

Unità didattica: algoritmi di visita di un grafo

[3-T]

Titolo: Algoritmi di visita di una struttura reticolare

Argomenti trattati:

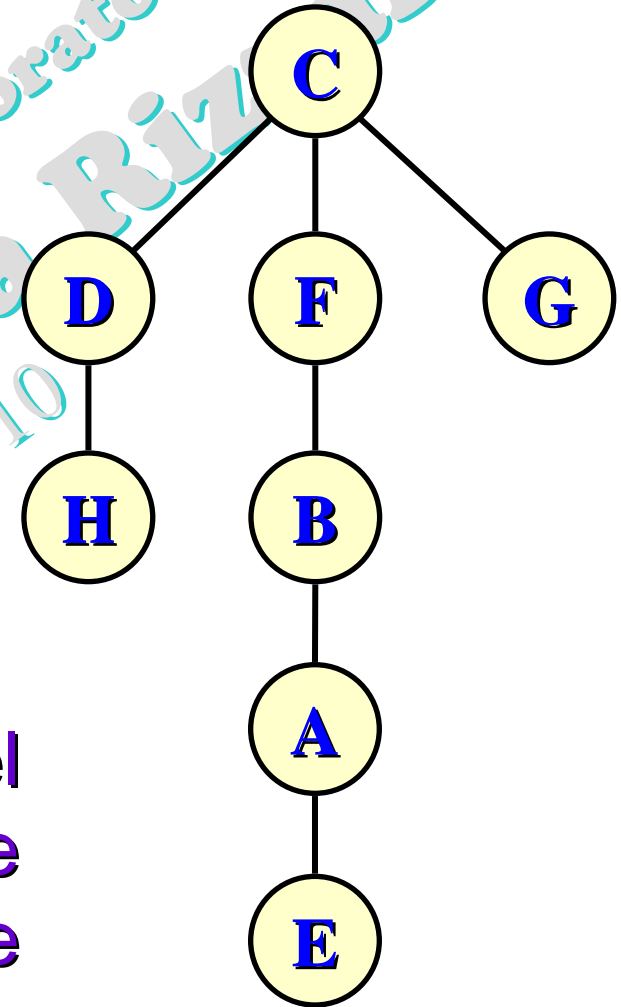
- ✓ Visita in ordine anticipato di un albero
- ✓ Visita di un grafo: algoritmo di visita in profondità (Depth First Search)

Prerequisiti richiesti: pila, visita di un albero, rappresentazione di un grafo

Algoritmo di visita in ordine anticipato per gli alberi generici

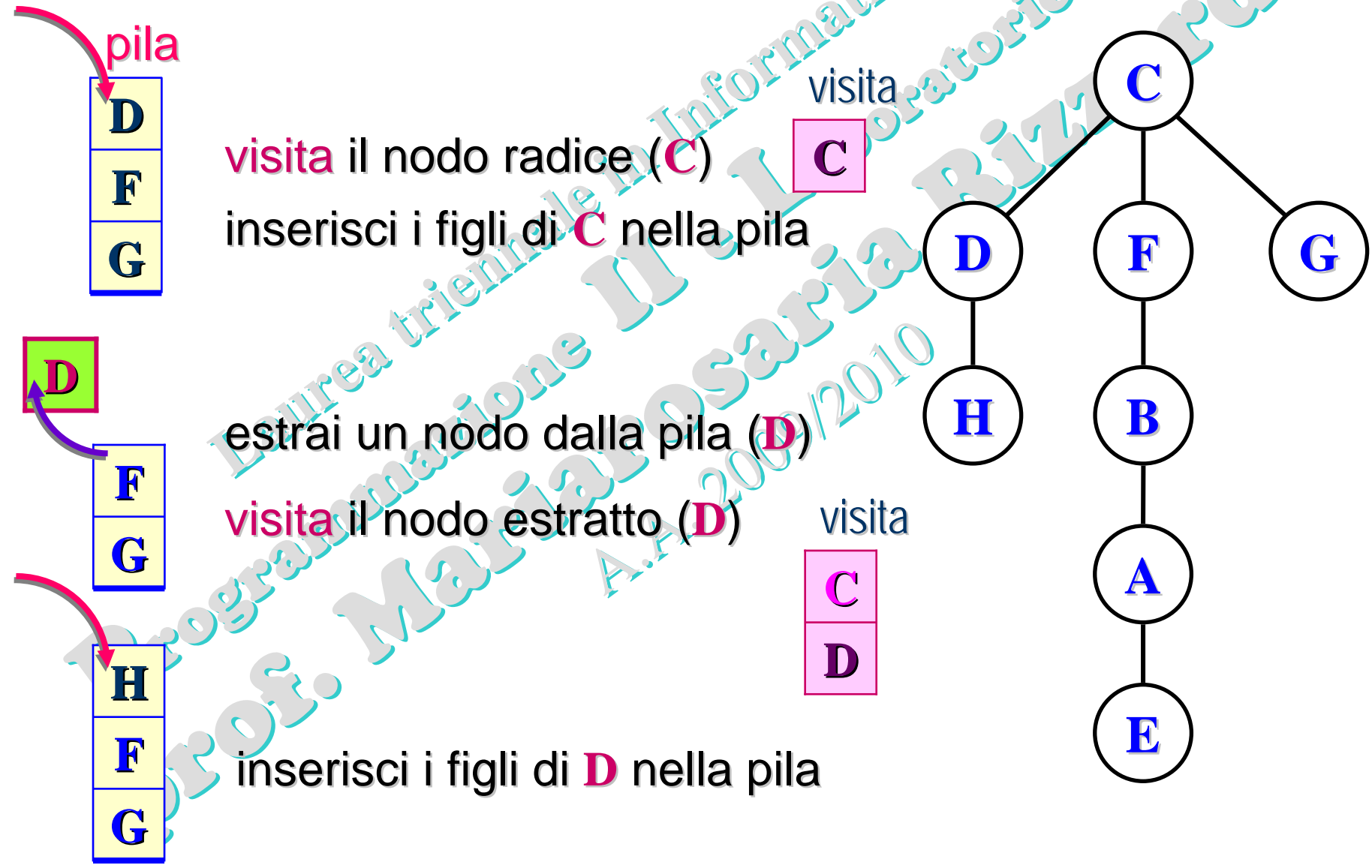
Usando una **pila** si possono visitare i nodi di un **albero qualsiasi** con un algoritmo simile alla visita in ordine anticipato di un albero binario.

L'ordine anticipato, qui, va inteso nel senso che si visita un nodo e soltanto dopo si passa a visitare un suo figlio.



Idea dell'algoritmo di visita in ordine anticipato per gli alberi generici

[1]



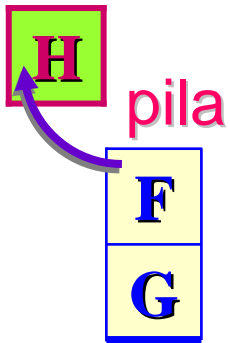
Idea dell'algoritmo di visita in ordine anticipato per gli alberi generici

[2]

P2_10_03.4

Strutture dinamiche reticolari

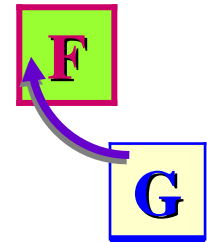
(prof. M. Rizzardi)



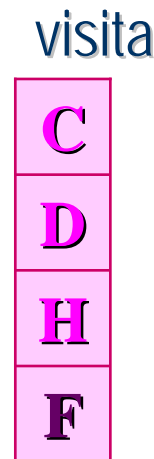
estrai un nodo dalla pila (**H**)
visita il nodo estratto (**H**)



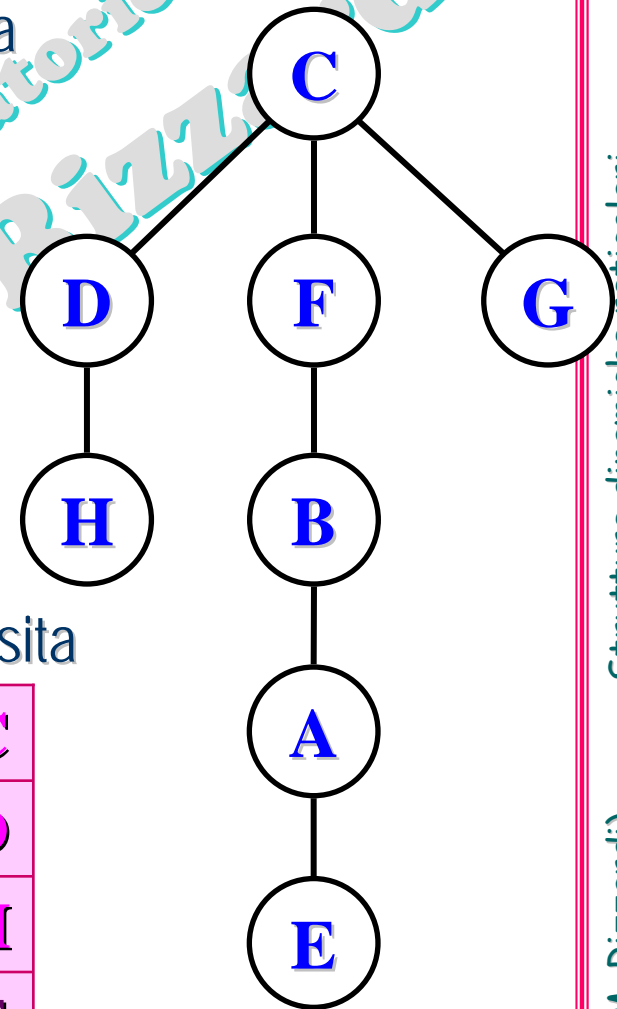
inserisci i figli di **H** nella pila
(non ce ne sono)



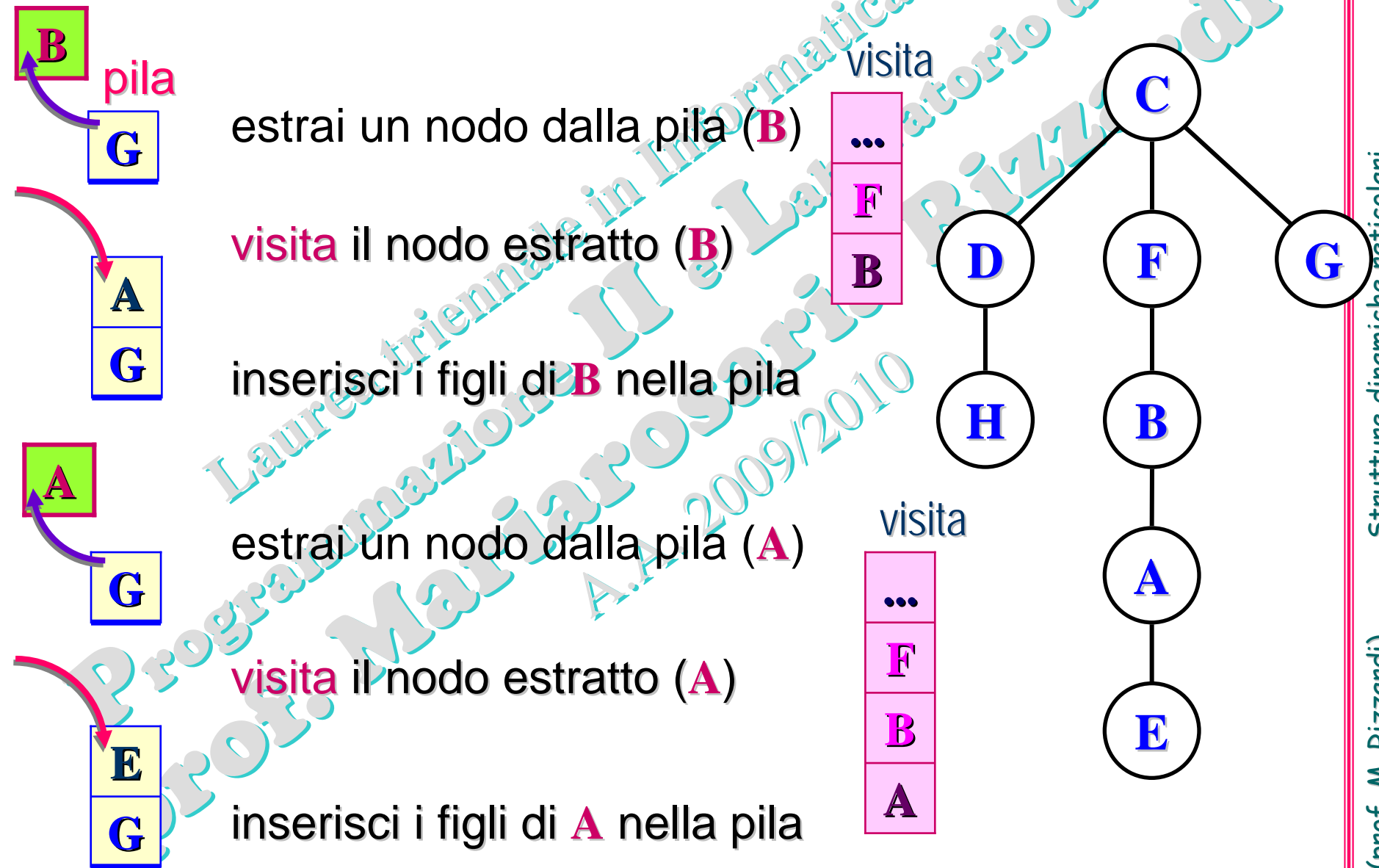
estrai un nodo dalla pila (**F**)
visita il nodo estratto (**F**)



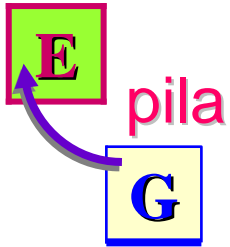
inserisci i figli di **F** nella pila



Idea dell'algoritmo di visita in ordine anticipato per gli alberi generici [3]



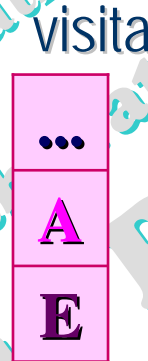
Idea dell'algoritmo di visita in ordine anticipato per gli alberi generici [4]



estrai un nodo dalla pila (**E**)

visita il nodo estratto (**E**)

inserisci i figli di **E** nella pila
(non ce ne sono)



estrai un nodo dalla pila (**G**)

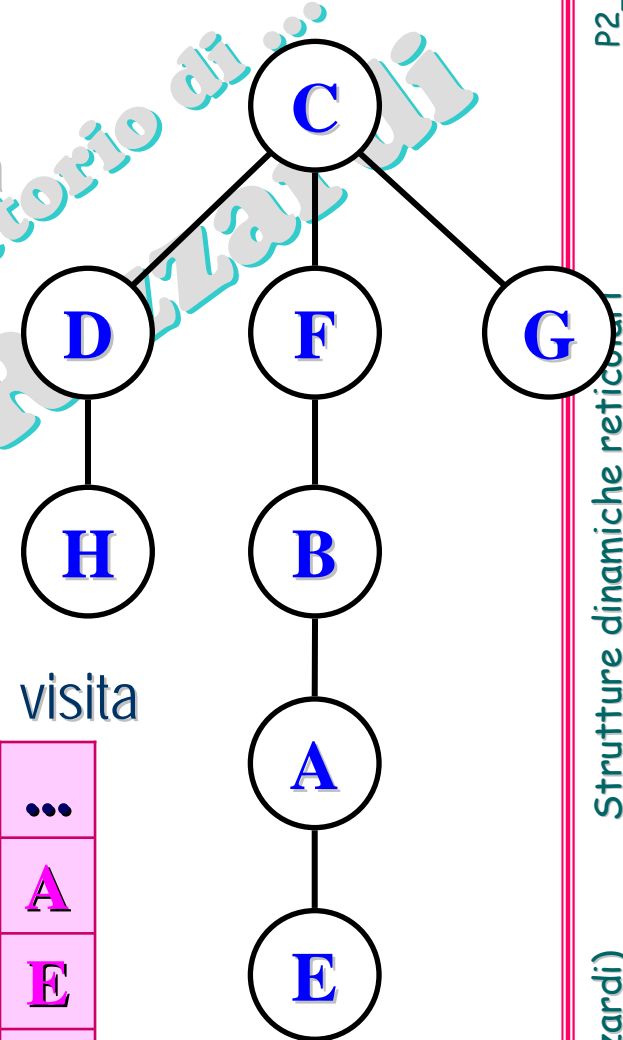
visita il nodo estratto (**G**)

inserisci i figli di **G** nella pila
(non ce ne sono)

visita



estrai un nodo dalla pila, ma la **pila** è **vuota**:
quindi ... **fine algoritmo!**



???



L'algoritmo **DFS** (**Depth First Search**) di **Visita in Profondità di un grafo** viene descritto molto semplicemente in forma ricorsiva:

visita un nodo V
(ricorsivamente) visita ogni nodo U adiacente a V
e non ancora visitato

Se il **grafo** è **connesso**, saranno raggiunti tutti i nodi, altrimenti viene visitato il sottografo connesso cui V appartiene

Nell'algoritmo **DFS** (Depth First Search) ricorsivo la **pila** (**stack**), usata per ricordare i vertici non ancora visitati, è gestita implicitamente dalla ricorsione.

```
DFS (Graph G, Vertice N)
```

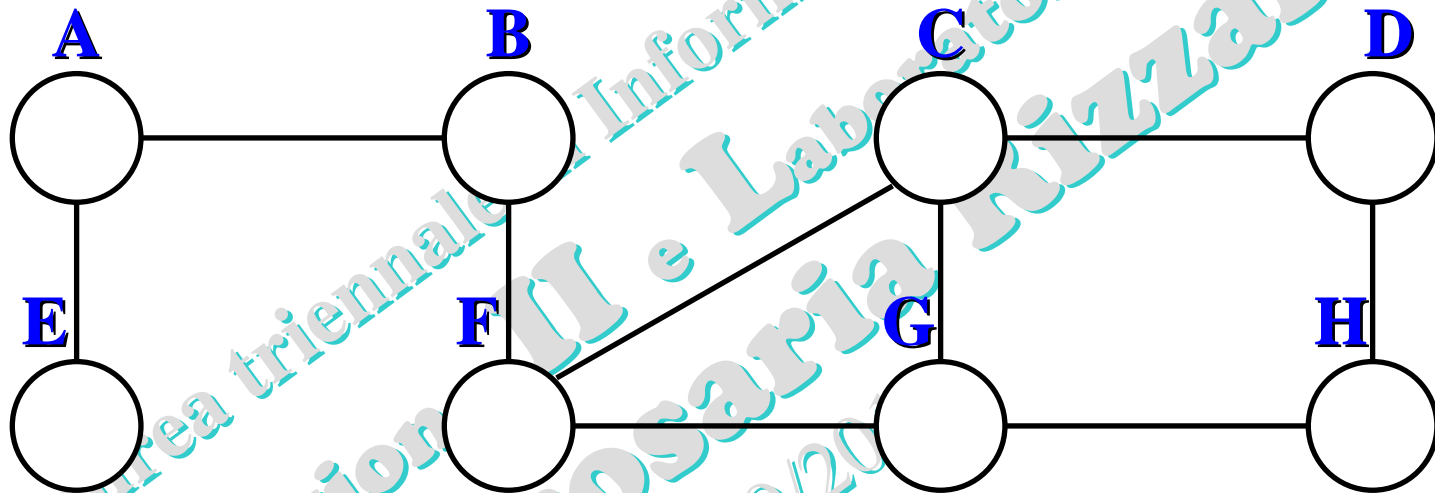
```
{  
    tutti i vertici di G sono "non visitati";  
    DFS_visit(G, N);  
}
```

```
DFS_visit (Graph G, Vertice V)
```

```
{ se V è "non visitato", allora  
    { visita V;  
      V è "visitato";  
      per "tutti i nodi U adiacenti a V"  
        DFS_visit (G, U);  
    }  
}
```


Algoritmi di visita di un grafo: visita in profondità (DFS - Depth First Search)

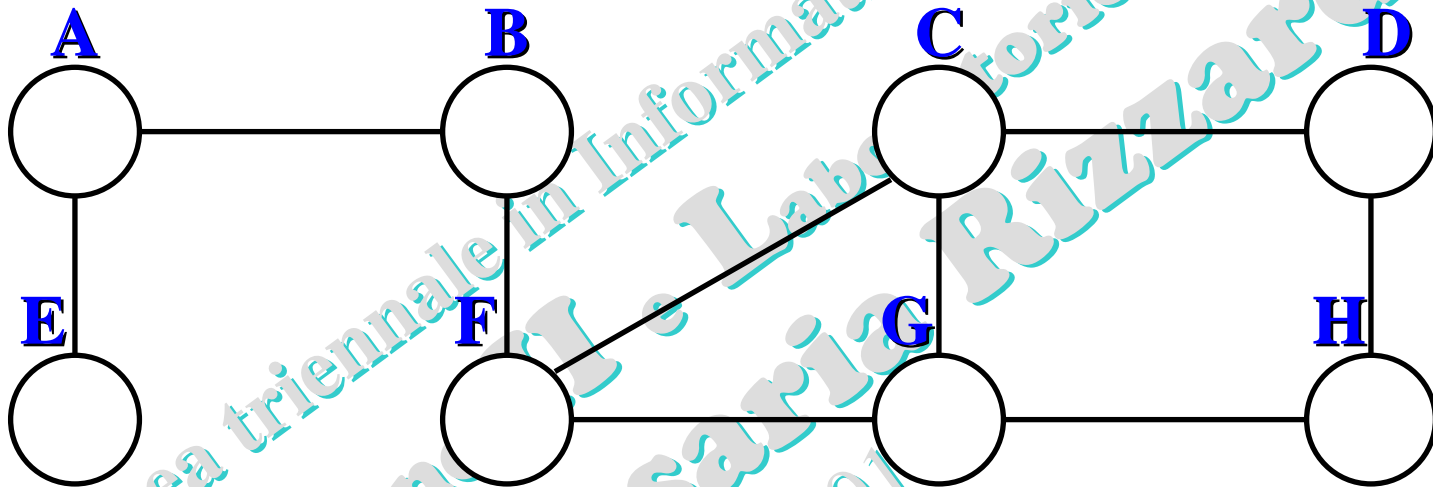
ESEMPIO



L'algoritmo iterativo della **visita in profondità** di un grafo viene realizzato utilizzando esplicitamente una **pila**: ogni volta che si visita un nuovo nodo, si inseriscono i suoi nodi adiacenti nella pila. Ad ogni passo si estrae un nodo dalla pila e, se non è stato visitato, lo si visita.

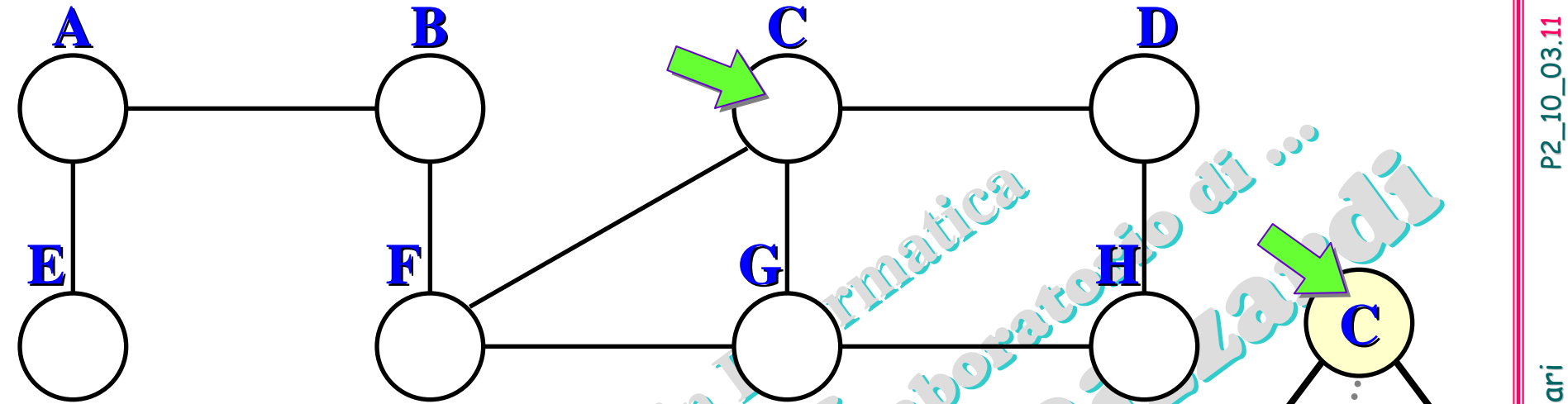
Algoritmi di visita di un grafo: visita in profondità

ESEMPIO



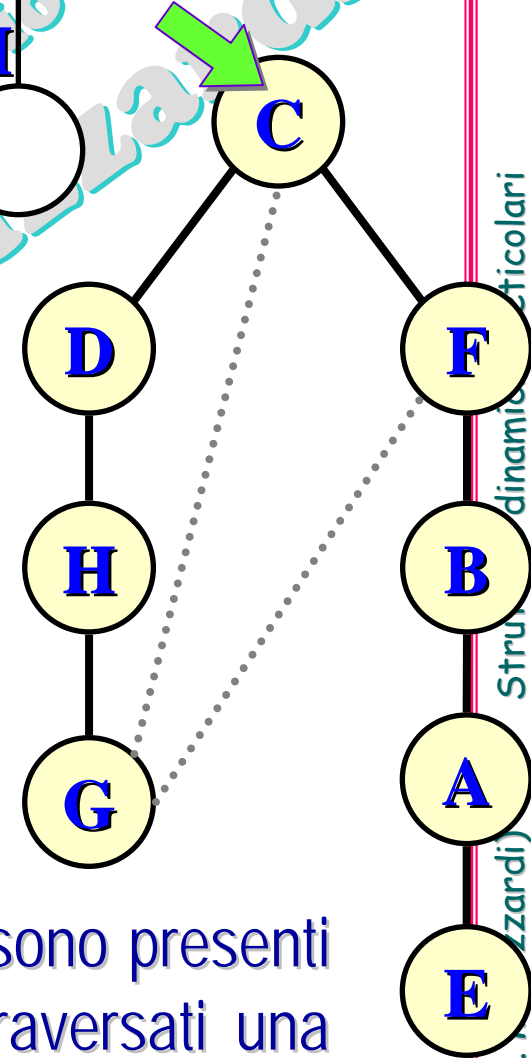
Per descrivere il funzionamento dell'algoritmo si utilizza la **colorazione** dei vertici (cioè un **flag**) con le regole:

- Inizialmente tutti i vertici sono **bianchi** ("non visitati").
- Un vertice è colorato in **grigio** quando viene "visitato".



L'algoritmo DFS coincide con la visita in ordine anticipato di un albero qualsiasi la cui radice è il nodo da cui si parte.

L'algoritmo DFS trasforma il grafo in un albero (albero di ricerca depth-first).



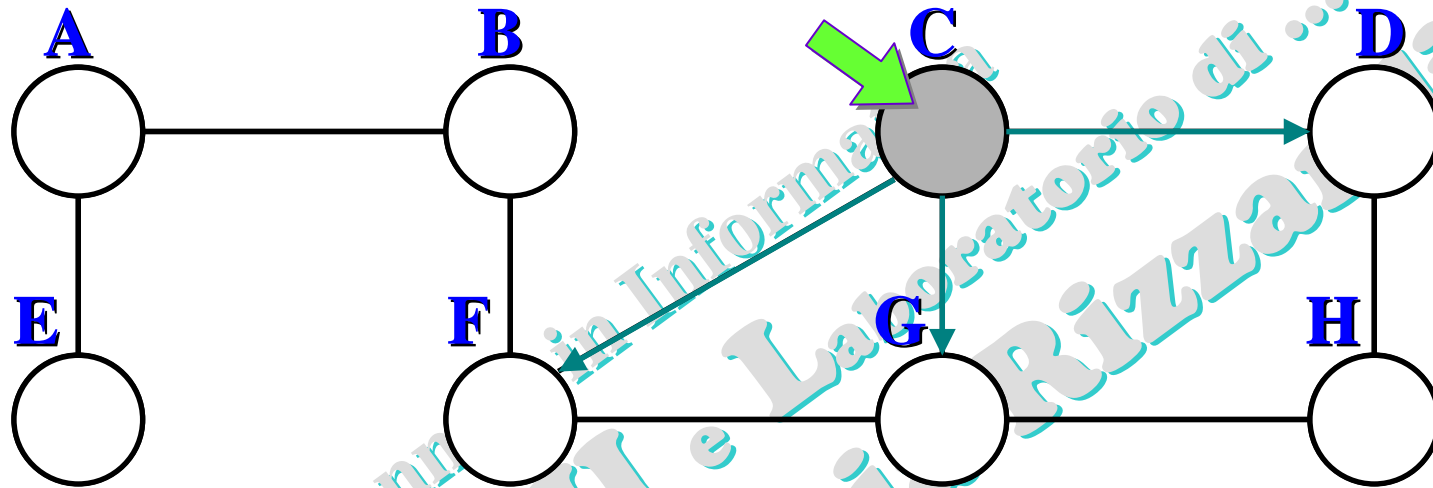
Nell'albero di ricerca depth-first corrispondente non sono presenti tutti gli archi del grafo ma quelli presenti sono attraversati una volta. Nella DFS tutti gli archi del grafo sono attraversati due volte.

Algoritmo Depth First Search iterativo

```
procedure DFS_visit_iterat (var G: Grafo, var V: Vertice)
{
  visita V;
  inserisci nella pila i nodi adiacenti a V;
  while pila "non vuota"
    estrai nodo U dalla pila;
    if nodo U "non visitato"
      visita U;
    endif
    inserisci nella pila i nodi adiacenti a U;
  endwhile
}
```

Visita in profondità - DFS

idea algoritmo [1]



Inizialmente:

- Viene assegnato il colore **bianco** (= "non visitato") a tutti i vertici.
- Se ad es. si comincia la visita da **C** come vertice **sorgente**: **C** viene colorato in grigio (= "visitato").
- Sono inseriti nella **pila** i **nodi adiacenti a C** (ciò equivale a percorrere gli archi che partono da **C**).



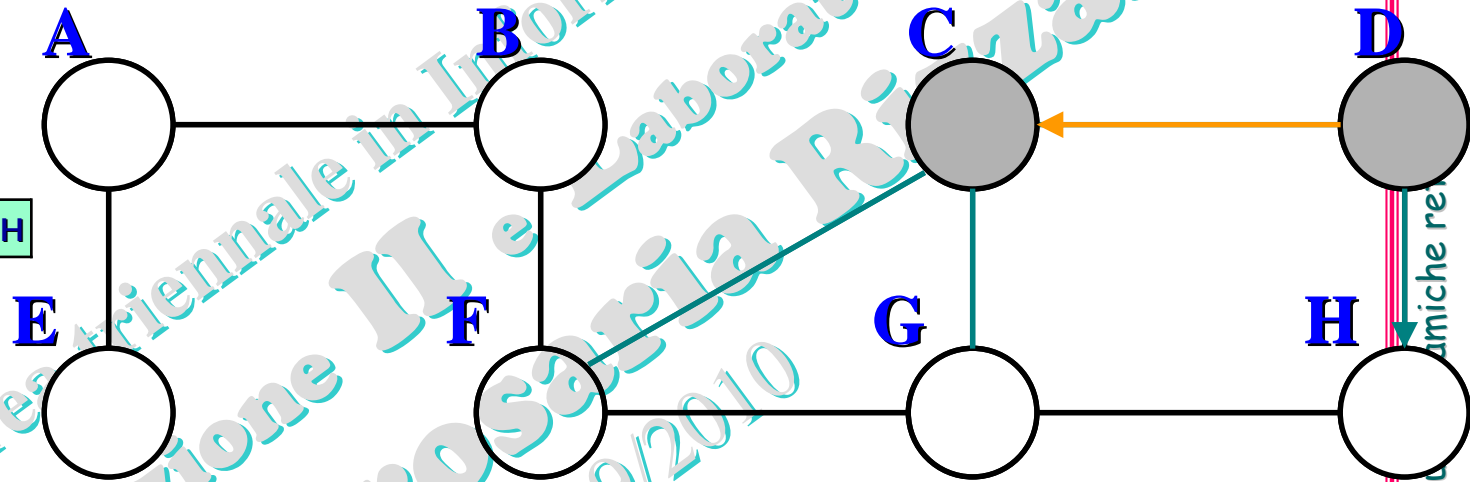
	A	B	C	D	E	F	G	H
A	0	1	0	0	1	0	0	0
B	1	0	0	0	0	1	0	0
C	0	0	0	1	0	1	1	0
D	0	0	1	0	0	0	0	1
E	1	0	0	0	0	0	0	0
F	0	1	1	0	0	0	1	0
G	0	0	1	0	0	1	0	0
H	0	0	0	1	0	0	1	0
	A	B	C	D	E	F	G	H

P2_10_03.14

Es

"bia"

- Estrae nodo dalla pila (**D**), essendo "bianco" lo visita e lo colora in grigio.



- Inserisce nella **pila** i **vertici** **adiacenti** a **D**:
nodi **C**, **H** (ciò equivale a percorrere gli archi che
partono da **D** – l'arco arancio è stato percorso 2 volte).



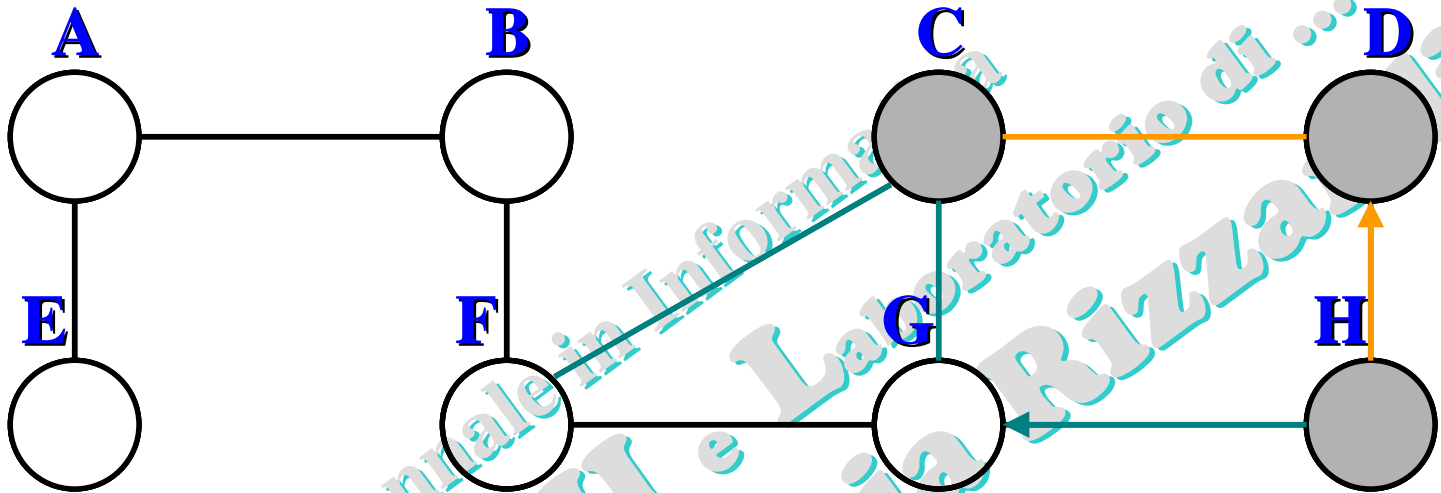
- Estrae nodo dalla pila (C), essendo "grigio" non lo visita.



A	B	C	D	E	F	G	H
	1	0	0	1	0	0	0
		0	0	0	1	0	0
			1	0	1	1	0
				0	0	0	1
					0	0	0
						1	0
							1

Visita in profondità - DFS

idea algoritmo [3]



- Estrae nodo dalla pila (**H**), essendo "bianco" lo visita e lo colora in grigio.

H

F G pila

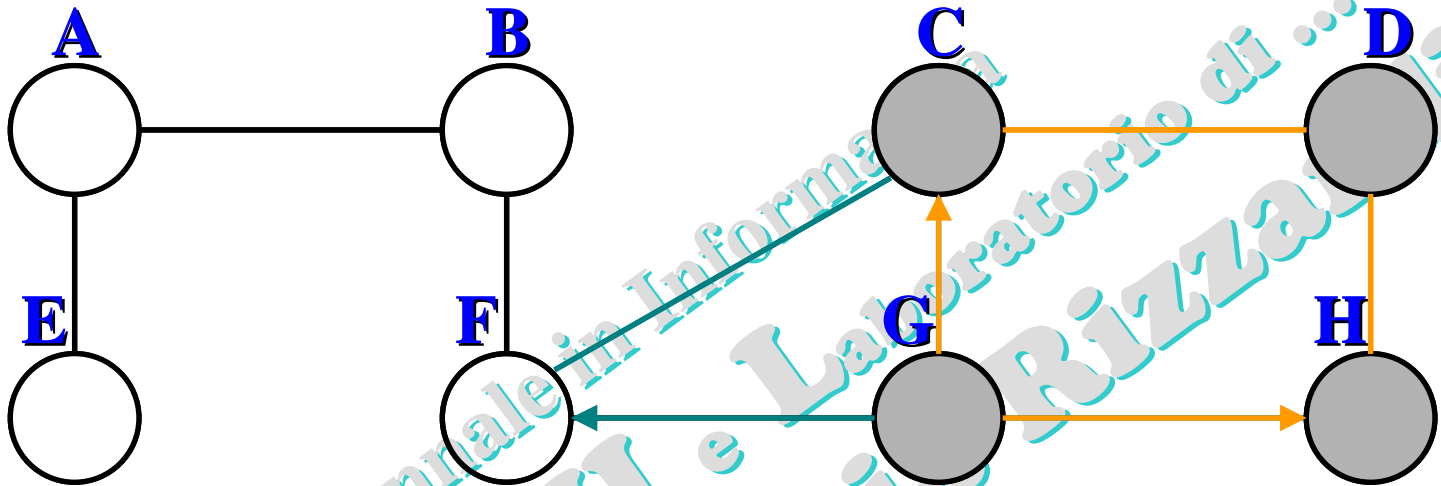
- Inserisce nella **pila** i vertici adiacenti ad **H**: nodi **D**, **G** (ciò equivale a percorrere gli archi che partono da **H** – l'arco arancio è stato percorso 2 volte).

D G F G

A	B	C	D	E	F	G	H	
1	0	0	1	0	0	0	0	A
0	0	0	1	0	0	0	0	B
1	0	1	1	0	0	0	0	C
0	0	0	1	0	0	0	0	D
0	0	0	0	0	0	0	0	E
1	0	0	0	0	0	0	0	F
1	0	0	0	0	0	0	0	G
0	0	0	0	0	0	0	0	H

Visita in profondità - DFS

idea algoritmo [4]



- Estrae nodo dalla pila (**D**) essendo "grigio" **non** lo visita.
- Estrae nodo dalla pila (**G**), essendo "bianco" lo visita e lo colora in grigio.
- Inserisce nella **pila** i vertici adiacenti ad **G**: nodi **C, F, H** (ciò equivale a percorrere gli archi che partono da **G** – l'arco arancio è stato percorso 2 volte).

D

pila

G F G

G

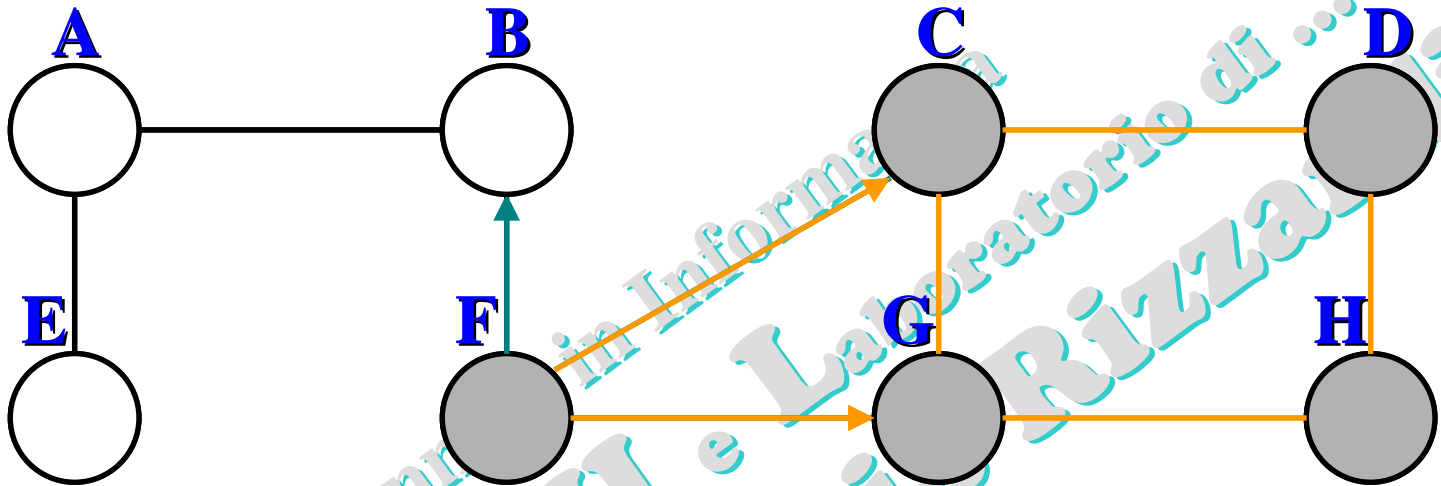
F G

C F H F G

A	B	C	D	E	F	G	H	
1	0	0	1	0	0	0	0	A
0	0	0	1	0	0	0	0	B
	1	0	1	1	0	0	0	C
	0	0	0	1	0	0	0	D
	0	0	0	0	0	0	0	E
		1	0	0	0	0	0	F
			1	0	0	0	0	G
				1	0	0	0	H

Visita in profondità - DFS

idea algoritmo [5]



- Estrae nodo dalla pila (**C**) essendo "grigio" **non** lo visita.
- Estrae nodo dalla pila (**F**), essendo "bianco" lo visita e lo colora in grigio.
- Inserisce nella **pila** i vertici adiacenti ad **F**: nodi **B**, **C**, **G** (ciò equivale a percorrere gli archi che partono da **F** – l'arco arancio è stato percorso 2 volte).

C

pila

F H F G

F

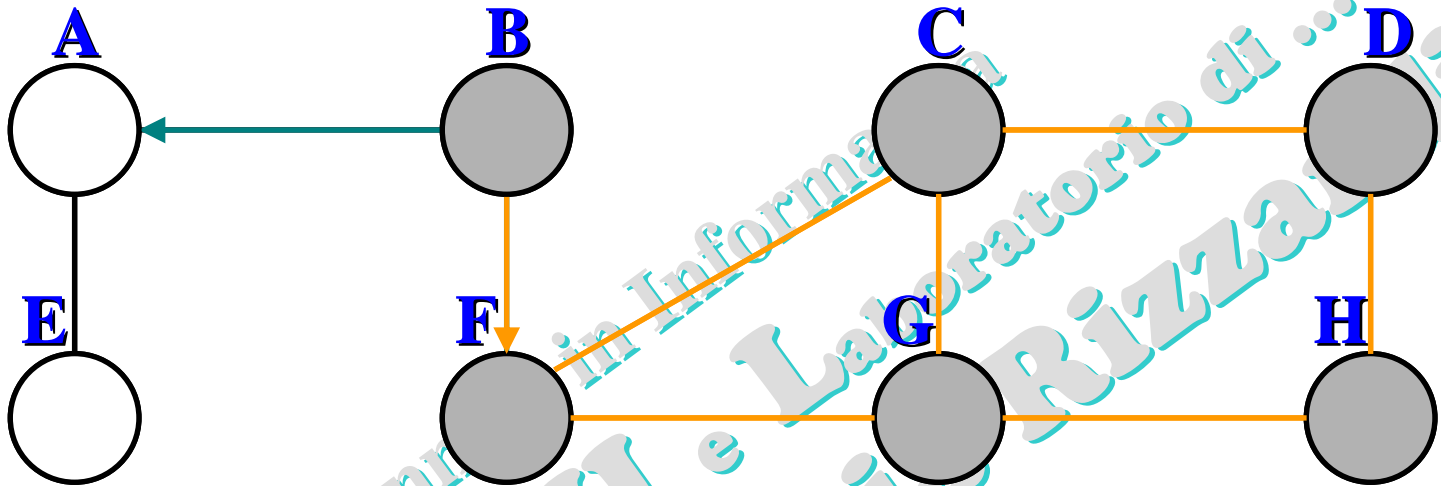
H F G

B C G H F G

A	B	C	D	E	F	G	H	
1	0	0	1	0	0	0	0	A
0	0	0	0	1	0	0	0	B
0	0	0	0	0	1	0	0	C
0	0	0	0	0	0	1	0	D
0	0	0	0	0	0	0	1	E
0	0	0	0	0	0	0	0	F
0	0	0	0	0	0	0	0	G
0	0	0	0	0	0	0	0	H

Visita in profondità - DFS

idea algoritmo [6]



- Estrae nodo dalla pila (**B**) essendo "bianco" lo visita e lo colora in grigio.
- Inserisce nella **pila** i vertici adiacenti a **B**: nodi **A**, **F** (ciò equivale a percorrere gli archi che partono da **B** – l'arco arancio è stato percorso 2 volte).

B

pila

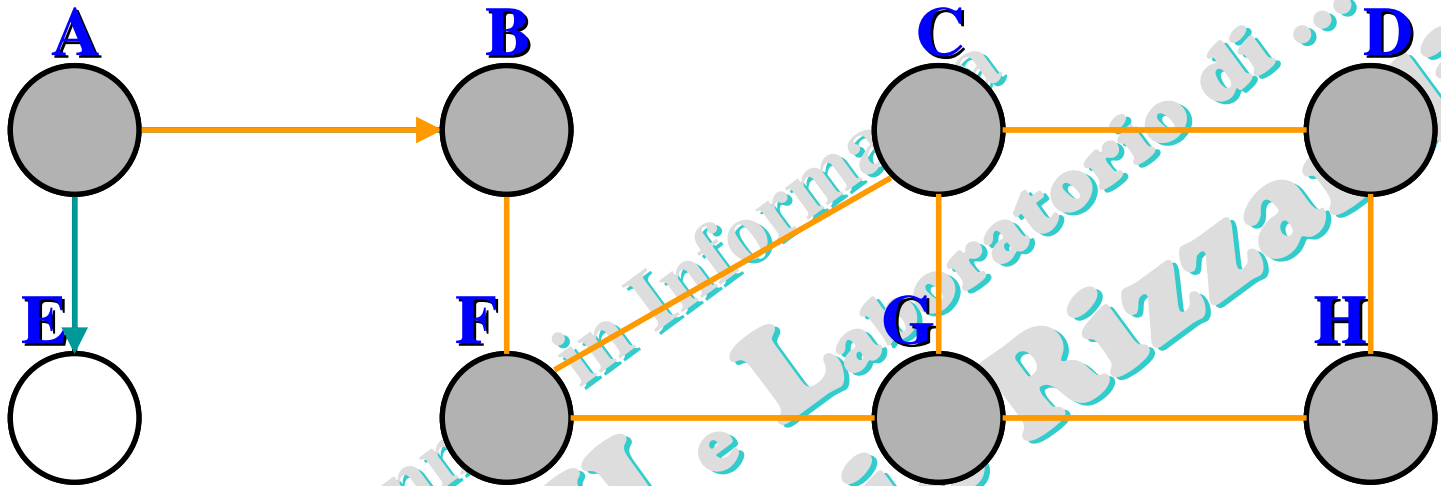
C G H F G

A F C G H F G

A	B	C	D	E	F	G	H	
	1	0	0	1	0	0	0	A
	0	0	0	1	0	0	0	B
		1	0	1	1	0	0	C
			0	0	0	1	0	D
				0	0	0	1	E
					1	0	0	F
						1	0	G
							1	H

Visita in profondità - DFS

idea algoritmo [7]



- Estrae nodo dalla pila (**A**) essendo "bianco" lo visita e lo colora in grigio.



pila

F C G H F G

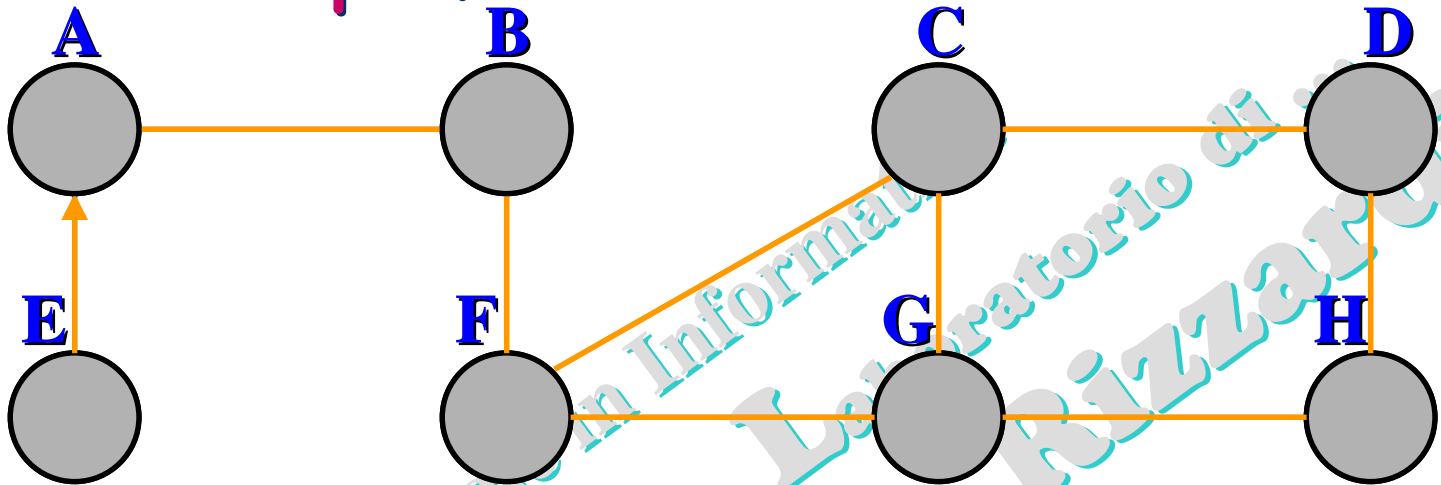
- Inserisce nella **pila** i vertici adiacenti ad **A**: nodi **B**, **E** (cioè equivale a percorrere gli archi che partono da **A** – l'arco arancio è stato percorso 2 volte).

B E F C G H F G

A	B	C	D	E	F	G	H	
1	0	0	1	0	0	0	0	A
0	0	0	1	0	0	0	0	B
1	0	1	1	0	0	0	0	C
0	0	0	1	0	0	0	0	D
0	0	0	0	1	0	0	0	E
1	0	0	0	0	1	0	0	F
1	0	0	0	0	0	1	0	G
0	0	0	0	0	0	0	1	H

Visita in profondità - DFS

idea algoritmo [8]



- Estrae nodo dalla pila (**B**) essendo "grigio" **non** lo visita.
- Estrae nodo dalla pila (**E**), essendo "bianco" lo visita e lo colora in grigio.
- Inserisce nella **pila** i vertici adiacenti ad **E**: nodo **A** (ciò equivale a percorrere gli archi che partono da **E** – l'arco arancio è stato percorso 2 volte).

B

pila

E F C G H F G

F C G H F G

E

A F C G H F G

- Svuota la pila estraendo, uno alla volta, i nodi che ormai sono tutti "grigi".

ordine di visita: C, D, H, G, F, B, A, E

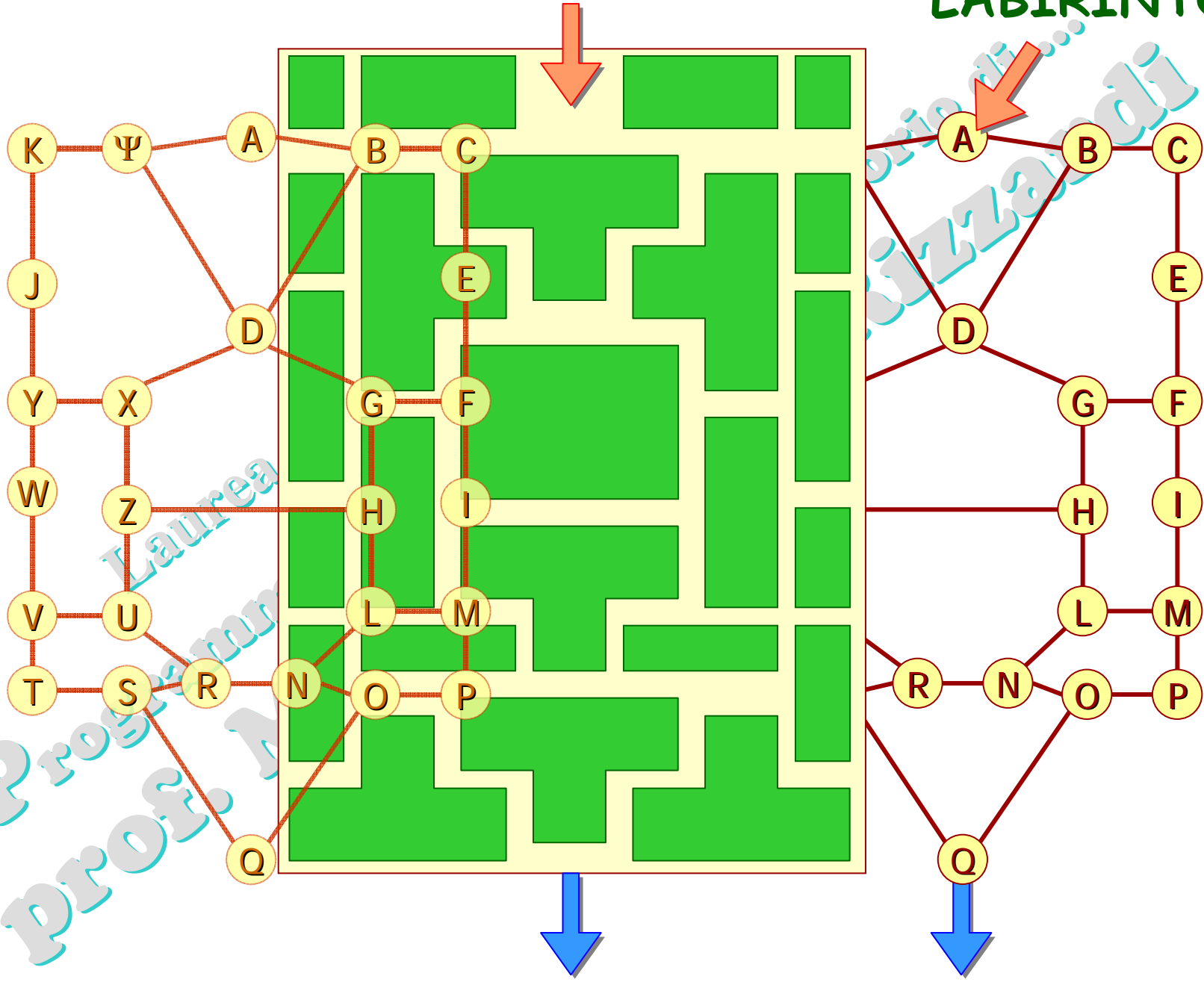
A	B	C	D	E	F	G	H	
	1	0	0	1	0	0	0	A
	0	0	0	1	0	0	0	B
	1	0	1	1	0			C
		0	0	0	1			D
			0	0	0			E
				1	0			F
					1			G
								H

Algoritmo Depth First Search iterativo migliorato

```
procedure DFS_visit_iterat (var G: Grafo, var N: integer, var V: Vertice)
{
  visita V; v_count:=1;
  inserisci nella pila i nodi adiacenti a V;
  while pila "non vuota" and v_count < N
    estrai nodo U dalla pila;
    if nodo U "non visitato"
      visita U; v_count:=v_count+1;
    endif
    inserisci nella pila i nodi adiacenti a U;
  endwhile
}
```

N numero totale
di vertici nel grafo

Applicazione dell'algoritmo DFS: attraversamento di un LABIRINTO



Esercizi:

1

Scrivere *function C* per la visita in ordine anticipato di un albero qualsiasi. [liv. 2]

2

Scrivere *function C* per la visita di un grafo mediante l'algoritmo *Depth First Search* iterativo. Applicare l'algoritmo al grafo che descrive il labirinto della pagina precedente per trovare l'uscita Q a partire dall'ingresso A. [liv. 3]