

Algoritmi e Strutture di Dati

Visita in profondità di un grafo

m.patrignani

240-visita-in-profondita-01 copyright ©2019 maurizio.patrignani@uniroma3.it

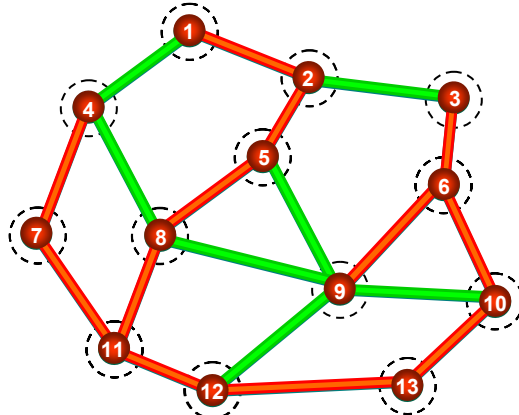
Nota di copyright

- 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

240-visita-in-profondita-01 copyright ©2019 maurizio.patrignani@uniroma3.it

Visita in profondità (Depth-First Search)

- A partire da un nodo v si percorre il grafo fino a che si trovano nodi non visitati
- Quando tutti i vicini sono visitati si torna indietro verso v per verificare che non ci siano nodi adiacenti non visitati



240-visita-in-profondita-01 copyright ©2019 maurizio.patrignani@uniroma3.it

Procedura DFS (liste di adiacenza)

- DFS di un grafo rappresentato con liste di adiacenza

```
DFS(g,v)          /* g è un grafo rappresentato tramite liste di adiacenza */
1.  for i = 0 to g.A.length-1
2.      color[i] = 0          /* zero = non raggiunto */
3.  DFS_visit(g,v,color)     /* comincia una DFS da v */
```

```
DFS_visit(g,i,color)    /* g è un grafo, i è un indice, color un array */
1.  color[i] = 1        /* uno = raggiunto */
2.  x = g.A[i]
3.  while x != NULL
4.      v = x.key        /* v è l'indice di un nodo adiacente ad i */
5.      if color[v] == 0    /* se v non ancora visitato... */
6.          DFS_visit(g,v,color) /* ...continua la visita da v */
7.      x = x.next        /* passa al prossimo adiacente di i */
```

240-visita-in-profondita-01 copyright ©2019 maurizio.patrignani@uniroma3.it

Visite di un grafo non connesso

- Come nel caso della BFS, per eseguire una DFS di un grafo non necessariamente connesso è sufficiente lanciare diverse visite con lo stesso array color
 - per esempio nel caso di grafo rappresentato con liste/matrice di adiacenza il codice potrebbe essere il seguente

```
DFS_non_connesso(g)      /* g è rappresentato tramite liste di adiacenza */
1. for i = 0 to g.A.length-1
2.   color[i] = 0          /* zero = non raggiunto */
3. for i = 0 to g.A.length-1
4.   if color[i] == 0      /* se i non ancora visitato... */
5.     DFS_visit(g,i,color) /* ...comincia una DFS da qui */
```

240-visita-in-profondita-01 copyright ©2019 maurizio.patrigiani@uniroma3.it

Complessità della visita DFS

- In una visita in profondità
 - su ogni nodo viene lanciata **DFS_visit** una sola volta
 - ogni arco (adiacenza) è considerata sia dal nodo di partenza che dal nodo di arrivo
- Dunque la complessità è $\Theta(n+m)$

240-visita-in-profondita-01 copyright ©2019 maurizio.patrigiani@uniroma3.it

Esercizi sulle visite di grafi

1. Scrivi lo pseudocodice della funzione ALBERO-RICOPRENTE(g, u)

- input: un grafo non orientato g rappresentato da un array $g.A$ di liste di adiacenza e un nodo u
- output: un array $parent$ dove $parent[v]$ è l'indice del nodo da cui è stato raggiunto v in una DFS a partire dal nodo u
 - il $parent[u]$ è convenzionalmente posto uguale a -1

2. Variante:

- produci in output un albero t (connesso e di grado arbitrario) con le seguenti caratteristiche
 - i nodi di t hanno gli stessi indici dei nodi del grafo
 - c'è un arco in t diretto da un nodo x ad un nodo y solo se nella DFS il nodo y è stato raggiunto da x

240-visita-in-profondita-01 copyright ©2019 maurizio.patrignani@uniroma3.it

Esercizi sulle visite di grafi

3. Scrivi lo pseudocodice della procedura DFS(g, v) nel caso in cui il grafo sia rappresentato da una matrice di adiacenza $g.A$

4. Scrivi lo pseudocodice della procedura DFS(g, v) che restituisce in output un array $ordine$ dove $ordine[v]$ è il numero d'ordine con cui il nodo v è stato visitato nella visita in profondità

240-visita-in-profondita-01 copyright ©2019 maurizio.patrignani@uniroma3.it

Esercizi sulle visite di grafi

5. Scrivi la funzione in linguaggio C
`int COMPONENTI_UGUALI (grafo* g)`
che prende in input un grafo g rappresentato tramite oggetti e riferimenti e verifica se tutte le componenti connesse hanno lo stesso numero di nodi
6. Scrivi la funzione in linguaggio C
`int COMPONENTI_BILANCIATE (grafo* g)`
che prende in input un grafo g rappresentato tramite oggetti e riferimenti e verifica se tutte le componenti connesse hanno tanti nodi quanti archi