Discreta II: Pasos para las demos

Mansilla, Kevin Gaston*

18 de junio de 2023

1) **Cual es la complejidad del algoritmo de Edmonds-Karp? Probarlo (Nota: en la prueba se definen unas distancias, y se prueba que esas distancias no disminuyen en pasos sucesivos de EK. Ud. puede usar esto sin necesidad de probarlo.)**

1) Un lado se vuelve crítico al pasar de f_k a f_{k+1} si se satura o vacia. Definir el paso k y j para un vertice z

2) Analizar $\overleftarrow{xz} \in E$, $\exists l : k \leqslant l \leqslant j$ ver caso backward y fordward para l.

3) Entonces la conclución para k. $d_k(z) = d_k(x) + 1$ (fordward) y $d_k(x) = d_k(z) + 1$ (backward).

4) Usar todo lo anterior para llegar a $d_l(t) = d_k(t) + 2$ que es la conclusión de la pruba, es que una vez que un lado se vuelve crítico solo puede volver a ser crítico si la distancia entre s y t aumenta en por lo menos 2. que es $\frac{n-1}{2}$ veces.

5) Resumen de la prueba. hay m lados odnde cada lado se vuelve critico $O(n)$ veces por lo que es $O(mn)$ más la complejidad de BFS que es $O(m)$ por lo que la complejidad es $O(m^2n)$.

2) **Probar que si, dados vértices x, z y flujo f definimos a la distancia entre x y z relativa a f como la longitud del menor $f-$camino aumentante entre x y z, si es que existe tal camino, o infinito si no existe o 0 si $x = z$, denotandola por $d_f(x, z)$, y definimos $d_k(x) = d_{f_k}(s, x)$, donde f_k es el $k-$ésimo flujo en una corrida de Edmonds-Karp, entonces $d_k(x) \leqslant d_{k+1}(x)$**

1) Definir $A = \{y : d_{k+1}(y) < d_k(y)\}$ y tratar de probar que $A = \emptyset$ por absurdo.

2) Tomar un elemento $x \in A$ tal que $d_{k+1}(x) \leqslant d_{k+1}(y)$ $\forall y \in A$. entonces en x se cumple lo mismo que en y. Y fundamental es tomar un $f_{k+1}-$ca entre s y x de menor longitud (EK).

3) Sea z el vertice inmediatamente anterior a x como el camino es de longitud minima $d_{k+1}(x) = d_{k+1}(z) + 1$.

4) Como $z \notin A$ se da lo opuesto que en A. Ademas existe un f_k- camino aumentante entre s y z y tomamos el de longitud minima (EK).

*kevingston47@gmail.com

5) Primero analizamos el camino $s \dots \overleftarrow{xz} \dots t$ y hay que llegar a $0 < 2$.

6) Analizar caso fordward del f_k-ca y se llega a $0 < 0$

7) Conclusiones de la prueba

3) **Cual es la complejidad del algoritmo de Dinic? Probarla en ambas versiones: Dinitz original y Dinic-Even. (no hace falta probar que la distancia en networks auxiliares sucesivos aumenta)**

1) **Dinitz original**

   a) Definir el colorario de complejidad de hallar un FB.
   b) Definir que la complejidad de hallar un FB es $O(mn)$.
   c) Depurar en el primer NA, como hay r niveles es $O(n)$.
   d) Cada camino satura al menos un lado $O(m)$ caminos.
   e) Hallar la complejidad de todos los podar.
   f) Primero revisar los vertices es $O(1)$ pero como hay n vertices es $O(n)$.
   g) Borrar lados, no queremos la de un podar sino la de todos, pues un podar puede ser muy grande y va redciendo a medida que se borran lados. Entonces es $O(m)$.
   h) Conclusión de la prueba.

2) **Dinic-Even**

   a) La complejidad se halla usando el corolario de complejidad de hallar un FB.
   b) La complejidad de hallar un FB es $O(mn)$.
   c) Una corrida es una palabra que se obtiene con DFS (dar ejemplo).
   d) Definir $AVANZAR$, $RETROCEDER$ y $INCREMENTAR\_E\_INICIALIZAR$. y dar complejidades.
   e) Calcular la complejidad de una palabra $A \dots AX$ es $O(m)$.
   f) Calcular cuantas palabras hay, es $O(n)$.
   g) Conclusión

4) **Cual es la complejidad del algoritmo de Wave? Probarla. (no hace falta probar que la distancia en networks auxiliares sucesivos aumenta).**

1) Corolario de flujos bloqueantes.

2) Definir los fwb, bwb y cantidad de olas $O(n)$.

3) Calcular la complejidad de los fwb dividiendolo en dos partes primero S que es la complejidad total de los fwb saturado y P que es la complejidad parcial de fwb.

4) Los mismo para los bwb con V para los bwb vacios y Q para los bwb parciales.

5) Conclución $S + P + V + Q = O(n^2)$.

5) **Probar que la distancia en networks auxiliares sucesivos aumenta.**

1) Definir NA, NA', d y d' para probar que $d(t) < d'(t)$.

2) Suponemos $d'(t) < \infty$ por lo que existe al menos un camino aumentante, entre s y t en el network original, por lo tanto existe un camino dirigido de s a t en el NA'.

3) Sea $s = x_0, x_1, \ldots, x_n = t$ un camino dirigido en NA'. Como NA' es por niveles entonces $d(x_i) = i$. Lo más importante es que ese camno no puede estar en NA, porque para pasar de NA a NA' se bloquean todos los caminos de NA por lo tanto si ese camino estuviera en NA se hubiera bloqueado y no estaría en NA' sino es camino en NA entonces puede suceder: Falta un vertice o Falta un lado.

4) Analizar le caso en que falte un vertice. tomar un x cualquiera por lo que $x_i \notin NA$ entonces $d(t) \leqslant d(x_i)$, esar EK.

- Analizar el caso en que falte un lado, tiene dos casos.

5) Caso 1, $d(x_{i+1}) < i + 1$ tengo que llegar a $d(t) < d'(t)$

6) Caso 2, $d(x_{i+1}) = i + 1$.

6) **Si f es flujo las siguientes son equivalentes:**

   1. $\exists S$ **corte:** $v(f) = cap(S)$

   2. f **es maximal.**
   $(1 = 2)$ **dice:**
   "f **maximal** $\Longleftrightarrow \exists S$ **corte** $v(f) = cap(S)$"
   y se suele llamar 'max-flow-min-cut theorem'.

   3. $\nexists f-$**caminos aumentanes entre** s **y** t
   y si se cumplen, el s **es minimal.**

Definir y demostrar en ese orden.

- Si f es flujo las siguientes son equivalentes:

   1. $\exists S$ corte: $v(f) = cap(S)$
   2. f es maximal.
   $(1 = 2)$ dice:
   "f maximal $\Longleftrightarrow \exists S$ corte $v(f) = cap(S)$"
   y se suele llamar 'max-flow-min-cut theorem'.
   3. $\nexists f-$caminos aumentanes entre s y t
   y si se cumplen, el s es minimal.

La prueba es 1) $\Rightarrow$ 2) $\Rightarrow$ 3) $\Rightarrow$ 1).

1) $\Rightarrow$ 2) es fácil. 2) $\Rightarrow$ 3) se hace por contrareciproca.

3) $\Rightarrow$ 1) es la parte más difícil.

1) Definir $S = \{s\} \cup \{x : \exists \text{ un } f-\text{ca de } s \text{ a } x\}$.

2) Como f es flujo y S corte, entonces $v(f) = f(S, \overline{S}) - f(\overline{S}, S)$. entonces $f(S, \overline{S}) = \sum_{x \in S, y \in \overline{S}, xy \in E} f(\overrightarrow{xy})$.

3) Tomar un par (x, y) cualquiera $x \in S, y \notin S, xy \in E$ entonces existe un ca entre $s \dots x$

4) suponfamos que $f(\overrightarrow{xy}) < c(\overrightarrow{xy})$ no se satura. se llega a un absurdo $y \in S$

5) Ahora lado backward $f(\overline{S}, S) = \sum_{x \notin S, y \in S, xy \in E} f(\overrightarrow{xy})$.

6) Se toma un par cualquiera (x, y) mismo analisis que en el paso 3. pero para backward, ahora el absurdo es $x \in S$.

7) **Probar que $2-$COLOR es polinomial.**

8) **Enunciar y probar el Teorema de Hall.**

Teorema 1 *(Hall) Si $G = (\bar{X} \cup \bar{Y}, E)$ es bipartito con partes $\bar{X}$ e $\bar{Y}$, entonces existe matching completo de $\bar{X}$ en $\bar{Y}$ si y solo si $|S| \leqslant |\Gamma(S)|\ \forall S \subseteq \bar{X}$.*

1) $\Rightarrow$ Definir la función inyectiva $\bar{X} \cap \bar{Y}$ tal que $\psi(x) \in E$, por lo tanto:

$$\psi(S) \subseteq \Gamma(S)$$

2) $\Leftarrow$ Se demuestra por contrareciproc, es decir que si no existe matching completo de $\bar{X}$ en $\bar{Y}$ entonces al correr el algoritmo llegamos a un matching maximal que no cubre a $\bar{X}$. Que es equivalente a hallar un flujo maximal entero f cuyo valor no es $|\bar{X}|$.

3) Al hallar f, tambien hallamos un corte minimal que vamos a denotar por c (seria la última cola, al correr EK).

4) Sea $S = c \cap \bar{X}$, $T = c \cap \bar{Y}$, T forma parte de c por lo tanto forma parte de la última cola, entonces todos sus elmeentos fueron agregados por alguien (pues $S \notin T$), ese alguien debe ser vecino y como el grafo es bipartito y $T \subseteq \bar{Y}$, esos vecnios deben estar en $\bar{X}$. Pero ademas deben haber estado en la cola, es decir, que están en c. Entonces el vecino estaba en S. Gracias a esto $T \subseteq \Gamma(S)$ y $\Gamma(S) \subseteq T$.

5) Hay que probar estas influciones

6) $\Gamma(S) \subseteq T$. Sea $y \in \Gamma(S)$, entonces $\exists x \in S : xy \in E$ (x esta en la cola). Supones que $f(\overrightarrow{xy}) = 0$ entonces $xy \in c$ y $y \in T$. Supones que $f(\overrightarrow{xy}) = 1$ entonces x no puede agregar a y a la cola pero $x \in S$, entonces algun vertice z agrego a x a la cola. Seguir en base a esto

9) **Enunciar y probar el teorema del matrimonio de Konig**

Teorema 2 *(matrimonio de konig) Todo grafo bipartito regular tiene un matching perfecto (todos los vertices dorman parte del matching).*

1) Se parte dado un conjunto de vertices definiendo $E_w = \{zw \in E : z \in W\}$.

2) Sean $\bar{X}$ y $\bar{Y}$ las partes de G y suponemos $w \subseteq \bar{Y}$ (completamente contenido en $\bar{X}$).

10) **Probar que si G es bipartito entonces $\chi^{'}(G) = \Delta(G)$**

11) **Probar la complejidad $O(n^4)$ del algoritmo Hungaro y dar una idea de como se la puede reducir a $O(n^3)$.**

12) **Enunciar el teorema de la cota de Hamming y probarlo**

13) **Probar que si H es matriz de chequeo de C, entonces**

$$\delta(C) = \min j : \exists \textbf{ un conjunto de } j \textbf{ columnas LD de } H$$

(LD es linealmente dependiente)

14) **(Fundamental de código ciclico) Sea C un código ciclico de longitud n con generador $g(x)$ entonces:**

   1) $C = \{p(x) \in \mathbb{Z}_2(x) : gr(p) < n \wedge g(x)|p(x)\}$ **por esto se dice que C es generador (son los multiplicos de $g(x)$ de menor grado).**

   2) $C = \{v(x) \odot g(x) : v \in \mathbb{Z}_2(x)\}$ **son los multiplos de g modulares.**

   3) Si $k = Dim(C)$ entonces $gr(g) = n - k$.

   4) $g(x)|(1 + x^n)$**.**

   5) Si $g(x) = g_0 + g_1 x + \ldots +$ entonces $g_0 = 1$.

15) **Probar que 3SAT es NP-completo**

16) **Probar que $3-$COLOR es NP-completo**