

Algoritmi e Strutture di Dati

I grafi

rappresentati con matrici e liste di adiacenza

m.patrignani

210-grafi-matrici-e-liste-02 copyright ©2019 maurizio.patrignani@uniroma3.it

Nota di copyright

- queste slides sono protette dalle leggi sul copyright
- il titolo ed il copyright relativi alle slides (inclusi, ma non limitatamente, immagini, foto, animazioni, video, audio, musica e testo) sono di proprietà degli autori indicati sulla prima pagina
- le slides possono essere riprodotte ed utilizzate liberamente, non a fini di lucro, da università e scuole pubbliche e da istituti pubblici di ricerca
- ogni altro uso o riproduzione è vietata, se non esplicitamente autorizzata per iscritto, a priori, da parte degli autori
- gli autori non si assumono nessuna responsabilità per il contenuto delle slides, che sono comunque soggette a cambiamento
- questa nota di copyright non deve essere mai rimossa e deve essere riportata anche in casi di uso parziale

210-grafi-matrici-e-liste-02 copyright ©2019 maurizio.patrignani@uniroma3.it

Contenuto

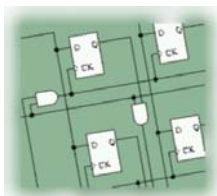
- Definizione di grafi diretti e indiretti
- Rappresentazione di grafi tramite:
 - matrici di adiacenza
 - liste di adiacenza
- Esercizi su grafi

210-grafi-matrici-e-liste-02

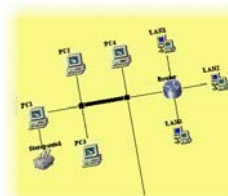
copyright ©2019 maurizio.patrignani@uniroma3.it

Grafi nelle applicazioni

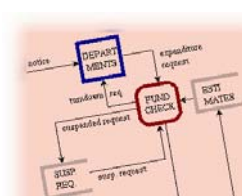
- Molti diagrammi utilizzati in ingegneria sono dei grafi



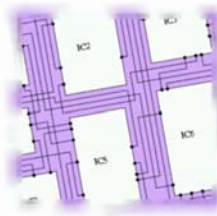
scemi circuitali



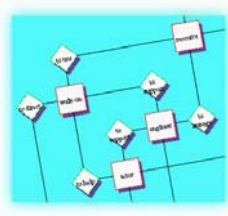
topologie di rete



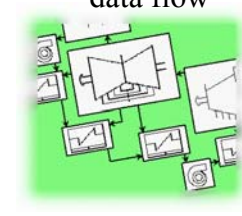
data flow



circuiti integrati



diagrammi ER



impianti industriali

210-grafi-matrici-e-liste-02

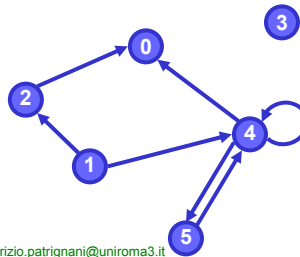
copyright ©2019 maurizio.patrignani@uniroma3.it

Grafi diretti

- Un *grafo orientato* (o *diretto*) $G=(V,E)$ è costituito da un insieme di nodi V e un insieme di archi E
 - ogni arco è una coppia ordinata di nodi (u,v)
- Denotiamo con n il numero dei nodi ($n = |V|$) e con m il numero degli archi ($m = |E|$)
 - si ha sempre $m \in O(n^2)$
- Esempio:

$$V = \{0, 1, 2, 3, 4, 5\}$$

$$E = \{(2,0) (1,2) (4,0) (4,4) (4,5) (5,4) (1,4)\}$$



210-grafi-matrici-e-liste-02

copyright ©2019 maurizio.patrignani@uniroma3.it

Grafi diretti e relazioni

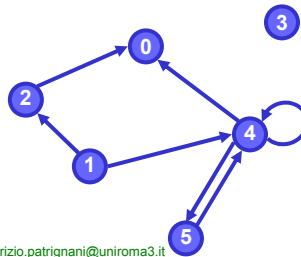
- La versatilità dei grafi diretti deriva dal fatto che essi corrispondono a relazioni binarie
- Per esempio
 - contatti tra utenti di una rete di telefonia
 - dipendenze tra invocazioni di metodi in un software
 - partecipazioni di aziende nel capitale di altre
 - rapporti di eredità
 - rapporti di precedenza tra attività
 - reti sociali
 - ...

210-grafi-matrici-e-liste-02

copyright ©2019 maurizio.patrignani@uniroma3.it

Archi uscenti ed entranti

- Dato un nodo u
 - un suo *arco uscente* è un arco $(u,v) \in E$
 - un suo *arco entrante* è un arco $(v,u) \in E$
 - un *nodo adiacente* è un nodo v per cui esiste $(u,v) \in E$
 - il suo *grado di uscita* è il numero dei suoi archi uscenti
 - il suo *grado di ingresso* è il numero dei suoi archi entranti
- Un nodo u è detto...
 - *sorgente* se non ha archi entranti
 - *pozzo* se non ha archi uscenti

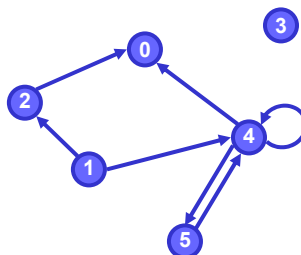


210-grafi-matrici-e-liste-02

copyright ©2019 maurizio.patrignani@uniroma3.it

Cammini

- Un *cammino* (è sottinteso che sia diretto) è una sequenza di nodi u_1, u_2, \dots, u_k tali che per $i=1,2, \dots, k-1$ esistono gli archi (u_i, u_{i+1})
- Il cammino è detto *semplice* se tutti i suoi nodi sono distinti
- Il numero di archi è la *lunghezza* del cammino

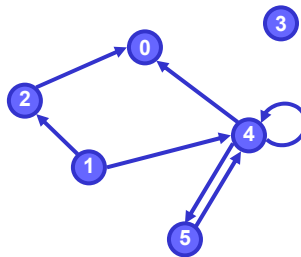


210-grafi-matrici-e-liste-02

copyright ©2019 maurizio.patrignani@uniroma3.it

Cicli

- Un *ciclo* è un cammino (non semplice) in cui il primo e l'ultimo nodo coincidono
- Un ciclo è detto *semplice* se il primo e l'ultimo nodo sono gli unici nodi che coincidono
- Un *cappio* (o *loop*) è un ciclo di un solo arco (e un solo nodo)
- Un *grafo semplice* è un grafo senza cappi
- Un grafo diretto è *aciclico* se non ha cicli (diretti)



210-grafi-matrici-e-liste-02

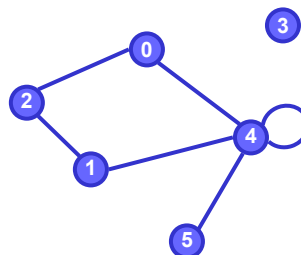
copyright ©2019 maurizio.patrigiani@uniroma3.it

Grafi non orientati

- In un grafo *non orientato* (o *non diretto*) $G=(V,E)$ l'insieme degli archi E è un insieme di coppie non ordinate (u,v)
 - (u,v) e (v,u) rappresentano lo stesso arco
- Graficamente si conviene di rappresentare una sola linea tra i nodi u e v

$$V = \{0, 1, 2, 3, 4, 5\}$$

$$E = \{(0,2) (0,4) (1,2) (1,4) (4,4) (4,5)\}$$



210-grafi-matrici-e-liste-02

copyright ©2019 maurizio.patrigiani@uniroma3.it

Il tipo astratto di dato grafo

- **Domini**
 - il dominio di interesse è l'insieme G dei grafi (diretti o non diretti)
 - dominio di supporto: l'insieme degli iteratori NI per i nodi
 - dominio di supporto: l'insieme degli iteratori AI per gli archi
 - dominio di supporto: i booleani $B = \{true, false\}$
- **Costanti**
 - il grafo vuoto
 - gli iteratori non validi per nodi e archi
- **Operazioni**
 - trova il primo nodo del grafo:
 - trova il prossimo nodo:
 - trova il primo arco di un nodo:
 - trova il nodo adiacente tramite l'arco:
 - trova il prossimo arco:
 - determina se due nodi sono adiacenti
 - aggiunge un nodo al grafo:
 - aggiunge un arco tra due nodi:

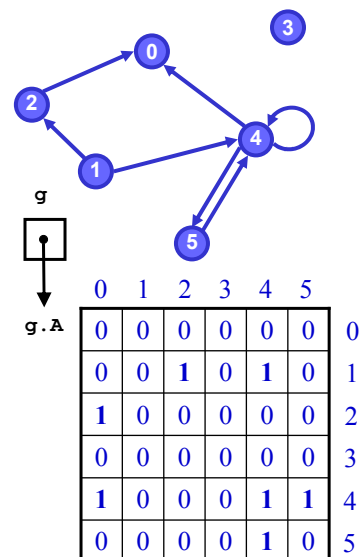
$FIRST_NODE: G \rightarrow NI$
 $NEXT_NODE: G \times NI \rightarrow NI$
 $FIRST_EDGE: G \times NI \rightarrow AI$
 $ADJ_NODE: G \times NI \times AI \rightarrow NI$
 $NEXT_EDGE: G \times NI \times AI \rightarrow AI$
 $ARE_ADJ: G \times NI \times NI \rightarrow B$
 $ADD_NODE: G \rightarrow NI$
 $ADD_EDGE: G \times NI \times NI \rightarrow AI$

210-grafi-matrici-e-liste-02

copyright ©2019 maurizio.patrignani@uniroma3.it

Realizzazione mediante matrice di adiacenza

- Rappresentazione preferita per grafi densi
 - cioè per i quali il numero degli archi m è prossimo ad n^2
- Consente di sapere rapidamente se c'è un arco tra due nodi
- Usa una matrice (o un array di array) in cui l'elemento in posizione (i,j) segnala se esiste l'arco (i,j)
- Occupa $\Theta(n^2)$ spazio

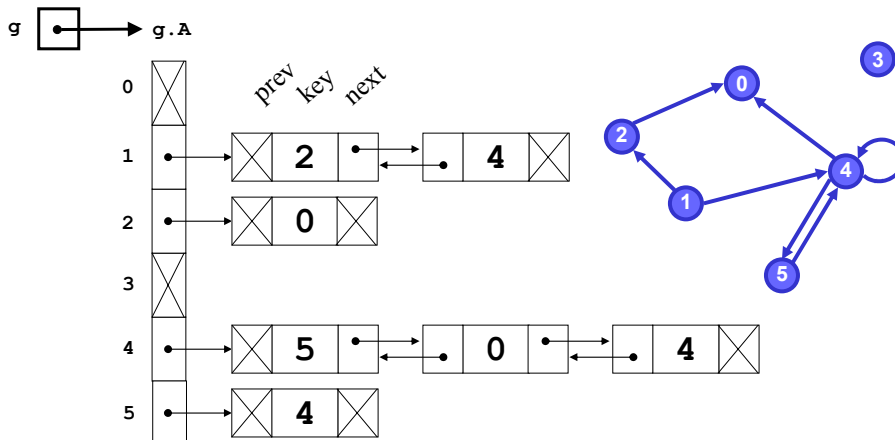


210-grafi-matrici-e-liste-02

copyright ©2019 maurizio.patrignani@uniroma3.it

Realizzazione tramite liste di adiacenza

- Si fa uso di un array A di liste doppiamente concatenate
 - generalmente si mette nell'array direttamente il riferimento al primo elemento della lista



210-grafi-matrici-e-liste-02

copyright ©2019 maurizio.patrignani@uniroma3.it

Rappresentazione dei grafi: liste di adiacenza

- Questa rappresentazione occupa spazio $O(n) + O(m)$
 - nel caso peggiore, siccome $m = O(n^2)$ utilizza uno spazio $O(n^2)$ come le rappresentazioni con matrici di adiacenza
 - in numerose applicazioni, però, $m \in O(n)$
 - in questo caso le rappresentazioni con liste di adiacenza sono preferibili
- La lista di adiacenza di un nodo può essere lunga $O(n)$
- Percorrendo tutte le liste di adiacenza di tutti i nodi si impiega un tempo $O(n) + O(m)$
 - che diventa $O(n^2)$ se il grafo è denso

210-grafi-matrici-e-liste-02

copyright ©2019 maurizio.patrignani@uniroma3.it

Realizzazioni del tipo astratto di dato grafo

- Matrice di adicenza
 - l'iteratore per un nodo è un intero
 - è valido se è nell'intervallo legittimo per l'array g.A
 - l'iteratore per un arco è ancora un intero
 - è valido se è nell'intervallo legittimo e se la cella corrispondente dell'array g.A contiene un uno
 - si rinuncia a realizzare efficientemente l'operazione di aggiunta di un nodo
- Liste di adiacenza
 - l'iteratore per un nodo è un intero
 - è valido se è nell'intervallo legittimo per l'array g.A
 - l'iteratore per un arco è un riferimento ad un elemento di una lista
 - l'iteratore non valido è NULL
 - anche in questo caso l'operazione di aggiunta di un nodo richiede una gestione non semplice e non sempre efficiente

210-grafi-matrici-e-liste-02 copyright ©2019 maurizio.patrignani@uniroma3.it

Rappresentazione di grafi non orientati

- Per ogni arco non orientato (u,v) vengono rappresentati i due archi orientati (u,v) e (v,u)
- Nel caso di matrice di adiacenza
 - la matrice è simmetrica
- Nel caso di liste di adiacenza
 - se la lista del nodo i contiene il nodo j , allora la lista del nodo j contiene il nodo i

210-grafi-matrici-e-liste-02 copyright ©2019 maurizio.patrignani@uniroma3.it

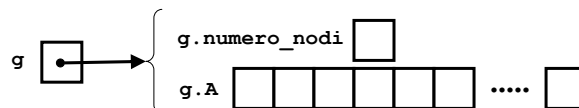
Grafi pesati sugli archi

- Sono grafi in cui ad ogni arco e è associato un peso w_e
- Nella rappresentazione tramite matrici di adiacenza si usano i valori dei pesi al posto degli uni:
 - si assume che non esistano archi con peso zero e si usa lo zero per rappresentare l'assenza dell'arco
 - si usa w_e per rappresentare un arco di peso w_e
- Nella rappresentazione tramite liste di adiacenza
 - ogni elemento della lista ha, oltre all'indice del nodo adiacente, anche un attributo weight con valore w_e

210-grafi-matrici-e-liste-02 copyright ©2019 maurizio.patrignani@uniroma3.it

Realizzazioni in linguaggio C

- La rappresentazione dei grafi in linguaggio C segue gli stessi paradigmi della rappresentazione in pseudocodifica
 - tuttavia, poiché gli array in linguaggio C non hanno un campo “length” che ne riveli la lunghezza occorre aggiungere un campo “numero_nodi” ad entrambe le rappresentazioni



210-grafi-matrici-e-liste-02 copyright ©2019 maurizio.patrignani@uniroma3.it

Esercizi su matrici di adiacenza

- Dato un grafo g rappresentato tramite una matrice di adiacenza $g.A$ (un array di array)
 1. scrivi una procedura `LISTE(g)` che ne restituisca la sua rappresentazione mediante un array di liste di adiacenza doppiamente concatenate
 2. scrivi una procedura `GRADO_USCITA(g,u)` che calcoli il grado di uscita del nodo con indice u
 3. scrivi una procedura `GRADO_INGRESSO(g,u)` per il calcolo del grado di ingresso del nodo con indice u
 4. scrivi una procedura `GRADO_USCITA_MEDIO(g)` per il calcolo del grado di uscita medio dei nodi del grafo
 5. scrivi una procedura `GRAFO_SEMPLICE(g)` che verifica se il grafo è semplice (privo di cappi)
- Discuti la complessità degli algoritmi che hai proposto

210-grafi-matrici-e-liste-02 copyright ©2019 maurizio.patrignani@uniroma3.it

Esercizi su liste di adiacenza 1/3

- Dato un grafo diretto g rappresentato tramite un array $g.A$ di liste di adiacenza doppiamente concatenate
 6. scrivi una procedura `MATRICE(g)` che ne restituisca la sua rappresentazione mediante una matrice di adiacenza
 7. scrivi una procedura `GRADO_USCITA(g,u)` che calcoli il grado di uscita del nodo con indice u
 8. scrivi una procedura `GRADO_INGRESSO(g,u)` per il calcolo del grado di ingresso del nodo con indice u
 9. scrivi una procedura `GRADO_USCITA_MEDIO(g)` per il calcolo del grado di uscita medio dei nodi del grafo
 10. scrivi una procedura `GRAFO_SEMPLICE(g)` che verifica se il grafo è semplice (privo di cappi)
- Discuti la complessità degli algoritmi che hai proposto

210-grafi-matrici-e-liste-02 copyright ©2019 maurizio.patrignani@uniroma3.it

Esercizi su liste di adiacenza 2/3

- Dato un grafo diretto g rappresentato tramite un array $g.A$ di liste di adiacenza doppiamente concatenate
 11. scrivi lo pseudocodice della funzione $VERIFICA_ARCO(g,u,v)$ che restituisce **true** se esiste l'arco che va dal nodo identificato dall'indice u al nodo indentificato dall'indice v e **false** altrimenti
 12. scrivi lo pseudocodice della funzione $VERIFICA_NON_ORIENTATO(g)$ che restituisce **true** se il grafo presenta un arco (u,v) per ogni arco (v,u) e **false** altrimenti
 - puoi utilizzare la funzione $VERIFICA_ARCO(g,u,v)$
 13. scrivi lo pseudocodice della funzione $VERIFICA_POZZO(g,u)$ che restituisce **true** se il nodo identificato dall'indice u non ha archi uscenti, **false** altrimenti
 14. scrivi lo pseudocodice della funzione $VERIFICA_SORGENTE(g,u)$ che restituisce **true** se il nodo identificato dall'indice u non ha archi entranti, **false** altrimenti
- Discuti la complessità degli algoritmi che hai proposto

210-grafi-matrici-e-liste-02 copyright ©2019 maurizio.patrignani@uniroma3.it

Esercizi su liste di adiacenza 3/3

- Dati due grafi $g1$ e $g2$ rappresentati tramite array di liste di adiacenza doppiamente concatenate
 15. scrivi lo pseudocodice della funzione $VERIFICA_UNIONE(g1,g2)$ che verifica che tra ogni possibile coppia di nodi ci sia un arco in $g1$ o in $g2$ (o in entrambi)
 - puoi supporre che $g1$ e $g2$ abbiano lo stesso numero di nodi ($g1.A.length=g2.A.length$)
 16. scrivi lo pseudocodice della funzione $VERIFICA_POZZI_E_SORGENTI(g1,g2)$ che restituisce **true** se tutti i pozzi di $g1$ sono sorgenti di $g2$ e tutte le sorgenti di $g1$ sono pozzi di $g2$ e restituisce **false** altrimenti
 - puoi supporre che $g1$ e $g2$ abbiano lo stesso numero di nodi
 - puoi utilizzare le funzioni $VERIFICA_POZZO(g,u)$ e $VERIFICA_SORGENTE(g,u)$
- Discuti la complessità degli algoritmi che hai proposto

210-grafi-matrici-e-liste-02 copyright ©2019 maurizio.patrignani@uniroma3.it