ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΑΝΑΛΥΣΗ ΑΛΓΟΡΙΘΜΩΝ

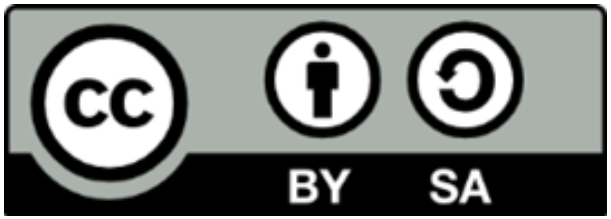
Ενότητα 10γ: Αλγόριθμοι Γραφημάτων- Διερεύνηση Πρώτα σε Βάθος (DFS)- Τοπολογική Ταξινόμηση

Μαρία Σατρατζέμη
Τμήμα Εφαρμοσμένης Πληροφορικής

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Μακεδονίας» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Depth-First Search

- ***Πρώτα σε Βάθος διερεύνηση*** *(Depth-First Search)* είναι μια άλλη στρατηγική για να εξερευνήσουμε ένα γράφημα
  - Εξερευνούμε “βαθύτερα” στο γράφημα οποτεδήποτε αυτό είναι δυνατόν
  - Οι ακμές εξερευνώνται με αφετηρία από την πιο πρόσφατα εντοπισμένη κορυφή *v* από την οποία εκκινούν μη εξερευνημένες ακμές
  - Αφού εξερευνηθούν όλες οι ακμές της *v*, η διερεύνηση «επιστρέφει» στην κορυφή από την οποία εντοπίστηκε η *v*

Depth-First Search

- Η κορυφή χρωματίζεται λευκή
- Στη συνέχεια χρωματίζεται γκρι όταν *εντοπίζεται*
- Στη συνέχεια μαύρη όταν *περατώνεται* (όταν ολοκληρώνεται η εξέταση της λίστας γειτνίασης της)

Αλγόριθμος Depth-First Search

```
DFS(G)
1.      for each vertex u ∈ G->V
2.          color[u] = WHITE;
3.          p[u] = null
4.      time = 0;
5.      for each vertex u ∈ G->V
6.          if (color[u] == WHITE)
7.              DFS_Visit(u);
```

```
DFS_Visit(u)
1.      color[u] = GREY;
2.      time = time+1;
3.      d[u] = time;
4.      for each v ∈ Adj[u]
5.          if (color[v] == WHITE)
6.              p[v] = u;
7.              DFS_Visit(v);
8.      color[u] = BLACK;
9.      time = time+1;
10.     f[u] = time;
```

Αλγόριθμος Depth-First Search

```
DFS(G)
1.      for each vertex u ∈ G->V
2.          color[u] = WHITE;
3.          p[u] = null
4.      time = 0;
5.      for each vertex u ∈ G->V
6.          if (color[u] == WHITE)
7.              DFS_Visit(u);
```

Η μεταβλητή d[u] καταγράφει τη χρονική στιγμή που εντοπίζεται για πρώτη φορά η κορυφή u (και χρωματίζεται γκρι)

```
DFS_Visit(u)
1.      color[u] = GREY;
2.      time = time+1;
3.      d[u] = time;
4.      for each v ∈ Adj[u]
5.          if (color[v] == WHITE)
6.              p[v] = u;
7.              DFS_Visit(v);
8.      color[u] = BLACK;
9.      time = time+1;
10.     f[u] = time;
```

Αλγόριθμος Depth-First Search

```
DFS(G)
1.    for each vertex u ∈ G->V
2.        color[u] = WHITE;
3.        p[u] = null
4.    time = 0;
5.    for each vertex u ∈ G->V
6.        if (color[u] == WHITE)
7.            DFS_Visit(u);
```

```
DFS_Visit(u)
1.    color[u] = GREY;
2.    time = time+1;
3.    d[u] = time;
4.    for each v ∈ Adj[u]
5.        if (color[v] == WHITE)
6.            p[v] = u;
7.            DFS_Visit(v);
8.    color[u] = BLACK;
9.    time = time+1;
10.   f[u] = time;
```

Η μεταβλητή f[u] καταγράφει τη χρονική στιγμή που ολοκληρώνεται η εξέταση της λίστας γειτνίασης της u (και χρωματίζεται μαύρη)

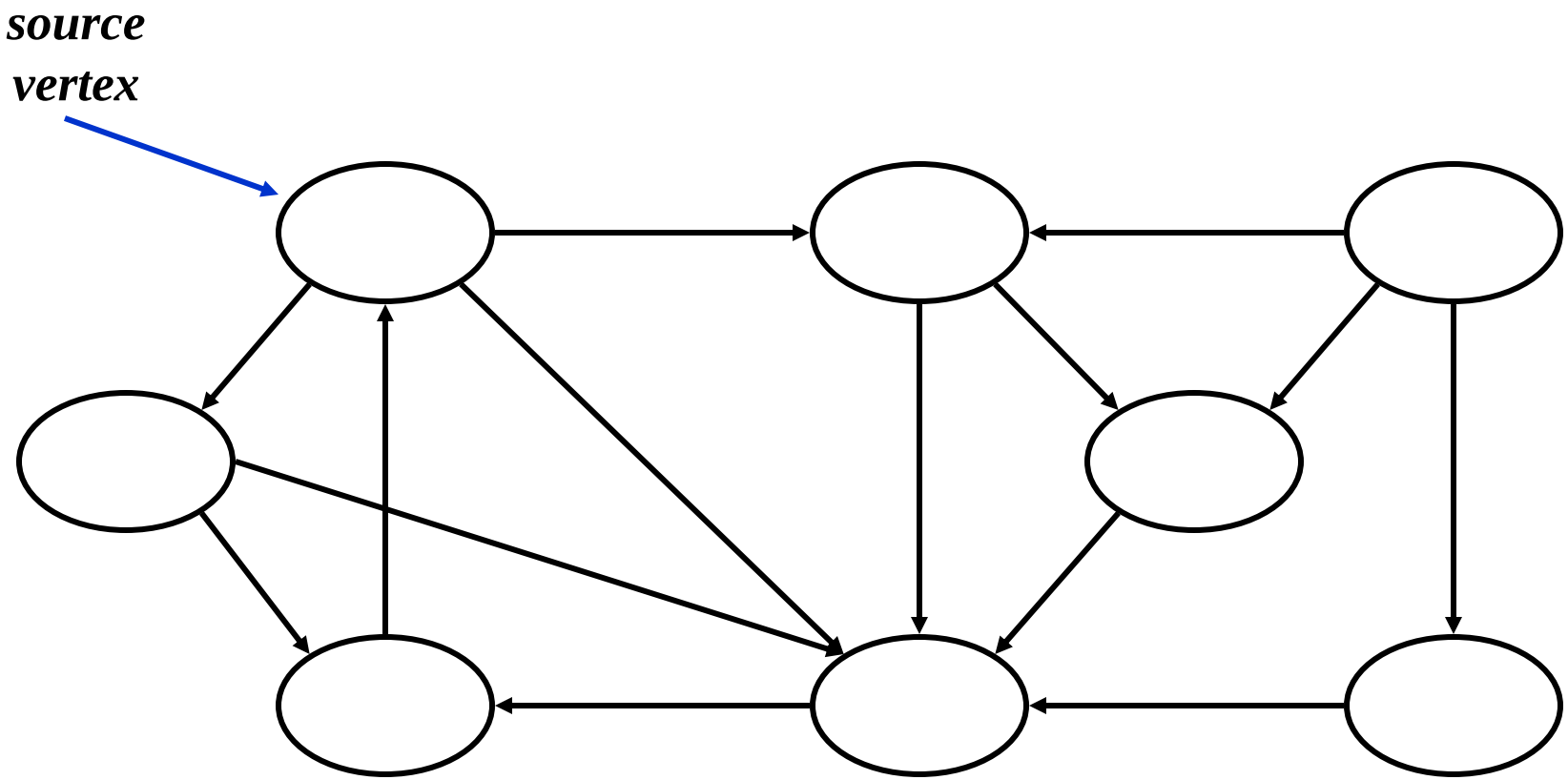
Αλγόριθμος Depth-First Search

```
DFS(G)
1.    for each vertex u ∈ G->V
2.        color[u] = WHITE;
3.        p[u] = null
4.    time = 0;
5.    for each vertex u ∈ G->V
6.        if (color[u] == WHITE)
7.            DFS_Visit(u);
```

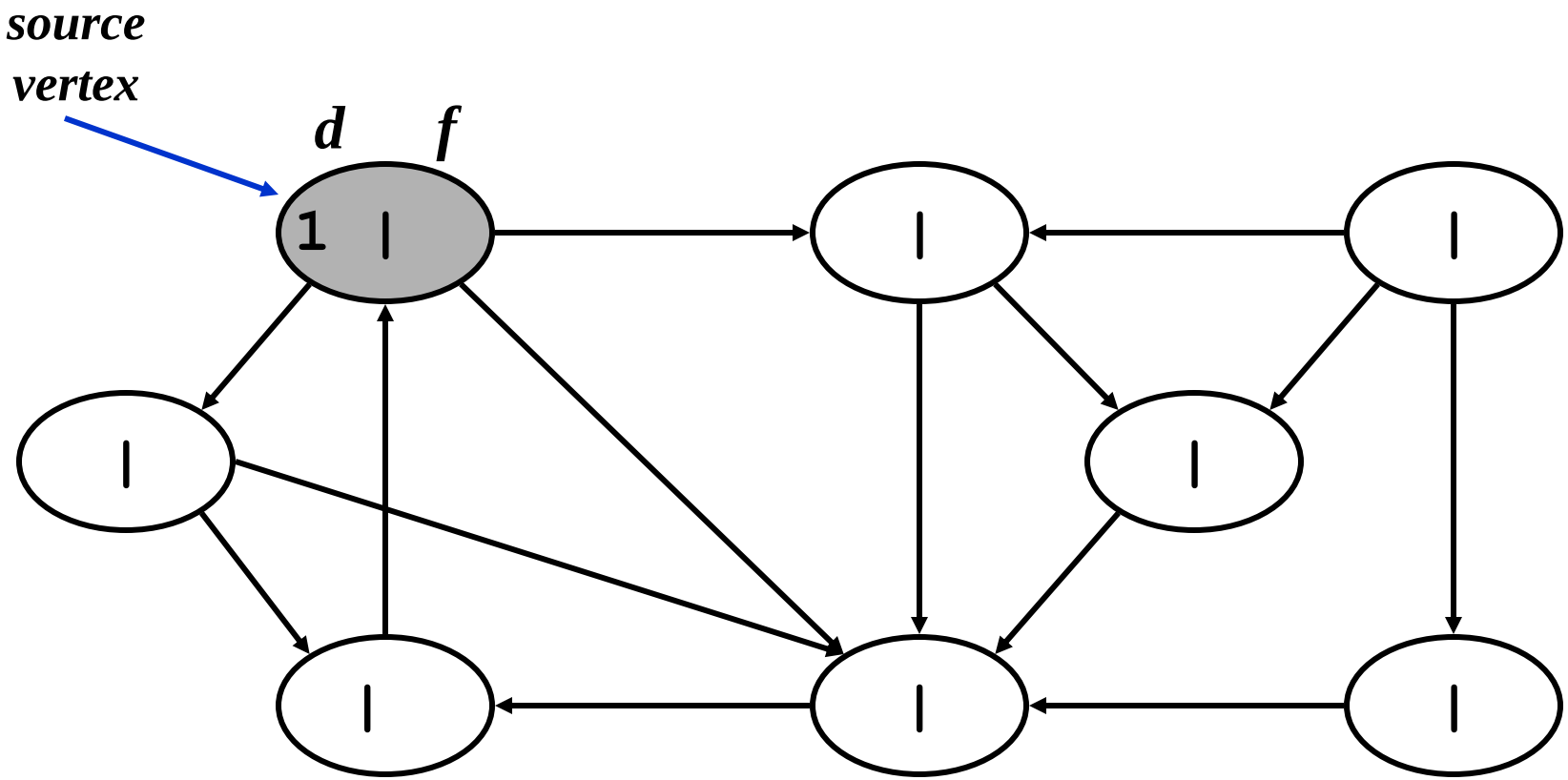
```
DFS_Visit(u)
1.    color[u] = GREY;
2.    time = time+1;
3.    d[u] = time;
4.    for each v ∈ Adj[u]
5.        if (color[v] == WHITE)
6.            p[v] = u;
7.            DFS_Visit(v);
8.    color[u] = BLACK;
9.    time = time+1;
10.   f[u] = time;
```

Θα χρωματιστούν όλες οι κορυφές μαύρες?

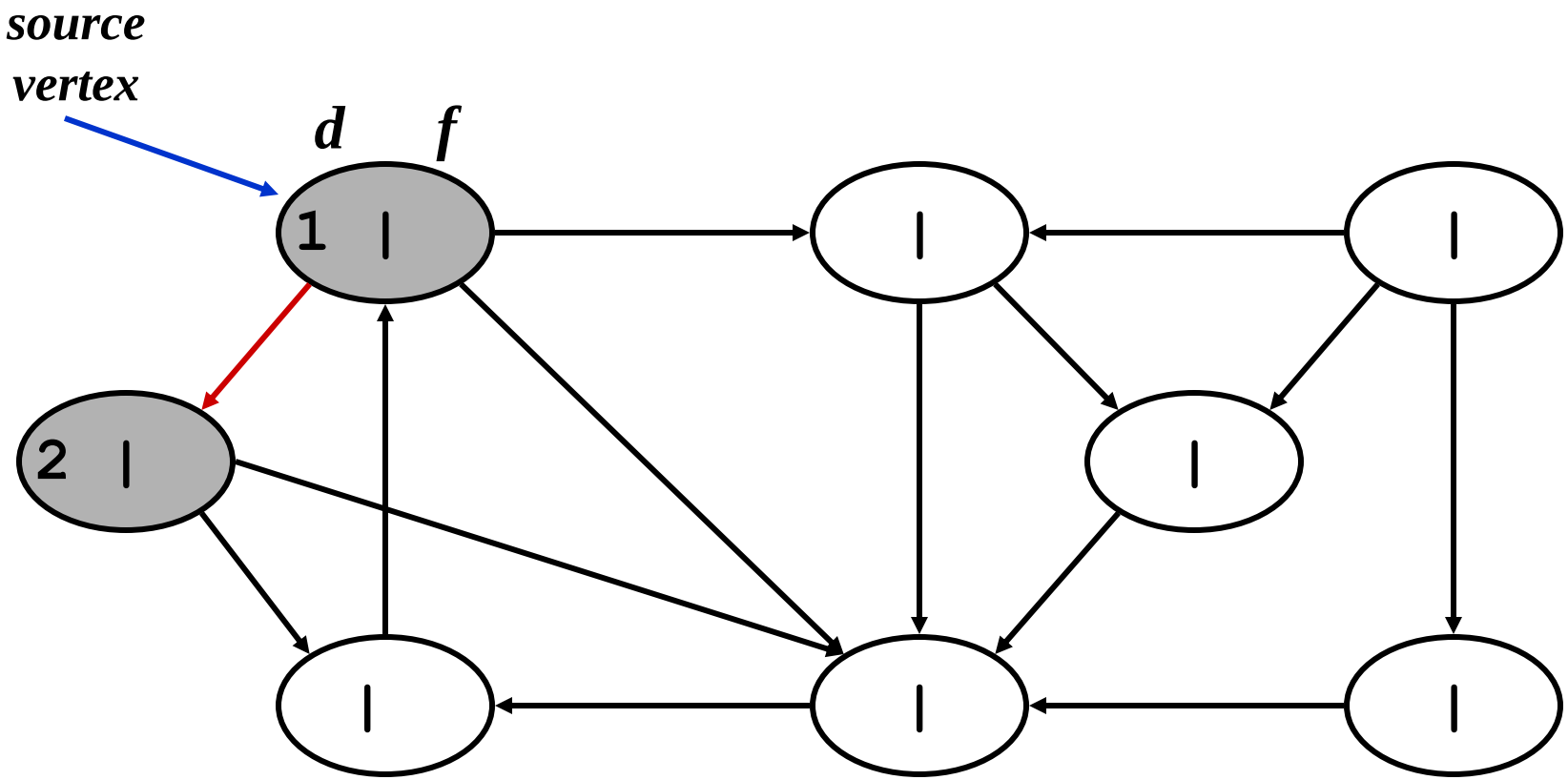
DFS Example



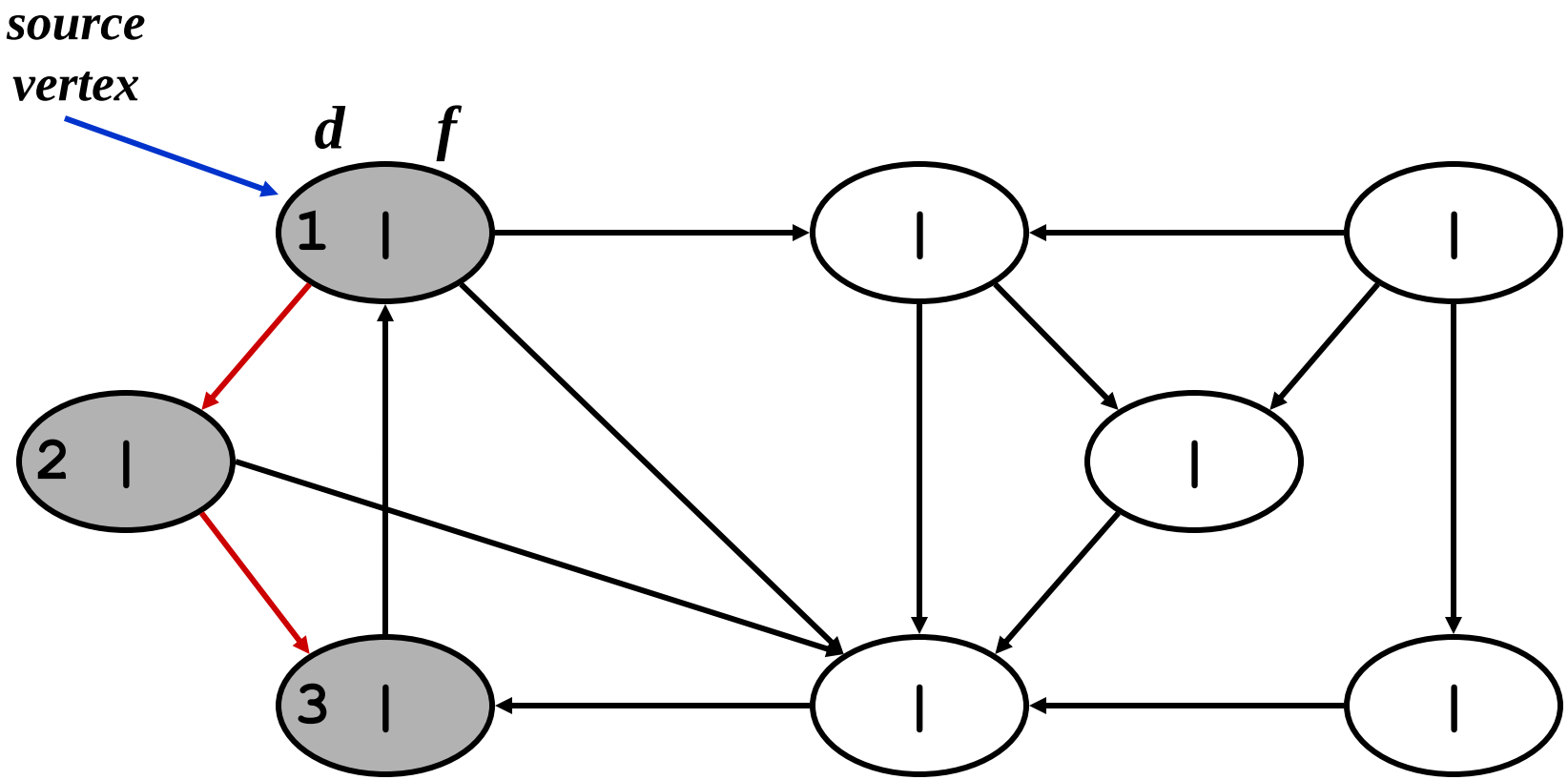
DFS Example



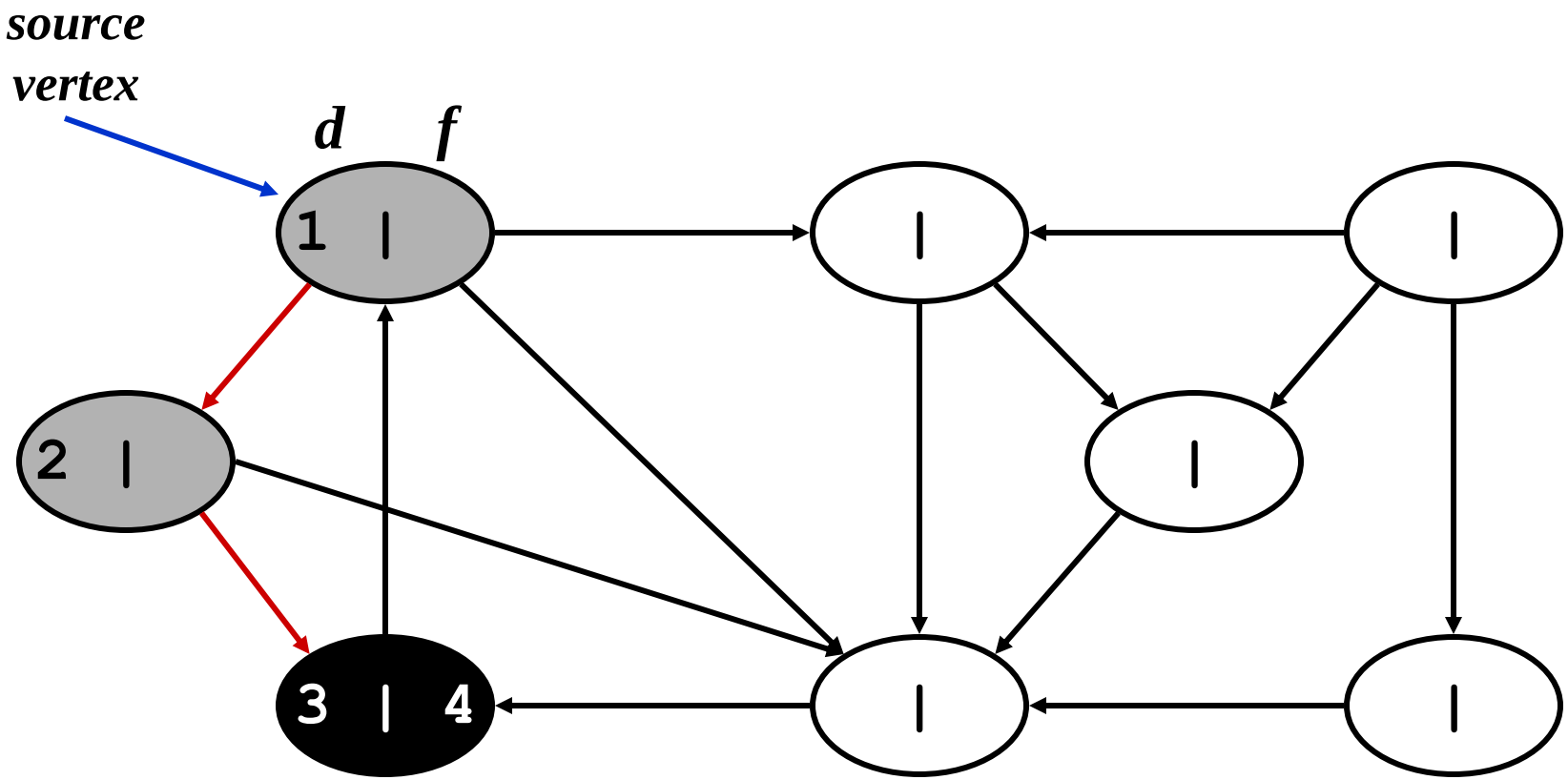
DFS Example



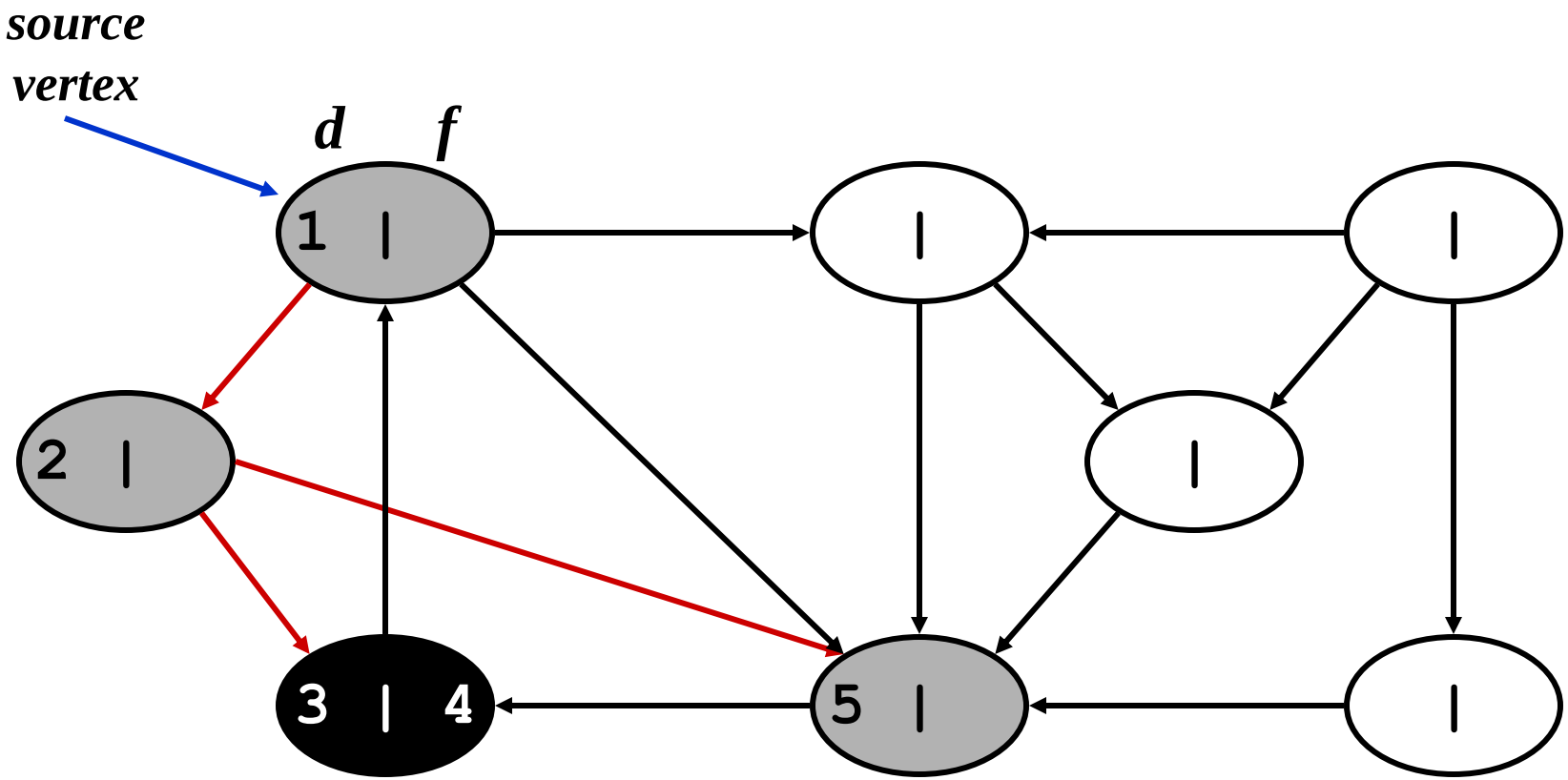
DFS Example



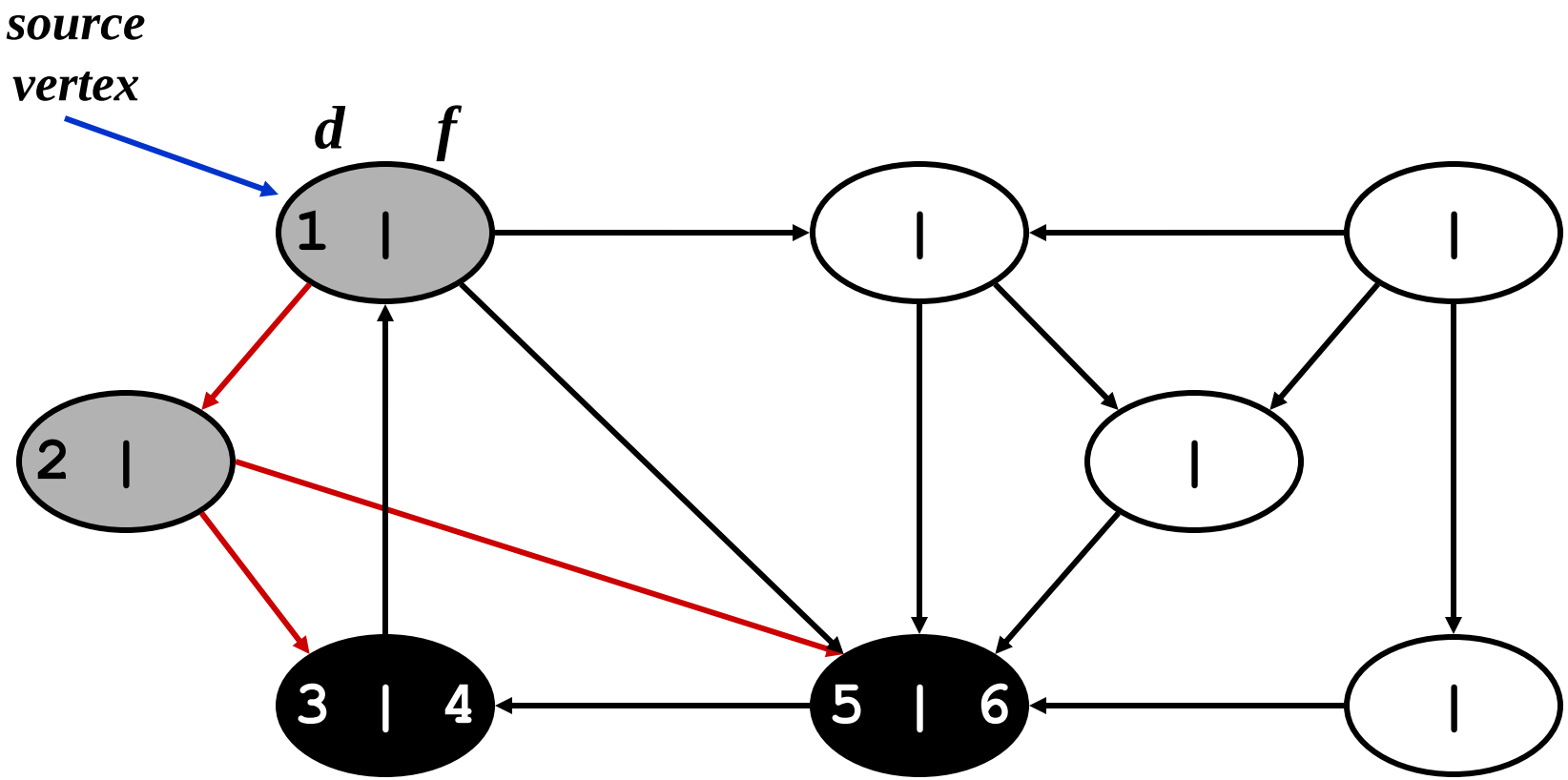
DFS Example



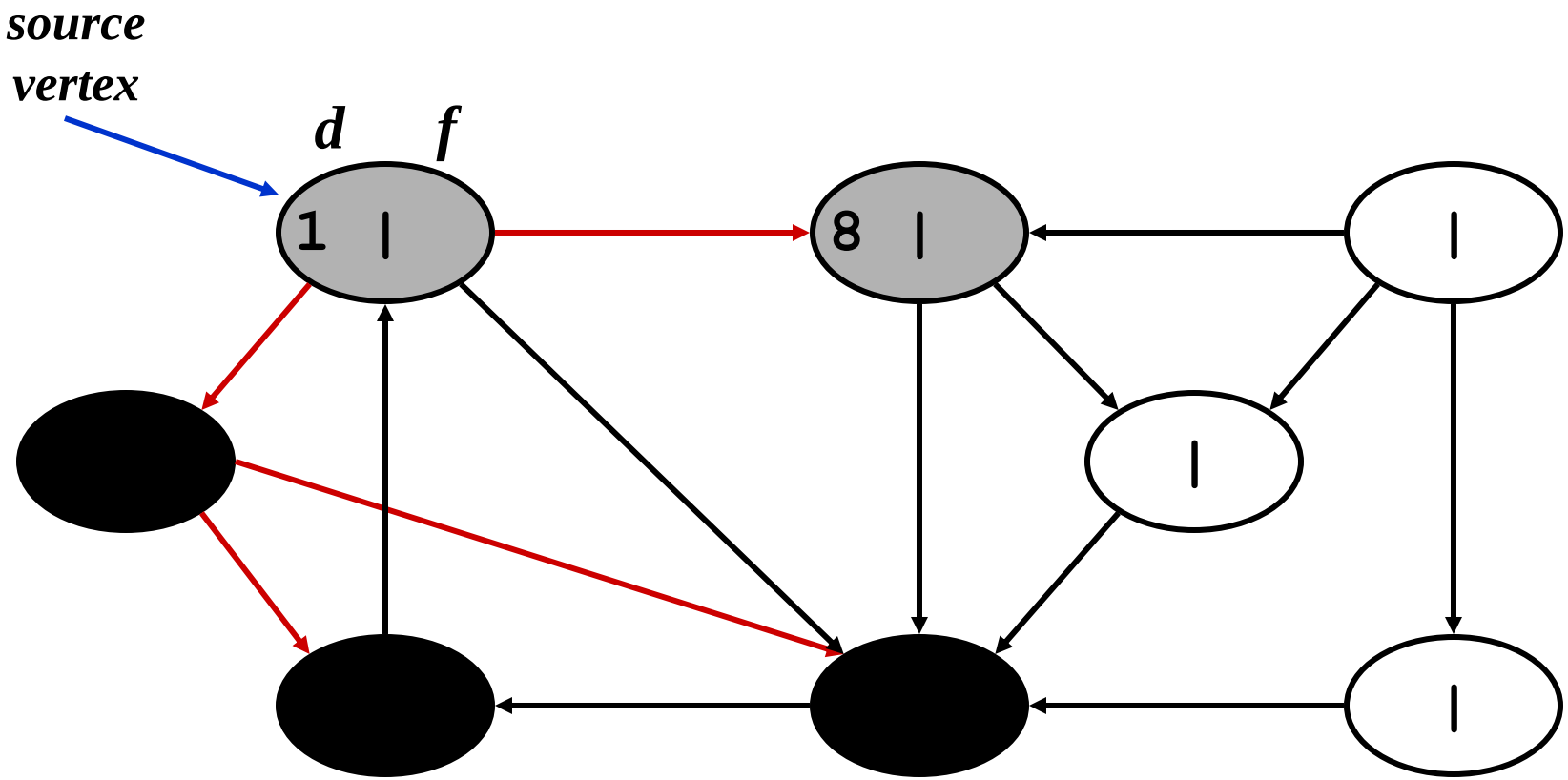
DFS Example



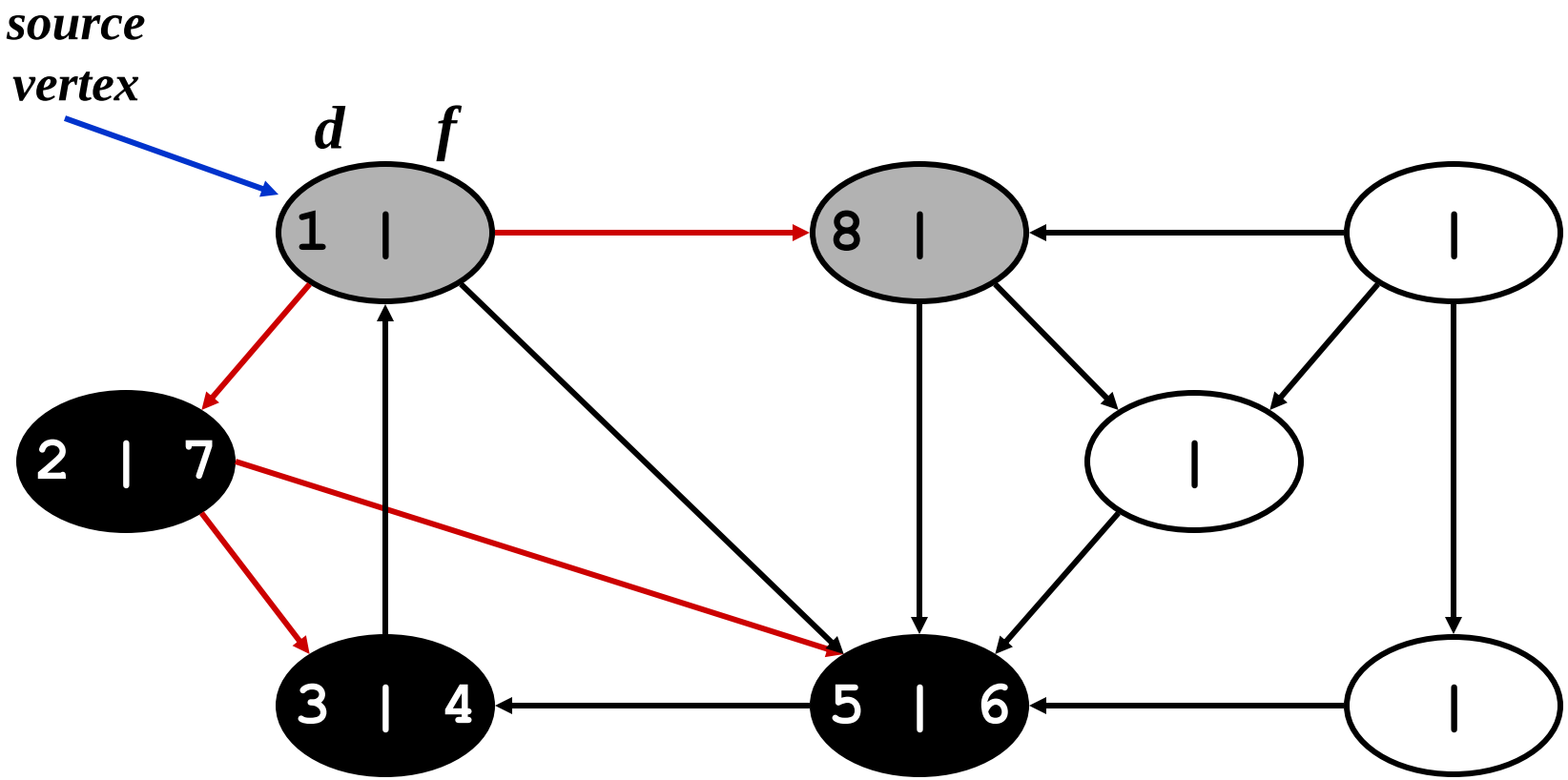
DFS Example



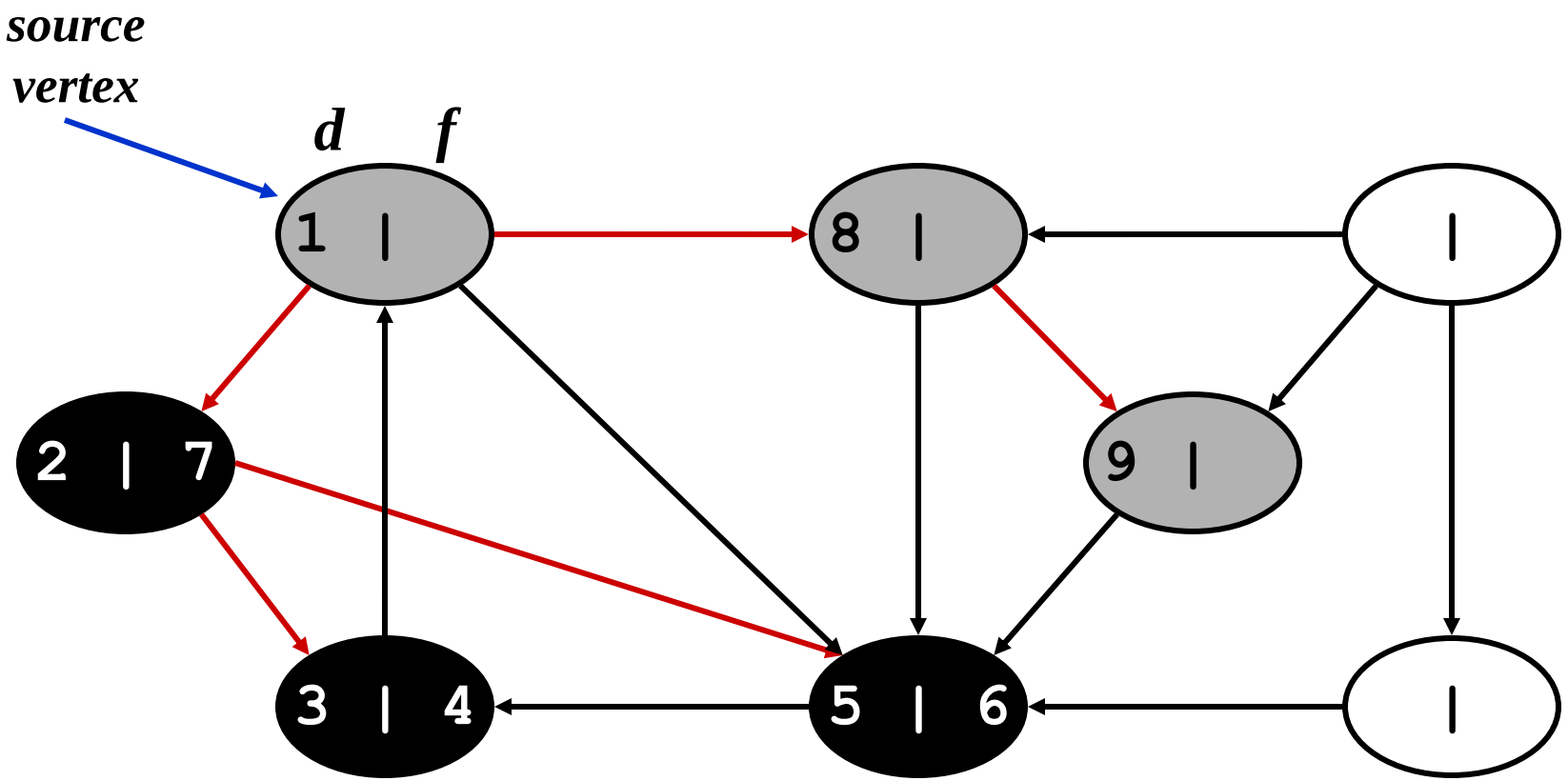
DFS Example



DFS Example

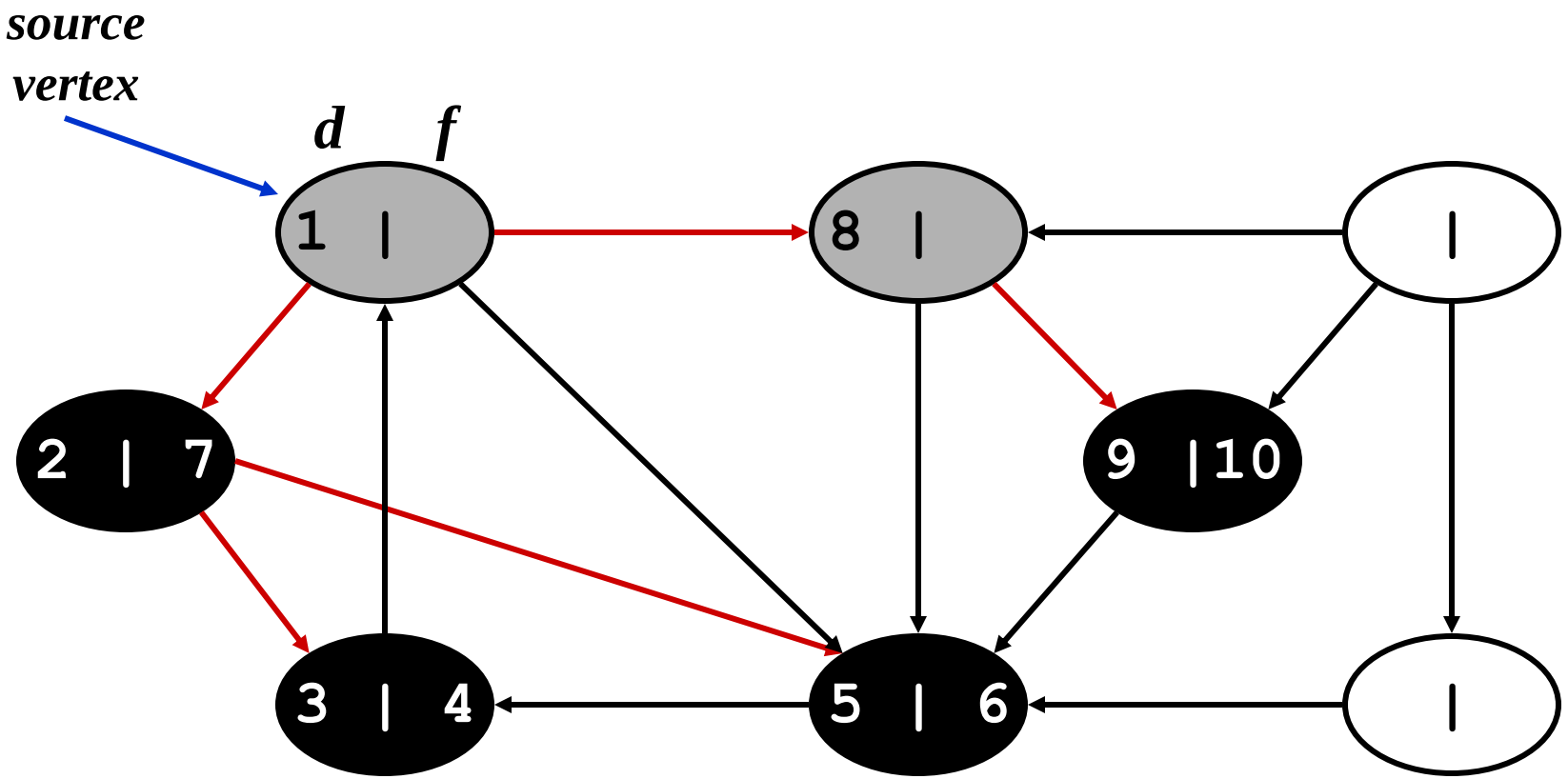


DFS Example

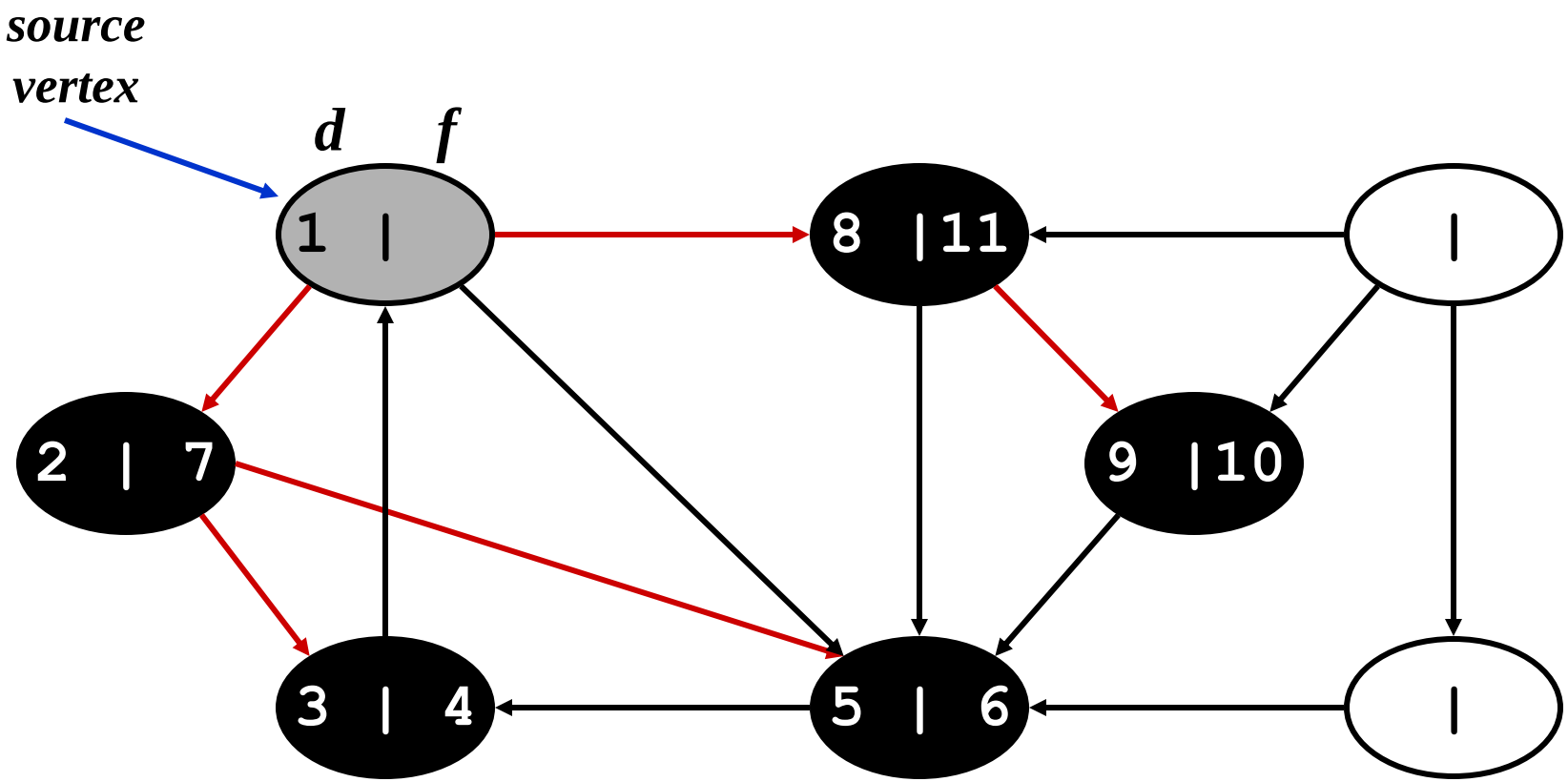


Τι εκφράζουν οι γκρι κορυφές?

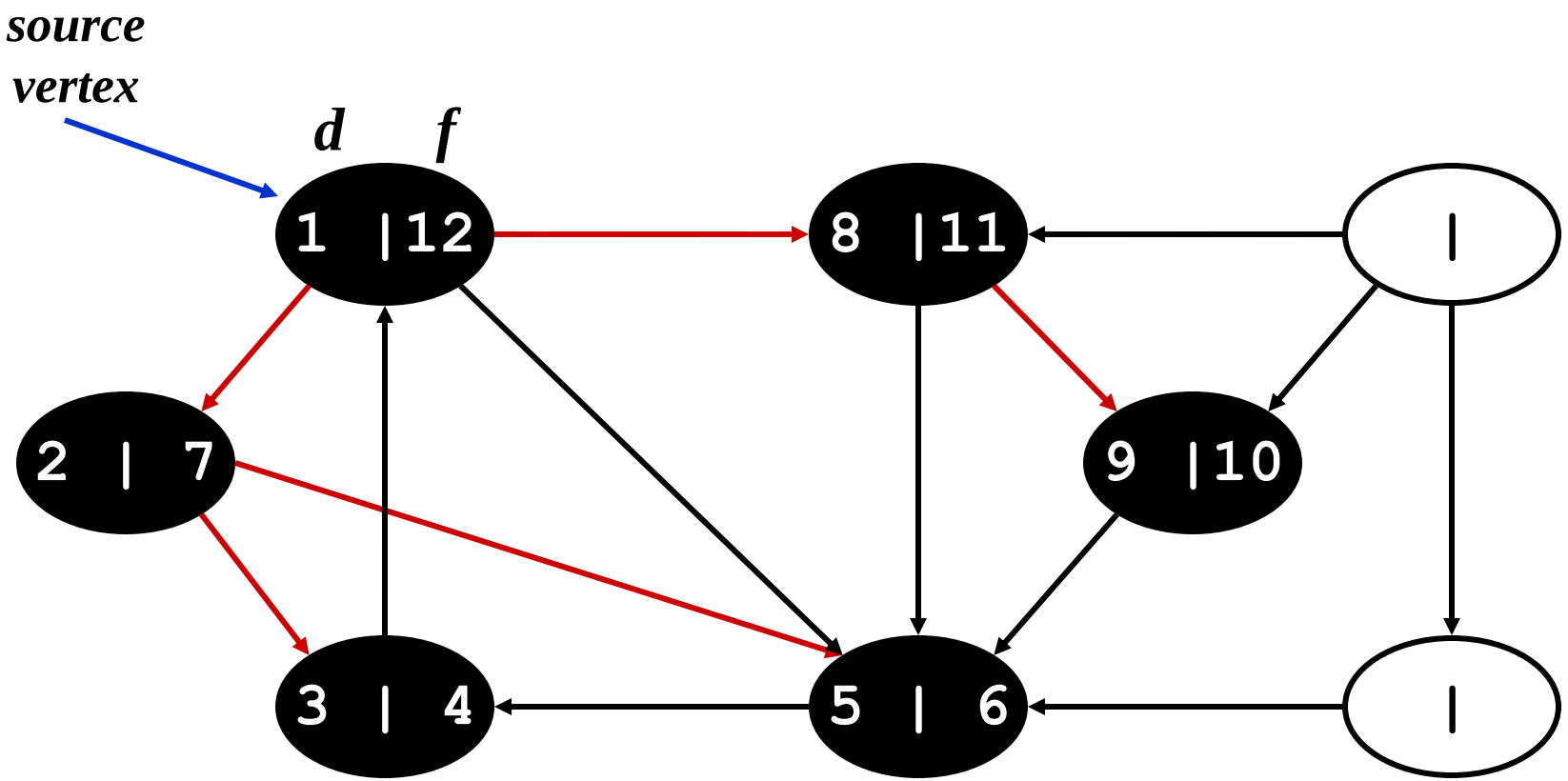
DFS Example



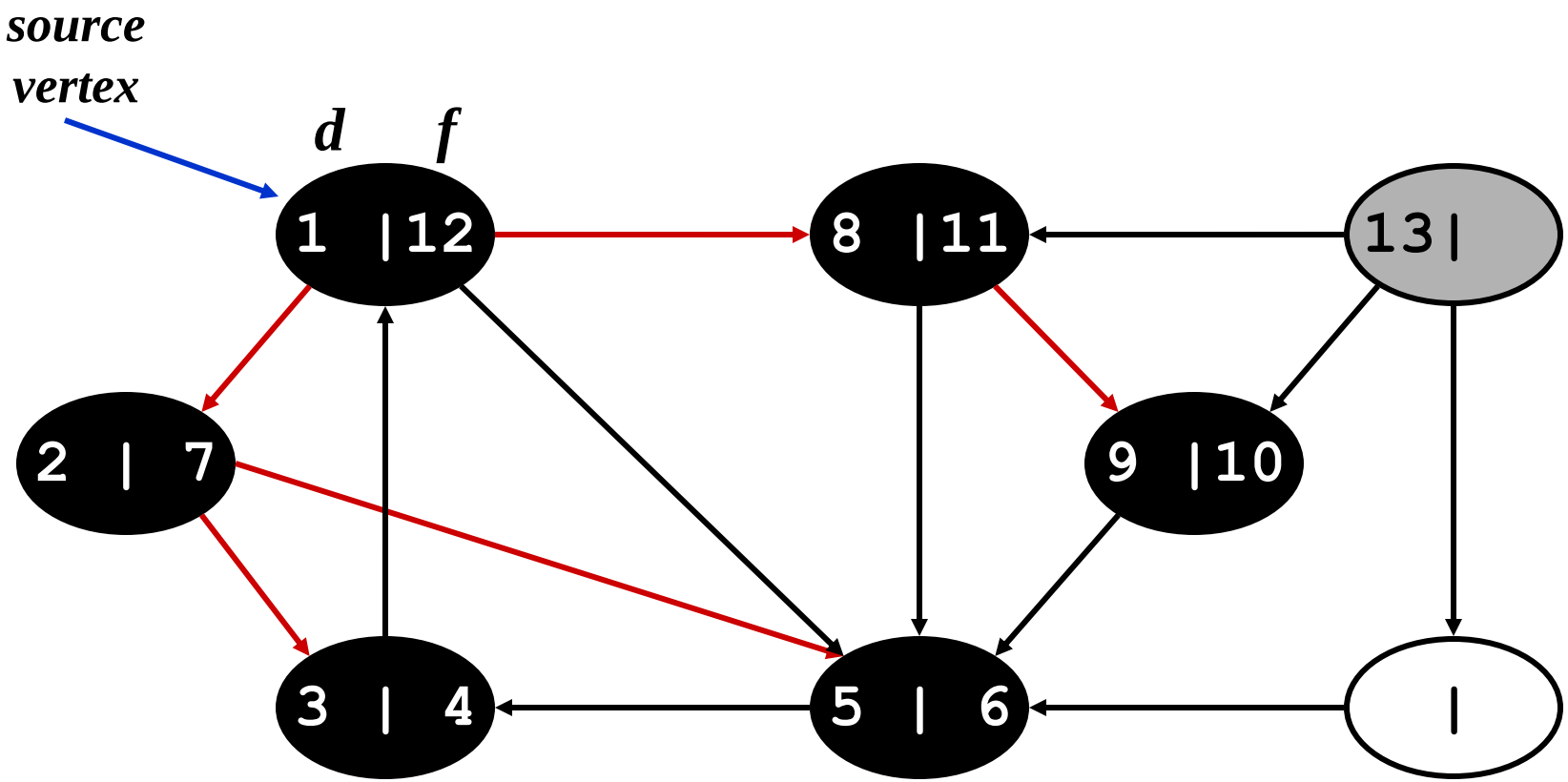
DFS Example



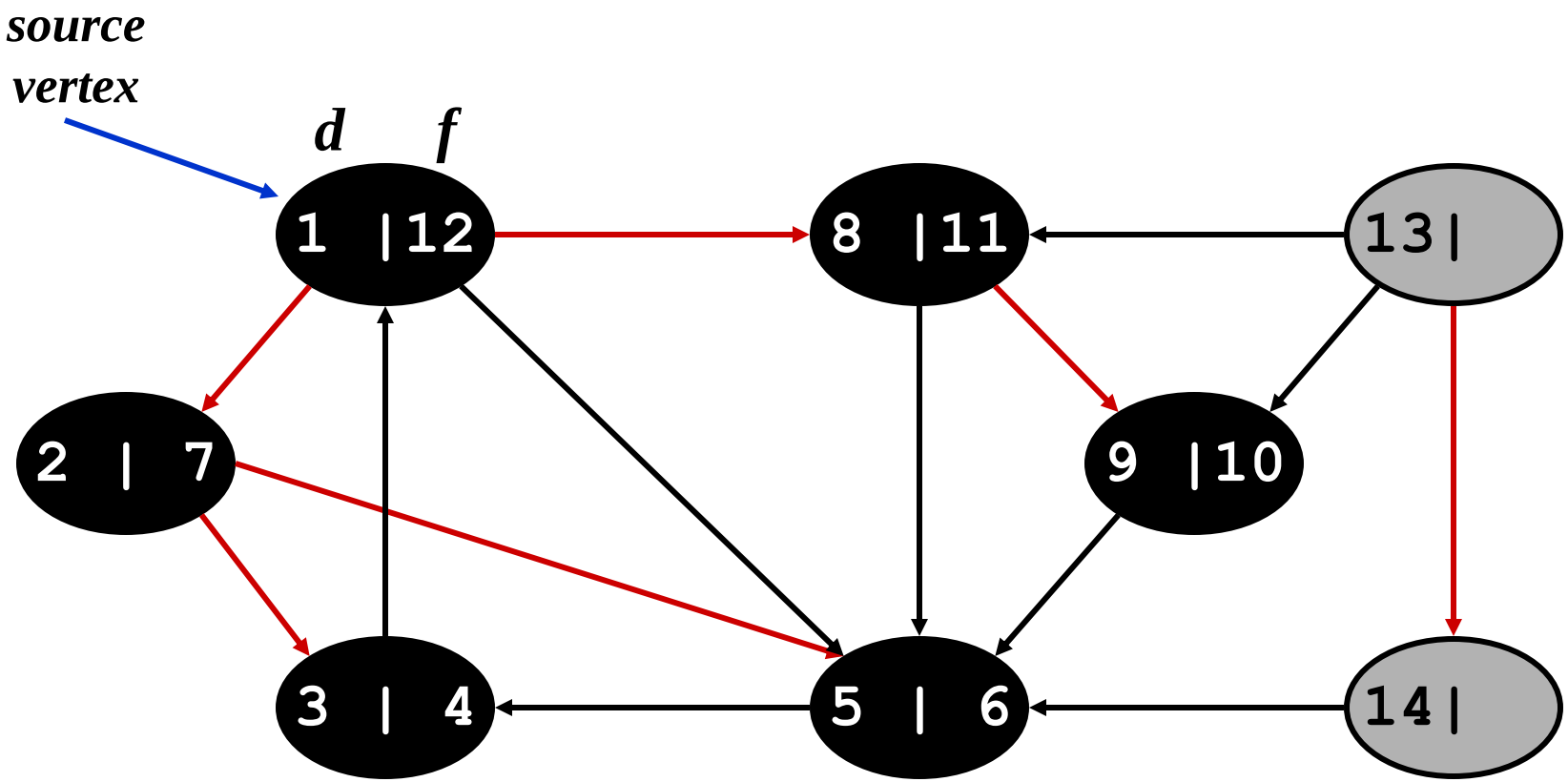
DFS Example



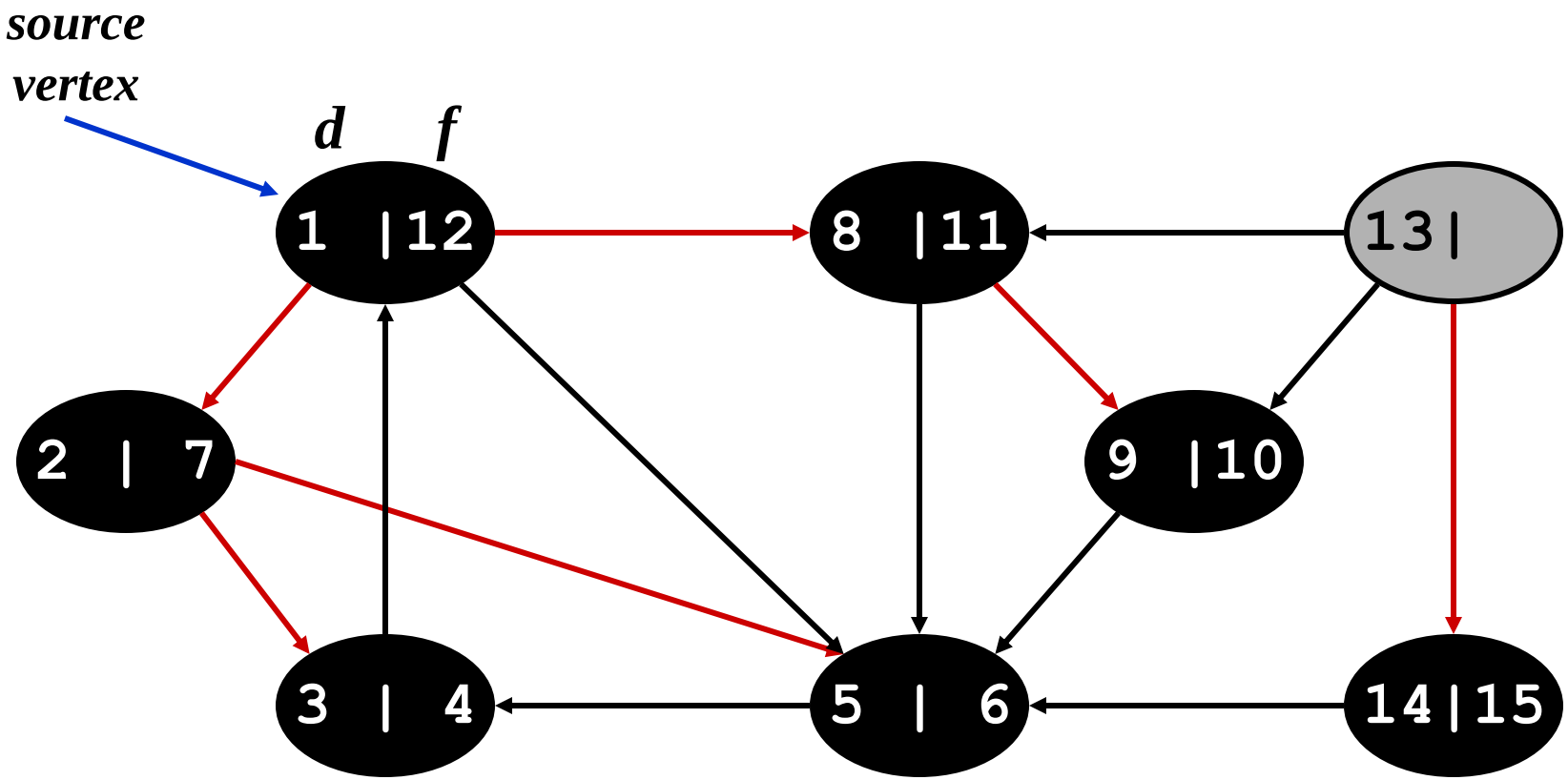
DFS Example



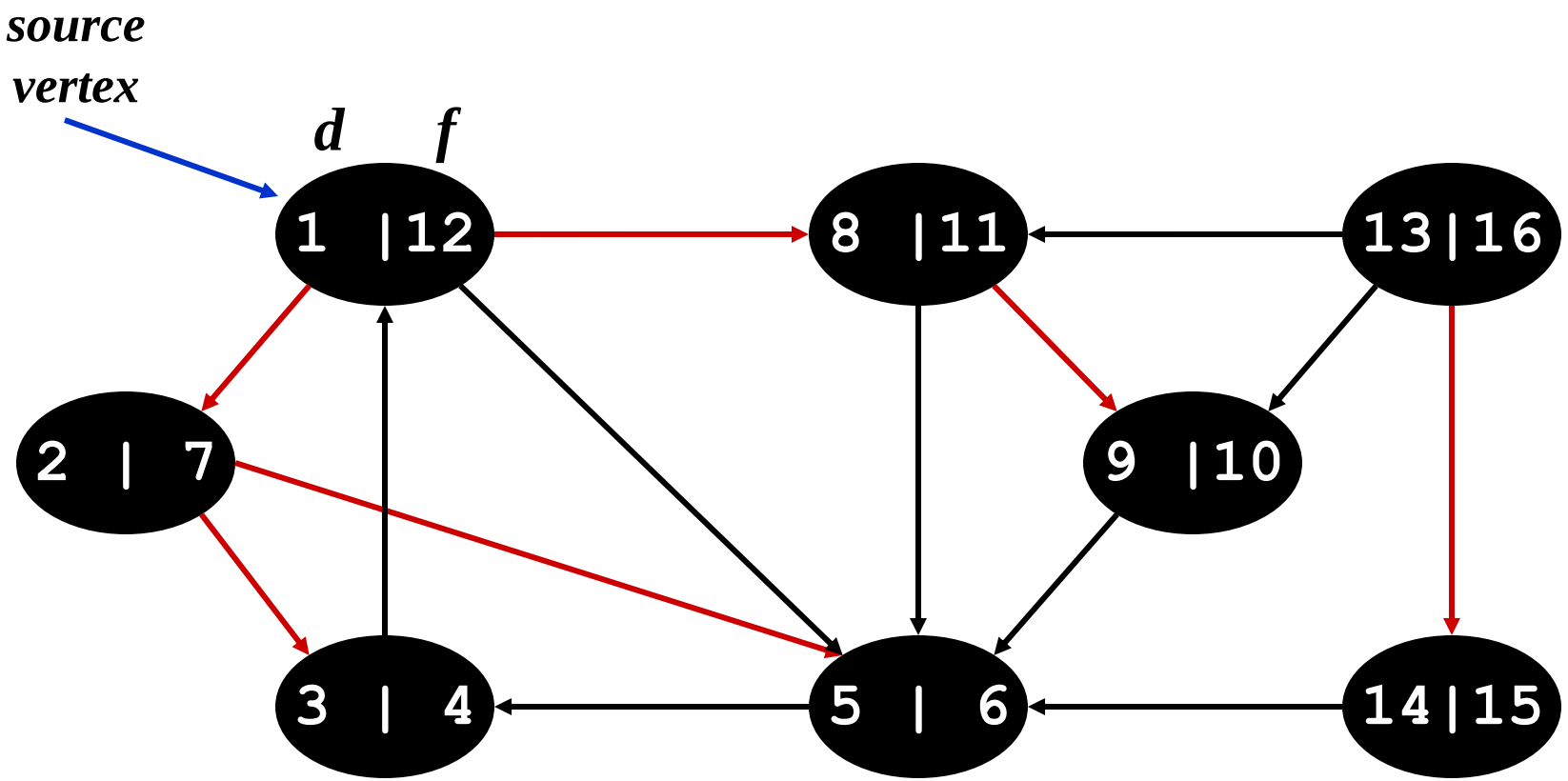
DFS Example



DFS Example



DFS Example



DFS : Ανάλυση

```
DFS(G)
1.    for each vertex u ∈ G->V
2.        color[u] = WHITE;
3.        p[u] = null
4.    time = 0;
5.    for each vertex u ∈ G->V
6.        if (color[u] == WHITE)
7.            DFS_Visit(u);
```

```
DFS_Visit(u)
1.    color[u] = GREY;
2.    time = time+1;
3.    d[u] = time;
4.    for each v ∈ Adj[u]
5.        if (color[v] == WHITE)
6.            p[v] = u;
7.            DFS_Visit(v);
8.    color[u] = BLACK;
9.    time = time+1;
10.   f[u] = time;
```

Ποιος θα είναι ο χρόνος εκτέλεσης?

DFS : Ανάλυση

```
DFS(G)
1.      for each vertex u ∈ G->V
2.          color[u] = WHITE;
3.          p[u] = null
4.      time = 0;
5.      for each vertex u ∈ G->V
6.          if (color[u] == WHITE)
7.              DFS_Visit(u);
```

χρόνος εκτέλεσης: $O(V^2)$ διότι η κλήσης της DFS_Visit για κάθε κορυφή, και η επανάληψη επί των Adj[] εκτελείται \|V\| φορές

```
DFS_Visit(u)
1.      color[u] = GREY;
2.      time = time+1;
3.      d[u] = time;
4.      for each v ∈ Adj[u]
5.          if (color[v] == WHITE)
6.              p[v] = u;
7.              DFS_Visit(v);
8.      color[u] = BLACK;
9.      time = time+1;
10.     f[u] = time;
```

DFS : Ανάλυση

```
DFS(G)
1.    for each vertex u ∈ G->V
2.        color[u] = WHITE;
3.        p[u] = null
4.    time = 0;
5.    for each vertex u ∈ G->V
6.        if (color[u] == WHITE)
7.            DFS_Visit(u);
```

```
DFS_Visit(u)
1.    color[u] = GREY;
2.    time = time+1;
3.    d[u] = time;
4.    for each v ∈ Adj[u]
5.        if (color[v] == WHITE)
6.            p[v] = u;
7.            DFS_Visit(v);
8.    color[u] = BLACK;
9.    time = time+1;
10.   f[u] = time;
```

Αλλά υπάρχει ένα αυστηρότερο φράγμα. Πόσες φορές θα κληθεί η DFS_Visit()?

DFS : Ανάλυση

Στην DFS(G) οι βρόχοι στις γραμμές 1-3 και στις γραμμές 5-7 απαιτούν χρόνο $\Theta(V)$ πέραν του χρόνου εκτέλεσης των κλήσεων της DFS_Visit.

Η διαδικασία της DFS_Visit καλείται ακριβώς μια φορά για κάθε κορυφή $v \in V$, δεδομένου ότι εκτελείται μόνο για λευκές κορυφές και το πρώτο πράγμα που κάνει είναι να χρωματίσει την κορυφή εισόδου σε γκρι.

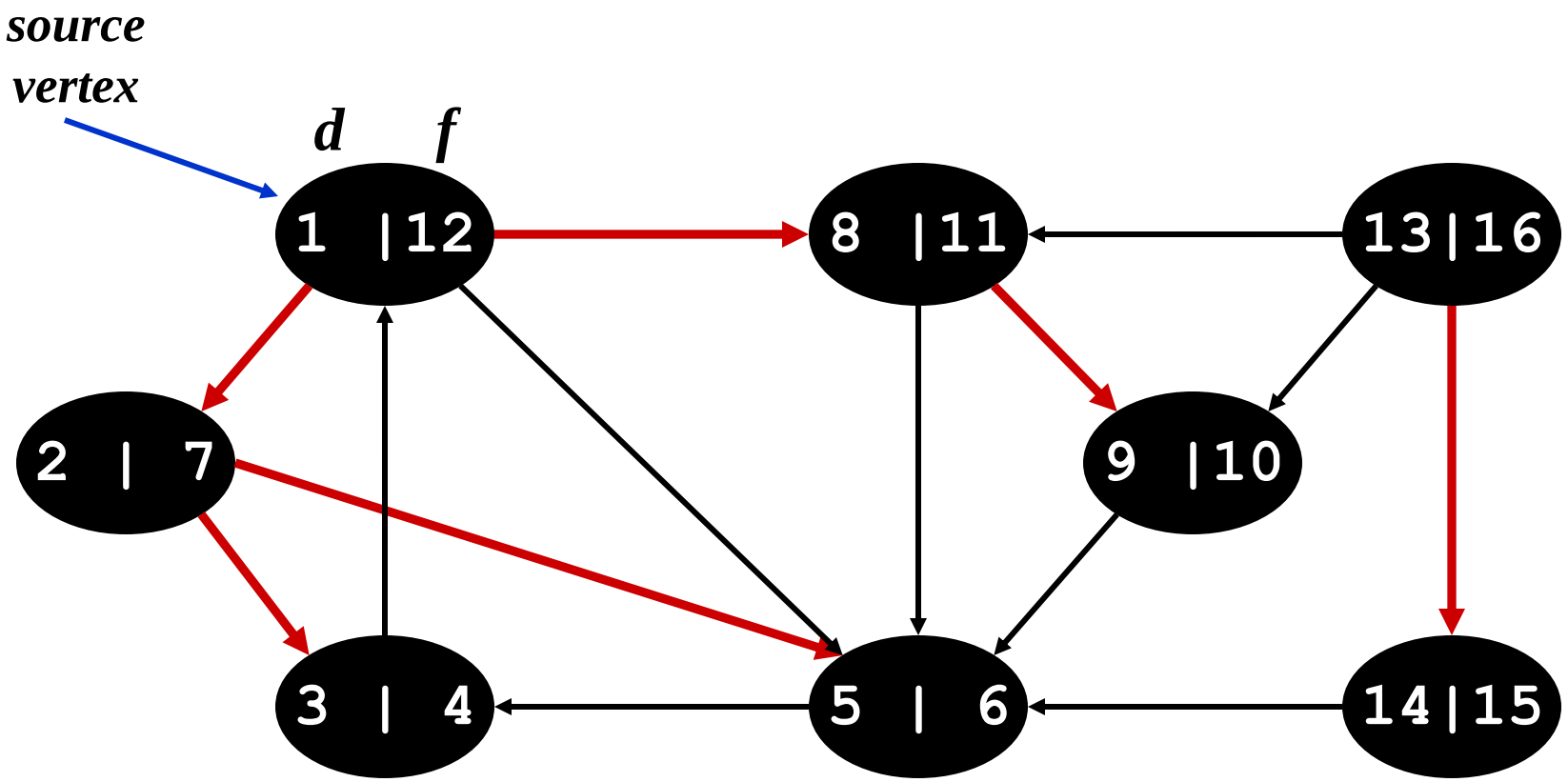
Κατά τη διάρκεια μιας μεμονωμένης εκτέλεσης της DFS_Visit(v), ο βρόχος στις γραμμές 4-7 εκτελείται $|Adj[v]|$ φορές. Δεδομένου ότι ισχύει:

$\sum_{v \in V} |Adj[v]| = \Theta(E)$ το συνολικό κόστος εκτέλεσης των γραμμών 4-7 της DFS_Visit είναι $\Theta(E)$. Επομένως, ο χρόνος εκτέλεσης της DFS είναι $\Theta(V + E)$.

DFS: Είδη ακμών

- DFS εισάγει μια σημαντική διάκριση μεταξύ των ακμών του γραφήματος :
  - *δενδρικές ακμές (Tree edge)*: συναντά μια (νέα) λευκή κορυφή
    - Οι δενδρικές ακμές ανήκουν στο καθοδικό δάσος G
    - Η ακμή (*u*, *v*) είναι δενδρική αν η *v* εντοπίζεται κατά την εξερεύνηση της (*u*, *v*).
    - *Είναι δυνατόν οι δενδρικές ακμές να σχηματίζουν κύκλους? Ναι ή όχι και γιατί ?*

DFS Example

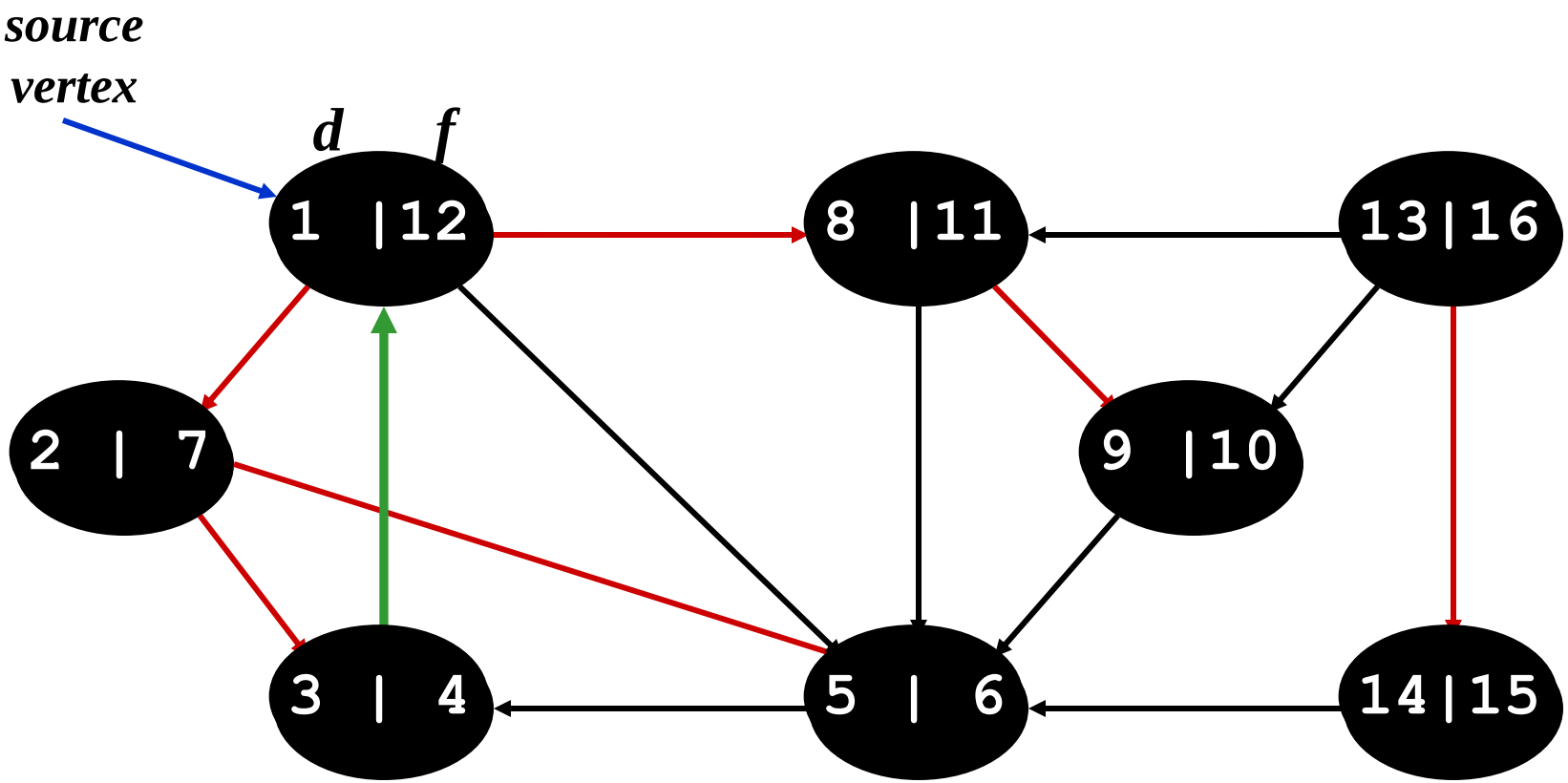


Δενδρικές ακμές
(Tree edges)

DFS: Είδη ακμών

- DFS εισάγει μια σημαντική διάκριση μεταξύ των ακμών του αρχικού γραφήματος :
  - *Δενδρικές ακμές (Tree edge)*: όταν συναντά μια νέα (λευκή) κορυφή
  - *Ανιούσες ακμές (Back edge)*: από απόγονο σε πρόγονο
    - Συναντά μια γκρι κορυφή (γκρι σε γκρι)

DFS Example



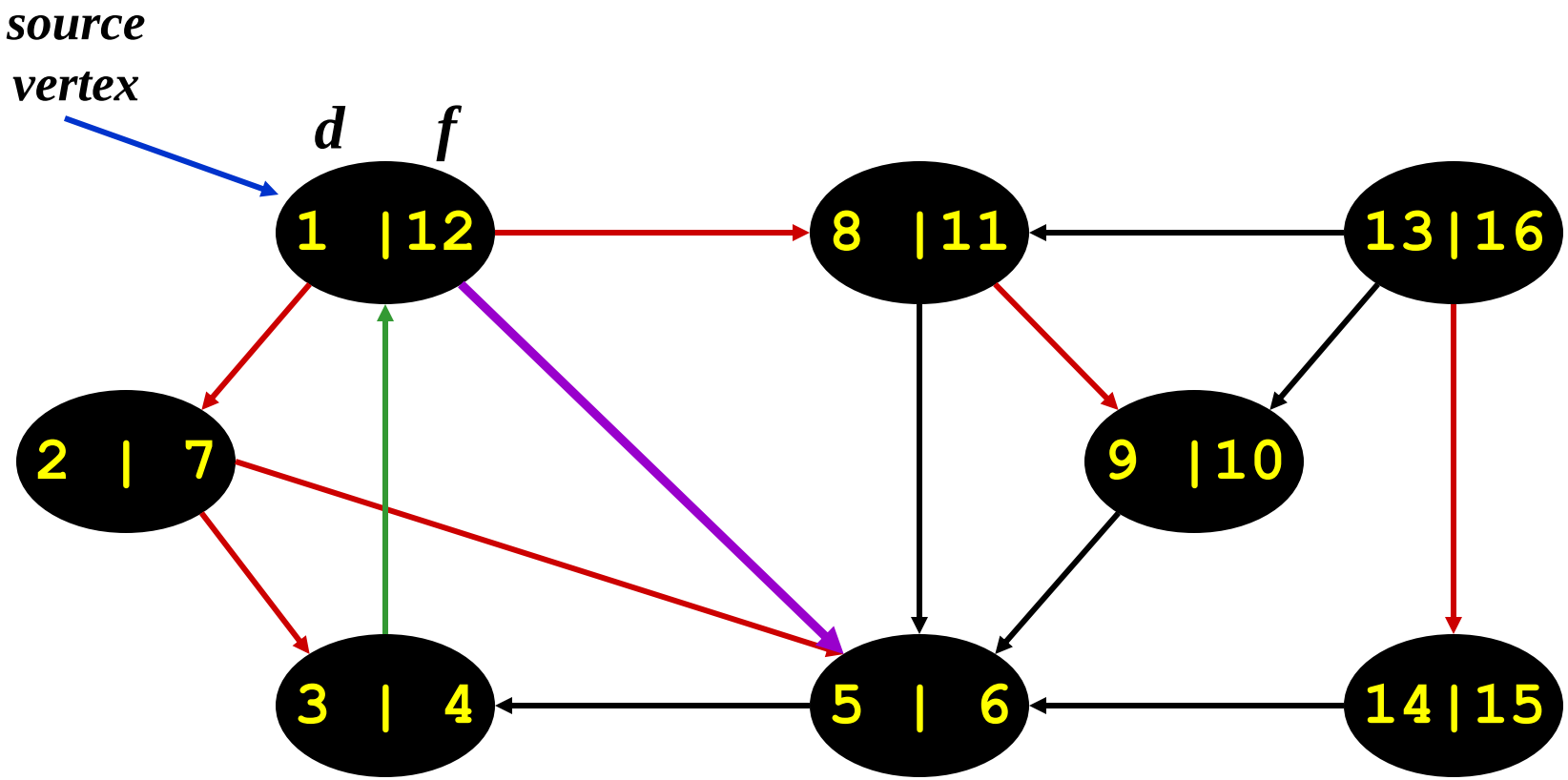
Δενδρικές ακμές
(Tree edges)

Ανιούσες ακμές
(Back edges)

DFS: Είδη ακμών

- DFS εισάγει μια σημαντική διάκριση μεταξύ των ακμών του αρχικού γραφήματος :
  - *Δενδρικές ακμές (Tree edge)*: όταν συναντά μια νέα (λευκή) κορυφή
  - *Ανιούσες ακμές (Back edge)*: από απόγονο σε πρόγονο
  - *Κατιούσες ακμές (Forward edge)*: από πρόγονο σε from απόγονο
    - Όχι ακμή δένδρου
    - Από γκρι κορυφή σε μαύρη

DFS Example



Δενδρικές ακμές (Tree edges)

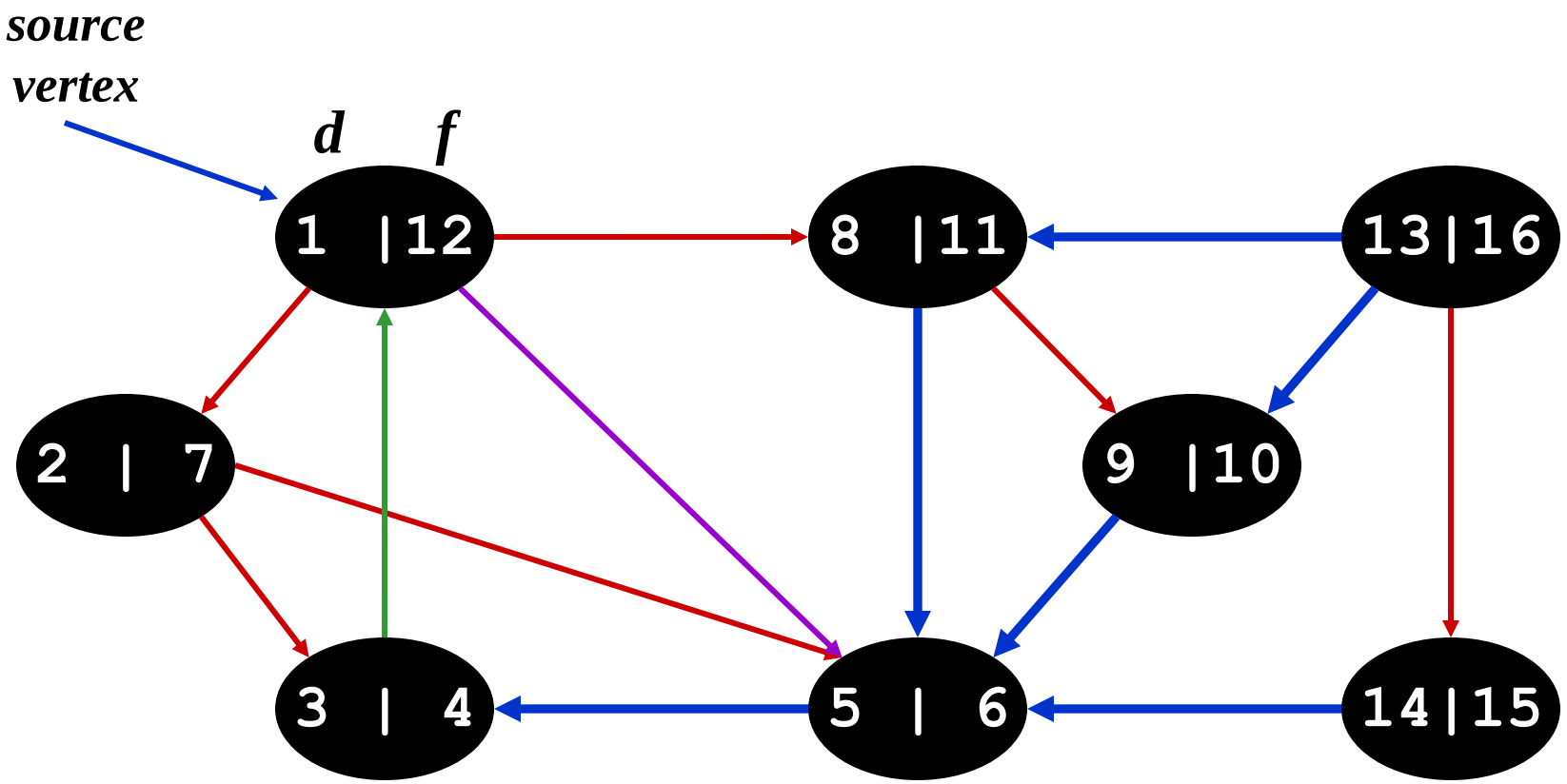
Ανιούσες ακμές (Back edges)

Κατιούσες ακμές Forward edges

DFS: Είδη ακμών

- DFS εισάγει μια σημαντική διάκριση μεταξύ των ακμών του αρχικού γραφήματος :
  - *Δενδρικές ακμές (Tree edge)*: όταν συναντά μια νέα (λευκή) κορυφή
  - *Ανιούσες ακμές (Back edge)*: από απόγονο σε πρόγονο
  - *Κατιούσες ακμές (Forward edge)*: από πρόγονο σε απόγονο
  - *Εγκάρσιες ακμές (Cross edge)* : συνδέουν κορυφές στο ίδιο καθοδικό δένδρο εφόσον οι κορυφές δεν συνδέονται με σχέση προγόνου-απογόνου ή διαφορετικά καθοδικά δένδρα
    - Από γκρι σε μαύρη ακμή

DFS Example



Δενδρικές ακμές (Tree edges)

Ανιούσες ακμές (Back edges)

Κατιούσες ακμές Forward edges

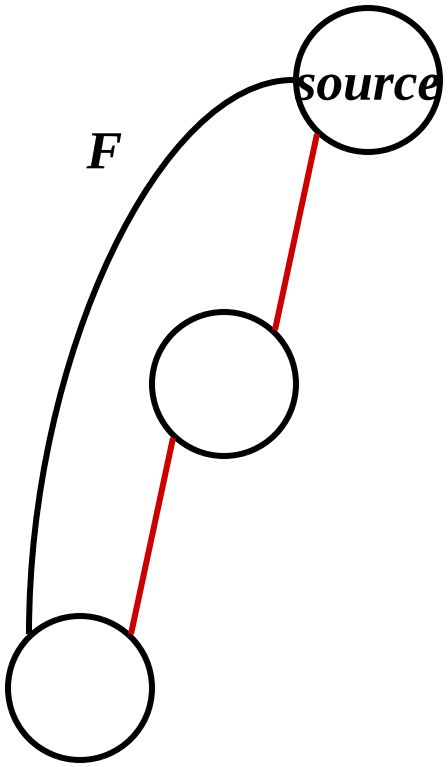
Εγκάρσιες ακμές Cross edges

DFS: Είδη ακμών

- DFS εισάγει μια σημαντική διάκριση μεταξύ των ακμών του αρχικού γραφήματος :
  - *Δενδρικές ακμές (Tree edge)*: όταν συναντά μια νέα (λευκή) κορυφή
  - *Ανιούσες ακμές (Back edge)*: από απόγονο σε πρόγονο
  - *Κατιούσες ακμές (Forward edge)*: από πρόγονο σε απόγονο
  - *Εγκάρσιες ακμές (Cross edge)* : μεταξύ δένδρου ή υποδένδρων
  - Σημείωση: δενδρικές & ανιούσες ακμές είναι σημαντικές; Οι περισσότεροι αλγόριθμοι δεν διακρίνουν τις κατιούσες και εγκάρσιες ακμές

DFS: Είδη ακμών

- Αν το G είναι μη προσανατολισμένο, ο DFS παράγει μόνο δενδρικές και ανιούσες ακμές



DFS: Είδη ακμών

- **Θεώρημα** : Σε μια σε βάθος διερεύνηση ενός μη προσανατολισμένου γραφήματος *G*, κάθε ακμή του *G* είναι είτε δενδρική είτε ανιούσα
- **Απόδειξη**: Εστω (*u*, *v*) μια τυχούσα ακμή του *G*, και ας υποθέσουμε ότι $d[u] < d[v]$. Σ' αυτή την περίπτωση η *v* θα πρέπει να εντοπίζεται και να περατώνεται πριν να περατωθεί η *u* (ενόσω η *u* είναι γκρι), αφού ανήκει στη λίστα γειτνίασης τη *u*.

  Αν η ακμή (*u*, *v*) εξερευνάται πρώτα κατά τη κατεύθυνση από την *u* προς την *v*, τότε μέχρι εκείνη τη χρονική στιγμή η *v* είναι μη εντοπισμένη (λευκή) διότι διαφορετικά η συγκεκριμένη ακμή θα είχε ήδη εξερευνηθεί κατά την κατεύθυνση από την v προς την *u*. Επομένως η ακμή καθίσταται δενδρική.

  Αν η ακμή (*u*, *v*) εξερευνάται πρώτα κατά τη κατεύθυνση από την v προς την *u* , τότε είναι ανιούσα, δεδομένου ότι η *u* είναι ακόμη γκρι τη στιγμή που ακμή εξερευνάται για πρώτη φορά.

DFS και Κύκλοι Γραφήματος

- Θεώρημα: Ένα προσανατολισμένο γράφημα είναι *άκυκλο* (*acyclic)* αν και μόνο αν η DFS διερεύνηση του G δε δημιουργεί ανιούσες ακμές
  - Αν είναι άκυκλο, δε υπάρχουν ανιούσες ακμές (ανιούσα ακμή συνεπάγεται κύκλο)
  - Αν δεν υπάρχουν ανιούσες ακμές τότε είναι άκυκλο
    - Όχι ανιούσες ακμές συνεπάγεται μόνο δενδρικές ακμές
    - Μόνο δενδρικές ακμές συνεπάγεται ότι έχουμε δένδρο ή δάσος το οποίο είναι εξ ορισμού άκυκλο
- Άρα εφαρμόζοντας DFS διερεύνηση προσδιορίζουμε αν ένα γράφημα έχει κύκλο

DFS και Κύκλοι Γραφήματος

```
DFS(G)
1.      for each vertex u ∈ G->V
2.          color[u] = WHITE;
3.          p[u] = null
4.      time = 0;
5.      for each vertex u ∈ G->V
6.          if (color[u] == WHITE)
7.              DFS_Visit(u);
```

```
DFS_Visit(u)
1.      color[u] = GREY;
2.      time = time+1;
3.      d[u] = time;
4.      for each v ∈ Adj[u]
5.          if (color[v] == WHITE)
6.              p[v] = u;
7.              DFS_Visit(v);
8.      color[u] = BLACK;
9.      time = time+1;
10.     f[u] = time;
```

Πως μπορεί να τροποποιηθεί ο παραπάνω αλγόριθμος ώστε να προσδιορίζει κύκλο?

DFS και Κύκλοι Γραφήματος

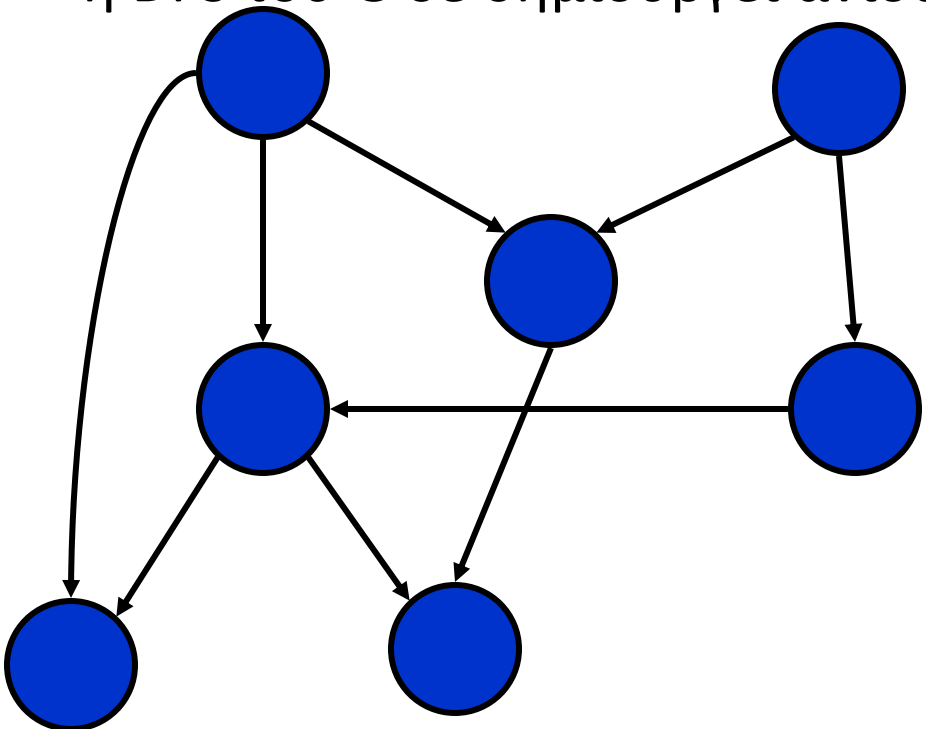
```
DFS(G)
1.      for each vertex u ∈ G->V
2.          color[u] = WHITE;
3.          p[u] = null
4.      time = 0;
5.      for each vertex u ∈ G->V
6.          if (color[u] == WHITE)
7.              DFS_Visit(u);
```

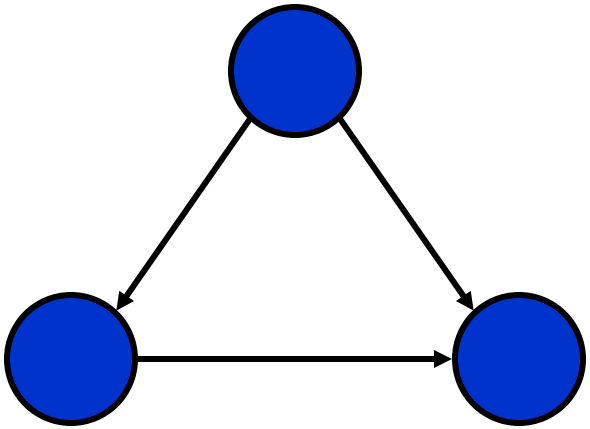
```
DFS_Visit(u)
1.      color[u] = GREY;
2.      time = time+1;
3.      d[u] = time;
4.      for each v ∈ Adj[u]
5.          if (color[v]==GREY and p[u] ≠v)
6.              return the cycle (u, p[u]),
                        (p[u], p[p[u]], … (v, u)
7.          if (color[v] == WHITE)
8.              p[v] = u;
9.              DFS_Visit(v);
10.     color[u] = BLACK;
11.     time = time+1;
12.     f[u] = time;
```

Προσθήκη «μπλε» εντολών στον DFS αλγόριθμο ώστε να προσδιορίζει κύκλο

Προσανατολισμένο Ακυκλο Γράφημα

- Ένα *προσανατολισμένο άκυκλο γράφημα* (*directed acyclic graph)* ή *DAG* είναι ένα προσανατολισμένο γράφημα χωρίς προσανατολισμένους κύκλους.
- Ένα προσανατολισμένο γράφημα G είναι άκυκλο αν και μόνο η DFS του G δε δημιουργεί ανιούσες ακμές.

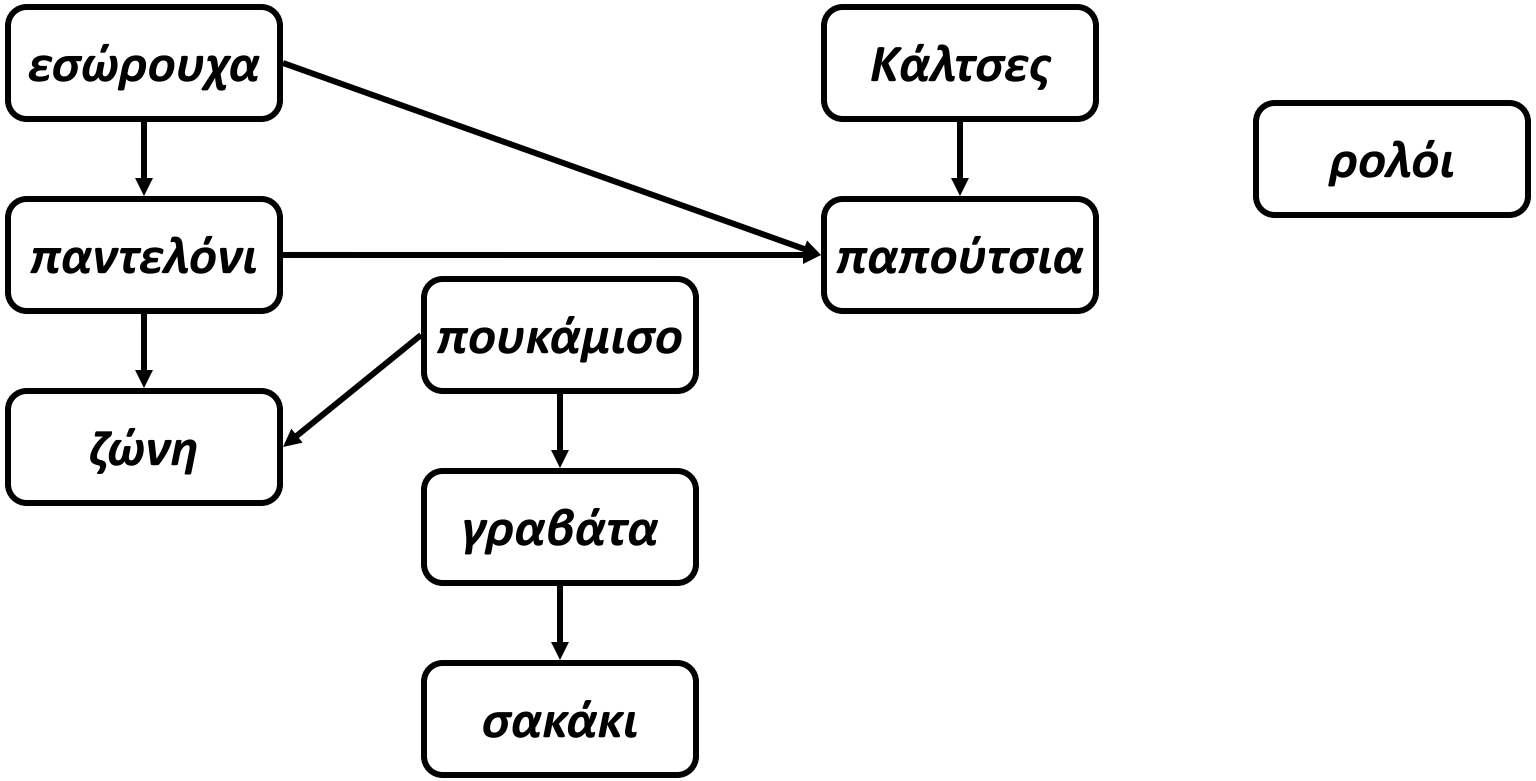




Τοπολογική Ταξινόμηση

- *Τοπολογική ταξινόμηση (Topological sort)* ενός DAG:
  - Γραμμική διάταξη όλων των κορυφών του γραφήματος *G* έτσι ώστε η κορυφή *u* να εμφανίζεται πριν από την κορυφή αν η ακμή $(u, v) \in G$
- Τα προσανατολισμένα άκυκλα γραφήματα χρησιμοποιούνται σε πολλές εφαρμογές που απαιτούν να υποδειχθεί μια σειρά προτεραιότητας σε ορισμένα γεγονότα.
- Παράδειγμα από την καθημερινή ζωή: *ντύσιμο*

Πρωινό ντύσιμο



Αλγόριθμος Τοπολογικής Ταξινόμησης

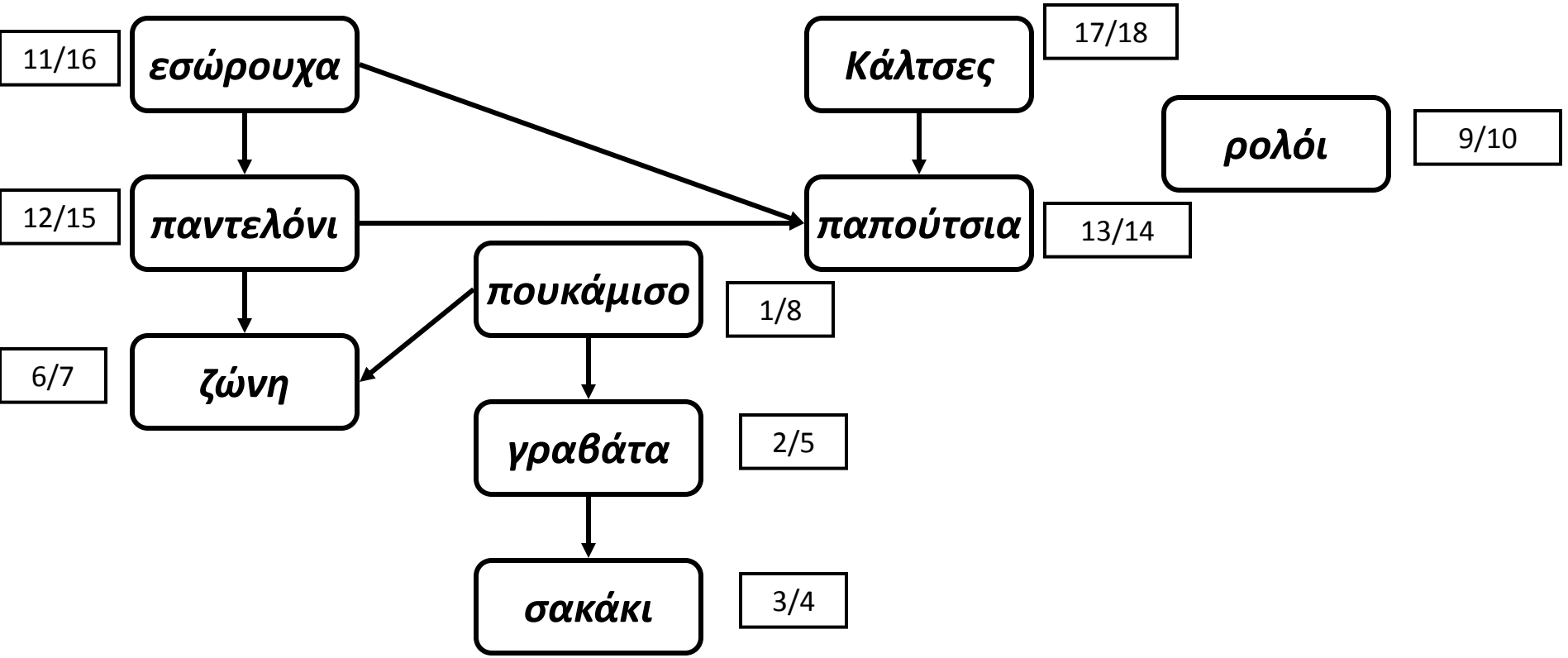
```
Topological-Sort()
{
```

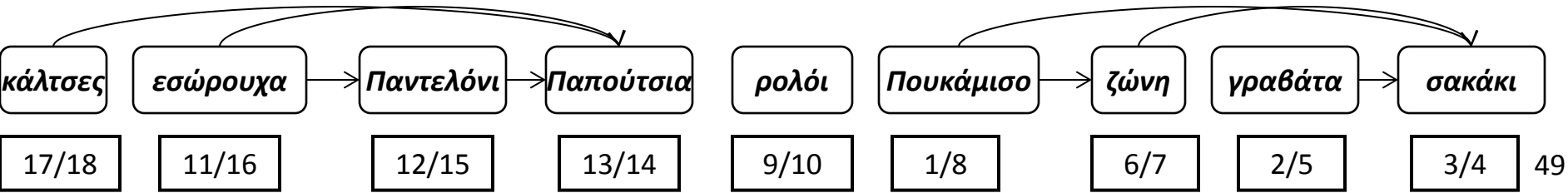
1. Καλούμε την DFS(G) για τον υπολογισμό του χρόνου περάτωσης $f[v]$ για κάθε κόμβο v
2. Μετά την περάτωση κάθε κόμβου, τον τοποθετούμε επικεφαλής μιας αλυσίδας (λίστας)
3. Επιστροφή η αλυσίδα κόμβων

```
}
```

- Χρόνος: $\Theta(V+E)$ δεδομένου ότι η DFS διερεύνηση απαιτεί χρόνο $\Theta(V+E)$ ενώ η εισαγωγή καθεμιάς από τις $|V|$ κορυφές στην κορυφή της αλυσίδας απαιτεί χρόνο $O(1)$.

Πρωινό ντύσιμο – τοπολογική ταξινόμηση





Τέλος Ενότητας