Graphes

Graphes

1. Définitions et exemples
2. Implémentations
3. Ordre topologique
4. Chemin le plus court
5. Dijkstra
6. Parcours

Graphes

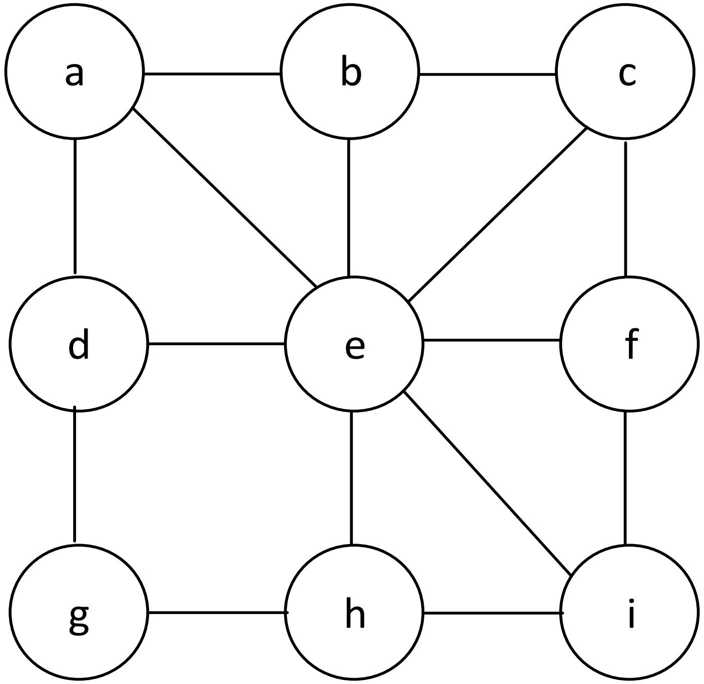
1. Définitions et exemples
2. Implémentations
3. Ordre topologique
4. Chemin le plus court
5. Dijkstra
6. Parcours

Graphes - définitions

- Un graphe est une paire G = (V,E) où V est un ensemble de **sommets** et E un ensemble d'**arêtes**. Chaque arête est une paire qui relie deux sommets du graphe.

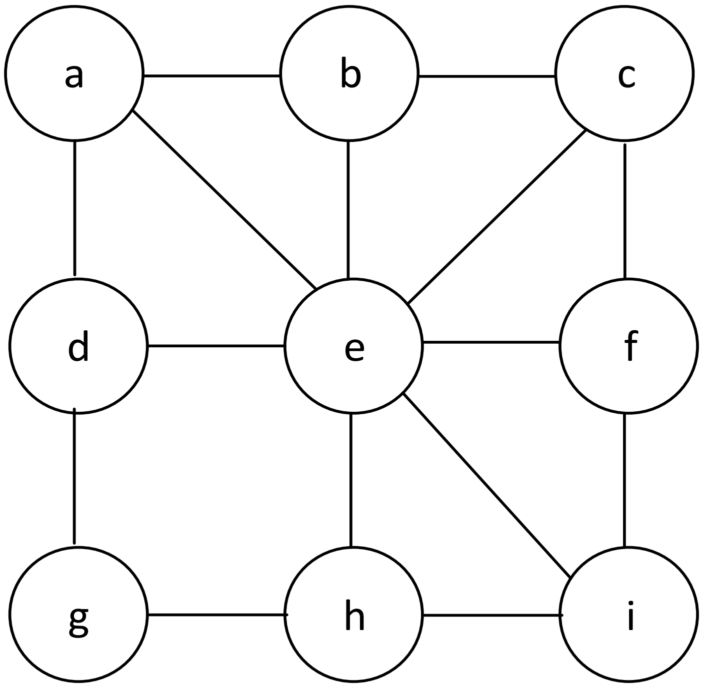
V = {a, b, c, d, e, f, g, h, i}

E = {(a, b), (a, d), (a, e),
(b, c), (b, d), (c, e), (c, f),
(d, e), (d, g), (e, f), (e, h), (e, i),
(f, i), (g, h), (h, i)}

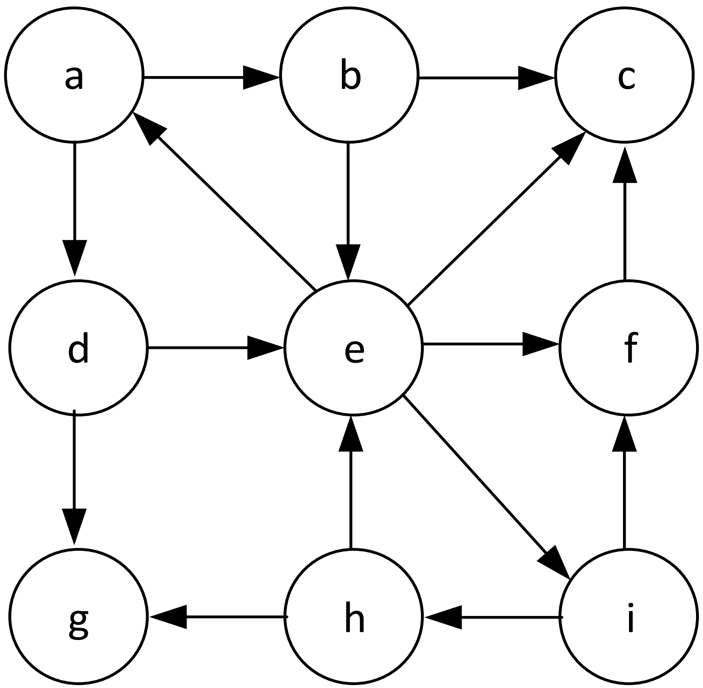


Graphes - définitions

- Graphe orienté: les sommets sont reliés par des **arcs** (arêtes orientées), qui relient un sommet origine à un sommet destination



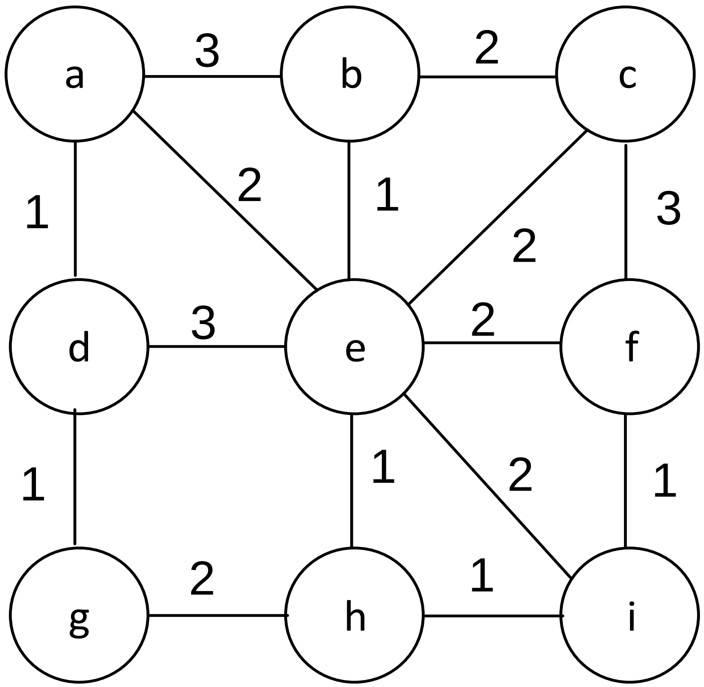
Graphe non-orienté

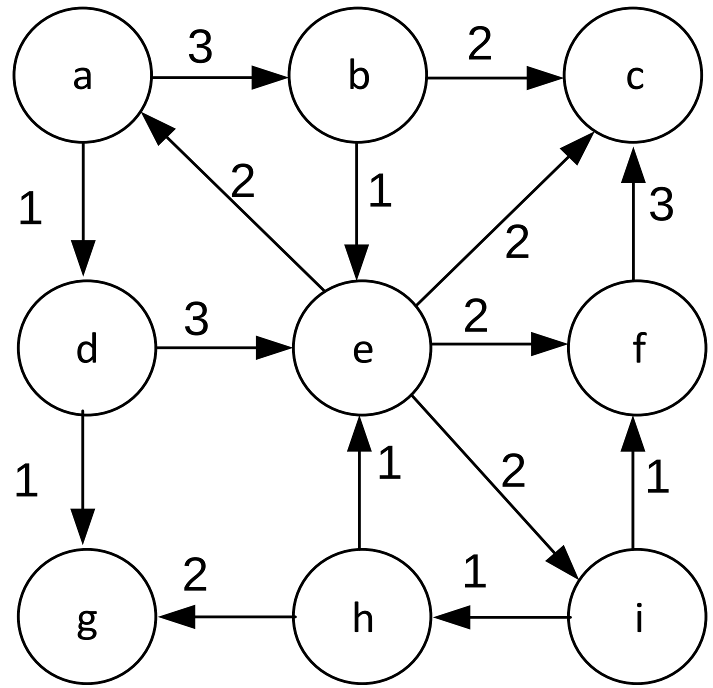


Graphe orienté

Graphes - définitions

- Un graphe est dit **valué** si les arêtes (ou arcs) ont une valeur indiquant le **coût** pour les traverser. On peut aussi parler de **poids** de chaque arête (arc).



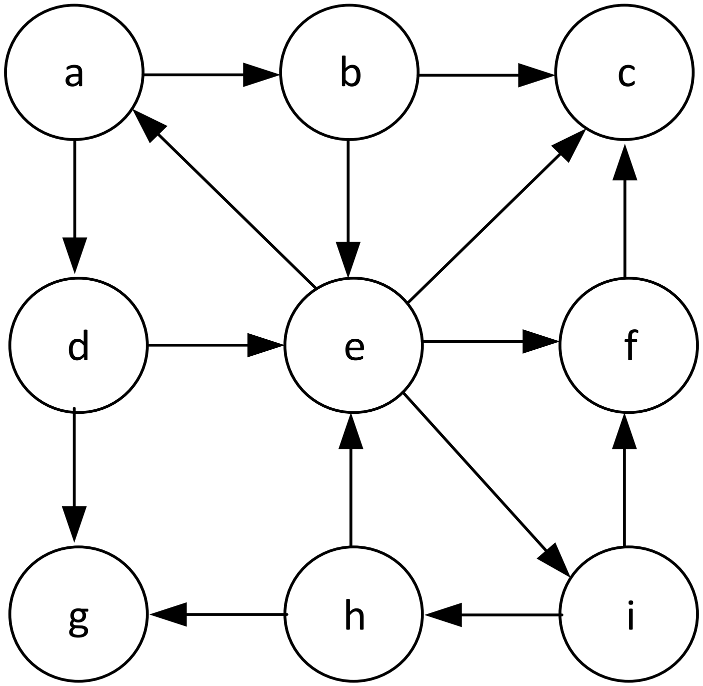


Graphes – définitions

- Un **chemin** est une séquence de sommets du graphe connectés par des arêtes
- La **longueur** d'un chemin correspond au nombre d'arêtes dans ce chemin
- Un **chemin simple** ne contient pas plus d'une fois le même sommet
- Un **cycle** est un chemin qui commence et termine au même sommet
- Un **graphe orienté acyclique** est un graphe orienté qui ne contient pas de cycle

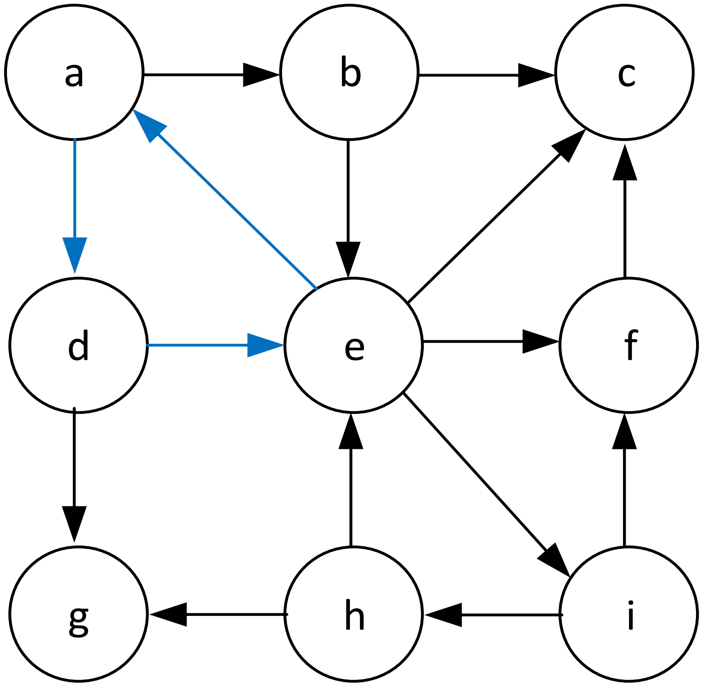
Graphes – définitions

- Est-ce un graphe orienté acyclique ?



Graphes – définitions

- Est-ce un graphe orienté acyclique ? **NON**



Graphes – définitions

- Graphe connexe → un chemin pour chaque paire de nœuds

- Graphes orientés
  - connexes → connexité forte
  - Non connexes, mais le graphe sous-jacent sans orientation est connexe → connexité faible

Graphes – définitions

- Un graphe complet comportant $|V|$ nœuds possède $|E| = (|V| - 1)\cdot(|V|)/2$ arcs

- On dit qu'un graphe est dense si $|E|$ est $\Theta(|V|^2)$

- On dira qu'un graphe est peu dense si $|E|$ est $\Theta(|V|)$

Graphes – exemples

- Réseaux
  - Sociaux
  - Ordinateurs
  - Transports

- Théorie des langages
  - Automates
  - Graphe de flot de contrôle
  - Graphes d’appel
  - Graphes des dépendances

Graphes – exemples

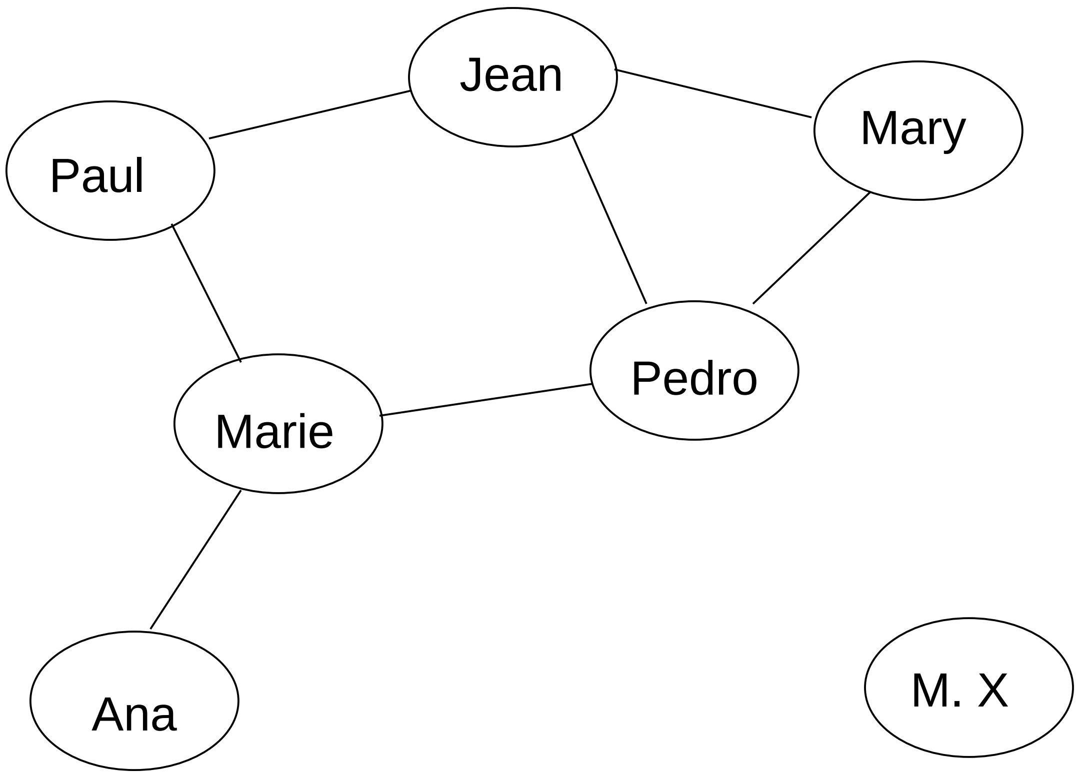
- Généalogies

- Biologie moléculaire
  - Chaînes métaboliques
  - Représentation de protéines et de molécules organiques

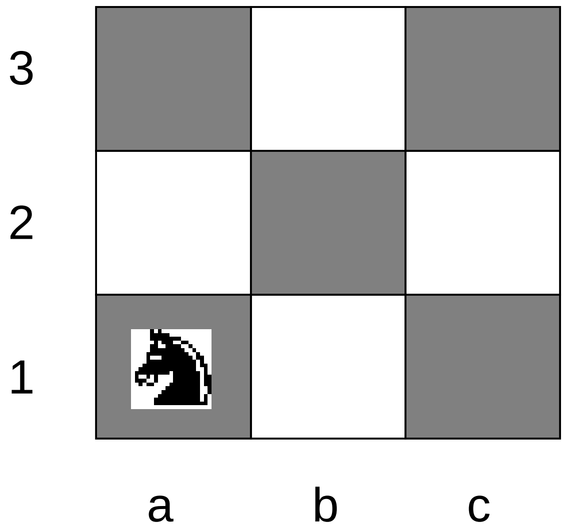
- Génie logiciel
  - Diagrammes UML (classe, interaction)
  - Diagramme de transition des Interfaces usager

Graphes - exemples

Graphe non orienté: chaque arête représente deux personnes qui se connaissent

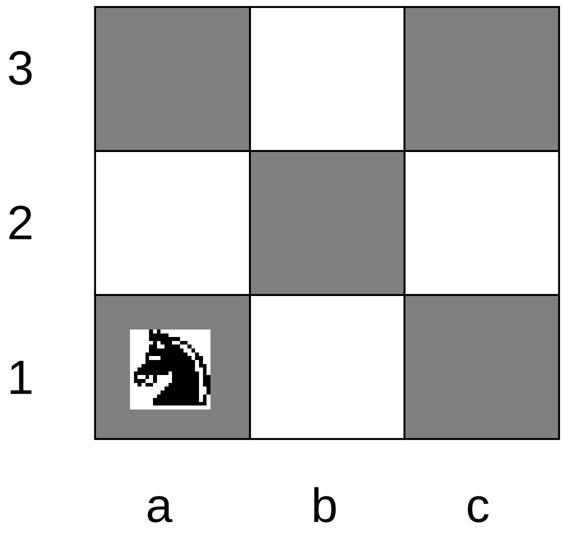


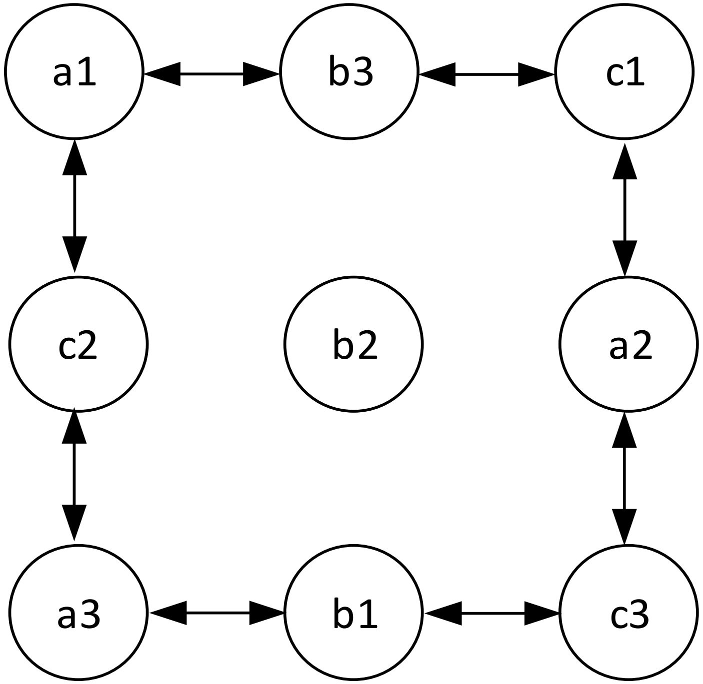
Graphes – exemples



Quelles sont les positions possibles du cavalier à partir de sa position actuelle?

Graphes – exemples





Remarque: dans la figure, chaque arc représente en fait deux arcs, soit un pour chaque orientation

Graphes

1. Définitions et exemples
2. Implémentations
3. Ordre topologique
4. Chemin le plus court
5. Dijkstra
6. Parcours

Graphes - implémentation

- Matrice d'adjacence
  - On suppose que les sommets du graphe sont étiquetés de 0 à N
  - S'il existe une arête du sommet *i* au sommet *j*, on met 1 à la position A[i][j], sinon on met INFINI comme valeur (autres codages existent)

Graphes – implémentation

- Matrice d'adjacence
  - Si le graphe est valué, on met à la position A[i][j], le poids associé à l'arête
  - Si le graphe est peu dense, ce qui est souvent le cas, il y aura beaucoup de 0 dans la matrice

Graphes – implémentation

- Listes d’adjacence
  - Pour chaque sommet, on associe une liste de tous les autres sommets auquel il est lié par une arête dont il est l’origine
  - En principe (tout comme avec la matrice d’adjacence), il faut une table qui associe l’identificateur de chaque sommet à un numéro interne dans la représentation
  - En Java, cette table peut nous retourner une référence sur la structure qui représente le sommet

Graphes

1. Définitions et exemples
2. Implémentations
3. Ordre topologique
4. Chemin le plus court
5. Dijkstra
6. Parcours

Ordre topologique

- Graphes orientes acycliques

- Définition
  - Ordre sur les nœuds du graphe dans lequel l'existence d'un chemin entre x et y implique que x précède y

Algorithme

```
void topsort( ) throws CycleFoundException
{
    for( int counter = 0; counter < NUM_VERTICES; counter++ )
    {
        Vertex v = findNewVertexOfIndegreeZero( );
        if( v == null )
            throw new CycleFoundException( );
        v.topNum = counter;
        for each Vertex w adjacent to v
            w.indegree--;
    }
}
```

Algorithme

```
void topsort( ) throws CycleFoundException
{
    for( int counter = 0; counter < NUM_VERTICES; counter++ )
    {
        Vertex v = findNewVertexOfIndegreeZero( );
        if( v == null )
            throw new CycleFoundException( );
        v.topNum = counter;
        for each Vertex w adjacent to v
            w.indegree--;
    }
}
```

Problème?

Algorithme

```
void topsort( ) throws CycleFoundException
{
    for( int counter = 0; counter < NUM_VERTICES; counter++ )
    {
        Vertex v = findNewVertexOfIndegreeZero( );
        if( v == null )
            throw new CycleFoundException( );
        v.topNum = counter;
        for each Vertex w adjacent to v
            w.indegree--;
    }
}
```

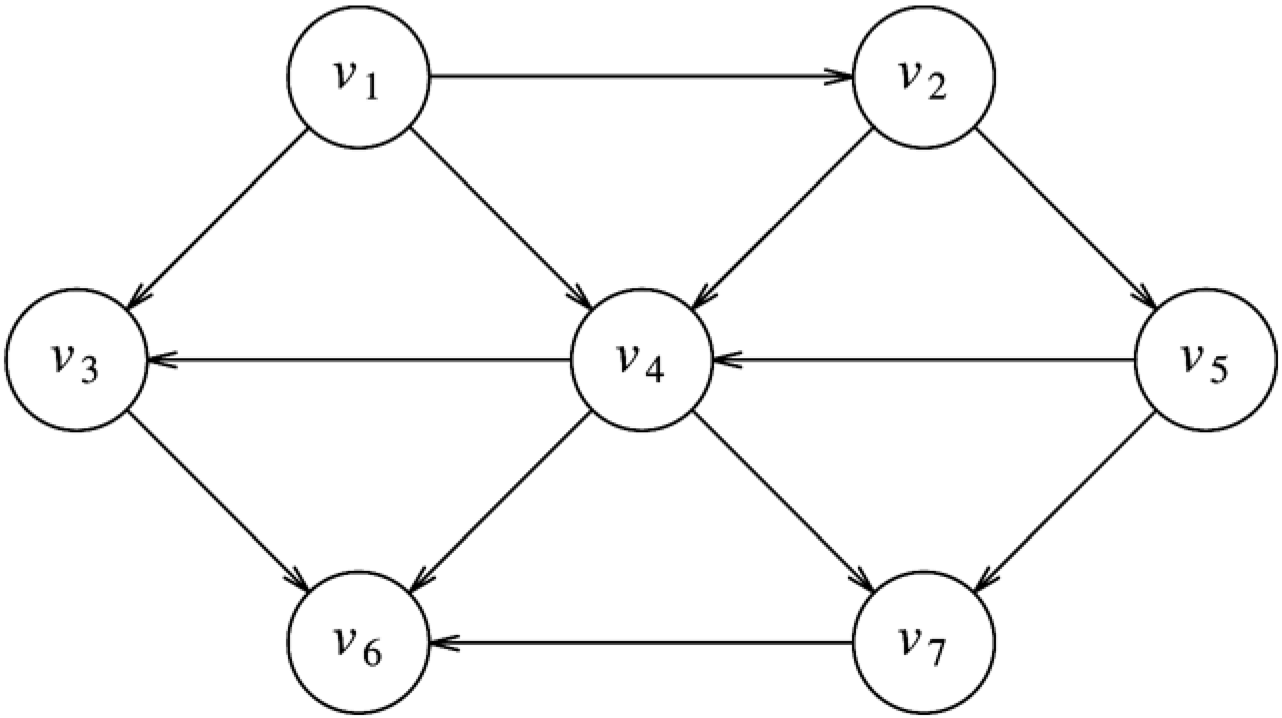
Complexité: $O(|V|^2)$

Algorithme amélioré (?)

```
void topsort( ) throws CycleFoundException
{
    Queue<Vertex> q = new Queue<Vertex>( );
    int counter = 0;

    for each Vertex v
        if( v.indegree == 0 )
            q.enqueue( v );

    while( !q.isEmpty( ) )
    {
        Vertex v = q.dequeue( );
        v.topNum = ++counter;  // Assign next number

        for each Vertex w adjacent to v
            if( --w.indegree == 0 )
                q.enqueue( w );
    }

    if( counter != NUM_VERTICES )
        throw new CycleFoundException( );
}
```

- Complexité: O(|E| + |V|)
- Algorithme avec une file (liste de travail)

Exemple



Simulation

	Indegree Before Dequeue #						
Vertex	1	2	3	4	5	6	7
v_1	0	0	0	0	0	0	0
v_2	1	0	0	0	0	0	0
v_3	2	1	1	1	0	0	0
v_4	3	2	1	0	0	0	0
v_5	1	1	0	0	0	0	0
v_6	3	3	3	3	2	1	0
v_7	2	2	2	1	0	0	0
Enqueue	v_1	v_2	v_5	v_4	v_3, v_7		v_6
Dequeue	v_1	v_2	v_5	v_4	v_3	v_7	v_6

Graphes

1. Définitions et exemples
2. Implémentations
3. Ordre topologique
4. Chemin le plus court
5. Dijkstra
6. Parcours

Plus court chemin sans poids

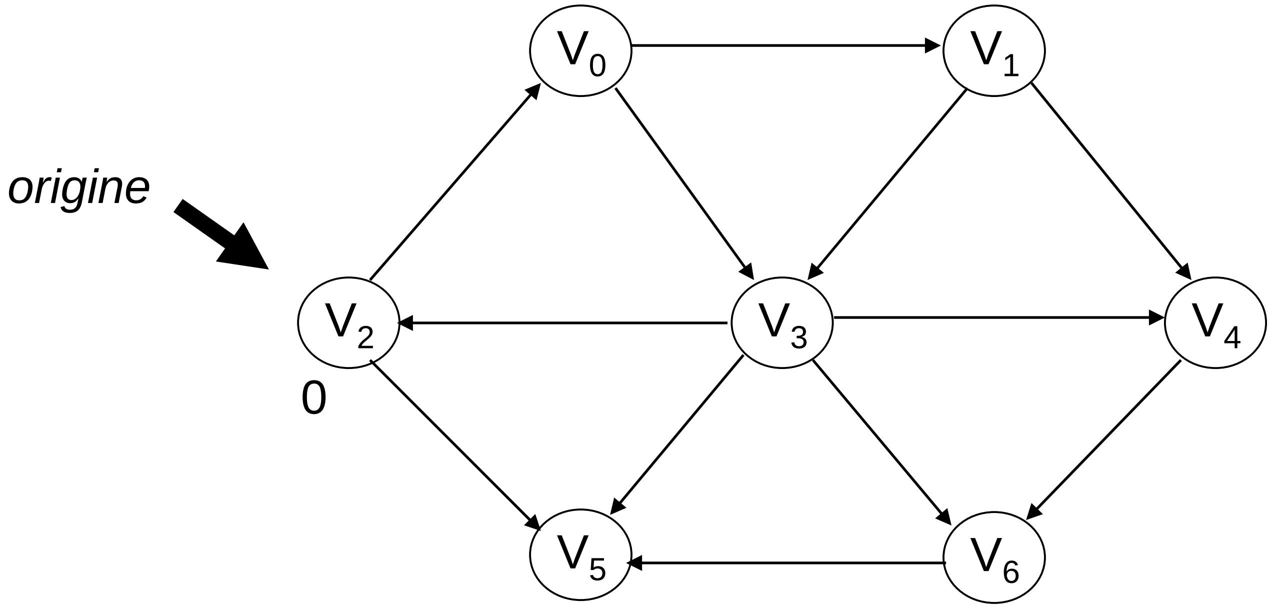
- Graphe oriente
- Nœud de départ
- Coût associé aux arêtes
  - Longueur du chemin

Algorithme

```
void unweighted( Vertex s )
{
    for each Vertex v
    {
        v.dist = INFINITY;
        v.known = false;
    }

    s.dist = 0;

    for( int currDist = 0; currDist < NUM_VERTICES; currDist++ )
        for each Vertex v
            if( !v.known && v.dist == currDist )
            {
                v.known = true;
                for each Vertex w adjacent to v
                    if( w.dist == INFINITY )
                    {
                        w.dist = currDist + 1;
                        w.path = v;
                    }
            }
}
```

- Complexité: $O(|V|^2)$

Exemple



Simulation

Nœuds	Distance	Connu?	Parent
V_0	∞	Faux	-
V_1	∞	Faux	-
V_2	∞	Faux	-
V_3	∞	Faux	-
V_4	∞	Faux	-
V_5	∞	Faux	-
V_6	∞	Faux	-

Nœuds	Distance	Connu?	Parent
V_0	*1*	*Faux*	*V_2*
V_1	∞	Faux	-
V_2	**0**	**Vrai**	**-**
V_3	∞	Faux	-
V_4	∞	Faux	-
V_5	*1*	*Faux*	*V_2*
V_6	∞	Faux	-

Nœuds	Distance	Connu?	Parent
V_0	**1**	**Vrai**	**V_2**
V_1	*2*	*Faux*	*V_0*
V_2	0	Vrai	-
V_3	*2*	*Faux*	*V_0*
V_4	∞	Faux	-
V_5	1	Faux	V_2
V_6	∞	Faux	-

Nœuds	Distance	Connu?	Parent
V_0	1	Vrai	V_2
V_1	2	Faux	V_0
V_2	0	Vrai	-
V_3	2	Faux	V_0
V_4	∞	Faux	-
V_5	**1**	**Vrai**	**V_2**
V_6	∞	Faux	-

Simulation

Nœuds	Distance	Connu?	Parent
V_0	1	Vrai	V_2
V_1	**2**	**Vrai**	**V_0**
V_2	0	Vrai	-
V_3	2	Faux	V_0
V_4	*3*	*Faux*	*V_1*
V_5	1	Vrai	V_2
V_6	∞	Faux	-

→

Nœuds	Distance	Connu?	Parent
V_0	1	Vrai	V_2
V_1	2	Vrai	V_0
V_2	0	Vrai	-
V_3	**2**	**Vrai**	**V_0**
V_4	3	Faux	V_1
V_5	1	Vrai	V_2
V_6	*3*	*Faux*	*V_3*

Nœuds	Distance	Connu?	Parent
V_0	1	Vrai	V_2
V_1	2	Vrai	V_0
V_2	0	Vrai	-
V_3	2	Vrai	V_0
V_4	**3**	**Vrai**	**V_1**
V_5	1	Vrai	V_2
V_6	3	Faux	V_3

→

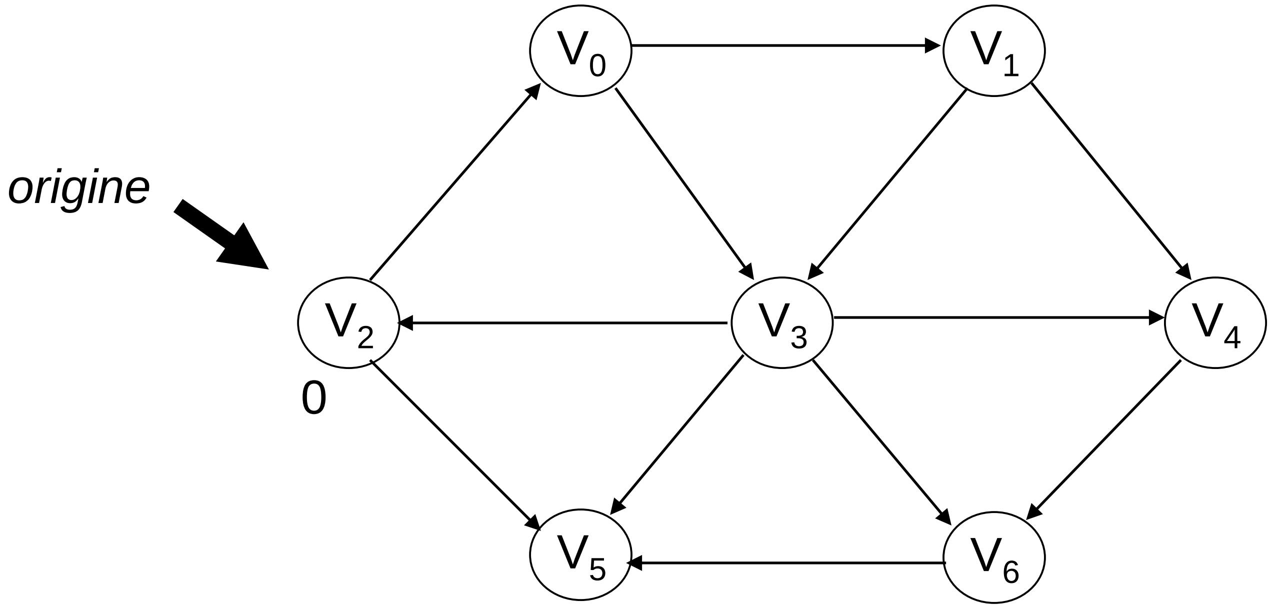
Nœuds	Distance	Connu?	Parent
V_0	1	Vrai	V_2
V_1	2	Vrai	V_0
V_2	0	Vrai	-
V_3	2	Vrai	V_0
V_4	3	Vrai	V_1
V_5	1	Vrai	V_2
V_6	**3**	**Vrai**	**V_3**

Algorithme amélioré (?)

```
void unweighted( Vertex s ) {
    Queue<Vertex> q = new Queue<Vertex>( );
    for each Vertex v
        v.dist = INFINITY;
    s.dist = 0;
    q.enqueue( s );
    while( !q.isEmpty( ) ) {
        Vertex v = q.dequeue( );
        for each Vertex w adjacent to v
            if( w.dist == INFINITY ) {
                w.dist = v.dist + 1;
                w.path = v;
                q.enqueue( w );
            }
    }
}
```

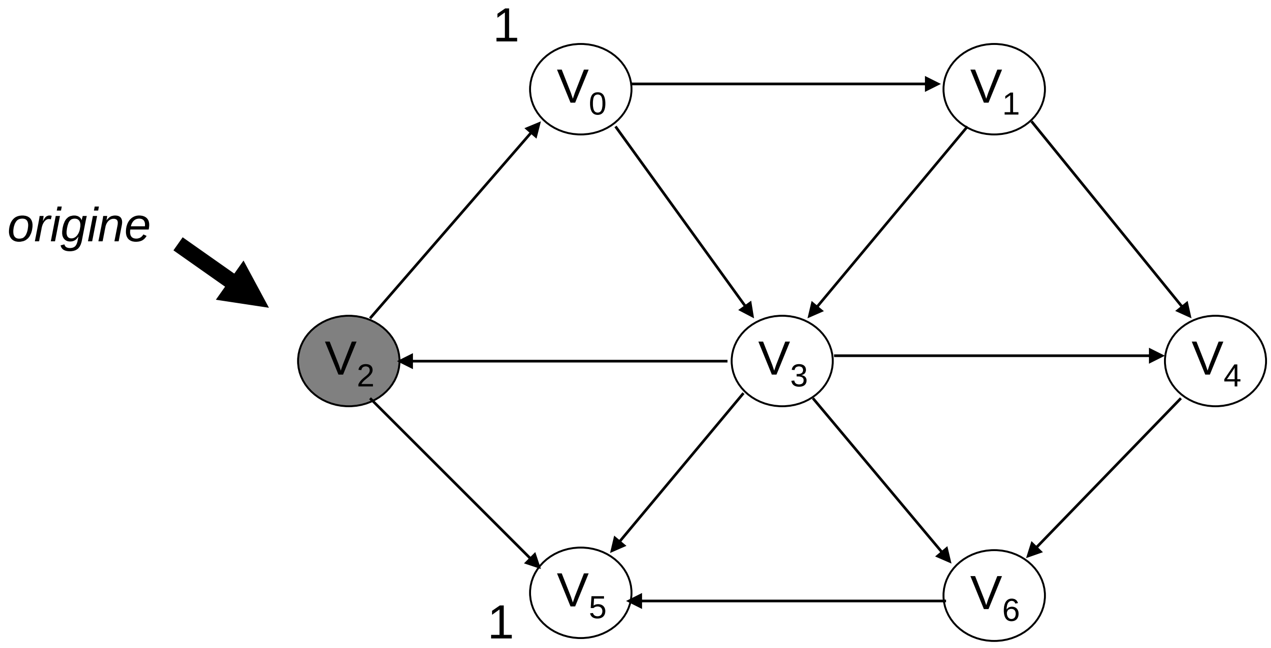
- Complexité: O(|E| + |V|)

Exemple avec file



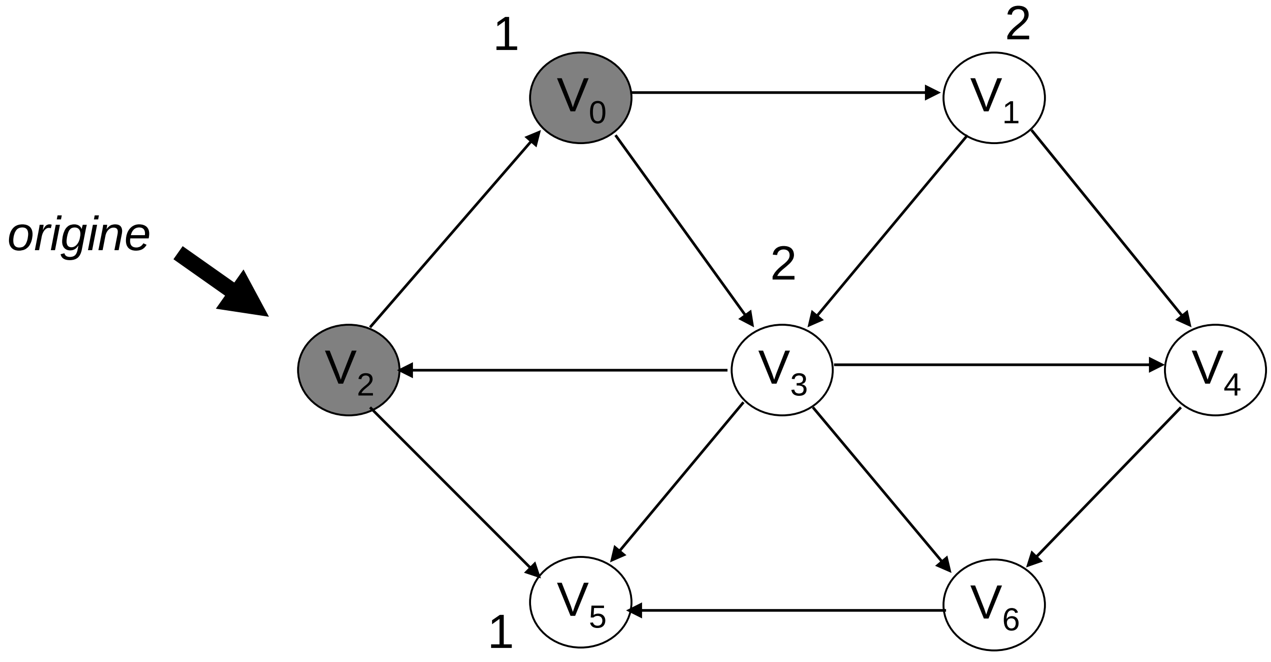
File: V_2

Exemple avec file



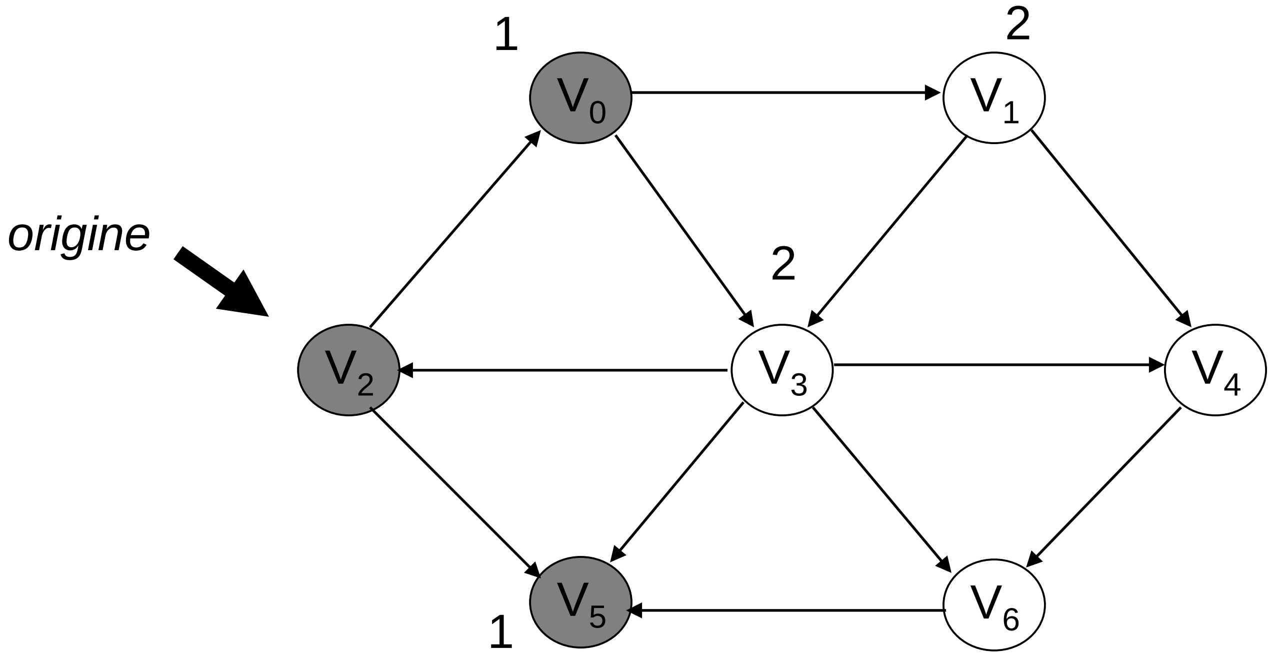
File: V_0 V_5

Exemple avec file



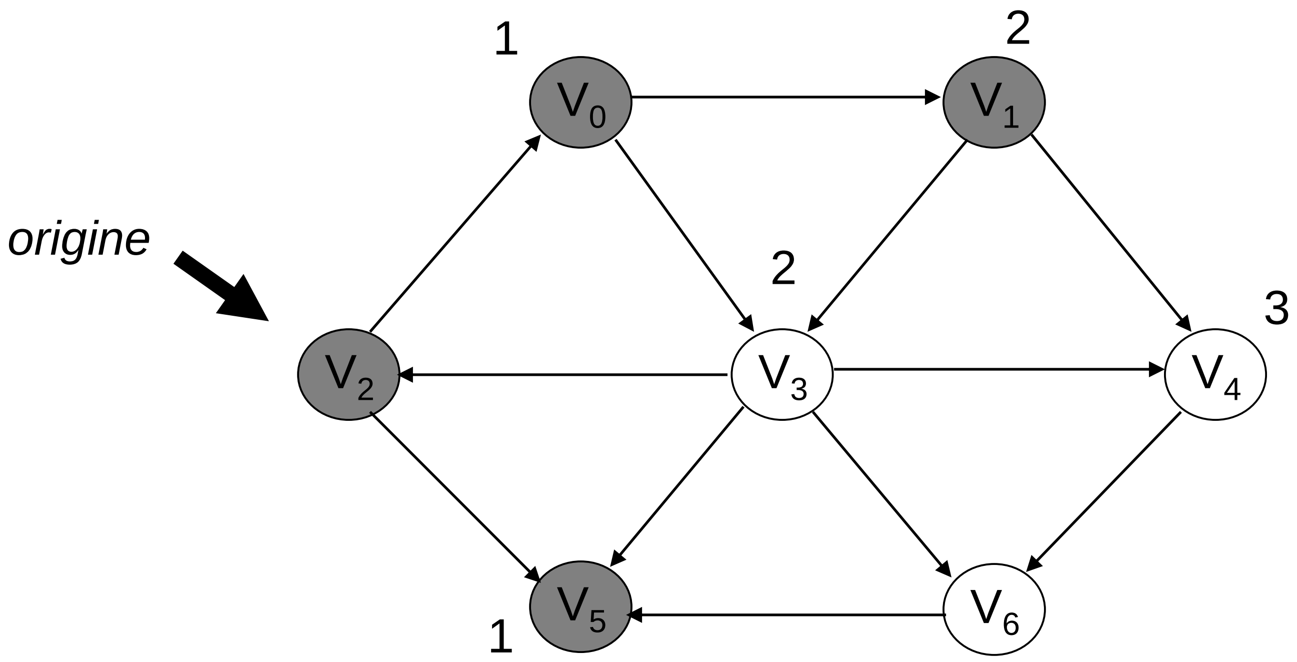
File: V_5 V_1 V_3

Exemple avec file



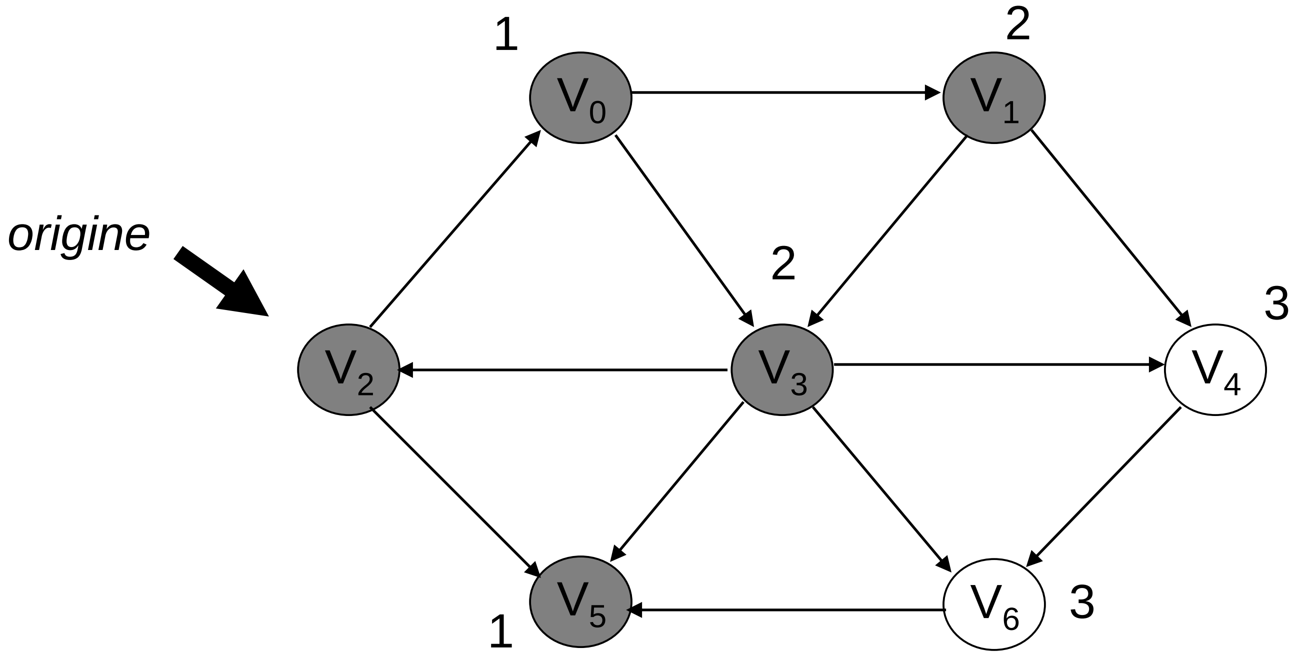
File: V_1 V_3

Exemple avec file



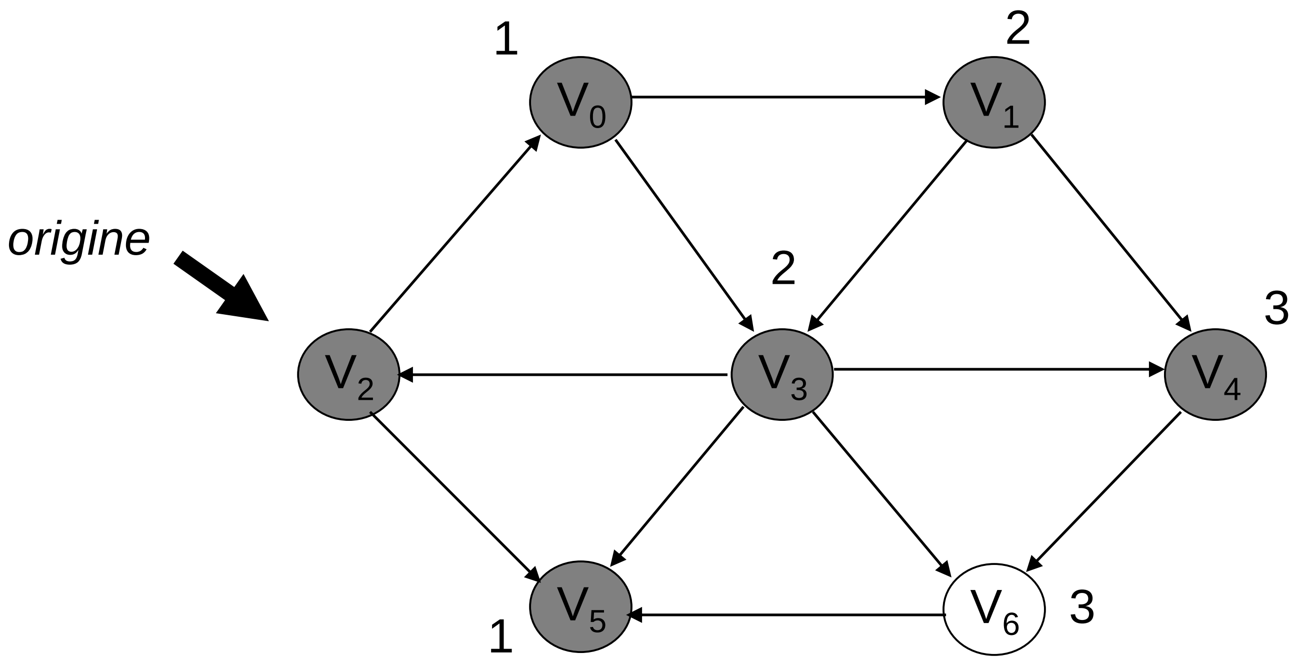
File: V_3 V_4

Exemple avec file



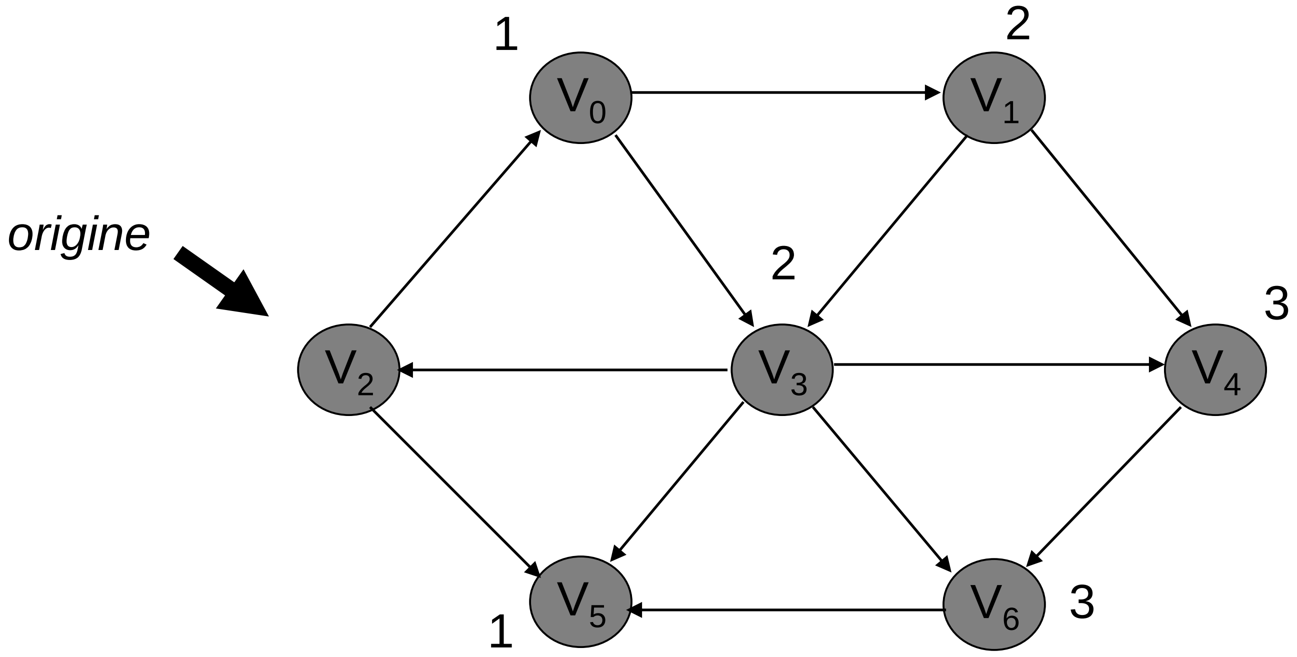
File: V_4 V_6

Exemple avec file



File: V_6

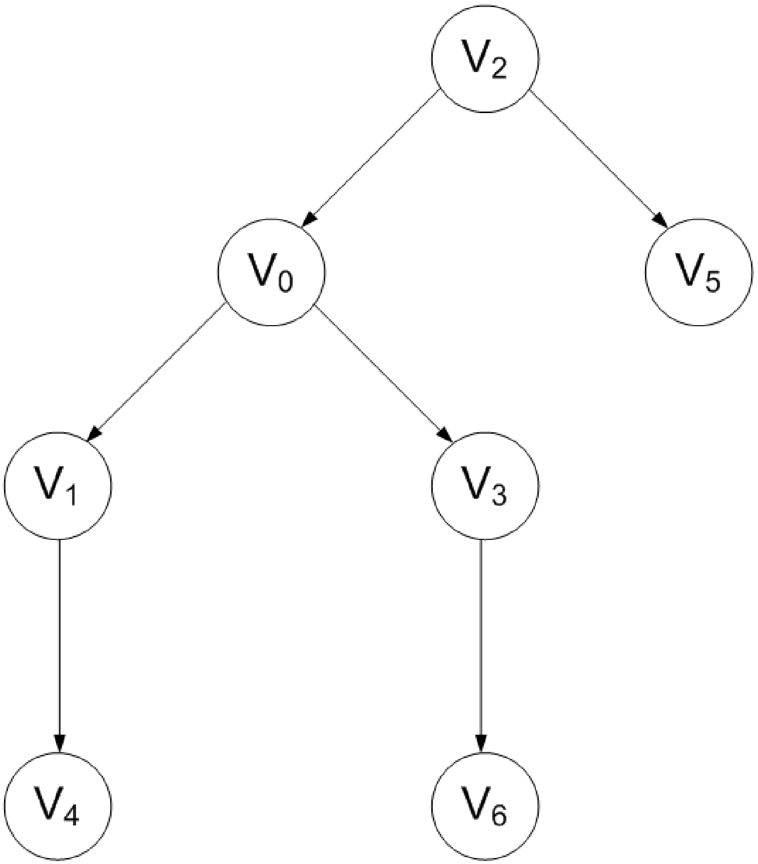
Exemple avec file



File: vide

Arbre équivalent

(parcours par niveaux)



Graphes

1. Définitions et exemples
2. Implémentations
3. Ordre topologique
4. Chemin le plus court
5. Dijkstra
6. Parcours

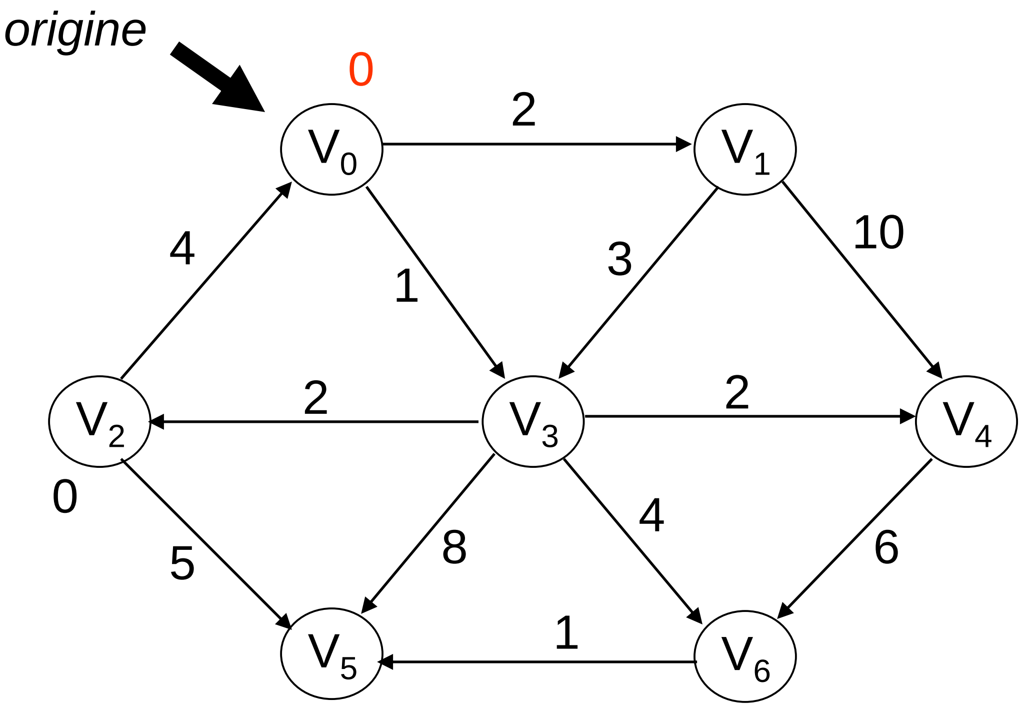
Plus court chemin avec poids

- Graphe oriente
- Nœud de départ
- Coût associé aux arêtes
  - Poids (non négatif)

```
void dijkstra( Vertex s )
{
    for each Vertex v
    {
        v.dist = INFINITY;
        v.known = false;
    }

    s.dist = 0;

    for( ; ; )
    {
        Vertex v = smallest unknown distance vertex;
        if( v == NOT_A_VERTEX )
            break;
        v.known = true;

        for each Vertex w adjacent to v
            if( !w.known )
                if( v.dist + cvw < w.dist )
                {
                    // Update w
                    decrease( w.dist to v.dist + cvw
                    w.path = v;
                }
    }
}
```

- Complexité: $O(|V|^2)$

Exemple



Simulation

Nœud	Distance V_0	Distance V_1	Distance V_2	Distance V_3	Distance V_4	Distance V_5	Distance V_6
-	0	∞	∞	∞	∞	∞	∞
V_0	<u>0</u>	**2**	∞	**1**	∞	∞	∞
V_3	0	2	**3**	<u>1</u>	**3**	**9**	**5**
V_1	0	<u>2</u>	3	1	3	9	5
V_2	0	2	<u>3</u>	1	3	**8**	5
V_4	0	2	3	1	<u>3</u>	8	5
V_6	0	2	3	1	3	**6**	<u>5</u>

Analyse (Dijkstra)

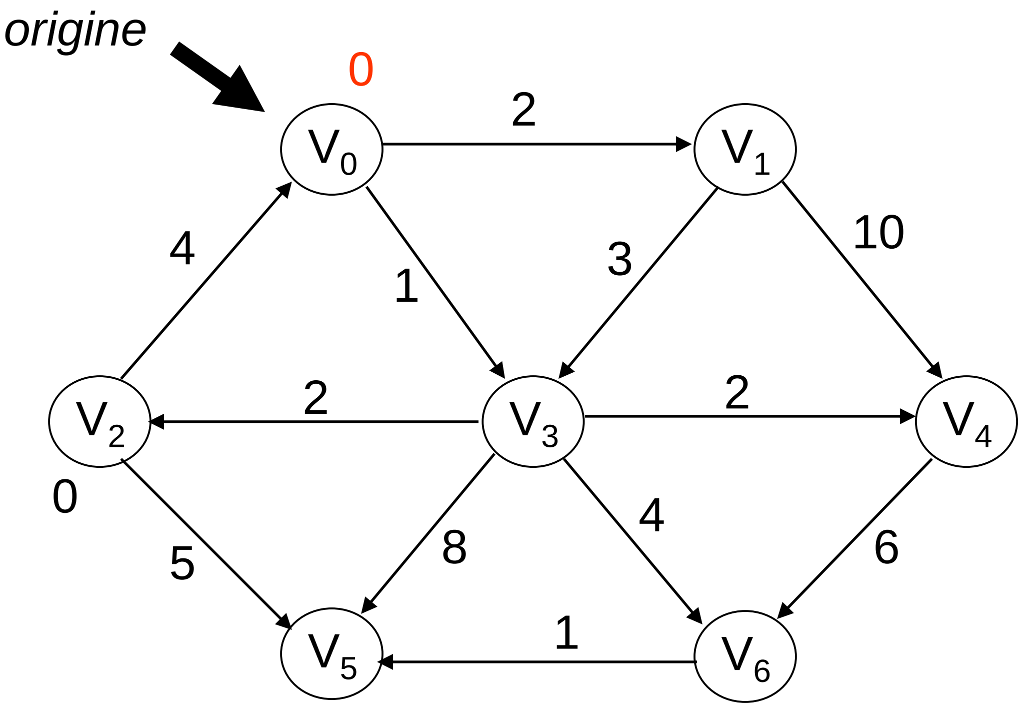
- Recherche séquentielle du minimum nœud traverse $O(|V|^2)$

  – Graphe compacte (« dense »)
    - $|E| = \Theta(|V|^2) \rightarrow O(|E|)$

  – Graphe éparpillé (« sparse »)
    - $|E| = \Theta(|V|)$

Algorithme de Dijkstra

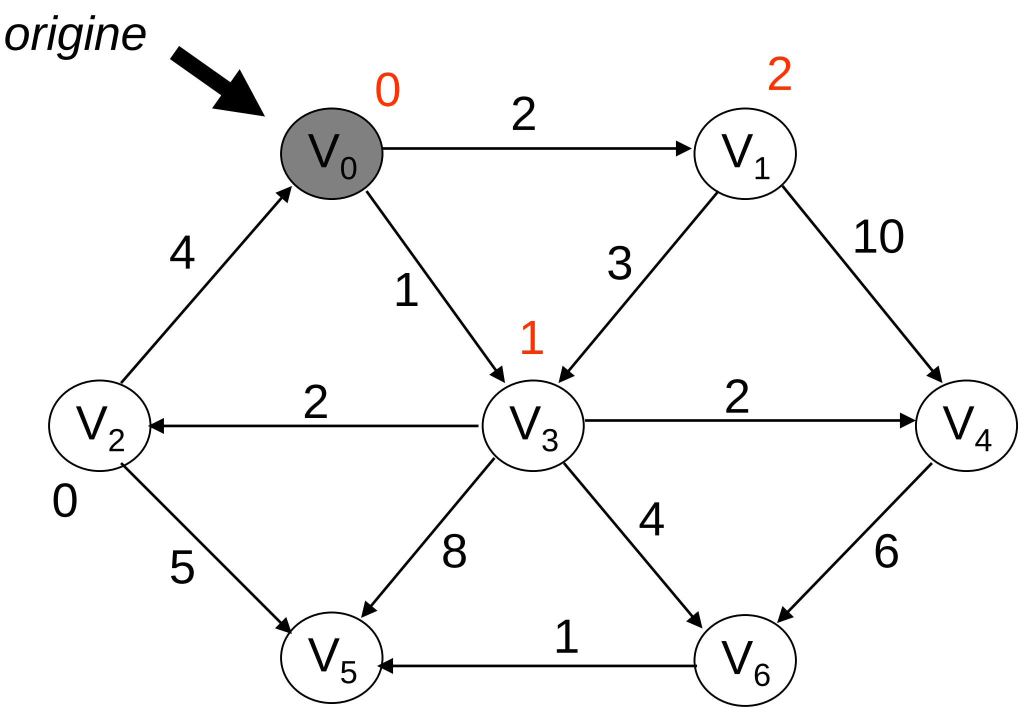
(avec file de priorité)



File de priorité: $(V_0,0)$

Algorithme de Dijkstra

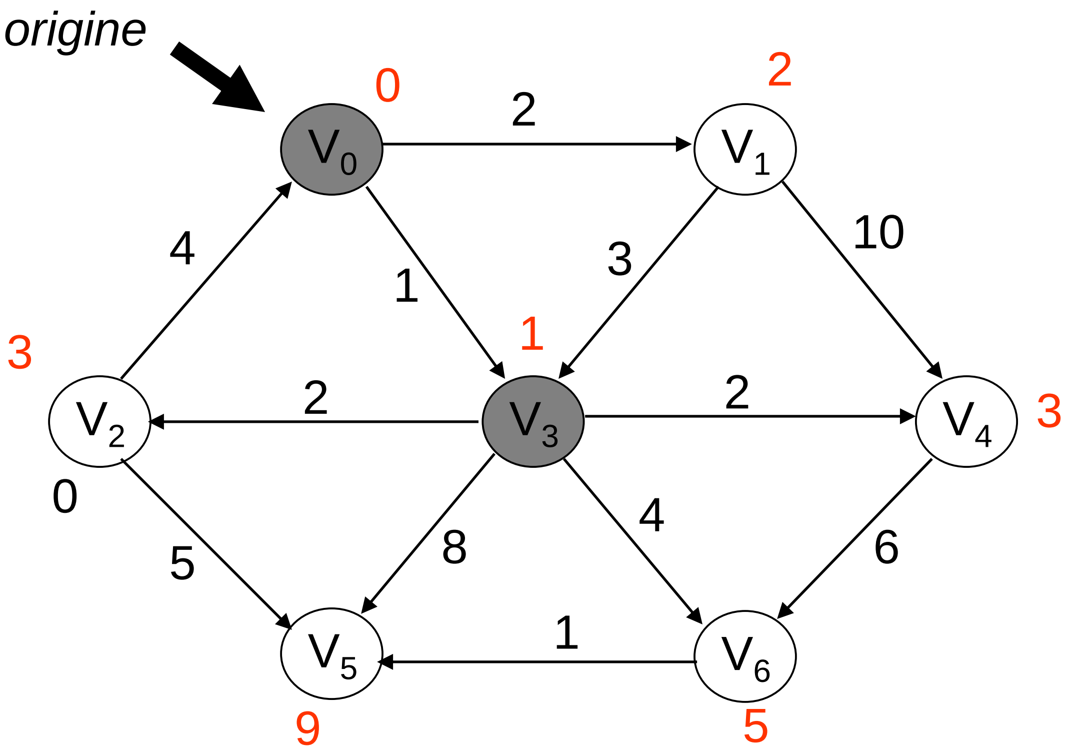
(avec file de priorité)



File de priorité: $(V_3, 1)$ $(V_1, 2)$

Algorithme de Dijkstra

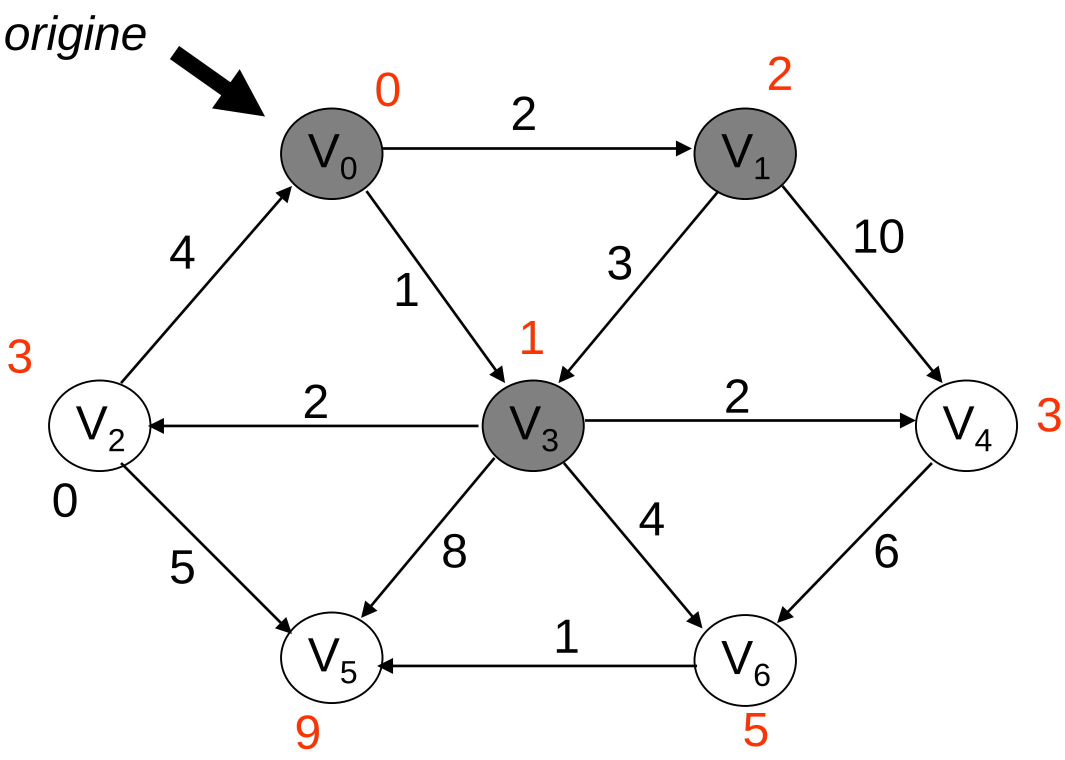
(avec file de priorité)



File de priorité: $(V_1, 2)$ $(V_2, 3)$ $(V_4, 3)$ $(V_6, 5)$ $(V_5, 9)$

Algorithme de Dijkstra

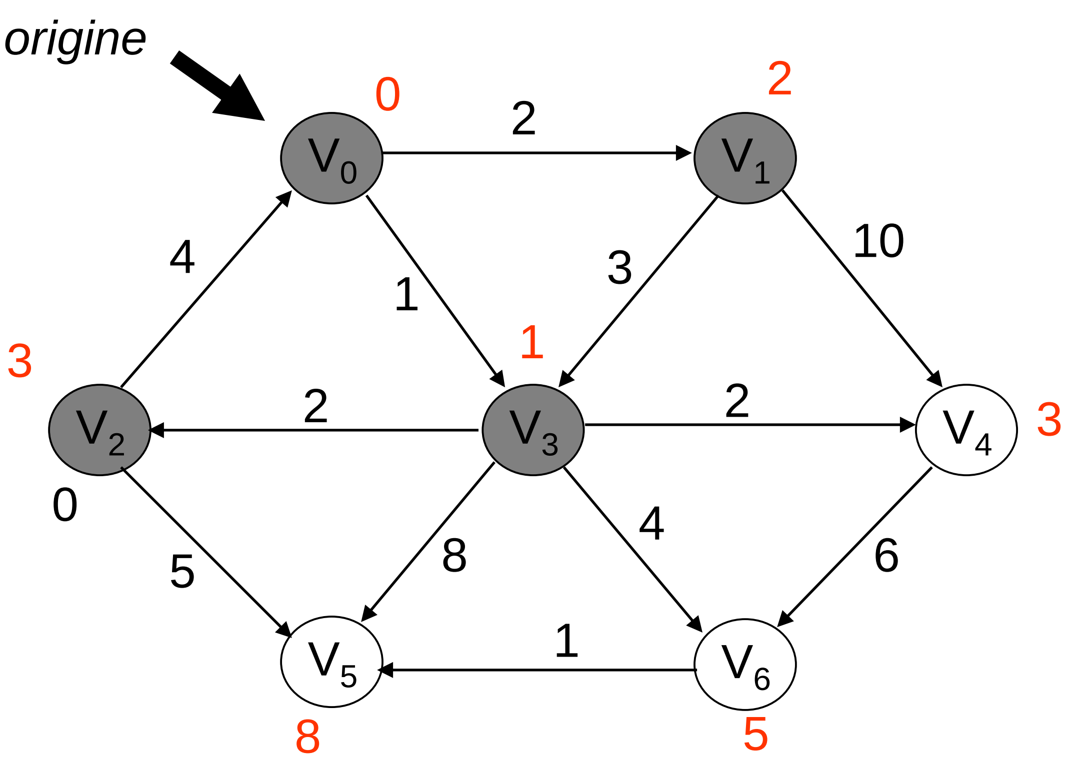
(avec file de priorité)



File de priorité: (V_2 ,3) (V_4 ,3) (V_6 ,5) (V_5 ,9)

Algorithme de Dijkstra

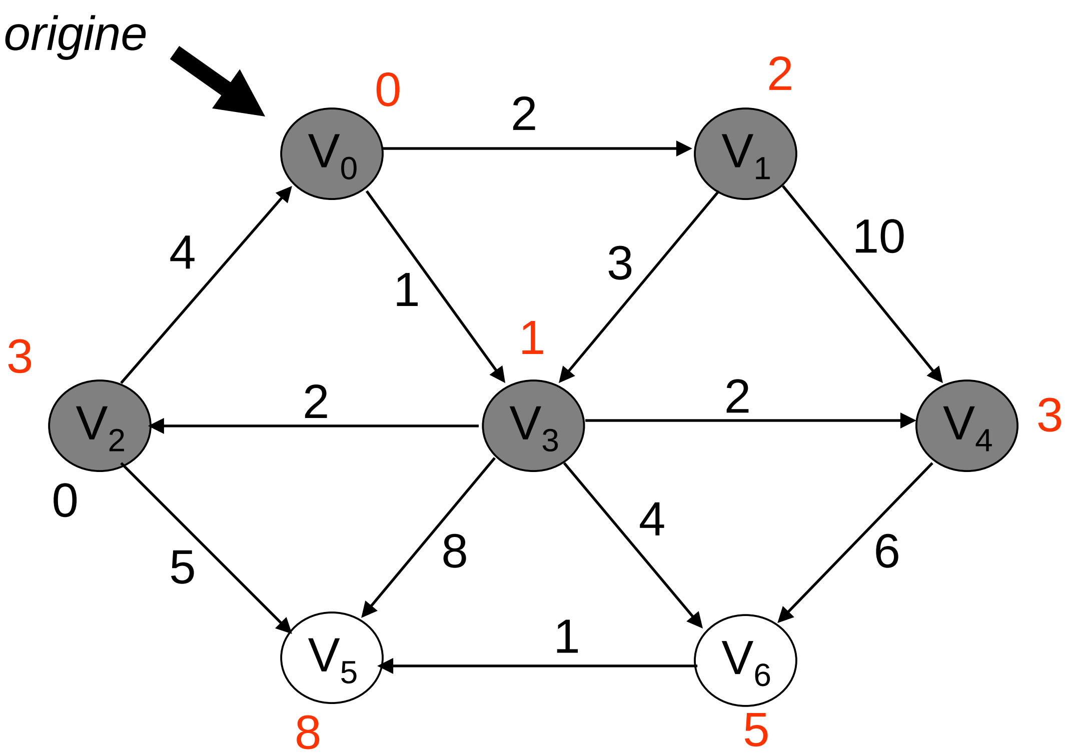
(avec file de priorité)



File de priorité: $(V_4, 3)$ $(V_6, 5)$ $(V_5, 8)$

Algorithme de Dijkstra

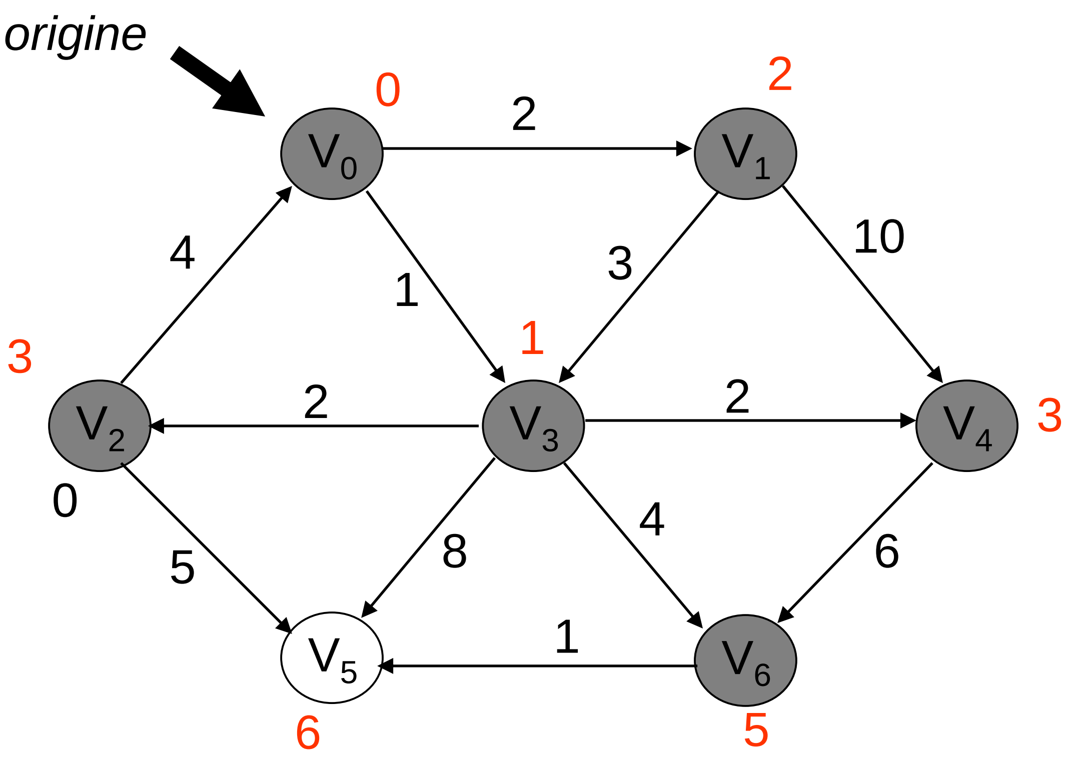
(avec file de priorité)



File de priorité: $(V_6, 5)$ $(V_5, 8)$

Algorithme de Dijkstra

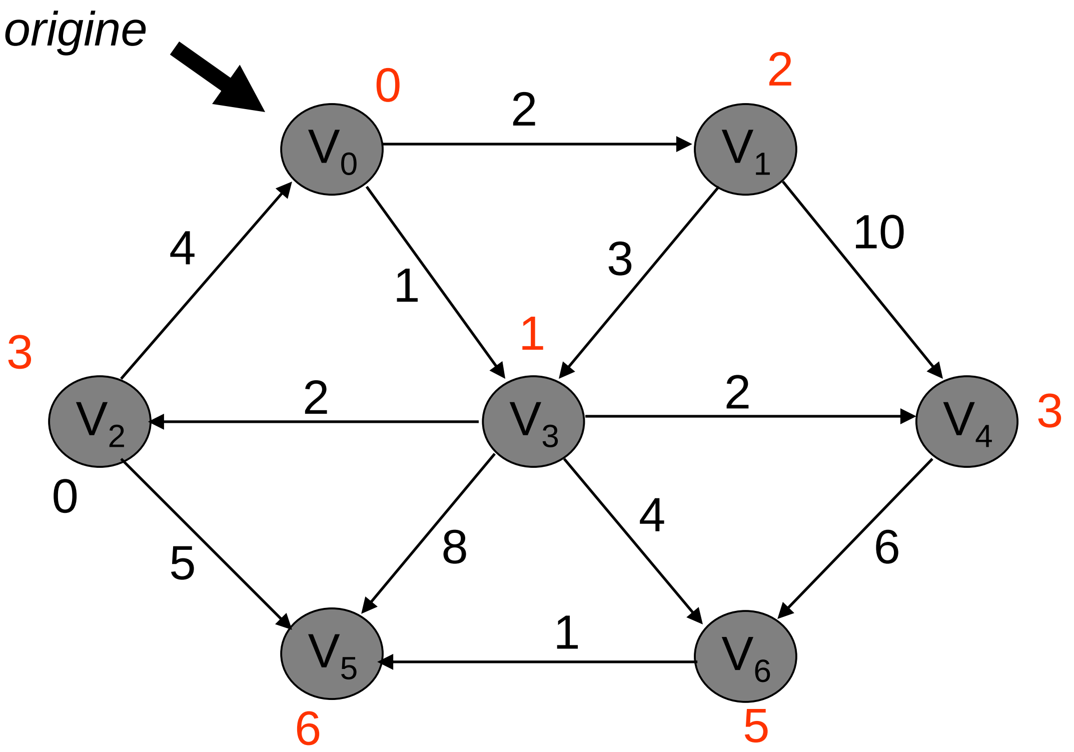
(avec file de priorité)



File de priorité: (V_5 ,8)

Algorithme de Dijkstra

(avec file de priorité)



File de priorité: Vide

Graphes

1. Définitions et exemples
2. Implémentations
3. Ordre topologique
4. Chemin le plus court
5. Dijkstra
6. Parcours

Algorithmes de visite

1. Breadth First Search (équivalent à par niveau)

   Vu au tri topologique

2. Depth First Search (équivalent à pré-ordre)

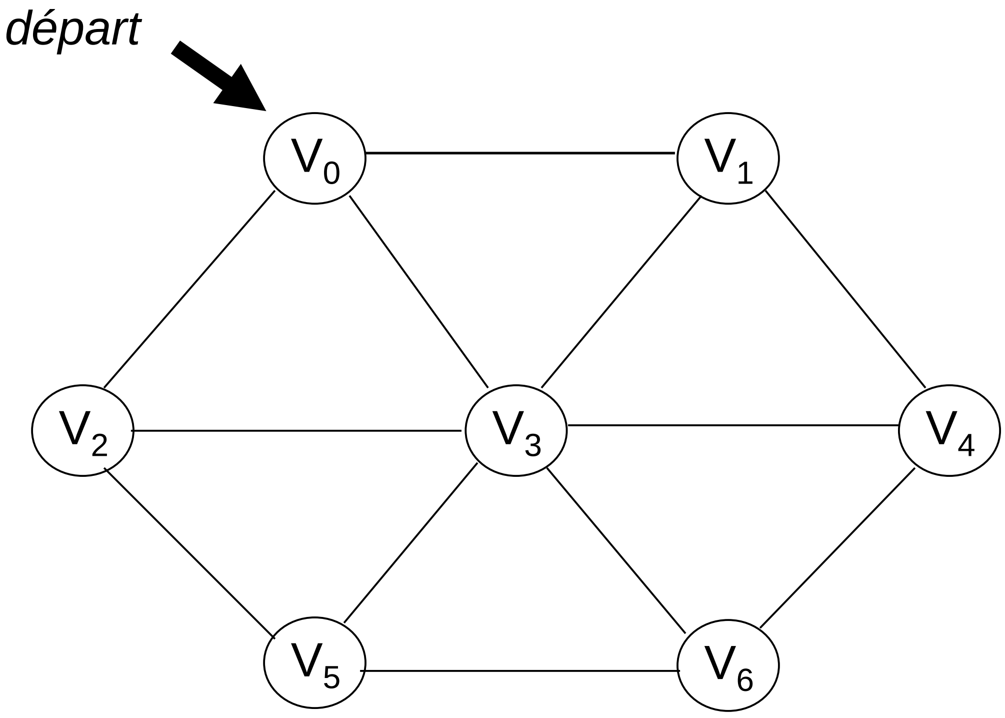
   On part d'un nœud,

   Visite ses enfants

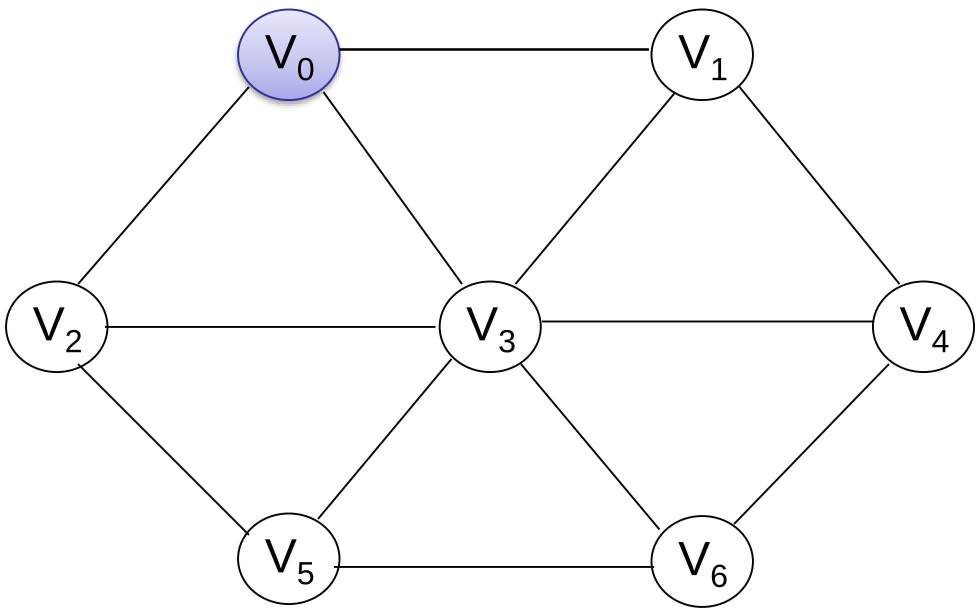
   Pour chacun de ses enfants, on refait la même chose

   Chaque nœud visité est marqué comme tel

Ex. 1 BFS – graphe non orienté

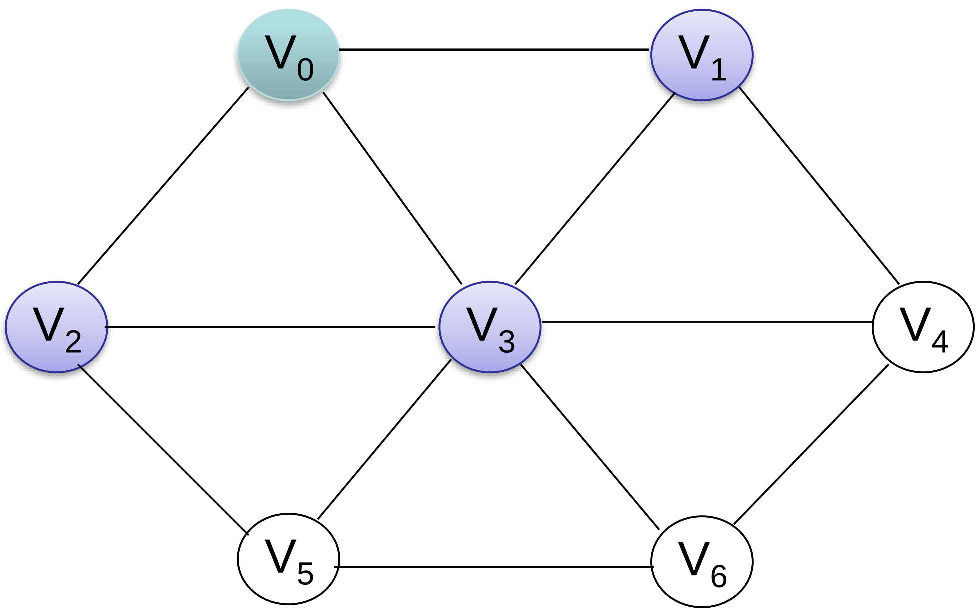


Ex. 1 BFS – graphe non orienté



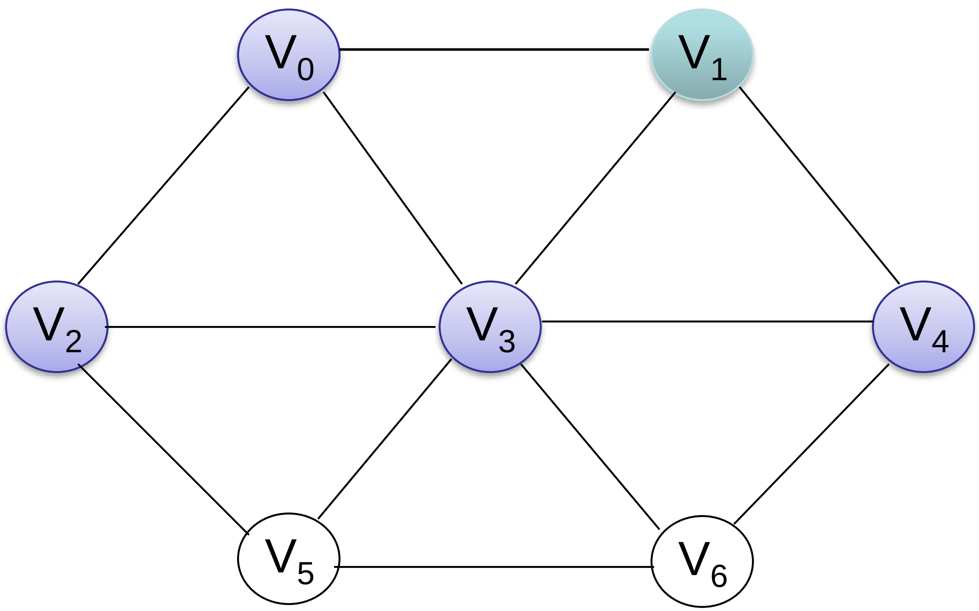
V_0,

Ex. 1 BFS – graphe non orienté



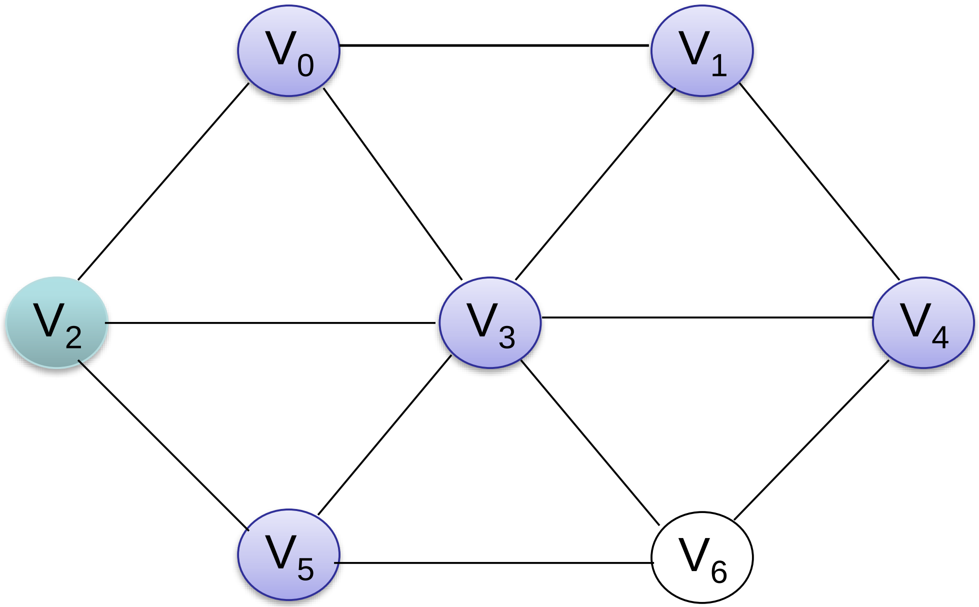
V_0, V_1, V_2, V_3,

Ex. 1 BFS – graphe non orienté



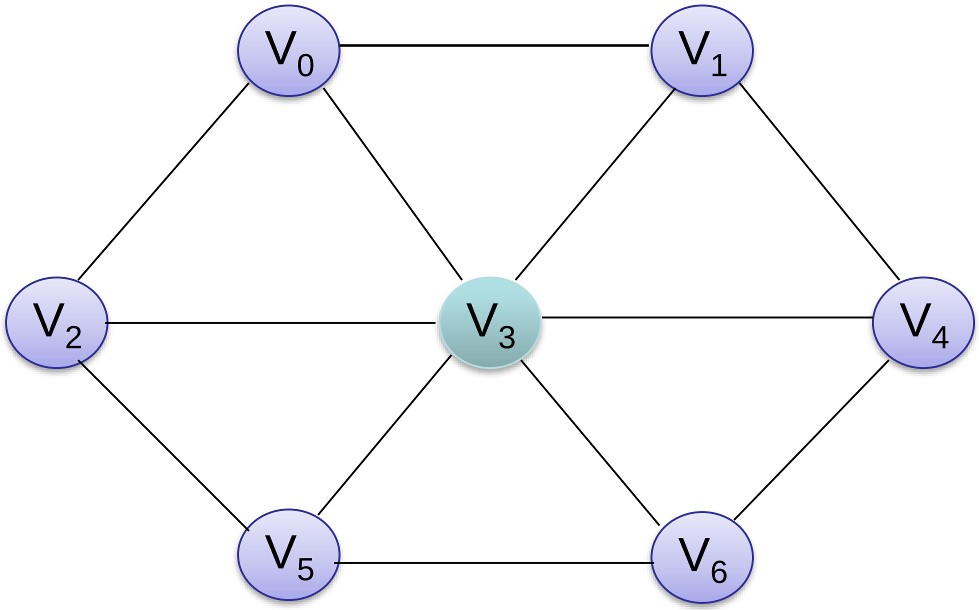
V_0, V_1, V_2, V_3, V_4,

Ex. 1 BFS – graphe non orienté



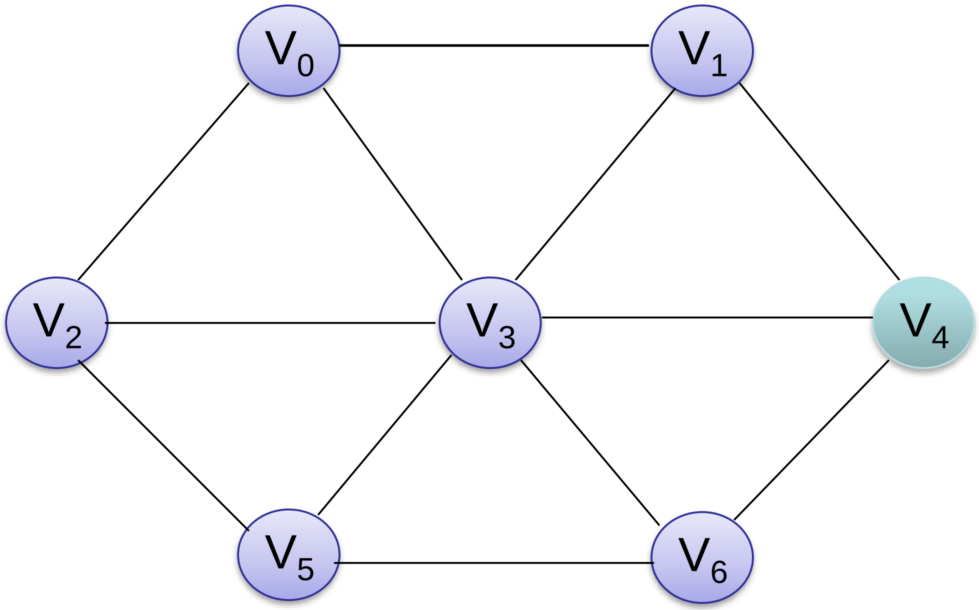
V_0, V_1, V_2, V_3, V_4, V_5,

Ex. 1 BFS – graphe non orienté



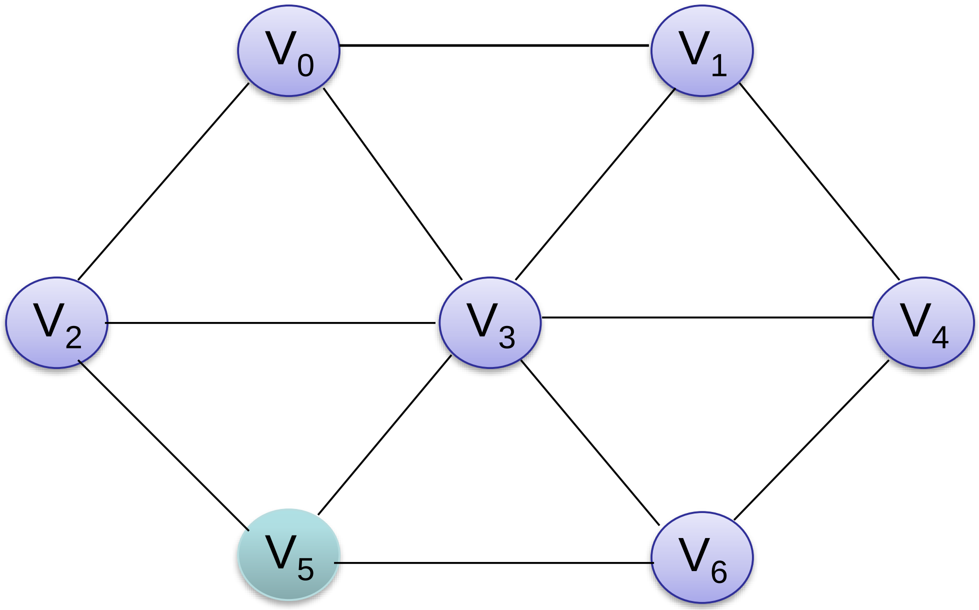
V_0, V_1, V_2, V_3, V_4, V_5, V_6

Ex. 1 BFS – graphe non orienté



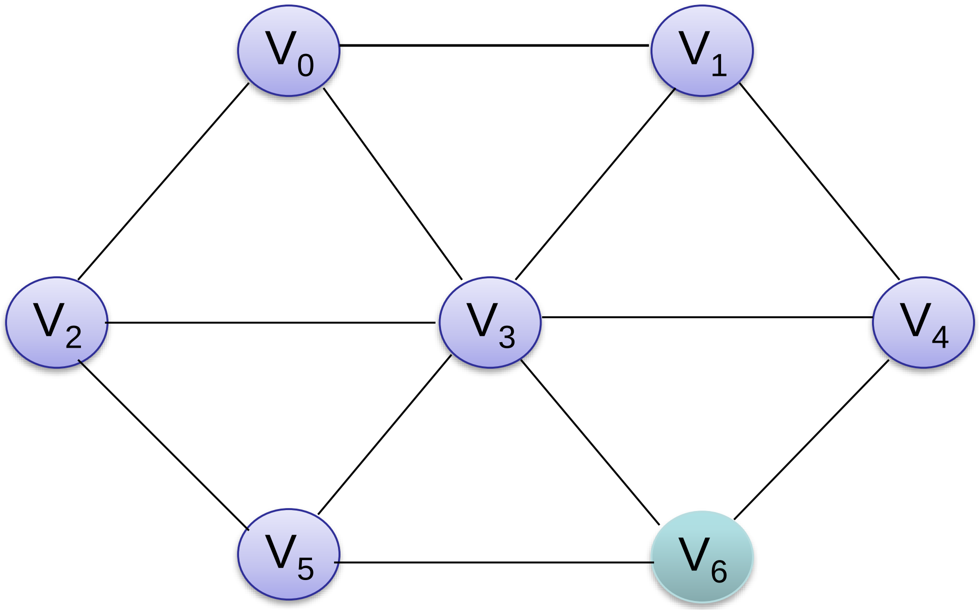
V_0, V_1, V_2, V_3, V_4, V_5, V_6

Ex. 1 BFS – graphe non orienté



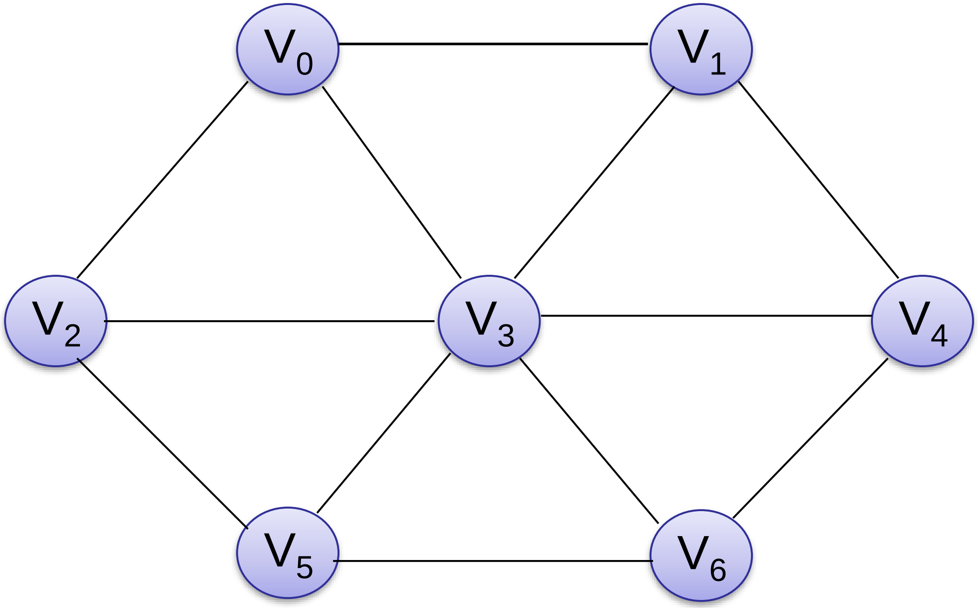
V_0, V_1, V_2, V_3, V_4, V_5, V_6

Ex. 1 BFS – graphe non orienté



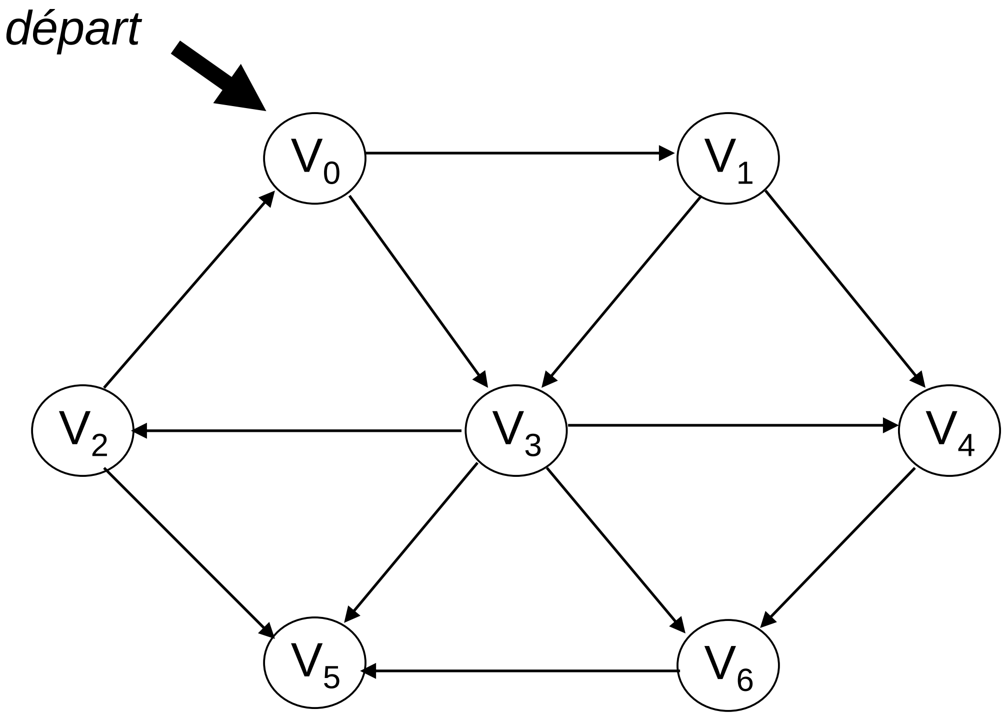
V_0, V_1, V_2, V_3, V_4, V_5, V_6

Ex. 1 BFS – graphe non orienté



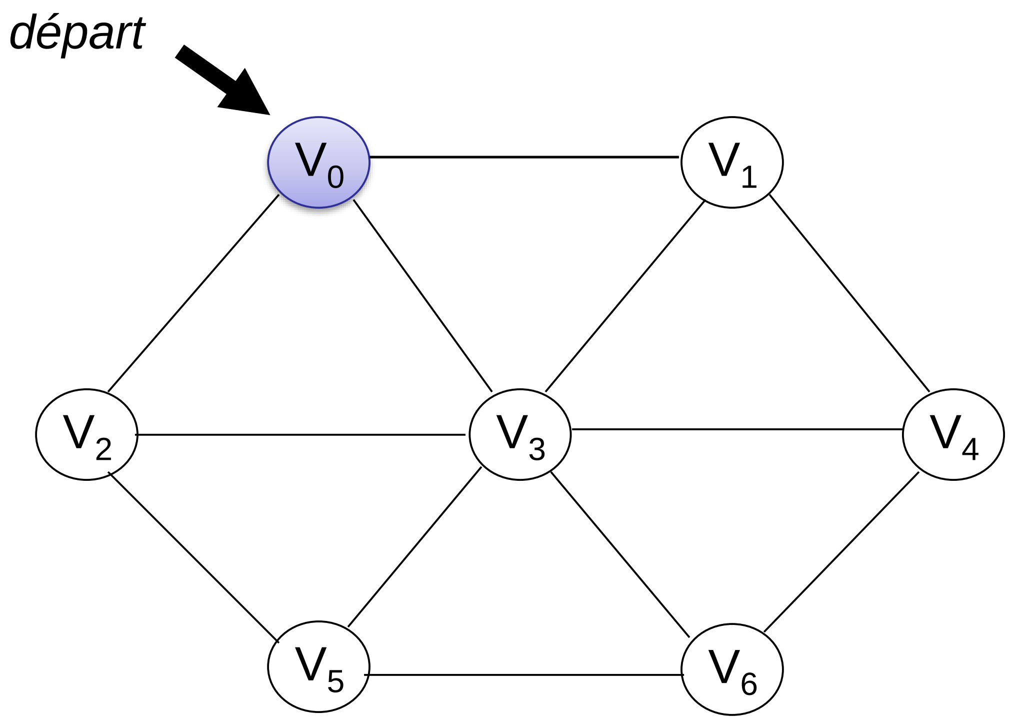
V_0, V_1, V_2, V_3, V_4, V_5, V_6. FIN

Ex. 2 BFS – graphe orienté



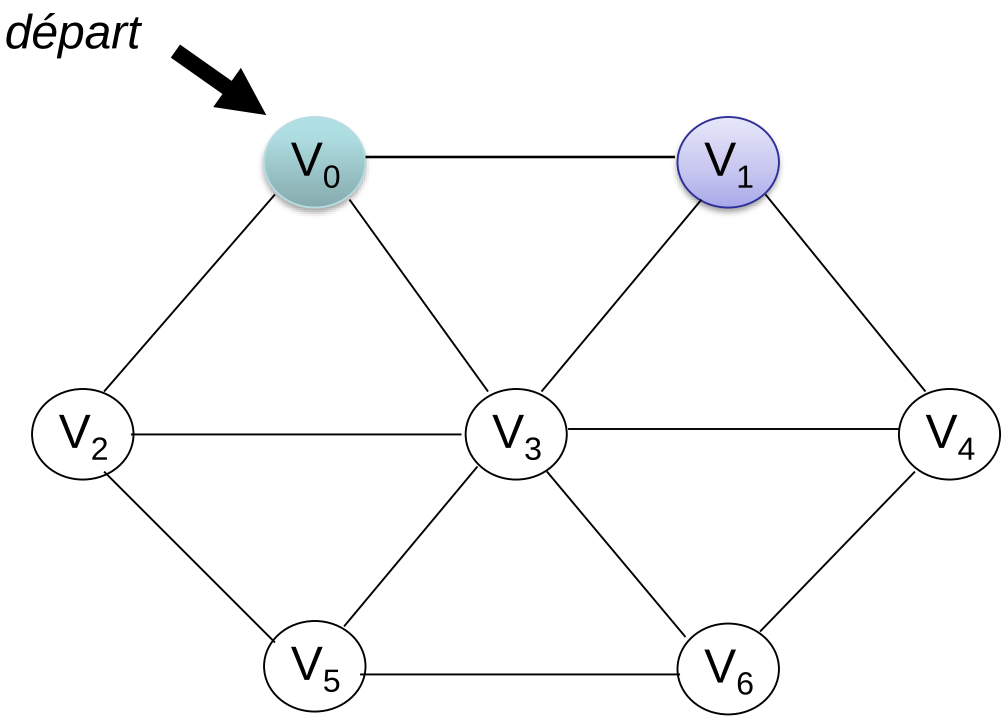
V_0, V_1, V_3, V_4, V_2, V_5, V_6.

Ex. 3 DFS – graphe non orienté



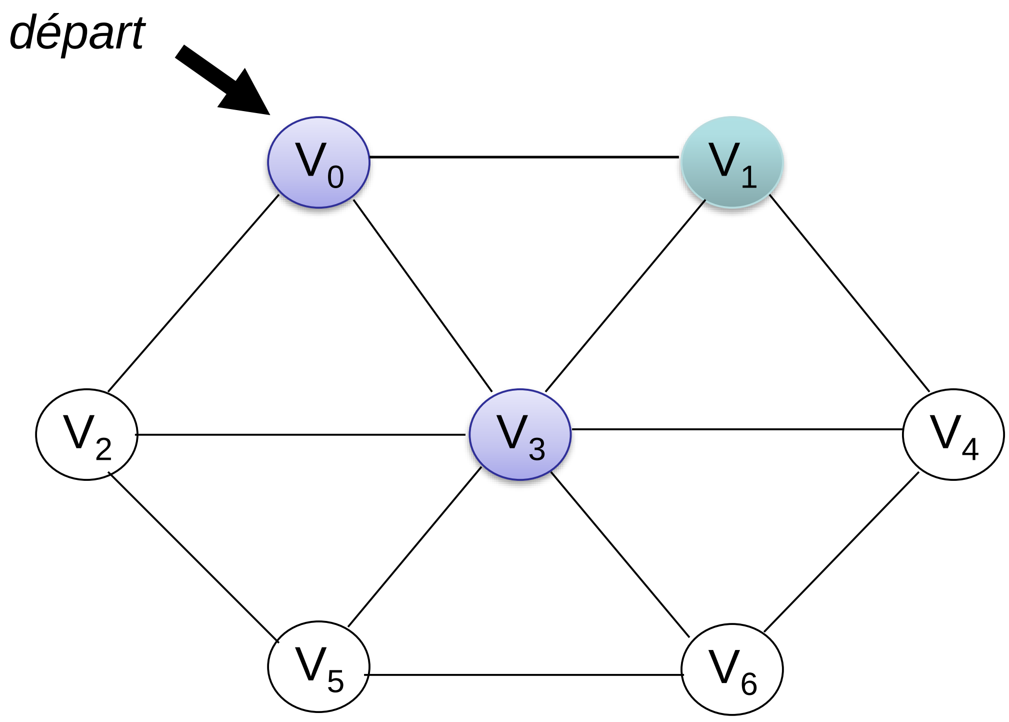
V_0,

Ex. 3 DFS – graphe non orienté



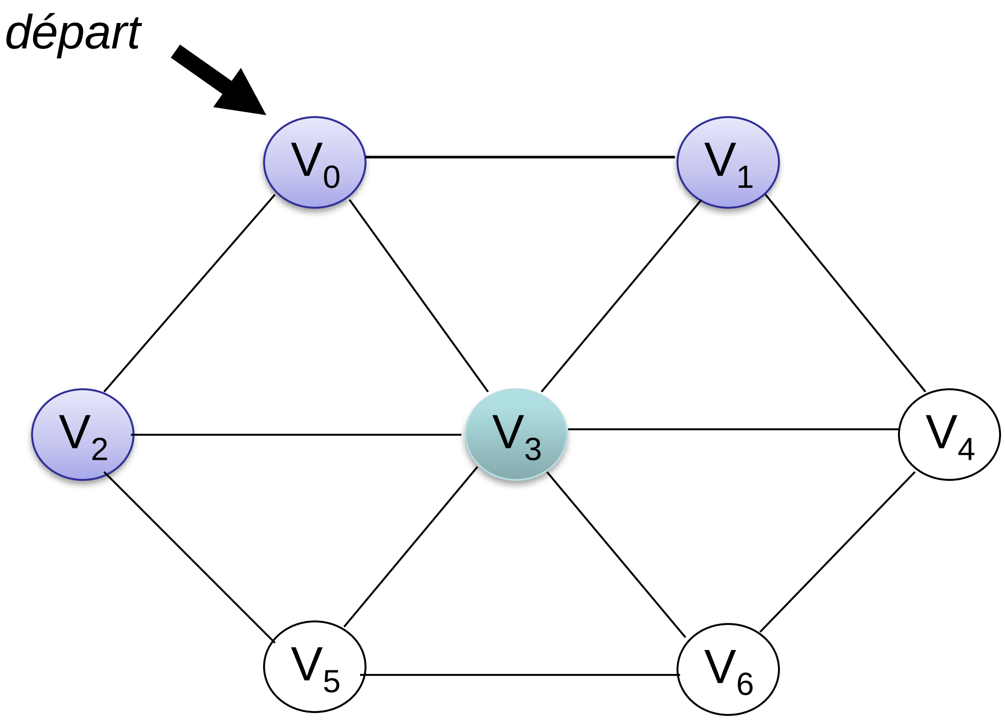
V_0, V_1,

Ex. 3 DFS – graphe non orienté



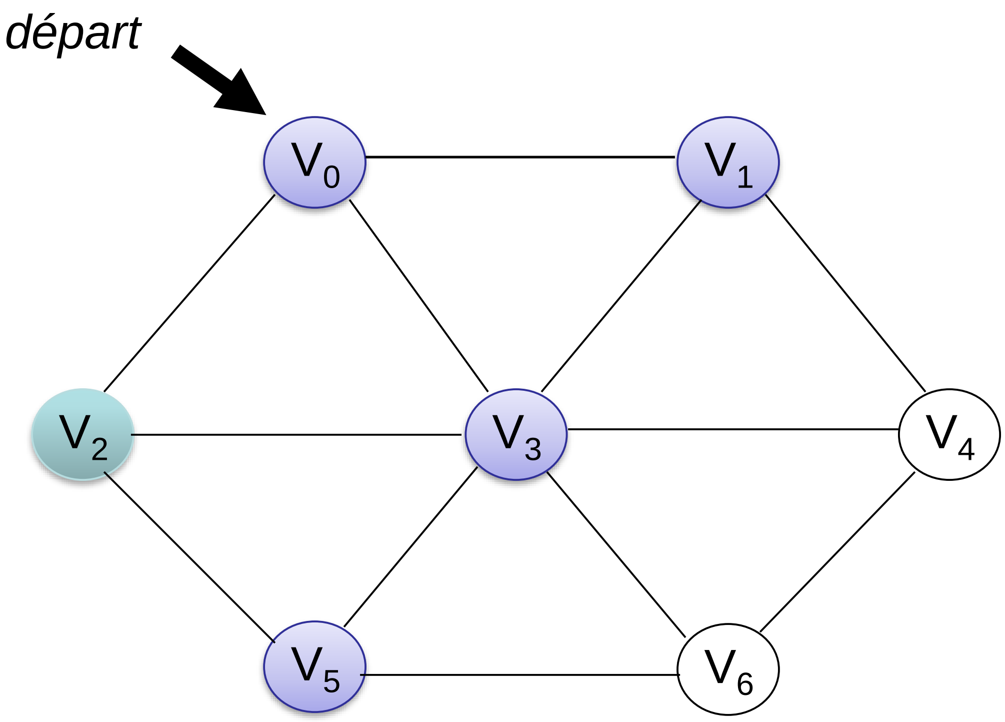
V_0, V_1, V_3,

Ex. 3 DFS – graphe non orienté



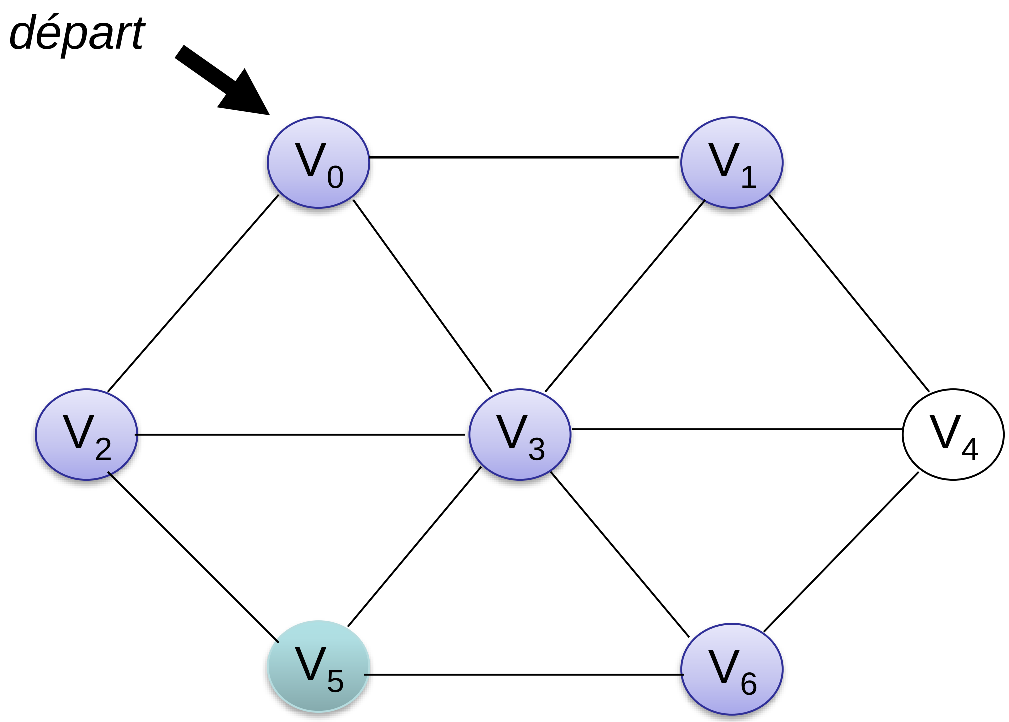
V_0, V_1, V_3, V_2,

Ex. 3 DFS – graphe non orienté



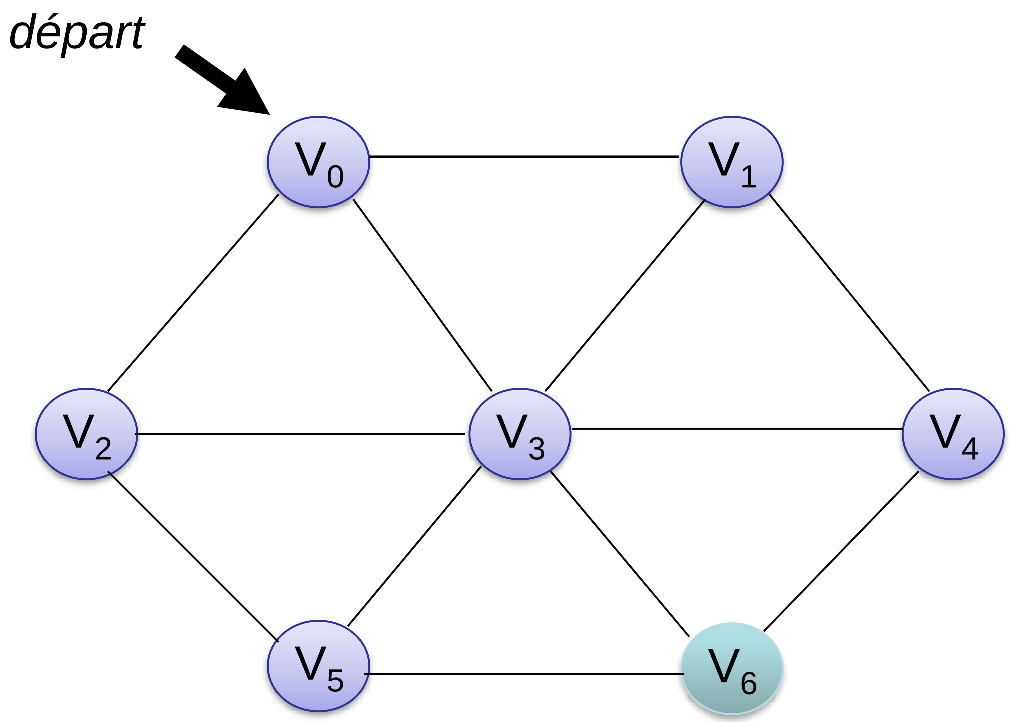
V_0, V_1, V_3, V_2, V_5,

Ex. 3 DFS – graphe non orienté



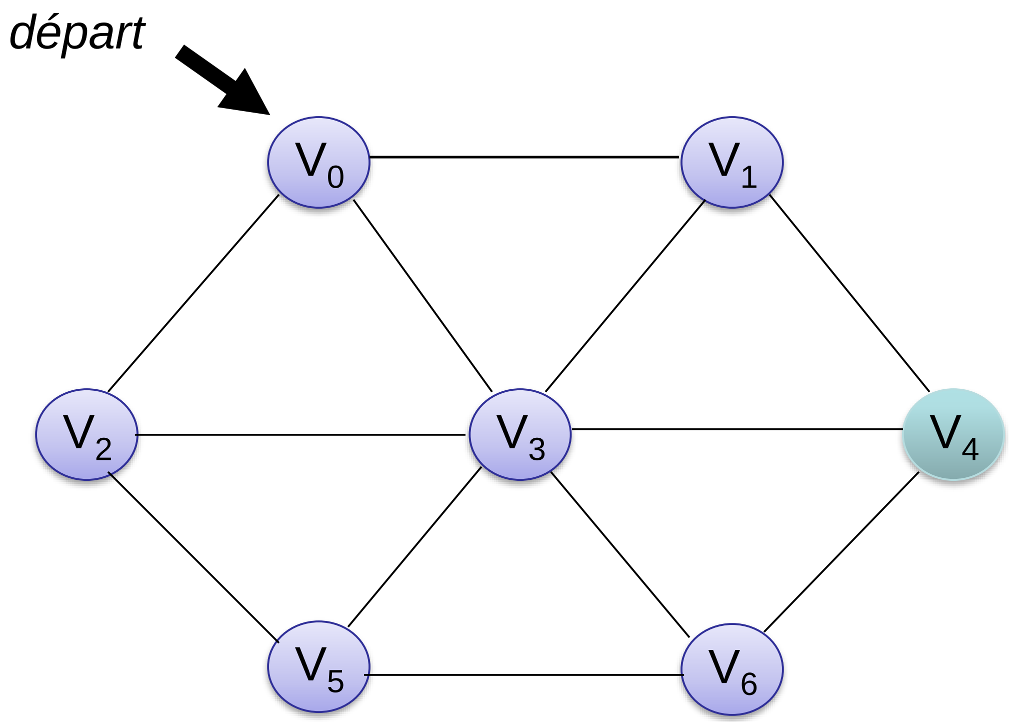
V_0, V_1, V_3, V_2, V_5, V_6,

Ex. 3 DFS – graphe non orienté



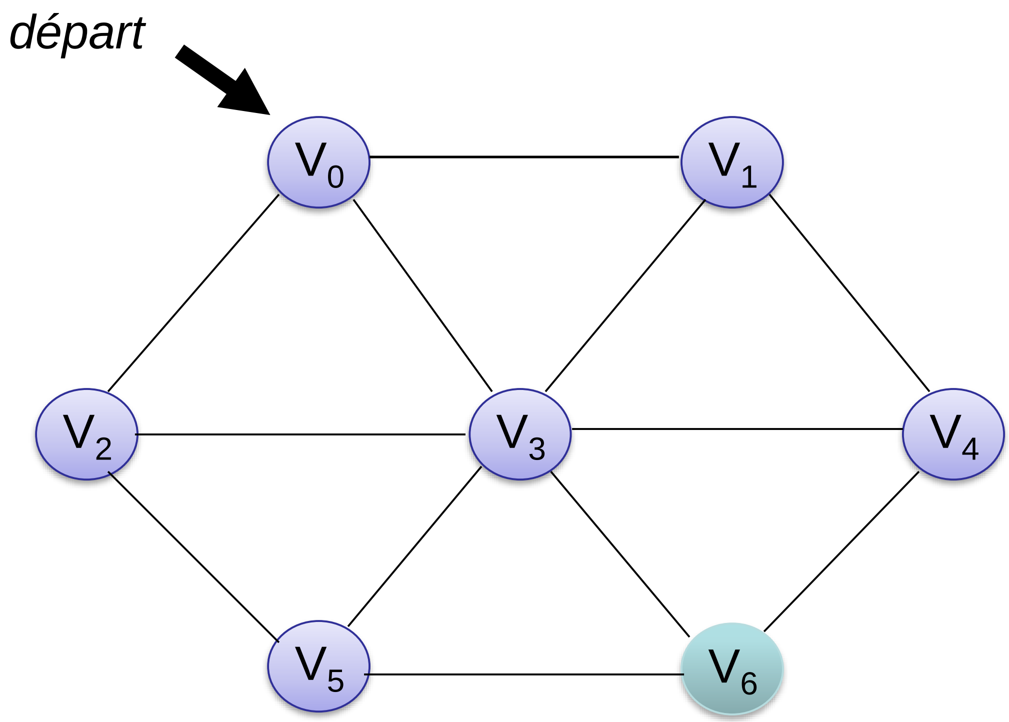
V_0, V_1, V_3, V_2, V_5, V_6, V_4,

Ex. 3 DFS – graphe non orienté



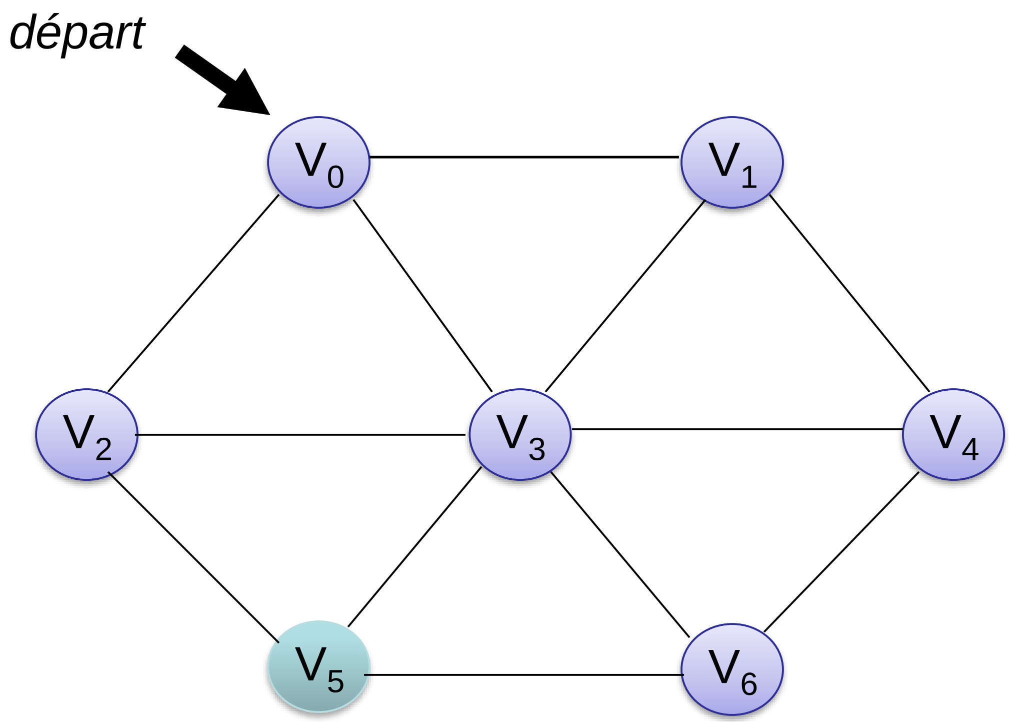
V_0, V_1, V_3, V_2, V_5, V_6, V_4,

Ex. 3 DFS – graphe non orienté



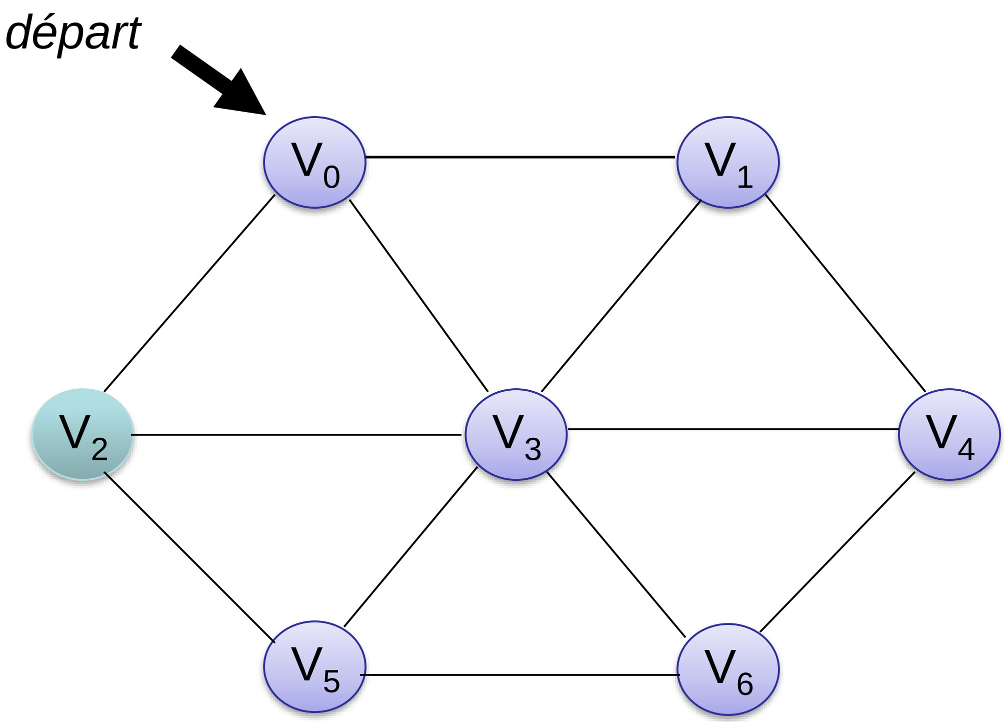
V_0, V_1, V_3, V_2, V_5, V_6, V_4,

Ex. 3 DFS – graphe non orienté



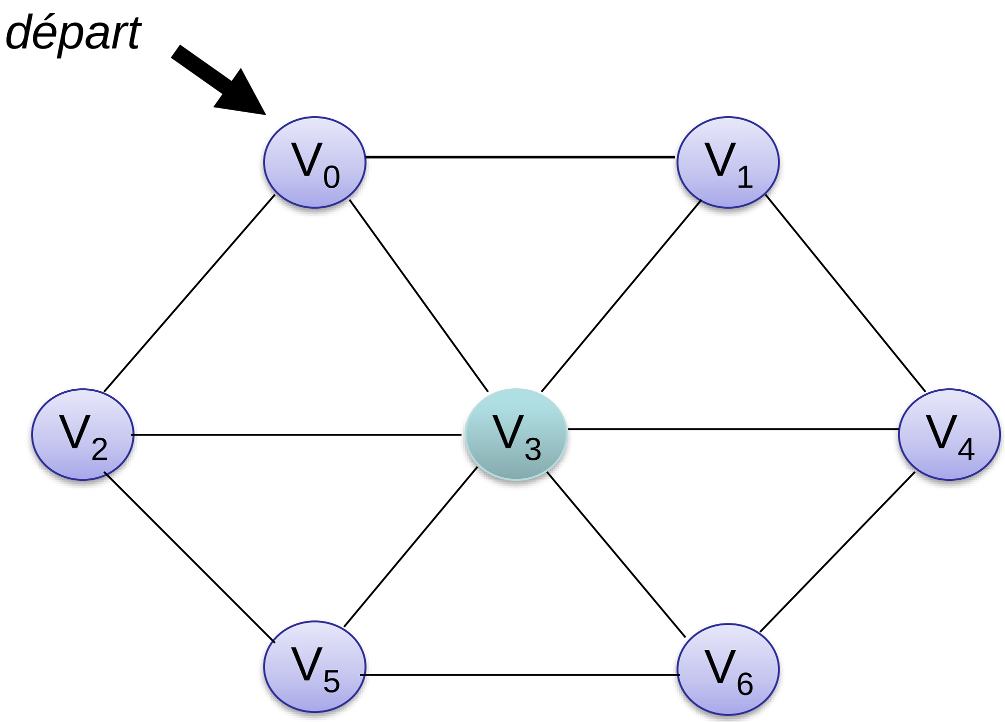
V_0, V_1, V_3, V_2, V_5, V_6, V_4,

Ex. 3 DFS – graphe non orienté



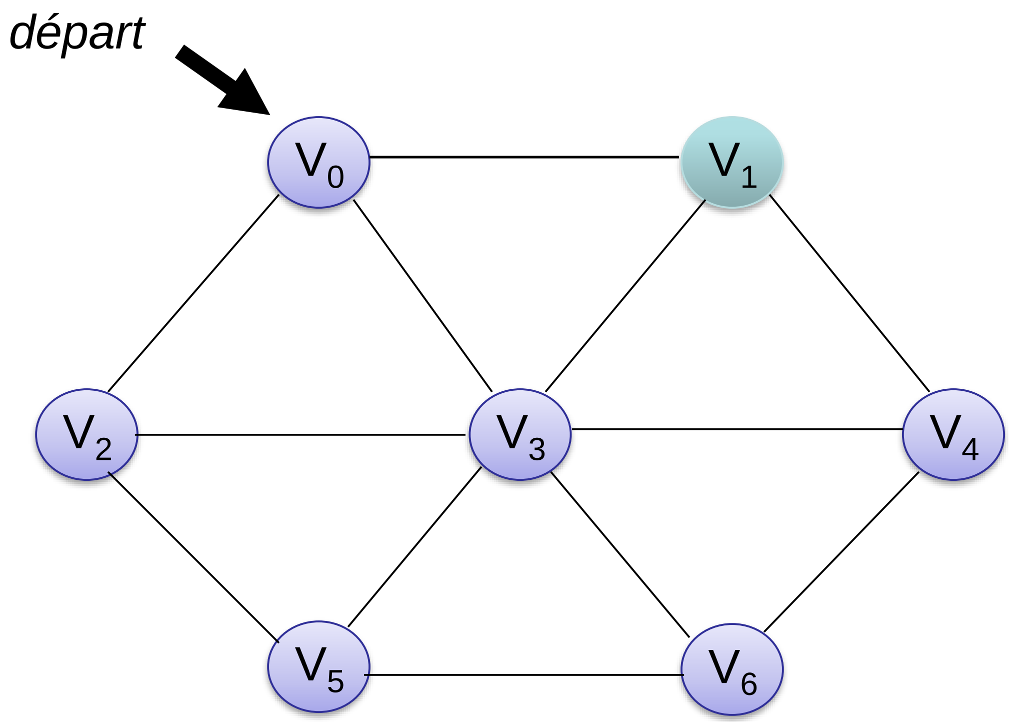
V_0, V_1, V_3, V_2, V_5, V_6, V_4,

Ex. 3 DFS – graphe non orienté



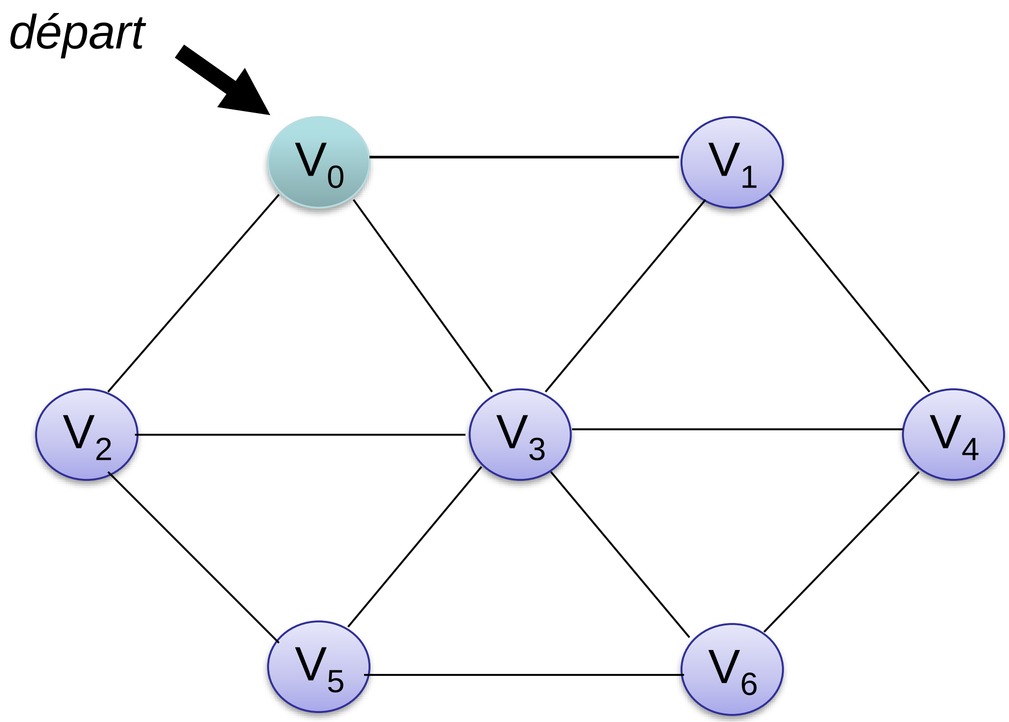
V_0, V_1, V_3, V_2, V_5, V_6, V_4,

Ex. 3 DFS – graphe non orienté



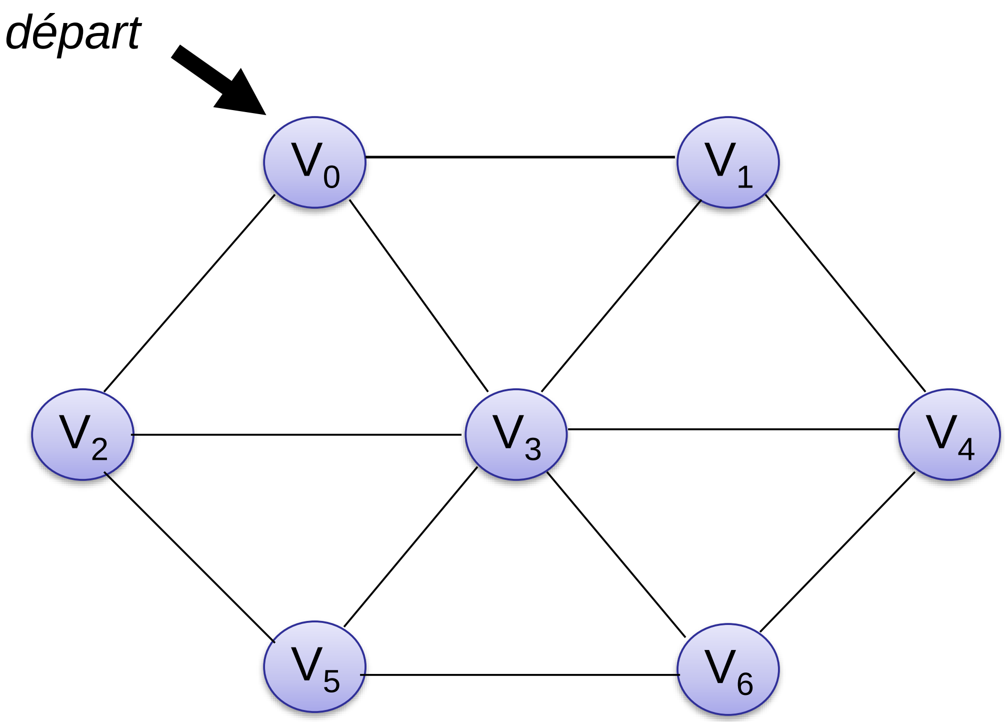
V_0, V_1, V_3, V_2, V_5, V_6, V_4,

Ex. 3 DFS – graphe non orienté



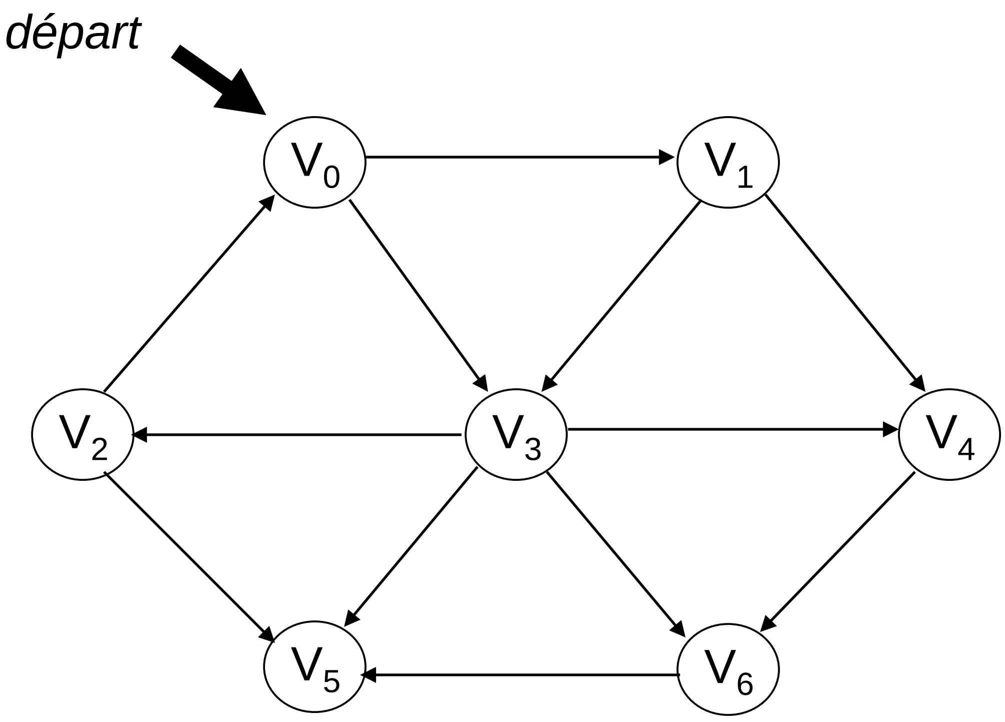
V_0, V_1, V_3, V_2, V_5, V_6, V_4,

Ex. 3 DFS – graphe non orienté



V_0, V_1, V_3, V_2, V_5, V_6, V_4. FIN

Ex. 4 DFS – graphe orienté



V_0, V_1, V_3, V_2, V_5, V_4, V_6.