


Algoritmi e Strutture Dati

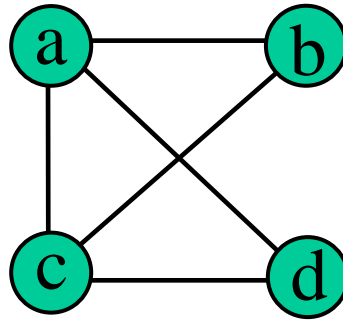
Capitolo 11 Visite di grafi



Strutture dati per rappresentare grafi

Grafi non diretti

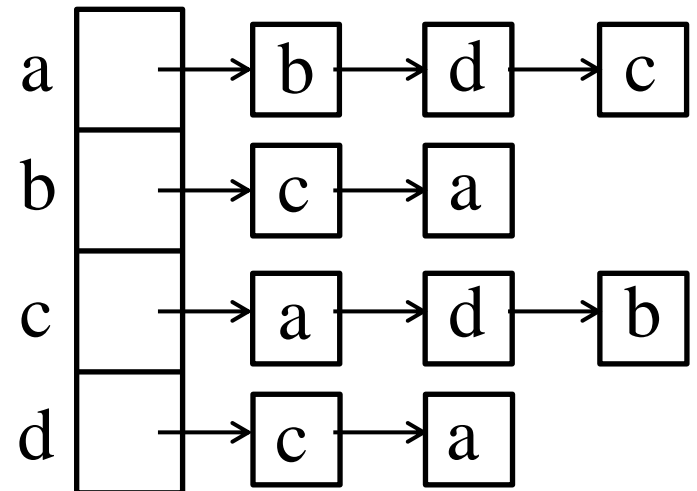
Quanto spazio?



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

Matrice di adiacenza

$O(n^2)$

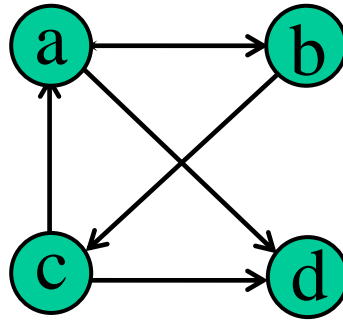


liste di adiacenza

$O(m + n)$

Grafi diretti

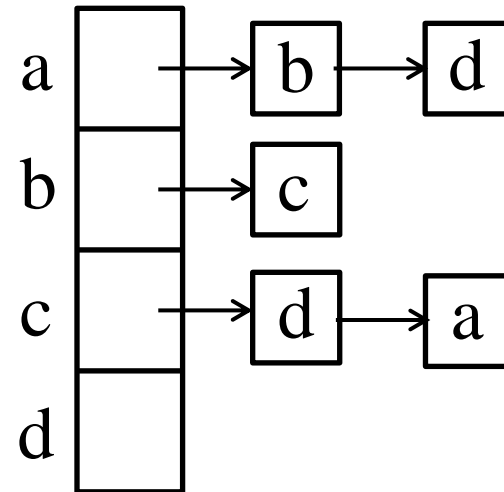
Quanto spazio?



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

Matrice di adiacenza

$O(n^2)$



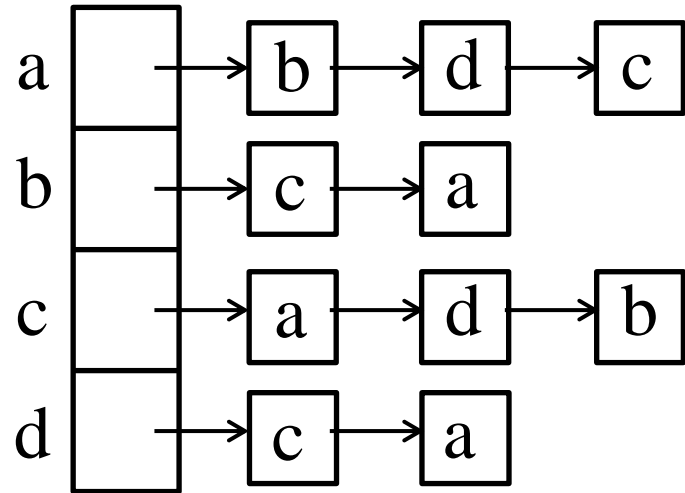
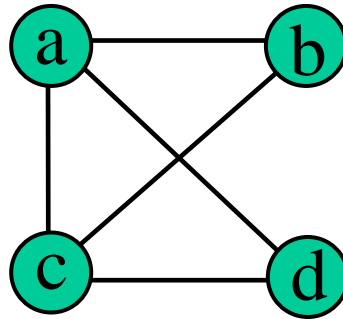
liste di adiacenza

$O(m + n)$

Grafi non diretti

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

Matrice di adiacenza



liste di adiacenza

Operazione:

elenco archi
incidenti in v

c'è arco (u,v) ?

matrice di a.

$O(n)$

$O(1)$

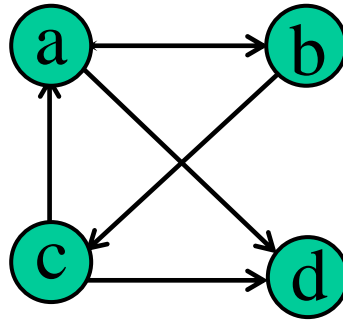
liste di a.

$O(\delta(v))$

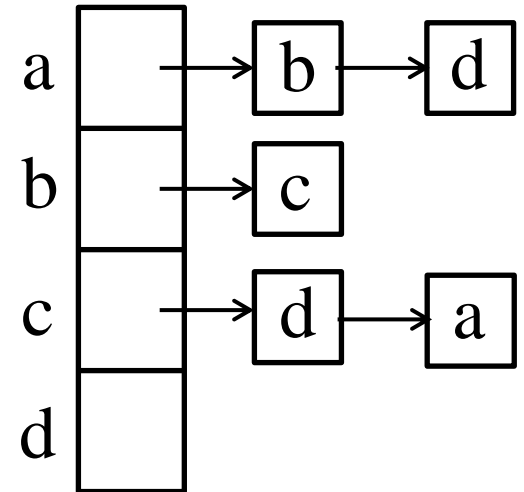
$O(\min\{\delta(u), \delta(v)\})$

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

Matrice di adiacenza



Grafi diretti



liste di adiacenza

Operazione:

elenco archi
uscenti da v

c'è arco (u,v)?

matrice di a.

$O(n)$

$O(1)$

liste di a.

$O(\delta(v))$

$O(\delta(u))$

*quali parti del grafo
sono raggiungibili
da un certo
nodo?*

...eseguo una *visita* del grafo

Scopo e tipi di visita

- una visita di un grafo G permette di esaminare i nodi e gli archi di G **in modo sistematico** (se G è connesso)
- genera un **albero** di visita
- problema di base in molte applicazioni
- esistono vari tipi di visite con diverse proprietà:
 - **visita in ampiezza (BFS=breadth first search)**
 - **visita in profondità (DFS=depth first search)**

Visita in ampiezza

dato un grafo G (non pesato) e un nodo s , trova tutte le distanze/cammini minimi da s verso ogni altro nodo v

applicazioni

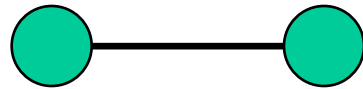
- web crawling
 - come google trova nuove pagine da indicizzare
- social networking
 - trovare gli amici che potresti conoscere
- network broadcast
 - un nodo manda un messaggio a tutti gli altri nodi della rete
- garbage collection
 - come scoprire memoria non più raggiungibile che si può liberare
- 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

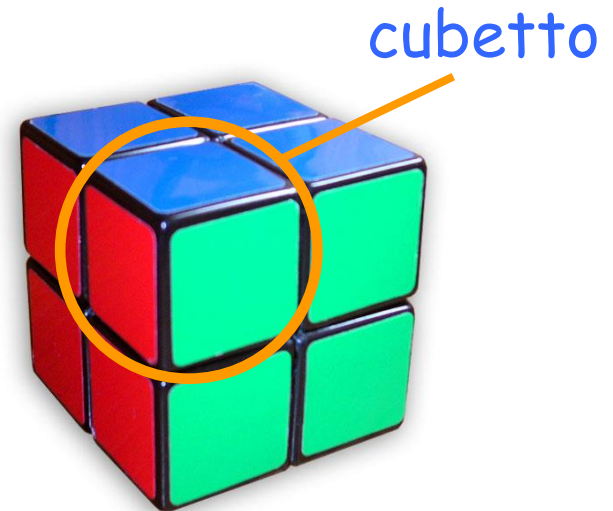
- un vertice per ogni possibile stato del cubo
- un arco fra due configurazioni se l'una è ottenibile dall'altra tramite una mossa



grafo non diretto

$$\# \text{verciti} \leq 8! \times 3^8$$

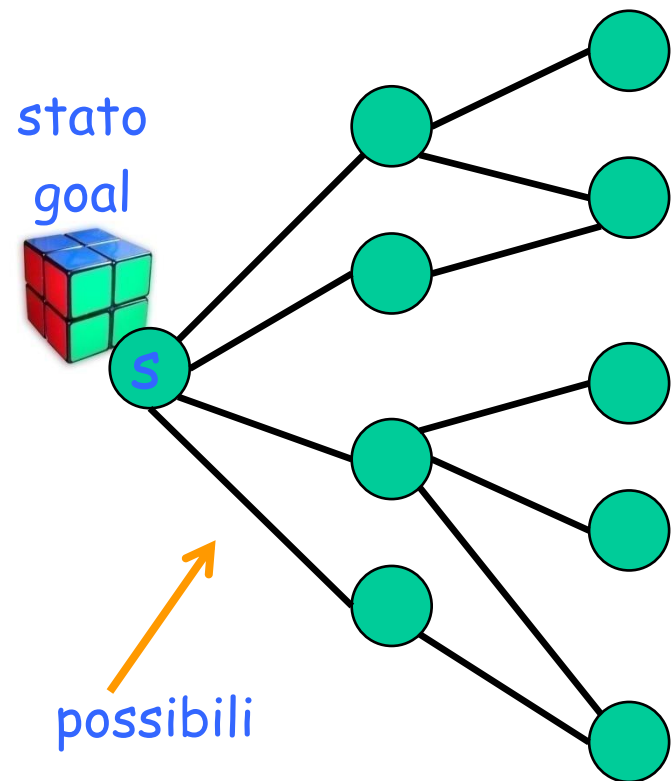
$$= 264.539.520$$



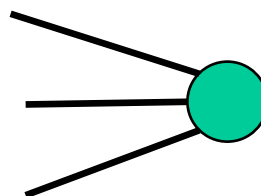
cubo di Rubik: 2x2x2

eccentricità di s (God's number)

God's number



...

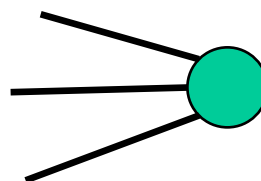


2x2x2: 11

3x3x3: 20

4x4x4: ???

...



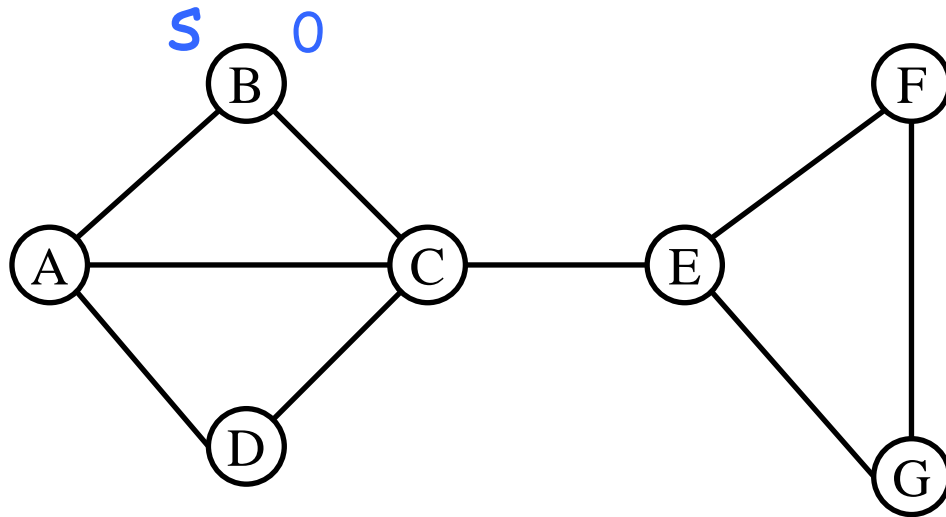
$n \times n \times n: \Theta(n^2 / \log n)$

Visita in ampiezza

algoritmo visitaBFS(*vertice* s) \rightarrow *albero*

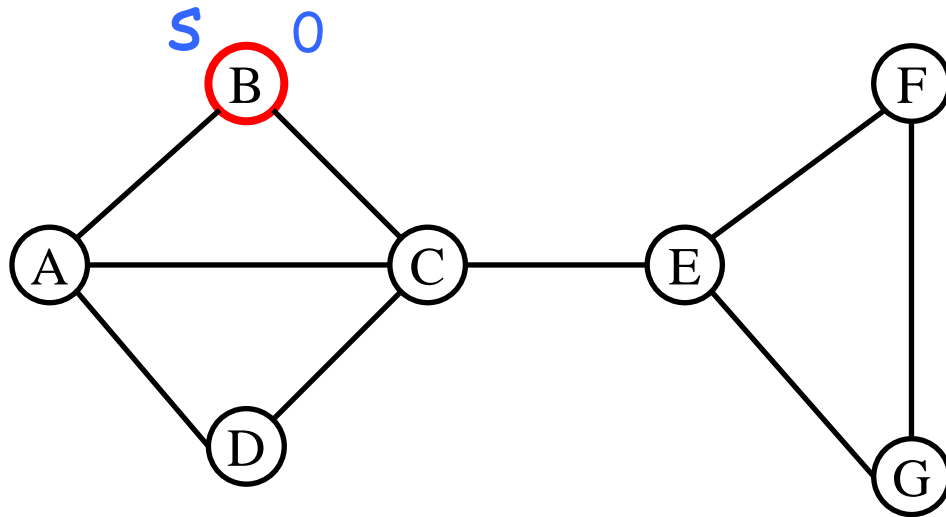
1. rendi tutti i vertici non marcati
2. $T \leftarrow$ albero formato da un solo nodo s
3. Coda F
4. marca il vertice s
5. $F.enqueue(s)$
6. **while** (**not** $F.isEmpty()$) **do**
7. $u \leftarrow F.dequeue()$
8. **for each** (arco (u, v) in G) **do**
9. **if** (v non è ancora marcato) **then**
10.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

Un esempio

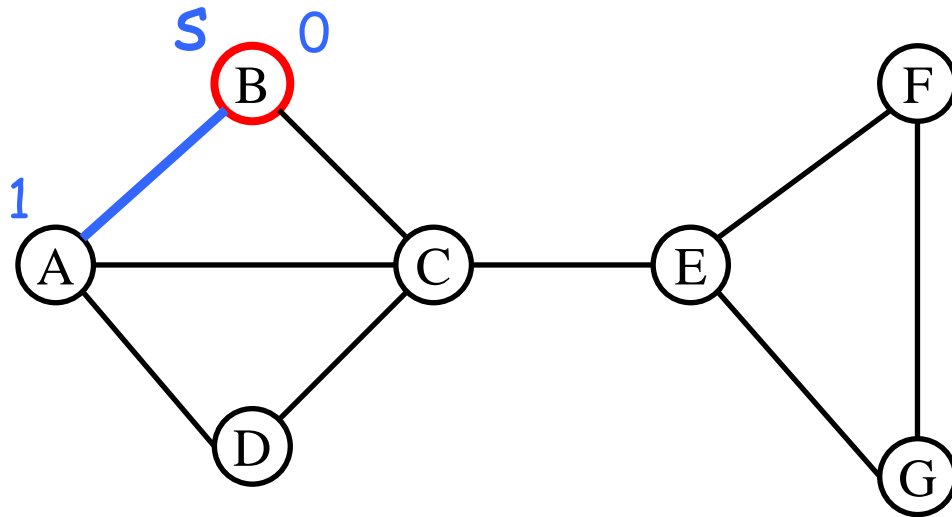


→
B

Un esempio

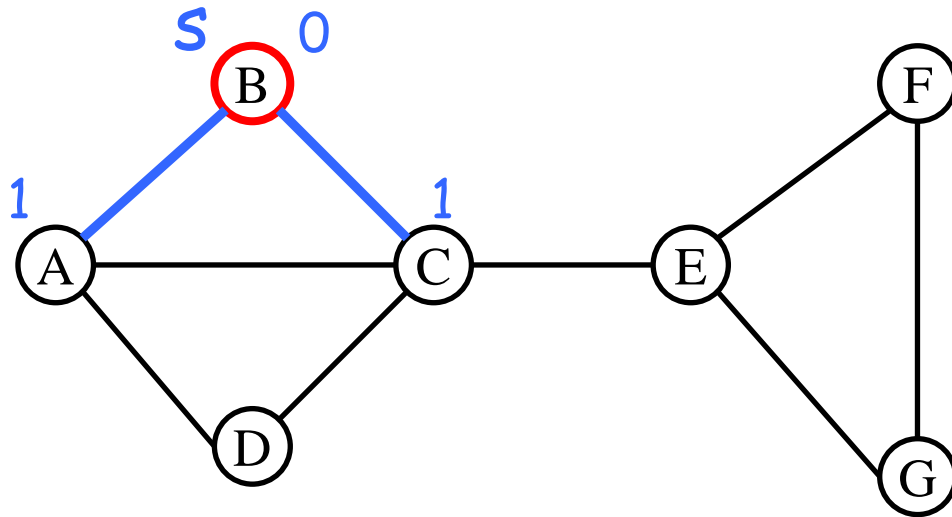


Un esempio



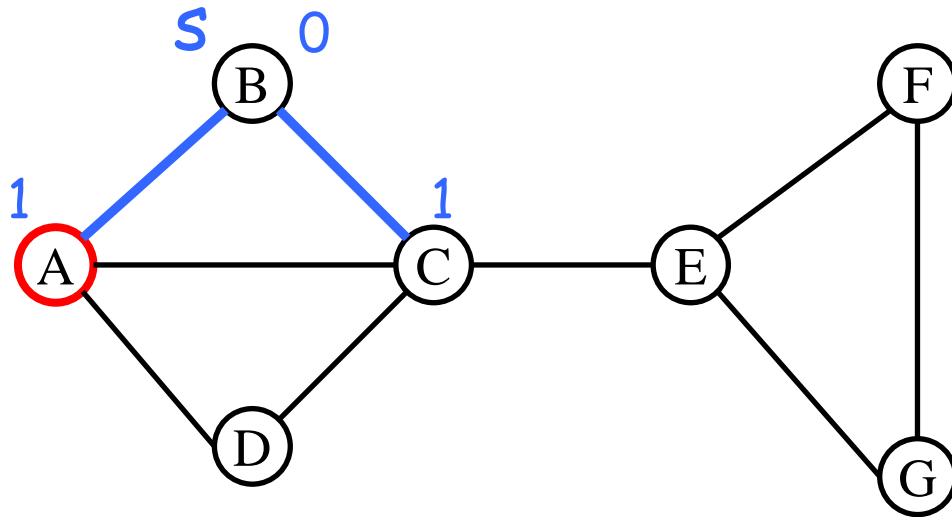
A

Un esempio



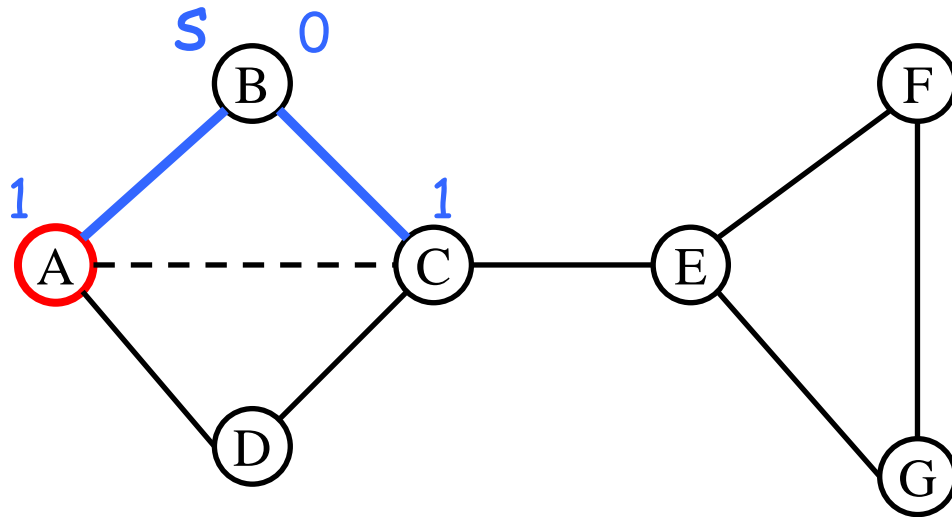
C A

Un esempio



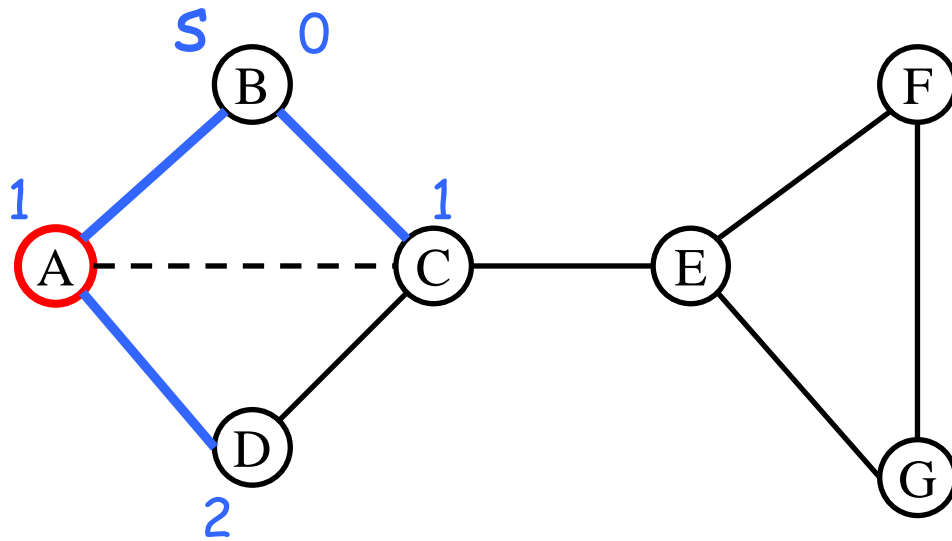
$\xrightarrow{\quad}$
C

Un esempio



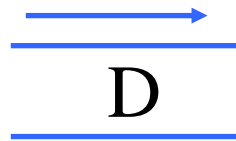
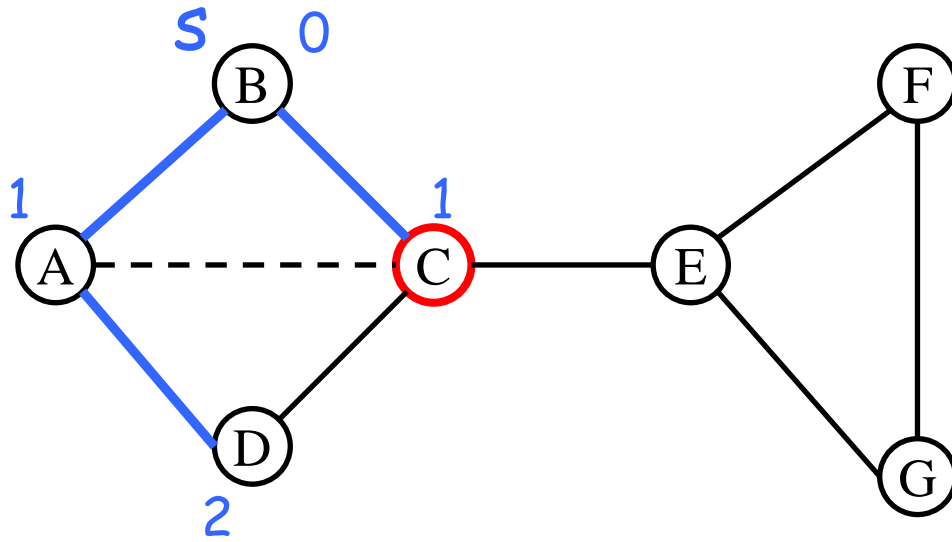
→
C

Un esempio

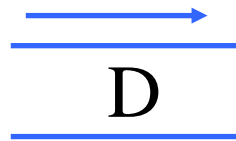
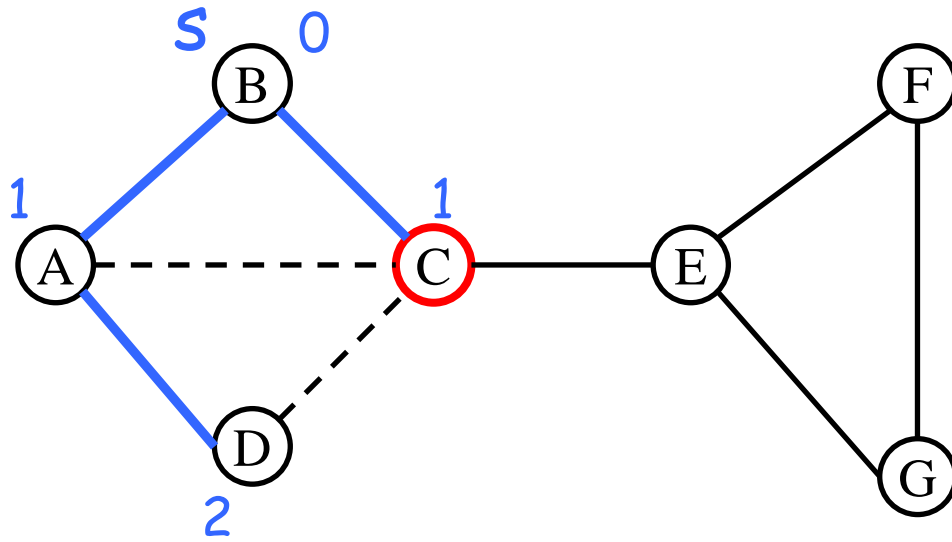


D C

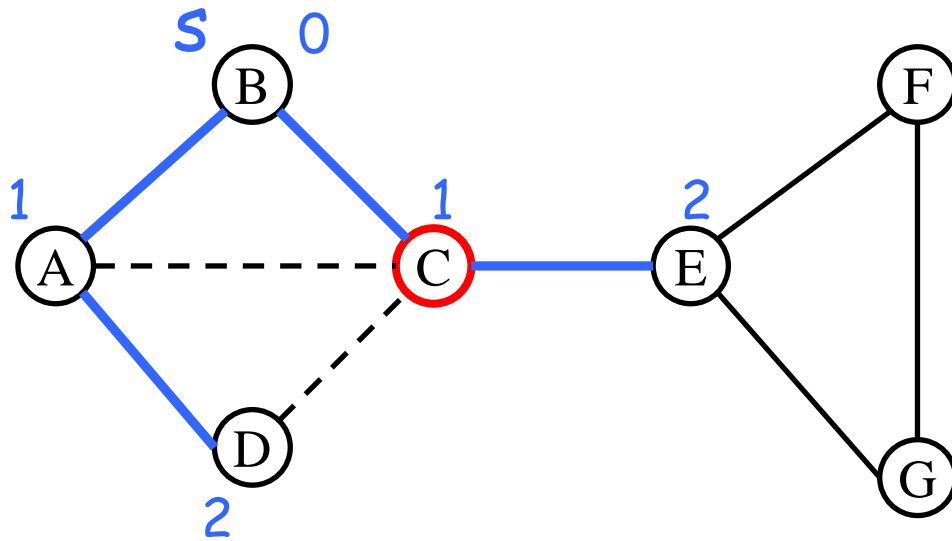
Un esempio



Un esempio

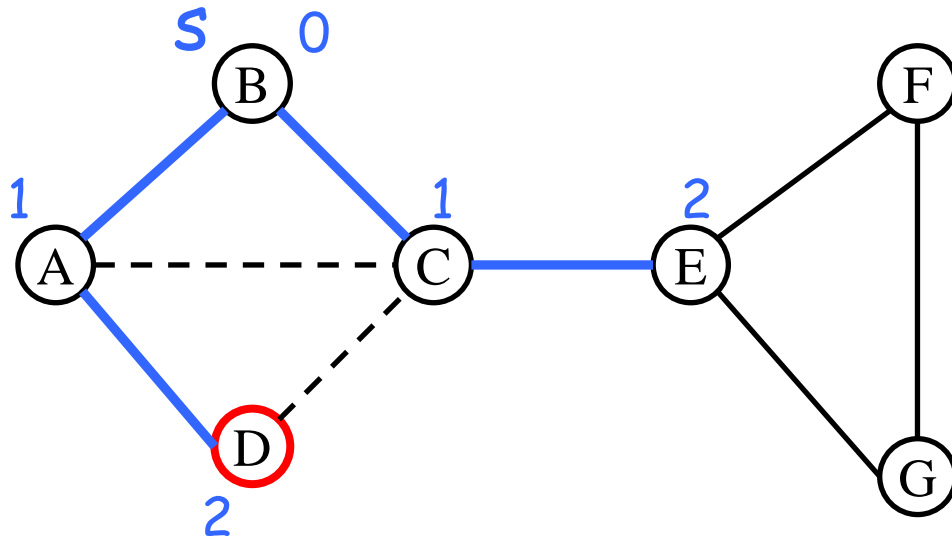


Un esempio

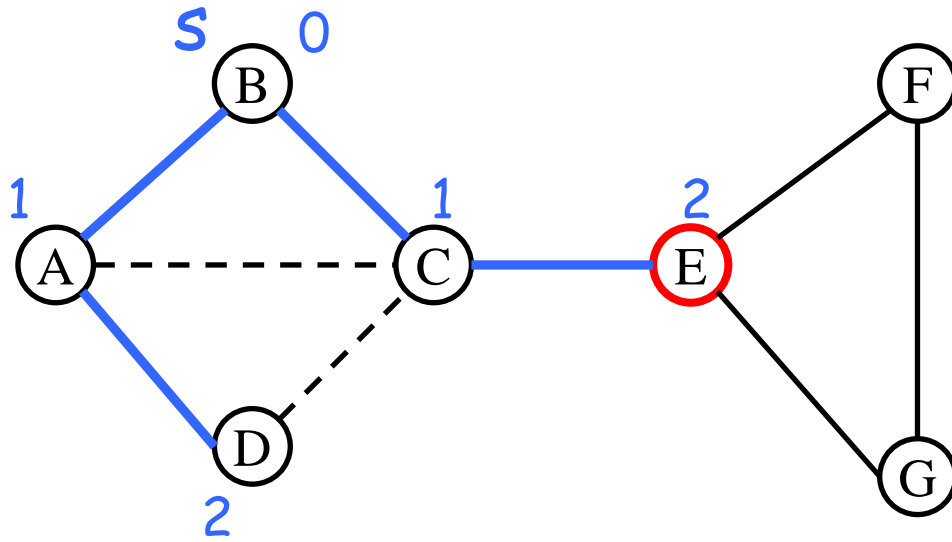


→
E D

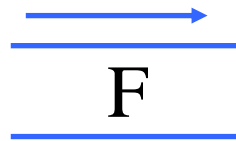
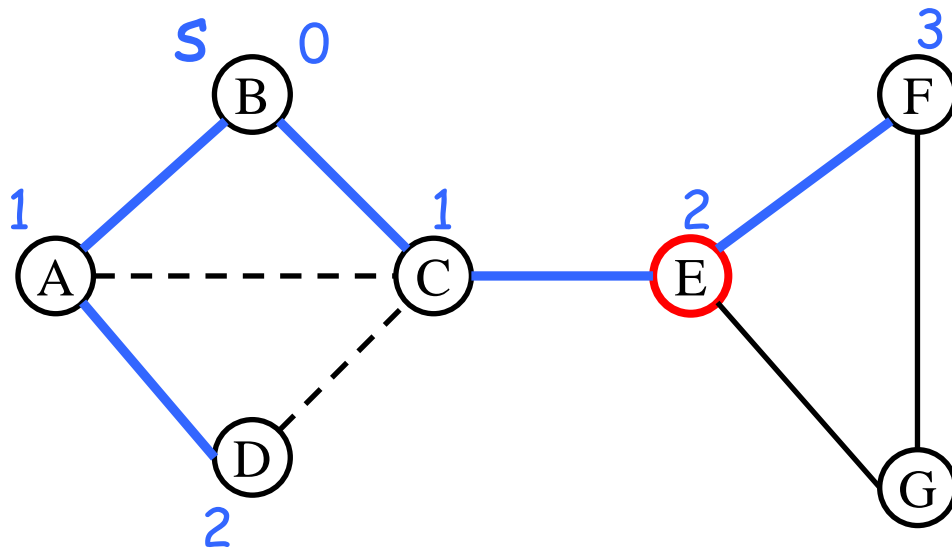
Un esempio



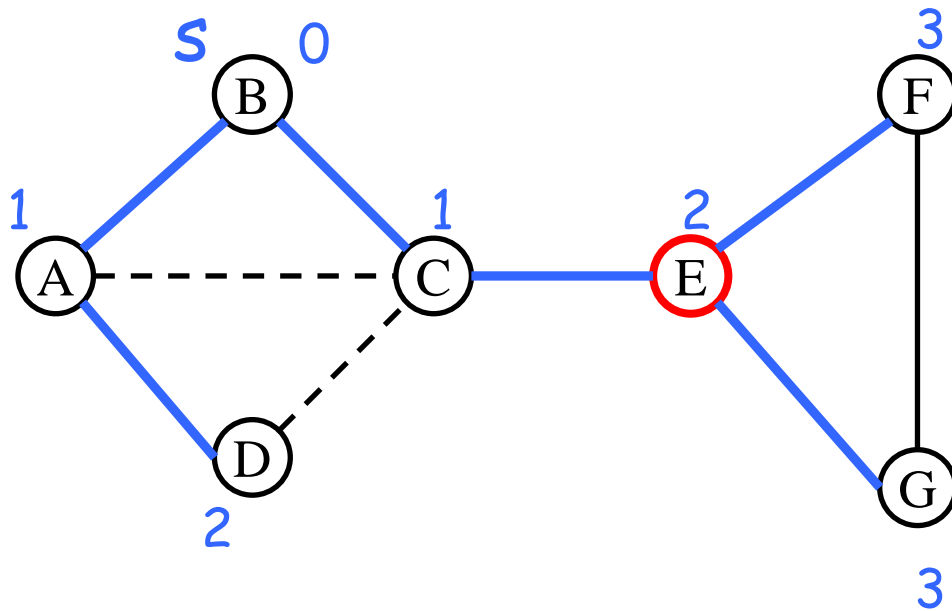
Un esempio



Un esempio

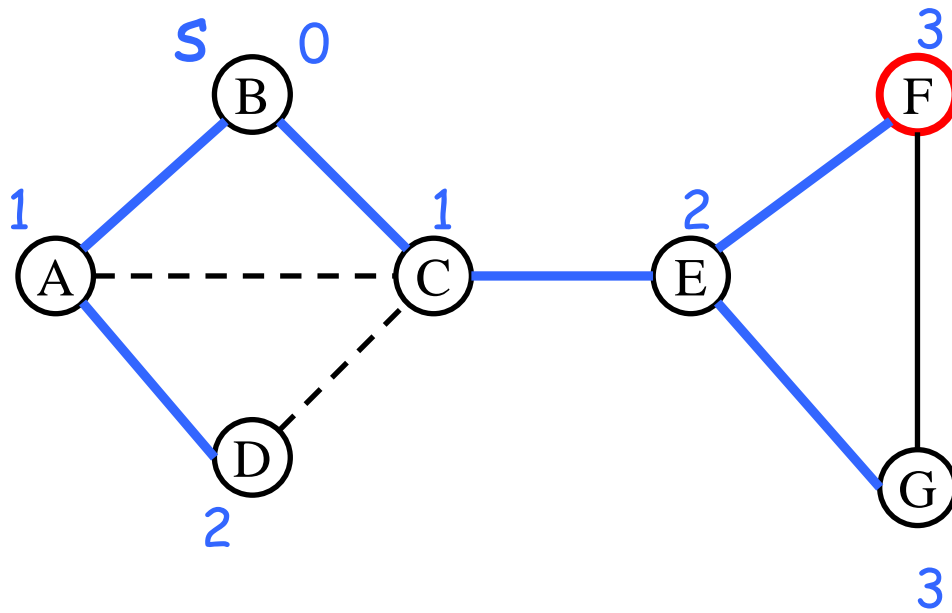


Un esempio

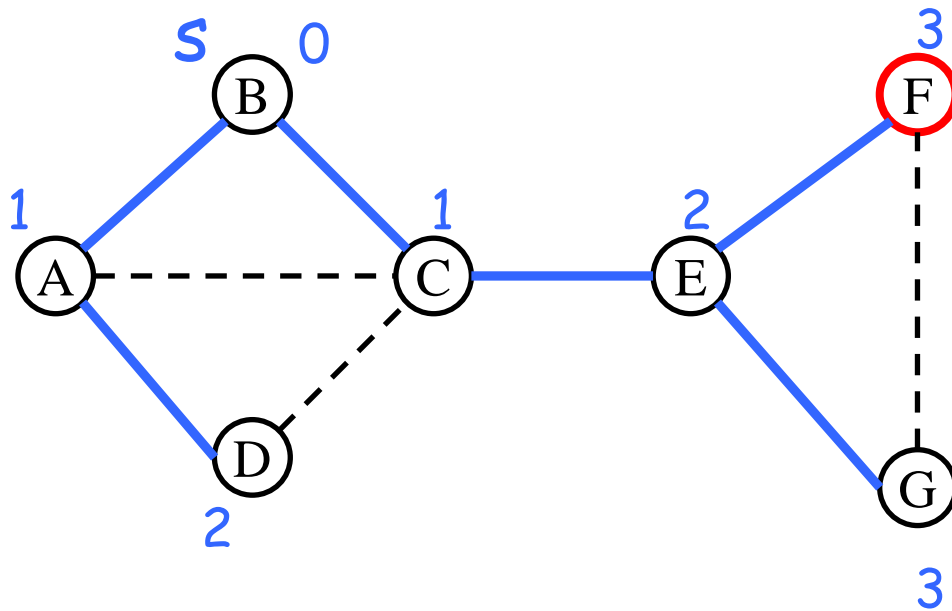


$\xrightarrow{\quad}$
GF

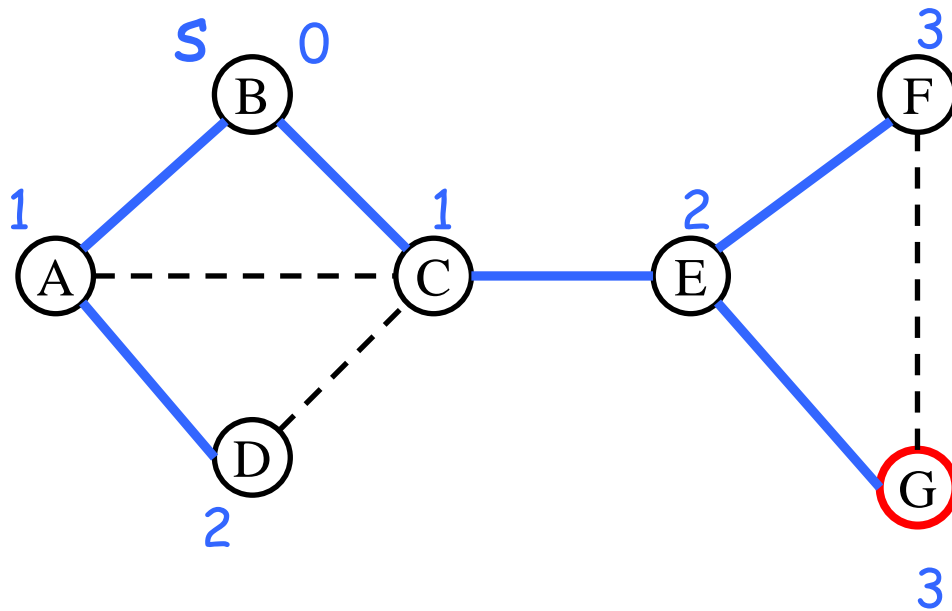
Un esempio



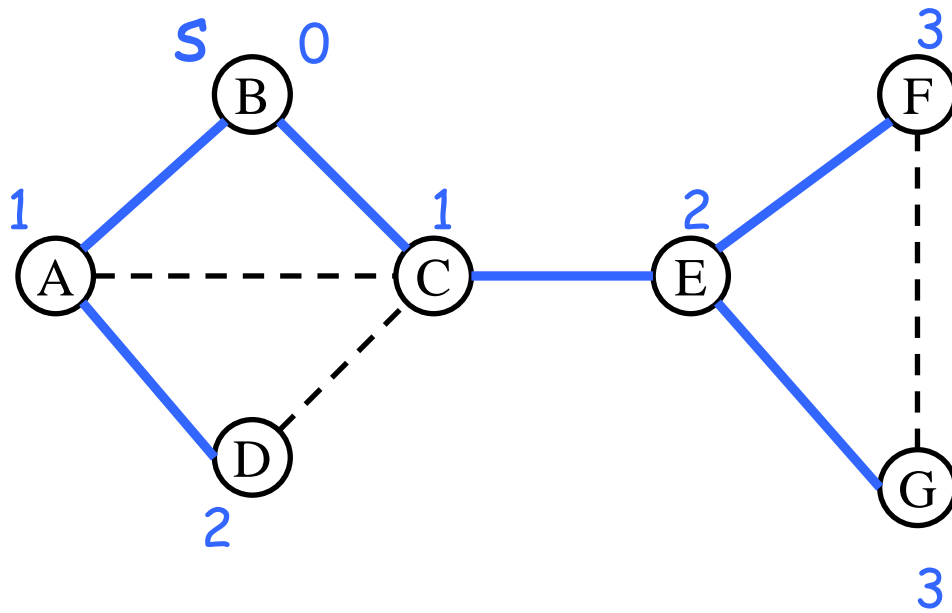
Un esempio



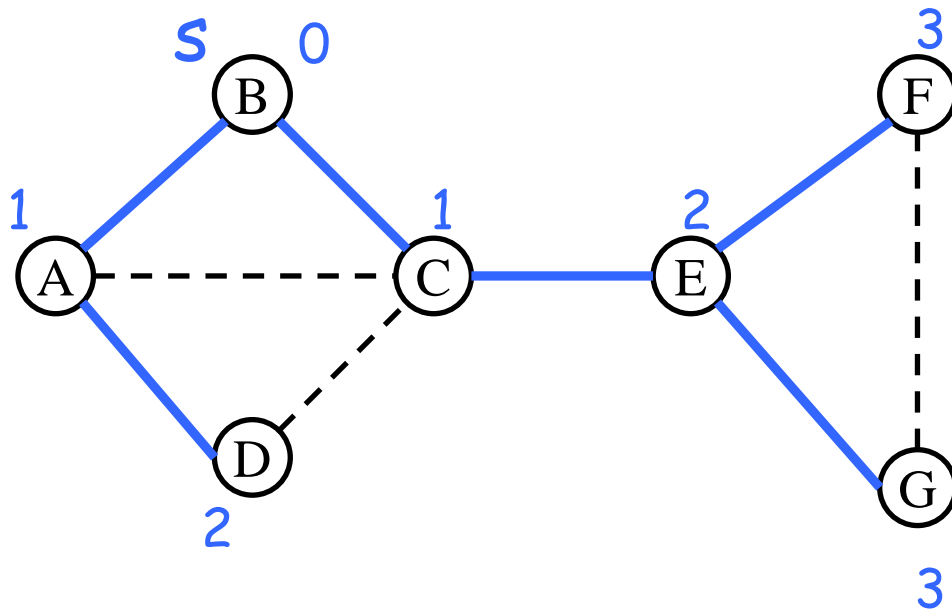
Un esempio



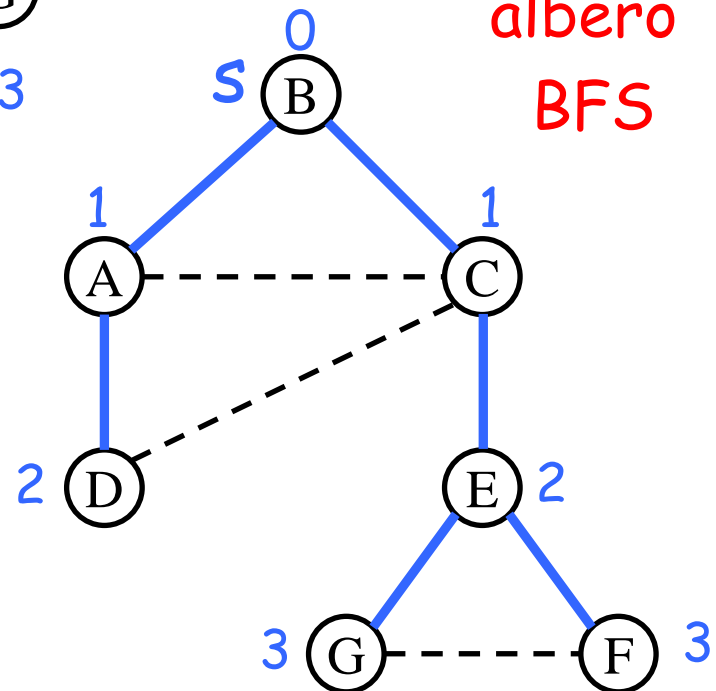
Un esempio



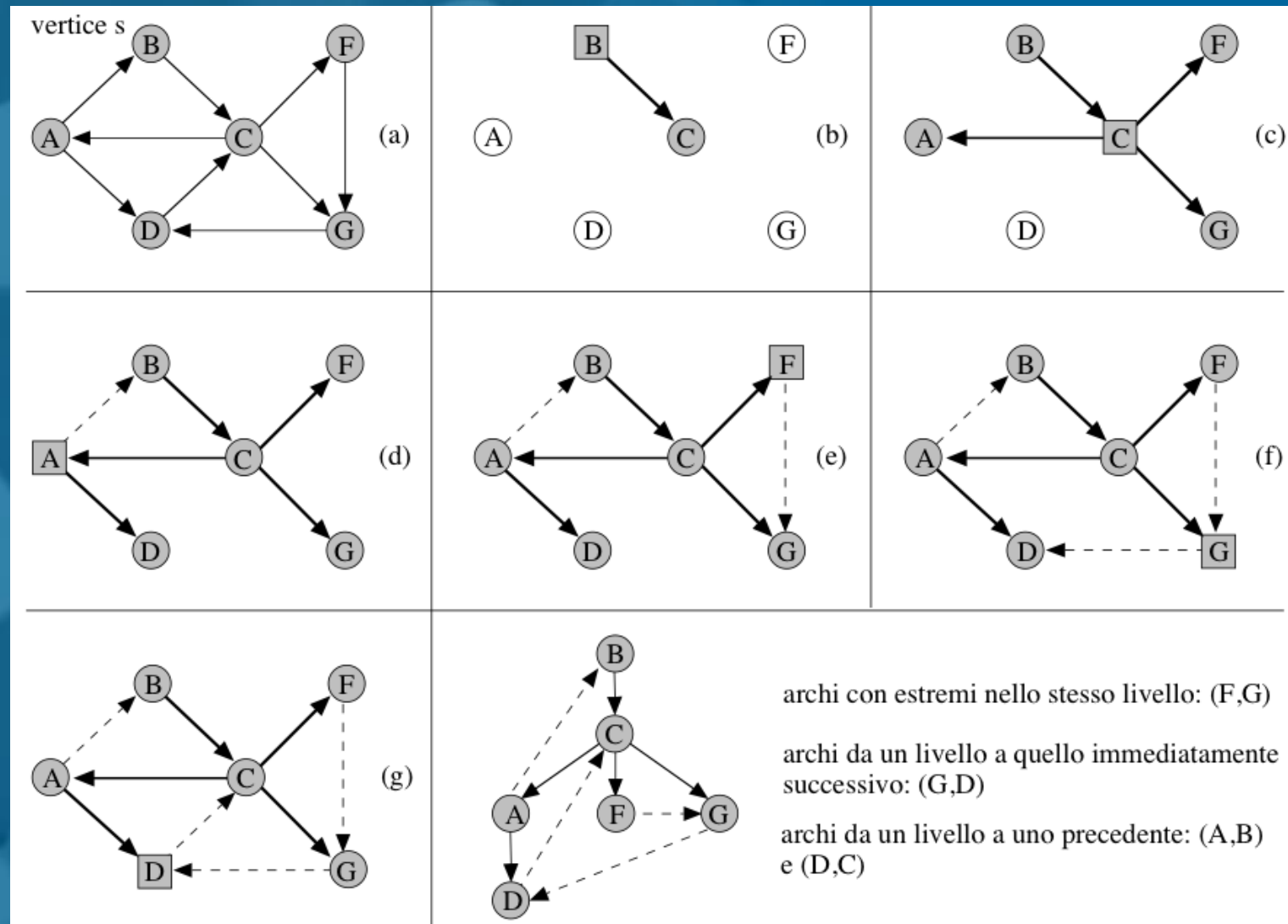
Un esempio



albero dei
cammini di G
radicato in s



Esempio: grafo orientato



Costo della visita in ampiezza

grafo rappresentato con matrice di adiacenza

algoritmo visitaBFS(*vertice* s) \rightarrow *albero*

1. rendi tutti i vertici non marcati
2. $T \leftarrow$ 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. $u \leftarrow F.dequeue()$
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) \rightarrow *albero*

1. rendi tutti i vertici non marcati
2. $T \leftarrow$ 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. $u \leftarrow F.dequeue()$
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!

Teorema

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)

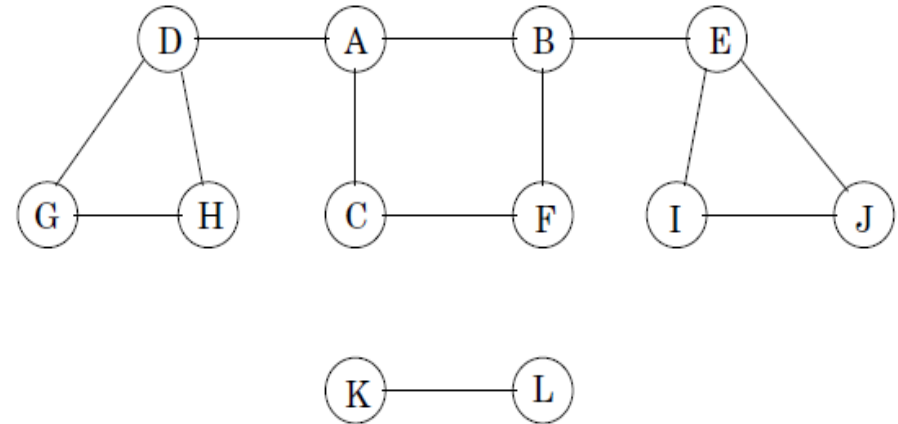
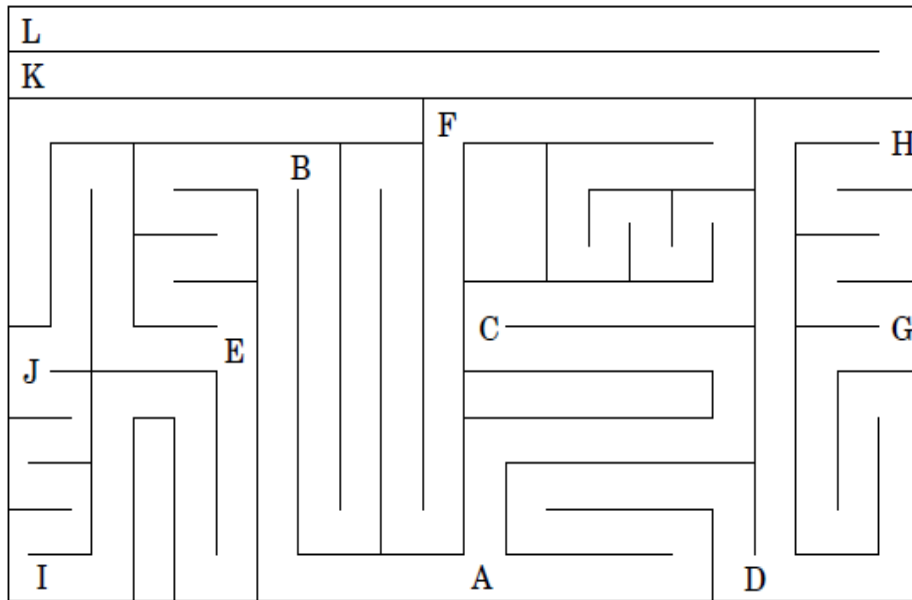
dimostrazione informale

- 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 (archi uscenti); 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 (archi uscenti); 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 (archi uscenti); 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



corda: per
tornare
indietro se
necessario



variabile booleana:
dice se un nodo è stato
già visitato

pila: push vuol dire srotolare
pop vuol dire arrotolare

Visita in profondità

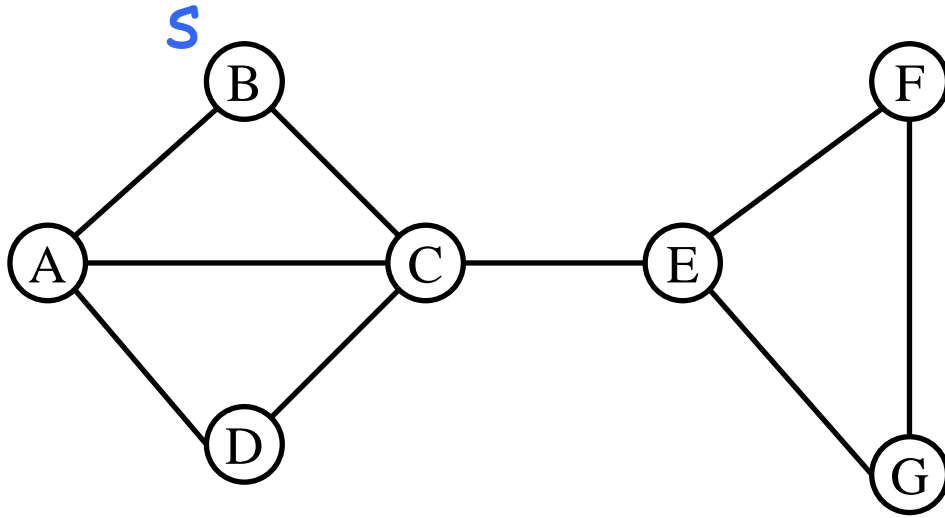
procedura visitaDFSRicorsiva(*vertice* v , *albero* T)

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)

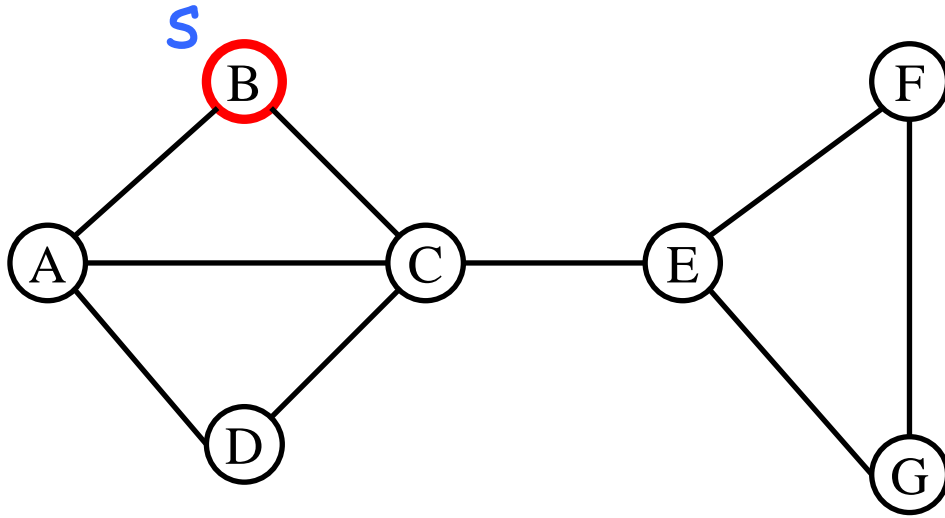
algoritmo visitaDFS(*vertice* s) \rightarrow *albero*

6. $T \leftarrow$ albero vuoto
7. visitaDFSRicorsiva(s, T)
8. **return** T

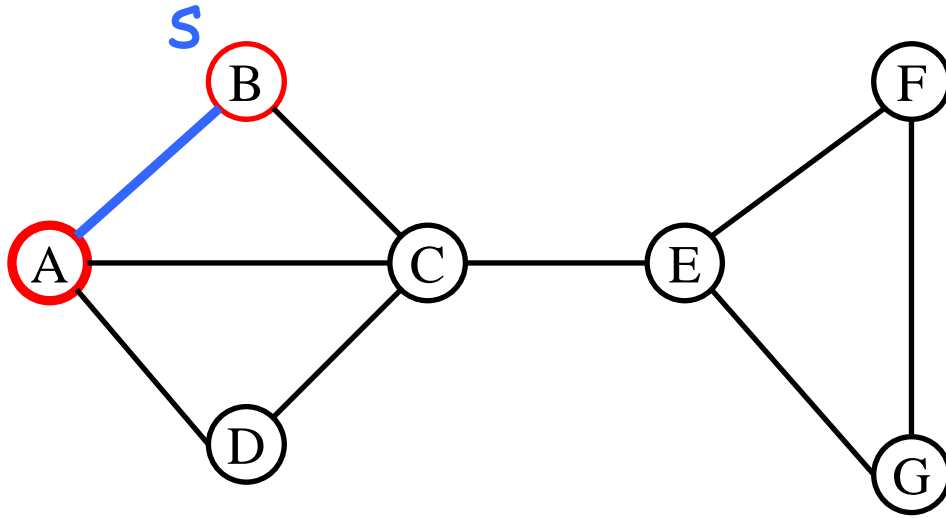
Un esempio: visita DFS



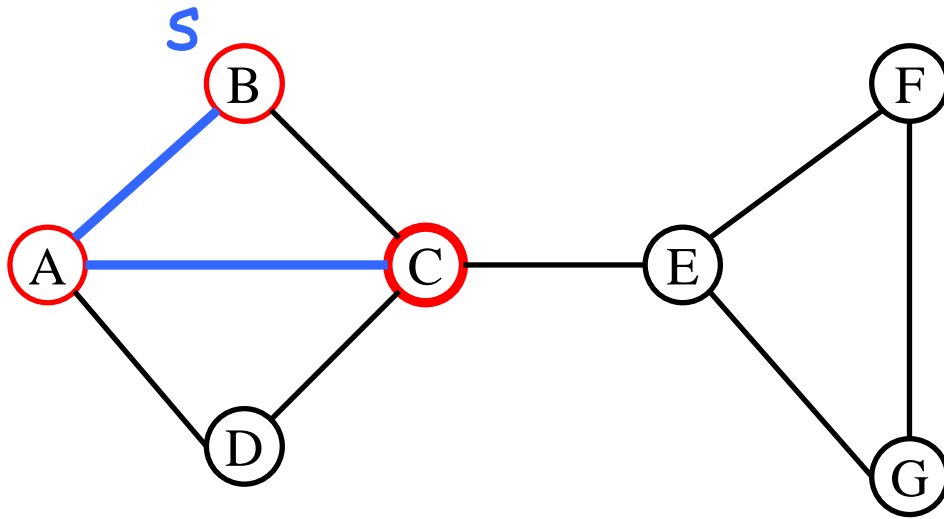
Un esempio: visita DFS



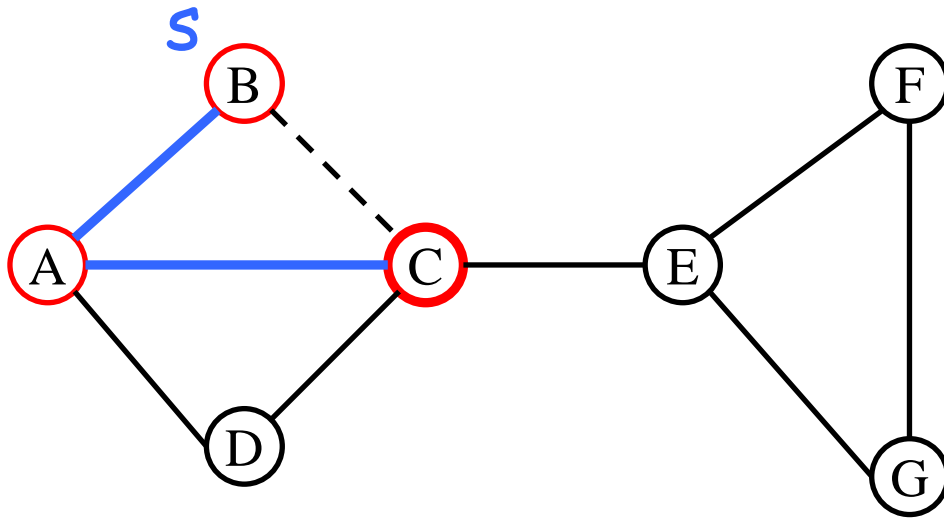
Un esempio: visita DFS



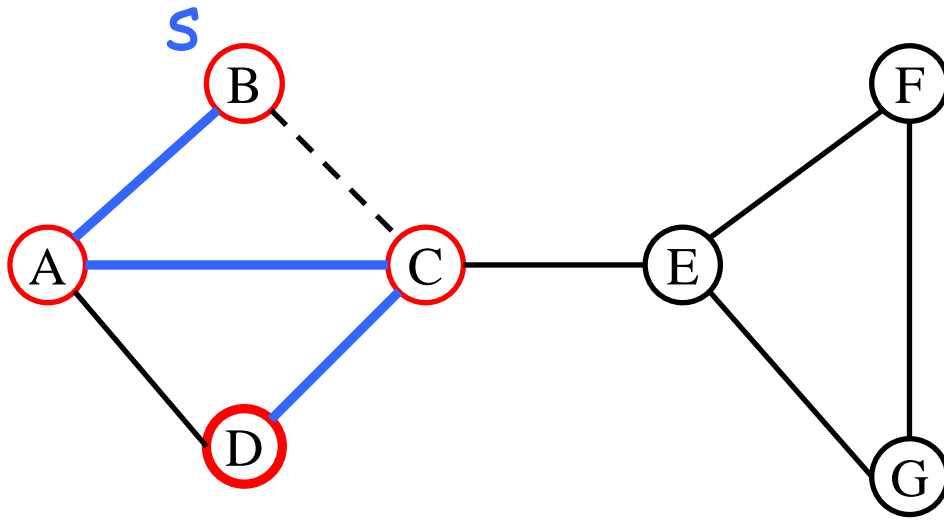
Un esempio: visita DFS



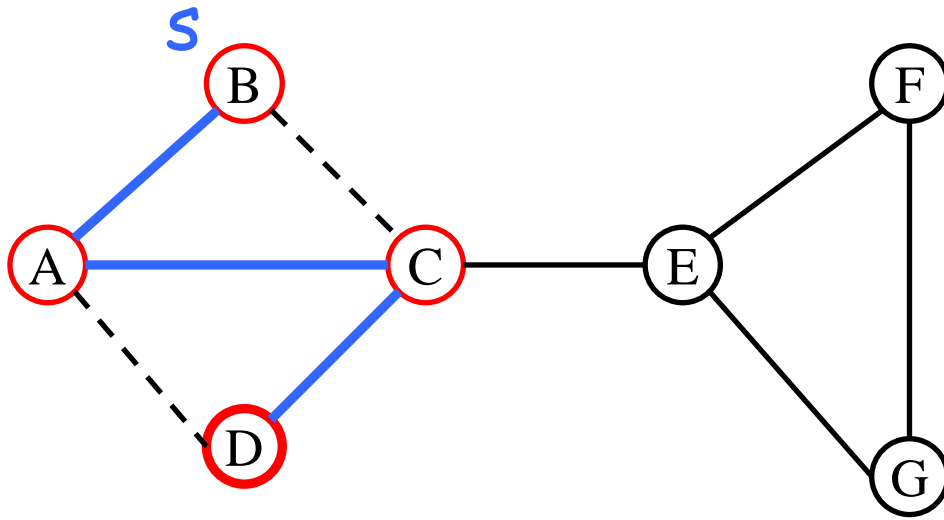
Un esempio: visita DFS



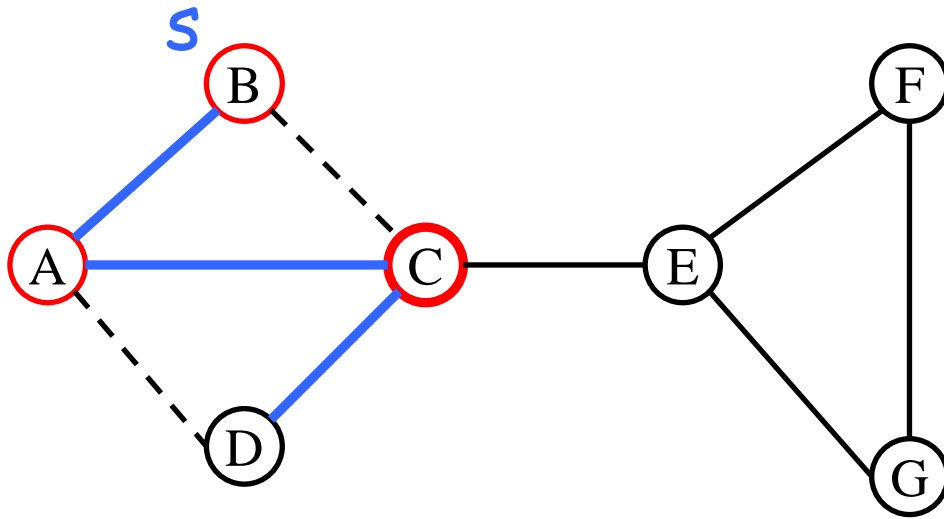
Un esempio: visita DFS



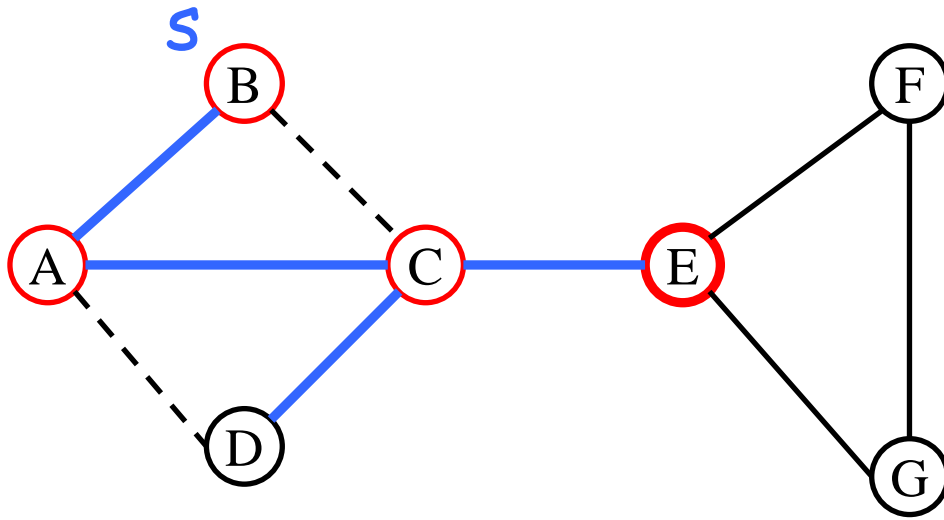
Un esempio: visita DFS



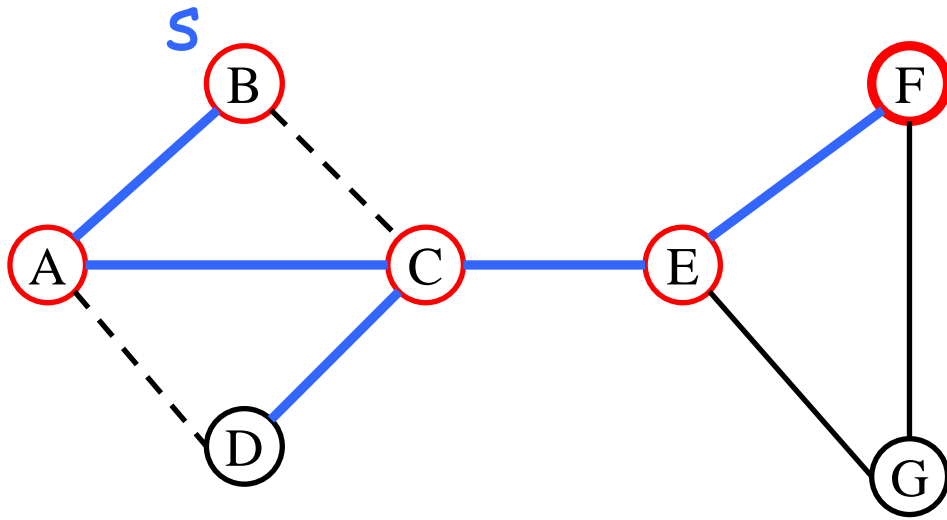
Un esempio: visita DFS



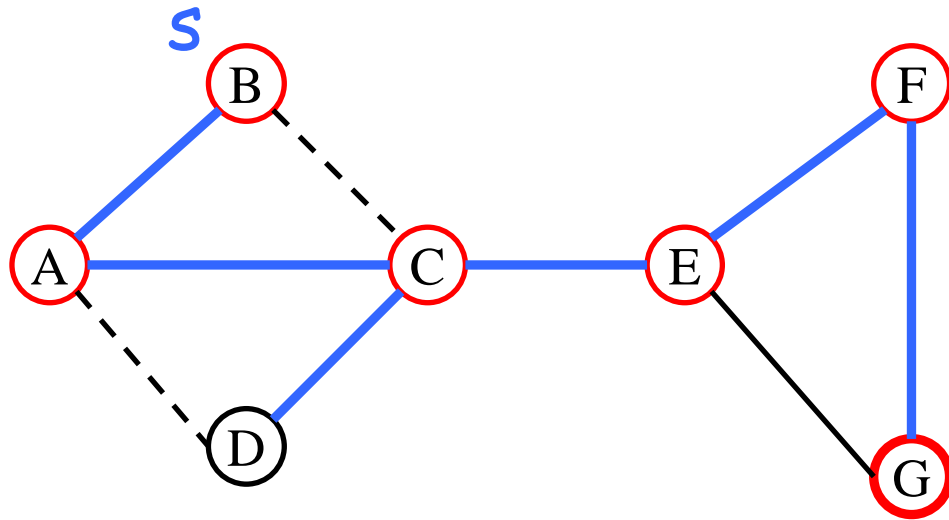
Un esempio: visita DFS



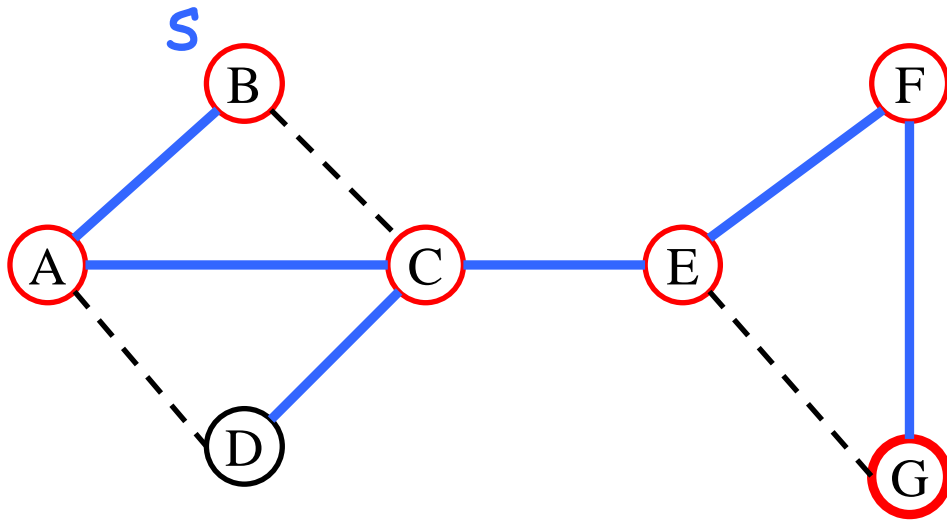
Un esempio: visita DFS



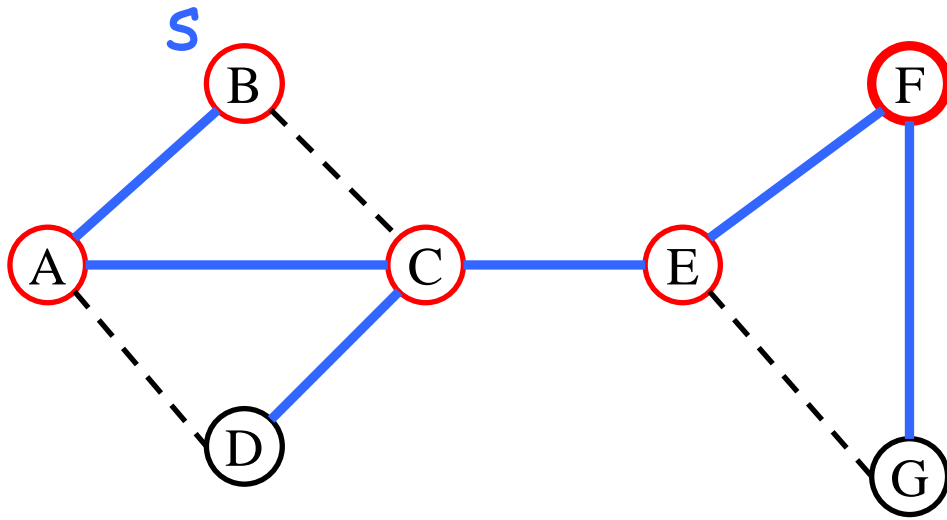
Un esempio: visita DFS



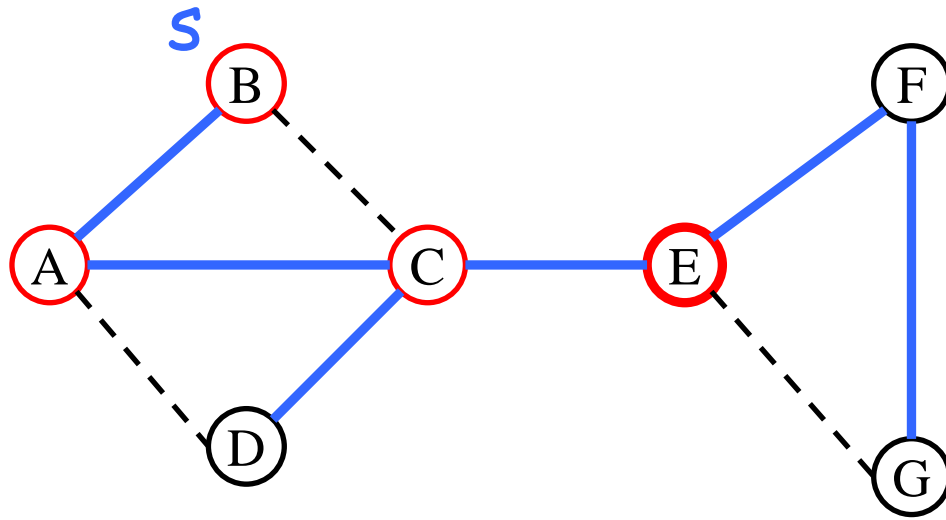
Un esempio: visita DFS



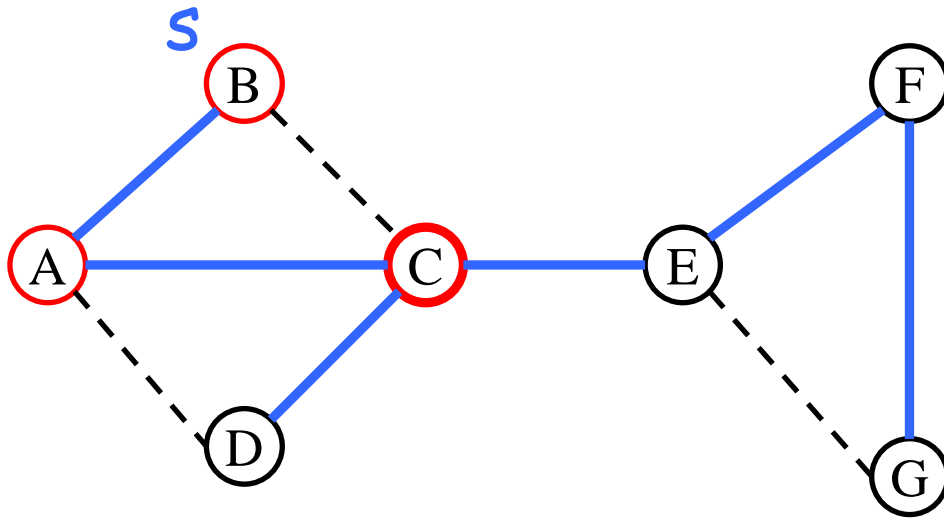
Un esempio: visita DFS



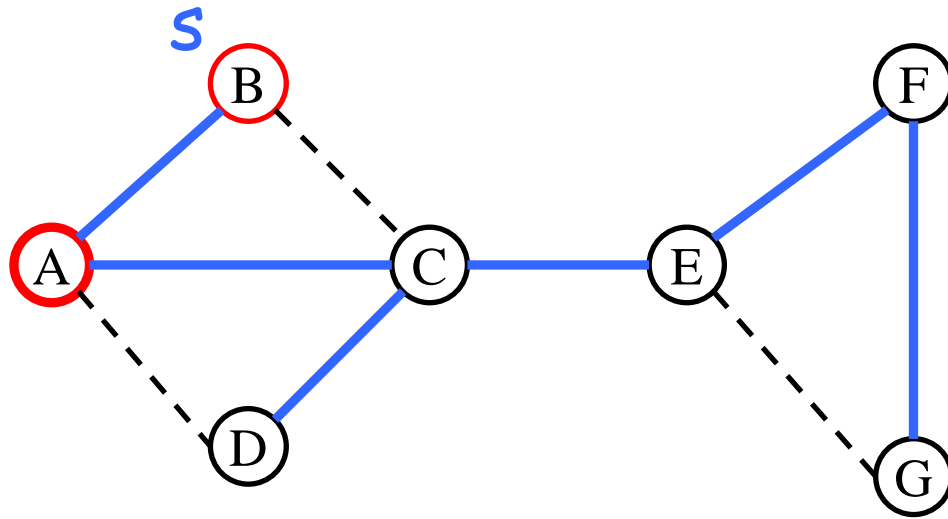
Un esempio: visita DFS



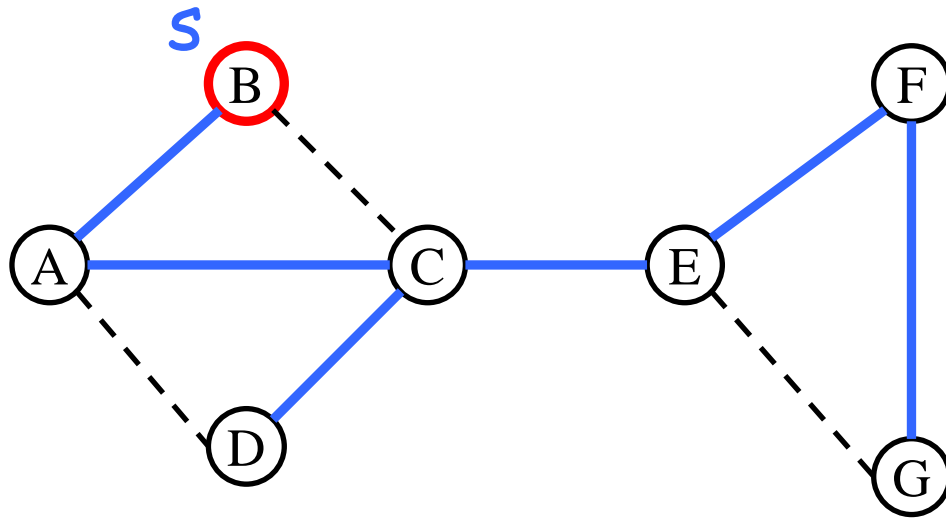
Un esempio: visita DFS



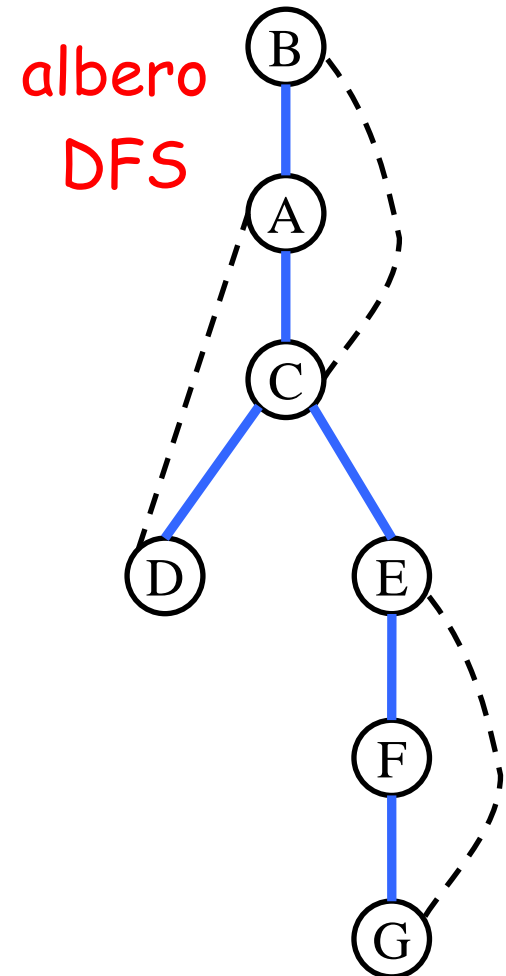
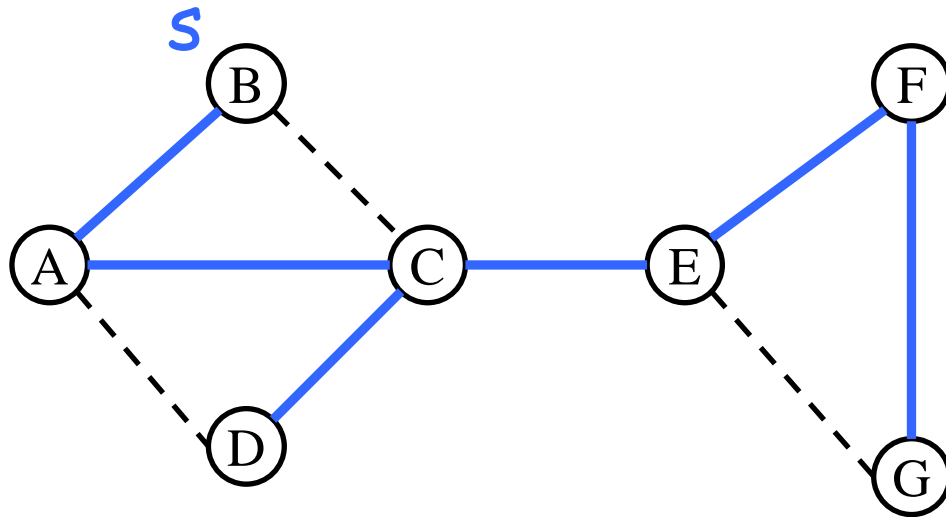
Un esempio: visita DFS



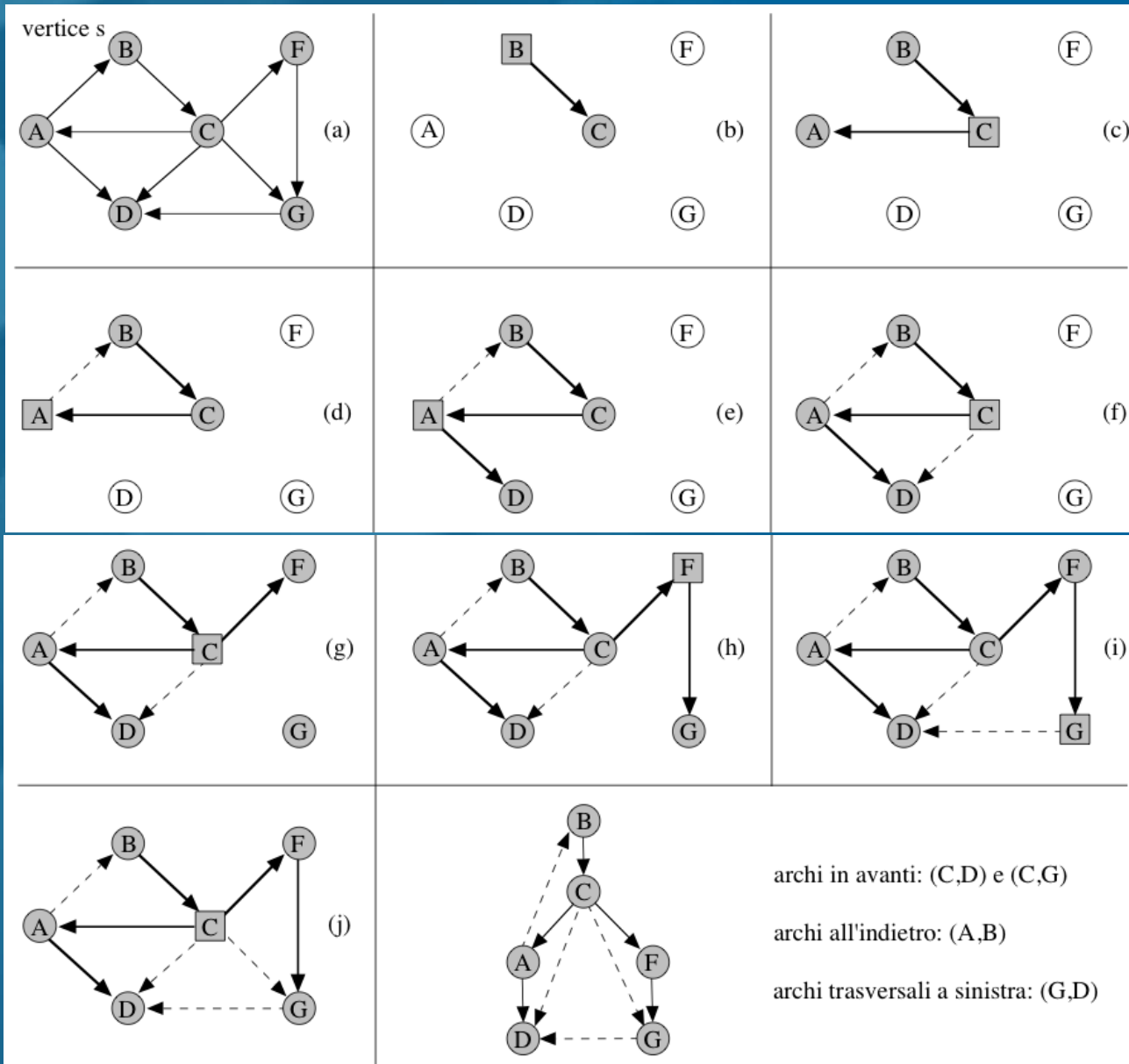
Un esempio: visita DFS



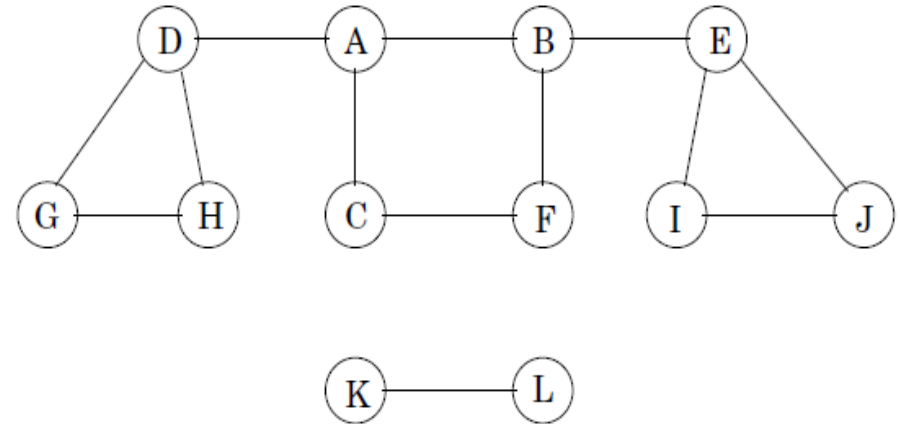
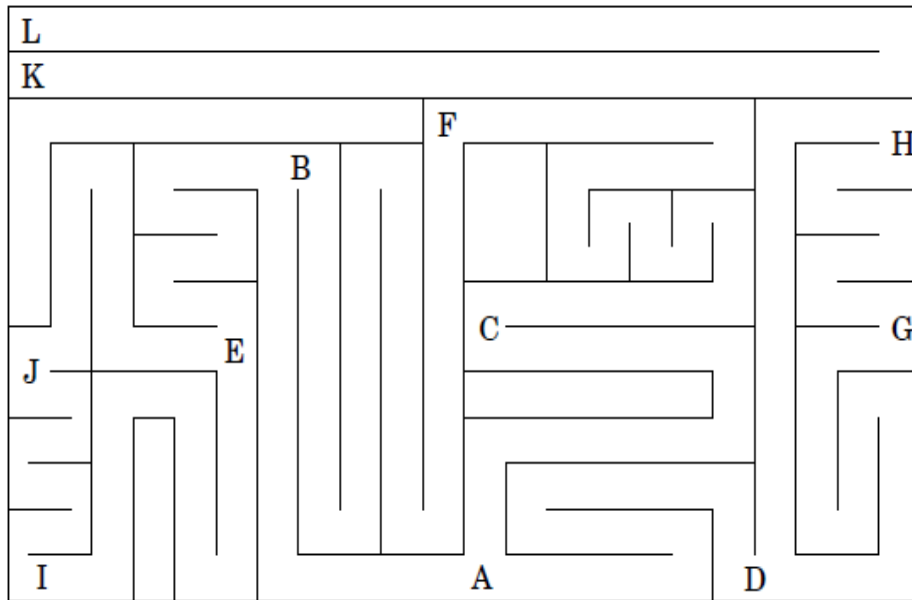
Un esempio: visita DFS



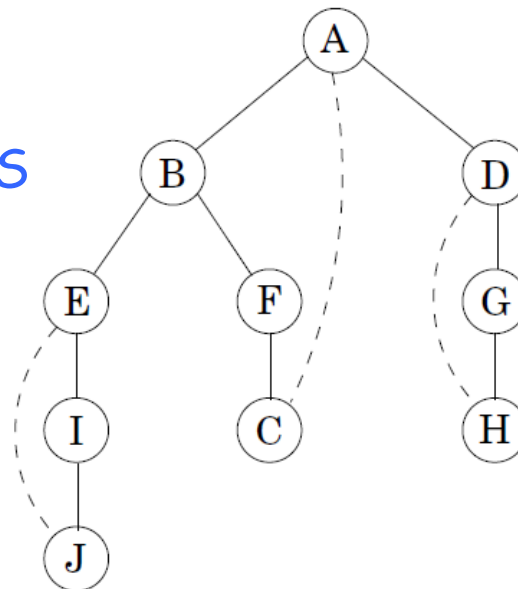
Esempio: grafo orientato



...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**