Problèmes et variantes de domination romaine dans les graphes d'intervalles

Nicolas Schivre

23 juin 2023

Contexte

La LIFO :

- Laboratoire Fondamentale d'Informatique d'Orléans
- 5 équipes de recherche
- 45 chercheurs et 30 (post) doctorants

L'équipe GAMoC :

- Graphes, Algorithmies et Modèles de Calcul
- 7 chercheurs et 6 (post) doctorants
- Algorithmique des graphes
- Modèles de calcul

Sommaire

Problème du voyageur du commerce

Principe

Chercher un cycle hamiltonien de moindre poids dans le graphe.

Problème du voyageur du commerce

Principe

Chercher un cycle hamiltonien de moindre poids dans le graphe.

Definition

Cycle hamiltonien : Cycle passant par tous les sommets du graphe.

Problème du voyageur du commerce

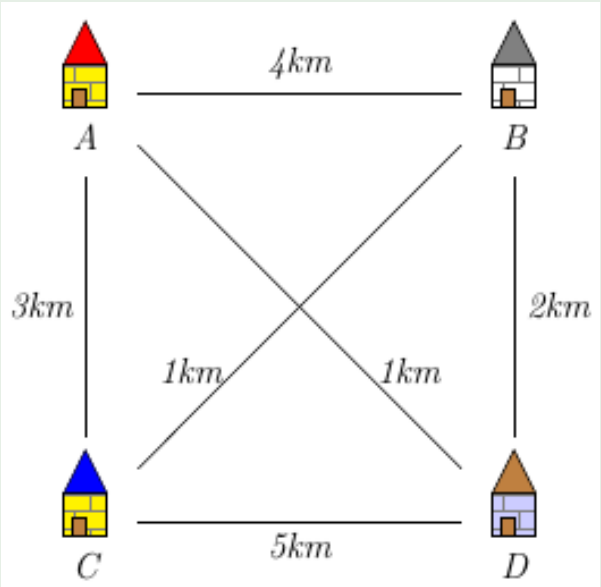
Principe

Chercher un cycle hamiltonien de moindre poids dans le graphe.

Definition

Cycle hamiltonien : Cycle passant par tous les sommets du graphe.

Exemple : Wikipédia



Problème du voyageur du commerce

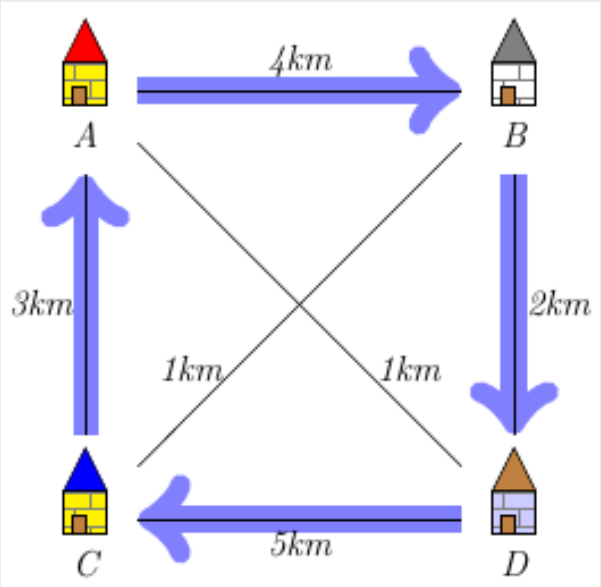
Principe

Chercher un cycle hamiltonien de moindre poids dans le graphe.

Definition

Cycle hamiltonien : Cycle passant par tous les sommets du graphe.

Exemple : une solution pas optimale



Problème du voyageur du commerce

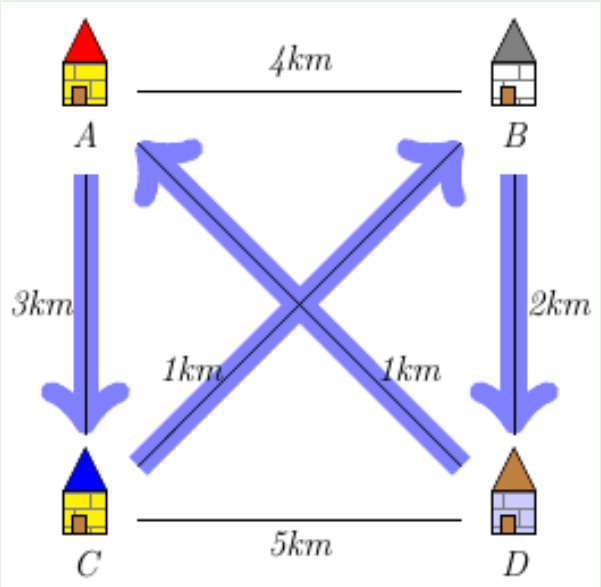
Principe

Chercher un cycle hamiltonien de moindre poids dans le graphe.

Definition

Cycle hamiltonien : Cycle passant par tous les sommets du graphe.

Exemple : une solution optimale



Problème de coloration

Principe

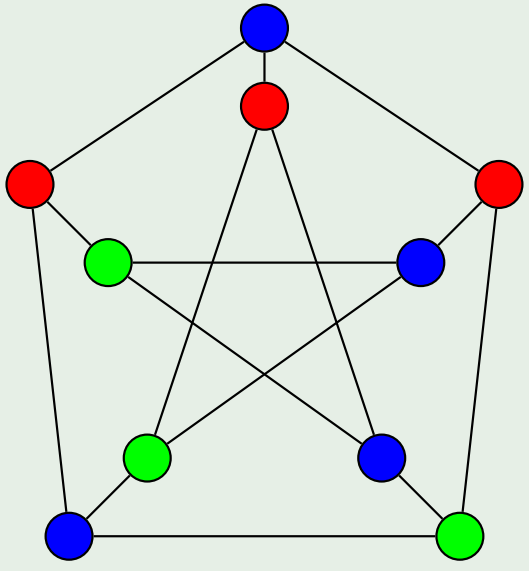
Attribuer une couleur à chaque sommet telle que pour chaque paire de sommets voisins, les sommets aient des couleurs différentes.

Problème de coloration

Principe

Attribuer une couleur à chaque sommet telle que pour chaque paire de sommets voisins, les sommets aient des couleurs différentes.

Exemple : coloration du graphe de Petersen



Problème de domination

Principe

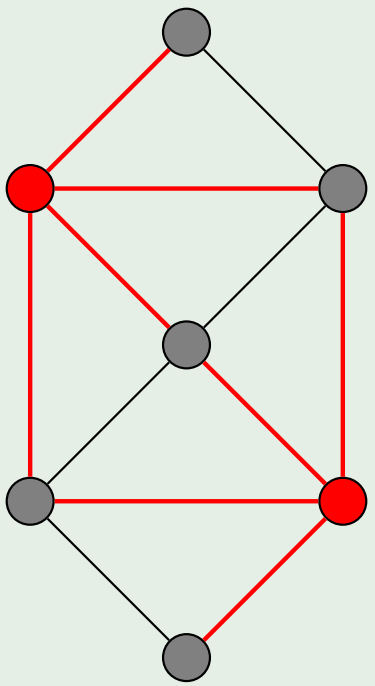
Calculer un ensemble de sommets tel que chaque sommet appartient à cet ensemble ou a un voisin dans l'ensemble.

Problème de domination

Principe

Calculer un ensemble de sommets tel que chaque sommet appartient à cet ensemble ou a un voisin dans l'ensemble.

Exemple : domination d'un graphe



Réduction polynomiale

Réduire des problèmes

Soit X et Y deux problèmes.

Réduction polynomiale

Réduire des problèmes

Soit X et Y deux problèmes.

Y est réductible polynomialement à X si :

Réduction polynomiale

Réduire des problèmes

Soit X et Y deux problèmes.

Y est réductible polynomialement à X si :

Considérant une boite noire résolvant X en une étape polynomiale Y est résolvable avec un nombre polynomial d'appels à la boite noire plus un nombre polynomiale d'opérations de base.

Réduction polynomiale

Réduire des problèmes

Soit X et Y deux problèmes.

Y est réductible polynomialement à X si :

Considérant une boite noire résolvant X en une étape polynomiale Y est résolvable avec un nombre polynomial d'appels à la boite noire plus un nombre polynomiale d'opérations de base.

Une notation simple

$Y \leq_p X$ signifie que Y est polynomialement réductible à X.

Définition

Des classes de complexité

P pour *Polynomial time*.

NP pour *Nondeterministic Polynomial time*.

Définition

Des classes de complexité

P pour *Polynomial time*.

NP pour *Nondeterministic Polynomial time*.

Toute solution à un problème appartenant à NP est vérifiable en temps polynomial.

Définition

Des classes de complexité

P pour *Polynomial time*.

NP pour *Nondeterministic Polynomial time*.

Toute solution à un problème appartenant à NP est vérifiable en temps polynomial.

Un problème X est NP-complet si :

- X appartient à la classe NP.
- $\forall$ problèmes $Y \in$ NP on a $Y \leq_p X$.

Définition

Des classes de complexité

P pour *Polynomial time*.

NP pour *Nondeterministic Polynomial time*.

Toute solution à un problème appartenant à NP est vérifiable en temps polynomial.

Un problème X est NP-complet si :

- X appartient à la classe NP.
- $\forall$ problèmes $Y \in$ NP on a $Y \leq_p X$.

Les problèmes NP-difficiles

2^{eme} propriété de NP-complet uniquement.

Enjeux et intérêts

Un problème du millénaire

$$P = NP\,?$$

Enjeux et intérêts

Un problème du millénaire

$$P = NP ?$$

Si P = NP

Résoudre tous les problèmes de NP en temps polynomial.

Enjeux et intérêts

Un problème du millénaire

$$P = NP ?$$

Si P = NP

Résoudre tous les problèmes de NP en temps polynomial.

Si P ≠ NP

Aucun problème NP-complet ne serait résoluble en temps polynomial.

Enjeux et intérêts

Un problème du millénaire

$$P = NP ?$$

Si P = NP

Résoudre tous les problèmes de NP en temps polynomial.

Si P ≠ NP

Aucun problème NP-complet ne serait résoluble en temps polynomial.

Domination, un problème NP-complet

On va donc se restreindre à une certaine classe de graphes pour le résoudre.

Une représentation par des intervalles de la droite réelle

Un début et une fin

Un intervalle i est représenté par un début $d(i)$ et une fin $f(i)$,

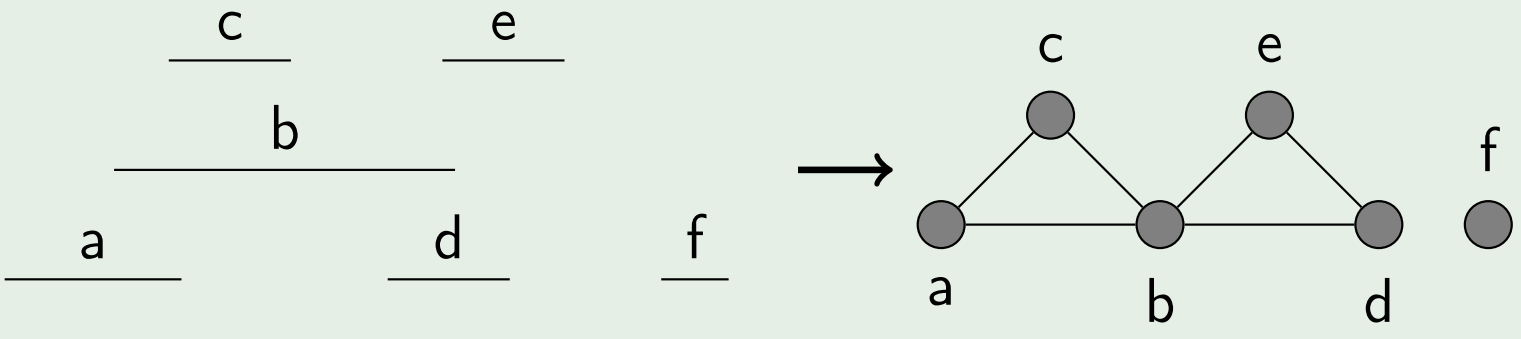
aussi appelé gauche $l(i)$ (*left*) et droite $r(i)$ (*right*).

Une représentation par des intervalles de la droite réelle

Un graphe d'intersection

Graphe d'intersection d'un ensemble d'intervalles de la droite réelle.

Un ensemble d'intervalles et son graphe d'intersections



Limites de la classe

Des graphes impossible

Impossible de construire un cycle induit de longueur ≥ 4.

Definition

Cycle induit : Cycle sans corde.

Definition

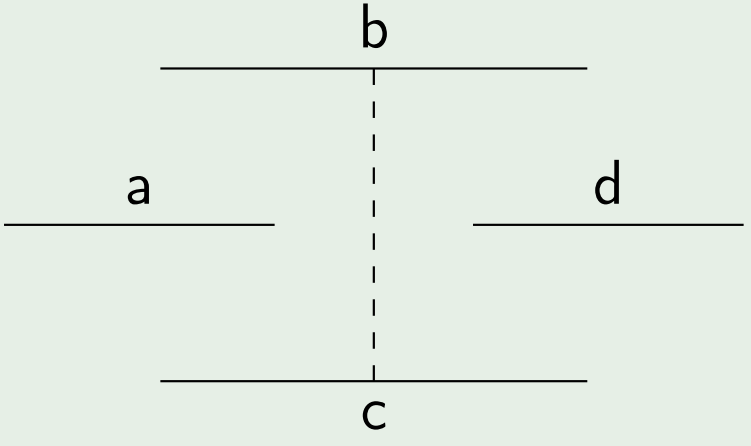
Corde : Arrête reliant deux sommets non adjacent d'un cycle.

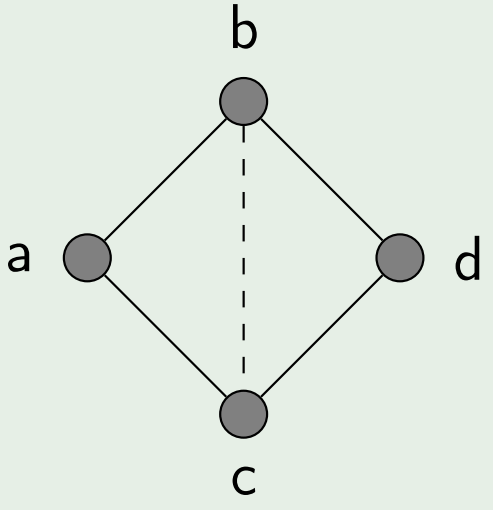
Limites de la classe

Definition

C_4 : Cycle de taille 4.

Exemple : Pas de C_4 dans les graphes d'intervalles





Modèle d'intervalles normalisés

Une représentation normalisée

Représentation des n intervalles sur un axe indexé de 1 à $2n$.

Chaque indice représente soit le début, soit la fin d'un intervalle.

Modèle d'intervalles normalisés

Une représentation normalisée

Représentation des n intervalles sur un axe indexé de 1 à $2n$.

Chaque indice représente soit le début, soit la fin d'un intervalle.

Facilités algorithmique

Facilite le parcours du graphe d'intervalles et la conception d'algorithmes.

Modèle d'intervalles normalisés

Une représentation normalisée

Représentation des n intervalles sur un axe indexé de 1 à $2n$.

Chaque indice représente soit le début, soit la fin d'un intervalle.

Facilités algorithmique

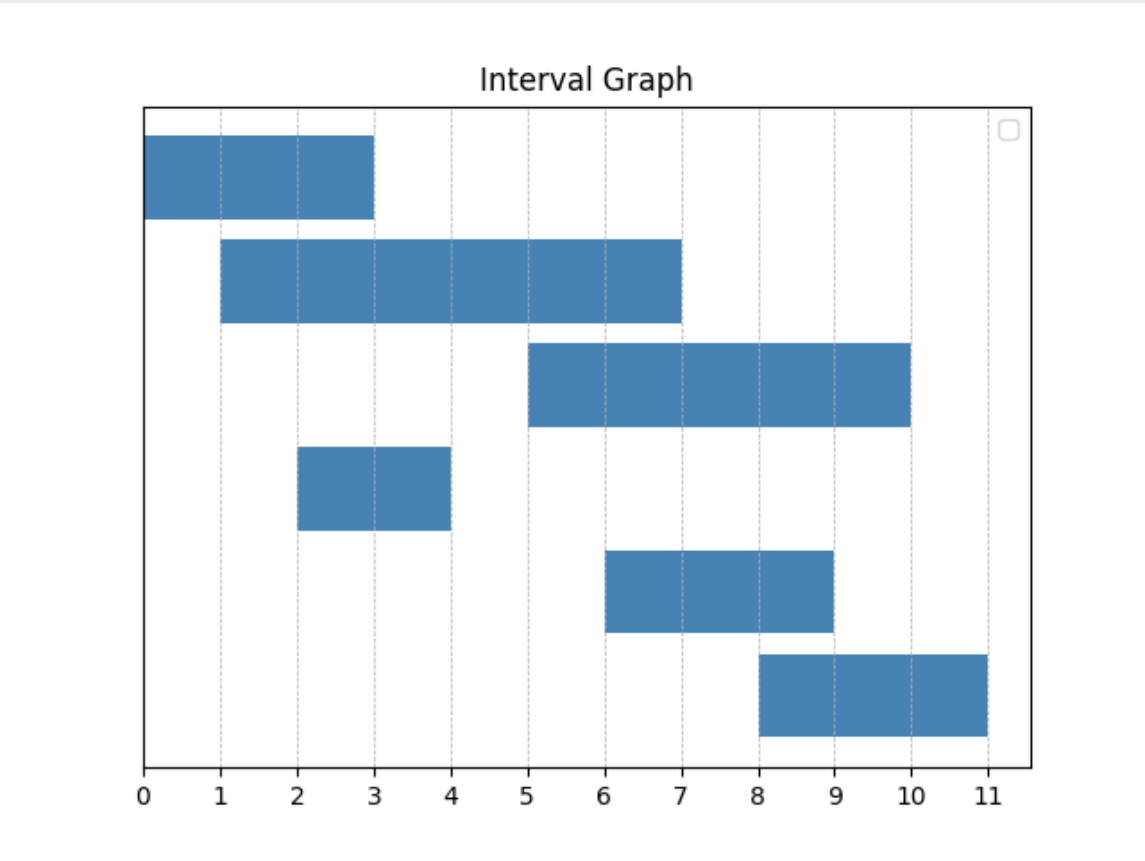
Facilite le parcours du graphe d'intervalles et la conception d'algorithmes.

Algorithmes linéaire

Rend le tri efficace (linéaire) grâce l'utilisation du tri par dénombrement.

Modèle d'intervalles normalisés

Exemple : généré à partir d'un script python [1]



1. Script hébergé à l'adresse : https ://github.com/Itasuka/roman-domination

Définitions et notations : domination romaine

Un sous-problème de domination

Deux ensembles de sommets dominant différents : $V1$ et $V2$.

Les sommets non dominant font partis de l'ensemble $V0$.

Définitions et notations : domination romaine

Un sous-problème de domination

Deux ensembles de sommets dominant différents : $V1$ et $V2$.

Les sommets non dominant font partis de l'ensemble $V0$.

Definition

$V1$: Ensemble des sommets se dominant uniquement eux-même.

Définitions et notations : domination romaine

Un sous-problème de domination

Deux ensembles de sommets dominant différents : $V1$ et $V2$.

Les sommets non dominant font partis de l'ensemble $V0$.

Definition

$V1$: Ensemble des sommets se dominant uniquement eux-même.

Definition

$V2$: Ensemble des sommets se dominant eux-même ainsi que leurs voisins.

Définitions et notations : domination romaine

Un sous-problème de domination

Deux ensembles de sommets dominant différents : $V1$ et $V2$.

Les sommets non dominant font partis de l'ensemble $V0$.

Definition

$V1$: Ensemble des sommets se dominant uniquement eux-même.

Definition

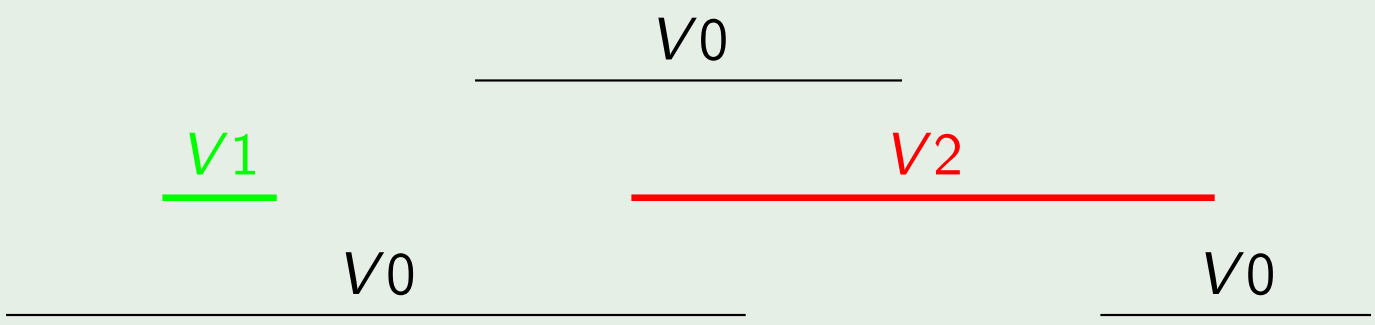
$V2$: Ensemble des sommets se dominant eux-même ainsi que leurs voisins.

Un coût particulier

Le coût d'une solution est $\gamma_R(G) = |V1| + 2 * |V2|$.

Exemple d'application

Exemple : une solution optimale de domination romaine



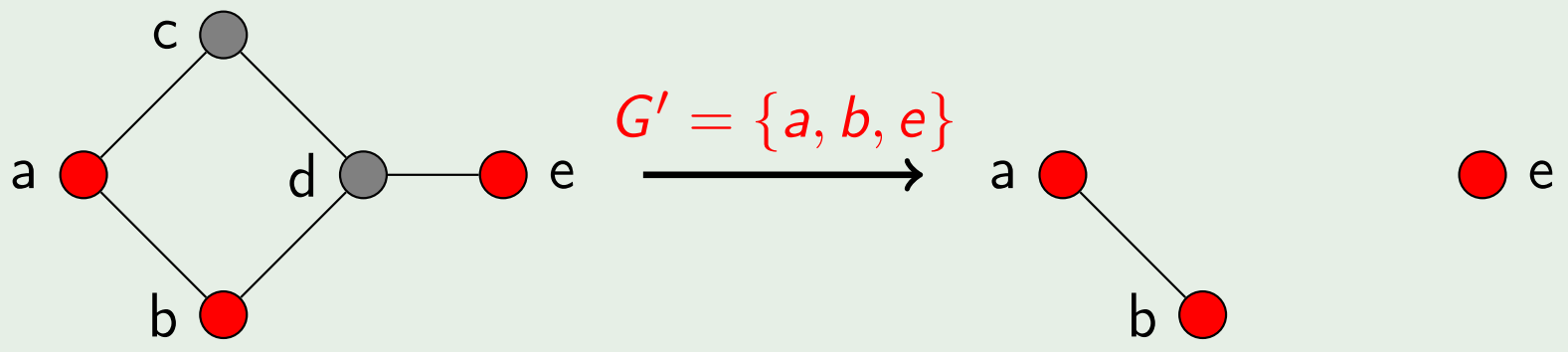
$$\gamma_R(G) = 3$$

Définition et notations

Définition : sous-graphe induit

G' est un sous-graphe induit d'un graphe $G = (V, E)$ si pour tout couple $(u, v) \in V$, u et v sont connectés dans $G' = (V', E')$ si et seulement si ils sont connectés dans G.

Exemple : Sous-graphe induit



Définition et notations

Une variante de domination romaine

Une contrainte supplémentaire :

Un sommet de type $V2$ ne peut être isolé dans le sous-graphe induit par l'ensemble $V1 \cup V2$.

Définition et notations

Une variante de domination romaine

Une contrainte supplémentaire :

Un sommet de type $V2$ ne peut être isolé dans le sous-graphe induit par l'ensemble $V1 \cup V2$.

Autrement dit

S'il existe un sommet isolé dans le sous-graphe induit par l'ensemble solution $V1 \cup V2$ alors ce sommet est de type $V1$.

Définition et notations

Une variante de domination romaine

Une contrainte supplémentaire :

Un sommet de type $V2$ ne peut être isolé dans le sous-graphe induit par l'ensemble $V1 \cup V2$.

Autrement dit

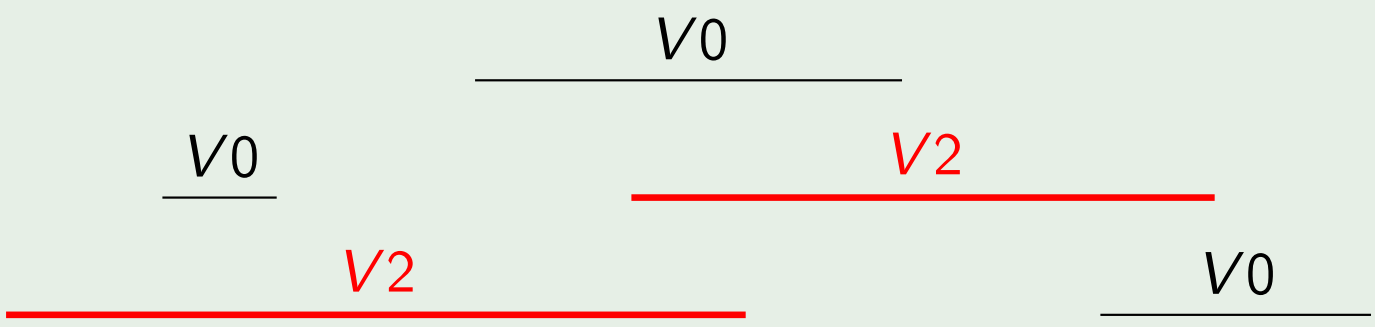
S'il existe un sommet isolé dans le sous-graphe induit par l'ensemble solution $V1 \cup V2$ alors ce sommet est de type $V1$.

Coût d'une solution

Identique à la domination romaine, $\gamma_{QTRD}(G) = |V1| + 2 * |V2|$.

Exemple d'application

Exemple : une solution optimale de domination romaine quasi totale



$$\gamma_{QTRD}(G) = 4$$

Un algorithme par disjonction de cas

Inspiré de précédents résultats

Adaptation d'un algorithme par programmation dynamique pour la domination romaine de M. Liedloff et d'autres chercheurs dans l'article « *Efficient algorithms for roman domination on some classes of graphs* » (Discr. Appl. Math., 2008)

Un algorithme par disjonction de cas

Inspiré de précédents résultats

Adaptation d'un algorithme par programmation dynamique pour la domination romaine de M. Liedloff et d'autres chercheurs dans l'article « *Efficient algorithms for roman domination on some classes of graphs* » (Discr. Appl. Math., 2008)

Utilisation de la programmation dynamique :

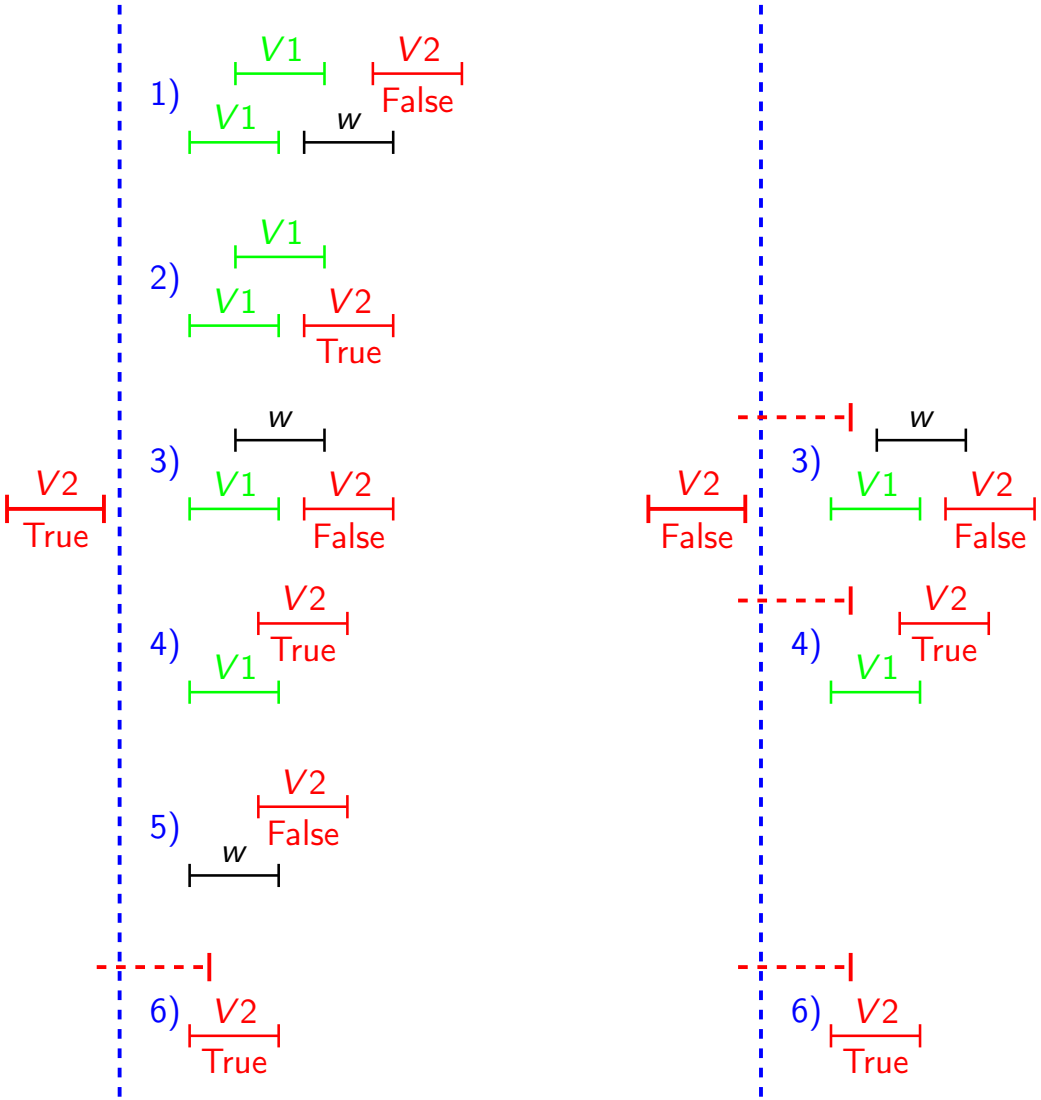
Structure d'une sous-solution

Une sous-solution $opt[i, Bool]$ est une solution considérant les i premiers intervalles, pour laquelle :

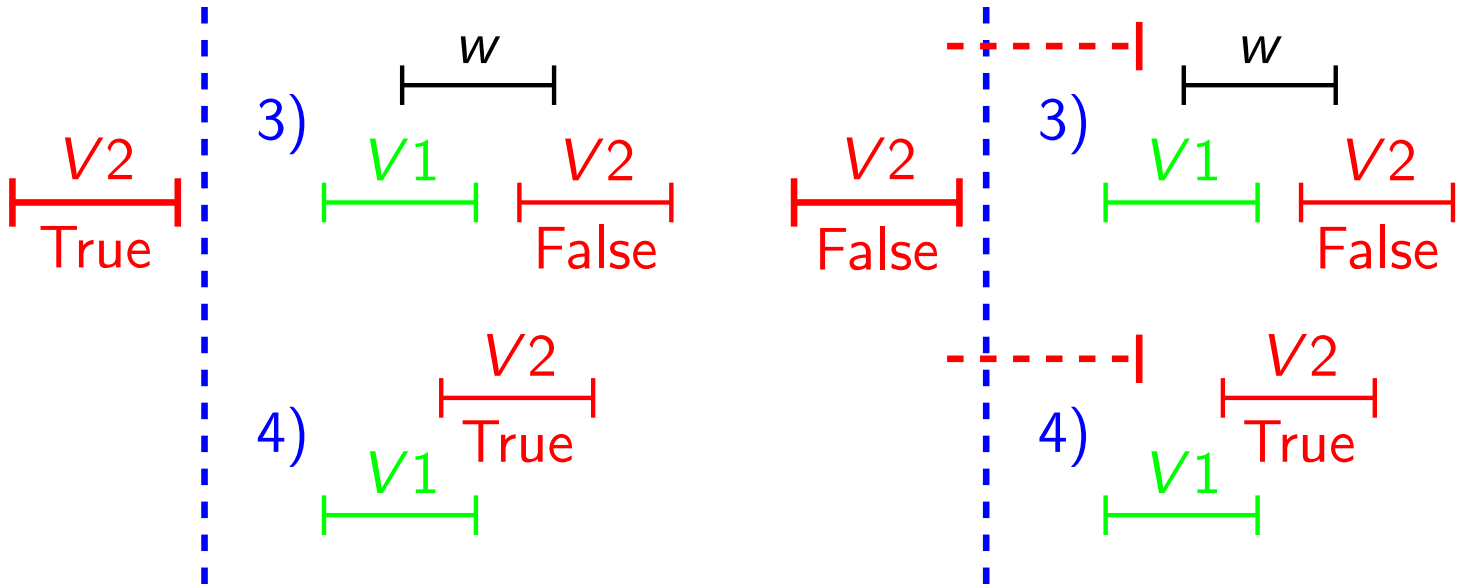
- i est un sommet de type $V2$, et
- tel que i n'est pas isolé dans la solution ssi $Bool$ est $True$.

Les intervalles du modèle sont triés par date de fin croissante.

Les différents cas d'une extension d'une sous-solution



Exemple de l'extension 3-4



Une procédure

Data: Un sommet v d'un graphe G=(V,E) ayant déjà une sous-solution (V_1^v, V_2^v).
Result: Une extension de (V_1^v, V_2^v) selon le 1^{er} et 2^{eme} cas à partir d'une sous-solution.

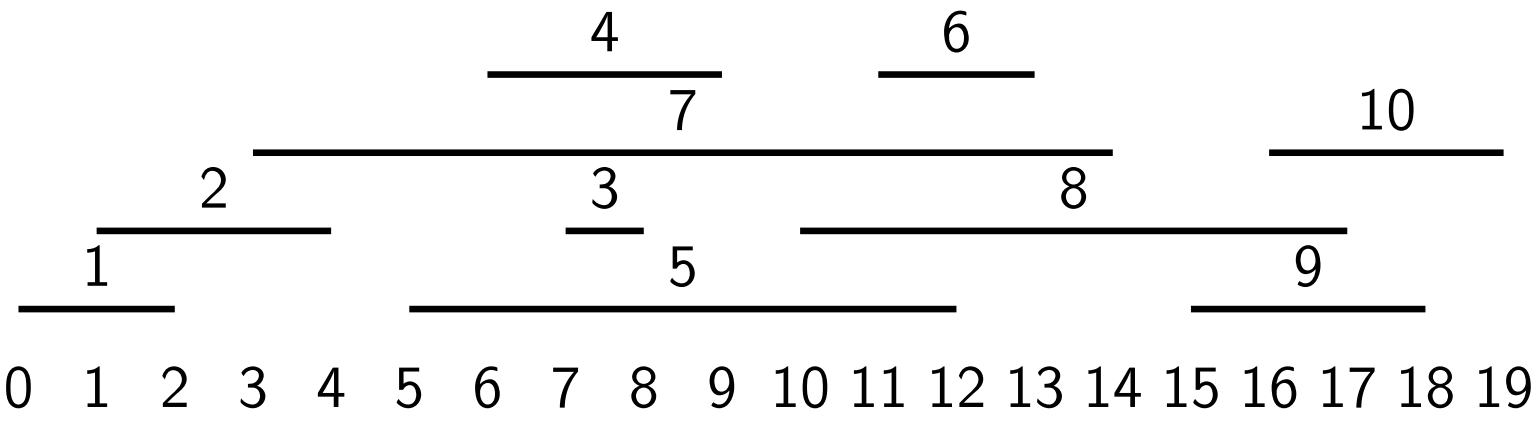
$i_1 \longleftarrow \min(droite(x)), x \in V, droite(v) < gauche(x)$
if $i_1 \neq Nil$ **then**
  $i'_1 \longleftarrow \min(droite(x)), x \in V \setminus \{i_1\}, droite(v) < gauche(x)$
  if $i'_1 \neq Nil$ **then**
    $w \longleftarrow \min(droite(x)), x \in V \setminus \{i_1, i'_1\}, droite(v) < gauche(x)$
    if $w \neq Nil$ **then**
      $i_2 \longleftarrow \max(droite(x)), x \in N[w]$
      if $i_2 \notin N[i_1]$ **then**
        if $i_2 \notin N[i'_1]$ **then**
          $opt[i_2, False] \longleftarrow \min(opt[i_2, False], opt[v, prop] + 4)$
        end
        else
          $opt[i_2, True] \longleftarrow \min(opt[i_2, True], opt[v, prop] + 4)$
        end
      end
      $i_{22} \longleftarrow \max(droite(x)), x \in N[i'_1]$
      if $i_{22} \neq i'_1$ **and** $i_{22} \notin N[i_1]$ **then**
        $opt[i_{22}, True] \longleftarrow \min(opt[i_{22}, True], opt[v, prop] + 4)$
      end
    end
    else
      $n \longleftarrow \max(droite(x)), x \in V$
      $opt[n, True] \longleftarrow \min(opt[n, True], opt[v, prop] + 2)$
    end
  end
end

L'algorithme résolvant QTRD

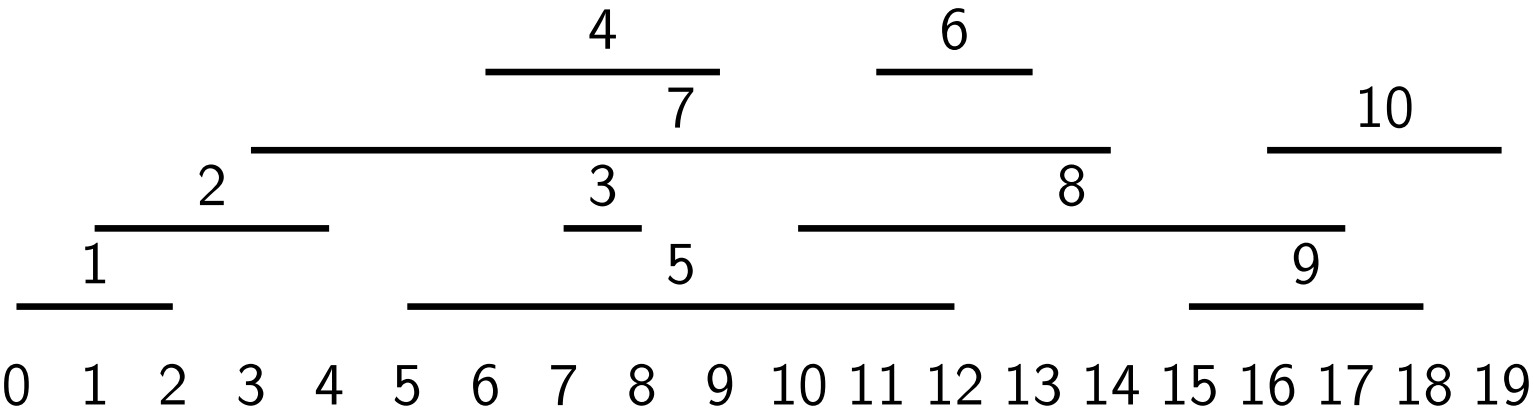
Data: Un Graphe d'intervalles G=(V,E) construit selon un modèle normalisé.
Result: Une valeur minimale de QTRD dans G $\gamma_{QTRD}(G)$.

$maxValue \longleftarrow 2 * |V|$
$initialValue \longleftarrow (-1, -1)$
$opt[initialValue, True] \longleftarrow 0$
for $v \in V$ **do**
    $opt[v, True] \longleftarrow maxValue$
    $opt[v, False] \longleftarrow maxValue$
end
Solution1-2($initialValue$)
Solution3-4($initialValue$, $True$)
Solution5-6($initialValue$, $True$)
$Trierlegraphepardatedefincroissante$ **for** $v \in V$ **do**
    if $opt[v, True] \neq maxValue$ **then**
        Solution1-2(v)
        Solution3-4(v, $True$)
        Solution5-6(v, $True$)
    end
    if $opt[v, False] \neq maxValue$ **then**
        Solution3-4(v, $False$)
        Solution5-6(v, $False$)
    end
end
$n \longleftarrow \max(droite(x)), x \in V$
return $\gamma_{QTRD}(G) = opt[n, True]$

Exemple d'exécution



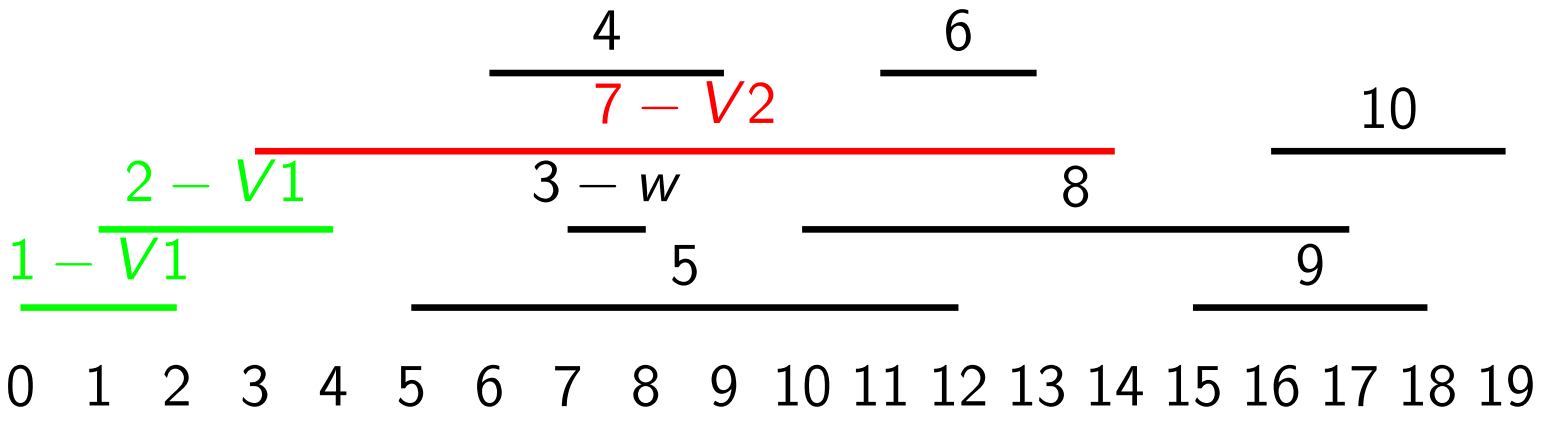
Exemple d'exécution



i	0	1	2	3	4	5	6	7	8	9	10
True	**0**	20	20	20	20	20	20	20	20	20	20
False	20	20	20	20	20	20	20	20	20	20	20

Table – *Opt* - Initialisation

Exemple d'exécution

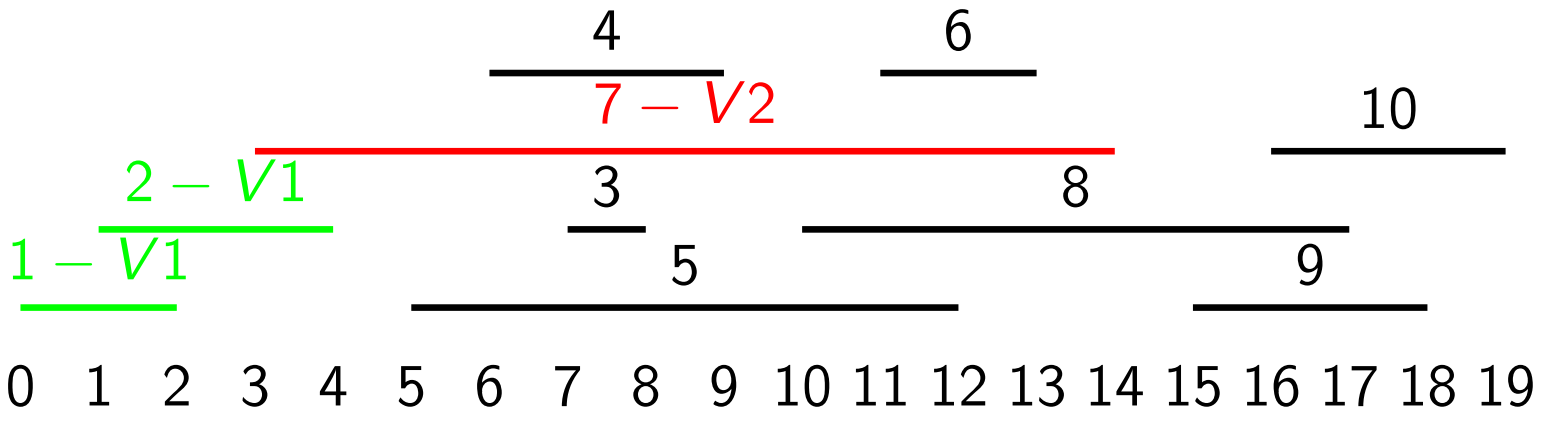


i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	20	20	20	20	20	**4**	20	20	20
False	20	20	20	20	20	20	20	20	20	20	20

Table – *Opt* - Cas de base 1

$$\mathbf{4} < 20$$

Exemple d'exécution

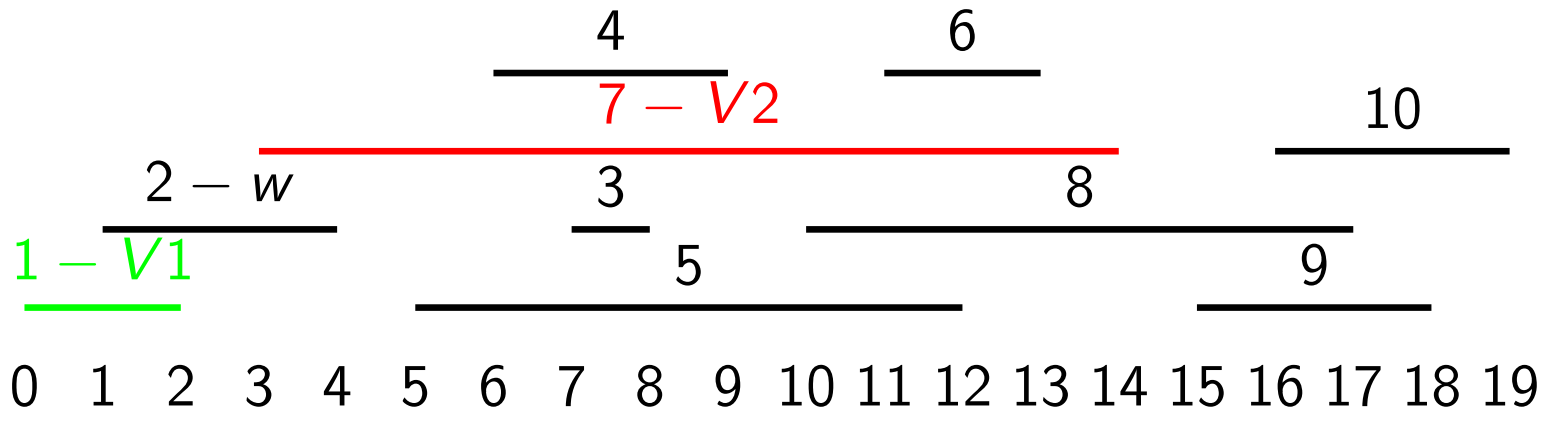


i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	20	20	20	20	20	**4**	20	20	20
False	20	20	20	20	20	20	20	20	20	20	20

Table – *Opt* - Cas de base 2

$$\mathbf{4} = 4$$

Exemple d'exécution

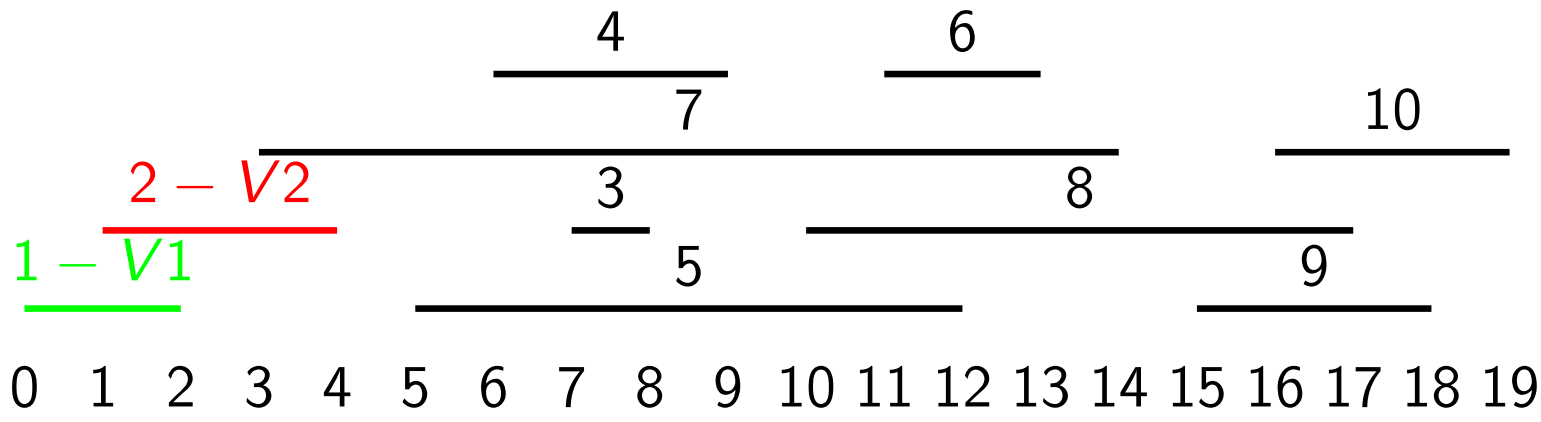


i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	20	20	20	20	20	4	20	20	20
False	20	20	20	20	20	20	20	**3**	20	20	20

Table – *Opt* - Cas de base 3

3 < 20

Exemple d'exécution

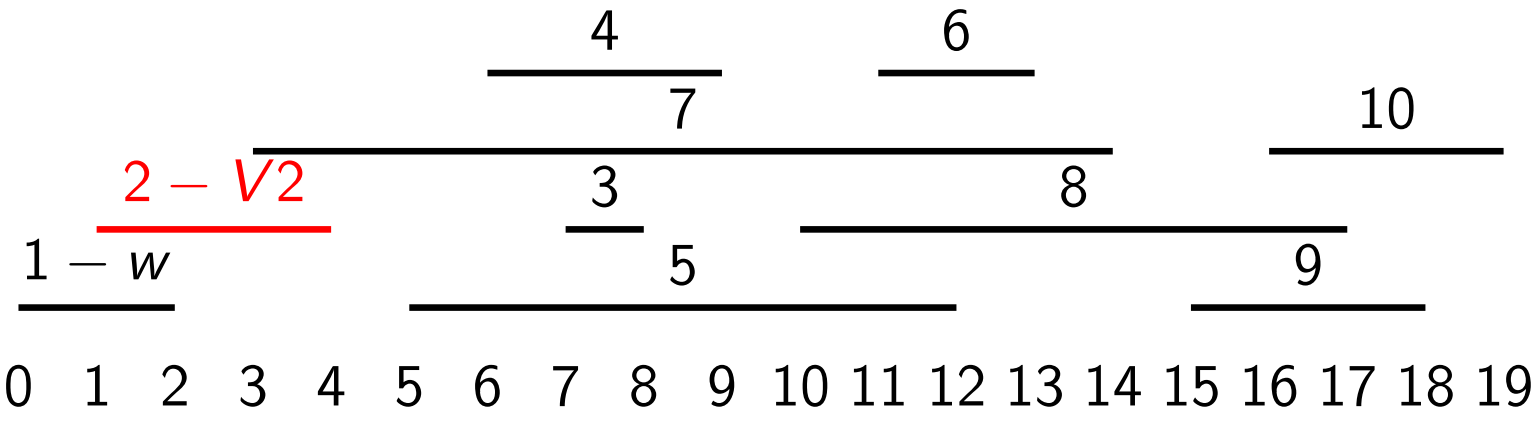


i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	**3**	20	20	20	20	4	20	20	20
False	20	20	20	20	20	20	20	3	20	20	20

Table – *Opt* - Cas de base 4

$$\mathbf{3} < 20$$

Exemple d'exécution

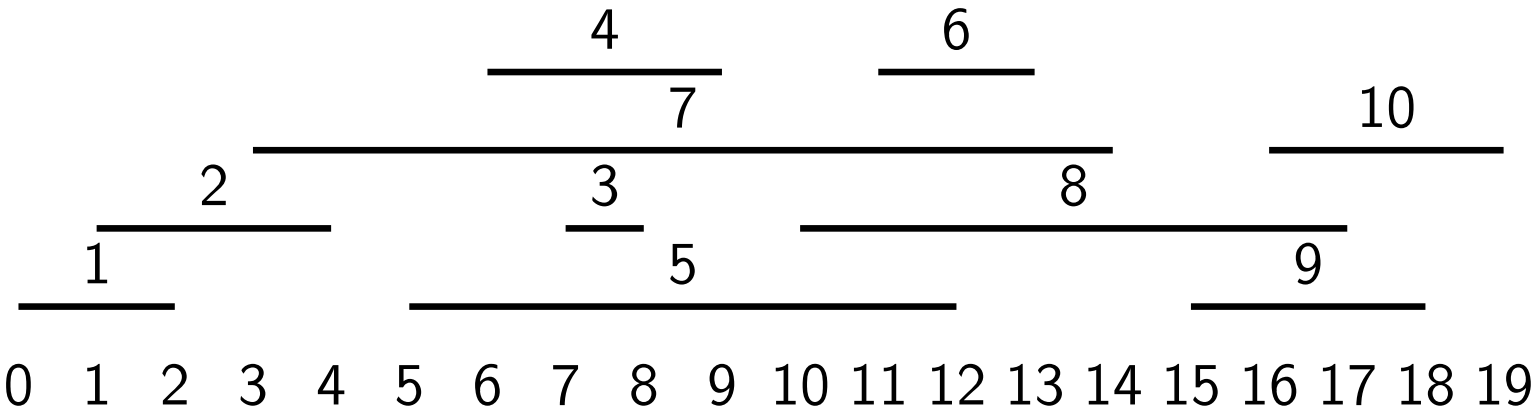


i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	3	20	20	20	20	4	20	20	20
False	20	20	**2**	20	20	20	20	3	20	20	20

Table – *Opt* - Cas de base 5

$$\mathbf{2} < 20$$

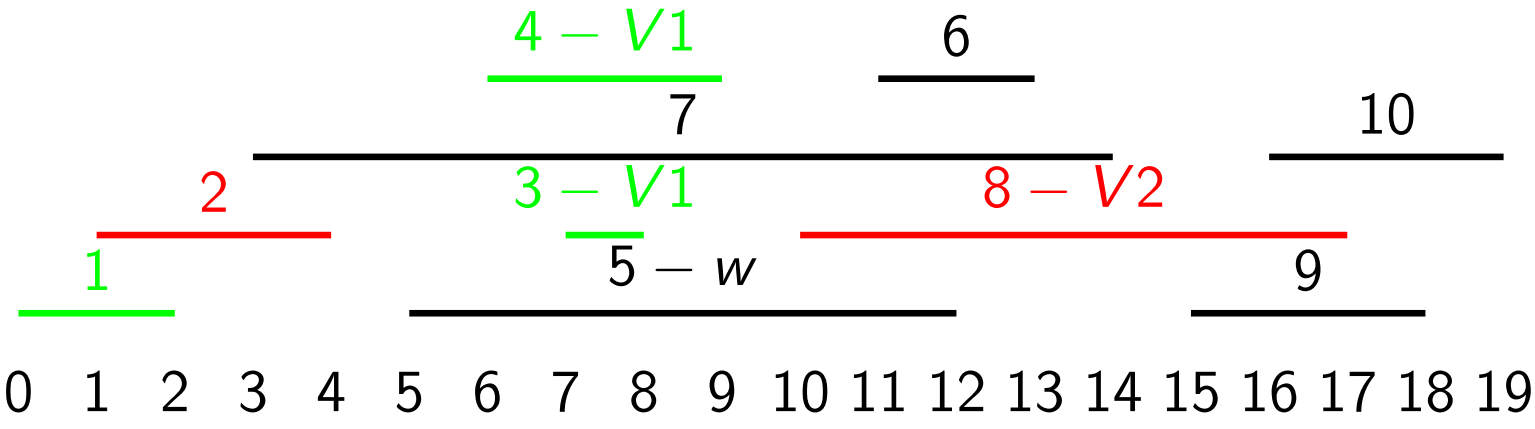
Exemple d'exécution



i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	3	20	20	20	20	4	20	20	20
False	20	20	2	20	20	20	20	3	20	20	20

Table – *Opt* - Cas de base 6

Exemple d'exécution

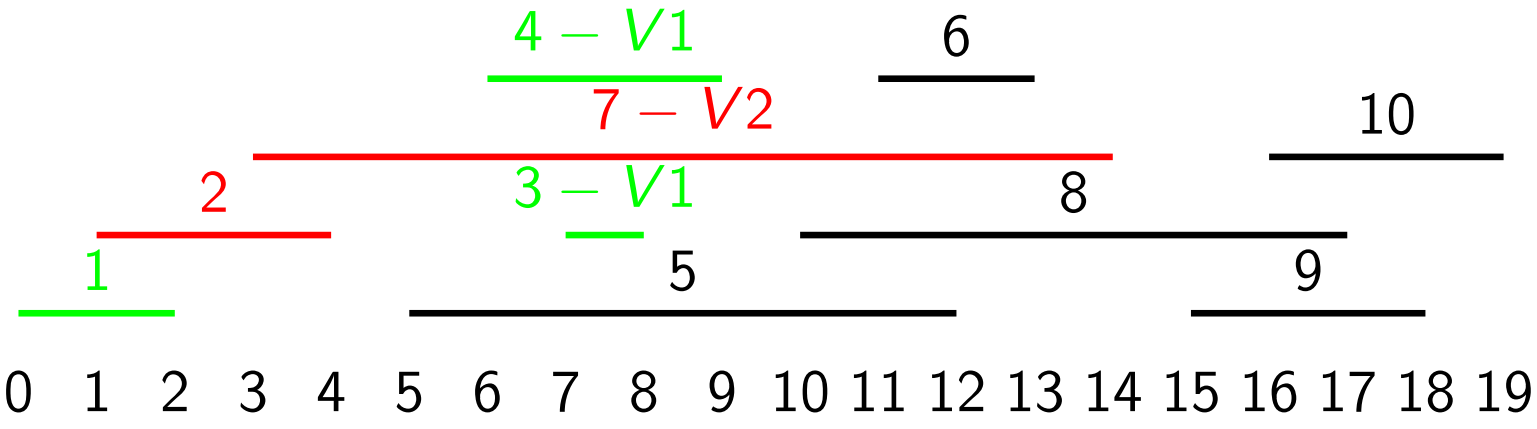


i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	3	20	20	20	20	4	20	20	20
False	20	20	2	20	20	20	20	3	**7**	20	20

Table – *Opt* - Itération $i = 2$ - Cas 1 - True

$$\mathbf{7} < 20$$

Exemple d'exécution

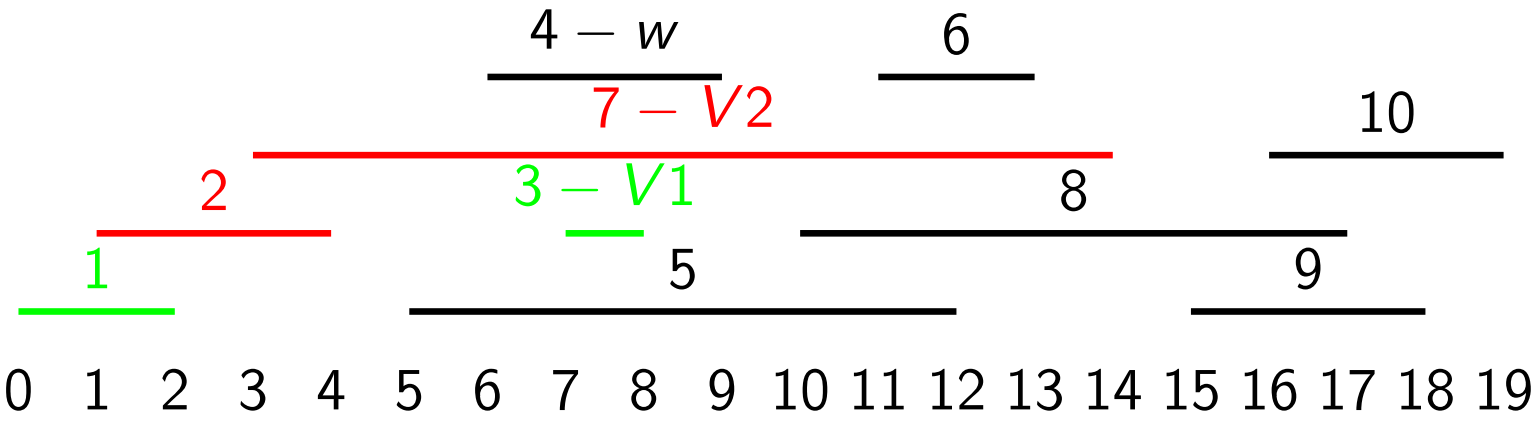


i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	3	20	20	20	20	**4**	20	20	20
False	20	20	2	20	20	20	20	3	7	20	20

Table – *Opt* - Itération $i = 2$ - Cas 2 - True

$$\mathbf{7} > 4$$

Exemple d'exécution

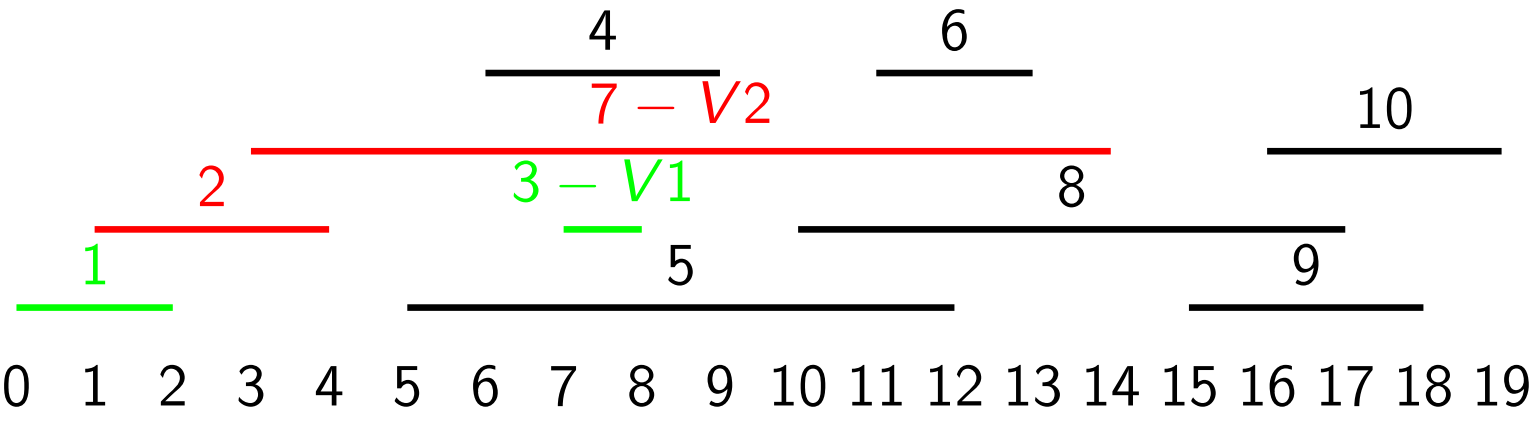


i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	3	20	20	20	20	**4**	20	20	20
False	20	20	2	20	20	20	20	3	7	20	20

Table – *Opt* - Itération $i = 2$ - Cas 3 - True

$$\mathbf{6} > 4$$

Exemple d'exécution

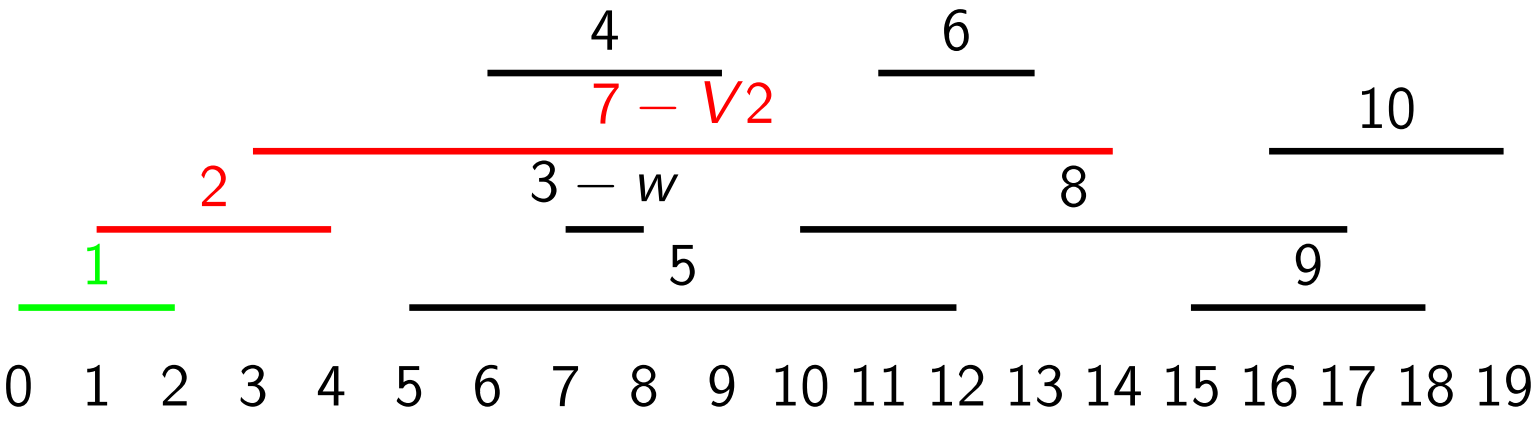


i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	3	20	20	20	20	**4**	20	20	20
False	20	20	2	20	20	20	20	3	7	20	20

Table – *Opt* - Itération $i = 2$ - Cas 4 - True

$$\mathbf{6} > 4$$

Exemple d'exécution

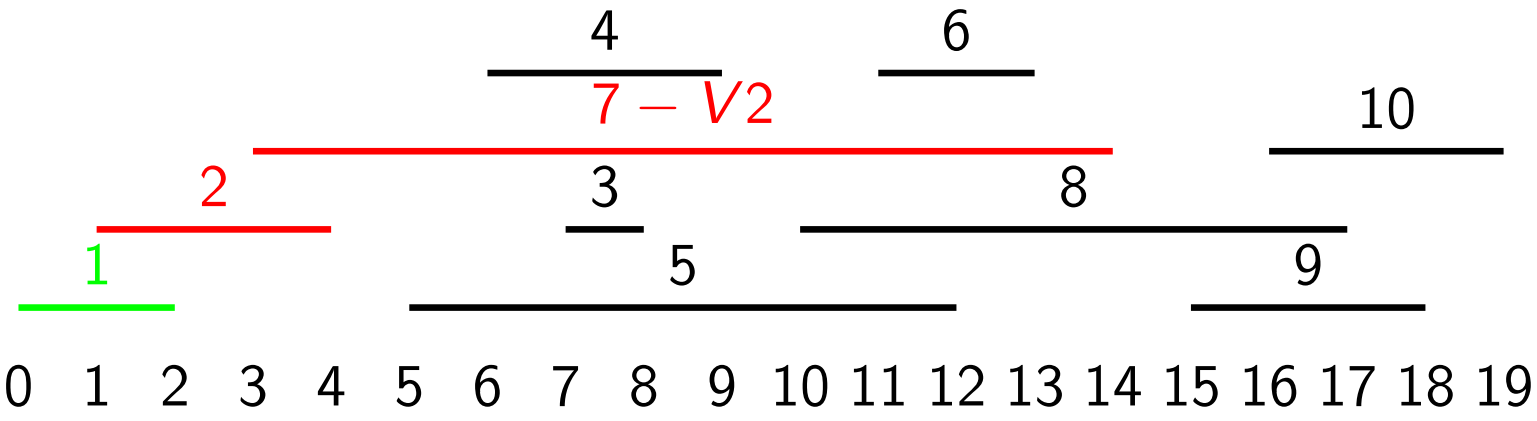


i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	3	20	20	20	20	**4**	20	20	20
False	20	20	2	20	20	20	20	3	7	20	20

Table – *Opt* - Itération $i = 2$ - Cas 5 - True

$$\mathbf{5} > 4$$

Exemple d'exécution

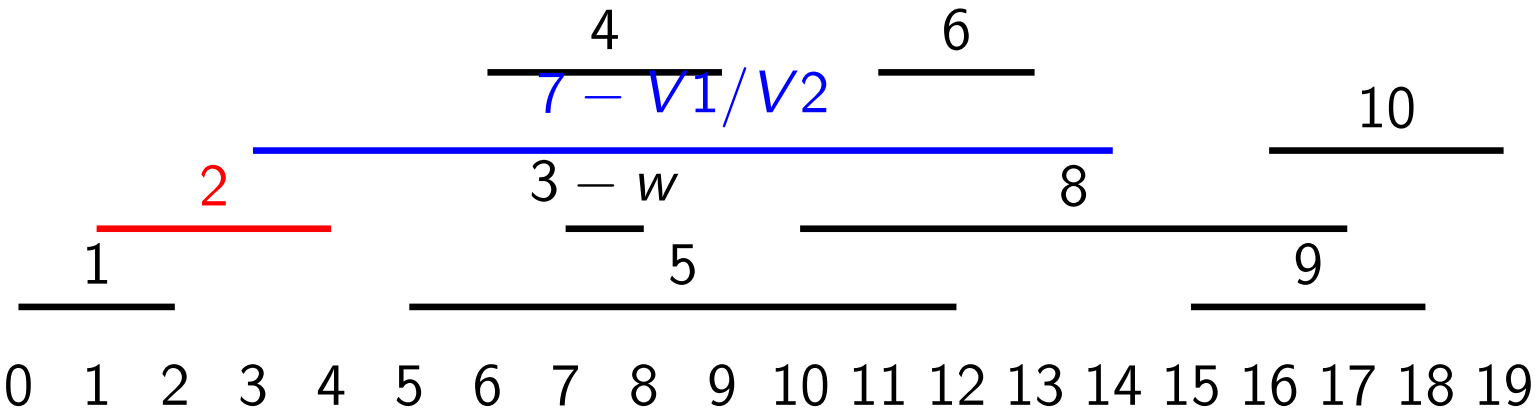


i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	3	20	20	20	20	**4**	20	20	20
False	20	20	2	20	20	20	20	3	7	20	20

Table – *Opt* - Itération $i = 2$ - Cas 6 - True

5 > 4

Exemple d'exécution

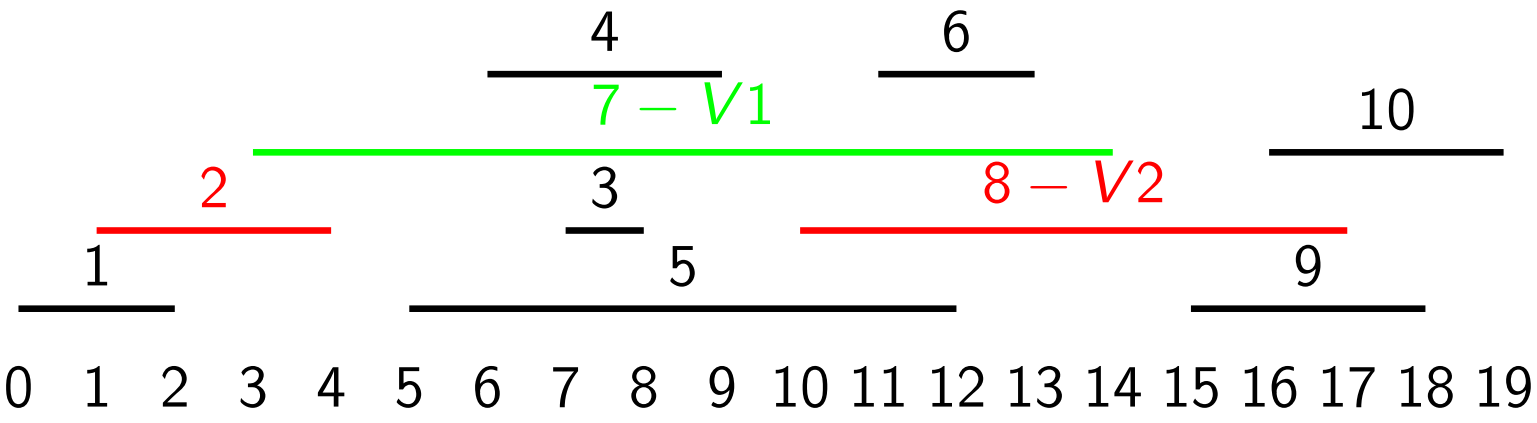


i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	3	20	20	20	20	4	20	20	20
False	20	20	2	20	20	20	20	3	7	20	20

Table – *Opt* - Itération $i = 2$ - Cas 3 - False

L'intervalle 7 ne peut être à la fois dans $V1$ et $V2$.

Exemple d'exécution

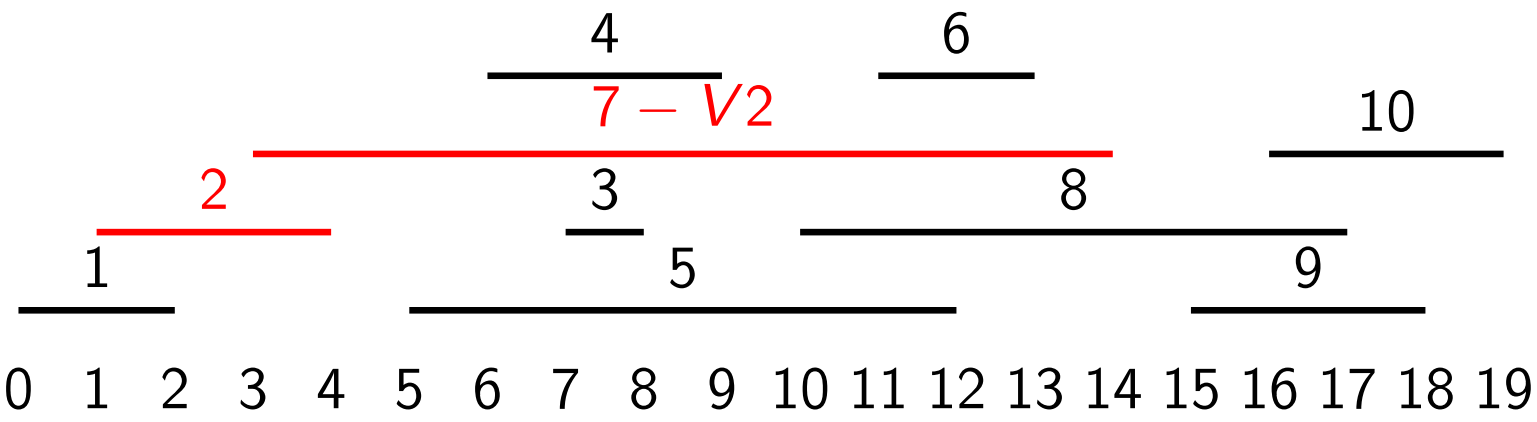


i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	3	20	20	20	20	4	20	20	20
False	20	20	2	20	20	20	20	3	7	20	20

Table – *Opt* - Itération $i = 2$ - Cas 4 - False

L'intervalle 8 n'est pas voisin avec le premier intervalle non dominé.

Exemple d'exécution

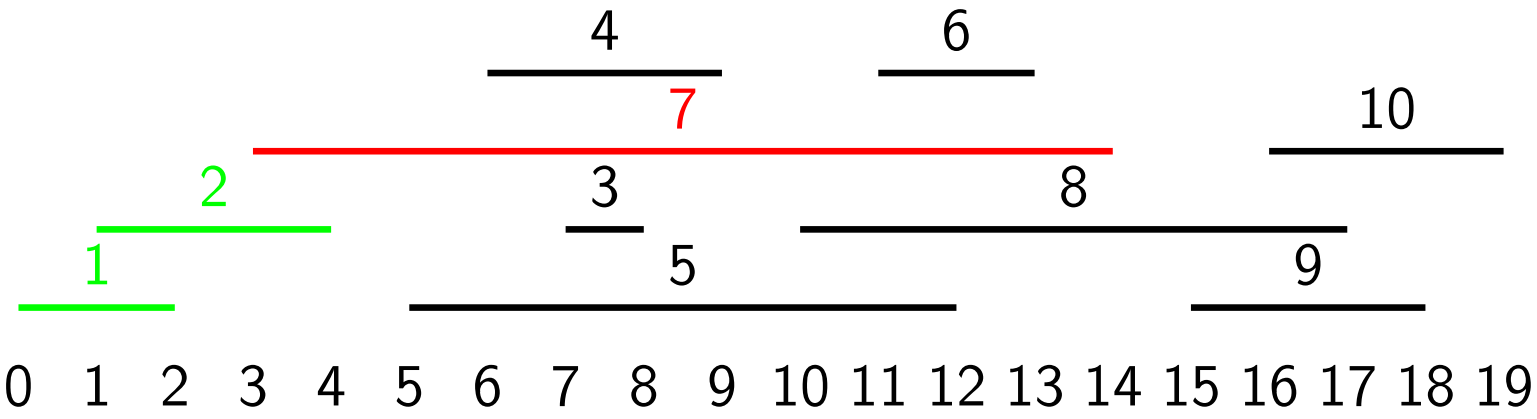


i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	3	20	20	20	20	**4**	20	20	20
False	20	20	2	20	20	20	20	3	7	20	20

Table – *Opt* - Itération $i = 2$ - Cas 6 - False

$$\mathbf{4} = 4$$

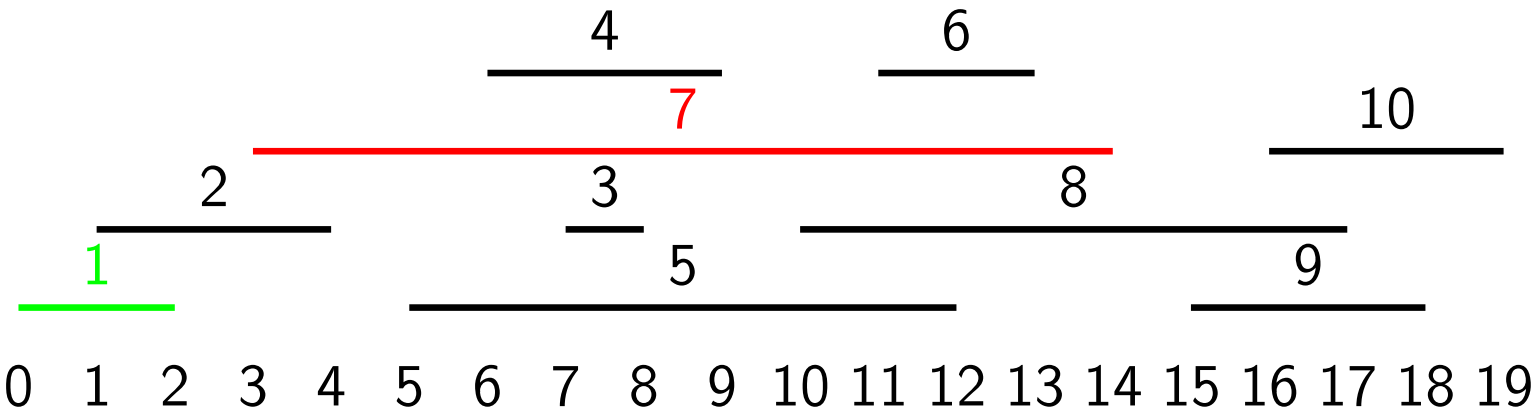
Un peu plus vite...



i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	3	20	20	20	20	4	**6**	20	**6**
False	20	20	2	20	20	20	20	3	7	20	20

Table – *Opt* - Itération $i = 7$ - True

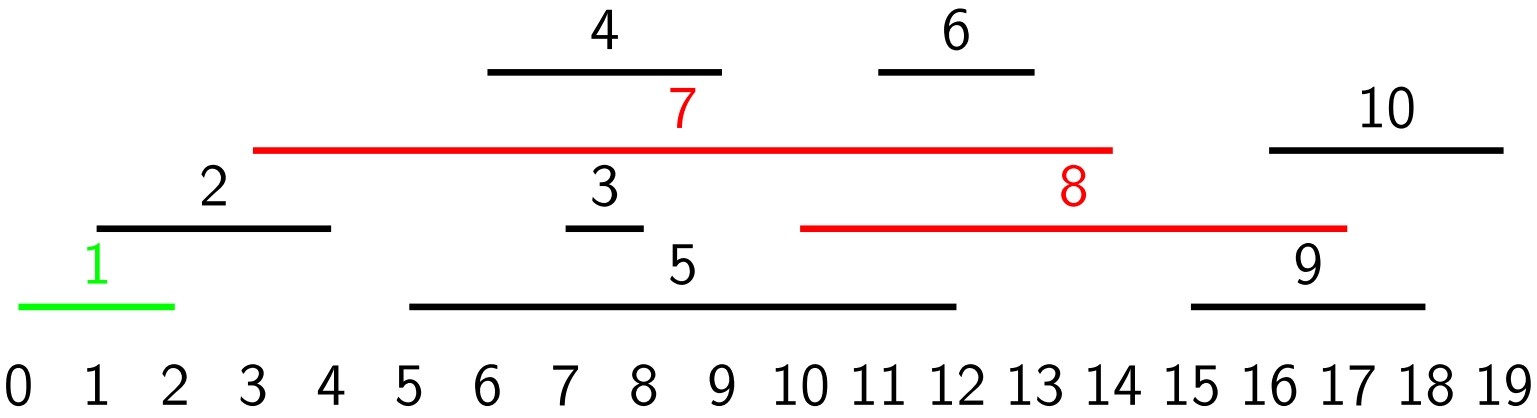
Un peu plus vite...



i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	3	20	20	20	20	4	6	20	**5**
False	20	20	2	20	20	20	20	3	7	20	20

Table – *Opt* - Itération $i = 7$ - False

Un peu plus vite...



i	0	1	2	3	4	5	6	7	8	9	10
True	0	20	3	20	20	20	20	4	6	20	**5**
False	20	20	2	20	20	20	20	3	7	20	20

Table – *Opt* - Fin

On obtient une solution optimale de coût 5.

Conclusion

Une complexité linéaire ?

L'étude de la complexité de cet algorithme polynomial reste encore à faire...

Conclusion

Une complexité linéaire ?

L'étude de la complexité de cet algorithme polynomial reste encore à faire...

D'autres problèmes à étudier

- Domination romaine totale
- Domination romaine indépendante
- Domination romaine signée

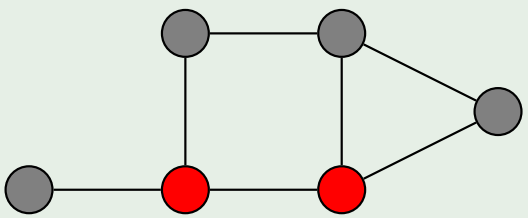
Merci de votre attention !

Domination romaine totale

Un chemin dominant

Aucun sommet de l'ensemble $V1 \cup V2$ n'est isolés dans le sous-graphe induit par la solution.

Exemple : Domination romaine totale



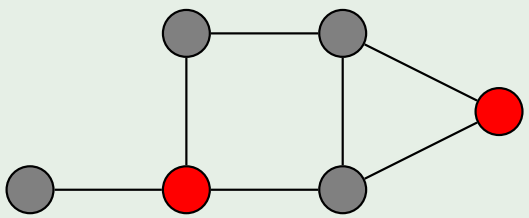
$$\gamma_{TRD}(G) = 4$$

Domination romaine indépendante

Une domination solitaire

Tous les sommets de l'ensemble $V1 \cup V2$ sont isolés dans le sous-graphe induit par la solution.

Exemple : Domination romaine indépendante



$$\gamma_{IRD}(G) = 4$$

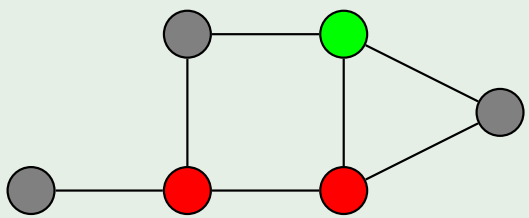
Domination romaine signée

Un entourage calculé

Une valeur pour chaque sommet est désormais définis comme suit :

- $V0 = -1$
- $V1 = 1$
- $V2 = 2$
- Chaque voisinage fermé d'un sommet doit posséder une valeur au moins égale à 1.
- Chaque sommet de type $V0$ doit avoir un voisin de type $V2$.

Exemple : Domination romaine signée



$$\gamma_{SRD}(G) = 5$$