$
basta hacer $\rightarrow g(x) \rightarrow (g(x) \cdot q(x))$

Codificar 1001:

$$1001 = (1+x^3) \cdot (1+x^2+x^3)$$

$$= 1+x^2+x^3+x^5+x^6$$

$$= 1+x^2+x^5+x^6$$

$$= (1010011)_2$$

(cíclicos)

→ (Idea método 2 codificación): $\forall p, [(p \bmod g) + p]$ es múltiplo de g pues

$$((p \bmod g) + p) \bmod g = p \bmod g + p \bmod g = 0$$

Usaremos este truco pero con cuidado pues $\{0,1\}^k \rightarrow \{0,1\}^n$

porque esta función no es inyectiva: ej: $q \rightarrow q \cdot (q \bmod g)$ no funciona

Lo que se tiene que hacer es: $q \rightarrow (x^{n-k} q \bmod g) + x^{n-k} q$ \circledast

$$\text{ej: } g(x) = 1 + x^2 + x^3 \quad n=7 \quad q(x) = 1100 = 1+x \quad k=4$$

$$x \cdot q(x) = x^3 + x^4 \rightarrow x^3 + x^4 \bmod g = 1 + x^2 + 1 + x + x^2 = x$$

$$q \bmod g = 0$$

$$(1 + x^2 + x^3) \bmod g = 0$$

$$(1 + x^2) \bmod g + x^3 \bmod g = 0$$

$$1 + x^2$$

$$\Rightarrow x^3 \bmod g = 1 + x^2$$

$$x^4 \bmod g$$

$$x \cdot x^3 \bmod g$$

$$x \cdot (1 + x^2) \bmod g$$

$$(x + x^3) \bmod g$$

$$1 + x + x^2$$

usando \circledast

$$(x^3 + x^4) \bmod g + x^3 + x^4 = x + x^3 + x^4$$

$$= 0101100$$

4

Ahora nos construimos la matriz generadora

$$100 \rightarrow 0 \rightarrow 1 \rightarrow (x^{n-k} \bmod g) + x^{n-k}$$

$$010 \rightarrow 0 \rightarrow x \rightarrow (x^{n-k+1} \bmod g) + x^{n-k+1}$$

tracción

6.

matriz generadora mat1

multiplicar una base de $\{0,1\}^k$ por $g(x)$

$$\text{ej: } 1 \cdot 1 \Rightarrow \{0,1\}^4$$

$$\begin{array}{l}
 1000 \rightarrow 1 \cdot g \\
 0100 \rightarrow x \cdot g \\
 0010 \rightarrow x^2 \cdot g \\
 0001 \rightarrow x^3 \cdot g
 \end{array}
 = \left. \begin{array}{l}
 g \\
 gx \\
 gx^2 \\
 gx^3
 \end{array} \right\}
 \text{ }$$

$$G = \begin{bmatrix} g \\ gx \\ gx^2 \\ gx^3 \end{bmatrix}$$

(7)

→ (polinomio que quedar): $g \mid 1+x^n \Rightarrow \exists h(x): 1+x^n = g(x) \cdot h(x)$

$h(x) = \frac{1+x^n}{g(x)}$ Suponemos que $p \in C \Rightarrow p = qg$ para algún q del grado apropiado.

$$ph = qgh = q(1+x^n) \Rightarrow ph \bmod (1+x^n) = 0 \text{ es decir } p(x) \odot h(x) = 0$$

Si cumple → la palabra él se en el código.

matriz de cheques: $B = [A \mid \text{Id}] \Rightarrow H = [\text{Id} \mid A^t]$

$$\boxed{x^n \bmod g = 1 \Rightarrow g \mid x^n + 1 \text{ fimp}}$$

Error de trapping

- Suponemos que C es cíclico de longitud n con pol generador $g(x)$ y contiene errores. Supongamos que mandamos $v \in C$ y llega $w = v + e$ con $\|e\| \leq t$

Si tomamos mod g : $(w \bmod g) = (v + e) \bmod g = \underbrace{v \bmod g}_{=0 \text{ pues } v \in C} + e \bmod g = e \bmod g$

Pero esto mismo vale para las rotaciones

$$\bullet \text{rot}^i(w) = \text{rot}^i(v) + \text{rot}^i(e) \quad \text{rot}^i(e) \text{ tiene } \|e\| \leq t$$

$$\bullet \text{rot}^i(w) \bmod g = \underbrace{\text{rot}^i(v) \bmod g}_{=0} + \text{rot}^i(e) \bmod g = \text{rot}^i(e) \bmod g$$

Si para algún i $\text{gr}(\text{rot}^i(e)) < \text{gr}(g(x)) \Rightarrow \text{rot}^i(e) \bmod g = \text{rot}^i(e)$, equiv.

es decir que todos los "1" del error están en una ventana de longitud $n-k = \text{gr}(g(x))$. Si pasa eso: $\text{rot}^i(w) \bmod g = \text{rot}^i(e) \Rightarrow e = \text{rot}^{-i}(\text{rot}^i(w) \bmod g)$

En términos de polinomios, el algoritmo queda:

$$S_0 = \cancel{w \bmod g} \quad \text{Síntrome} \quad \rightarrow \text{palabra pue lleva}$$

$$S_i = (x \cdot S_{i-1} \bmod g) \quad i \geq 1$$

$$\text{hasta que } \|S_i\| \leq t \rightarrow \text{error} = x^{n-i} S_i \bmod (1+x^n)$$

Luego la palabra más probable es $w(x) = w(x) + \text{error}$
falto ejemplo

Como ver que $g \mid (1+x^n)$,

tiene que cumplir

$$(1+x^n) \bmod g = 0$$

$$\boxed{x^n \bmod g = 1}$$