Formation « graphes »

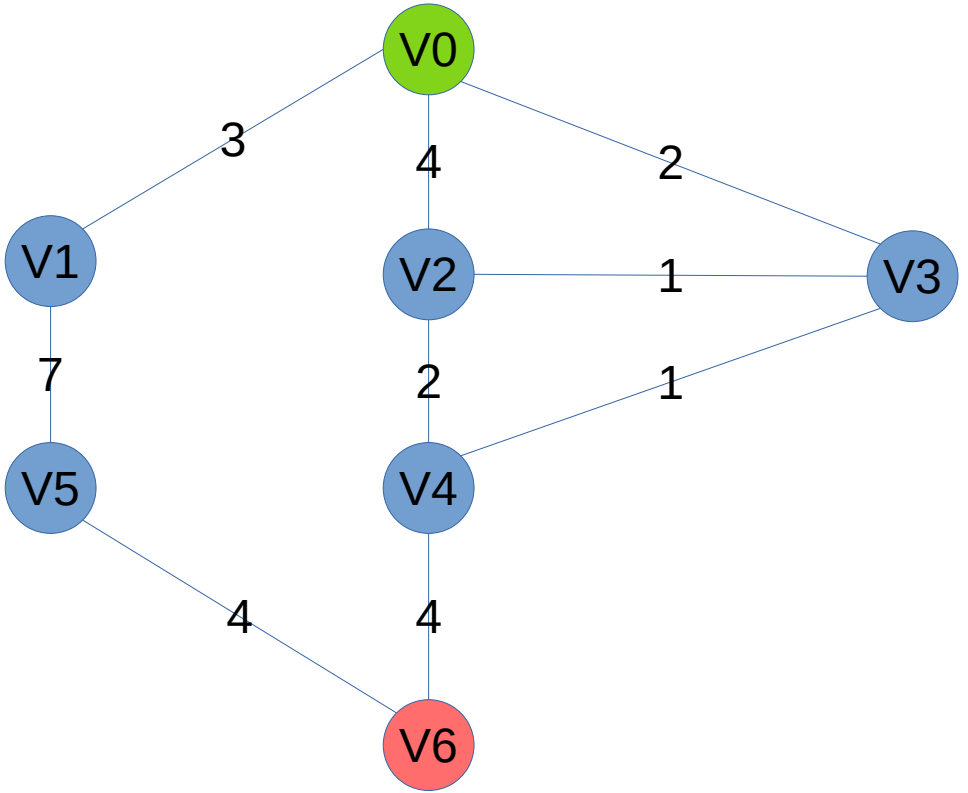
26 mai 2025

Algorithme A*

- But : recherche du plus cours chemin

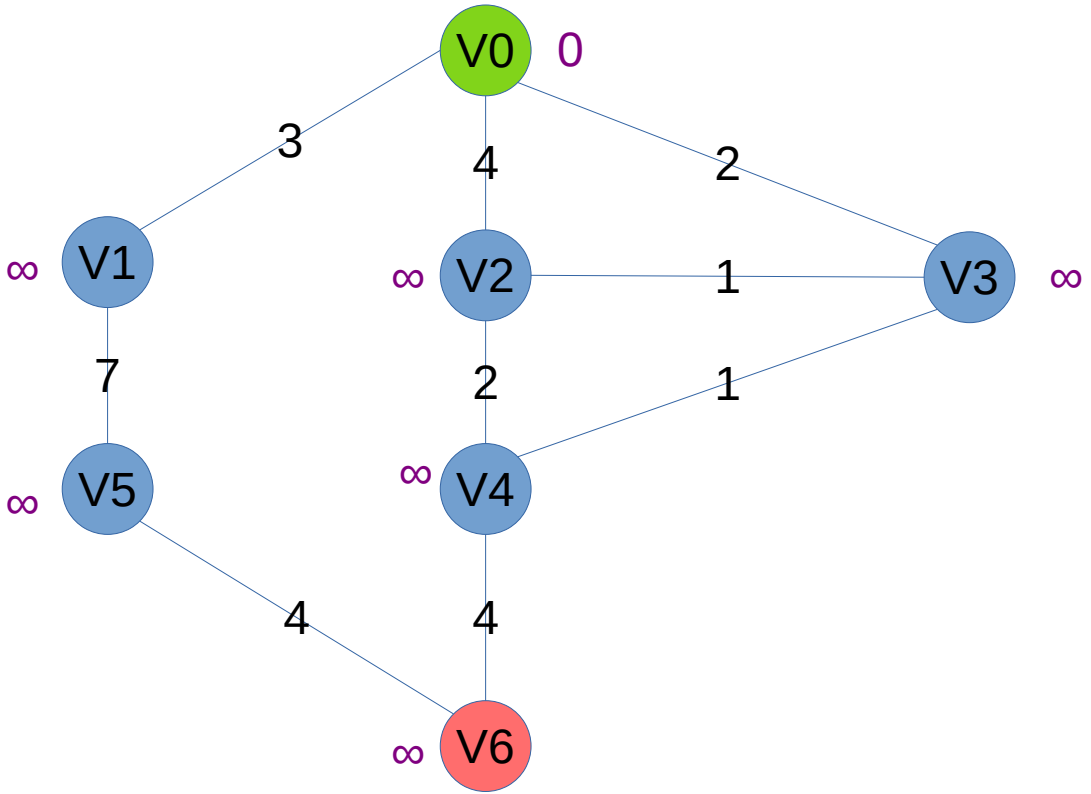
  => Rôle similaire à Dijkstra
- Pré-requis : avoir une heuristique de la distance d’un nœud à l’arrivée
- Principe : le même que Dijkstra, mais on traite les nœuds par distance potentielle à l’arrivée croissante

Graphe exemple



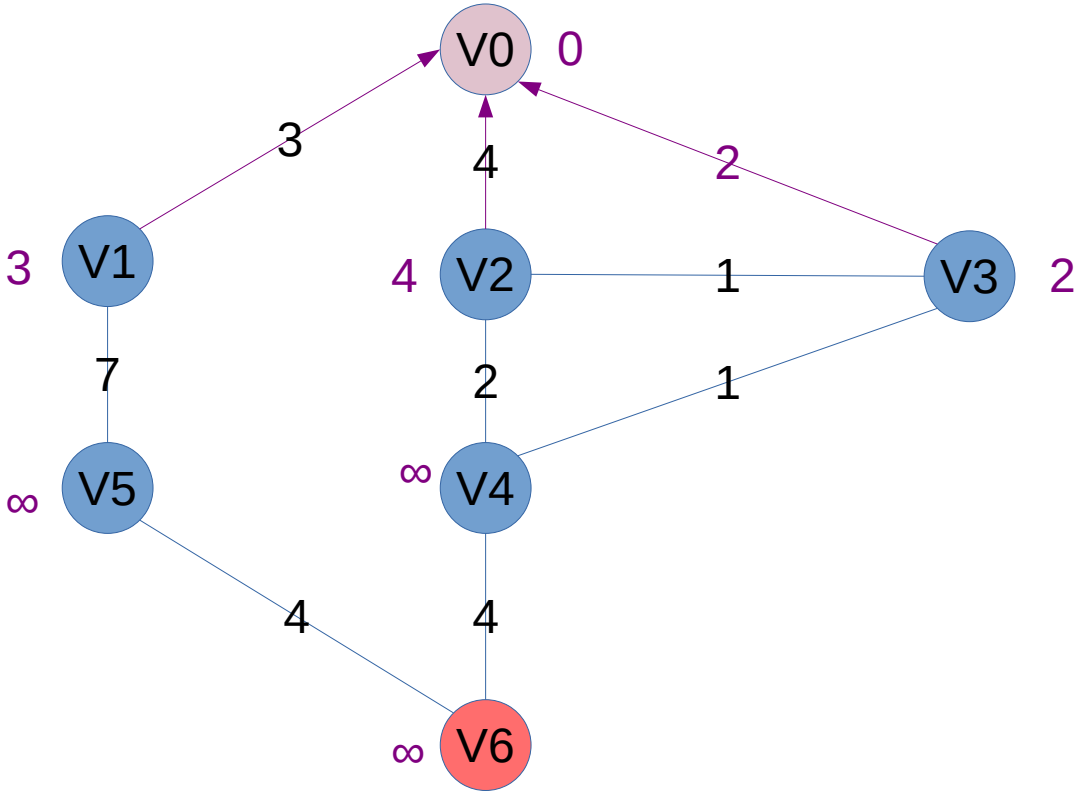
Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

Dijkstra



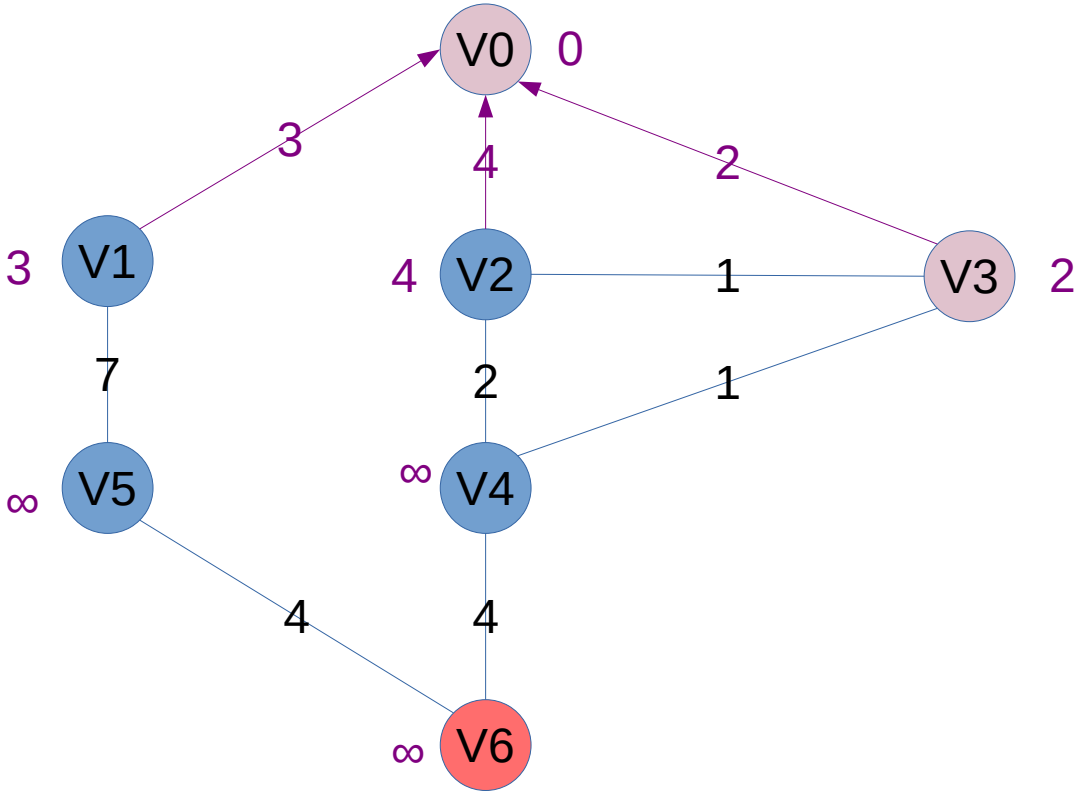
Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

Dijkstra



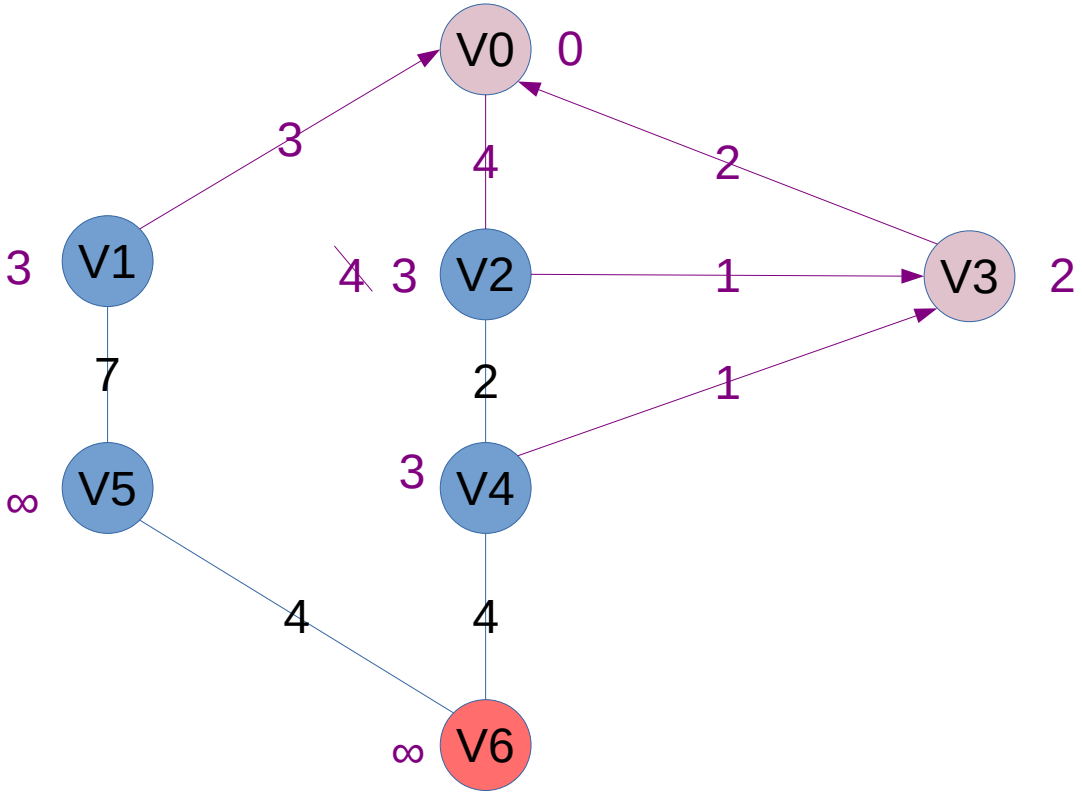
Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

Dijkstra



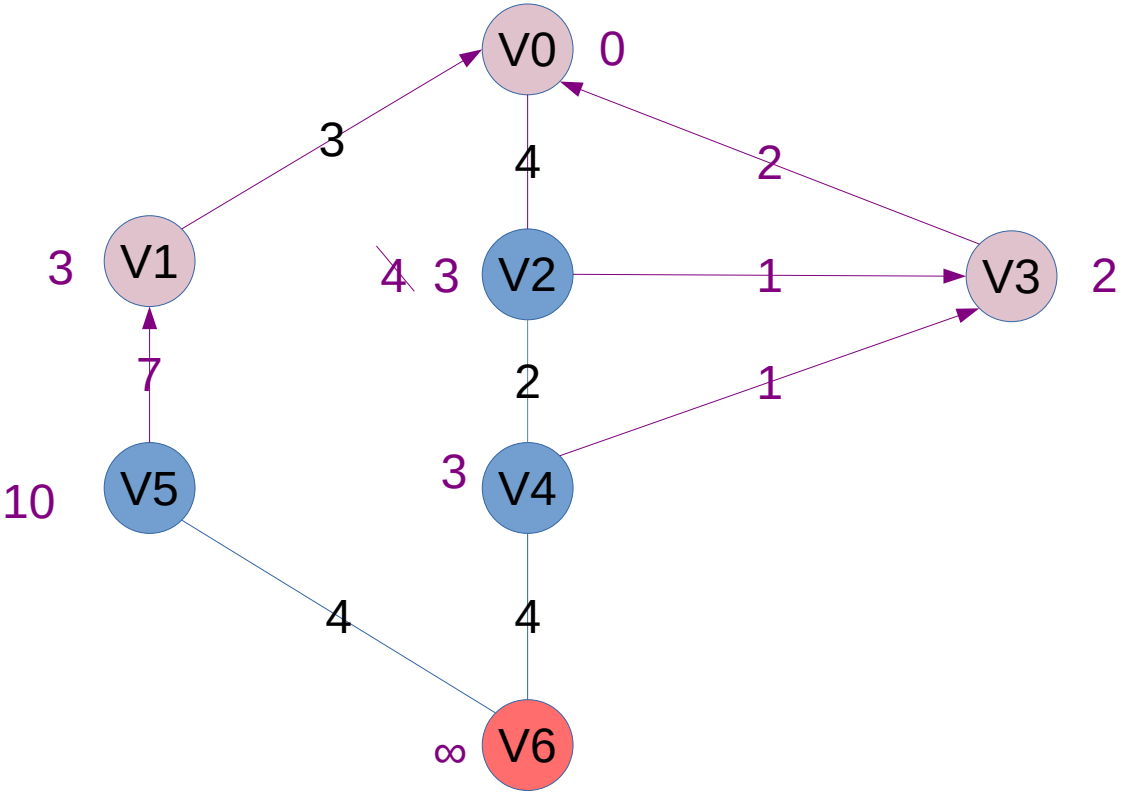
Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

Dijkstra



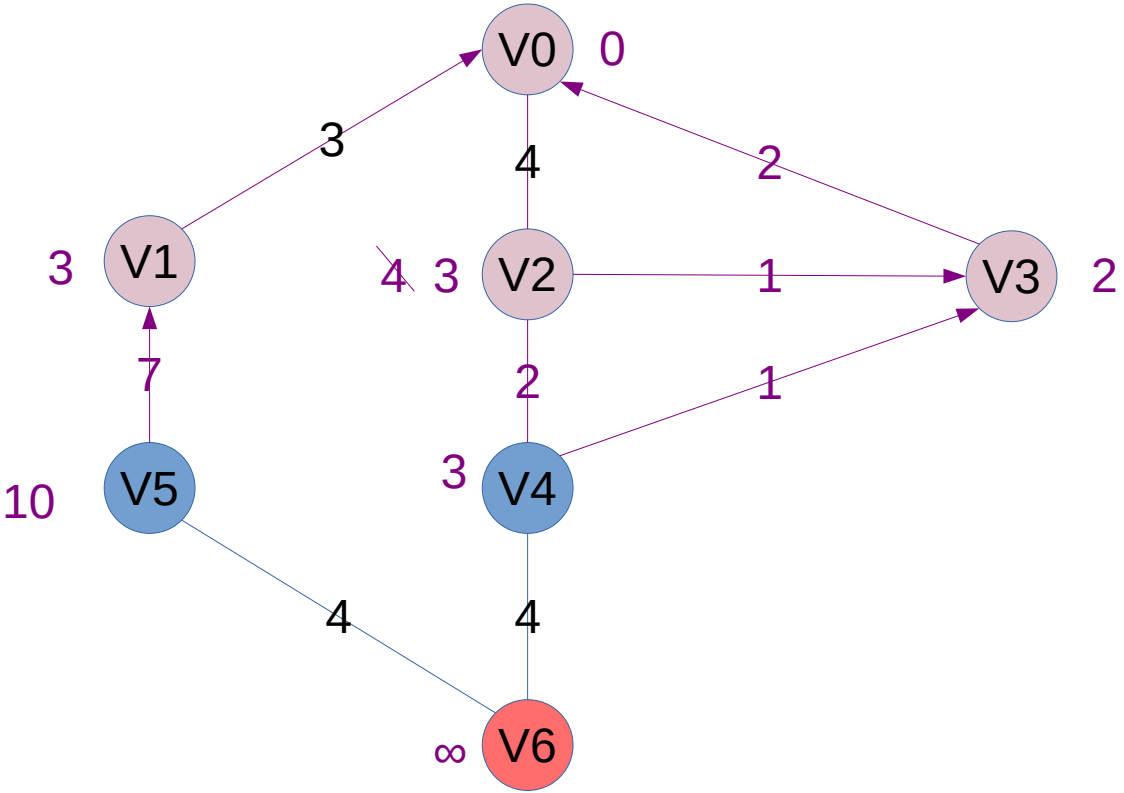
Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

Dijkstra



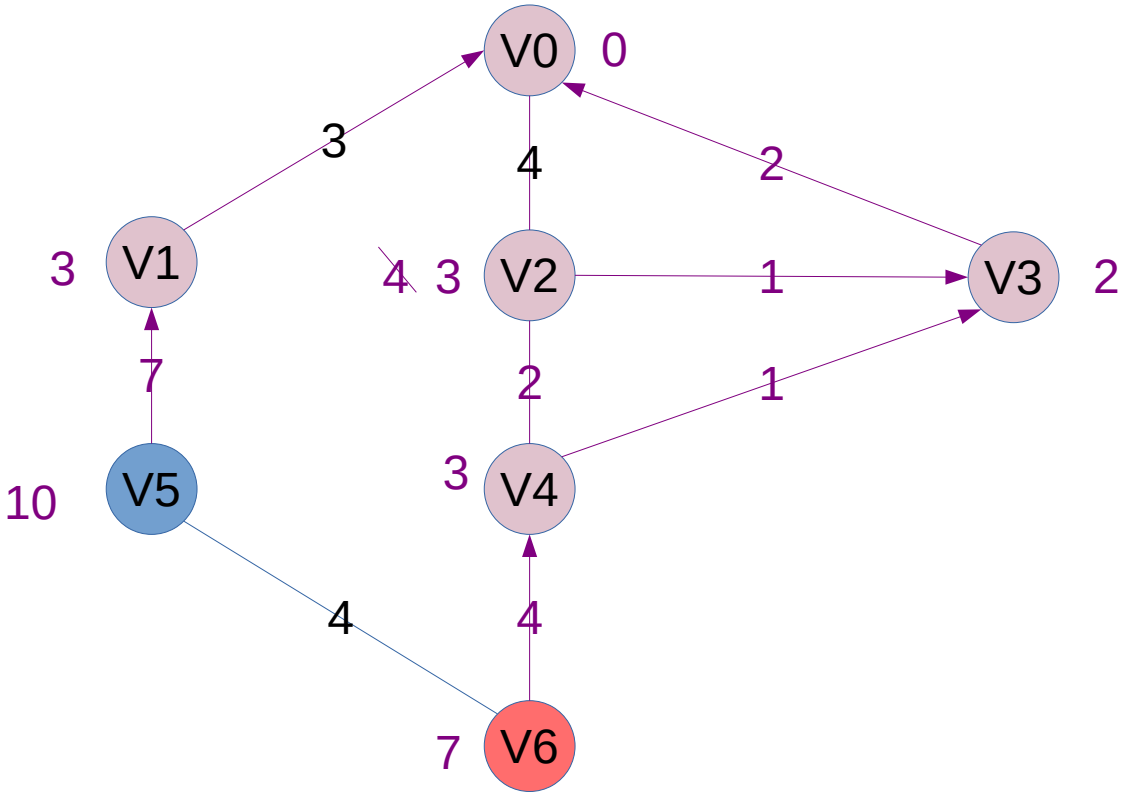
Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

Dijkstra



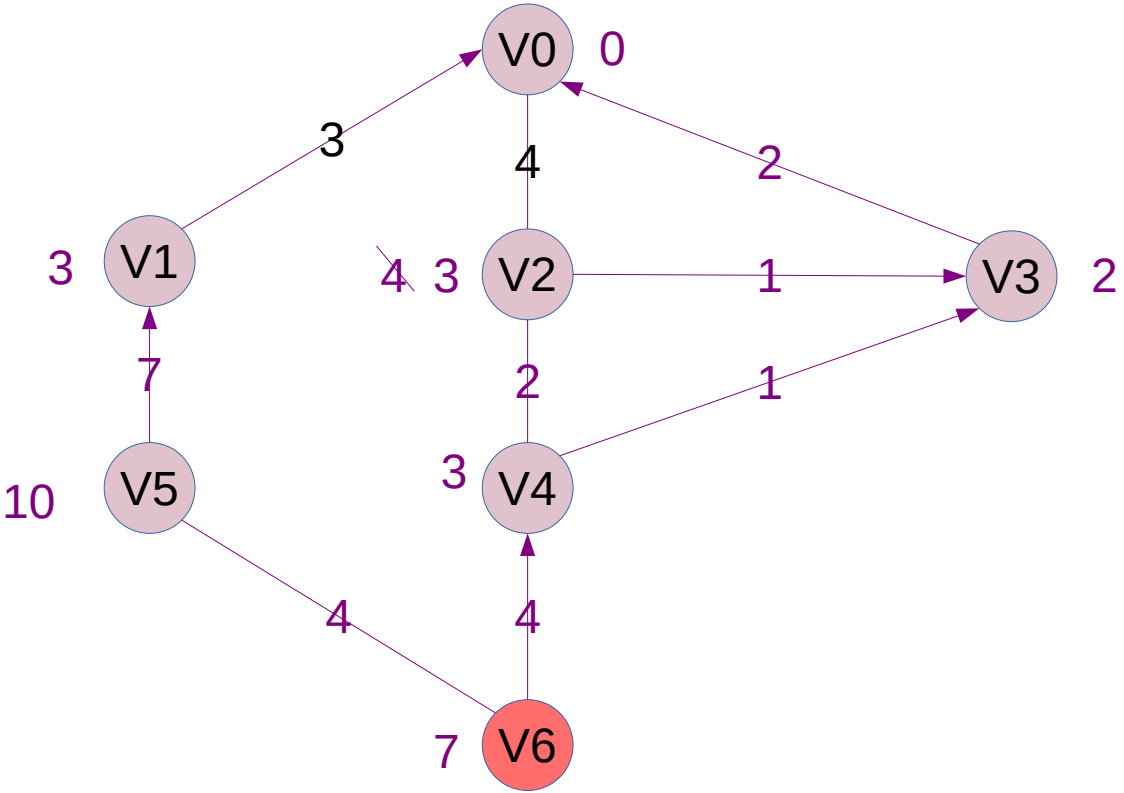
Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

Dijkstra



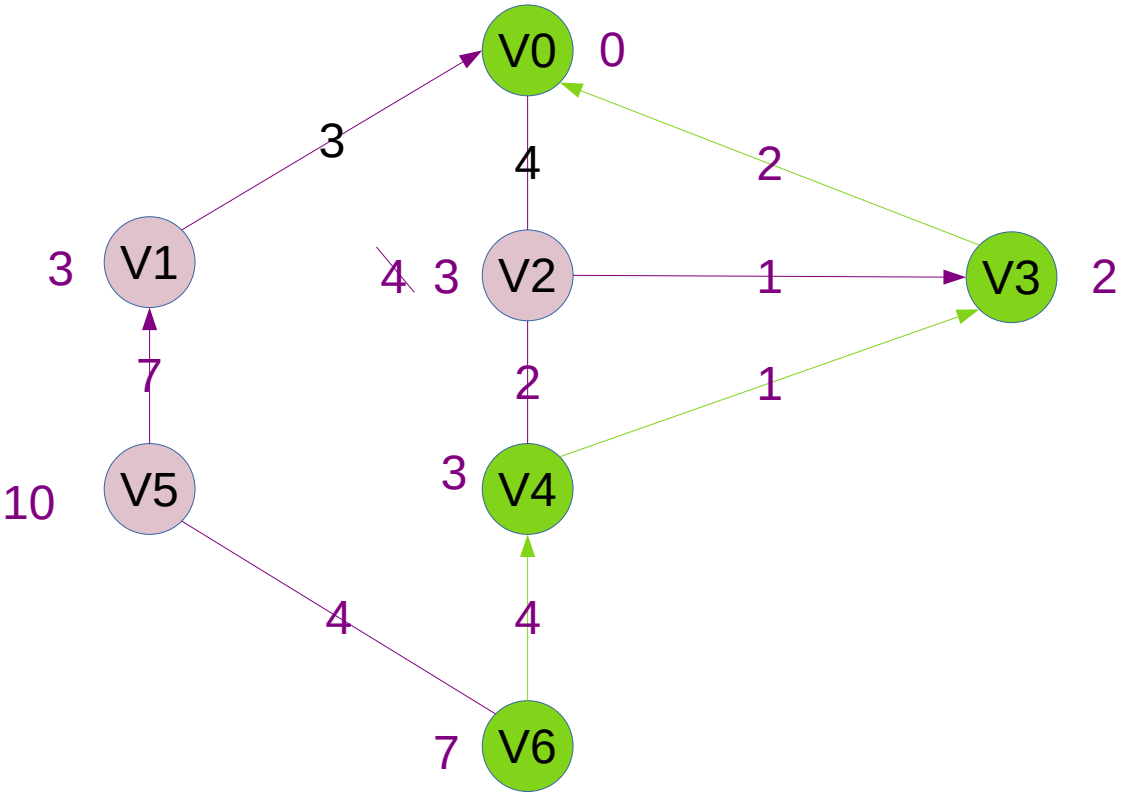
Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

Dijkstra



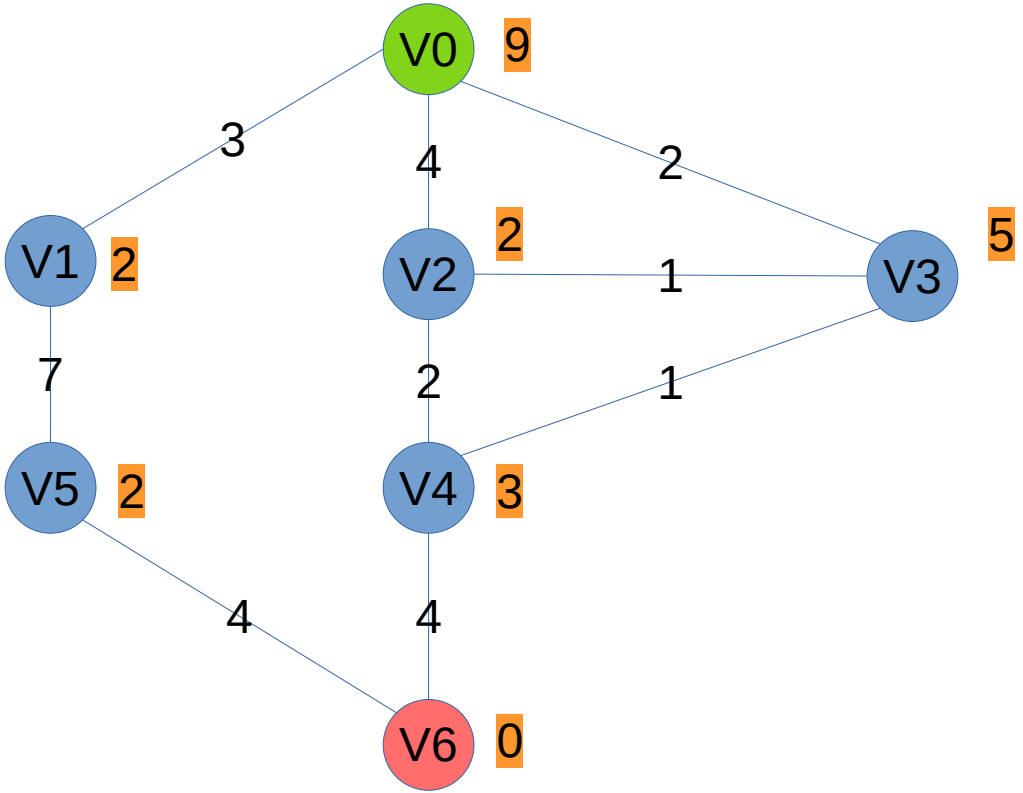
Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

Dijkstra



Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

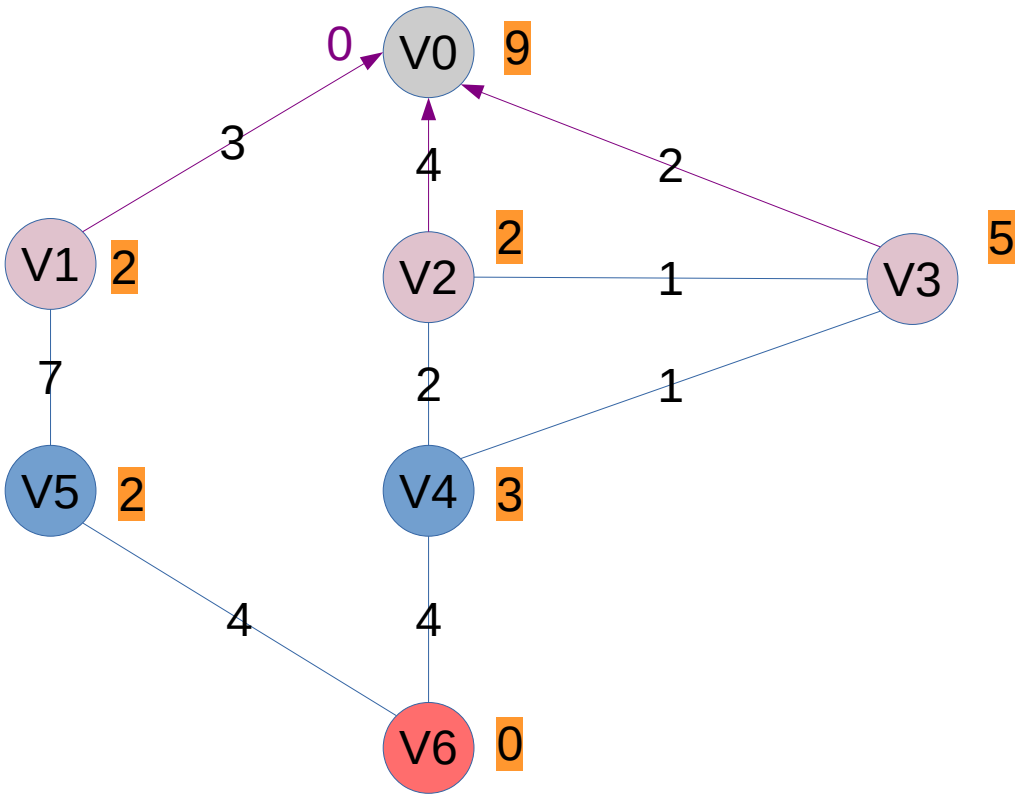
A* : une heuristique affecte une distance à chaque nœud



Fermé
    nœuds déjà traités
Ouvert
    nœud à traiter classés par distance heuristique croissante
Algo
    Similaire à Dijskstra pour le reste
Au début
    Ouvert = [(V0, 9)]
    Fermé : {}

Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

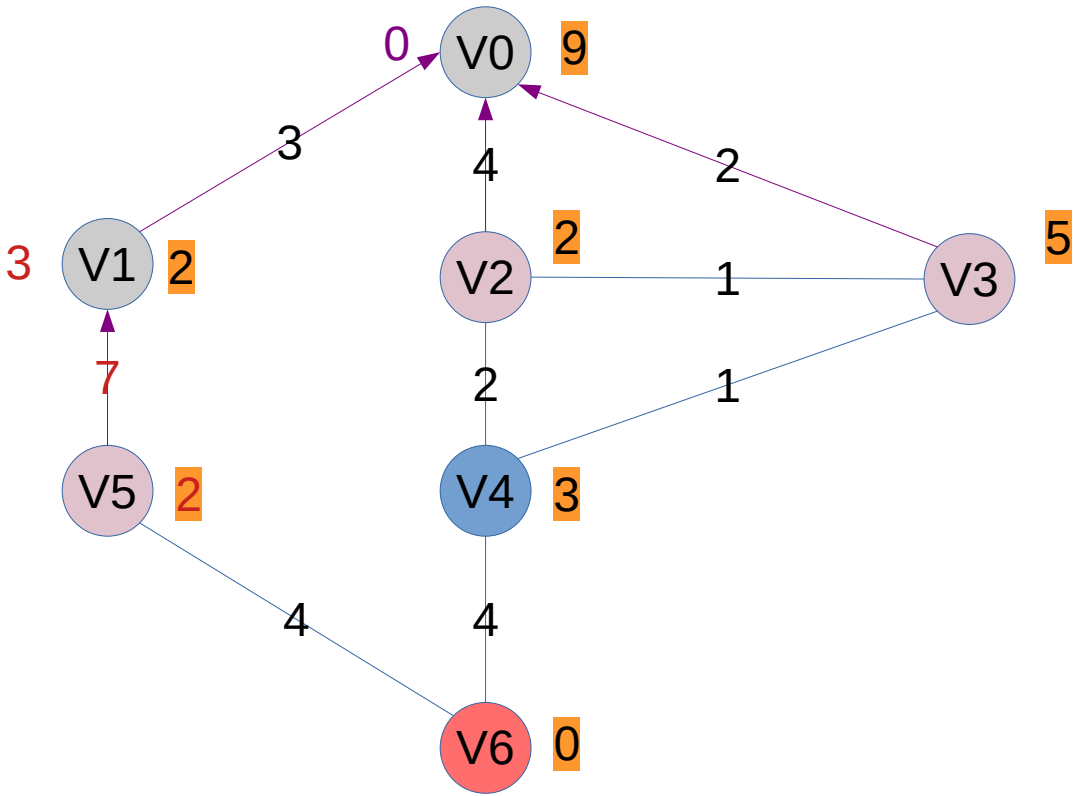
A* : déroulement



Fermé : {V0}
Ouvert = [(V1, 5), (V2, 6), (V3, 7)]

Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

A* : déroulement

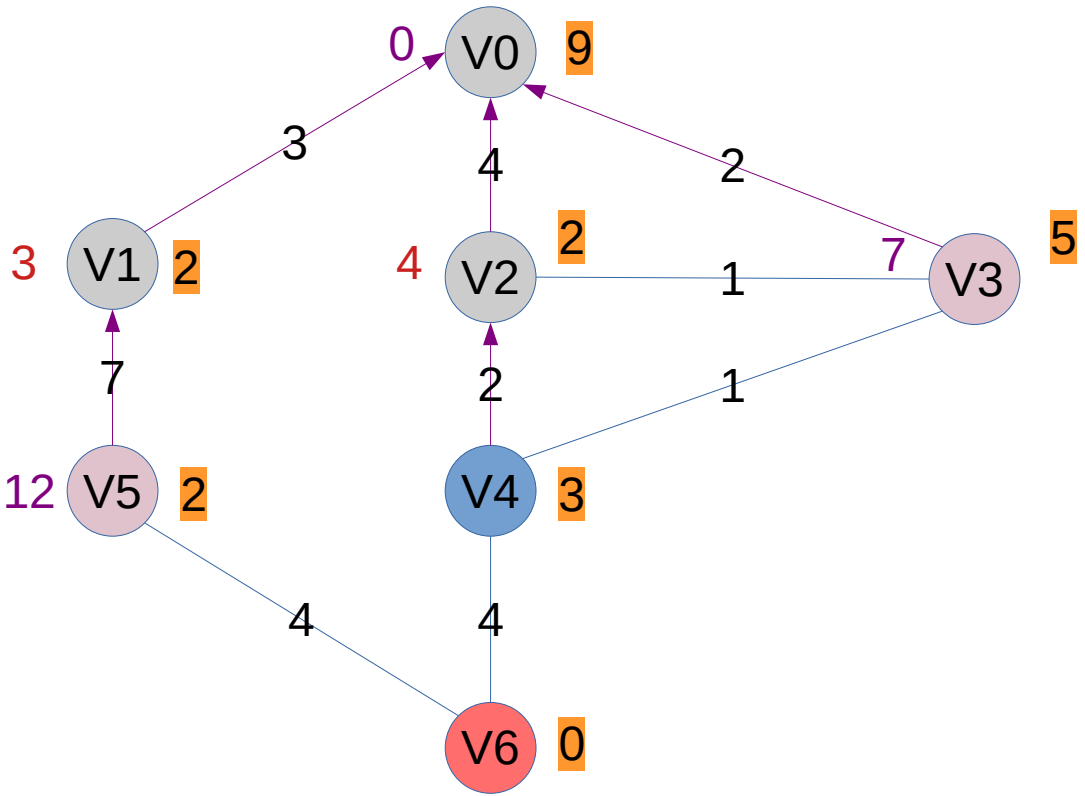


3+7+2

Fermé : {V0, V1}
Ouvert = [(V2, 6), (V3, 7), (V5, 12)]

Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

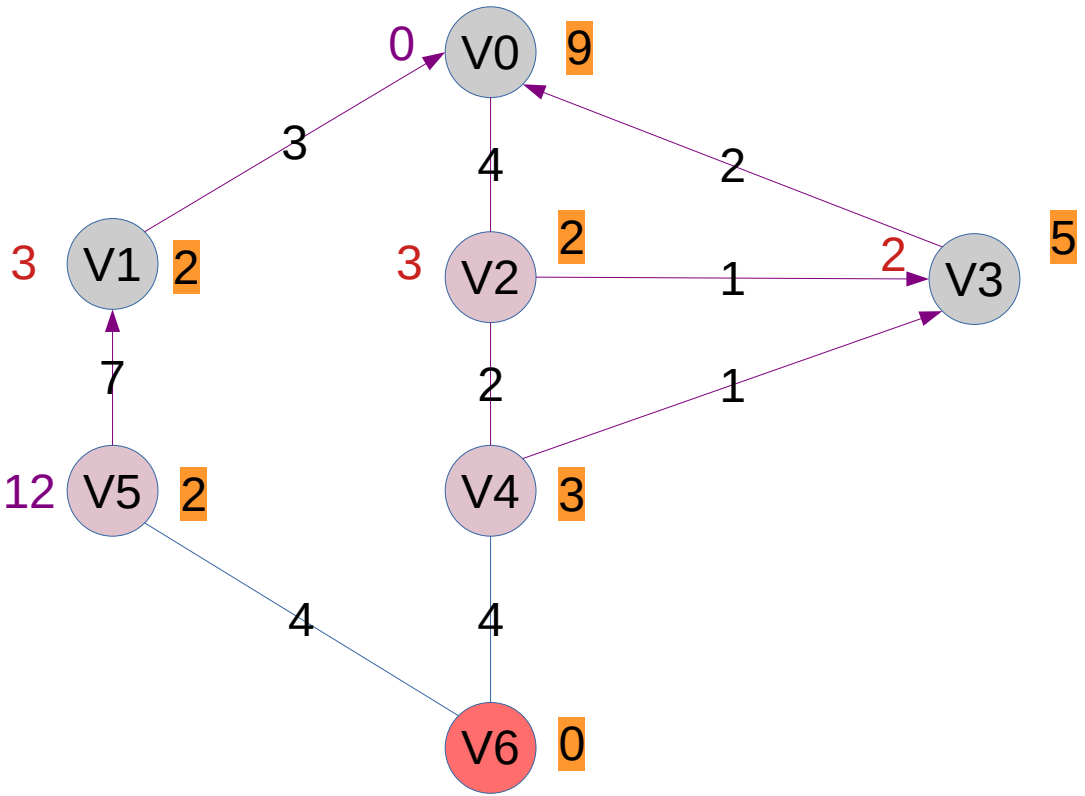
A* : déroulement



Fermé : {V0, V1, V2}
Ouvert = [(V3, 7), (V4, 9), (V5, 12)]

Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

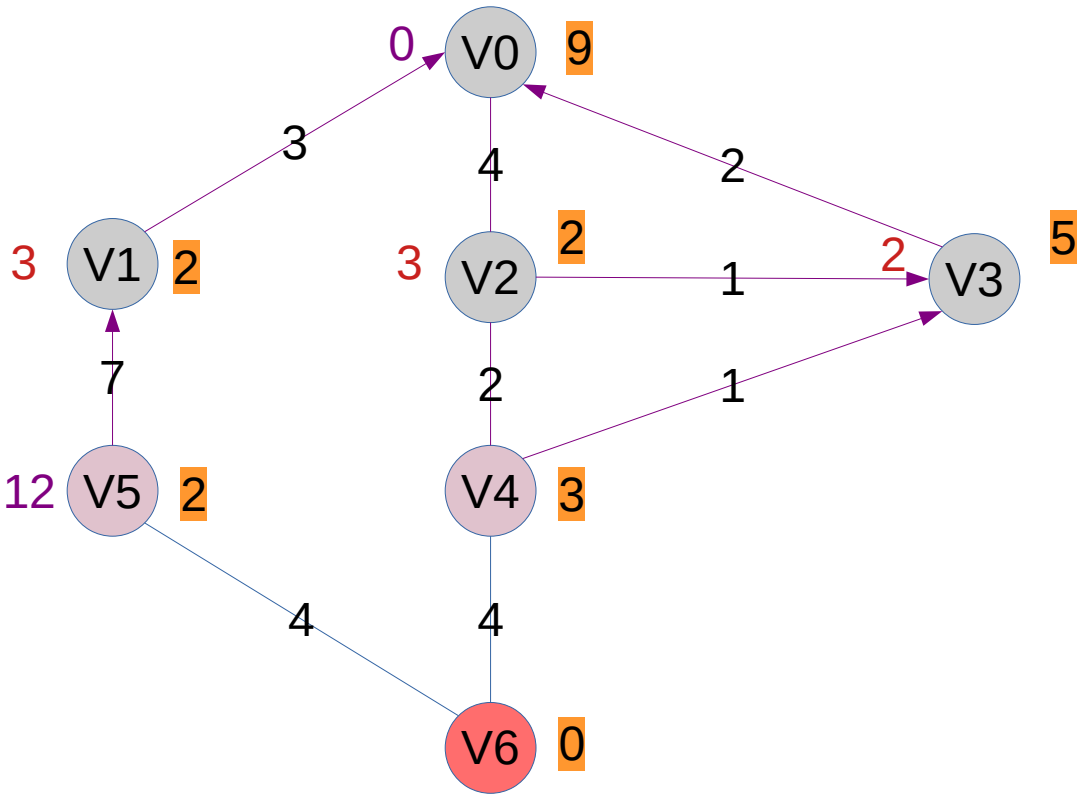
A* : déroulement



Fermé : {V0, V1, V3}
Ouvert = [(V2, 5), (V4, 6), (V5, 12)]

Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

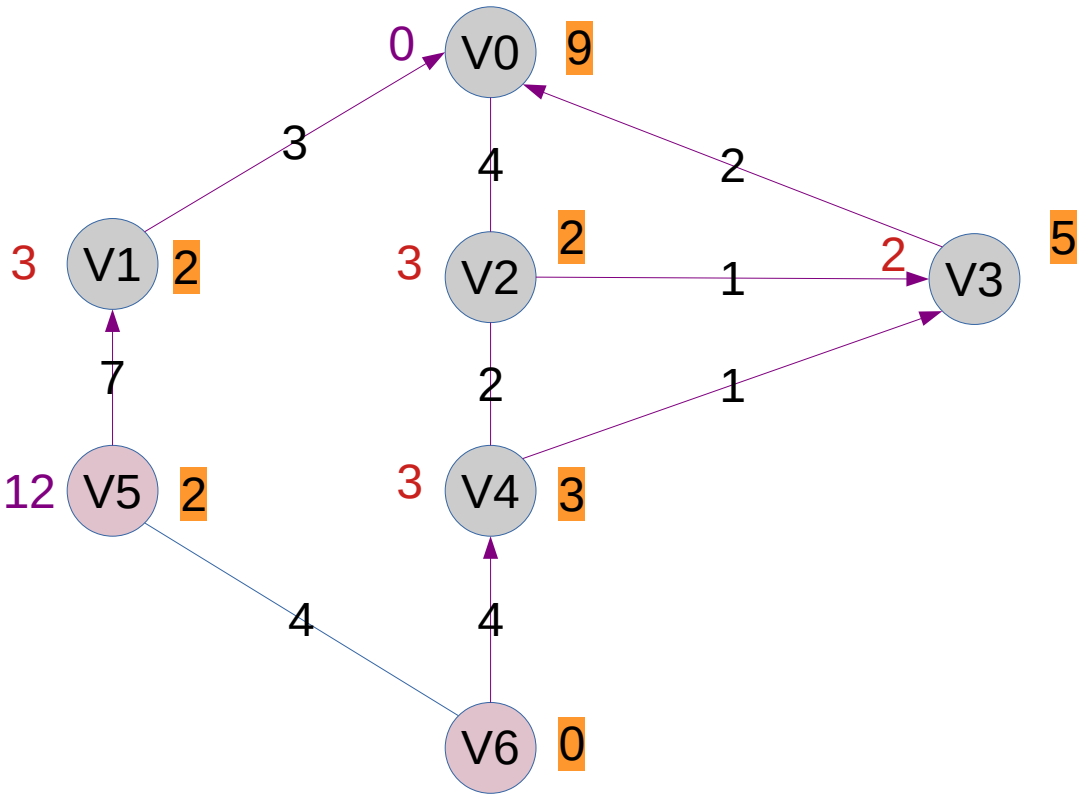
A* : déroulement



Fermé : {V0, V1, V3, V2}
Ouvert = [(V4, 6), (V5, 12)]

Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

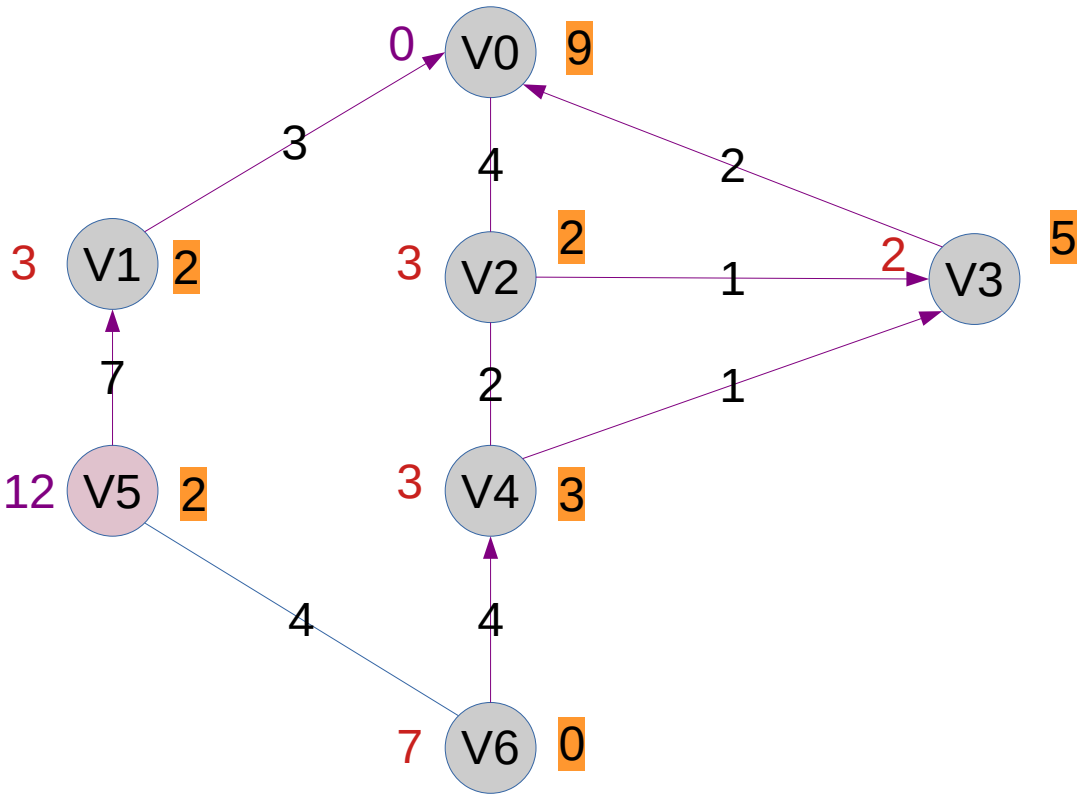
A* : déroulement



Fermé : {V0, V1, V3, V2, V4}
Ouvert = [(V6, 7), (V5, 12)]

Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

A* : déroulement



Fermé : {V0, V1, V3, V2, V4, V6}
Ouvert = [(V5, 12)]

Graphe repris de https://gdac.uqam.ca/inf4230/diapos/04-recherche-heuristique.pdf

A* et Dijsktra

- Quelques remarques :
  - Si h(n) = 0 ∀ n, A* = Dijkstra
  - A* optimal si h(n) ≤ d(n) ∀ n
  - A* meilleur si sous-estimation pas trop faible
- Illustration de la comparaison
  - https://qiao.github.io/PathFinding.js/visual/

Kruskal

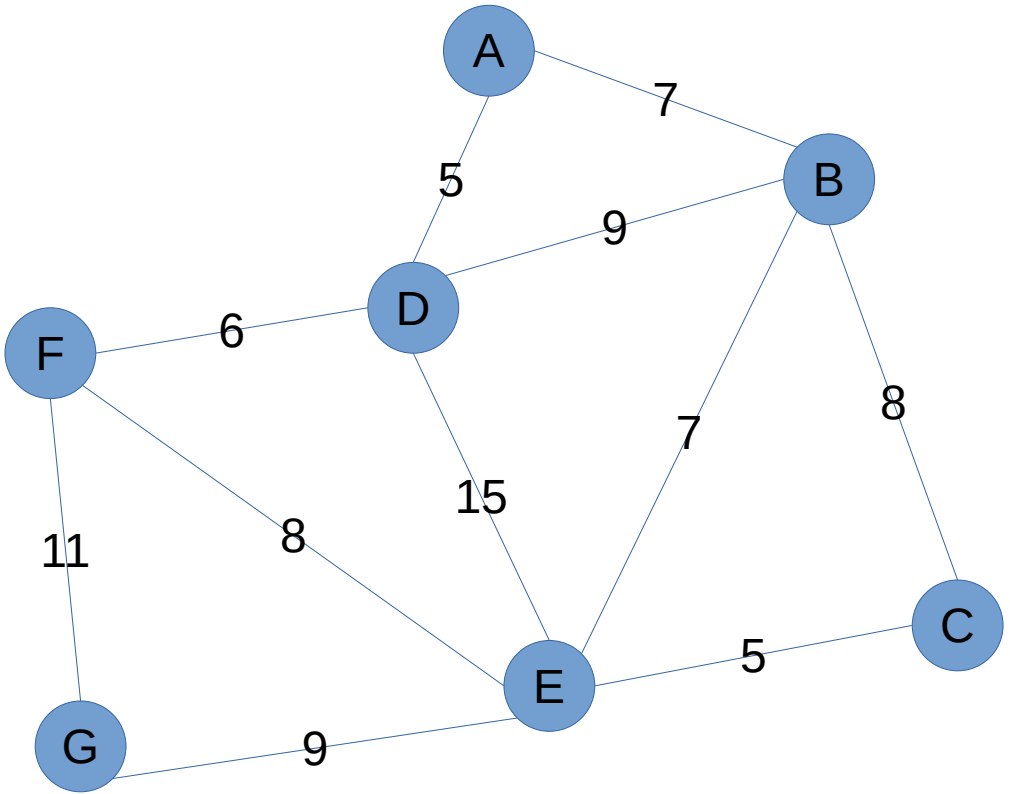
- But

  Calculer l’arbre de recouvrement minimal d’un graphe

- Application

  Relier des villes par des longueurs minimales de câbles par exemple

Graphe Exemple



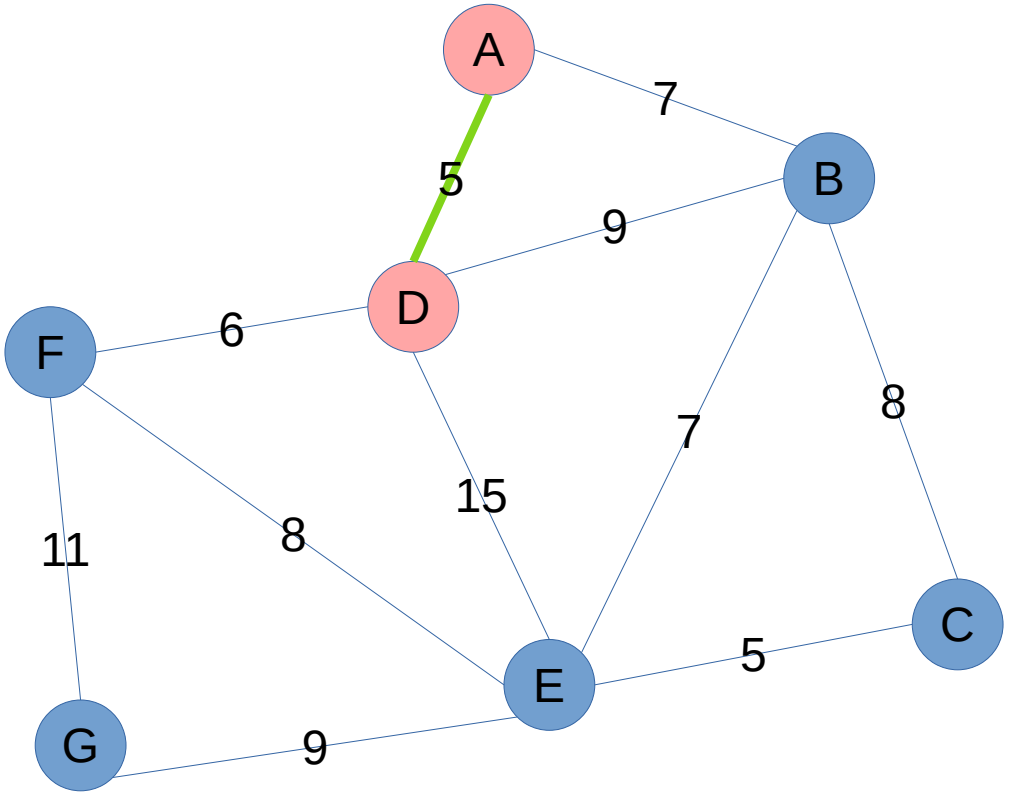
Principe :

on prend les arêtes ne formant pas de cycle par ordre croissant de poids

on s'arrête quand tous les nœuds sont couverts

Graphe repris de https://fr.wikipedia.org/wiki/Algorithme_de_Kruskal

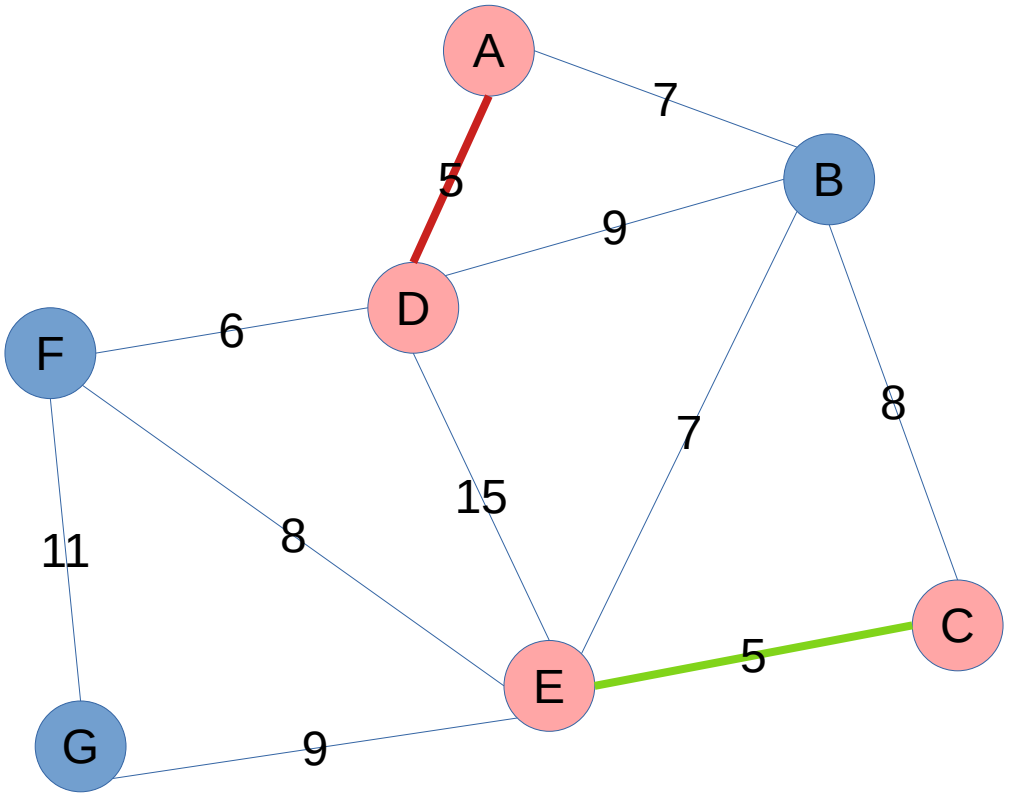
Kruskal : déroulement



A – D (5)
C – E (5)
D – F (6)
A – B (7)
B – E (7)
F – E (8)
B – C (8)
B – D (9)
E – G (9)
F – G (11)

Graphe repris de https://fr.wikipedia.org/wiki/Algorithme_de_Kruskal

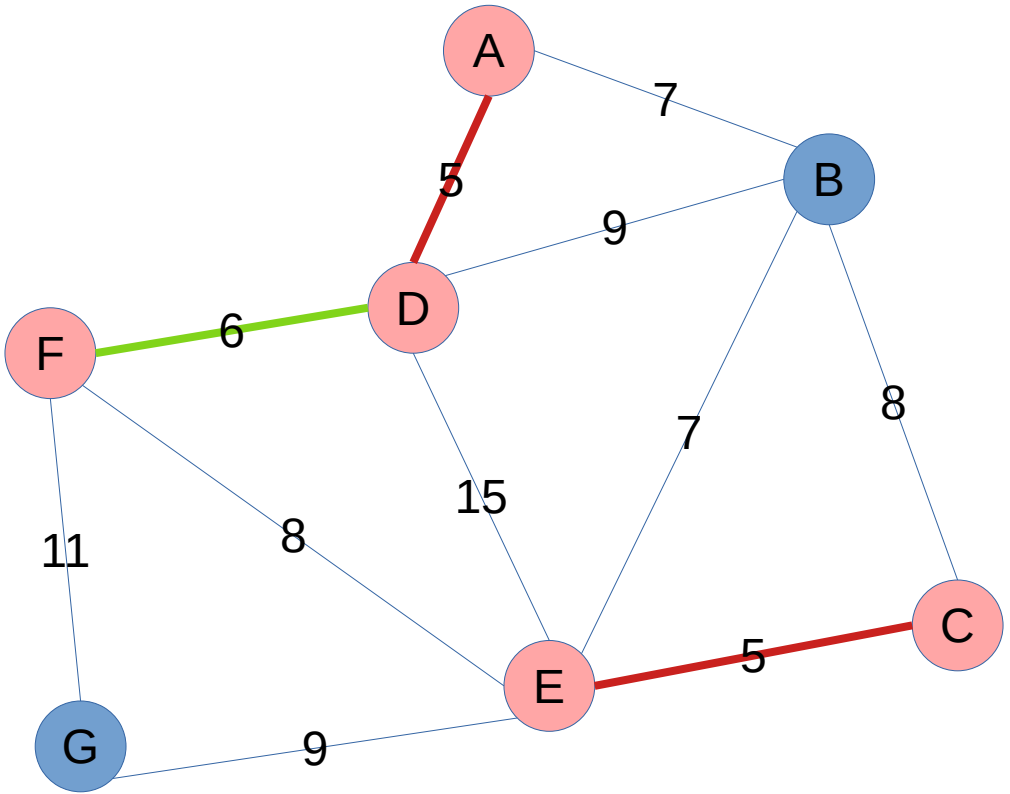
Kruskal : déroulement



A – D (5)
C – E (5)
D – F (6)
A – B (7)
B – E (7)
F – E (8)
B – C (8)
B – D (9)
E – G (9)
F – G (11)

Graphe repris de https://fr.wikipedia.org/wiki/Algorithme_de_Kruskal

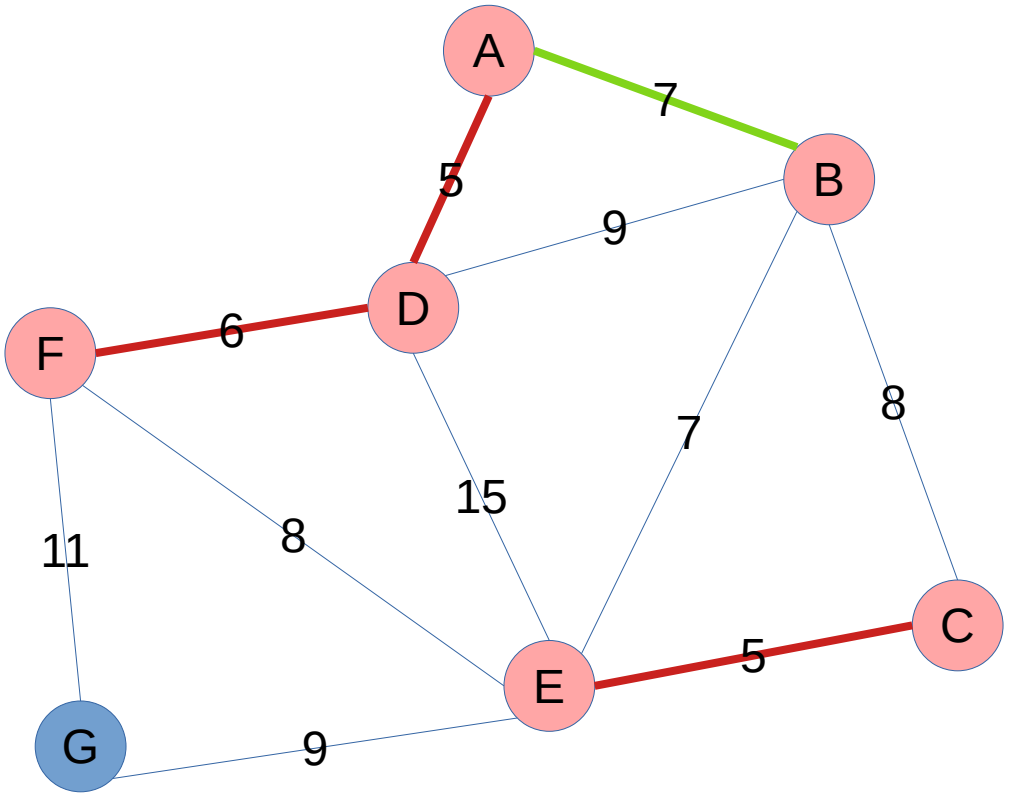
Kruskal : déroulement



A – D (5)
C – E (5)
D – F (6)
A – B (7)
B – E (7)
F – E (8)
B – C (8)
B – D (9)
E – G (9)
F – G (11)

Graphe repris de https://fr.wikipedia.org/wiki/Algorithme_de_Kruskal

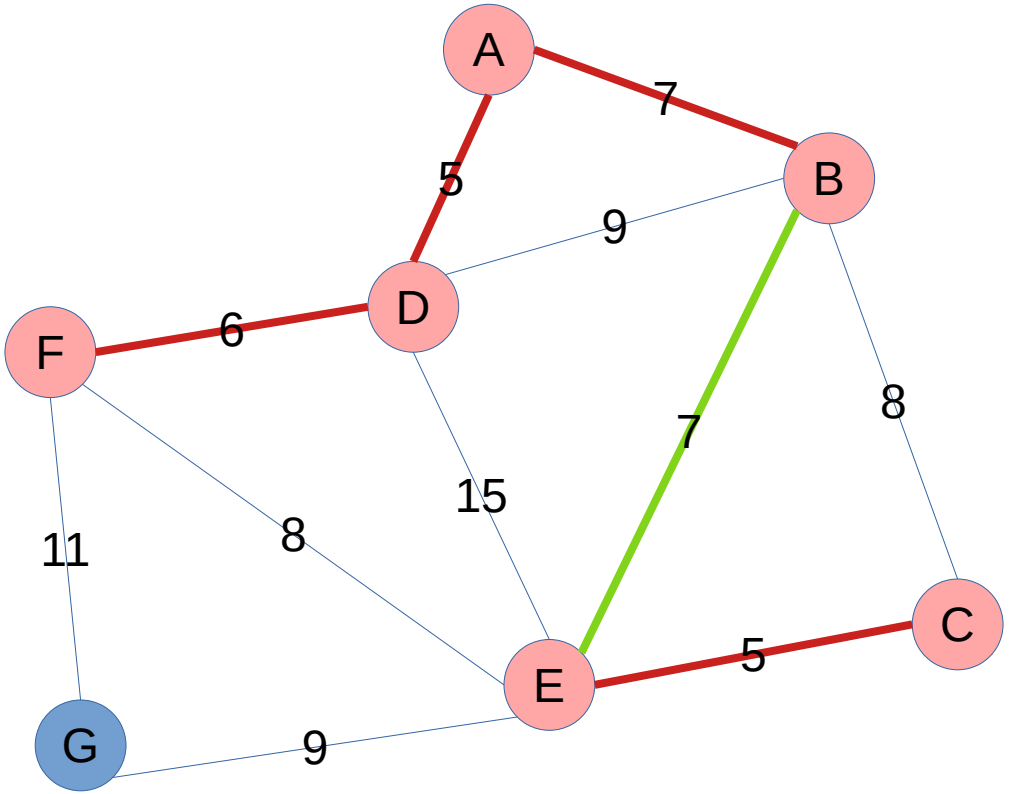
Kruskal : déroulement



A – D (5)
C – E (5)
D – F (6)
A – B (7)
B – E (7)
F – E (8)
B – C (8)
B – D (9)
E – G (9)
F – G (11)

Graphe repris de https://fr.wikipedia.org/wiki/Algorithme_de_Kruskal

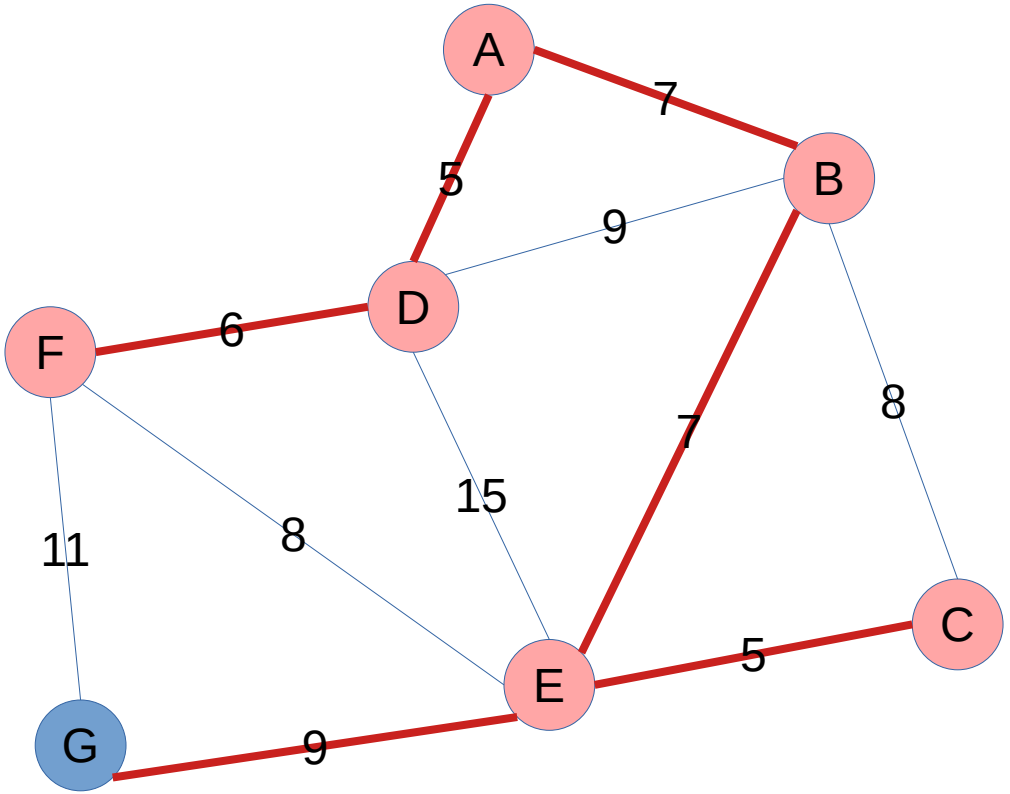
Kruskal : déroulement



A – D (5)
C – E (5)
D – F (6)
A – B (7)
B – E (7)
F – E (8)
B – C (8)
B – D (9)
E – G (9)
F – G (11)

Graphe repris de https://fr.wikipedia.org/wiki/Algorithme_de_Kruskal

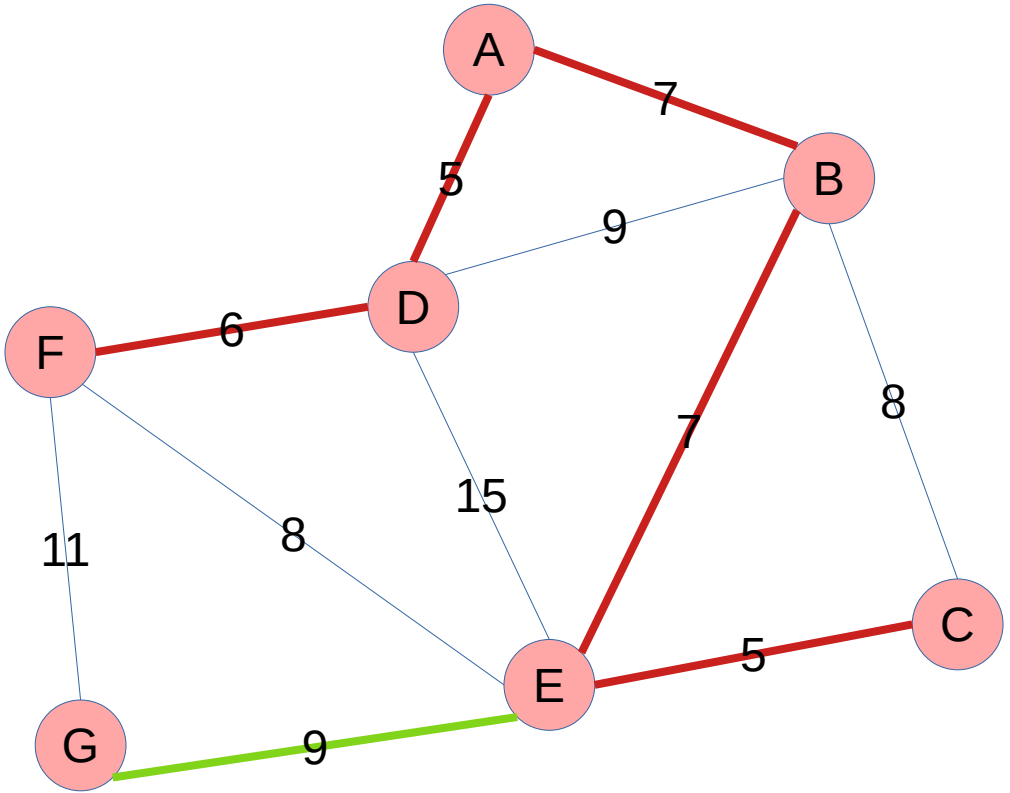
Kruskal : déroulement



A – D (5)
C – E (5)
D – F (6)
A – B (7)
B – E (7)
F – E (8)
B – C (8)
B – D (9)
E – G (9)
F – G (11)

Graphe repris de https://fr.wikipedia.org/wiki/Algorithme_de_Kruskal

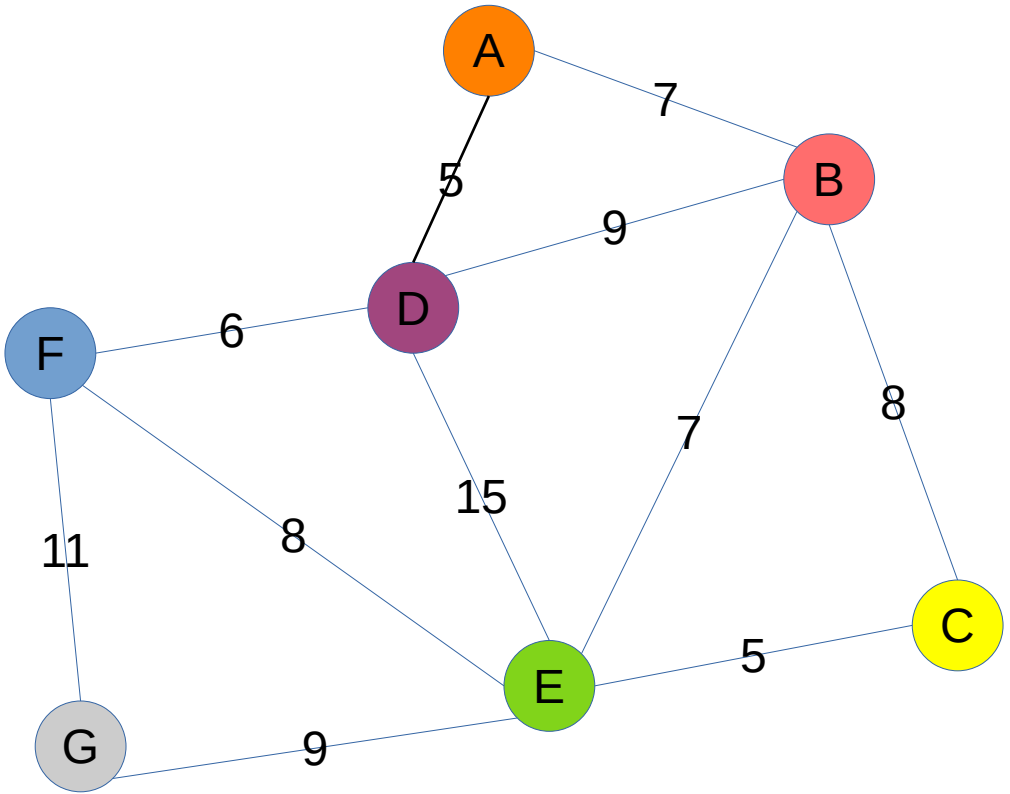
Kruskal : déroulement



A – D (5)
C – E (5)
D – F (6)
A – B (7)
B – E (7)
F – E (8)
B – C (8)
B – D (9)
E – G (9)
F – G (11)

Graphe repris de https://fr.wikipedia.org/wiki/Algorithme_de_Kruskal

Kruskal revisité



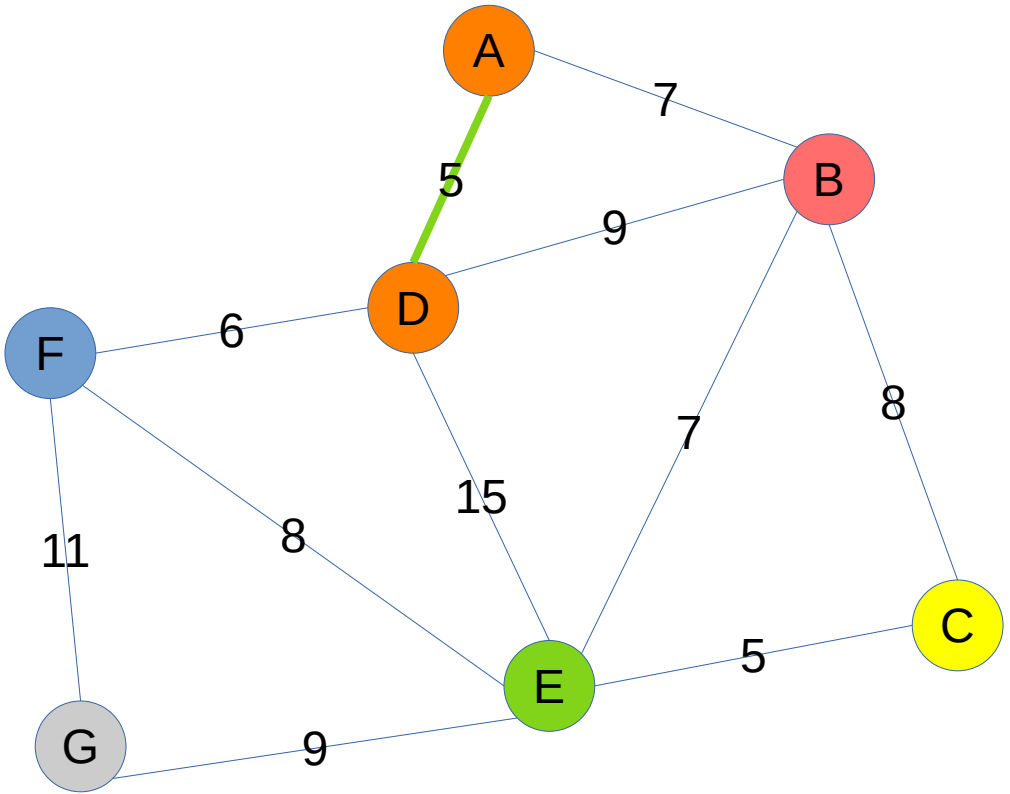
A – D (5)
C – E (5)
D – F (6)
A – B (7)
B – E (7)
F – E (8)
B – C (8)
B – D (9)
E – G (9)
F – G (11)

Classes d'équivalence :

{A}
{B}
{C}
{D}
{E}
{F}
{G}

Graphe repris de https://fr.wikipedia.org/wiki/Algorithme_de_Kruskal

Kruskal revisité



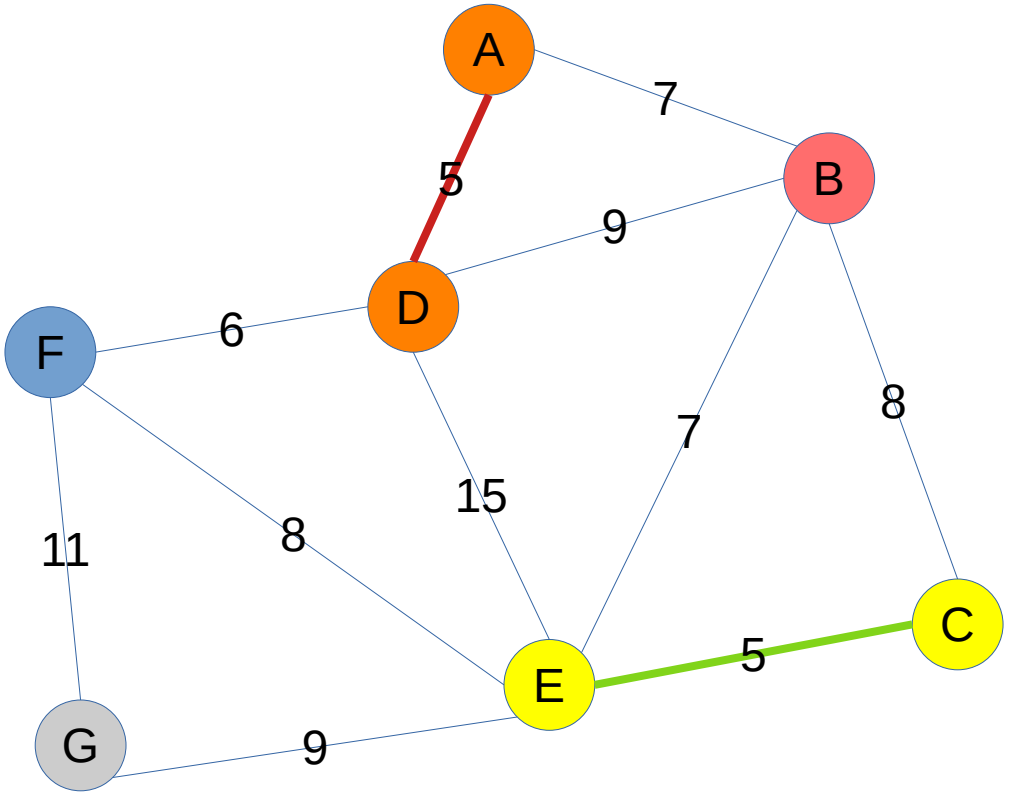
A – D (5)
C – E (5)
D – F (6)
A – B (7)
B – E (7)
F – E (8)
B – C (8)
B – D (9)
E – G (9)
F – G (11)

Classes d'équivalence :

{A, D}
{B}
{C}
{E}
{F}
{G}

Graphe repris de https://fr.wikipedia.org/wiki/Algorithme_de_Kruskal

Kruskal : revisité



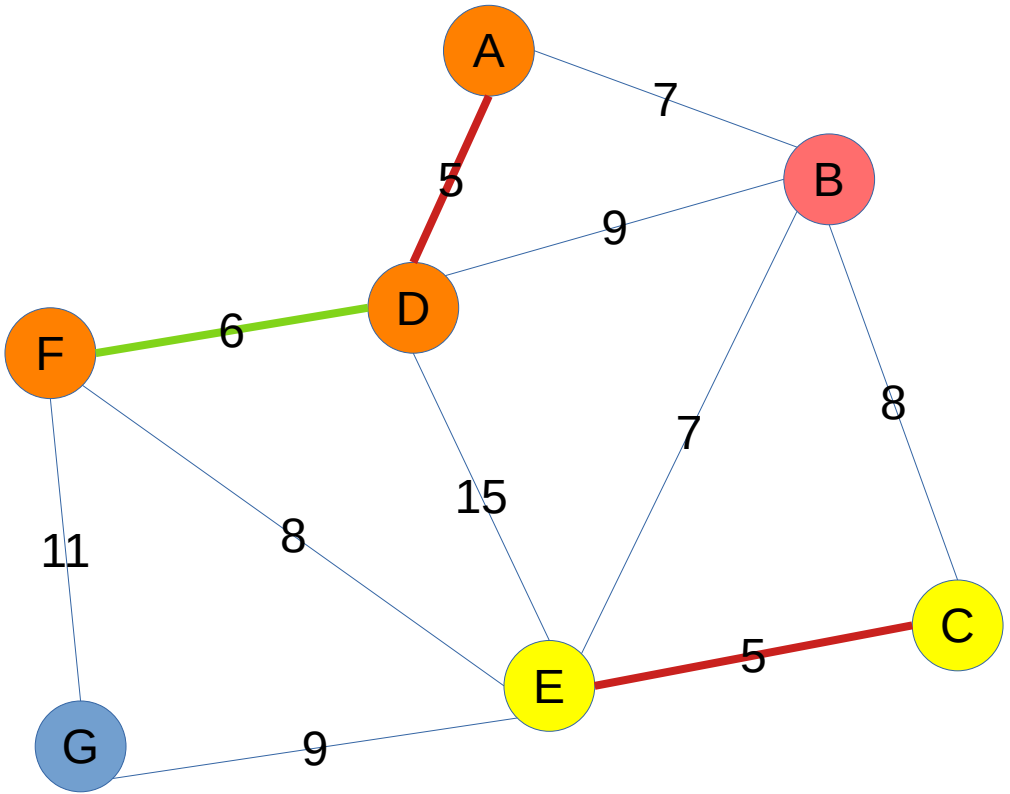
A – D (5)
C – E (5)
D – F (6)
A – B (7)
B – E (7)
F – E (8)
B – C (8)
B – D (9)
E – G (9)
F – G (11)

Classes d'équivalence :

{A, D}
{B}
{C, E}
{F}
{G}

Graphe repris de https://fr.wikipedia.org/wiki/Algorithme_de_Kruskal

Kruskal : déroulement



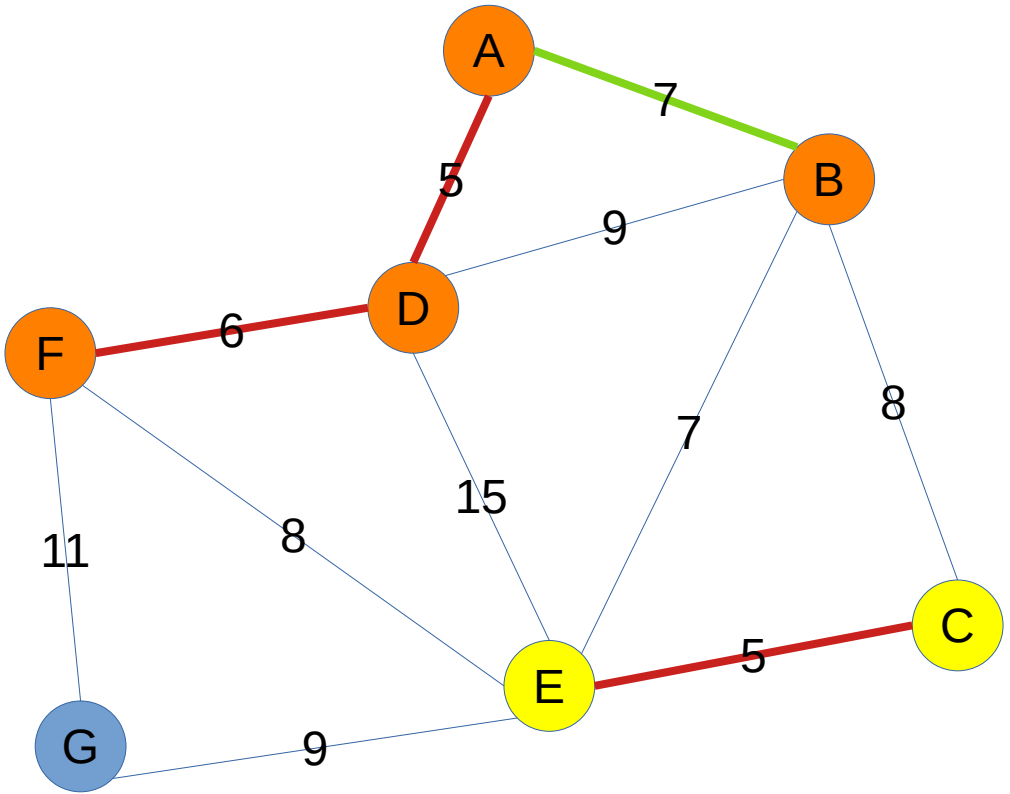
A – D (5)
C – E (5)
D – F (6)
A – B (7)
B – E (7)
F – E (8)
B – C (8)
B – D (9)
E – G (9)
F – G (11)

Classes d'équivalence :

{A, D, F}
{B}
{C, E}
{G}

Graphe repris de https://fr.wikipedia.org/wiki/Algorithme_de_Kruskal

Kruskal : déroulement



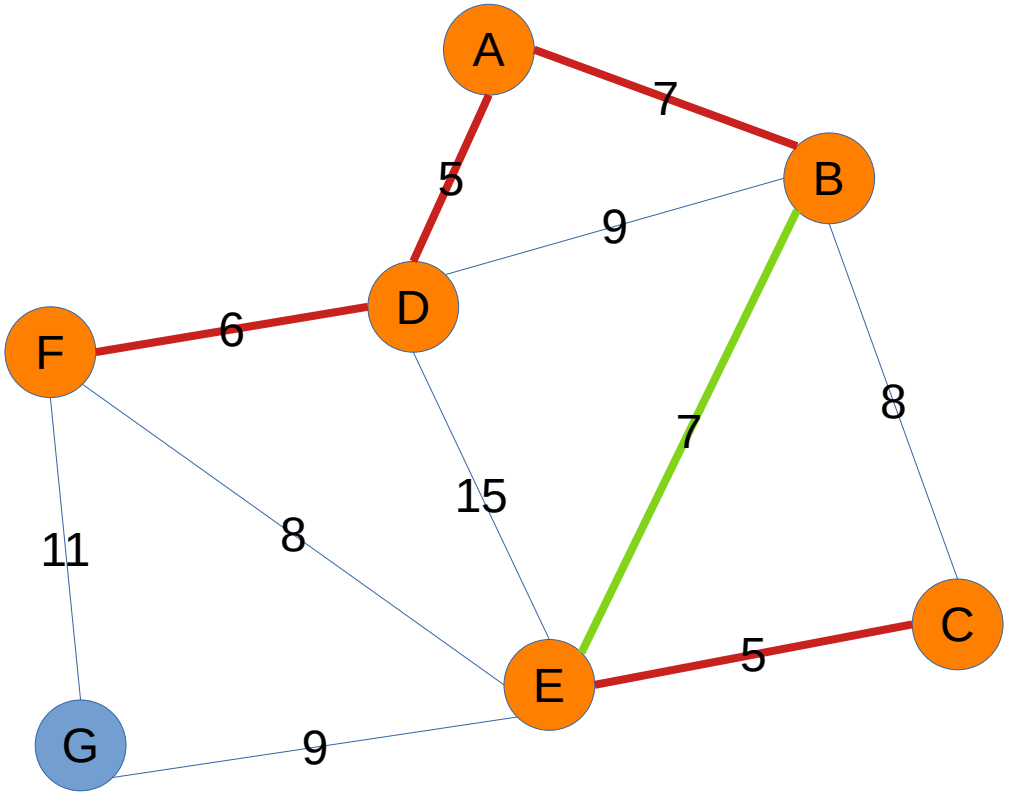
A – D (5)
C – E (5)
D – F (6)
A – B (7)
B – E (7)
F – E (8)
B – C (8)
B – D (9)
E – G (9)
F – G (11)

Classes d'équivalence :

{A, D, F, B}
{C, E}
{G}

Graphe repris de https://fr.wikipedia.org/wiki/Algorithme_de_Kruskal

Kruskal : déroulement

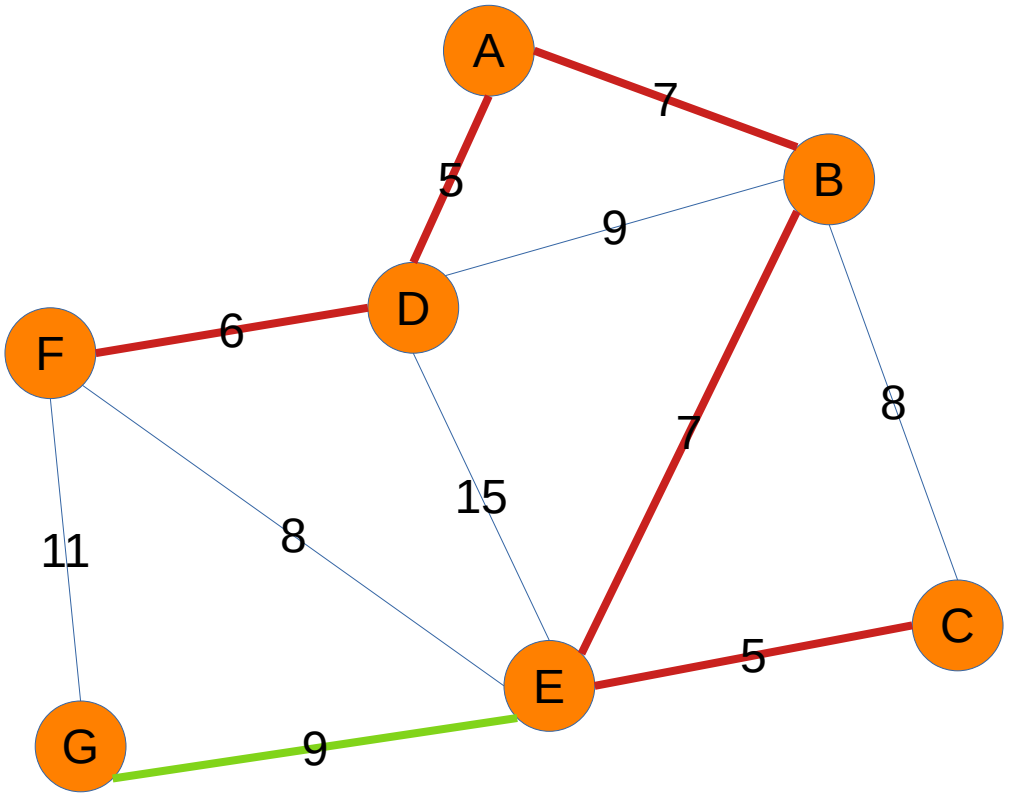


A – D (5)
C – E (5)
D – F (6)
A – B (7)
B – E (7)
F – E (8)
B – C (8)
B – D (9)
E – G (9)
F – G (11)

Classes d'équivalence :
{A, D, F, B, C, E}
{G}

Graphe repris de https://fr.wikipedia.org/wiki/Algorithme_de_Kruskal

Kruskal : déroulement



A – D (5)
C – E (5)
D – F (6)
A – B (7)
B – E (7)
F – E (8)
B – C (8)
B – D (9)
E – G (9)
F – G (11)

Classes d'équivalence :
{A, D, F, B, C, E, G}

Graphe repris de https://fr.wikipedia.org/wiki/Algorithme_de_Kruskal

Gestion efficaces des classes d’équivalence : la stucture Union-Find

- La base
  - On utilise une forêt (1 arbre par classe)
  - La racine de chaque arbre est le représentant de la classe
  - Lors de l’union de 2 classes, on fusionne les arbres
- Pour améliorer l’efficacité
  - Chaque représentant a un « rang » (initialisé à 0)
  - Lors de la fusion :
    - si les rangs sont différents, c’est la racine de plus petit rang qui est adoptée
    - Si les rangs sont identiques, on choisit l’une des racines et on augmente son rang de 1
    - À chaque recherche du représentant, on met à jour le parent

Application sur l’exemple

Classes d’équivalence :

{A}
{B}
{C}
{D}
{E}
{F}
{G}



Application sur l’exemple

Classes d’équivalence :

{A, D}
{B}
{C}
{E}
{F}
{G}



Application sur l’exemple

Classes d’équivalence :

{A, D}

{B}

{C, E}

{F}

{G}



Application sur l’exemple

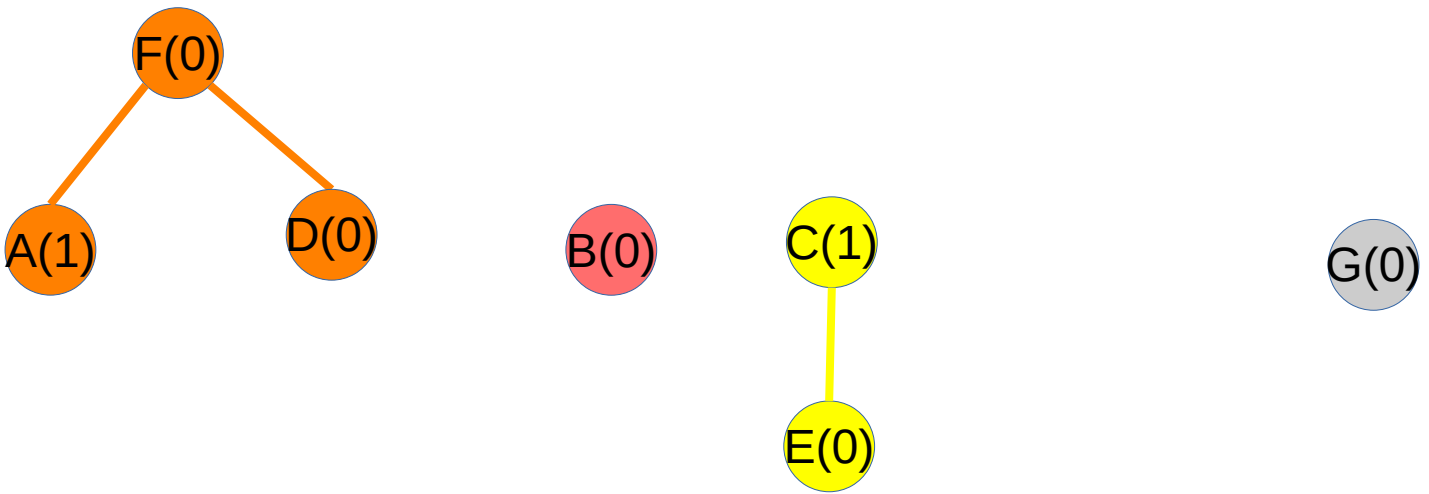
Classes d’équivalence :

{A, D, F}

{B}

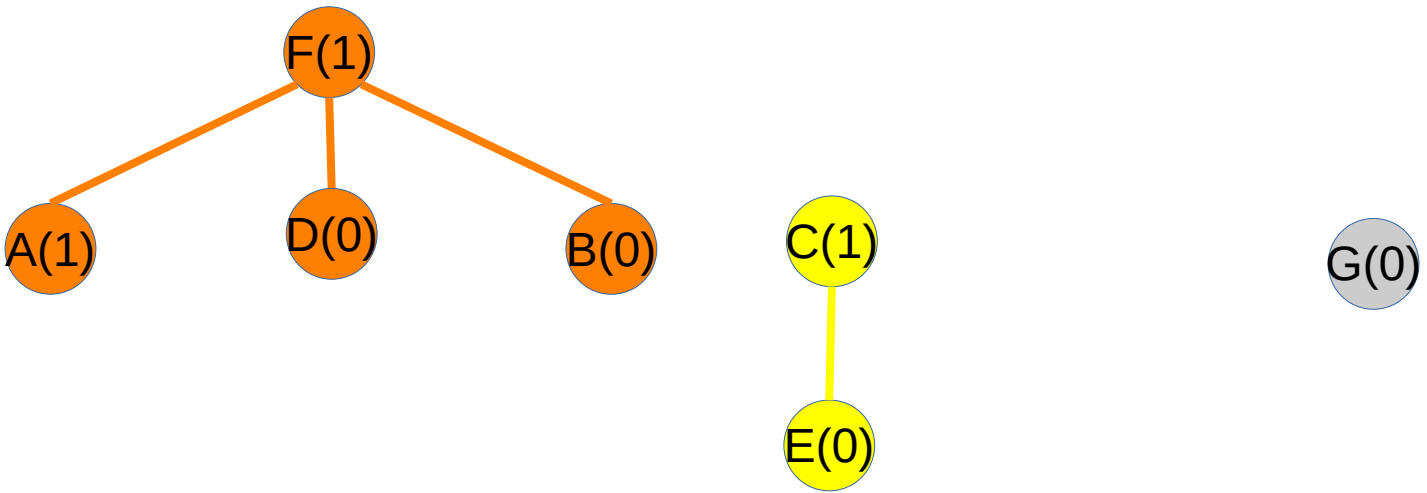
{C, E}

{G}



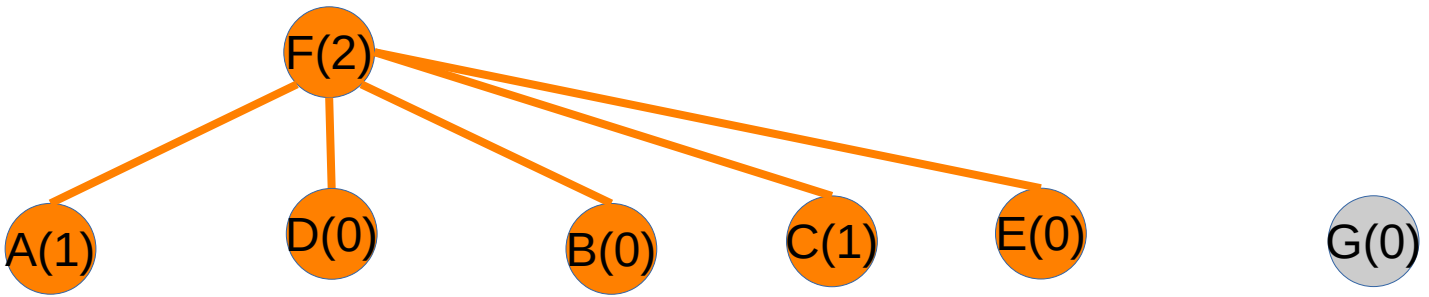
Application sur l’exemple

Classes d’équivalence :
{A, D, F, B}
{C, E}
{G}

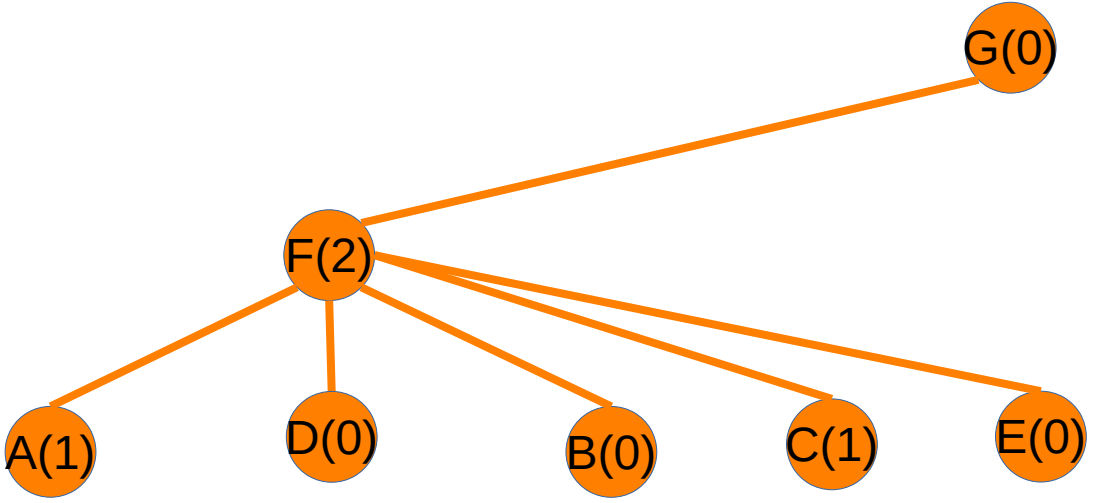


Application sur l’exemple

Classes d’équivalence :
{A, D, F, B, C, E}
{G}



Application sur l’exemple



Classes d’équivalence :
{A, D, F, B, C, E, G}

Pour aller plus loin

- Sujet Amérique du nord n°1 2025
- Galéjade
- Algorithmes fourmis (exemple dans NetLogo)