Matematica Discreta II

→ 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"

→ (notacion): $\{x,y\} \to xy = yx$

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

→ (grado de valencia): el grado de valencia de un vertice x es $d(x) = |\Gamma(x)|$

$\delta = \delta(G) = \min\{d(x)\ x \in \mathbb{N}\}$
$\Delta = \Delta(G) = \max\{d(x),\ x \in \mathbb{N}\}$

Un grafo es regular si $\delta = \Delta$
"la cant de vecinos de x"

→ un grafo es planar si se puede dibujar sin que ninguna arista se cruce con otra

→ (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 ciclico si $V = \{1, 2, \dots, n\}$ y $E = \{12, 23, \dots, n1\}$ 

→ Un grafo es completo si $V = \{1, \dots, n\}$ $\underbrace{E = \{A \subseteq V \mid |A| = 2\}}_{\text{todas los conexiones posibles}}$ 

→ (convencion): $n = |V|$ y $m = |E|$

Coloreo

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

El coloreo es propio si $xy \in E \Rightarrow c(x) \neq c(y)$ vecinos tengan distintos colores

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

(·) Una propiedad obvia es que $\chi(G) \leq n$ ayuda casos simetricos: reducir todos los colores a 1 (por simetria) y luego probar que no se puede con $k-1$

Para probar un coloreo propio en un grafo G:

(1) probar que existe un coloreo propio (decir cual) de k colores

(2) Demostrar que es el minimo (por absurdo), es decir $\nexists$ coloreo con $k-1$ colores

→ (propiedad): · Si H es un subgrafo de $G \Rightarrow \chi(H) \leq \chi(G)$

· $\chi(K_n) = n$

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

Algoritmo de Greedy para coloreo

→ (greedy para coloreo): requiere un orden de los vertices $x_1, \ldots, x_n$

$c(x_1) = 0$

para $i \geq 2$

$$c(x_i) = \min\{\overbrace{k \geq 0;\ c(x_i) \neq k}^{\text{minimo color posible}} : \overbrace{\forall j < i : x_i \in \Gamma(x_i)}^{\text{vecinos anteriores}}\}$$

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

→ (complejidad de greedy): $\sum O(d(x_i)) = O(\sum d(x_i)) \underset{\downarrow}{=} O(2n) = O(n)$

T. apreton de manos

colorear
— ciclos pares — necesito 2 colores
— ciclos impares — necesito 3 colores

→ (camino): Un camino entre dos vertices x e y es una sucecion de vertices $x_1, \ldots, x_n$ tq $x_1 = x$, $x_r = y$, $x_i x_{i+1} \in E \ \forall i \in \{1, 2, \ldots, r-1\}$

no conexo

→ Un grafo es conexo si tiene una sola componente conexa

puede llegar de cualquier pto a cualquier pto 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 que G sea un ciclo impar o un grafo completo.

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

↳ puede decir que greedy con un mal ordenamiento puede dar mas cant de colores

→ (Teorema VIT): Sea $G=(V,E)$ un grafo cuyos vertices estan coloreados con un coloreo propio c con r colores $\{0, \ldots, r-1\}$. Sea π una permutacion de los numeros $0, \ldots, r-1$. Sea $V_i = \{x \in V : c(x_i) = i\}$ $i=0, \ldots, r-1$.
Ordenamos los vertices poniendo primero los vertices de $V_{\pi(0)}$, luego $V_{\pi(1)}$, hasta $V_{\pi(r-1)}$. Entonces greedy en ese orden colorea a G con r colores o menos.

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

→ (grafo bipartito): se dice que un grafo es bipartito si $\chi(G)=2$, osea que $\exists\, x, y \subseteq V$ tq: $\left.\begin{array}{l} \cdot\ V = x \cup y \\ \cdot\ x \cap y = \emptyset \end{array}\right\}$ "partir el grafo en 2"

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

Flujos y Networks

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

→ (network): un network es una tripla (V,E,C) donde (V,E) es un grafo dirigido y $\underbrace{C}_{\text{costo/capacidad}} : E \to \mathbb{R}_{\geq 0}$

→ (notacion): • Si G esta definido en E y $A, B \subseteq V$, definimos

$$g(A,B) = \sum_{\substack{x \in A \\ y \in B \\ (x,y) \in E}} g(x,y)$$ (llegada / salida) Suma de un subgrafo

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

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

• $\overrightarrow{xy} = (x,y)$

vecinos hacia adelante

→ (definicion): • $\Gamma^+(x) = \{y : (x,y) \in E\}$ $out_g(x) = \sum_{y \in \Gamma^+(x)} g(\overrightarrow{xy})$

• $\Gamma^-(x) = \{y : (y,x) \in E\}$ $in_g(x) = \sum_{y \in \Gamma^-(x)} g(\overrightarrow{yx})$

vecinos hacia atras

→ (definicion): Dado un network (V,E,C) y vertices $s,t \in V$. Un flujo de s a t es una funcion f en los lados con:

s → t

(a) $0 \leq f(\overrightarrow{xy}) \leq C(\overrightarrow{xy})$ $\forall \overrightarrow{xy} \in E$

(b) $in_f(x) = out_f(x)$ $\forall x \neq s,t$ todo lo que entra sale

(c) $in_f(s) = out_f(t) = 0$ nada entra a s y nada sale de t

→ (def): el valor de un flujo es $v(f) = out_f(s) - \overset{0}{in_f(s)} = out_f(s)$

→ (Flujo maximal): Un flujo es maximal si $v(g) \leq v(f)$ $\forall g$ flujo de s a t

↳ puede no ser unico pero si el mayor.

→ (prop): $v(f) = in_f(t)$

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

Completar algoritmo greedy

escribir ideas

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

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

II) $\overrightarrow{x_{i+1} x_i} \in E \wedge f(\overrightarrow{x_{i+1} x_i}) > 0$ que haya mandado flujo para poder (backward) devolverlo

Copiar Algoritmo de FF y idea

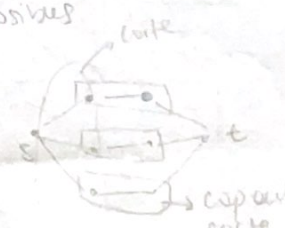
Por eso t nunca pertenece a S

los elementos del corte son los vertices alcanzados por la ultima it.

la capacidad de un corte son los pesos de los lados tq $\overrightarrow{xy}$ con $x \in S$ e $y \notin S$

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

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



la $\sum$ de los conexiones de los del conjunto con uno de afuera

→ (def): Un corte es minimal si: $cap(S) \leq cap(T)$ $\forall$ T corte posible.

→ (lema): al cambiar el flujo f como en el algoritmo de F-F a un f', lo que se queda es flujo y $V(f') = V(f) + \varepsilon$

→ (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 $\Rightarrow v(f) \leq Cap(S)$

C) si f flujo las siguientes son equivalentes:

(1) · $\exists\, S$ corte tq $\boxed{v(f) = cap(S)}$

(2) · f es maximal

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

Si ocurre alguna de las 3 $\Rightarrow$ $\boxed{S \text{ es minimal}}$

→ (corolario): Si F-F termina, termina con flujo maximal

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

Idea y Alg (o ej) de EK

EK = F-F + DFS
C.a de menor long

→ Diniz 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 greedy en el n.a. (modificar f)
4) Repetir 2 hasta que $t \notin$ n.a.

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

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

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

→ 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 $r = \boxed{d_f}(s,t)$ la menor distancia de camino aumentante entre s y t.

niveles $\begin{cases} V_0 = \{s\} \\ V_i = \{x : d_f(s,x) = i\} \quad 0<i<r \\ V_r = \{t\} \end{cases}$

$\underbrace{s}_{V_0} \quad \dots \quad \underbrace{t}_{V_r}$

→ (lados y capacidades de n.a):

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

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

→ (Corolario): la complejidad de calcular el algoritmo que use network aux. es: $\underbrace{O(n)}_{\text{cant n.a}} \cdot \underbrace{O(m)}_{\text{complejidad de construir n.a.}}$ + complejidad de hallar flujo bloqueante.

→ (Teorema): La complejidad de Diniz original es $O(n^2 m)$

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

→ (diferencia entre dinitz y dinitz-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 dinitz que coma un algo aparte de depuracion.

Algo de Dinic-even.