Resumen de teoremas para el final de Matemática Discreta II

Agustin Curto, agucurto95@gmail.com

2016

Índice general

Capítulo 1

Parte A

1.1. La complejidad de EDMONS-KARP

Teorema: La complejidad de $\langle E - K \rangle$ con $n = |V|$ y $m = |E|$ es $\mathcal{O}(nm^2)$.

Prueba: Sean: $f_0, f_1, f_2, \ \ldots$ la sucesión de flujos creados por $\langle E - K \rangle$. Es decir, el paso k crea f_k.

Para cada k definimos funciones:

- $d_k(x) =$ "distancia" entre s y x en el paso k en caso de existir, si no ∞.
- $b_k(x) =$ "distancia" entre x y t en el paso k en caso de existir, si no ∞.

"Distancia": longitud del menor camino aumentante entre dos vértices.

Observaciónes:

1.
   - $d_k(s) = 0$
   - $b_k(t) = 0$
2. Sabemos que las distancias de $\langle E - K \rangle$ no disminuyen en pasos sucesivos, como esto será útil para esta demostración llamaremos $\circledast$ a la demostración de:

$$\begin{aligned} d_k(x) \leq d_{k+1}(x) \\ b_k(x) \leq b_{k+1}(x) \end{aligned}$$

Llamemos *crítico* a un lado disponible en el paso k pero no disponible en el paso *k+1*. Es decir, si xy es un lado $\Rightarrow$ xy se satura ó yx se vacía en el paso k.

Supongamos que al construir f_k el lado xy se vuelve crítico, el camino: $s \ldots$ x,y $\ldots$ t se usa para construir f_k.

$$\begin{aligned} d_k(t) &= d_k(x) + b_k(x) \qquad (1) \\ &= d_k(x) + b_k(y) + 1 \end{aligned}$$

Para que xy pueda ser *crítico* nuevamente debe ser usado en la otra dirección (*i.e* yx). Sea j el paso posterior a k en el cual se usa el lado en la otra dirección, el camino $s \cdots$ y,x $\cdots$ t se usa para construir f_j.

$$
\begin{aligned}
d_j(t) &= d_j(x) + b_j(x) \qquad (2)\\
&= d_j(y) + 1 + b_j(x)
\end{aligned}
$$

Entonces:

$$
\text{De (1) y (2)} \Rightarrow \begin{cases} d_j(x) = d_j(y) + 1 \ \star \\ d_k(y) = d_k(x) + 1 \ \dagger \end{cases}
$$

Luego:

$$
\begin{aligned}
d_j(t) &= d_j(x) + b_j(x) & \\
&= d_j(y) + 1 + b_j(x) & \text{Por } \dagger \\
&\geq d_k(y) + 1 + b_k(x) & \text{Por } \circledast \\
&= d_k(x) + 1 + 1 + b_k(x) & \text{Por } \star \\
&= d_k(t) + 2 & \\
\Rightarrow d_j(t) &\geq d_k(t) + 2 &
\end{aligned}
$$

Por lo tanto cuando un lado se vuelve crítico recien puede volver a saturarse cuando la distancia de s a t haya aumentado en por lo menos 2. Puede existir $\mathcal{O}(n/t)$ tales aumentos, es decir:

$$
\#\ \text{Veces que un lado puede volverse crítico} = \mathcal{O}(n).
$$

$$
\begin{aligned}
\therefore Complejidad(\langle E-K\rangle) &= (\#pasos) * Compl(1\ paso) \\
&= (\#\text{veces que un lado se vuelve crítico}) * (\#lados) * Compl(BFS) \\
&= \mathcal{O}(n) * \mathcal{O}(m) * \mathcal{O}(m) \\
&= \mathcal{O}(nm^2)
\end{aligned}
$$

1.2. Las distancias de Edmonds-Karp no disminuyen en pasos sucesivos

Teorema: Sean: $f_0, f_1, f_2, \ \ldots$ la sucesión de flujos creados por $\langle E-K\rangle$. Es decir, el paso k crea f_k.

Para cada k definimos funciones:

- $d_k(x) =$ "distancia" entre s y x en el paso k en caso de existir, si no ∞.
- $b_k(x) =$ "distancia" entre x y t en el paso k en caso de existir, si no ∞.

"Distancia": longitud del menor camino aumentante entre dos vértices.

Queremos probar que:

1. $d_k(x) \leq d_{k+1}(x)$

2. $b_k(x) \leq b_{k+1}(x)$

Prueba: Lo probaremos por inducción y solo para d_k ya que para b_k la prueba es análoga.

$$\text{HI: } H(i) = \{\forall_z : d_{k+1}(z) \leq i, \textit{ vale } d_k(z) \leq d_{k+1}(z)\}$$

1. Caso Base: $\boxed{\text{i} = 0}$ $\quad H(0) = \{\forall_z : d_{k+1}(z) \leq 0,\ d_k(z) \leq d_{k+1}(z)\}$

   Pero $d_{k+1}(z) \leq 0 \Rightarrow z = s$

$$\begin{aligned} d_k(z) &= d_k(s) \\ &= 0 \\ &\leq d_{k+1}(s) \\ &\leq d_{k+1}(z) \\ \Rightarrow d_k(z) &\leq d_{k+1}(z) \end{aligned}$$

2. Caso Inductivo: Supongamos ahora que vale H(i), veamos que vale H($)i+1$.

   Sea z con $d_{k+1}(z) \leq i{+}1$, si $d_{k+1}(z) \leq i$ vale H(i) para z.

$$\therefore\ d_{k+1}(z) \leq d_{k+1}(z)$$

   Supongamos que $\boxed{d_{k+1}(z) = i + 1}$

   Entonces existe un camino aumentante, relativo a f_k, de la forma: $s = z_0,\ z_1,\ \ldots\ z_i,\ z_{i+1} = z$.

   Sea $\boxed{x = z_i}$

   - Caso 1: Existe algun camino aumentante, relativo a f_{k-1} de la forma $s,\ \ldots\ x,\ z$. $\Rightarrow \boxed{d_k(x) \leq d_k(x) + 1}$

     Pues al haber un camino $\underbrace{s,\ \ldots\ x,}_{d_k(x)}$ z, llamemosle A, de longitud $d_k(x) + 1$ entre s y z, sabemos que el minimo de todos los caminos de s a z seran $\leq$ A.

   - Caso 2: No existe un camino aumentante, relativo a f_{k-1}, pero si existe un camino aumentante relativo a f_k. Por lo tanto el lado xz no esta "disponible" en el paso k, ya que xz está saturado zx está vacío relativo a f_{k-1}. Para construir f_k usamos un camino de la forma $s,\ \ldots\ z,\ x$. Es decir:

     1) $f_{k-1}(xz) = C(xz)$ pero $f_k(xz) < C(xz),\ f_k$ devuelve flujo por xz ó
     2) $f_{k-1}(zx) = 0$ pero $f_k(zx) > 0,\ f_k$ manda flujo por zx.

     Como $\langle E - K \rangle$ funciona con BFS, ese camino usado pra construir f_k debe ser de longitud mínima. Es decir: $\quad d_k(x) = d_k(z) + 1$

$$\begin{aligned} d_k(z) &= d_k(x) - 1 \\ &\leq d_k(x) + 1 \end{aligned}$$

   Conclusión: En cualquiera de los dos casos tenemos:

$$\boxed{d_k(x) \leq d_k(x) + 1}$$

Ahora bien: $d_{k+1}(x) = d_{k+1}(z_i) = i \quad \Rightarrow \quad$ vale H(i) para x.
$\therefore d_k(z) \leq d_{k+1}(x)$

$$\begin{aligned} d_{k+1}(x) &= d_{k+1}(z_i) \\ &= i \\ &\Rightarrow \text{H}(i) \text{ vale para x.} \\ \therefore\ d_k(z) &\leq d_{k+1}(x) \end{aligned}$$

Por lo tanto:

$$\begin{aligned} d_k(z) &\leq d_k(x) + 1 \\ &\leq d_{k+1}(x) + 1 \\ &= i + 1 \\ &= d_{k+1}(z) \\ &\Rightarrow \text{H}(i{+}1) \text{ vale.} \end{aligned}$$

1.3. La complejidad de DINIC

Teorema: La complejidad del algoritmo de Dinic es $\mathcal{O}(n^2 m)$.

Prueba:Vimos que la distancia entre s y t en networks auxiliares consecutivos aumenta y puede ir a lo sumo entre 1 y $n-1$ entonces hay a lo sumo $\mathcal{O}(n)$ networks auxiliares.

Complejidad(Dinic)= $\mathcal{O}(n)*$ Compl(Hallar un flujo bloqueante en un NA con Dinic)

Para probar que la complejidad de Dinic es $\mathcal{O}(n^2 m)$ debemos probar que complejidad del paso bloqueante es $\mathcal{O}(nm)$.

Sea:

- A = Avanzar()
- R = Retroceder()
- I = Incrementar_Flujo + Inicialización ($\mathcal{O}(1)$)

Una corrida de Dinic luce como:

AA ... AIAAARA ... AIAARAAARR ... IA ...

Dividamos la corrida en subpalabras del tipo:

$* \underbrace{AA\ \dots\ A}_{TodasA's} I$

$* \underbrace{AA\ \dots\ A}_{TodasA's} R$

Nota: el número de A's puede ser 0.

Debemos determinar:

1. Cual es la complejidad de cada subpalabra.
2. Cuantas palabras hay de cada tipo.

Complejidad de cada subpalabra
Recordemos que:

$$\text{A:}\left[\begin{array}{l} P[i+1] = \text{algún elemento de } \Gamma^{+}(P[i]) \\ i = i+1 \end{array}\right.$$

$$\Rightarrow \text{A es } \mathcal{O}(1)$$

$$\text{R:}\left[\begin{array}{l} P[i+1] = \text{borrar } P[i-1]P[i] \text{ del NA} \\ i = i-1 \end{array}\right.$$

$$\Rightarrow \text{R es } \mathcal{O}(1)$$

$$\text{I:}\left[\begin{array}{l} P[i+1] = \text{recorre 2 veces, un camino de longitud } d = d(t) \end{array}\right.$$

$$\Rightarrow \text{I es } \mathcal{O}(d)$$

Por lo tanto:

$$\begin{aligned} Compl(\underbrace{A\ \ldots\ A}_{j\ veces}R) &= \underbrace{\mathcal{O}(1) +\ \ldots\ \mathcal{O}(1)}_{j\ veces} + \mathcal{O}(1) \\ &= \mathcal{O}(j) + \mathcal{O}(1) \\ &= \mathcal{O}(j) \end{aligned}$$

Pero como cada A hace $i = i+1$ y tenemos $0 \leq i \leq d \ \Rightarrow\ j \leq d$.

$$\therefore\ Compl(A\ \ldots\ AR) = \mathcal{O}(d)$$

Similarmente:

$$\begin{aligned} Compl(A\ \ldots\ AI) &= \underbrace{\mathcal{O}(1) +\ \ldots\ \mathcal{O}(1)}_{\leq d\ veces} + \mathcal{O}(1) \\ &= \mathcal{O}(d) + \mathcal{O}(1) \\ &= \mathcal{O}(d) \end{aligned}$$

Cantidad de subpalabras

- R tiene la instrucción "**borrar lado**". Como los lados borrados quedan borrados hay a lo sumo m R's, es decir:

$$\therefore\ \#(A\ \ldots\ AR's) \leq m$$

- I tiene también línes de la forma:

  if ... **then**
      borrar lado
  end if

  Lo que está dentro del **if** se cumple al menos una vez, es decir:

$$\therefore\ \#(A\ \ldots\ AI's) \leq m$$

  Este análisis muestra que:

$$\therefore \ \#(A \ \ldots \ AR's) + \#(A \ \ldots \ AI's) \leq m$$

Por lo tanto hay $\leq m$ palabras, cada una de complejidad $\mathcal{O}(d)$.

$$\begin{aligned} \therefore \ Compl(Paso\ Bloqueante) &= \mathcal{O}(m) + \mathcal{O}(md) \\ &= \mathcal{O}(mn) \end{aligned}$$

ya que $d \leq n$.

1.4. La complejidad de WAVE

1.5. La distancia entre NA sucesivos aumenta

Capítulo 2

Parte B

2.1. 2-COLOR es polinomial

2.2. Teorema Max-Flow Min-Cut

Teorema:

a) Si f es flujo y S es corte $\Rightarrow$ V(f) $\leq$ Cap(S).

b) Si V(f) = Cap(S) $\Rightarrow$ f es maximal y S es minimal.

c) Si f es maximal $\Rightarrow$ $\exists$ S con V(f) = Cap(S).

Prueba: Demostraremos primero que $V(f) = f(S, \overline{S}) - f(\overline{S}, S)$ donde f es un flujo y S un corte. Esto nos ayudará en la demostración del ítem a.

Observemos que:

- $f(A \cup B, C) = f(A, C) + f(B, C)$: A y B disjuntos.
- $f(A, B \cup C) = f(A, B) + f(A, C)$: B y C disjuntos.
- $f(A, B) = \sum\limits_{\substack{x \in A \\ y \in B}} f(x, y)$.

Sea x $\in$ S $\Rightarrow$ x $\neq$ t.

$$\text{f(x, V) - f(V, x)} = \begin{cases} V(f) & Si\ x = s \\ 0 & Si\ x \neq s\ pues\ t \notin S \end{cases}$$

Luego:

$$\sum_{x \in S} (f(x, V) - f(V, x)) = 0 + 0 \cdots + V(f) = V(f) \tag{1}$$

$$V(f) = \sum_{x \in S} f(x, V) - \sum_{x \in S} f(V, x) \tag{2}$$

$$= f(S, V) - f(V, S) \tag{3}$$

$$= f(S, S \cup \overline{S}) - f(S \cup \overline{S}, S) \tag{4}$$

$$= f(S, S) + f(S, \overline{S}) - f(S, S) - f(\overline{S}, S) \tag{5}$$

a) $\mathbf{V}(f) \leq \mathbf{Cap(S)}$

2.3. Complejidad del Hungaro es $\mathcal{O}(n^4)$

Teorema: La complejidad del algoritmo Húngaro es $\mathcal{O}(n^4)$.
Prueba:

1. La complejidad del matching inicial es $\mathcal{O}(n^2)$, ya que:

   Restar mínimo de cada fila:

   $$(\mathcal{O}(n^2) + \mathcal{O}(n^2)) * n = \mathcal{O}(n^2) \text{ Idem para las columnas.}$$

2. Llamemos **extender** el matching, a incrementar su número de filas en 1, i.e agregar una fila más al matching.

   $$\#\ extensiones\ de\ matching = \mathcal{O}(n)$$

   Resta ver la complejidad de cada **extender**.

3. En cada extensión vamos a ir revisando filas y columnas, donde escanear una fila es $\mathcal{O}(n)$ y se realizan n escaneos, por lo que sería $\mathcal{O}(n^2)$ sin considerar que se debe realizar un cambio de matriz.

   Hacer un cambio de matriz es $\mathcal{O}(n^2)$.

   - Buscar $m = \min S\ x\ \overline{\Gamma(S)} \rightarrow \mathcal{O}(n^2)$
   - Restar m de $S \rightarrow \mathcal{O}(n^2)$
   - Sumar m a $\Gamma(S) \rightarrow \mathcal{O}(n^2)$

   Luego la implementación NAIVE lanzaría nuevamente el algoritmo desde cero. La forma correcta es continuar con el matching que teniamos, ya que el mismo no se pierde.

   $$\begin{bmatrix} A & A \\ B & C \end{bmatrix}$$

   TO DO

   Debemos ver cuantos Cambios de matriz hay antes de extender nuevamente un matching

   Lema Interno: Luego de un cambio de matriz, se extiende el matching (i.e se termina el **extender**), o bien se aumenta S.

   Prueba:

   $$\begin{bmatrix} A & A \\ B & C \end{bmatrix}$$

   Al restar $m = \min S\ \Gamma(S)$ de las filas de S, habrá un nuevo cero en alguna fila i ($\in S$) y columna j ($\in \Gamma(S)$) entonces la columna se etiquetará con i y se revisará. Tenemos dos resultados posible:

   a) j está libre (i.e no forma parte del matching) $\Rightarrow$ extendemos el matching.

b) j forma parte de matching $\Rightarrow \exists$ fila k matcheada con j. En este caso, la fila k se etiquetará con j, por lo que el "nuevo" $S \geq S \cup \{k\}$.

Entonces se termina con una extensión o se produce un nuevo S de cardinalidad, al menos $|S| + 1$.

Fin lema interno

Luego como $|S|$ solo puede crecer $\mathcal{O}(n)$ veces, tenemos que hay a lo sumo n **cambios de matriz** antes de extender el matching. Entonces:

$$\text{Complejidad(1 Extensión)} = \underbrace{\mathcal{O}(n)}_{\#CM} * \underbrace{\mathcal{O}(n^2)}_{Compl(CM)} + \underbrace{\mathcal{O}(n^2)}_{Busqueda\ n\ filas\ x\ n\ columnas}$$

$$\therefore \text{Complejidad(Húngaro)} = \underbrace{\mathcal{O}(n^2)}_{Matchinginicial} + \underbrace{\mathcal{O}(n)}_{\#extensiones} * \underbrace{\mathcal{O}(n^3)}_{Compl(extension)}$$

2.4. Teorema de Hall

Teorema: Sea G = (x ∪ y, E) grafo bipartito ⇒ ∃ matching completo de X a Y ⇔ $|S| = |\Gamma(S)| \forall S \subseteq X$.

Prueba:

⇒) Si M es matching comple de X a Y entonces oberservemos que M induce una función inyectiva de X a Y.

$$f(x) = \text{único y : xy} \in \text{M}.$$

1. Si $S \subseteq X \Rightarrow |S| = |\Gamma(S)|$.

   Además por definición de f, $f(x) \in \Gamma(x)$.

2. Si $x \in S \Rightarrow f(x) \in \Gamma(S) \Rightarrow f(S) \subseteq \Gamma(S)$.

De 1 y 2 $\Rightarrow |S| \leq |\Gamma(S)|$.

⇐) Supongamos que no es cierto, entonces G es bipartito con $|S| \leq |\Gamma| \ \forall \ S \subseteq X$ pero no tiene matching completo de X a Y. Es equivalente a ver que: Si ∄ un matching completo $\Rightarrow \exists \ S \subseteq X : |S| > |\Gamma(S)|$.

Corramos el algoritmo para hallar matching. Al finalizar habrá filas sin matcher (las de s). Sean:

-

2.5. Teorema del matrimonio

2.6. Si G es bipartito $\Rightarrow \chi'(G) = \Delta$

2.7. Teorema cota de Hamming

2.8. Sea H una matriz de chequeo de un código C, pruebe que:

2.8.1. $\delta(C) =$ mínimo número de columnas linealmente dependientes de H

2.8.2. Si H no tiene la columna cero ni columnas respetidas $\Rightarrow$ C corrige al menos un error

2.9. Sea C un código cíclico de dimensión k y longitud n y sea $g(x)$ su polinomio generador, probar que:

2.9.1. C está formado por los múltiplos de $g(x)$ de grado menor a n

2.9.2. El grado de $g(x)$ es $n - k$

2.9.3. $g(x)$ divide a $1 + x^n$

Capítulo 3

Parte C

3.1. 4-COLOR $\leq_p$ SAT

3.2. 3-SAT es NP-Completo

3.3. 3-COLOR es NP-Completo

Bibliografía

[1] Curto Agustín , «Matemática Discreta II, apuntes de clase», *FaMAF, UNC*.

[2] Maximiliano Illbele, «Resumen de Discreta II, 16 de agosto de 2012», *FaMAF, UNC*.