Algoritmos en digráficas

Miguel Raggi

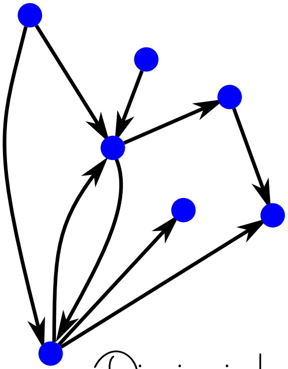
Algoritmos en grafos
Escuela Nacional de Estudios Superiores
UNAM

23 de mayo de 2018

Índice:

Índice:

¿Qué es un digrafo?



Dirigido

DAG

Definición (Directed Acyclic Graph)

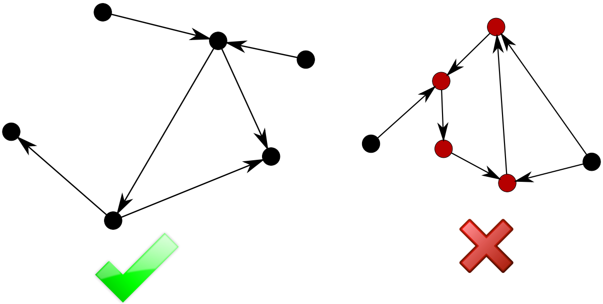
Una digráfica es acíclica si no tiene ciclos dirigidos.

DAG

Definición (Directed Acyclic Graph)

Una digráfica es acíclica si no tiene ciclos dirigidos.

Ejemplo:



Índice:

Fuertemente Conexa

Definición

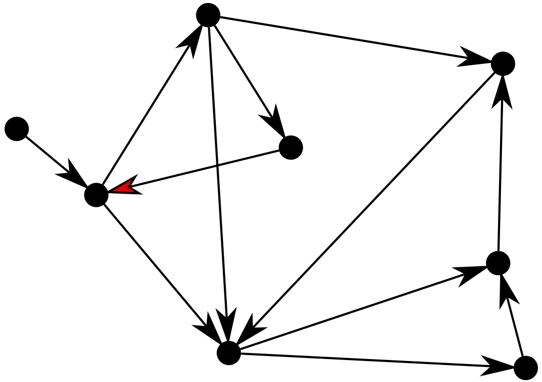
Sea D una digráfica. Decimos que es fuertemente conexa si para cualquier pareja de vértices x, y existe una trayectoria (dirigida) de x a y.

Componentes Fuertemente Conexas

Definición

Dada una digráfica D, las componentes fuertemente conexas son conjuntos de vértices C maximales con la siguiente propiedad: $\forall a, b \in C$, existe camino (dirigido) en D de a a b y también de b a a.

Ejemplo:

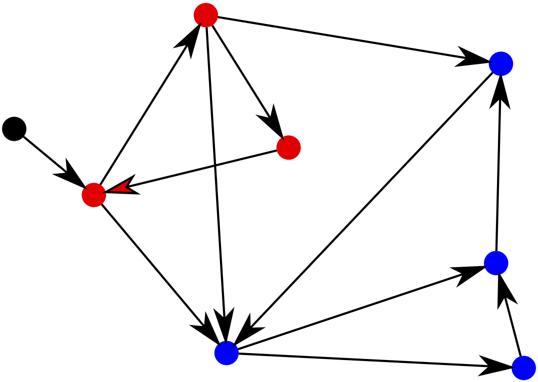


Componentes Fuertemente Conexas

Definición

Dada una digráfica D, las componentes fuertemente conexas son conjuntos de vértices C maximales con la siguiente propiedad: $\forall a, b \in C$, existe camino (dirigido) en D de a a b y también de b a a.

Ejemplo:



Componentes Fuertemente Conexas

- Vamos a tomar $\sim$ la relación de equivalencia en vértices $x \sim y$ si hay camino de x a y y de y a x.

Componentes Fuertemente Conexas

- Vamos a tomar $\sim$ la relación de equivalencia en vértices $x \sim y$ si hay camino de x a y y de y a x.
- ¿Qué estructura tienen las SCC?

Componentes Fuertemente Conexas

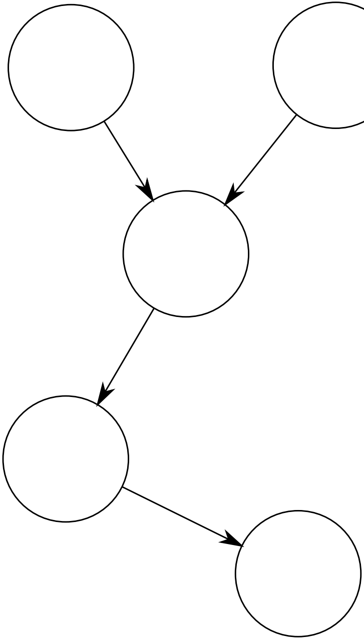
- Vamos a tomar $\sim$ la relación de equivalencia en vértices $x \sim y$ si hay camino de x a y y de y a x.
- ¿Qué estructura tienen las SCC?
- Pues tienen estructura de orden parcial!

Componentes Fuertemente Conexas

- Vamos a tomar $\sim$ la relación de equivalencia en vértices $x \sim y$ si hay camino de x a y y de y a x.
- ¿Qué estructura tienen las SCC?
- Pues tienen estructura de orden parcial!
- Es decir, para dos clases de equivalencia C_1 y C_2 decimos que $C_1 > C_2$ si hay un camino de C_1 a C_2 (todos los vértices).

Componentes Fuertemente Conexas

- Vamos a tomar $\sim$ la relación de equivalencia en vértices $x \sim y$ si hay camino de x a y y de y a x.
- ¿Qué estructura tienen las SCC?
- Pues tienen estructura de orden parcial!
- Es decir, para dos clases de equivalencia C_1 y C_2 decimos que $C_1 > C_2$ si hay un camino de C_1 a C_2 (todos los vértices).
- Entonces esto me da un orden parcial en el conjunto de clases de equivalencia.

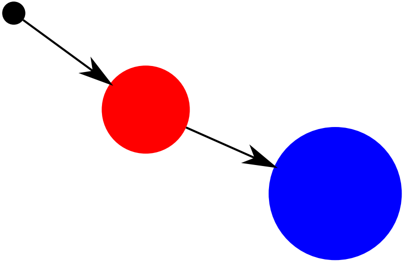


El esqueleto

Definición

Dada una digráfica, el esqueleto es la digráfica sin ciclos que se obtiene al "comprimir" los vértices de cada componente conexa en uno solo. Es decir, los vértices del esqueleto son las componentes fuertemente conexas de la digráfica original, y las aristas en el esqueleto son las que unían componentes en la original, sin repetir.

Ejemplo:

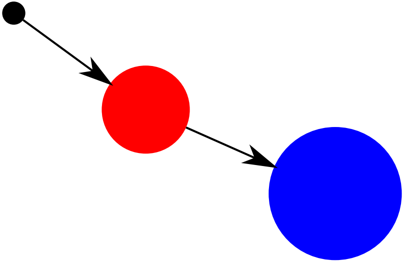


El esqueleto

Definición

Dada una digráfica, el esqueleto es la digráfica sin ciclos que se obtiene al "comprimir" los vértices de cada componente conexa en uno solo. Es decir, los vértices del esqueleto son las componentes fuertemente conexas de la digráfica original, y las aristas en el esqueleto son las que unían componentes en la original, sin repetir.

Ejemplo:



Nota: El esqueleto es obviamente una digráfica acíclica.

Índice:

Orden topológico en una DAG

Definición

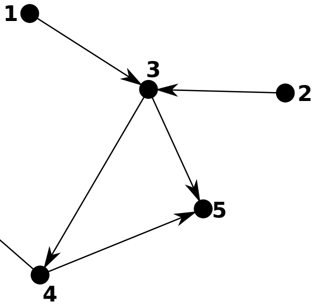
Sea D una digráfica acíclica. Un orden topológico es un orden de los vértices en el cual todas las aristas van de un vértice "menor" a uno "mayor":

Orden topológico en una DAG

Definición

Sea D una digráfica acíclica. Un orden topológico *es un orden de los vértices en el cual todas las aristas van de un vértice "menor" a uno "mayor":*

Ejemplo:

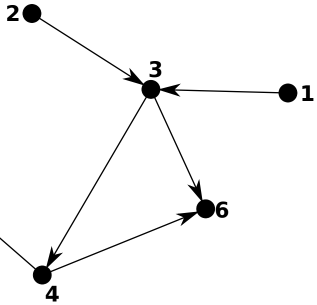


Orden topológico en una DAG

Definición

Sea D una digráfica acíclica. Un orden topológico *es un orden de los vértices en el cual todas las aristas van de un vértice "menor" a uno "mayor":*

Ejemplo:



Algoritmo para TopoSort

Problema

Encuentra un algoritmo que encuentre un orden topológico en una DAG.

Hay muchos, pero el más sencillo es DFS (y marcar en post-orden).
T es una cola.

- Repetidamente, toma un vértice v no explorado para hacer DFS desde ahí.
- Visita todos los hijos u de v.
- Marca el vértice v como explorado (post-orden!) y añádelo a T.

Al final regresa T.

Notas

- Lo anterior es si ya estamos seguros que el grafo es una DAG.

Notas

- Lo anterior es si ya estamos seguros que el grafo es una DAG. Si no es, podría ciclarse infinitamente.

Notas

- Lo anterior es si ya estamos seguros que el grafo es una DAG. Si no es, podría ciclarse infinitamente.
- Pero podemos detectarlo:

Notas

- Lo anterior es si ya estamos seguros que el grafo es una DAG. Si no es, podría ciclarse infinitamente.
- Pero podemos detectarlo:
- Todos los nodos empiezan "cerrados".
- Vamos a hacer DFS, y la primera vez que vemos un nodo, lo "abrimos", y cuando terminamos de procesarlo, lo "cerramos con llave" (así que ya no puede abrirse de nuevo)

Notas

- Lo anterior es si ya estamos seguros que el grafo es una DAG. Si no es, podría ciclarse infinitamente.
- Pero podemos detectarlo:
- Todos los nodos empiezan "cerrados".
- Vamos a hacer DFS, y la primera vez que vemos un nodo, lo "abrimos", y cuando terminamos de procesarlo, lo "cerramos con llave" (así que ya no puede abrirse de nuevo)

Pseudo-código

Para cada nodo v cerrado,

- procesa(v)

donde:

procesa(v):

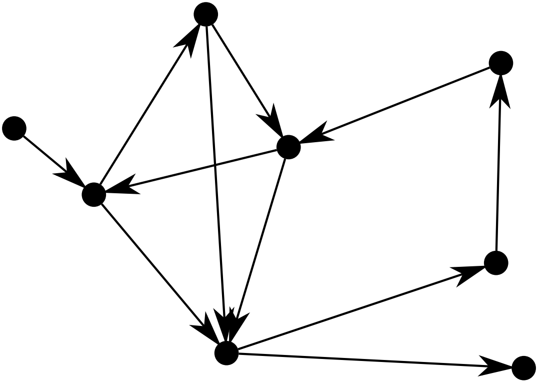
- Si v está "cerrado con llave", regresa
- Si v está "abierto", error!! (no es DAG)
- Abre v.
- Para cada vecino u de v:
  - visita(u)
- Cierra v con llave y añádelo a T

Índice:

LONGEST SIMPLE PATH

Problema

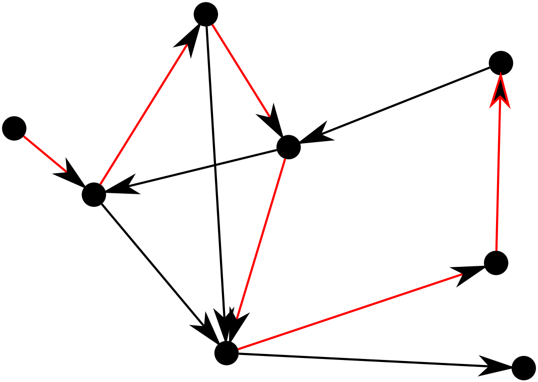
Sea D una gráfica dirigida con pesos positivos en las aristas. Es decir, $W : E(D) \to \mathbb{R}^+$. Queremos, con ayuda de la computadora, encontrar caminos simples que sean lo más largo posible. Es decir, caminos que no repitan vértices cuya suma de pesos de sus aristas sea lo más posible.



LONGEST SIMPLE PATH

Problema

Sea D una gráfica dirigida con pesos positivos en las aristas. Es decir, $W : E(D) \to \mathbb{R}^+$. Queremos, con ayuda de la computadora, encontrar caminos simples que sean lo más largo posible. Es decir, caminos que no repitan vértices cuya suma de pesos de sus aristas sea lo más posible.



LONGEST SIMPLE PATH en DAGs

- Dada una digráfica acíclica, es fácil y rápido encontrar el camino más largo usando programación dinámica.

LONGEST SIMPLE PATH en DAGs

- Dada una digráfica acíclica, es fácil y rápido encontrar el camino más largo usando programación dinámica.
- Es muy sencillo: A cada vértice le asociamos un numerito que representará “el tamaño del camino más largo que termina aquí”.

LONGEST SIMPLE PATH en DAGs

- Dada una digráfica acíclica, es fácil y rápido encontrar el camino más largo usando programación dinámica.
- Es muy sencillo: A cada vértice le asociamos un numerito que representará “el tamaño del camino más largo que termina aquí”.
- Tomamos un orden topológico cualquiera y procesamos los vértices en ese orden.

LONGEST SIMPLE PATH en DAGs

- Dada una digráfica acíclica, es fácil y rápido encontrar el camino más largo usando programación dinámica.
- Es muy sencillo: A cada vértice le asociamos un numerito que representará “el tamaño del camino más largo que termina aquí”.
- Tomamos un orden topológico cualquiera y procesamos los vértices en ese orden.
- Procesar un vértice v consiste de:

LONGEST SIMPLE PATH en DAGs

- Dada una digráfica acíclica, es fácil y rápido encontrar el camino más largo usando programación dinámica.
- Es muy sencillo: A cada vértice le asociamos un numerito que representará "el tamaño del camino más largo que termina aquí".
- Tomamos un orden topológico cualquiera y procesamos los vértices en ese orden.
- Procesar un vértice v consiste de:
  - Fijarse en los padres de v, que al cabo todos ellos ya fueron procesados.

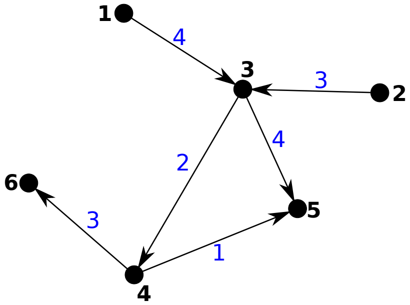
LONGEST SIMPLE PATH en DAGs

- Dada una digráfica acíclica, es fácil y rápido encontrar el camino más largo usando programación dinámica.
- Es muy sencillo: A cada vértice le asociamos un numerito que representará “el tamaño del camino más largo que termina aquí”.
- Tomamos un orden topológico cualquiera y procesamos los vértices en ese orden.
- Procesar un vértice v consiste de:
  - Fijarse en los padres de v, que al cabo todos ellos ya fueron procesados.
  - El numerito de v será el máximo (numerito del padre más la arista que va del padre a v).

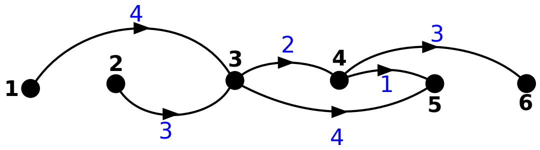
LONGEST SIMPLE PATH en DAGs

- Dada una digráfica acíclica, es fácil y rápido encontrar el camino más largo usando programación dinámica.
- Es muy sencillo: A cada vértice le asociamos un numerito que representará "el tamaño del camino más largo que termina aquí".
- Tomamos un orden topológico cualquiera y procesamos los vértices en ese orden.
- Procesar un vértice v consiste de:
  - Fijarse en los padres de v, que al cabo todos ellos ya fueron procesados.
  - El numerito de v será el máximo (numerito del padre más la arista que va del padre a v).
- Cuando el algoritmo termina, el número más grande encontrado representará el tamaño del camino más largo.

Ejemplo en DAGs



Ejemplo en DAGs



Ejemplo en DAGs