Università Roma Tre – Dipartimento di Matematica e Fisica

Percorso Abilitante Speciale
Classe A048 – Matematica Applicata

Corso di Informatica

Algoritmi su Grafi

Marco Liverani
(liverani@mat.uniroma3.it)

Sommario

- Grafi: alcune definizioni fondamentali
- Alberi: definizione e proprietà
- Rappresentazione di un grafo o di un albero
- Problemi su grafi e alberi
- Algoritmo per la visita in ampiezza di un grafo
- Algoritmo per la visita in profondità di un grafo

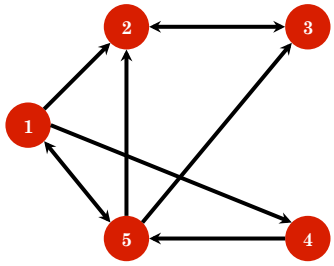
Grafi: definizioni principali 1

- Un grafo G è un oggetto matematico definito come una **coppia ordinata** $G=(V,E)$, dove $V=\{v_1, \ldots, v_n\}$ è l'insieme dei ***vertici*** (o *nodi*, o *punti*) ed E è l'insieme degli ***spigoli*** (o *archi*, o *lati*).
- E è una relazione binaria su V, ovvero un sottoinsieme del prodotto cartesiano $V \times V$: $e=(v_i,v_j) \in E \subseteq V \times V$.
- Se il grafo G è ***non orientato*** la relazione E è simmetrica: $(v_i,v_j) \in E \Leftrightarrow (v_j,v_i) \in E$.
- Se il grafo G è ***orientato*** $(v_i,v_j) \in E$ non implica necessariamente che $(v_j,v_i) \in E$.

Grafi: definizioni principali 2

- Sia $G=(V,E)$. Notazione:
  - Numero di vertici: $|V| = \boldsymbol{n}$;
  - Numero di spigoli: $|E| = \boldsymbol{m}$.
- Sia $G=(V,E)$ un grafo orientato. Se $(u,v) \in E$ allora si dice che lo spigolo è incidente i vertici u e v; lo spigolo è ***uscente*** dal vertice u ed ***entrante*** nel vertice v.
- Il ***grado*** di un vertice è dato dal numero di spigoli incidenti su di esso.
- Se $(u,v) \in E$ allora v è ***adiacente*** ad u (se il grafo è orientato allora non è detto che u sia adiacente a v).

Grafi: definizioni principali 3



$G = (V,E)$ è un grafo orientato
$V = \{1, 2, 3, 4, 5\} \Rightarrow |V| = 5$
$E = \{(1,2), (1,4), (1,5), (2,3), (3,2), (4,5), (5,1), (5,2), (5,3)\} \Rightarrow$
$\Rightarrow |E| = 9$

Cammini

- Un ***cammino*** di lunghezza k da un vertice u ad un vertice u' in G è una sequenza $(v_0, v_1, \ldots, v_k)$ di vertici tali che $u=v_0$, $u'=v_k$ e $(v_{i-1}, v_i) \in E$, per $i=1, \ldots, k$.
- Se esiste un cammino p da u a u', allora si dice che u' è ***raggiungibile*** da u.
- Un cammino è ***semplice*** se tutti i suoi vertici sono distinti. Un cammino da u a u' è un ***ciclo*** se $u=u'$. Se il ciclo è di lunghezza 1 allora è un ***cappio***.
- Un grafo G privo di cicli è ***aciclico***.

Grafi connessi, componenti connesse

- Un grafo **non** orientato è ***connesso*** se ogni coppia di vertici è collegata da un cammino in G.
- Le ***componenti connesse*** di un grafo sono le classi di equivalenza dei vertici sotto la relazione "raggiungibile da".
- Un grafo orientato è ***fortemente connesso*** se ogni coppia di vertici è collegata da un cammino. Le ***componenti fortemente connesse*** di un grafo sono le classi di equivalenza dei vertici sotto la relazione "mutuamente raggiungibile".

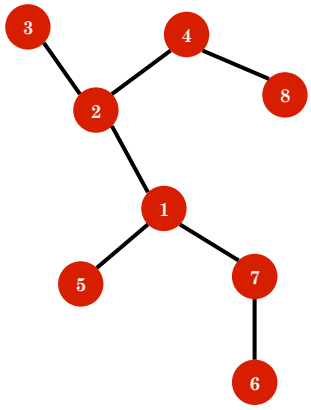
Alberi liberi

- Un ***albero*** è un grafo *non orientato*, *connesso* e *aciclico*.
- Un grafo non orientato e aciclico, ma non connesso, è una ***foresta***.
- Sia $G=(V,E)$ un grafo non orientato. Affermazioni equivalenti (*Teorema di caratterizzazione degli alberi*):
  - G è un albero;
  - due vertici qualsiasi di G sono connessi da un unico cammino semplice;
  - G è connesso, ma se qualunque spigolo di G venisse rimosso, allora G diventerebbe non connesso;
  - G è connesso e $|E|=|V|-1$;
  - G è aciclico e $|E|=|V|-1$;
  - G è aciclico, ma se venisse aggiunto uno spigolo qualsiasi il grafo risultante non sarebbe più aciclico.

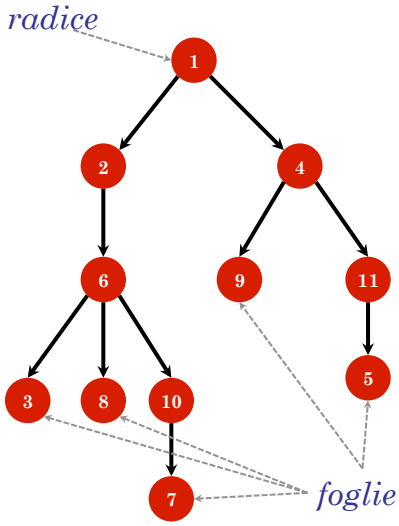
Alberi radicati

- Un ***albero radicato*** è un albero in cui uno dei vertici (detto ***radice***) si distingue dagli altri.
- Se u è un vertice dell'albero radicato A ed r è la sua radice, allora ogni vertice v sull'unico cammino da r a u è un ***antenato*** di u e u è un ***discendente*** di v.
- Se (u,v) è uno spigolo di un albero radicato, allora si dice che u è il ***padre*** di v e che v è un ***figlio*** di u. Se anche (u,w) è uno spigolo dell'albero, allora v e w sono ***cugini***.
- Un vertice senza figli è una ***foglia*** dell'albero.
- La lunghezza del cammino più lungo dalla radice ad una foglia è l'***altezza*** dell'albero.

Alberi: esempi



Albero (libero)

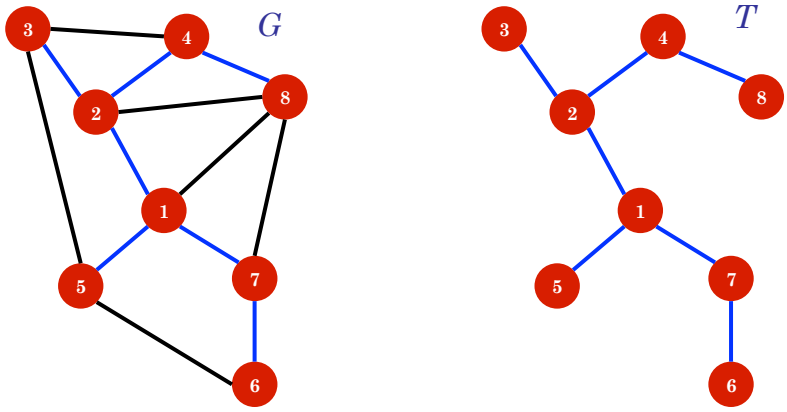


Albero radicato orientato

Alberi ricoprenti (spanning tree) 1

- Sia $G=(V,E)$ un grafo connesso. Sia $T=(V,E')$ un sottografo di G connesso. Se T è un albero allora si dice che T è un ***albero ricoprente*** (***spanning tree***) di G.
  Un grafo G può avere anche più di un albero ricoprente.
- Se G non è connesso allora l'insieme degli alberi che ricoprono le sue componenti connesse è una ***foresta ricoprente*** (***spanning forest***).

Alberi ricoprenti (spanning tree) 2

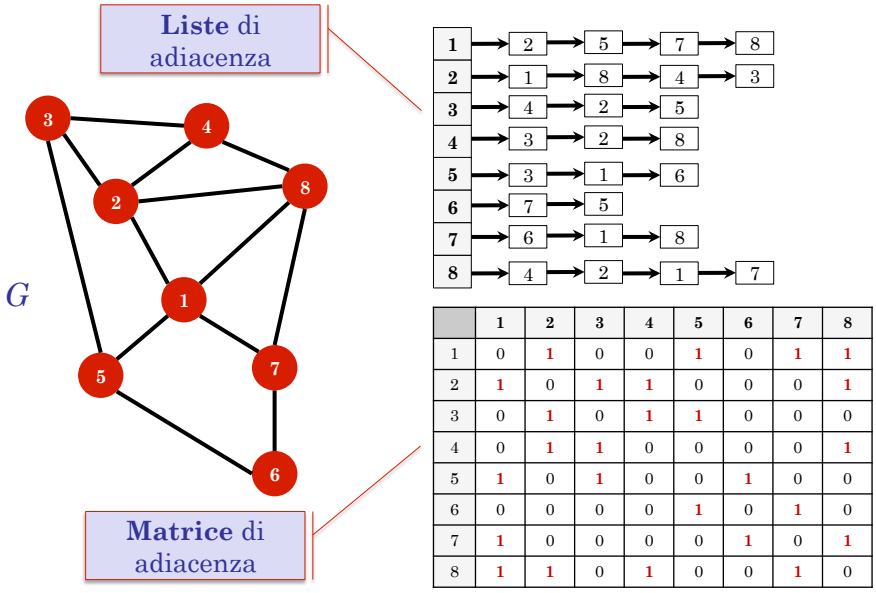


Grafo $G=(V,E)$ e un albero ricoprente $T=(V,E')$ con $E' \subset E$

Rappresentazione di grafi ed alberi 1

- Nella memoria della macchina un grafo può essere rappresentato mediante:
  - ***Liste di adiacenza***: per ogni vertice $u \in V$ viene costruita una lista dei veritici $v_1, \ldots, v_k$ adiacenti a u, ossia tali che $(u,v_i) \in E$ per ogni $i=1, \ldots, k$.
  - ***Matrice di adiacenza***: il grafo viene rappresentato con una matrice quadrata di ordine $n=|V|$, in cui $M_{i,j}=1$ se $(v_i,v_j) \in E$, e $M_{i,j}=0$ altrimenti.
- Per grafi *sparsi* (con pochi spigoli rispetto al numero dei vertici) è conveniente la rappresentazione mediante liste di adiacenza.

Rappresentazione di grafi ed alberi 2



	1	2	3	4	5	6	7	8
1	0	1	0	0	1	0	1	1
2	1	0	1	1	0	0	0	1
3	0	1	0	1	1	0	0	0
4	0	1	1	0	0	0	0	1
5	1	0	1	0	0	1	0	0
6	0	0	0	0	1	0	1	0
7	1	0	0	0	0	1	0	1
8	1	1	0	1	0	0	1	0

Problemi ed algoritmi sui grafi

- Esiste un vasto insieme di problemi interessanti sui grafi, spesso dotati di algoritmi risolutori efficienti ed eleganti.
- Alcuni esempi:
  - Visita in ampiezza ed in profondità di un grafo;
  - Ricerca del minimo albero ricoprente (*minimum spanning tree* – algoritmo di Kruskal);
  - Ricerca del cammino minimo per raggiungere ogni altro vertice a partire da una sorgente (algoritmo di Dijkstra);
  - Calcolo del massimo flusso su una rete (algoritmo di Ford e Fulkerson, algoritmo di Belman e Ford, ecc.).

Problemi di ottimizzazione

- Alcuni problemi consistono nella ricerca di una configurazione del grafo tale da massimizzare o minimizzare una determinata proprietà. Questi problemi vanno sotto il nome di ***problemi di ottimizzazione***.
- Formalmente viene definita una ***funzione obiettivo*** f i cui valori sono calcolati sulla base di particolari configurazioni del grafo (es.: un peso associato ai vertici o agli spigoli del grafo, il numero di vertici in ogni componente, ecc.). Si deve individuare una configurazione del grafo tale da rendere massimo o minimo il valore di f.

Visita di un grafo

- Partendo da un vertice di un grafo G si vogliono percorrere alcuni spigoli del grafo in modo da raggiungere (se possibile) ogni altro vertice del grafo
- Perché potrebbe non essere possibile?
  - Perché il grafo non è connesso
  - Perché gli spigoli del grafo non sono orientati in modo opportuno
- Due strategie fondamentali:
  - ***visita in ampiezza***
  - ***visita in profondità***

Visita in ampiezza 1

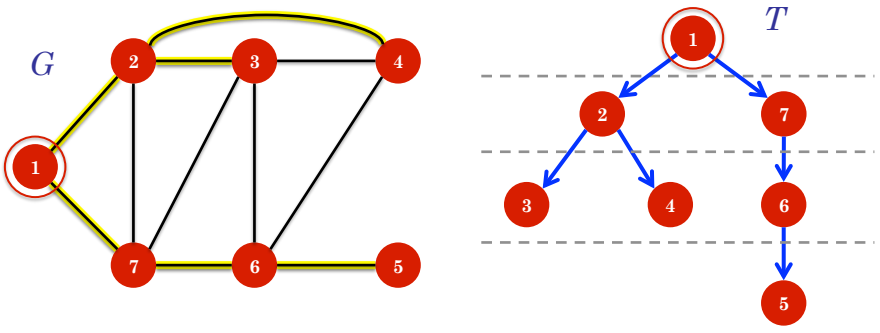
- Partendo dalla "sorgente" s della visita (il punto di partenza) mi allontano meno possibile da s: visito un vertice a distanza k da s solo quando ho visitato tutti i vertici a distanza k–1 da s
- Se raggiungo tutti i vertici del grafo costruisco un albero ricoprente con radice in s, costituito dai ***cammini più brevi*** per raggiungere ogni vertice del grafo a partire da s

Visita in ampiezza 2

1. Colora tutti i vertici $v_1, \ldots, v_n$ di bianco: $c(v_k) = 1$, per $k=1, 2, \ldots, n$
2. Poni $d(v_k) = \infty$, per $k=1, 2, \ldots, n$
3. Poni $\pi(v_k) = \text{null}$, per $k=1, 2, \ldots, n$
4. Poni $d(s) = 0$, $c(s) = 2$ (grigio), $Q = \{s\}$
5. Fintanto che $Q \neq \emptyset$ ripeti:
6. Estrai da Q il vertice u
7. Per ogni vertice v adiacente a u ripeti:
8. Se $c(v) = 1$ (v è bianco) allora
9. $c(v) = 2$, $d(v) = d(u) + 1$, $\pi(v) = u$
10. Fine iterazione
11. $c(u) = 3$ (nero)
12. Fine iterazione

Complessità: $O(n+m)$

Visita in ampiezza 3



Grafo $G=(V,E)$ e un albero di visita in profondità $T=(V,E')$, suddiviso su livelli di diversa distanza dalla sorgente

Visita in profondità 1

- Partendo dalla "sorgente" s della visita (il punto di partenza) mi allontano il più possibile da s
- Non calcolo i cammini più brevi, ma il procedimento è utile per risolvere molti altri problemi:
  - presenza di cicli
  - connessione del grafo
  - ordinamento topologico dei vertici
  - ecc.

Visita in profondità 2

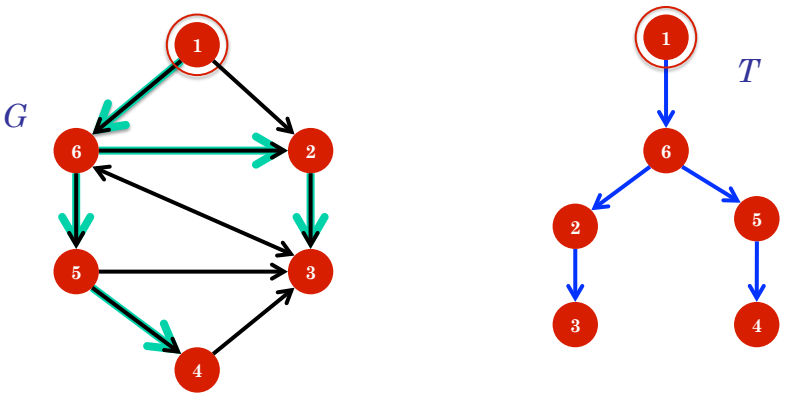
1. Colora tutti i vertici $v_1, \ldots, v_n$ di bianco: $c(v_k) = 1$, per $k=1, 2, \ldots, n$
2. Poni $\pi(v_k) = \text{null}$, per $k=1, 2, \ldots, n$
3. VISITA(G, s)

Complessità: $O(n+m)$

VISITA(G, u)

1. Poni $c(u) = 2$ (grigio)
2. Per ogni vertice v adiacente a u ripeti:
3. Se $c(v) = 1$ (v è bianco) allora
4. $\pi(v) = u$, VISITA(G, v)
5. Fine iterazione
6. Poni $c(u) = 3$ (nero)

Visita in profondità 3



Grafo orientato $G=(V,E)$ e un albero ricoprente $T=(V,E')$ ottenuto con la visita in profondità