U.F.R SCIENCES ET TECHNIQUES
Département d'Informatique
B.P. 1155
64013 PAU CEDEX
Téléphone secrétariat : 05.59.40.79.64
Télécopie : 05.59.40.76.54

SUR LE PARCOURS DE GRAPHE

I- PARCOURS EN PROFONDEUR
II- PARCOURS EN LARGEUR

Introduction

La plupart de problèmes utilisant le modèle de graphe nécessitent un examen **exhaustif** :

- des sommets
- et/ou des arcs ou arêtes du graphe.

On va étudier deux types de parcours correspondant à des **stratégies d'exploration** très générales:

- le parcours en profondeur,
- le parcours en largeur.

Stratégie d’exploration en profondeur

Elle consiste :

- à partir d’**un sommet donné**,
- à suivre un chemin **le plus loin possible**,
- puis à faire des **retours en arrière** pour **reprendre** tous les chemins non explorés.

Stratégie d’exploration en largeur

Elle consiste :

- à partir d’un **sommet donné**,
- à explorer le graphe « *niveau par niveau* ».

I- Parcours en profondeur

1- Algorithme

On associe au graphe un tableau de **marquage** M **booléen** défini tel que:

M(i) = vrai   si le sommet i a été rencontré,

M(i) = faux   sinon.

Au départ, tous les sommets sont non marqués:

$\forall\ i \in [1,taille]$ M(i) = faux

L'algorithme consiste:

- à **choisir** un sommet de départ s,

- à le **marquer** : M(s) := vrai,

- à **suivre un chemin issu de s**, aussi loin que possible en **marquant** les sommets rencontrés,

- arrivé en fin de chemin, on **revient au dernier choix fait** et on parcourt un autre chemin.

/* Parcours de tous les chemins issus d'un sommet s */

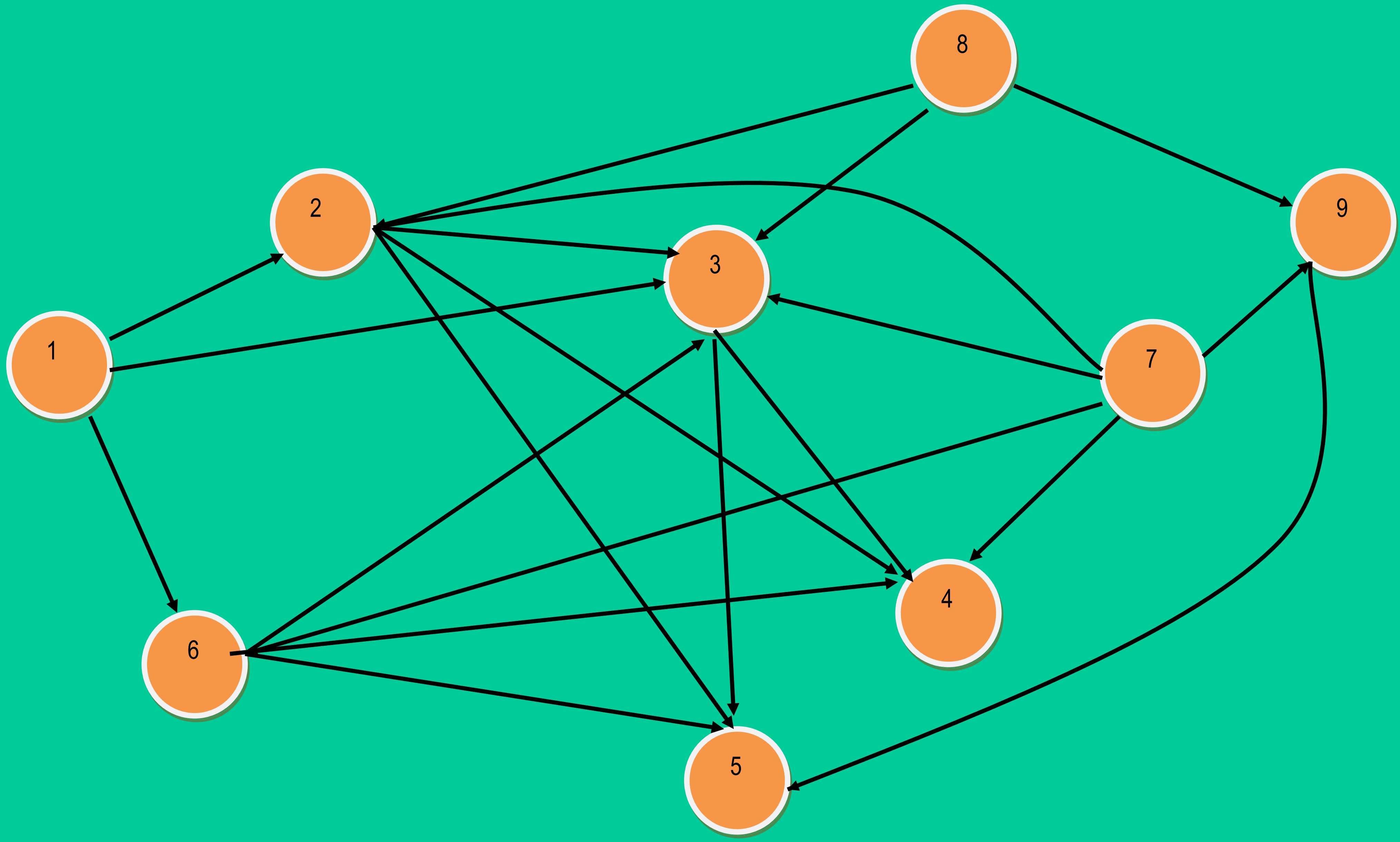
```
profondeur(entier s, GRAPHE g, booléen M[MAX])
    début
    entier i, x ;  /* x désigne le numéro d'un sommet
    M[s] := vrai ;
    pour i :=1 , i<=d°+(s,g)
        /* parcours de tous les chemins issus de s
        début
        x:=ième_succ(i,s,g);
        si M[x]) = faux   profondeur (x,g,M) ;
        fin
    fin /* profondeur
```

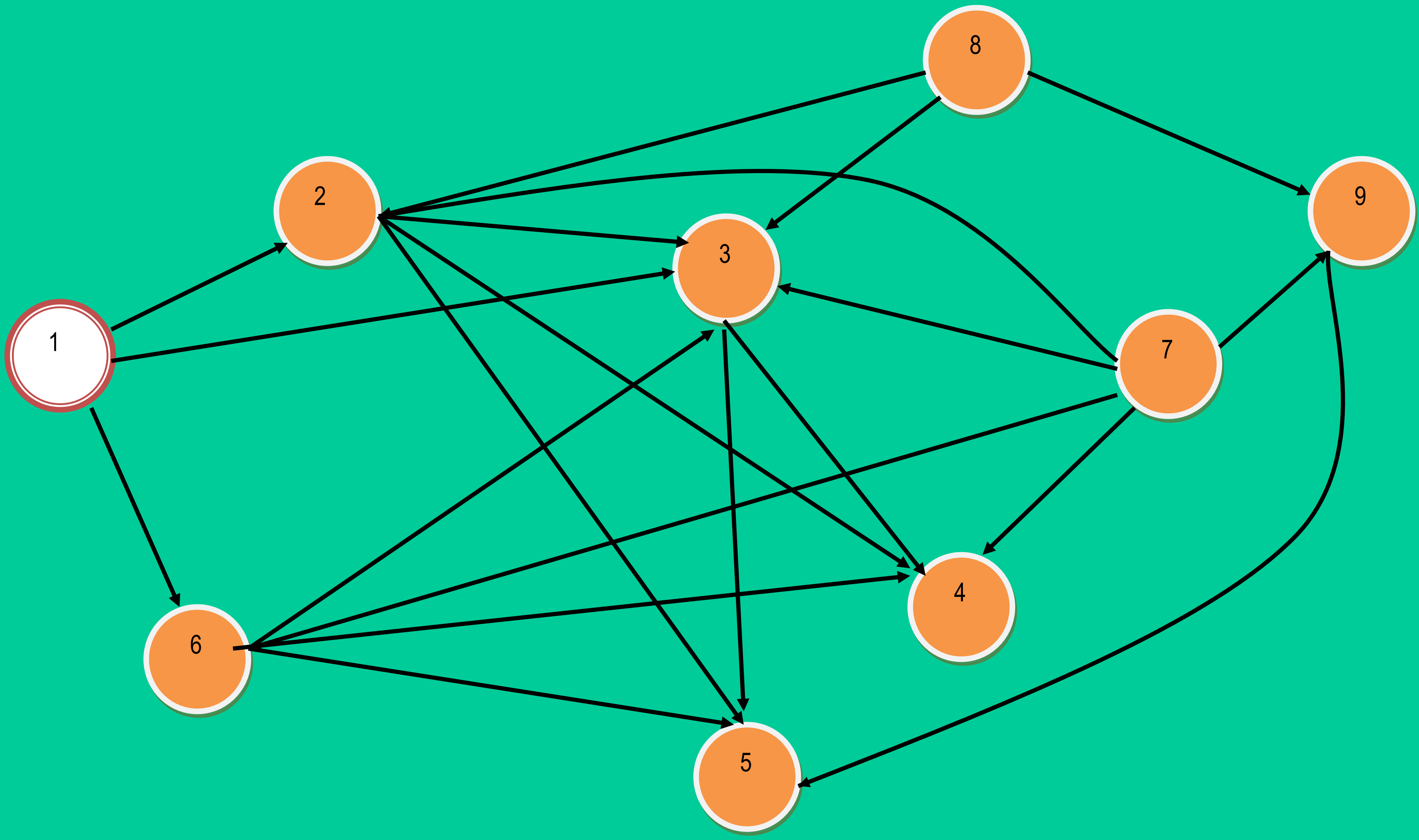
/ Parcours de **tous** les chemins issus de **tous** les sommets du graphe */*

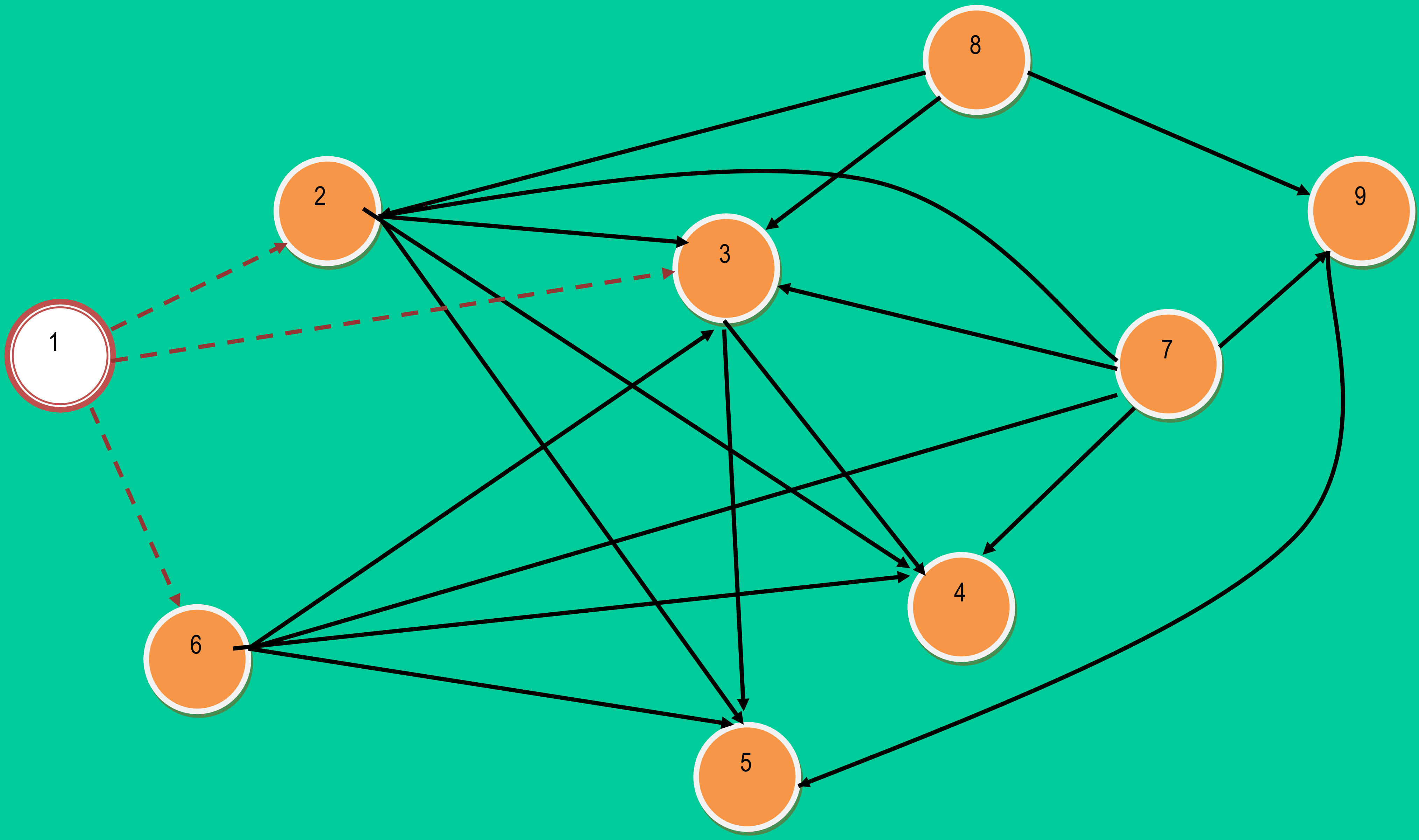
```
parcours(GRAPHE g)
début
    entier s ;
    booléen M[MAX];
    n := taille(g);    /* n désigne la taille du graphe g

    /* au départ aucun sommet n'est marqué
    pour s:=1, s<=n    M[s] := faux ;
```

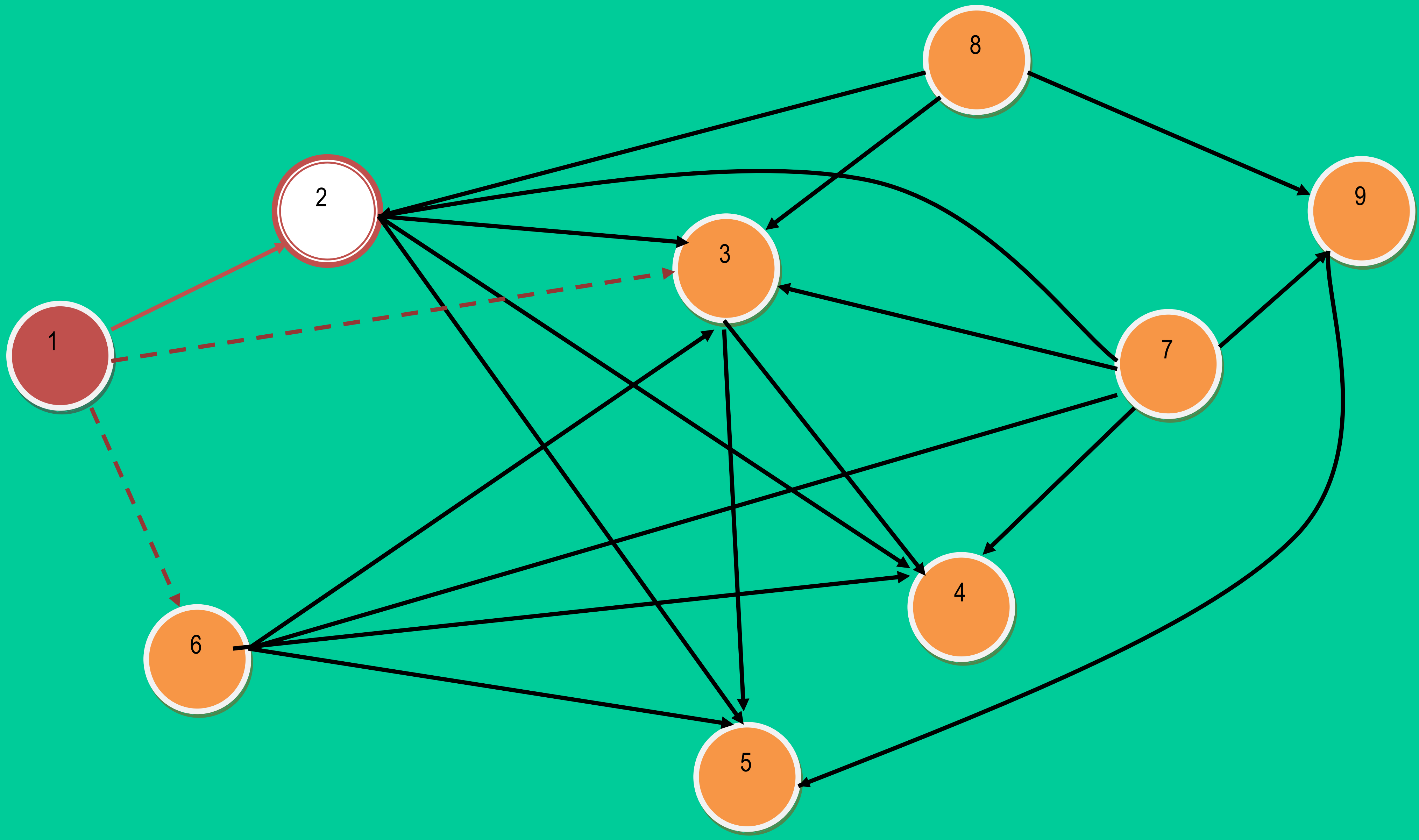
```
    /* parcours de tous les chemins issus de tous les sommets non marqués
    pour s :=1 , s<=n
            si M[s]) = faux  profondeur (s,g,M) ;
    /*  fin de parcours de tous les chemins du graphe
fin /* parcours
```

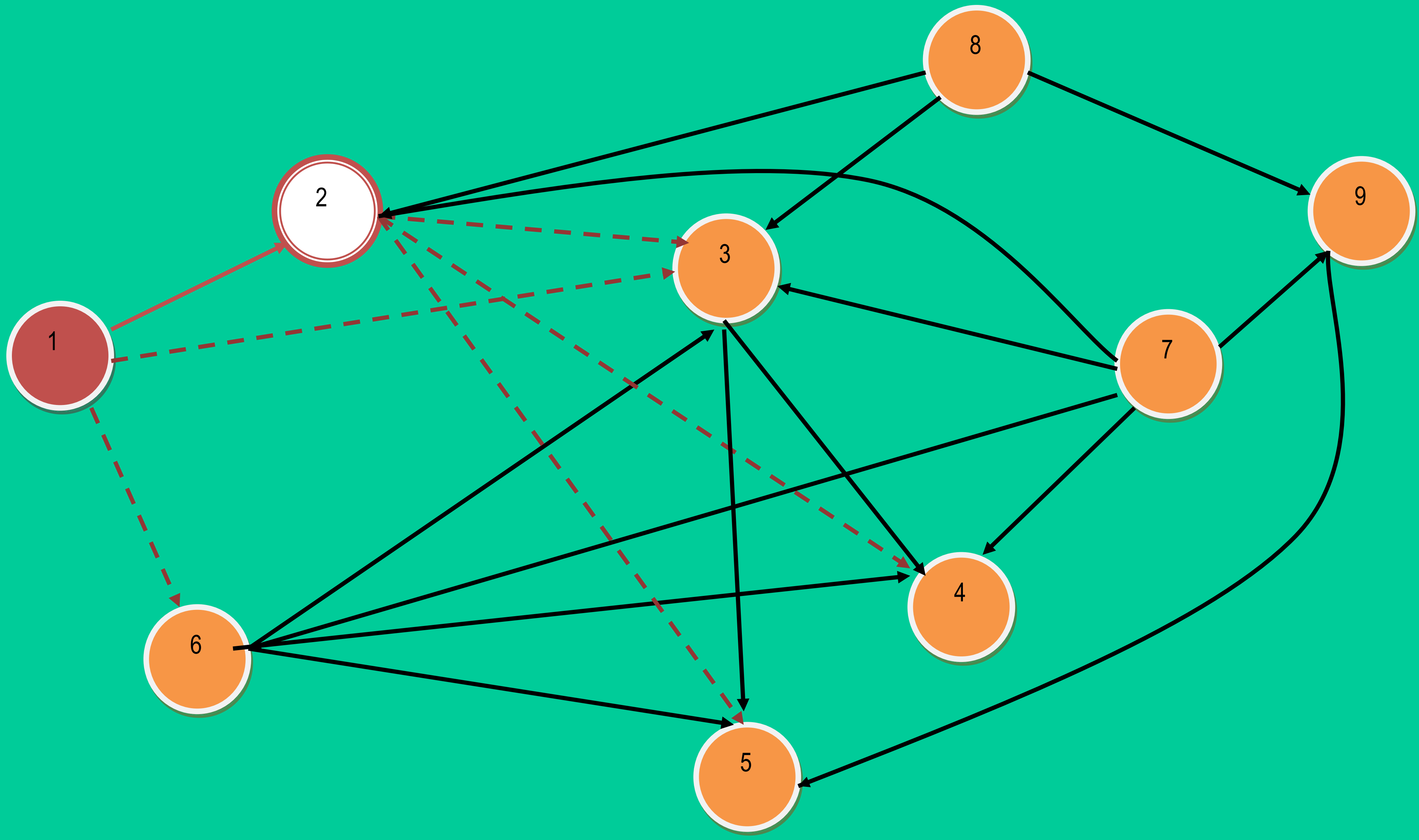
3-Application

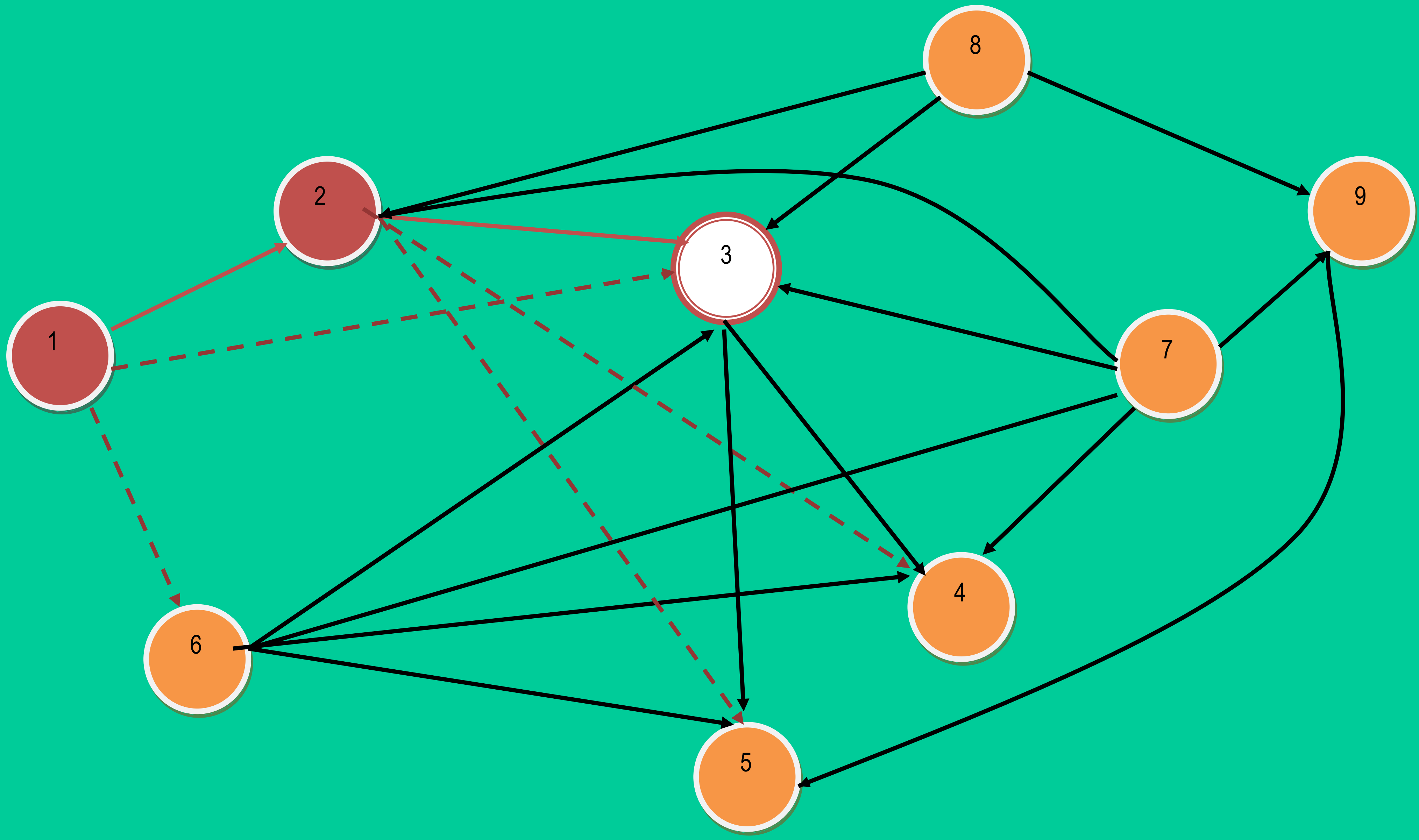


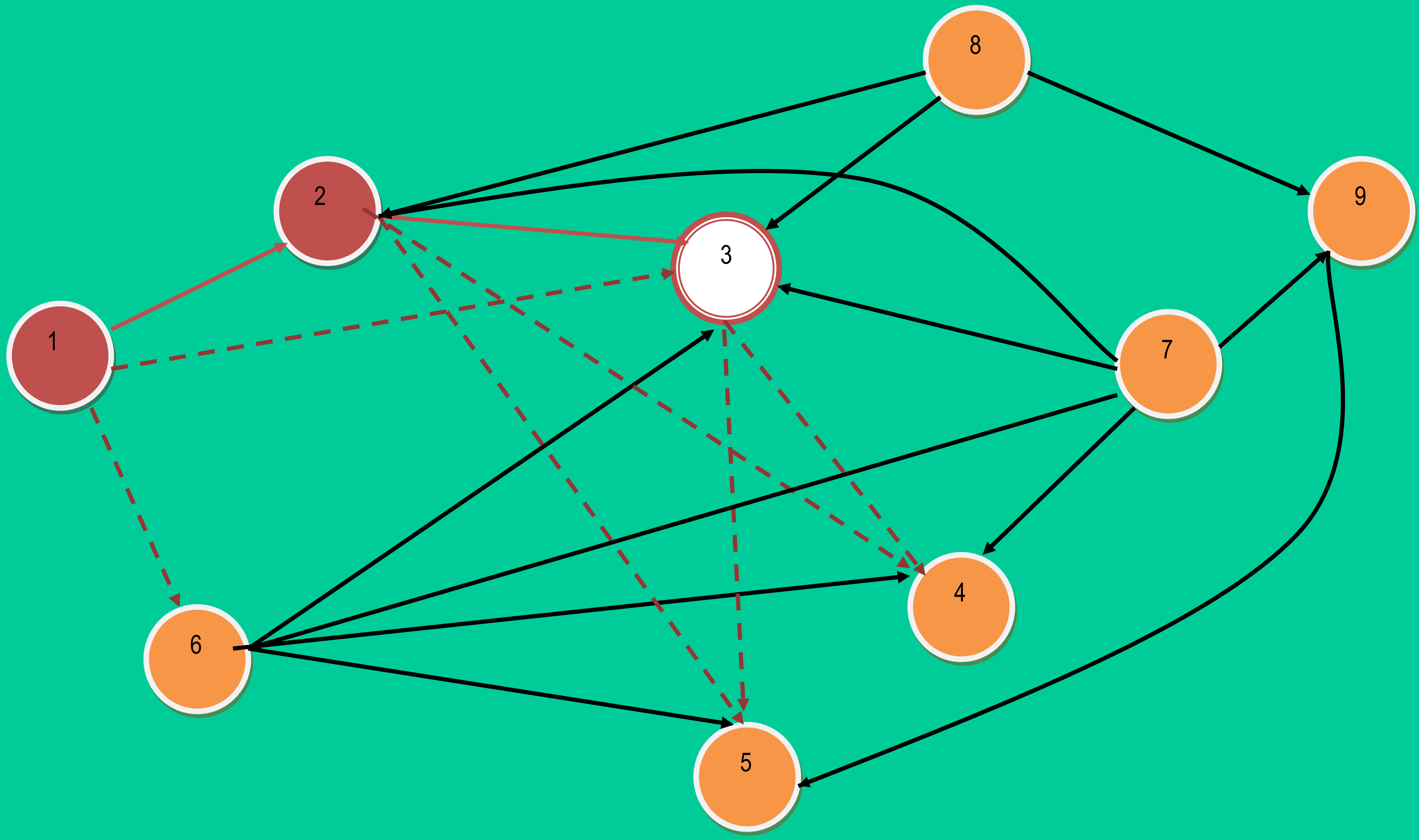

1
2
3
4
5
6
7
8
9

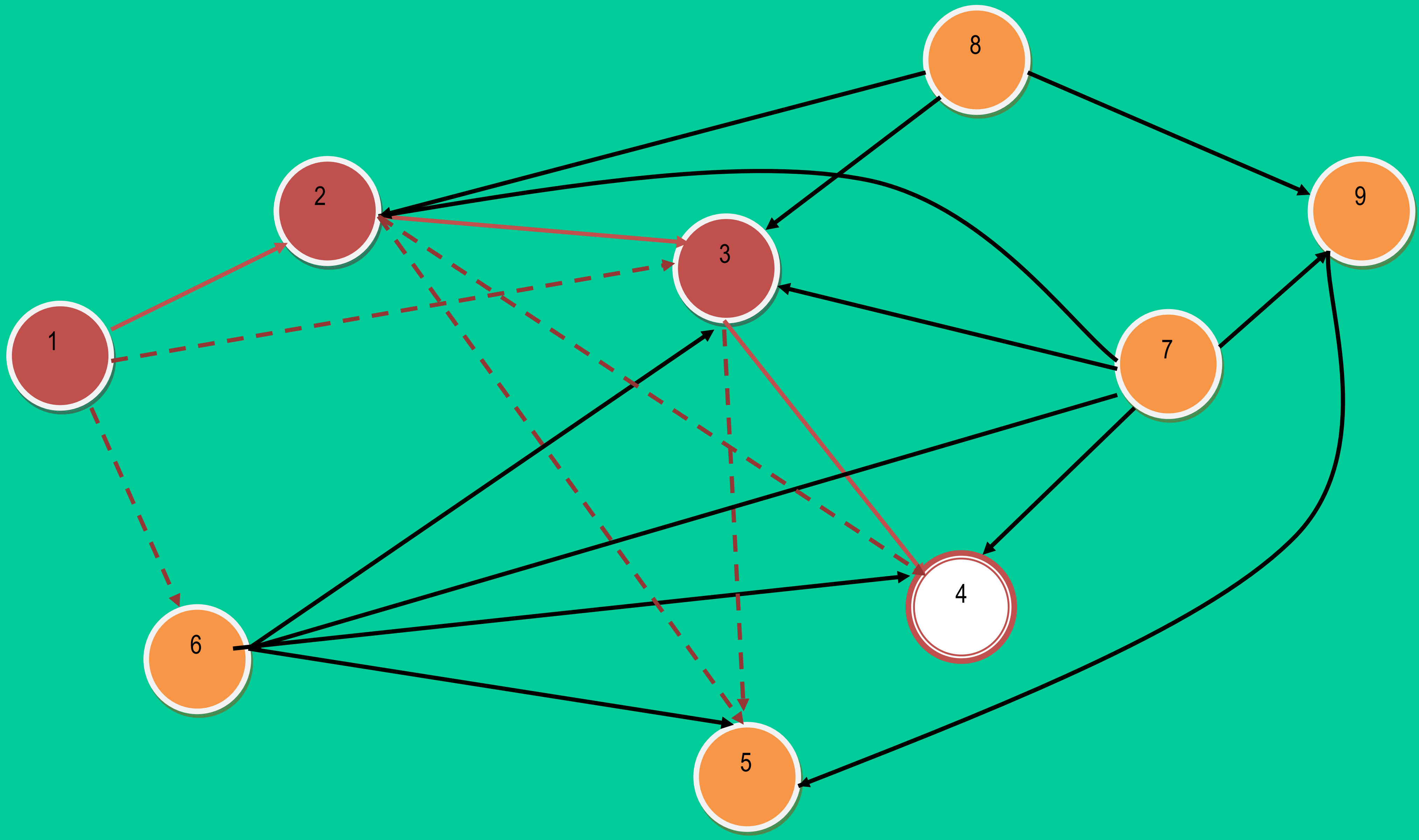

1
2
3
4
5
6
7
8
9

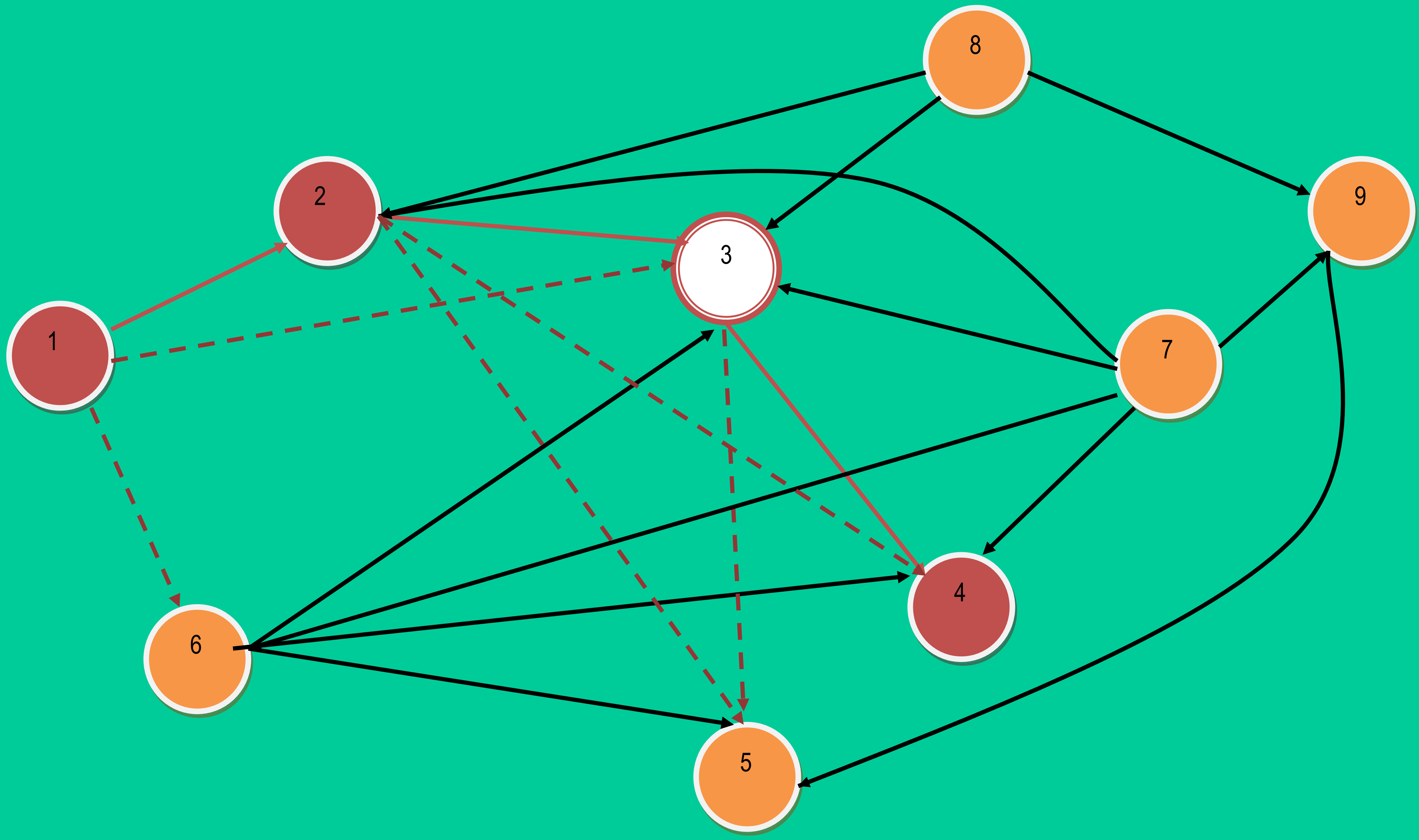

8
2
9
3
1
7
4
6
5

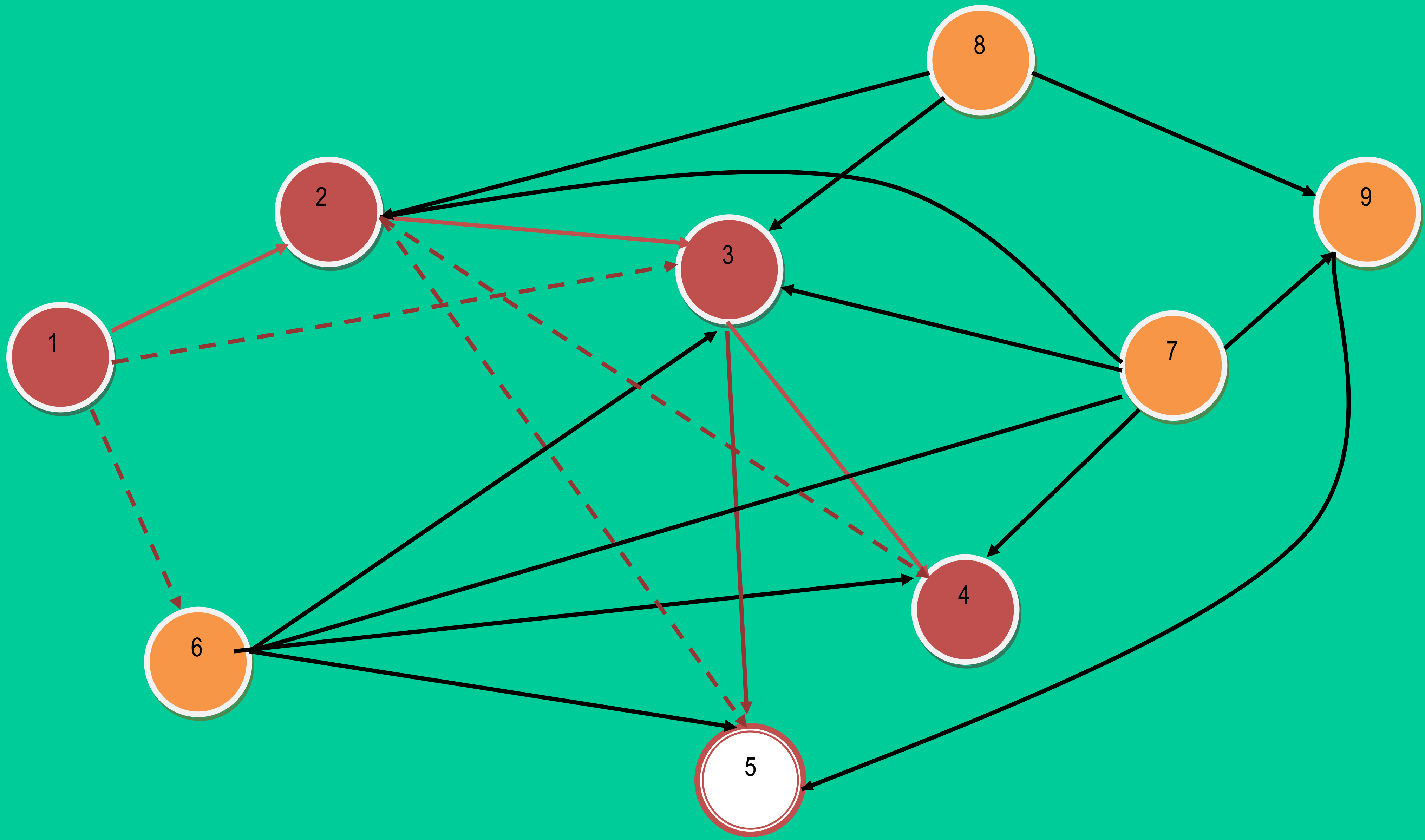

8
2
3
9
1
7
4
6
5

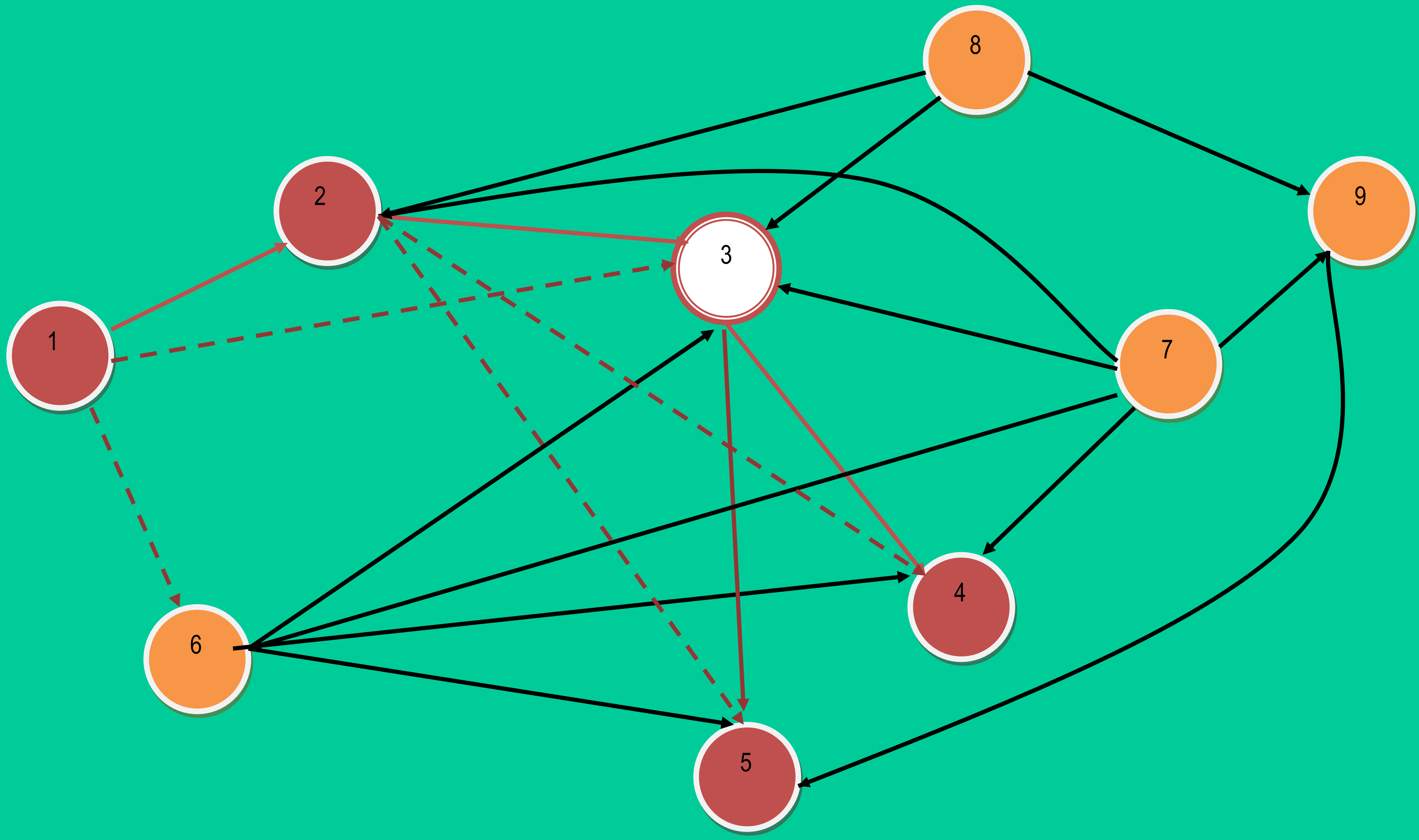

8
2
3
9
1
7
6
4
5

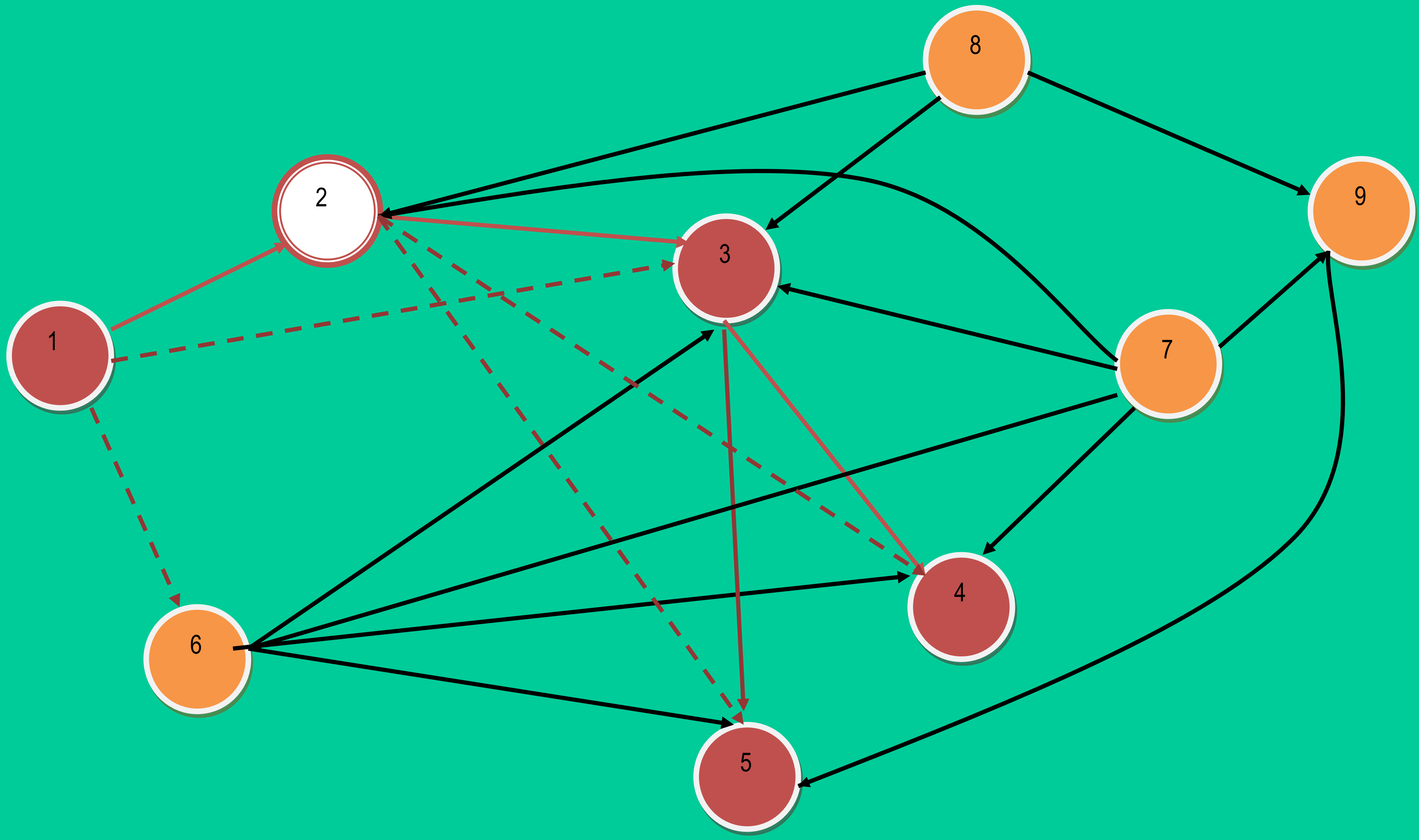

1
2
3
4
5
6
7
8
9

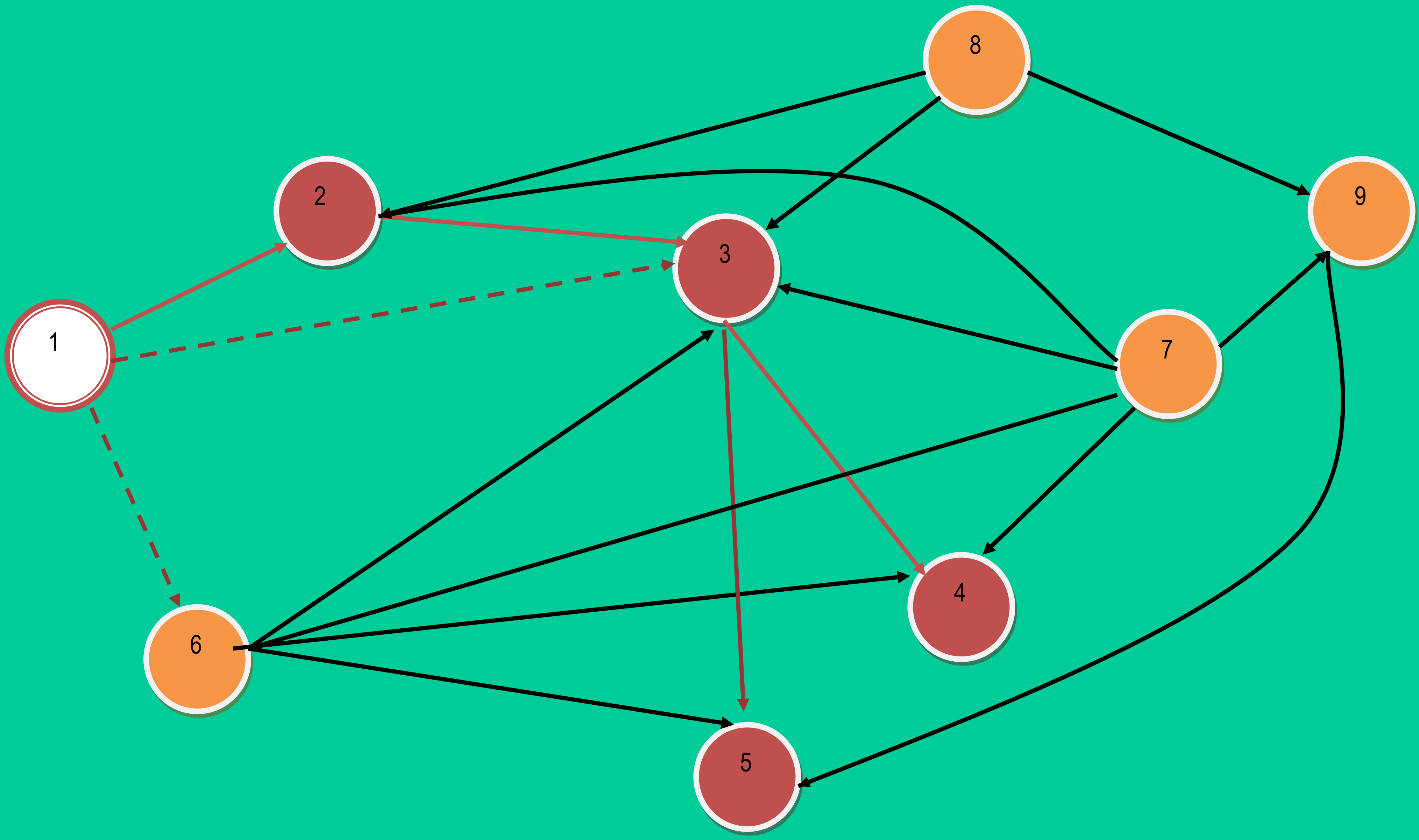

8
2
3
9
1
7
4
6
5

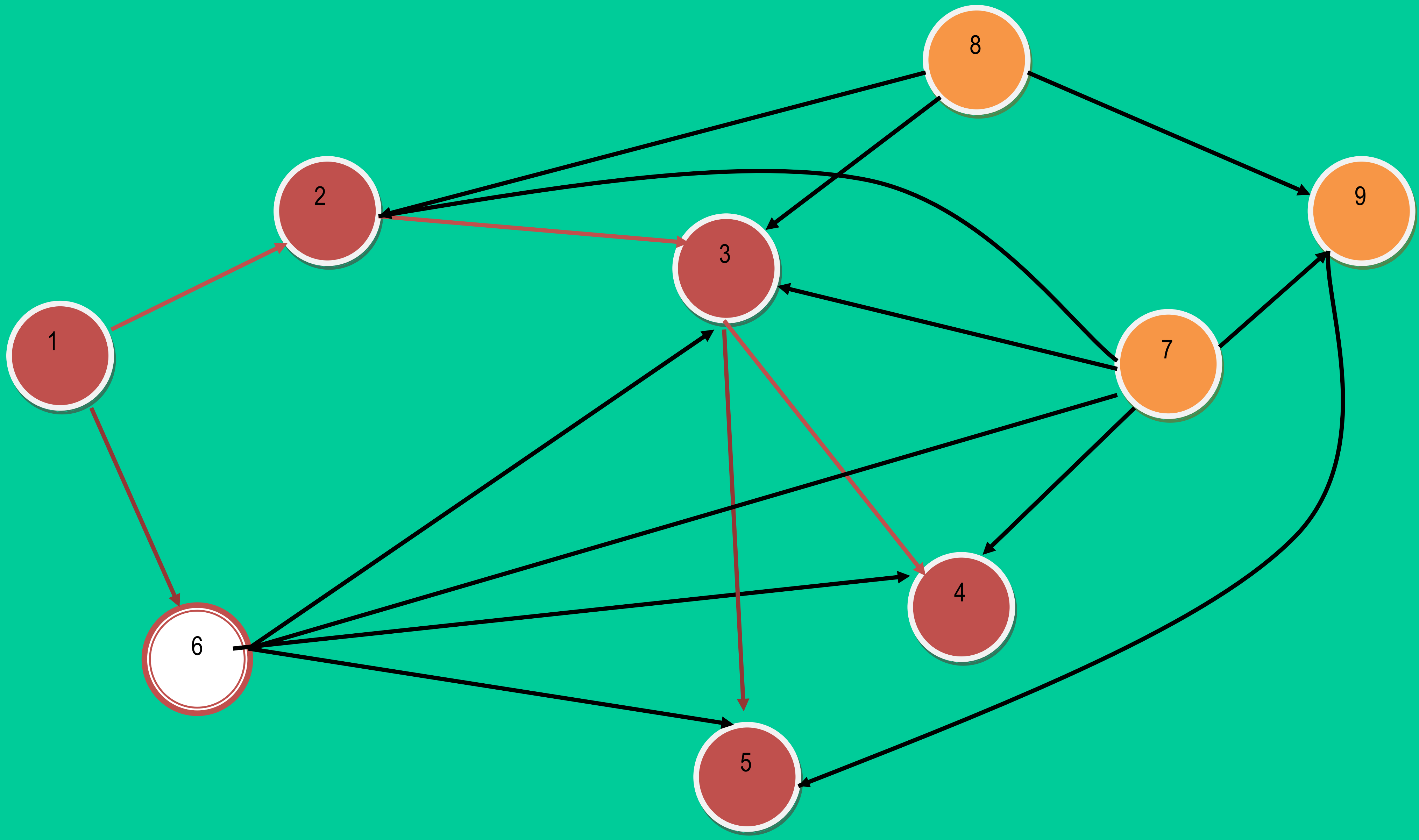

8
2
3
9
1
7
4
6
5

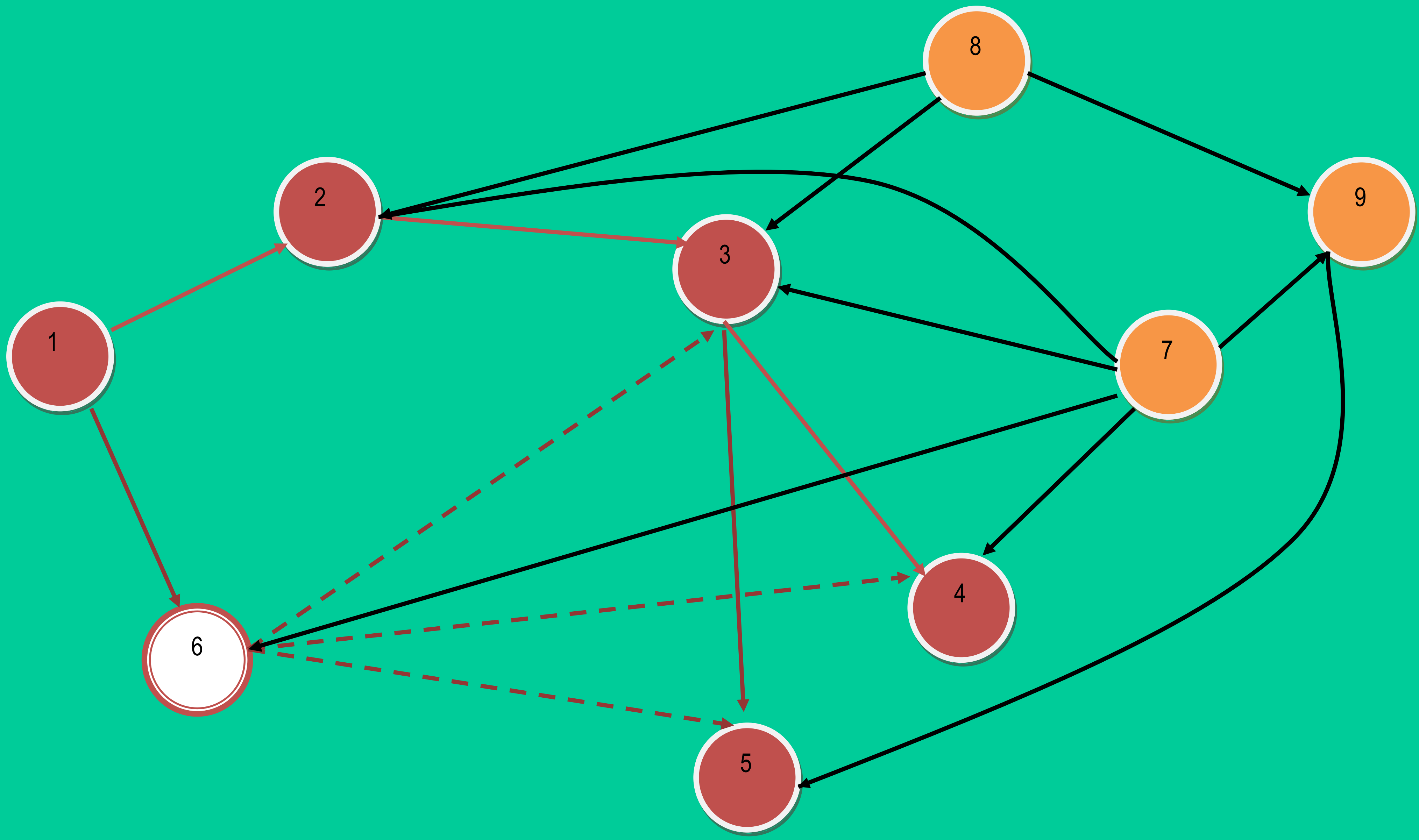

8
2
9
3
1
7
6
4
5

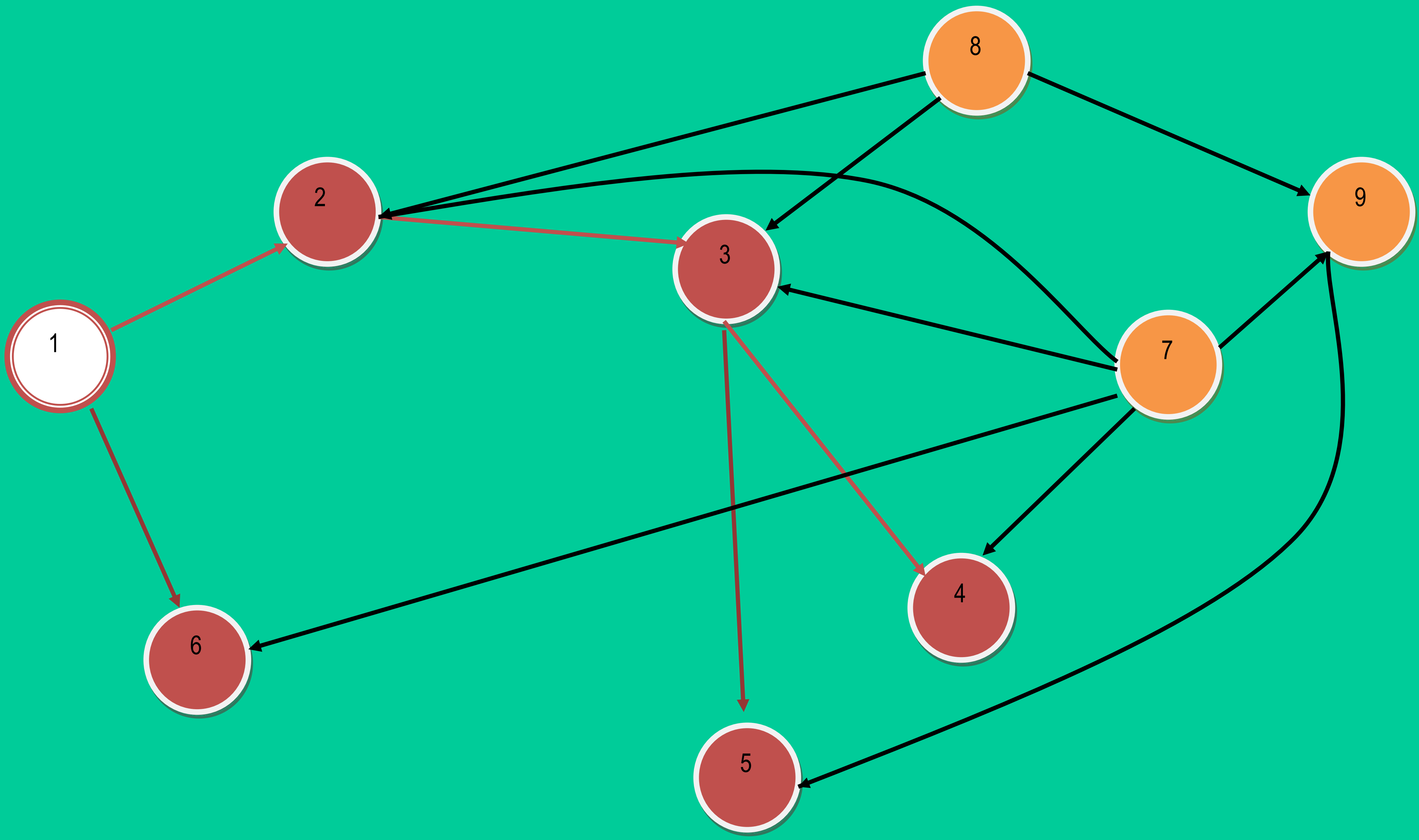

8
2
3
9
1
7
4
6
5

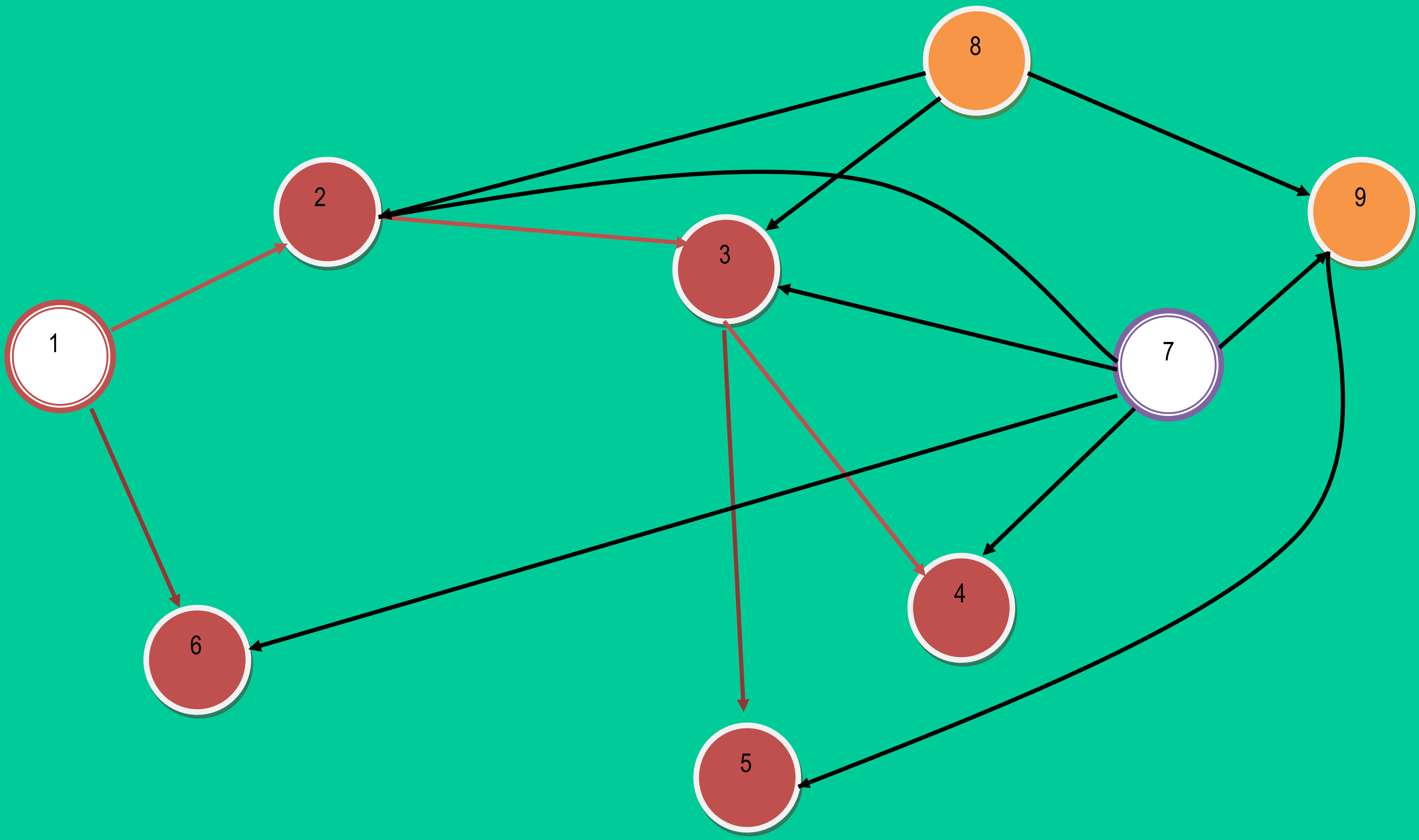

8
2
3
9
1
7
4
6
5

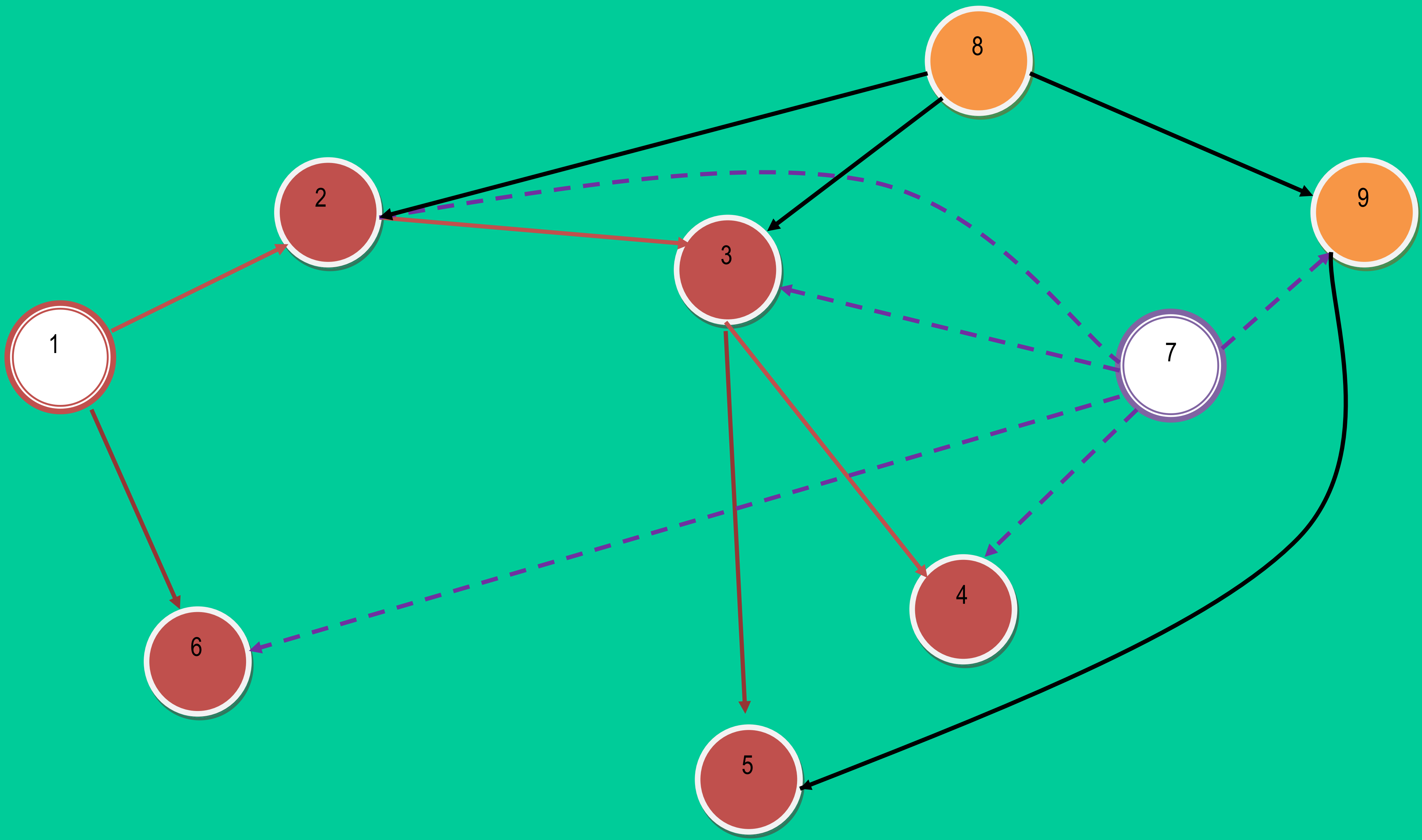

8
2
3
9
1
7
6
4
5

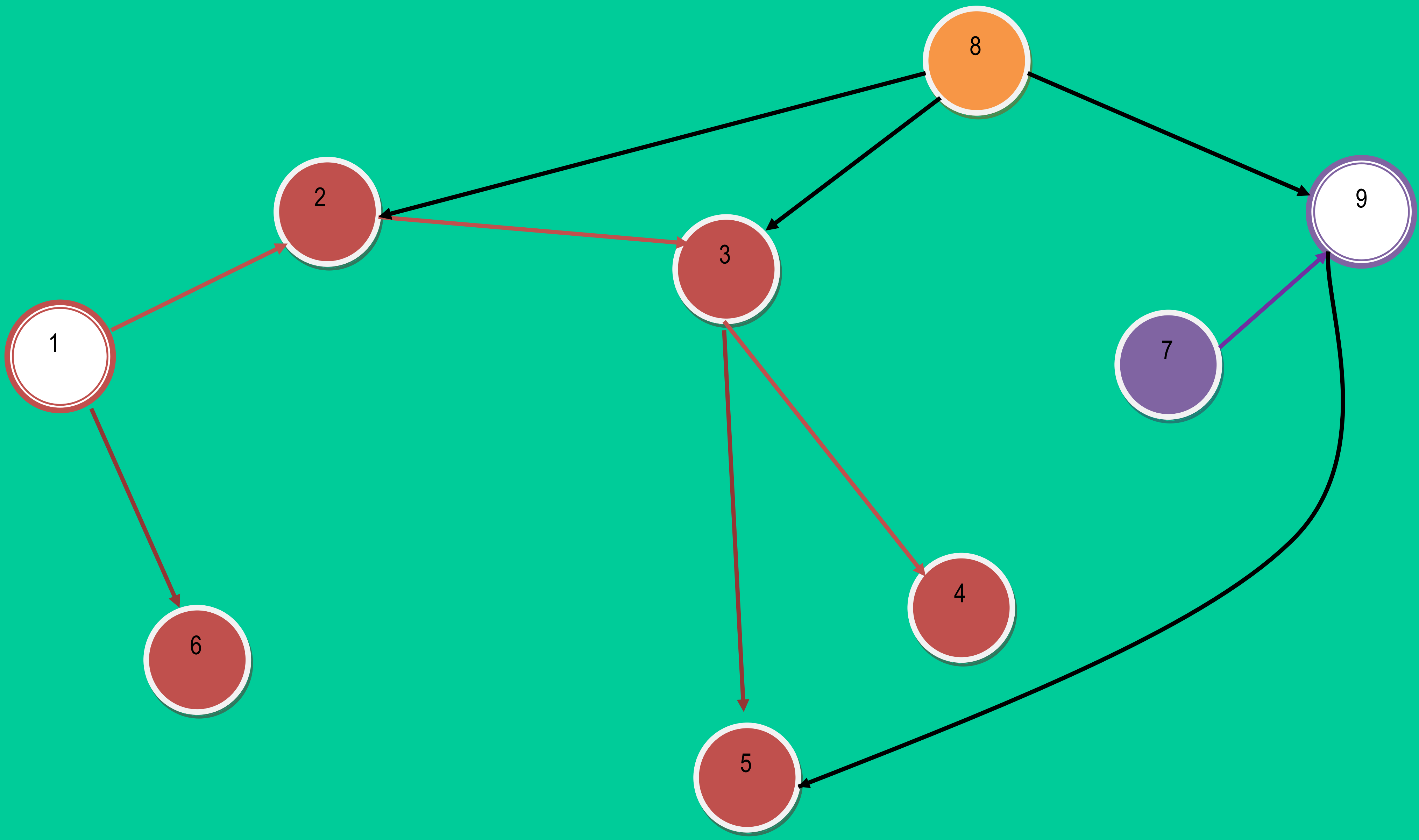

8
2
3
9
1
7
4
6
5

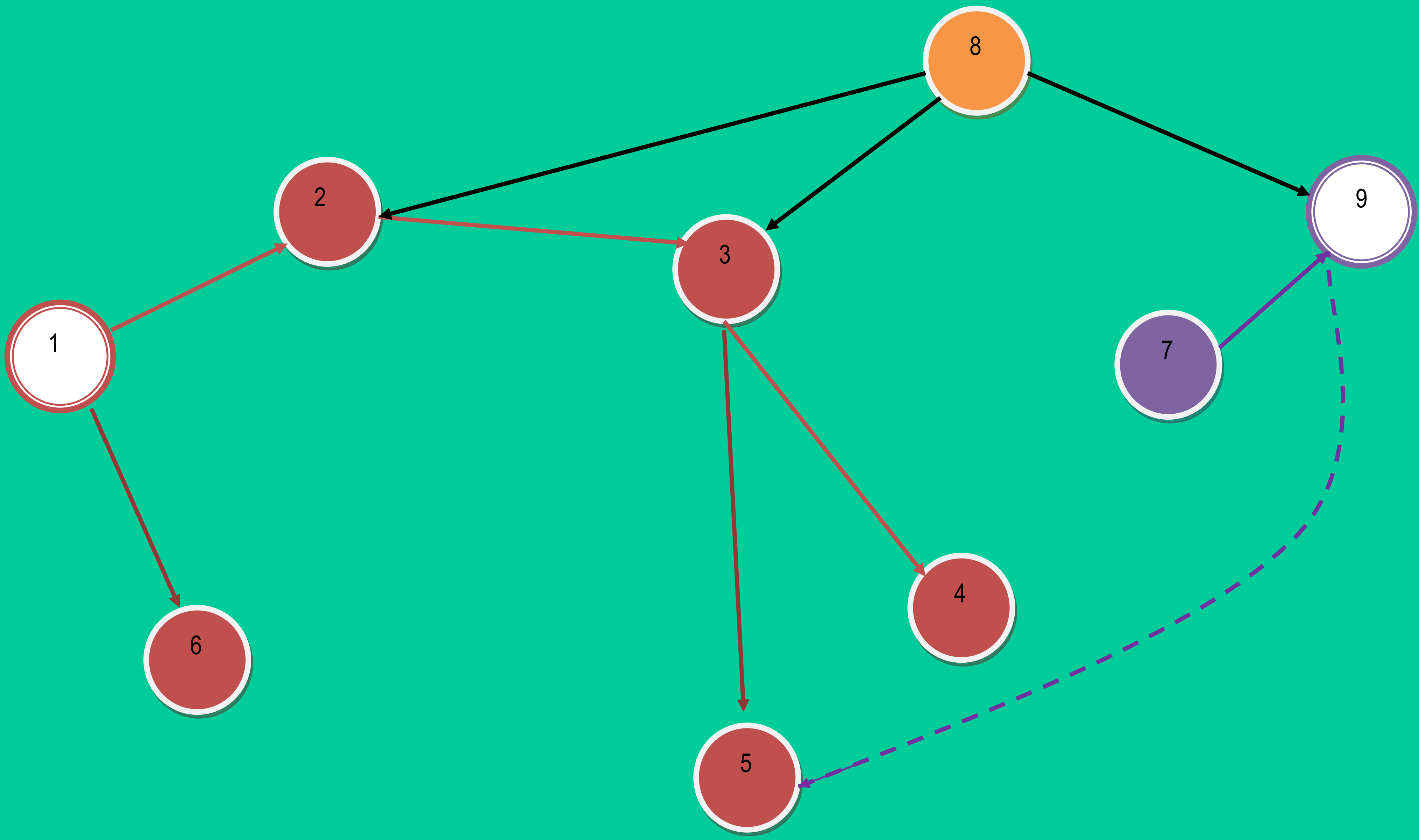

8
2
3
9
1
7
6
4
5

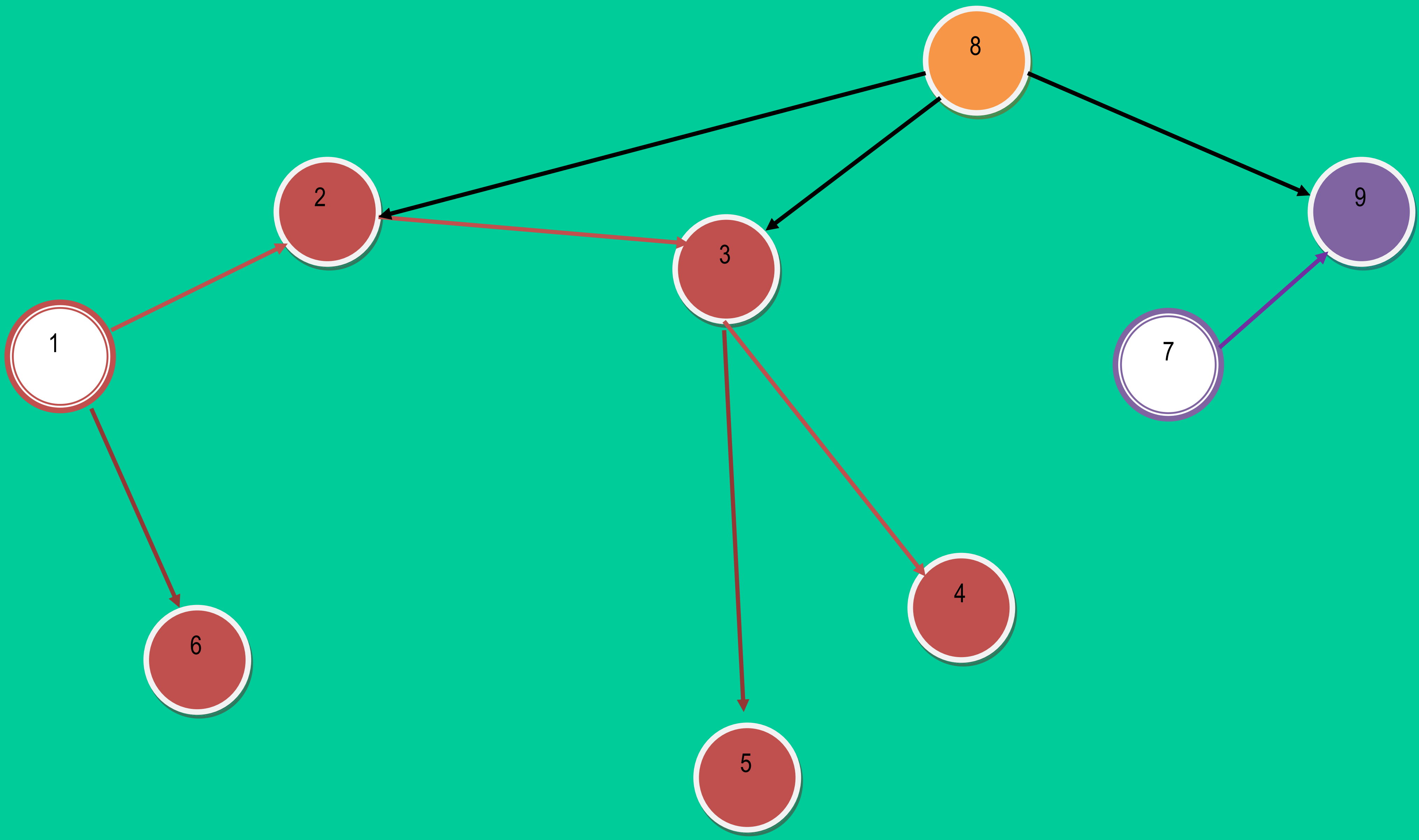

8
2
9
3
1
7
4
6
5

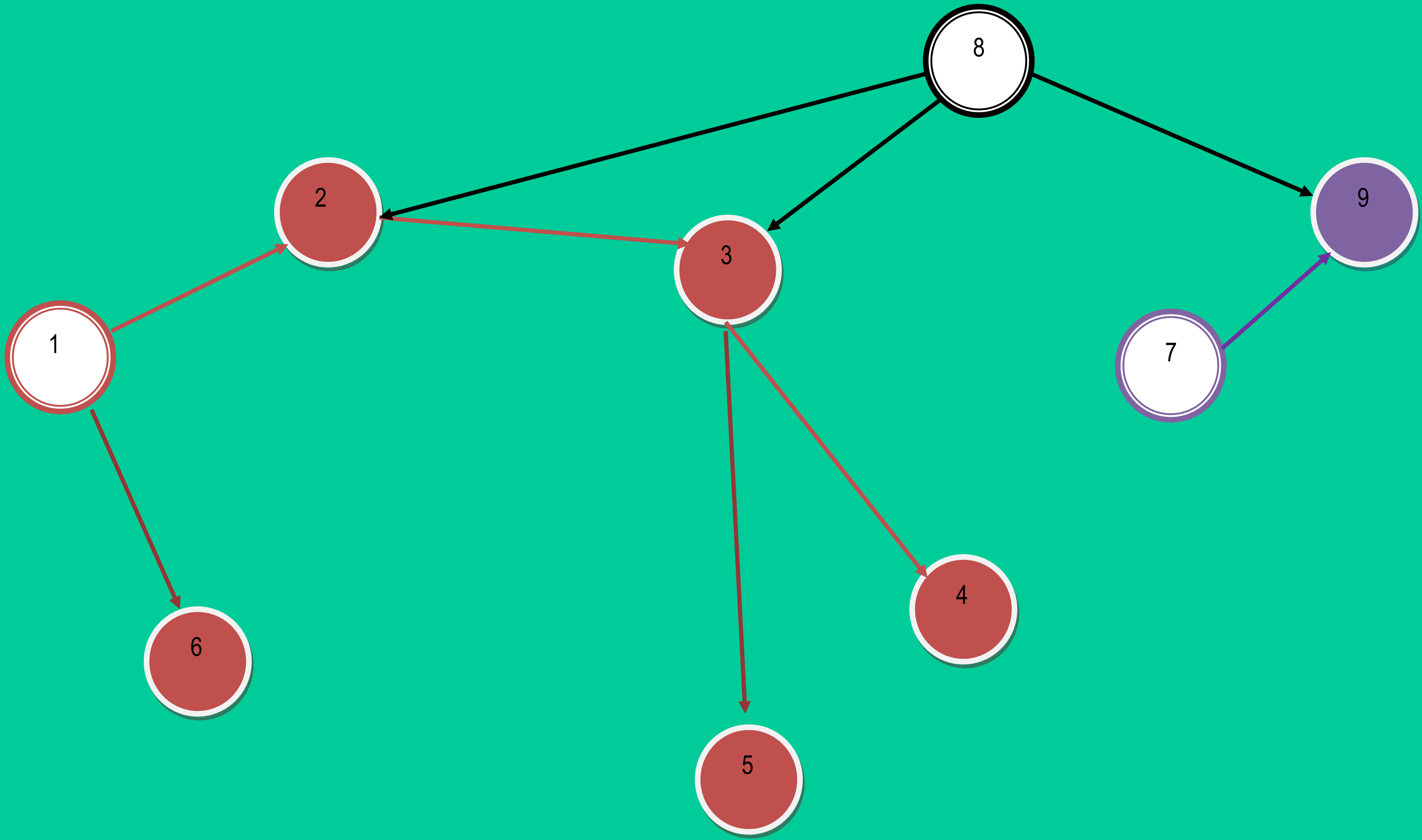

1
2
3
4
5
6
7
8
9

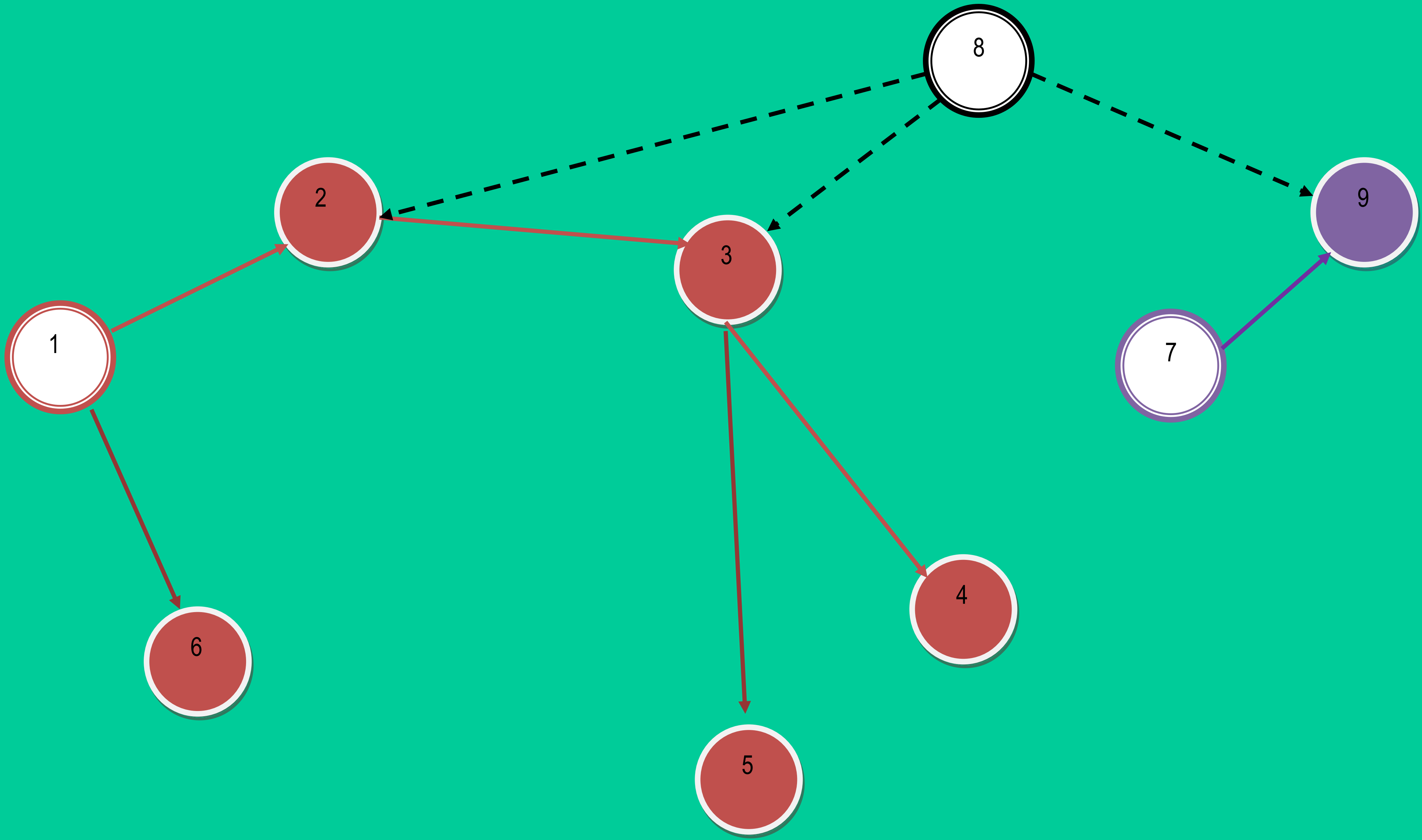

8
2
9
3
1
7
4
6
5

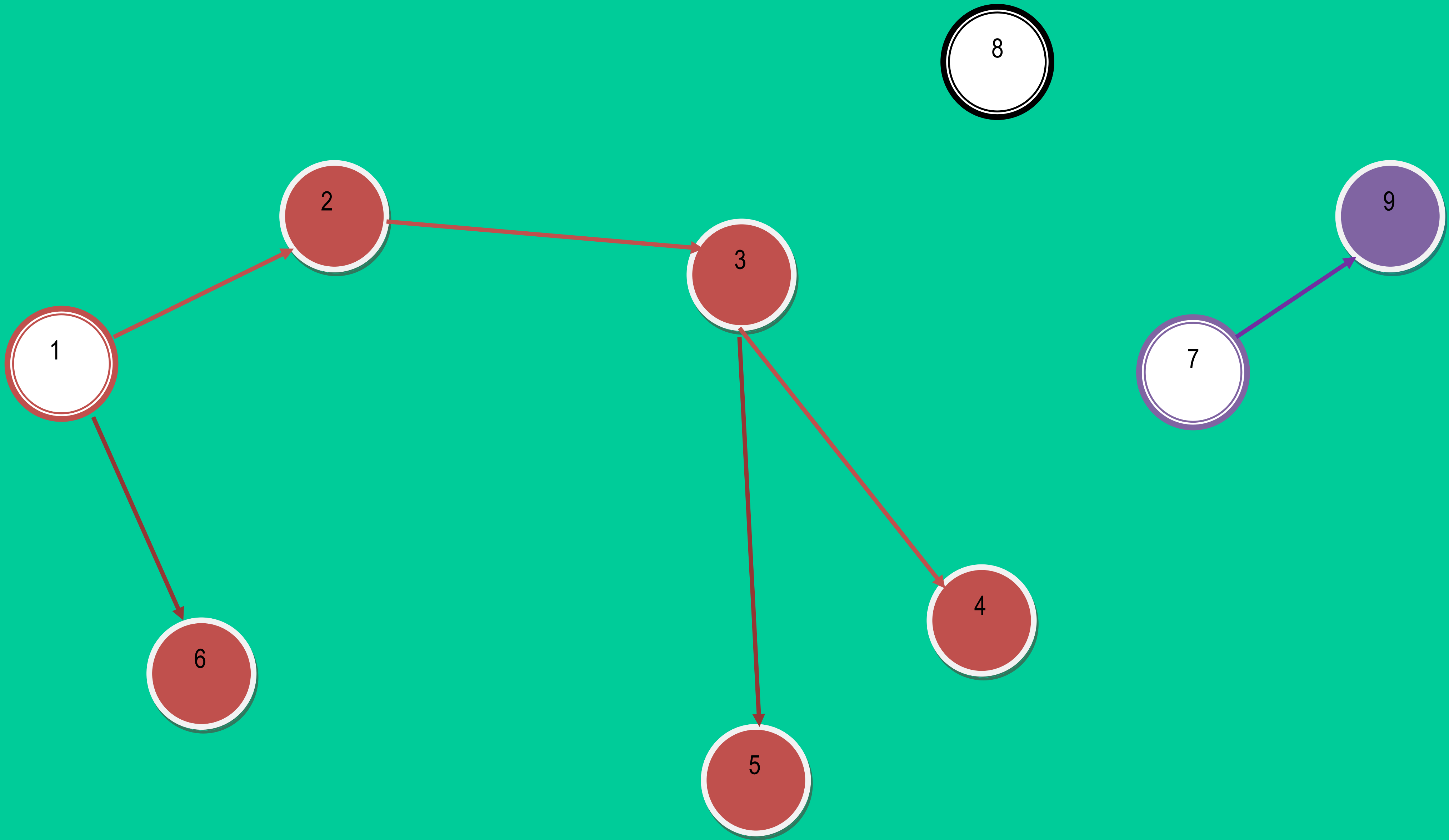

8
2
3
9
1
7
4
6
5


8
2
3
9
1
7
6
4
5


8
2
3
9
1
7
6
4
5


8
2
3
9
1
7
4
6
5


8
2
3
9
1
7
4
6
5


8
2
3
9
1
7
4
6
5

REMARQUES IMPORTANTES

1- Les trois sous-graphes connexes engendrés sont des **arborescences.**

2- Les sommets 1,7 et 8 sont leurs **racines**.

3-Les trois arbres assurent une **couverture** des sommets du graphe original sans traverser un **cycle**.

II- Parcours en largeur

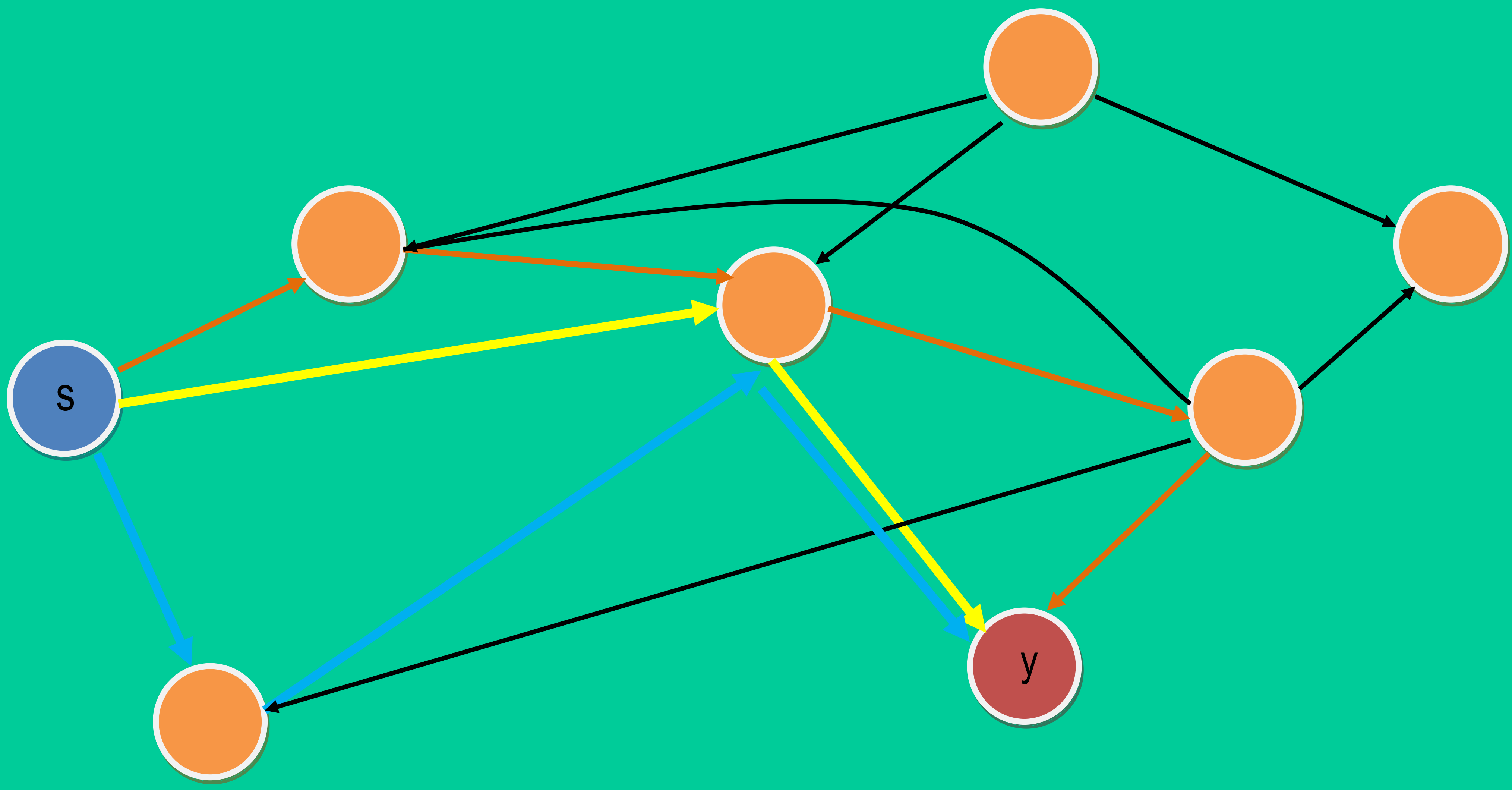
Rappels:

Dans un graphe orienté G, la **distance topologique**, notée **d(s,y)** :

- d'un sommet s,
- à un sommet y,

est la longueur du **plus court chemin** issu de s et allant vers y.

d(s,y)= 2



Algorithme

Le parcours en **largeur** consiste :

- à partir d'un sommet **s de départ**,
- à visiter d'abord **tous** les sommets qui sont à une **distance 1** de s,
- puis **tous** ceux qui sont à la **distance 2**,
- puis **tous** ceux à la **distance 3**,
- et ainsi de suite.

/* visiter **tous** les **descendant**s mon marqués d'un sommet **s** du graphe g

```
largeur(entier s ; GRAPHE g ; booléen M[MAX])
début
entier v,w,i ;
FILE F;
F := fileVide() ; / * initialiser la file F
M[s] := vrai; /* marquer le sommet de départ s
F := ajouter (F,s) ; /*  empiler s
tant_que   estVide(F) = faux
        début
            v:= premier(F) ; F:= retirer(F) ;
```

```
        pour i:=1, i<= d°+(v)
                début
                w:= ieme_succ(i,v,g) ;
                si M[w] = faux
                    début
                     visiter(w,g); M[w]:= vrai;
                     F:=ajouter(F,w);
                  fin /*fin  si
                fin  /*fin  pour
   fin /*fin  tant que
fin
```

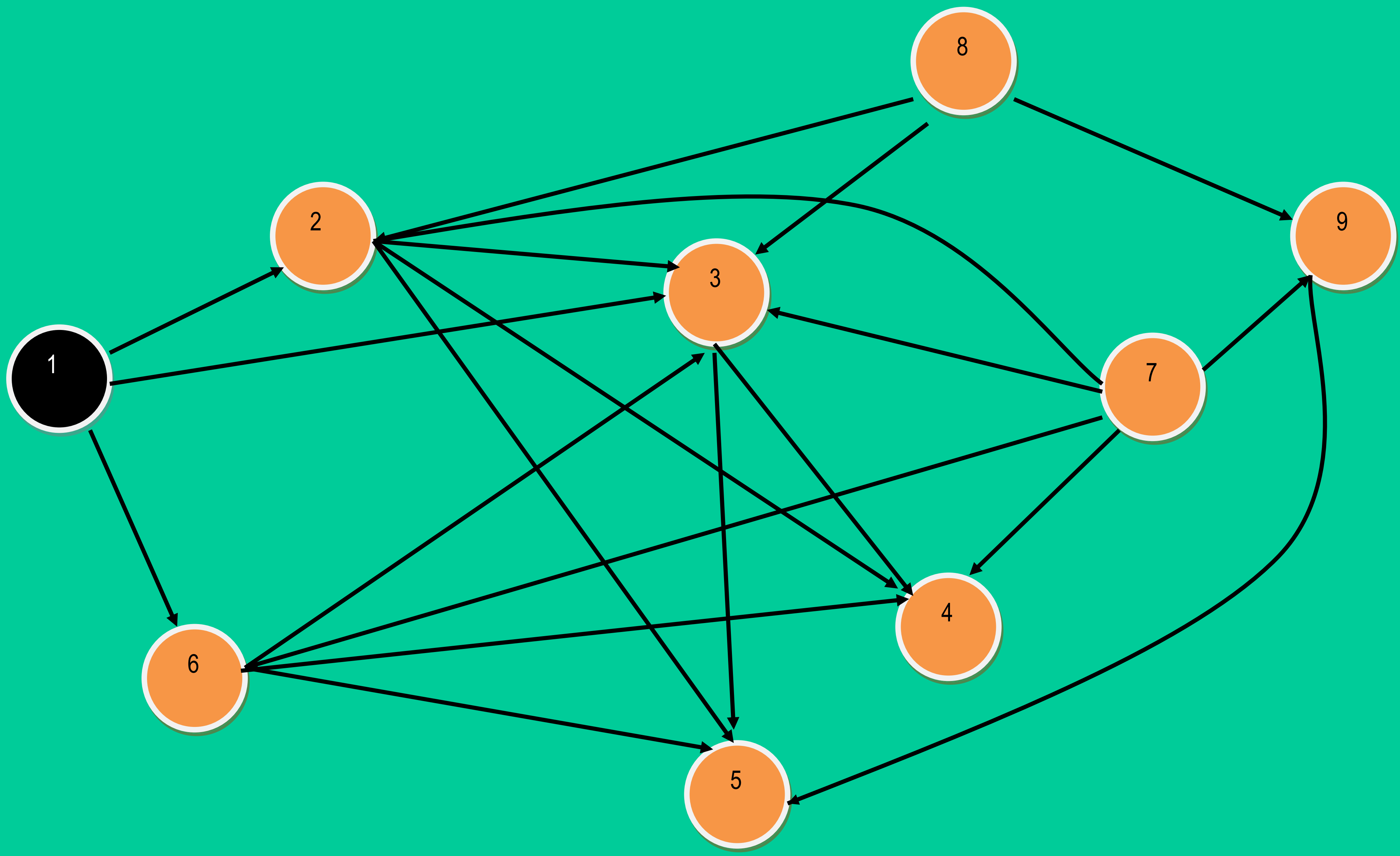
/ visiter **tous** les **descendants** mon marqués de **tous** les sommets du graphe g */*

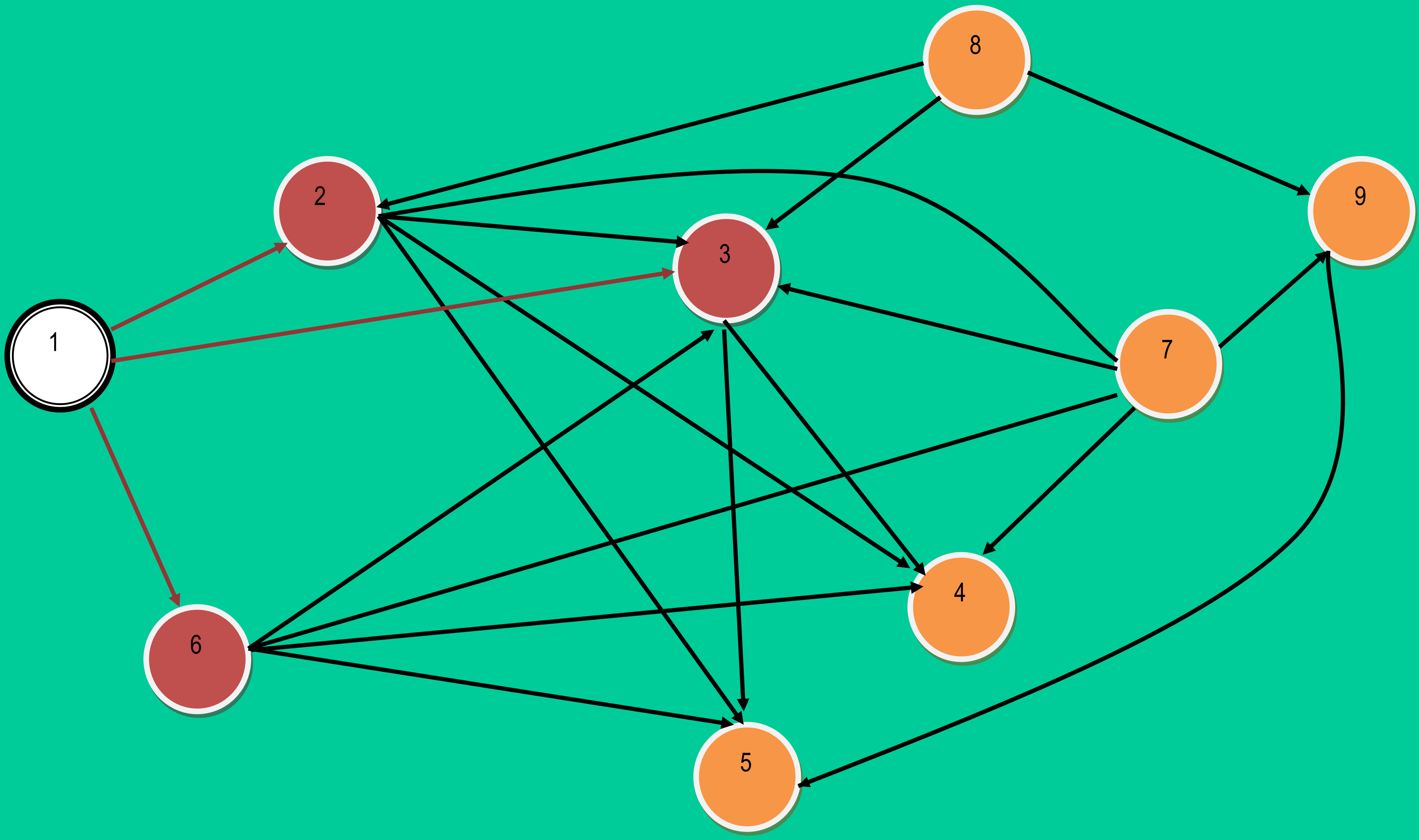
```
parcours (GRAPHE g)
début
entier s ;
booléen M[MAX];
n := taille(g);    /* taille() retourne la taille du graphe g

/* au départ aucun sommet n'est marqué */
pour s:=1, s<= n    M[s] := faux ;
```

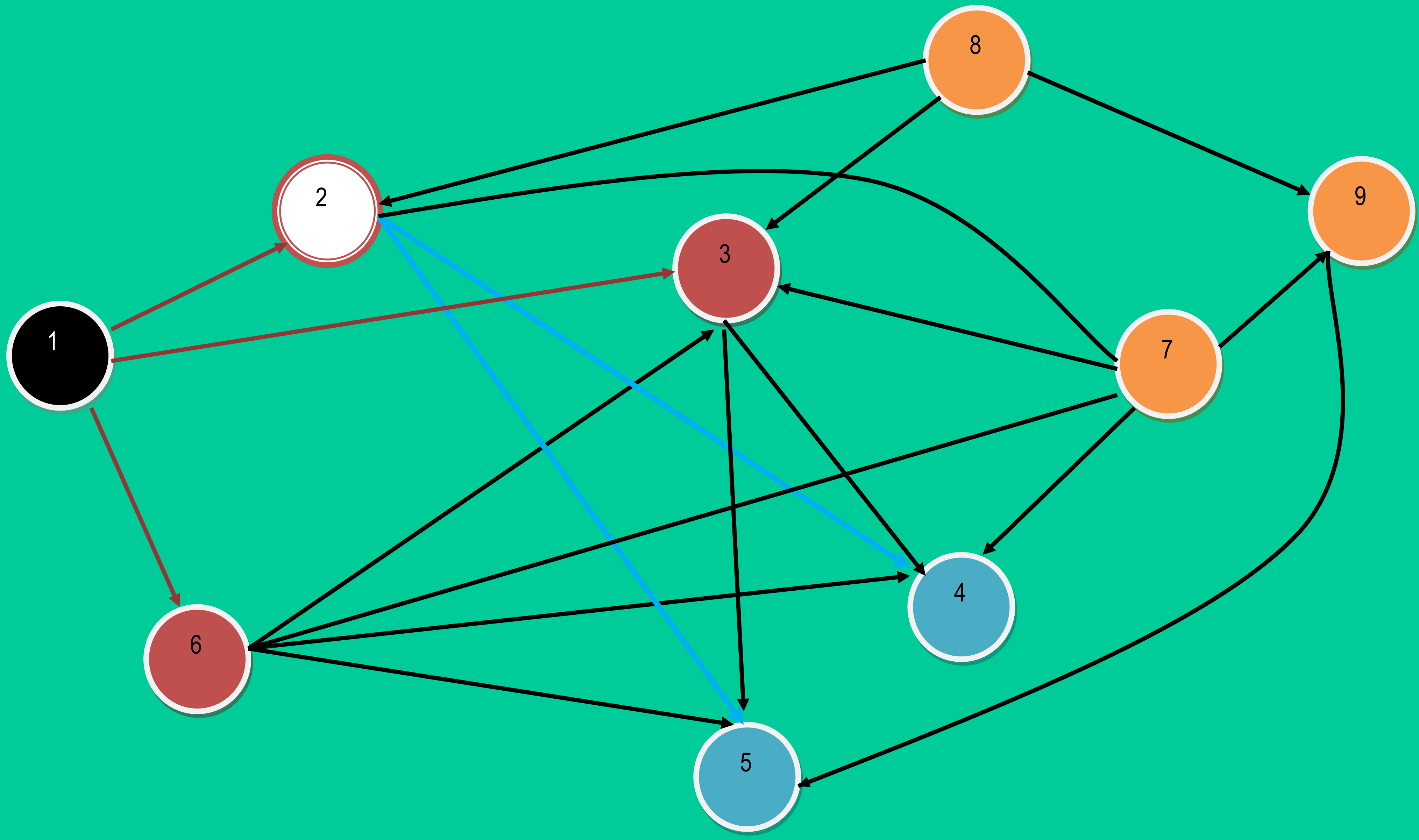
/ visiter pour chaque sommet s, non marqué, de tous ses* ***descendants*** *non déjà marqués */*

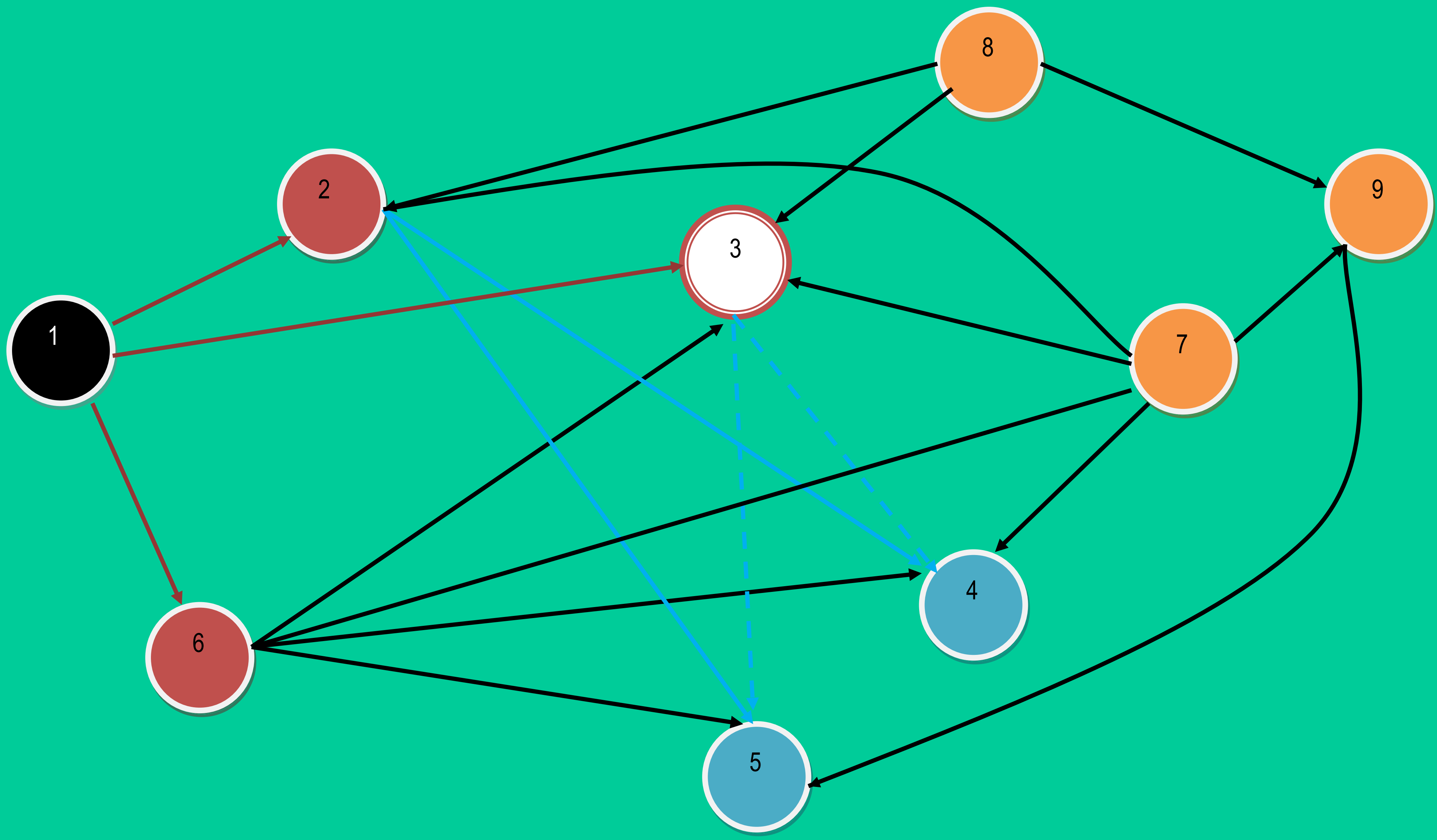
```
pour s :=1, s <=n
            si M[s]) = faux  alors  largeur (s,g,M) ;

fin /* parcours
```

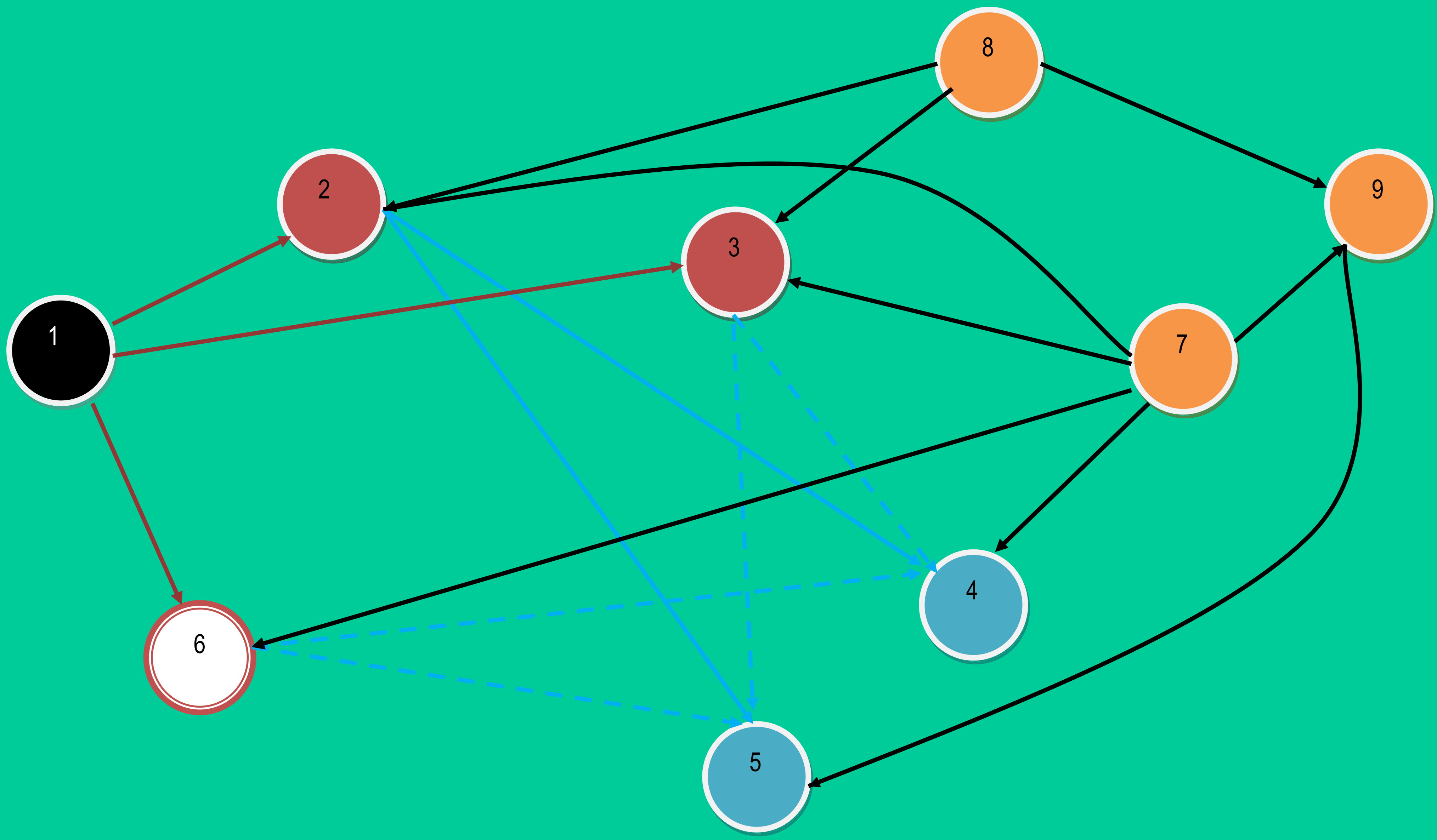
Application

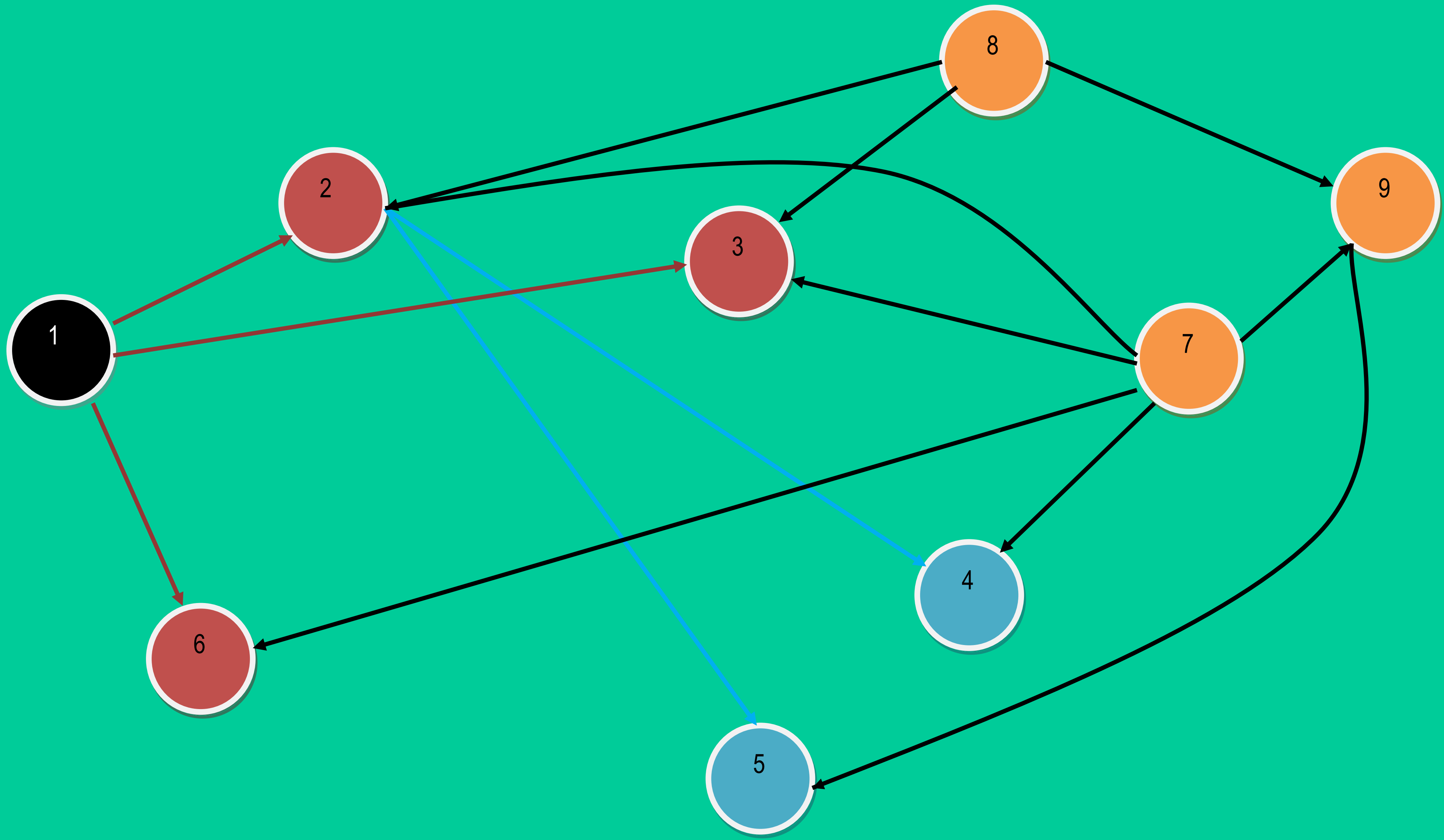


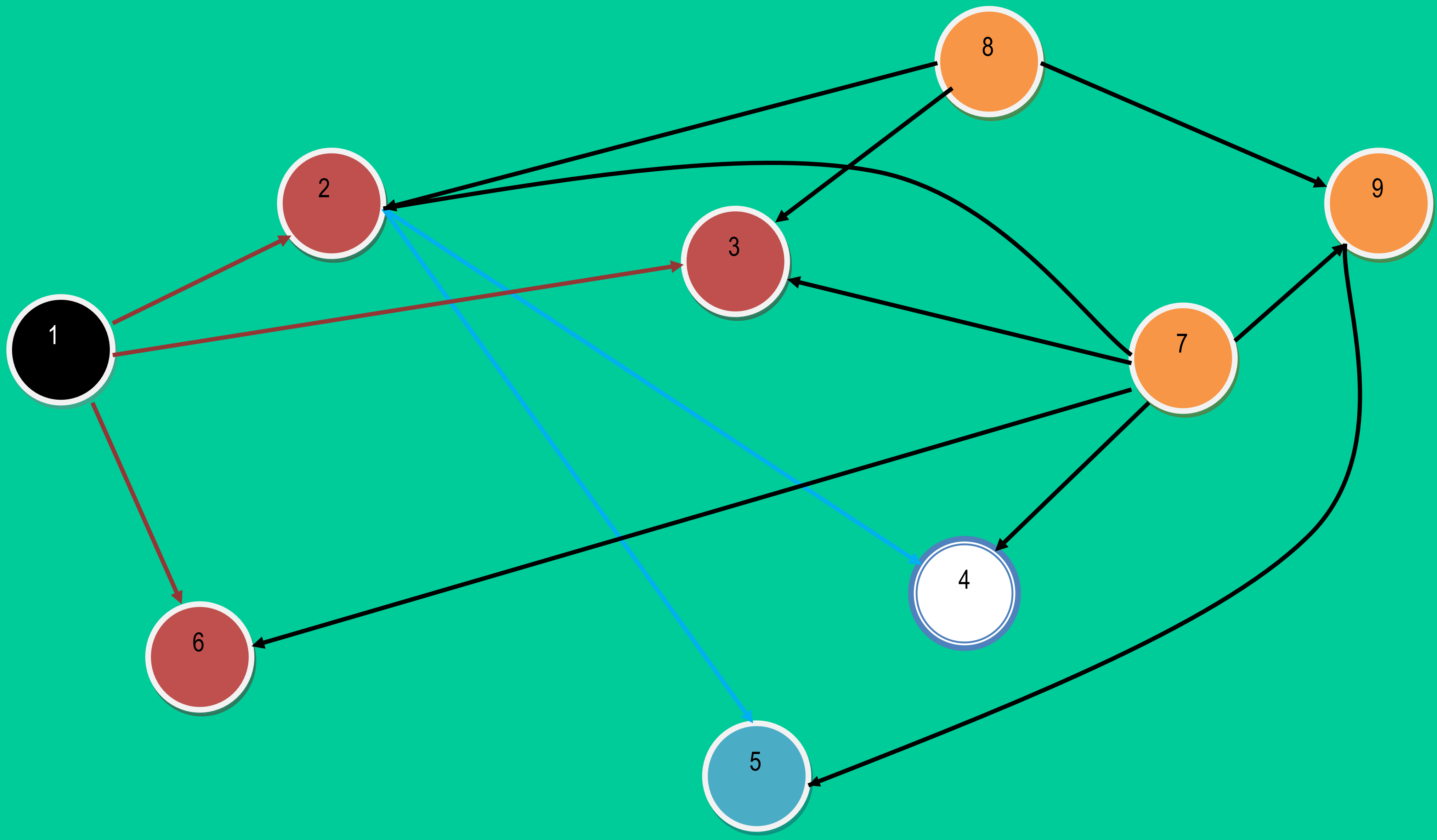

8
2
3
9
1
7
4
6
5

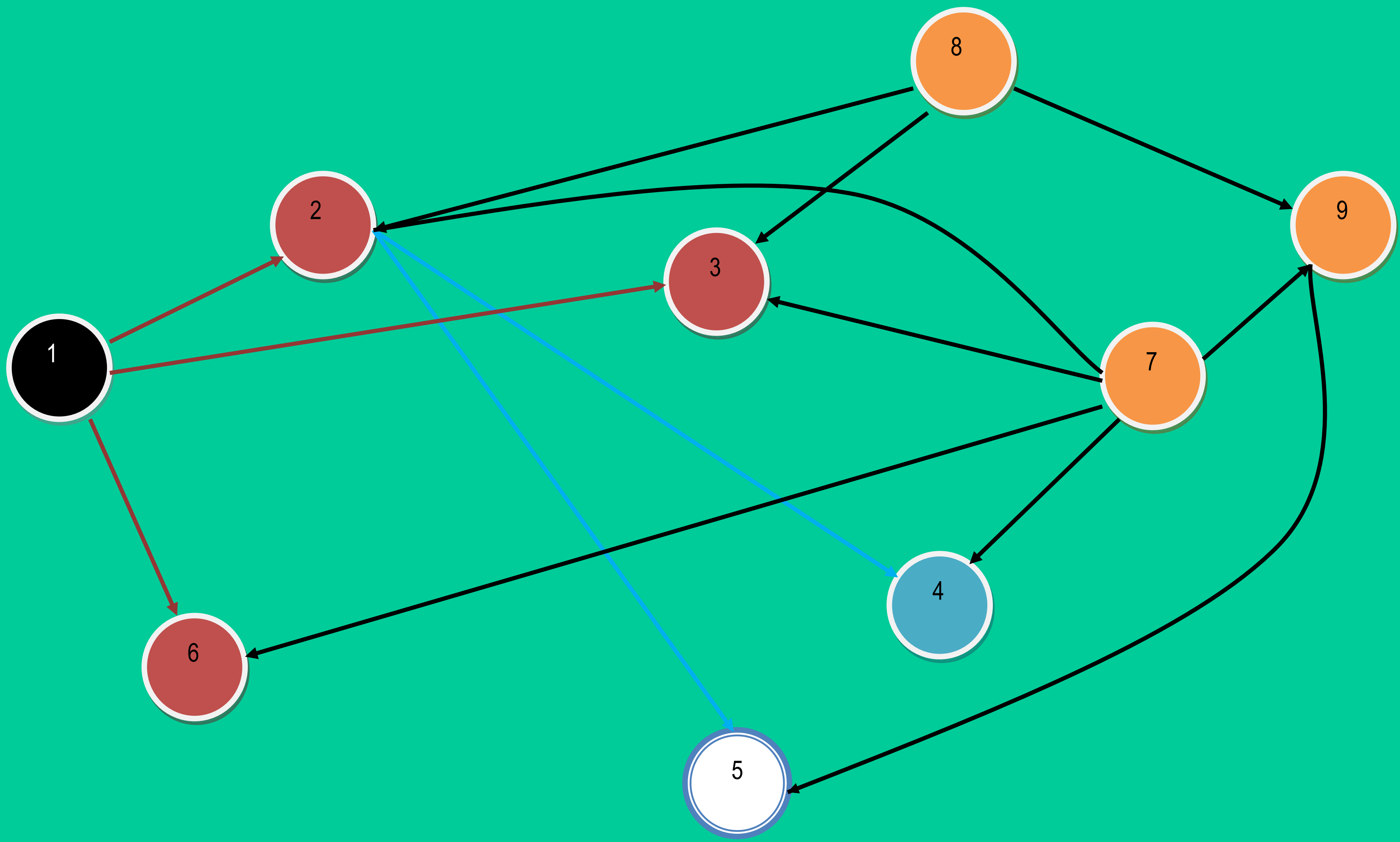

8
2
9
3
1
7
4
6
5

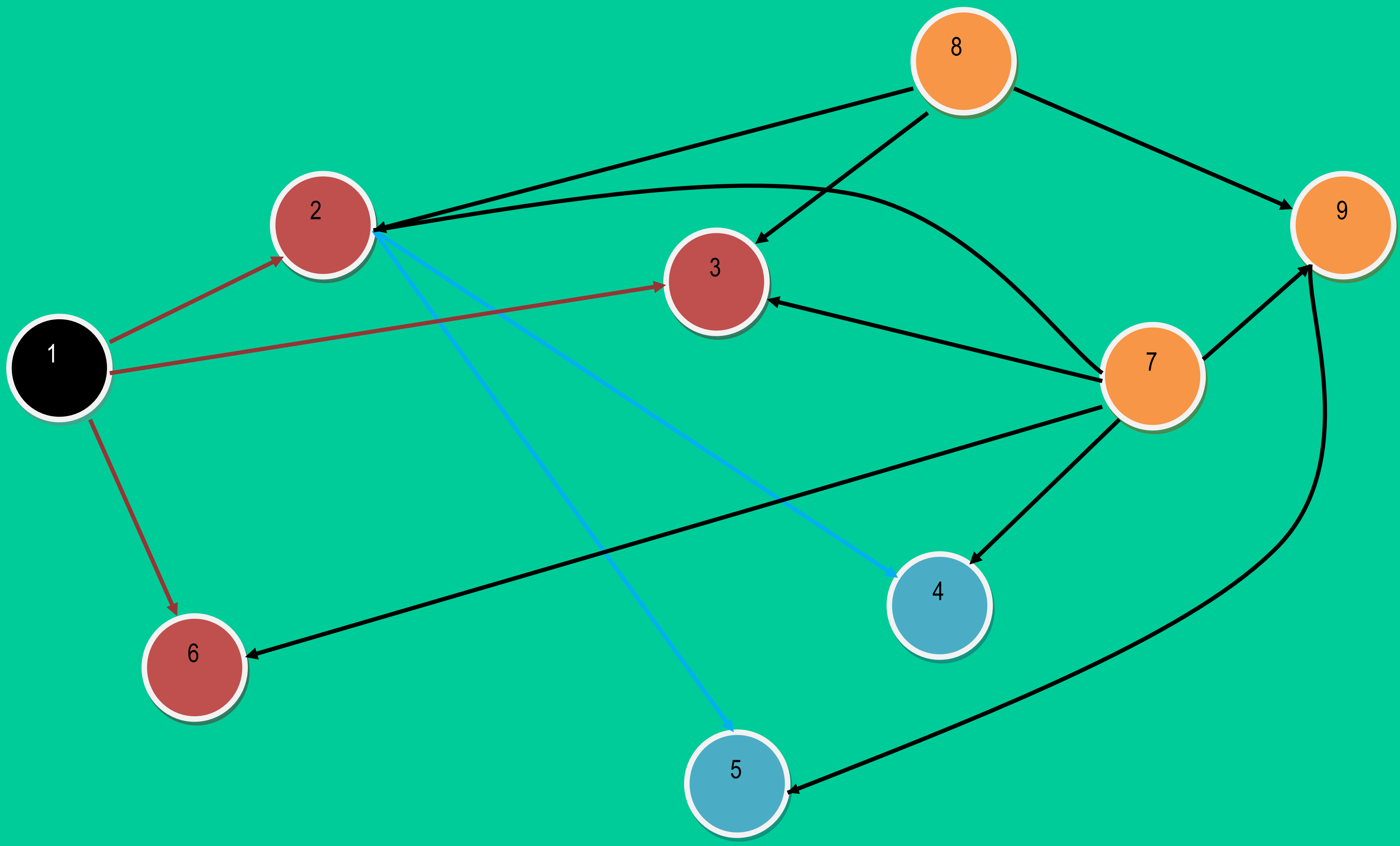

1
2
3
4
5
6
7
8
9

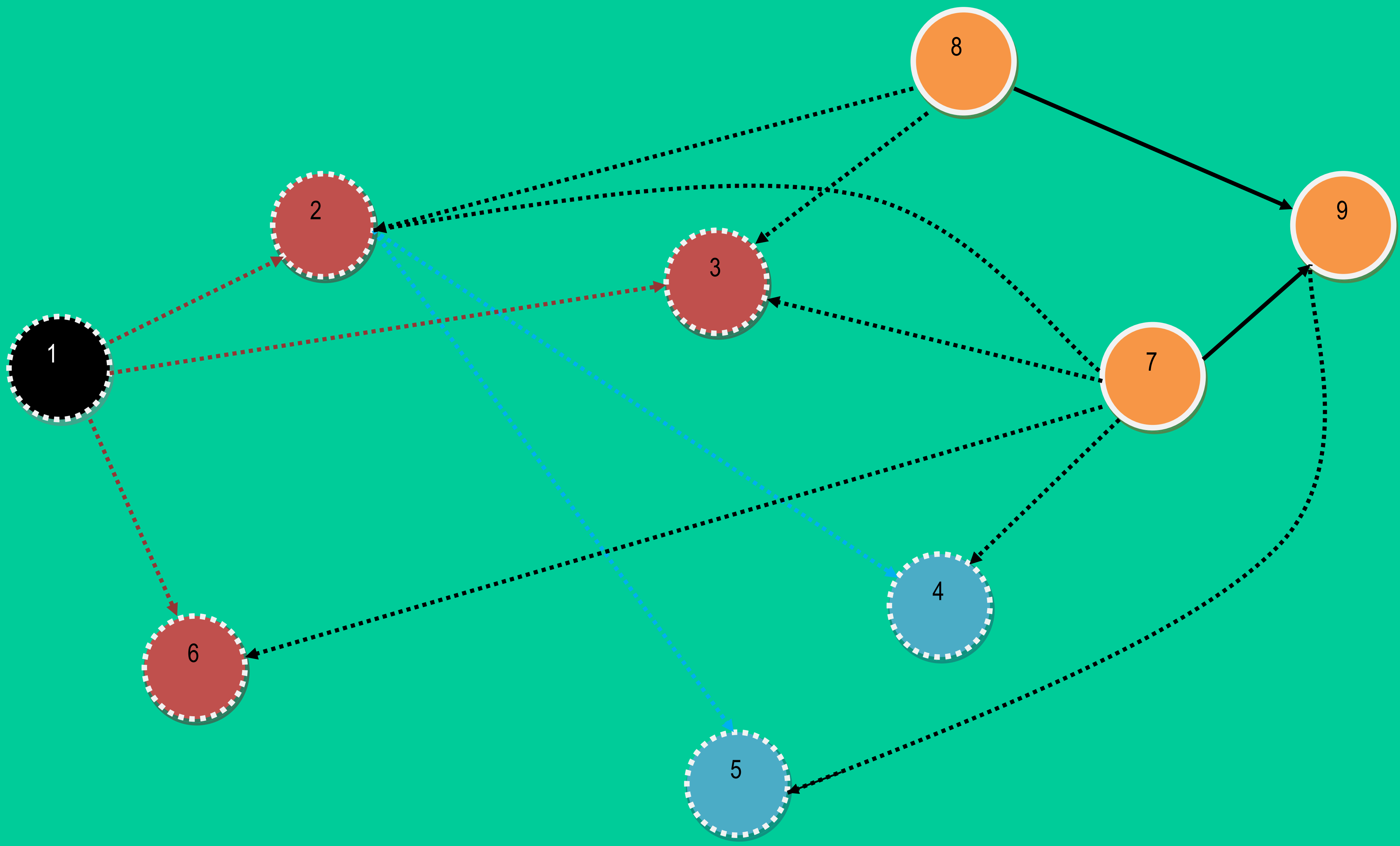

1
2
3
4
5
6
7
8
9

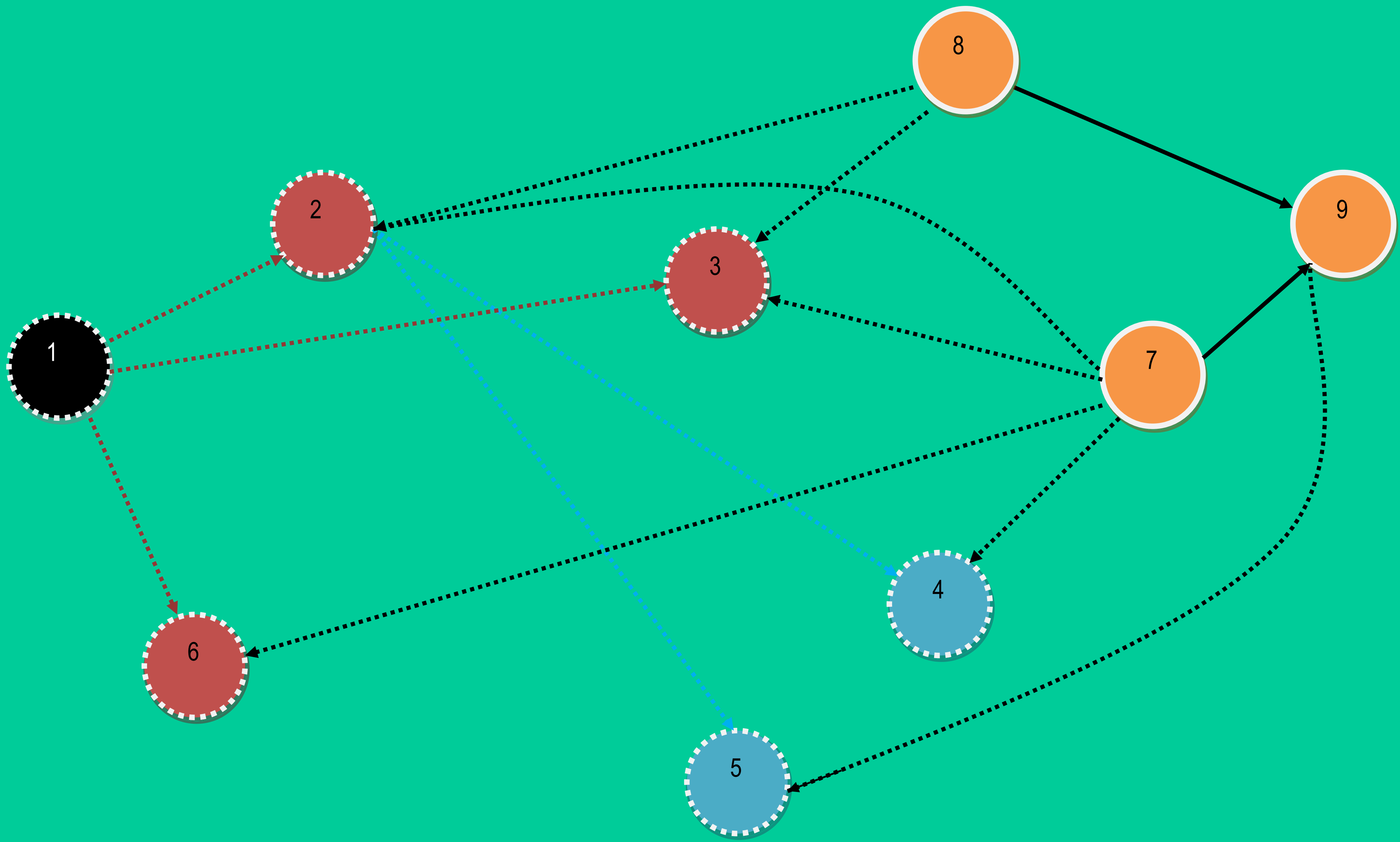

8
2
9
3
1
7
4
6
5

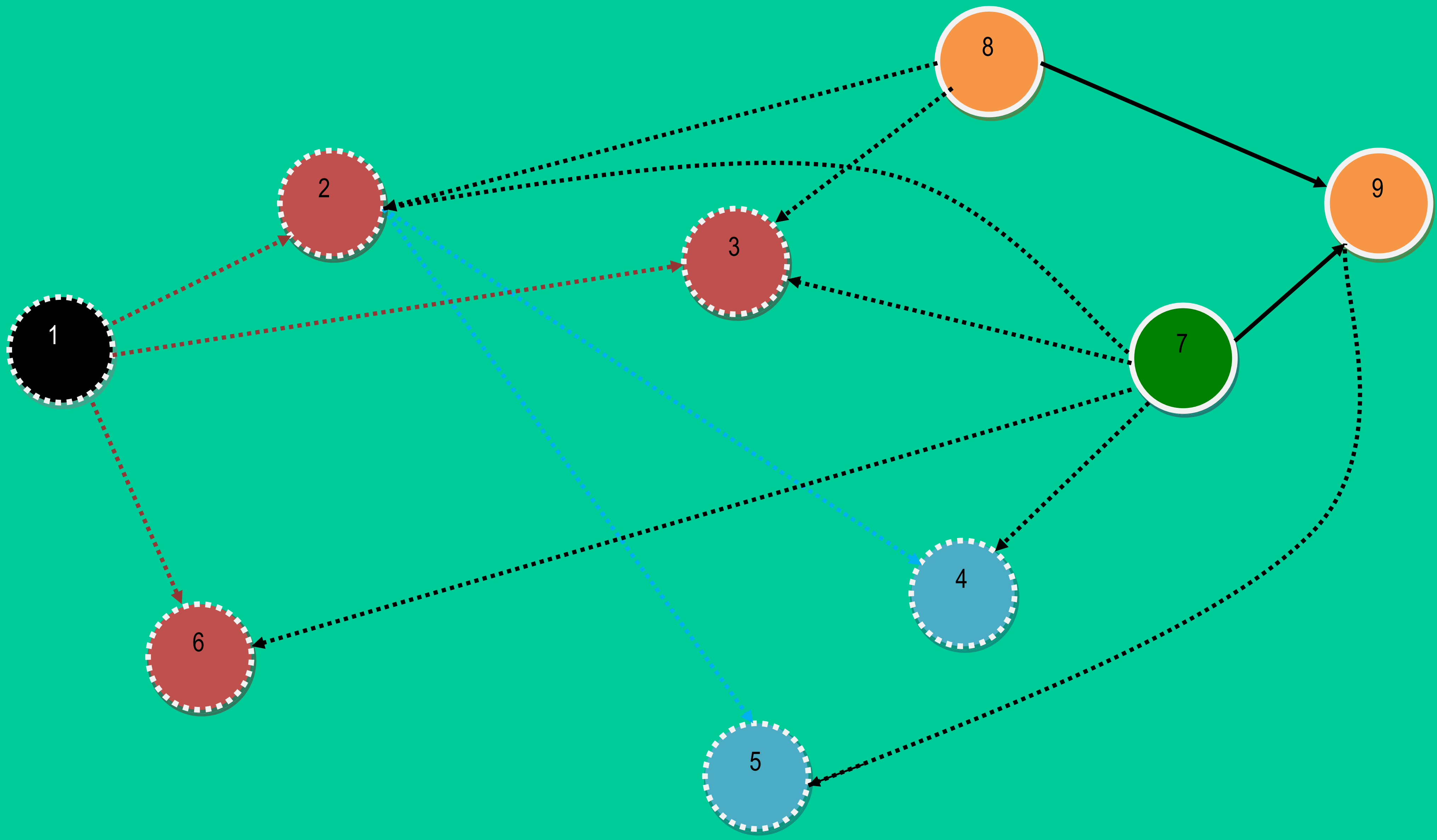

8
2
3
9
1
7
4
6
5

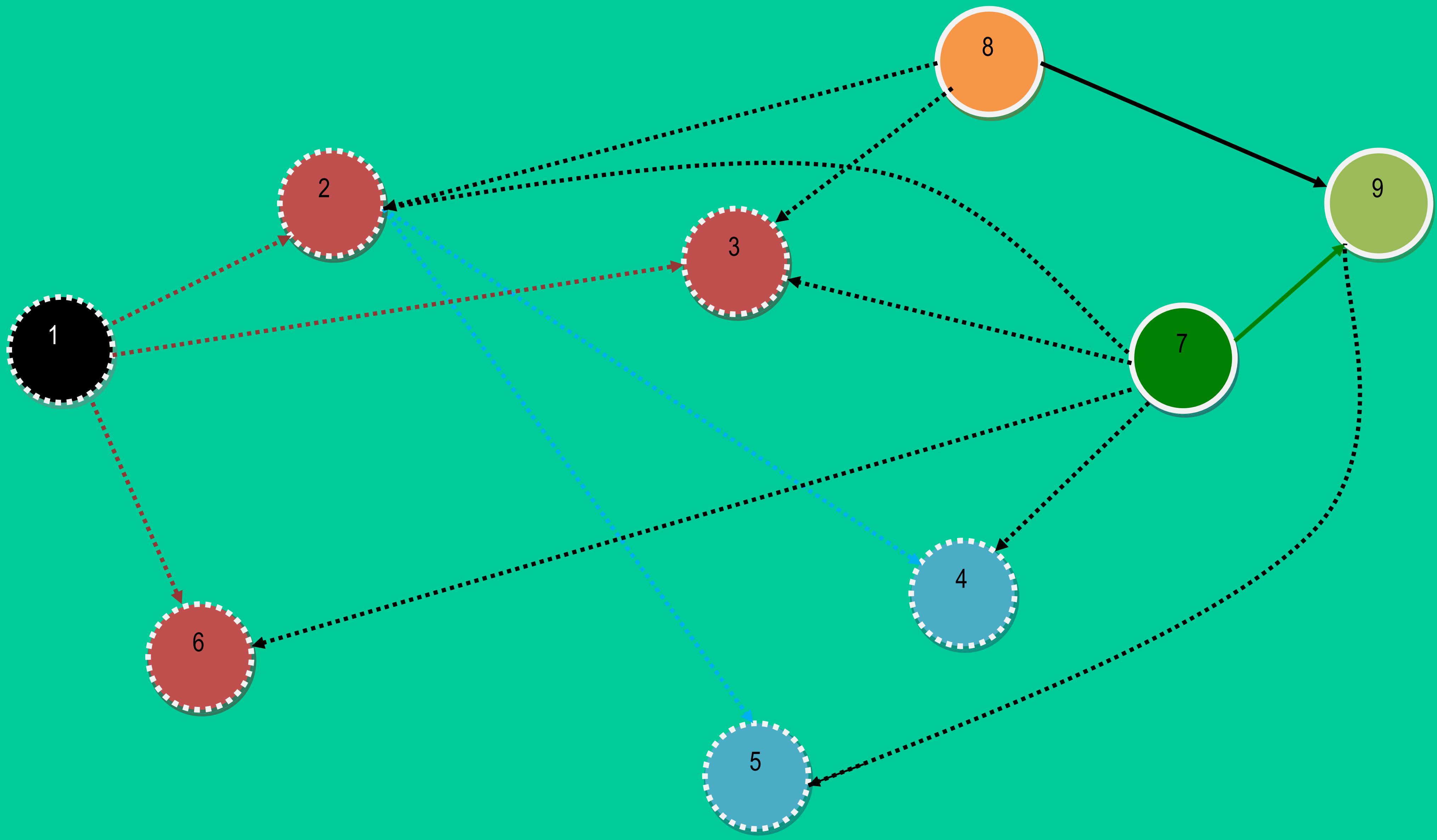

8
2
3
9
1
7
4
6
5

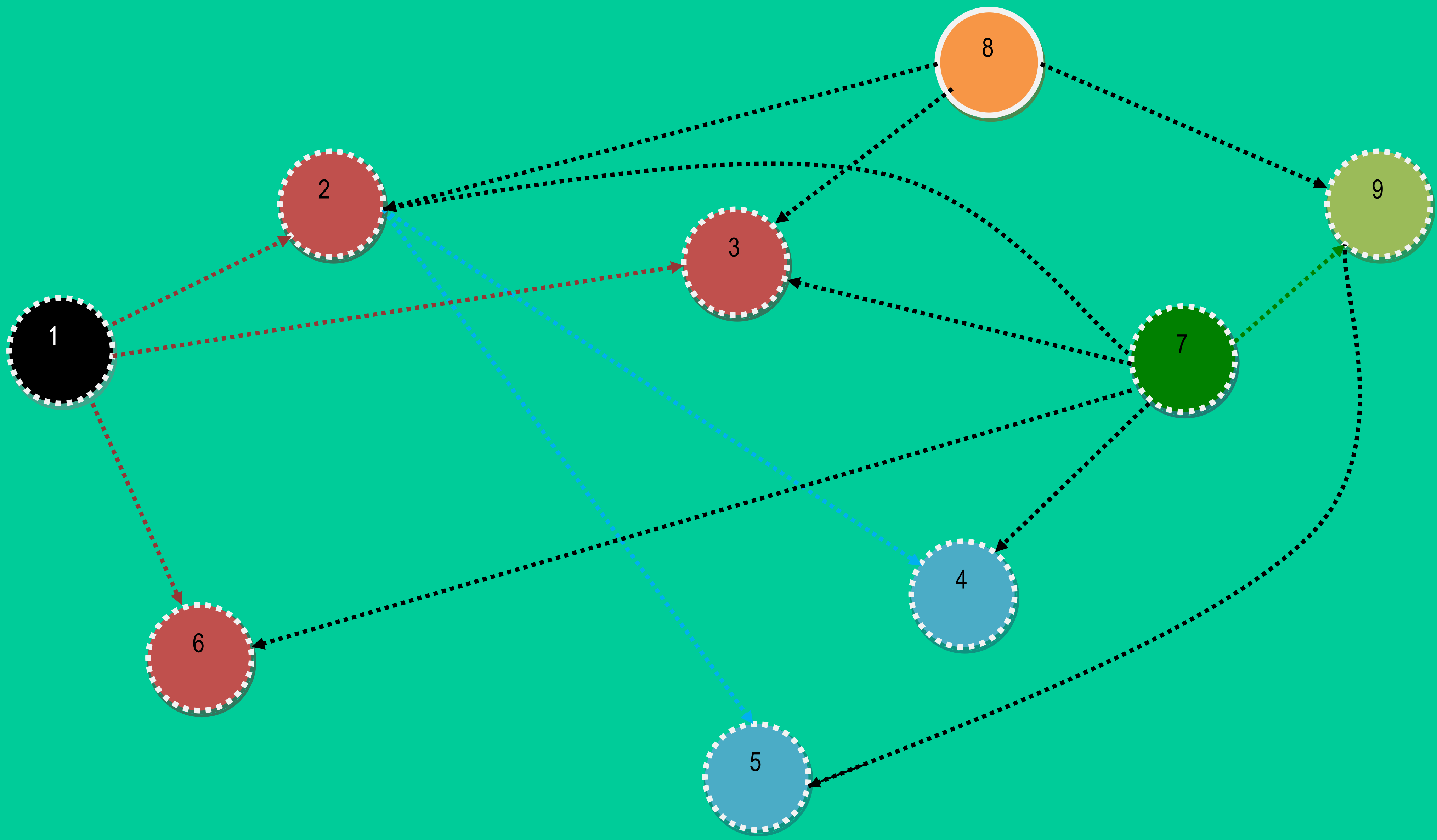

8
2
3
9
1
7
4
6
5

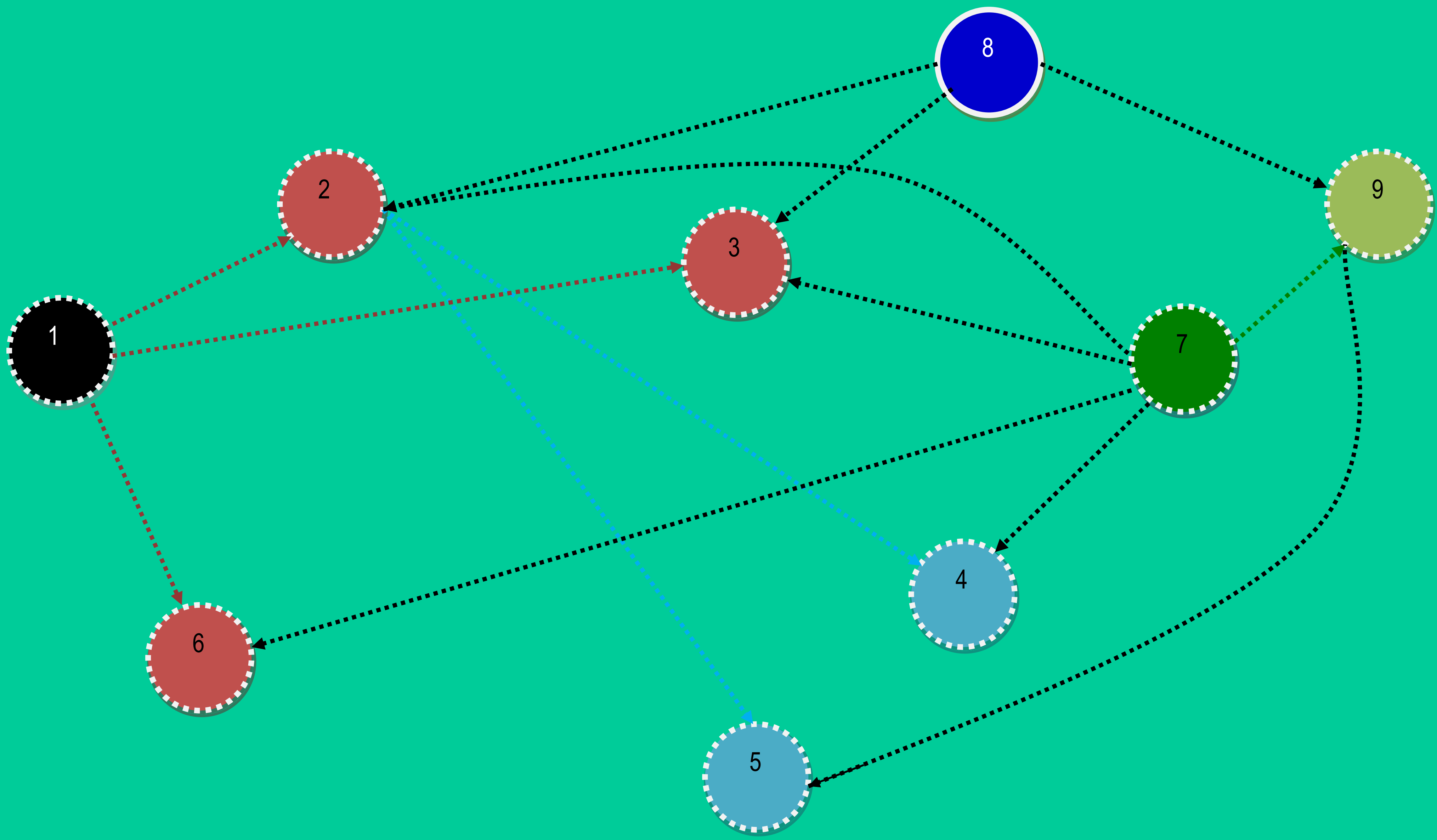

8
2
3
9
1
7
4
6
5

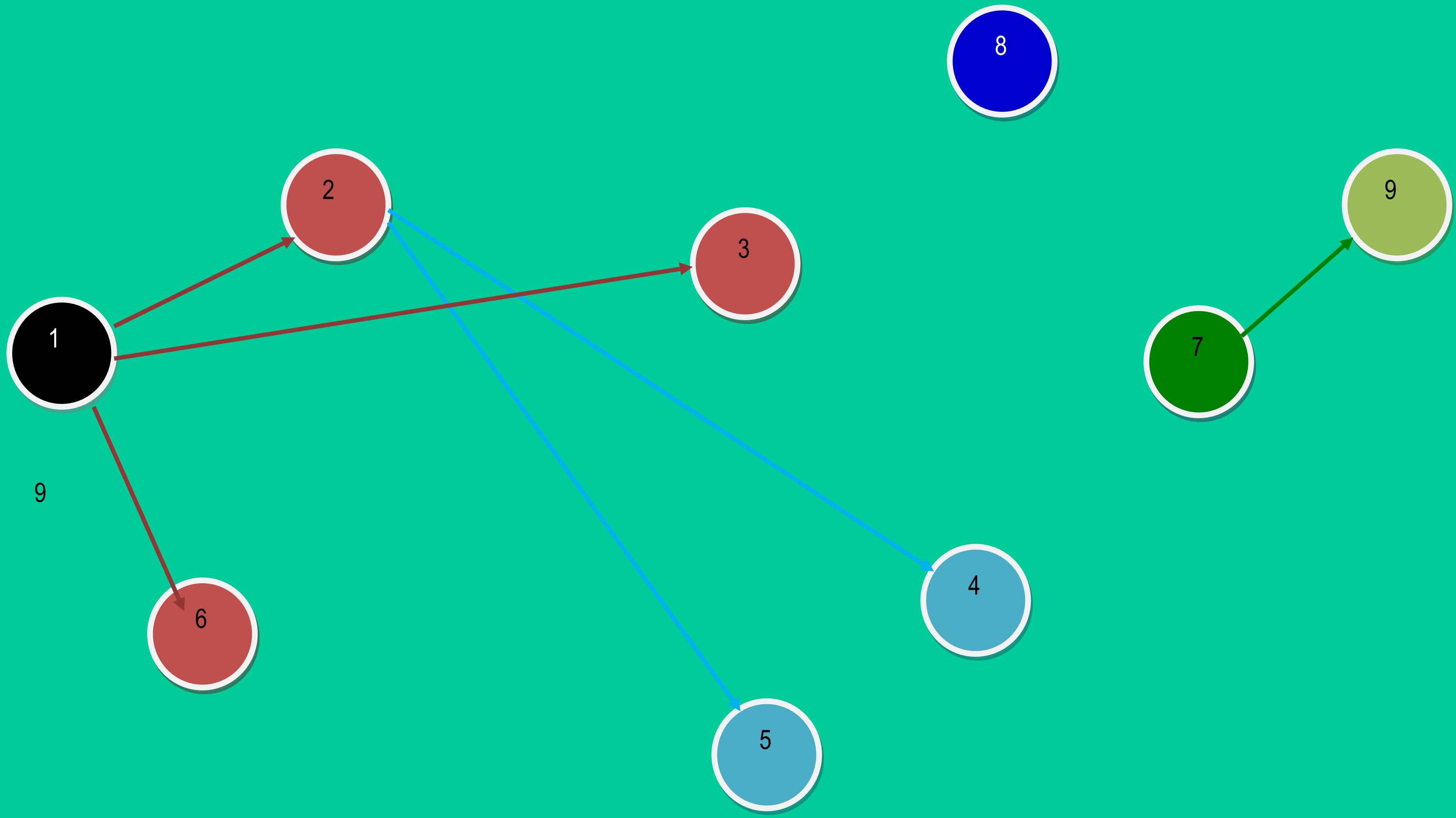

8
2
3
9
1
7
4
6
5


1
2
3
4
5
6
7
8
9


8
2
3
9
1
7
4
6
5


8
2
3
9
1
7
4
6
5


8
2
3
9
1
7
4
6
5


8
2
9
3
1
7
9
4
6
5

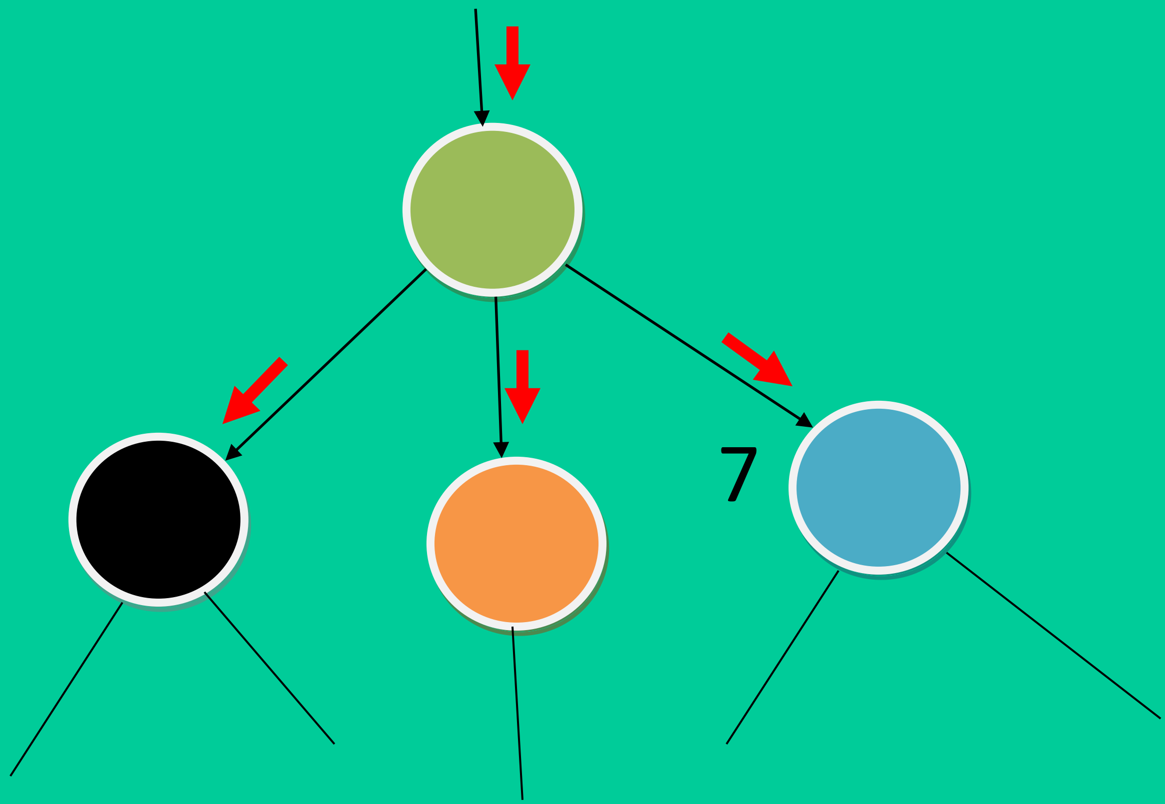
Ordre d'exploration

Deux ordres naturels d'exploration sont inclus dans le parcours en profondeur:

- l'ordre **préfixe**
- l'ordre **suffixe**.

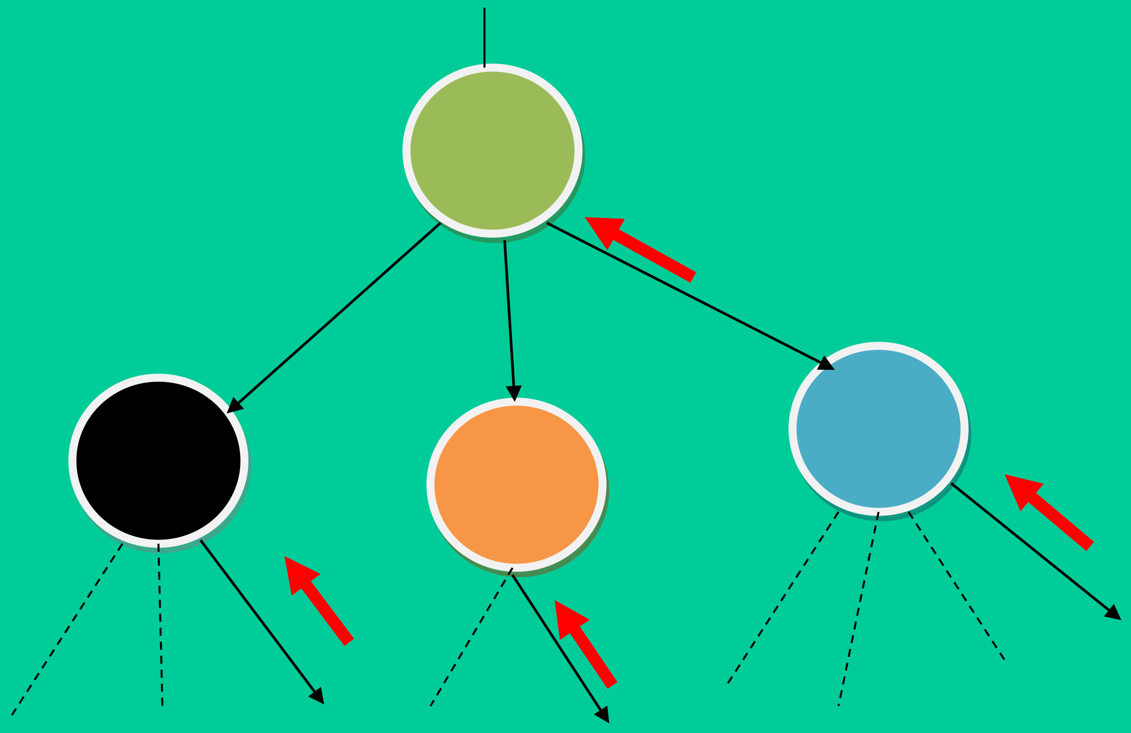
1-l'ordre préfixe:

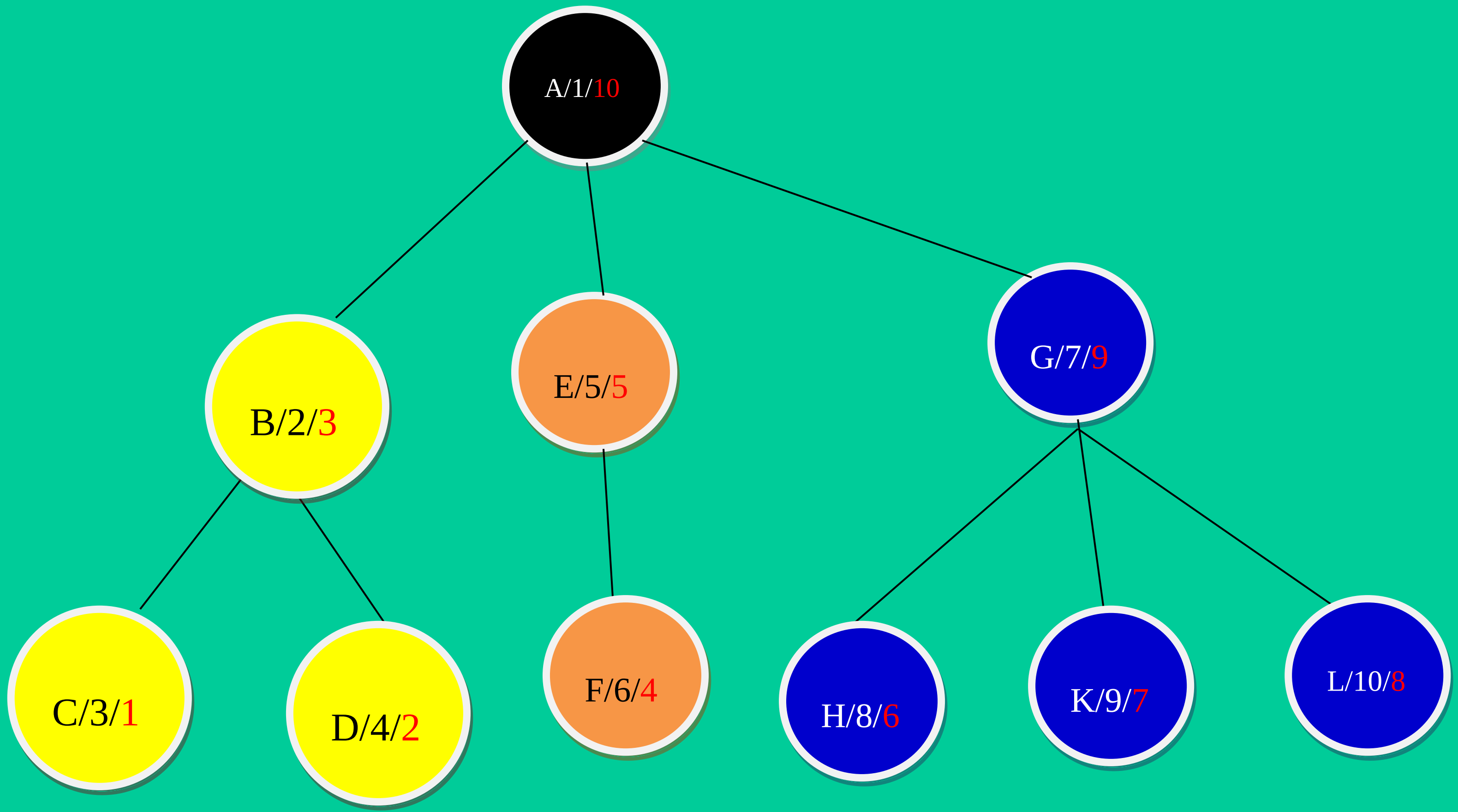
Chaque nœud n'est pris en compte que lors du **premier passage**.



2-l’ordre suffixe:

Chaque nœud n’est pris en compte que lors du **dernier passage**.

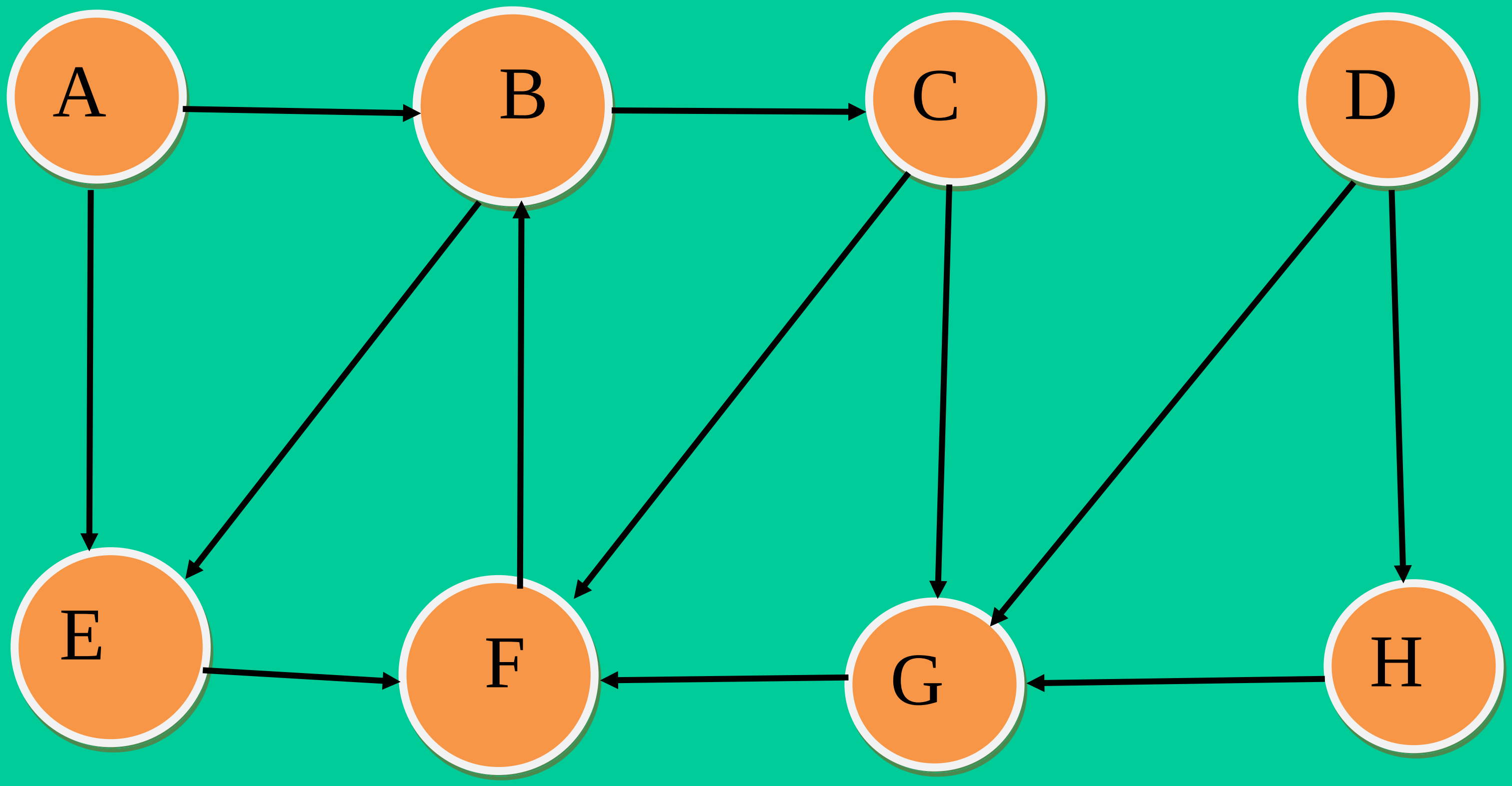


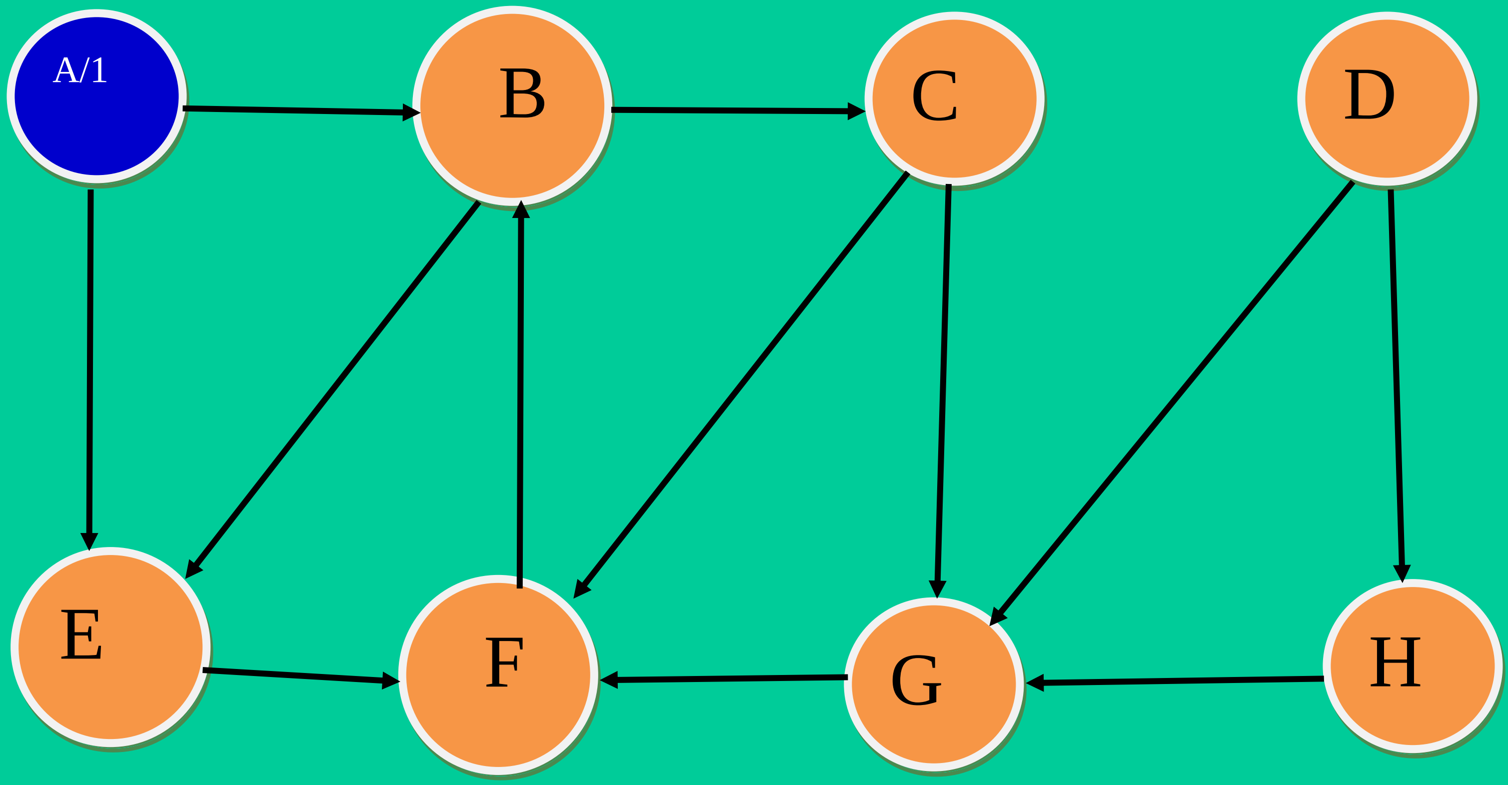


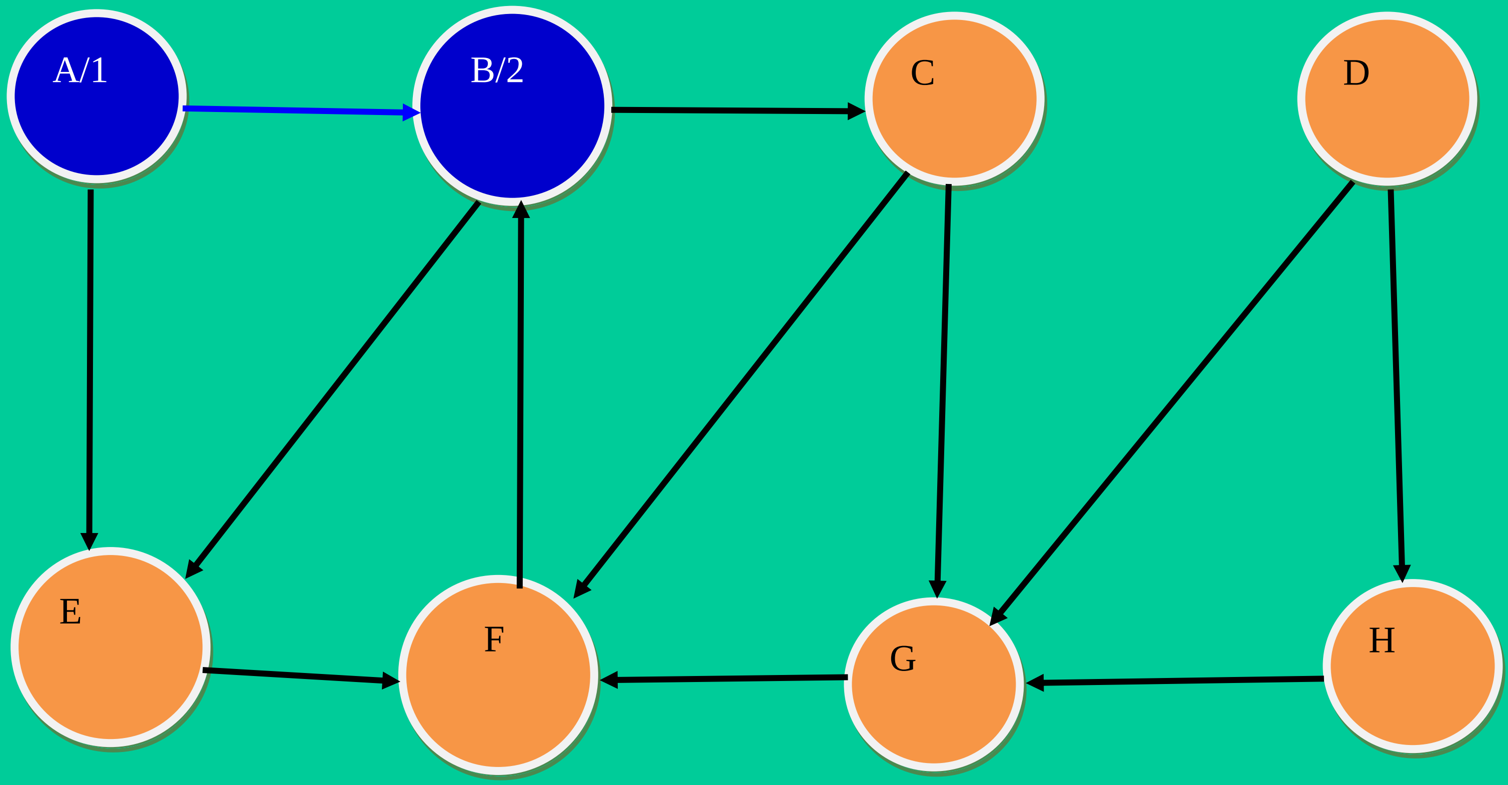
Ordre préfixe : **A B C D E F G H K L**

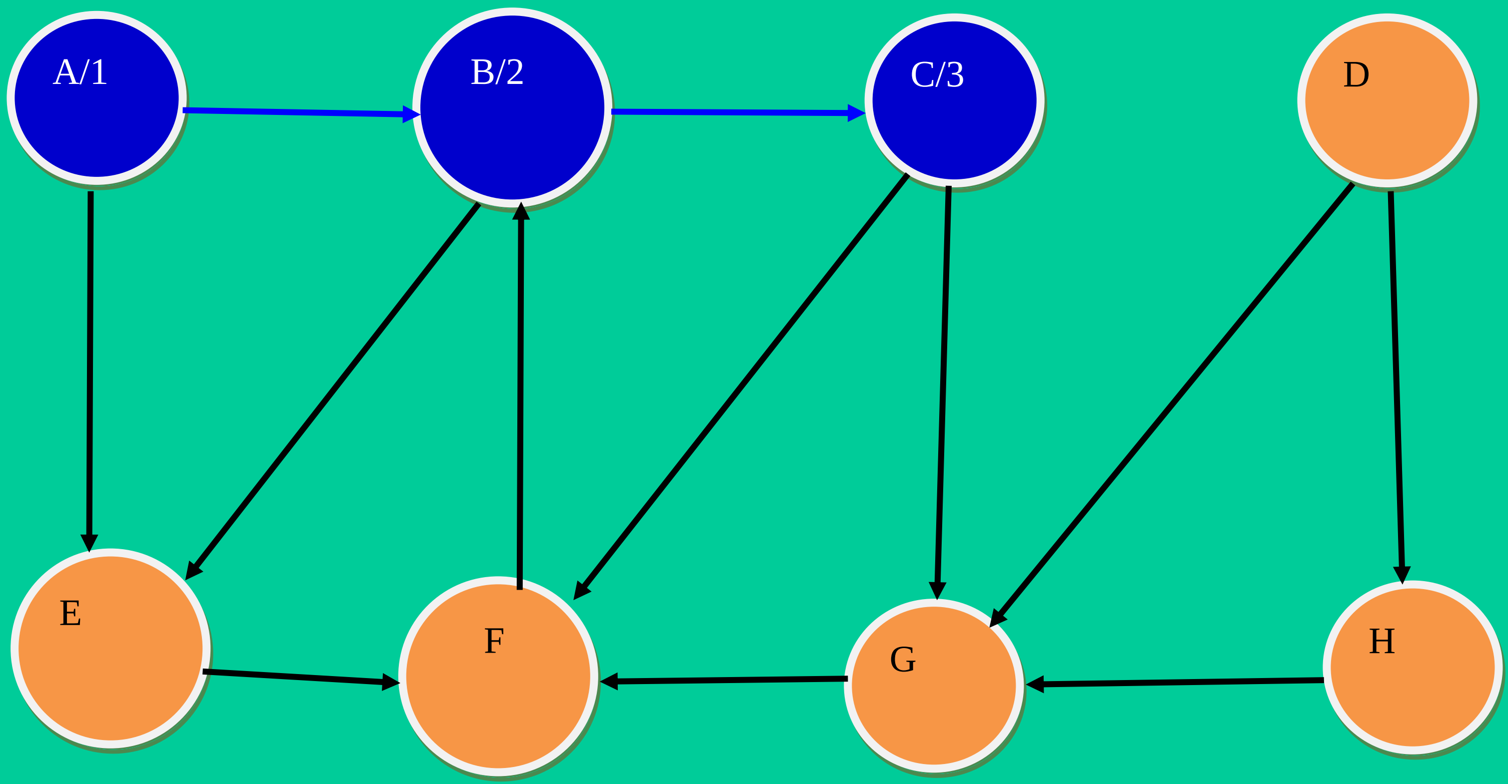
Ordre suffixe : **C D B F E H K L G A**

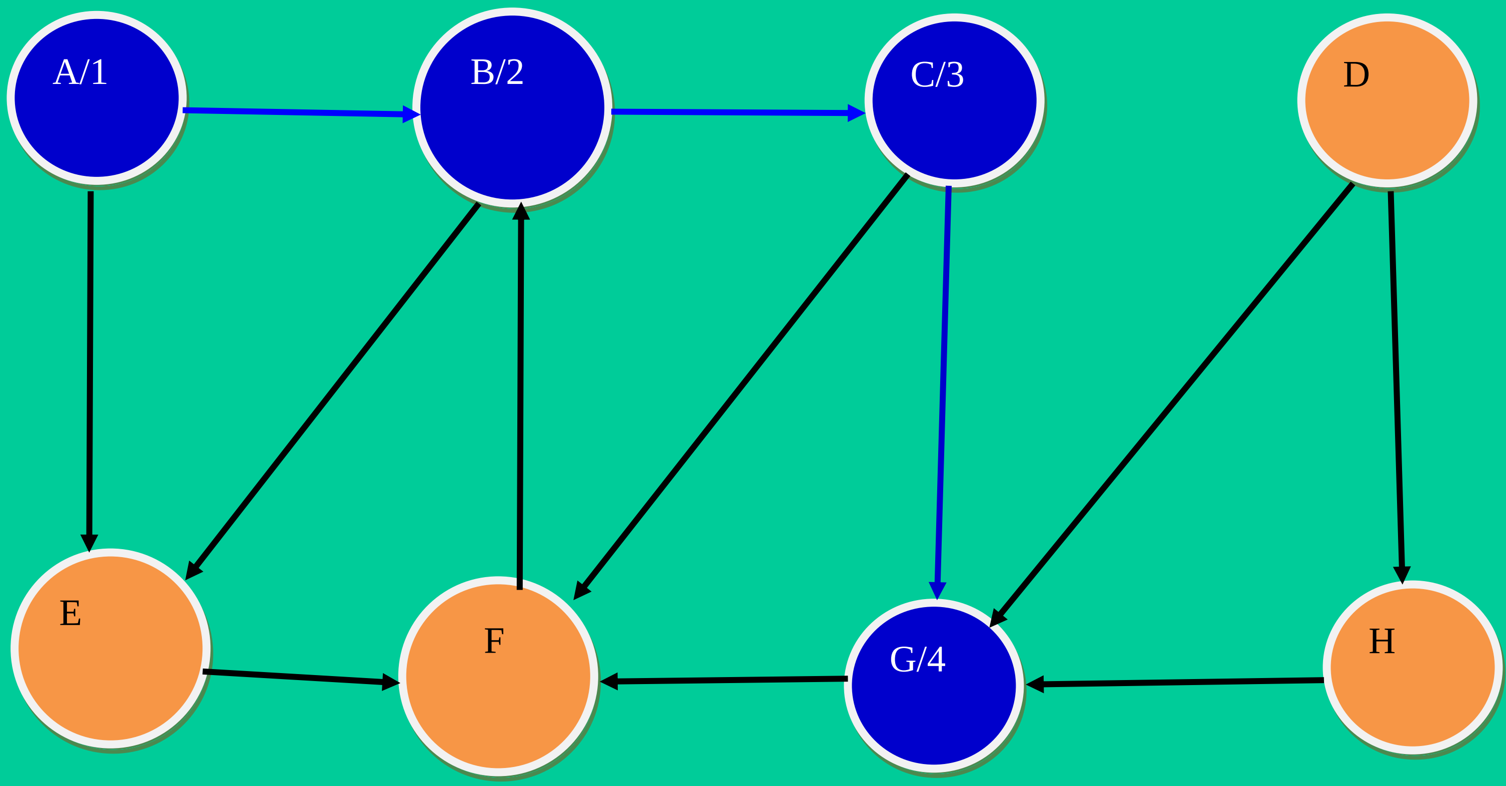
3-l'ordre date début / date fin

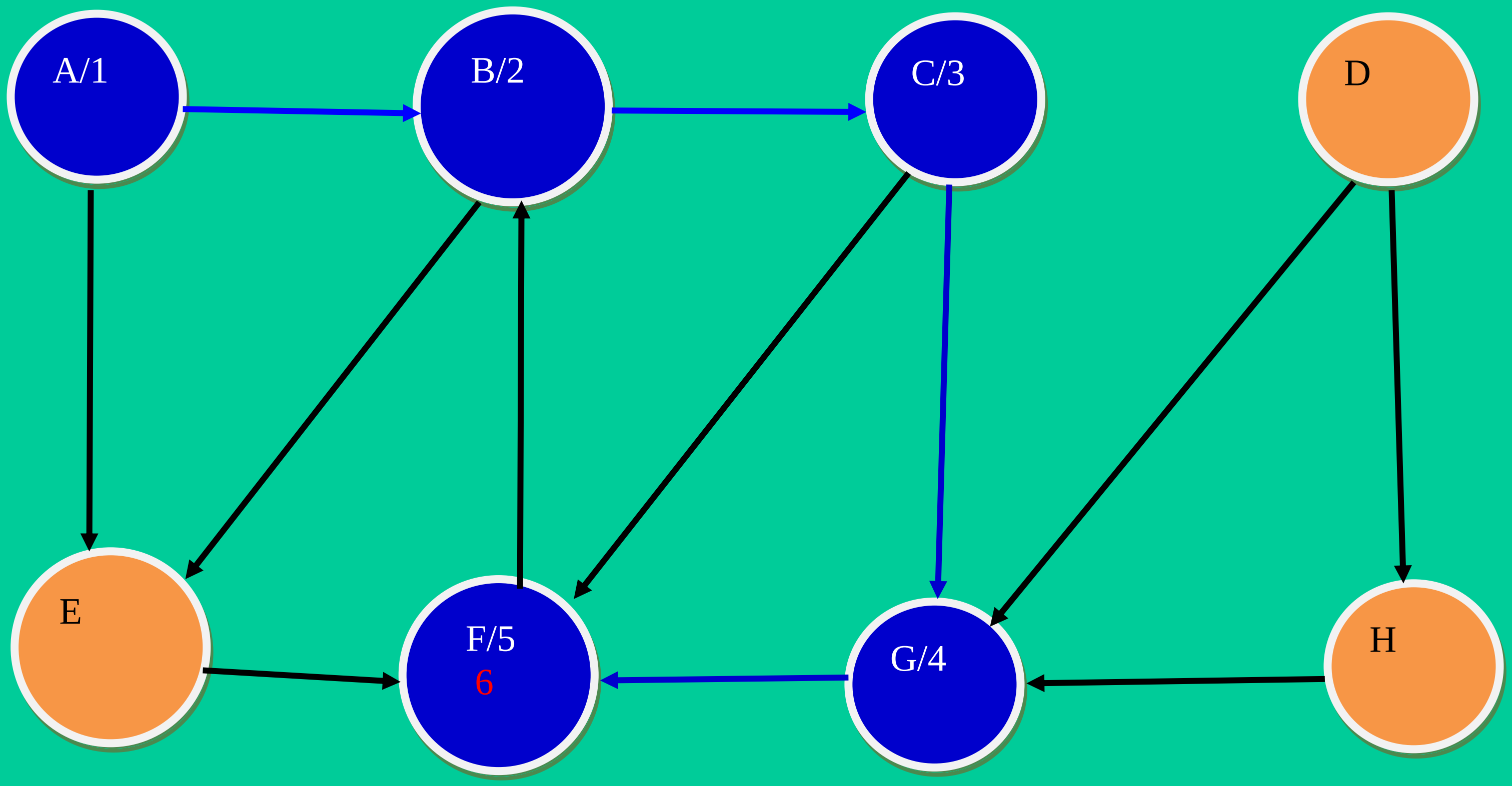


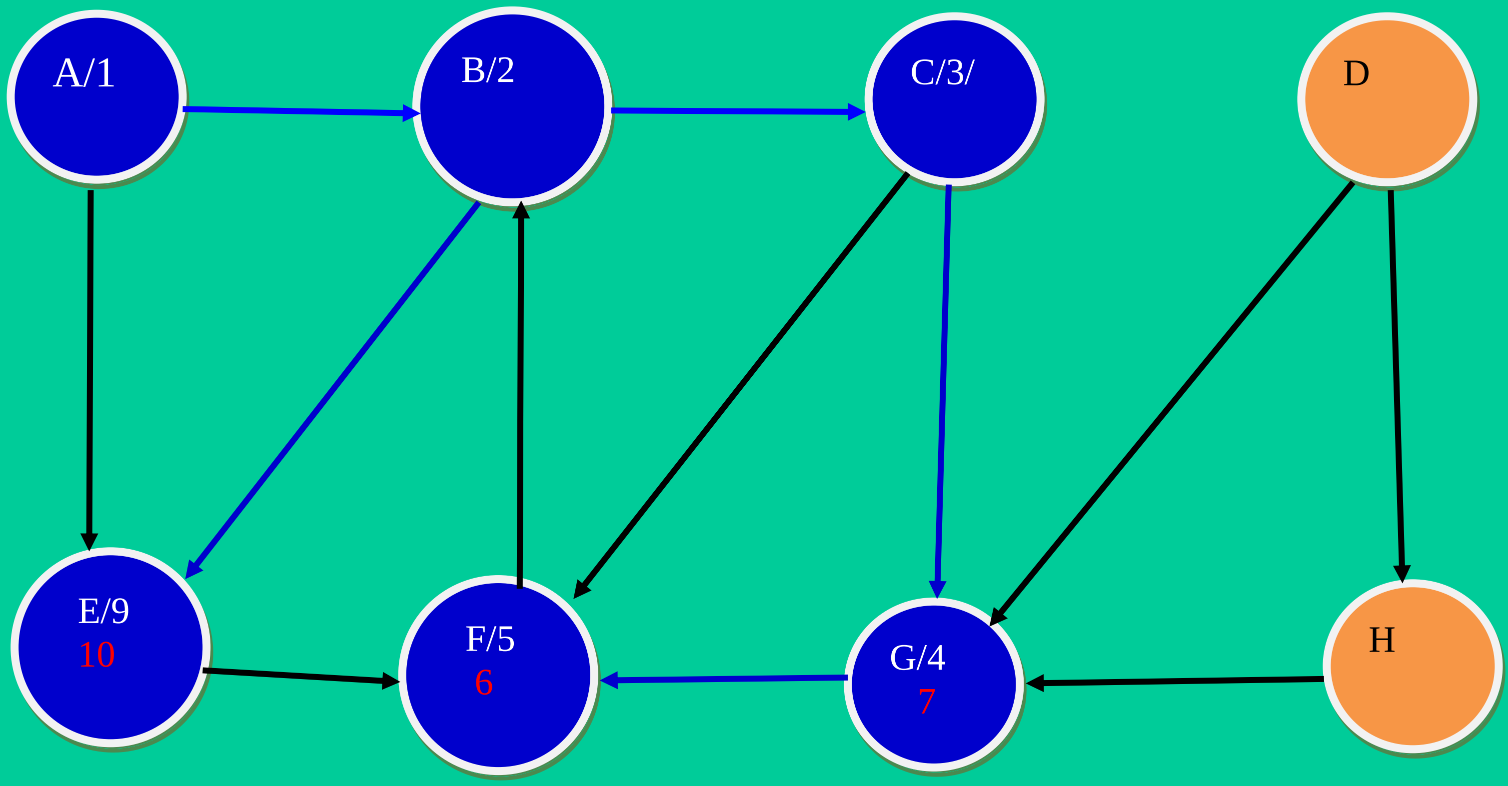

A/1
B
C
D
E
F
G
H

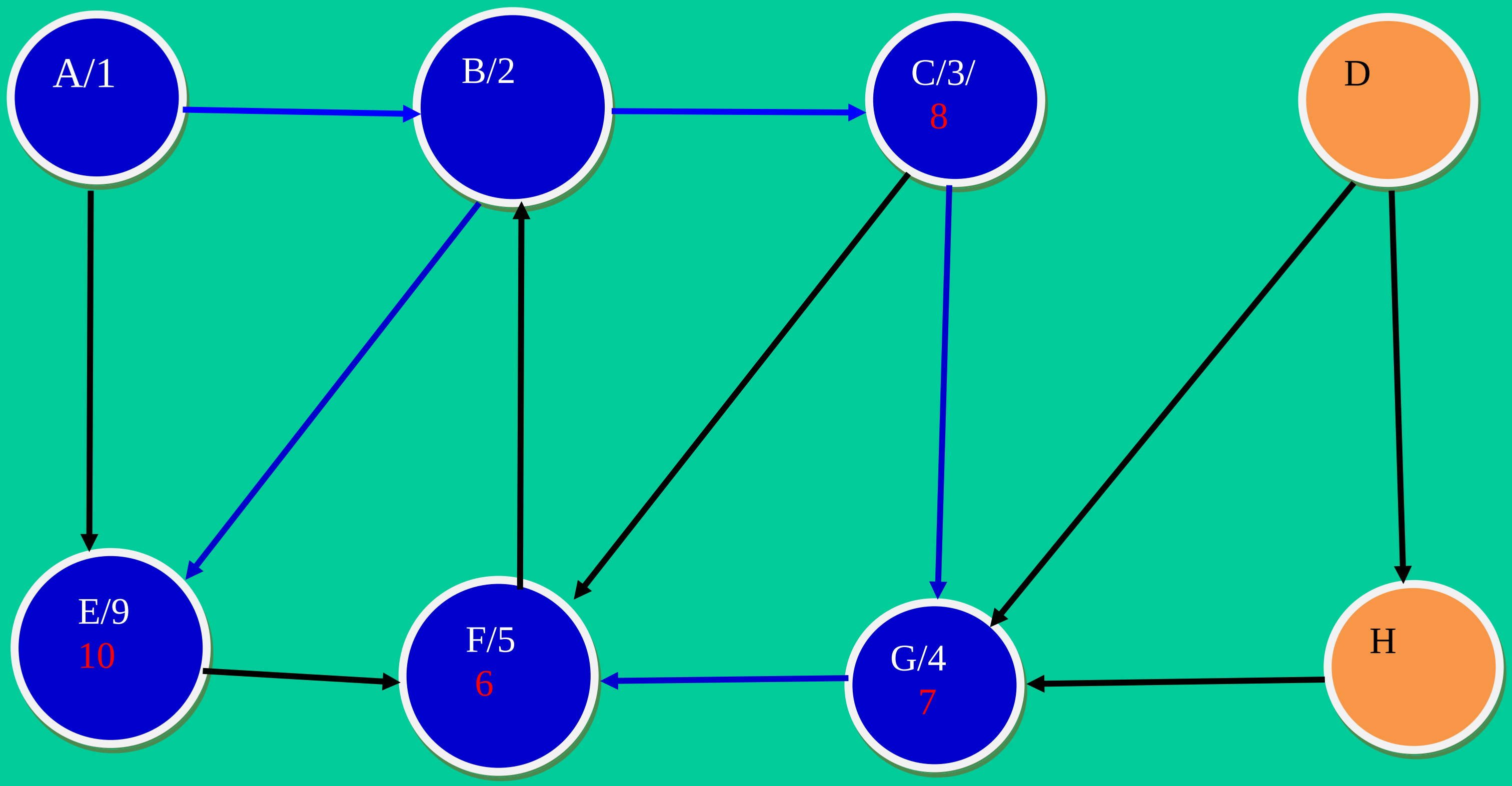

A/1
B/2
C
D
E
F
G
H

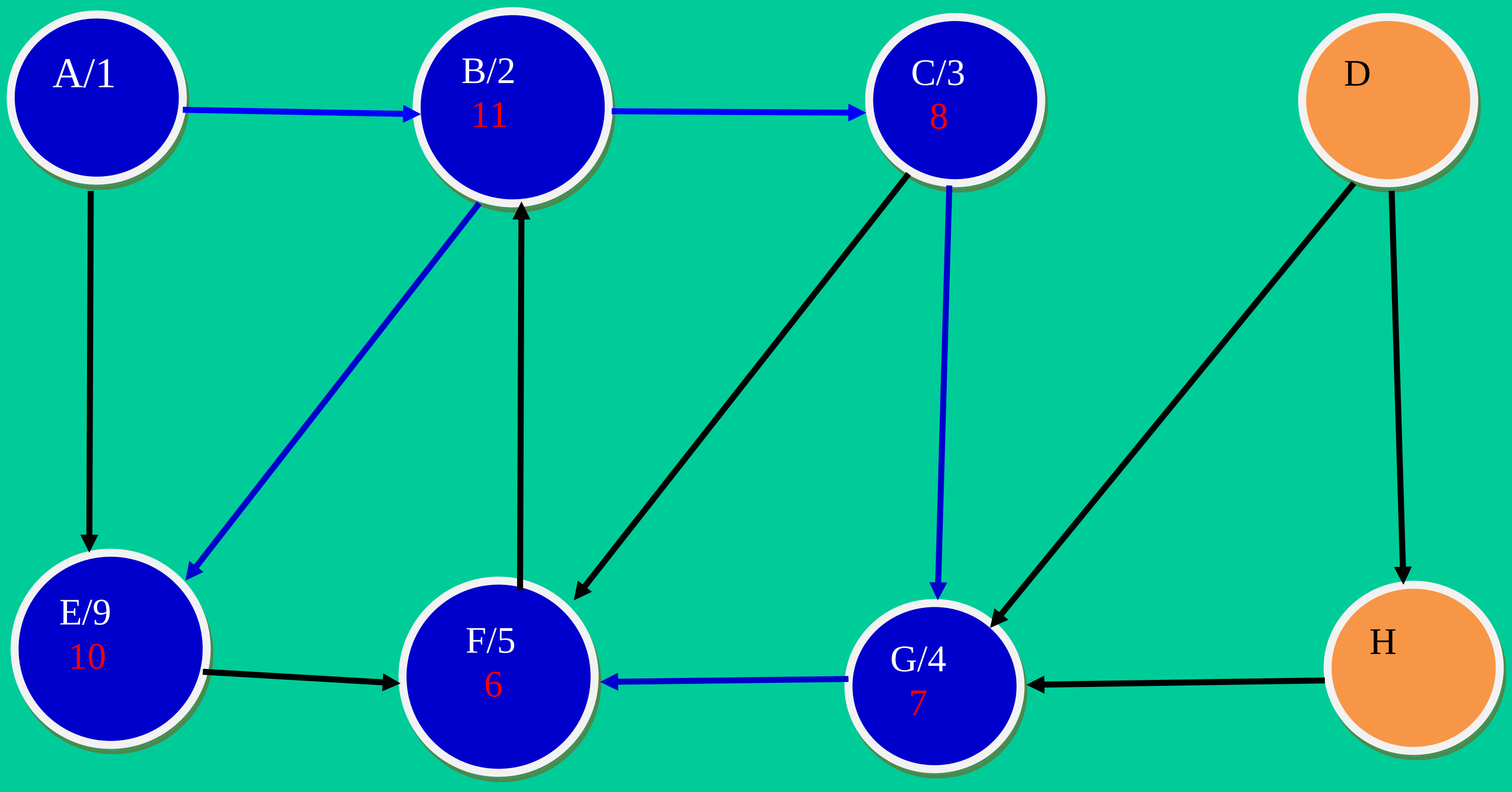

A/1
B/2
C/3
D
E
F
G
H

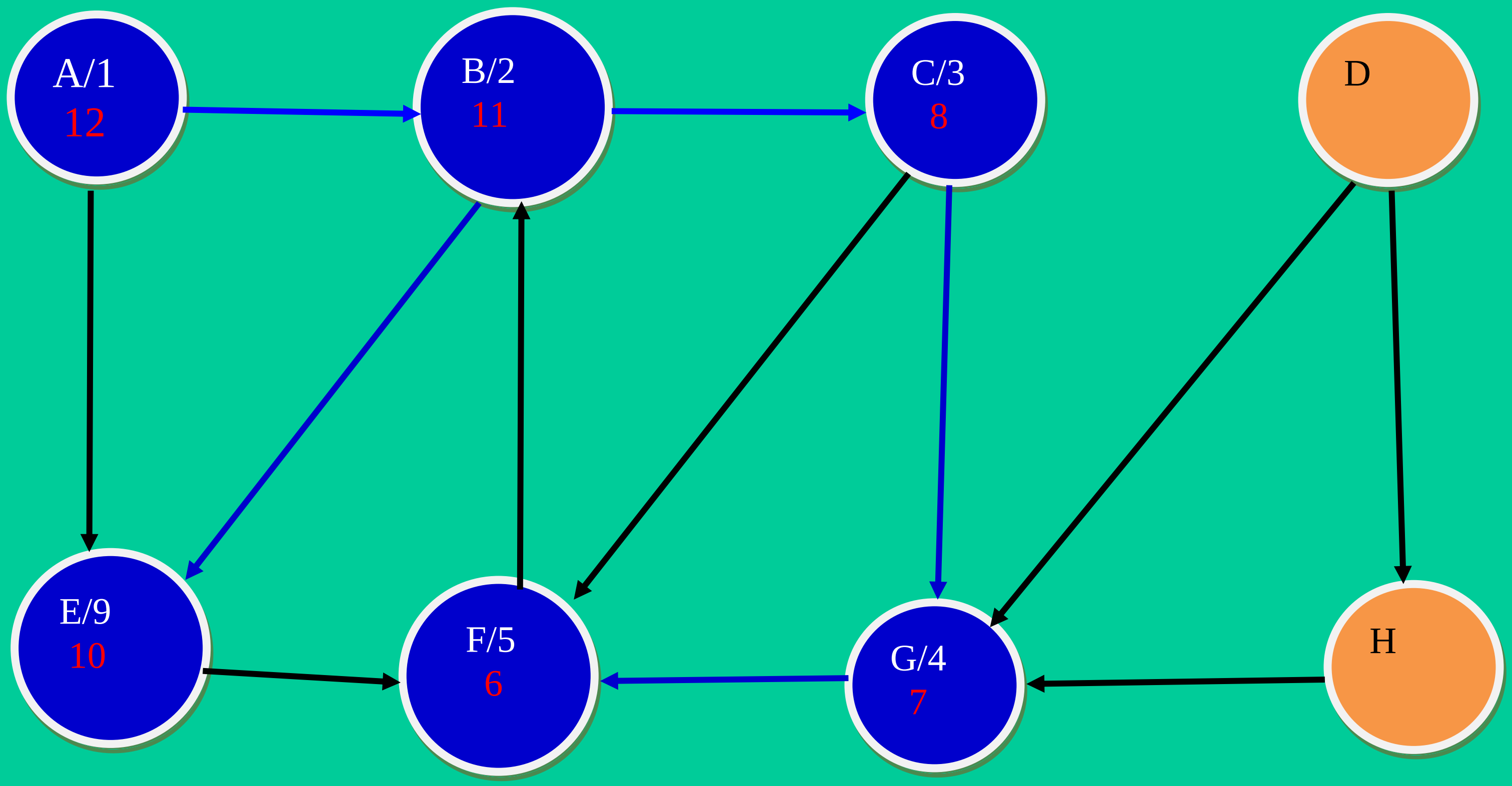

A/1
B/2
C/3
D
E
F
G/4
H

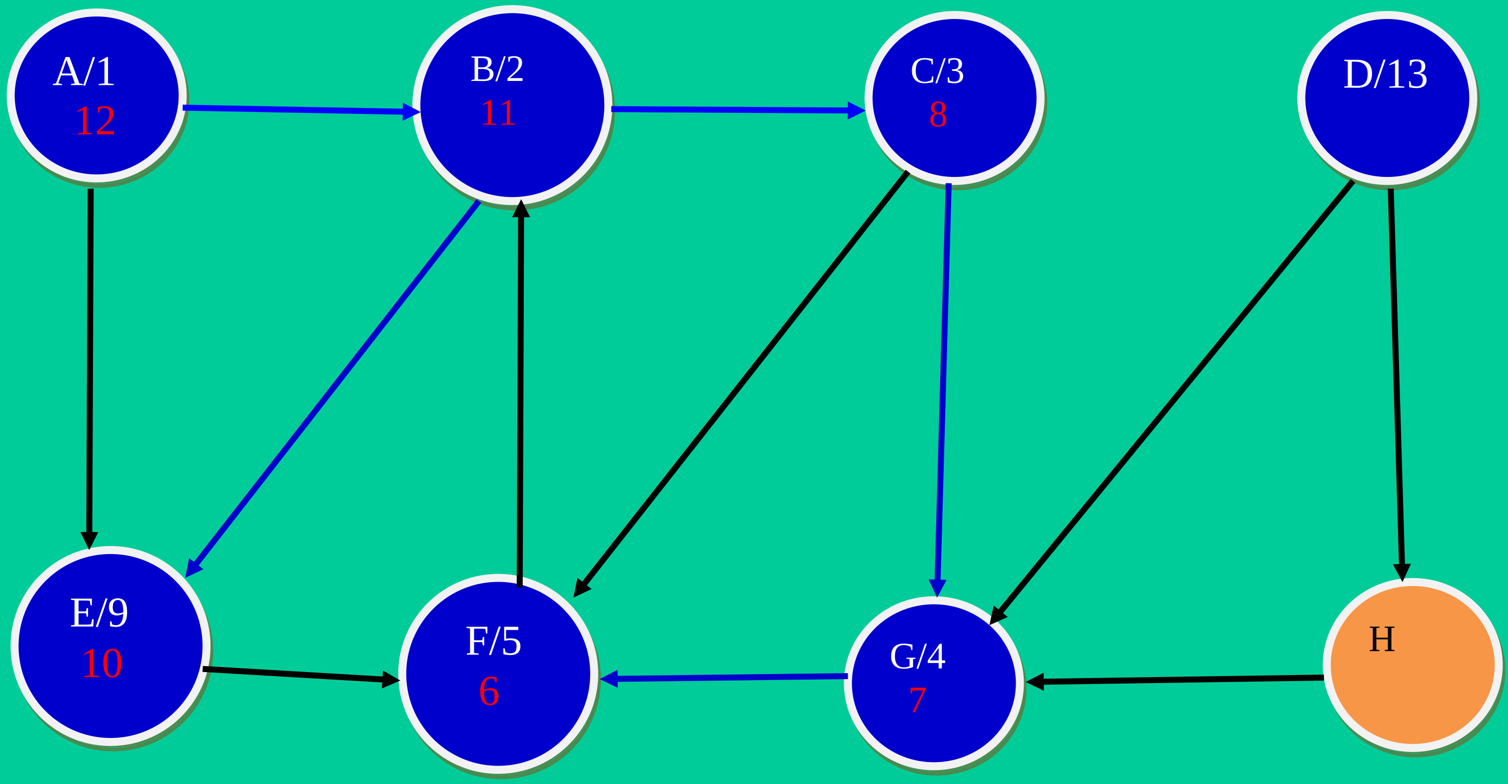

A/1
B/2
C/3
D
E
F/5
6
G/4
H

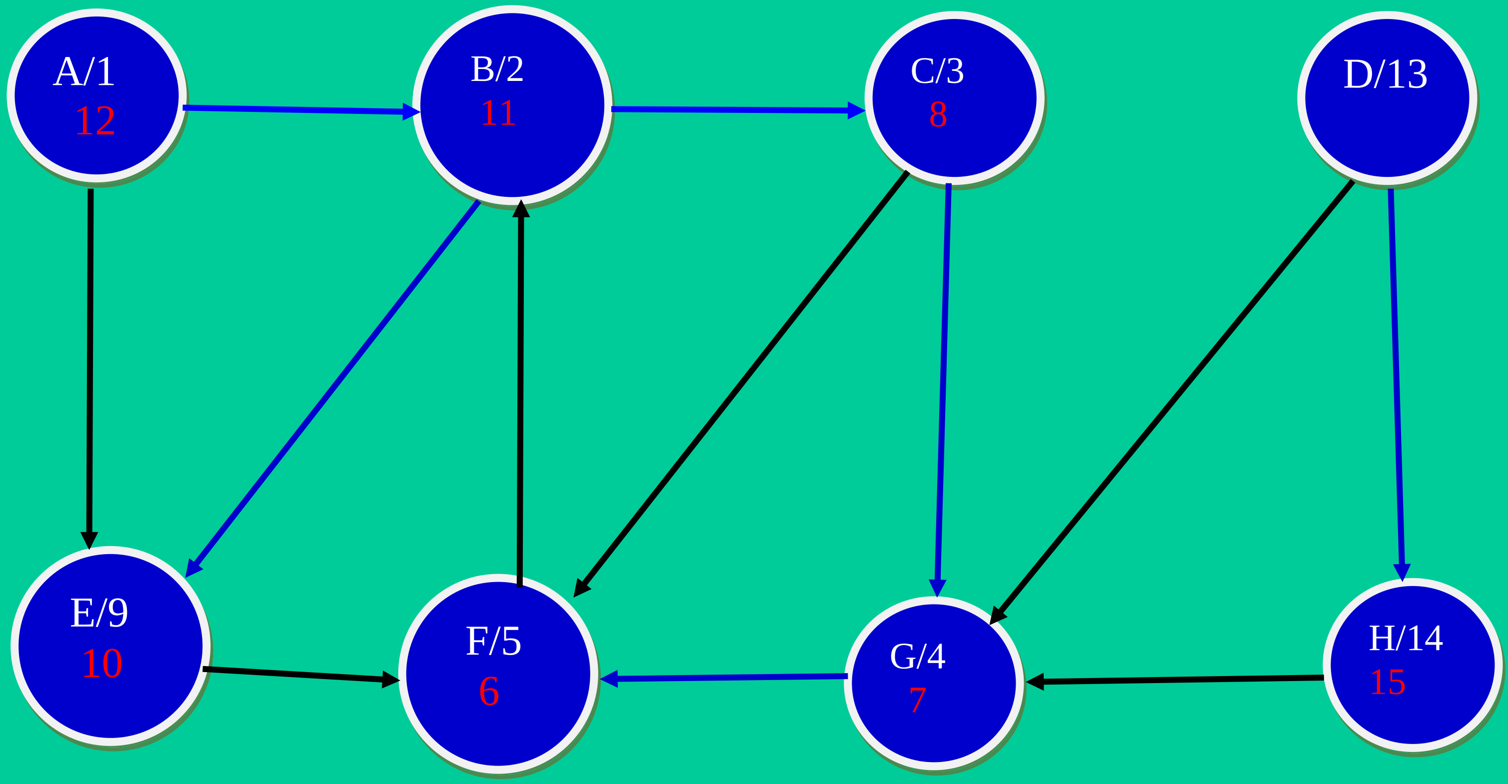

A/1
B/2
C/3/
D
E/9
10
F/5
6
G/4
7
H

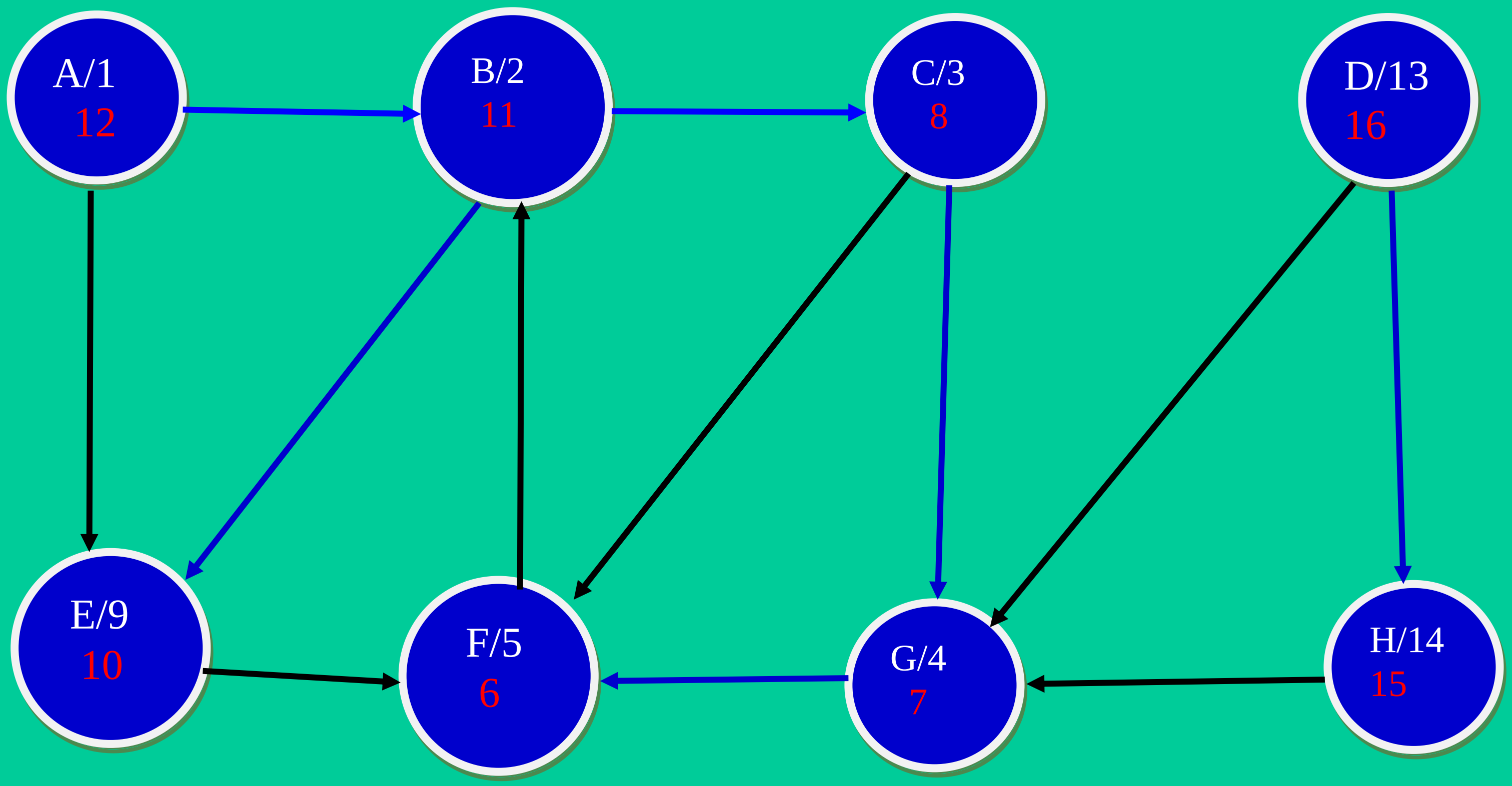

A/1
B/2
C/3/
8
D
E/9
10
F/5
6
G/4
7
H

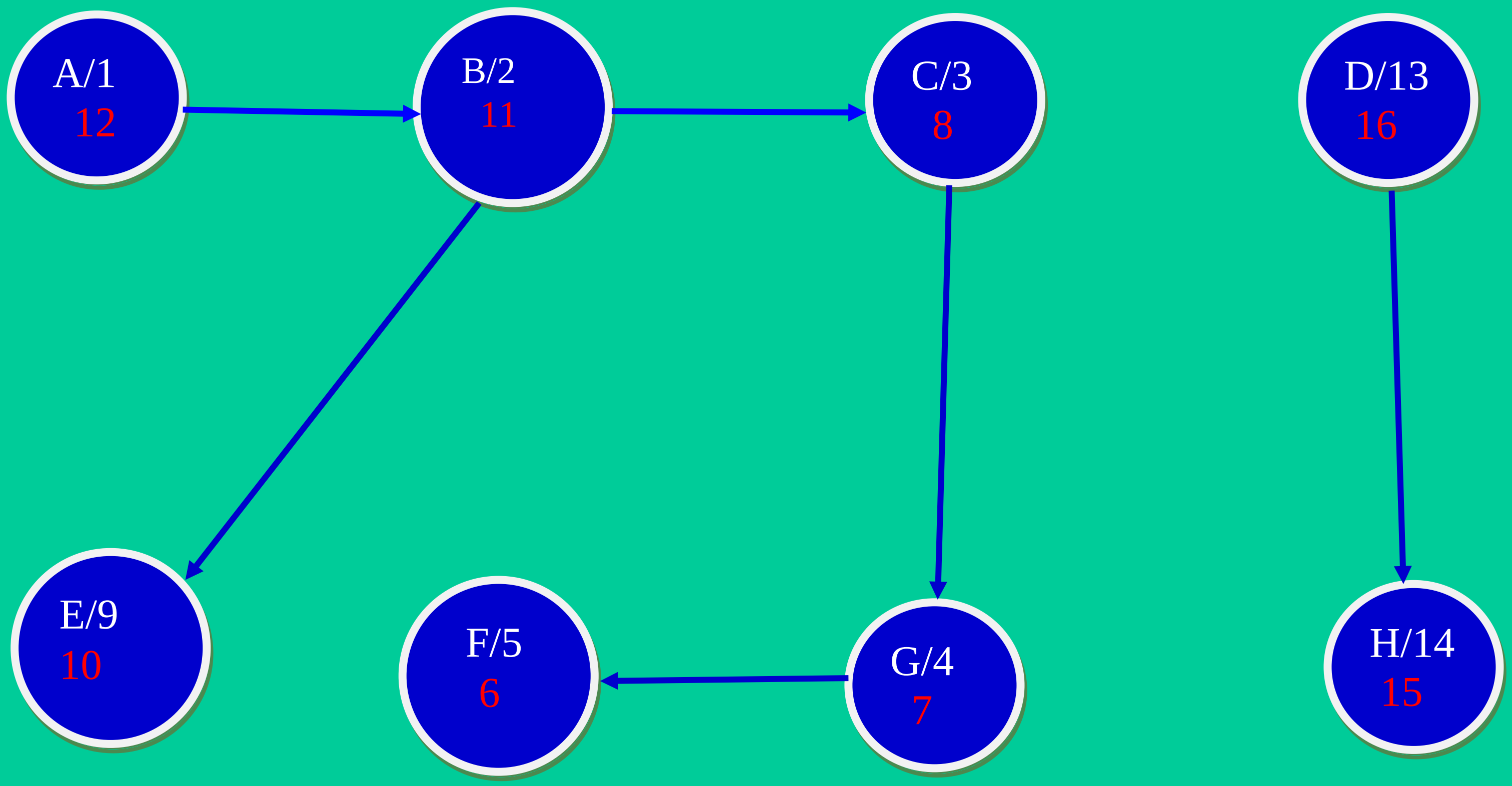

A/1
B/2
11
C/3
8
D
E/9
10
F/5
6
G/4
7
H


A/1
12
B/2
11
C/3
8
D
E/9
10
F/5
6
G/4
7
H


A/1
12
B/2
11
C/3
8
D/13
E/9
10
F/5
6
G/4
7
H

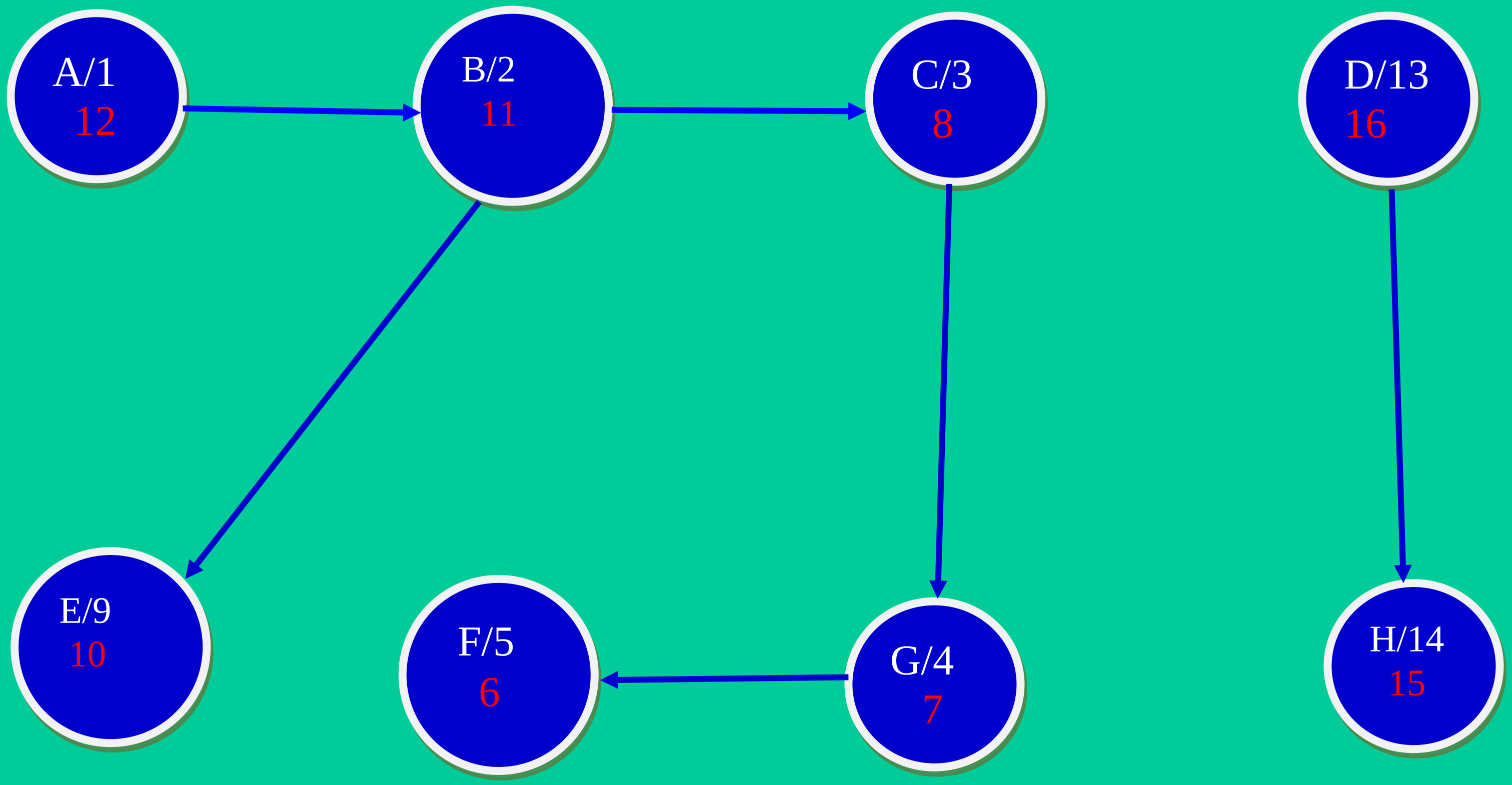

A/1
12
B/2
11
C/3
8
D/13
E/9
10
F/5
6
G/4
7
H/14
15


A/1
12
B/2
11
C/3
8
D/13
16
E/9
10
F/5
6
G/4
7
H/14
15



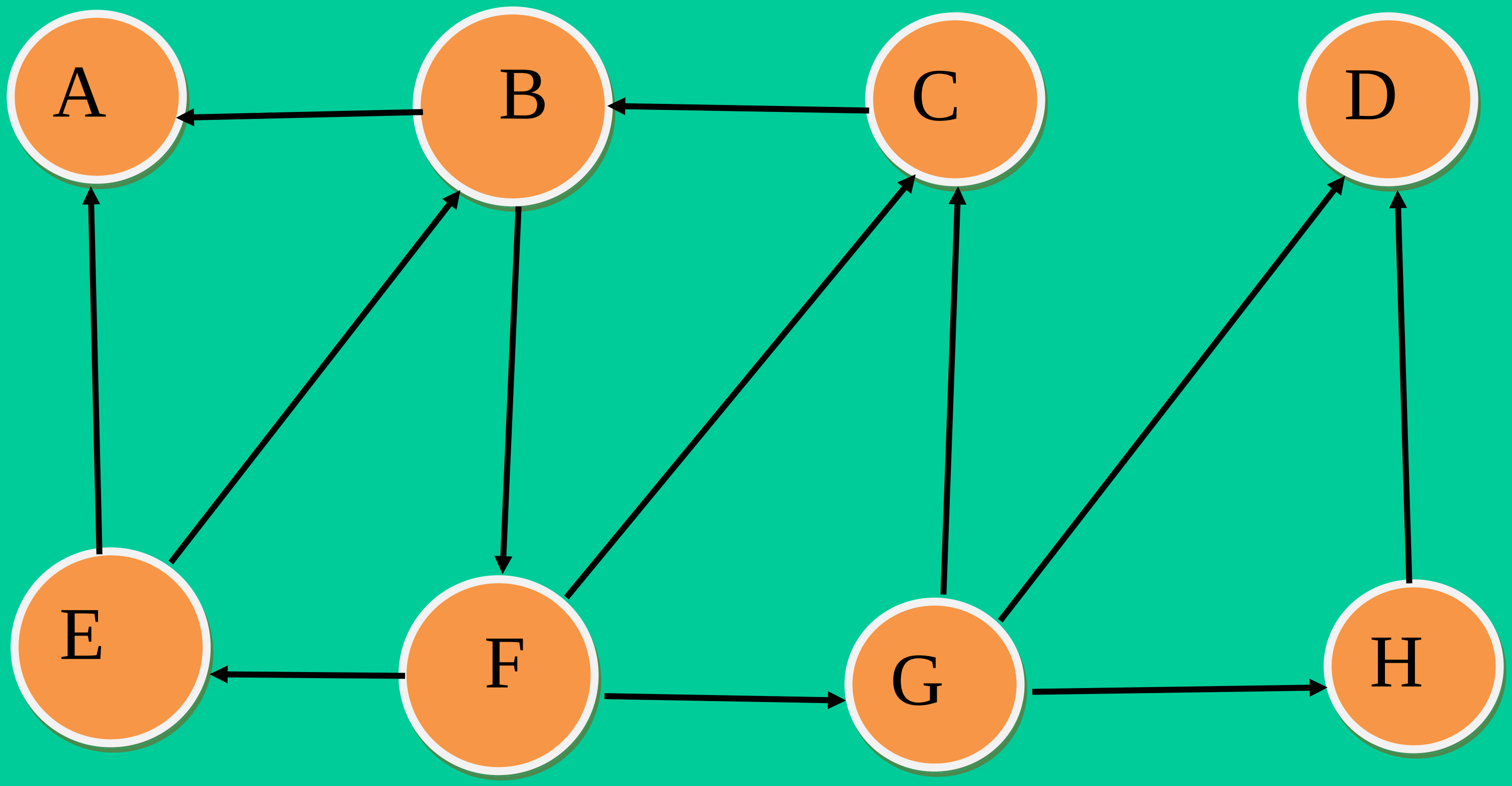
Tri des sommets par ordre **décroissant** des **dates fin**

D-H-A-B-E-C-G-F

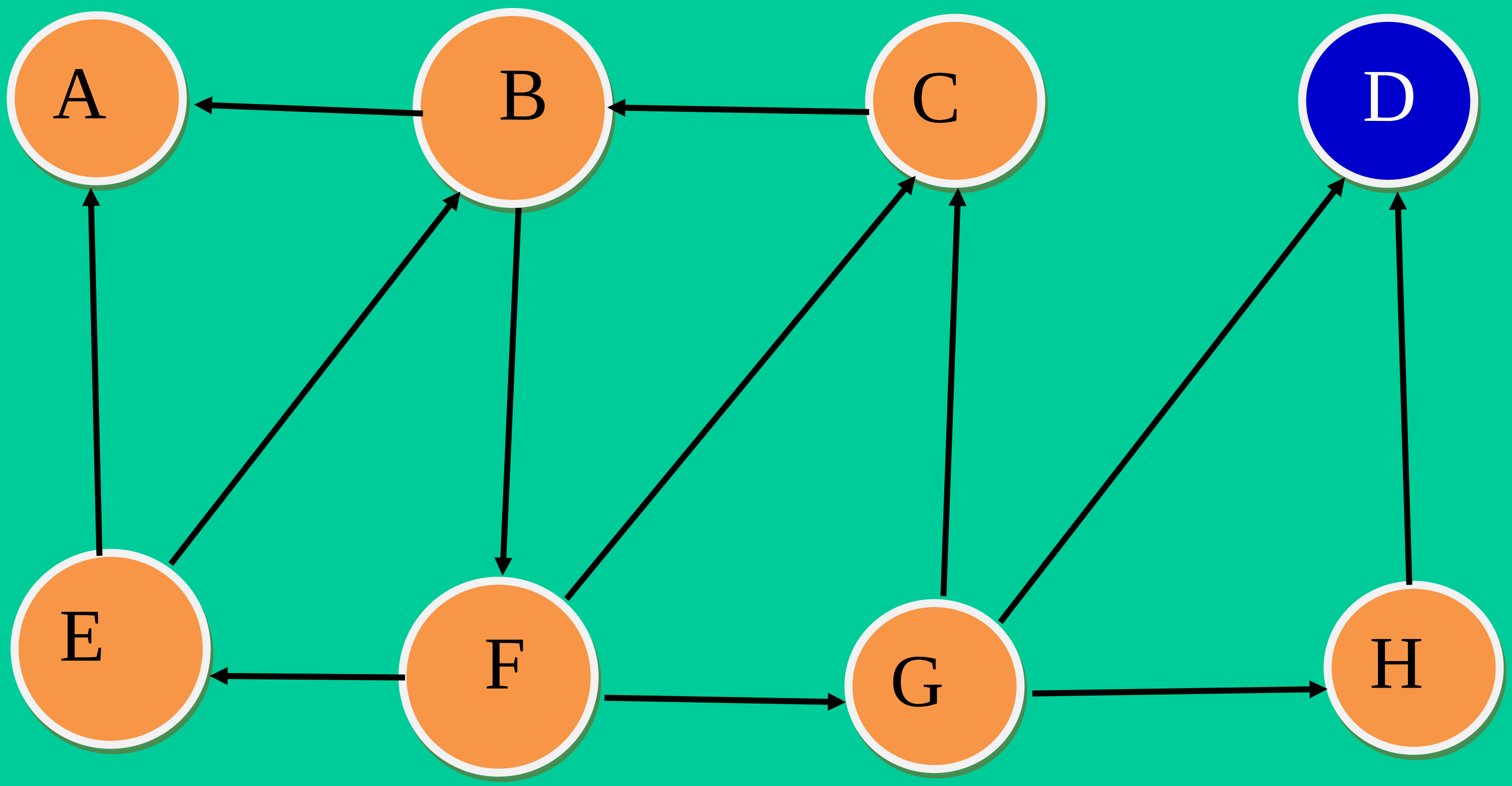


A/1
12
B/2
11
C/3
8
D/13
16
E/9
10
F/5
6
G/4
7
H/14
15

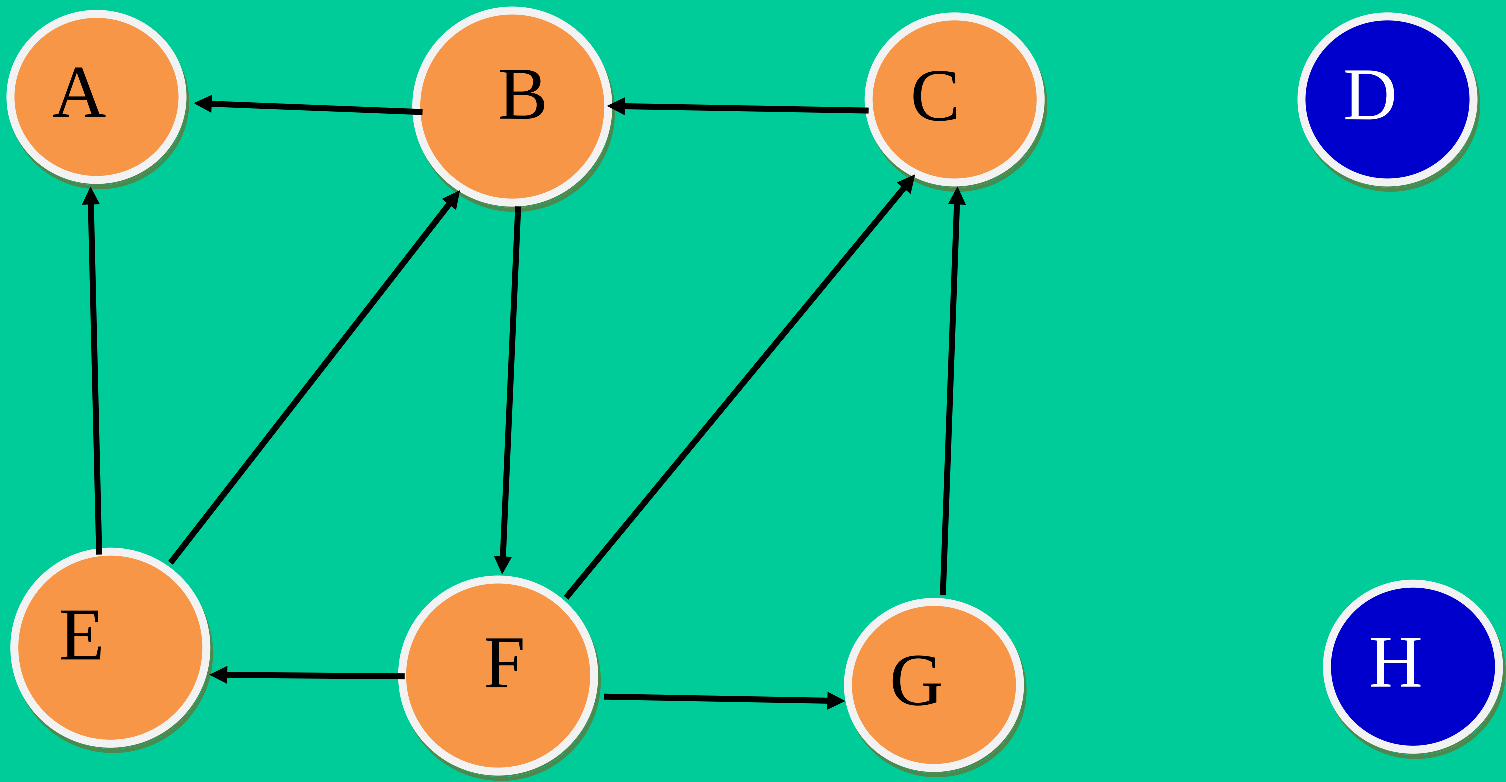
On construit le **graphe dual** G^t



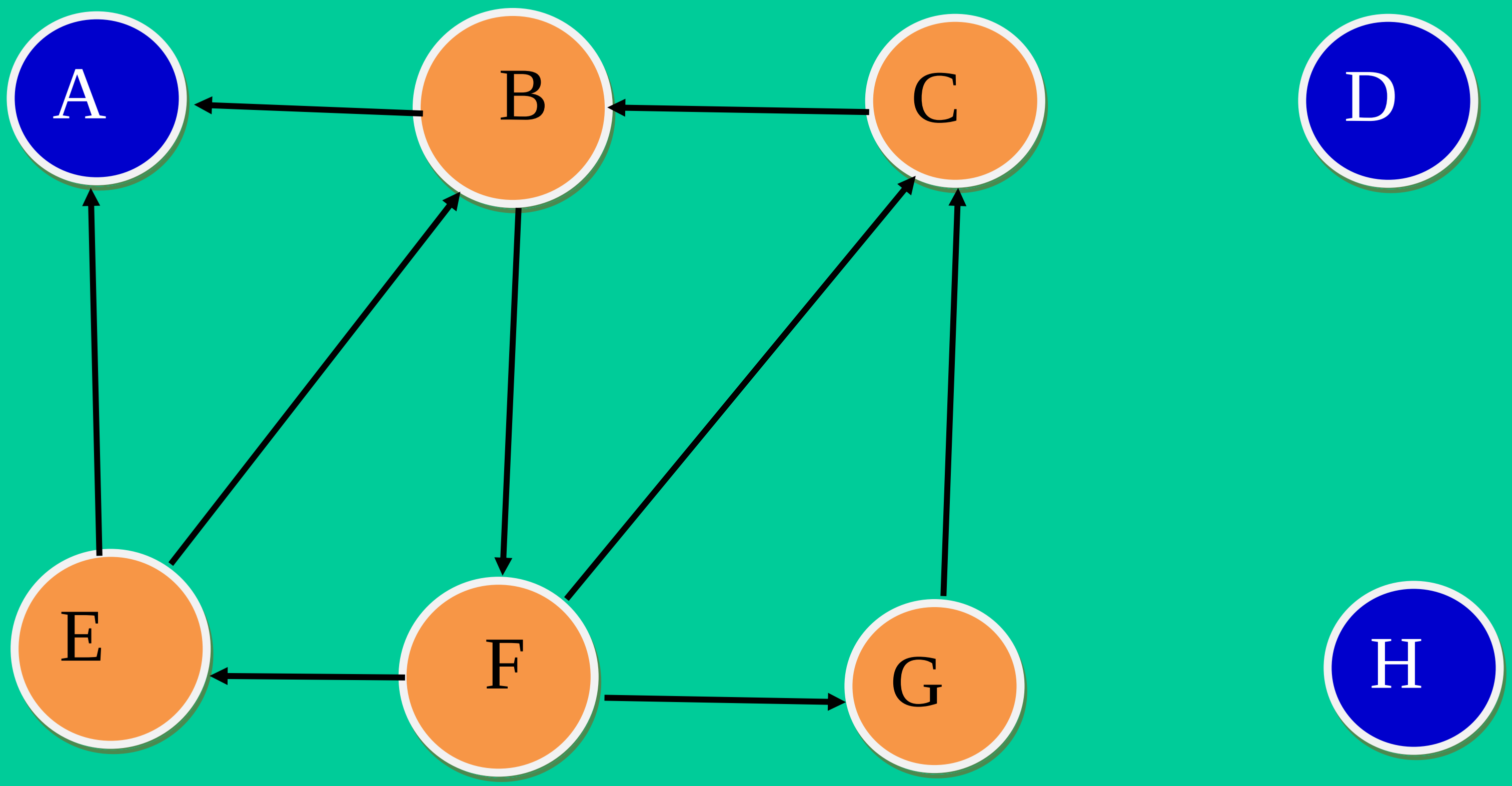
Lancer parcours en profondeur en partant de **D**$_{(16)}$

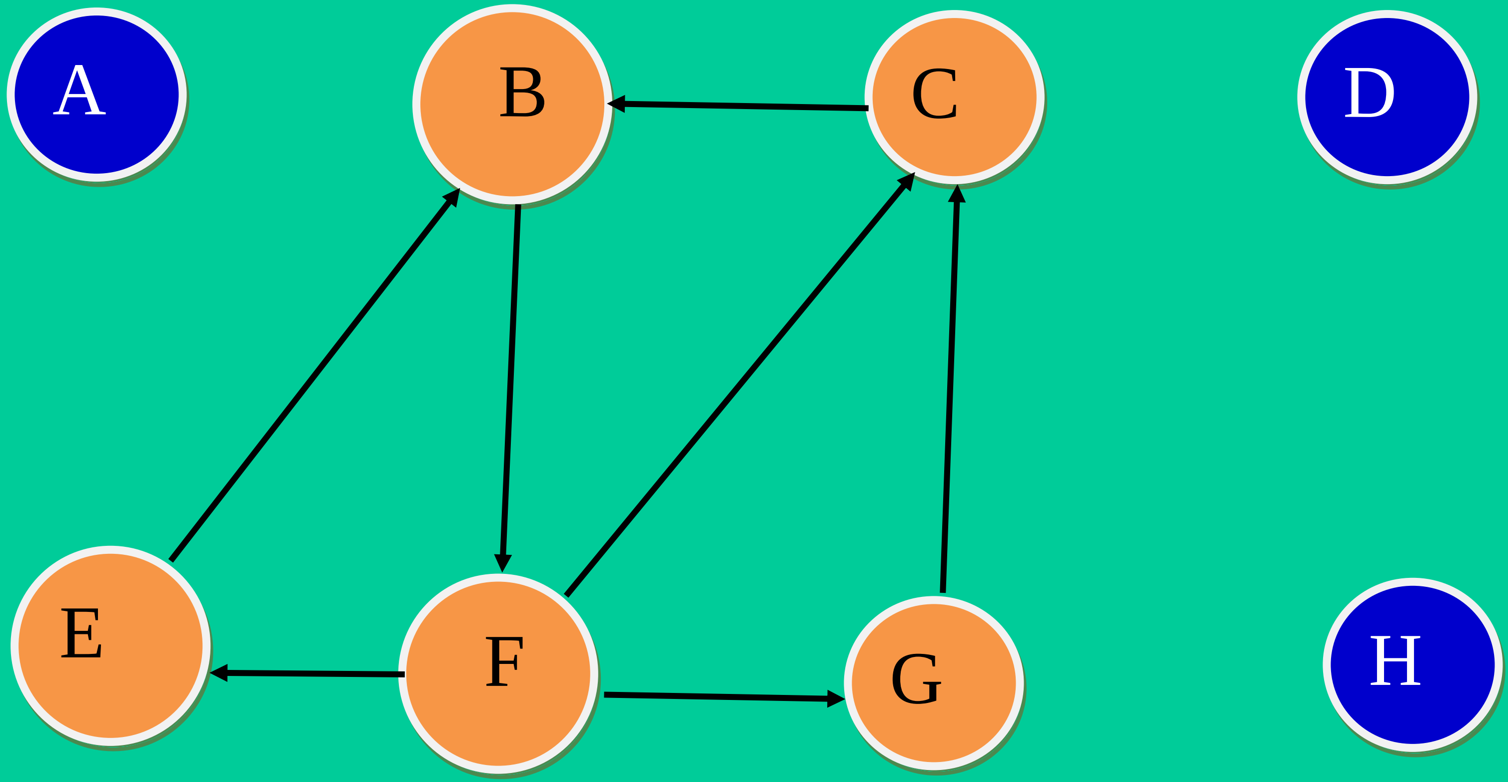


Lancer parcours en profondeur en partant de **H**$_{(15)}$

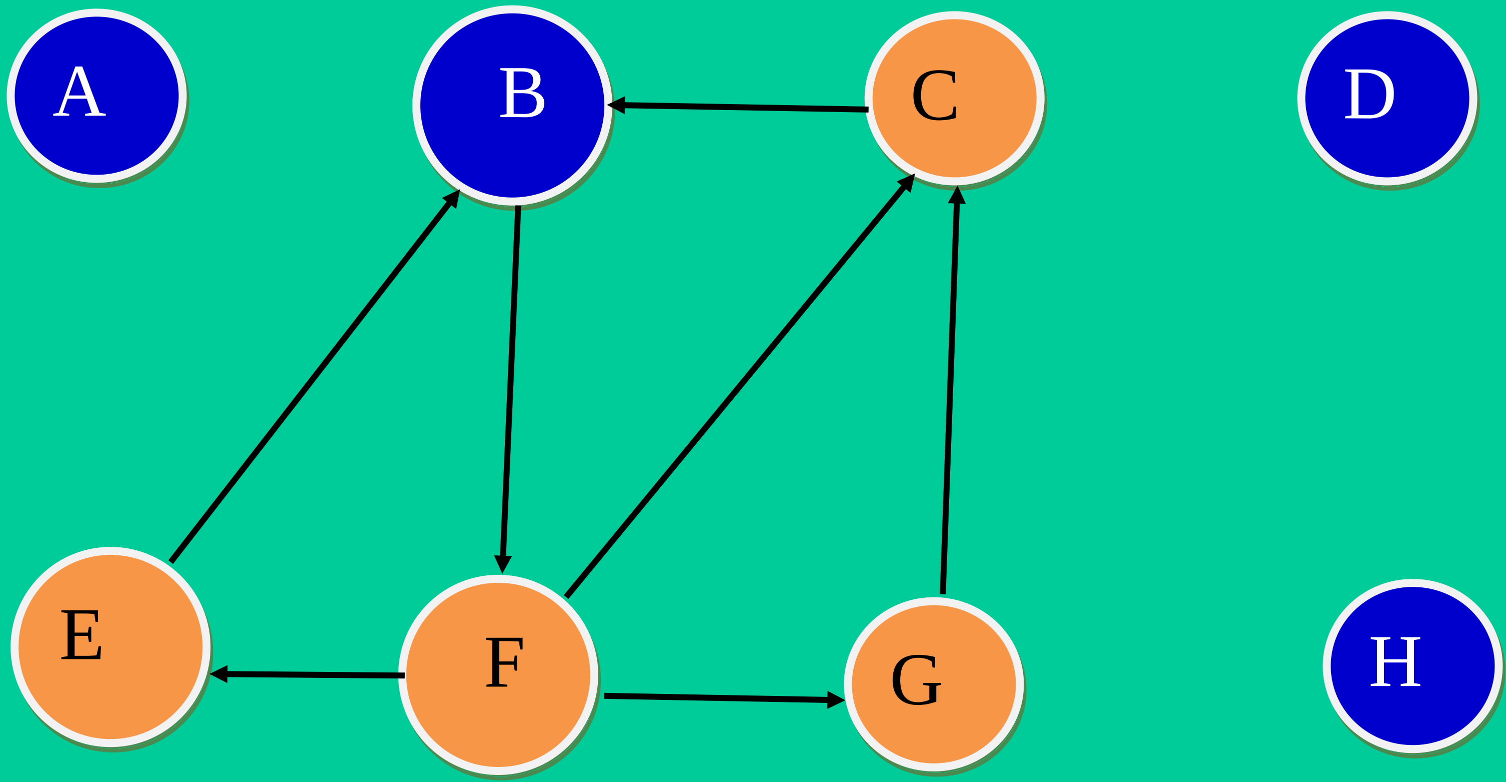


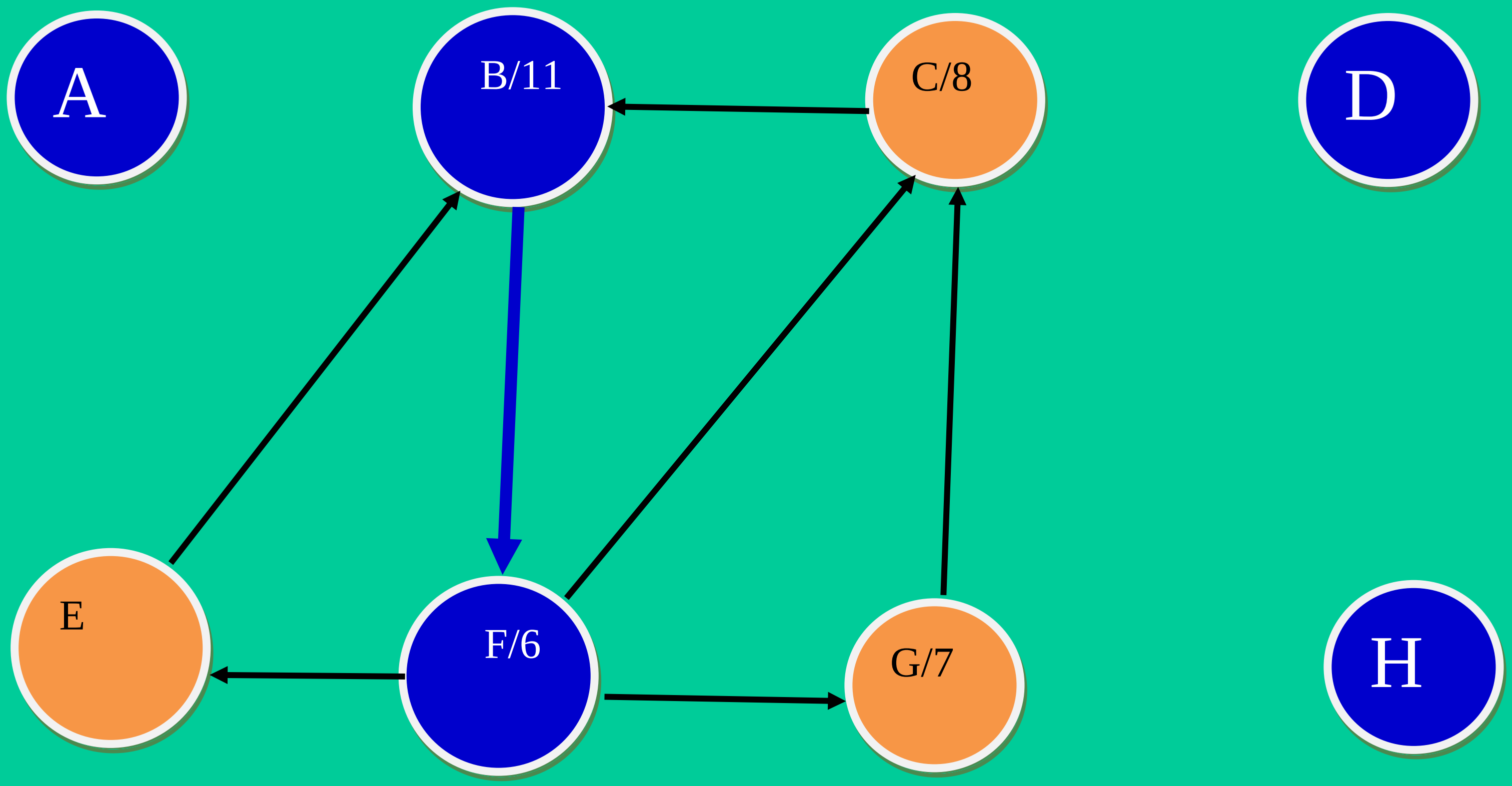
Lancer parcours en profondeur en partant de **A**$_{(12)}$

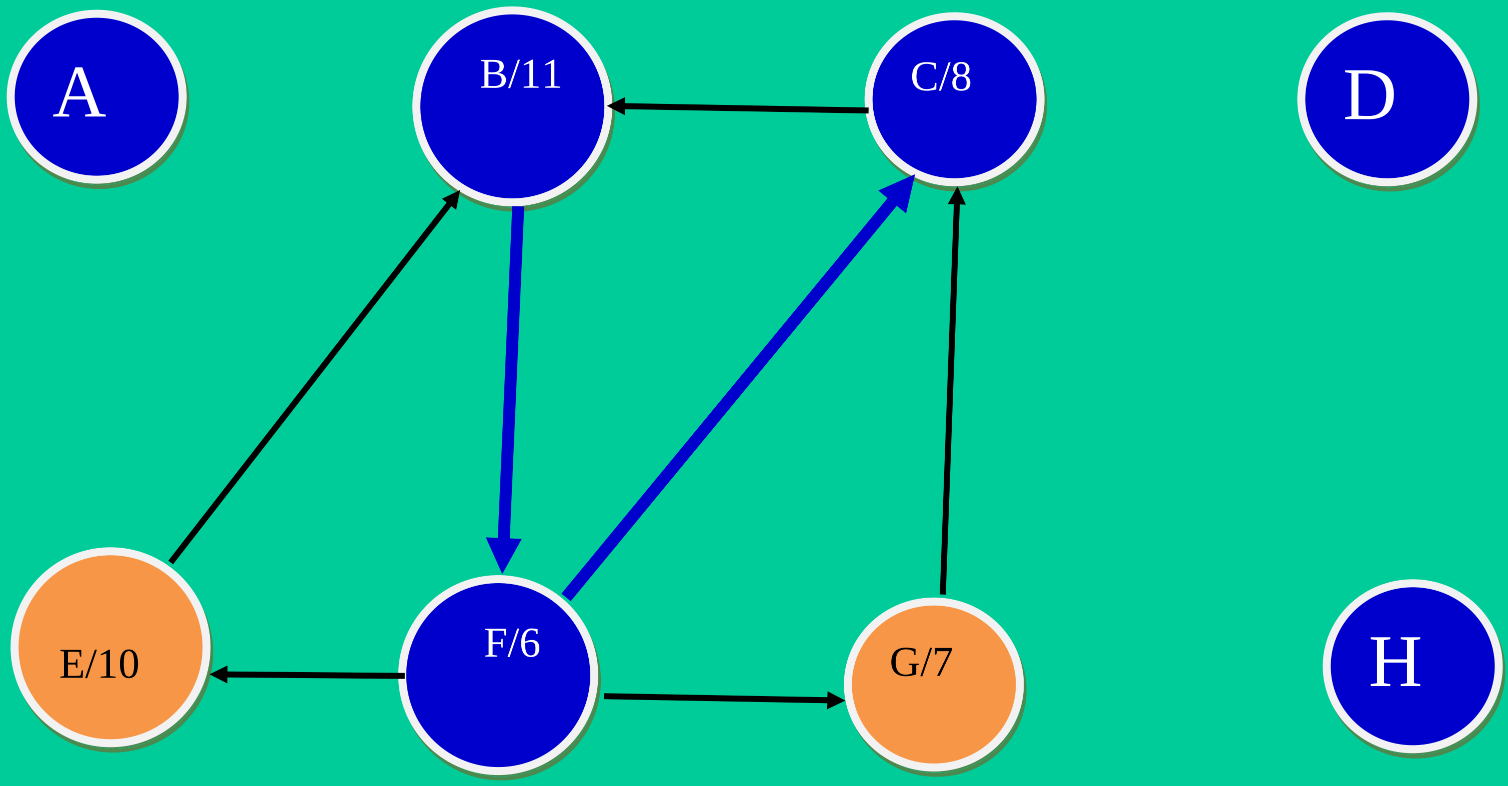


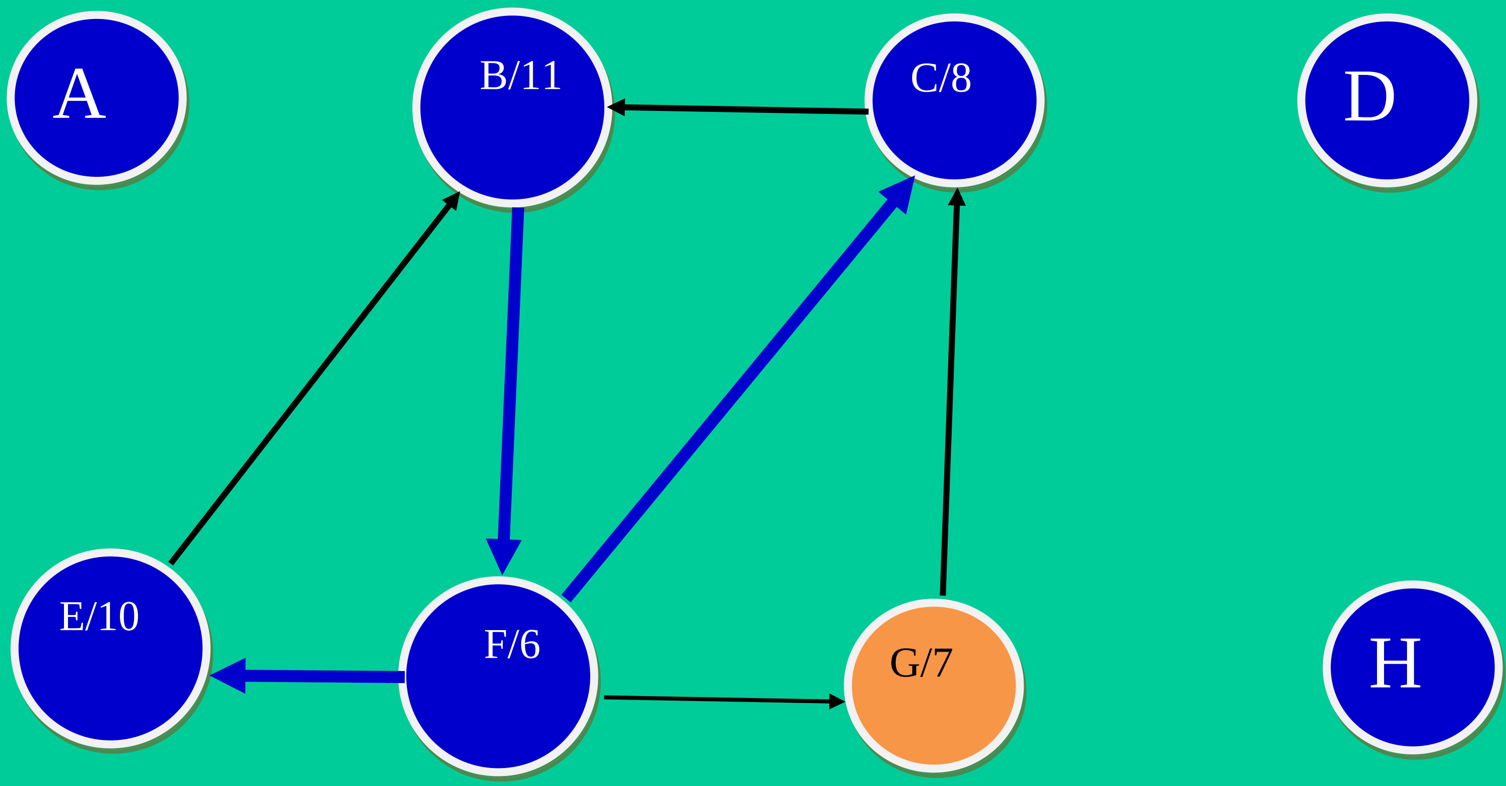

A
B
C
D
E
F
G
H

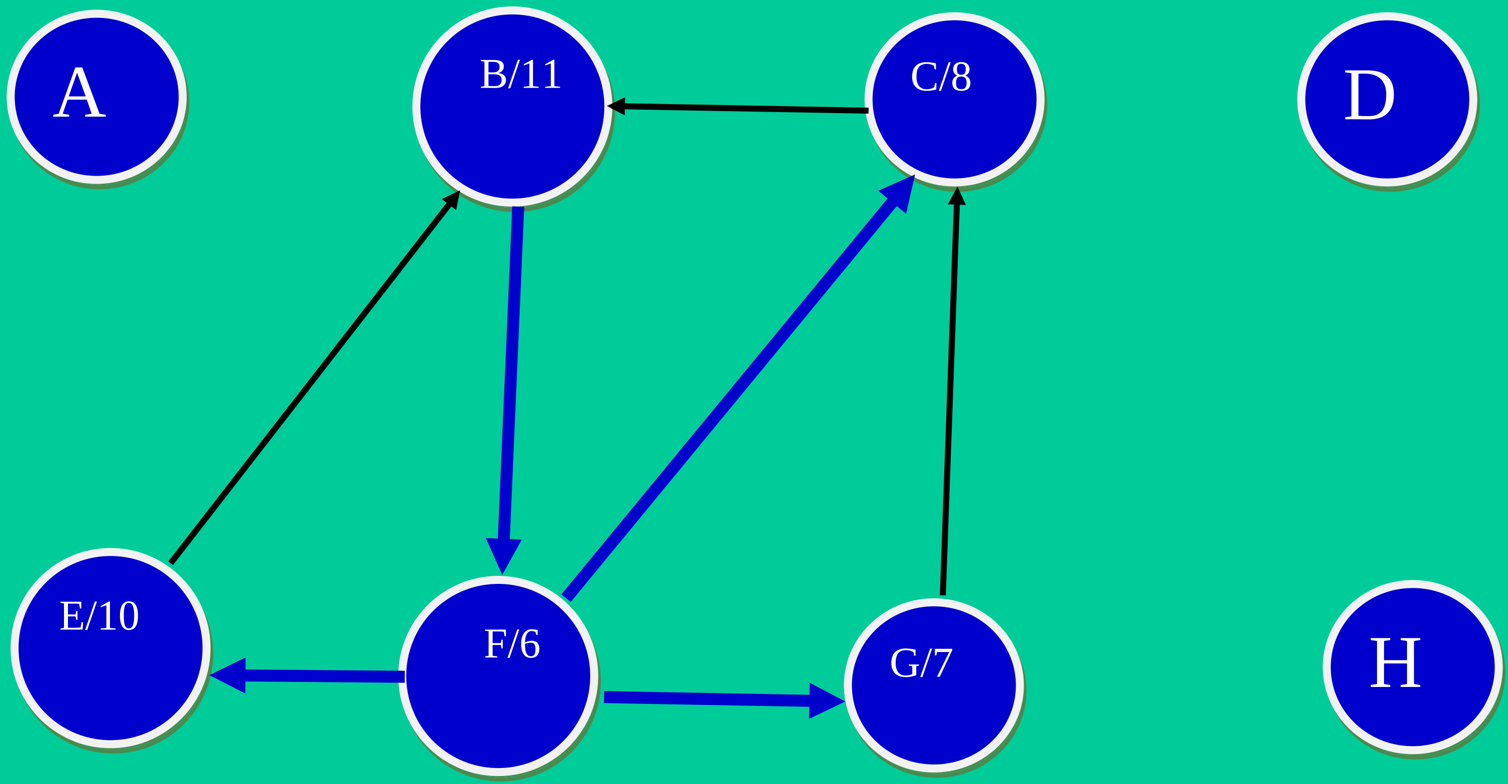
Lancer parcours en profondeur en partant de B(11)

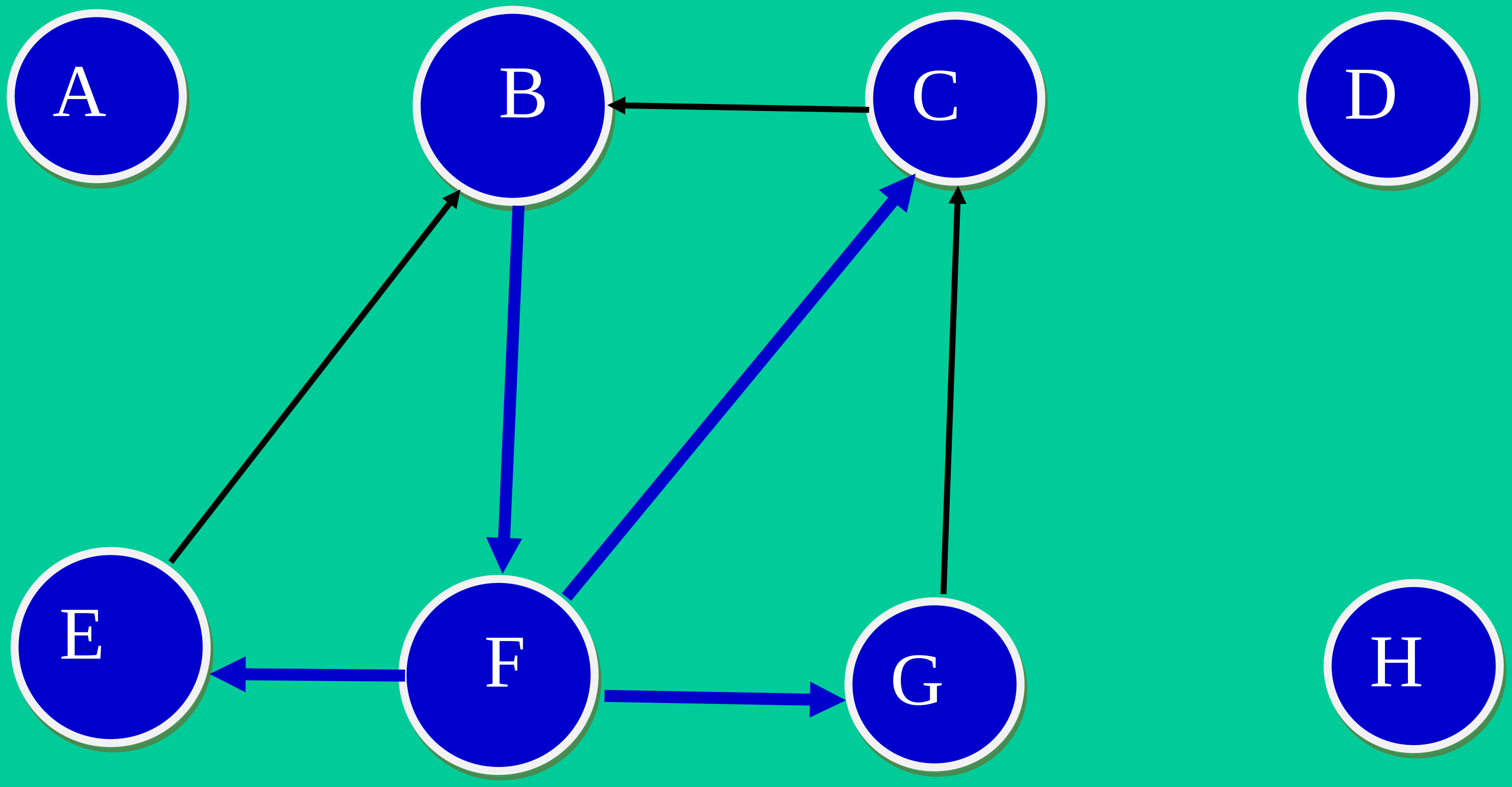



A
B/11
C/8
D
E
F/6
G/7
H

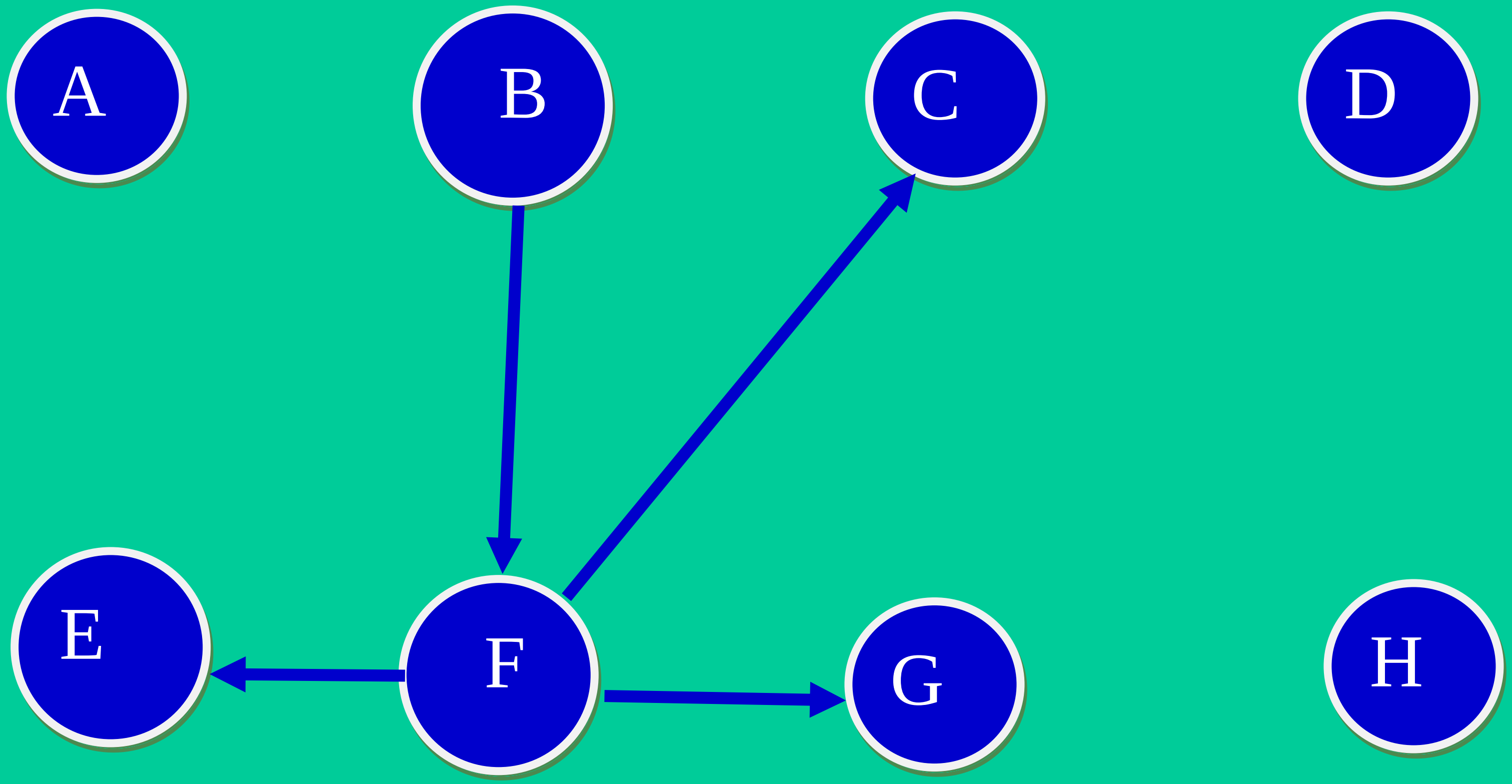

A
B/11
C/8
D
E/10
F/6
G/7
H


A
B/11
C/8
D
E/10
F/6
G/7
H

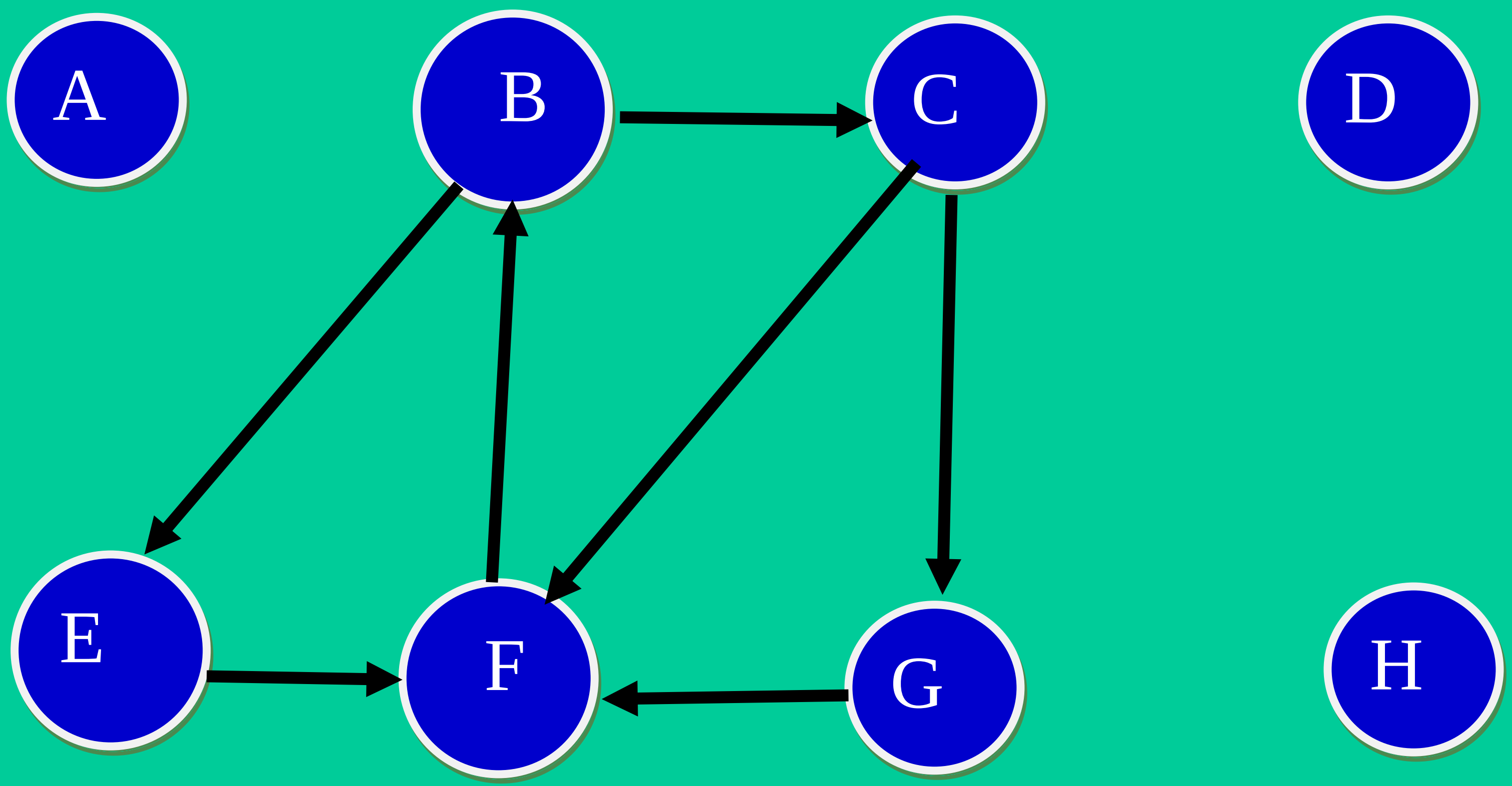

A
B/11
C/8
D
E/10
F/6
G/7
H

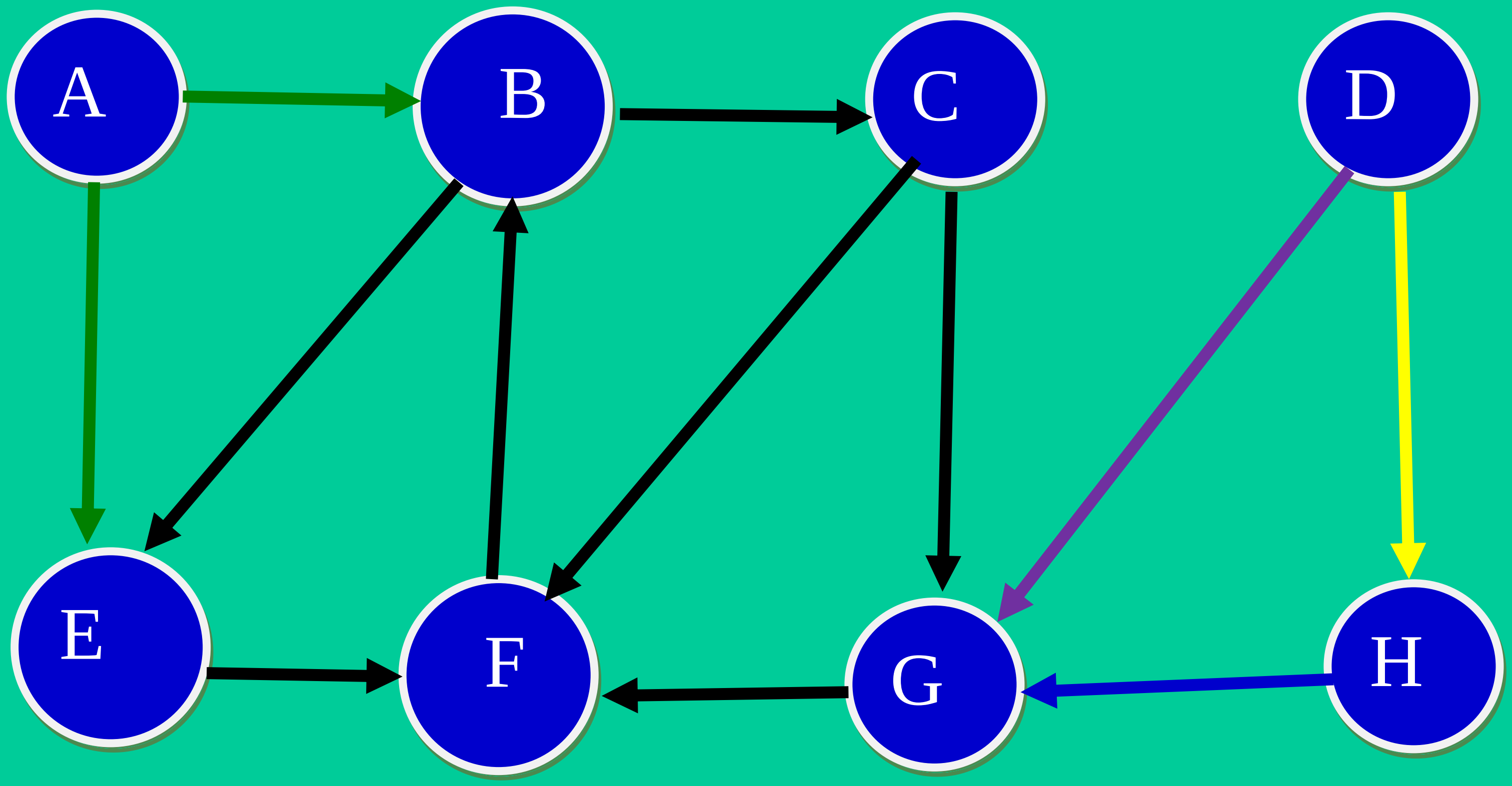

A
B
C
D
E
F
G
H

Le résultat est une forêt de 4 arborescences



Leurs sommets sont ceux de 4 **composantes fortement connexes** dans le graphe G




A
B
C
D
E
F
G
H

Graphe quotient