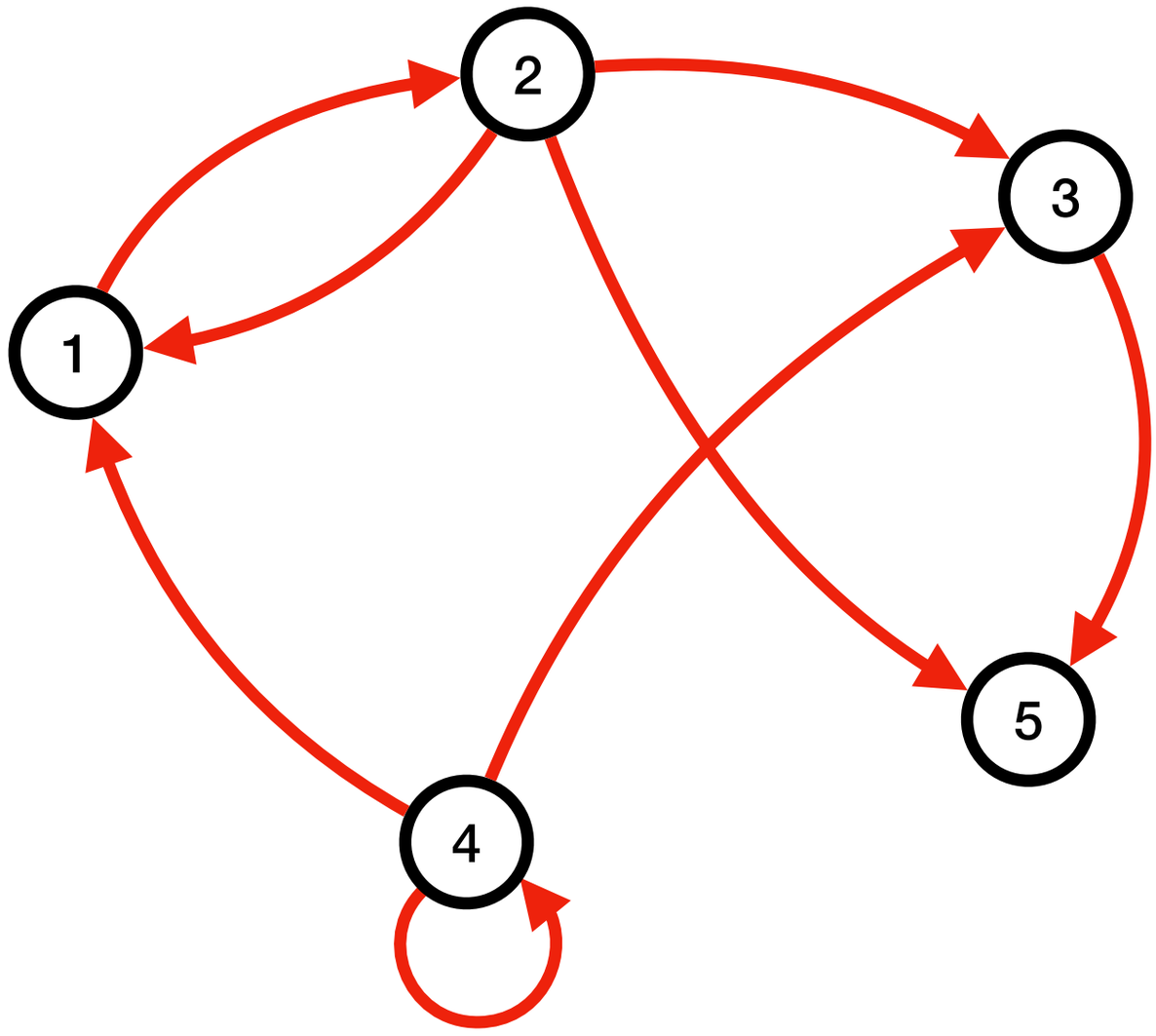
Introduction à l’informatique CM6

Antonio E. Porreca
aeporreca.org/introinfo

Graphes

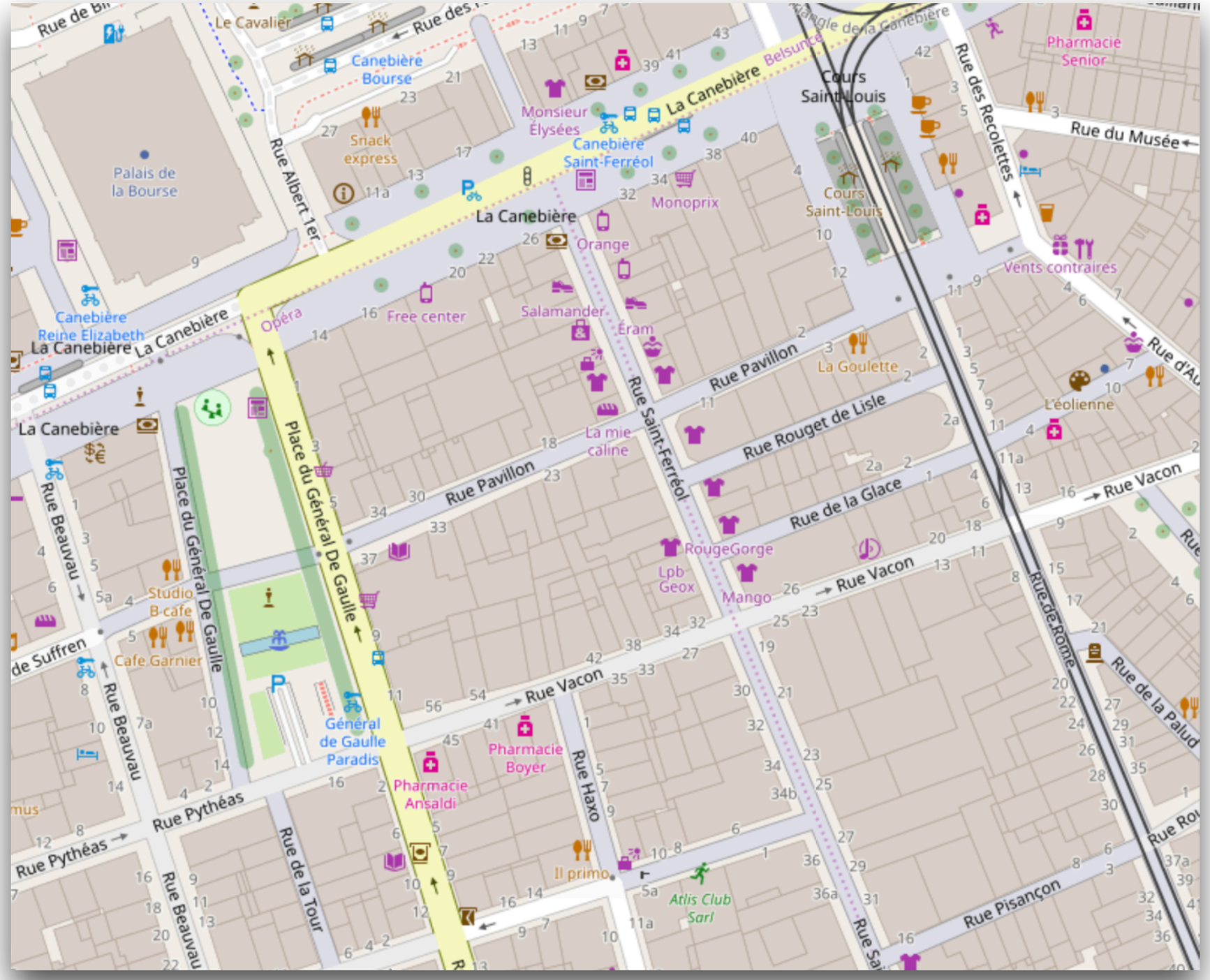
Graphes (orientés)

sommets (nœuds, points)



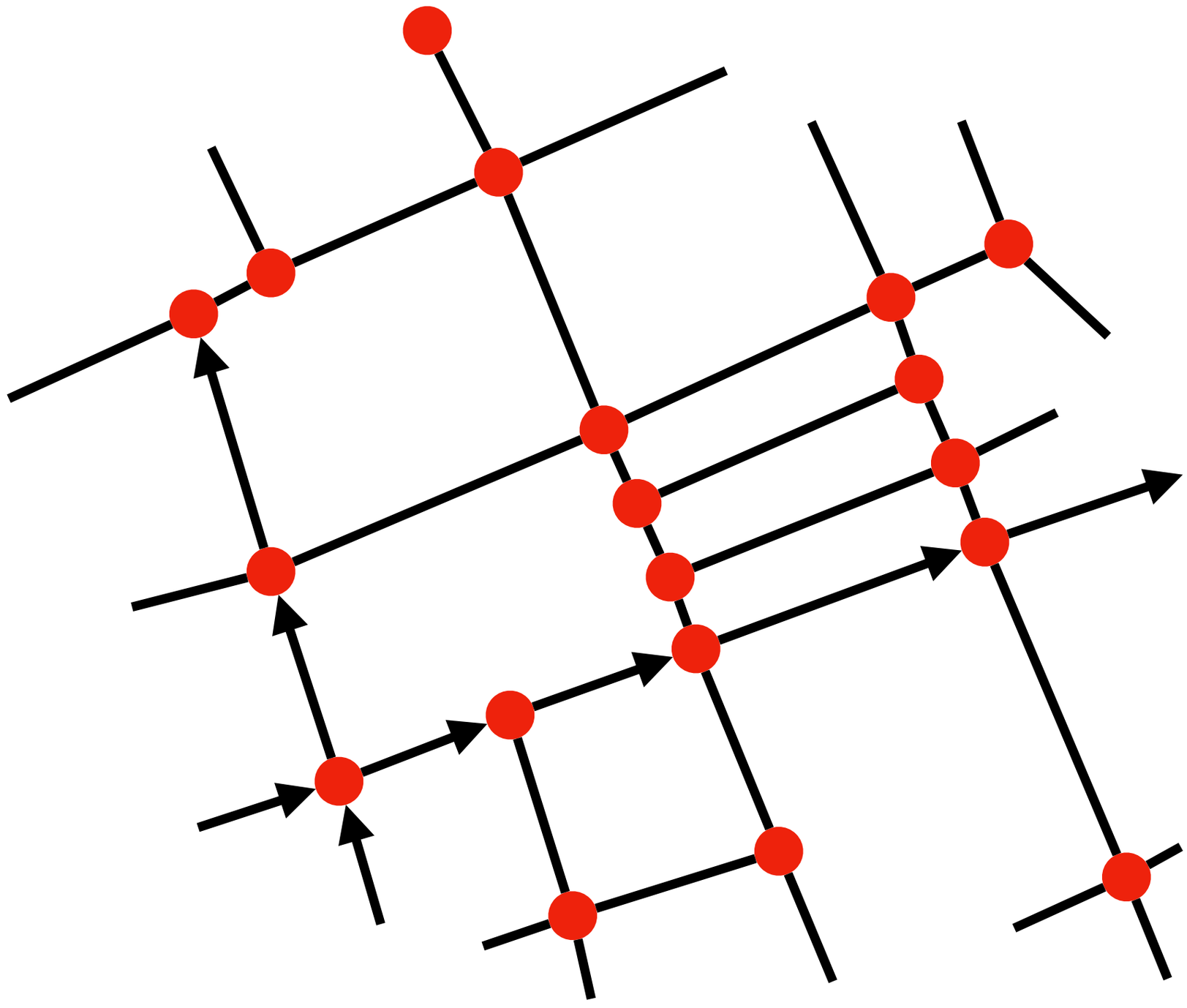
arcs

Plan d’une ville

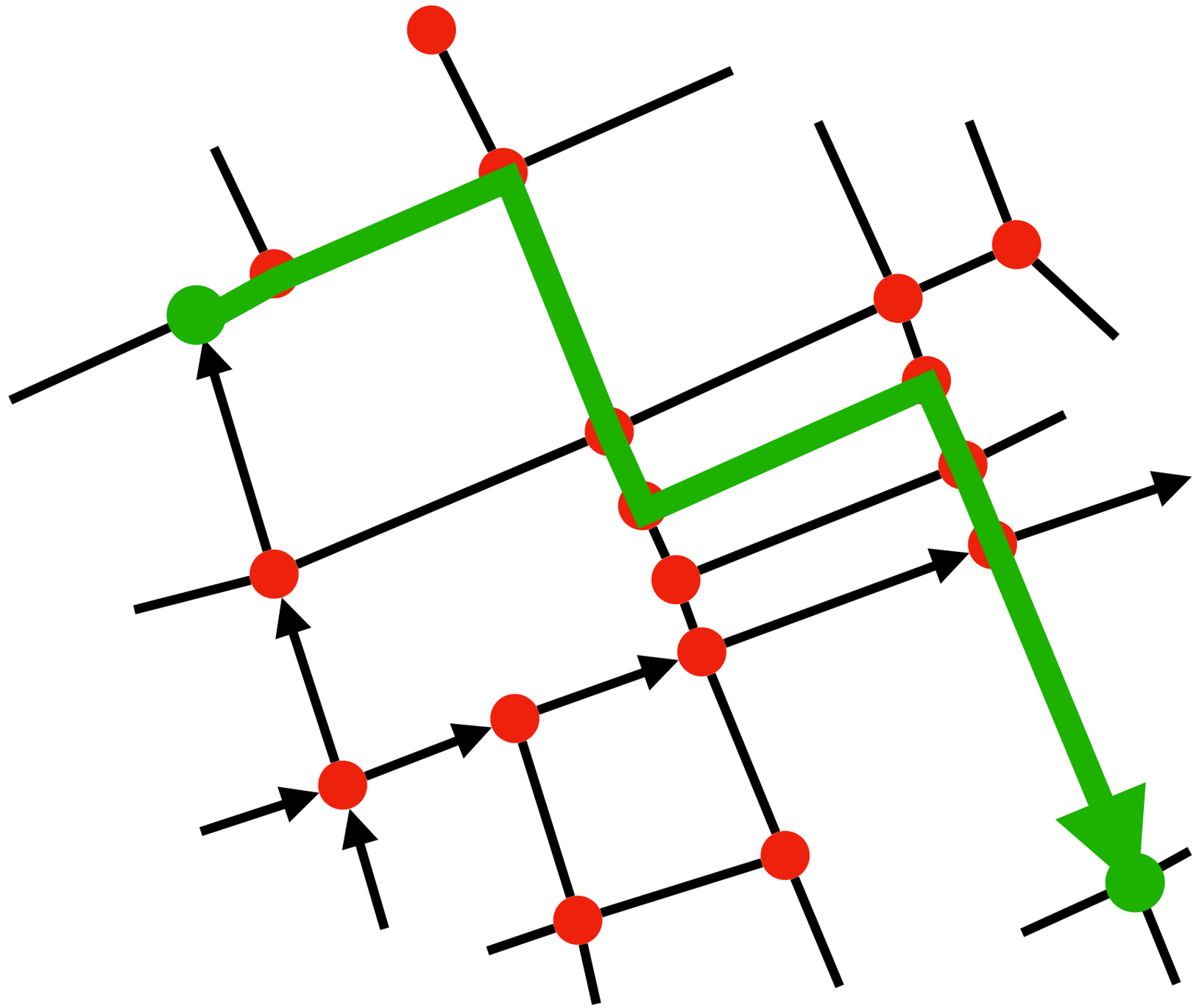


Plan d’une ville

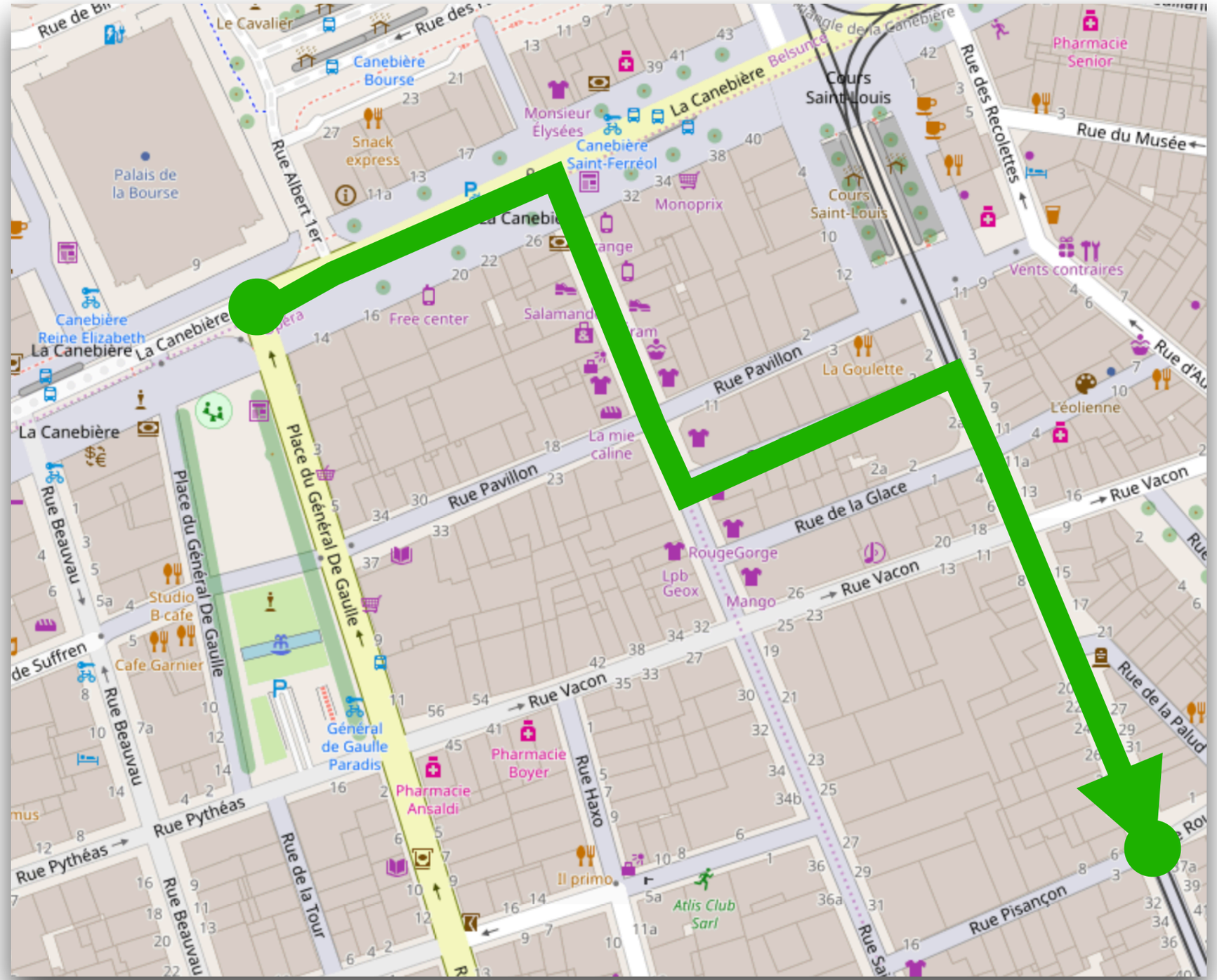
Plan d’une ville



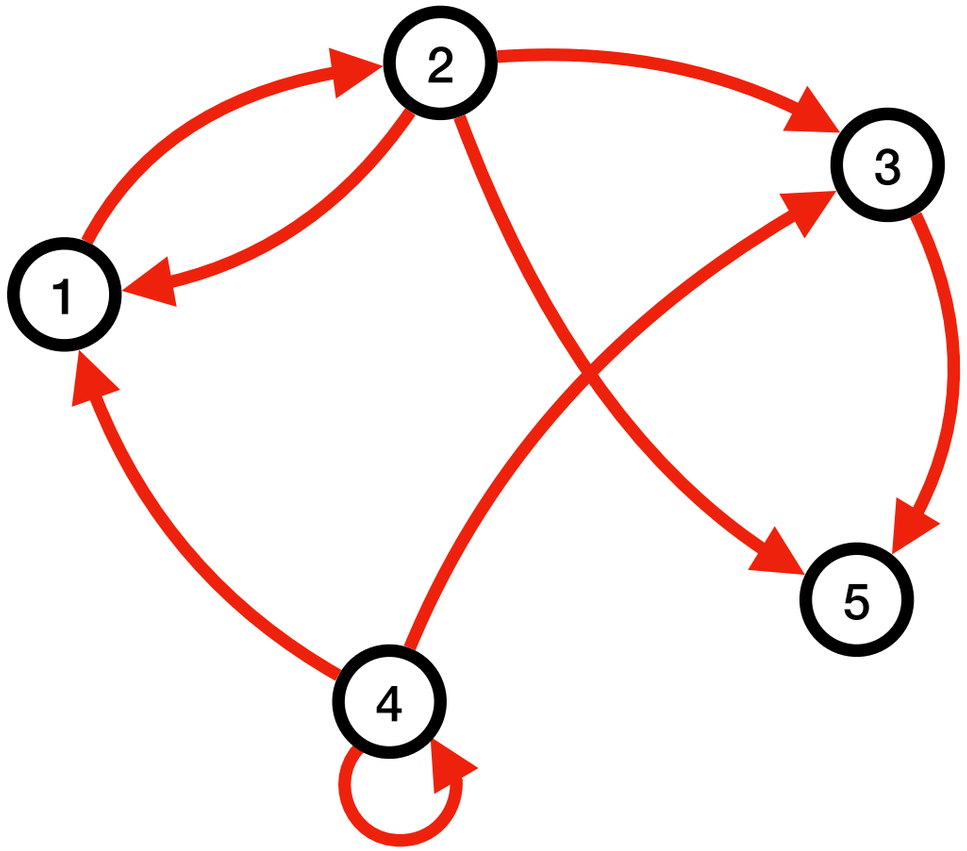
Plan d’une ville



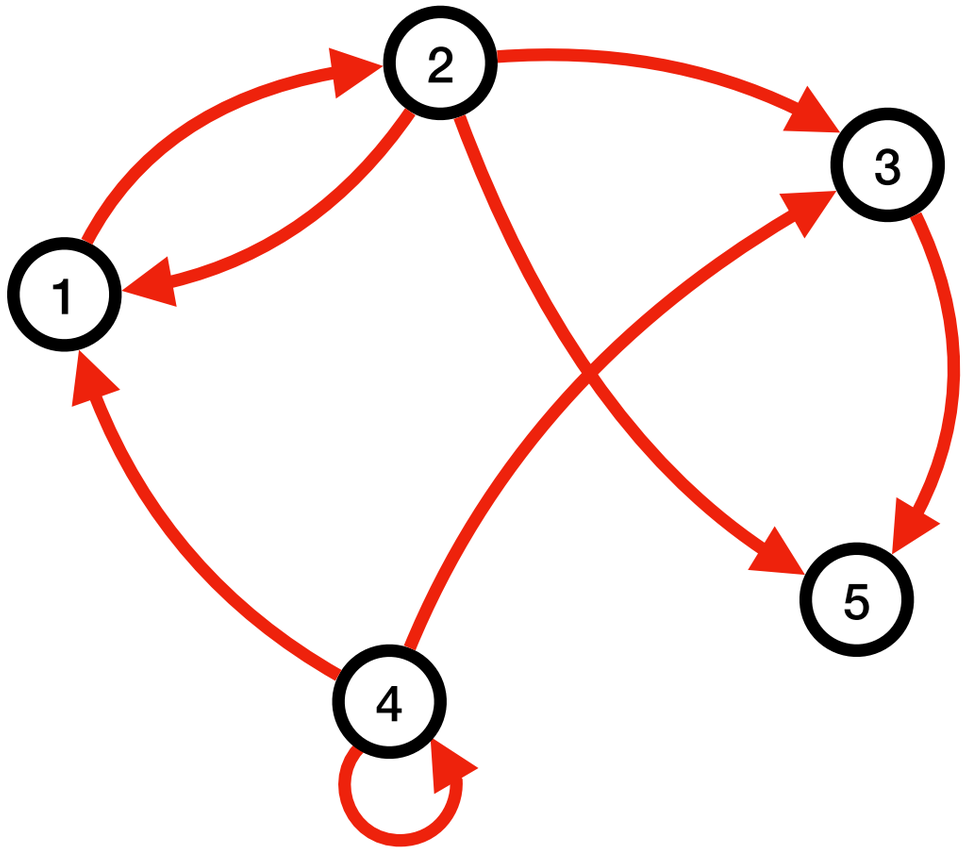
Plan d’une ville



Matrices d’adjacence

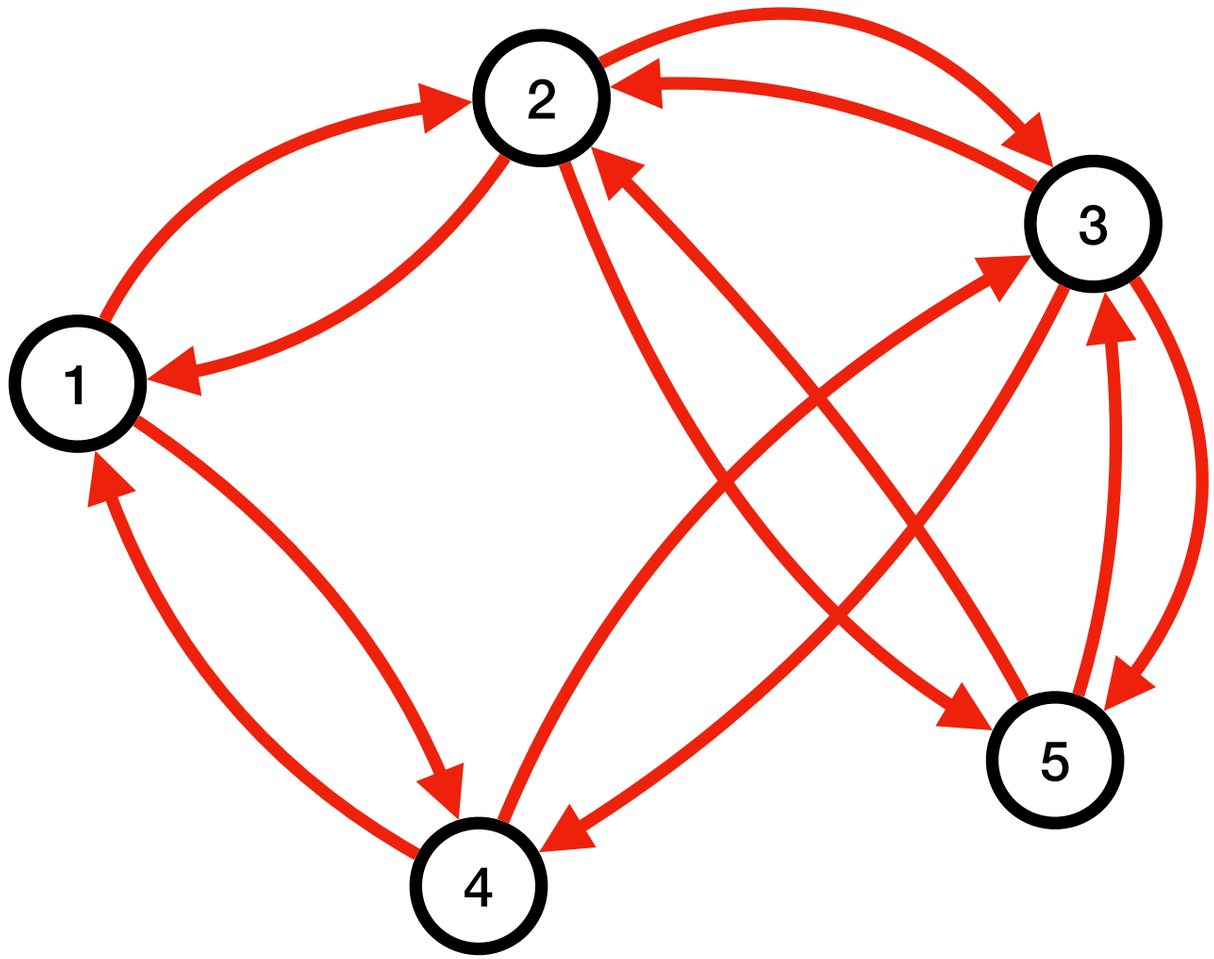


Matrices d’adjacence

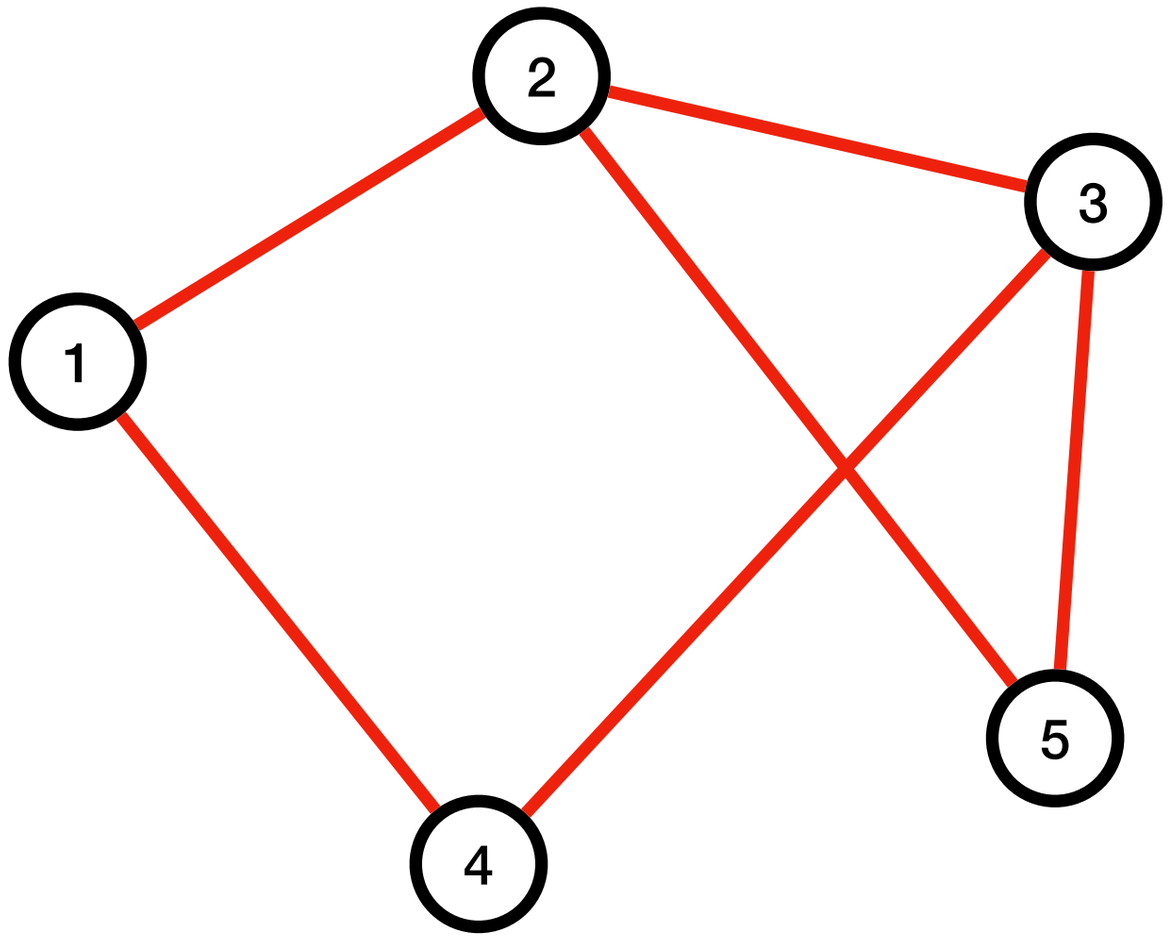


	1	2	3	4	5
1	0	**1**	0	0	0
2	**1**	0	**1**	0	**1**
3	0	0	0	0	**1**
4	**1**	0	**1**	**1**	0
5	0	0	0	0	0

Graphes non orientés

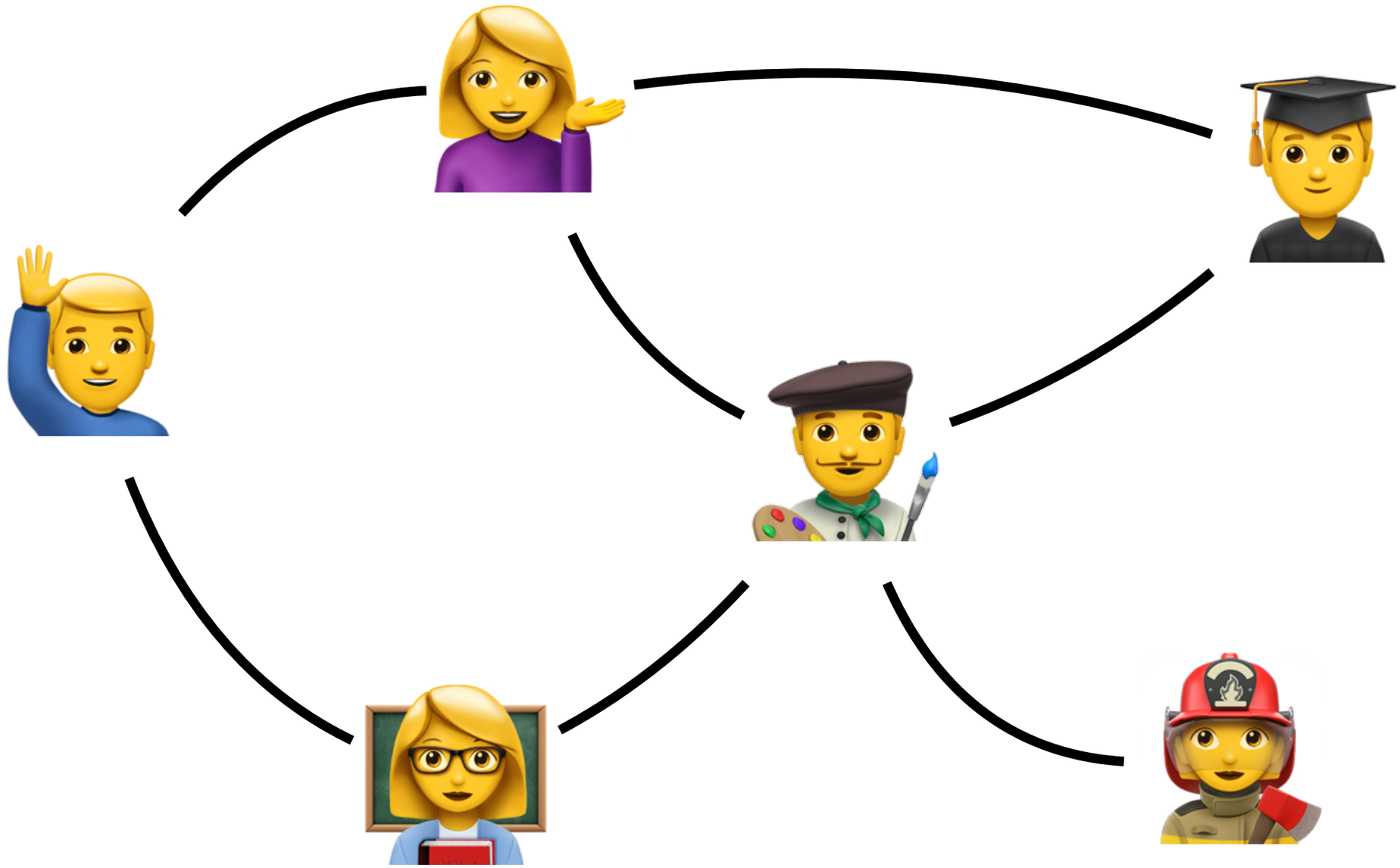


Graphes non orientés



arêtes

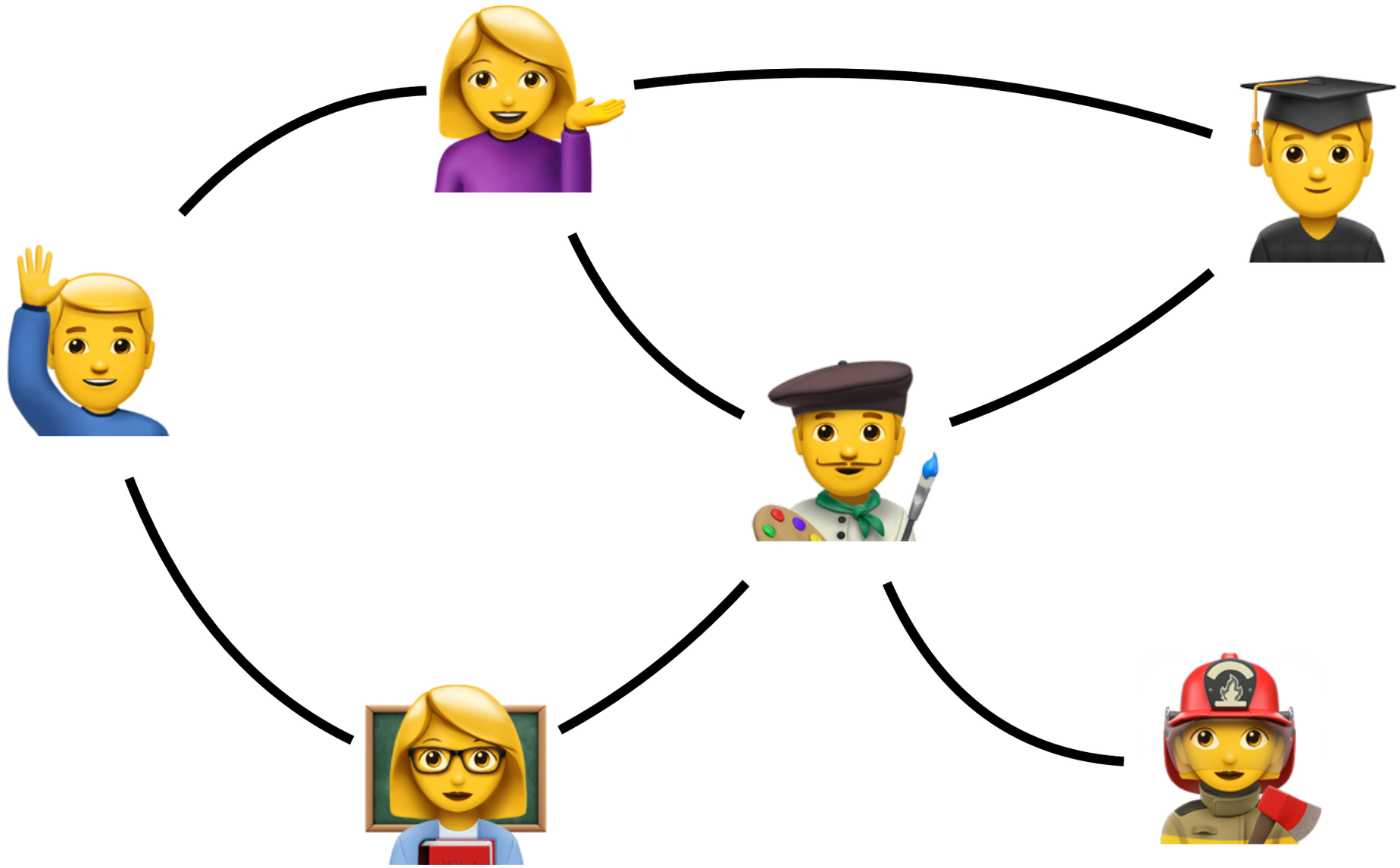
Réseau social (symétrique)



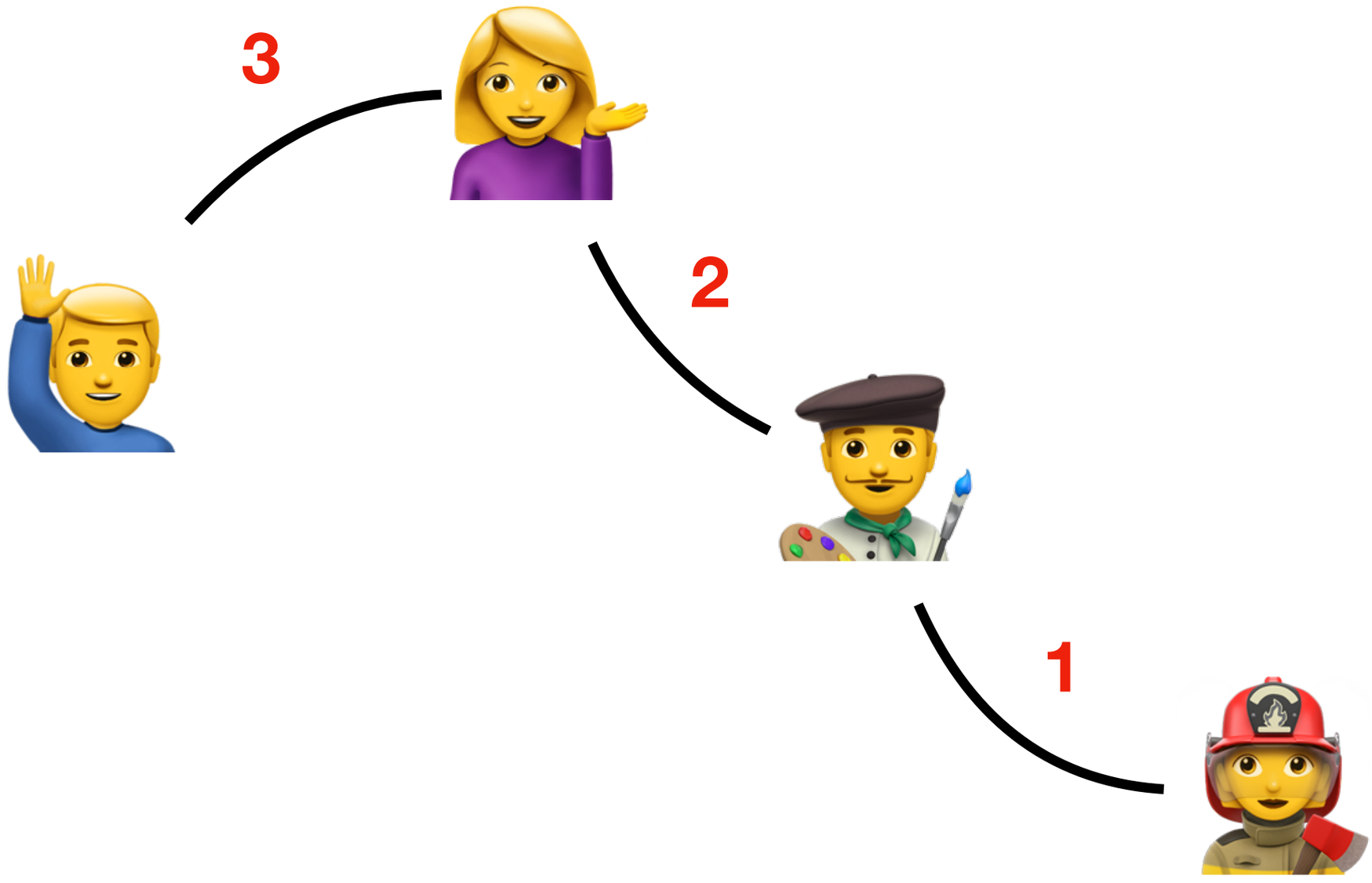
Each person in the world (at least among the 1.59 billion people active on Facebook) is connected to every other person by an average of three and a half other people.

https://research.fb.com/three-and-a-half-degrees-of-separation/

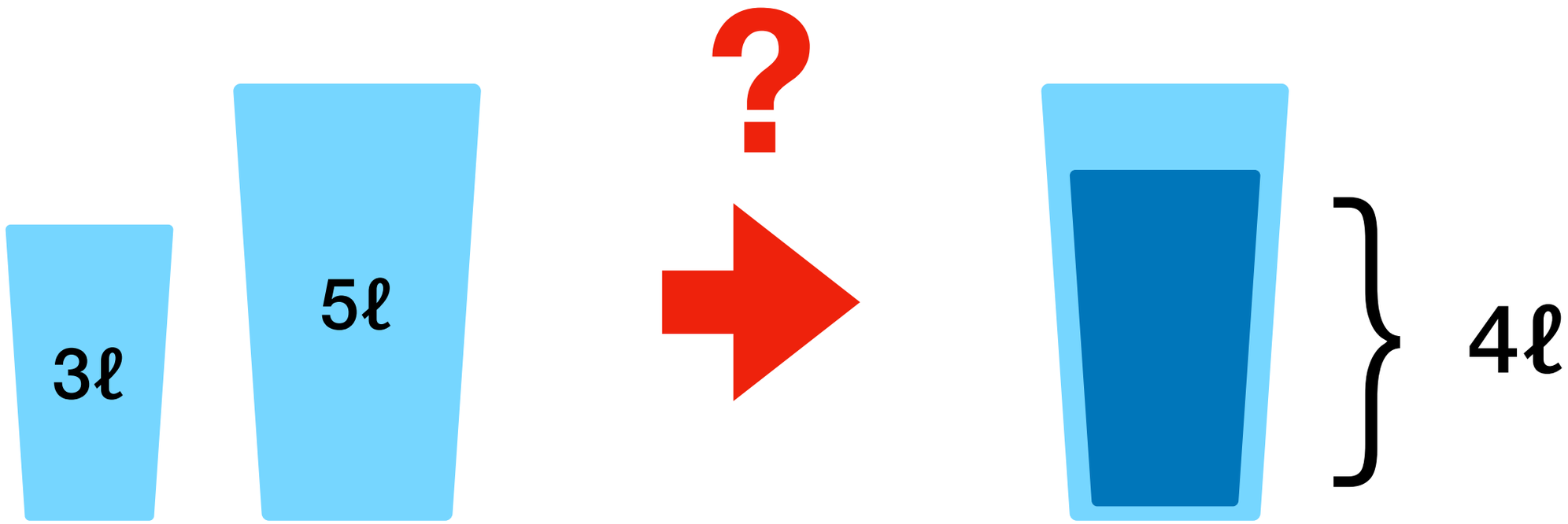
Diamètre d’un graphe



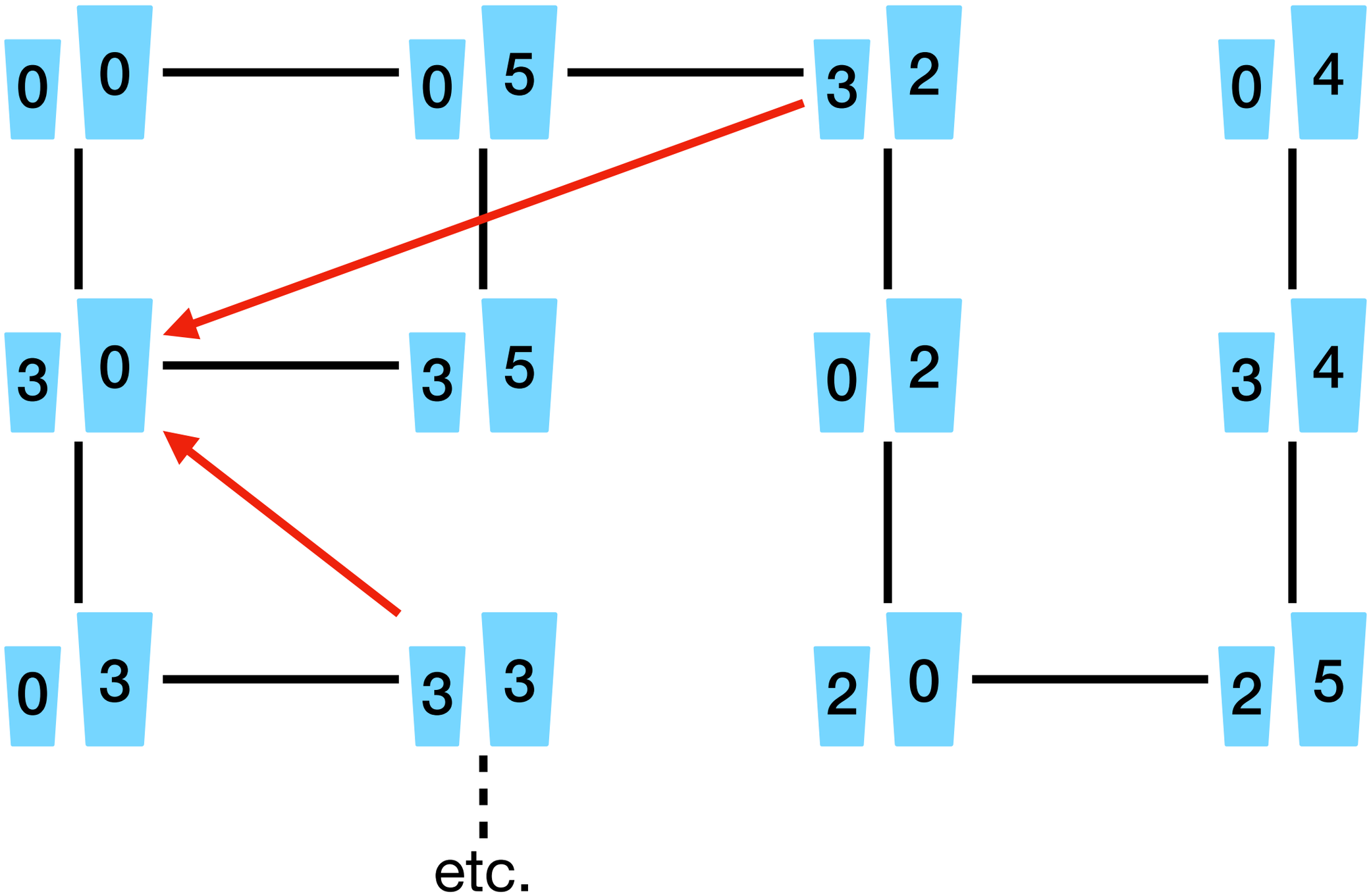
Diamètre d’un graphe



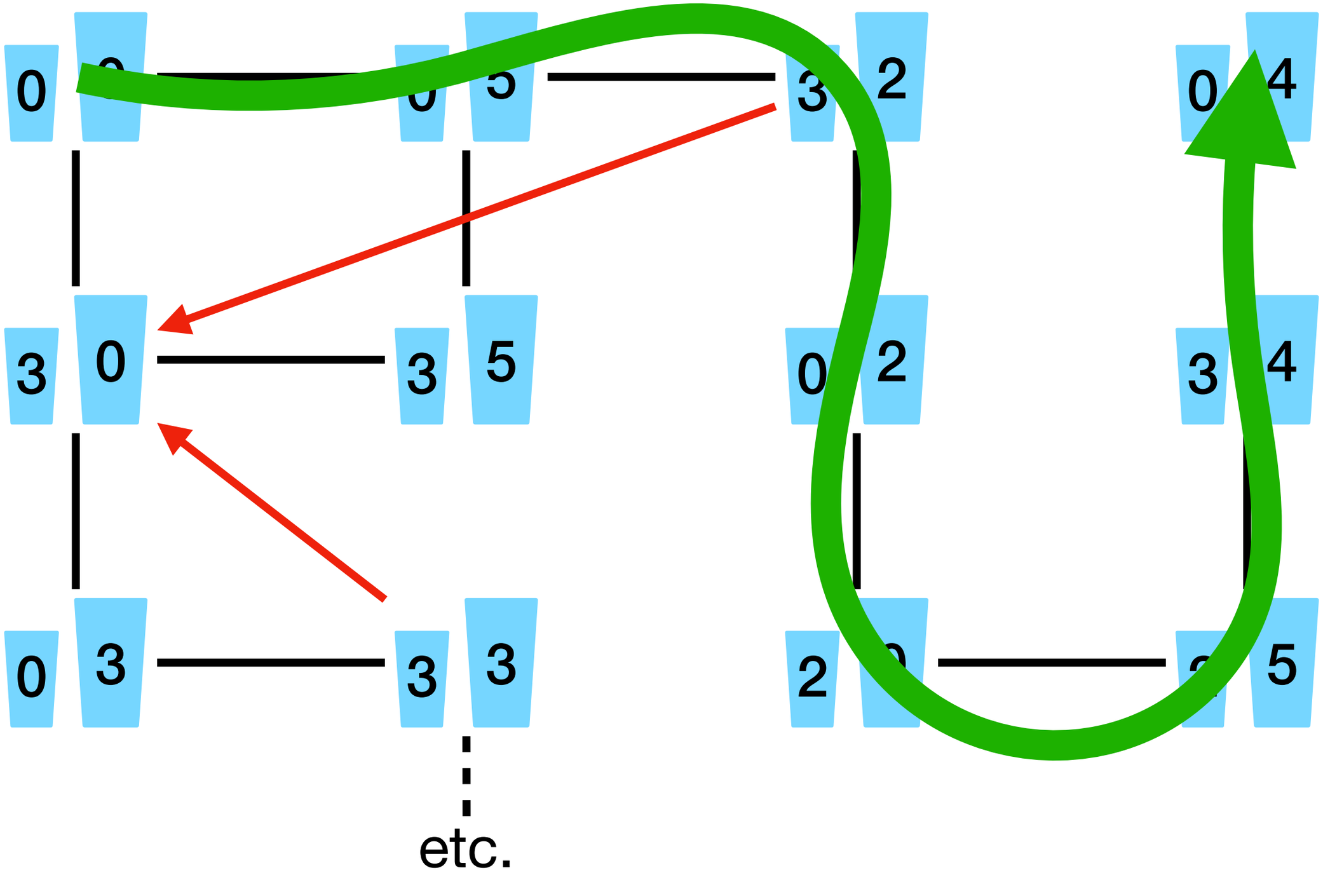
Enigme des récipients



Graphe des configurations

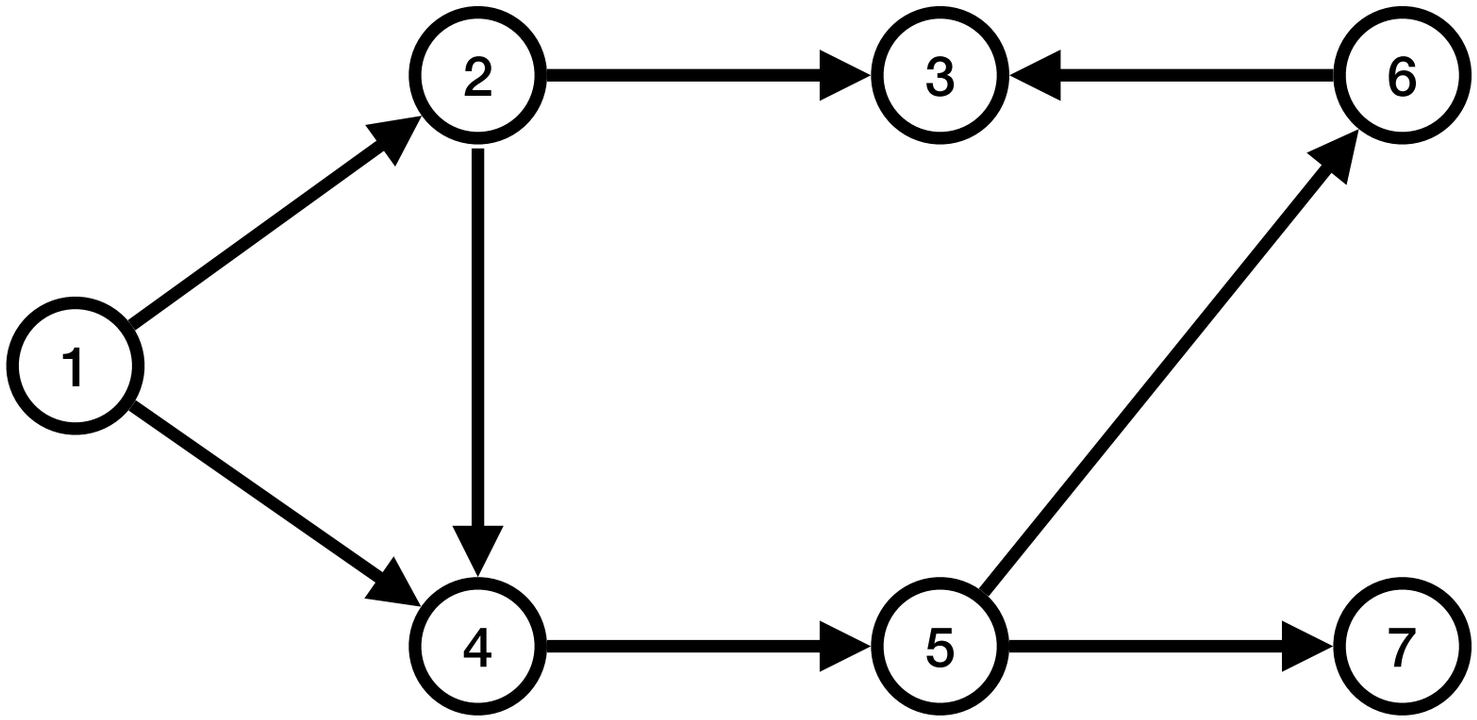


Graphe des configurations

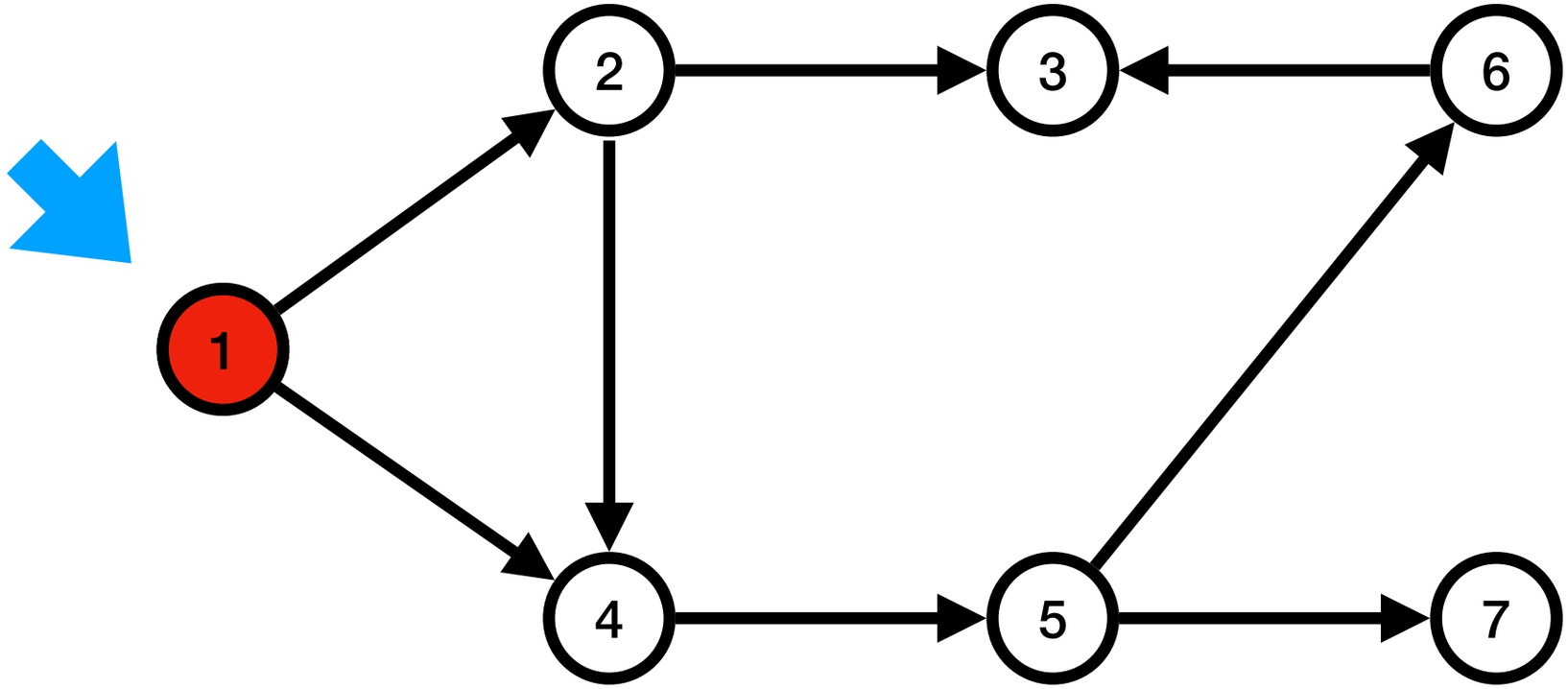


Parcours de graphes

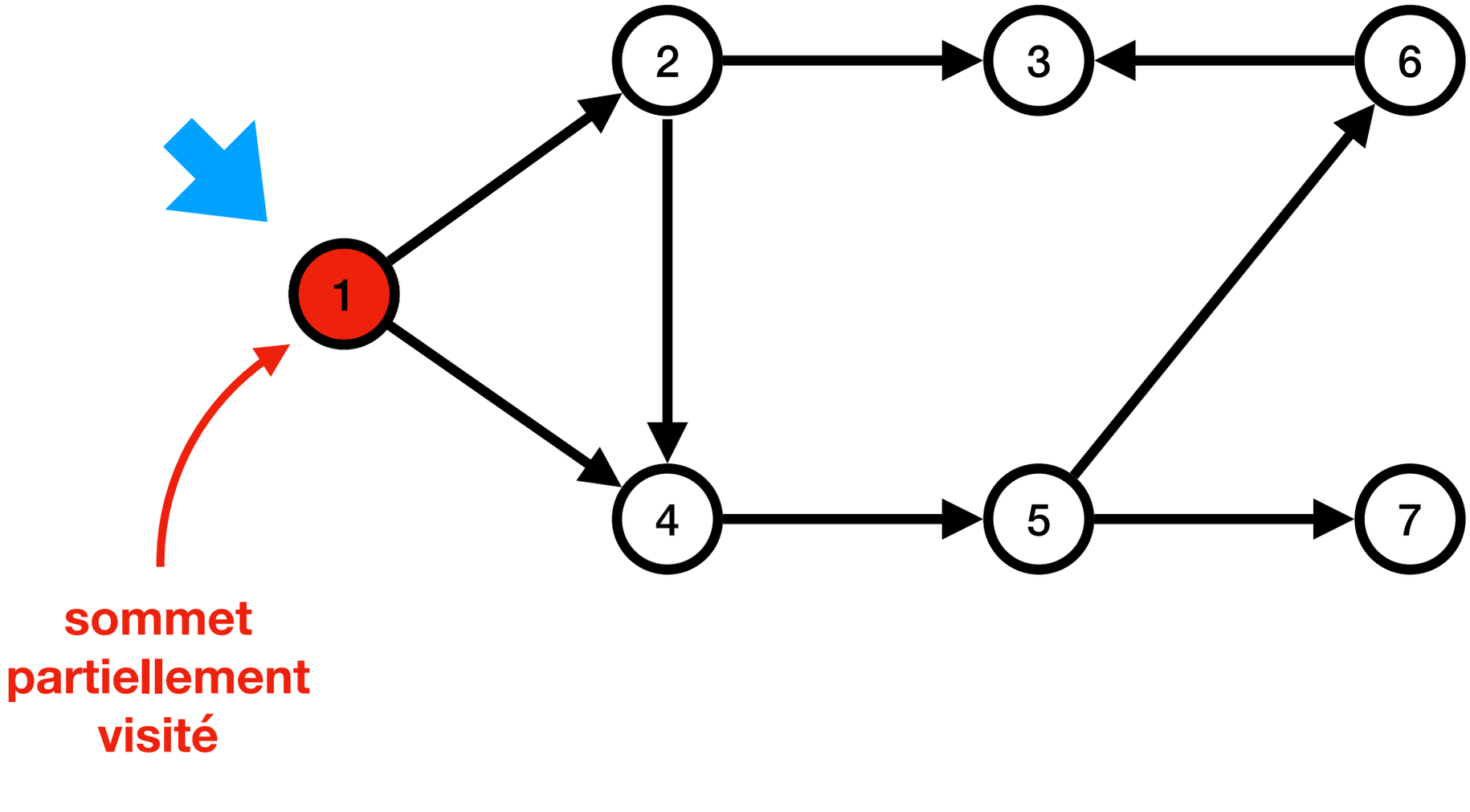
Parcours en largeur



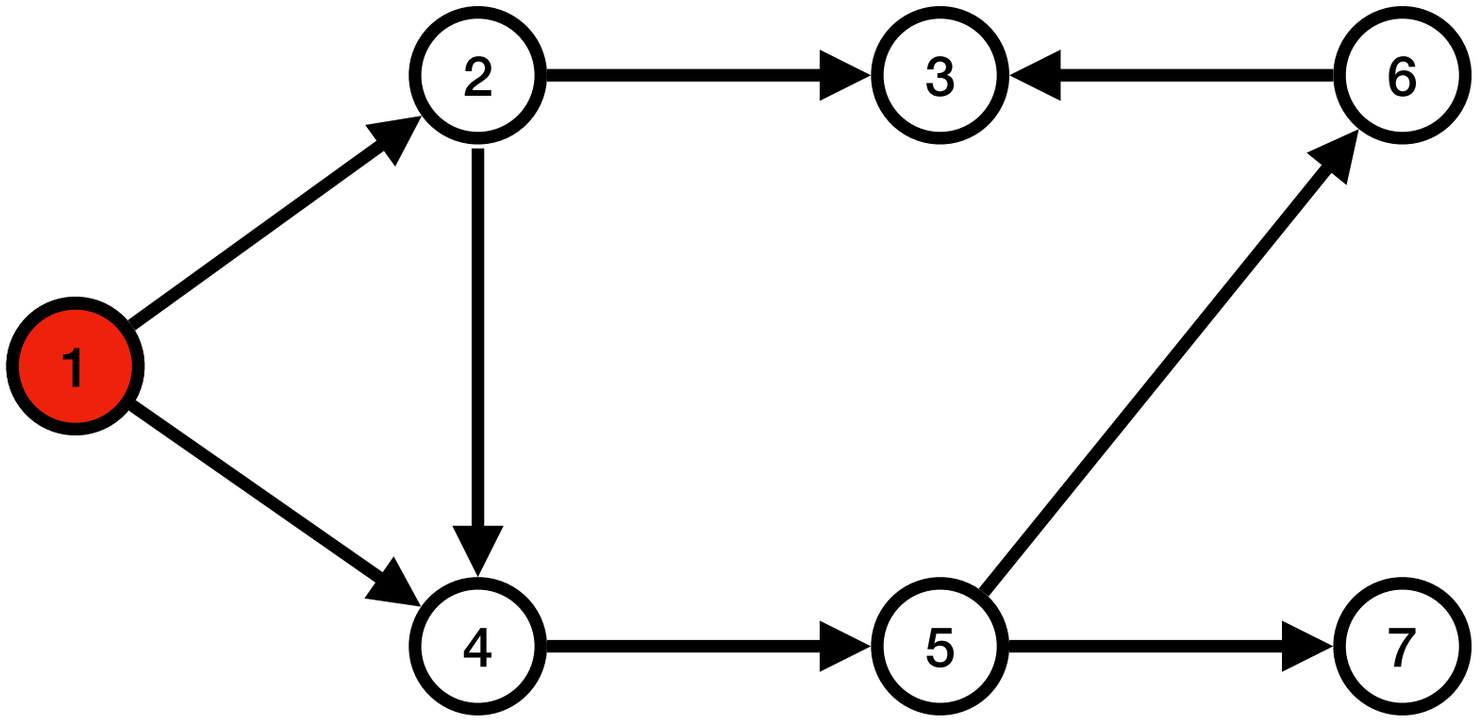
Parcours en largeur



Parcours en largeur

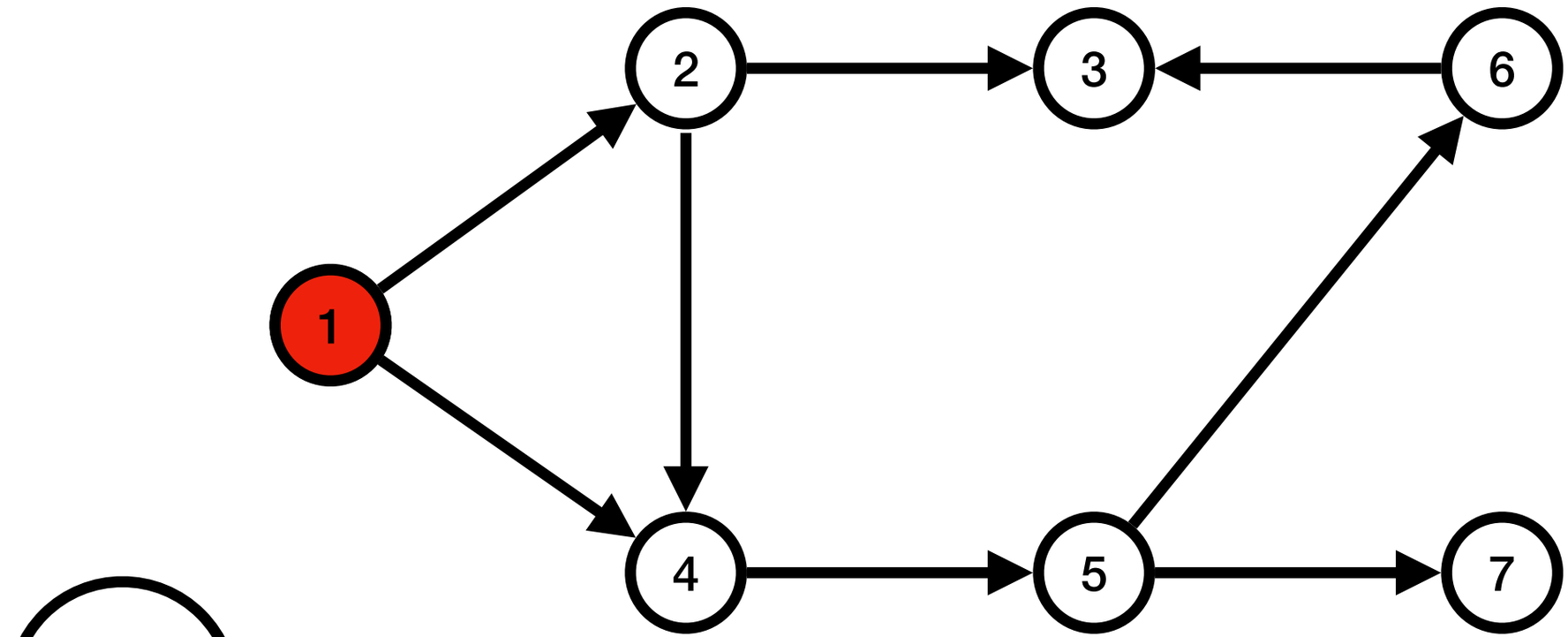


Parcours en largeur



File → 1

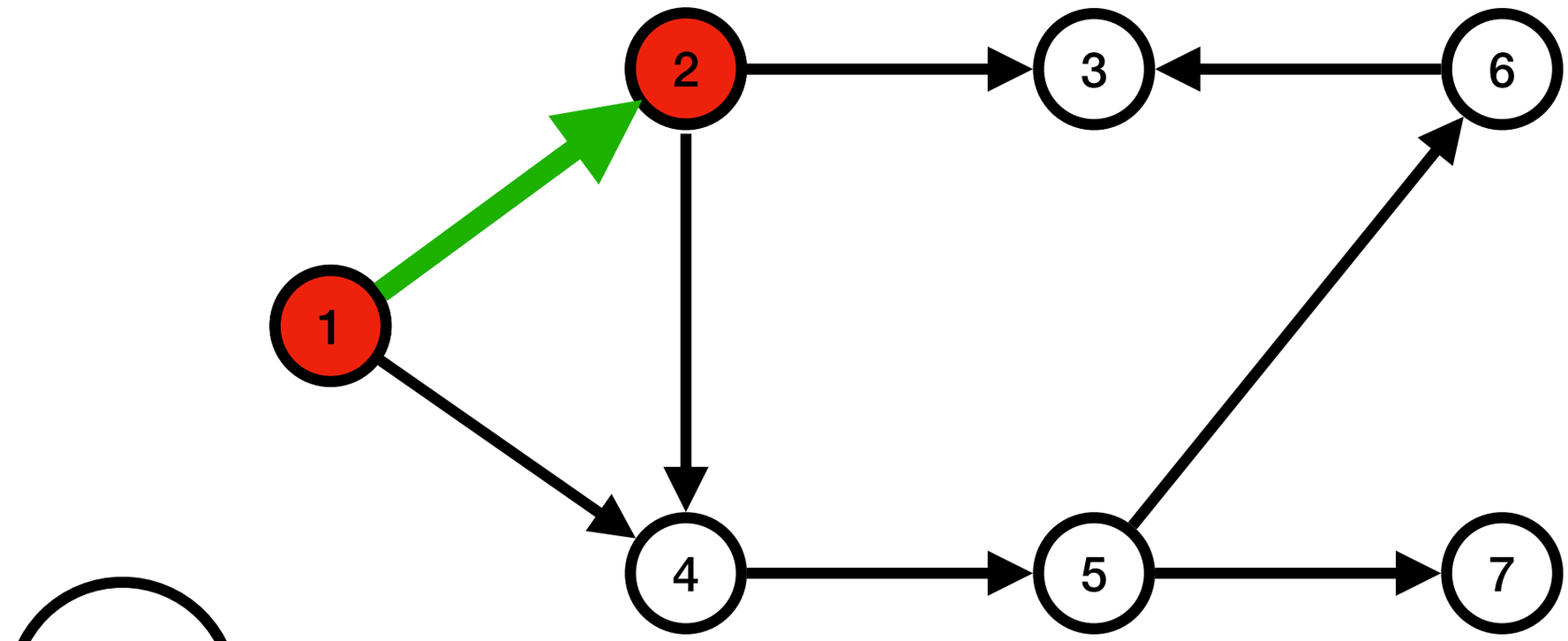
Parcours en largeur



1

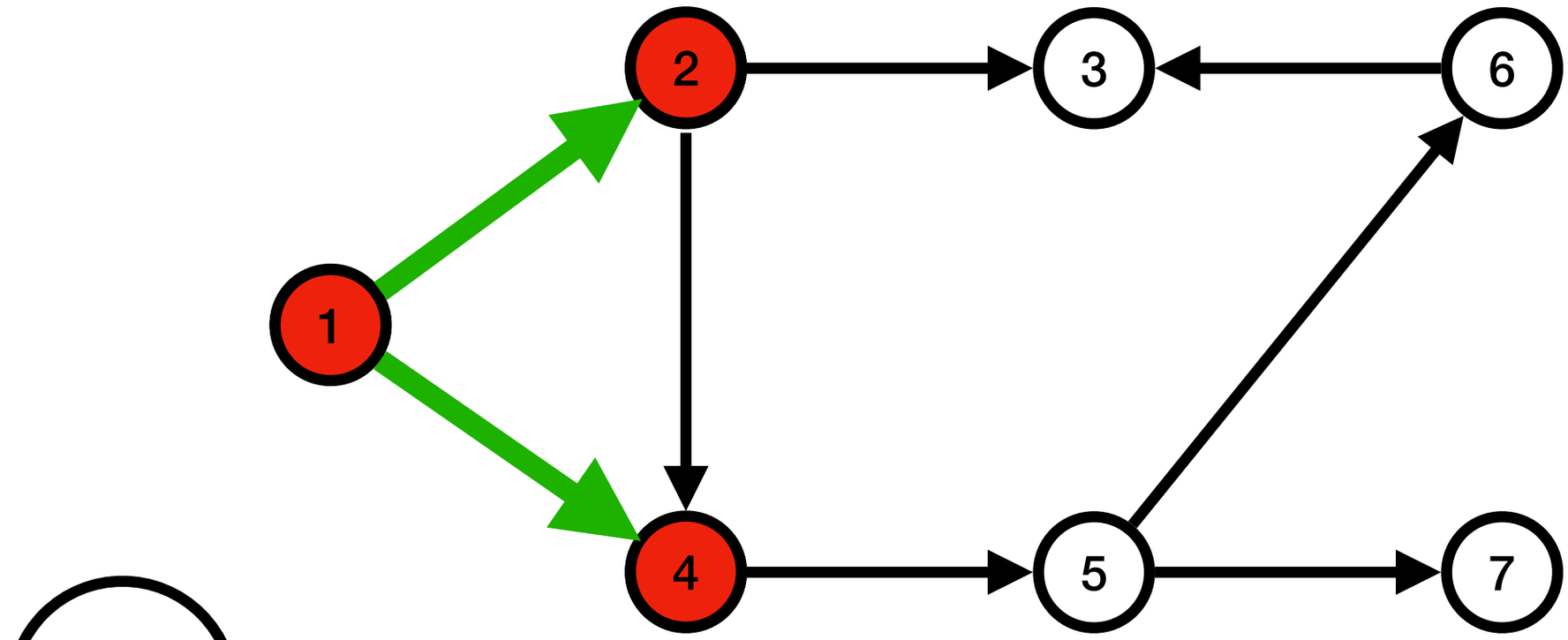
File →

Parcours en largeur



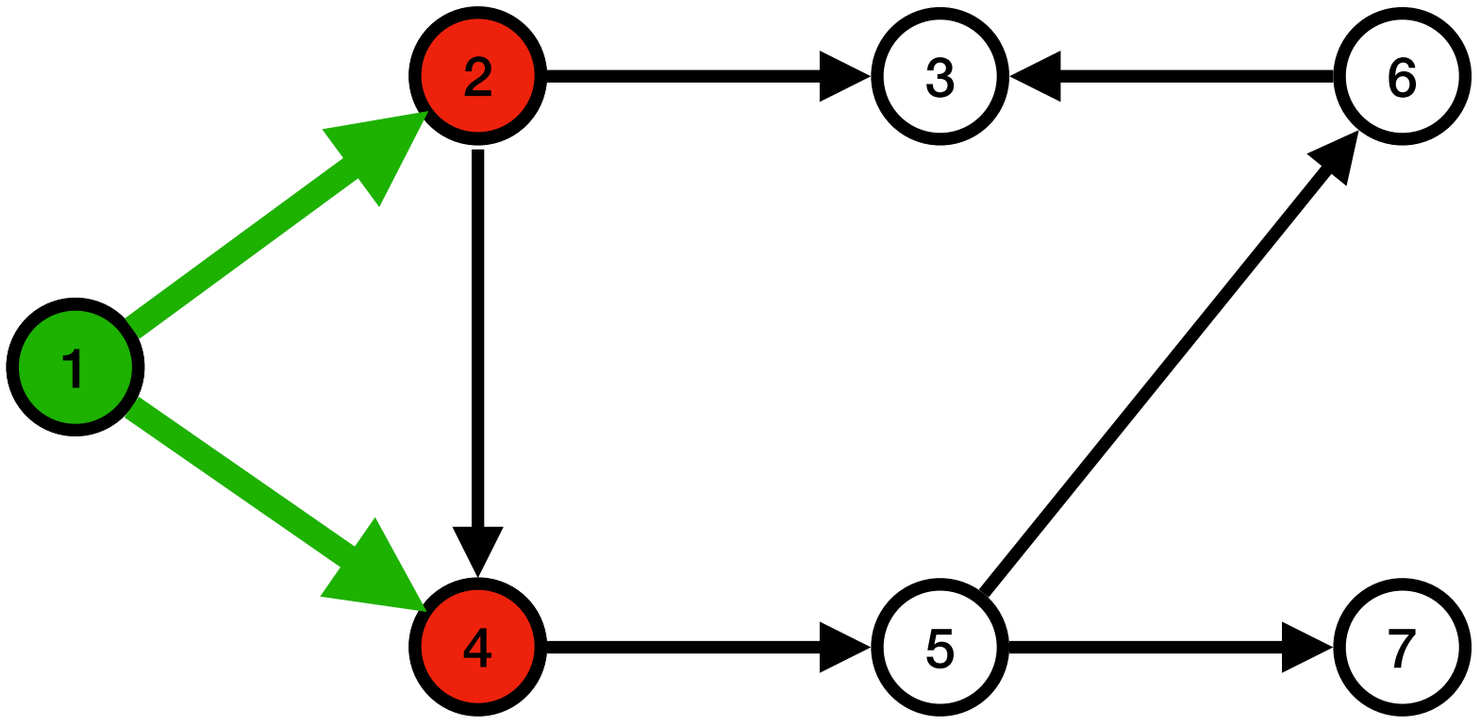
File → 

Parcours en largeur



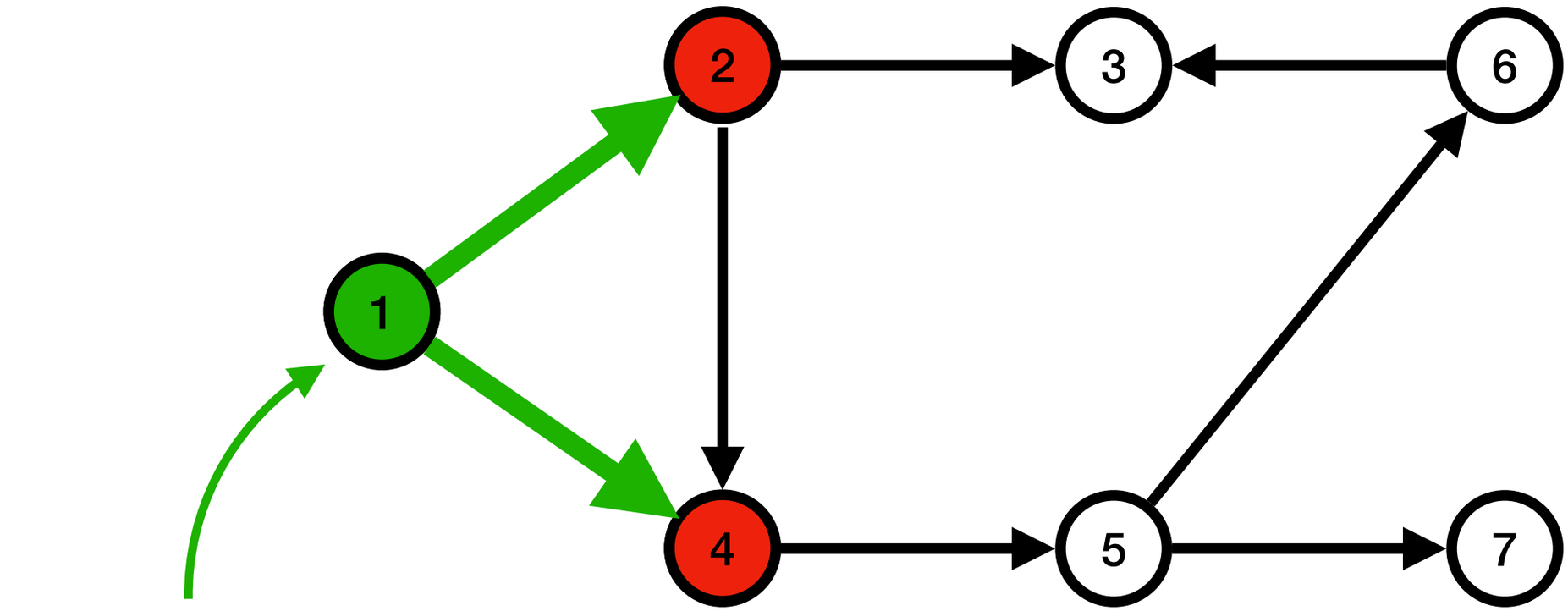
File → 2 4

Parcours en largeur



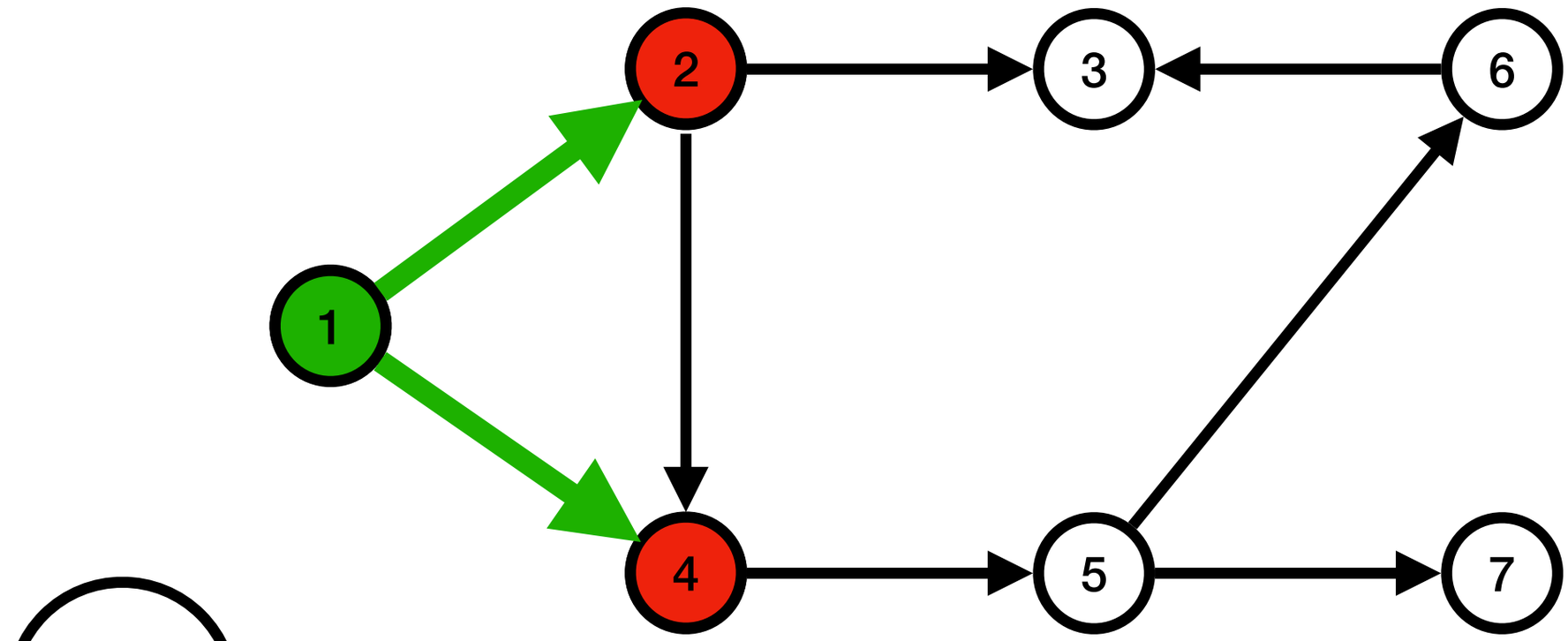
File → 2 4

Parcours en largeur



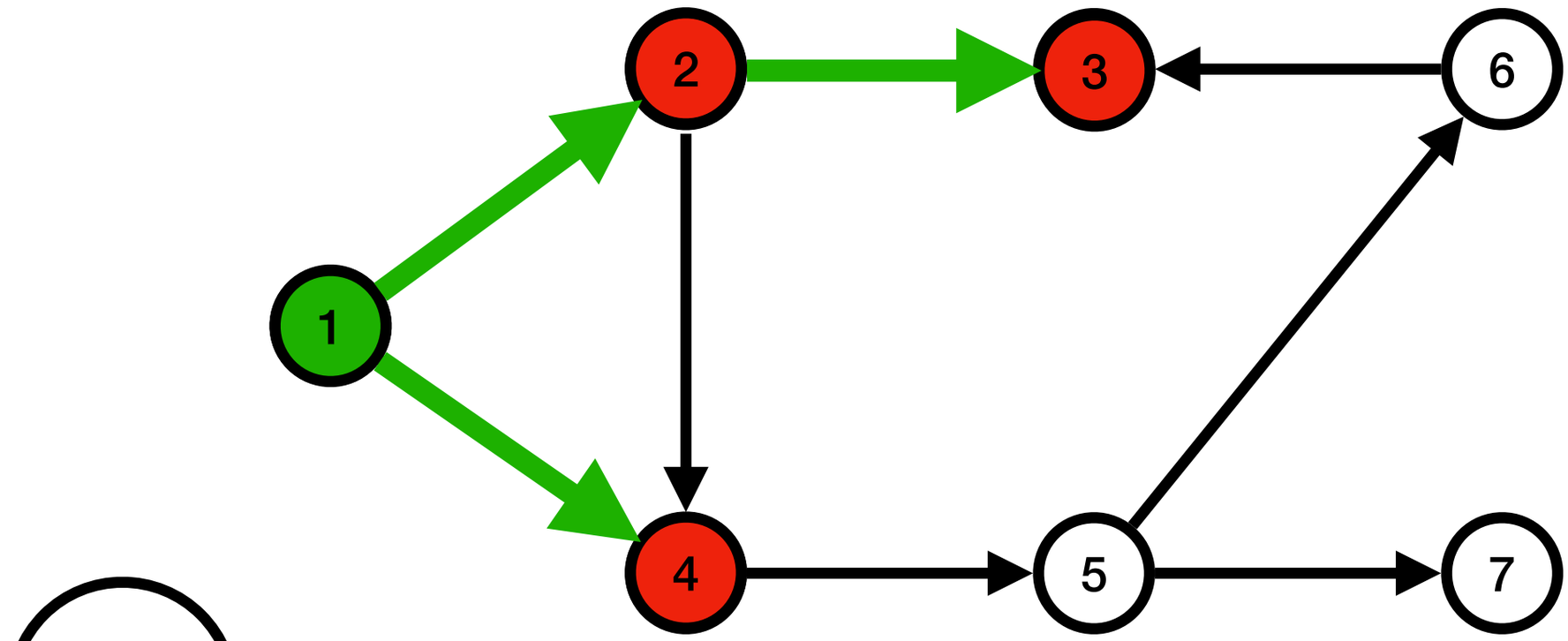
File → 2 4

Parcours en largeur



File → 

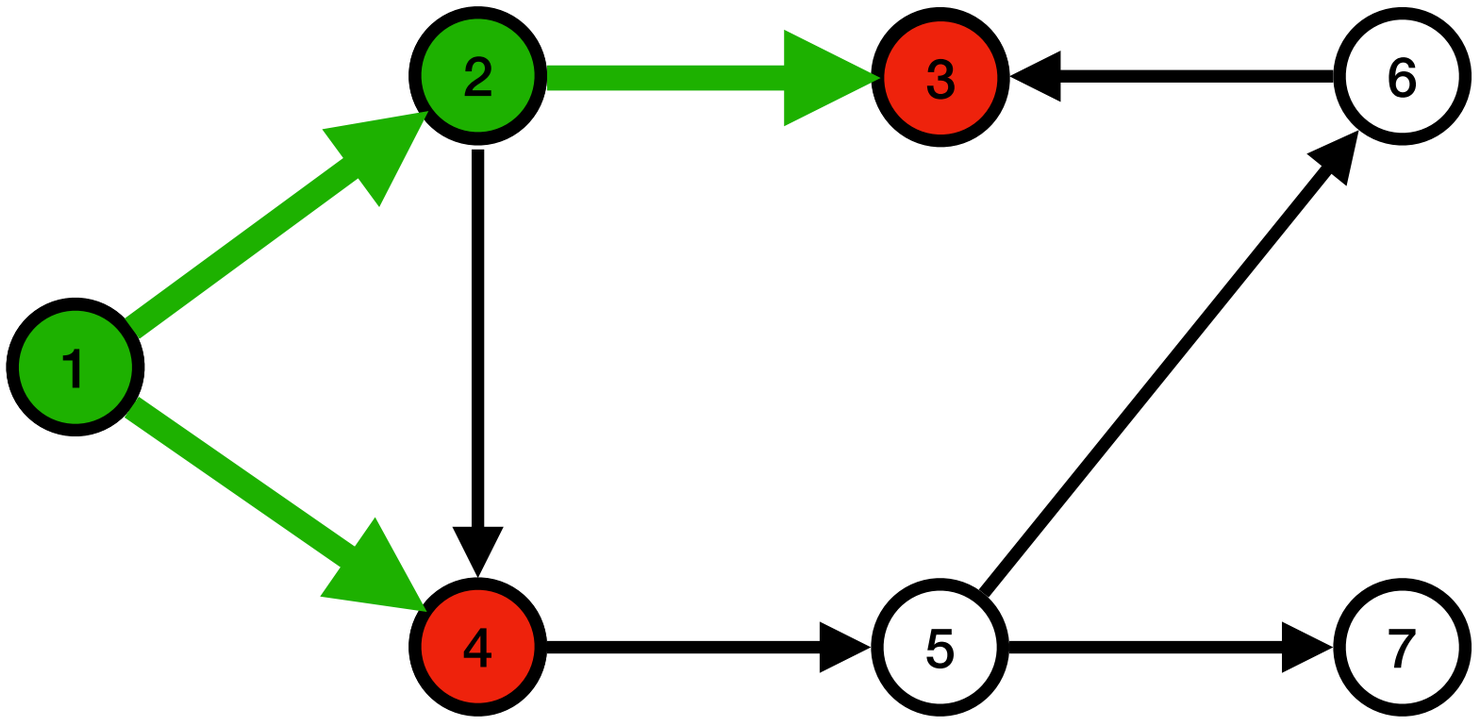
Parcours en largeur



2

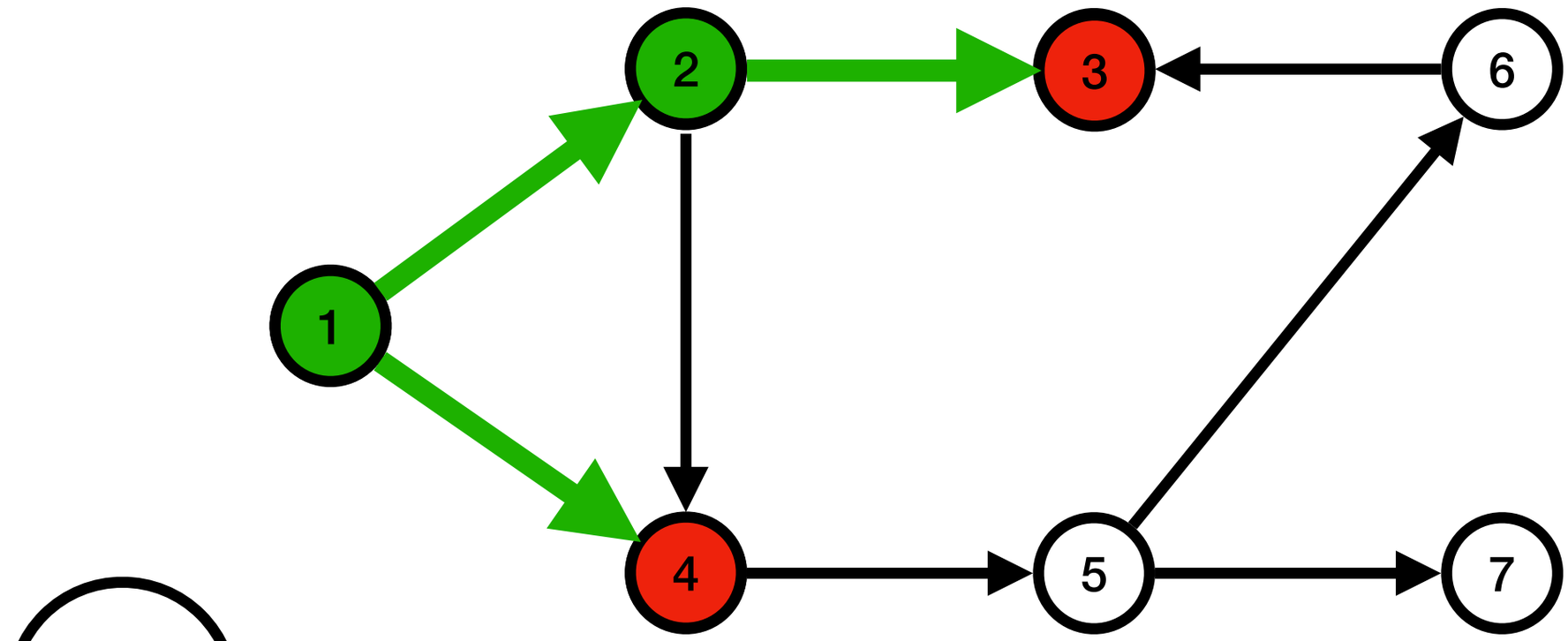
File → 4 3

Parcours en largeur



File → 4 3

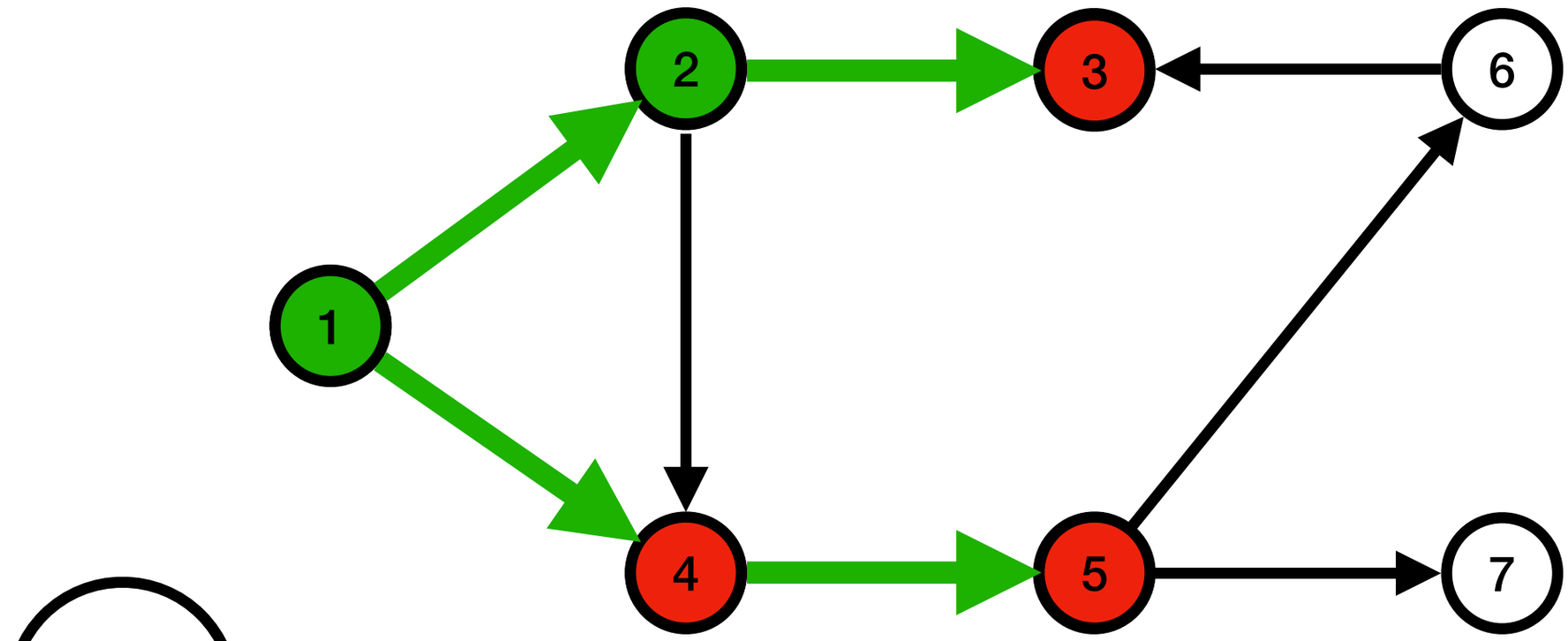
Parcours en largeur



4

File → 3

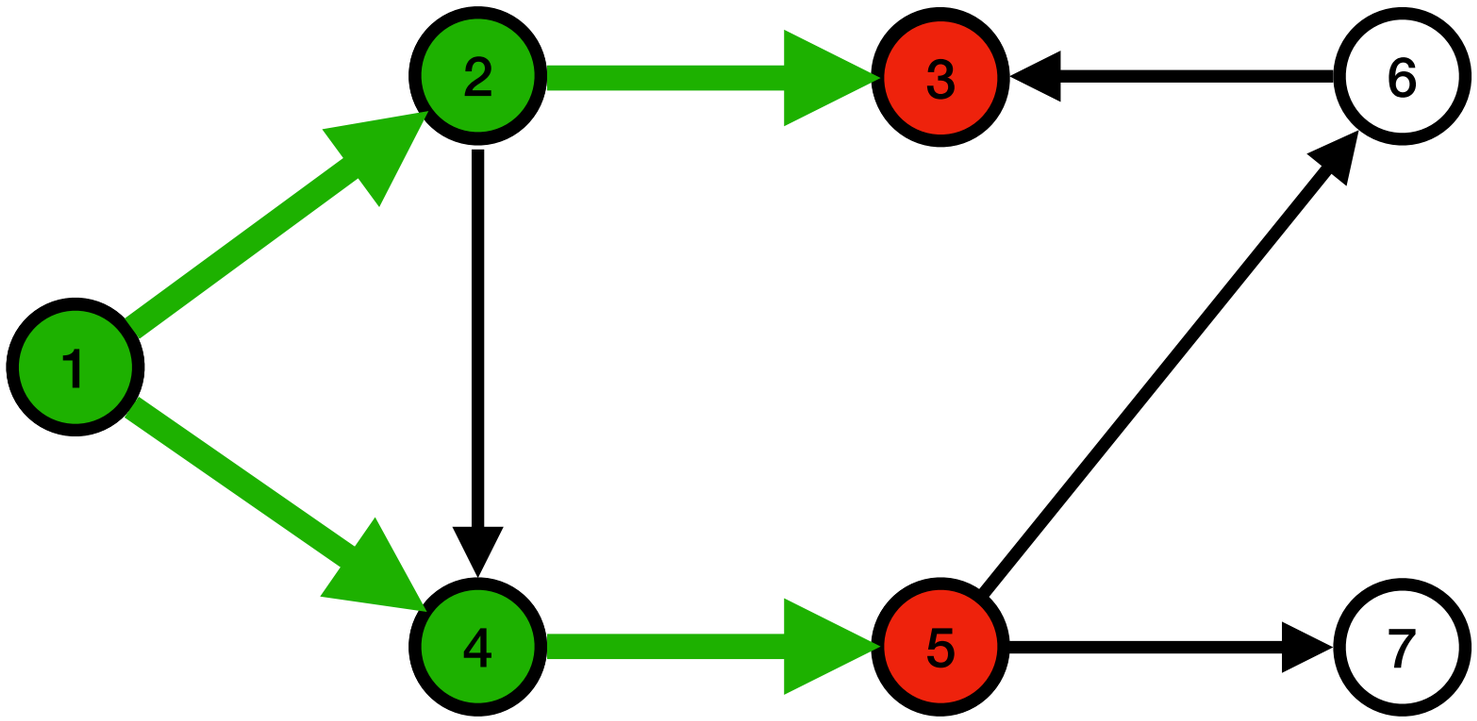
Parcours en largeur



4

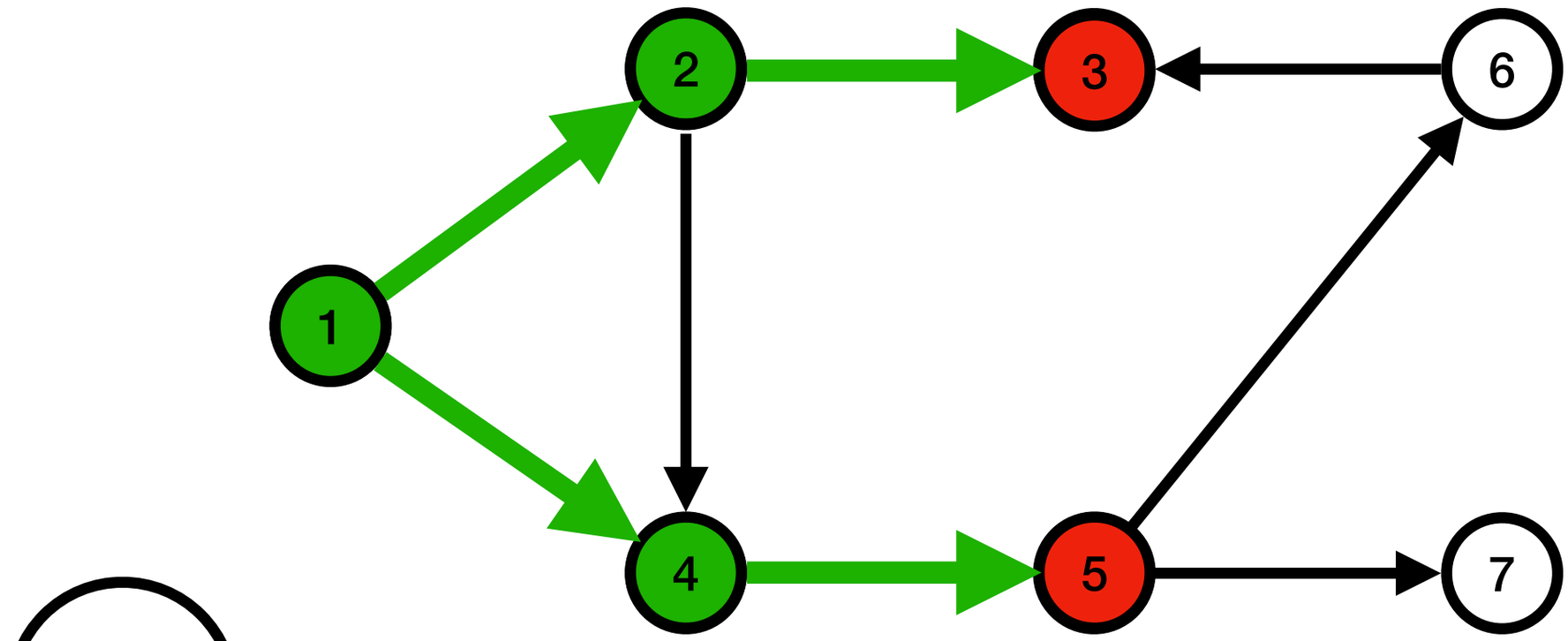
File → 3 5

Parcours en largeur



File → 3 5

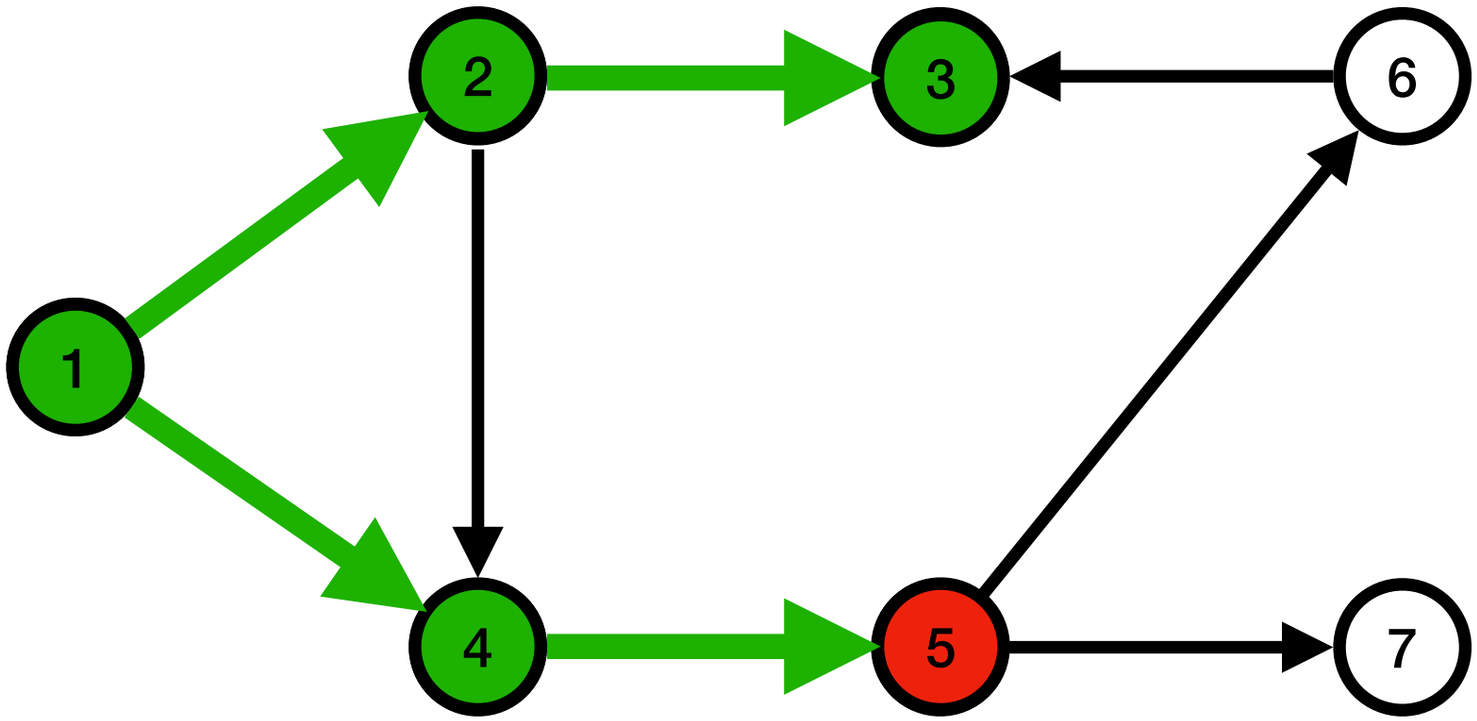
Parcours en largeur



3

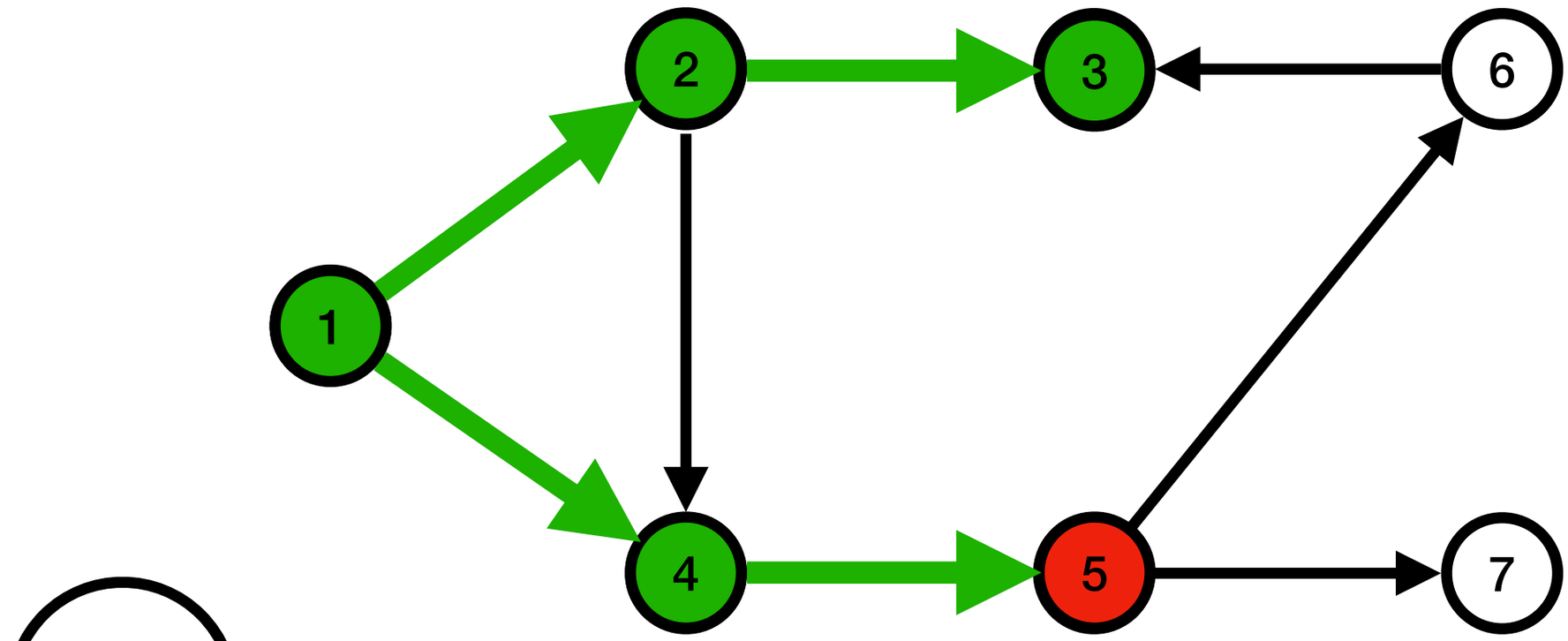
File → 5

Parcours en largeur



File → 5

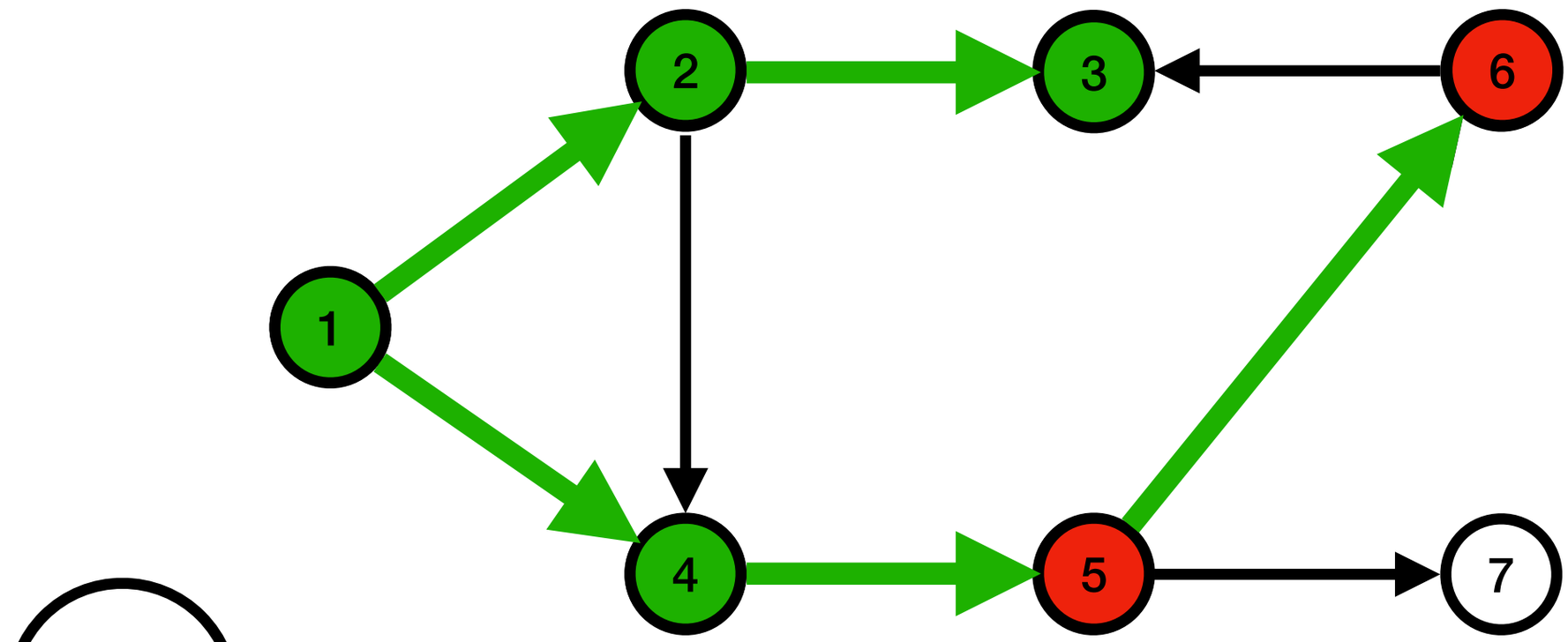
Parcours en largeur



5

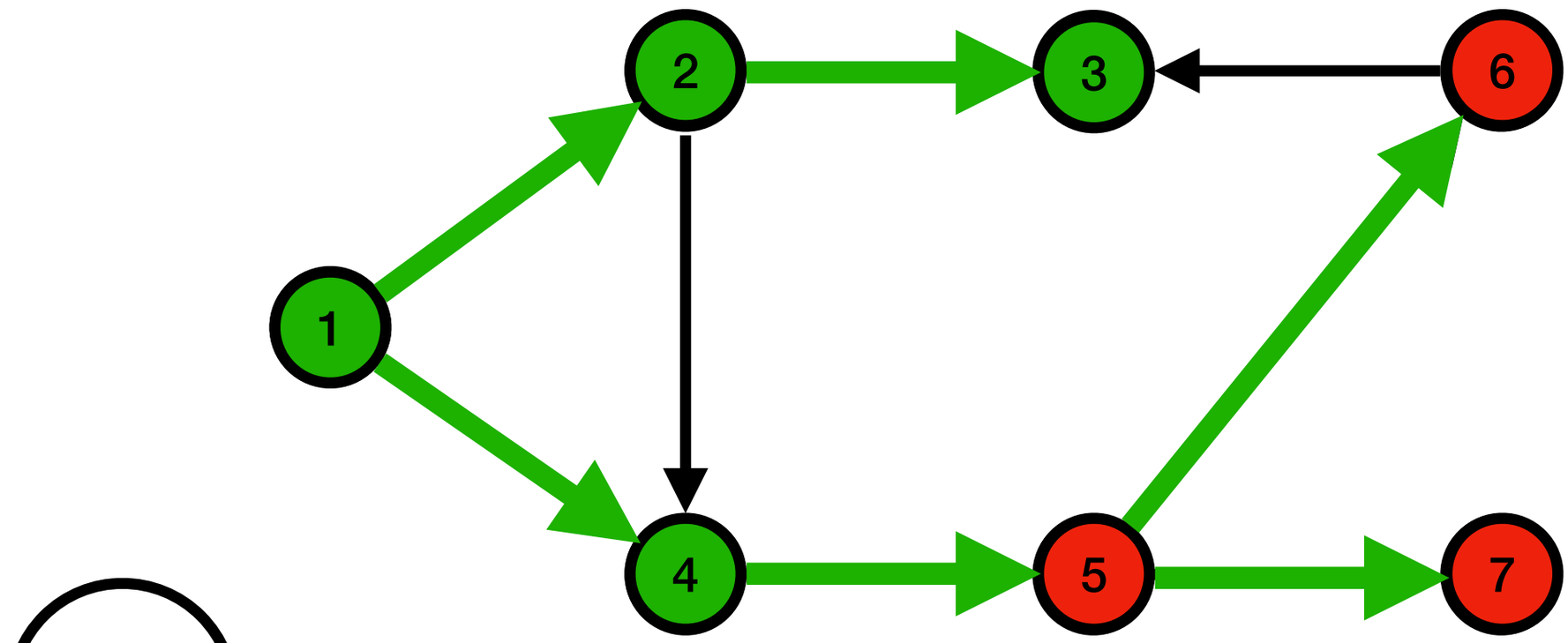
File →

Parcours en largeur



File → 

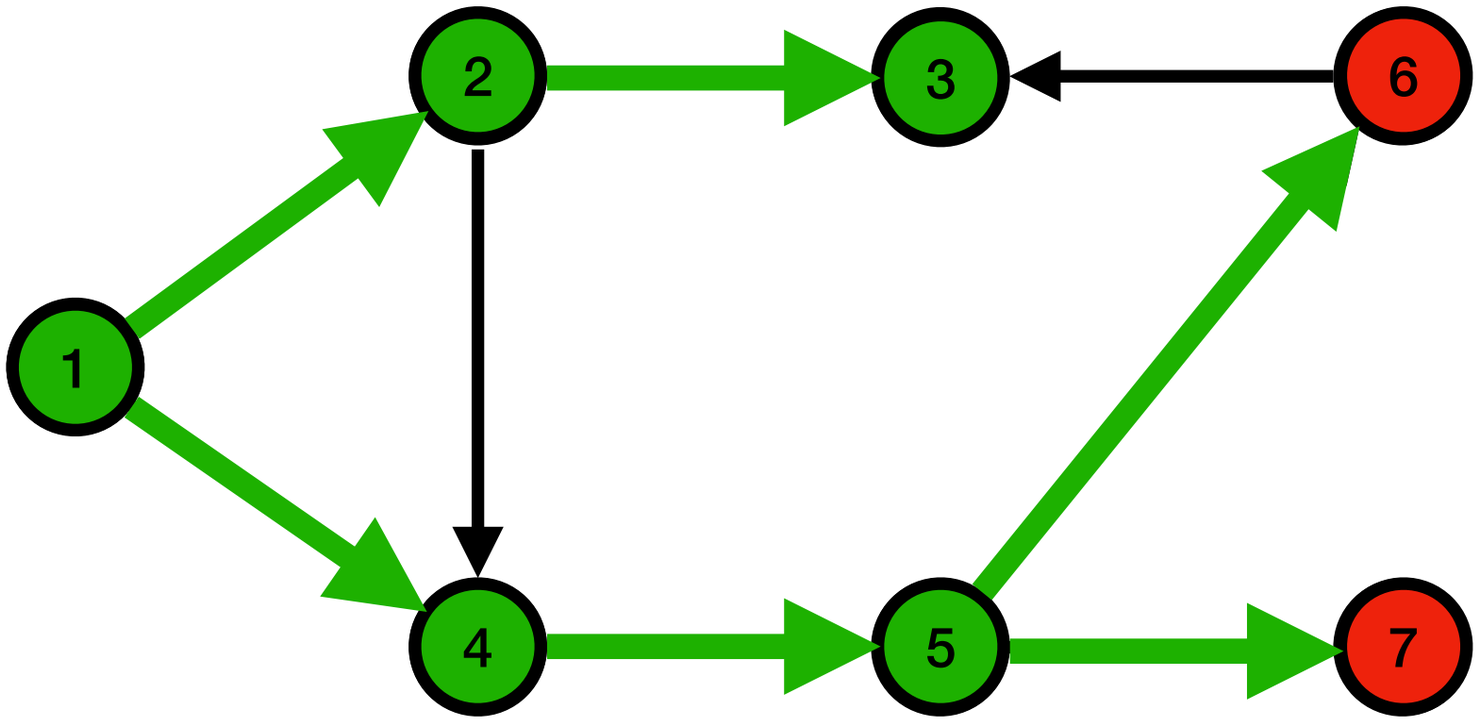
Parcours en largeur



5

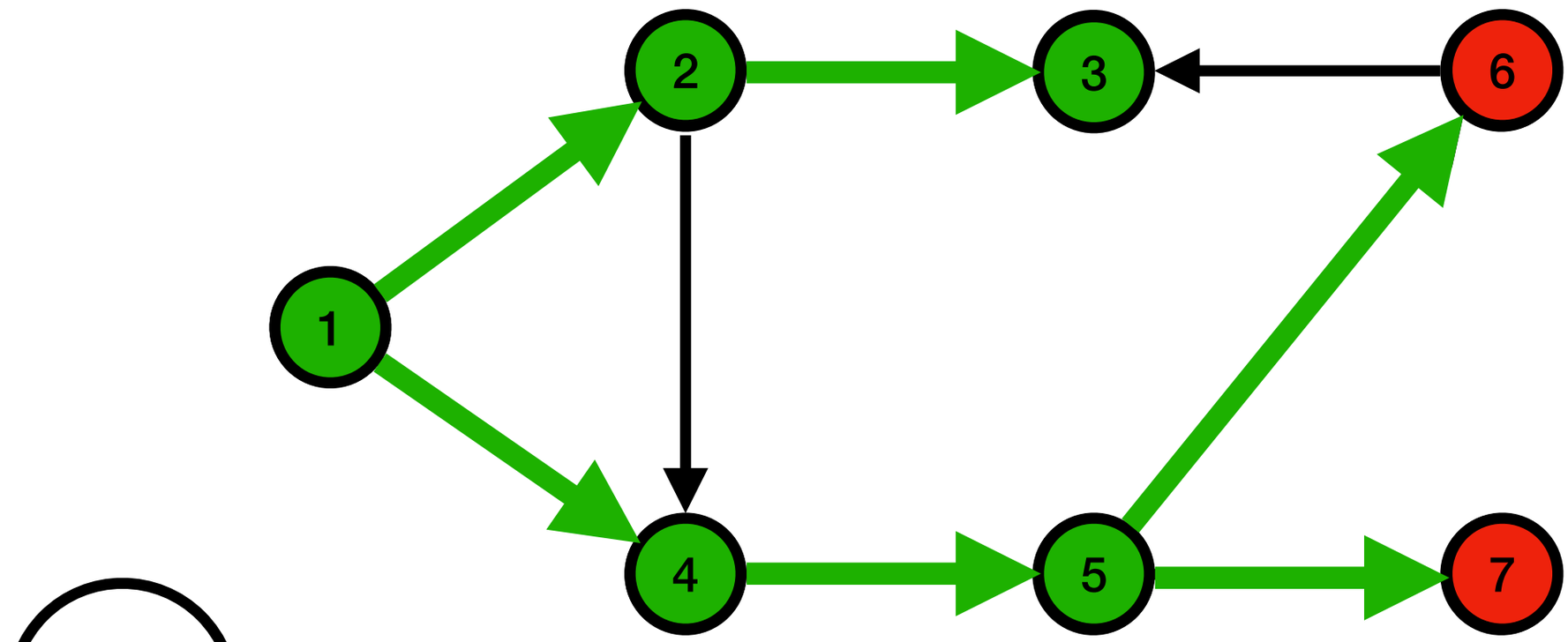
File → 6 7

Parcours en largeur



File → 6 7

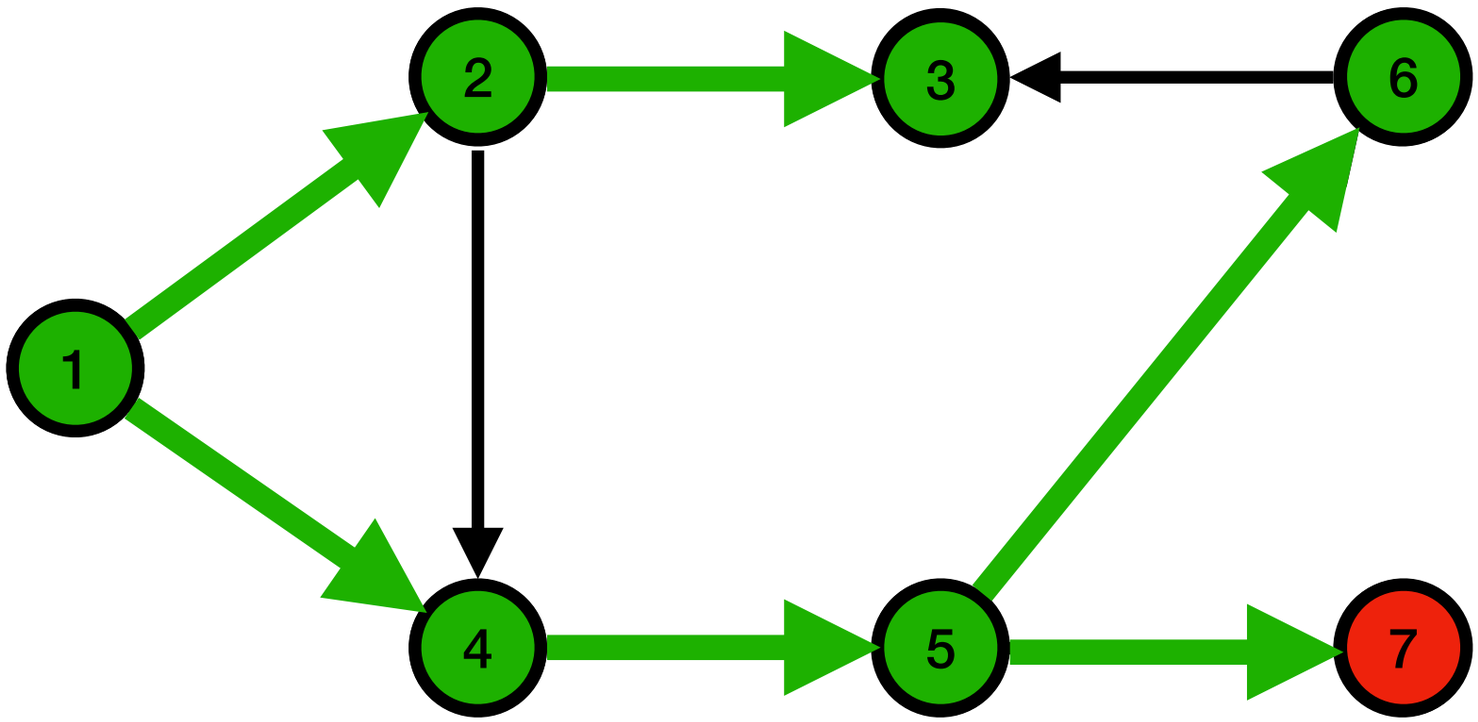
Parcours en largeur



6

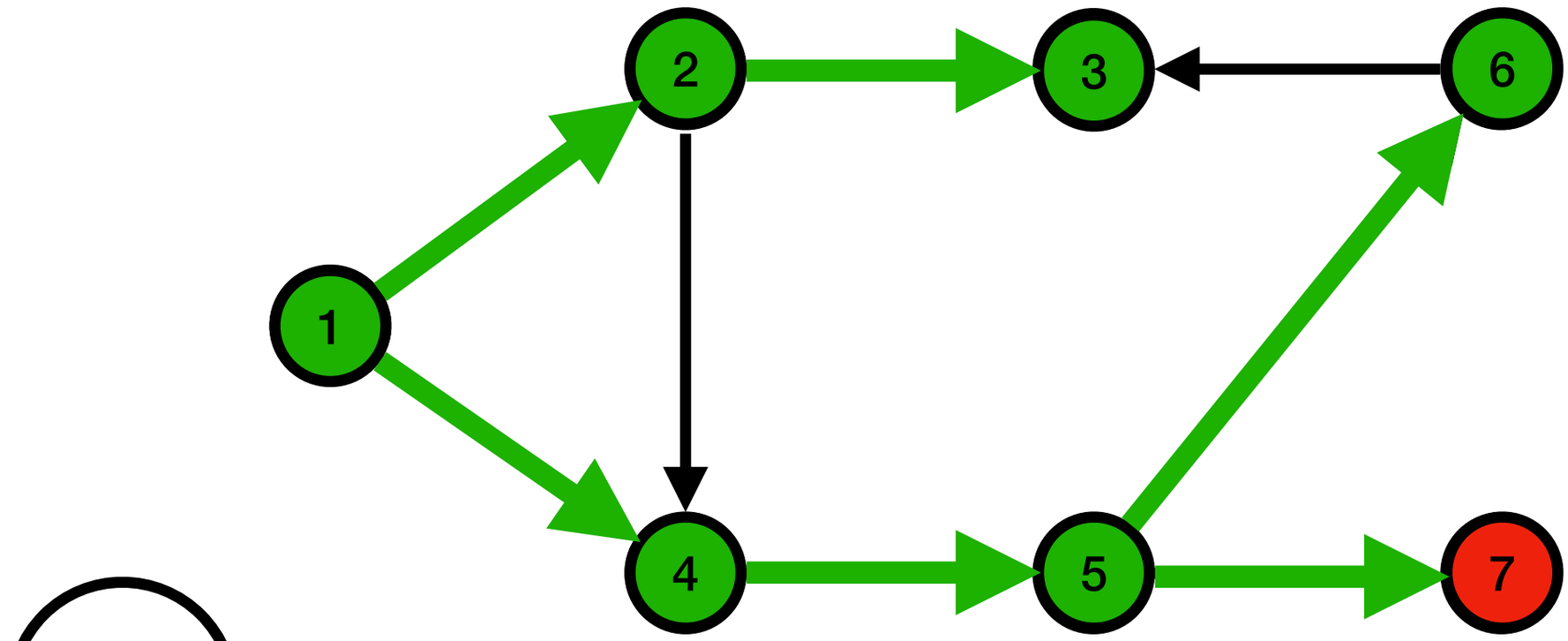
File → 7

Parcours en largeur



File → 7

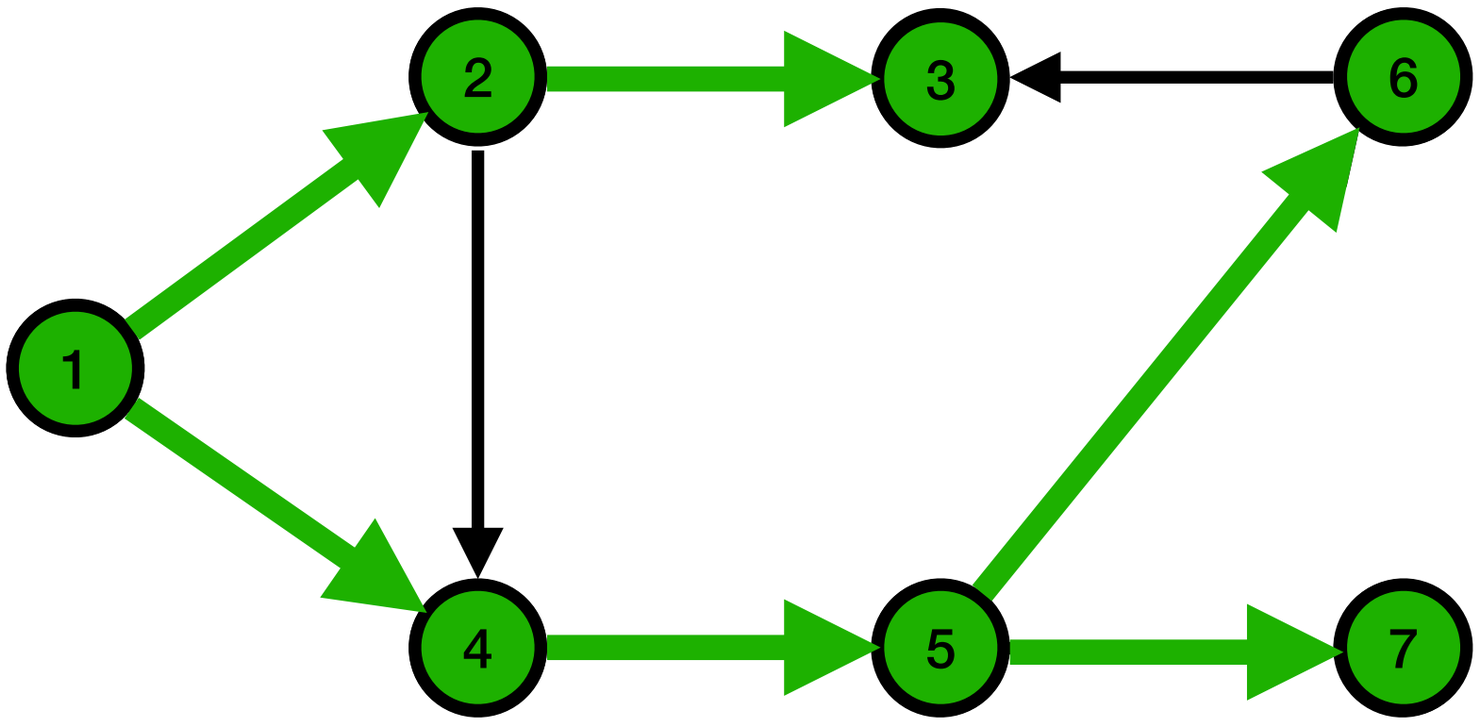
Parcours en largeur



7

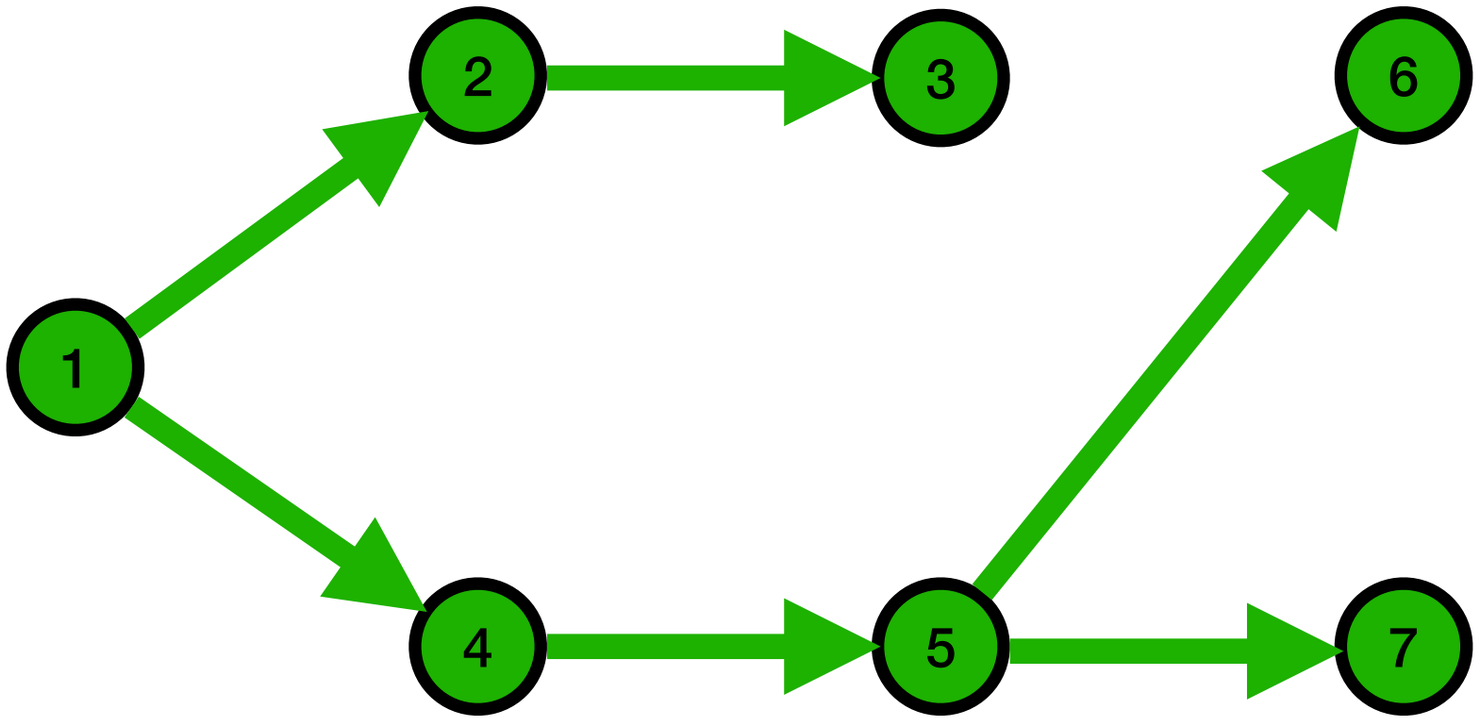
File →

Parcours en largeur



File →

Parcours en largeur



Structure de données « file »

$F := \varnothing$

Structure de données « file »

$F := \varnothing$

$F \rightarrow$

Structure de données « file »

enfiler(F, 1)

F →

Structure de données « file »

enfiler(F, 1)

F → 1

Structure de données « file »

enfiler(F, 2)

F → 1 2

Structure de données « file »

x ≔ défiler(F)

F → 2

Structure de données « file »

enfiler(F, 3)

F → 2 3

Structure de données « file »

enfiler(F, 4)

F → 2 3 4

Structure de données « file »

enfiler(F, 5)

F → 2 3 4 5

Structure de données « file »

x ≔ défiler(F)

F → 3 4 5

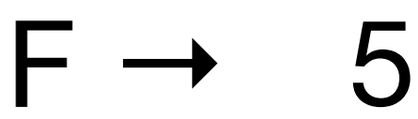
Structure de données « file »

x ≔ défiler(F)

F → 4 5

Structure de données « file »

x ≔ défiler(F)



Structure de données « file »

x ≔ défiler(F)

F →

Structure de données « file »

F →

Code pour la file

Code pour la file

```
def nouvelle_file():
    return []
```

Code pour la file

```
def nouvelle_file():
    return []

def est_vide(F):
    return F == []
```

Code pour la file

```
def nouvelle_file():
    return []

def est_vide(F):
    return F == []

def enfiler(F, x):
    F.append(x)
```

Code pour la file

```
def nouvelle_file():
    return []

def est_vide(F):
    return F == []

def enfiler(F, x):
    F.append(x)

def défiler(F):
    x = F.pop(0)
    return x
```

Parcours en largeur

```
def parcours_en_largeur(G, s):
    n = len(G)
    H = nouveau_graphe(n)          # graphe vide
    F = nouvelle_file()            # file vide
    couleur = ['blanc'] * n        # n cases blanches
    couleur[s] = 'rouge'
    enfiler(F, s)
    while not est_vide(F):
        u = défiler(F)
        for v in range(n):
            if G[u][v] == 1 and couleur[v] == 'blanc':
                couleur[v] = 'rouge'
                enfiler(F, v)
                H[u][v] = 1
        couleur[u] = 'vert'
    return H                       # graphe des chemins min
```