CM todos a todos y DAG's

Ezequiel Companeetz

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

1er Cuatrimestre de 2024

Programa de hoy

1. Aclaraciones importantes
2. Repaso de definiciones y algoritmos
3. Aplicaciones de CM y DAG
4. Ejercicios introductorios
5. Ejercicios CM todos a todos
6. Conclusiones

Situación docente

Computación - Docentes (Ay1 y JTP) - Salarios + Alumnos = Situación Actual

Buenas noticias

¡Cifu is back! Una alegría para TDA

Problemas de CM

Problemas a modelar

- Camino mínimo entre dos vértices
- Camino mínimo entre un vértice y el resto de vértices
- Camino mínimo entre todos los pares de vértices

¿Qué algoritmos resuelven estos problemas?

Problemas de CM

Problemas a modelar

- Camino mínimo entre dos vértices
- Camino mínimo entre un vértice y el resto de vértices
- Camino mínimo entre todos los pares de vértices

¿Qué algoritmos resuelven estos problemas?

Variantes de cada problema

- Grafos con aristas no dirigidas
- Digrafo sin pesos en las aristas
- Digrafo con/sin aristas de peso negativo
- Digrafo con/sin ciclos de peso negativo.

Algoritmos de CM

Variables a tener en cuentas

- Cada algoritmo se puede medir de muchas maneras distintas:
    - Complejidad temporal
    - Complejidad espacial
    - Implementación sencilla o compleja
    - ¿Calcula información parcial útil? ¿Cuál es su invariante? (i.e. Qué pasa si modifico el grafo luego de haber resuelto el problema?)

Algoritmos de la clase de hoy

- Dijkstra (uno contra todos)
- Floyd-Warshall (todos contra todos)
- Dantzig (todos contra todos)

Repaso de propiedades

Principio de optimalidad

Todo sub camino de un camino mínimo es un camino mínimo, siempre que no haya ciclos de pesos negativos.

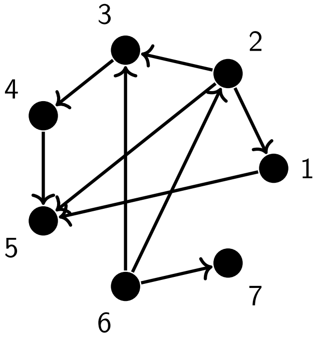
Árbol de caminos mínimos

Sea G un grafo, s un vértice de G y G no posee aristas de costo negativo alcanzables desde s. Sea W el conjunto de vértices alcanzables desde s en G. Entonces existe un árbol orientado T con raíz s y $V(T) = W$, tal que para todo $v \in W$, el camino de s a v en T es un camino mínimo de s a v en G.

DAGs

Definición

Un digrafo D es un DAG (*directed acyclic graph*) si no tiene ciclos dirigidos.

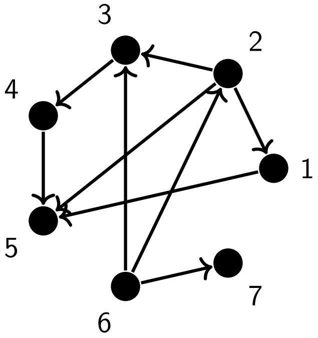


¿Cómo puedo saber si un grafo es un DAG?

DAGs

Definición

Un digrafo D es un DAG (*directed acyclic graph*) si no tiene ciclos dirigidos.



¿Cómo puedo saber si un grafo es un DAG? DFS o BFS, en $O(n+m)$.

DAGs y orden topológico

Orden Topológico

Definimos al orden topológico (o también conocido cómo *orden de actualización seguro inverso*) de un digrafo $D = (V, E)$ como un ordenamiento de los nodos $v_1 v_2 \dots v_n$ tal que para todo eje $v_i v_j \in E$ vale que $i < j$.

¿Cuál sería un orden topológico para nuestro DAG anterior?

DAGs y orden topológico

Orden Topológico

Definimos al orden topológico (o también conocido cómo *orden de actualización seguro inverso*) de un digrafo $D = (V, E)$ como un ordenamiento de los nodos $v_1 v_2 \dots v_n$ tal que para todo eje $v_i v_j \in E$ vale que $i < j$.
¿Cuál sería un orden topológico para nuestro DAG anterior? ¿Este es válido $v_6, v_7, v_2, v_3, v_4, v_5, v_1$? ¿Y este $v_6, v_7, v_2, v_3, v_4, v_1, v_5$?

- ¿Todo digrafo tiene un orden topológico?

DAGs y orden topológico

Orden Topológico

Definimos al orden topológico (o también conocido cómo *orden de actualización seguro inverso*) de un digrafo $D = (V, E)$ como un ordenamiento de los nodos $v_1 v_2 \ldots v_n$ tal que para todo eje $v_i v_j \in E$ vale que $i < j$.
¿Cuál sería un orden topológico para nuestro DAG anterior? ¿Este es válido $v_6, v_7, v_2, v_3, v_4, v_5, v_1$? ¿Y este $v_6, v_7, v_2, v_3, v_4, v_1, v_5$?

- ¿Todo digrafo tiene un orden topológico?
- ¿Todo digrafo que admita un orden topológico es un DAG?

DAGs y orden topológico

Orden Topológico

Definimos al orden topológico (o también conocido cómo *orden de actualización seguro inverso*) de un digrafo $D = (V, E)$ como un ordenamiento de los nodos $v_1 v_2 \dots v_n$ tal que para todo eje $v_i v_j \in E$ vale que $i < j$.
¿Cuál sería un orden topológico para nuestro DAG anterior? ¿Este es válido $v_6, v_7, v_2, v_3, v_4, v_5, v_1$? ¿Y este $v_6, v_7, v_2, v_3, v_4, v_1, v_5$?

- ¿Todo digrafo tiene un orden topológico?
- ¿Todo digrafo que admita un orden topológico es un DAG?
- ¿Todo DAG tiene un orden topológico? ¿Es único ese orden tópologico?

DAGs y orden topológico

Orden Topológico

Definimos al orden topológico (o también conocido cómo *orden de actualización seguro inverso*) de un digrafo $D = (V, E)$ como un ordenamiento de los nodos $v_1 v_2 \dots v_n$ tal que para todo eje $v_i v_j \in E$ vale que $i < j$.
¿Cuál sería un orden topológico para nuestro DAG anterior? ¿Este es válido $v_6, v_7, v_2, v_3, v_4, v_5, v_1$? ¿Y este $v_6, v_7, v_2, v_3, v_4, v_1, v_5$?

- ¿Todo digrafo tiene un orden topológico?
- ¿Todo digrafo que admita un orden topológico es un DAG?
- ¿Todo DAG tiene un orden topológico? ¿Es único ese orden tópologico?
- ¿Cuál es la máxima cantidad de ejes que puede tener un DAG de n nodos?

Problemas sobre DAGs

Sobre DAGs muchos problemas son fáciles, gracias al uso del orden topológico. Este lo podemos obtener con el algoritmo de Kahn o inviertiendo el post-order de *DFS*. Por ejemplo:

Problemas sobre DAGs

Sobre DAGs muchos problemas son fáciles, gracias al uso del orden topológico. Este lo podemos obtener con el algoritmo de Kahn o inviertiendo el post-order de *DFS*. Por ejemplo:

Problemas

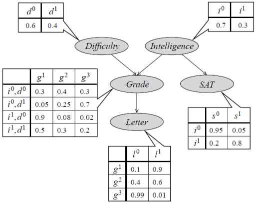
- ¿Cuál es la cantidad de caminos que van de v a w?
- Si el digrafo es pesado, ¿Cuál es el camino de menor costo de v a w?
- ¿Y el de mayor peso?
- ¿Con qué técnica se resuelven estos problemas? ¿Qué propiedad de los caminos mínimos utilizamos?

¿Por qué no podemos hacer lo mismo en digrafos generales?

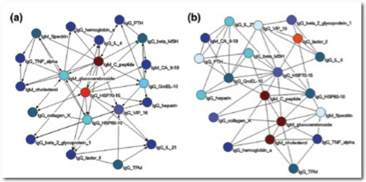
Aplicaciones de CM todos a todos

Manejo de tráfico urbano, buscar recorridos óptimos entre varios puntos

Problemas que podemos modelar cómo DAGs



Redes Bayesianas. Son DAG's que modelan causalidad.



Redes de dependencias, cómo la que vimo en el parcial

Sasha is back

Enunciado

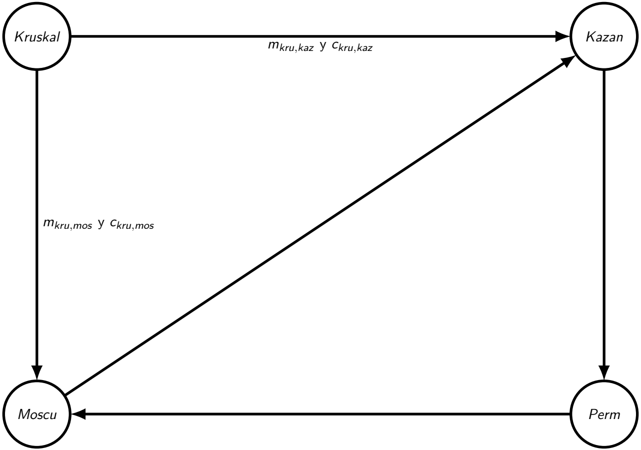
Sasha termino de cursar un cuatri agotador, por lo que va a volver a Kruskal para irse de vacaciones. Ya terminaron las obras con el mapa anterior que le dejo al presidente, e inclusive se agregaron más rutas. Así quedó un mapa de n ciudades y m rutas. Cada ruta e_i tiene un cierto peaje asociado p_i y se tarda m_i minutos en recorrerla. Queremos encontrar el camino de *Kruskal* a *Kazan* que minimice el costo de los peajes a pagar y que se pueda recorrer en menos de t[a] minutos.

[a] t es "chico" ($t \in \mathcal{O}(n)$(

Especificaciones

- El formato del input es un número N y M siendo las ciudades y rutas respectivamente. Seguidos de M líneas de la forma c_x, c_y, p_i, m_i con las dos ciudades que conecta cada ruta y su respectivo tiempo y peaje.

Dibujemos un ejemplo



¿Quienes sería V y E en este caso? ¿Hay algún problema con que tenga ciclos este grafo?

Analicemos el problema

- Son "casi" dos caminos mínimos en simultáneo

Analicemos el problema

- Son "casi" dos caminos mínimos en simultáneo: para el tiempo no queremos un camino mínimo, e incluso puede ser contraproducente.
- Recordemos que t es "chico" ¿Cómo se les ocurre que podemos modelar este problema?

Analicemos el problema

- Son "casi" dos caminos mínimos en simultáneo: para el tiempo no queremos un camino mínimo, e incluso puede ser contraproducente.
- Recordemos que t es "chico" ¿Cómo se les ocurre que podemos modelar este problema?
- ¡Podemos llevar la cuenta del tiempo en los nodos! Para esto podemos crear un digrafo G_t ¿Cómo nos quedarían los vértices dé G_t?

Analicemos el problema

- Son "casi" dos caminos mínimos en simultáneo: para el tiempo no queremos un camino mínimo, e incluso puede ser contraproducente.
- Recordemos que t es "chico" ¿Cómo se les ocurre que podemos modelar este problema?
- ¡Podemos llevar la cuenta del tiempo en los nodos! Para esto podemos crear un digrafo G_t ¿Cómo nos quedarían los vértices dé G_t?
- Sus vértices son los pares (c, t') donde $c \in V$ y $0 \leq t' \leq t$. ¿Qué aristas los conectan?

Analicemos el problema

- Son "casi" dos caminos mínimos en simultáneo: para el tiempo no queremos un camino mínimo, e incluso puede ser contraproducente.
- Recordemos que t es "chico" ¿Cómo se les ocurre que podemos modelar este problema?
- ¡Podemos llevar la cuenta del tiempo en los nodos! Para esto podemos crear un digrafo G_t ¿Cómo nos quedarían los vértices dé G_t?
- Sus vértices son los pares (c, t') donde $c \in V$ y $0 \leq t' \leq t$. ¿Qué aristas los conectan?
- Conectamos (c_1, t_1) con (c_2, t_2) con un eje de peso p_j si el eje $c_1 c_2 = e_j$ cumple $m_j = t_2 - t_1$. ¿Qué representa cada arista?

Solución

- ¿Cómo hacemos para resolver el problema a partir de este digrafo G_t? ¿Ya está completo nuestro digrafo?

Solución

- ¿Cómo hacemos para resolver el problema a partir de este digrafo G_t? ¿Ya está completo nuestro digrafo?
- A todos los nodos de la forma $(Kazan, t')$ para todos los t' le agregamos un eje de costo 0 al nodo *final* ¿Cómo resolvemos el problema?

Solución

- ¿Cómo hacemos para resolver el problema a partir de este digrafo G_t? ¿Ya está completo nuestro digrafo?
- A todos los nodos de la forma $(Kazan, t')$ para todos los t' le agregamos un eje de costo 0 al nodo *final* ¿Cómo resolvemos el problema?
- Queremos encontrar un camino de costo mínimo de $(Kruskal, 0)$ a *final* en G_t ¿Cuál es el tamaño de G_t? ?

Solución

- ¿Cómo hacemos para resolver el problema a partir de este digrafo G_t? ¿Ya está completo nuestro digrafo?
- A todos los nodos de la forma $(Kazan, t')$ para todos los t' le agregamos un eje de costo 0 al nodo *final* ¿Cómo resolvemos el problema?
- Queremos encontrar un camino de costo mínimo de $(Kruskal, 0)$ a *final* en G_t ¿Cuál es el tamaño de G_t? ?
- Tiene $\mathcal{O}(nt)$ nodos y $\mathcal{O}(mt)$ ejes. Ahora podemos utilizar Dijkstra para calcular el camino en $\mathcal{O}(mt \ \log(nt))$. ¿Se puede hacer en mejor tiempo? ¿Qué forma tiene el digrafo?

DAGs, here we go again

- Es un digrafo sin ciclos. ¿Por qué sabemos que no tiene ciclos? ¿Qué implicaría un ciclo en nuestro modelo?
- ¿Cómo podemos calcular el camino mínimo en un DAG?

CM de v a s en DAGs

$$d(v) = \begin{cases} 0, & \text{si } v = s \\ \min\{w(u,v) + d(u) : u \to v \in E\}, & \text{si no.} \end{cases}$$

- ¿Vale esto siempre, aunque no tenga ciclos?
- ¿Me sirve para hacer programación dinámica?
- ¿Cómo me queda la complejidad del algoritmo ahora?

CM de v a s en DAGs

$$d(v) = \begin{cases} 0, & \text{si } v = s \\ \min\{w(u,v) + d(u) : u \to v \in E\}, & \text{si no.} \end{cases}$$

- ¿Vale esto siempre, aunque no tenga ciclos? *Sí, es matemáticamente correcto*
- ¿Me sirve para hacer programación dinámica? *Solo si no tengo ciclos. ¿Por qué?*
- ¿Cómo me queda la complejidad del algoritmo ahora? *Lineal en base al tamaño del grafo, en este caso es $\mathcal{O}(mt + nt)$*
- Con esto podríamos escribir la función top down. Si queremos la forma bottom-up deberíamos calcular el orden topológico y comenzar en s hasta llegar a t para obtener el mínimo.

Solución final

- Construyo el grafo G_t, agregando al nodo *final*. $\mathcal{O}(nt + mt)$
- Calculo camino mínimo de $(Kruskal, 0)$ a *final*. $\mathcal{O}(nt + mt)$
- Le damos el camino a Sasha así se puede ir de vacaciones. $\mathcal{O}(1)$
- Complejidad total: $\mathcal{O}(nt + mt)$.
- Si el valor de "t" no está acotado, entonces nuestra solución puede ser exponencial.

Variaciones Ejercicio 1

Posibles variaciones del ejercicio

¿Cómo cambiaría nuestra solución si ahora nos dicen que queremos encontrar la ruta a cada una de las ciudades y que cumpla las mismas especificaciones?

Variaciones Ejercicio 1

Posibles variaciones del ejercicio

¿Cómo cambiaría nuestra solución si ahora nos dicen que queremos encontrar la ruta a cada una de las ciudades y que cumpla las mismas especificaciones?

- Deberíamos agregar un nodo *final* para cada una de las ciudades.
- Luego, con los algoritmos que vimos no solo tenemos el camino mínimo de un vértice a otro, sino que podemos obtener el camino mínimo de *Kruskal* a todos. ¿Cómo?
- Utilizando Dijkstra o PD sobre DAG's, lo cual también nos va a dar el árbol de caminos mínimos.

Variaciones Ejercicio 1

Posibles variaciones del ejercicio

- ¿Qué ocurre si ahora queremos contar todos los recorridos posibles desde *Kruskal* a *Kazan* que tarden menos de t minutos?

Variaciones Ejercicio 1

Posibles variaciones del ejercicio

- ¿Qué ocurre si ahora queremos contar todos los recorridos posibles desde *Kruskal* a *Kazan* que tarden menos de t minutos?
  - Usando PD cómo tenemos un DAG podemos contar la cantidad de caminos en un tiempo polinomial. Podemos crear una función CantidadRecorridos = CR, que nos calcule la CR de un nodo v a Kazan:

$$CR(v = (ciudad, t)) = \begin{cases} \sum_{x \in N^+(v)} CR(x), & \text{si } ciudad \neq \text{Kazan} \\ 1, & \text{if } ciudad = \text{Kazan} \\ 0, & \text{cc} \end{cases}$$

- Para pensar: ¿Qué ocurre si ahora Natasha (hermana de Sasha) quiere encontrar el camino hasta que tarde menos tiempo entre todos los que minimizan el costo de los peajes a pagar?[a]

[a] Spoiler: No se pica tanto

Rayuela Rectangular

Enunciado (Parte 1)

Sasha volvió a Argentina y decidió hacer turismo en el parque centenario. Ahí se encontró una rayuela rectangular recién pintada. Esta rayuela, de dimensiones $p \times q$, tiene números todos distintos escritos en los casilleros. Los números no son consecutivos (es decir, la rayuela puede tener el 1 y el 4 pero no el 2) y tampoco hay un orden claro para recorrerla.
Junto a la rayuela Sasha descubrió las reglas particulares para jugar sobre este tablero rectangular:

- Estando en una posición (i, j) solo se puede saltar a aquellas posiciones que están en la misma línea horizontal o vertical que (i, j). Aparte la casilla destino no pueden estar a mayor distancia que k, y su número debe ser mayor al de la casilla (i, j).
- Al saltar de un casillero con número x a otro con número $x + r$ se ganan r puntos.
- El juego termina cuando le jugadore no puede realizar más saltos válidos.

Rayuela Rectangular

Enunciado (Parte 2)

Sasha es competitive, y por ende quiere encontrar el casillero inicial desde el cual puede obtener la máxima cantidad de puntos.

El input de nuestro problema es la matriz R con los valores de cada casilla de la rayuela y la distancia k.

Modelando ando

- Igual que en el caso anterior vamos tener que crear un digrafo $D = (V, E)$ que nos modele cómo sería un recorrido de Sasha para esta rayuela. ¿Cómo se les ocurre modelar estos estados?

Modelando ando

- Igual que en el caso anterior vamos tener que crear un digrafo $D = (V, E)$ que nos modele cómo sería un recorrido de Sasha para esta rayuela. ¿Cómo se les ocurre modelar estos estados?
- Uno podría ver la similitud con el ejercicio 8 de la práctica 4, el problema es que acá no sabemos si hay una cota para los números. Por lo que definir cada vértice v cómo la tripla (suma_acumulada,i, j) nos podría generar un digrafo inecesariamente grande. ¿Se les ocurre una idea mejor?

Armando nuestro modelo

- Exacto! Armamos un grafo cuyos vértices $v \in V$ son los pares (i, j) donde $0 \leq i < p$ y $0 \leq j < q$. ¿Que aristas vamos a tener?

Armando nuestro modelo

- Exacto! Armamos un grafo cuyos vértices $v \in V$ son los pares (i, j) donde $0 \leq i < p$ y $0 \leq j < q$. ¿Que aristas vamos a tener?
- Conectamos $v = (i_i, j_1)$ con $w = (i_2, j_2)$ con un eje de peso $R[w]$ si se cumple que:
  - v y w están en la misma línea horizontal o vertical y su distancia es menor k: $w = (i_i + k_1, j_1) \wedge (i_i, j_1 + k_1)$ con $-k \leq k_1 \leq k$.
  - El valor del casillero de w es mayor que el de v: $R[v] < R[w]$.
- ¿Qué representa cada arista? ¿Nos falta representar algún otro estado?

Armando nuestro modelo

- Nos falta modelar que Sasha pueda comenzar en cualquier posición y determinar cuándo es que termina. ¿Que se les ocurre?

Armando nuestro modelo

- Nos falta modelar que Sasha pueda comenzar en cualquier posición y determinar cuándo es que termina. ¿Que se les ocurre?
- Podemos crear un nodo *inicio* y otro *fin* que representen estos estados. ¿Que aristas van a tener estos?

Armando nuestro modelo

- Nos falta modelar que Sasha pueda comenzar en cualquier posición y determinar cuándo es que termina. ¿Que se les ocurre?
- Podemos crear un nodo *inicio* y otro *fin* que representen estos estados. ¿Que aristas van a tener estos?
- *inicio* se va a conectar a todos los vértices de D con costo 0 y todos los vértices $v \in V$ tales que $d_{out}(v) = 0$ (¿Por qué puede pasar esto?) van a tener una arista hacia *fin*.
- Ahora ya tenemos nuestro modelo. ¿Cómo se resuelve este problema? ¿Cumple alguna propiedad nuestro digrafo?

¡Es un DAG!

Representación verídica de Sasha si la existencia de un ciclo fuera probada

Solución

- Armamos nuestro digrafo D con sus nodos y aristas pertinentes. Lo cual nos va a costar $\mathcal{O}(p \cdot q \cdot k)$ en el peor caso.

Solución

- Armamos nuestro digrafo D con sus nodos y aristas pertinentes. Lo cual nos va a costar $\mathcal{O}(p \cdot q \cdot k)$ en el peor caso.
- Luego calculamos el camino máximo entre *inicio* y *fin* utilizando el algoritmo de camino máximo en DAG's. $\mathcal{O}(p \cdot q \cdot k)$, pues es lineal en base al tamaño del grafo. ¿Sobre qué nodo vamos a correr nuestra función de camino máximo?
- Por último devolvemos el segundo nodo de algún camino máximo, el cual va a ser un casillero inicial desde el cuál puede obtener la máxima cantidad de puntos. $\mathcal{O}(1)$
- ¿Se puede mejorarAhí el modelo?

Solución

- Armamos nuestro digrafo D con sus nodos y aristas pertinentes. Lo cual nos va a costar $\mathcal{O}(p \cdot q \cdot k)$ en el peor caso.
- Luego calculamos el camino máximo entre *inicio* y *fin* utilizando el algoritmo de camino máximo en DAG's. $\mathcal{O}(p \cdot q \cdot k)$, pues es lineal en base al tamaño del grafo. ¿Sobre qué nodo vamos a correr nuestra función de camino máximo?
- Por último devolvemos el segundo nodo de algún camino máximo, el cual va a ser un casillero inicial desde el cuál puede obtener la máxima cantidad de puntos. $\mathcal{O}(1)$
- ¿Se puede mejorarAhí el modelo? Sí, eso queda para pensar.

Tips para DAG's

- Leer bien el enunciado y anotar todos los datos que se nos presentan para poder modelar el problema
- Intentar no modificar los algoritmos y usarlos como caja negra. Esto podemos hacerlo al aprovechar los resultados que te dan los mismos, para así resolver el problema (cómo la matriz de $Floyd - Warshall$). También podemos modelar el problema de tal forma que solo tengamos que correr nuestro algoritmo sobre nuestro modelo y eso nos dé la respuesta (cómo el caso del nodo *final* en el ejercicio 1).
- Estén atentos a si el digrafo con el que están tratando es un *DAG*, puesto que eso puede reducir la complejidad de su algoritmo significativamente.

¿Sale un break?