# Algorithme de vérification du raisonnement énergétique en $O(n \log^2 n)$ et algorithme de filtrage en $O(n^2 \log n)$.

**Yanick Ouellet**

Claude-Guy Quimper

23 février 2018

Université Laval

# Introduction : Problème à résoudre

## Étudiant

- Plusieurs travaux à remettre dans la session

# Introduction : Problème à résoudre

## Étudiant

- Plusieurs travaux à remettre dans la session
- Estime le temps de chaque travail

# Introduction : Problème à résoudre

## Étudiant

- Plusieurs travaux à remettre dans la session
- Estime le temps de chaque travail
- Souhaite savoir la date à laquelle il doit commencer s'il veut remettre tous ses travaux à temps

# Introduction : Problème à résoudre

## Étudiant

- Plusieurs travaux à remettre dans la session
- Estime le temps de chaque travail
- Souhaite savoir la date à laquelle il doit commencer s'il veut remettre tous ses travaux à temps
- Malheureusement pour l'étudiant, c'est NP-Difficile :(

# Plan de la présentation

1. Introduction à la programmation par contraintes
2. Problèmes d'ordonnancement
3. Algorithmes et structures de données
4. Résultats expérimentaux

# Programmation par contraintes

## Objectif

- Résoudre des problèmes NP-Difficiles
- Résoudre des instances industrielles dans un temps raisonnable

# Modélisation du problème

## Variables

- Type (Entier, Réel, Booléen, Ensemble, etc.)
- Domaine

## Contraintes

- Restreint le domaine des variables
- Unaire (Une variable)
- Binaire (Deux variables)
- Globales

# Exemple d'un modèle

## Problème

$X + Y \leq Z$

## Variables

- $X \in \{0..10\}$
- $Y \in \{0..10\}$
- $Z \in \{0..10\}$

## Contrainte

- $X + Y \leq Z$

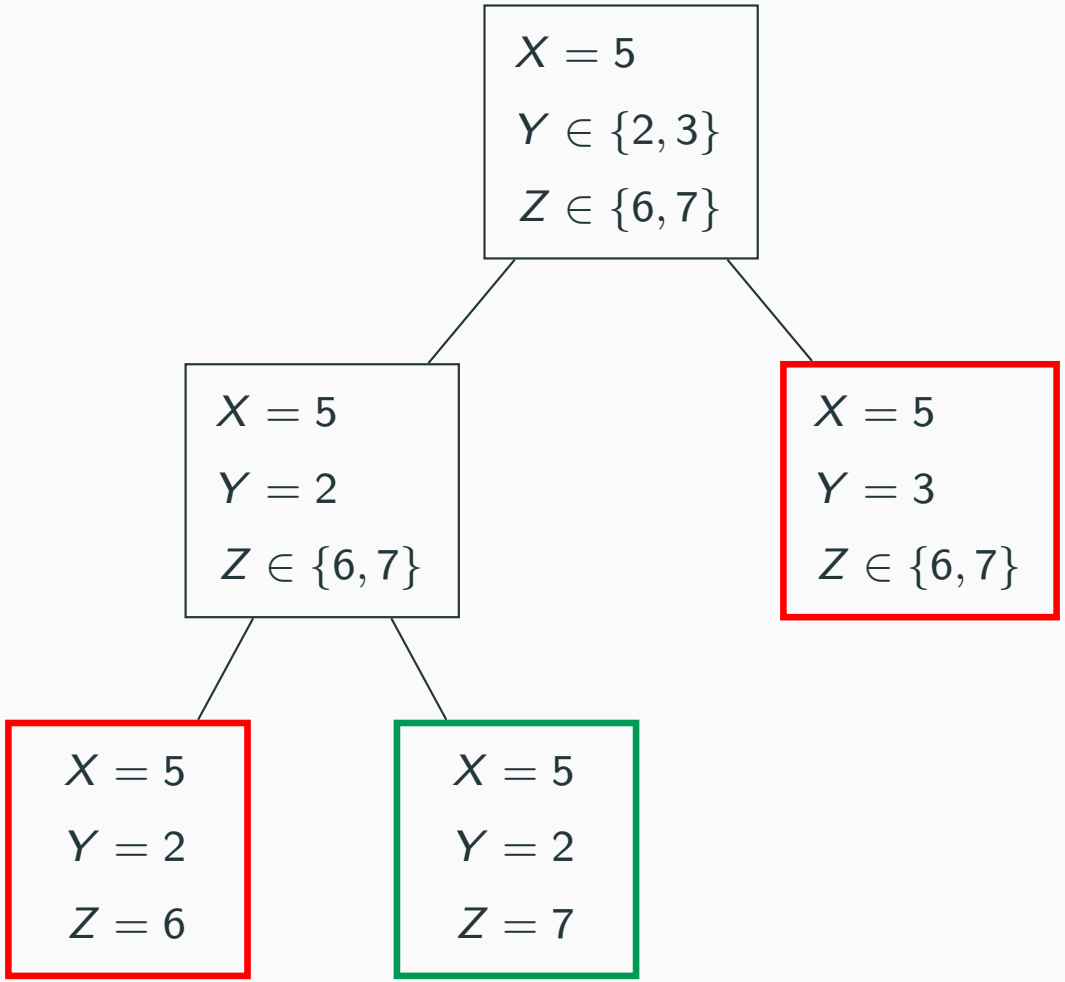
# Résolution d'un problème de programmation par contraintes

## Solveur

- Logiciel spécialisé qui prend en entrée un modèle et retourne une solution
- Construit un arbre de recherche
  - Assignation d'une valeur à une variable
  - Algorithmes de vérification
  - Algorithmes de filtrage

# Arbre de recherche

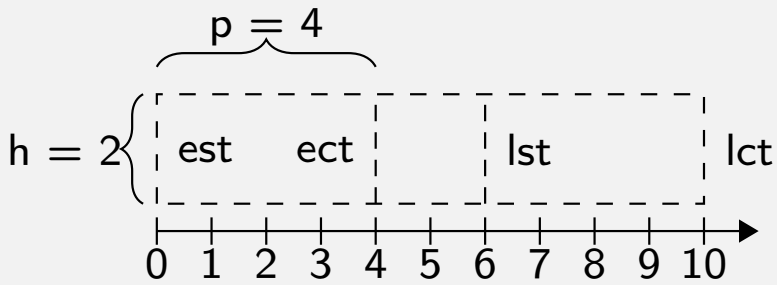
Contrainte $X + Y \leq Z$



# Ordonnancement

# Anatomie d'une tâche

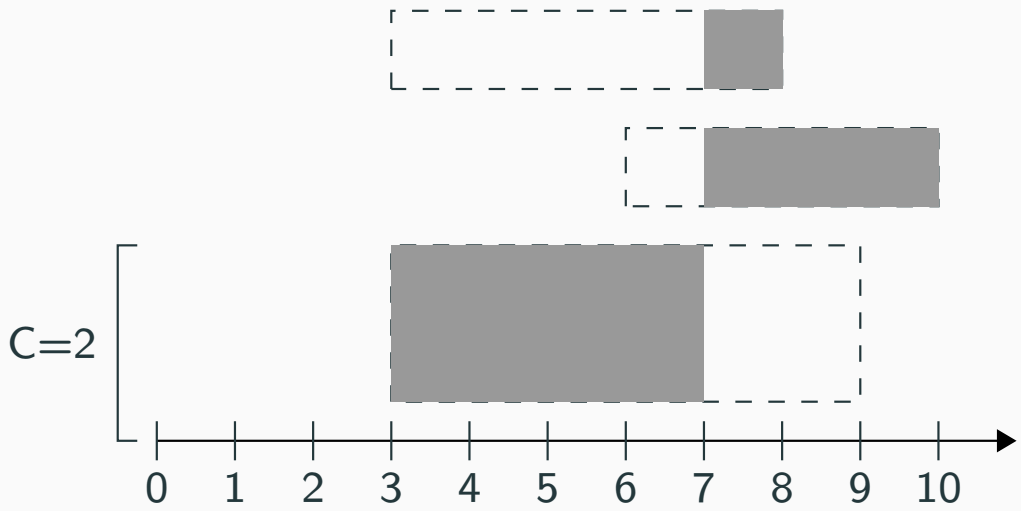
## Tâche




## Notation

- est : Earliest Starting Time
- ect : Earliest Completion Time
- lst : Latest Starting Time
- lct : Latest Completion Time
- p : Processing time
- h : Height

# Contrainte cumulative

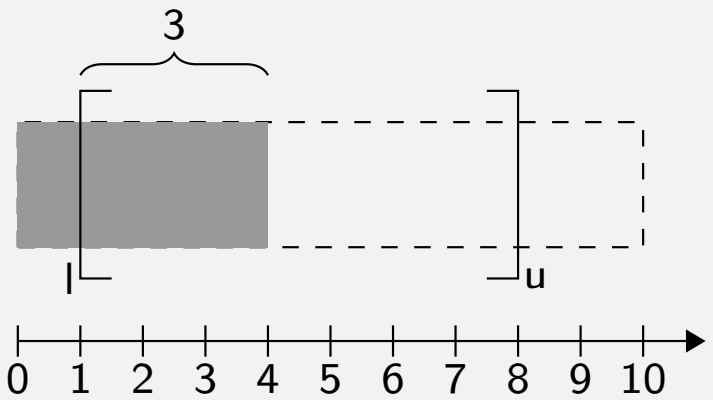



## Contrainte cumulative

- Ensemble de tâches
- La hauteur des tâches en cours d'exécution à un instant $t$ est au plus $C$

# Intersection minimum

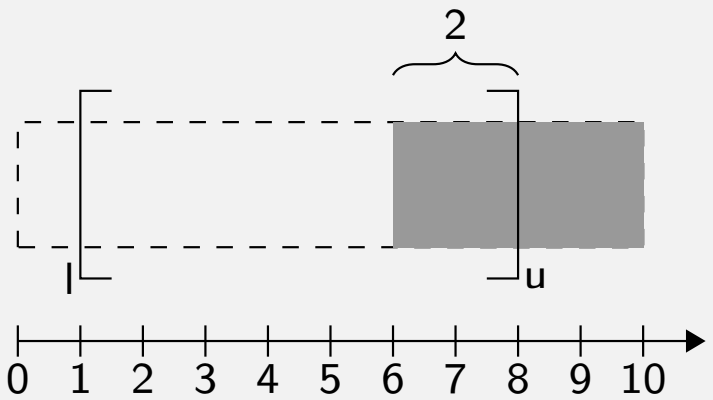
## Placement à gauche (Left Shift)




- $LS(1, 8) = 3$

# Intersection minimum

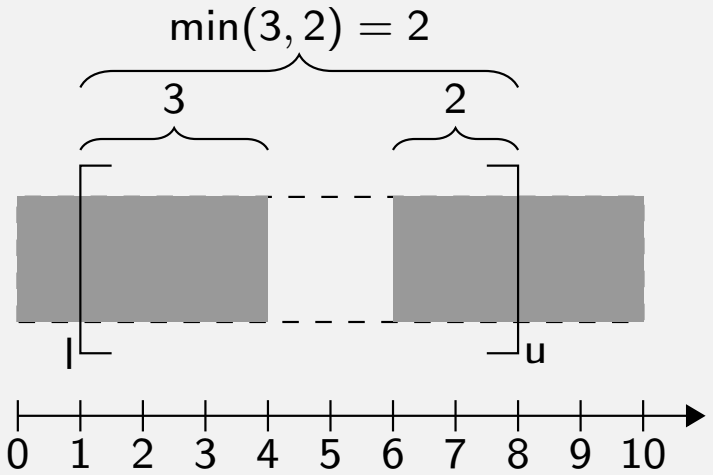
## Placement à droite (Right Shift)




- $LS(1, 8) = 3$
- $RS(1, 8) = 2$

# Intersection minimum

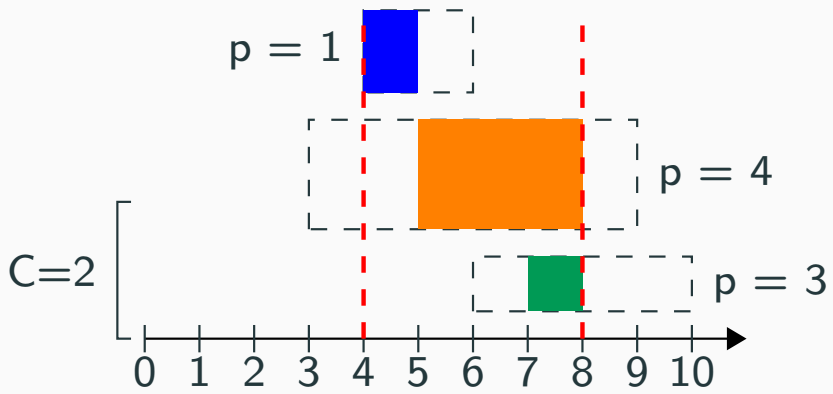
## Intersection minimum




- $LS(1, 8) = 3$
- $RS(1, 8) = 2$
- $MI(1, 8) = \min(LS(1, 8), RS(1, 8)) = 2$

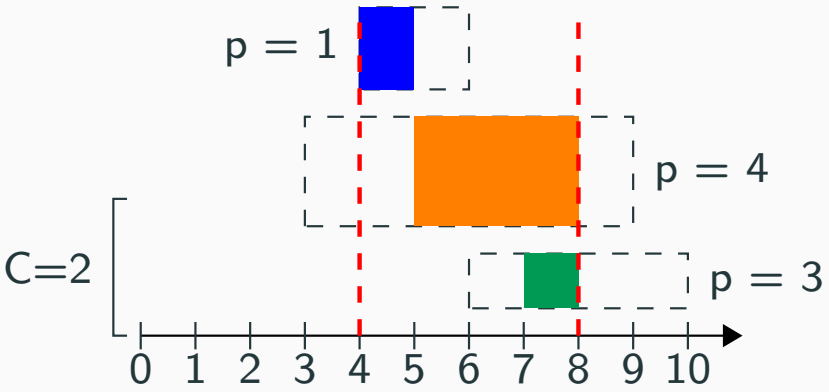
## Raisonnement énergétique

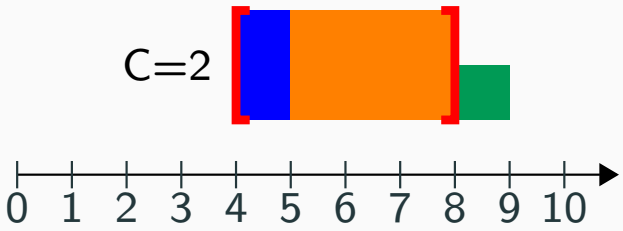
- $\text{Slack}(l, u) = C \cdot (u - l) - MI(l, u)$
- Si le Slack est négatif, il y a incohérence
- Il est suffisant de considérer $O(n^2)$ intervalles



## Raisonnement énergétique

- Slack$(l, u) = C \cdot (u - l) - MI(l, u)$
- Si le Slack est négatif, il y a incohérence
- Il est suffisant de considérer $O(n^2)$ intervalles



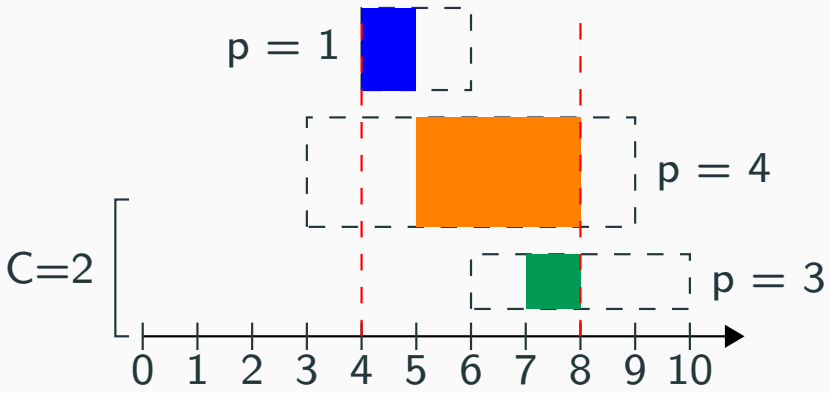
# Algorithme de vérification en $O(n \log^2 n)$

# Algorithme de vérification en $O(n \log^2 n)$

## Idée générale

- Utiliser la matrice du Slack
  - Lignes : Bornes inférieures d'un intervalle
  - Colonne : Bornes supérieures d'un intervalle
  - Une entrée représente le Slack dans un intervalle
- Si une entrée est inférieure à 0, il y a incohérence

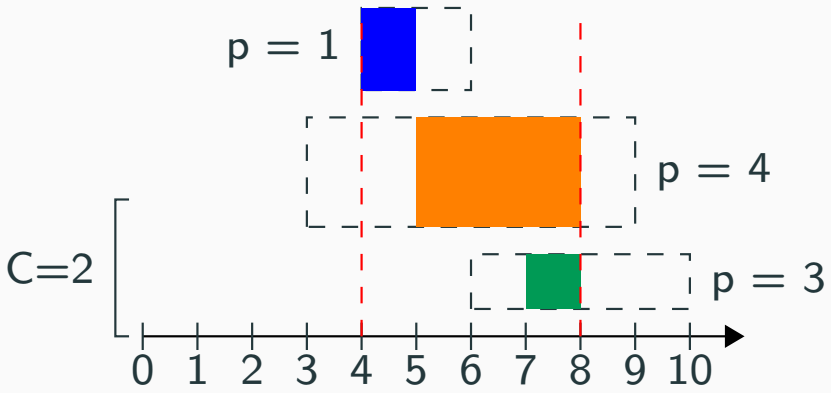
# Exemple

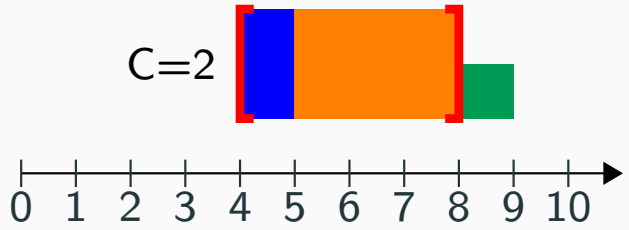



## Matrice

| | 1 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | | | 2 | 4 | 2 | 2 | 1 | 0 | 1 | 3 | 5 | 7 |
| 4 | | | | 2 | 0 | 0 | -1 | 0 | 1 | 3 | 5 | 7 |
| 5 | | | | | 0 | 0 | 1 | 2 | 3 | 5 | 7 | 9 |
| 6 | | | | | | 0 | 1 | 2 | 3 | 5 | 7 | 9 |
| 7 | | | | | | | 1 | 2 | 4 | 6 | 8 | 10 |
| 9 | | | | | | | | | 2 | 4 | 6 | 8 |

# Exemple




## Matrice

| | 1 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | | | 2 | 4 | 2 | 2 | 1 | 0 | 1 | 3 | 5 | 7 |
| 4 | | | | 2 | 0 | 0 | -1 | 0 | 1 | 3 | 5 | 7 |
| 5 | | | | | 0 | 0 | 1 | 2 | 3 | 5 | 7 | 9 |
| 6 | | | | | | 0 | 1 | 2 | 3 | 5 | 7 | 9 |
| 7 | | | | | | | 1 | 2 | 4 | 6 | 8 | 10 |
| 9 | | | | | | | | | 2 | 4 | 6 | 8 |

# Algorithme sous-quadratique

## Objectif

- Trouver le minimum en vérifiant $O(n \log n)$ entrées
- Calcul en $O(\log n)$ d'une entrée

## Matrice

| | 1 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | | | 2 | 4 | 2 | 2 | 1 | 0 | 1 | 3 | 5 | 7 |
| 4 | | | | 2 | 0 | 0 | -1 | 0 | 1 | 3 | 5 | 7 |
| 5 | | | | | 0 | 0 | 1 | 2 | 3 | 5 | 7 | 9 |
| 6 | | | | | | 0 | 1 | 2 | 3 | 5 | 7 | 9 |
| 7 | | | | | | | 1 | 2 | 4 | 6 | 8 | 10 |
| 9 | | | | | | | | | 2 | 4 | 6 | 8 |

# Algorithme sous-quadratique

## Objectif

- Trouver le minimum en vérifiant $O(n \log n)$ entrées
- Calcul en $O(\log n)$ d'une entrée

## Solution

- Utiliser des propriétés de la matrice

# Matrice de monge

## Matrice de Monge

- Soit une matrice $M$ de taille $n \times m$
- $M[i,j] - M[i+1,j] \leq M[i,j+1] - M[i+1,j+1]$
  $\forall 1 < i \leq n, 1 < j \leq m$

## Exemple

| | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 7 | 5 | 3 | 1 | 0 | 1 | 2 | 2 | 4 | 2 | | |
| 4 | 7 | 5 | 3 | 1 | 0 | -1 | 0 | 0 | 2 | | | |
| 5 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | 0 | | | | |
| 6 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | | | | | |
| 7 | 10 | 8 | 6 | 4 | 2 | 1 | | | | | | |
| 9 | 8 | 6 | 4 | 2 | | | | | | | | |

# Matrice de monge

## Matrice de Monge

- Soit une matrice $M$ de taille $n \times m$
- $M[i,j] - M[i+1,j] \leq M[i,j+1] - M[i+1,j+1]$
  $\forall 1 < i \leq n, 1 < j \leq m$

## Exemple

| | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 7 | 5 | 3 | 1 | 0 | 1 | 2 | 2 | 4 | 2 | | |
| 4 | 7 | 5 | 3 | 1 | 0 | -1 | 0 | 0 | 2 | | | |
| 5 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | 0 | | | | |
| 6 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | | | | | |
| 7 | 10 | 8 | 6 | 4 | 2 | 1 | | | | | | |
| 9 | 8 | 6 | 4 | 2 | | | | | | | | |

# Enveloppe d'une matrice de Monge

## Enveloppe

- L'enveloppe d'une colonne $j$ est la ligne $i$ où $M[i, j]$ est minimal
- Chaque ligne $i$ défini une fonction $f_i(j) = M[i, j]$
- Les fonctions de deux lignes se croisent au plus une fois

## Exemple

| | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 7 | 5 | 3 | 1 | 0 | 1 | 2 | 2 | 4 | 2 | | |
| 4 | 7 | 5 | 3 | 1 | 0 | -1 | 0 | 0 | 2 | | | |
| 5 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | 0 | | | | |
| 6 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | | | | | |
| 7 | 10 | 8 | 6 | 4 | 2 | 1 | | | | | | |

# Enveloppe d'une matrice de Monge

## Calcul de l'enveloppe

- Calcul ligne par ligne
- Recherche de la plus petite colonne où l'élément de la ligne courante est plus petit que ceux des lignes précédentes

## Exemple

| | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 7 | 5 | 3 | 1 | 0 | 1 | 2 | 2 | 4 | 2 | | |
| 4 | 7 | 5 | 3 | 1 | 0 | -1 | 0 | 0 | 2 | | | |
| 5 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | 0 | | | | |
| 6 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | | | | | |
| 7 | 10 | 8 | 6 | 4 | 2 | 1 | | | | | | |

# Enveloppe d'une matrice de Monge

## Calcul de l'enveloppe

- Calcul ligne par ligne
- Recherche de la plus petite colonne où l'élément de la ligne courante est plus petit que ceux des lignes précédentes

## Exemple

| | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 7 | 5 | 3 | 1 | 0 | 1 | 2 | 2 | 4 | 2 | | |
| 4 | 7 | 5 | 3 | 1 | 0 | -1 | 0 | 0 | 2 | | | |
| 5 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | 0 | | | | |
| 6 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | | | | | |
| 7 | 10 | 8 | 6 | 4 | 2 | 1 | | | | | | |

# Enveloppe d'une matrice de Monge

## Calcul de l'enveloppe

- Calcul ligne par ligne
- Recherche de la plus petite colonne où l'élément de la ligne courante est plus petit que ceux des lignes précédentes

## Exemple

| | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 7 | 5 | 3 | 1 | 0 | 1 | 2 | 2 | 4 | 2 | | |
| 4 | 7 | 5 | 3 | 1 | 0 | -1 | 0 | 0 | 2 | | | |
| 5 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | 0 | | | | |
| 6 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | | | | | |
| 7 | 10 | 8 | 6 | 4 | 2 | 1 | | | | | | |

# Algorithme de vérification

## Algorithme

- Calculer l'enveloppe à l'aide de recherches binaires sur les colonnes
- Vérifier, pour chaque colonne, si l'élément sur l'enveloppe de la colonne est négatif

## Exemple

| | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 7 | 5 | 3 | 1 | 0 | 1 | 2 | 2 | 4 | 2 | | |
| 4 | 7 | 5 | 3 | 1 | 0 | -1 | 0 | 0 | 2 | | | |
| 5 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | 0 | | | | |
| 6 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | | | | | |
| 7 | 10 | 8 | 6 | 4 | 2 | 1 | | | | | | |

# Algorithme de vérification

## Analyse

- Matrice de $O(n)$ lignes par $O(n^2)$ colonnes

# Algorithme de vérification

## Analyse

- Matrice de $O(n)$ lignes par $O(n^2)$ colonnes
- Jamais construite explicitement

# Algorithme de vérification

## Analyse

- Matrice de $O(n)$ lignes par $O(n^2)$ colonnes
- Jamais construite explicitement
- Seulement $O(n)$ colonnes sont traitées explicitement

# Algorithme de vérification

## Analyse

- Matrice de $O(n)$ lignes par $O(n^2)$ colonnes
- Jamais construite explicitement
- Seulement $O(n)$ colonnes sont traitées explicitement
- $O(n \log n)$ calculs d'éléments

# Algorithme de vérification

## Objectif

- Trouver le minimum en vérifiant $O(n \log n)$ entrées
- Calcul en $O(\log n)$ d'une entrée

## Exemple

| | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 7 | 5 | 3 | 1 | 0 | 1 | 2 | 2 | 4 | 2 | | |
| 4 | 7 | 5 | 3 | 1 | 0 | -1 | 0 | 0 | 2 | | | |
| 5 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | 0 | | | | |
| 6 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | | | | | |
| 7 | 10 | 8 | 6 | 4 | 2 | 1 | | | | | | |
| 9 | 8 | 6 | 4 | 2 | | | | | | | | |

# Algorithme de vérification

## Objectif

- Trouver le minimum en vérifiant $O(n \log n)$ entrées
- Calcul en $O(\log n)$ d'une entrée

## Exemple

| | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 7 | 5 | 3 | 1 | 0 | 1 | 2 | 2 | 4 | 2 | | |
| 4 | 7 | 5 | 3 | 1 | 0 | -1 | 0 | 0 | 2 | | | |
| 5 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | 0 | | | | |
| 6 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | | | | | |
| 7 | 10 | 8 | 6 | 4 | 2 | 1 | | | | | | |
| 9 | 8 | 6 | 4 | 2 | | | | | | | | |

# Algorithme de vérification

## Objectif

- Trouver le minimum en vérifiant $O(n \log n)$ entrées
- Calcul en $O(\log n)$ d'une entrée

## Solution

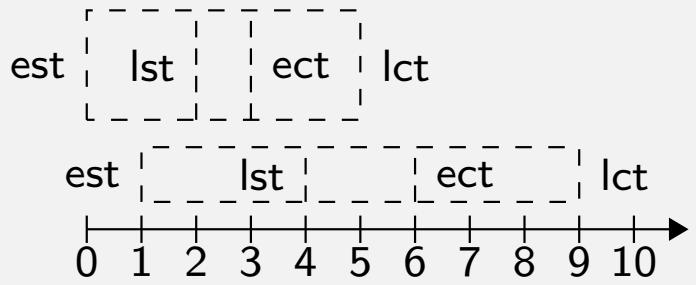
- Décomposer le calcul du slack en sous-problèmes
- Résoudre les sous-problèmes à l'aide de structure de données

# Calcul d'une entrée en $O(\log n)$

## Deux types d'énergie

- Énergie Fixe : Doit absolument s'exécuter entre le $\mathrm{lst}$ et le $\mathrm{ect}$
- Énergie Libre : Le reste de l'énergie

## Exemple

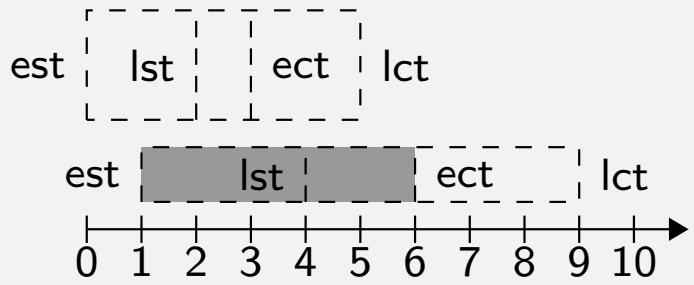


# Calcul d'une entrée en $O(\log n)$

## Deux types d'énergie

- Énergie Fixe : Doit absolument s'exécuter entre le $\mathrm{lst}$ et le $\mathrm{ect}$
- Énergie Libre : Le reste de l'énergie

## Exemple

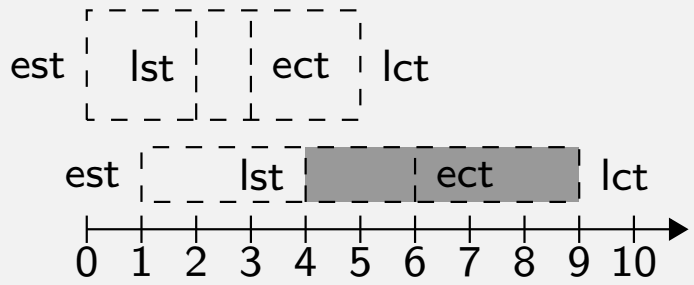


# Calcul d'une entrée en $O(\log n)$

## Deux types d'énergie

- Énergie Fixe : Doit absolument s'exécuter entre le $\mathrm{lst}$ et le $\mathrm{ect}$
- Énergie Libre : Le reste de l'énergie

## Exemple

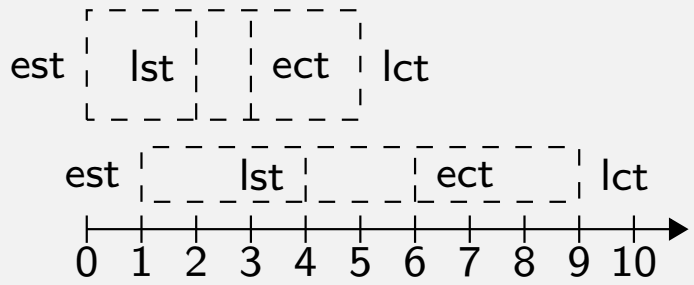


# Calcul d'une entrée en $O(\log n)$

## Décomposition en segments

- Segment de longueur 1 pour chaque unité d'Énergie Fixe

## Exemple

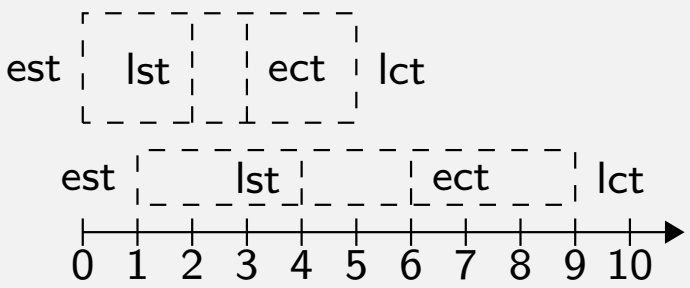


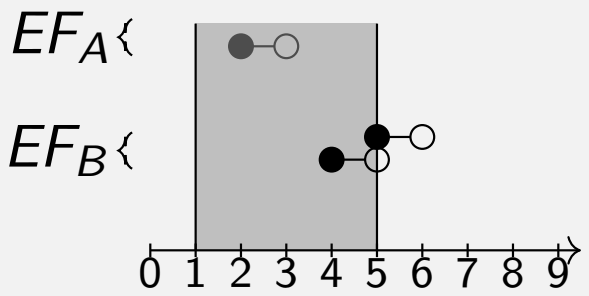
# Calcul d'une entrée en $O(\log n)$

## Décomposition en segments

- Segment de longueur 1 pour chaque unité d'Énergie Fixe

## Exemple



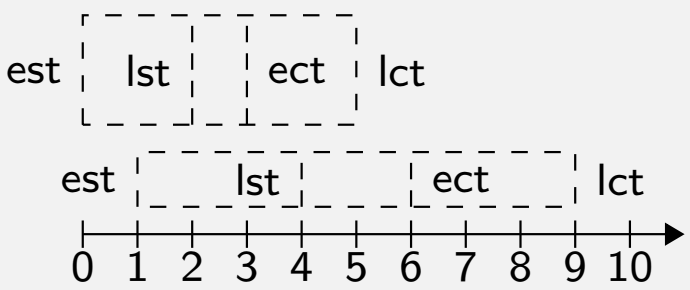



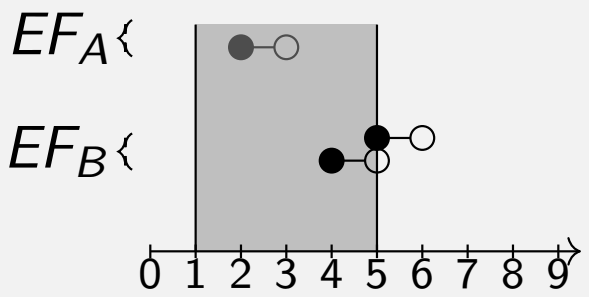
# Calcul d'une entrée en $O(\log n)$

## Décomposition en segments

- Segment de longueur 1 pour chaque unité d'Énergie Fixe
- Segment défini par les bornes du plus petit interval incluant une unité d'Énergie Libre

## Exemple



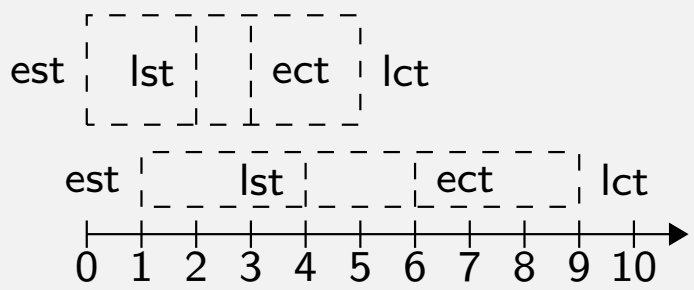



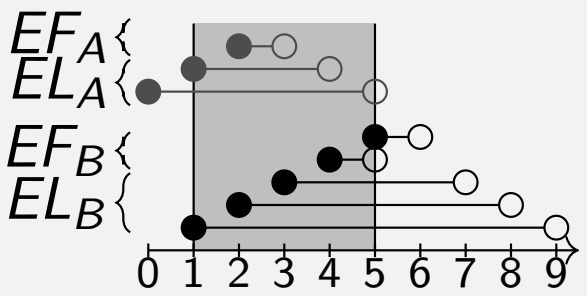
# Calcul d'une entrée en $O(\log n)$

## Décomposition en segments

- Segment de longueur 1 pour chaque unité d'Énergie Fixe
- Segment défini par les bornes du plus petit interval incluant une unité d'Énergie Libre

## Exemple



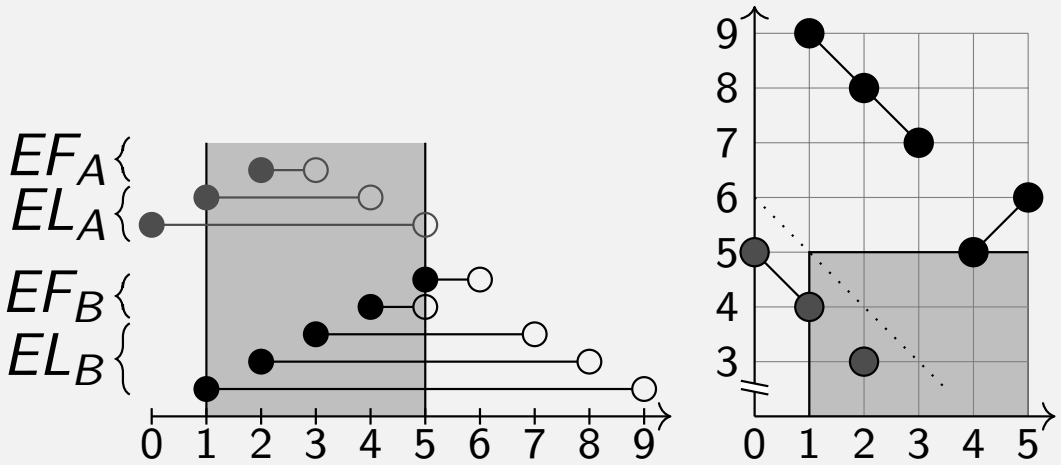



# Calcul d'une entrée en $O(logn)$

## Transformation de ligne à plan

- Les segments sont transformés en points
  - L'orgine du segment est l'abscisse
  - La fin du segment est l'ordonnée

## Exemple



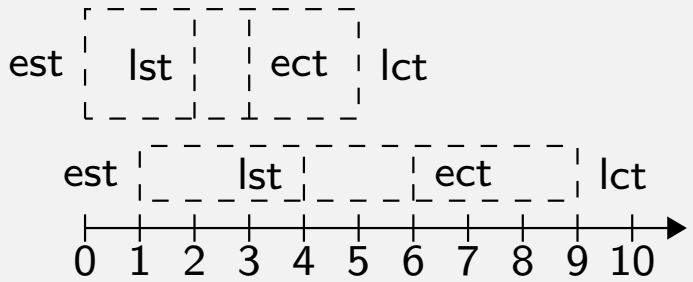
# Calcul de l'Énergie Fixe avec les Sommes Partielles

## Construction des Sommes Partielles

- Vecteur T de temps où l'Énergie Fixe commence et termine
- Vecteur Y : $Y_t$ est la somme des hauteurs des tâches ayant de l'Énergie Fixe au temps $T_t$
- Vecteur Z : $Z_t$ est la somme de l'Énergie Fixe dans l'intervalle $[0, T_t)$

## Exemple

| | | | | |
|---|---|---|---|---|
| T : | 2 | 3 | 4 | 6 |
| Y : | 2 | 0 | 1 | 0 |
| Z : | 0 | 2 | 2 | 4 |



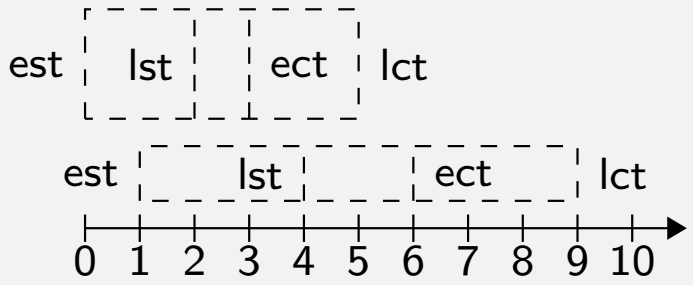
# Calcul de l'Énergie Fixe avec les Sommes Partielles

## Requête un intervalle

- Requête sur l'intervalle $[l, u]$
- Recherche binaire de $t_1$ tel que $T_{t_1-1} < l \leq T_{t_1}$
- Recherche binaire de $t_2$ tel que $T_{t_2} \leq u < T_{t_2+1}$
- Énergie Fixe $= Z_{t_2} + Y_{t_2}(u - T_{t_2}) - (Z_{t_1} + Y_{t_1}(l - T_{t_1}))$

## Exemple

| | | | | |
|---|---|---|---|---|
| T : | 2 | 3 | 4 | 6 |
| Y : | 2 | 0 | 1 | 0 |
| Z : | 0 | 2 | 2 | 4 |



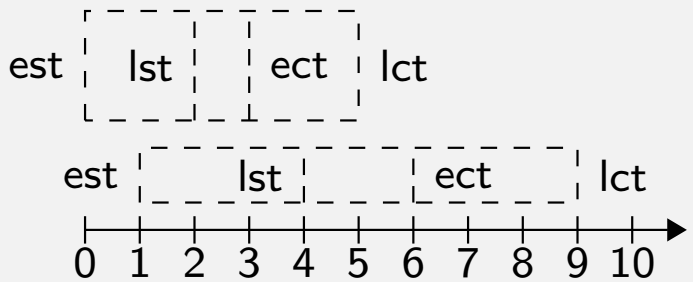
# Calcul de l'Énergie Fixe avec les Sommes Partielles

## Requête un intervalle

- Requête sur l'intervalle $[3, 5]$
- Recherche binaire de $t_1$ tel que $T_{t_1-1} < l \leq T_{t_1}$
- Recherche binaire de $t_2$ tel que $T_{t_2} \leq u < T_{t_2+1}$
- Énergie Fixe $= Z_{t_2} + Y_{t_2}(u - T_{t_2}) - (Z_{t_1} + Y_{t_1}(l - T_{t_1}))$

## Exemple

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| T : | 2 | 3 | 4 | 6 |
| Y : | 2 | 0 | 1 | 0 |
| Z : | 0 | 2 | 2 | 4 |



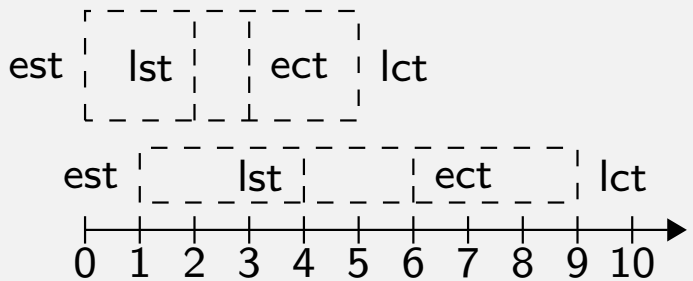
# Calcul de l'Énergie Fixe avec les Sommes Partielles

## Requête un intervalle

- Requête sur l'intervalle $[3, 5]$
- Recherche binaire de $t_1$ tel que $T_{t_1-1} < 3 \leq T_{t_1}$
- Recherche binaire de $t_2$ tel que $T_{t_2} \leq u < T_{t_2+1}$
- Énergie Fixe $= Z_{t_2} + Y_{t_2}(u - T_{t_2}) - (Z_{t_1} + Y_{t_1}(l - T_{t_1}))$

## Exemple

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| T : | 2 | 3 | 4 | 6 |
| Y : | 2 | 0 | 1 | 0 |
| Z : | 0 | 2 | 2 | 4 |



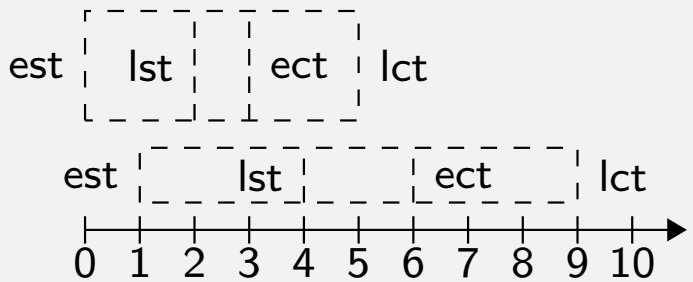
# Calcul de l'Énergie Fixe avec les Sommes Partielles

## Requête un intervalle

- Requête sur l'intervalle $[3, 5]$
- Recherche binaire de $t_1 = 1$ tel que $T_{t_1-1} < 3 \leq T_{t_1}$
- Recherche binaire de $t_2$ tel que $T_{t_2} \leq u < T_{t_2+1}$
- Énergie Fixe $= Z_{t_2} + Y_{t_2}(u - T_{t_2}) - (Z_{t_1} + Y_{t_1}(l - T_{t_1}))$

## Exemple

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| T : | 2 | 3 | 4 | 6 |
| Y : | 2 | 0 | 1 | 0 |
| Z : | 0 | 2 | 2 | 4 |



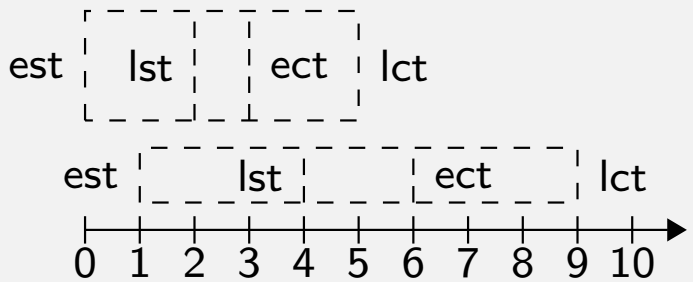
# Calcul de l'Énergie Fixe avec les Sommes Partielles

## Requête un intervalle

- Requête sur l'intervalle $[3, 5]$
- Recherche binaire de $t_1 = 1$ tel que $T_{t_1-1} < 3 \leq T_{t_1}$
- Recherche binaire de $t_2$ tel que $T_{t_2} \leq 5 < T_{t_2+1}$
- Énergie Fixe $= Z_{t_2} + Y_{t_2}(u - T_{t_2}) - (Z_{t_1} + Y_{t_1}(l - T_{t_1}))$

## Exemple

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| T : | 2 | 3 | 4 | 6 |
| Y : | 2 | 0 | 1 | 0 |
| Z : | 0 | 2 | 2 | 4 |



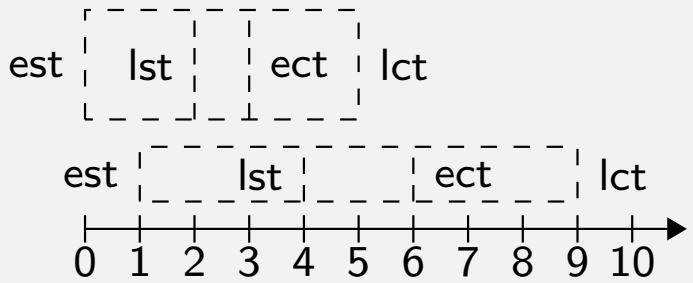
# Calcul de l'Énergie Fixe avec les Sommes Partielles

## Requête un intervalle

- Requête sur l'intervalle $[3, 5]$
- Recherche binaire de $t_1 = 1$ tel que $T_{t_1-1} < 3 \leq T_{t_1}$
- Recherche binaire de $t_2 = 2$ tel que $T_{t_2} \leq 5 < T_{t_2+1}$
- Énergie Fixe $= Z_{t_2} + Y_{t_2}(u - T_{t_2}) - (Z_{t_1} + Y_{t_1}(l - T_{t_1}))$

## Exemple

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| T : | 2 | 3 | 4 | 6 |
| Y : | 2 | 0 | 1 | 0 |
| Z : | 0 | 2 | 2 | 4 |



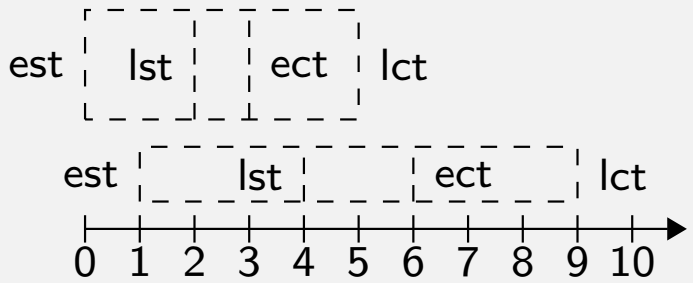
# Calcul de l'Énergie Fixe avec les Sommes Partielles

## Requête un intervalle

- Requête sur l'intervalle $[3, 5]$
- Recherche binaire de $t_1 = 1$ tel que $T_{t_1-1} < 3 \leq T_{t_1}$
- Recherche binaire de $t_2 = 2$ tel que $T_{t_2} \leq 5 < T_{t_2+1}$
- Énergie Fixe $= Z_{t_2} + Y_{t_2}(u - T_{t_2}) - (Z_{t_1} + Y_{t_1}(l - T_{t_1}))$
- EF $= 2 + 1\ (5 - 4)$

## Exemple

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| T : | 2 | 3 | 4 | 6 |
| Y : | 2 | 0 | 1 | 0 |
| Z : | 0 | 2 | 2 | 4 |



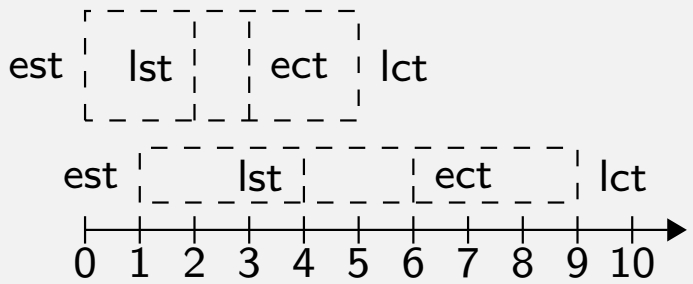
# Calcul de l'Énergie Fixe avec les Sommes Partielles

## Requête un intervalle

- Requête sur l'intervalle $[3, 5]$
- Recherche binaire de $t_1 = 1$ tel que $T_{t_1-1} < 3 \leq T_{t_1}$
- Recherche binaire de $t_2 = 2$ tel que $T_{t_2} \leq 5 < T_{t_2+1}$
- Énergie Fixe $= Z_{t_2} + Y_{t_2}(u - T_{t_2}) - (Z_{t_1} + Y_{t_1}(l - T_{t_1}))$
- EF $= 2 + 1\ (5 - 4)$ - $(2 + 0\ (3 - 3))$

## Exemple

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| T : | 2 | 3 | 4 | 6 |
| Y : | 2 | 0 | 1 | 0 |
| Z : | 0 | 2 | 2 | 4 |



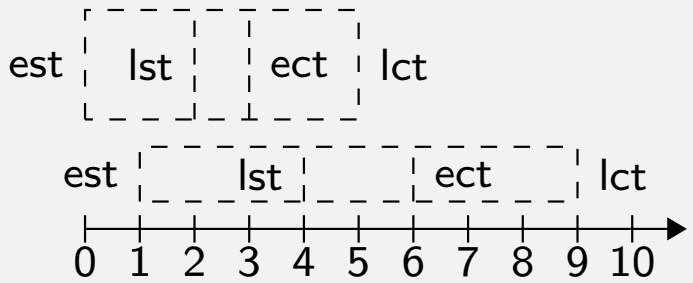
# Calcul de l'Énergie Fixe avec les Sommes Partielles

## Requête un intervalle

- Requête sur l'intervalle $[3, 5]$
- Recherche binaire de $t_1 = 1$ tel que $T_{t_1-1} < 3 \leq T_{t_1}$
- Recherche binaire de $t_2 = 2$ tel que $T_{t_2} \leq 5 < T_{t_2+1}$
- Énergie Fixe $= Z_{t_2} + Y_{t_2}(u - T_{t_2}) - (Z_{t_1} + Y_{t_1}(l - T_{t_1}))$
- EF $= 2 + 1\ (5 - 4)$ - $(2 + 0\ (3 - 3))$ $= 3 - 2 = 1$

## Exemple

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| T : | 2 | 3 | 4 | 6 |
| Y : | 2 | 0 | 1 | 0 |
| Z : | 0 | 2 | 2 | 4 |



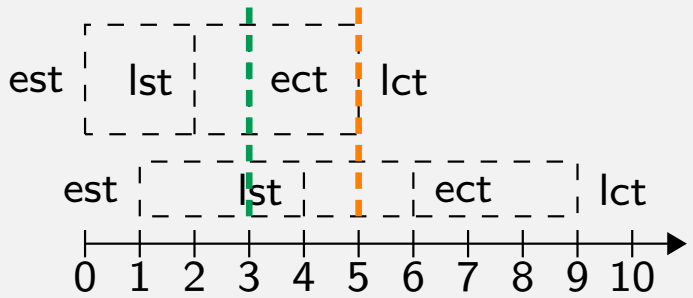
# Calcul de l'Énergie Fixe avec les Sommes Partielles

## Requête un intervalle

- Requête sur l'intervalle $[3, 5]$
- Recherche binaire de $t_1 = 1$ tel que $T_{t_1-1} < 3 \leq T_{t_1}$
- Recherche binaire de $t_2 = 2$ tel que $T_{t_2} \leq 5 < T_{t_2+1}$
- Énergie Fixe $= Z_{t_2} + Y_{t_2}(u - T_{t_2}) - (Z_{t_1} + Y_{t_1}(l - T_{t_1}))$
- EF $= 2 + 1\ (5 - 4)$ - $(2 + 0\ (3 - 3))$ $= 3 - 2 = 1$

## Exemple

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| T : | 2 | 3 | 4 | 6 |
| Y : | 2 | 0 | 1 | 0 |
| Z : | 0 | 2 | 2 | 4 |

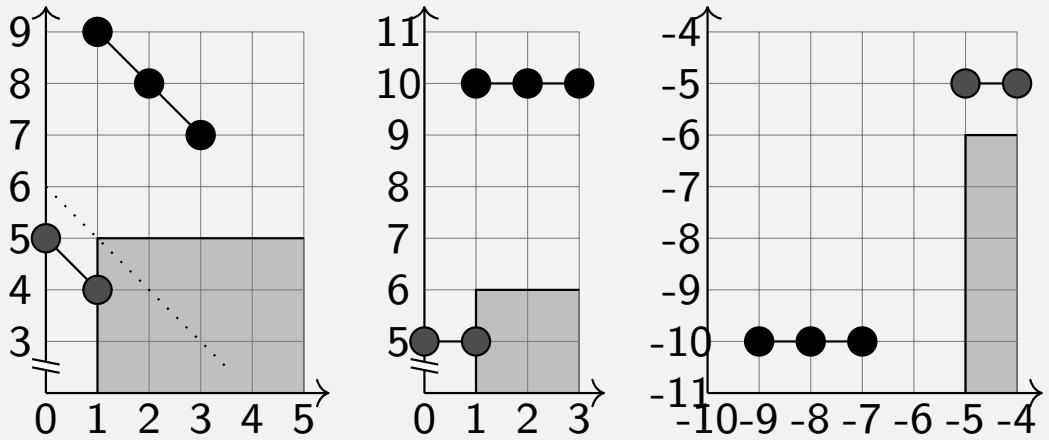


# Calcul de l'Énergie Libre

## Transformations géométriques

- Séparation de $EL$ en deux : $EL^1$ et $EL^2$
- Chaque point $(a, b) \in EL$ génère deux nouveaux points
  - $(a, a + b) \in EL^1$
  - $(-b, -a - b) \in EL^2$
- On applique la même transformation à la requête

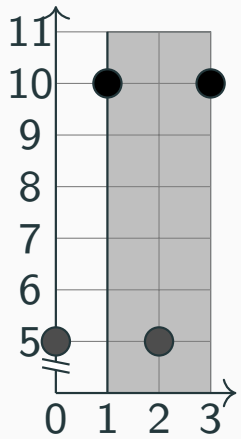
## Exemple



# Range-Tree à une dimension

## Range-Tree à une dimension

- Requête $Q(\chi)$ : somme pondérée des points dont l'abscisse est plus grande ou égale à $\chi$
- Chaque point est une feuille
- Chaque noeud résume ses enfants



# Range-Tree à une dimension

## Range-Tree à une dimension

- Requête $Q(\chi)$ : somme pondérée des points dont l'abscisse est plus grande ou égale à $\chi$
- Chaque point est une feuille
- Chaque noeud résume ses enfants

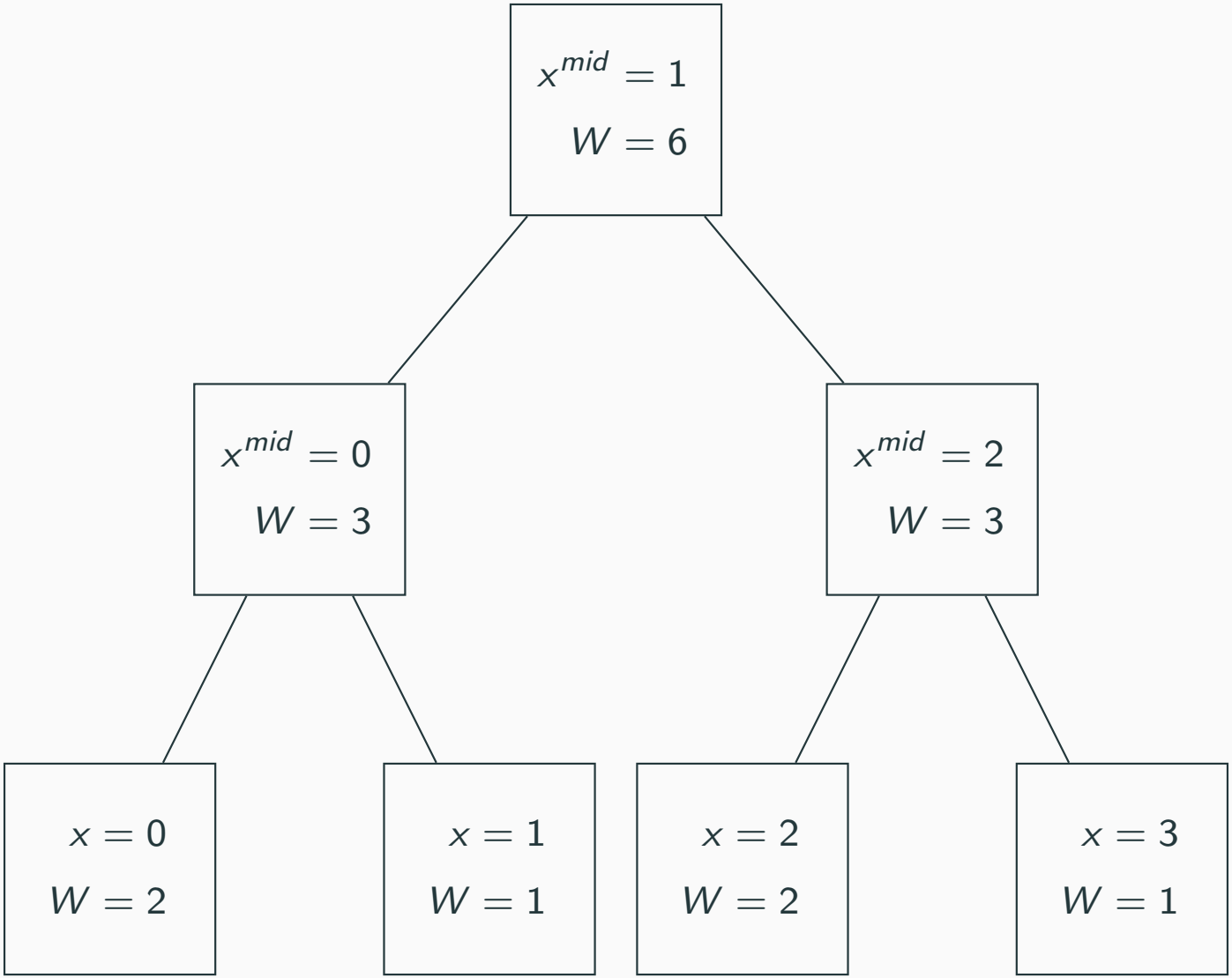
## Feuille

$x$ : Abcsisse du point
$W$ : Poids du point

## Noeud

$x^{mid}$ : Plus grand $x$ dans le sous-arbre de gauche
$W$ : Somme des $W$ des enfants

# Range-Tree à une dimension



# Range-Tree à une dimension

## Requête

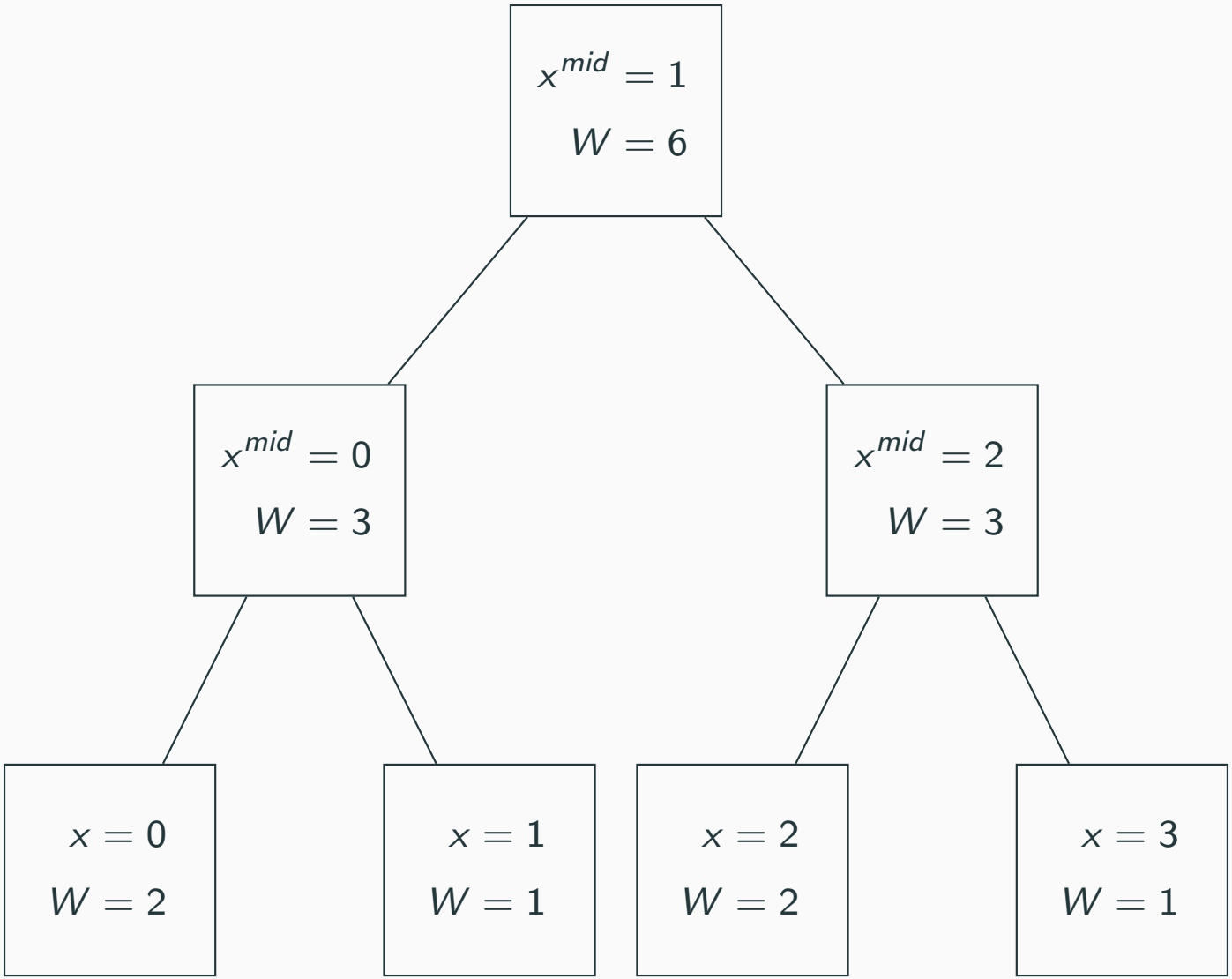
- Requête $Q(\chi)$ : somme pondérée des points dont l'abscisse est plus grande ou égale à $\chi$
- Algorithme
  1. $v = Racine$
  2. Tant que $v$ n'est pas une feuille
     - Si $x_v^{mid} \geq \chi$, on rapporte $right(v)$ et $v = left(v)$
     - Sinon $v = right(v)$
  3. Si $x_v \geq \chi$, on rapporte $v$

## Rapporter un noeud $v$

- Additionner $W_v$ au total

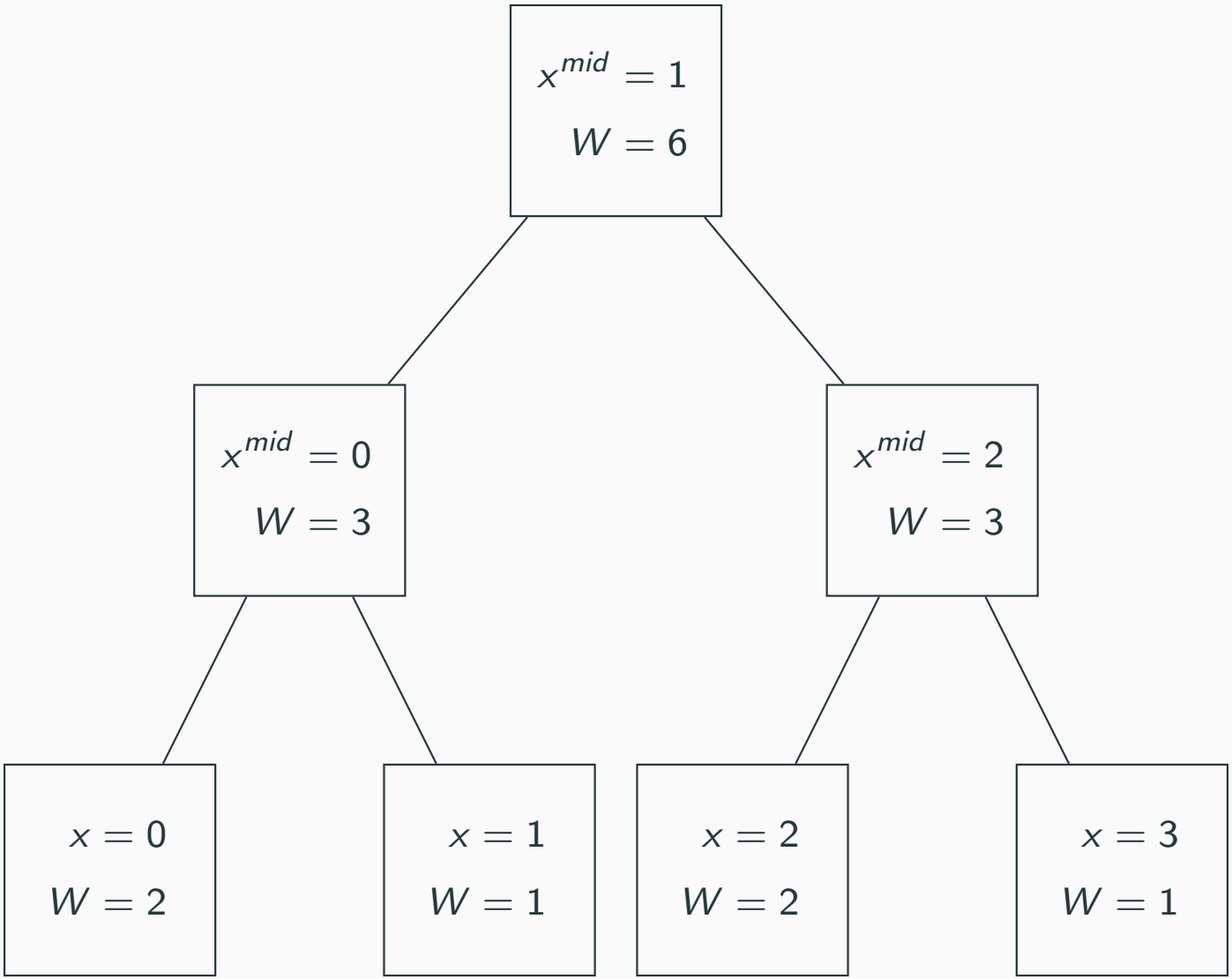
# Range-Tree à une dimension

Exemple avec $Q(1)$



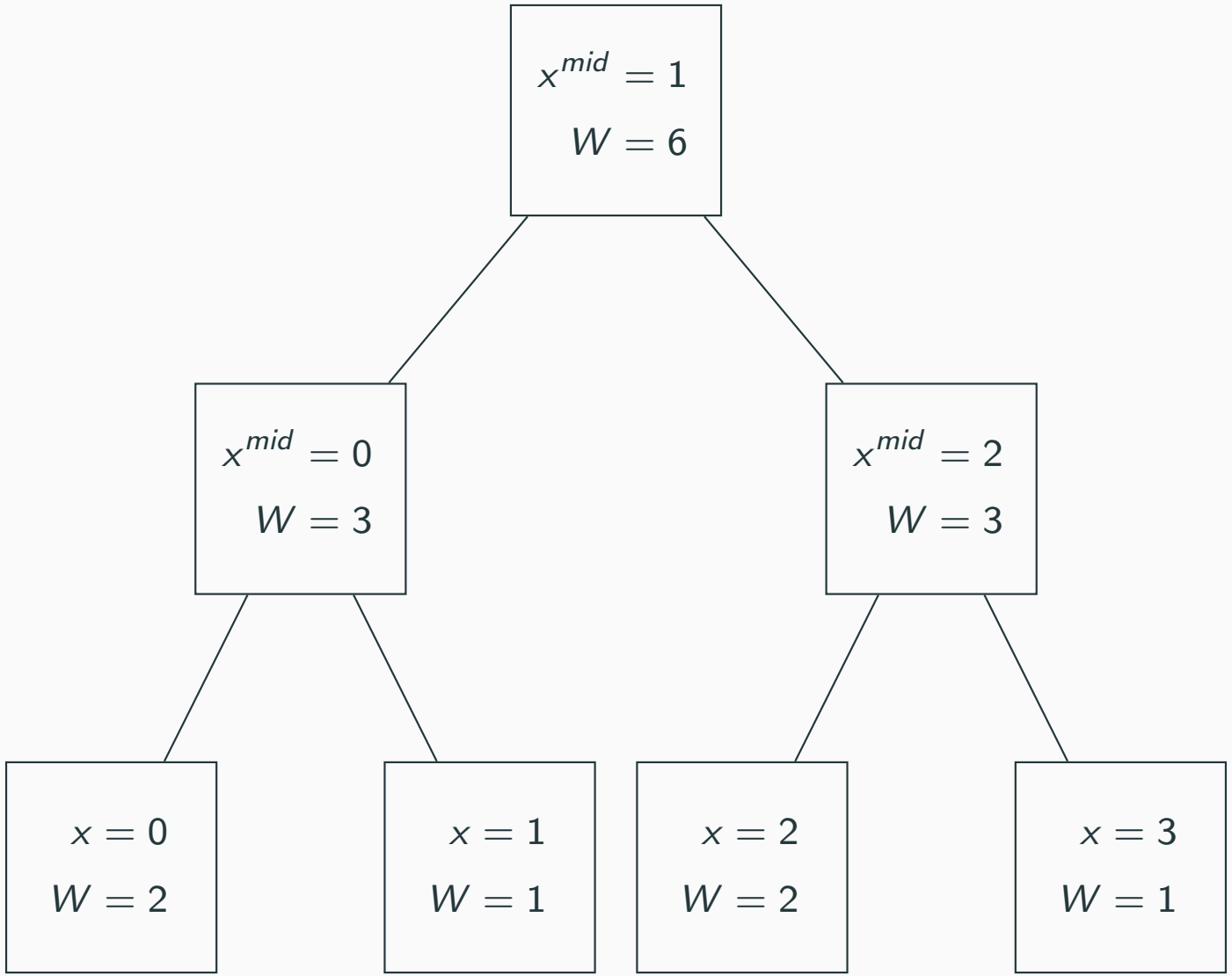
# Range-Tree à une dimension

Exemple avec $Q(1) = 3$



# Range-Tree à une dimension

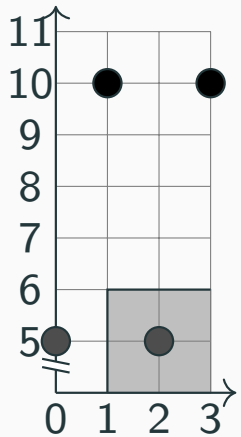
Exemple avec $Q(1) = 3 + 1 = 4$



# Range-Tree à deux dimensions

## Range-Tree à une dimension

- Requête $Q(\chi, \gamma)$ : somme pondérée des points avec l'abscisse $\geq \chi$ et l'ordonnée $\leq \gamma$
- Nouveaux vecteurs :
  - $Y$ : Ordonnées des points (vecteur trié)
  - $W$ : Poids
  - $L$ et $R$ : Pointeurs vers des éléments $Y$ des enfants



# Range-Tree à deux dimensions

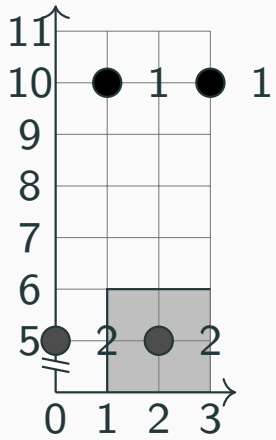
## Range-Tree à une dimension

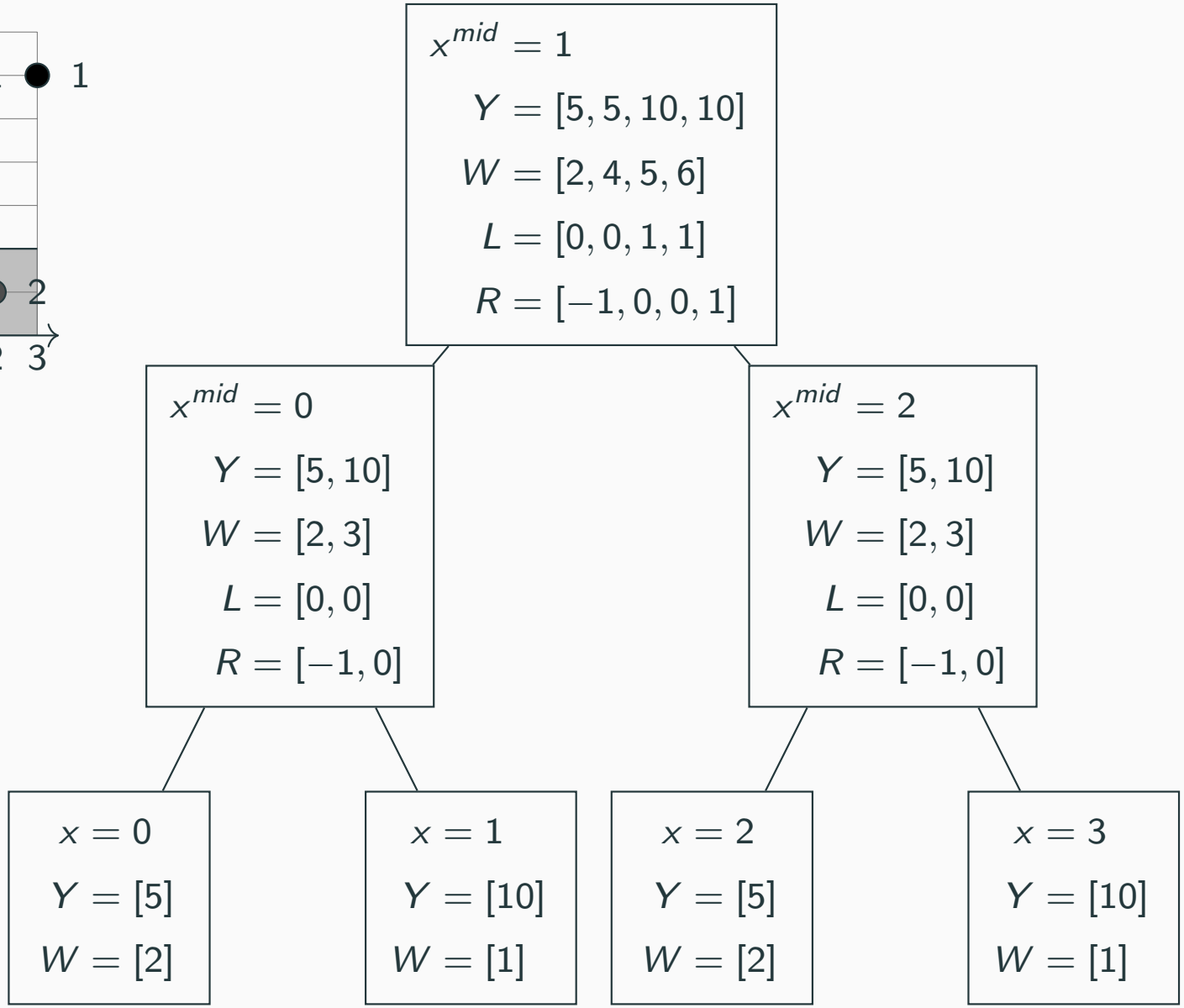
- Requête $Q(\chi, \gamma)$ : somme pondérée des points avec l'abscisse $\geq \chi$ et l'ordonnée $\leq \gamma$
- Nouveaux vecteurs :
  - $Y$ : Ordonnées des points (vecteur trié)
  - $W$ : Poids
  - $L$ et $R$ : Pointeurs vers des éléments $Y$ des enfants

## Construction des vecteurs

- Bottom-up, similaire à Fusion du tri fusion
- $W_i$ : Somme des poids des points dont l'ordonnée est inférieure à $Y_i$
- $L_i$ : Pointe vers la plus grande valeur plus petite ou égale à $Y_i$ dans le vecteur $Y$ de l'enfant gauche

# Range-Tree à deux dimensions



# Range-Tree à deux dimensions

## Requête

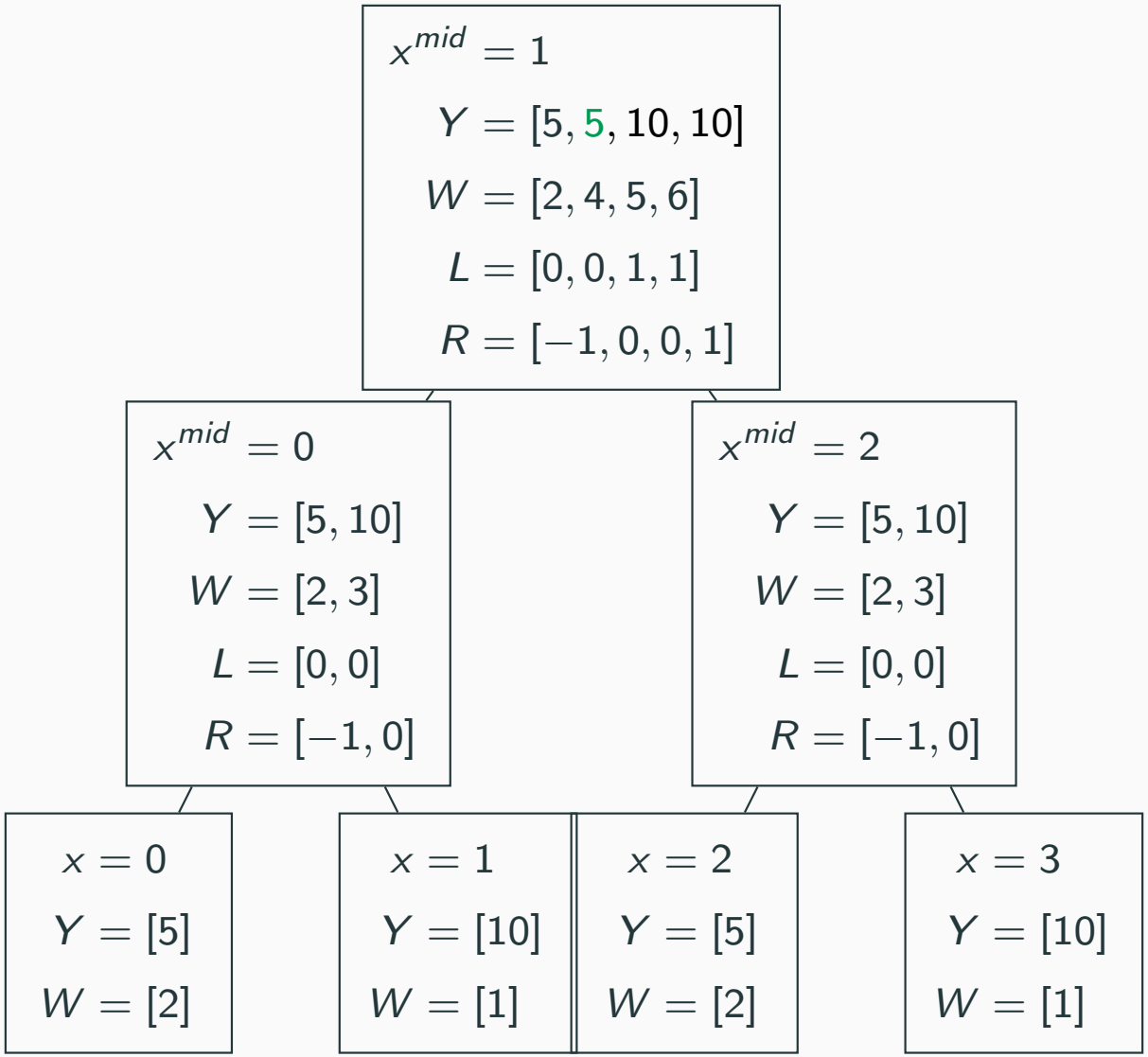
- Recherche du plus grand $i$ tel que $Y_i \leq \gamma$ à la racine
- Lors du parcours
  - $i = L_i$ si on descend à gauche
  - $i = R_i$ si on descend à droite

## Rapporter un noeud $v$

- Additionner $W_i$ à la somme

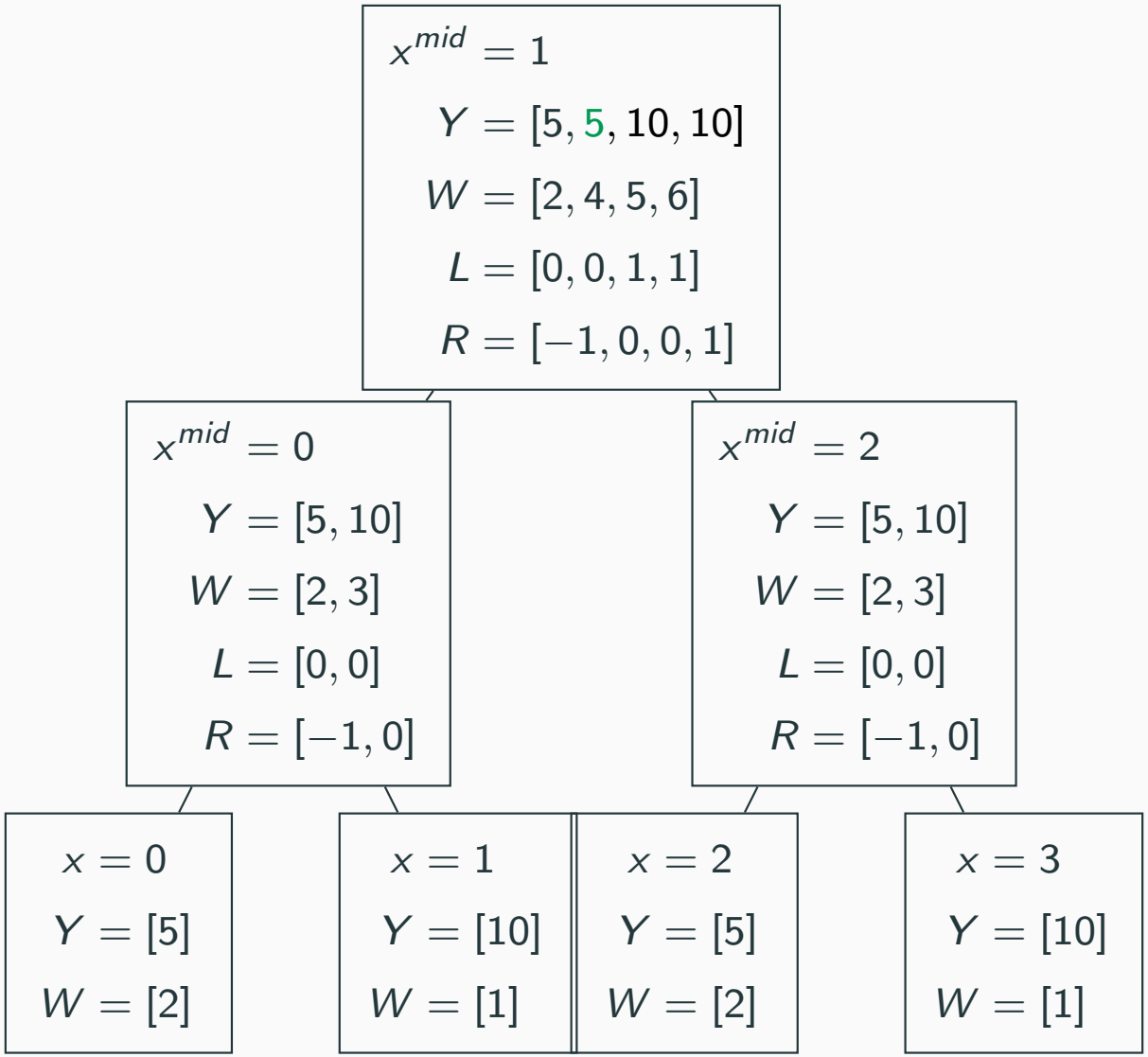
# Range-Tree à deux dimensions

Exemple avec $Q(1, 6)$



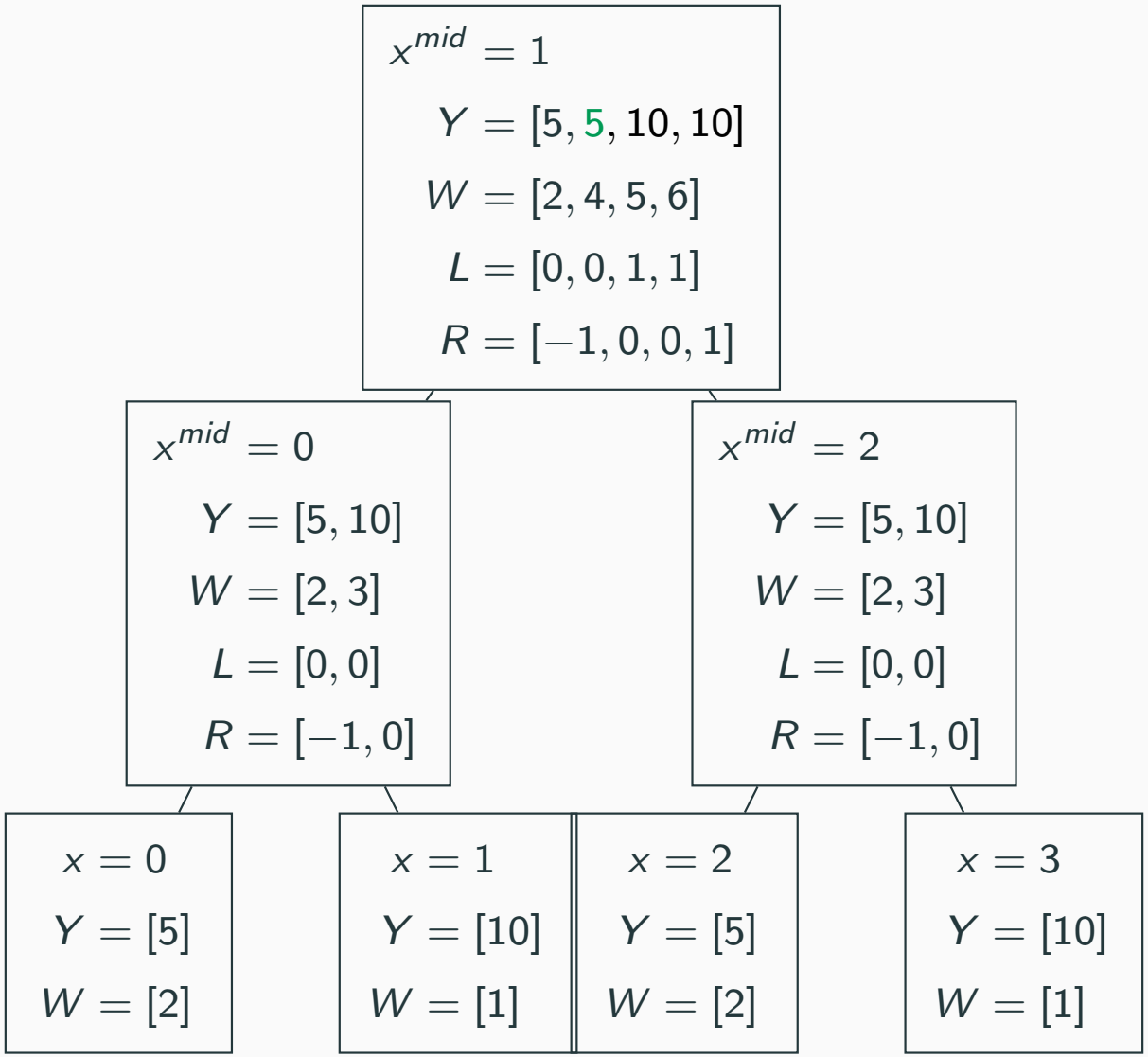
# Range-Tree à deux dimensions

Exemple avec $Q(1, 6)$



# Range-Tree à deux dimensions

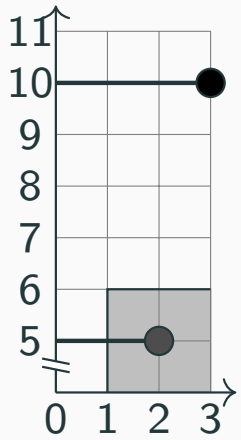
Exemple avec $Q(1, 6) = 2$



# Range-Tree à deux dimension

## Requête sur des rayons

- Requête $Q(\chi, \gamma)$ : somme pondérée des portions de rayons plus grands ou égal à $\chi$ et plus petits ou égal à $\gamma$
- Rayon : $\langle -\infty, x, y, w \rangle$



# Range-Tree à deux dimension

## Requête sur des rayons

- Requête $Q(\chi, \gamma)$ : somme pondérée des portions de rayons plus grands ou égal à $\chi$ et plus petits ou égal à $\gamma$
- Rayon : $\langle -\infty, x, y, w \rangle$

## Nouveaux attributs

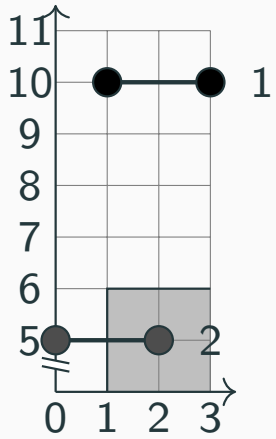
- $x^{min}$ : Plus petit $x$ d'une feuille
- $\Sigma_i$ : Somme pondérée à partir de $x^{min}$ de tous les rayons plus petits ou égal à $Y_i$

# Range-Tree à deux dimensions

## Transformation des segments en rayons

- Chaque segments $\langle x_i, x'_i, y_i, w_i \rangle$ génère deux rayons
  - $\langle -\infty, x_i, y_i, -w_i \rangle$
  - $\langle -\infty, x'_i, y_i, w_i \rangle$
- Avant $x_i$, les poids s'annulent

# Range-Tree à deux dimensions




$x^{mid} = 1$
$Y = [5, 5, 10, 10]$
$W = [-2, 0, -1, 0]$
$x^{min} = 0$
$\Sigma = [0, 4, 3, 6]$

$x^{mid} = 0$
$Y = [5, 10]$
$W = [-2, -3]$
$x^{min} = 0$
$\Sigma = [0, -1]$

$x^{mid} = 2$
$Y = [5, 10]$
$W = [2, 3]$
$x^{min} = 2$
$\Sigma = [0, 1]$

$x = 0$
$Y = [5]$
$W = [-2]$

$x = 1$
$Y = [10]$
$W = [-1]$

$x = 2$
$Y = [5]$
$W = [2]$

$x = 3$
$Y = [10]$
$W = [1]$

# Range-Tree à deux dimensions

## Requête

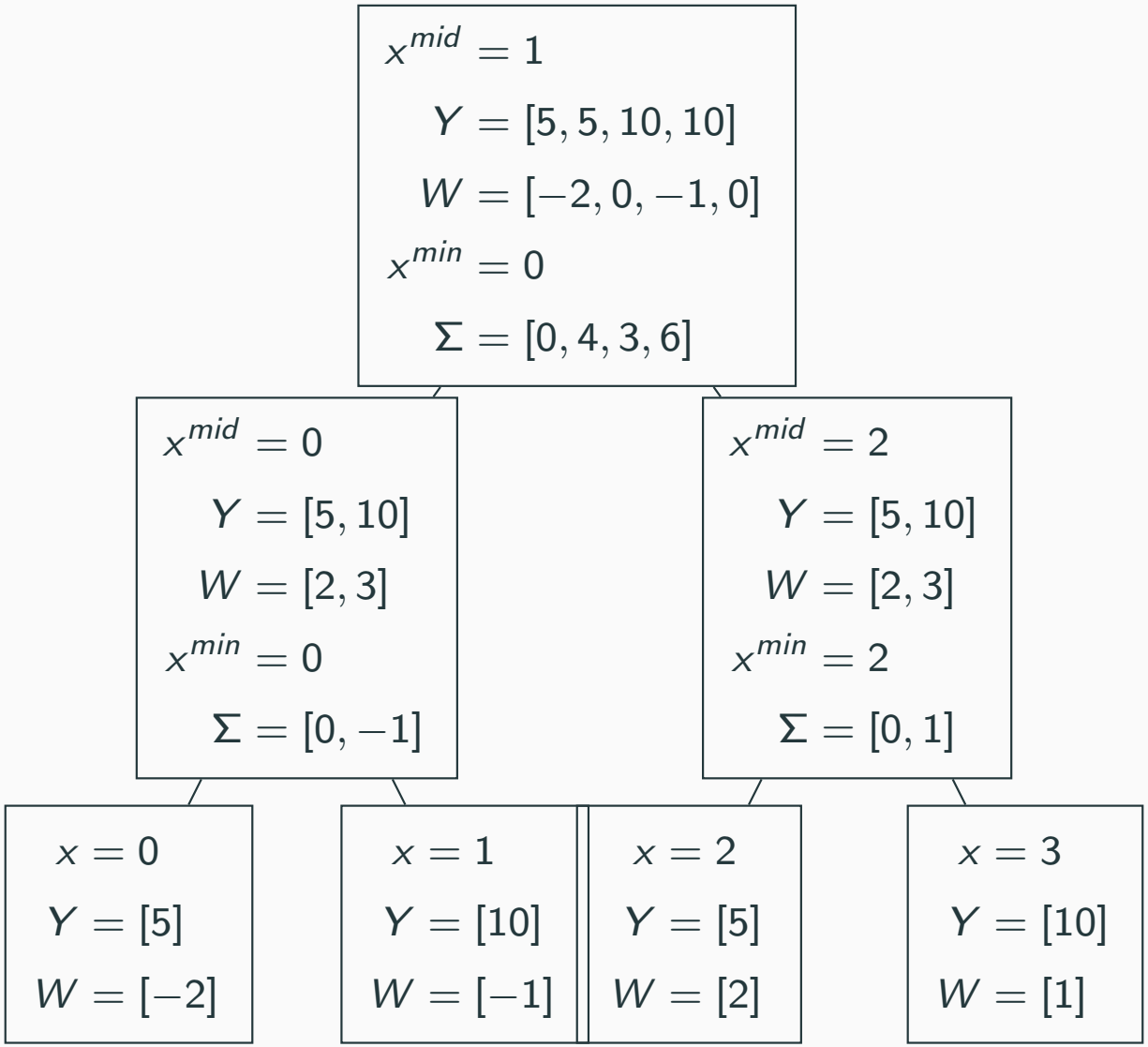
- Le parcours demeure inchangé

## Rappoter un noeud

- Additionner $\Sigma_i + W_i(x^{min} - \chi)$ au total

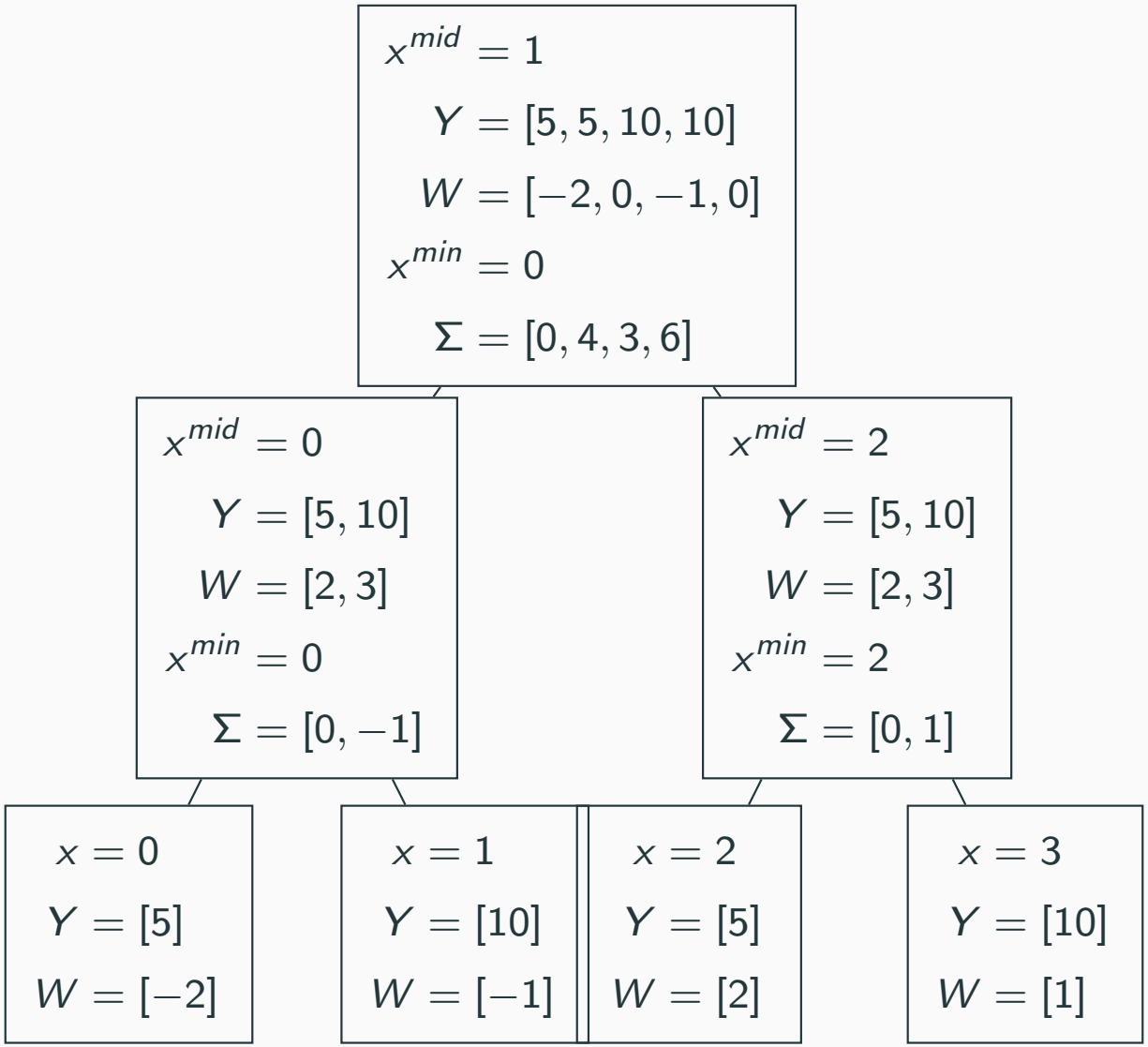
# Range-Tree à deux dimensions

Exemple avec $Q(1, 6)$



# Range-Tree à deux dimensions

Exemple avec $Q(1,6) = 0 + 2(2-1) = 2$



# Range-Tree

## Résumé des Range-Tree

- Construction en $O(n \log n)$
- Calcule en $O(\log n)$ la somme pondérée de portions de segments dans un rectangle semi-ouvert
- Plusieurs variantes possibles pour différents problèmes
- Se généralise à plusieurs dimensions

# Algorithme de vérification

## Objectif

- Trouver le minimum en vérifiant $O(n \log n)$ entrées
- Calcul en $O(\log n)$ d'une entrée
- Algorithme en $O(n \log^2 n)$

## Exemple

| | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 7 | 5 | 3 | 1 | 0 | 1 | 2 | 2 | 4 | 2 | | |
| 4 | 7 | 5 | 3 | 1 | 0 | -1 | 0 | 0 | 2 | | | |
| 5 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | 0 | | | | |
| 6 | 9 | 7 | 5 | 3 | 2 | 1 | 0 | | | | | |
| 7 | 10 | 8 | 6 | 4 | 2 | 1 | | | | | | |

# Algorithme de filtrage

# Algorithme de filtrage

## Algorithme

1. Filtrer une tâche à son plus tôt
2. Exécuter l'algorithme de vérification
3. S'il y a échec, ajuster le $\mathrm{est}$ de la tâche

# Algorithme de filtrage

## Algorithme

1. Filtrer une tâche à son plus tôt
2. Exécuter l'algorithme de vérification
3. S'il y a échec, ajuster le $\mathrm{est}$ de la tâche

## Analyse

- $O(n \log^2 n)$ pour une tâche

# Algorithme de filtrage

## Algorithme

1. Filtrer une tâche à son plus tôt
2. Exécuter l'algorithme de vérification
3. S'il y a échec, ajuster le $\mathrm{est}$ de la tâche

## Analyse

- $O(n \log^2 n)$ pour une tâche
- $O(n^2 \log^2 n)$ pour n tâches en pire cas

# Algorithme de filtrage

## Algorithme

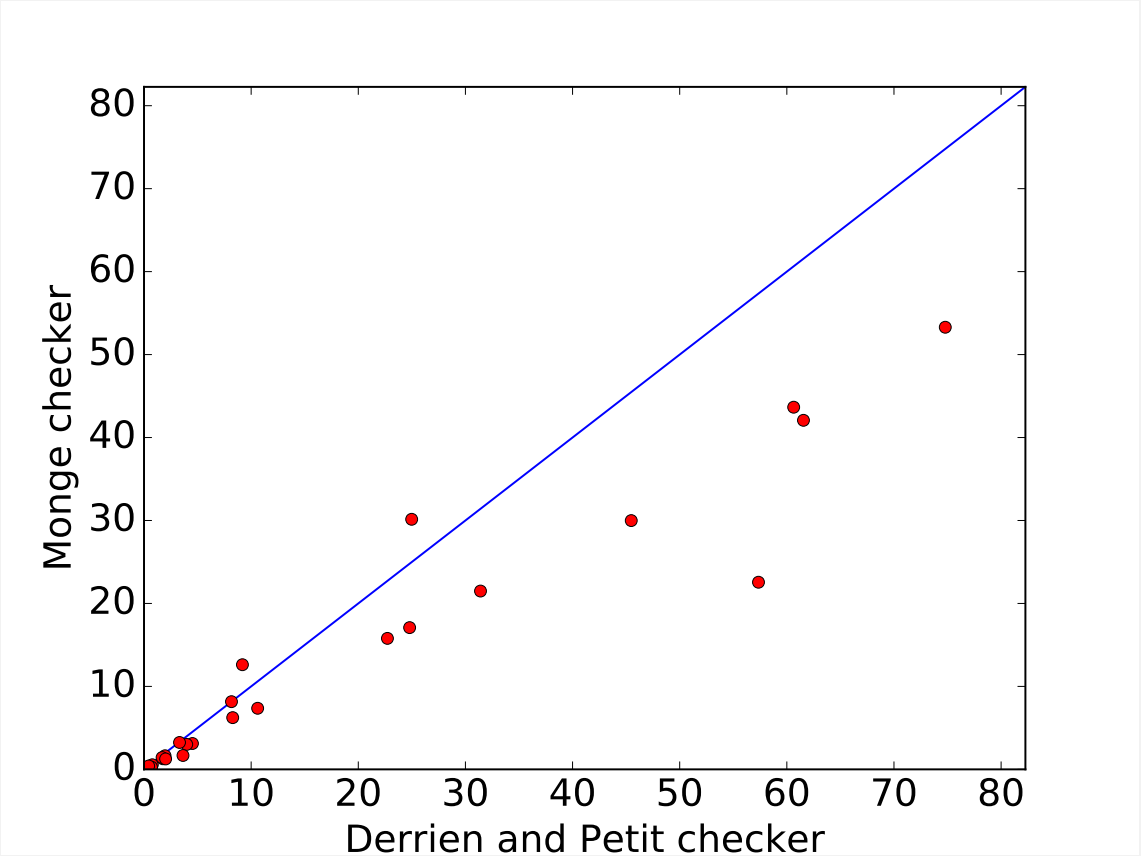
1. Filtrer une tâche à son plus tôt
2. Exécuter l'algorithme de vérification
3. S'il y a échec, ajuster le $\mathrm{est}$ de la tâche

## Analyse

- $O(n \log^2 n)$ pour une tâche
- $O(n^2 \log^2 n)$ pour n tâches en pire cas
- $O(n^2 \log n)$ pour n tâches en cas moyen

# Résultats

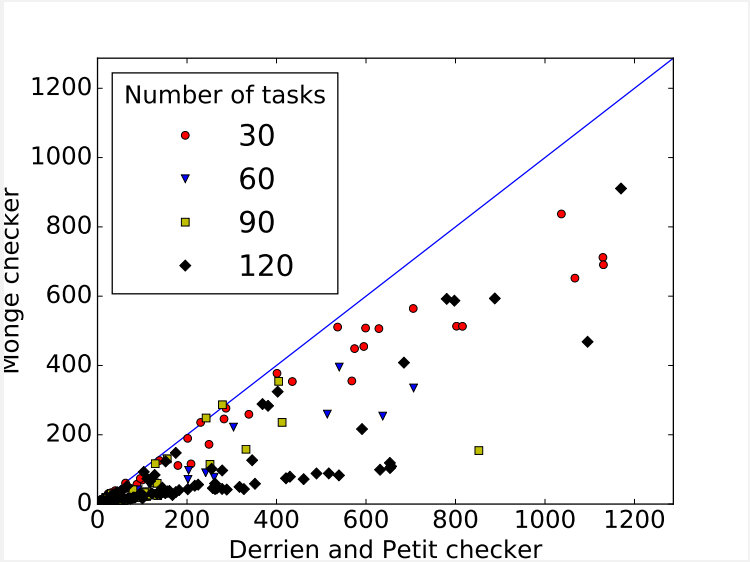
## Algorithme de vérification



# Résultats

## Algorithme de filtrage

| Benchmark | $n$ | % time |
|---|---|---|
| BL | 20 | 0.92 |
| BL | 25 | 0.89 |
| PSPLIB | 30 | 0.94 |
| PSPLIB | 60 | 0.57 |
| PSPLIB | 90 | 0.44 |
| PSPLIB | 120 | 0.48 |



# Conclusion

## Contributions

- Raisonnement énergétique avec seulement $O(n \log n)$ intervalles
- Amélioration du temps d'exécution d'algorithmes de vérification et de filtrage
- Adaptation de structure de données

## Résumé de l'amélioration de la complexité

| Algorithme | État de l'art | Algorithme de Monge |
|---|---|---|
| Vérification | $O(n^2)$ | $O(n \log^2 n)$ |
| Filtrage | $O(n^2 \log^2 n)$ | $O(n^2 \log n)$ |

# Fin

Merci !