Grafi (II parte)

Progettazione di Algoritmi a.a. 2023-24

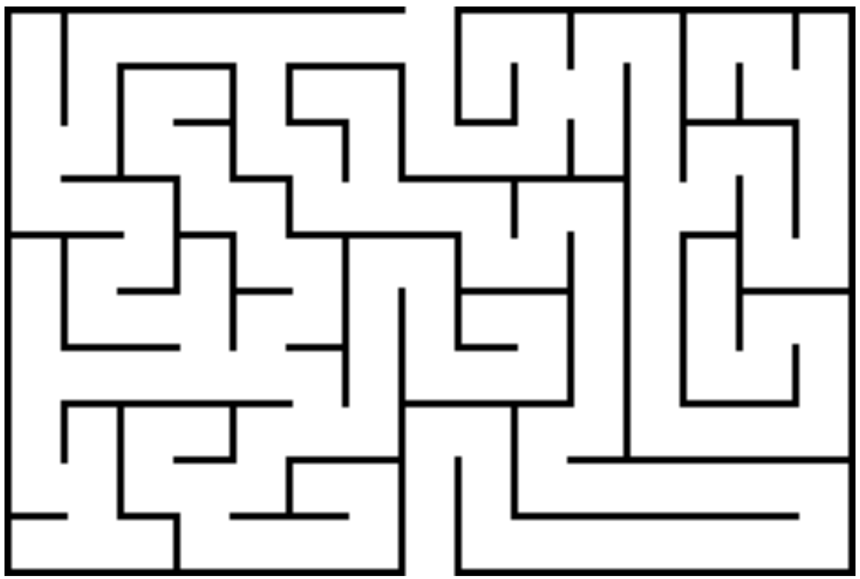
Matricole congrue a 1

Docente: Annalisa De Bonis

Depth first search (visita in profondità)

- La visita in profondità riproduce il comportamento di una persona che esplora un labirinto di camere interconnesse

  - La persona parte dalla prima camera (nodo s) e si sposta in una delle camere accessibili dalla prima (nodo adiacente ad s), di lì si sposta in una delle camere accessibili dalla seconda camera visitata e così via fino a quando raggiunge una camera da cui non è possibile accedere a nessuna altra camera non ancora visitata. A questo punto torna nella camera precedentemente visitata e di lì prova a raggiungere nuove camere.



Depth first search (visita in profondità)

- La visita DFS parte dalla sorgente s e si spinge in profondità fino a che non è più possibile raggiungere nuovi nodi.

  - La visita parte da s, segue uno degli archi uscenti da s ed esplora il vertice v a cui porta l'arco.
  - Una volta in v, se c'è un arco uscente da v che porta in un vertice w non ancora esplorato allora l'algoritmo esplora w
  - Uno volta in w segue uno degli archi uscenti da w e così via fino a che non arriva in un nodo del quale sono già stati esplorati tutti i vicini.
  - A questo punto l'algoritmo fa backtrack (torna indietro) fino a che torna in un vertice a partire dal quale può visitare un vertice non ancora esplorato in precedenza.

Depth first search: pseudocodice

```
DFS(u):
  Mark u as "Explored" and add u to R
  For each edge (u,v) incident to u
    If v is not marked "Explored" then
      Recursively invoke DFS(v)
    Endif
  Endfor
```

R = insieme dei vertici raggiunti

Analisi: (assumendo G rappresentato con liste di adiacenza)

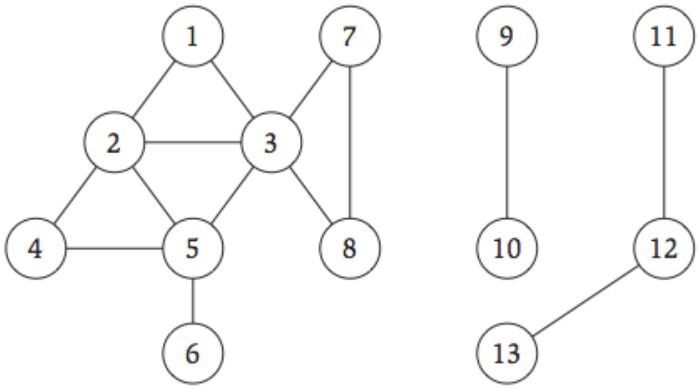
- Se ignoriamo il tempo delle chiamate ricorsive al suo interno, ciascuna visita ricorsiva richiede tempo O(1+deg(u)): O(1) per marcare u e aggiungerlo ad R e O(deg(u)) per eseguire il for.
- Se inizialmente invochiamo DFS su un nodo s, allora DFS viene invocata ricorsivamente su tutti i nodi raggiungibili a partire da s. Il costo totale è quindi al più

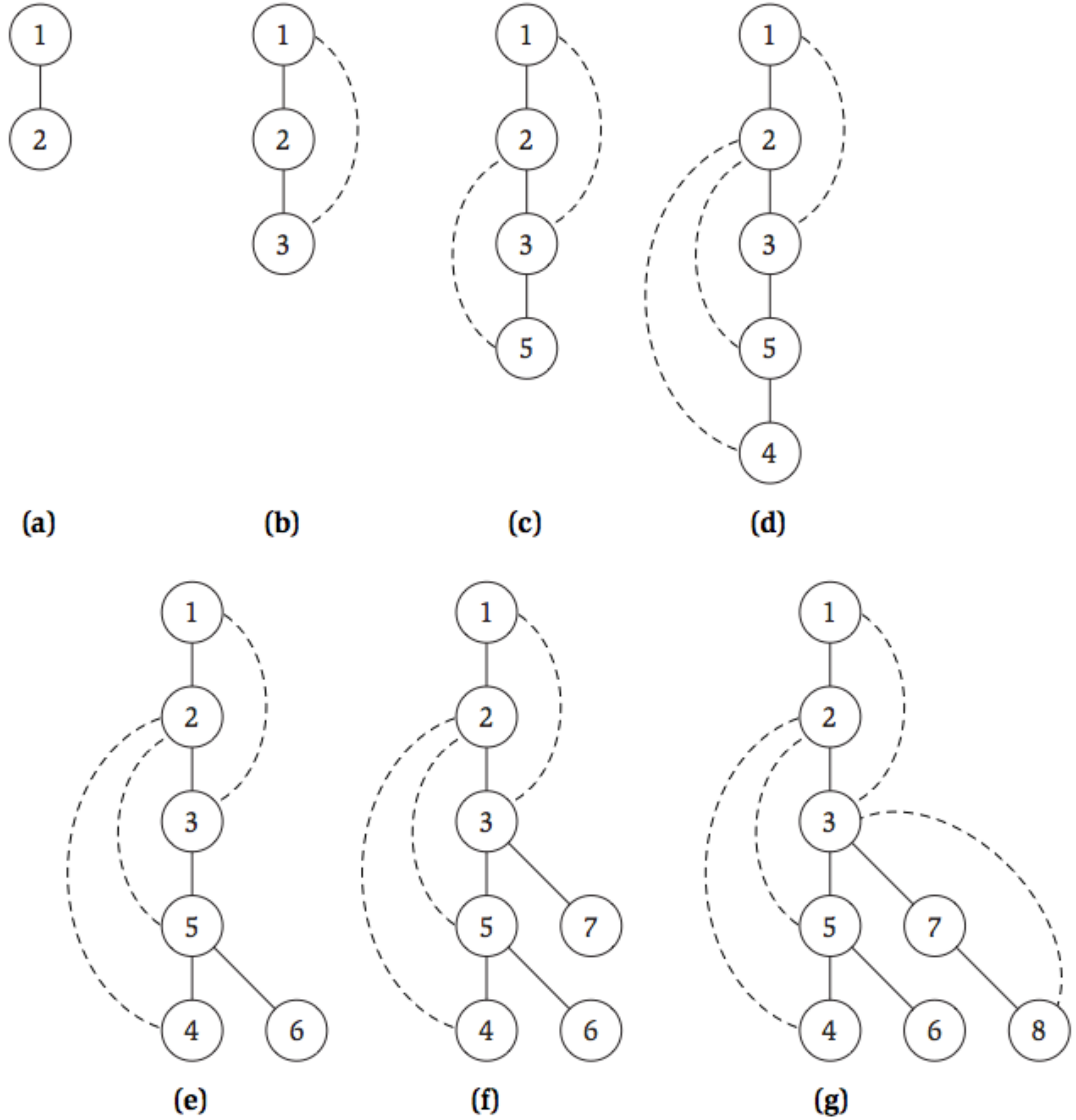
$$\sum_{u\in V} O(1+deg(u)) = O(\sum_{u\in V} 1 + \sum_{u\in V} deg(u))$$
$$= O(n+m)$$

Esempio di esecuzione di DFS

I nodi nelle liste di adiacenza vengono esaminati in ordine crescente

G





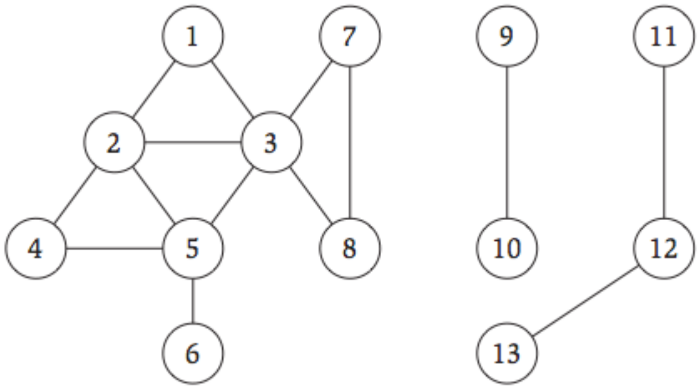
Depth First Search Tree (Albero DFS)

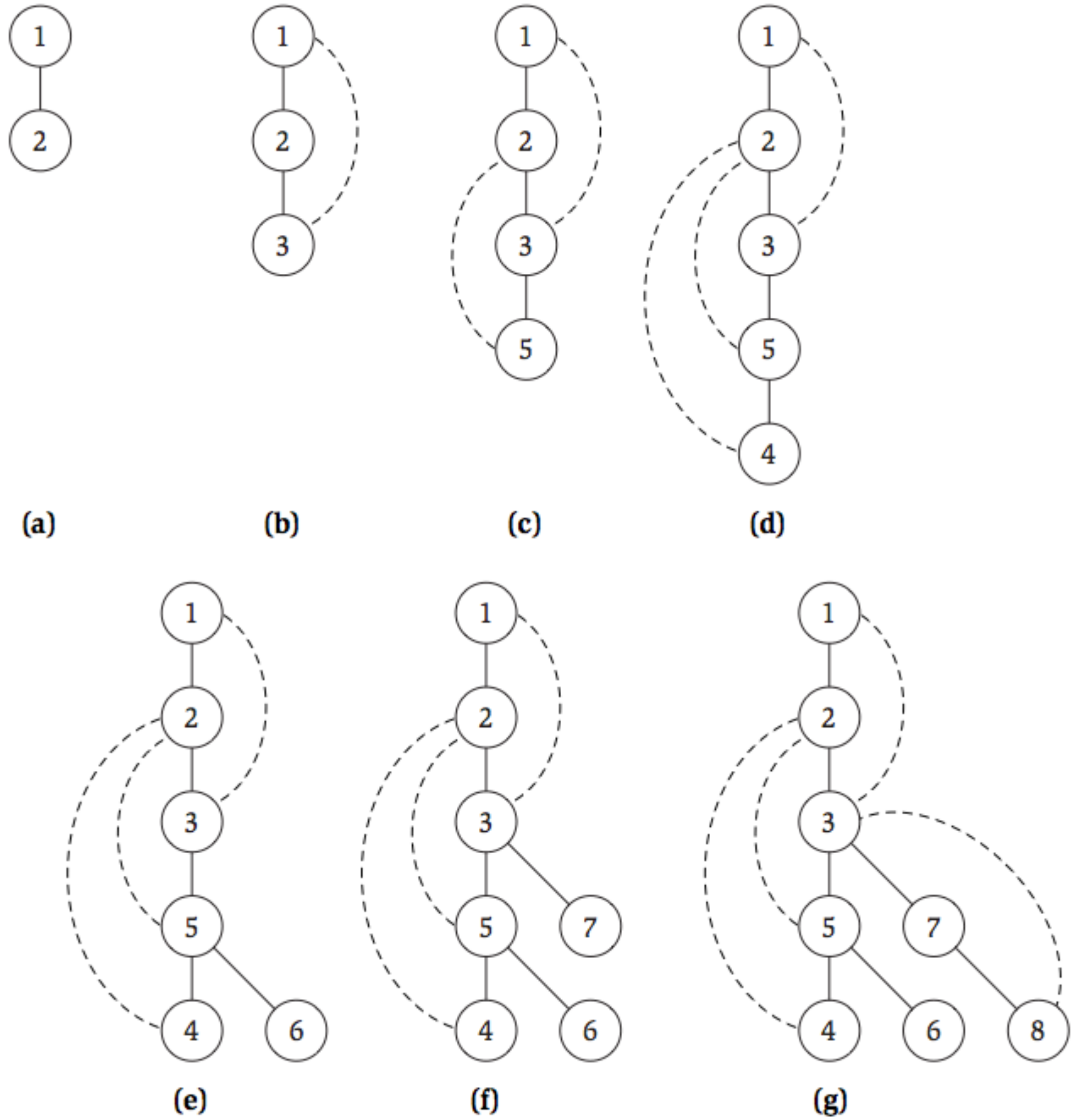
- Proprietà. L'algoritmo DFS produce un albero che ha come radice la sorgente s e come nodi tutti i nodi del grafo raggiungibili da s.

- L'albero si ottiene in questo modo:
  - Consideriamo il momento in cui viene invocata DFS(v)
    - Ciò avviene durante l'esecuzione di DFS(u) per un certo nodo u. In particolare durante l'esame dell'arco (u,v) nella chiamata DFS(u).
    - In questo momento, aggiungiamo l'arco (u,v) e il nodo v all'albero

Esempio di albero DFS

G





Gli archi non tratteggiati fanno parte del DFS tree.

Albero DFS

- Proprietà 1. Per una data chiamata ricorsiva DFS(u), tutti i nodi che vengono etichettati come "Esplorati" tra l'inizio e la fine della chiamata DFS(u), sono discendenti di u nell'albero DFS.

Dim. Proprietà 1.

- Sia x un nodo esplorato tra l'inizio e la fine della chiamata DFS(u)
- La dimostrazione è per induzione sul numero m di chiamate ricorsive iniziate dopo l'inizio di DFS(u) e non ancora terminate **quando viene invocata** DFS(x) (esclusa DFS(u) e inclusa DFS(x)).
- **Base. m=1** → Una sola chiamata cominciata dopo l'inizio di DFS(u) e che si deve ancora concludere nel momento in cui esploriamo x → questa chiamata è proprio DFS(x) → DFS(x) è invocata nel foreach di DFS(u) e in questo caso x diventa figlio di u →la proprieta` è soddisfatta (es. u=5, x=6 in slide precedente)
- **Passo induttivo.** Supponiamo vera la proprieta` fino a m-1>=1 e dimostriamo che è vera per m. Siccome m>=2 → ci deve essere almeno una chiamata a DFS che non è ancora terminata prima che venga invocata DFS(x) → DFS(x) non è invocata nel foreach di DFS(u). Infatti, se DFS(x) venisse invocata nel foreach di DFS(u) allora in quel momento sarebbero gia` terminate tutte le chiamate sui nodi adiacenti ad u esaminati prima di x e tutte le chiamate da esse innescate e sarebbe m=1.
- Sia z il nodo adiacente ad x per cui si ha che DFS(z) invoca DFS(x)**. Questo vuol dire dopo l'inizio di DFS(u) fino al momento in cui viene invocata DFS(z) sono state effettuate al piu` m-1 chiamate ricorsive.** Possiamo quindi applicare l'ipotesi induttiva e dire che z è discendente di u. Siccome x è figlio di z allora x è anch'esso discendente di u.

Albero DFS

Questa proprieta` vale solo se il grafo è non direzionato.

- Proprietà 2. Sia T un albero DFS e siano x e y due nodi di T collegati dall'arco (x,y) in G. Si ha che x e y sono l'uno antenato dell'altro in T.

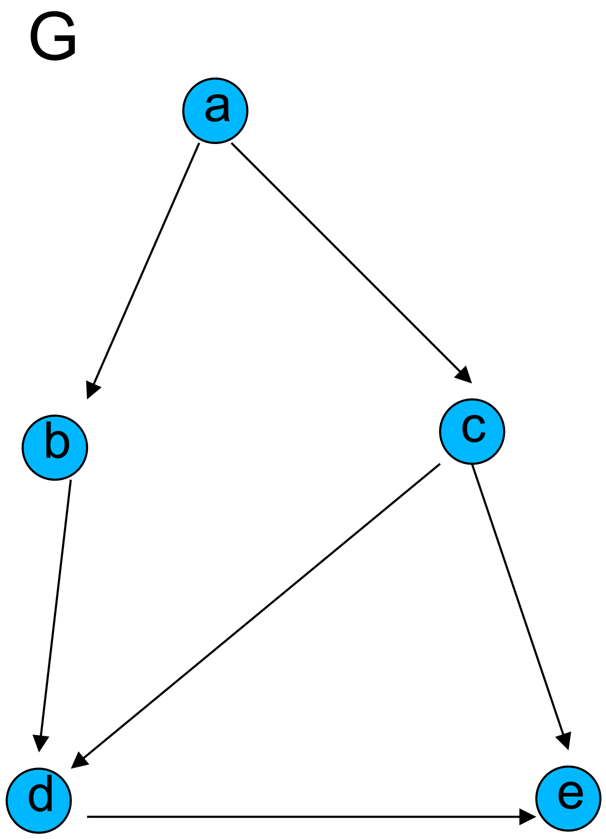
Dim. Proprietà 2

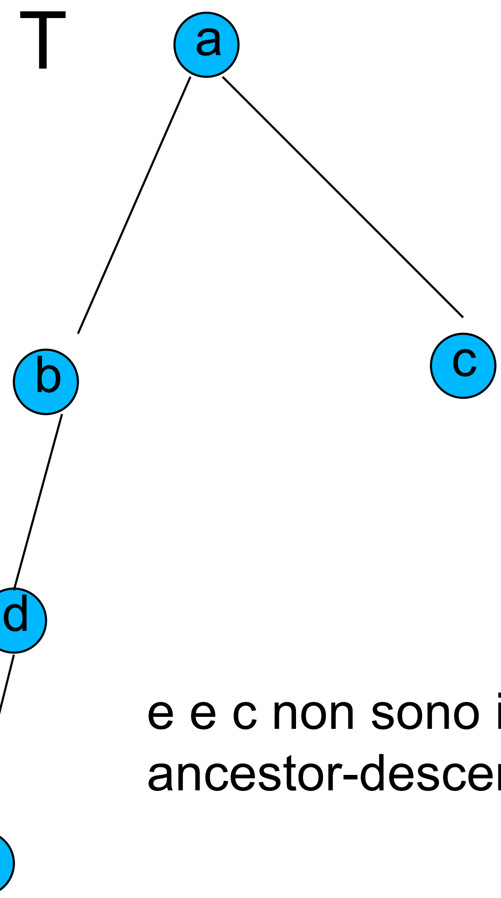
- **Caso (x,y) e` in T.** In questo caso la proprietà e` ovviamente soddisfatta.
- **Caso (x,y) non e` in T.** Supponiamo senza perdere di generalità che DFS(x) venga invocata prima di DFS(y). Ciò vuol dire che quando viene invocata DFS(x), y non è ancora etichettato come "Esplorato".
- La chiamata DFS(x) esamina l'arco (x,y) e per ipotesi non inserisce (x,y) in T. Ciò si verifica solo se y è già stato etichettato come "Esplorato". Siccome y non era etichettato come "Esplorato" all'inizio di DFS(x) vuol dire è stato esplorato tra l'inizio e la fine della chiamata DFS(x). La proprietà 1 implica che y è discendente di x.

Albero DFS

Facciamo vedere che la proprieta` 2 non vale in generale per i grafi direzionati.

Usiamo un controesempio:





e e c non sono in relazione ancestor-descendant