

→ 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 lazos"

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

→ (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 $d(x) = |\Gamma(x)|$

$$\begin{aligned} S = S(G) &= \min \{d(x) : x \in V\} \\ \Delta = \Delta(G) &= \max \{d(x) : x \in V\} \end{aligned} \quad \left\{ \begin{array}{l} \text{Un grafo es regular si } S = \Delta \\ \text{"la cant de vecinos de } x \text{"} \end{array} \right.$$

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



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



→ (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 $xy \in E \Rightarrow c(x) \neq c(y)$ vecinos tienen distintos colores

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

(*) Una propiedad obvia es que $\chi(G) \leq n$ significa casos sencillos: reducir todos los colores a 1 (por simetría) 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 (destruir) de k colores

(2) Demostrar que es el mínimo (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, \dots, x_n

$c(x_1) = 0$

para $i \geq 2$ minimo color posible

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

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

→ (complejidad de greedy): $\equiv O(d(x_i)) = O(\sum d(x_i)) = O(2n) = O(n)$

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

T. apretón de manos

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

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

 no camino

→ 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 coloreo a todos los vertices, salvo a uno, con Δ colores o menos.

↳ puede ocurrir que greedy con un mal ordenamiento pueda 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 $\{0, \dots, r-1\}$. Sea π una permutacion de los numeros $0, \dots, r-1$. Sea $V_i = \{x \in V : C(x) = i\}$ $i=0, \dots, r-1$.

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
 $\bullet \exists x, y \subseteq V$ 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

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

→ (network): un network es una tripleta (V,E,C) donde (V,E) es un grafo

\bullet dirigido y $C: E \rightarrow \mathbb{R}_{\geq 0}$
costo/capacidad

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

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

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

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

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

→ (definición): $\Gamma^+(x) = \{y : (x,y) \in E\}$ vecinos hacia adelante $\text{out}_g(x) = \sum_{y \in \Gamma^+(x)} g(\vec{xy})$
 $\Gamma^-(x) = \{y : (y,x) \in E\}$ vecinos hacia atrás $\text{ing}_g(x) = \sum_{y \in \Gamma^-(x)} g(\vec{yx})$

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

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

$$(b) \text{inf}_f(x) = \text{out}_f(x) \quad \forall x \neq s, t \quad \text{todo lo que entra sale}$$

$$(c) \text{inf}_f(s) = \text{out}_f(t) = 0 \quad \text{nada entra a } s \text{ y nada sale de } t$$

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

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

→ (prop): $v(f) = \text{inf}_f(t)$

↳ puede no ser único pero si el mayor.

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

Computar algoritmo preedy

describir tarea

(5)

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

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

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

Copyr Algorithmo de FF y Iowa

Por eso t nunca pertenece a S

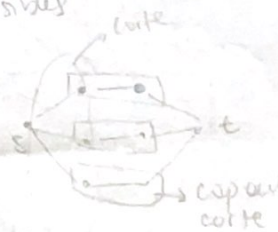
los elementos del corte son los vertices alcanzados por la ultima it.
la capacidad de un corte son los pesos de las lacos tq \overrightarrow{xy} con $x \in S$ e $y \notin S$

→ (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: $\text{Cap}(S) = c(S, \bar{S}) = \sum_{\substack{x \in S \\ y \notin S \\ \overrightarrow{xy} \in E}} c(\overrightarrow{xy})$

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

$\forall T$ corte posible.



la \sqsubseteq de los cortes en los de aumento con uno de otro

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

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

~~Time~~ y Algo (o ej) de E_k

$E_k = F + DFS$
C.2 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 preedy en el na. (modificar f)
- 4) Repetir 2 hasta que $t \notin na$.

→ (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. \equiv a que no pueda aumentar el flujo preedy.

→ (def): una network por 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 $r = d_f(s, t)$ la menor distancia de camino aumentando entre s y t .

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

→ (lados y capacidades de na): ~~en~~

• Si \vec{xy} es en lado de N , $f(\vec{xy}) < c(\vec{xy})$, $x \in V_i$, $y \in V_{i+1}$ entonces \vec{xy} es un lado del n.a. con capacidad $= c(\vec{xy}) - f(\vec{xy})$

• Si \vec{xy} es un lado de N con $f(\vec{xy}) > 0$, $y \in V_i$, $x \in V_{i+1}$ entonces \vec{yx} es un lado del na 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 que use network aux. es: $O(n) \cdot O(m) + \text{complejidad de hallar flujo bloqueante.}$

\downarrow \downarrow
 Cant n.a complejidad de
 construir n.a.

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

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

→ (diferencia entre Dinic y Dinic-even): even no va depurando a medida que los caminos se saturan, por lo tanto un dfs demora más que $O(n)$, pero va depurando a medida que se corre dfs, borrando los labels por los cuales tenemos que retroceder. A dif. de Dinic que como un algo aparte de depuración.

Algo de Dinic-even.