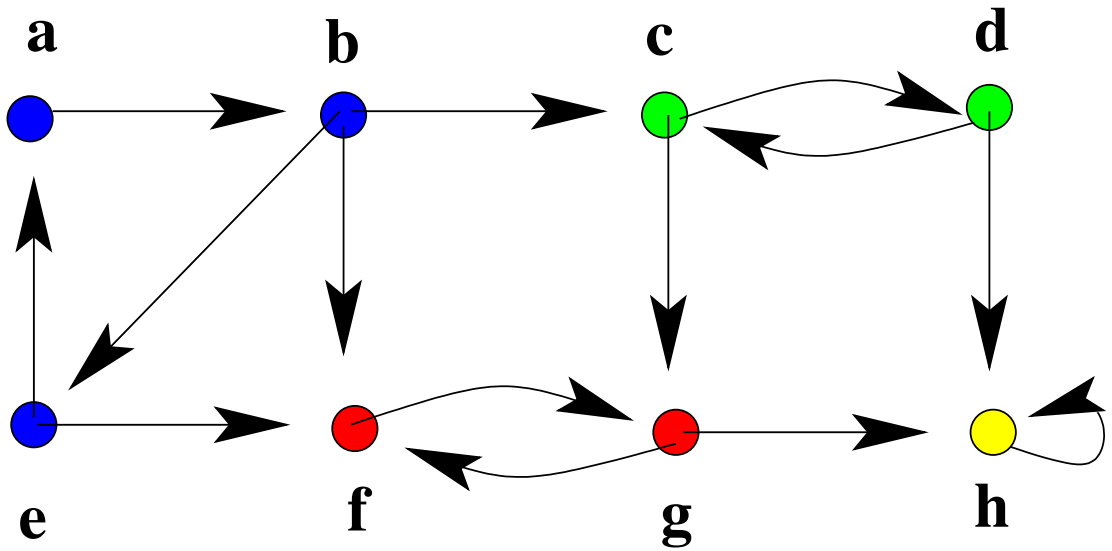
Componenti fortemente connesse

Sia $G = (V, E)$ un grafo diretto. Una **componente fortemente connessa** di G è un insieme **massimale** $C \subseteq V$ tale che, per ogni $u, v \in C$, esistono un cammino tra u e v e viceversa.

Equivalentemente, le componenti fortemente connesse di un grafo G sono le **classi di equivalenza** della relazione di mutua raggiungibilià.

Componenti fortemente connesse

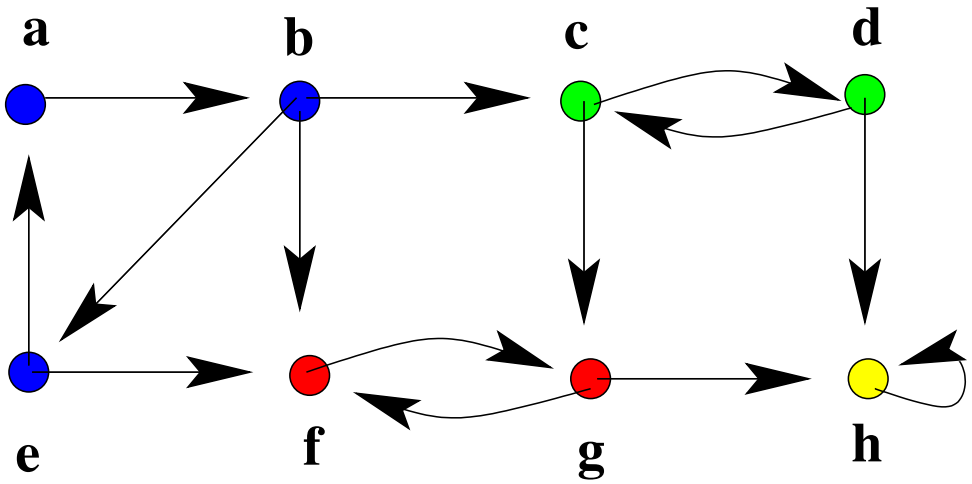


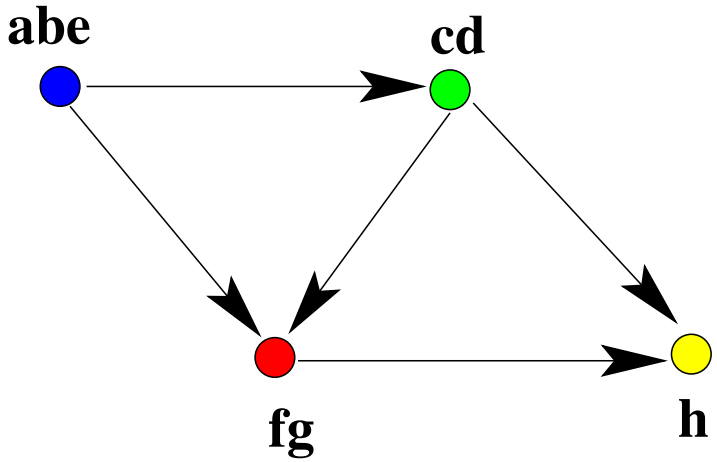
Grafo delle componenti

Dato un grafo orientato $G = (V, E)$ con k componenti fortemente connesse $C_1, \ldots, C_k$, il **grafo delle componenti** $G^C = (V^C, E^C)$ è definito come segue:

- $V^C = \{v_1, \ldots, v_k\}$;
- $E^C = \{(v_i, v_j) \mid \exists\ (x, y) \in E.\ x \in C_i \wedge y \in C_j \wedge i \neq j\}$

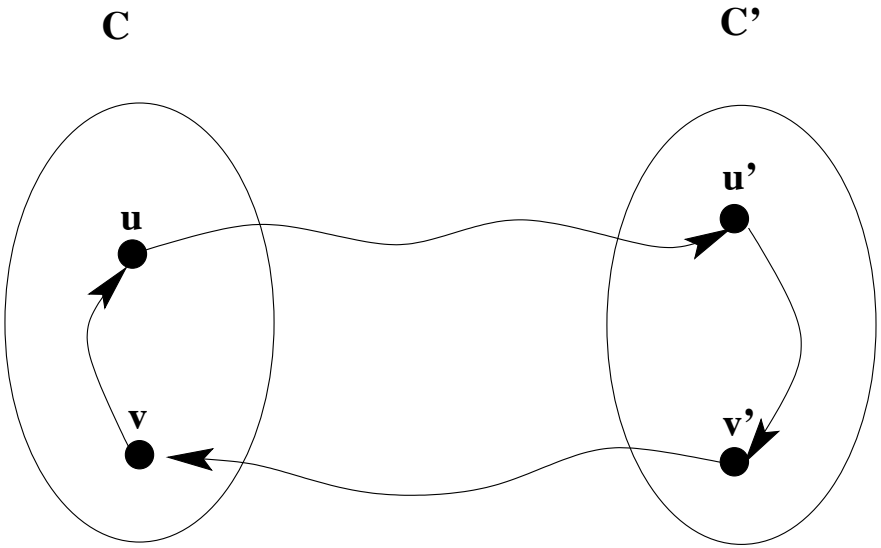
Grafo delle componenti





Grafo delle componenti

Teorema *Il grafo delle componenti è aciclico.*

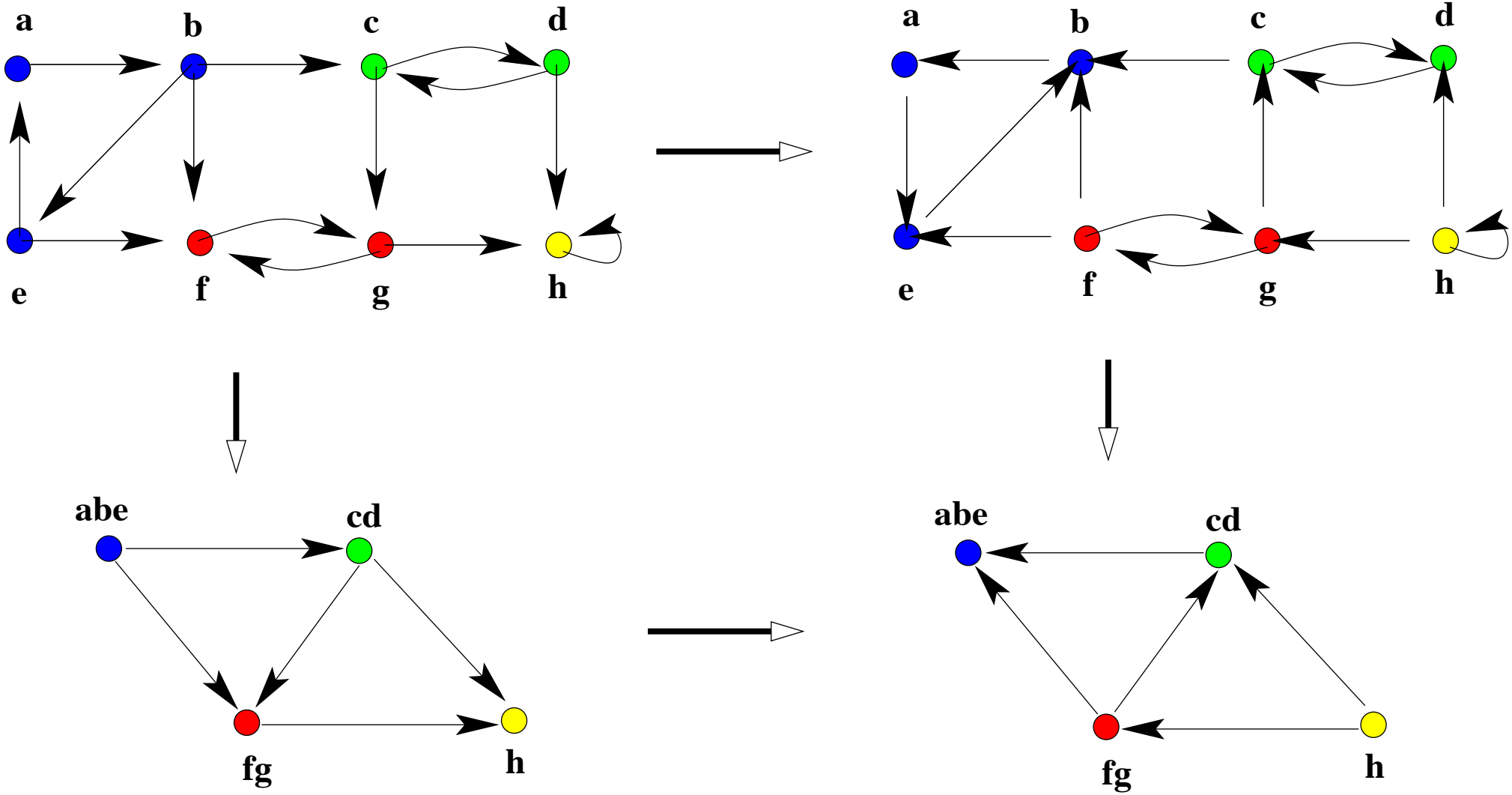


Grafo inverso

Dato un grafo diretto $G = (V, E)$, il **grafo inverso** di G è il grafo $G^{-1} = (V, E^{-1})$ ove

$$E^{-1} = \{(v, u) \ : \ (u, v) \in E\}$$

Si noti che le G e G^{-1} hanno le stesse componenti fortemente connesse; infatti se u e v sono mutuamente raggiungibili in G allora lo sono anche in G^{-1}.


a
b
c
d
e
f
g
h
a
b
c
d
e
f
g
h
abe
cd
fg
h
abe
cd
fg
h

Tempi di scoperta e di terminazione globali

Estendiamo la nozione di tempo di scoperta e tempo di fine ad un insieme di nodi.

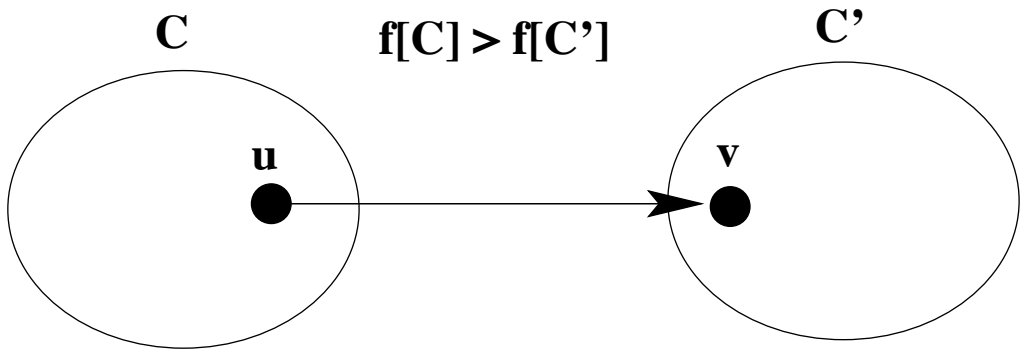
Sia $U \subseteq V$. Allora

$$d[U] = min_{u \in U} d[u]$$

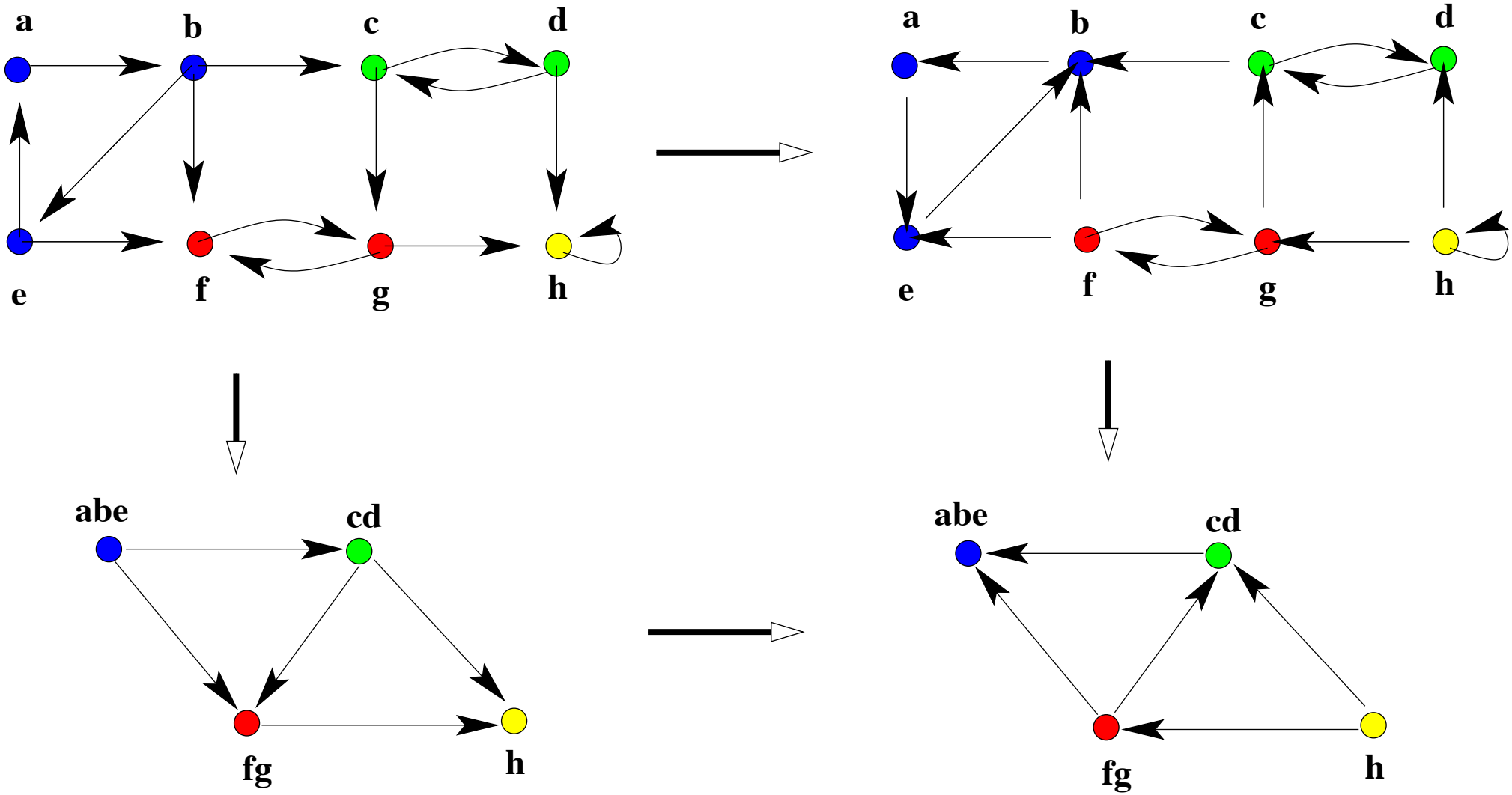
e

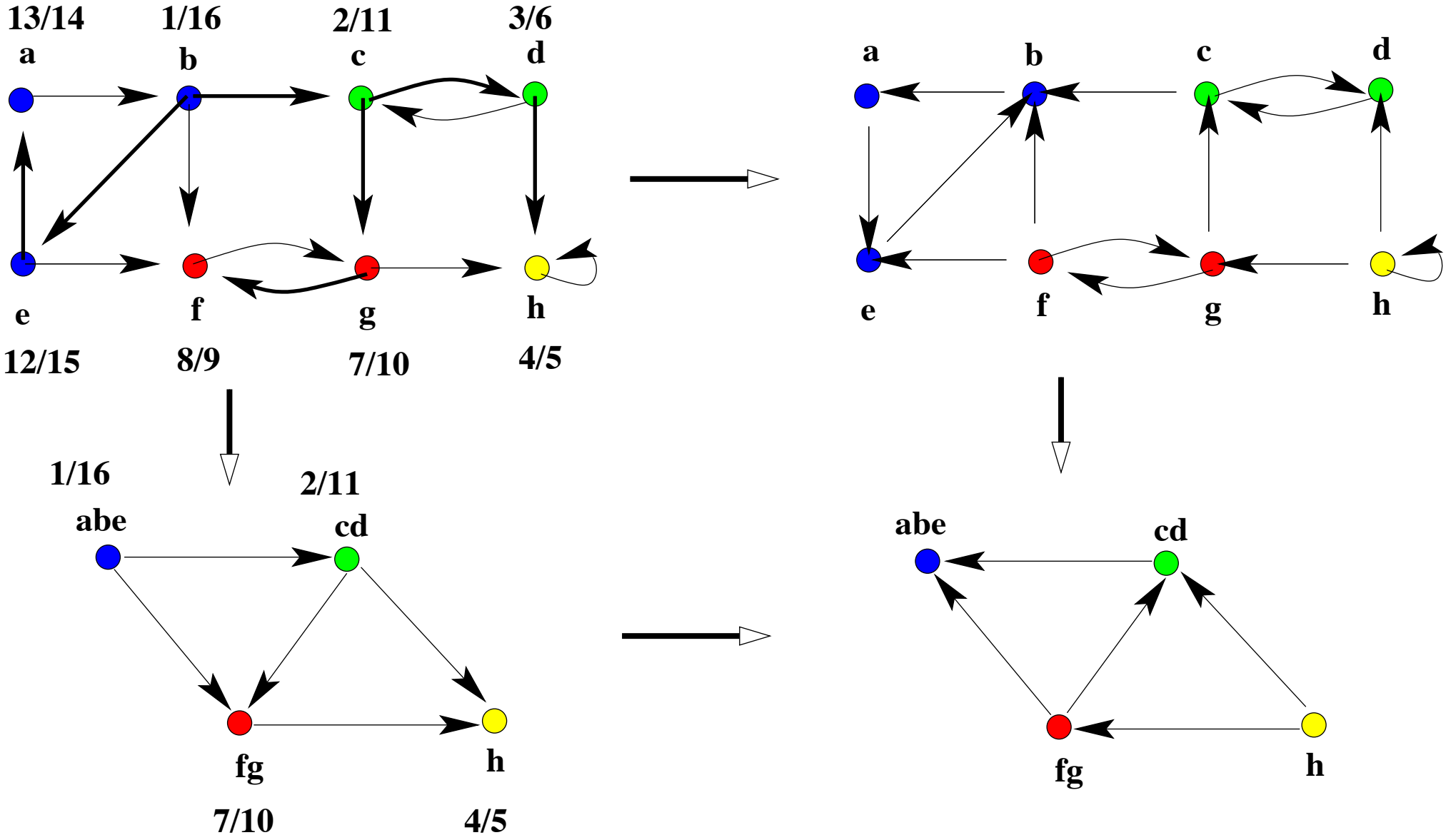
$$f[U] = max_{u \in U} f[u]$$

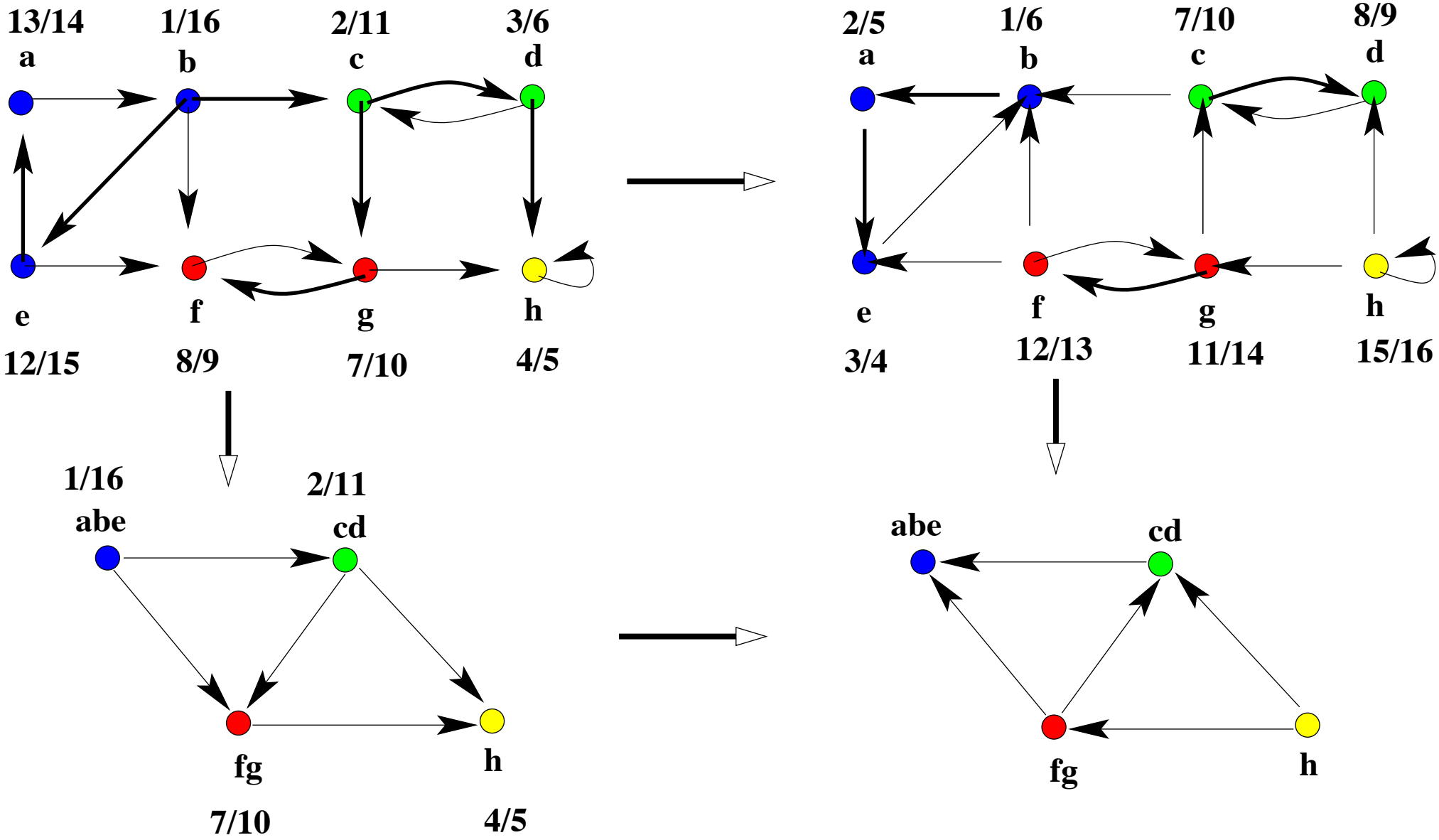
Teorema *Siano C e C' componenti fortemente connesse distinte di un grafo diretto $G = (V, E)$. Supponiamo esista un arco $(u, v) \in E$, dove $u \in C$ e $v \in C'$. Allora $f[C] > f[C']$.*



Corollario *Siano C e C' componenti fortemente connesse distinte di un grafo diretto $G = (V, E)$. Sia $G^{-1} = (V, E^{-1})$ il grafo inveso di G. Supponiamo esista un arco $(u, v) \in E^{-1}$, dove $u \in C$ e $v \in C'$. Allora $f[C] < f[C']$.*


a
b
c
d
e
f
g
h
a
b
c
d
e
f
g
h
abe
cd
fg
h
abe
cd
fg
h


13/14
a
1/16
b
2/11
c
3/6
d
e
12/15
f
8/9
g
7/10
h
4/5
a
b
c
d
e
f
g
h
1/16
abe
2/11
cd
fg
7/10
h
4/5
abe
cd
fg
h


13/14
a
1/16
b
2/11
c
3/6
d
e
12/15
f
8/9
g
7/10
h
4/5
2/5
a
1/6
b
7/10
c
8/9
d
e
3/4
f
12/13
g
11/14
h
15/16
1/16
abe
2/11
cd
fg
7/10
h
4/5
abe
cd
fg
h

SCC(G)

1. richiama DFS(G) e inserisci in una pila S ogni vertice di G una volta terminato;
2. costruisci il grafo inverso G^{-1};
3. richiama DFS(G^{-1}), ma nel ciclo principale di DFS considera i vertici prendendoli dalla pila S, cioè in ordine decrescente rispetto ai tempi di fine;
4. restituisci gli insiemi dei nodi degli alberi della DF-foresta associata alla visita di G^{-1}.

1: **SCC(G)**
2: $k \leftarrow 0$
3: $S \leftarrow \emptyset$
4: $DFS(G)$
5: $ListGraphInverse(G, G^{-1})$
6: $DFS2(G^{-1})$

```
1: DFS(G)
2: for each u ∈ V[G] do
3:     color[u] ← WHITE
4:     π[u] ← NIL
5: end for
6: time ← 0
7: for each u ∈ V[G] do
8:     if color[u] = WHITE then
9:         DFSVisit(u)
10:    end if
11: end for
```

```
1:  DFSVisit(u)
2:  color[u] ← GRAY
3:  time ← time + 1
4:  d[u] ← time
5:  for each v ∈ Adj[u] do
6:      if color[v] = WHITE then
7:          π[v] ← u
8:          DFSVisit(v)
9:      end if
10: end for
11: color[u] ← BLACK
12: time ← time + 1
13: f[u] ← time
14: Push(S, u)
```

```
1: ListGraphInverse(G, G^-1)
2: for each u ∈ V[G] do
3:     for each v ∈ Adj[u] do
4:         ListInsert(Adj^-1[v], u)
5:     end for
6: end for
```

```
1: DFS2(G^{-1})
2: for each u ∈ V[G^{-1}] do
3:     color[u] ← WHITE
4:     π[u] ← NIL
5: end for
6: time ← 0
7: while not StackEmpty(S) do
8:     u ← Pop(S)
9:     if color[u] = WHITE then
10:        DFSVisit2(u)
11:    end if
12: end while
```

```
1:  DFSVisit2(u)
2:  if π[u] = NIL then
3:      k ← k + 1
4:      scc[u] ← k
5:  else
6:      scc[u] ← scc[π[u]]
7:  end if
8:  color[u] ← GRAY
9:  time ← time + 1
10: d[u] ← time
```

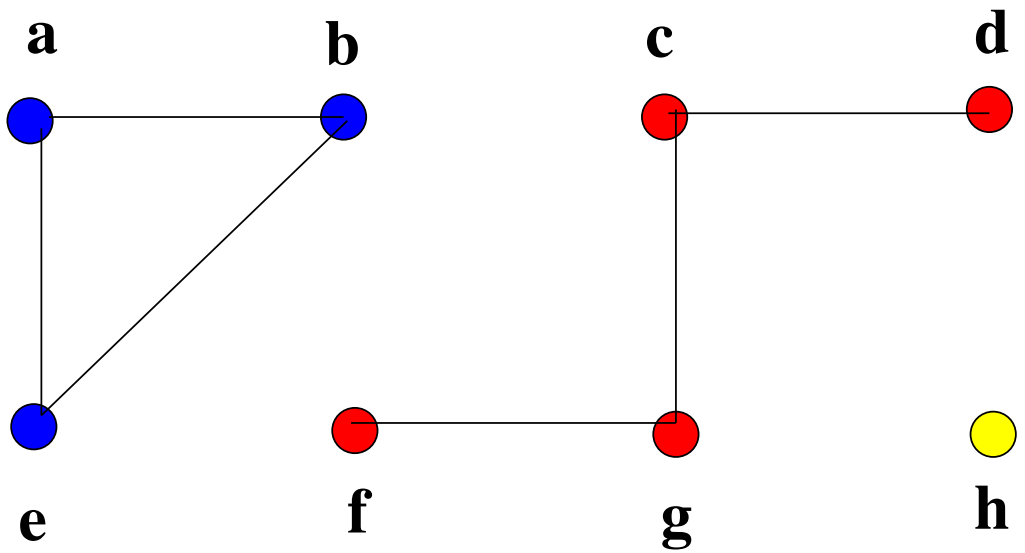
11: **for** each $v \in Adj^{-1}[u]$ **do**
12: **if** $color[v] =$ WHITE **then**
13: $\pi[v] \leftarrow u$
14: $DFSVisit2(v)$
15: **end if**
16: **end for**
17: $color[u] \leftarrow$ BLACK
18: $time \leftarrow time + 1$
19: $f[u] \leftarrow time$

Componenti connesse

Sia $G = (V, E)$ un grafo indiretto. Una **componente connessa** di G è un insieme **massimale** $C \subseteq V$ tale che, per ogni $u, v \in C$, esistono un cammino tra u e v.

Equivalentemente, le componenti connesse di un grafo G sono le **classi di equivalenza** della relazione di raggiungibilià.

Componenti connesse



Componenti connesse

Teorema *Dato un grafo indiretto G e due vertici u e v, esiste un cammino da u a v in G se e solo se u e v appartengono allo stesso albero della DF-foresta.*

Dimostrazione

$\Rightarrow$) Supponiamo per assurdo che esista un cammino da u a v sul grafo G ma che u e v appartengano ad **alberi differenti** della DF-foresta. Quindi, almeno un arco (x, y) del cammino da u a v deve connettere vertici appartenenti ad alberi distinti della DF-foresta. Dunque (x, y) è un **arco cross**. Ciò è assurdo in quanto una DFS su un grafo indiretto non genera archi cross.

$\Leftarrow$) Dato che la DF-foresta è un sottografo di G, se u e v appartengono allo stesso albero della DF-foresta, allora esiste un cammino in G che li connette.

```
1: CC(G)
2: k ← 0
3: for each u ∈ V[G] do
4:     color[u] ← WHITE
5:     π[u] ← NIL
6: end for
7: time ← 0
8: for each u ∈ V[G] do
9:     if color[u] = WHITE then
10:        DFSVisit(u)
11:    end if
12: end for
```

```
1:  DFSVisit(u)
2:  if π[u] = NIL then
3:      k ← k + 1
4:      cc[u] ← k
5:  else
6:      cc[u] ← cc[π[u]]
7:  end if
8:  color[u] ← GRAY
9:  time ← time + 1
10: d[u] ← time
```

```
11: for each v ∈ Adj[u] do
12:     if color[v] = WHITE then
13:         π[v] ← u
14:         DFSVisit(v)
15:     end if
16: end for
17: color[u] ← BLACK
18: time ← time + 1
19: f[u] ← time
```

Complessità computazionale

Il calcolo delle componenti fortemente connesse costa:

- $\Theta(n+m)$ per la prima visita in profondità;
- $\Theta(n+m)$ per il calcolo del grafo inverso;
- $\Theta(n+m)$ per la seconda visita in profondità.

In totale, $\Theta(3(n+m)) = \Theta(n+m)$.

Il calcolo delle componenti connesse costa come una visita in profondità, quindi $\Theta(n+m)$.