Algoritmia Avanzada

Teoría de Grafos

Dr. Arno Formella

Departamento de Informática
Universidad de Vigo

12/01/07

Algoritmia Avanzada: Teoría de Grafos

La página incial del curso es:
`http://www.ei.uvigo.es/~formella/doc/tc06`
Estos apuntes se acompañan con ilustraciones en pizarra dónde se explica las notaciones, las ideas de las comprobaciones, y el funcionamiento de los algoritmos.
El texto es meramente una brevísima introducción (5 horas) a la *Teoría de Grafos* donde se pincelan ciertos aspectos más bien para motivar y despertar interés por este campo fascinante de la informática.

Bibliografía

libros

- Reinhard Diestel. *Graph Theory.* 3rd edition, Springer Verlag, 2005.
  ISBN 3-540-26183-4.
  Existe una versión electrónica no imprimible:
  `http://www.math.uni-hamburg.de/home/diestel/books/graph.theory`
- Thomas H. Cormen, Charles E. Leiseron, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition.* McGraw Hill, 2001.
  ISNB 0-262-03292-7.

Bibliografía

enlaces

(como disponibles en enero 2007)

- `http://www.graphtheory.com`
- `http://www.ericweisstein.com/encyclopedias/books/GraphTheory.html`
- `http://mathworld.wolfram.com/Graph.html`
- `http://en.wikipedia.org/wiki/Graph_theory` (con cuidado)

Bibliografía

apuntes

(como disponibles en enero 2007)

- Gregorio Hernández Peñalver, Universidad Politécnica de Madrid, `http://www.dma.fi.upm.es/docencia/segundociclo/teorgraf`
  (en castellano)
- Steven C. Locke, Florida Atlantic University, `http://www.math.fau.edu/locke/graphthe.htm`
  (en orden alfabético)

Tareas

de estudio

El curso de doctorado ya tiene en su título *Desarrollo de Software*, por eso se ha pensado como tareas:

visualización de grafos:
: analisis de herramientas disponibles para visualizar grafos y la información que contienen (AGD, GraphViz, Dot, etc.)

librerías de programación:
: analisis de herramientas de programación para trabajar con grafos (Leda, GraphBase, GTL, Boost, etc.)

Se pide una breve analisis por lo menos según los siguientes criterios: completitud, complejidad, entorno de uso, algoritmos disponibles, filosofía de diseño, simplicidad de uso, aplicaciones, documentación y recursos disponibles...

Tareas

de programación

Para los que están más interesados en la parte de algoritmia, también hay dos opciones:

caminos más cortos: busca/desarrolla una implementación del algoritmo y analiza de forma estadística el uso de la cola de prioridad

caminos más cortos: busca/desarrolla un algoritmo que enumera todos los caminos con longitud como mucho k que existen entre dos vértices en un grafo plano.

camino de peso mínimo: busca/desarrolla un algoritmo que calcule el camino de peso mínimo entre dos vértices en un grafo ponderado, si se permite ciclos de pesos negativos (se pone como restricción adicional que ninguna arista se recorre más de una vez (pero sí se puede visitar un vértice más de una vez)).

Motivación

apps

Grafos de forma intuitiva se encuentran por ejemplo en las siguientes situaciones:

- planos o mapas de carreteras o calles
- redes de flujos (datos, líquidos, etc.)
- redes de transportes urbanos
- conexiones químicas entre átomos de una molécula
- relaciones de vecinidad en un mapamundi
- relaciones de interferencia entre antenas en un sistema de comunicación inalámbrica
- los enlaces entre documentos en el Internet

Es decir, un grafo es el *concepto abstracto* detrás de la representación de relaciones (aristas) entre entidades (vértices).

Ejemplos de problemas que se quiere resolver:

- Dado un traje completo para vestirse con el fin de ir a una boda en Galicia en invierno, es decir, llueve, ¿En qué orden hay que ponerse las cosas?
- Dado un callejero, ¿Es posible realizar un paseo que recorra cada calle exactamente una vez?
- Dado un callejero, ¿Cuál es el recorrido más corto que debe usar un cartero si tiene que recorrer cada calle por lo menos una vez?

Motivación

máx ejemplos

- Dado un callejero, ¿Cómo orientar las calles con sentido único, así que se siga podiendo ir de cada punto a cada punto?
- Dado una fiesta con igual número de chicas y chicos que se conocen o no, ¿Cuáles son las condiciones para que sea posible organizar un baile de conocidos (en parejas chica–chico)?
- Dado una descripción de las interconexiones de un circuito electrónico, ¿Cómo posicionar los chips con sus interconexiones en una placa?
- Dado un conjunto de casas por proteger de fuego, ¿Dónde posicionar los bomberos disponibles para minimizar su radio de acción?

Notaciones

intuitivas

Las notaciones que se suelen usar en la teoría de grafos son muy intuitivas y se basan ya casi siempre en la nomen clatura inglesa. Se aprovecha de interpretaciones sencillas y expresivas de los símbolos disponibles desde otras ramas de la matemática.

Notaciones

básicas

V	conjunto de vértices (nodos, puntos)
$[V]^r$	conjunto de todos los subconjuntos de V de tamaño r
$E \subseteq [V]^2$	conjunto de aristas
$v \in V$	vértice
$e = \{x, y\} \in E$	arista
$\{x, y\} \Longleftrightarrow: xy$	xy es arista
$G = (V, E)$	grafo
$V(G), E(G)$	vértices y aristas del grafo G
$v \in G :\Longleftrightarrow v \in V(G)$	v es vértice del grafo G
$e \in G :\Longleftrightarrow e \in E(G)$	e es arista del grafo G
$\lvert V \rvert =: n$	número de vértices
$\lvert V \rvert = \lvert V(G) \rvert = \lvert G \rvert$	notaciones equivalentes
$\lvert E \rvert =: m$	número de aristas
$\lvert E \rvert = \lvert E(G) \rvert = \lVert G \rVert$	notaciones equivalentes

Vocabulario

básico

trivial	si $\|G\| = 0$ o $\|G\| = 1$, el grafo es trivial
sobre	si $G = (V, E)$, G es un grafo sobre V
incidente	un vértice v es incidente a una arista e, si $v \in e$
	una arista e es incidente a un vértice v, si $v \in e$
adyacente	dos vértices v y w son adyacentes, si $\{v, w\} \in E$
	dos aristas e y f son adyacentes, si $e \cap f \neq \emptyset$
conecta	una arista conecta sus vértices
$X-Y$-arista	si $x \in X \subseteq V$ e $y \in Y \subseteq V$, xy es $X-Y$-arista
$E(X, Y)$	conjunto de $X-Y$-aristas

Notaciones

grado

$E(v) := E(v, V \setminus \{v\})$	conjunto de aristas incidentes a v
$N(v)$	conjunto de vértices adyacentes a v (vecinos)
$d(v) := \|E(v)\| = \|N(v)\|$	grado del vértice v
$d_G(v)$	grado del vértice $v \in G$
$\delta(G)$	grado mínima de los vértices en G
$\Delta(G)$	grado máxima de los vértices en G
$\epsilon(G) := \|E\|/\|V\|$	grado medio de los vértices en G

Notaciones

restas

$$
\begin{array}{ll}
G \setminus e & \text{grafo } (V, E \setminus \{e\}) \\
G \setminus v & \text{grafo } (V \setminus \{v\}, E \setminus E(v)) \\
G \setminus E' & \text{grafo } (V, E \setminus E') \\
G \setminus V' & \text{grafo } (V \setminus V', E \setminus E(V'))
\end{array}
$$

Vocabulario

vecinidad

vecino	v es vecino de w, si $vw \in E(v)$, es decir, si v y w son adyacentes
vecina	e es vecina de f, si $e \cap f \neq \emptyset$ es decir, e y f son incidentes al mismo vértice
independiente	vértices/aristas no adyacentess, un conjunto de vértices (aristas) mutuamente independientes es un conjunto independiente
completo	un grafo es completo, si todos sus vértices son vecinos
partición	el conjunto de conjuntos $\{V_0, ..., V_{r-1}\}$ es una partición de V, si $V = \bigcup_i V_i$, $V_i \neq \emptyset$, y $\forall i \neq j : V_i \bigcap V_j = \emptyset$

$\alpha(G)$	cardinalidad del conjunto de vértices independientes más grande del grafo G

Vocabulario

grafos

digrafo	las aristas están dirigidos, es decir, en vez de conjuntos $\{v, w\}$ se habla de parejas (v, w) o (w, v) es decir, $E \subseteq V \times V$
multigrafo	se permite más de una arista entre vértices
pseudografos	se permite bucles en vértices

Es decir, en la mayoría de los casos se entiende como grafo solamente el caso en el cual existe como mucho una arista (o arista dirigida) entre dos vertices diferentes.

Representación

dibujo

Se puede visualizar un grafo pintando sus vértices como puntos y sus aristas como líneas entre los puntos correspondientes.
¿Cuáles son las operaciones que se quieren realizar con un grafo (y sus componentes)?

Se puede almacenar un grafo con tres métodos básicos:

- una matriz de adyacencia
  - matriz cuatrada, y en el caso simple, binaria (y simétrica si no es digrafo) que codifica si existe una arista entre dos vértices
  - complejidad de memoria $\Omega(n^2)$
- listas de adyacencia
  - lista o array de vértices que contiene en cada entrada una lista de los vértices adyacentes
  - complejidad de memoria $\Theta(n + m)$

- tablas de dispersión
  - lista o array de vértices que contiene en cada entrada una tabla de dispersión de los vértices adyacentes
  - complejidad de memoria $\Theta(n + m)$

¿Cuáles son las principales ventajas y desventajas de cada uno de los métodos?
Existen más estructuras de datos para que ciertas operaciones sobre los grafos se pueden hacer más eficientes, especialmente para grafos planares.

Isomorfismo

definición

Sean $G = (V, E)$ y $G' = (V', E')$ dos grafos.
Si existe una bijección $\varphi : V \longrightarrow V'$ entre los vértices de los grafos de tal manera que

$$xy \in E \Longleftrightarrow \varphi(x)\varphi(y) \in E'$$

(es decir, si x e y son vecinos en G, lo son también $\varphi(x)$ y $\varphi(y)$ en G'), entonces G es isomorfo a G', $G \simeq G'$ o también $G = G'$, es decir, se dice *el grafo* G.

Una función f sobre grafos con $f(G) = f(G')$ si $G \simeq G'$ se llama invariante. Por ejemplo, invariantes son:

- n
- m
- ¿hay otras?

Invariante

abierta

No se conoce ninguna invariante sobre grafos que se puede calcular en tiempo polinomial (y determinista) que decida que dos grafos son isomorfos, pero tampoco se ha comprobado hasta hoy que el problema de isomorfismo entre grafos sea *NP*–completo.

NP-completo

definición

Recordamos que significa *NP*–completo:
Un problema pertence a la clase de problemas *NP*–completo, si existe una máquina de Turing no–determinista que resuelve el problema en tiempo polinomial respecto a la longitud de entrada y si todos los demás problemas dentro de la clase como mucho son más fáciles.

NP-completo

propiedades

Entonces sabemos sobre problemas *NP*–completos:

- Existe un algoritmo para su solución.
- Si alguien nos da una solución podemos comprobar en tiempo polinomial que de verdad es una solución.
- Si conocemos un algoritmo polinomial para un problema *NP*–completo sabemos implícitamente algoritmos para todos los problemas de la clase cuyos tiempos de cálculo siguen siendo polinomial.
- Siempre se puede usar búsqueda exhaustiva para resolver el problema de forma determinista (con tiempo de ejecución exponencial).
- La pregunta de existencia de algoritmos polinomiales y deterministas para problemas *NP*–completos es una de las grandes preguntas abiertas en la informática.

Isomorfismo

en NP

Dado una bijección de los nodos entre dos grafos, es muy fácil comprobar si los dos grafos son isomorfos: basta con comparar las matrizes de adyacencia que se puede realizar en tiempo $O(m)$.

Sean $G = (V, E)$ y $G' = (V', E')$ dos grafos.

$$G \cup G' := (V \cup V', E \cup E') \quad \text{unión de grafos}$$
$$G \cap G' := (V \cap V', E \cap E') \quad \text{intersección de grafos}$$

Vocabulario

parcial

grafo parcial	G' es grafo parcial de G, $V' \subseteq V$ y $E' \subseteq E$
subgrafo	G' es subgrafo de G, si $G' \cap G = G'$, es decir, G' es un grafo parcial de G que contiene todas las aristas de G cuyos vértices incidentes están en V'.
inducido	para $V' \subseteq V$ el grafo $(V', E(V', V'))$ es el subgrafo G' de G inducido por V'

Vocabulario

subgrafos

$G' \subseteq G$	G' es grafo parcial de G
$G[V']$	subgrafo de G sobre V' asumiendo $V' \subseteq V$
$G[G']$	subgrafo de G sobre $V(G')$ asumiendo $G' \subseteq G$

Vocabulario

partido

r-partido	un grafo es r-partido, si existe una partición de V en r conjuntos independientes
bipartido	2-partido

grafos k-regulares	$d(G) = \epsilon(G) = k \ \ (= \delta(G) = \Delta(G))$
grafos completos K^r	$G = (V, [V]^2), \lvert V \rvert = r$
caminos: P^k	$G = (\{v_0, \ldots, v_k\}, \{v_0v_1, v_1v_2, \ldots v_{k-1}v_k\})$
ciclos C^k	$G = (\{v_0, \ldots, v_k\}, \{v_0v_1, v_1v_2, \ldots v_{k-1}v_0\})$

Obviamente se puede describir un camino o ciclo por su secuencia de vértices.

longitud	número de aristas de un camino o ciclo
cíclico	un grafo que contiene un ciclo es cíclico
acíclico	un grafo que no contiene ningún ciclo es acíclico
cintura	un ciclo mínimo que un grafo contiene
circumferencia	ciclo máximo que un grafo contiene

$g(G)$	longitud de la cintura de G
	($g(G) = \infty$ si G acíclico)
$D(G)$	longitud de la circumferencia de G
	($D(G) = 0$ si G acíclico)

- grafos bipartidos B
- grafos bipartidos completos K^{n_1,n_2}
- hipercubos Q^k
  propiedades (con excepciones para Q^0 y Q^1):
  - $n = |V| = 2^k$
  - k-regular
  - bipartidos (¿Cómo particionar?)
  - $g(Q^k) = 4$
  - $D(Q^k) = 2^k$ (¿Cuál es un ciclo máximo?)

Distancia

métrica

$d(v, w)$ distancia entre dos vértices
siendo la longitud del camino más corto entre v y w

$d_G(v, w)$ distancia entre v y w en G

La distancia define una métrica, es decir,

1. $d(v, w) \geq 0$
2. $d(v, w) = 0 \iff v = w$
3. $d(v, w) = d(w, v)$
4. $d(u, w) \leq d(u, v) + d(v, w)$

Vocabulario

conexo

conexo	v y w son conexos, si $d(v,w) < \infty$;
	G es conexo,
	si todas las parejas $v, w \in V$ son conexas
disconexo	G es disconexo si no es conexo
puente	arista $e \in G$ es un puente
	G conexo tal que $G \setminus e$ es disconexo
vértice de corte	vértice $v \in G$
	G conexo tal que $G \setminus v$ es disconexo
componentes conexas	conjunto de subgrafos conexos y maximales

Notaciones

invariantes

$c(G)$	número de componentes conexas de G
$\kappa(G)$	cardinalidad mínima de un subconjunto de vértices de G tal que $G \setminus V$ sea disconexo
$\lambda(G)$	cardinalidad mínima de un subconjunto de aristas de G tal que $G \setminus E$ sea disconexo

k-conexo	G es k-conexo, si $\kappa(G) \geq k$
biconexo	2-conexo
k-aristoconexo	G es k-aristoconexo, si $\lambda(G) \geq k$
bloque	subgrafo máximo biconexo

Bosques

árboles

- si G no tiene ciclos, G es un bosque
- si G es un bosque y conexo, G es un árbol
- los componentes conexos de un bosque son árboles

Theorem

las siguientes propiedades son equivalentes:

- *G es un árbol*
- *entre cada pareja de vértices existe un camino único*
- *cada arista es un puente*
- *G es acíclico y* $n = m - 1$
- *G es conexo y* $n = m - 1$
- *G es acíclico y maximal en* $|E|$
- *G es conexo y minimal en* $|E|$

árbol con raíz	un árbol con un vértice marcado como raíz
árbol libre	un árbol sin marca en ningún vértice
árbol generador	T es árbol generador de un grafo G, si $T \subseteq G$ y $V(T) = V(G)$

Theorem

cada grafo G contiene un bosque generador, y si G es conexo, contiene un árbol generador (cuya raíz puede ser cualquier vértice)

Vocabulario

recorridos

recorrido euleriano	viaje entre dos vértices que no pasa varias veces por la misma arista
grafo euleriano	grafo con recorrido euleriano con todas sus aristas
grafo hamiltoniano	grafo con camino por todos sus vértices

Theorem (EULER)

$$\sum_v d(v) = 2m$$

Idea of proof

contar

Theorem

cada grafo tiene un número par de vértices con grado impar

Idea of proof

usa teorema de EULER

Theorem

cada grafo G contiene un camino P con $\|P\| \geq \delta(G)$, y cada grafo G con $\delta(G) \geq 2$ contiene un ciclo C con $|C| > \delta(G)$

Idea of proof

observa los vecinos del último vértice en un camino más largo

Theorem

si G no trivial: $\kappa(G) \leq \lambda(G) \leq \delta(G)$

Theorem

cada grafo G con $|E| > 1$ *contiene un subgrafo H con* $\delta(H) > \epsilon(H) \geq \epsilon(G)$

Theorem

G es bipartido con $|V_0| \neq |V_1| \implies$ *G no es hamiltoniano*

Theorem

$\forall v, w \in V, vw \notin E : d(v) + d(w) \geq n \implies G$ *es hamiltoniano*

Theorem

G *es hamiltoniano* $\implies \forall S \subset V : c(G \setminus S) \leq |S|$

Theorem

decidir si un grafo G es hamiltoniano es NP-completo

Theorem

G es bipartido $\iff$ *G no contiene ciclos impares*

Idea of proof

bi-coloración de un bosque generador

¿Algoritmo que decide que G es bipartido?

Theorem (EULER)

G es euleriano $\iff$ *G es conexo y* $\forall v \in V : d(v)$ *es par*

Idea of proof

analiza camino de longitud máxima que pasa por un vértice

¿Algoritmo que calcule un recorrido euleriano?

Theorem

G k-conexo $\Longrightarrow$ $|E| \geq \lceil kn/2 \rceil$

Theorem (WHITNEY)

G es 2-conexo $(|G| > 2) \quad \Longrightarrow \quad \forall v, w \in V \exists vPw, vQw : P \cap Q = \emptyset$

Es decir, existen dos caminos que no se intersecan entre todas las posibles parejas de vértices en G.

Theorem (MADER)

cada grafo G con $\epsilon(G) \geq 4k$ *contiene un grafo k-conexo como grafo parcial*

Algorithm

Exploración en profundidad (otros dicen recorrido en profundidad, o depth first search, DFS)

Con el algoritmo DFS se obtiene un árbol con raíz T y aristas de retroceso no pertenecientes a T. El algoritmo sirve entre otras cosas:

- determinar componentes conexas
- detectar puentes
- detectar vértices de corte
- detectar bloques
- ordenación topológica

Theorem

Sea T un árbol DFS de un grafo G conexo y v su raíz.
v es vértice de corte $\iff$ v tiene más de un hijo en T

Theorem

Sea T un árbol DFS de un grafo G conexo y v no sea la raíz.
v es vértice de corte $\iff$ no existe una arista de retroceso desde el subárbol debajo de v hacia un antecesor de v en T

DFS tiene complejidad $\Theta(n+m)$
(asumiendo listas de adyacencias, ¿y con los demás?)

Algorithm

Exploración en anchura (otros dicen recorrido en anchura, o breadth first search, BFS)

Con el algoritmo de BFS se obtiene un árbol con raíz T, aristas de retroceso no pertenecientes a T, y aristas de cruce. El algoritmo sirve entre otras cosas:

- determinar caminos lo más cortos entre vertices
- ordenación topológica

BFS tiene complejidad $\Theta(n+m)$ (asumiendo listas de adyacencias, ¿y con los demás?)

Sea G un grafo y $\overline{G}$ un grafo dirigido (digrafo).

fuertemente conexo	$\overline{G}$ es fuertemente conexo, si existe un recorrido entre cada pareja de vértices de G
orientable	G es orientable, si existe un grafo $\overline{G} \simeq G$ que es fuertemente conexo
componente fuertemente conexa	conjunto máximo de vértices de $\overline{G}$ cuyo subgrafo inducido es fuertemente conexo

$d_i(v)$ grado entrante
$d_o(v)$ grado saliente

Theorem (ROBBINS)

G orientable $\Longrightarrow$ *G es conexo y no contiene puentes.*

¿Algoritmo que calcule una orientación?
(¿Qué se entiende bajo una orientación óptima? Depende: minimizar el promedio de las distancias, minimizar el máximo de las distancias, minimizar el máximo de las diferencias entre las distancias en G y en $\overline{G}$ correspondientes.)

Se puede explorar también un digrafo en anchura (BFS) o en profundidad (DFS). A parte de las aristas del árbol y las aristas de retroceso, DFS produce aristas de progreso y aristas de cruce.
DFS se usa para producir una ordenación topológica de un digrafo acíclico, es decir, en el orden aparece un vértice v antes de un vértice w, si existe un camino desde v a w.
DFS se usa para determinar los componentes fuertemente conexos.
¿Algoritmo que calcule los componentes fuertemente conexos?

se puede seguir también recorridos eulerianos:

Theorem

$\overline{G}$ *es euleriano* $\iff$ $\overline{G}$ *es conexo y* $\forall v \in V : d_i(v) = d_o(v)$

¿Algoritmo que calcule un camino euleriano en un digrafo?

(G, W)	grafo ponderado con $W : E(G) \longrightarrow \mathbb{R}^+$
$w(G)$	peso de grafo G, $w(G) = \sum_{e \in G} w(e)$
$w(P)$	peso de camino P, $w(P) = \sum e \in P w(e)$
$d(v, w)$	distancia entre dos vértices, $d(v, w) = \min_P \{w(vPw)\}$
$dt(v) = \sum_w d(v, w)$	distancia total de un vértice

con $w(e) = 1 \forall e \in E$ se reproduce la distancia

$e(v)$	excentricidad, $e(v) = \max_{w \in G}\{d(v,w)\}$
$\mathrm{rad}(G)$	radio del grafo G, $\mathrm{rad}(G) = \min_{v \in G}\{e(v)\}$
$\mathrm{diam}(G)$	diámetro del grafo G, $\mathrm{diam}(G) = \max_{v \in G}\{e(v)\}$

Vocabulario

centro

centro	el centro del grafo G es
	el subgrafo de G inducido
	por los vértices con excentricidad mínima
mediana	la mediana del grafo G es
	el subgrafo de G inducido
	por los vértices con distancia total mínima

Theorem

G conexo $\Longrightarrow$ $\text{rad}(G) \leq \text{diam}(G) \leq 2 \cdot \text{rad}(G)$

Theorem

Todo grafo es centro de un grafo.

Theorem

El centro de un árbol consiste en uno o dos vértices.

¿Algoritmo que calcule el centro de un árbol?

dado un grafo G (o un digrafo $\overline{G}$) ponderado
preguntas interesantes son:

- ¿Cuál es un bosque generador con peso mínimo en G?
- dado un vertice $s \in \overline{G}$, ¿Cuáles son los caminos mínimos hacia los demás vértices?
- ¿Cuáles con los caminos mínimos entre todas las parejas de vértices?

Caminos

más cortos

- Hacia cada vértice $u \in G$ existe un camino desde s con distancia mínima.
- Las distancias a todos los vértices en uno de estos caminos a su vez son caminos mínimos.
- Si G es conexo, se puede construir un árbol con raíz $s \in G$ que determina todos los caminos mínimos hacia todos los vértices de G.
- ¿Algoritmo que calcule un bosque mínimo de un grafo G?

Algorithm

KRUSKAL, unir vorazmente árboles con aristas mínimas, complejidad $O(m \log n)$

Algorithm

PRIM, construir iterativamente un árbol con aristas mínimas, complejidad $O(m + n \log n)$

¿Algoritmo que calcule un árbol mínimo desde un vértice s en $\overline{G}$?

Algorithm

DIJKSTRA, complejidad $O(n^2)$

El algoritmo funciona igual con grafos como con digrafos.

¿Se puede permitir pesos negativos?
pueden existir ciclos negativos, es decir, ciclos con pesos negativos
¿Algoritmo que calcule un árbol mínimo desde un vértice s en G si G contiene pesos negativos?

Algorithm

BELLMANN–FORD, complejidad $O(mn)$

¿Algoritmo que calcule los caminos mínimos entre todas las parejas de vértices incluyendo el caso de pesos negativos?

Algorithm

FLOYD–WARSHALL, complejidad $O(n^3)$

Abmos algoritmos detectan la existencia de ciclos negativos en que caso terminan sin solución.

si se tiene más información sobre los grafos y sus pesos se puede mejorar los algoritmos:

- si los pesos están confinados por una constante W:
- si el número de aristas m está confinado por $O(n \log n)$:

Minores

definiciones

- contracción de arista
- minor
- minor topológico

Consecuencia:

Theorem (ROBERTSON/SEYMOUR)

Cada propiedad heredable sobre grafos se puede representar por un número finito de minores prohibidos.

- Como consecuencia se puede comprobar la existencia de un algoritmo con tiempo de cálculo cúbico que comprueba si un (ciclo de un) grafo forma un nudo en 3D.
- Pero hasta ahora no se conoce ni siquiera un algoritmo que lo hace.

Extremalidad

preguntas

se puede realizar preguntas interesantes como ciertas propiedades de grafos están relacionados con el tamaño del grafo (normalmente número de aristas)

- ¿Cuántas aristas tiene que tener un grafo como mínimo para que sea un grafo conexo?
- ¿Cuál es el número máximo de aristas que puede tener un grafo para que sea un grafo plano?

Vocabulario

planaridad

planar	un grafo es planar cuando se puede pintar sus vértices y sus aristas en un plano de tal manera que ninguna pareja de aristas se interseca
región	una región es un área del plano donde se ha pintado un grafo planar que esté confinado por aristas
grafo triangular	un grafo es triangular, si en su representación planar en el plano toda región está confinada por tres aristas del grafo

Theorem (EULER)

G planar y conexo con $n \leq 1$, m y l $\implies$ $n - m + l = 2$

Theorem

G planar con $n \geq 3$ $\implies$ $m \leq 3n - 6$

Theorem

G maximal planar si es un grafo triangular (y contiene $3n - 6$ *aristas)*

Theorem (KURATOWSKI)

G es planar $\iff$ *no contiene un* K^5 *o un* $K^{3,3}$ *como minor (o minor topológico)*

Coloración

tipos

- Coloración de vértices
- Coloración de aristas
- Coloración de grafos planos

Theorem

G planar $\Longrightarrow$ *los vértices de G son colorables con 4 colores*

Theorem

G planar que no contiene ningún triángulo $\Longrightarrow$ *los vértices de G son colorables con 3 colores*

$\chi(G)$ número mínimo de colores que se necesita para colorar los vértices de un grafo

$\chi'(G)$ número mínimo de colores que se necesita para colorar las aristas de un grafo

Theorem (Brooks)

G *conexo* $G \neq C^{impar}$ *y* $G \neq K^n \implies \chi(G) \leq \Delta(G)$

Theorem (Koenig)

G *bipartido* $\implies \chi'(G) = \Delta(G)$

Flujos

tipos

1. Flujos y redes
2. Flujos y cortes
3. Flujos y coloración

Vocabulario

flujos

viable	$f(e) \leq w(e)$, es decir, el flujo es como mucho igual a la capacidad
conservante	$\sum_v f_{in}(e) = \sum_v f_{out}(e)$, es decir, tanto como entra sale de un nodo (claro, excepto para fuente y afluente).

Theorem (FORD-FULKERSON)

Máximo flujo es igual a mínimo corte.

Algorithm

FORD–FULKERSON, complejidad $O(F * m)$ siendo F el máximo flujo (asumiendo enteros como capacidades).

Algorithm

EDMONDS–KARP, complejidad $O(nm^2)$.

Algorithm

DINIC, complejidad $O(n^2m)$.

Algorithm

KARZANOV-GOLDBERG-TARJAN, complejidad $O(n^3)$.

Se pueden variar las restricciones: capacidades máximas por nodos, más fuentes/afluentes, introducción de capacidades mínimas, etc.

Vocabulario

emparejamientos

emparejamiento	$M \subset E$ y $\forall e, f \in M : e \cup f = \emptyset$, es decir, pares de aristas no tienen vértices en común
perfecto	M cubre todos los vértices de G
M-alternado	un camino en G alternando con arista de M
M-aumento	camino M-alternado que se puede aumentar

Theorem (BERGE)

emparejamiento M es máximo $\Longrightarrow$ *G no contiene caminos M-aumento*

Theorem (HALL)

teorema del matrimonio: Sea $G = (U \cup V, E)$ un grafo bipartido. G tiene un emparejamiento completo (para U) $\Longrightarrow$
$\forall S \subseteq U : |N(S)| \geq |S|$, *siendo $N(S)$ conjunto de vecinos de S.*

Aplicaciones

ejemplos

1. Planificación
2. Diseño de circuitos
3. Juegos
4. Optimización
5. Teoría de Codificación
6. Visualización

Español

inglés

árbol	tree
árbol generador	spanning tree
arista	edge
arista de cruce	cross edge
arista de progreso	forward edge
arista de retroceso	back edge
articulación	articulation
bosque	forest
camino	path
cintura	girth
conexo	connected
dirigido	directed

Español

inglés

emparejamiento	matching
exploración en anchura	breadth first search
exploración en profundidad	depth first search
flujo	flow
fuertemente conexo	strongly connected
grado	degree
peso	weight
ponderado	weighted
puente	bridge
recorrido	walk
vecino	neighbor
vértice de corte	cutvertex

Inglés

español

articulation	articulación
back edge	arista de retroceso
breadth first search	exploración en anchura
bridge	puente
connected	conexo
cross edge	arista de cruce
cutvertex	vértice de corte
degree	grado
depth first search	exploración en profundidad
directed	dirigido
edge	arista
flow	flujo

Inglés

español

forest	bosque
forward edge	arista de progreso
girth	cintura
matching	emparejamiento
neighbor	vecino
path	camino
spanning tree	árbol generador
strongly connected	fuertemente conexo
tree	árbol
walk	recorrido
weight	peso
weighted	ponderado