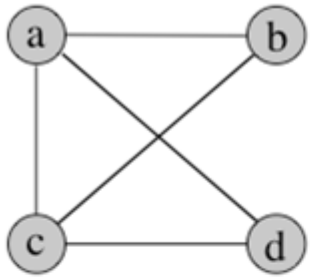
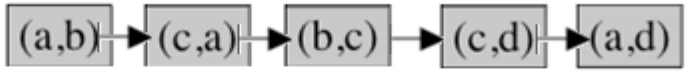
Strutture dati per rappresentare grafi

Grafi non diretti

quanto spazio?

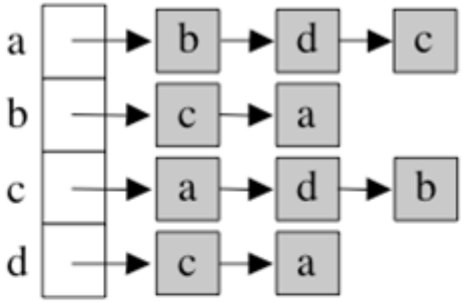


(a) Grafo non orientato G



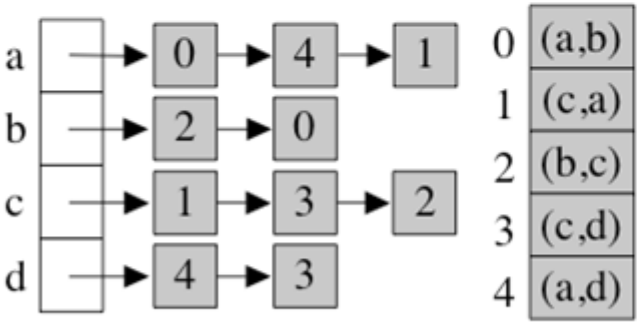
O(m)

(b) Lista di archi di G



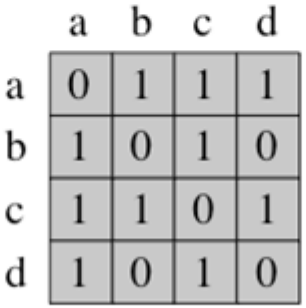
O(m + n)

(c) Liste di adiacenza di G



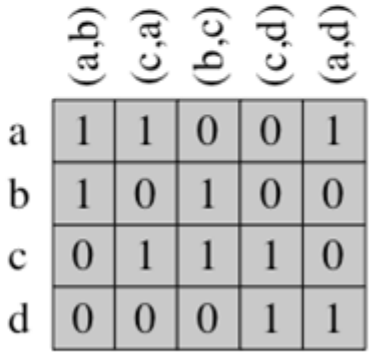
O(m + n)

(d) Liste di incidenza di G



$O(n^2)$

(e) Matrice di adiacenza di G

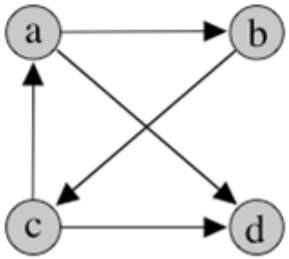


O(m n)

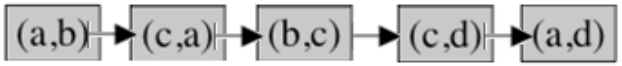
(f) Matrice di incidenza di G

Grafi diretti

quanto spazio?

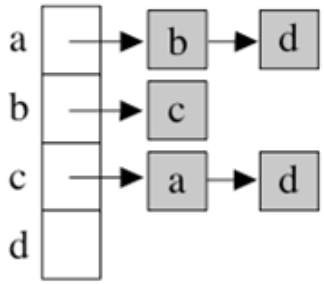


(a) Grafo orientato G



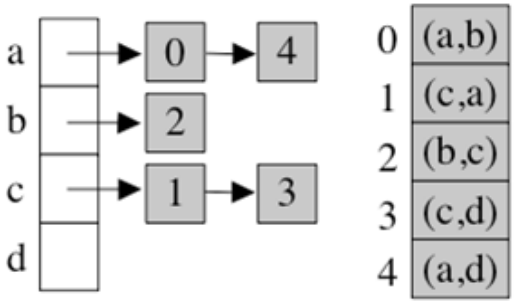
O(m)

(b) Lista di archi di G



O(m + n)

(c) Liste di adiacenza di G



(d) Liste di incidenza di G

O(m + n)

	a	b	c	d
a	0	1	0	1
b	0	0	1	0
c	1	0	0	1
d	0	0	0	0

(e) Matrice di adiacenza di G

$O(n^2)$

	(a,b)	(c,a)	(b,c)	(c,d)	(a,d)
a	1	-1	0	0	1
b	-1	0	1	0	0
c	0	1	-1	1	0
d	0	0	0	-1	-1

(f) Matrice di incidenza di G

O(m n)

Prestazioni della lista di archi (grafi non diretti)

Operazione	Tempo di esecuzione
`grado`(v)	$\Theta(m)$
`archiIncidenti`(v)	$\Theta(m)$
`sonoAdiacenti`(x, y)	$O(m)$
`aggiungiVertice`(v)	$O(1)$
`aggiungiArco`(x, y)	$O(1)$
`rimuoviVertice`(v)	$\Theta(m)$
`rimuoviArco`(e)	$O(1)$

Prestazioni liste di adiacenza (grafi non diretti)

Operazione	Tempo di esecuzione
`grado`(v)	$\Theta(\delta(v))$
`archiIncidenti`(v)	$\Theta(\delta(v))$
`sonoAdiacenti`(x, y)	$O(\min\{\delta(x), \delta(y)\})$
`aggiungiVertice`(v)	$O(1)$
`aggiungiArco`(x, y)	$O(1)$
`rimuoviVertice`(v)	$O(m)$
`rimuoviArco`$(e = (x, y))$	$O(\delta(x) + \delta(y))$

Prestazioni matrice di adiacenza (grafi non diretti)

Operazione	Tempo di esecuzione
grado(v)	$\Theta(n)$
archiIncidenti(v)	$\Theta(n)$
sonoAdiacenti(x, y)	$O(1)$
aggiungiVertice(v)	$\Theta(n)$
aggiungiArco(x, y)	$O(1)$
rimuoviVertice(v)	$O(n^2)$
rimuoviArco$(e = (x, y))$	$O(1)$

Algoritmi e Strutture Dati

Capitolo 11
Visite di grafi

quali parti del grafo
sono raggiungibili
da un certo
nodo?

...eseguo una *visita* del grafo

Scopo e tipi di visita

- Una visita (o attraversamento) di un grafo G permette di esaminare i nodi e gli archi di G in modo sistematico (se G è connesso)
- Problema di base in molte applicazioni
- Esistono vari tipi di visite con diverse proprietà: in particolare, visita in ampiezza (BFS=breadth first search) e visita in profondità (DFS=depth first search)

Algoritmo di visita generica

- La visita parte da un vertice s prescelto ed esplora seguendo una qualche regola uno dei suoi adiacenti
- Un vertice v raggiunto da u viene marcato come visitato se è stato incontrato per la prima volta, e viene quindi aggiunto alla frangia F di visita; inoltre, il nodo u diventa padre di v, e l'arco (u,v) viene etichettato come arco di visita
- Un vertice rimane nella frangia di visita fintantoché non sono stati esplorati tutti i suoi adiacenti
- La visita genera un albero di copertura T del grafo

Visite particolari

- Se la frangia F è implementata come coda si ha la visita in ampiezza (BFS)
- Se la frangia F è implementata come pila si ha la visita in profondità (DFS)

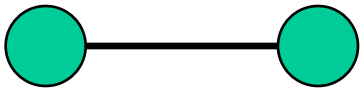
Visita in ampiezza

applicazioni

- web crawling
  - come google trova nuove pagine da indicizzare
- social networking
  - trovare gli amici che potresti conoscere
- network broadcast
  - un nodo manda un messaggio a tutti gli altri nodi della rete
- garbage collection
  - come scoprire memoria non più raggiungibile che si può liberare
- model checking
  - verificare una proprietà di un sistema
- risolvere puzzle
  - risolvere il Cubo di Rubik con un numero minimo di mosse

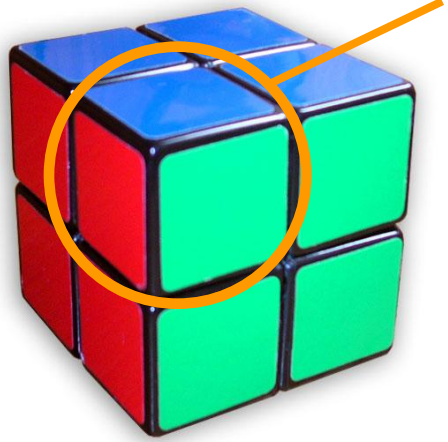
cubo di Rubik: 2x2x2

- grafo delle configurazioni
  - un vertice per ogni possibile stato del cubo
  - un arco fra due configurazioni se l'una è ottenibile dall'altra tramite una mossa



grafo non diretto

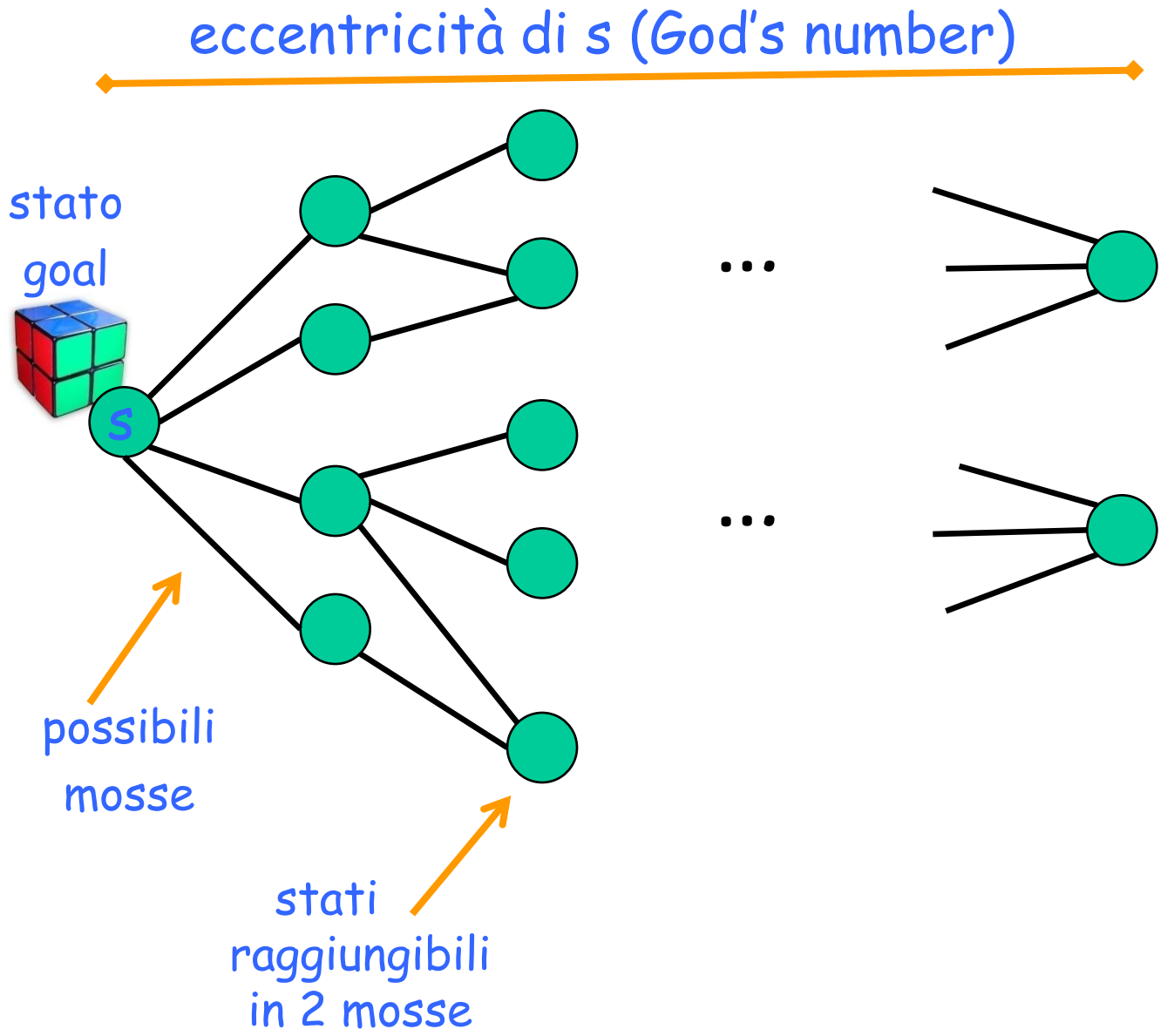
cubetto



#verciti $\leq 8! \times 3^8$

$= 264.539.520$

cubo di Rubik: 2x2x2



God's number

2x2x2: 11

3x3x3: 20

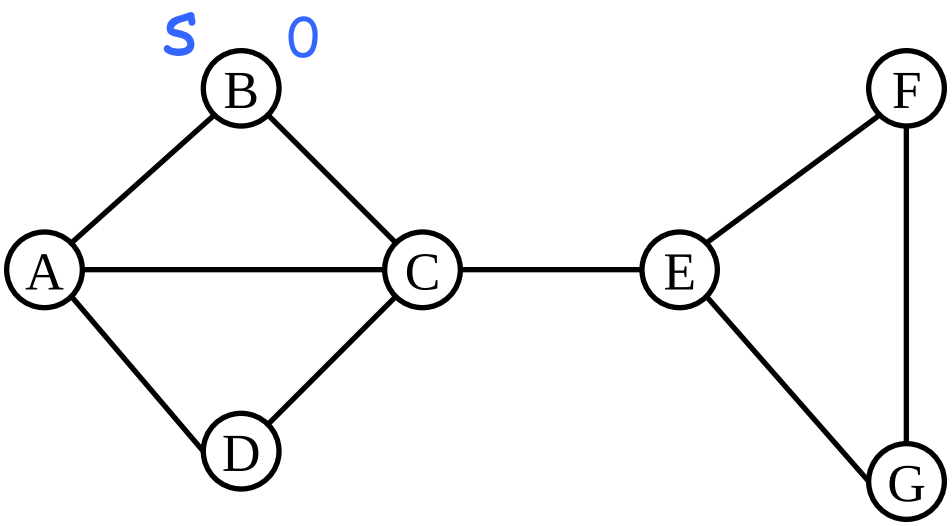
4x4x4: ???

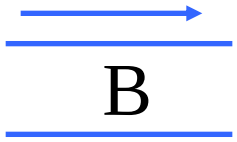
nxnxn: $\Theta(n^2/\log n)$

Visita in ampiezza

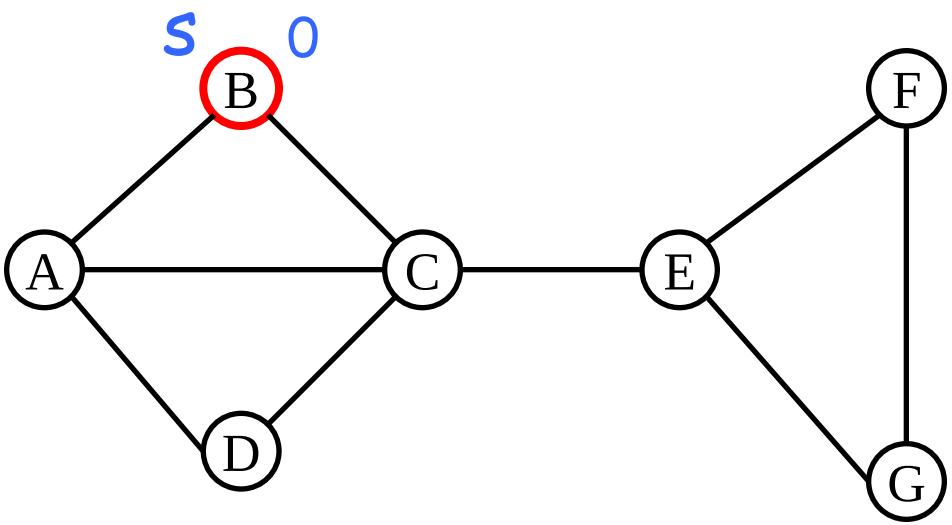
```
algoritmo visitaBFS(vertice s) → albero
1.    rendi tutti i vertici non marcati
2.    T ← albero formato da un solo nodo s
3.    Coda F
4.    marca il vertice s
5.    F.enqueue(s)
6.    while ( not F.isempty() ) do
7.        u ← F.dequeue()
8.        for each ( arco (u, v) in G ) do
9.            if ( v non è ancora marcato ) then
10.               F.enqueue(v)
11.               marca il vertice v
12.               rendi u padre di v in T
13.   return T
```

Un esempio

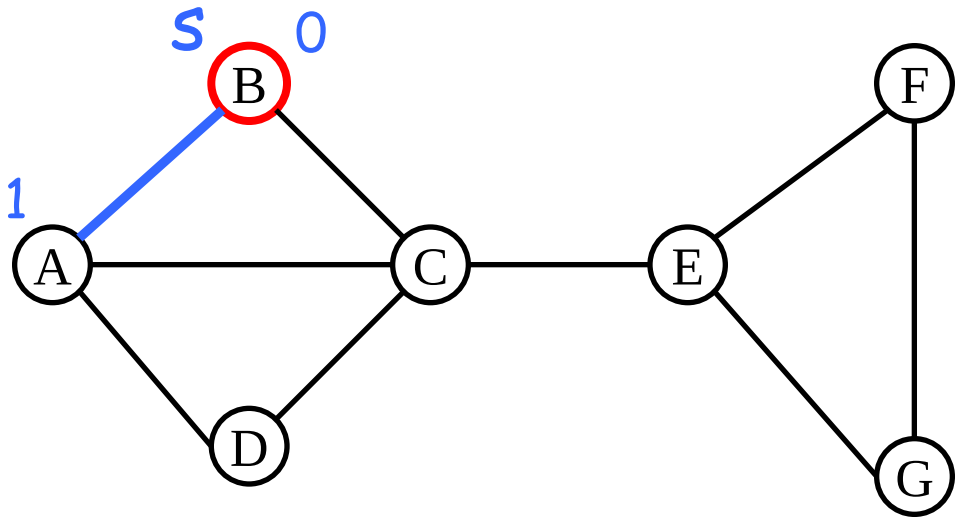


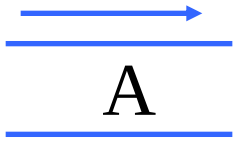


Un esempio

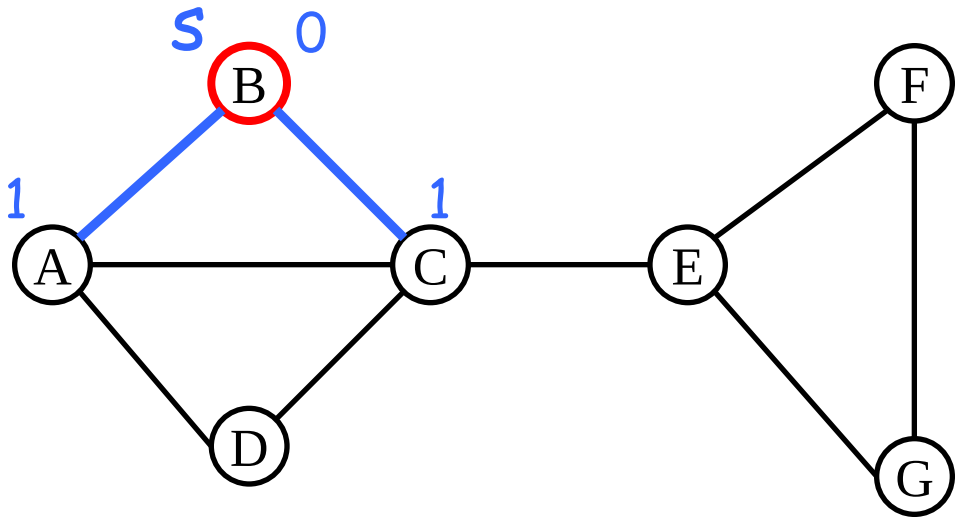


Un esempio



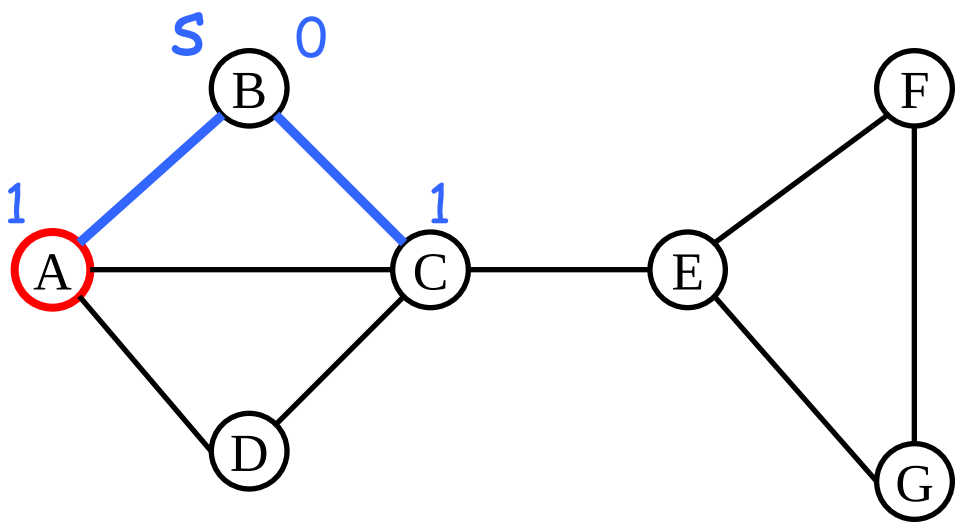


Un esempio



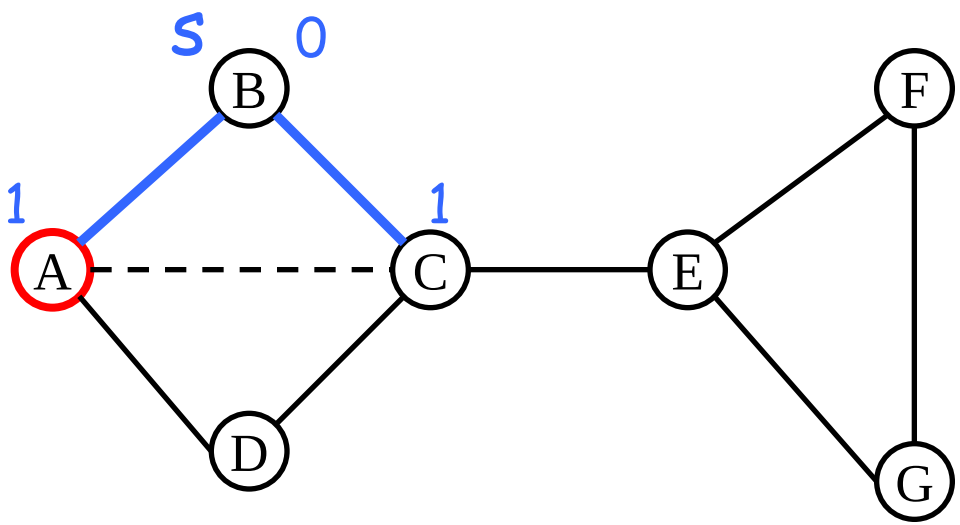
$\overrightarrow{CA}$

Un esempio



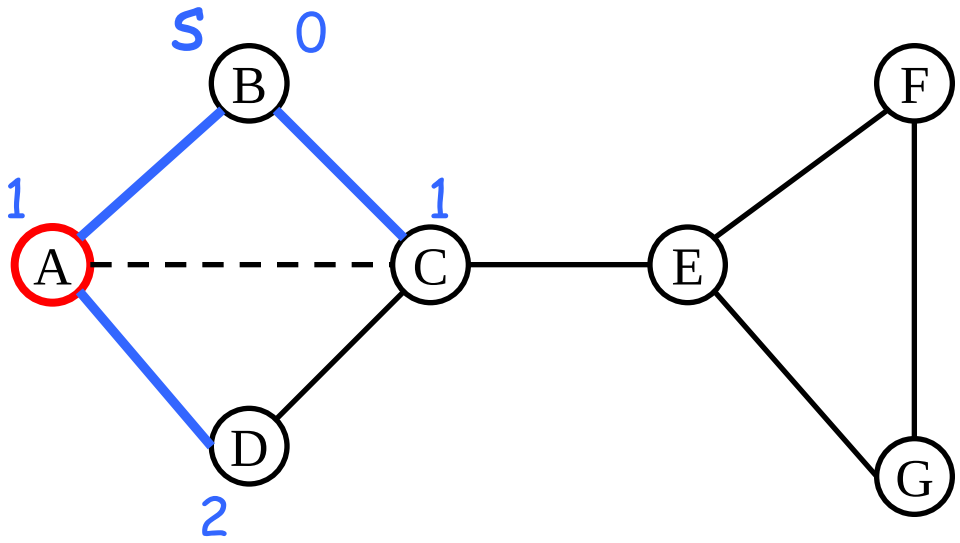
$\overrightarrow{C}$

Un esempio



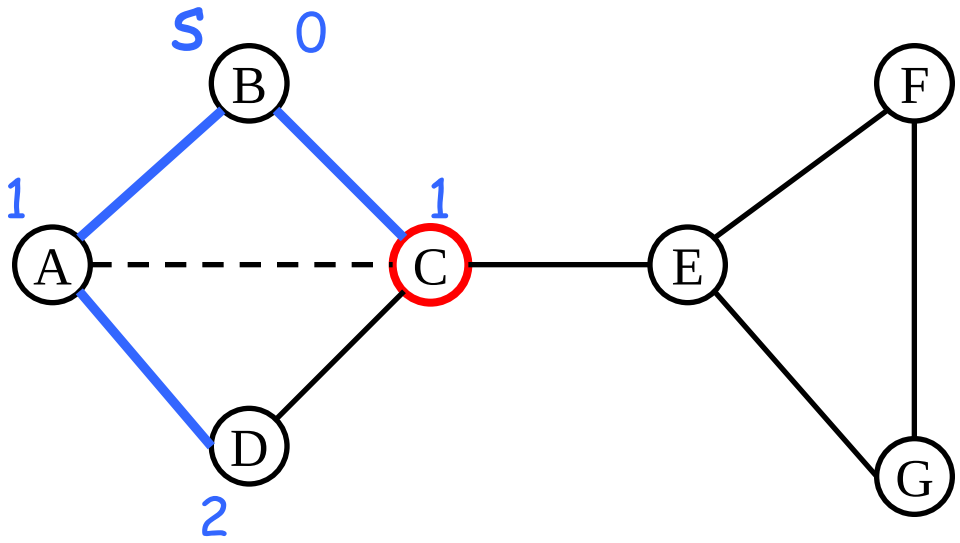
$\overrightarrow{C}$

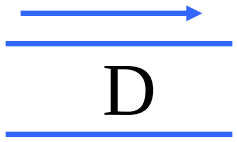
Un esempio



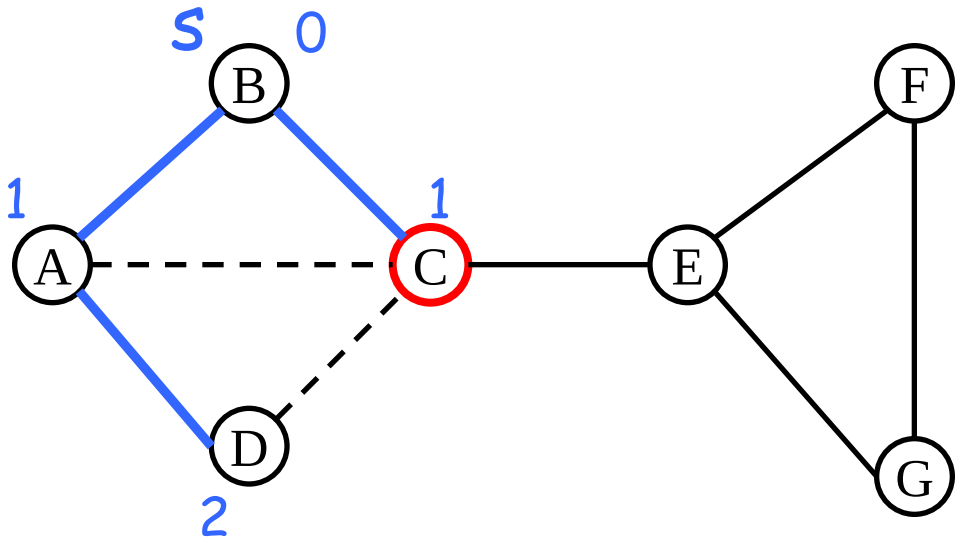
$\overrightarrow{DC}$

Un esempio



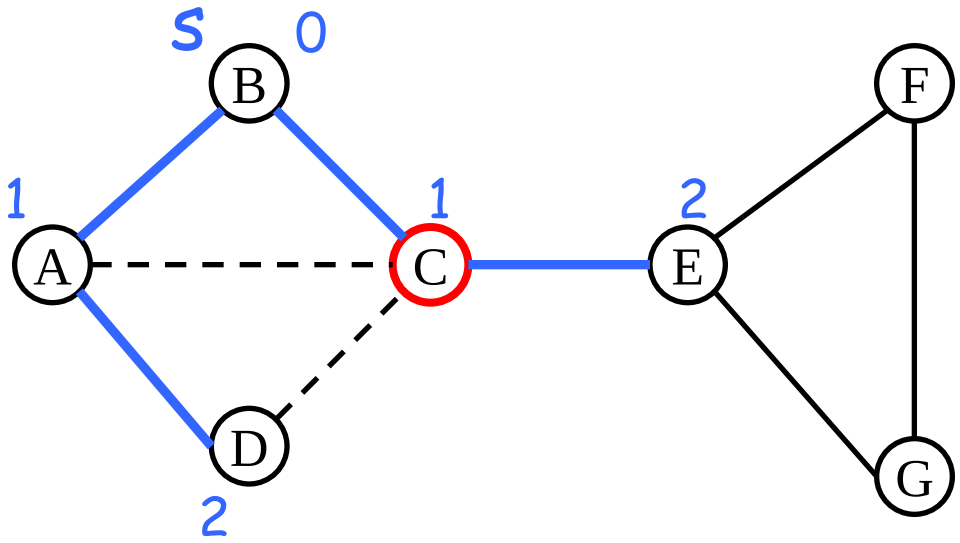


Un esempio



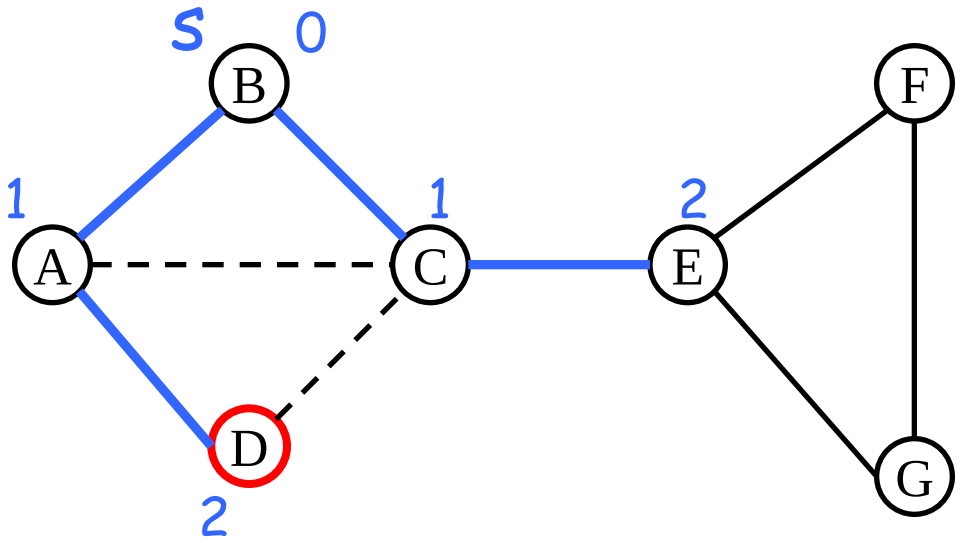
$\overrightarrow{D}$

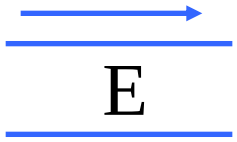
Un esempio



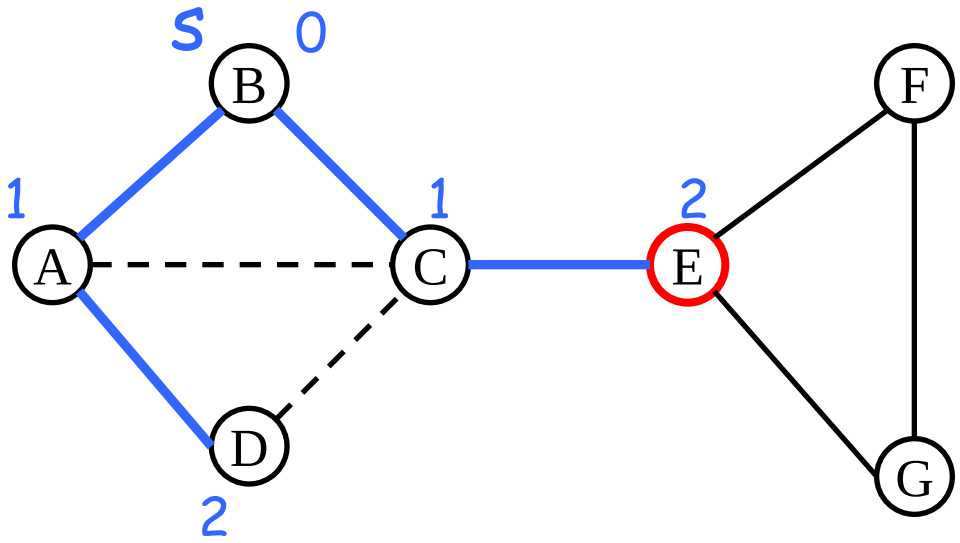
$\overrightarrow{ED}$

Un esempio

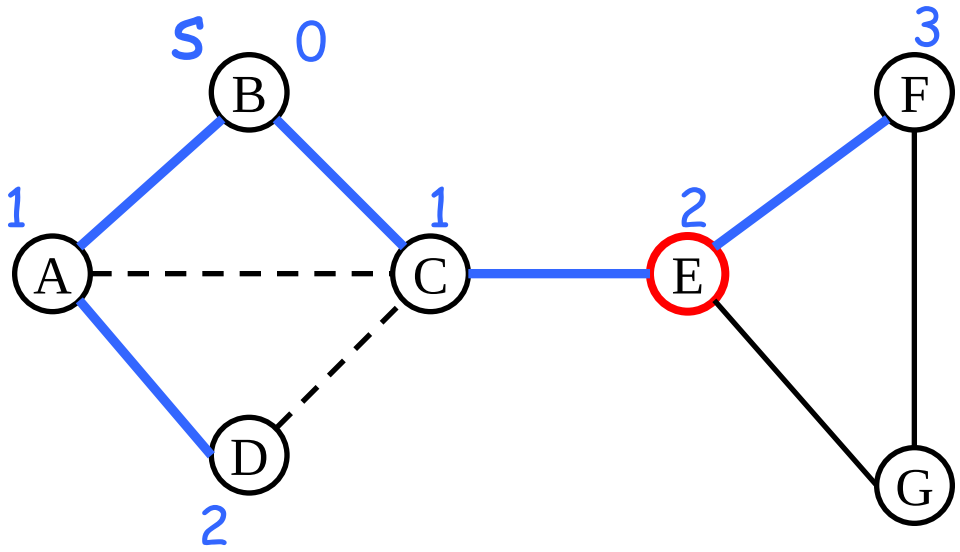


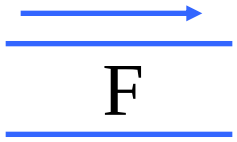


Un esempio

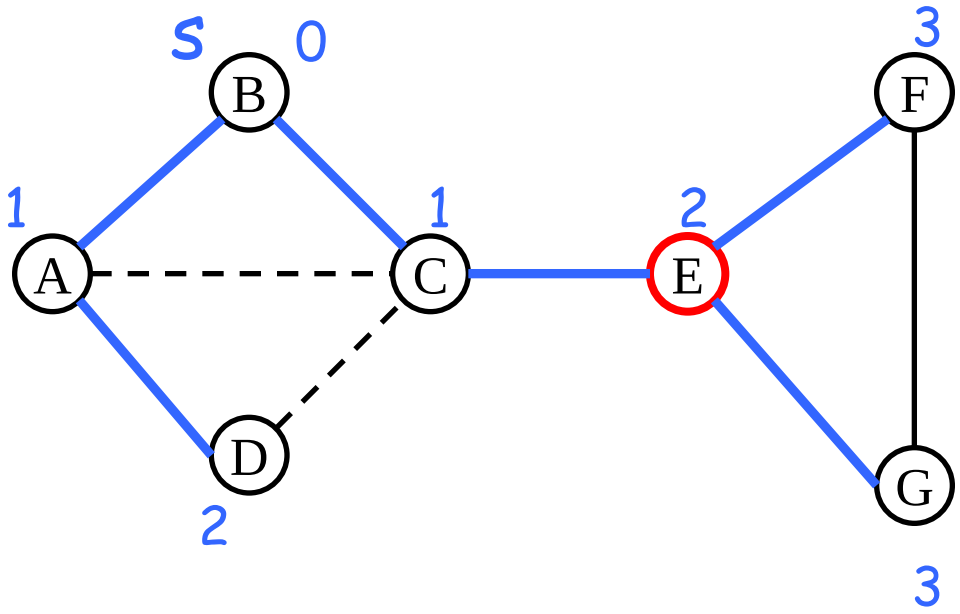


Un esempio

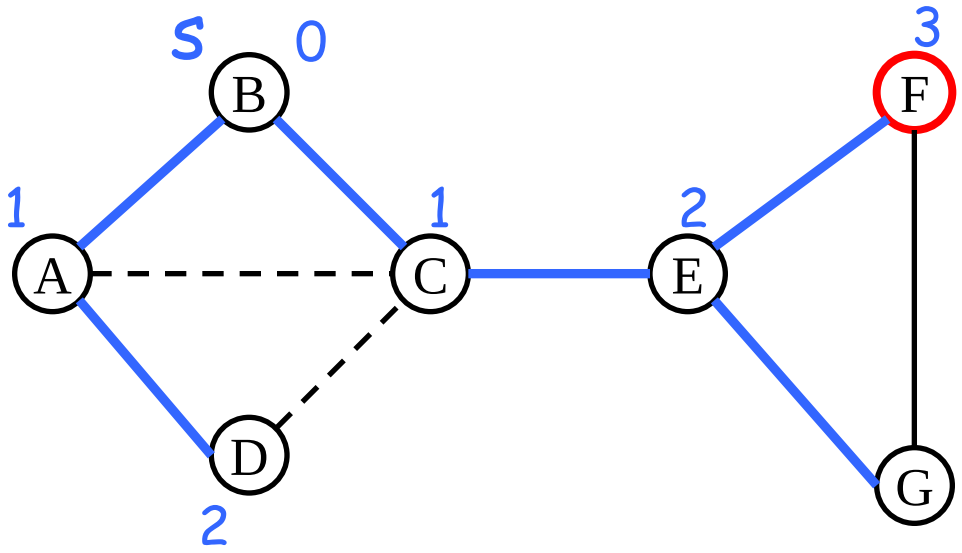




Un esempio

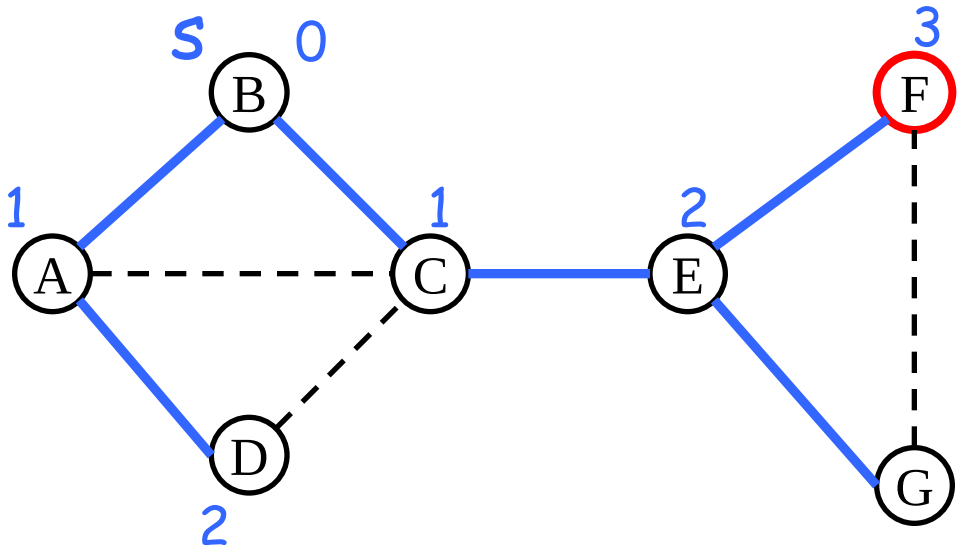


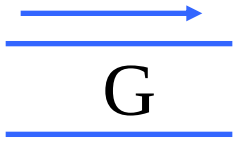
Un esempio



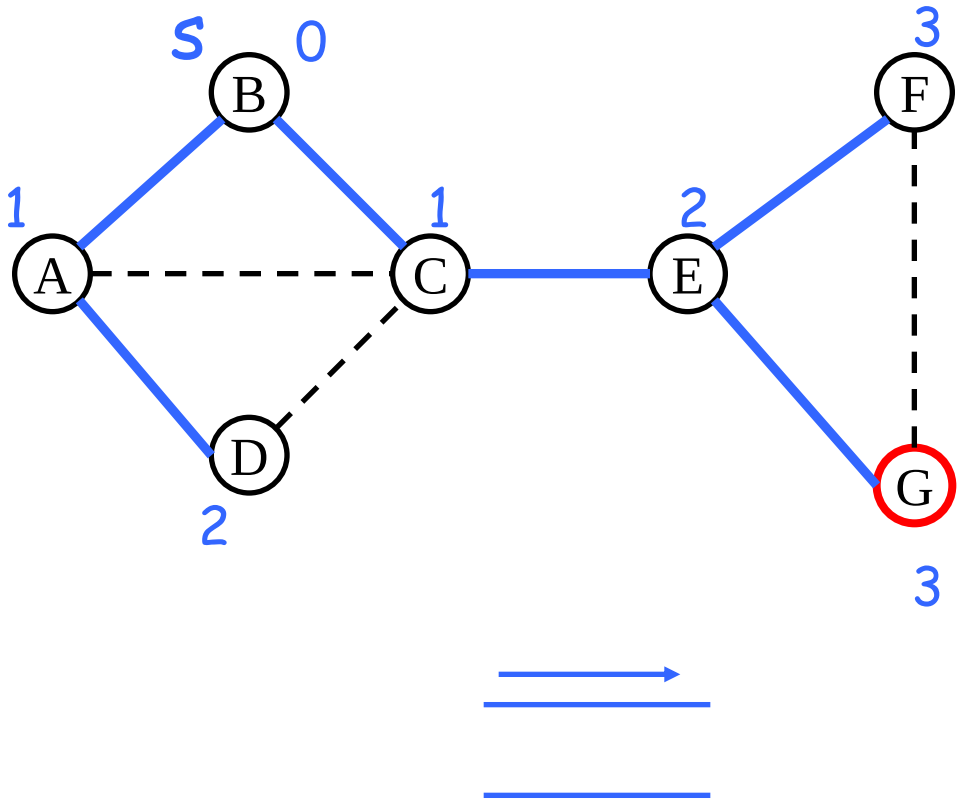
$\overrightarrow{G}$

Un esempio

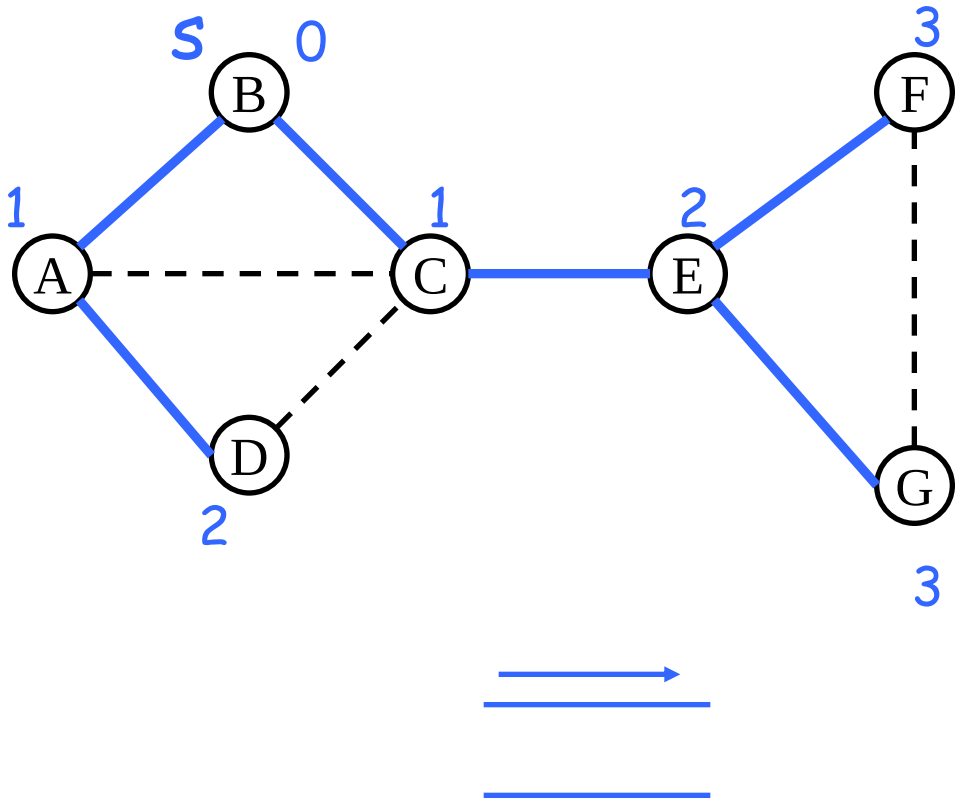




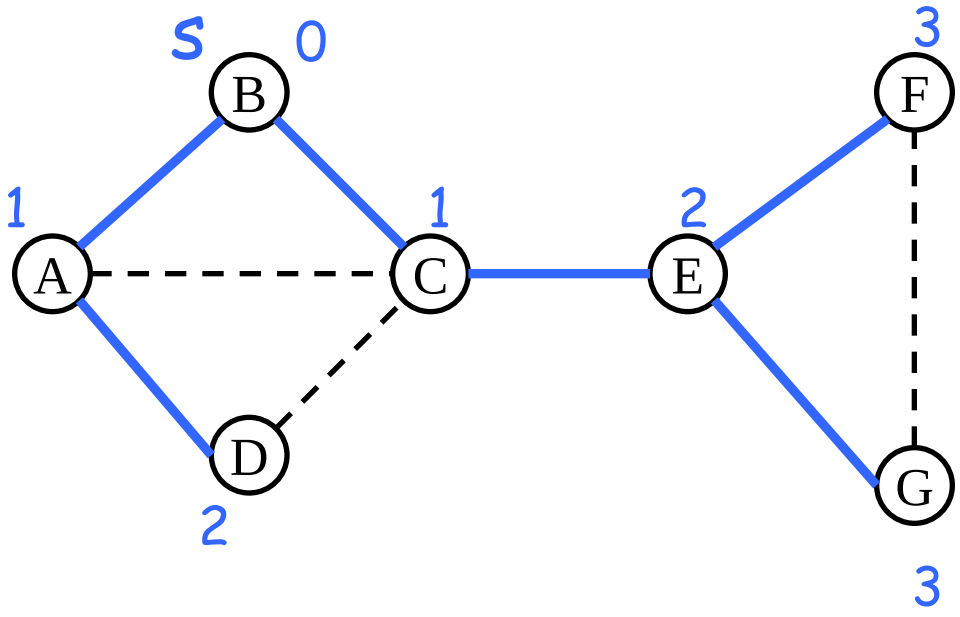
Un esempio



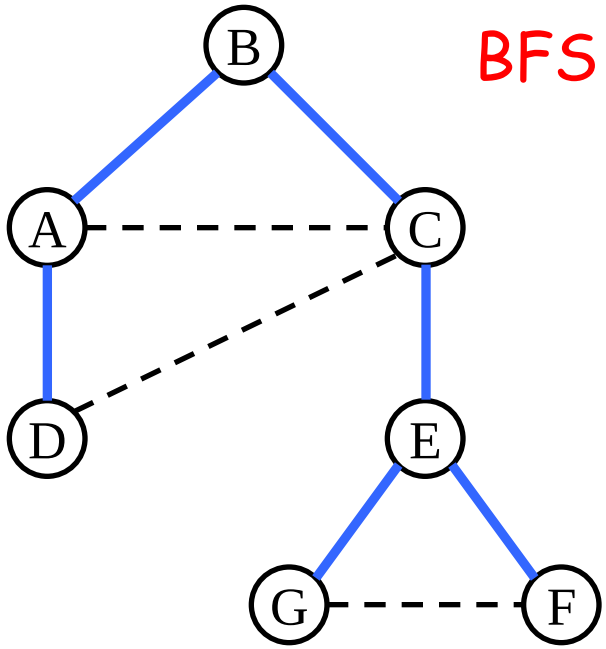
Un esempio



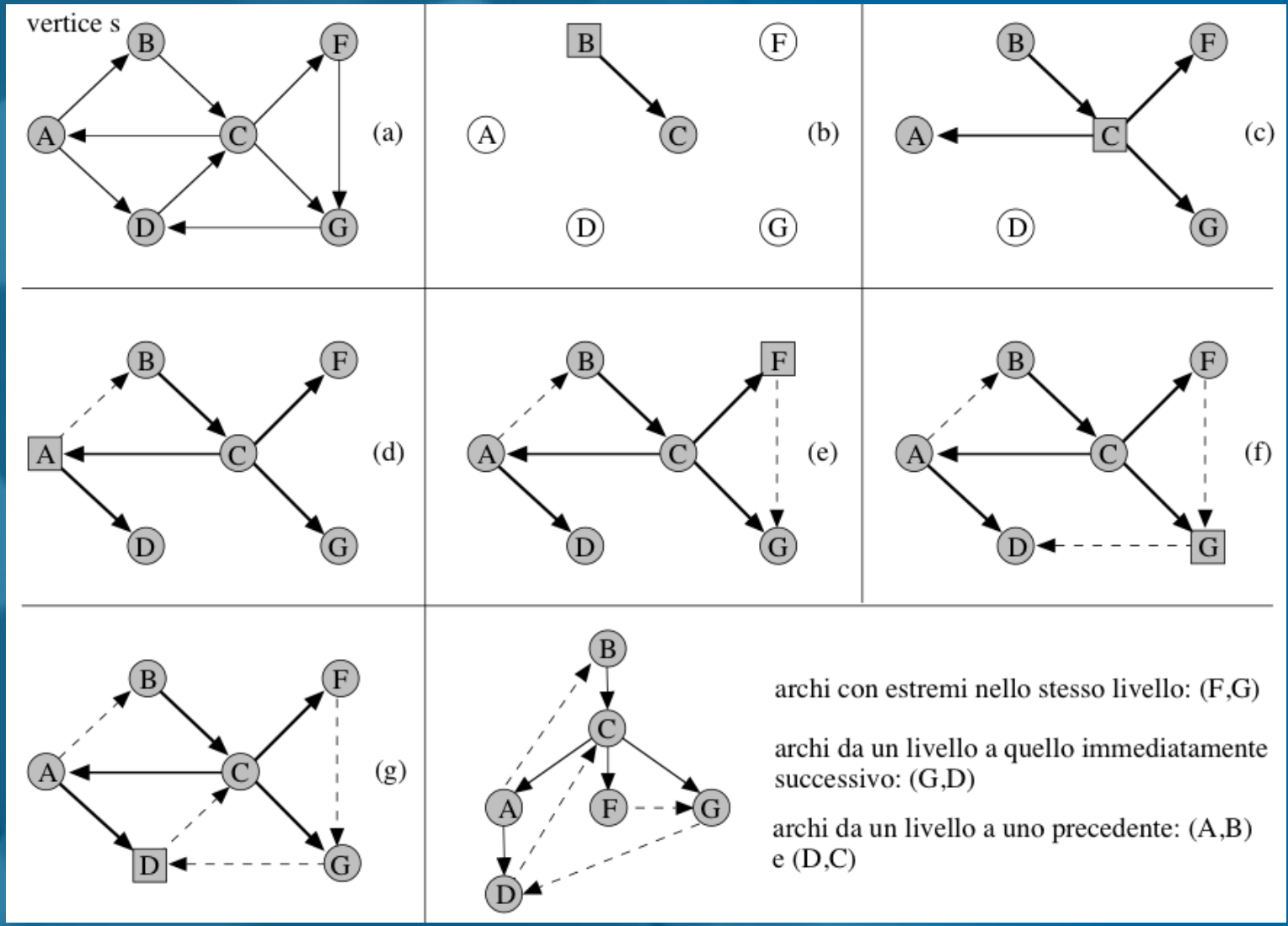
Un esempio



albero
BFS



Esempio: grafo orientato



Costo della visita in ampiezza

grafo rappresentato con matrice di adiacenza

```
algoritmo visitaBFS(vertice s) → albero
1.    rendi tutti i vertici non marcati
2.    T ← albero formato da un solo nodo s
3.    Coda F
4.    marca il vertice s
5.    F.enqueue(s)
6.    while ( not F.isempty() ) do
7.        u ← F.dequeue()
8.        for each ( arco (u, v) in G ) do
9.            if ( v non è ancora marcato ) then
10.               F.enqueue(v)
11.               marca il vertice v
12.               rendi u padre di v in T
13.   return T
```

$O(n^2)$

$O(n)$

Costo della visita in ampiezza

grafo rappresentato con liste di adiacenza

```
algoritmo visitaBFS(vertice s) → albero
1.    rendi tutti i vertici non marcati
2.    T ← albero formato da un solo nodo s
3.    Coda F
4.    marca il vertice s
5.    F.enqueue(s)
6.    while ( not F.isempty() ) do
7.        u ← F.dequeue()
8.        for each ( arco (u, v) in G ) do
9.            if ( v non è ancora marcato ) then
10.               F.enqueue(v)
11.               marca il vertice v
12.               rendi u padre di v in T
13.   return T
```

$O(m+n)$

$\sum_u O(\delta(u)) = O(m)$

$O(\delta(u))$

Costo della visita in ampiezza

Il tempo di esecuzione dipende dalla struttura dati usata per rappresentare il grafo (e dalla connettività o meno del grafo rispetto ad s):

- Liste di adiacenza: $O(m+n)$
- Matrice di adiacenza: $O(n^2)$

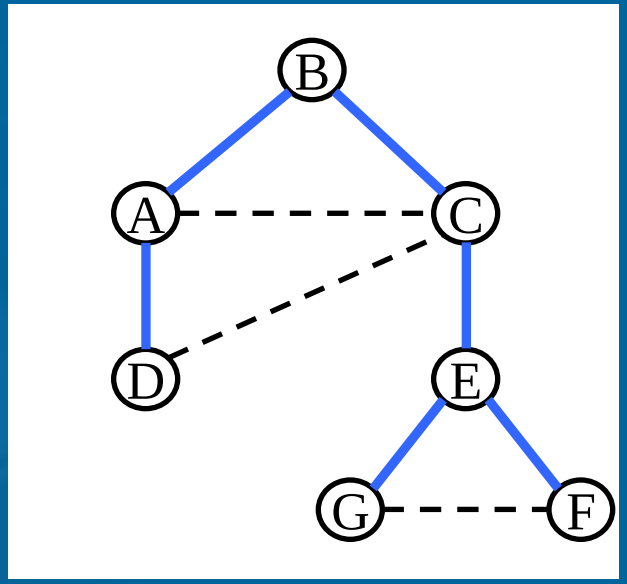
Osservazioni:

1. Si noti che se il grafo è connesso allora $m \geq n-1$ e quindi $O(m+n)=O(m)$
2. Ricordando che $m \leq n(n-1)/2$, si ha $O(m+n)=O(n^2)$

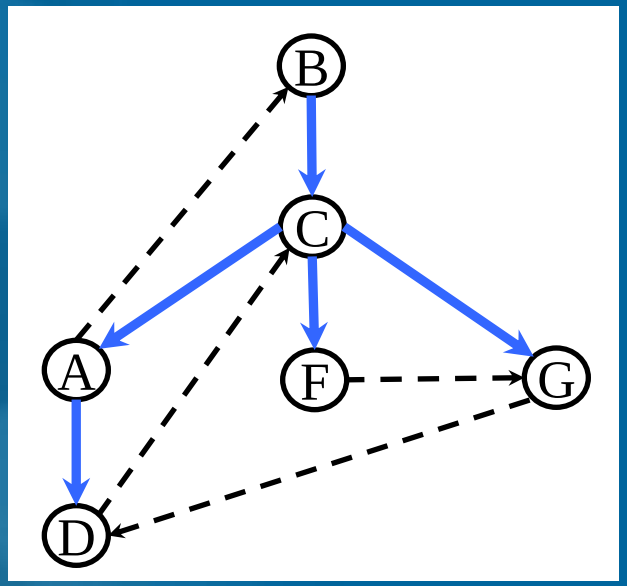
$\Rightarrow$ per $m=o(n^2)$ la rappresentazione mediante liste di adiacenza è temporalmente più efficiente!

Proprietà dell'albero BFS radicato in s

- Se il grafo è non orientato, per ogni arco (u,v) del grafo gli estremi u e v appartengono allo stesso livello o a livelli consecutivi dell'albero BFS



- Se il grafo è orientato, allora gli archi orientati verso il basso uniscono nodi sullo stesso livello o su livelli adiacenti, mentre gli archi orientati verso l'alto possono unire nodi su livelli non adiacenti



Proprietà dell'albero BFS radicato in s

- Per ogni nodo v, il livello di v nell'albero BFS è pari alla distanza di v dalla sorgente s (sia per grafi orientati che non orientati)
  - Perché? Conseguenza delle seguenti proprietà:

Proprietà 1

I nodi di G vengono inseriti nella coda F in ordine non decrescente di distanza dalla sorgente s

Proprietà 2

Quando un nodo v è inserito in F, il livello di v nell'albero BFS è uguale alla sua distanza da s

Proprietà dell'albero BFS radicato in s

- Per ogni nodo v, il livello di v nell'albero BFS è pari alla distanza di v dalla sorgente s (sia per grafi orientati che non orientati)

dimostrazione informale

- all'inizio inserisco s in F (che è a distanza 0 da se stesso) e gli assegno livello 0; chiaramente s è l'unico nodo a distanza 0.

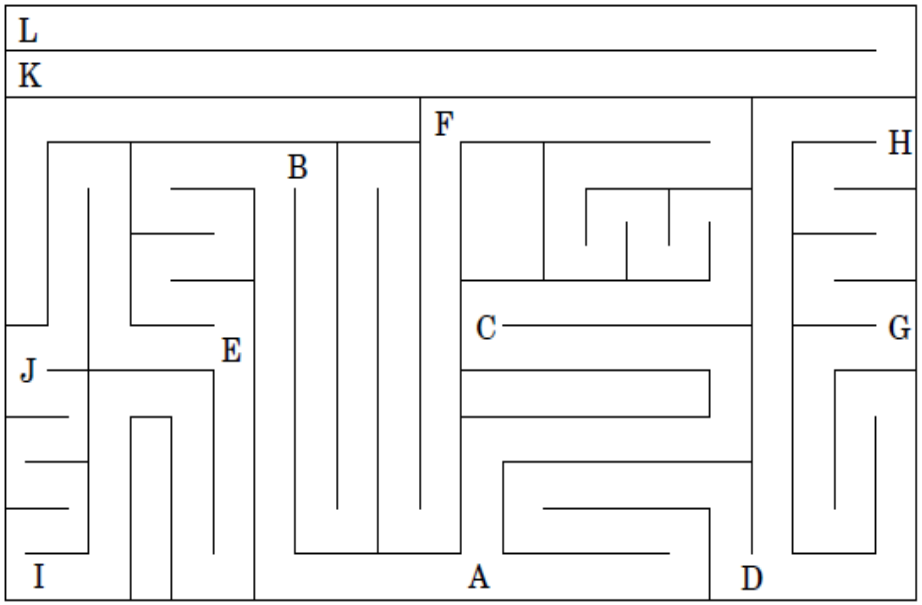
- estraggo s e guardo tutti suoi vicini; questi sono tutti i nodi a distanza 1 da s; li inserisco in F e assegno loro livello 1. Ora in F ho tutti i nodi a distanza 1.

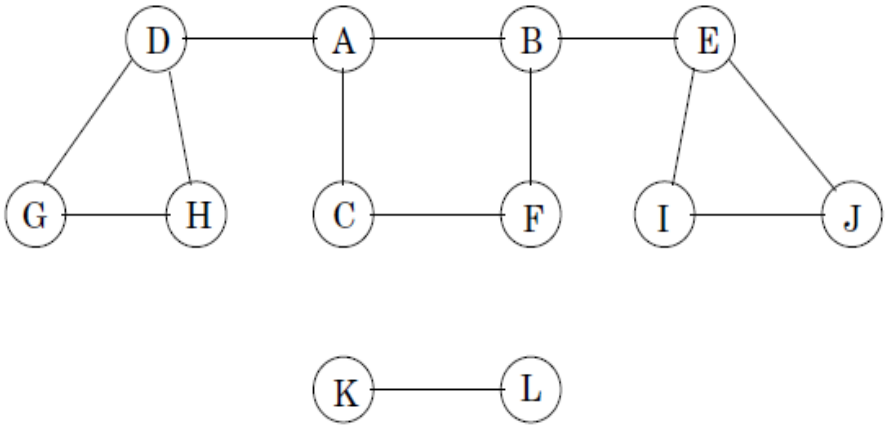
- estraggo uno a uno tutti i nodi di livello/distanza 1 e per ognuno guardo tutti suoi vicini; i vicini non marcati sono a distanza 2 da s; li inserisco in F e assegno loro livello 2; quando ho estratto e visitato tutti i nodi di livello 1, in F ho tutti i nodi a distanza 2 da s.

- estraggo uno a uno tutti i nodi di livello/distanza 2 e per ognuno guardo tutti suoi vicini; i vicini non marcati sono a distanza 3 da s...

Visita in profondità

un'analogia: esplorare un labirinto





Cosa mi serve?

gesso: per segnare le strade prese

variabile booleana: dice se un nodo è stato già visitato

corda: per tornare indietro se necessario

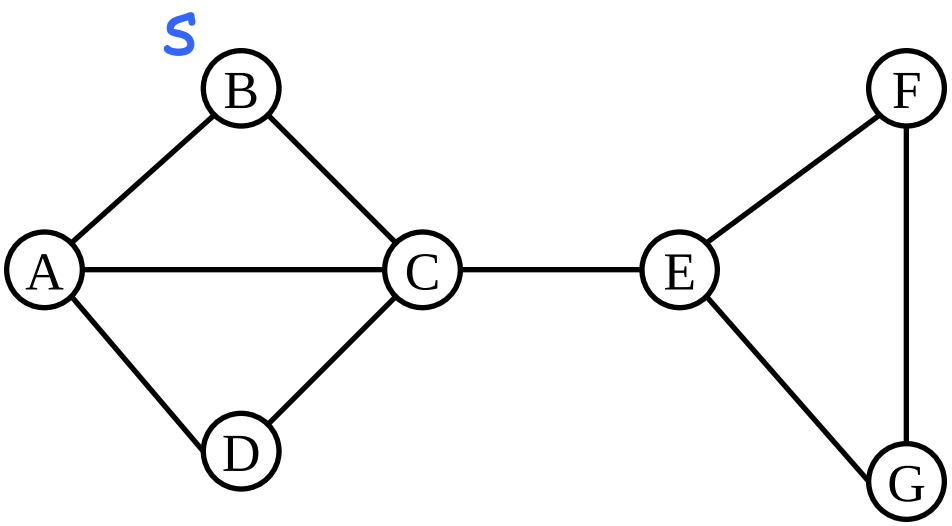
pila: push vuol dire srotolare
pop vuol dire arrotolare

Visita in profondità

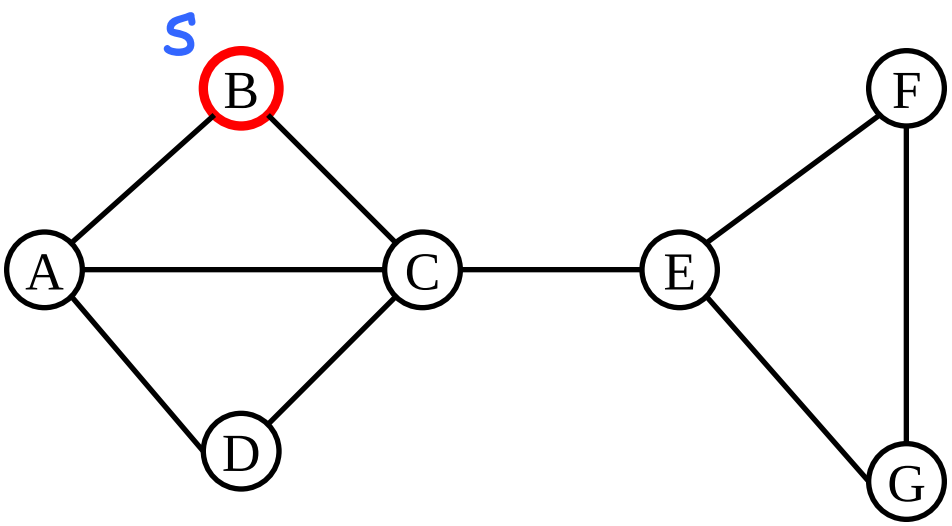
```
    procedura visitaDFSRicorsiva(vertice v, albero T)
1.      marca e visita il vertice v
2.      for each ( arco (v, w) ) do
3.          if ( w non è marcato ) then
4.              aggiungi l'arco (v, w) all'albero T
5.              visitaDFSRicorsiva(w, T)

    algoritmo visitaDFS(vertice s) → albero
6.      T ← albero vuoto
7.      visitaDFSRicorsiva(s, T)
8.      return T
```

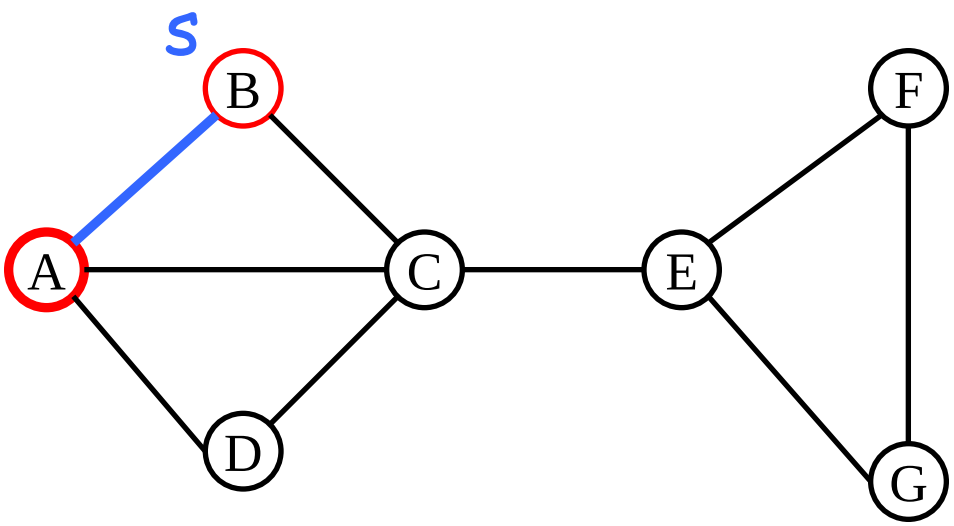
Un esempio: visita DFS



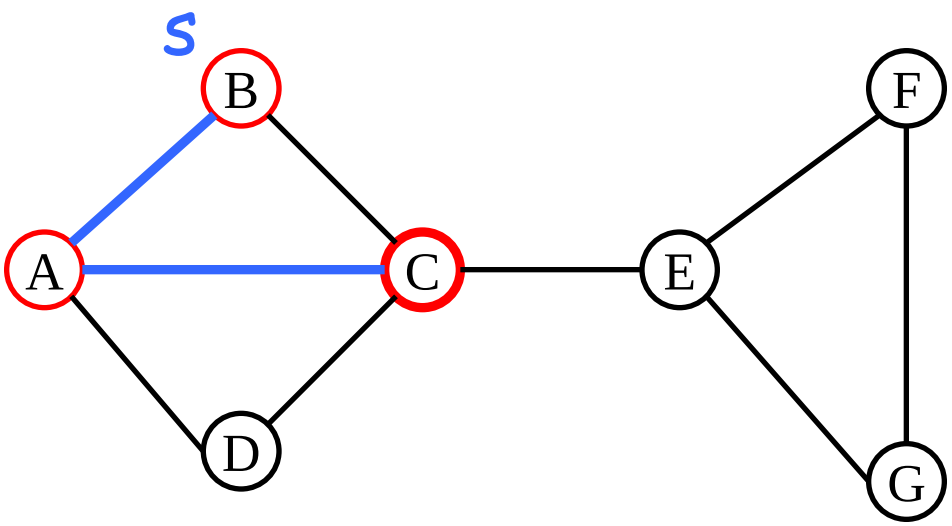
Un esempio: visita DFS



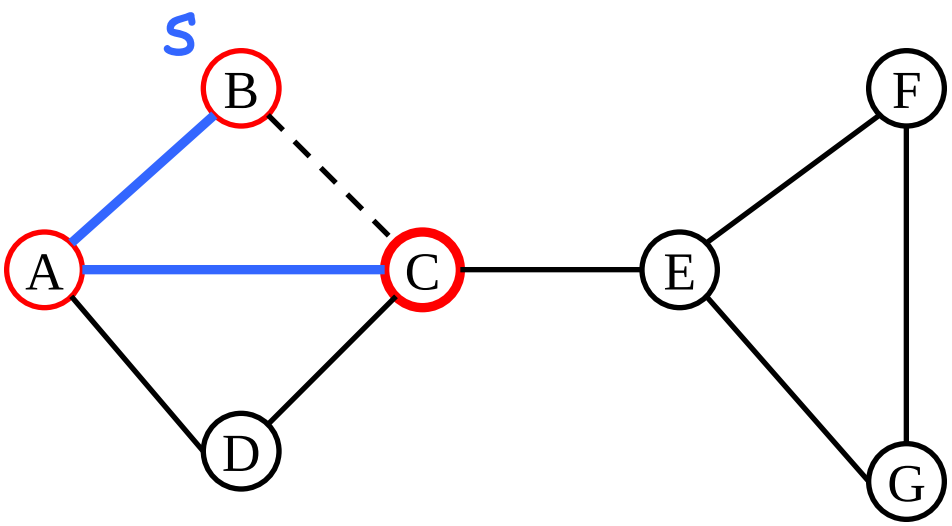
Un esempio: visita DFS



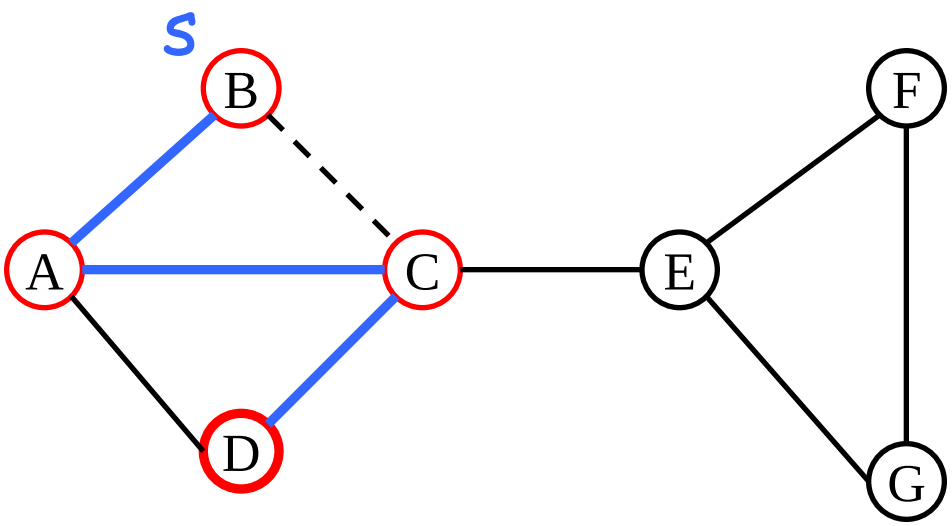
Un esempio: visita DFS



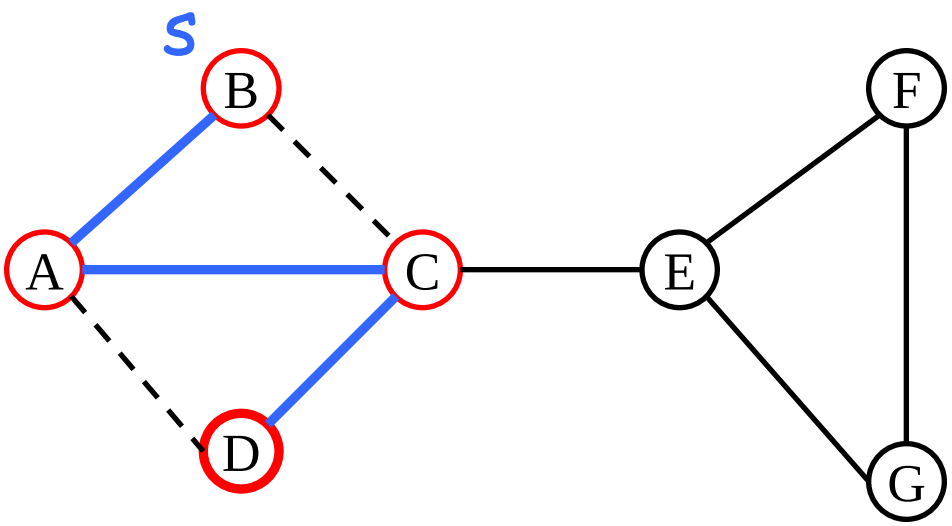
Un esempio: visita DFS



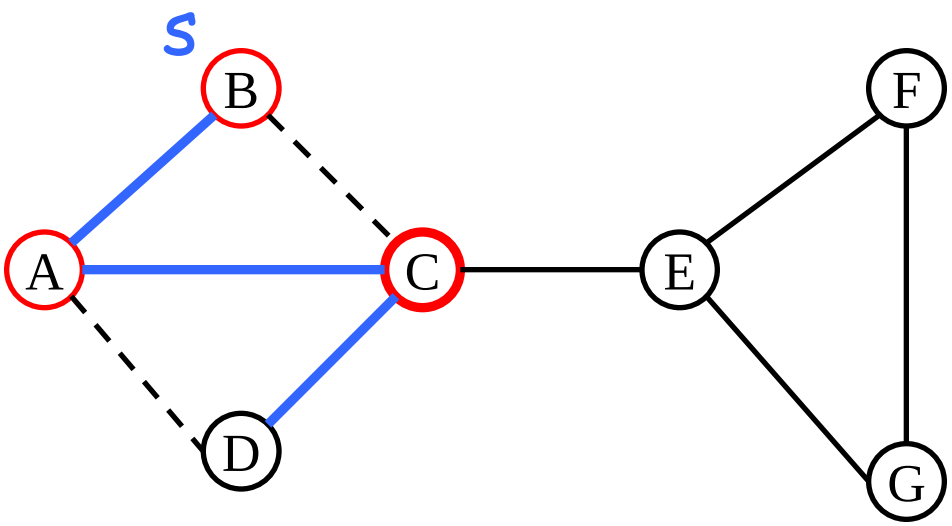
Un esempio: visita DFS



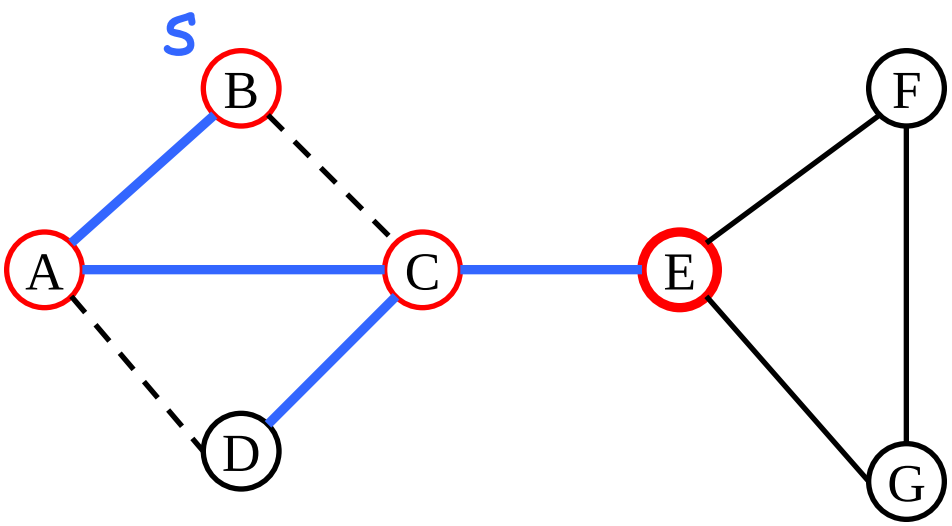
Un esempio: visita DFS



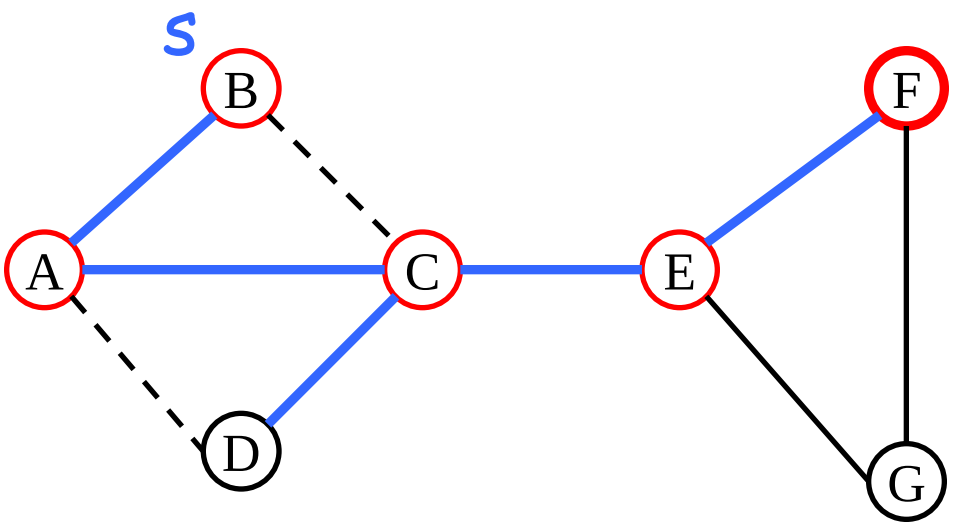
Un esempio: visita DFS



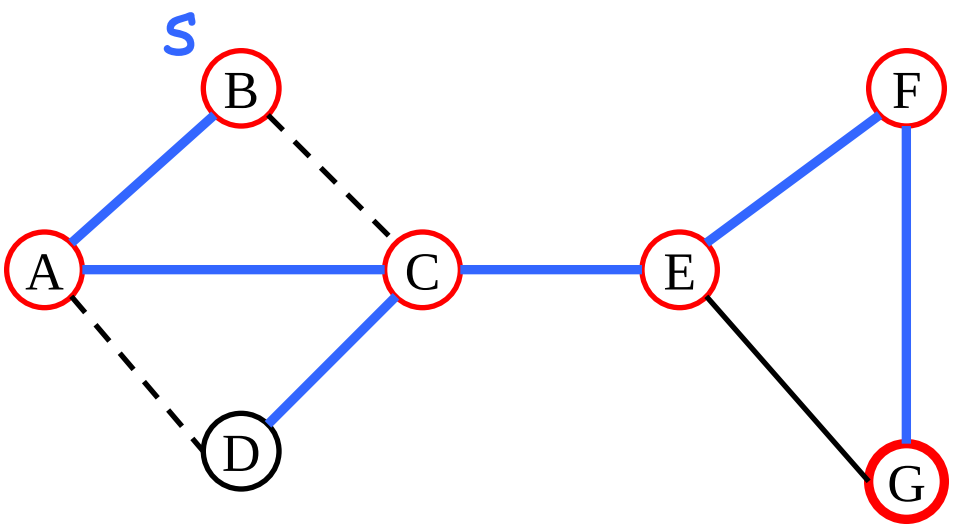
Un esempio: visita DFS



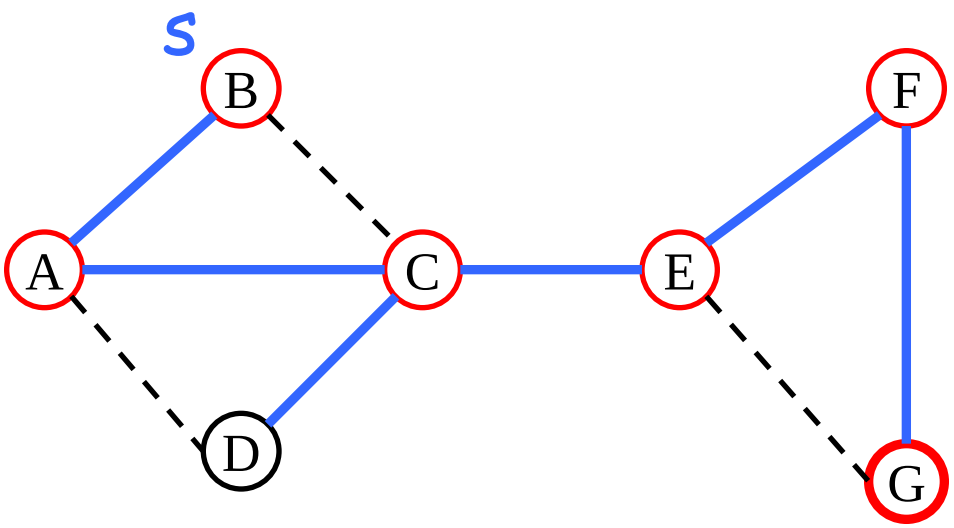
Un esempio: visita DFS



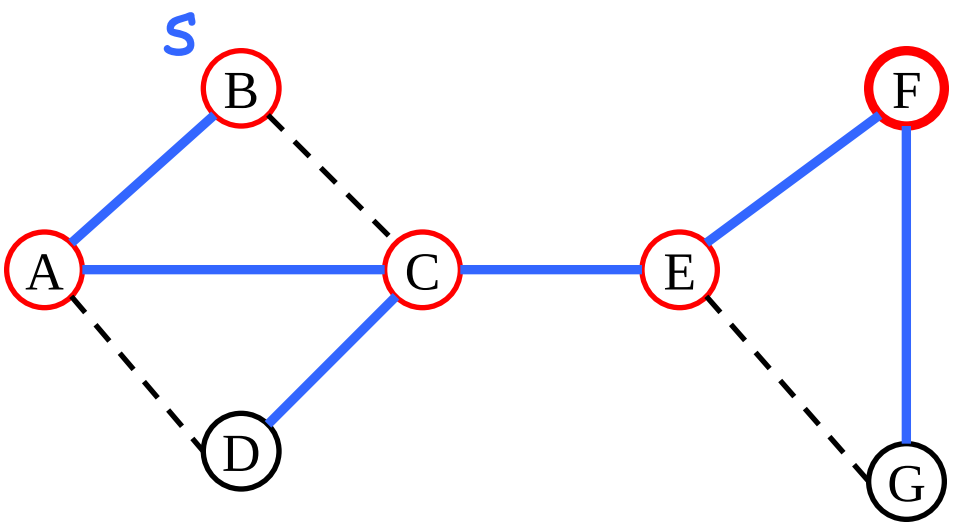
Un esempio: visita DFS



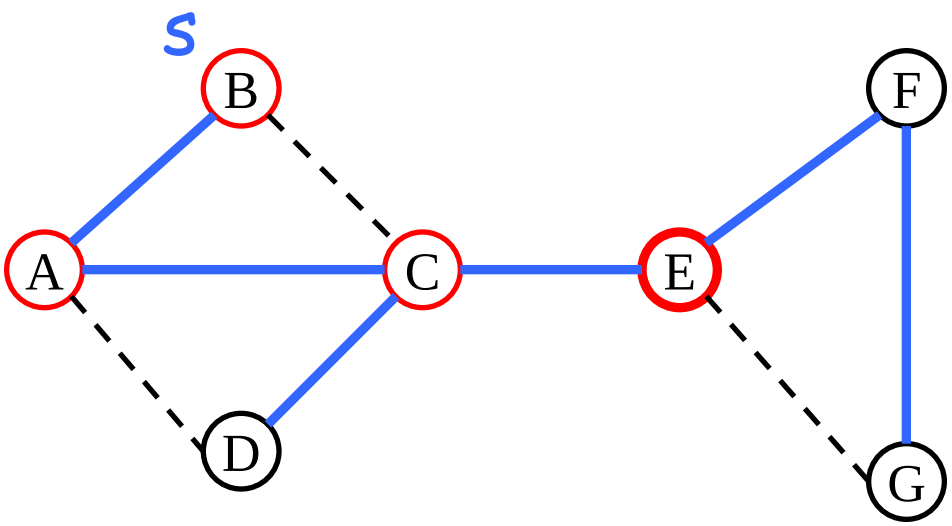
Un esempio: visita DFS



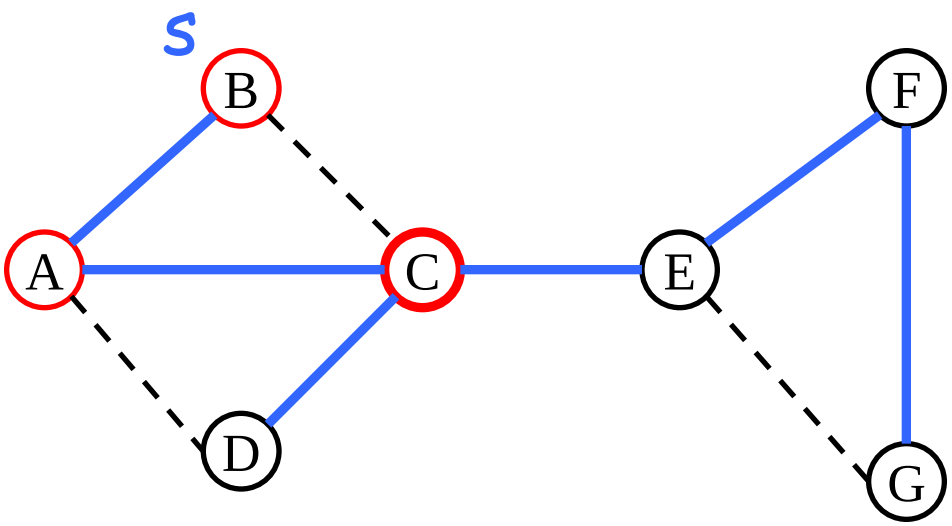
Un esempio: visita DFS



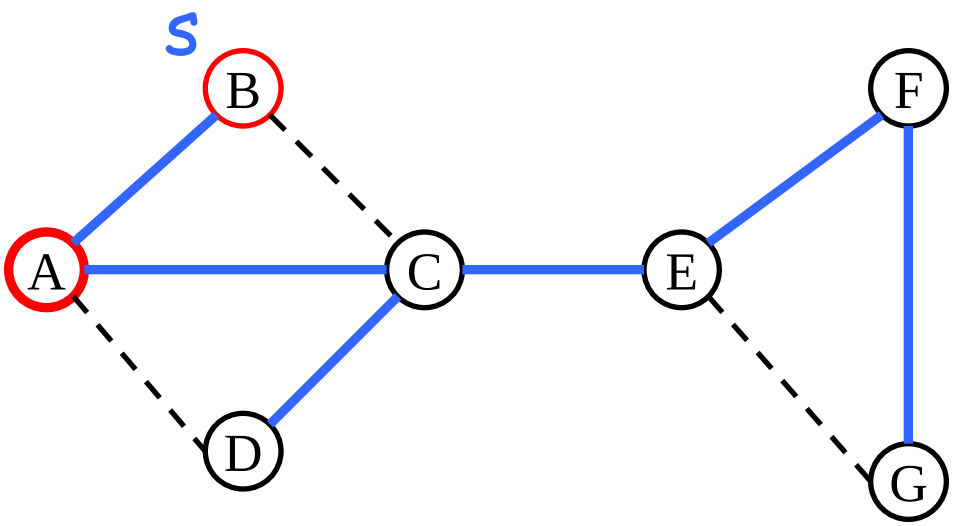
Un esempio: visita DFS



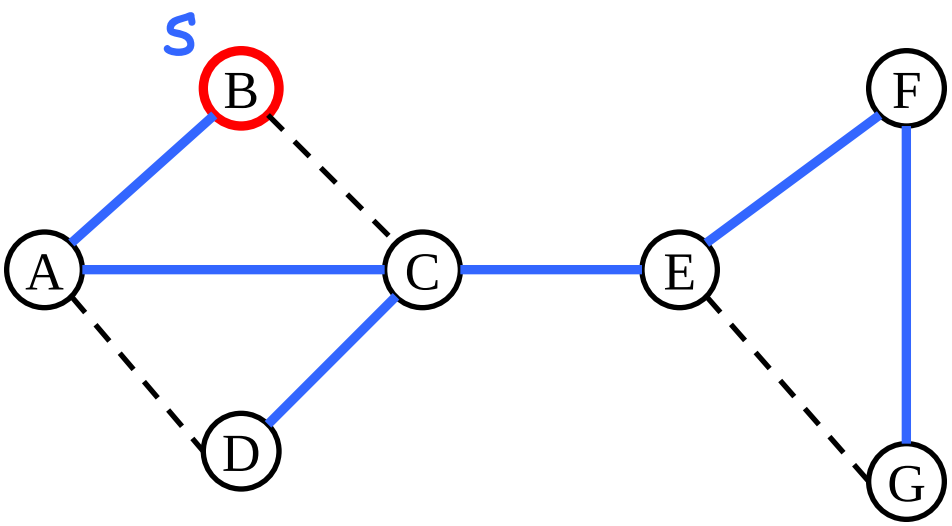
Un esempio: visita DFS



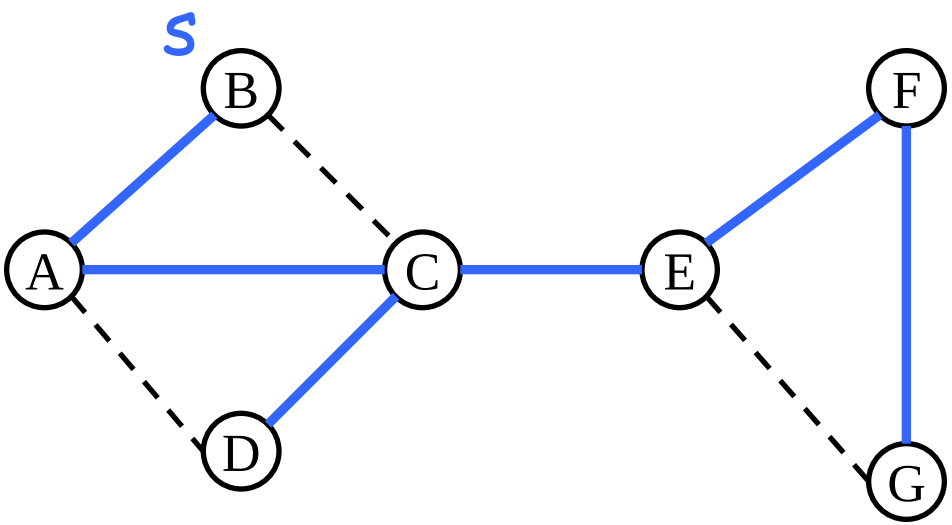
Un esempio: visita DFS



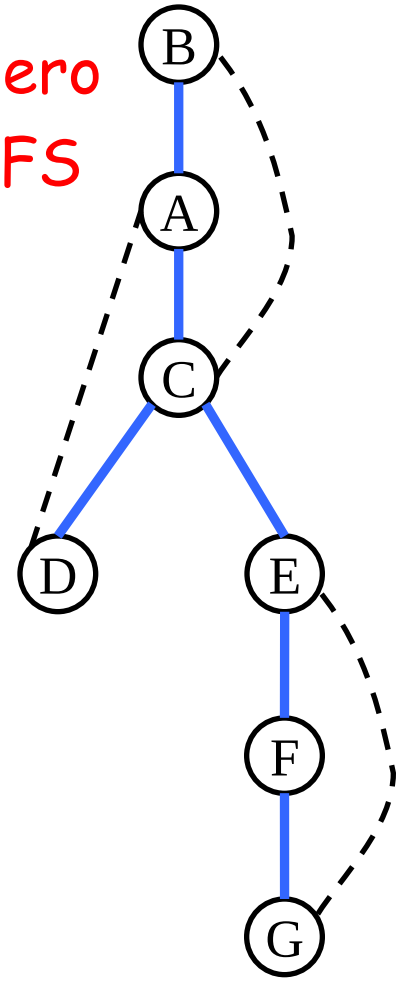
Un esempio: visita DFS



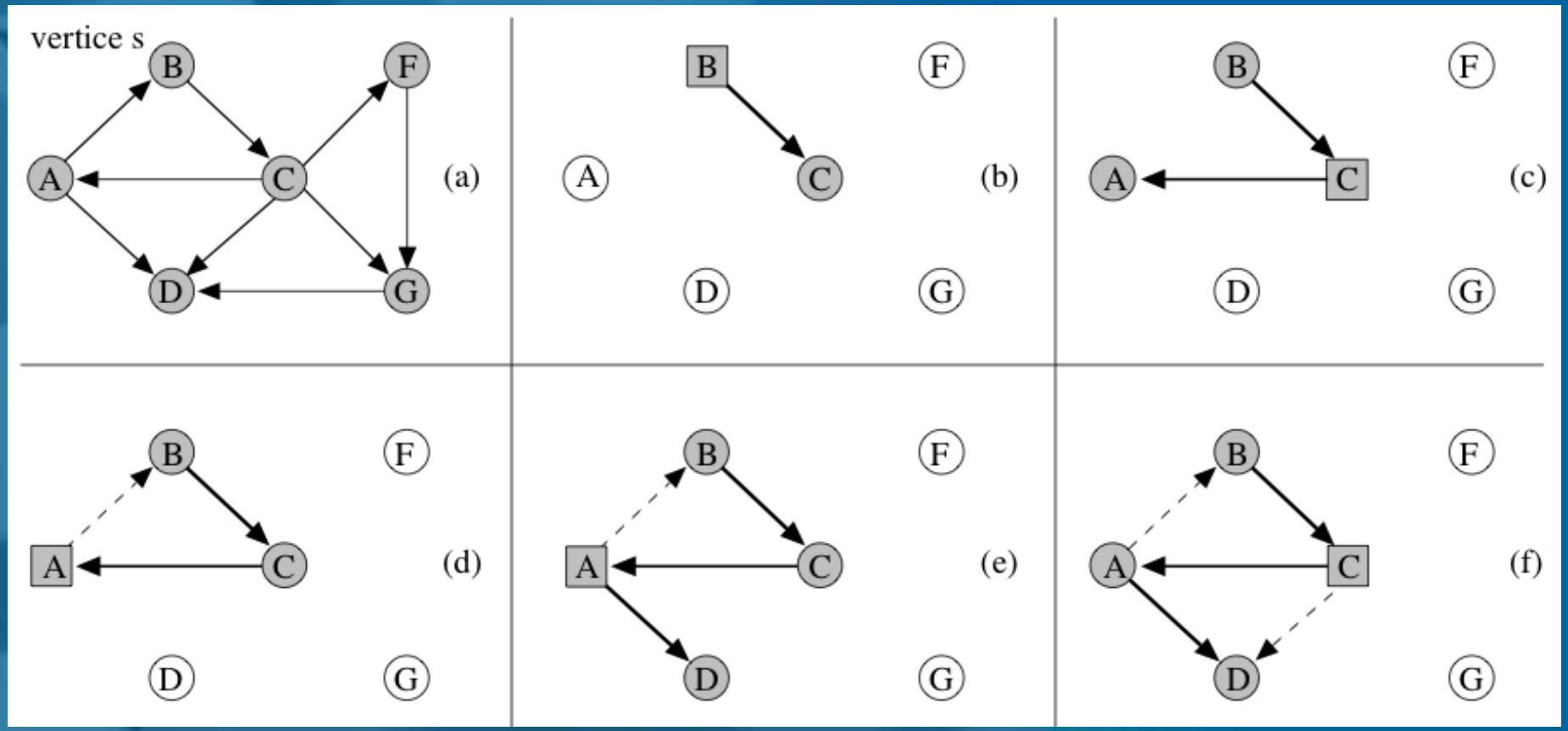
Un esempio: visita DFS



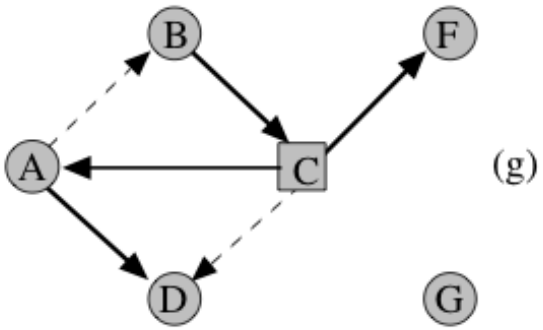
albero DFS



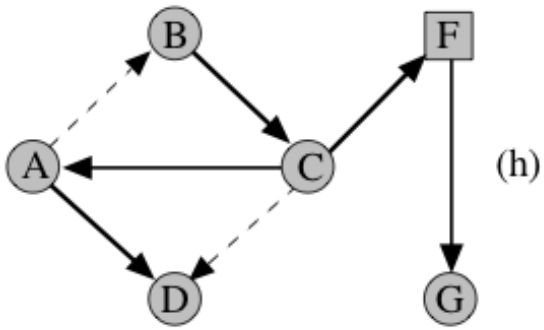
Esempio: grafo orientato (1/2)



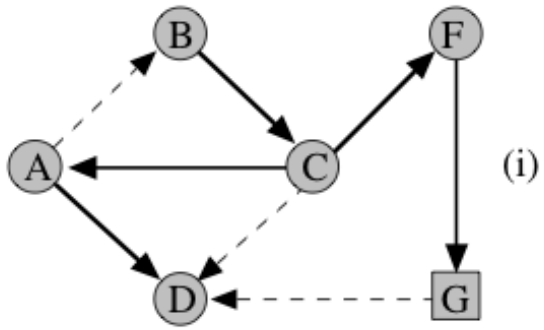
Esempio: grafo orientato (2/2)



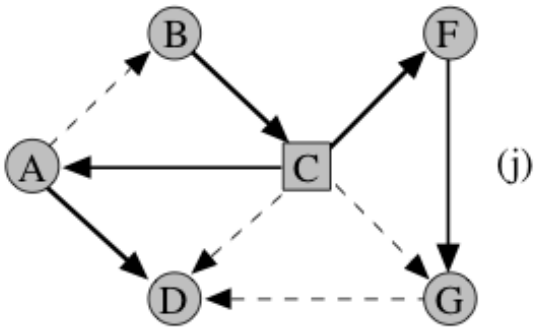
(g)



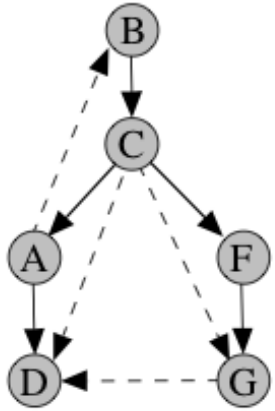
(h)



(i)



(j)

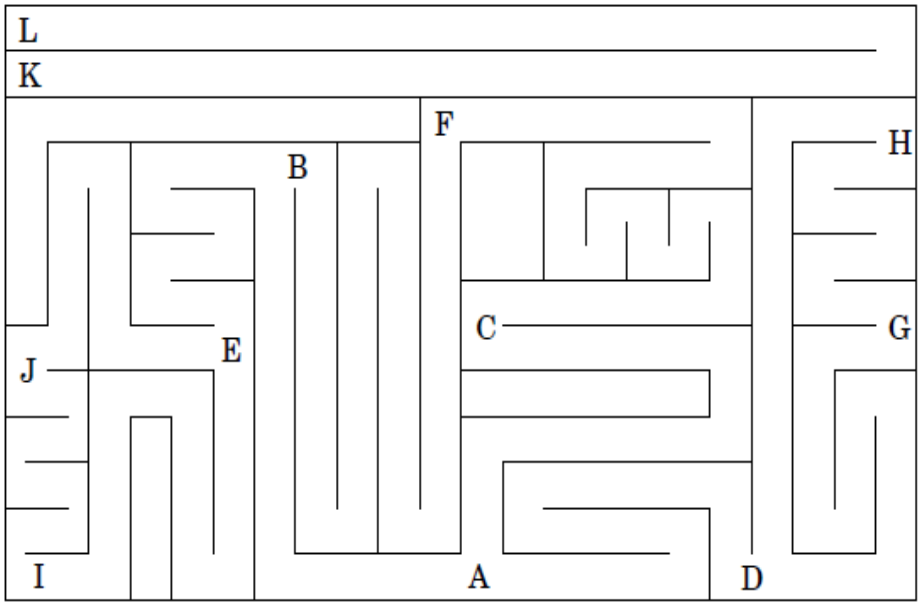


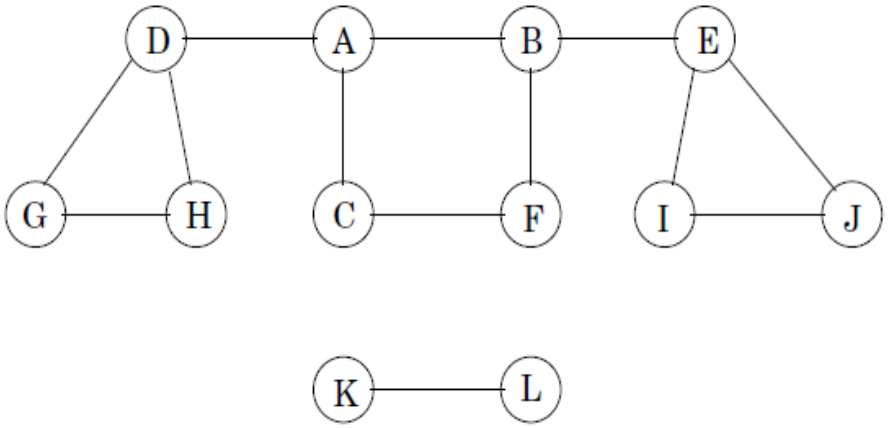
archi in avanti: (C,D) e (C,G)

archi all'indietro: (A,B)

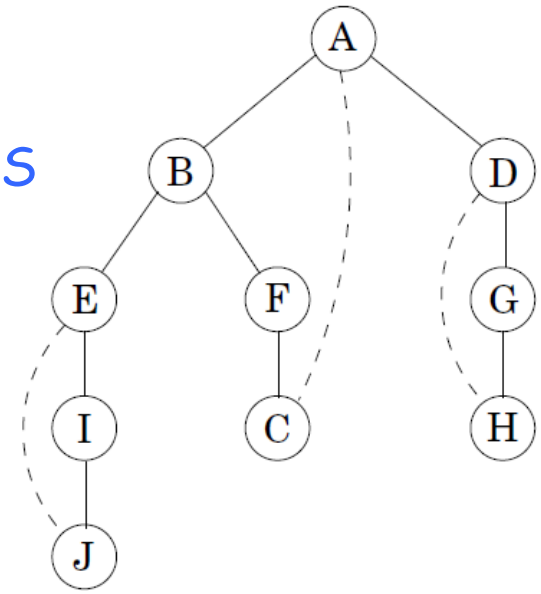
archi trasversali a sinistra: (G,D)

...tornando al labirinto





albero DFS



Costo della visita in profondità

Il tempo di esecuzione dipende dalla struttura dati usata per rappresentare il grafo (e dalla connettività o meno del grafo rispetto ad s):

- Liste di adiacenza: $O(m+n)$
- Matrice di adiacenza: $O(n^2)$

Proprietà dell'albero DFS radicato in s

- Se il grafo è non orientato, per ogni arco (u,v) si ha:
  - (u,v) è un arco dell'albero DFS, oppure
  - i nodi u e v sono l'uno discendente/antenato dell'altro
- Se il grafo è orientato, per ogni arco (u,v) si ha:
  - (u,v) è un arco dell'albero DFS, oppure
  - i nodi u e v sono l'uno discendente/antenato dell'altro, oppure
  - (u,v) è un arco trasversale a sinistra, ovvero il vertice v è in un sottoalbero visitato precedentemente ad u