algoritmi e strutture di dati

grafi

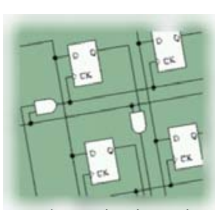
m.patrignani

contenuto

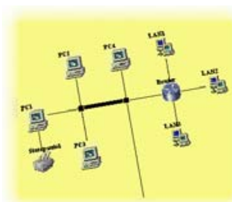
- definizione di grafi diretti e indiretti
- rappresentazione di grafi
  - matrici di adiacenza
  - liste di adiacenza
- esercizi su grafi

grafi nelle applicazioni

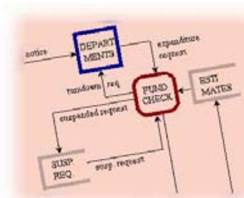
- molti diagrammi utilizzati in ingegneria sono dei grafi



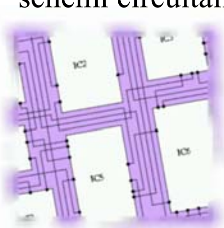
schemi circuitali



topologie di rete

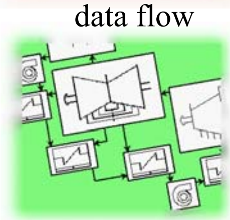


data flow



circuiti integrati

diagrammi ER



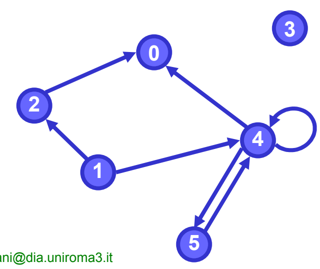
impianti industriali

130-grafi-05 copyright ©2013 patrignani@dia.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:

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

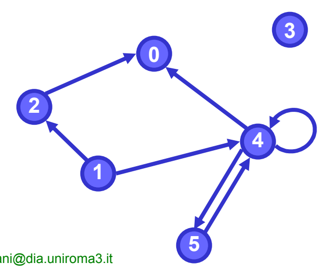


grafi diretti e relazioni

- la versatilità dei grafi diretti deriva dal fatto che essi corrispondo 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
  - …

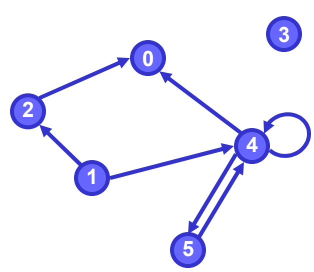
archi uscenti ed entranti

- 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



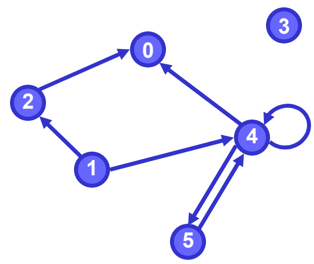
cammini

- un *cammino* (è sottinteso che sia diretto) è una sequenza di nodi $u_1, u_2, \ldots, u_k$ tali che per i=1,2, …,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



cicli

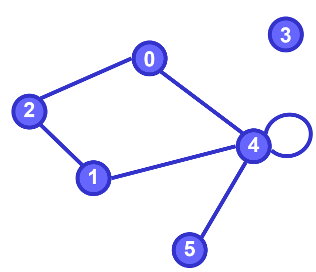
- un *ciclo* è un cammino (non semplice) in cui il primo e l'ultimo nodo coincidono
- un ciclo è detto *semplice* se il primo e l'ultimo nodo sono gli unici nodi che coincidono
- un *cappio* (o *loop*) è un ciclo di un solo arco (e un solo nodo)
- un *grafo semplice* è un grafo senza cappi
- un grafo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



grafi non orientati

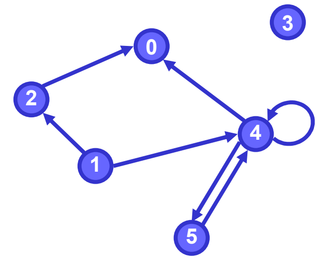
- in un grafo *non orientato* l'insieme degli archi è un insieme di coppie non ordinate (u,v)
  - (u,v) e (v,u) rappresentano lo stesso arco
- graficamente si conviene di rappresentare una sola linea tra i nodi u e v

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



rappresentazione di grafi: matrici di adiacenza

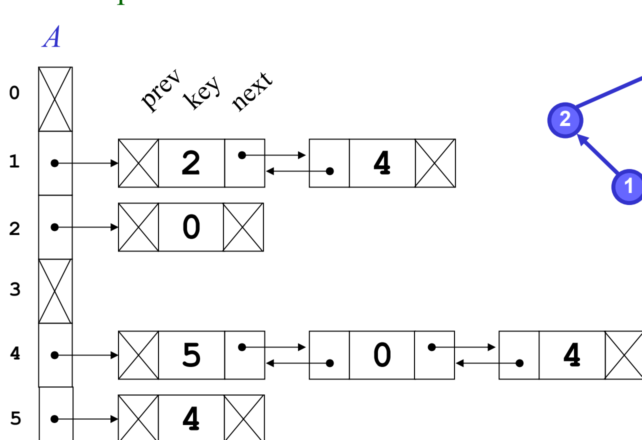
- rappresentazione preferita per grafi densi
  - cioè per i quali il numero degli archi m è prossimo ad n^2
- consente di sapere rapidamente se c'è un arco tra due nodi
- usa una matrice (o un array di array) in cui l'elemento in posizione (i,j) segnala se esiste l'arco (i,j)
- occupa $\Theta(n^2)$ spazio

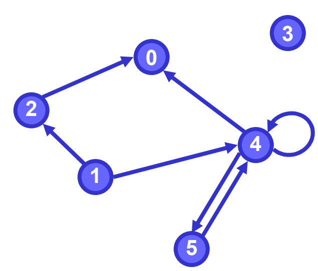


	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	**1**	0	**1**	0
2	**1**	0	0	0	0	0
3	0	0	0	0	0	0
4	**1**	0	0	0	**1**	**1**
5	0	0	0	0	**1**	0

rappresentazione dei grafi: 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





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

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

grafi pesati sugli archi

- sono grafi in cui ad ogni arco e è associato un peso w_e
- nella rapprentazione tramite matrici di adiacenza si usano i valori:

  0 per rappresentare l'assenza dell'arco

  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 il dato satellite w_e

esercizi su matrici di adiacenza

- dato un grafo rappresentato tramite una matrice di adiacenza A (un array di array)
  - scrivi una procedura `LISTE`(A) che ne costruisca la sua rappresentazione mediante un array di liste di adiacenza doppiamente concatenate
  - scrivi una procedura `GRADO-USCITA`(A,u) per il calcolo del grado di uscita del nodo con indic u
  - scrivi una procedura `GRADO-INGRESSO`(A,u) per il calcolo del grado di ingresso del nodo con indice u
  - scrivi una procedura `GRADO-USCITA-MEDIO`(A) per il calcolo del grado di uscita medio dei nodi del grafo
  - scrivi una procedura `GRAFO-SEMPLICE`(A) che verifica se il grafo è semplice (privo di cappi)
- discuti la complessità degli algoritmi che hai proposto

esercizi su liste di adiacenza 1/3

- dato un grafo diretto rappresentato tramite un array *A* di liste di adiacenza doppiamente concatenate
  - scrivi una procedura `MATRICE`(*A*) che ne costruisca la sua rappresentazione mediante un array di liste di adiacenza doppiamente concatenate
  - scrivi una procedura `GRADO-USCITA`(*A*,*u*) per il calcolo del grado di uscita del nodo con indice *u*
  - scrivi una procedura `GRADO-INGRESSO`(*A*,*u*) per il calcolo del grado di ingresso del nodo con indice *u*
  - scrivi una procedura `GRADO-USCITA-MEDIO`(*A*) per il calcolo del grado di uscita medio dei nodi del grafo
  - scrivi una procedura `GRAFO-SEMPLICE`(*A*) che verifica se il grafo è semplice (privo di cappi)
- discuti la complessità degli algoritmi che hai proposto

esercizi su liste di adiacenza 2/3

- dato un grafo diretto rappresentato tramite un array *A* di liste di adiacenza doppiamente concatenate
  - scrivi lo pseudocodice della funzione `VERIFICA-ARCO`(*A*,*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
  - scrivi lo pseudocodice della funzione `VERIFICA-NON-ORIENTATO`(*A*) 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`(*A*,*u*,*v*)
  - scrivi lo pseudocodice della funzione `VERIFICA-POZZO`(*A*,*u*) che restituisce **true** se il nodo identificato dall'indice *u* non ha archi uscenti, **false** altrimenti
  - scrivi lo pseudocodice della funzione `VERIFICA-SORGENTE`(*A*,*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

esercizi su liste di adiacenza 3/3

- dati due grafi *A1* e *A2* rappresentati tramite array di liste di adiacenza doppiamente concatenate
  - scrivi lo pseudocodice della funzione `VERIFICA-UNIONE`(*A1*,*A2*) che verifica che tra ogni possibile coppia di nodi ci sia un arco in *A1* o in *A2* (o in entrambi)
    - puoi supporre che *A1* e *A2* abbiano lo stesso numero di nodi (*A1.length=A2.length*)
  - scrivi lo pseudocodice della funzione `VERIFICA-POZZI-E-SORGENTI`(*A1*,*A2*) che restituisce **true** se tutti i pozzi di *A1* sono sorgenti di *A2* e tutte le sorgenti di *A1* sono pozzi di *A2* e restituisce **false** altrimenti
    - puoi suppore che *A1* e *A2* abbiano lo stesso numero di nodi
    - puoi utilizzare le funzioni `VERIFICA-POZZO`(*A*,*u*) e `VERIFICA-SORGENTE`(*A*,*u*)
- discuti la complessità degli algoritmi che hai proposto

