Arbres et graphes

Arbres, domaines d'utilisation

Ce sont des structures fondamentales utilisées dans de nombreux domaines :

- Informatique
- Sciences sociales
- Classification et analyse de données
- Théorie des questionnaires
- Recherche opérationnelle
- Intelligence artificielle
- Optimisation combinatoire
- Théorie des réseaux électriques

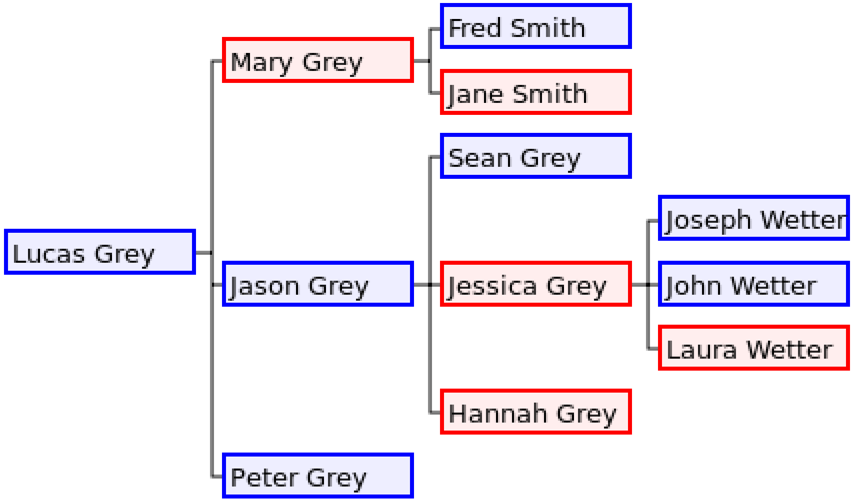
Arbres, définition

- Un arbre :
  un graphe non orienté, connexe, simple et sans cycle.

- Un graphe sans cycle qui **n'est pas connexe :**
  une **forêt**

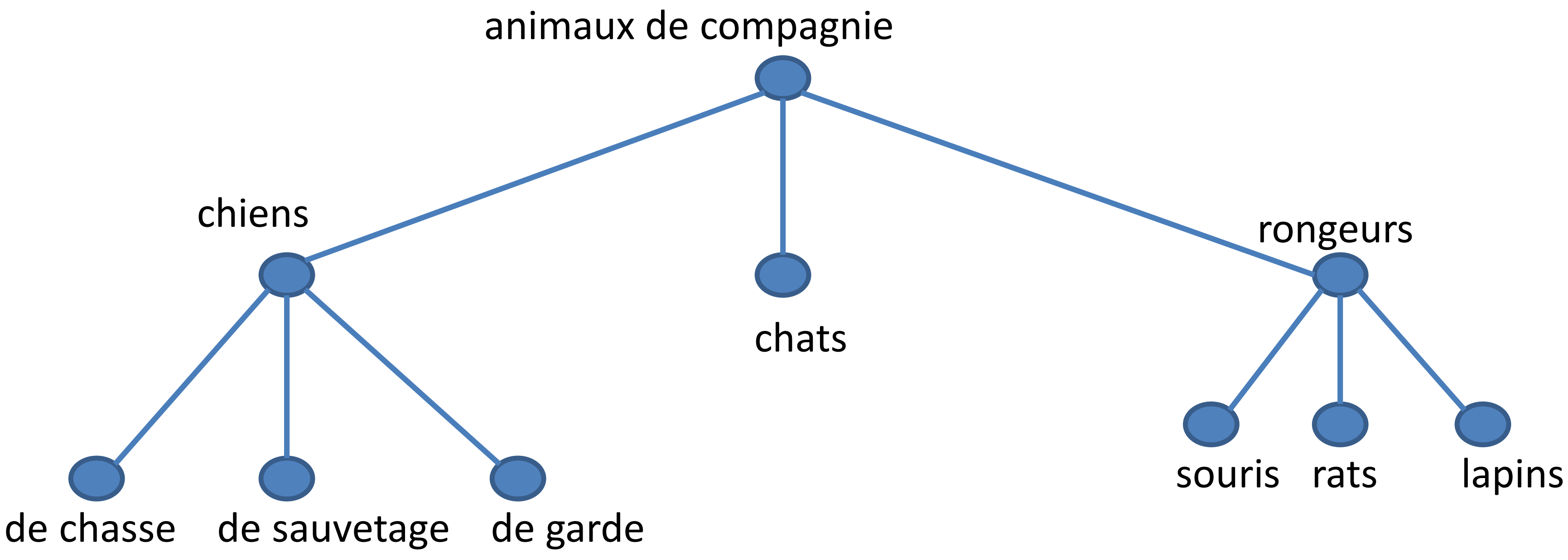
Arbres, exemples

- **<u>Exemple 1</u>** : Arbre généalogique d'une famille
-
    - sommets – membres de la famille
    - arêtes – liens de parenté



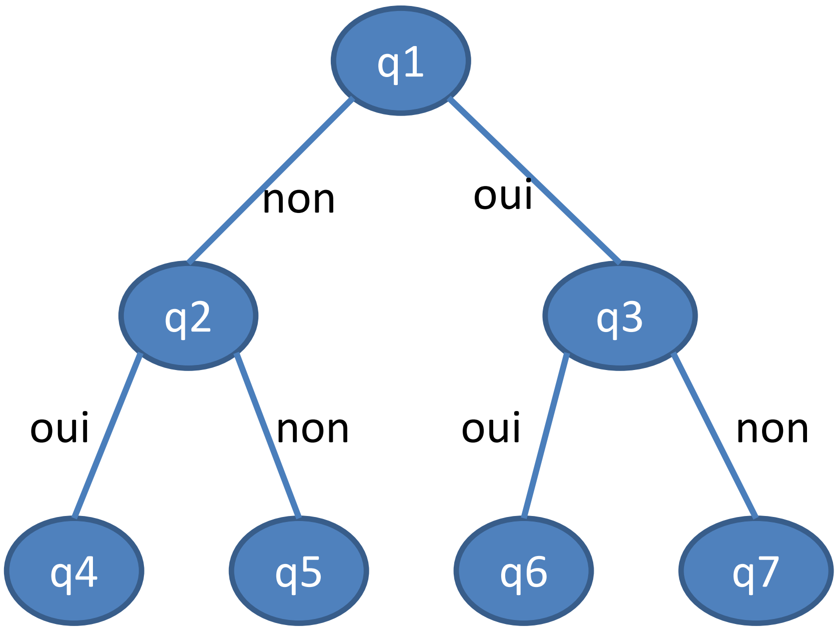
Arbres, exemples

<u>Exemple 2</u> : **Arbre de classification**



Arbres, exemples

Exemple 3 : Questionnaire



Arbre couvrant de poids minimum

Le problème de l’arbre de poids minimum

Soit G = <S, A> un graphe non orienté valué, et on note w(u) le poids de l’arête u $\in$ A.

Soit G′ = <S, A′> un graphe partiel quelconque de G.
(Rappel: un graphe partiel contient tous les sommets de G, mais pas toutes les arêtes).

On appelle poids de G′ la somme des poids de ses arêtes : $w\left(G'\right) = \sum_{u \in A'} w(u)$

Parmi les graphes partiels de G il y a des arbres : si G′ est connexe sans cycle, c’est un arbre. Plus précisément, c’est une **arbre couvrant.**

On veut trouver l’arbre couvrant de poids minimum τ^* sur l’ensemble de tous les arbres couvrants de G :

$w(\tau^*) = \min \{w(\tau)\}$.

Algorithme de Kruskal (1956)
$1^{ère}$ version (constructive)

Joseph Bernard Kruskal, Jr. (1928 – 2010)
était un mathématicien, statisticien, chercheur en informatique et psychométricien américain.

Principe

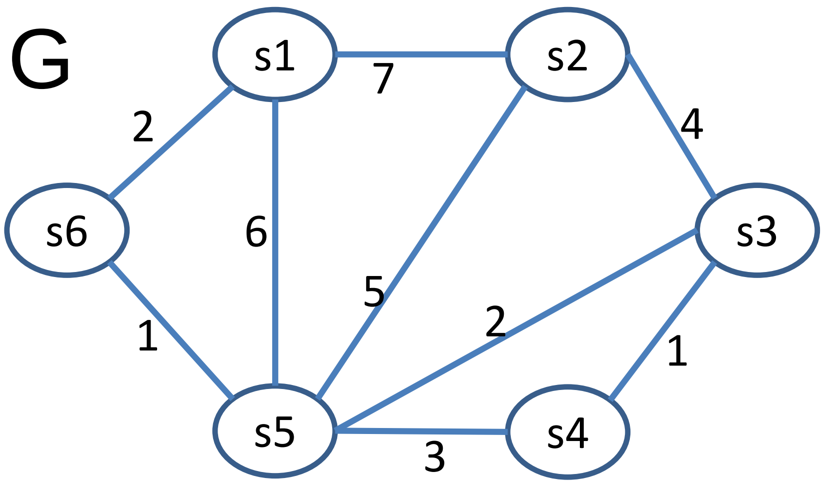
Soit G = <S, A> un graphe non orienté valué et connexe, de ***n*** sommets et ***p*** arêtes.

On part d'un graphe T vide.

On ajoute à T des arêtes de G, une par une, en choisissant à chaque étape, parmi les arêtes qui ne sont pas dans T, une arête de coût minimum, à condition qu'elle ne forme pas de cycle avec les arêtes qui sont déjà dans T.

Lorsqu'on a ajouté n-1 arêtes, sans créer de cycle, on a obtenu un arbre de recouvrement minimum (arbre couvrant de poids minimum).

Kruskal constructif : exemple



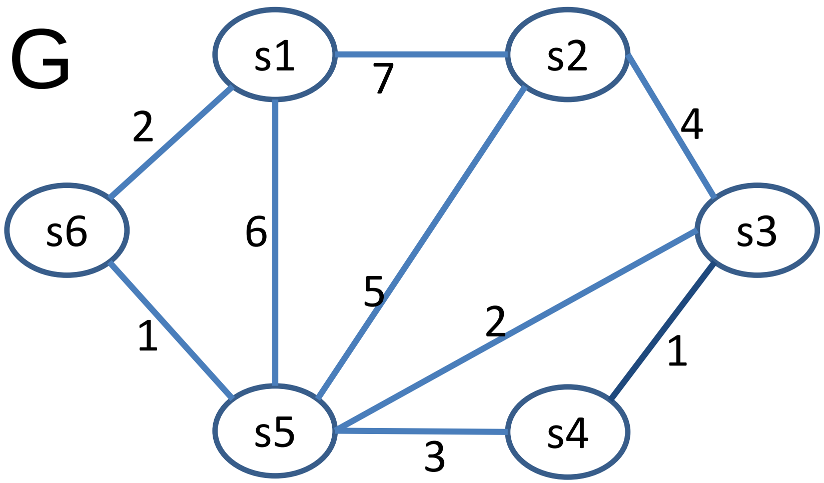
T

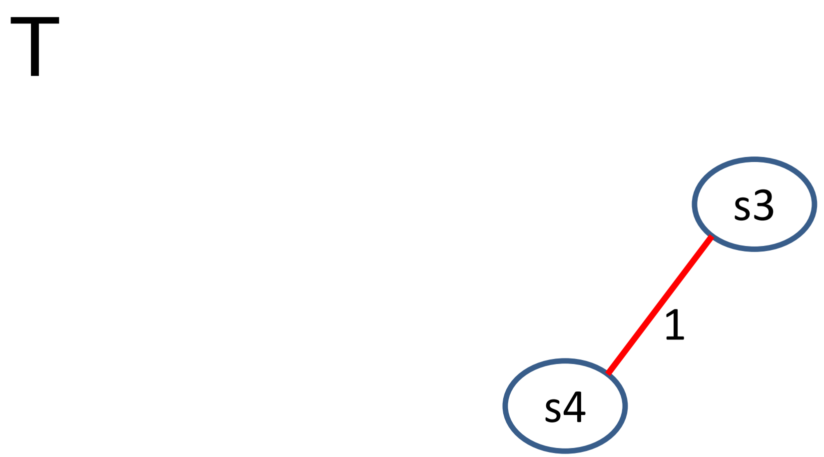
Arête	Poids
s5-s6	1
s3-s4	1
s1-s6	2
s3-s5	2
s4-s5	3
s2-s3	4
s2-s5	5
s1-s5	6
s1-s2	7

<u>Initialisation</u>

U = toutes les arêtes; T : graphe vide

Kruskal constructif : exemple





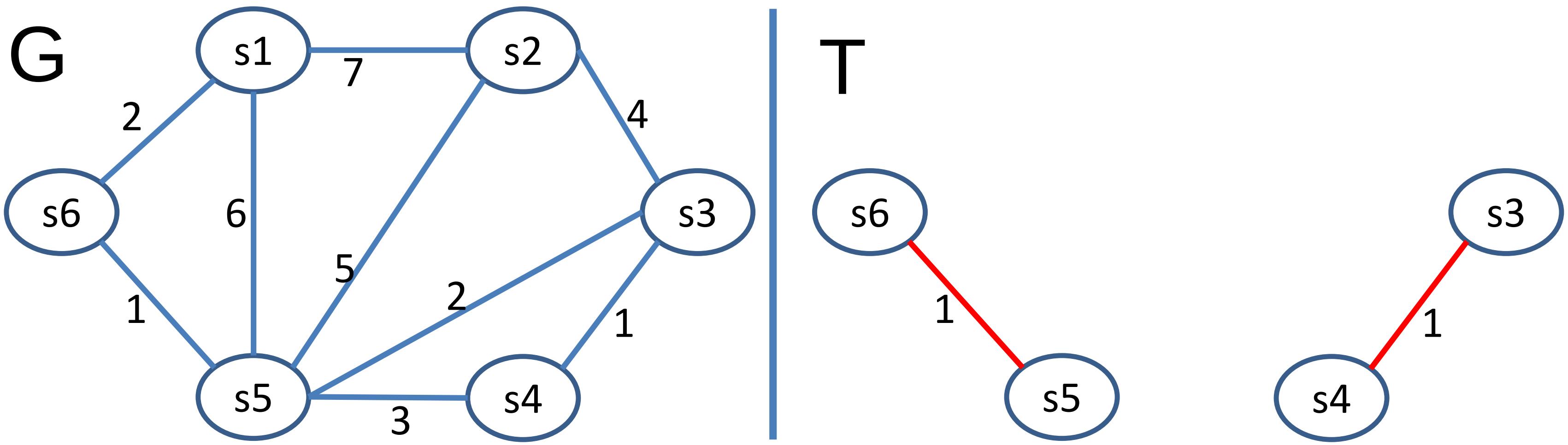
Arête	Poids
s5-s6	1
s3-s4	1
s1-s6	2
s3-s5	2
s4-s5	3
s2-s3	4
s2-s5	5
s1-s5	6
s1-s2	7

Itération 1: Arête(s) de poids minimum sur U: (s5,s6,1) et (s3, s4,1).

Choisissons (s3, s4,1) d'abord. U = U \ {(s3, s4,1)}.
Le nombre d'arêtes de T : i=1.

U: arêtes marqués en noir dans la table.

Kruskal constructif : exemple



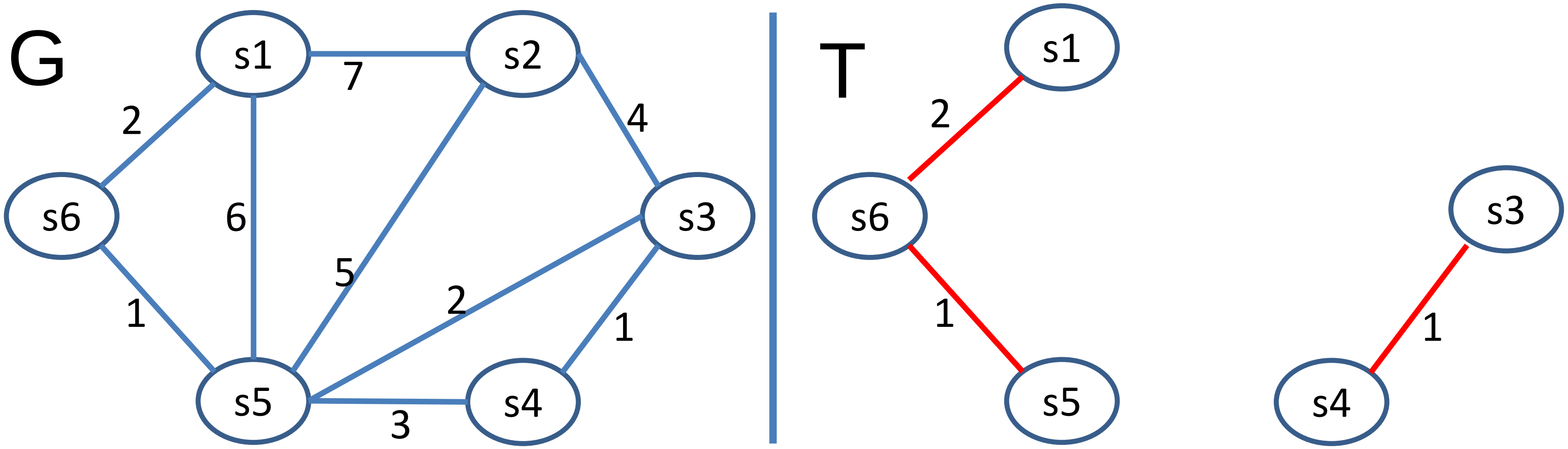
Arête	Poids
s5-s6	1
s3-s4	1
s1-s6	2
s3-s5	2
s4-s5	3
s2-s3	4
s2-s5	5
s1-s5	6
s1-s2	7

Itération 2: Arête(s) de poids minimum sur U: (s5,s6,1).

U = U \ {(s5, s6,1)}.
Le nombre d'arêtes de T : i=2.

U: arêtes marqués en noir dans la table.

Kruskal constructif : exemple



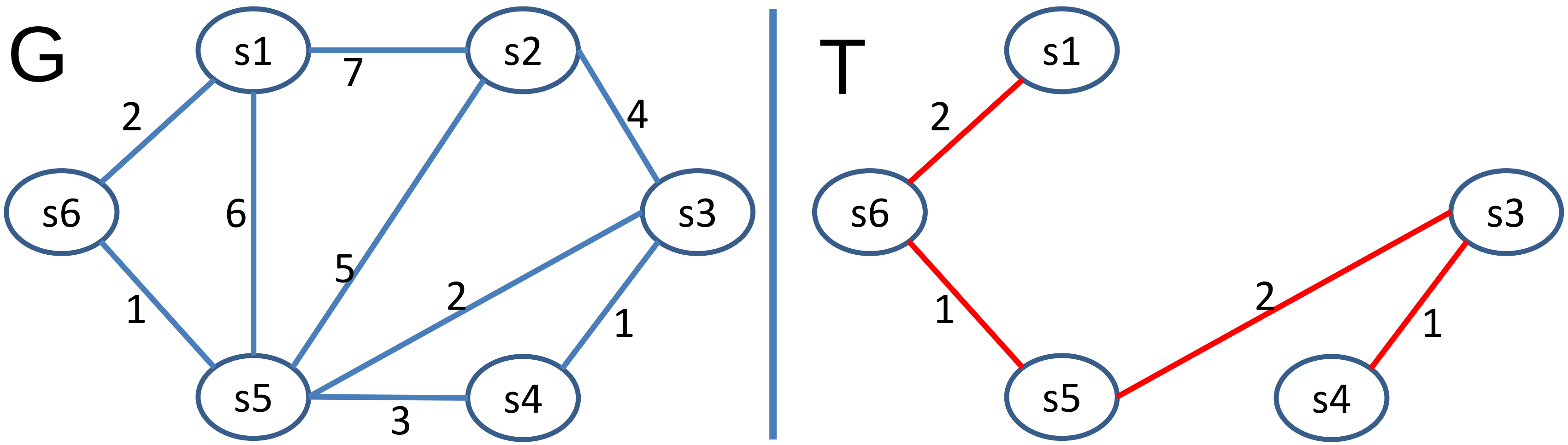
Arête	Poids
s5-s6	1
s3-s4	1
s1-s6	2
s3-s5	2
s4-s5	3
s2-s3	4
s2-s5	5
s1-s5	6
s1-s2	7

Itération 3: Arête(s) de poids minimum sur U: (s1,s6,2), (s3,s5,2).

Choisissons (s1, s6,2) d'abord. U = U \ {(s1, s6,2)}.
Le nombre d'arêtes de T : i=3.

U: arêtes marqués en noir dans la table.

Kruskal constructif : exemple



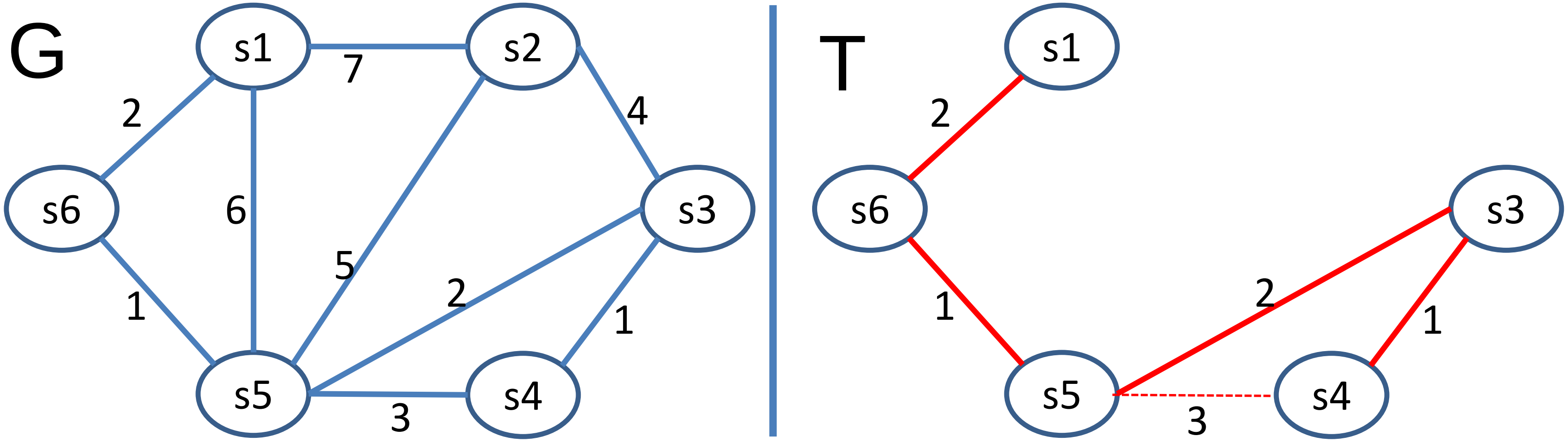
Arête	Poids
s5-s6	1
s3-s4	1
s1-s6	2
s3-s5	2
s4-s5	3
s2-s3	4
s2-s5	5
s1-s5	6
s1-s2	7

Itération 4: Arête(s) de poids minimum sur U: (s3,s5,2).

U = U \ {(s3, s5,2)}.
Le nombre d'arêtes de T : i=4.

U: arêtes marqués en noir dans la table.

Kruskal constructif : exemple



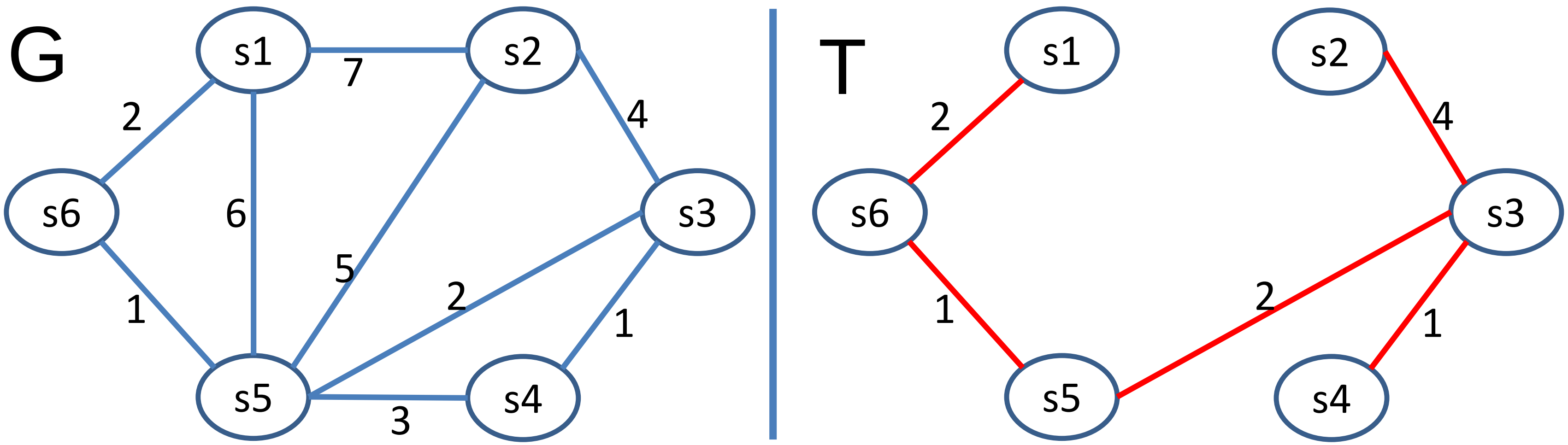
Arête	Poids
s5-s6	1
s3-s4	1
s1-s6	2
s3-s5	2
~~s4-s5~~	~~3~~
s2-s3	4
s2-s5	5
s1-s5	6
s1-s2	7

Itération 5: Arête(s) de poids minimum sur U: (s4,s5,3).

On rejette cette arête car elle créerait un cycle, mais on ne la laisse pas dans U : U = U \ {(s4, s5, 3)}.
Le nombre d'arêtes de T reste i=4.

U: arêtes marqués en noir dans la table.

Kruskal constructif : exemple



Arête	Poids
s5-s6	1
s3-s4	1
s1-s6	2
s3-s5	2
~~s4-s5~~	~~3~~
s2-s3	4
s2-s5	5
s1-s5	6
s1-s2	7

Itération 6: Arête(s) de poids minimum sur U: (s2,s3,4).

U = U \ {(s2, s3,4)}.
Le nombre d'arêtes de T : i=5. C'est le nombre d'arêtes constituant un arbre couvrant pour un graphe à 6 sommets. L'arbre de recouvrement minimum T est construit. Son poids est 1+1+2+2+4 = 10.

Algorithme de Kruskal (1956)
2^{ième} version (destructive)

<u>Principe</u>

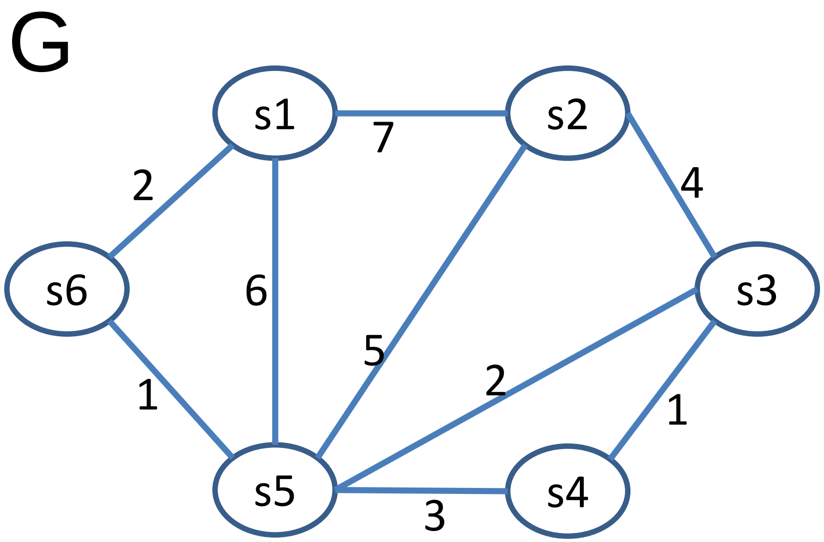
Soit G = <S, A> un graphe non orienté valué et connexe, de ***n*** sommets et ***p*** arêtes.

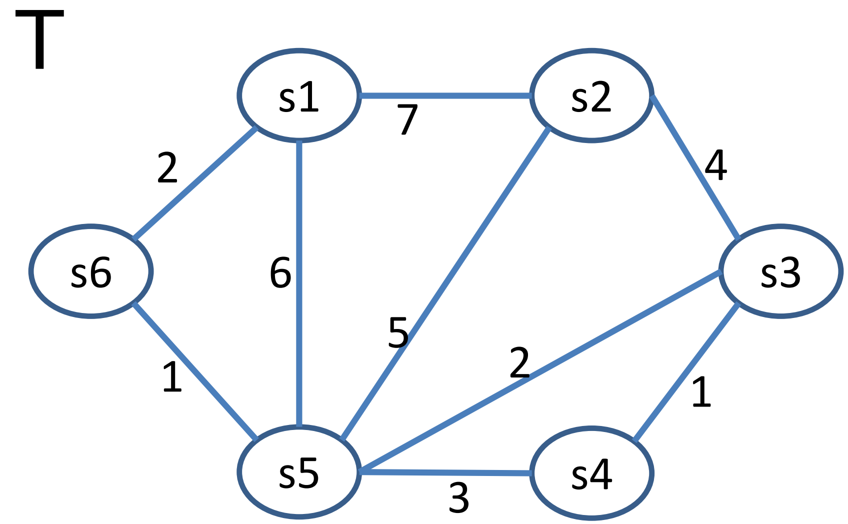
On part d'un graphe T = G

On retire de T des arêtes de G, une par une, en choisissant à chaque étape, parmi les arêtes qui ne sont pas dans T, une arête de coût maximum, de façon que T reste connexe.

Lorsque T possède n-1 arêtes, on a obtenu un arbre de poids minimum (arbre de recouvrement minimum).

Kruskal destructif : exemple





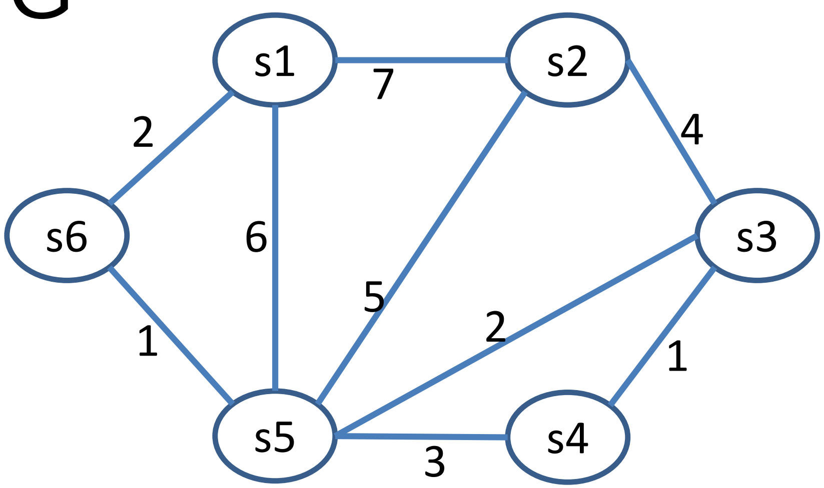
Arête	Poids
s1-s2	7
s1-s5	6
s2-s5	5
s2-s3	4
s4-s5	3
s1-s6	2
s3-s5	2
s5-s6	1
s3-s4	1

<u>Initialisation</u>

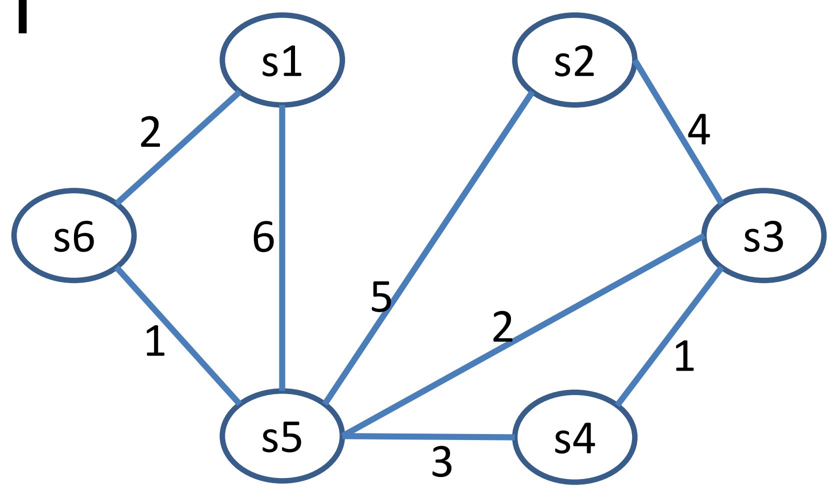
T = G

Kruskal destructif : exemple

G



T

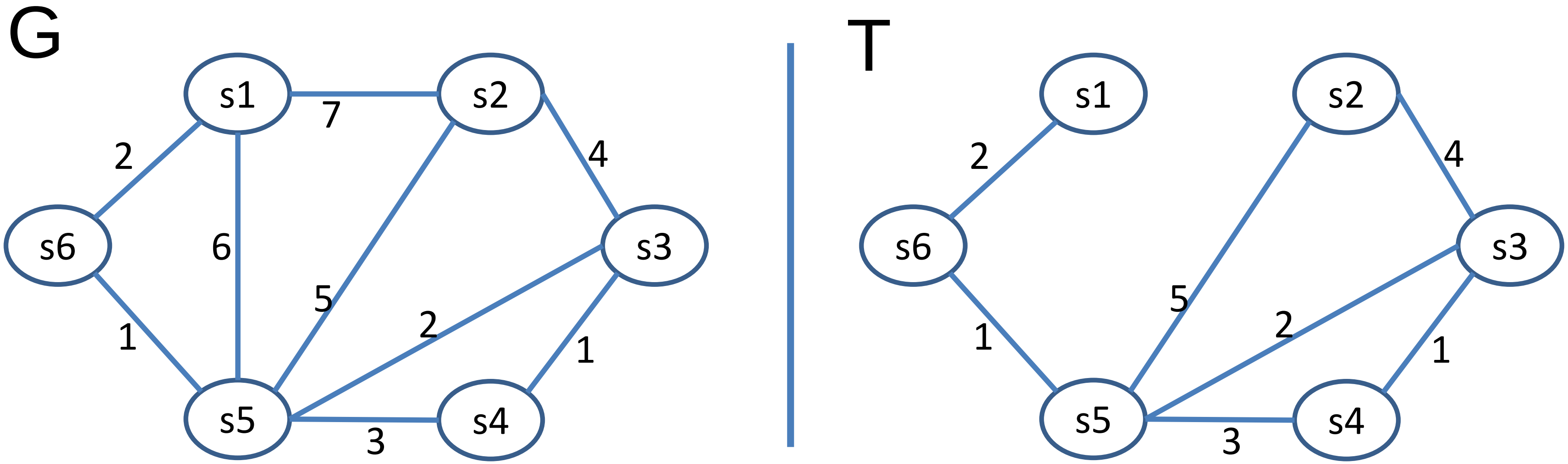


Arête	Poids
s1-s2	7
s1-s5	6
s2-s5	5
s2-s3	4
s4-s5	3
s1-s6	2
s3-s5	2
s5-s6	1
s3-s4	1

<u>Itération 1 :</u>

On retire l'arête (s1, s2, 7)

Kruskal destructif : exemple

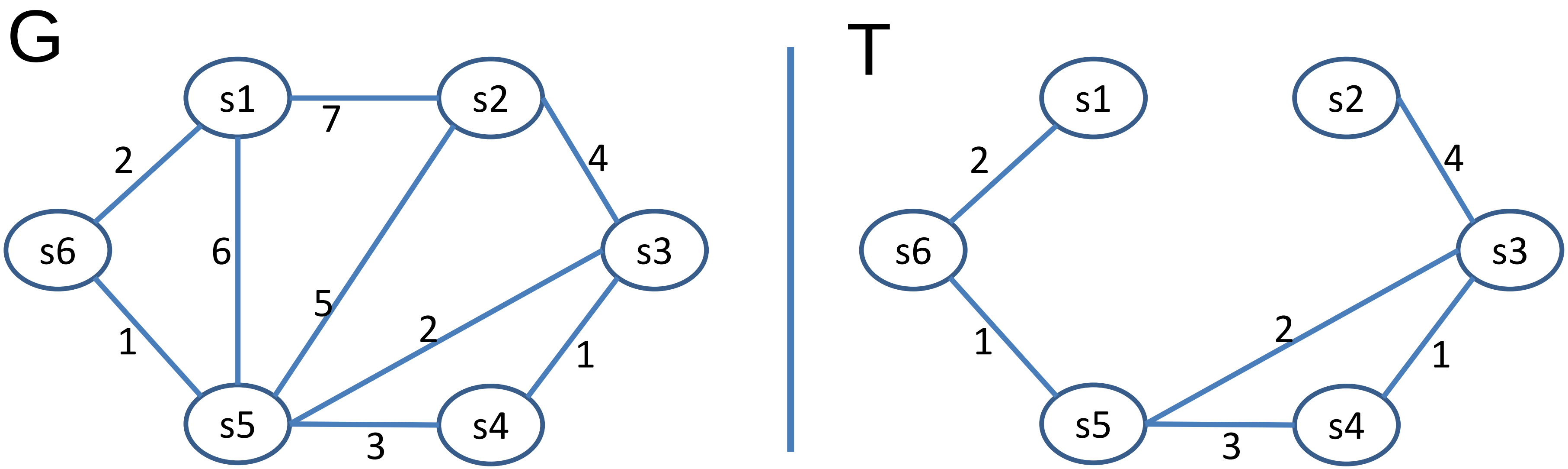


Arête	Poids
s1-s2	7
s1-s5	6
s2-s5	5
s2-s3	4
s4-s5	3
s1-s6	2
s3-s5	2
s5-s6	1
s3-s4	1

Itération 2 :

On retire l'arête (s1, s5, 6)

Kruskal destructif : exemple



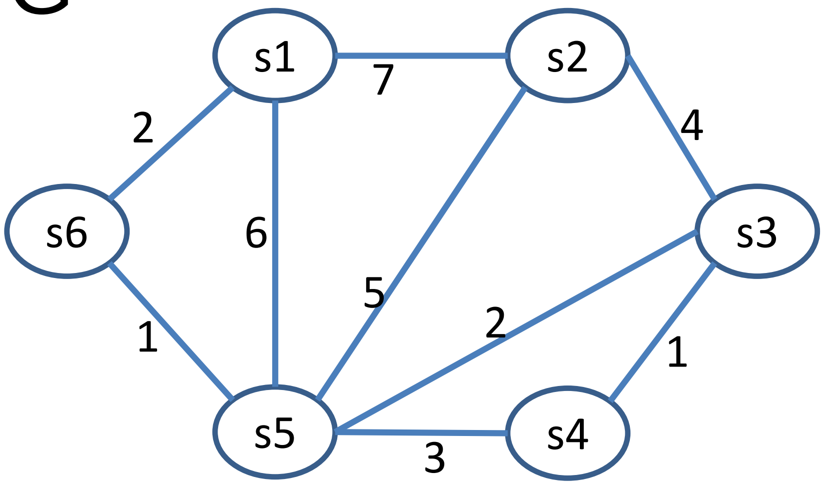
Arête	Poids
s1-s2	7
s1-s5	6
s2-s5	5
s2-s3	4
s4-s5	3
s1-s6	2
s3-s5	2
s5-s6	1
s3-s4	1

<u>Itération 3 :</u>

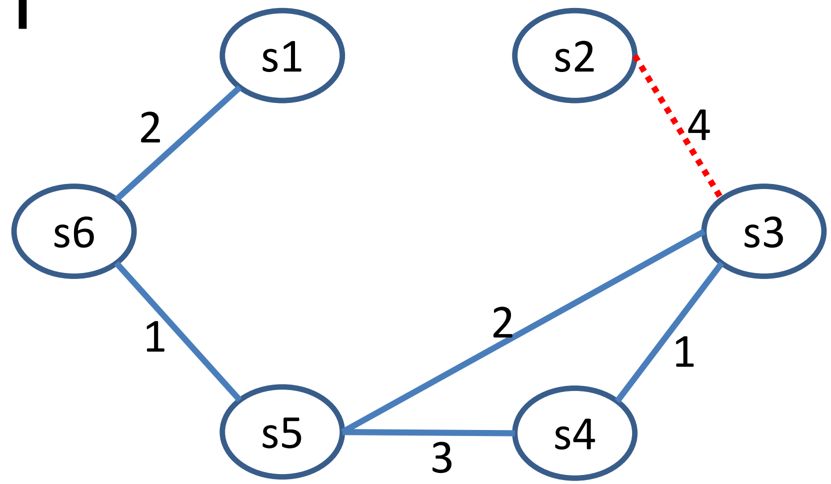
On retire l’arête (s2, s5, 5)

Kruskal destructif : exemple

G



T

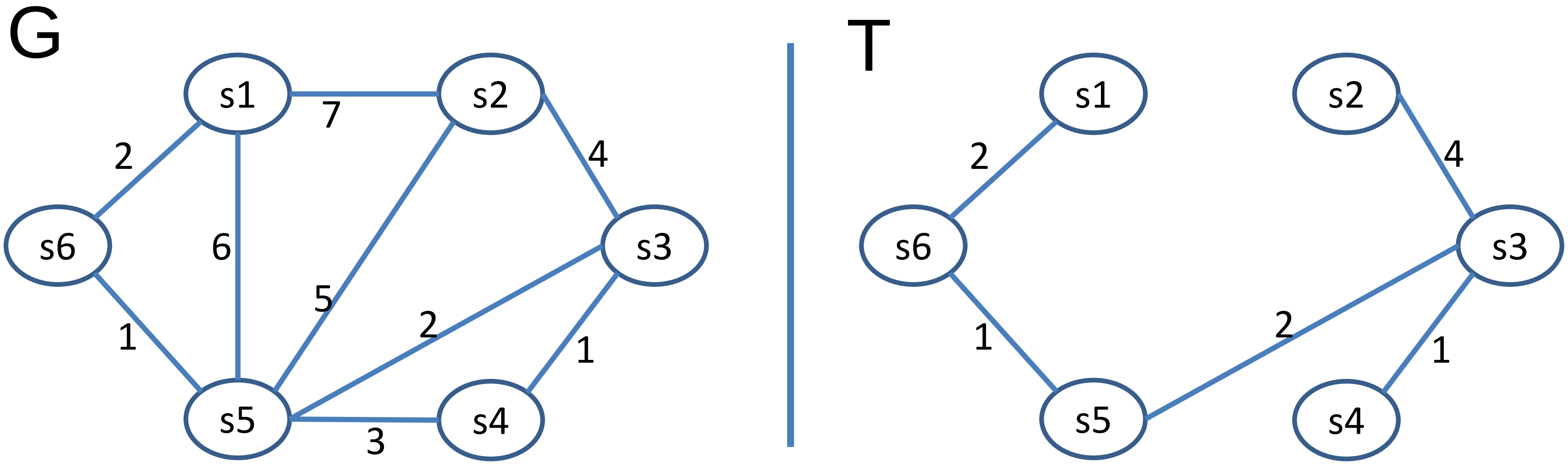


Arête	Poids
s1-s2	7
s1-s5	6
s2-s5	5
s2-s3	4
s4-s5	3
s1-s6	2
s3-s5	2
s5-s6	1
s3-s4	1

<u>Itération 4 :</u>

On essaie de retirer l'arête (s2, s3, 4). Or cela crée un graphe non connexe, et on ne le fait pas : l'arête reste dans la liste des arêtes de T.

Kruskal destructif : exemple



Arête	Poids
s1-s2	7
s1-s5	6
s2-s5	5
s2-s3	4
s4-s5	3
s1-s6	2
s3-s5	2
s5-s6	1
s3-s4	1

Itération 5 :

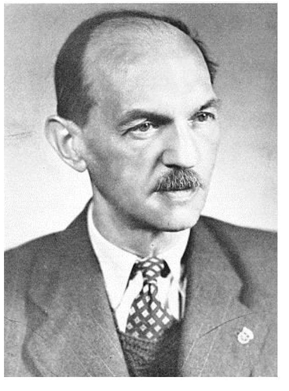
On retire l'arête (s4, s5, 3).
Le graphe T possède 5 arêtes pour 6 sommets, on arrête les itérations.
Le poids de l'arbre de recouvrement obtenu est 10.

Algorithme de Prim (1957)

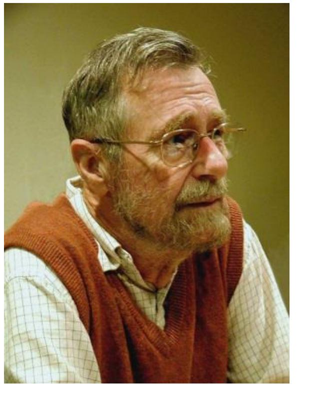
Robert Clay Prim (né en 1921 à Sweetwater, Texas) est un mathématicien et informaticien américain.

Développé en 1930 par un mathématicien tchèque **Vojtěch Jarník** et plus tard, indépendamment, par **Robert C. Prim** en 1957, et puis redécouvert en 1959 par **Edsger Dijkstra**. C'est pourquoi il est parfois appelé **algorithme DPJ**, **algorithme de Jarník,** ou **algorithme de Prim-Jarník.**

Prim



Jarník



Dijkstra

Algorithme de Prim, principe : faire pousser un arbre

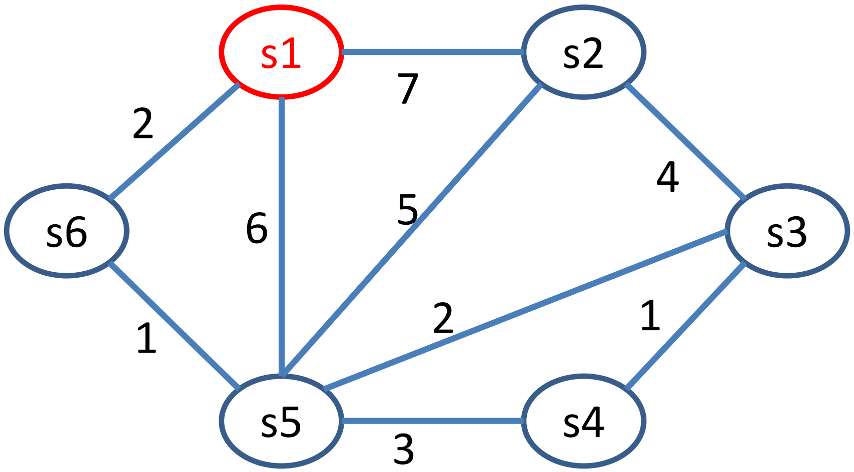
Principe voisin de celui de l'algorithme de Dijkstra :

A partir d'un graphe non orienté valué G = <S, A> on construit un arbre T en faisant grandir un ensemble CC de sommets appartenant à T.

À chaque itération, une nouvelle arête est ajoutée à T : on choisit l'arête de poids minimum parmi toutes les arêtes dont une extrémité appartient à T et l'autre non. De cette façon, à chaque itération un nouveau sommet est ajouté à l'arbre T et donc à l'ensemble CC.

Quand touts les sommets sont dans CC, on a terminé.

Algorithme de Prim, exemple

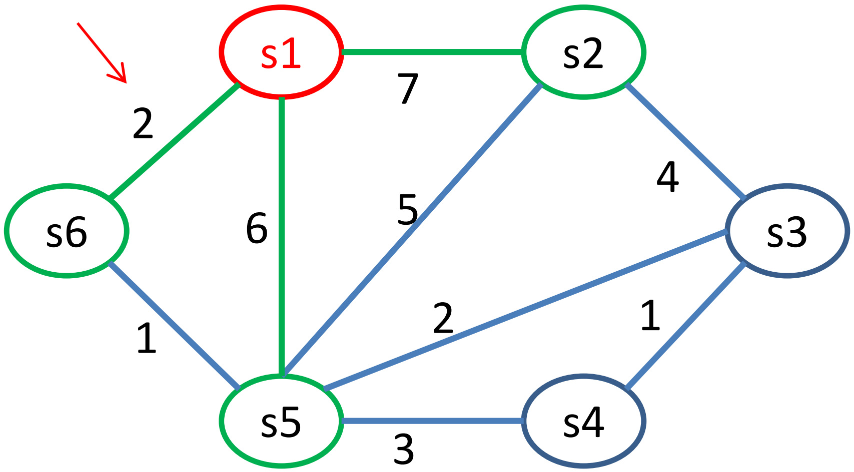


1. Initialisation: On choisit s1 comme racine.

CC={s1}, M ={s2,s3,s4,s5,s6}.

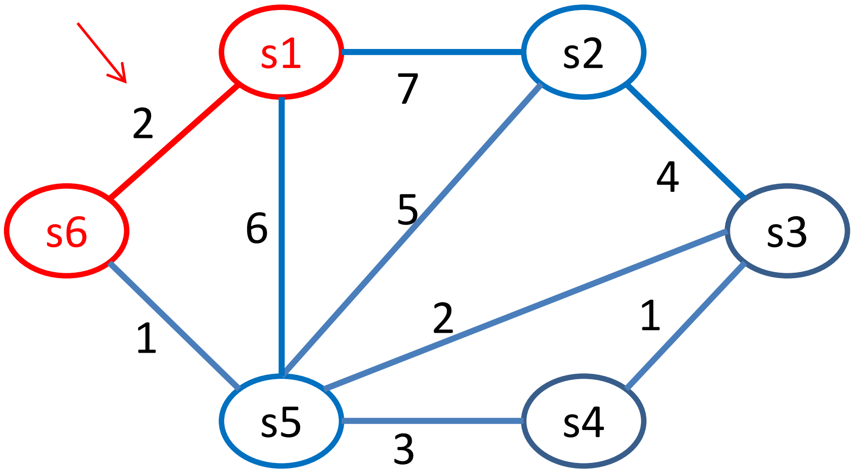
L'ensemble A' des arêtes de l'arbre T est vide : A' = ∅.

Algorithme de Prim, exemple



2a. Les sommets adjacents aux sommets de CC (actuellement, à s1) : s2, s5, s6. Les longueurs des arêtes dont une extrémité est dans CC, et l'autre dans M, sont: |(s1,s2)|=7, |(s1,s5)|=6, |(s1,s6)|=2. Le poids minimal est |(s1,s6)|=2.

Algorithme de Prim, exemple

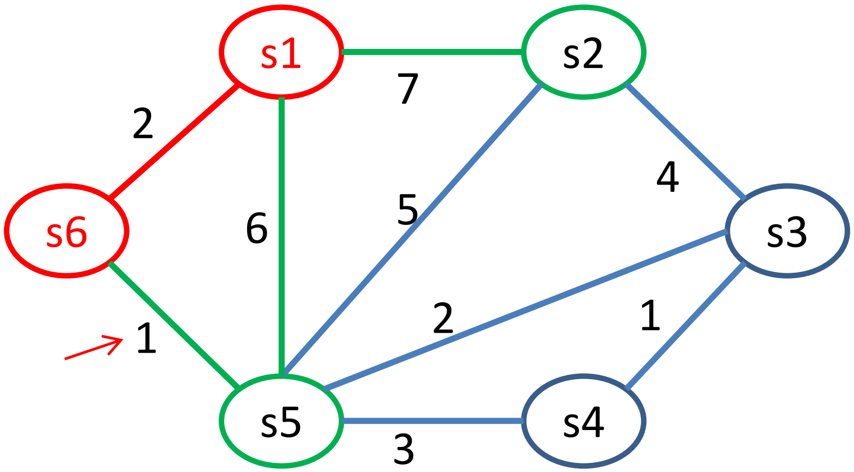


2b. Les sommets adjacents aux sommets de CC (actuellement, à s1) : s2, s5, s6. Les longueurs des arêtes dont une extrémité est dans CC, et l'autre dans M, sont: |(s1,s2)|=7, |(s1,s5)|=6, |(s1,s6)|=2. Le poids minimal est |(s1,s6)|=2.

On met le sommet s6 dans CC, et on ajoute (s1, s6) à A'.

CC={s1,s6}, M ={s2,s3,s4,s5}, A' = {(s1,s6)}.

Algorithme de Prim, exemple

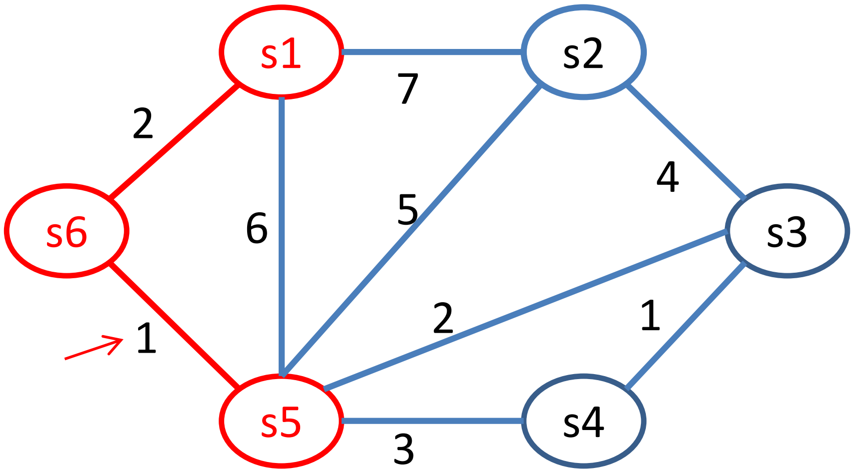


3a. Les sommets adjacents aux sommets de CC : s2, s5.

Les longueurs des arêtes dont une extrémité est dans CC, et l'autre dans M, sont:

|(s1,s2)|=7, |(s1,s5)|=6, |(s6,s5)|=1. Le poids minimal est |(s6,s5)|=1.

Algorithme de Prim, exemple



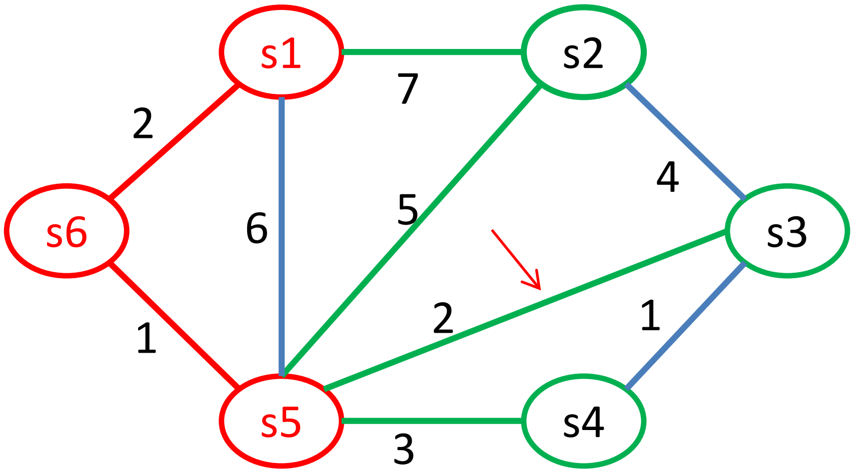
3b. Les sommets adjacents aux sommets de CC : s2, s5.

Les longueurs des arêtes dont une extrémité est dans CC, et l'autre dans M, sont: |(s1,s2)|=7, |(s1,s5)|=6, |(s6,s5)|=1. Le poids minimal est |(s6,s5)|=1.

On met le sommet s5 dans CC, et on ajoute (s6, s5) à A'.

CC={s1,s5,s6}, M ={s2,s3,s4}, A' = {(s1,s6), (s6,s5)}.

Algorithme de Prim, exemple

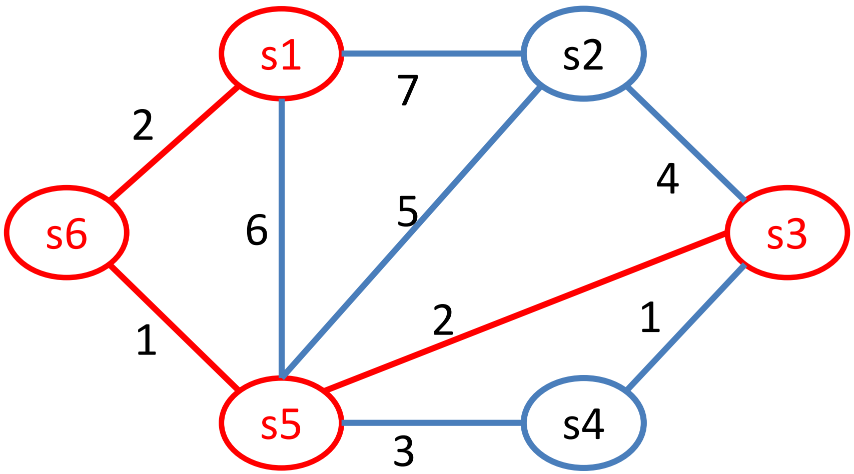


4a. Les sommets adjacents aux sommets de CC : s2, s3, s4.

Les longueurs des arêtes dont une extrémité est dans CC, et l'autre dans M, sont:

|(s1,s2)|=7, |(s5,s2)|=5, |(s5,s3)|=2, |(s5,s4)|=3. Le poids minimal est |(s5,s3)|=2.

Algorithme de Prim, exemple



4b. Les sommets adjacents aux sommets de CC : s2, s3, s4.

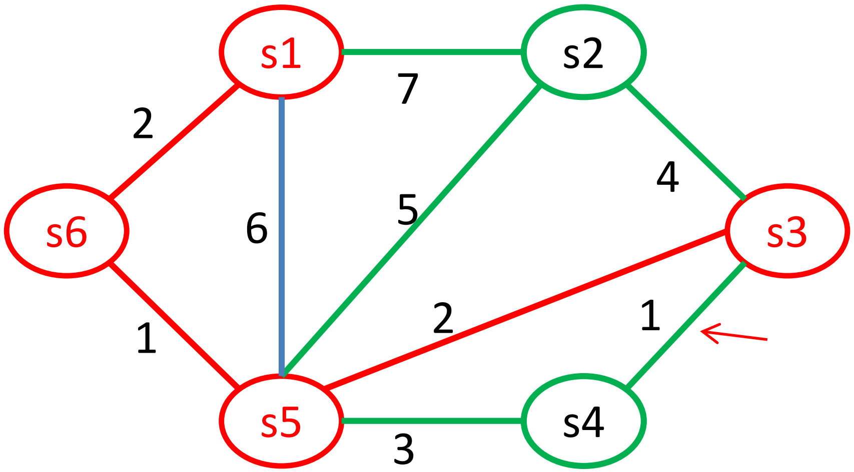
Les longueurs des arêtes dont une extrémité est dans CC, et l'autre dans M, sont:

|(s1,s2)|=7, |(s5,s2)|=5, |(s5,s3)|=2, |(s5,s4)|=3. Le poids minimal est |(s5,s3)|=2.

On met le sommet s3 dans CC, et on ajoute (s5, s3) à A'.

CC={s1,s3,s5,s6}, M ={s2,s4}, A' = {(s1,s6), (s6,s5), (s5,s3)}.

Algorithme de Prim, exemple



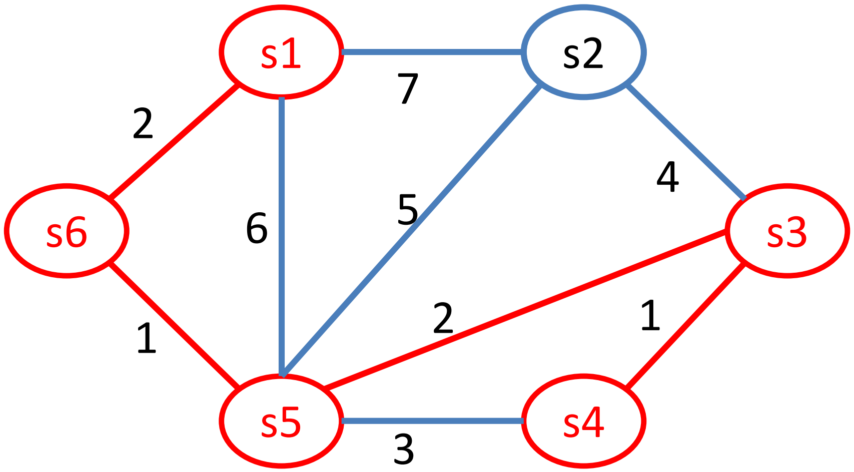
5a. Les sommets adjacents aux sommets de CC : s2, s4.

Les longueurs des arêtes dont une extrémité est dans CC, et l'autre dans M, sont:

|(s1,s2)|=7, |(s5,s2)|=5, |(s5,s4)|=3, |(s3,s4)|=1, |(s3,s2)|=4.

Le poids minimal est |(s3,s4)|=1.

Algorithme de Prim, exemple



5b. Les sommets adjacents aux sommets de CC : s2, s4.

Les longueurs des arêtes dont une extrémité est dans CC, et l'autre dans M, sont:

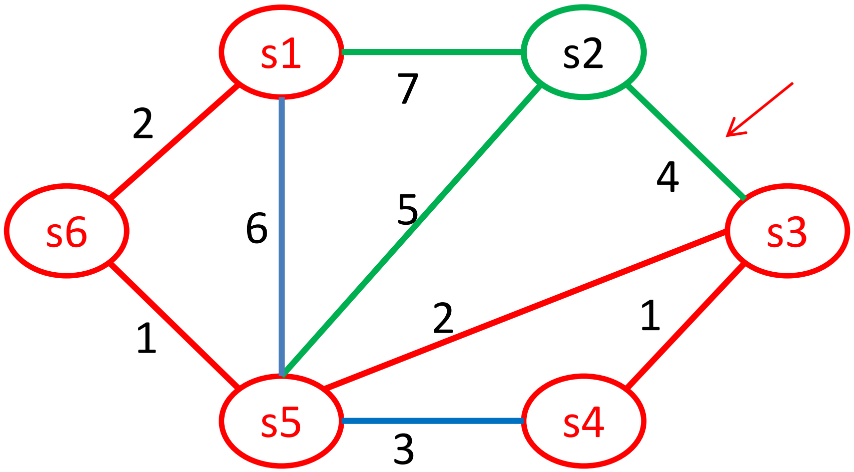
|(s1,s2)|=7, |(s5,s2)|=5, |(s5,s4)|=3, |(s3,s4)|=1, |(s3,s2)|=4.

Le poids minimal est |(s3,s4)|=1.

On met le sommet s4 dans CC, et on ajoute (s3, s4) à A'.

CC={s1,s3,s4,s5,s6}, M ={s2}, A' = {(s1,s6), (s6,s5), (s5,s3), (s3,s4)}.

Algorithme de Prim, exemple



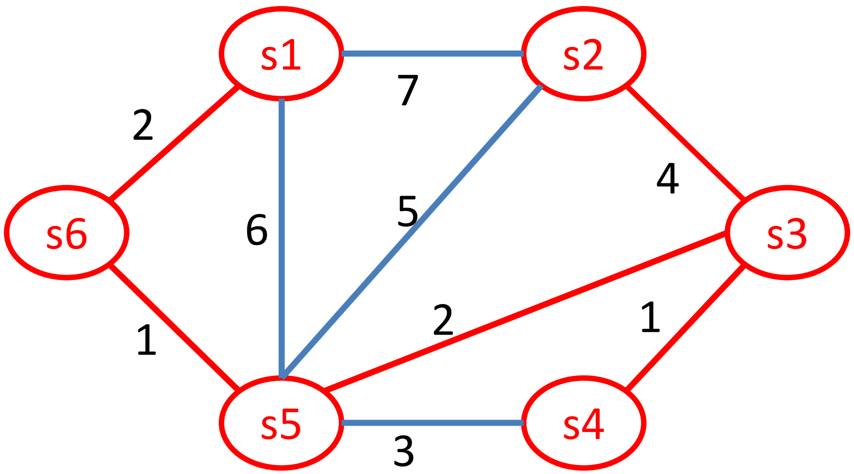
6a. Le sommet adjacent aux sommets de CC : s2.

Les longueurs des arêtes dont une extrémité est dans CC, et l'autre dans M, sont:

|(s1,s2)|=7, |(s5,s2)|=5, |(s3,s2)|=4.

Le poids minimal est |(s3,s2)|=4.

Algorithme de Prim, exemple



6b. Le sommet adjacent aux sommets de CC : s2.

Les longueurs des arêtes dont une extrémité est dans CC, et l'autre dans M, sont:

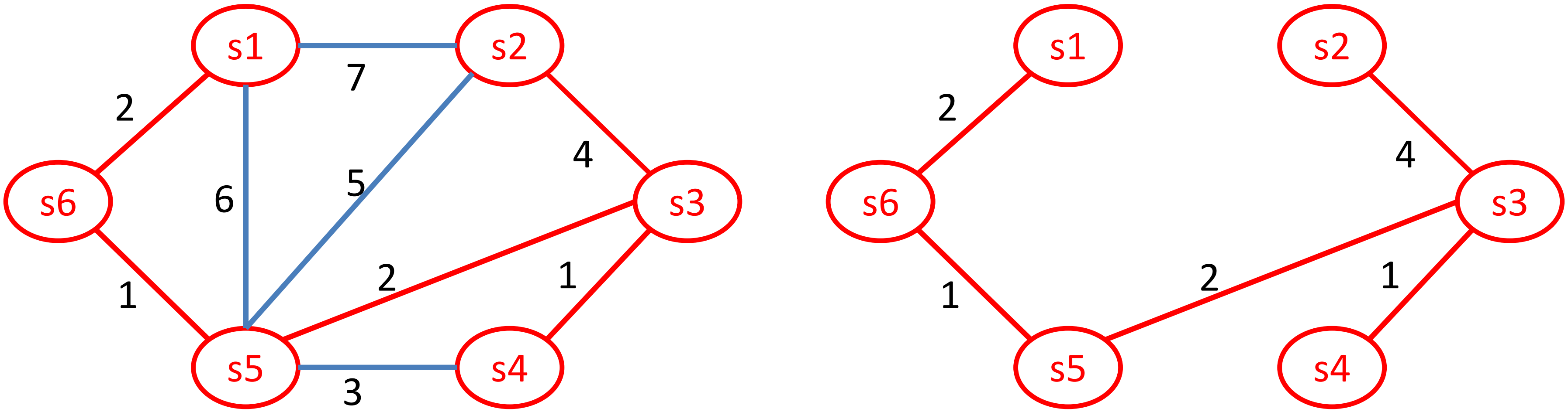
|(s1,s2)|=7, |(s5,s2)|=5, |(s3,s2)|=4.

Le poids minimal est |(s3,s2)|=4.

On met le sommet s2 dans CC, et on ajoute (s3, s2) à A'.

CC={s1,s2,s3,s4,s5,s6}, M = ∅, A' = {(s1,s6), (s6,s5), (s5,s3), (s3,s4), (s3,s2)}.

Algorithme de Prim, exemple



Tous les sommets ont passé à CC, l'arbre couvrant de poids minimum est construit.