MA3705. Algoritmos Combinatoriales 2020.
Profesor: José Soto
Escriba(s): Guillermo Cardenas y Cynthia Vega.
Fecha: 9 de octubre de 2020.

Cátedra 10

1. Paseos de largo mínimo con k aristas: PD

Motivación: Estamos en búsqueda de calcular en grafos dirigidos, paseos desde un punto a otro, claramente optimizando su largo. Esto se traduce a encontrar el paseo de largo mínimo entre un vértice y otro.

Para abordar este problema, buscaremos la reducción a un problema más simple, transformando lo anterior en buscar un paseo de largo mínimo con exactamente k arcos. Esta última condición es restrictiva, lo cual acota nuestras opciones y produce el efecto buscado de simplificar un problema tan amplio como el original anteriormente descrito.
En resumen, minimizamos paseos con k arcos y luego minimizamos sobre k para resolver el problema.

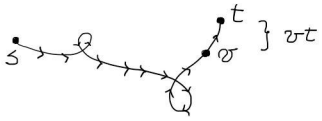
1.1. Paseos con exactamente k arcos.

- $G = (V, E)$ grafo **dirigido**
- $\mathcal{W}_{=k}(s,t)$: Paseos (dirigidos) de s a t con exactamente k arcos.
- Dado $\ell : E \to \mathbb{R}$, $d^{\mathcal{W}}_{=k}(s,t) = \min\{\ell(W) : W \in \mathcal{W}_{=k}(s,t)\}$

Teorema 1. *Sea $W \in \mathcal{W}_{=k}(s,t)$ con $d^{\mathcal{W}}_{=k}(s,t) = \ell(W)$ y $k \geq 1$, vt el último arco de W y $W' = W - vt$ entonces $\ell(W') = d^{\mathcal{W}}_{=k-1}(s,v)$*

Demostración

Idea: Imaginemos vt, el último arco del paseo W, como es el último en un paseo sobre el cual conocemos la cantidad de aristas (arcos) que posee (k aristas) es consistente que al definir $W' = W - vt$ este por definición tendrá exactamente una arista menos, luego se concluye la idea natural de proponer $\ell(W') = d^{\mathcal{W}}_{=k-1}(s,v)$



Veamos entonces que si tomamos un paseo $\mathcal{R} \in \mathcal{W}_{=k-1}(s,v)$ este podrá ser descrito a través de; $\ell(\mathcal{R}) = d^{\mathcal{W}}_{=k-1}(s,v)$.

Recordamos que por la definición de W' tenemos una primera desigualdad (1) $\ell(W') \geq d^{\mathcal{W}}_{=k-1}(s,v)$, con esto basta demostrar entonces la desigualdad hacia el otro lado

Para esto aprovecharemos el comportamiento lineal del operador ℓ, de la siguiente manera:

Primero notamos que, como ℓ tiene comportamiento lineal al separar la suma se tiene la igualdad:

$$\ell(\mathcal{R}) = \ell(\mathcal{R} + vt) - \ell(vt)$$

Donde aprovechamos la definición de W para notar que $\mathcal{R} + vt \in \mathcal{W}_{=k}(s,t)$ pues describe un paseo de s hasta v y continúa hasta t, lo cual nos dice que codifica un paseo de s a t, así mismo recordamos que W era nuestro paseo optimal, esto lo usaremos para acotar como sigue:

$$\ell(\mathcal{R}) = \ell(\mathcal{R} + vt) - \ell(vt)\ \ell(\mathcal{R}) \geq \ell(W) - \ell(vt) = \ell(W')$$

Donde la última linea la obtenemos de que W' lo definimos como $W - vt$. De esta manera obtenemos la desigualdad deseado, logrando el siguiente resultado:

$$\ell(\mathcal{W}') = \ell(\mathcal{R}) = d^{\mathcal{W}}_{=k-1}(s,v) \quad \blacksquare$$

1.2. Consecuencias: subpaseos óptimos

Teorema 2. *Sea $W \in \mathcal{W}_{=k}(s,t)$ con $d^{\mathcal{W}}_{=k}(s,t) = \ell(W)$ y $k \geq 1$, sv el primer arco de W y $W' = W - sv$ entonces $\ell(W') = d^{\mathcal{W}}_{=k-1}(s,v)$*

Es una buena observación notar que este segundo teorema tiene una forma equivalente a lo demostrado anteriormente, salvo, que para la versión anterior quitábamos el arco final y ahora removemos el arco del principio. Esto nos motiva a la selección de sub-paseos los cuales se puedan asegurar que tienen largo óptimo (mínimo de largo k).

Corolario 1 (subpaseos tienen largo mínimo (para números de arcos fijo))**.** *Sea $W \in \mathcal{W}_{=k}(s,t)$ con $d^{\mathcal{W}}_{=k}(s,t)$ y $k \geq 1$, sv el primer arco de W, $W' = W - sv$ y W' sub paseo de W de a a b, entonces $\ell(W') = d^{\mathcal{W}}_{=|W'|}(a,b)$*

1.3. Algoritmo para calcular $d^{\mathcal{W}}_{=k}(s,t)$

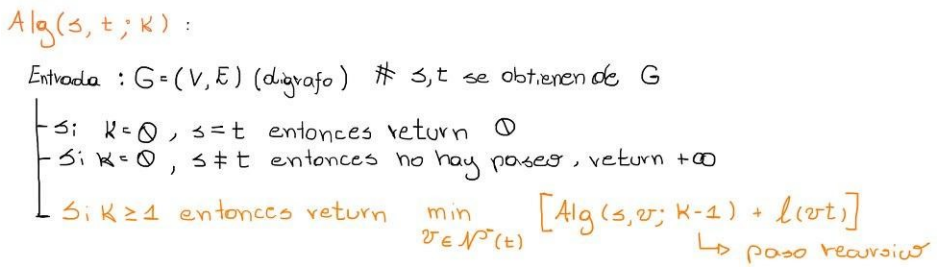
Algoritmo (s,t; k):

- **Entrada:** $G = (V,E)$ **digrafo,** $s,t \in V, \quad k,l : E \to \mathbb{R}$
- **If** $k = 0, s = t$ **Entonces devolver 0 (largo vacío = 0)**
- **If** $k = 0, s \neq t$ **Entonces, devolver ∞, esto se desarrolla por convención de un arco que no existe.**
- **If** $k \geq 1$ **Ejecutar algoritmo.**

Con esta estructura podemos imaginar la forma general de como será nuestro algoritmo, sin embargo primero revisaremos cuales son las posibles maneras en las cuales lo podemos codificar, para obtener un algoritmo igual de correcto como óptimo.

Posible algoritmo

Un primer acercamiento es a través de la recursividad, pues es lo más natural si apuntamos a usar los teoremas descritos anteriormente, sin embargo uno de los problemas que tiene la traducción de los teoremas 1 y 2 a código es la ausencia de información o más bien, el desconocimiento que tenemos del penúltimo vértice, es por esto que la manera recursiva soluciona esta carencia de información, mas, veremos que es poco óptima y suma demasiada complejidad a un algoritmo que podríamos tratar desde otra perspectiva.



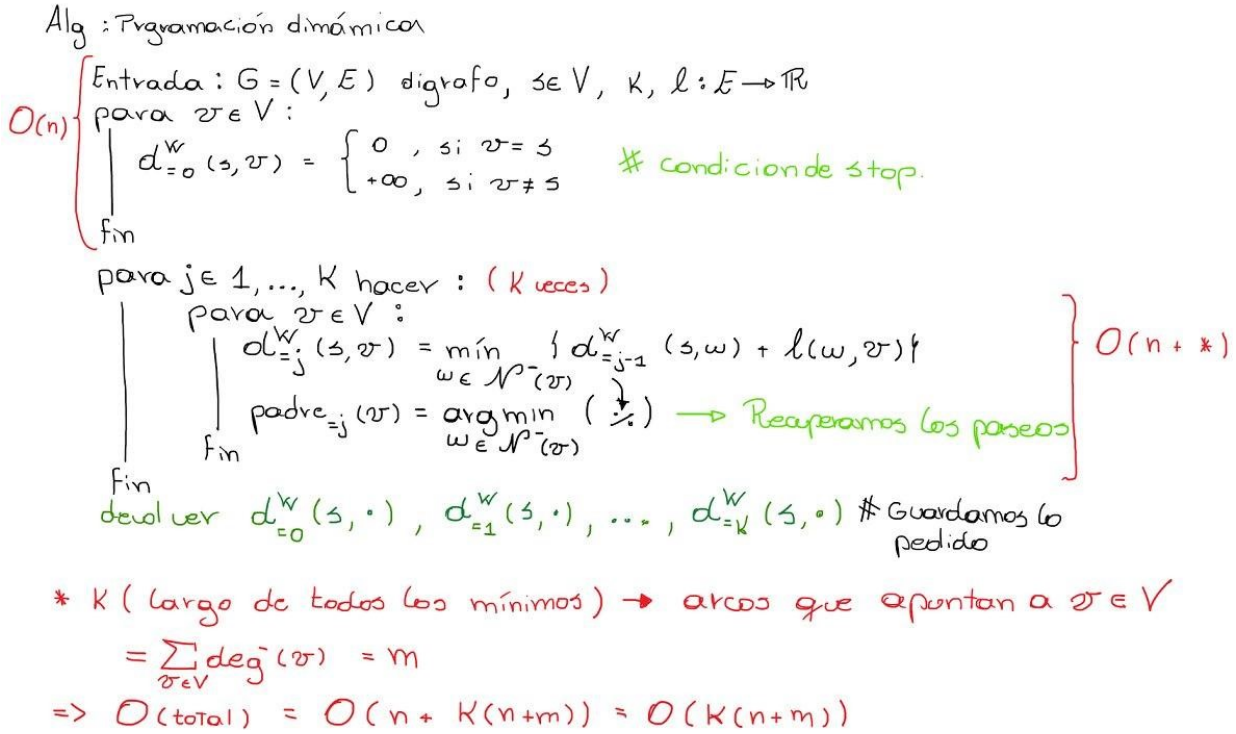
Correctitud del algoritmo: En efecto es necesario estudiarla, veamos primero si finaliza y efectivamente nos podemos dar cuenta de que estamos estableciendo recursión sobre k, parámetro el cual tiene bien definidas sus condiciones de término $k = 0$ dentro del mismo algoritmo. Además podemos asegurar el la correctitud en términos de finalización de la iteración pues como se menciona anteriormente, k en algún momento se hace 0

El problema principal que presenta este algoritmo recursivo es que no se aprovecha información potencial en las iteraciones. Es por esto que se propone abordarlo desde la programación dinámica como se verá a continuación.

Teorema 3. *si $k \geq 1$, $d^{\mathcal{W}}_{=k}(s,t) = \min_{v \in N^-(t)}\{d^{\mathcal{W}}_{=k-1}(s,v) + \ell(vt)\}$*

Mejor algoritmo; Con programación Dinámica

El objetivo de este algoritmo es re-utilizar información ya calculada en interaciones anteriores, de esta manera evitaremos la recursividad. Esto aumenta su eficiencia de manera considerable pues estamos evitando calcular resultados que se presentan hasta más de dos veces. Con el algoritmo que se presenta a continuación construiremos una tabla sobre la cual se irán registrando los resultados de cada iteración del algoritmo.



Explicación Algoritmo

Notemos las diferencias a grandes rasgos de nuestro algoritmo con programación dinámica y nuestra primera versión del algoritmo iterativo. En esta segunda versión mantenemos la condición que mantiene la correctitud (pues k eventualmente se hace 0 al igual que en la versión iterativa), sin embargo, cambia radicalmente en la estructura de ejecución misma al tomar $k \geq 1$.

Recordamos que uno de los principales objetivos de este algoritmos es la reutilización de los datos ya recolectados en las ejecuciones previas, que se representan en la imagen al final del algoritmo con *devolver:* ${d_{=0}}^{\mathcal{W}}(s,\cdot)$; ${d_{=1}}^{\mathcal{W}}(s,\cdot)$; $\cdots$; ${d_{=k}}^{\mathcal{W}}(s,\cdot)$ Además, nos es de suma utilidad guardar también de manera dinámica los paseos y no solo sus largos, de esta manera podremos recuperar cuando necesitemos, cual fué el paseo usado de largo mínimo.

Complejidad

Para calcular el orden de este algoritmo, como se muestra en la imagen anterior debemos analizarlo en "dos mitades", la mitad superior, la cual tiene Orden 1 para cada nodo lo que nos lleva a concluir que tiene orden $O(n)$ y la mitad inferior, la cual es un poco más compleja de analizar:

Desde una primera perspectiva podemos tener inmediatamente un multiplicador k correspondiente al ciclo "para". Haciendo un conteo rápido resaltamos que para cada vértice debemos calcular un mínimo el cual, con un análisis rápido nos da la intuición de que en el peor caso tendríamos que leer n datos, lo que nos lleva a inferir un orden de la sección inferior de $O(n^2)$

Esto nos da un orden en primera instancia del algoritmo igual a $O(n + kn^2)$

Sin embargo tenemos una segunda visión para analizar este algoritmo; se propone la siguiente manera de ver la mitad inferior, pensemos que estamos haciendo al calcular los mínimos, y estos a su vez se calculan dentro de un conjunto que podemos simplificar llamándolo lista o array para efectos prácticos de algoritmos; entonces, vemos cada elemento de la lista "visitandoüna vez cada elemento, es así como nos motiva a proponer que el orden de la mitad inferior estaría más relacionado con la suma de los largos de todas las listas, lo cual finalmente quedaría expresado como:

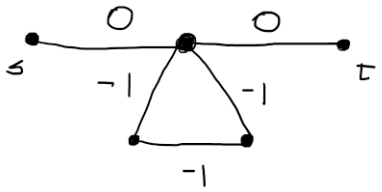
$$O(n + k \cdot \textit{largo total de todas las listas de minimos}) = O(n + \sum_{v \in V} deg^-(v))$$

Donde $deg^-(v)$ representa cada arco que apunta a cada nodo, lo que traduce la expresión completa a:

$$O(n + \sum_{v \in V} deg^-(v)) = O(n + k \cdot (n + m) = O(k \cdot (n + m))$$

Ciclos de largo negativo

Mas que largo, podemos interpretar el concepto propuesto como un coste negativo o un estado dentro de un proceso, aquí si tendría sentido definir un estado o coste negativo que lo podríamos interpretar como un largo negativo. Naturalmente queremos minimizar la pérdida y para esto podemos ver el siguiente ejemplo:



2. Largos Conservativos

Una definición rápida aprovechando el concepto anterior refiere a un largo conservativo como una función de largo que va de los arcos a los reales en la cual no hay ciclos (dentro del grafo) con largos negativos.

Ser conservativo significa no tener ciclos de largo negativo (en resumen)

2.1. Definiciones en largos conservativos para digrafo $G = (V, E)$

Definición 1. $\ell : E \to \mathbb{R}$ es *conservativa* si no hay ciclos en $G = (V, E)$ con largo negativo.

$\mathcal{W}_{\leq k}\{\ W$: paseo con $\leq k$ arcos $\}$		$\mathcal{P}_{\leq k}\{\ P$: camino con $\leq k$ arcos $\}$	
$d^{\mathcal{W}}_{\leq k}(s,t) = \min_{W \in \mathcal{W}_{\leq k}} \ell(W)$	$d^{\mathcal{W}}(s,t) = \min_{W \in \mathcal{W}} \ell(W)$	$d^{\mathcal{P}}_{\leq k}(s,t) = \min_{W \in \mathcal{P}_{\leq k}} \ell(W)$	$d^{\mathcal{W}}(s,t) = \min_{W \in \mathcal{W}_{\leq k}} \ell(W)$

Caminos tienen igual poder que paseos

$d^{\mathcal{W}}_{\leq k}(s,t) = d^{\mathcal{P}}_{\leq k}(s,t)$ y $d^{\mathcal{W}}(s,t) = d^{\mathcal{P}}(s,t)$

Demostración

(1) $d^{\mathcal{W}}_{\leq k}(s,t) \leq d^{\mathcal{P}}_{\leq k}(s,t)$

Los paseos contienen a los caminos entonces, si tomo el mínimo del conjunto de caminos, será menor al mínimo del conjunto de paseos;

Esto nos lleva a ver lo siguiente;

$$\mathcal{P}_{\leq k}(s,t) \subset \mathcal{W}_{\leq k}(s,t)$$

tomando mínimo a cada lado y aplicando ℓ podemos ver

$$\min_{W \in \mathcal{W}_{\leq k}(s,t)} \ell(W) \leq \min_{P \in \mathcal{P}_{\leq k}(s,t)} \ell(P)$$

Lo cual por definición nos lleva a la primera desigualdad

$$d^{\mathcal{W}}_{\leq k}(s,t) \leq d^{\mathcal{P}}_{\leq k}(s,t)$$

Para la segunda desigualdad nos tomaremos, de todos los paseos que alcanzan el mínimo tomemos $\mathcal{R}$ el paseo que alcanza $d^{\mathcal{W}}_{\leq k}(s,t)$ con menos arcos, llegaríamos a que $\mathcal{R}$ realmente es un camino, pues si no lo fuera (suponiendo contradicción) podríamos ver que eventualmente $\mathcal{R}$ repite un vértice generando un ciclo el cual llamaremos C. Si $\mathcal{R}$ repite vértices entonces podemos construir $\mathcal{R}' = \mathcal{R} - C$ y tendríamos que el largo $\ell(\mathcal{R}') = \ell(\mathcal{R}) - \ell(C)$, donde por hipótesis tenemos que C tiene largo no negativo, lo que contradice la condición de mínimo de $\mathcal{R}$, concluyendo con que $\mathcal{R}$ es realmente un camino y como es un camino la distancia de s a t usando caminos es menor a la distancia usando paseos. $\Rightarrow d^{\mathcal{P}}_{\leq k}(s,t) \leq d^{\mathcal{W}}_{\leq k}(s,t)$

2.2. Consecuencias

Si ℓ es conservativa entonces podermos escribir:

$$\begin{aligned} d_{\leq k}(s,t) &= d^{\mathcal{W}}_{\leq k}(s,t) = d^{\mathcal{P}}_{\leq k}(s,t) \\ d(s,t) &= d^{\mathcal{W}}(s,t) = d^{\mathcal{P}}(s,t) \\ d(s,t) &= d_{\leq n-1}(s,t) \end{aligned}$$

Más aún $d(\cdot,\cdot)$ es una *métrica de grafos* (no es métrica en el sentido habitual):

1. $d(s,t)$ puede ser negativa.
2. $d(v,v)$ pero $d(u,v) = 0 \nRightarrow u = v$.
3. $d(s,t)$ no es necesariamente igual a $d(t,s)$.
4. **Desigualdad triangular:** $d(s,t) \leq d(s,v) + d(v,t)$.

2.3. Desigualdad Triangular

Dos resultados importantes de la desigualdad triangular son los siguientes

- Desigualdad triangular no se tiene (necesariamente) si ℓ no es conservativa
- Desigualdad triangular implica Principio de Optimalidad de Bellmam

Teorema 4 (Si ℓ es conservativo)**.** *Sea P camino de s a t con $\ell(P) = d(s,t)$. Sea P' subcamino de P (de a a b), entonces $\ell(P') = d(a,b)$.*

2.4. Consecuencias

Teorema 5 (Si ℓ es conservativo)**.**

$$d(s,t) = \min_{w \in N^-(t)} d(s,w) + \ell(w,t)$$

$$d_{\leq k}(s,t) = \min\left(d_{\leq k-1}(s,t), \min_{w \in N^-(t)} d_{\leq k-1}(s,w) + \ell(w,t) \right)$$