Capitolo 9

Grafi

9.1 Definizioni preliminari

Definizione 9.1. Un *grafo G = (V, E)* consiste di:

- un insieme V di *vertici*;
- un insieme E di coppie di vertici, detti *archi*

Definizione 9.2. Un grafo si dice *non orientato* se E rappresenta una relazione *simmetrica* in V, ossia $(u, v) \in E \Leftrightarrow (v, u) \in E$, altrimenti si dice *orientato*; nel primo caso si ha $|E| \leq n \cdot (n-1)/2$, mentre nel secondo $|E| \leq n^2$.

Definizione 9.3. Dato un grafo $G = (V, E)$, $G' = (V', E')$ è *sottografo* di G se $V' \subseteq V$ e $E' \subseteq E \cap (V \times V')$.

Definizione 9.4. Dato un grafo $G = (V, E)$, $G' = (V', E')$ è *sottografo indotto* di G se $V' \subseteq V$ e $E' = E \cap (V \times V')$ (ossia, se contiene tutti e soli gli archi presenti in E i cui vertici appartengono entrambi a V').

Definizione 9.5. Dato $G = (V, E)$, un *cammino* in G è una sequenza di vertici $\pi = \langle x_0, x_1, ..., x_k \rangle$ tali che $(x_i, x_{i+1}) \in E, \forall i = 0..k-1$; la *lunghezza* del cammino è pari al numero di archi che lo compongono. π si dice *cammino semplice* se $i \neq j \Rightarrow x_i \neq x_j \forall i, j = 0..k$.

Definizione 9.6. Il vertice v si dice *raggiungibile* da u se esiste un cammino π tale che $x_0 = u$ e $x_k = v$.

Definizione 9.7. Dato un cammino π, un *sottocammino* è una sequenza di vertici $\langle x_i, x_{i+1}, ..., x_j \rangle$ con $0 \leq i \leq j \leq k$.

Definizione 9.8. Un *ciclo* è un cammino dove $x_0 = x_k$; un ciclo si dice *semplice* se tutti i vertici intermedi sono distinti.

Definizione 9.9. La *distanza* $\delta(u, v)$ tra due vertici u e v è il numero minimo di archi di un qualsiasi cammino da u a v; se v non è raggiungibile da u, allora $\delta(u, v) = \infty$.

Definizione 9.10. Se un cammino π da u a v è tale che $length(\pi) = \delta(u, v)$, allora π è un *cammino minimo.*

Definizione 9.11. Un grafo non orientato $G = (V, E)$ è *connesso* se, per ogni $u, v \in V$, esiste un cammino da u a v.

Definizione 9.12. Sia $G = (V, E)$ un grafo non orientato; $V' \subseteq V$ è *componente connessa* se:

- il sottografo indotto da V' è connesso;
- se $V'' \subseteq V$ induce un sottografo connesso e $V' \subseteq V''$, allora $V' = V''$ (*vincolo di massimalità*).

Definizione 9.13. Un *albero libero* è un grafo non orientato, connesso e aciclico.

Proprietà 9.1. Sia $G = (V, E)$ un albero libero (grafo non orientato, aciclico e connesso), con $|V| = n$ e $|E| = m$: allora $m = n - 1$.

Dimostrazione. Procediamo per induzione su n.

Caso base: si ha $n = 1$, ossia l'albero con un nodo (e nessun arco); si ha $m = 0 = n - 1$.

Ipotesi induttiva: assumiamo valida la proprietà per ogni $k < n$.

Passo induttivo: sia $n > 1$; prendiamo $v \in V$ e consideriamo il sottografo indotto da $V \setminus \{v\}$; ovvero consideriamo $G' = (V', E')$ con $V' = V - \{v\}$ e $E' = E \cap (V' \times V')$. Siano $V_1, V_2, ..., V_k$ le componenti connesse di G'. Definiamo n_i la cardinalità di V_i. Osservando che $V_1 \cup V_2 \cup ... \cup V_k = V \setminus \{v\}$ deduciamo che

$$n_1 + n_2 + ... + n_k = n - 1.$$

Sia E_i l'insieme degli archi della componente connessa V_i, definiamo $m_i = |E_i|$. Per definizione, V_i è connessa. Inoltre V_i è aciclica essendo sottografo di un grafo aciclico. Inoltre $|V_i| < n$. Possiamo quindi usare l'ipotesi induttiva s V_i e otteniamo

$$m_i = n_i - 1.$$

Poiché il grafo di partenza G è connesso, v deve essere congiunto con *almeno* un arco a ciascuna componente connessa. Poiché il grafo di partenza G è aciclico, v può essere congiunto da *al più* un arco ad ogni componente connessa (altrimenti si formerebbe un ciclo). Concludiamo che da v escono esattamente k archi, esattamente uno per ogni componente connessa. Ora, poiché

$$E = \left(\bigcup_{i=1}^{k} E_i \right) \cup \{\text{archi uscenti da } v\},$$

otteniamo

$$|E| = \sum_{1=1}^{k} m_i + k = \sum_{i=1}^{k} (n_i - 1) + k = \sum_{i=1}^{k} n_i - \sum_{i=1}^{k} 1 + k = n - 1 - k + k = n - 1$$

come volevasi dimostrare. □

Proprietà 9.2. Sia $G = (V, E)$ un grafo non orientato e aciclico con $|E| = |V| - 1$. allora G è connesso.

Dimostrazione. Consideriamo le componenti connesse di G, siano esse $V_1 \dots V_k$. Definendo $|V_i| = n_i$ abbiamo $\sum_{i=1}^{k} n_i = n$. Consideriamo gli insiemi degli archi delle componenti connesse, siano essi $E_1 \dots E_k$. Definendo $|E_i| = m_i$ abbiamo $\sum_{i=1}^{k} m_i = m$. Poiché G è aciclico, anche ogni sua componente connessa è aciclica. Essendo pure connessa, ogni componente connessa è un albero libero. Grazie alla Proprietà 9.1 possiamo affermare che $m_i = n_i - 1$ per ogni componente connessa $i = 1 \dots k$. Ora abbiamo

$$m = \sum_{i=1}^{k} m_i = \sum_{i=1}^{k} (n_i - 1) = \sum_{i=1}^{k} n_i - \sum_{i=1}^{k} 1 = n - k$$

e otteniamo $m = n - k$. Per ipotesi sappiamo che $m = n - 1$, quindi deve essere $k = 1$. Concludiamo che G ha un'unica componente connessa ed è quindi connesso. □

Proprietà 9.3. Sia $G = (V, E)$ un grafo non orientato e connesso con $|E| = |V| - 1$. allora G è aciclico.

Dimostrazione. Assumiamo per assurdo che G ammetta un ciclo π. Senza perdere di generalità, possiamo assumere che π sia un ciclo semplice. Tale ciclo è composto da k archi e da k nodi distinti (con $k \geq 2$). Osserviamo innanzitutto che non può essere $k = n$, altrimenti avremmo che $E \geq n$ e andremmo contro le ipotesi. Quindi $k < n$. Ci sarà allora un insieme di nodi $N' \subseteq N$ con h nodi del grafo che non fanno parte del ciclo. Ovviamente deve essere $h + k = n$. Poiché il grafo è connesso, deve esserci almeno un nodo in N' connesso con almeno un nodo in N', sia esso u_1. Se ora anche $N' \setminus \{u_1\} \neq \emptyset$, poiché il grafo è connesso, deve esserci almeno un nodo in $N' \setminus \{u_1\}$ connesso con un nodo di π o con u_1, sia esso u_2. Iterando il ragionamento, concludiamo che ognuno dei nodi in N' ha almeno un arco incidente. Quindi i nodi del grafo devono essere almeno $h + k$. Otteniamo quindi

$$|E| \geq n$$

e questo è assurdo perché va contro le ipotesi iniziali. Concludiamo quindi che G non può avere cicli. □

Definizione 9.14. Sia $G = (V, E)$ un grafo non orientato: il *grado* di un vertice $u \in V$ è definito come

$$deg(u) = |\{v \in V : (u, v) \in E\}|$$

Sia ora $G = (V, E)$ un grafo orientato: il *grado entrante* e il *grado uscente* di un vertice $u \in V$ sono definiti, rispettivamente, come

$$deg_{in}(u) = |\{v \in V : (v, u) \in E\}|$$
$$deg_{out}(u) = |\{v \in V : (u, v) \in E\}|$$

Lemma 9.1. *Sia $G = (V, E)$ un grafo non orientato; si ha:*

$$\sum_{u \in V} deg(u) = 2 \cdot |E|$$

Sia ora $G = (V, E)$ un grafo orientato; si ha:

$$\sum_{u \in V} deg_{in}(u) = |E|$$
$$\sum_{u \in V} deg_{out}(u) = |E|$$

9.2 Rappresentazione di grafi

Una struttura dati di tipo grafo è descritta dal seguente schema generale:

Dati: un insieme di vertici (di tipo *vertice*) e di archi (di tipo *arco*).

Operazioni:

`numVertici()` $\rightarrow$ *intero*
Restituisce il numero di vertici presenti nel grafo.

`numArchi()` $\rightarrow$ *intero*
Restituisce il numero di archi presenti nel grafo.

`grado(`*vertice v*`)` $\rightarrow$ *intero*
Restituisce il numero di archi incidenti sul vertice v.

`archiIncidenti(`*vertice v*`)` $\rightarrow \langle arco, arco, ..., arco \rangle$
Restituisce, uno dopo l'altro, gli archi incidenti sul vertice v.

`estremi(`*arco e*`)` $\rightarrow \langle vertice, vertice \rangle$
Restituisce gli estremi x e y dell'arco $e = (x, y)$.

`opposto(`*vertice x, arco e*`)` $\rightarrow$ *vertice*
Restituisce y, l'estremo dell'arco $e = (x, y)$ diverso da x.

`sonoAdiacenti(`*vertice x, vertice y*`)` $\rightarrow$ *booleano*
Restituisce *true* se esiste l'arco (x, y), *false* altrimenti.

`aggiungiVertice(`*vertice v*`)` $\rightarrow$ *void*
Inserisce un nuovo vertice v.

`aggiungiArco(`*vertice x, vertice y*`)` $\rightarrow$ *void*
Inserisce un nuovo arco tra i vertici x e y.

`rimuoviVertice(`*vertice v*`)` $\rightarrow$ *void*
Cancella il vertice v e tutti gli archi ad esso incidenti.

`rimuoviArco(`*arco e*`)` $\rightarrow$ *void*
Cancella l'arco e.

9.2.1 Lista di archi

È una rappresentazione che si basa sull'utilizzo di:

- una struttura per rappresentare i vertici (tipo arraylist);
- una lista per rappresentare gli archi.

9.2.2 Liste di adiacenza

In questa rappresentazione, ogni vertice viene associato ad una lista contenente i suoi vertici adiacenti (struttura realizzata con array di liste).

9.2.3 Matrice di adiacenza

È una rappresentazione basata sull'uso di una matrice $n \times n$ le cui righe e colonne sono indicizzate dai vertici del grafo:

$$M[u,v] = \begin{cases} 1 & \text{se } (u,v) \in E \\ 0 & \text{altrimenti} \end{cases}$$

Si tratta di una rappresentazione utile per il seguente fatto:

Proprietà 9.4. Sia $M_k = \underbrace{M \cdot M \cdot ... \cdot M}_{kvolte}$: si ha $M^k[u,v] = 1$ se e solo se esiste un cammino di k vertici che collega u a v.

9.2.4 Confronto tra le rappresentazioni

Operazione	**Lista di archi**	**Lista di adiacenza**	**Matrice di adiacenza**
`grado`(v)	$O(m)$	$O(deg(v))$	$O(n)$
`archiIncidenti`(v)	$O(m)$	$O(deg(v))$	$O(n)$
`sonoAdiacenti`(x, y)	$O(m)$	$O(\min\{deg(x), deg(y)\})$	$O(1)$
`aggiungiVertice`(v)	$O(1)$	$O(1)$	$O(n^2)$
`aggiungiArco`(x, y)	$O(1)$	$O(1)$	$O(1)$
`rimuoviVertice`(v)	$O(m)$	$O(m)$	$O(n^2)$
`rimuoviArco`(e)	$O(m)$	$O(deg(x) + deg(y))$	$O(1)$

Tabella 9.1: Tempi di esecuzione

	Lista di archi	**Lista di adiacenza**	**Matrice di adiacenza**
Spazio	$O(m+n)$	$O(m+n)$	$O(n^2)$

Tabella 9.2: Spazio occupato