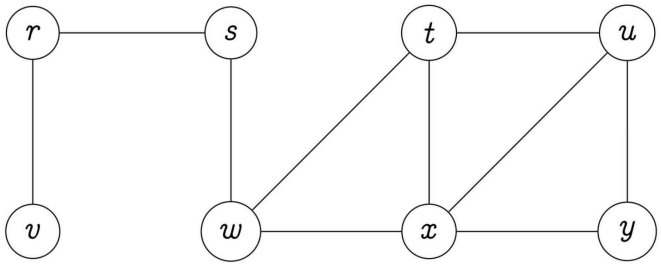
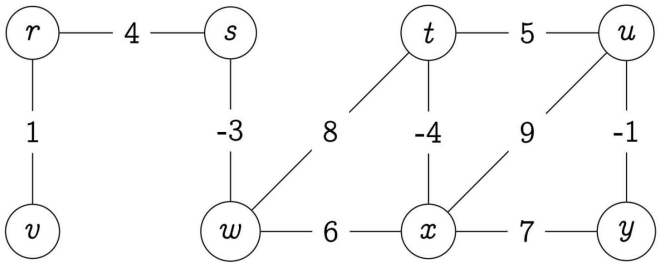
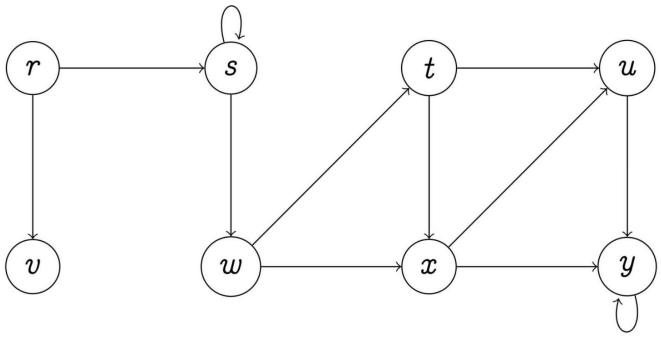
Esempio di grafo non orientato UNDIRECTED



Esempio di grafo pesato WEIGHTED



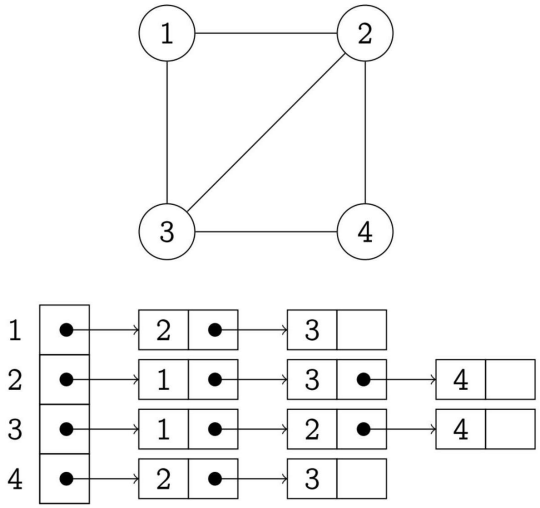
Esempio di grafo orientato DIRECTED



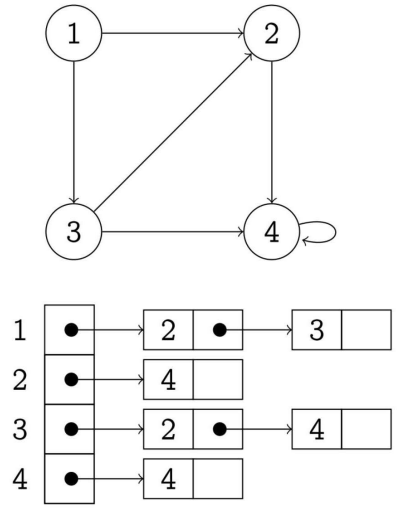
Grafo: definizione

- Un Grafo è una coppia $G = (V, E)$
- $V = \{v_1, v_2, \ldots, v_n\}$ è l'insieme dei vertici o nodi del grafo
- $E = \{e_1, e_2, \ldots, e_m\}$ è l'insieme degli archi del grafo
- Grafi orientati e non orientati
- Grafi pesati e non pesati

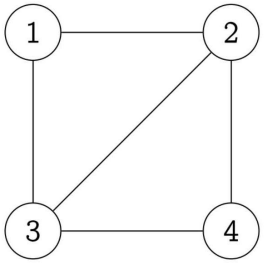
Grafo non orientato: liste di adiacenza



Grafo orientato: liste di adiacenza



Grafo non orientato: matrice di adiacenza



M x M = costo percorso minimo

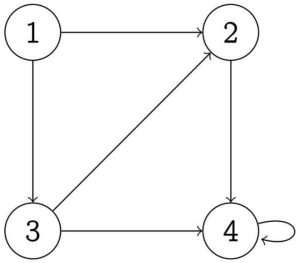
Nodo → Nodo

$$M = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

i = 2
j = 3

non orientato
⇒ simmetrica

Grafo orientato: matrice di adiacenza



$$M = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dimensione di un grafo e spazio di memoria occupata

- Sia G un grafo con n nodi e m archi
- G ha una dimensione pari a $n + m$
- Un grafo rappresentato con liste di adiacenza occupa memoria pari a $\Theta(n + m)$
- Un grafo rappresentato con matrice di adiacenza occupa memoria pari a $\Theta(n^2)$
- Un grafo è sparso se $m = O(n)$
- Si noti che in generale $m = O(n^2)$

Graph-visit: ordine di visita dei nodi

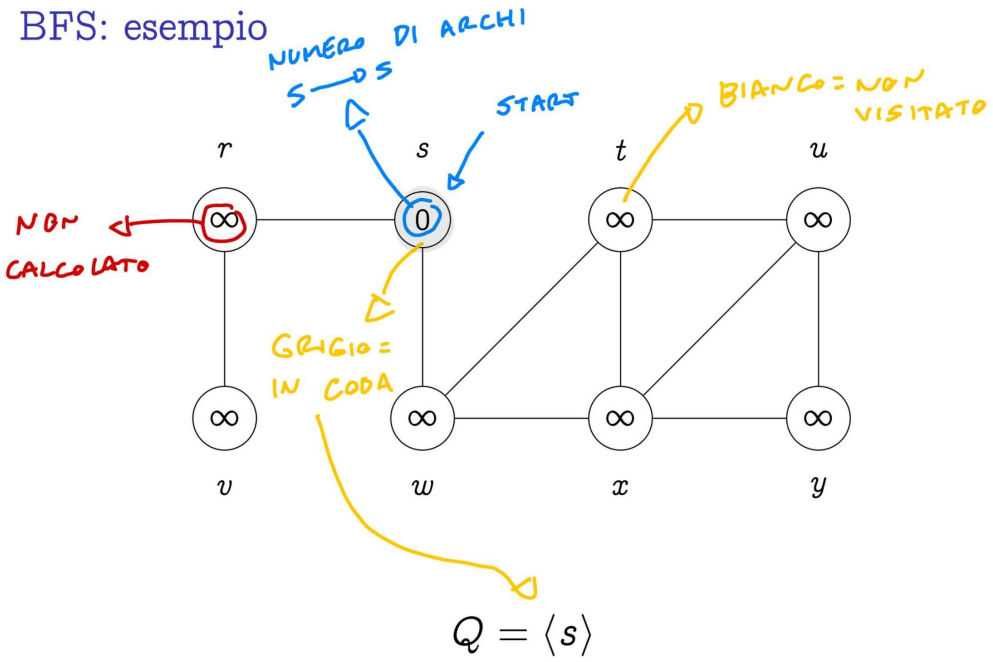
BSF-visit è in grado di calcolare l'albero dei cammini minimi e la distanza di ogni nodo dalla sorgente della visita

Visita di un grafo in ampiezza: BFS

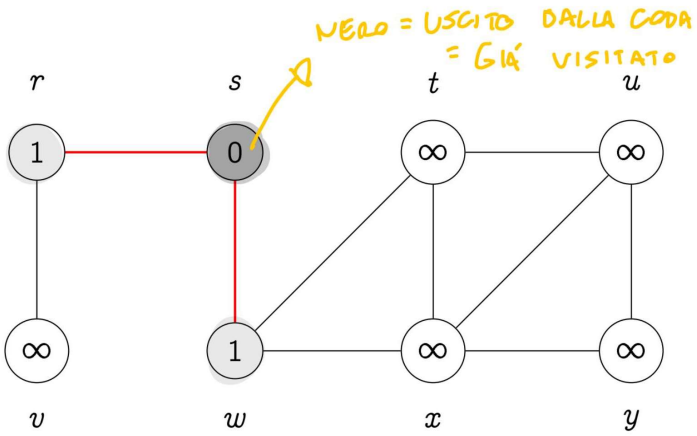
BREADTH-FIRST SEARCH "mi allontano solo se costretto"

```
BFS-Visit(G, s)                                   ← non sensibile all'ordine
 1  for each vertex u ∈ G.V − {s}                 ┐
 2      u.color = WHITE; u.d = ∞; u.π = NIL       ┘ Θ(n)   (u.π = padre)
 3  s.color = GRAY; s.d = 0; s.π = NIL
 4  Q = ∅
 5  Enqueue(Q, s)
 6  while Q ≠ ∅                                   ┐
 7      u = Dequeue(Q)                            │
 8      for each v ∈ G.Adj[u]   adiacenti; nodo u  │
 9          if v.color == WHITE                   │ Θ(m)
10              v.color = GRAY; v.d = u.d + 1; v.π = u
11              Enqueue(Q, v)                     │
12      u.color = BLACK;                          ┘
```

BFS: esempio

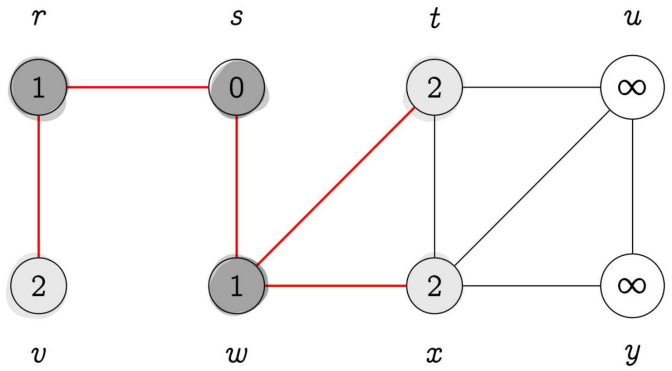


BFS: esempio



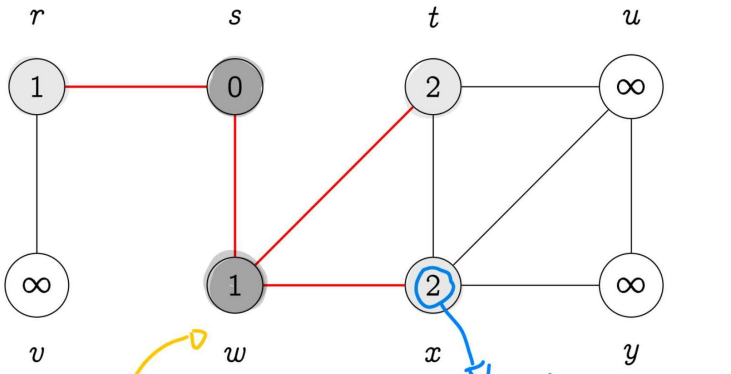
$$Q = \langle w, r \rangle$$

BFS: esempio



$$Q = \langle t, x, v \rangle$$

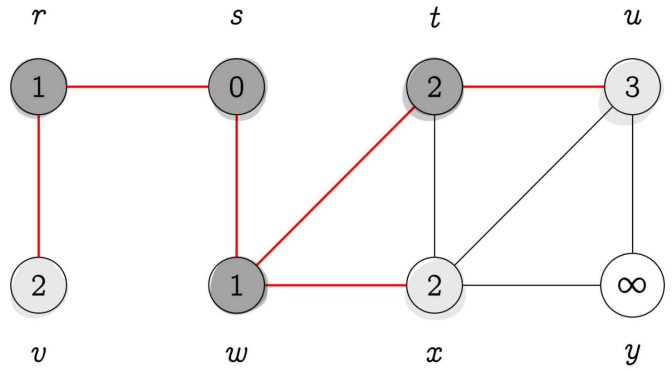
BFS: esempio



$$Q = \langle r, t, x \rangle$$

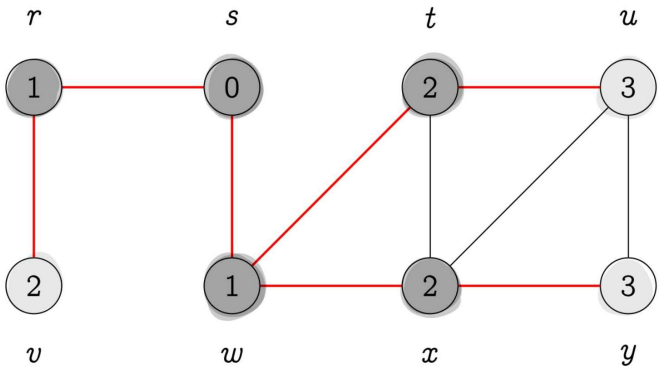
s già nero

BFS: esempio



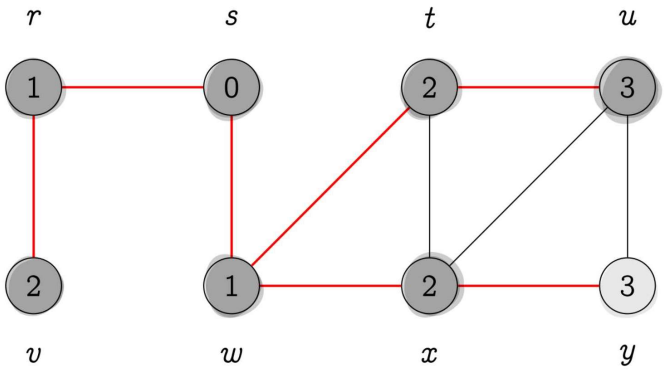
$$Q = \langle x, v, u \rangle$$

BFS: esempio



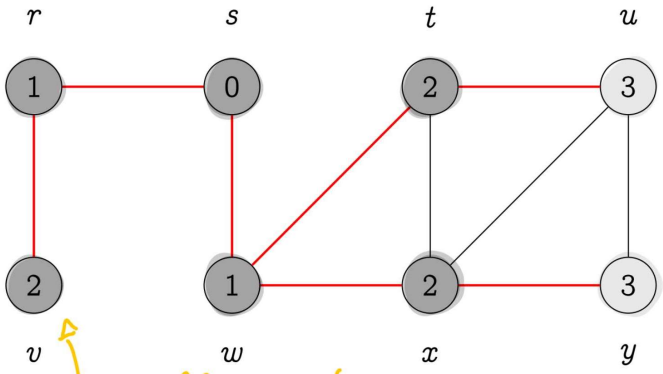
$Q = \langle v, u, y \rangle$

BFS: esempio



$Q = \langle y \rangle$

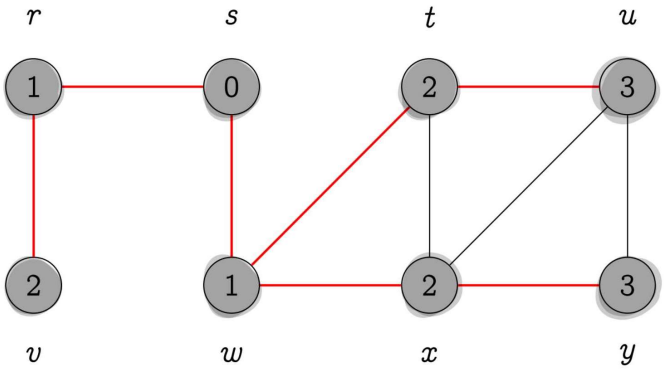
BFS: esempio



no collegamenti non neri

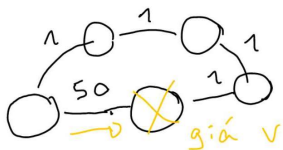
$Q = \langle u, y \rangle$

BFS: esempio



Non funziona per grafi pesati

$Q = \langle \rangle \implies$ fine



già visitato

Implementazione e costo della BFS

- G implementato con liste di adiacenza
- Q implementata con vettori (inserimento ed estrazioni di costo costante)
- G ha n nodi e m archi
- Istruzioni $1-3$: costo $\Theta(n)$
- Istruzioni $4-12$: costo $\Theta(m)$
- Costo complessivo $\Theta(n+m)$

Ricostruzione e stampa di un cammino

partenza BFS

```
PRINT-PATH(G, s, v)
1  if v == s
2        print s
3  elseif v.π == NIL
4        print no path from s to v exists
5  else PRINT-PATH(G, s, v.π)
6        print v
```

Costo computazionale BFS

Costo computazionale BFS

La visita BFS ha un costo computazionale $\Theta(n+m)$ ed è quindi un algoritmo ottimo

DFS

DEPTH-FIRST SEARCH

"resto vicino solo se costretto"

BFS con pila

```
DFS(G)
1  for each vertex u ∈ G.V
2        u.color = WHITE
3        u.π = NIL
4  time = 0
5  for each vertex u ∈ G.V
6        if u.color == WHITE
7              DFS-VISIT(G, u)
```

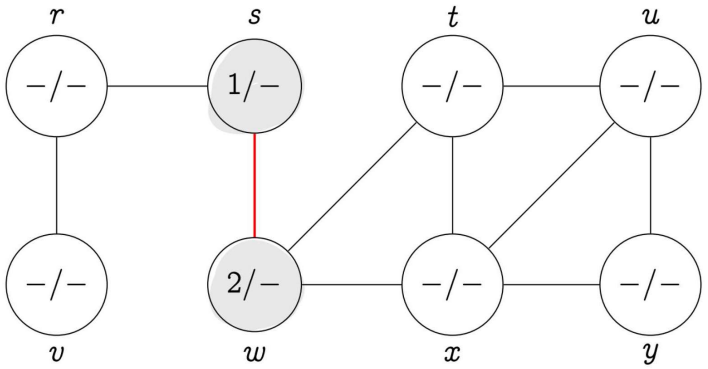
pila tramite ricorsione

DFS sulla singola componente connessa

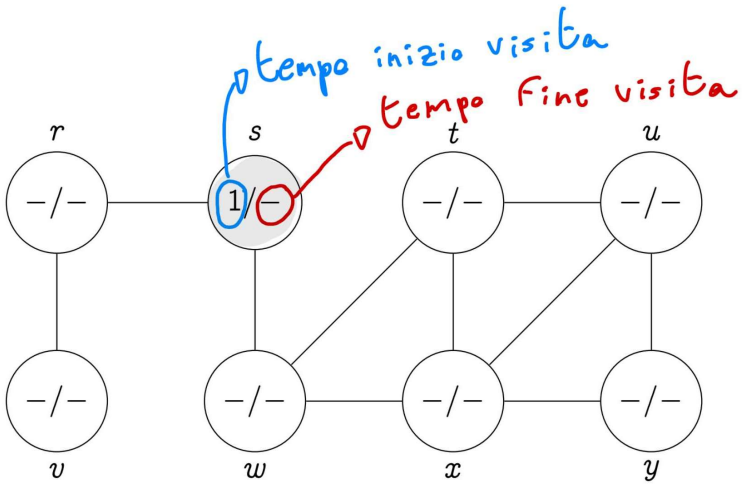
bianco

```
DFS-VISIT(G, u)
1   time = time + 1
2   u.d = time          START
3   u.color = GRAY
4   for each vertex v ∈ G.Adj[u]
5       if v.color == WHITE
6           v.π = u
7           DFS-VISIT(G, v)
8   u.color = BLACK
9   time = time + 1
10  u.f = time          END
```

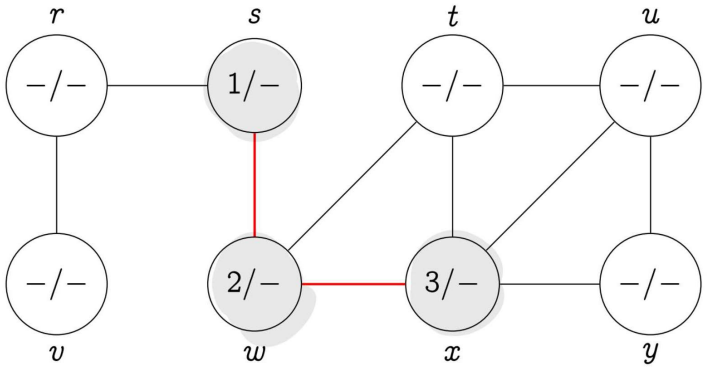
DFS: esempio



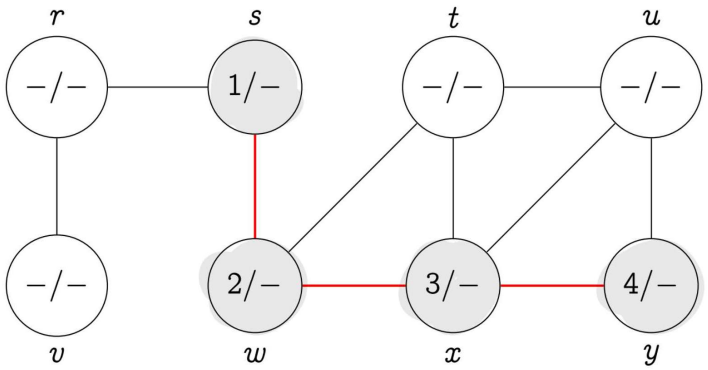
DFS: esempio



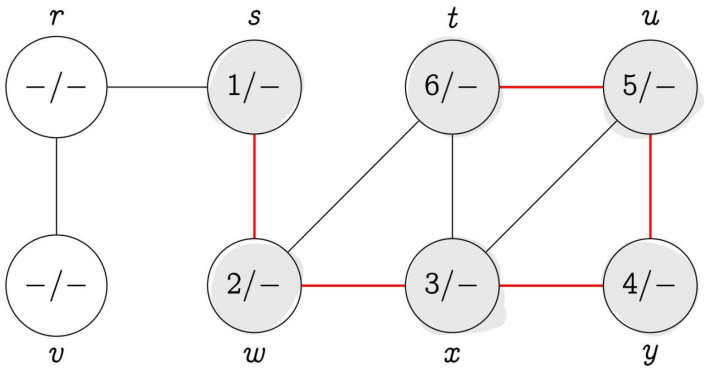
DFS: esempio



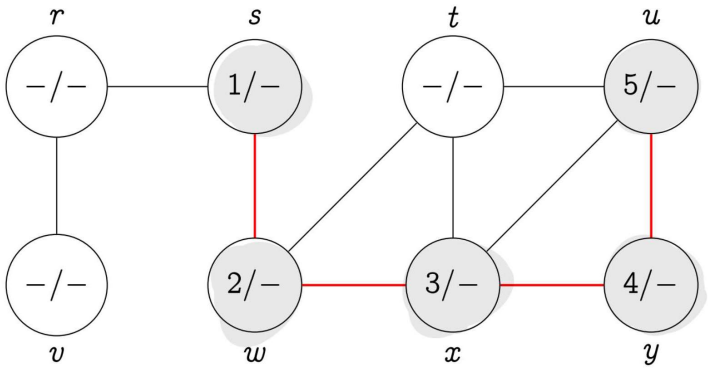
DFS: esempio



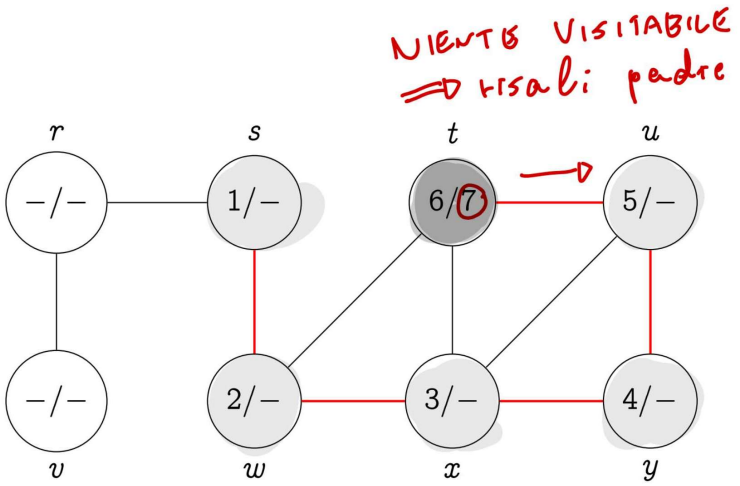
DFS: esempio



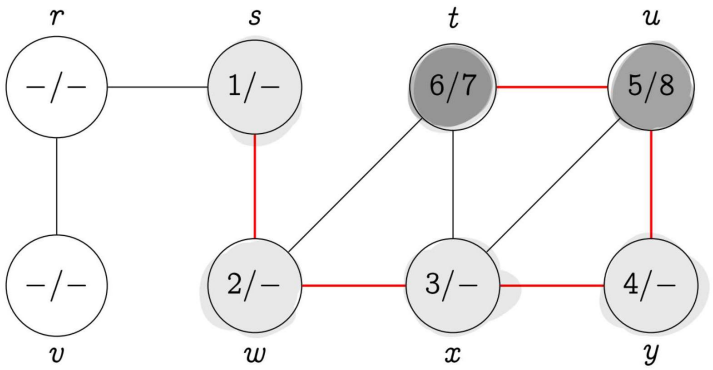
DFS: esempio



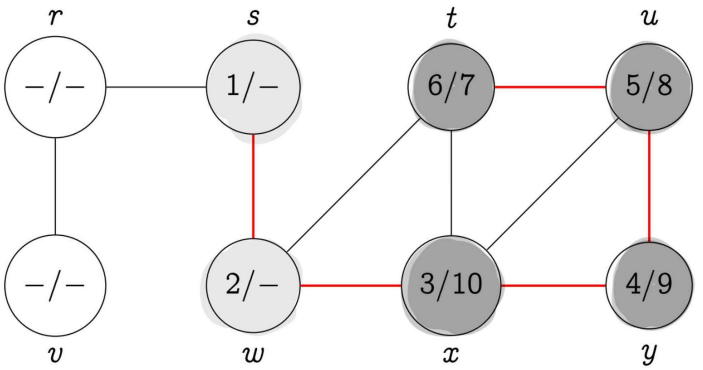
DFS: esempio



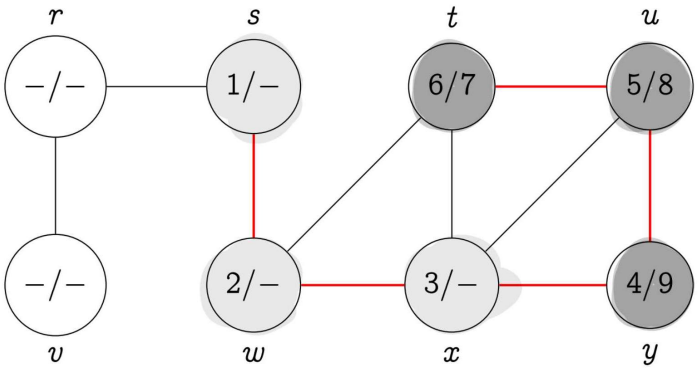
DFS: esempio



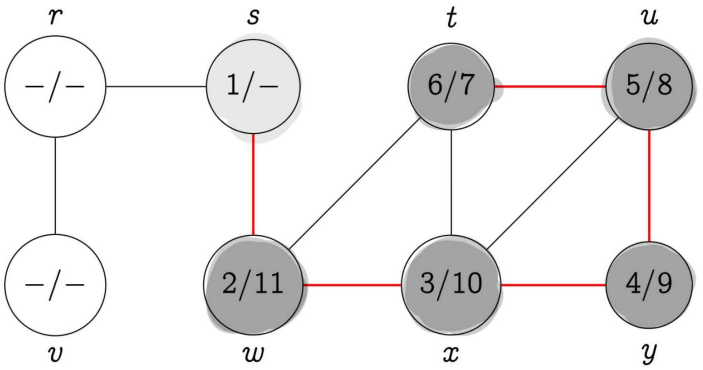
DFS: esempio



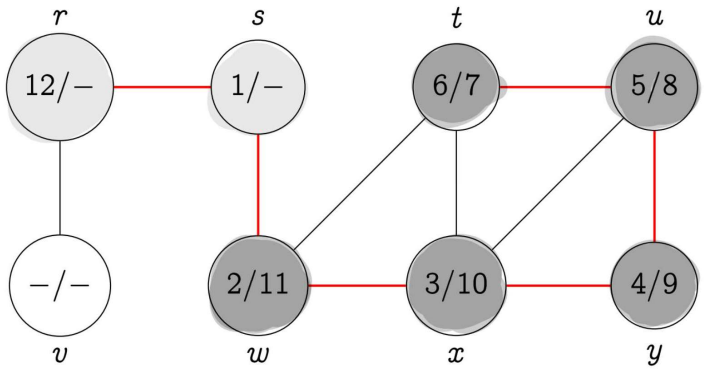
DFS: esempio



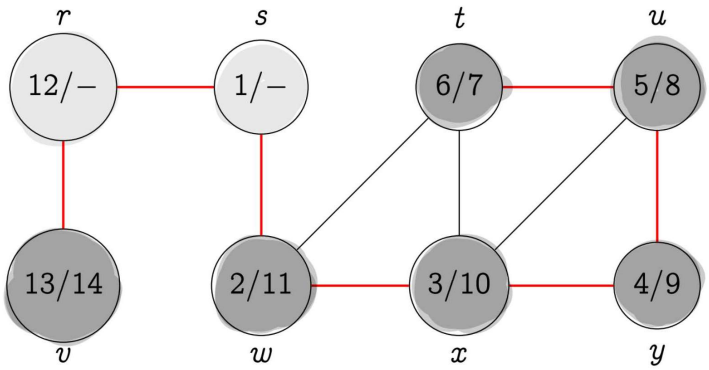
DFS: esempio



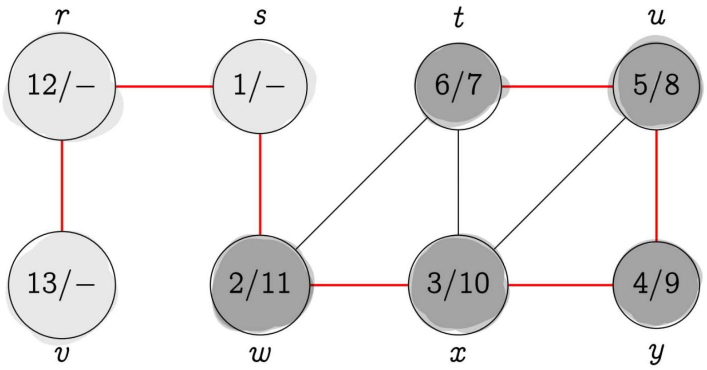
DFS: esempio



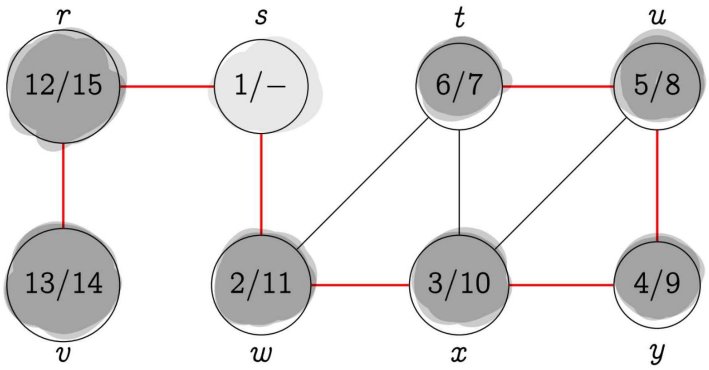
DFS: esempio



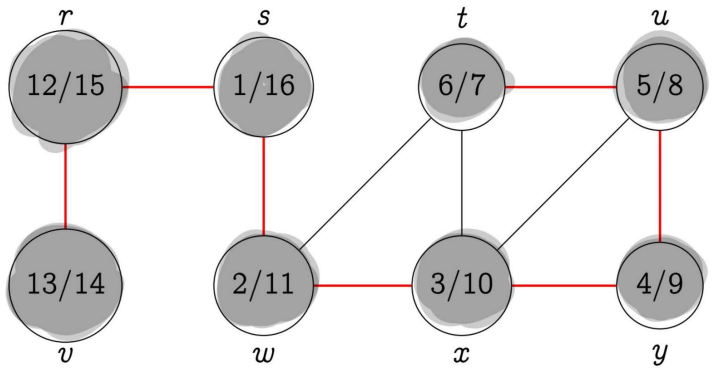
DFS: esempio



DFS: esempio



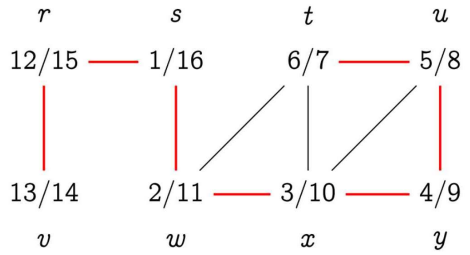
DFS: esempio



Implementazione e costo della DFS

- G implementato con liste di adiacenza
- G ha n nodi e m archi
- Analisi di ogni componente connessa separatamente
- Costo complessivo $\Theta(n+m)$ (per esercizio)

DFS: parentesi ben formate



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
(s	(w	(x	(y	(u	(t	t)	u)	y)	x)	w)	(r	(v	v)	r)	s)

inizio-fine visita

parentesi ben strutturate

Costo computazionale DFS

Costo computazionale DFS

La visita DFS ha un costo computazionale $\Theta(n+m)$ ed è quindi un algoritmo ottimo