Algoritmi (modulo di laboratorio)

Corso di Laurea in Matematica

Roberto Cordone

DI - Università degli Studi di Milano

Lezioni:	Martedì 8.30 - 10.30 in aula 8 e Z4	Mercoledì 10.30 - 13.30 in aula 2 e Z4
	Giovedì 15.30 - 18.30 in aula 2 e Z4	Venerdì 10.30 - 12.30 in aula 400 e Z4
Ricevimento:	su appuntamento (Dipartimento di Informatica)	
E-mail:	roberto.cordone@unimi.it	
Pagina web:	http://homes.di.unimi.it/~cordone/courses/2022-algo/2022-algo.html	
Sito Ariel:	https://mgoldwurmasd.ariel.ctu.unimi.it	

Lezione 9: Grafi

Milano, A.A. 2021/22

Ogni relazione binaria su un insieme base finito $V = \{v_1, \dots, v_n\}$ si può descrivere elencando le coppie di elementi di V in relazione

$$E = \{\{i,j\} : i \in V, j \in V, i \text{ e } j \text{ sono legate}\} \Rightarrow E \subseteq V \times V$$

Un modo standard di rappresentare una relazione binaria è il grafo $G = (V, E)$, cioè una coppia di insiemi:

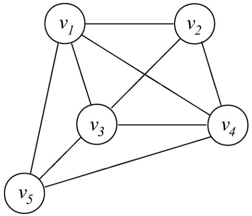
- un insieme V di oggetti elementari detti vertici
- un insieme E di coppie non ordinate di oggetti di V detti lati

Un grafo si rappresenta disegnando i vertici come punti (o cerchi) e i lati come linee

$$V = \{v_1, v_2, v_3, v_4, v_5\}$$

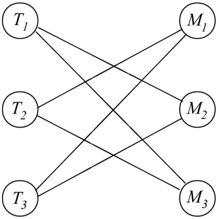
$$E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_1, v_5\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}, \{v_3, v_5\}, \{v_4, v_5\}\}$$

Si notino le parentesi graffe: la coppia non è ordinata



Esempi

- reti stradali: i vertici sono città, i lati strade
- reti elettriche: i vertici sono impianti, stazioni o utenti, i lati linee elettriche
- reti di telecomunicazione: i vertici sono trasmettitori, ripetitori e ricevitori, i lati collegamenti
- reti sociali: i vertici sono utenti, i lati relazioni umane
- giochi: i vertici sono posizioni, i lati mosse
- relazioni di (in)compatibilità; i vertici sono oggetti/persone, i lati coppie di oggetti/persone (in)compatibili



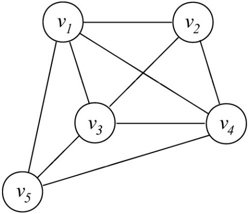
V include operazioni (*task*) e macchine
$V = \{T_1, T_2, T_3, M_1, M_2, M_3\}$

Il lato $\{i, j\}$ indica che il task i
può essere eseguito dalla macchina j

Non si può eseguire un task su un task,
né una macchina su una macchina

Topologia di un grafo

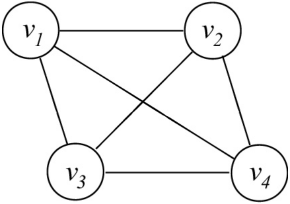
- i e j sono i vertici estremi del lato $\{i, j\}$
- due vertici i e j sono adiacenti se il lato $\{i, j\}$ esiste
- il lato $\{i, j\}$ è incidente nei vertici i e j
- il grado δ_v di un vertice v è il numero di lati incidenti



- v_1 e v_2 sono i vertici estremi di $\{v_1, v_2\}$
- v_3 e v_4 sono adiacenti (il lato $\{v_3, v_4\}$ esiste)
- il lato $\{v_3, v_5\}$ è incidente ai vertici v_3 e v_5
- il grado del vertice v_3 è $\delta_{v_3} = 4$

Un grafo è completo quando ogni coppia di vertici corresponde a un lato

$$E = \{\{v_i, v_j\} : v_i \in V, v_j \in V, i < j\}$$



Tutti i grafi con n vertici hanno

$$m \leq \binom{n}{2} = \frac{n\,(n-1)}{2} \text{ lati}$$

(con l'uguaglianza per i grafi completi)

Se sono ammessi gli autoanelli, un grafo completo ha

$$E = \{\{v_i, v_j\} : v_i \in V, v_j \in V, i \leq j\} \qquad m = \frac{n\,(n+1)}{2}$$

Sottografi

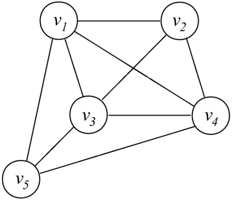
$H = (U, X)$ è un sottografo di $G = (V, E)$ se

- è un grafo
- $U \subseteq V$ e $X \subseteq E$

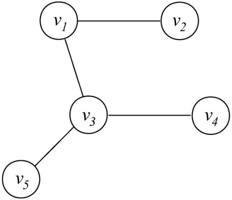
È un sottografo ricoprente quando $U = V$

È un sottografo indotto quando $X = E_U = \{\{u, v\} \in E : u, v \in U\}$

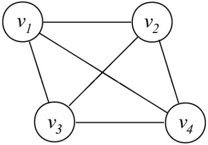
$G = (V, E)$



$H_1 = (U_1, X_1)$



$H_2 = (U_2, X_2)$



$V = \{v_1, v_2, v_3, v_4, v_5\}$

$E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_1, v_5\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}, \{v_3, v_5\}, \{v_4, v_5\}\}$

$U_1 = \{v_1, v_2, v_3, v_4, v_5\} = V$

$X_1 = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_3, v_4\}, \{v_3, v_5\}\}$

$U_2 = \{v_1, v_2, v_3, v_4\}$

$X_2 = \{\{v_1, v_2], \{v_1, v_3], \{v_1, v_4], \{v_2, v_3], \{v_2, v_4], \{v_3, v_4]\} = E_{U_2}$

Connessione

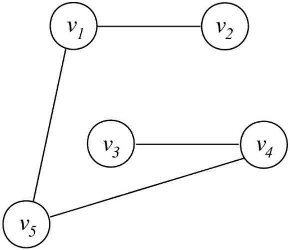
- Un cammino è una sequenza di lati, in cui ognuno condivide un estremo col lato precedente e l'altro col lato successivo (*se esistono*)

$$P = \left(\left\{v_{\pi_0}, v_{\pi_1}\right\}, \left\{v_{\pi_1}, v_{\pi_2}\right\}, \ldots, \left\{v_{\pi_{k-1}}, v_{\pi_k}\right\}\right)$$

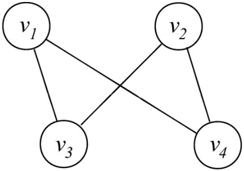
I vertici estremi v_{π_0} e v_{π_k} sono connessi

- Un ciclo è un cammino il cui primo e ultimo vertice coincidono

$$v_{\pi_k} = v_{\pi_0}$$



$P = (\{v_1, v_2\}, \{v_1, v_5\}, \{v_4, v_5\}, \{v_3, v_4\})$



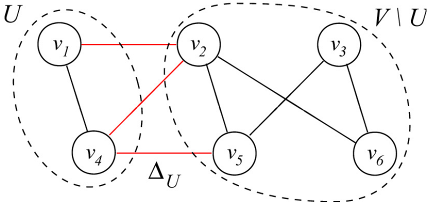
$P = (\{v_1, v_3\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_1, v_4\})$

In un grafo connesso ogni coppia di vertici è connessa da un cammino

Tagli

- Dato un sottoinsieme di vertici $U \subset V$, il taglio indotto Δ_U è il sottoinsieme di lati con un estremo in U e l'altro in $V \setminus U$

$$\Delta_U = \{\{u, v\} \in E : |\{u, v\} \cap U| = |\{u, v\} \cap (V \setminus U)| = 1\}$$



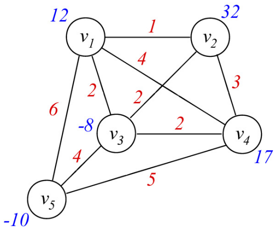
$$U = \{v_1, v_4\}$$

$$\Delta_U = \{\{v_1, v_2\}, \{v_2, v_4\}, \{v_4, v_5\}\}$$

Grafi pesati

Si possono definire uno o più pesi sui vertici/lati

- Un grafo pesato sui vertici (V, E, w) è un grafo $G = (V, E)$ i cui vertici sono associati a informazioni quantitative $w : V \to \mathbb{R}$
- Un grafo pesato sui lati (V, E, c) è un grafo $G = (V, E)$ i cui lati sono associati a informazioni quantitative $c : E \to \mathbb{R}$



Applicazione	vertici	lati
reti stradali	viaggi generati o attratti	lunghezze, tempi o costi di viaggio
reti elettriche	energia prodotta o consumata	costo di costruzione delle linee
reti di telecomunicazione	domanda di traffico	capacità o costo dei collegamenti
reti sociali	valore individuale	forza della relazione
giochi	valore della posizione	probabilità o costo della mossa
relazione di (in)compatibilità	utilità dell'elemento	forza della (in)compatibilità

Modelli basati su grafi (1)

Qual è l'insieme massimo di persone che posso raggiungere per conoscenza?

L'insieme dei vertici V include tutti gli individui (io sono il vertice $i \in V$); l'insieme dei lati E tutte le conoscenze (coppie di individui che si conoscono)

Si trovi il sottoinsieme di massima cardinalità $U \subseteq V$ che include solo vertici u tali che esista un cammino P_{iu} fra i e u

$$U = \left\{u \in V : \exists P_{iu} = \left(\{v_{\pi_0}, v_{\pi_1}\}, \ldots, \{v_{\pi_{k-1}}, v_{\pi_k}\}\right) \text{ with } v_{\pi_0} = i, v_{\pi_k} = u\right\}$$

È vero che ognuno è a sei passi di distanza da ogni altra persona del mondo attraverso una catena di conoscenze?

L'insieme dei vertici V include gli individui; l'insieme dei lati E le conoscenze

Si trovi per ogni individuo $v \in V$ il sottoinsieme di massima cardinalità $U_v^6 \subseteq V$ che include solo vertici u tali che esista un cammino P_{vu}^6 di al più 6 lati fra v e u

$$U_v^6 = \left\{u \in V : \exists P_{vu}^6 = \left(\{v_{\pi_0}, v_{\pi_1}\}, \ldots, \{v_{\pi_{k-1}}, v_{\pi_k}\}\right) \text{ with } v_{\pi_0} = v, v_{\pi_k} = u, k \leq 6\right\}$$

Se $U_v^6 = V$ per ogni $v \in V$, la proprietà dei "sei gradi di separazione" è valida

Modelli basati su grafi (2)

Si calcoli il numero di Erdős di un matematico

L'insieme dei vertici V include tutti i matematici (quello dato è u, Erdős è v); l'insieme dei lati E include tutte le coppie con un lavoro pubblicato insieme

si trovi il cammino di minima cardinalità P_{uv} fra u e v

$$\min |P_{uv}| \text{ such that } P_{uv} = \left(\left\{u, v_{\pi_1}\right\}, \ldots, \left\{v_{\pi_{k-1}}, v\right\}\right)$$

Un museo consiste di un insieme di corridoi, che si incrociano in sale. Dove bisogna posizionare le guardie per averne una vicina ad ogni corridoio? Quante guardie servono per controllare l'intero museo?

L'insieme dei vertici V include tutte le sale, l'insieme dei lati E tutti i corridoi

Si trovi il sottoinsieme di vertici di minima cardinalità $U \subseteq V$ tale che ogni lato del grafo sia adiacente ad almeno un vertice di U

$$\min |U| \text{ such that } X = \{\{u, v\} \in E : \{u, v\} \cap U \neq \emptyset\} = E$$

Modelli basati su grafi (3)

Quali linee ferroviarie bisogna bombardare per distruggere ogni collegamento fra un centro industriale nemico e il fronte?

L'insieme dei vertici V include tutte le stazioni (v è il centro industriale, V^* raccoglie le stazioni al fronte), l'insieme dei lati include tutte le linee ferroviarie

$$\begin{aligned} &\min |\Delta_U| \\ &\qquad \Delta_U = \{\{u,v\} \in E : |\{u,v\} \cap U| = |\{u,v\} \cap (V \setminus U)| = 1\} \\ &\qquad U \ni v \\ &\qquad U \subseteq V \setminus V^* \end{aligned}$$

Dato un insieme di possibili investimenti finanziari, il loro rendimento atteso (ROI) e la matrice di correlazione a coppie, qual è il sottoinsieme più redditizio di investimenti scorrelati a coppie?

L'insieme dei vertici V include gli investimenti, il peso w_v fornisce il rendimento dell'investimento $v \in V$, l'insieme dei lati include tutte le coppie correlate

$$\max \sum_{v \in U} w_v \text{ such that } U \subseteq V \text{ e } E_U = \emptyset$$

Qual è la catena più corta di cambi di una lettera da `GATTO` *a* `PESCE`*?*

... fatelo da voi

Grafi orientati

Se la relazione binaria è asimmetrica, l'ordine degli elementi nelle coppie è significativo

Il modello è una coppia di insiemi $G = (N, A)$ detta grafo orientato

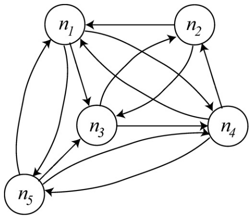
- un insieme di oggetti elementari detti nodi
- un insieme di coppie ordinate di oggetti detti archi

Un grafo orientato si rappresenta disegnando i nodi come punti (o cerchi) gli archi come linee e il loro orientamento con frecce

$$N = \{n_1, n_2, n_3, n_4, n_5\}$$

$$\begin{aligned} A = \{&(n_1, n_3), (n_1, n_4), (n_1, n_5), (n_2, n_1), (n_2, n_3), \\ &(n_3, n_2), (n_3, n_4), (n_4, n_1), (n_4, n_2), (n_4, n_5), \\ &(n_5, n_1), (n_5, n_3), (n_5, n_4)\} \end{aligned}$$

Si notino le parentesi rotonde: la coppia è ordinata



Cammini, cicli e tagli orientati

- i è la coda e j è la testa dell'arco (i, j)
- l'arco (i, j) è un arco uscente for i, un arco entrante per j
- il grado uscente δ_i^+ di un nodo $i \in N$ è il numero degli archi uscenti
- il grado entrante δ_i^- di un nodo $i \in N$ è il numero degli archi entranti
- un cammino orientato è una sequenza di archi la cui testa coincide con la coda del successivo (*tranne per l'ultimo arco*)

  $$P = \left(\left(i_{\pi_0}, i_{\pi_1} \right), \left(i_{\pi_1}, i_{\pi_2} \right), \ldots, \left(i_{\pi_{k-1}}, i_{\pi_k} \right) \right)$$

  I nodi i_{π_0} e i_{π_k} sono fortemente connessi e in un grafo fortemente connesso ogni coppia di nodi è fortemente connessa
- un ciclo orientato (circuito) è un cammino orientato il cui primo e ultimo nodo coincidono

  $$i_{\pi_k} = i_{\pi_0}$$
- dato un sottoinsieme di nodi $U \subset N$, la sezione uscente (entrante) Δ_U^+ (Δ_U^-) è il sottoinsieme di archi con coda (testa) in U e testa (coda) in $N \setminus U$

  $$\Delta_U^+ = \{(i, j) \in A : i \in U, j \in N \setminus U\}$$
  $$\Delta_U^- = \{(i, j) \in A : i \in N \setminus U, j \in U\}$$

Modelli basati sui grafi orientati

Alcune reti sociali considerano relazioni orientate fra gli utenti ("follower" e "leader")

In molti giochi, le posizioni evolvono in altre posizioni irreversibilmente (ad es., catture negli scacchi, le pedine nella dama, il tris...)

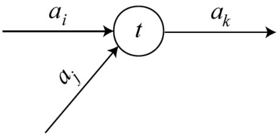
Nelle reti stradali urbane, molte strade sono a senso unico

Un progetto spesso è formato da attività soggette a una relazione binaria di precedenza, che richiede di terminare un'attività prima di cominciarne un'altra: $a_i \prec a_k$ e $a_j \prec a_k$

Modello activity-on-arc (*AOA*)

nodo $\leftrightarrow$ evento "milestone"

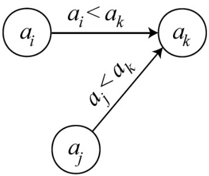
arco $\leftrightarrow$ attività



Modello activity-on-node (*AON*)

nodo $\leftrightarrow$ attività

arco $\leftrightarrow$ precedenza



Grafi: operazioni

Ogni grafo non orientato corrisponde a un grafo orientato simmetrico

Sia $\mathcal{G}_N$ l'insieme di tutti i grafi orientati su un dato insieme di nodi N

I grafi orientati ammettono tipicamente le seguenti operazioni

- aggiunta di un arco: dato un grafo e una coppia ordinata di nodi, inserisce un arco fra i due nodi nel grafo dato

$$\text{insarco} : \mathcal{G}_N \times N \times N \to \mathcal{G}_N$$

E se l'arco esiste già?

- eliminazione di un arco: dato un grafo e una coppia ordinata di nodi, cancella dal grafo dato l'arco fra i due nodi

$$\text{cancarco} : \mathcal{G}_N \times N \times N \to \mathcal{G}_N$$

E se l'arco non esiste?

- verifica di esistenza: dato un grafo e una coppia ordinata di nodi, verifica se l'arco fra i due nodi esiste o no

$$\text{esistearco} : \mathcal{G}_N \times N \times N \to \mathbb{B} \qquad (\text{ovvero } \{0, 1\})$$

Molte altre funzioni possono essere utilmente definite: lo faremo poi

Grafi: operazioni

In matematica basta definire un oggetto per crearlo

Nelle implementazioni concrete, però questo non sempre vale: potrebbe occorrere qualche inizializzazione o allocazione dinamica

Per motivi tecnici, quindi è opportuno definire anche

- creazione: crea un grafo vuoto sull'insieme dei nodi N

$$\text{creagrafo} : N \rightarrow \mathcal{G}_N$$

- distruzione: distrugge un grafo

$$\text{distruggegrafo} : \mathcal{G}_N \rightarrow ()$$

Grafi: implementazioni

I nodi sono messi in corrispondenza biunivoca con numeri naturali

$$N \leftrightarrow \{1, \ldots, |N|\}$$

Le funzioni di peso sui nodi sono rappresentate da vettori/tabelle

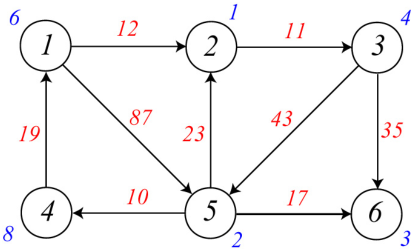
Per gli archi, ci sono tre rappresentazioni principali:

1. lista degli archi: una semplice lista/tabella che include tutti gli archi
2. matrice di adiacenza: una matrice quadrata le cui celle corrispondono a coppie di nodi
3. forward (backward) star: una lista/tabella che include per ogni nodo $i \in N$ una lista/tabella di archi uscenti (entranti)

Le funzioni di peso sugli archi si includono facilmente in ogni rappresentazione

Nei grafi non orientati, non occorrono forward e backward star (sono uguali!): ogni vertice ha una lista di incidenza

Lista degli archi



$N \rightarrow \{1, 2, 3, 4, 5, 6\}$

$w \rightarrow [\ 6\ \ 1\ \ 4\ \ 8\ \ 2\ \ 3\ ]$

$A \rightarrow ((1,2),(1,5),(2,3),(3,5),(3,6),(4,1),(5,2),(5,4),(5,6))$

$c \rightarrow [\ 12\quad 87\quad 11\quad 43\quad 35\quad 19\quad 23\quad 10\quad 17\ ]$

- Vantaggio: rappresentazione compatta $(\Theta(|A|))$
- Svantaggio: ricerca inefficiente di un arco dato $(\Theta(|A|))$

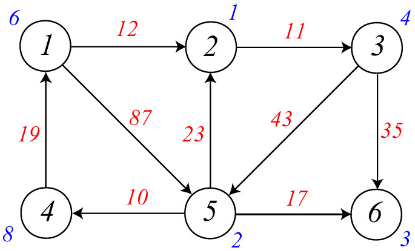
Lista degli archi: implementazione in C

```
typedef struct _grafo grafo;

struct _grafo
{
   int n;                    (numero dei nodi)
   int m;                    (numero degli archi)
   listaarchi L;             (lista degli archi)
};
```

Ovviamente, occorre anche una libreria per gestire archi e liste di archi
L'esempio che segue usa l'implementazione a puntatori

```
typedef struct _arco arco;
typedef arco* listaarchi;
typedef arco* posarco;
typedef int nodo;

#define NO_ARC NULL
#define NO_NODE 0
```

```
struct _arco
{
   nodo orig, dest;
   posarco succ, pred;
};
```



$$N \rightarrow \{1, 2, 3, 4, 5, 6\}$$
$$w \rightarrow [\,6 \;\; 1 \;\; 4 \;\; 8 \;\; 2 \;\; 3\,]$$

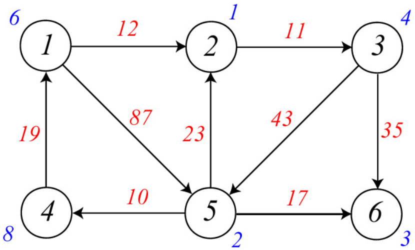
$$(A, c) \rightarrow \begin{bmatrix} - & 12 & - & - & 87 & - \\ - & - & 11 & - & - & - \\ - & - & - & - & 43 & 35 \\ 19 & - & - & - & - & - \\ - & 23 & - & 10 & - & 17 \\ - & - & - & - & - & - \end{bmatrix}$$

dove "–" è un valore numerico convenzionale per indicare "nessun arco"

- Vantaggio: ricerca molto efficiente di un arco dato ($\Theta(1)$)
- Svantaggio: occupazione di memoria enorme ($\Theta(|N|^2)$)

Matrice di adiacenza: implementazione in C

```
typedef struct _grafo grafo;
struct _grafo
{
   int n;                          (numero dei nodi)
   int m;                          (numero degli archi)
   boolean **MatriceAdiacenza;     (matrice di adiacenza)
};
```



$$N \;\rightarrow\; \{1,\ 2,\ 3,\ 4,\ 5,\ 6\} \qquad (A, c) \rightarrow \left[\begin{array}{l} (2,12) \rightarrow (5,87) \perp \\ (3,11) \perp \\ (5,43) \rightarrow (6,35) \perp \\ (1,19) \perp \\ (2,23) \rightarrow (4,10) \rightarrow (6,17) \perp \\ \quad \perp \end{array}\right]$$

$$w \;\rightarrow\; [\,6\ \ 1\ \ 4\ \ 8\ \ 2\ \ 3\,]$$

dove $\perp$ indica una lista vuota

- (S)Vantaggi: ricerca di efficienza intermedia di un arco dato $(\Theta\left(\delta_v^+\right))$
  occupazione di memoria intermedia $(\Theta\left(|N| + |A|\right))$
- Svantaggio: nessuna informazione sugli archi entranti
  (a meno che sia accompagnato dal vettore delle *backward star*)

Forward star: implementazione in C

```c
typedef struct _grafo grafo;
struct _grafo
{
   int n;                        (numero dei nodi)
   int m;                        (numero degli archi)
   listaarchi *FS;               (vettore delle forward star)
   listaarchi *BS;               (vettore delle backward star: può mancare)
};
```

Ovviamente, occorre anche una libreria per gestire archi e liste di archi
L'esempio che segue usa l'implementazione a puntatori

```c
typedef struct _arco arco;
typedef arco* listaarchi;
typedef arco* posarco;
typedef int nodo;

#define NO_ARC NULL
#define NO_NODE 0
```

```c
struct _arco
{
   nodo orig, dest;
   posarco succ, pred;
};
```