Théorie des jeux

Modèles basés sur les états pour les jeux

Mohammed Ismail SMAHI

Mars 2020

Sommaire

Modalités

Déroulement

- Cours (1h30 chaque semaine)
- Travaux pratiques (1h30 chaque semaine)

Evaluation

- Un test de travaux pratiques ($\approx$ 33% de la note finale)
- Un examen final ($\approx$ 66% de la note finale)
- Eventuellement des points bonus pour les "bons" joueurs

Algorithme *Minimax*

Caractéristiques des jeux

- Basé sur le théorème du Minimax de von Neumann.
- La chance n'intervient pas.
- Règle de décision pour les jeux à somme nulle : les gains d'un joueur représentent exactement les pertes de l'autre joueur.
- Formulé pour des jeux à somme nulle à deux joueurs jouant alternativement.

Algorithme Minimax

Procedure miniMax(N, maxPlayer)

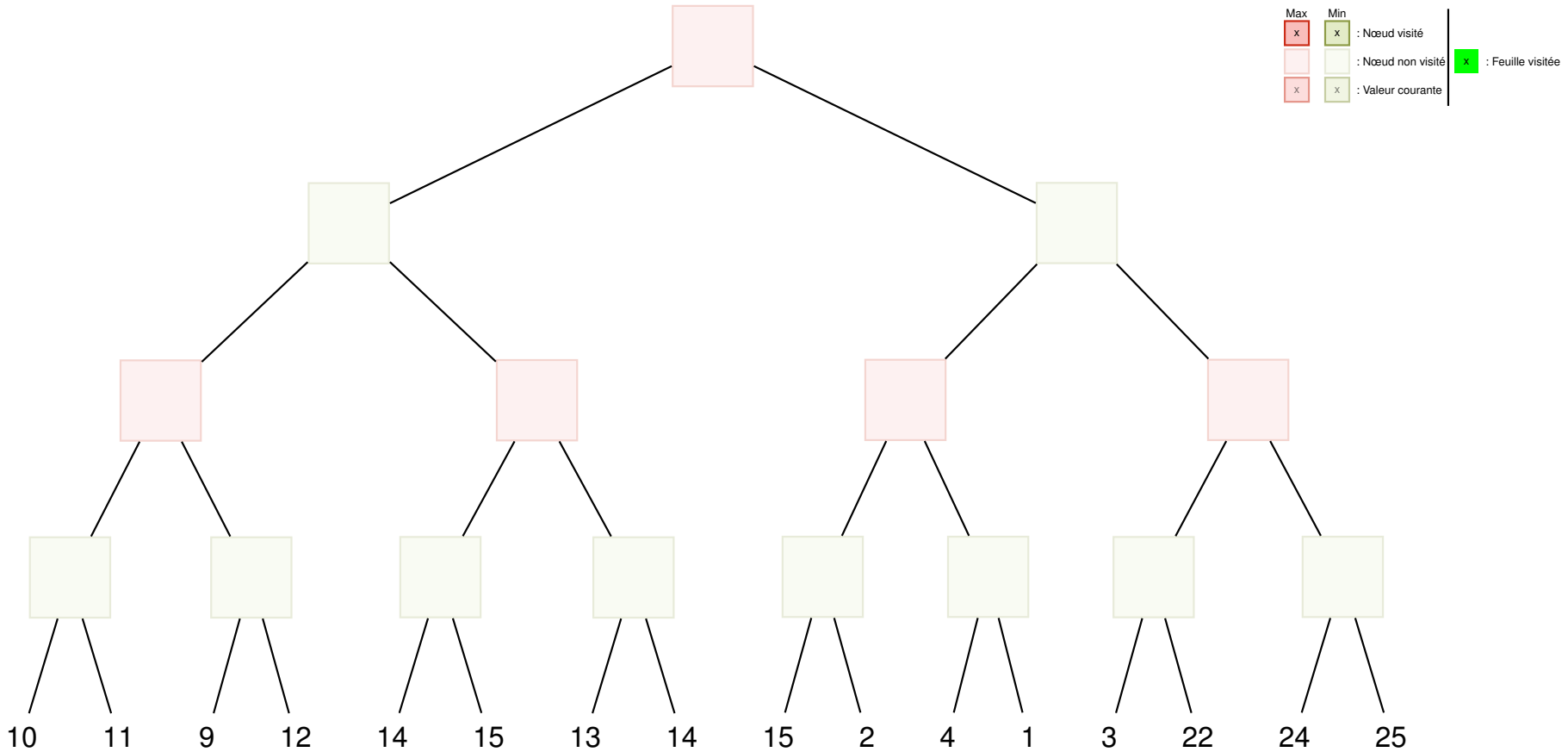
if N is Leaf **then**
    return score(N)
end if

if maxPlayer **then**
    $v \leftarrow -\infty$
    for each $n \in N.successeurs$ **do**
        $v \leftarrow \max(v, miniMax(n, false))$
    end for
else
    $v \leftarrow +\infty$
    for each $n \in N.successeurs$ **do**
        $v \leftarrow \min(v, miniMax(n, true))$
    end for
end if

return v

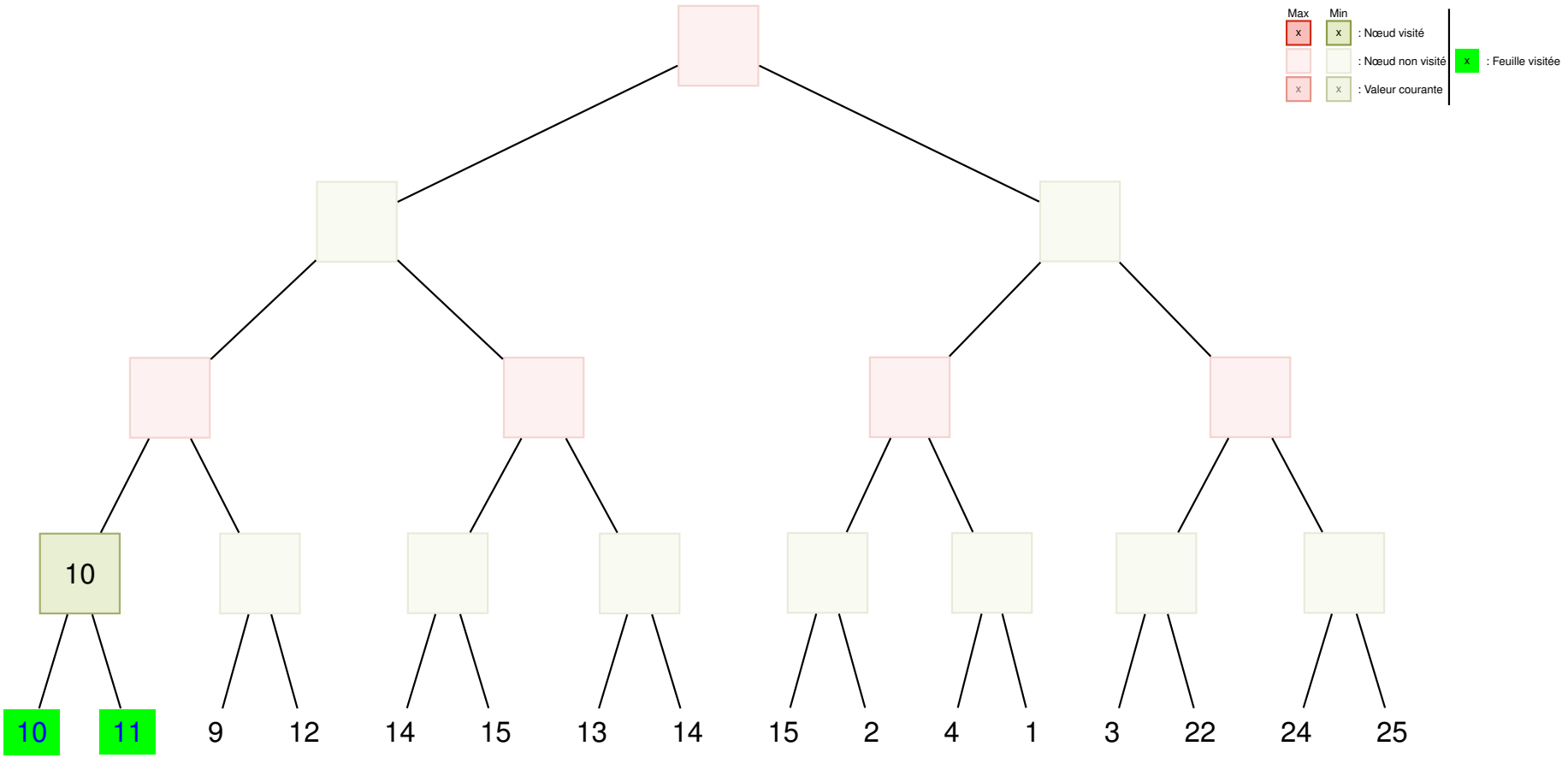
Algorithme *Minimax*

Exemple



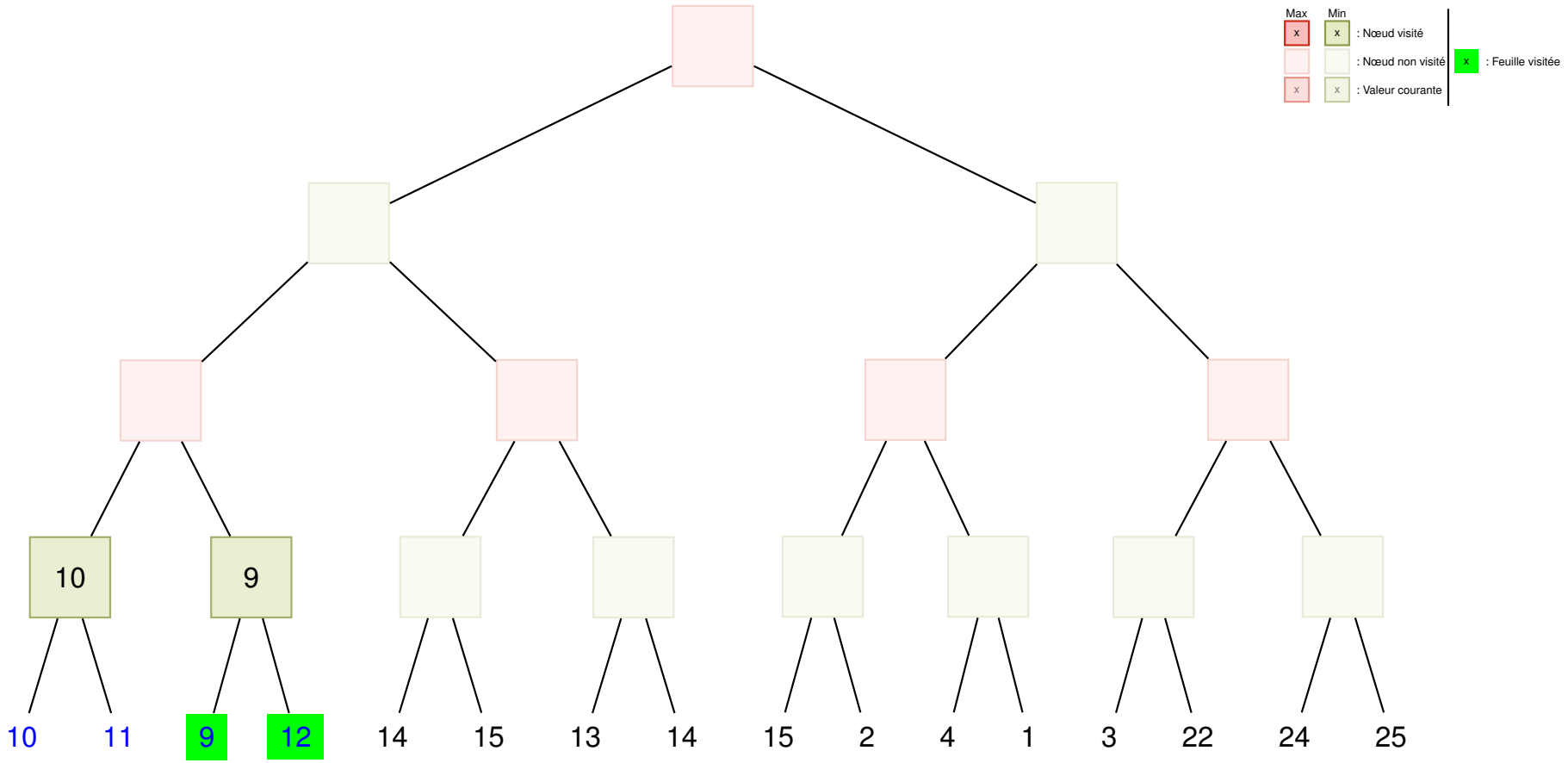
Algorithme *Minimax*

Exemple



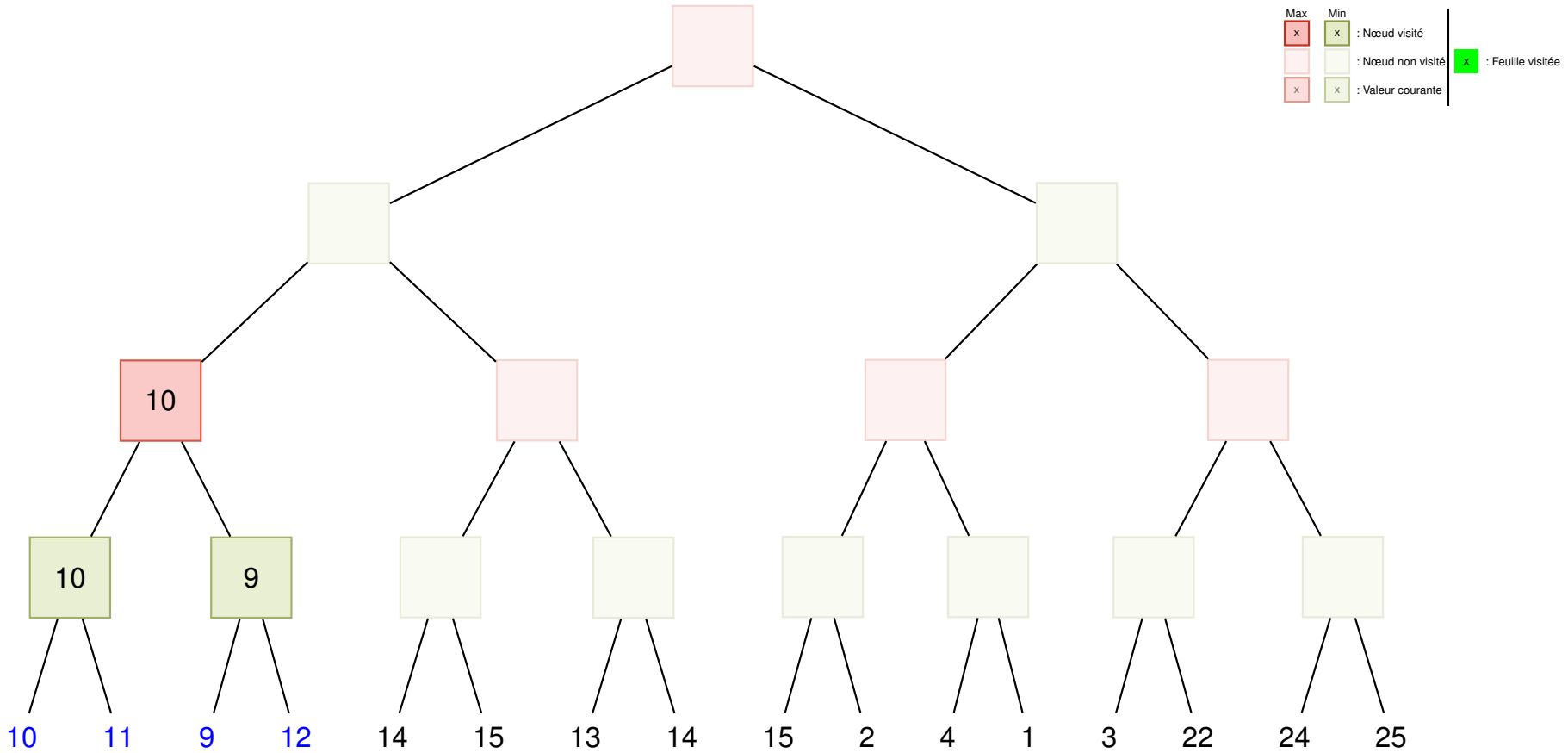
Algorithme *Minimax*

Exemple



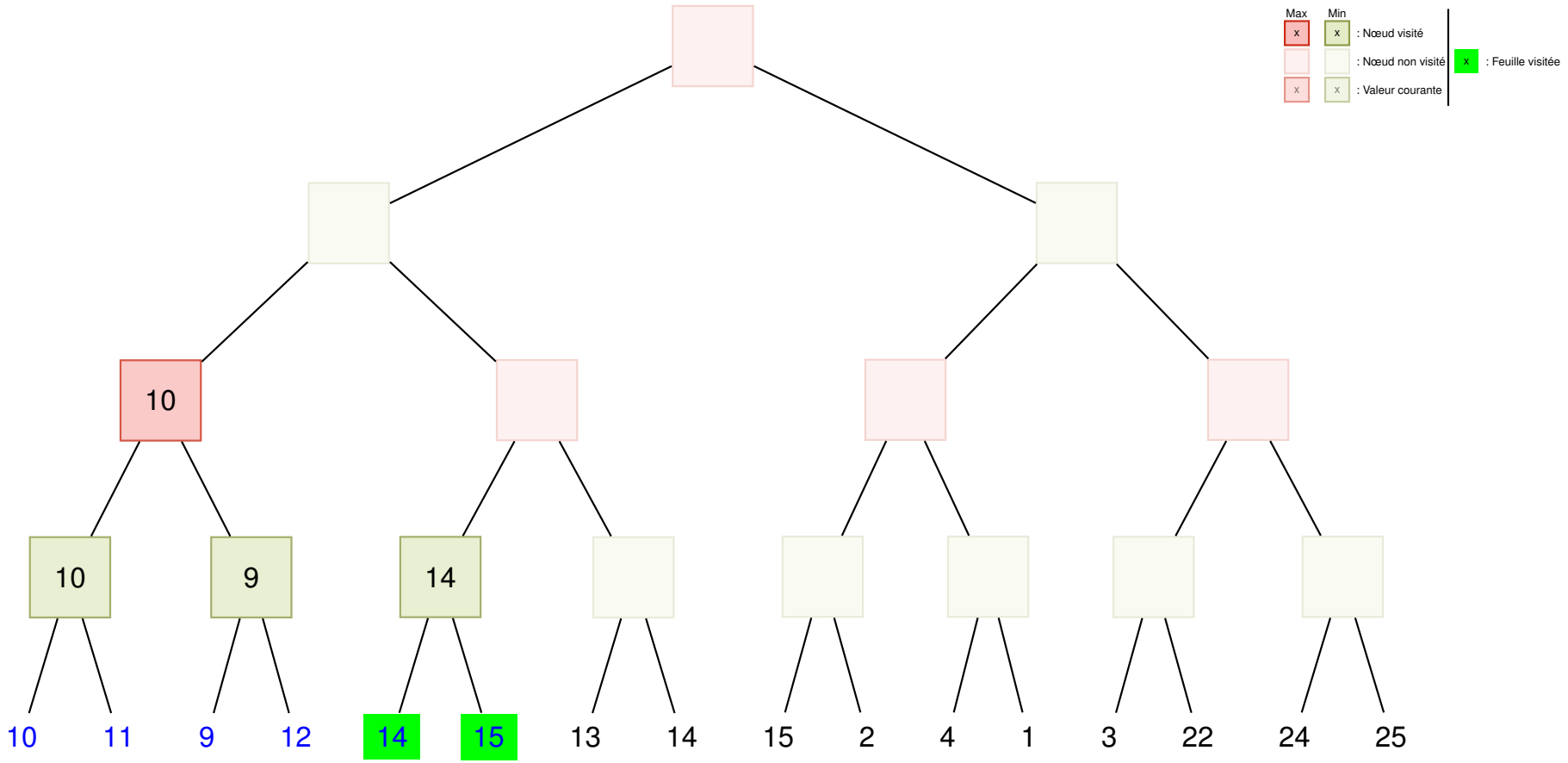
Algorithme *Minimax*

Exemple



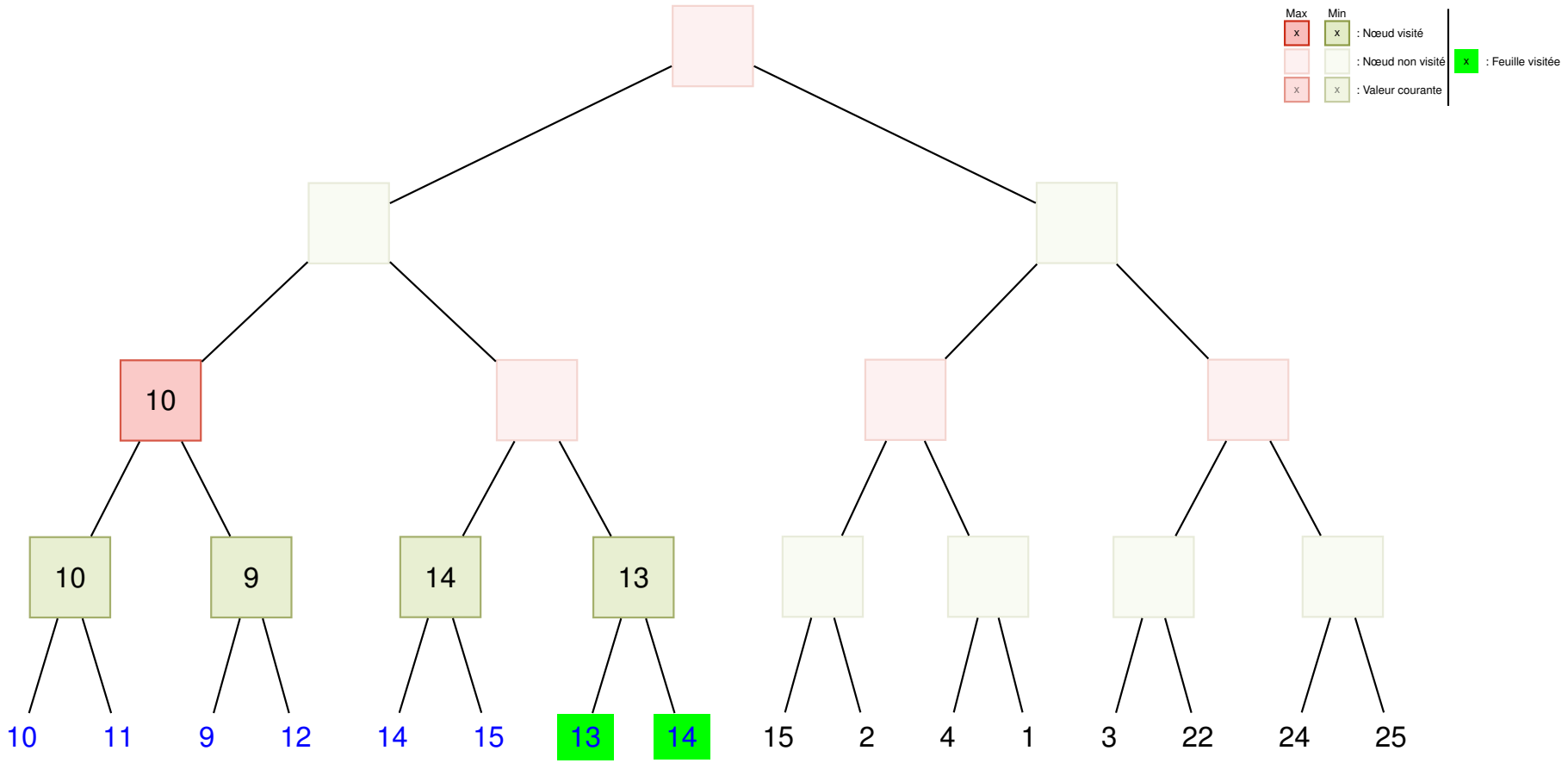
Algorithme *Minimax*

Exemple



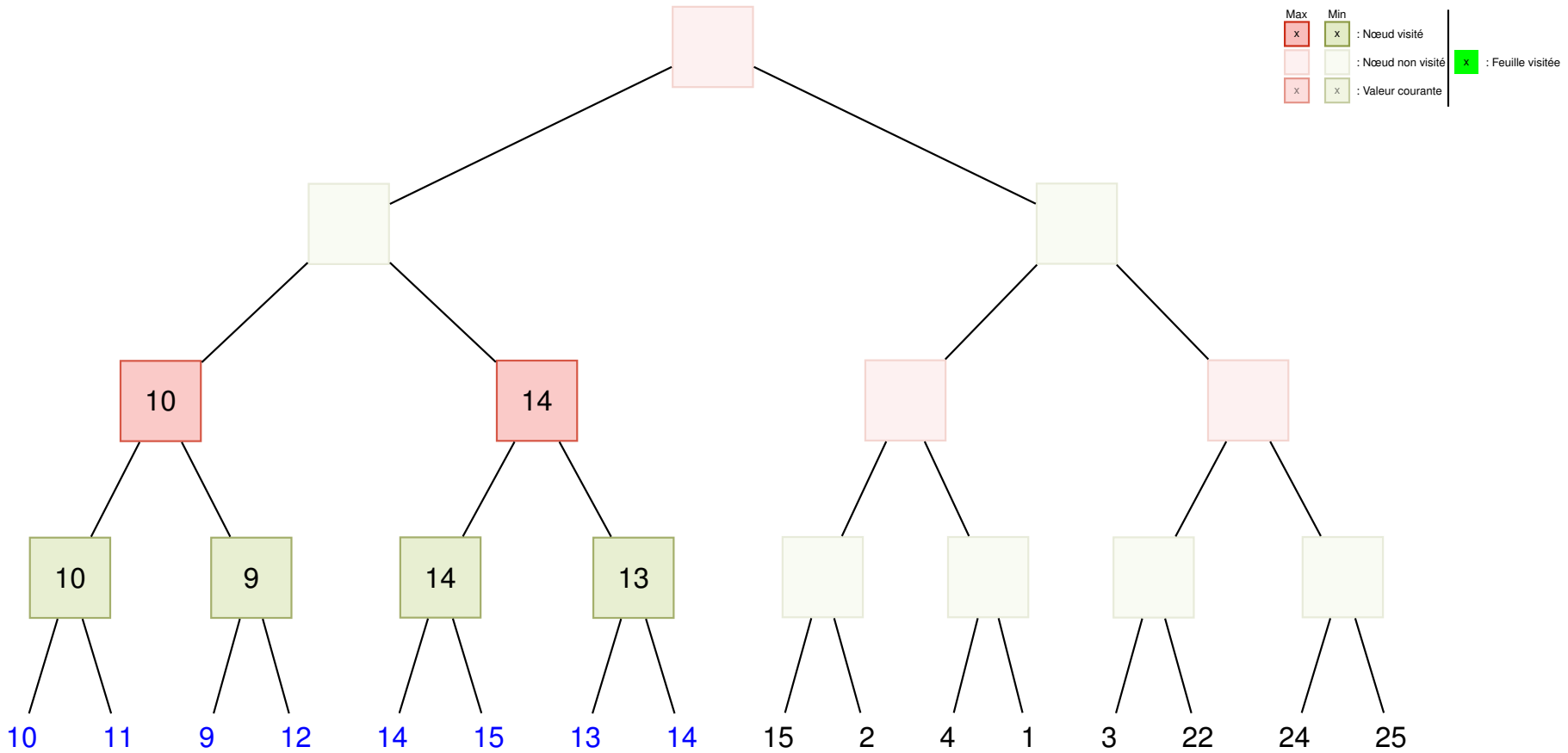
Algorithme *Minimax*

Exemple



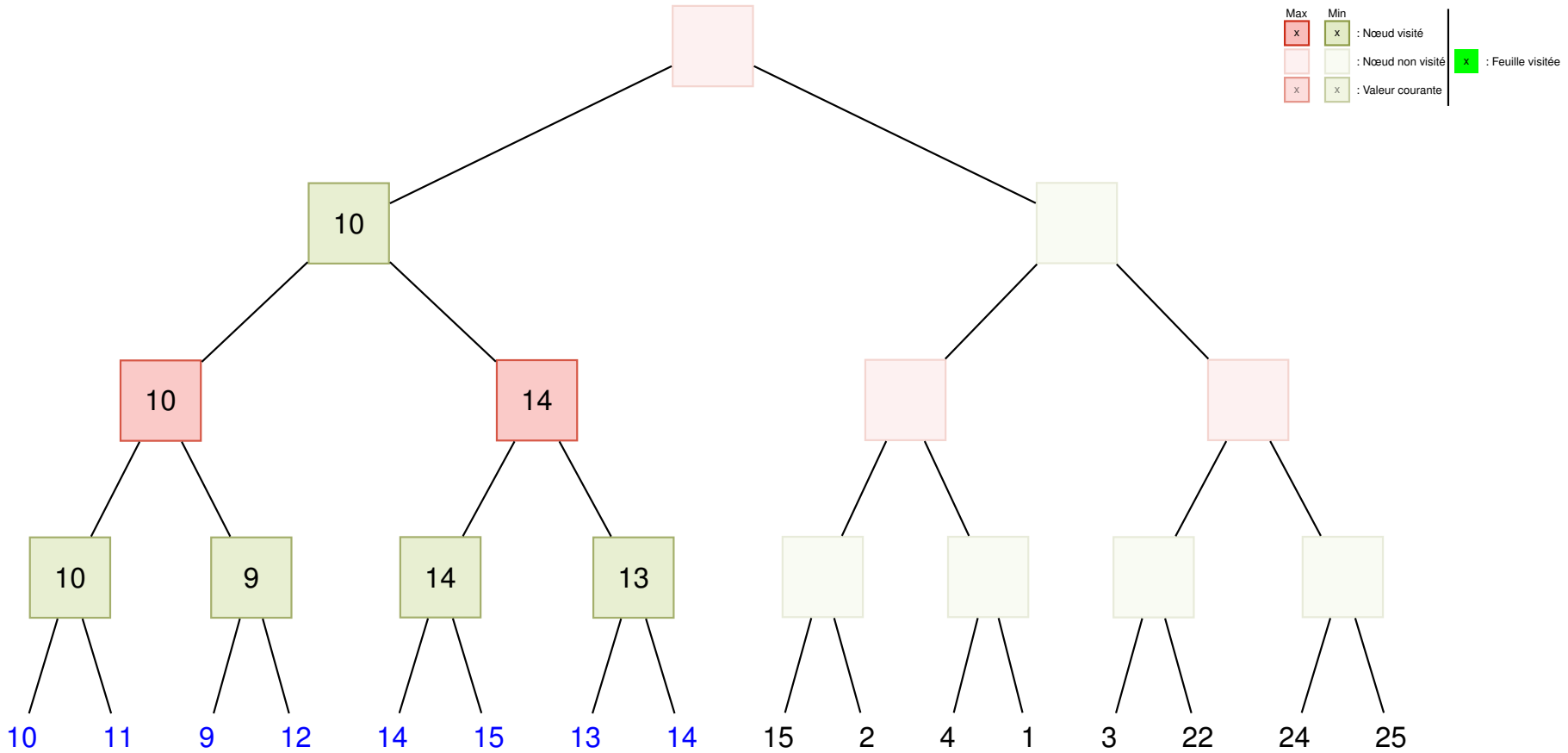
Algorithme *Minimax*

Exemple



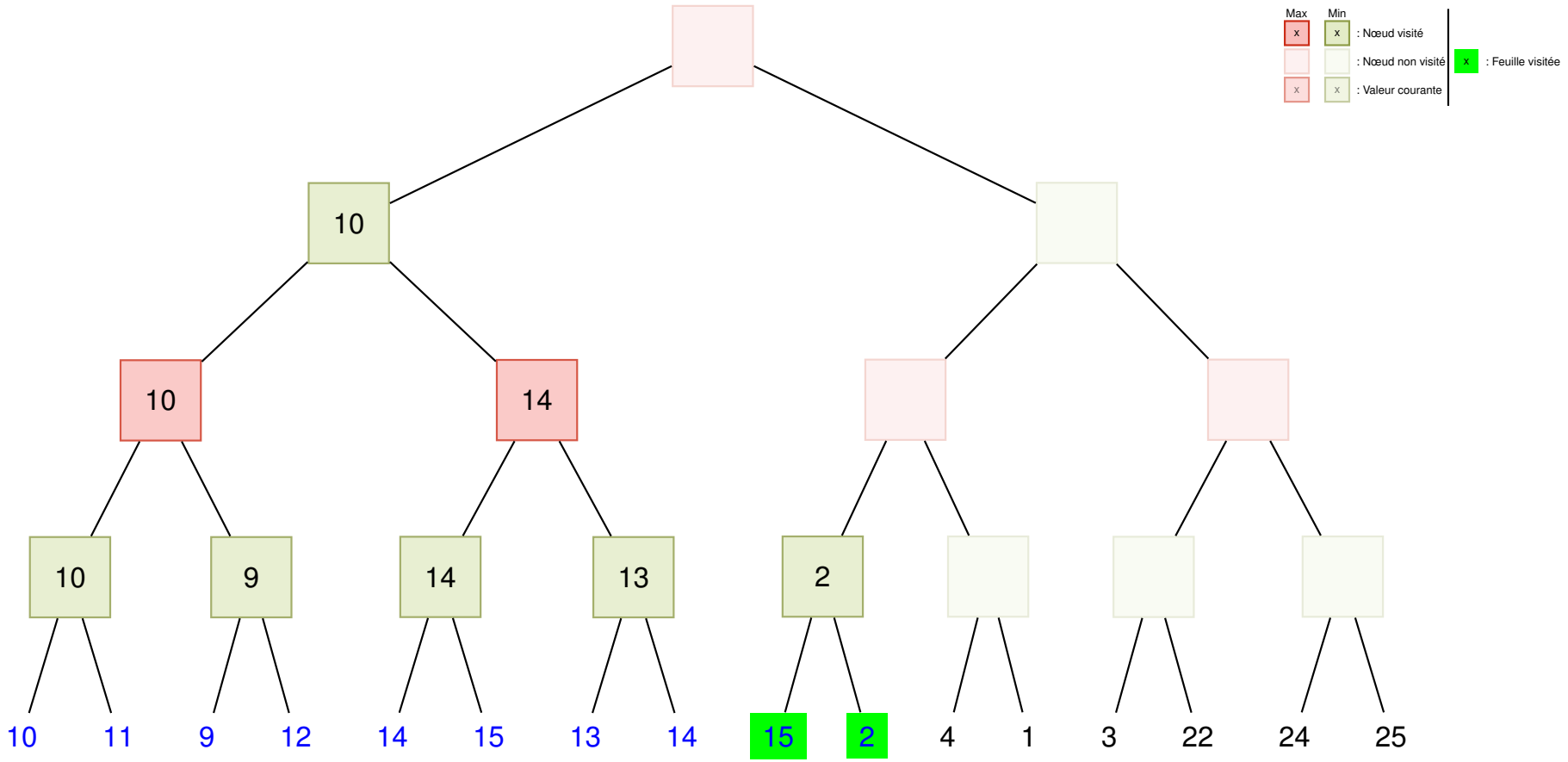
Algorithme *Minimax*

Exemple



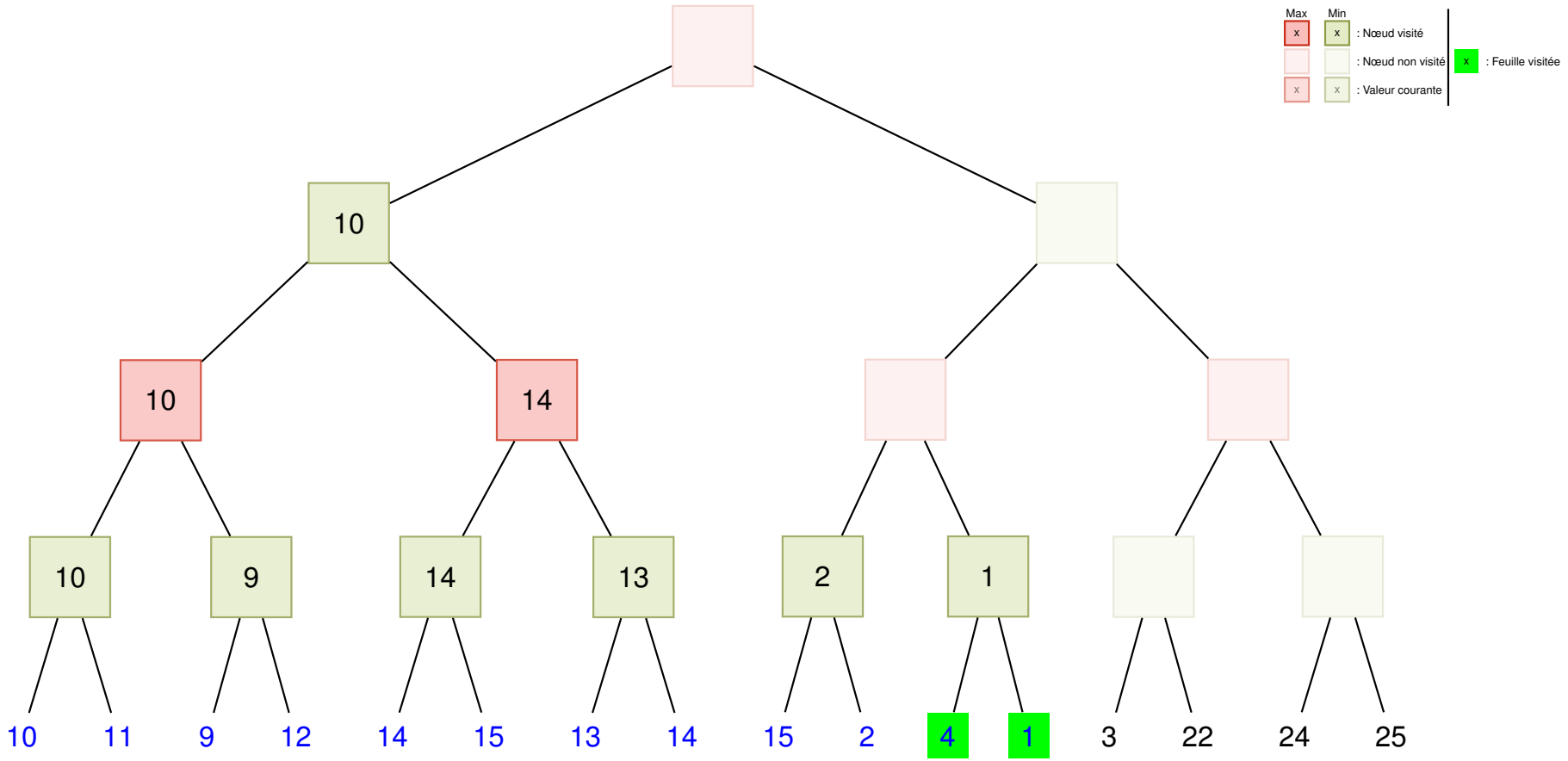
Algorithme *Minimax*

Exemple



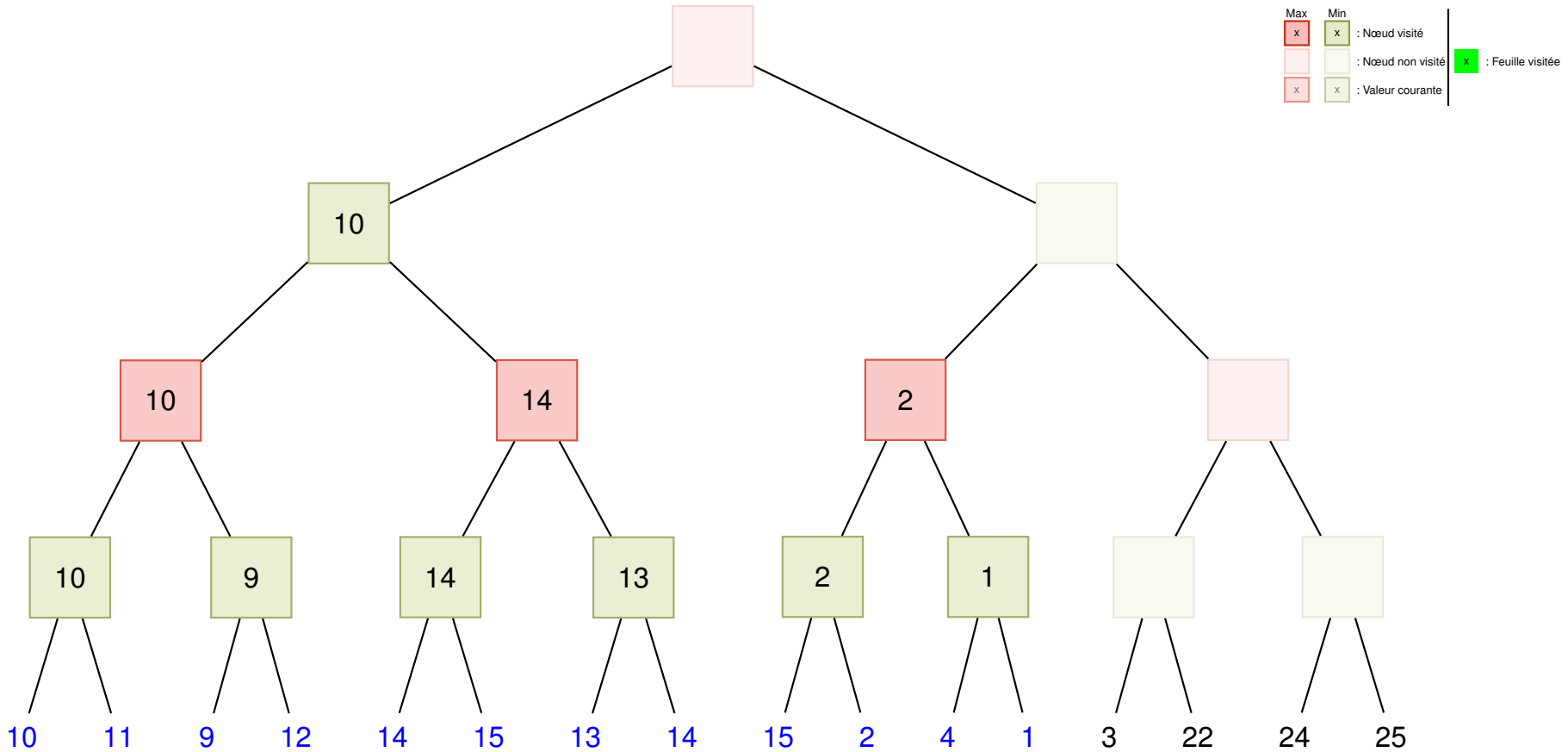
Algorithme *Minimax*

Exemple



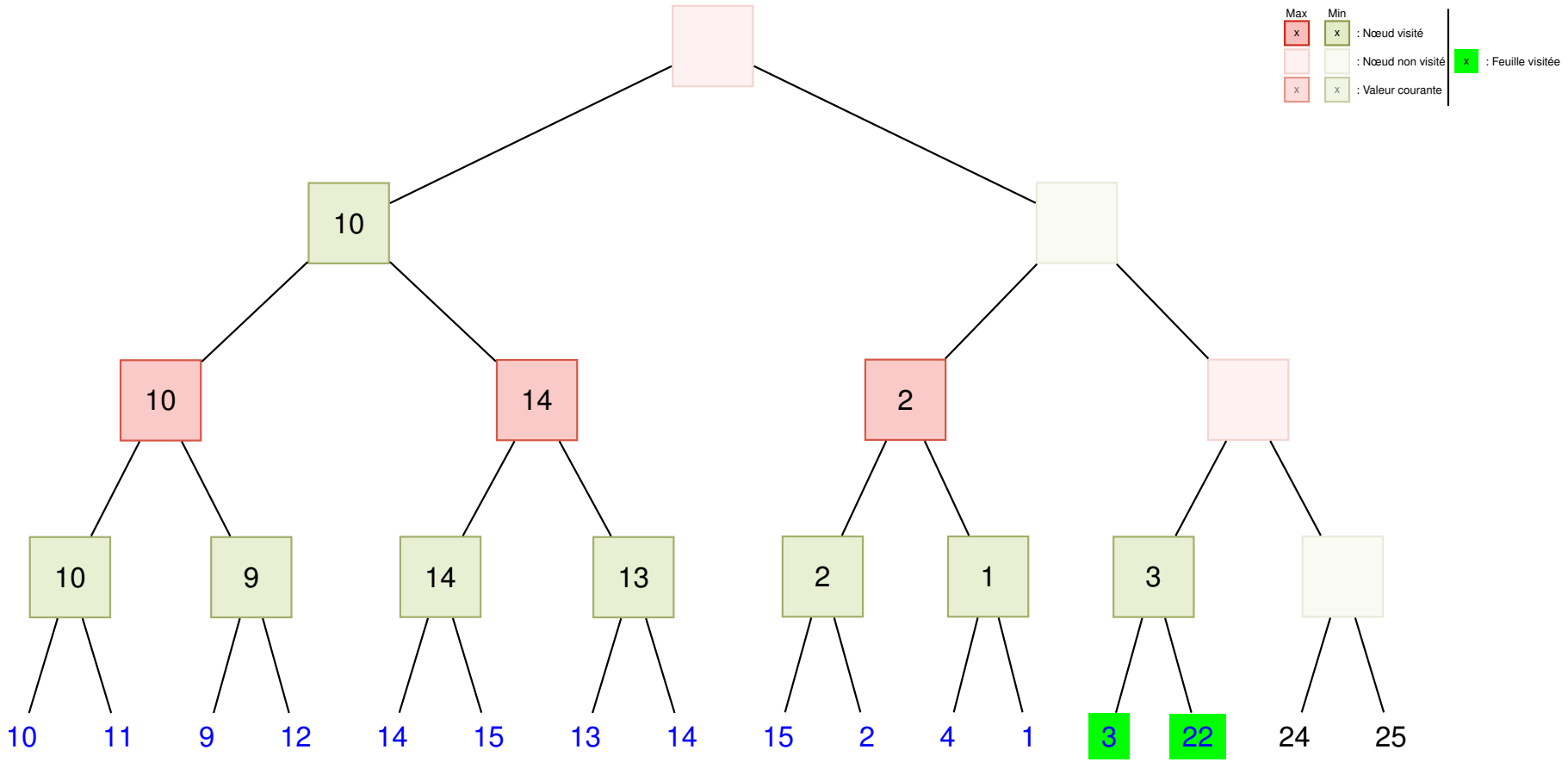
Algorithme *Minimax*

Exemple



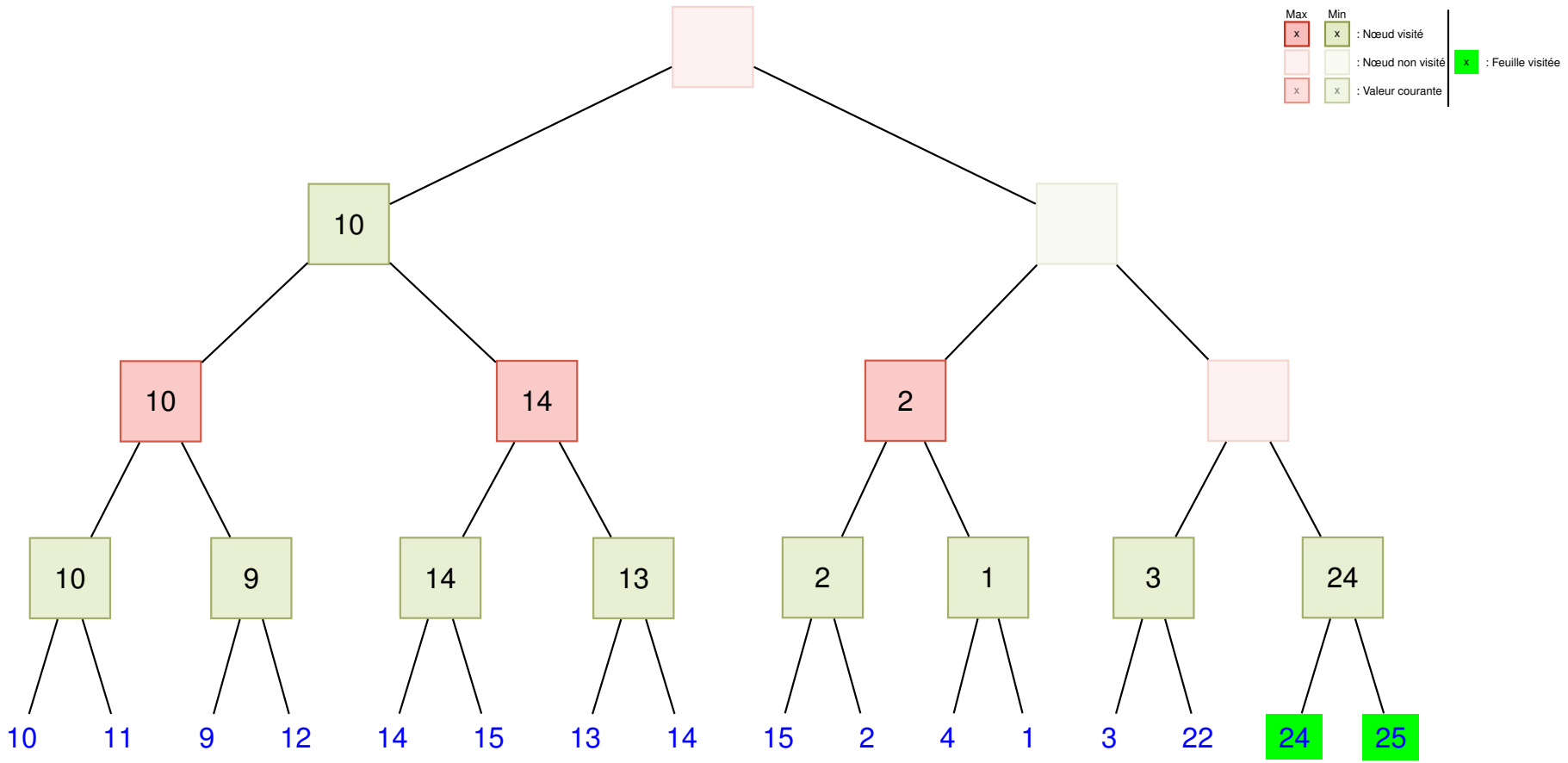
Algorithme *Minimax*

Exemple



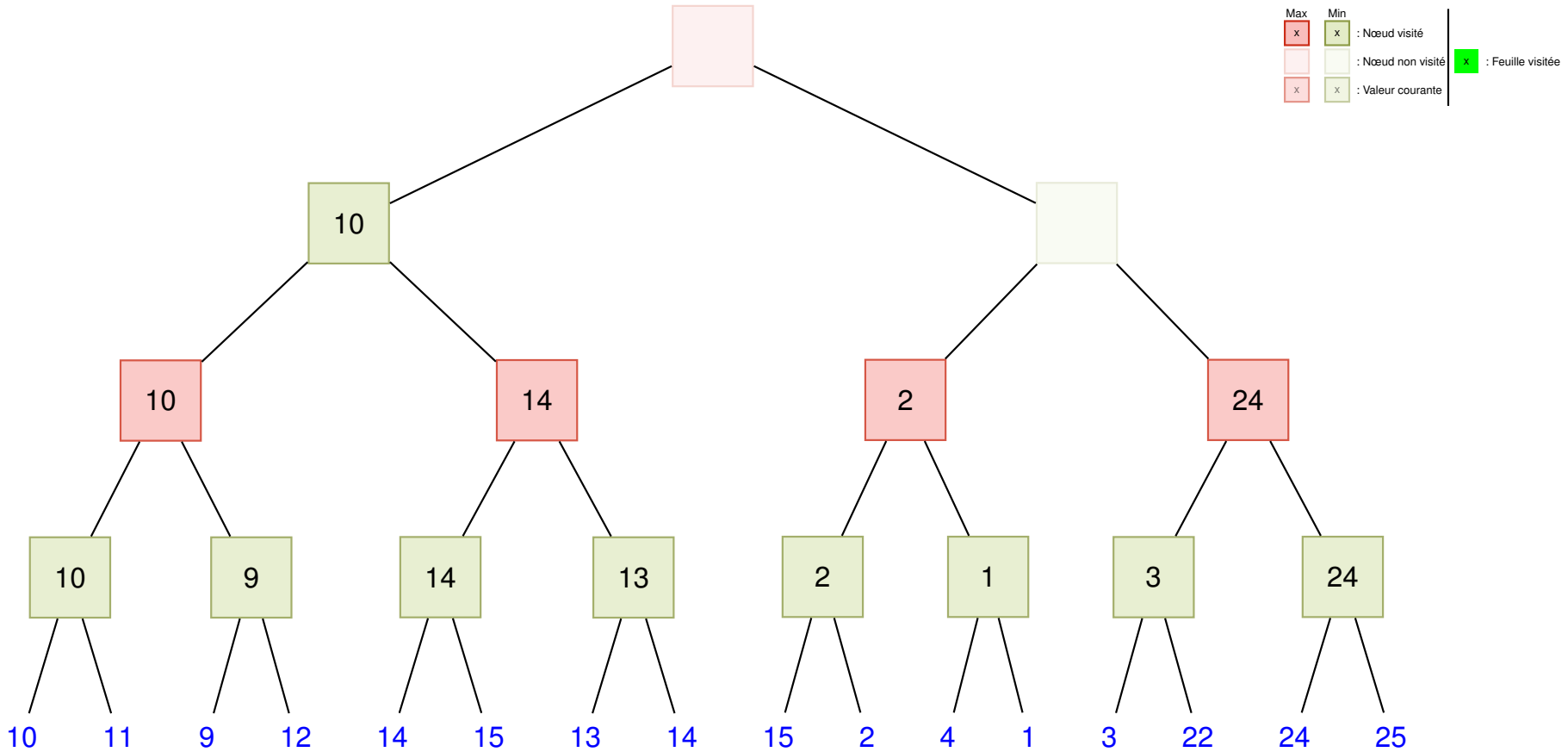
Algorithme *Minimax*

Exemple



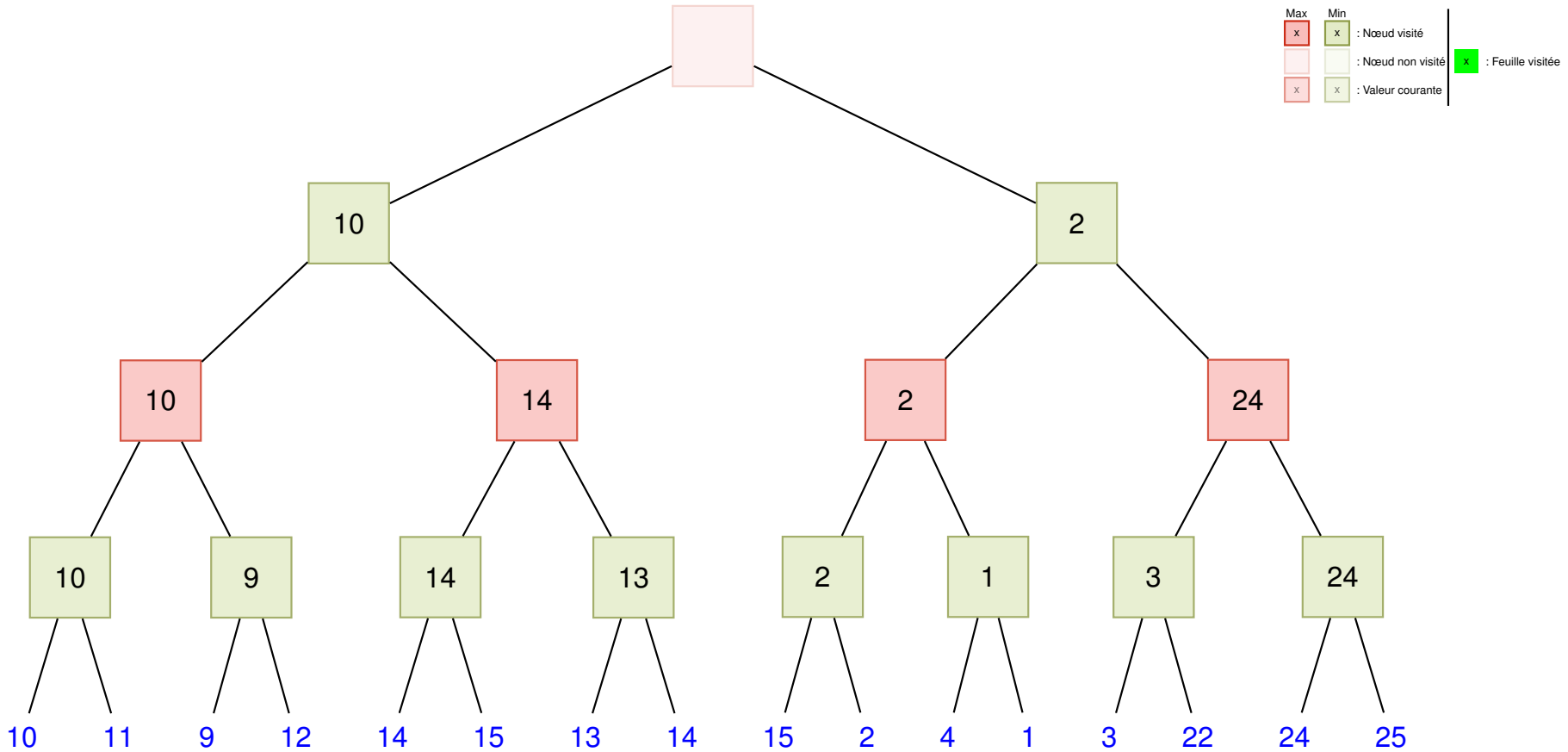
Algorithme *Minimax*

Exemple



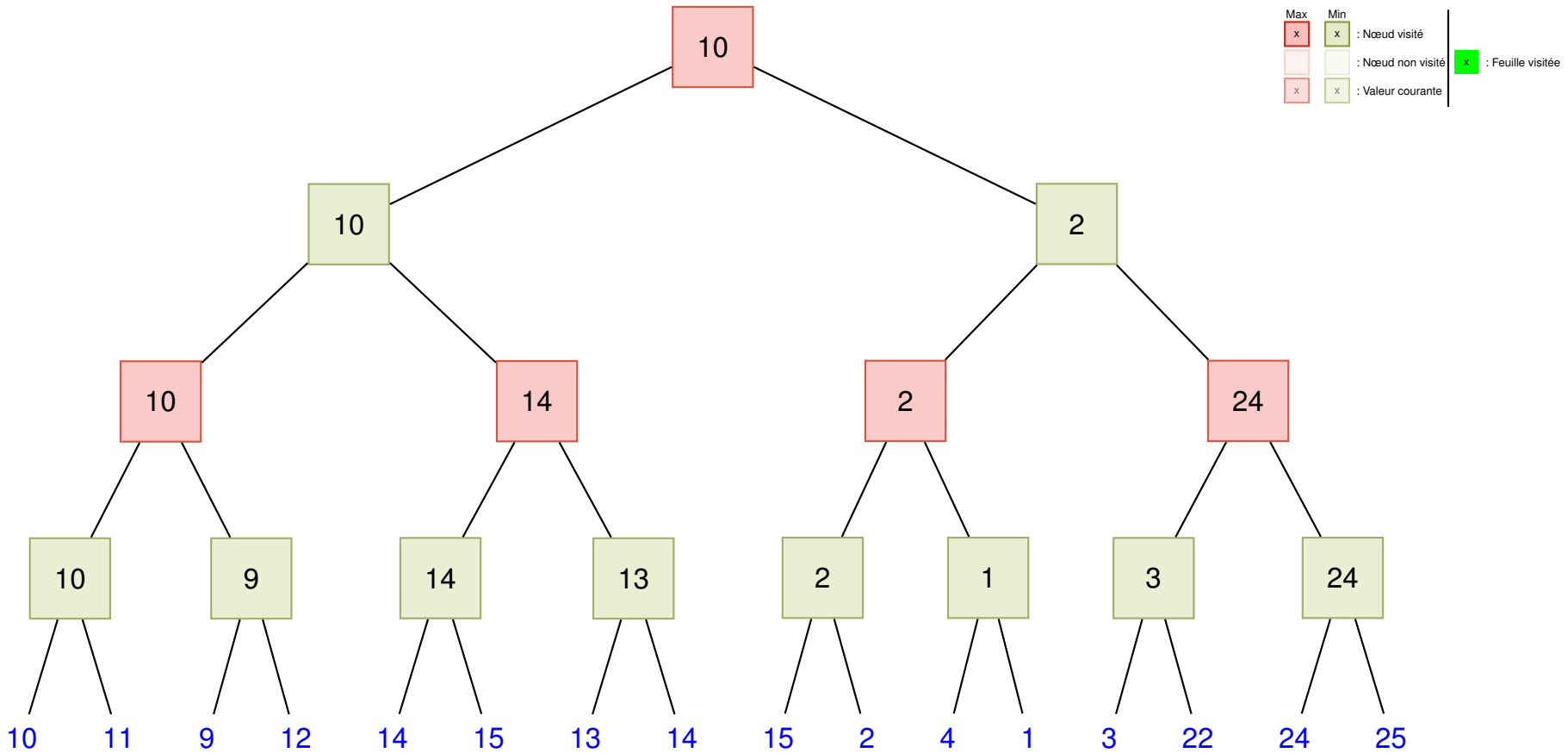
Algorithme *Minimax*

Exemple



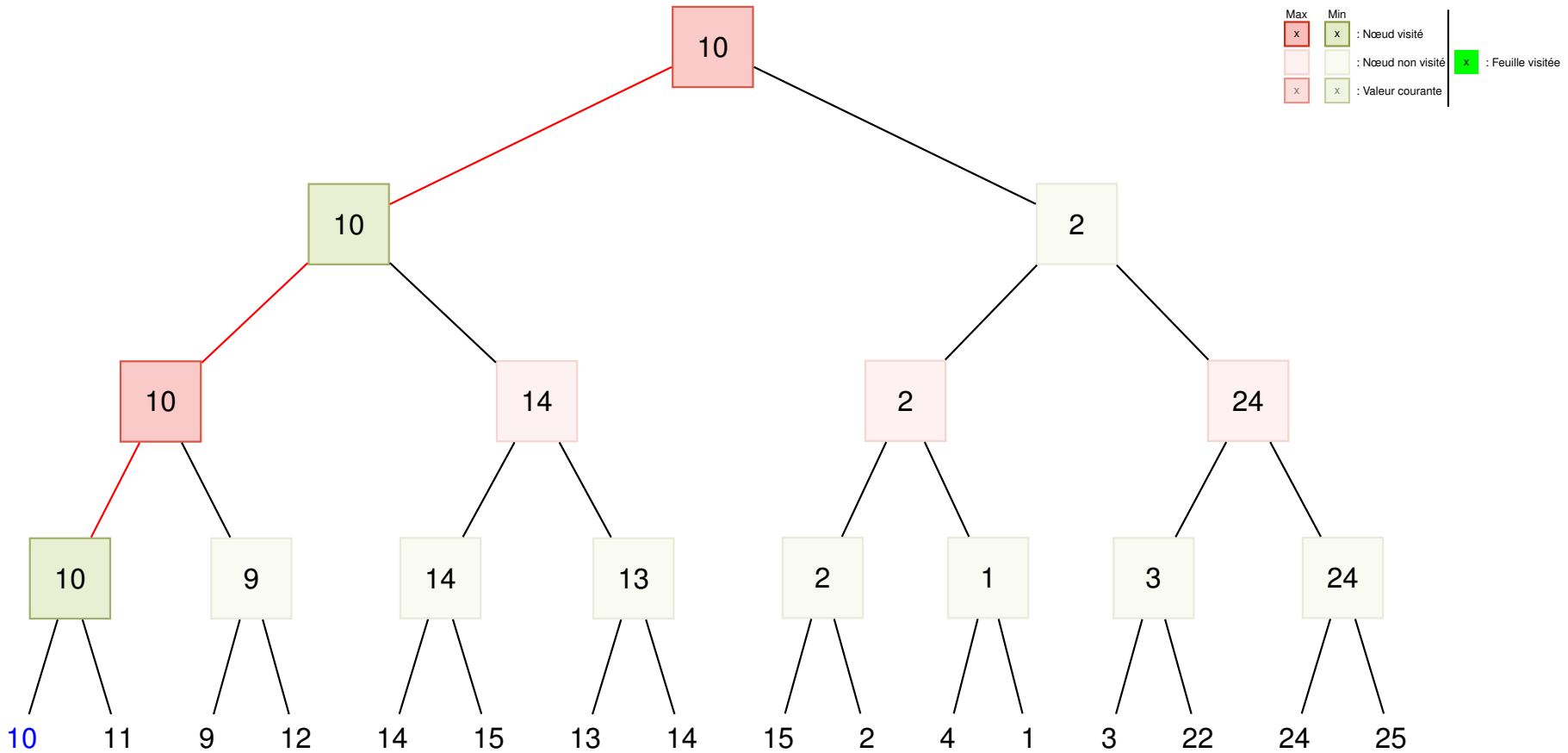
Algorithme *Minimax*

Exemple



Algorithme *Minimax*

Exemple



Algorithme Minimax

Inconvénients de la méthode

Problèmes Minimax

- Exploration complète de l'arbre, de l'espace d'états
- Possibilité d'élaguer des branches en fonction des découvertes !

Introduction

Principe Alpha-Beta

L'élagage alpha-beta est une méthode exacte d'optimisation employée sur l'algorithme de minimax et a pour but d'éviter l'exploration de parties inutiles de l'arbre de jeu :

- Alpha - Beta est une stratégie visant à réduire le nombre de noeuds explorés par la méthode "Minimax".
- Pour chaque noeud, elle calcule la valeur de la position ainsi que deux valeurs, alpha et beta.

Principe $\alpha - \beta$

Coupure Alpha-Beta

- Coupure Alpha : Si la valeur d'un successeur *n* d'un nœud Min est inférieure à la valeur courante d'un nœud Max ancêtre, alors les frères de *n* n'ont pas besoin d'être explorés
- Coupure Beta : Si la valeur d'un successeur *n* d'un nœud Max est supérieure à la valeur courante d'un nœud MIN ancêtre, alors les frères de *n* n'ont pas besoin d'être explorés

Élagage fils d'un nœud MIN

Élagage fils d'un nœud MAX

Seuils $\alpha - \beta$

Principe

Utilisation de deux variables seuils, α et β :

- Le seuil α : pour un nœud à maximiser, α vaut la plus grande valeur de ses successeurs et, pour un nœud à minimiser, α est celui du prédécesseur.
  Si la valeur de s devient inférieure ou égale à α,l'exploration de sa descendance peut être arrêtée $\longrightarrow$ **Coupure Alpha**
- Le seuil β : pour un nœud à maximiser, β est celui du prédécesseur et, pour un nœud à minimiser, β vaut la plus petite valeur de ses successeurs.
  Si la valeur de s devient supérieure ou égale à β, l'exploration de sa descendance peut être arrêtée $\longrightarrow$ **Coupure Beta**
- α et β sont initialisés réciproquement à $-\infty$ et $+\infty$

Autrement dit :

Chaque joueur garde en mémoire la meilleure valeur qu'il puisse espérer (appelée α chez le joueur maximisant et β chez le joueur minimisant).
À une étape donnée, la condition $\beta < \alpha$ signifie que le chemin optimal ne peut pas passer par la branche actuelle puisque le joueur qui précédait avait une meilleure option à sa disposition

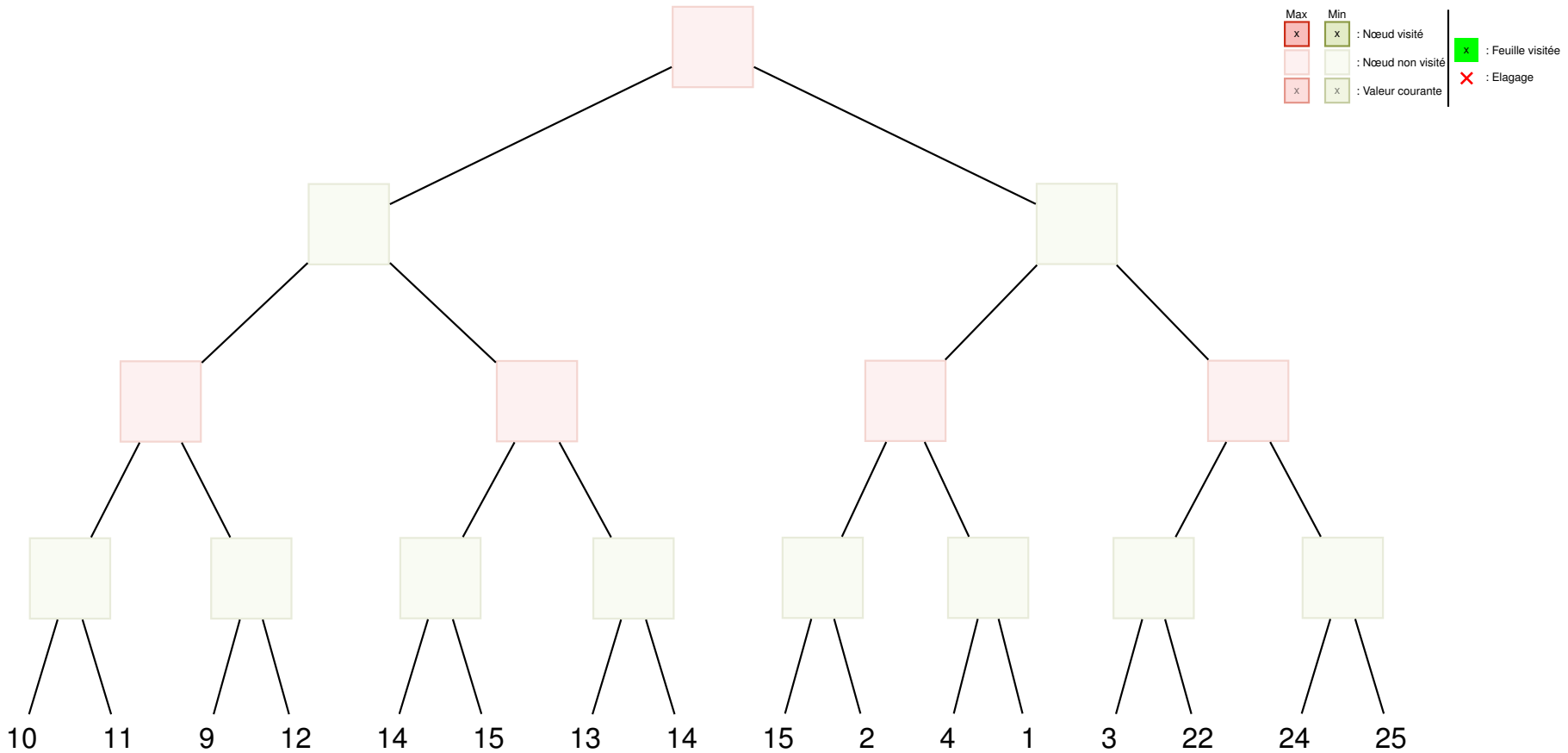
Algorithme Alpha-Beta

Procedure alpha-beta(N, α, β)

if N is Leaf **then**
  return score(N)
else
  if N is Min node **then**
    $v \leftarrow +\infty$
    for each $n_i \in N.successeurs$ **do**
      $v \leftarrow \min(v, \text{alpha-beta}(n_i, \alpha, \beta))$
      if $\alpha \geqslant v$ **then** ▷ coupure α
        return v
      end if
      $\beta \leftarrow \min(\beta, v)$
    end for
  else
    $v \leftarrow -\infty$
    for each $n_i \in N.successeurs$ **do**
      $v \leftarrow \max(v, \text{alpha-beta}(n_i, \alpha, \beta))$
      if $v \geqslant \beta$ **then** ▷ coupure β
        return v
      end if
      $\alpha \leftarrow \max(\alpha, v)$
    end for
  end if
  return v
end if

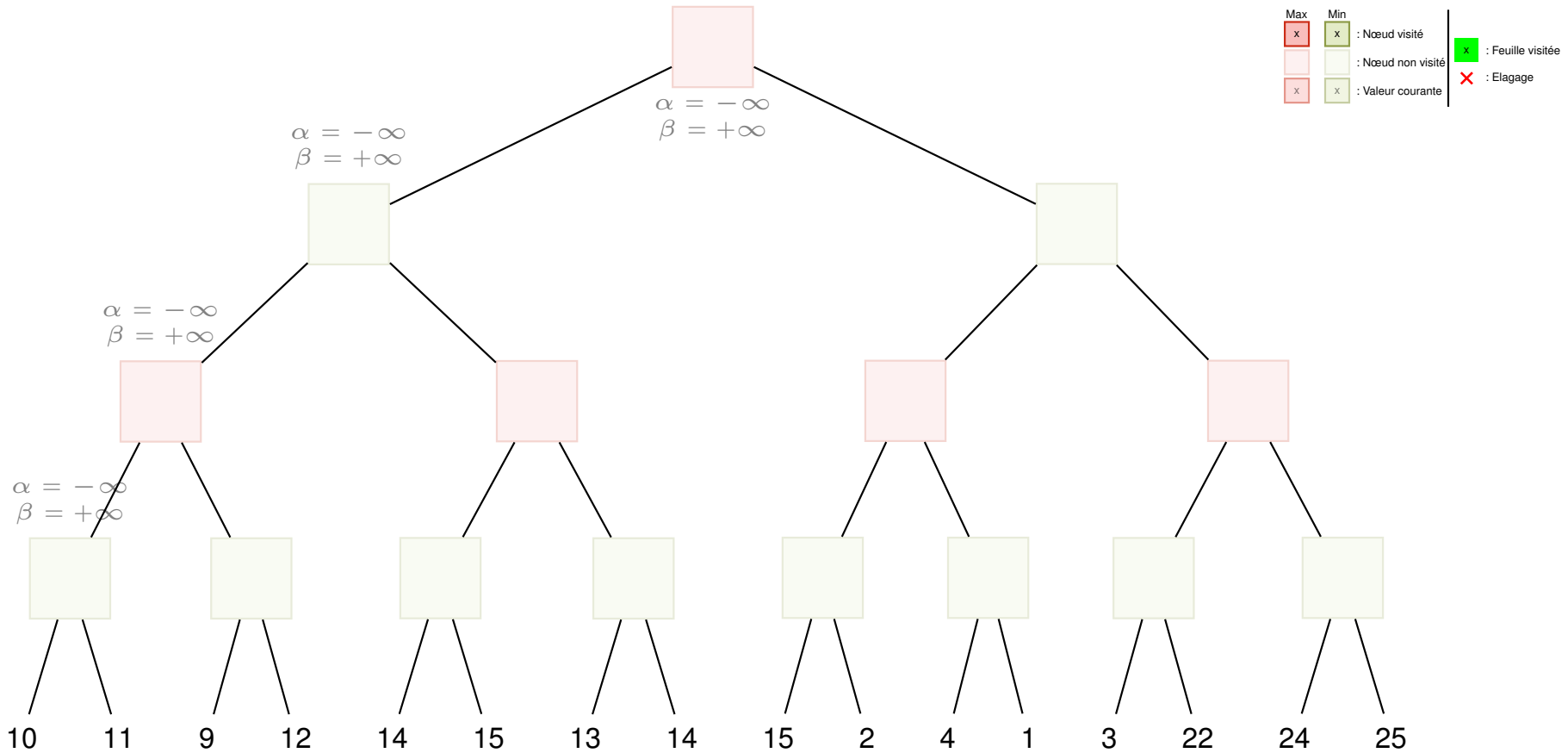
Exemples

Exemple 1



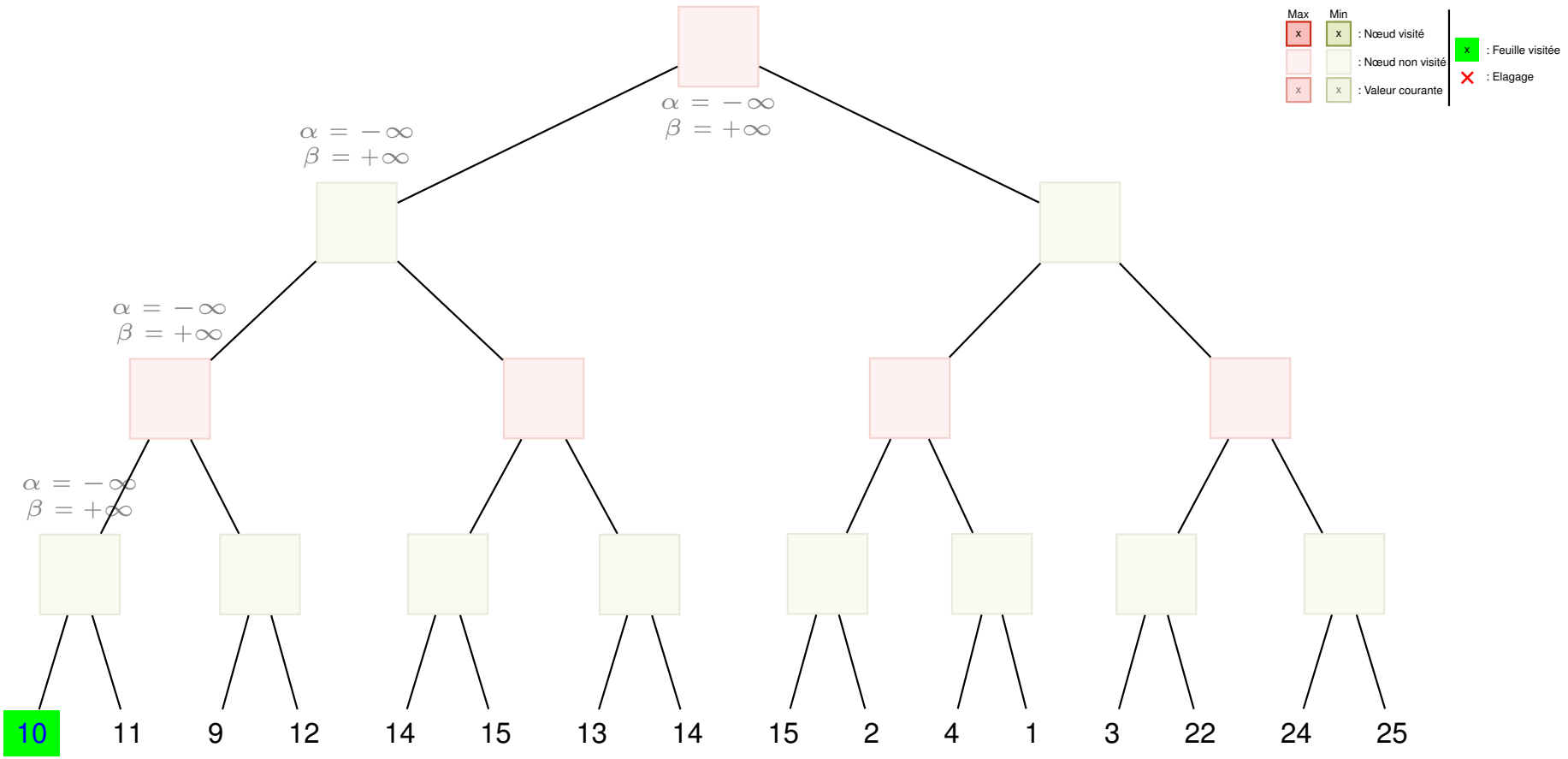
Exemples

Exemple 1



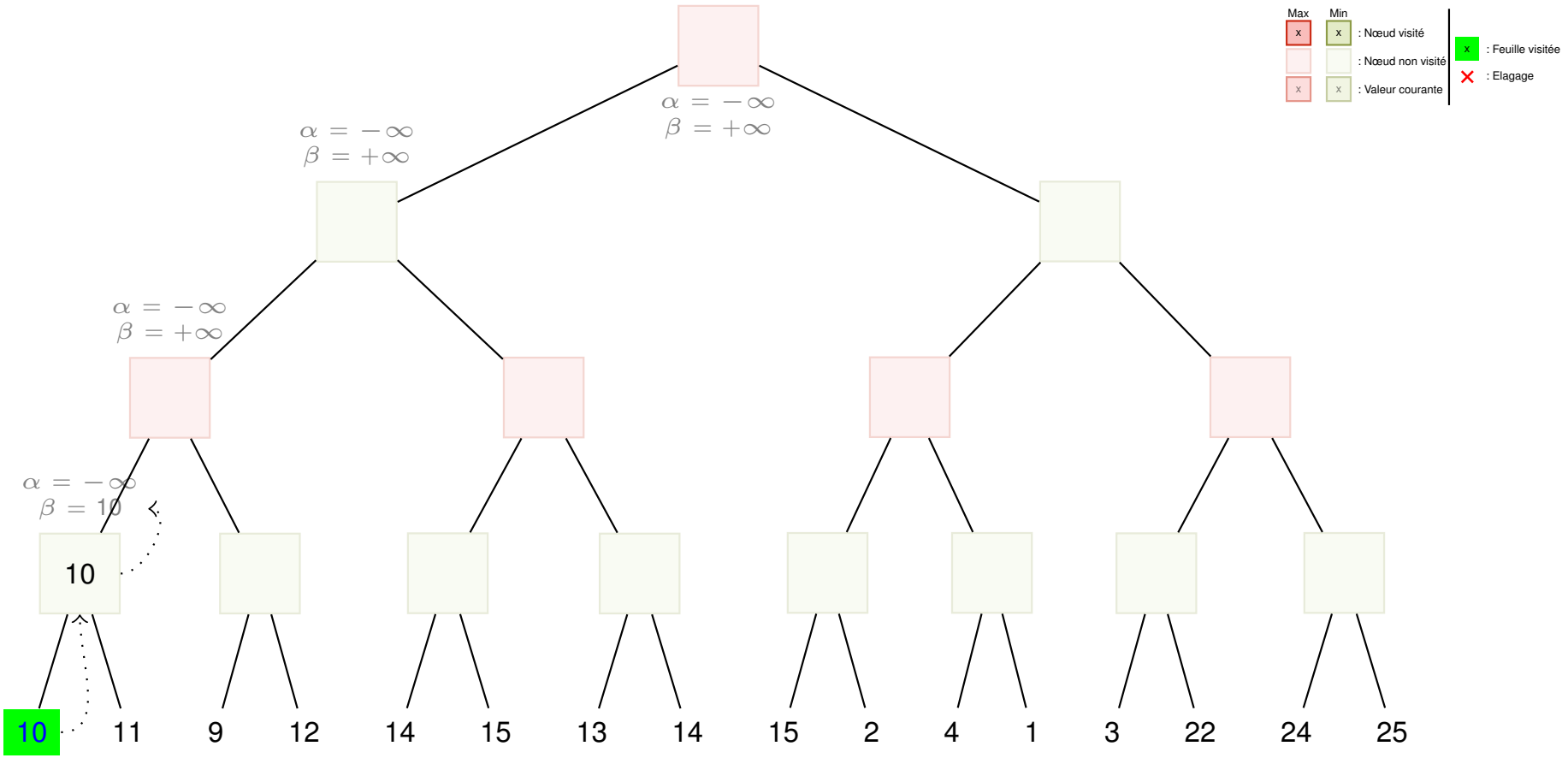
Exemples

Exemple 1



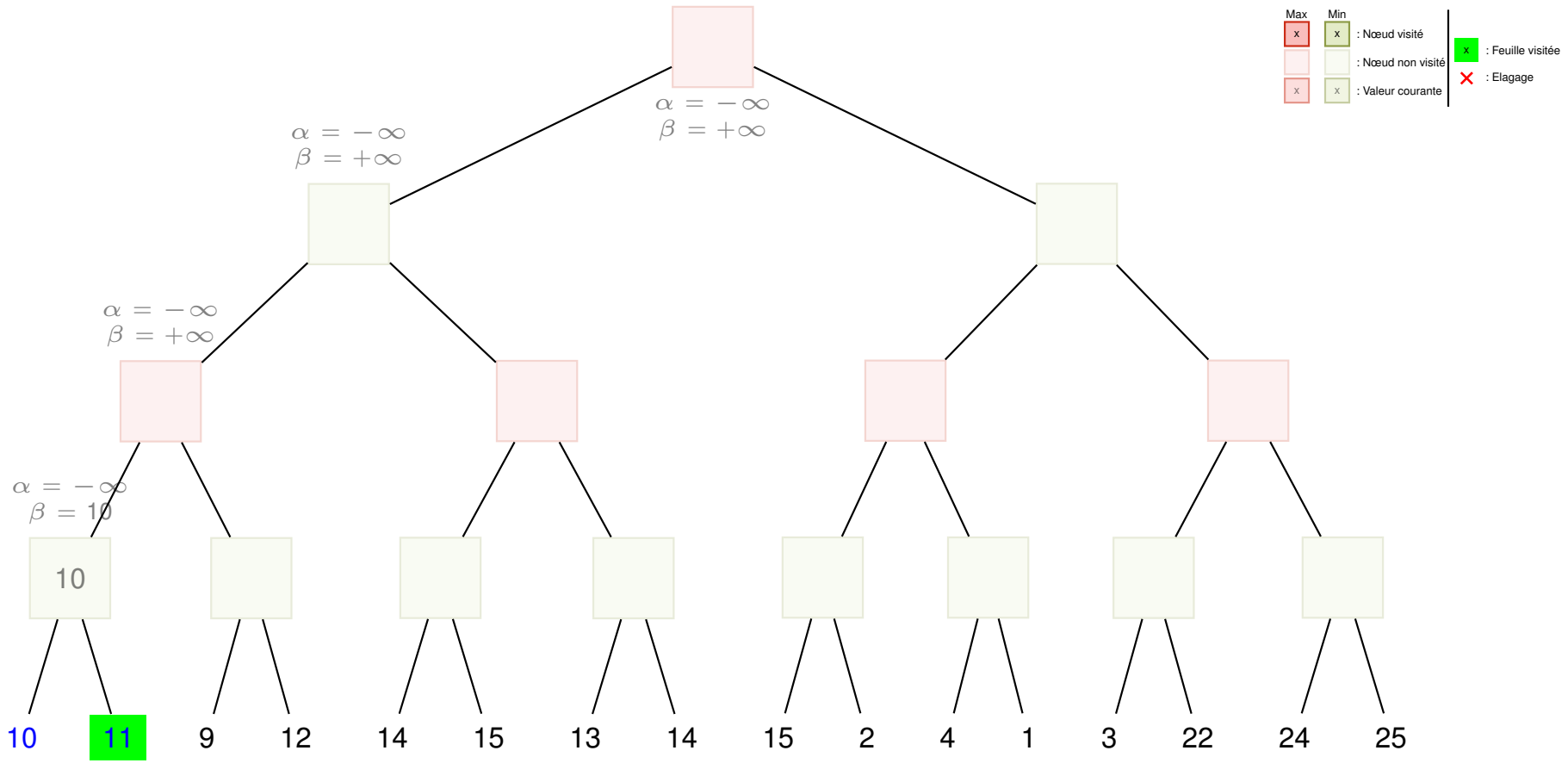
Exemples

Exemple 1



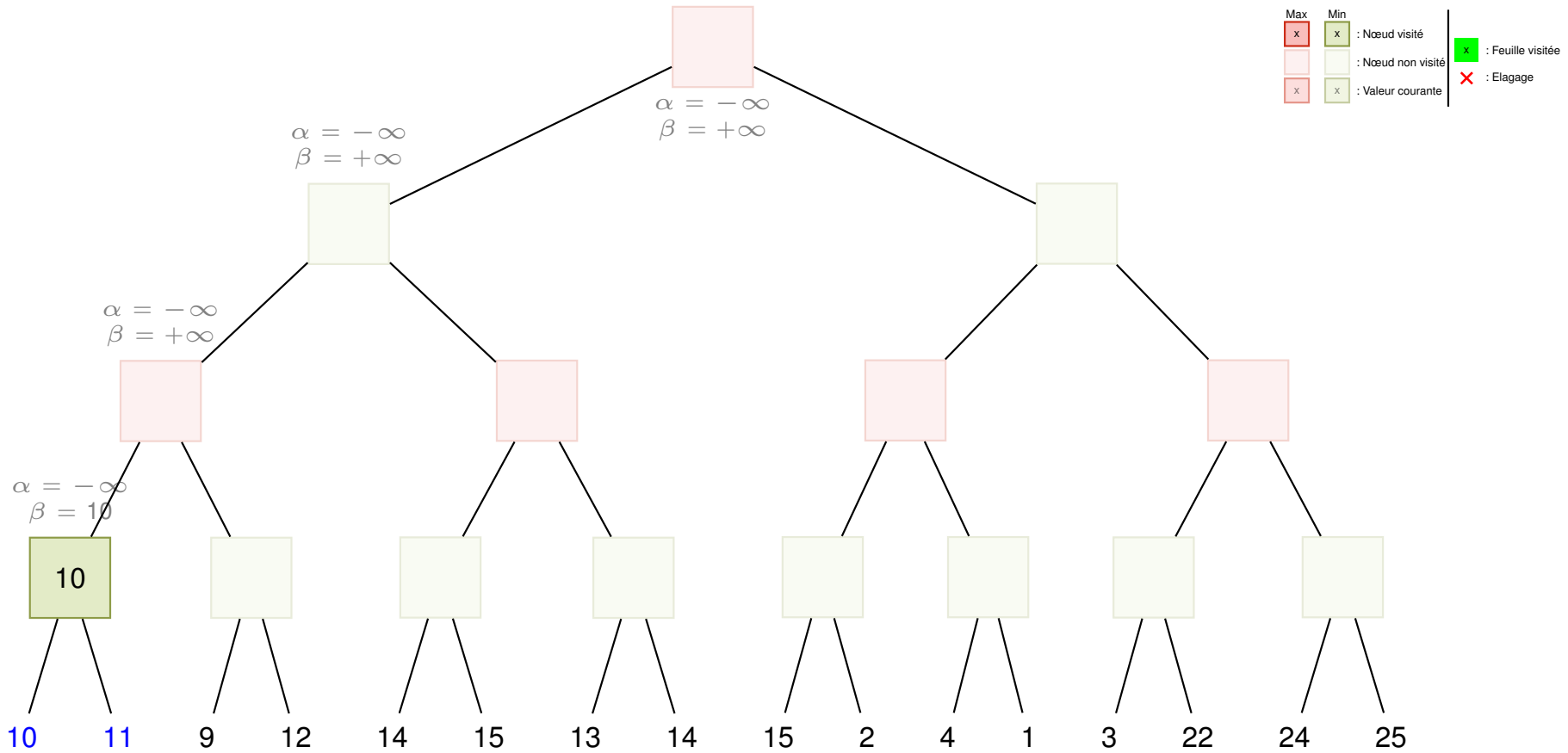
Exemples

Exemple 1



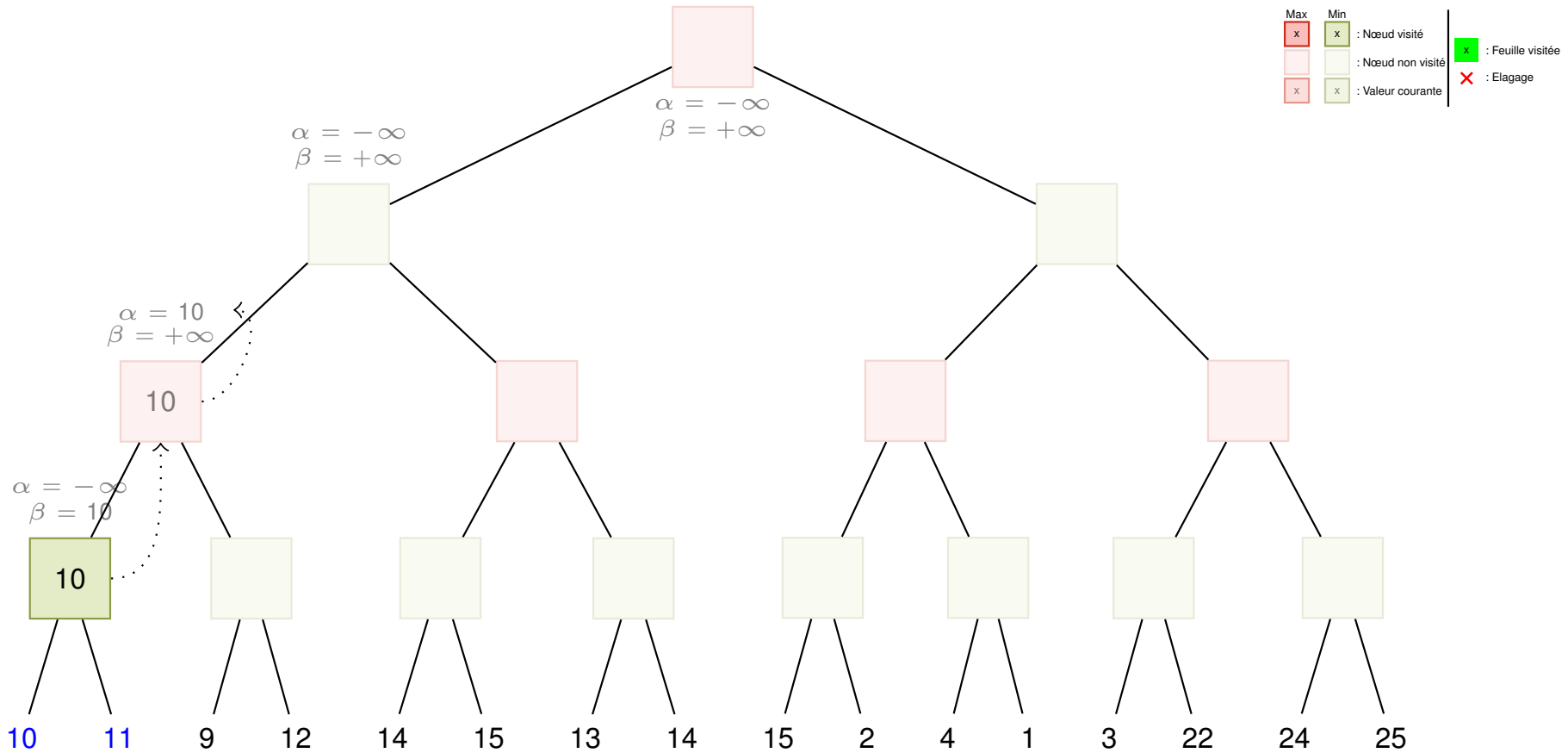
Exemples

Exemple 1



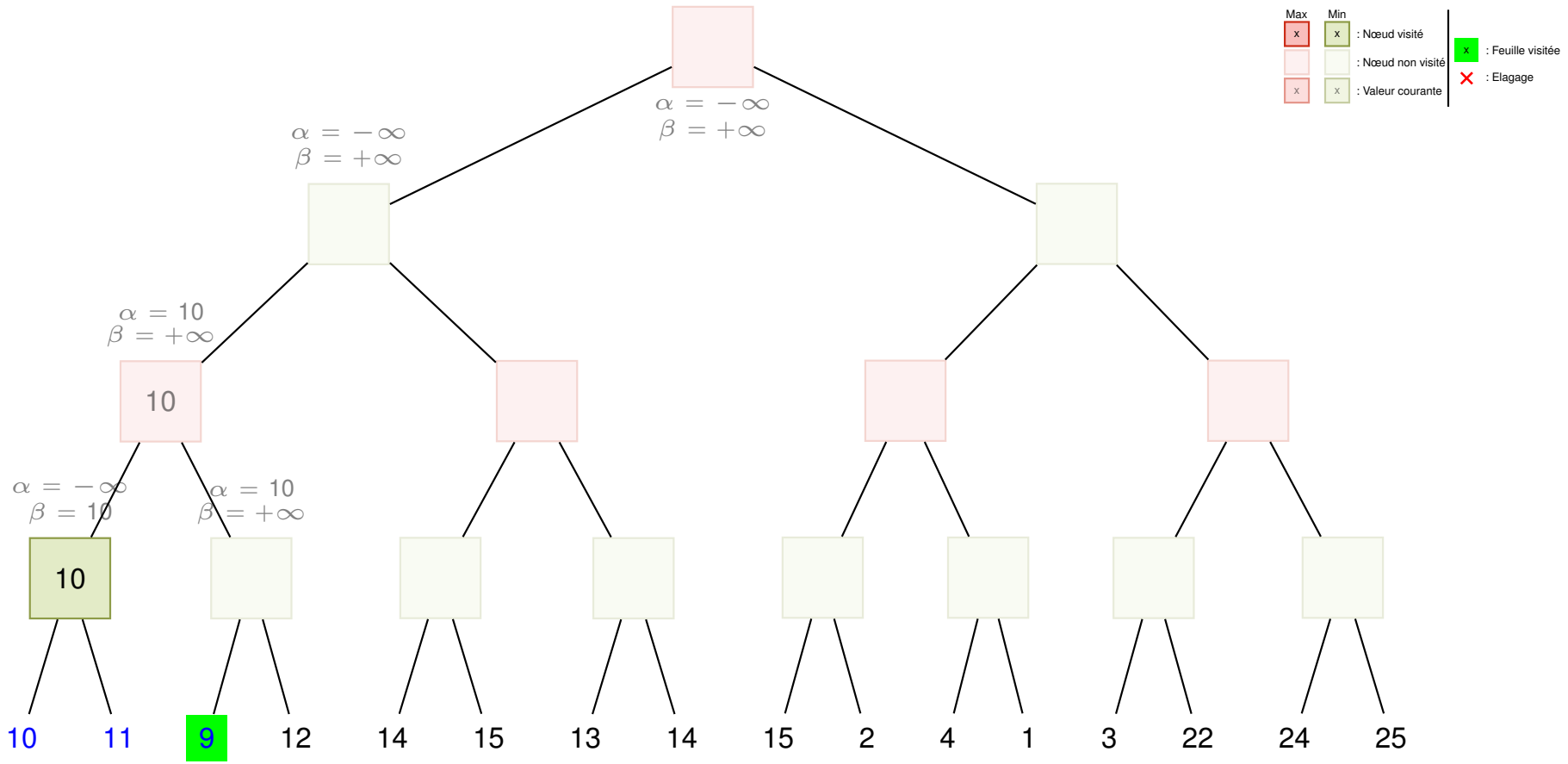
Exemples

Exemple 1



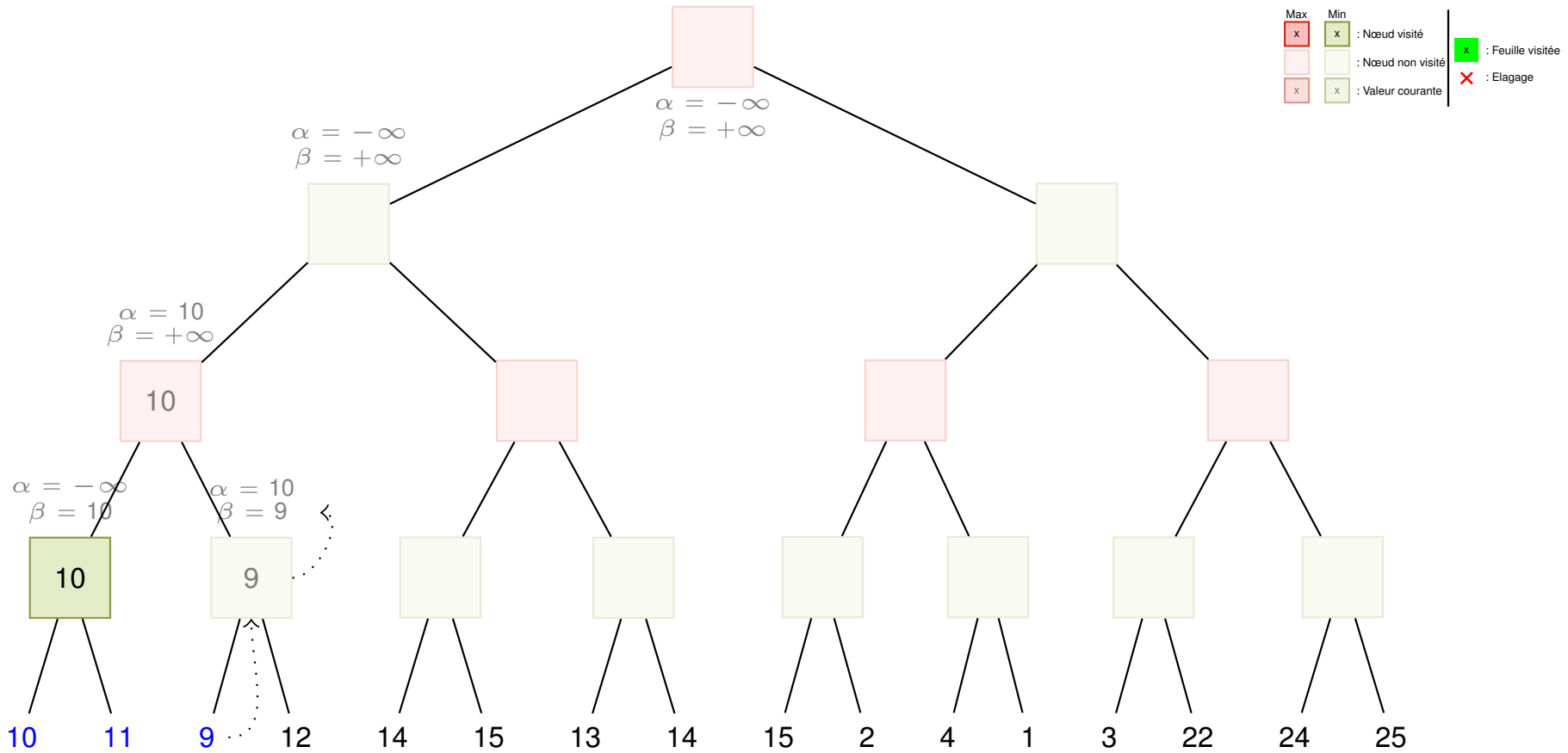
Exemples

Exemple 1



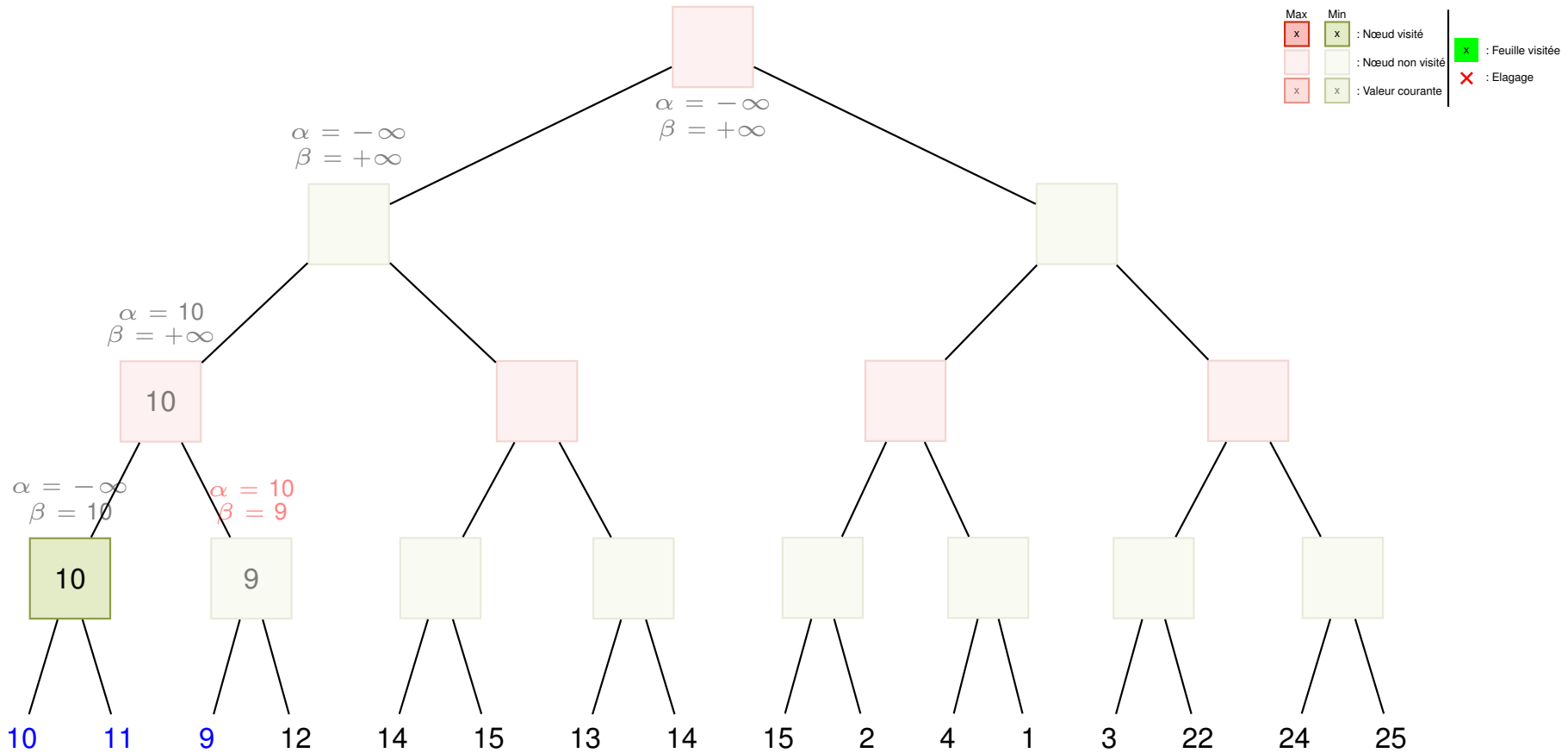
Exemples

Exemple 1



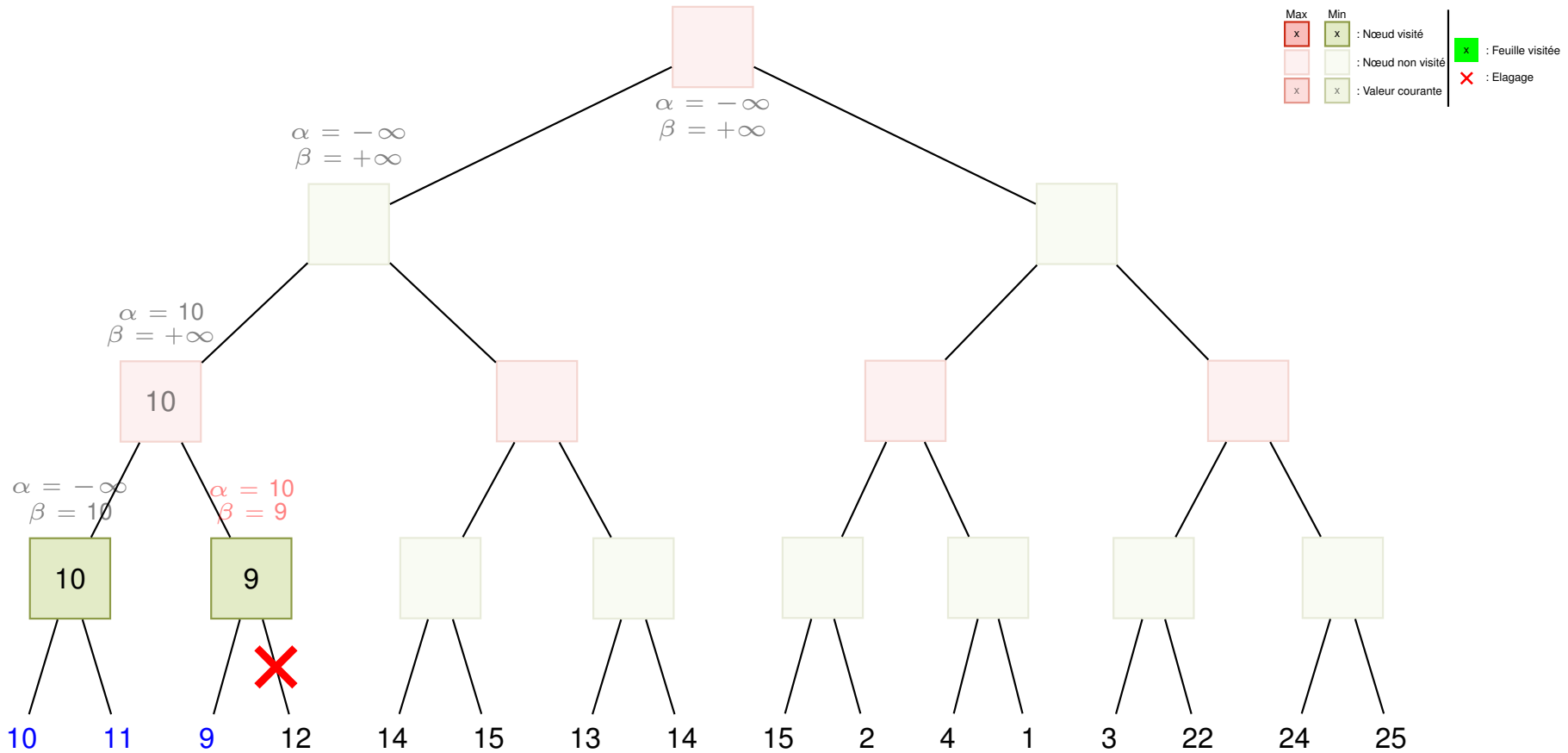
Exemples

Exemple 1



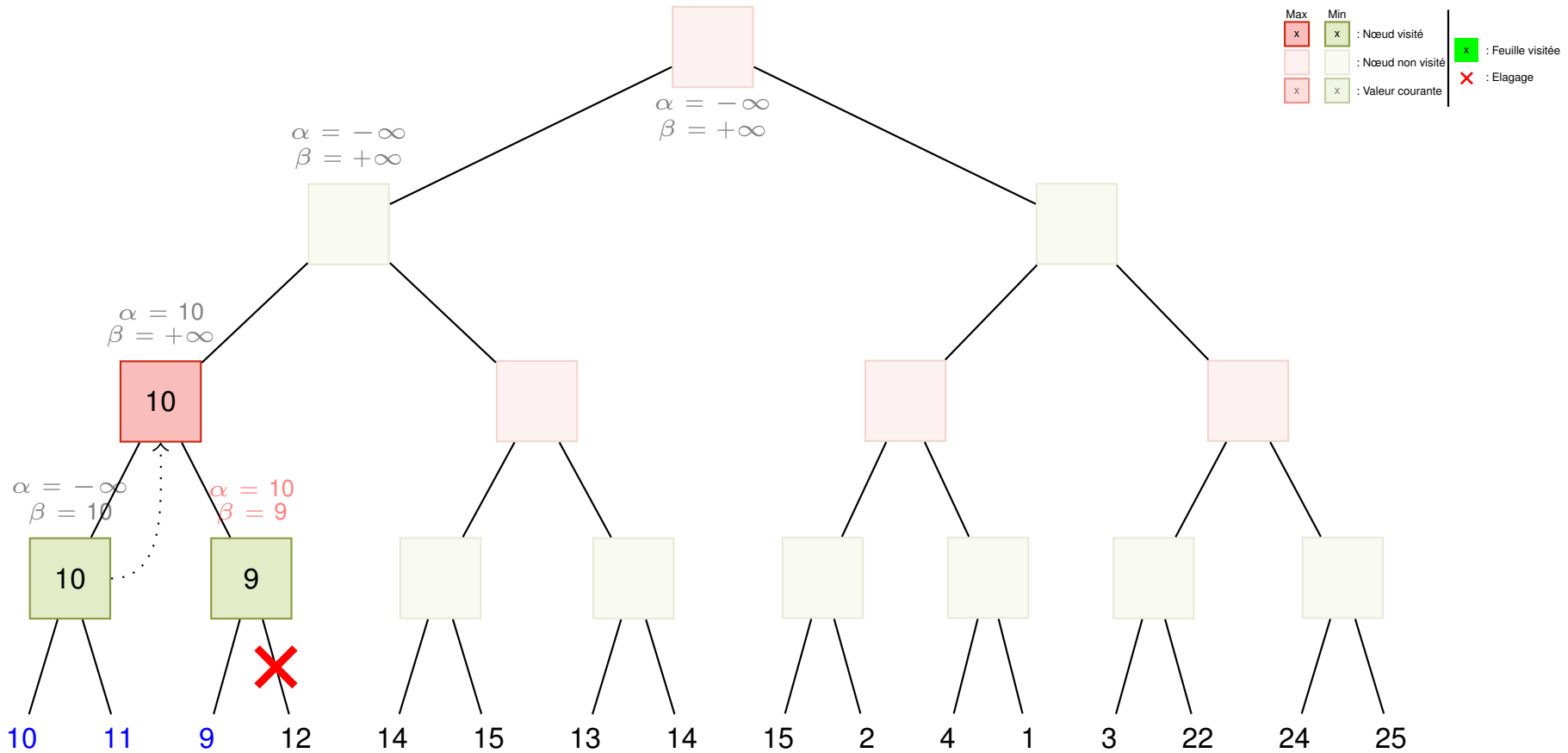
Exemples

Exemple 1



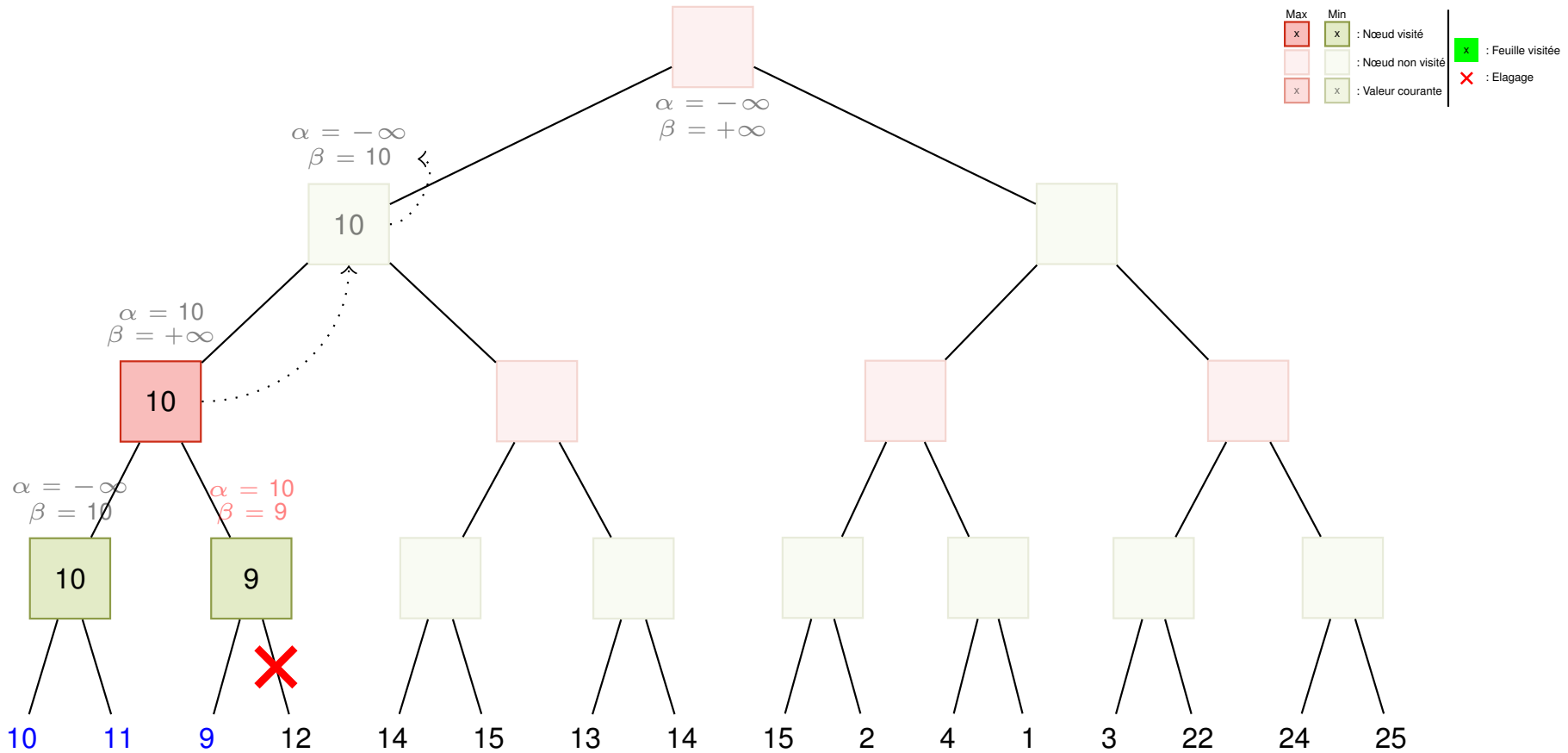
Exemples

Exemple 1



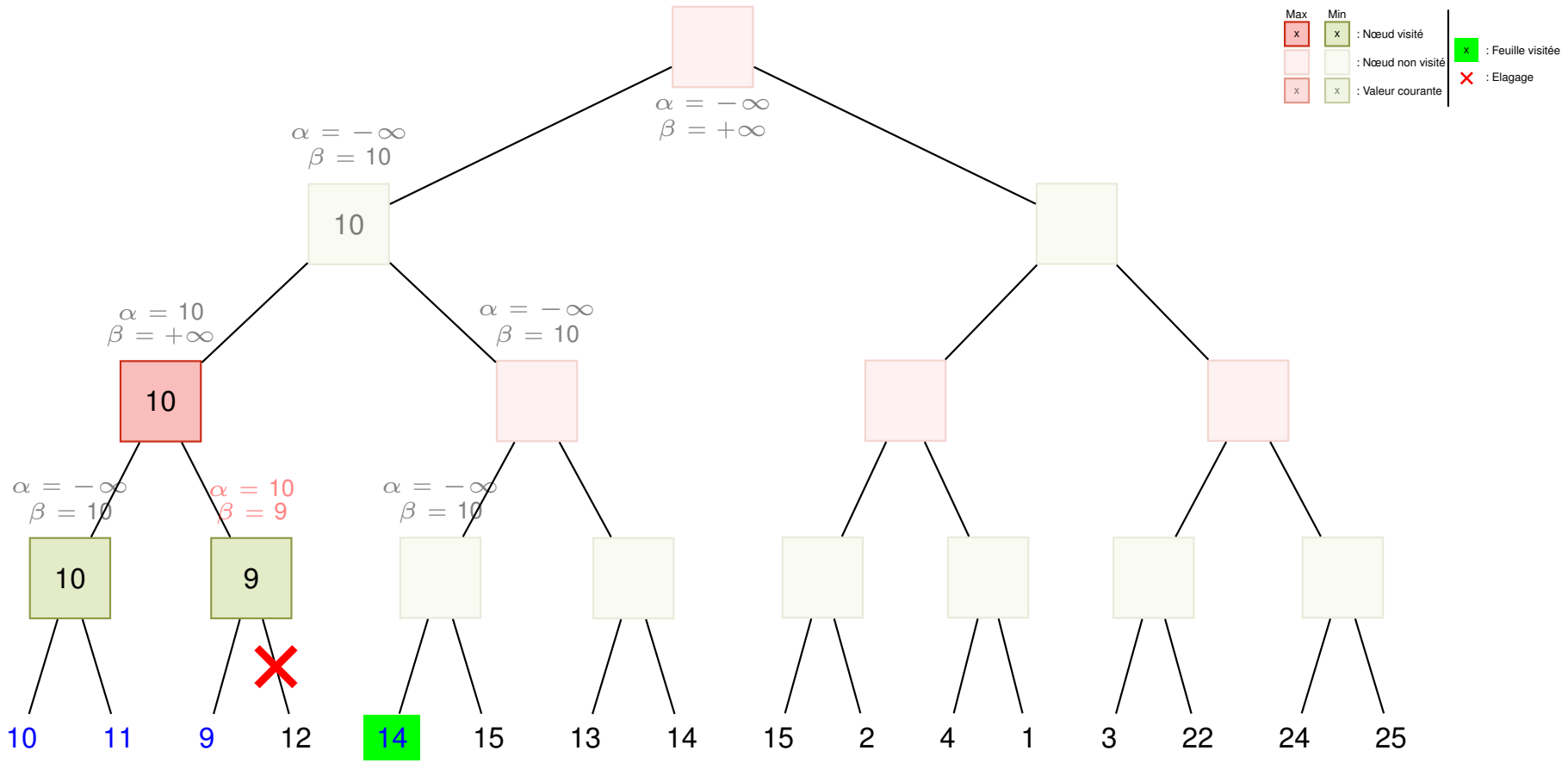
Exemples

Exemple 1



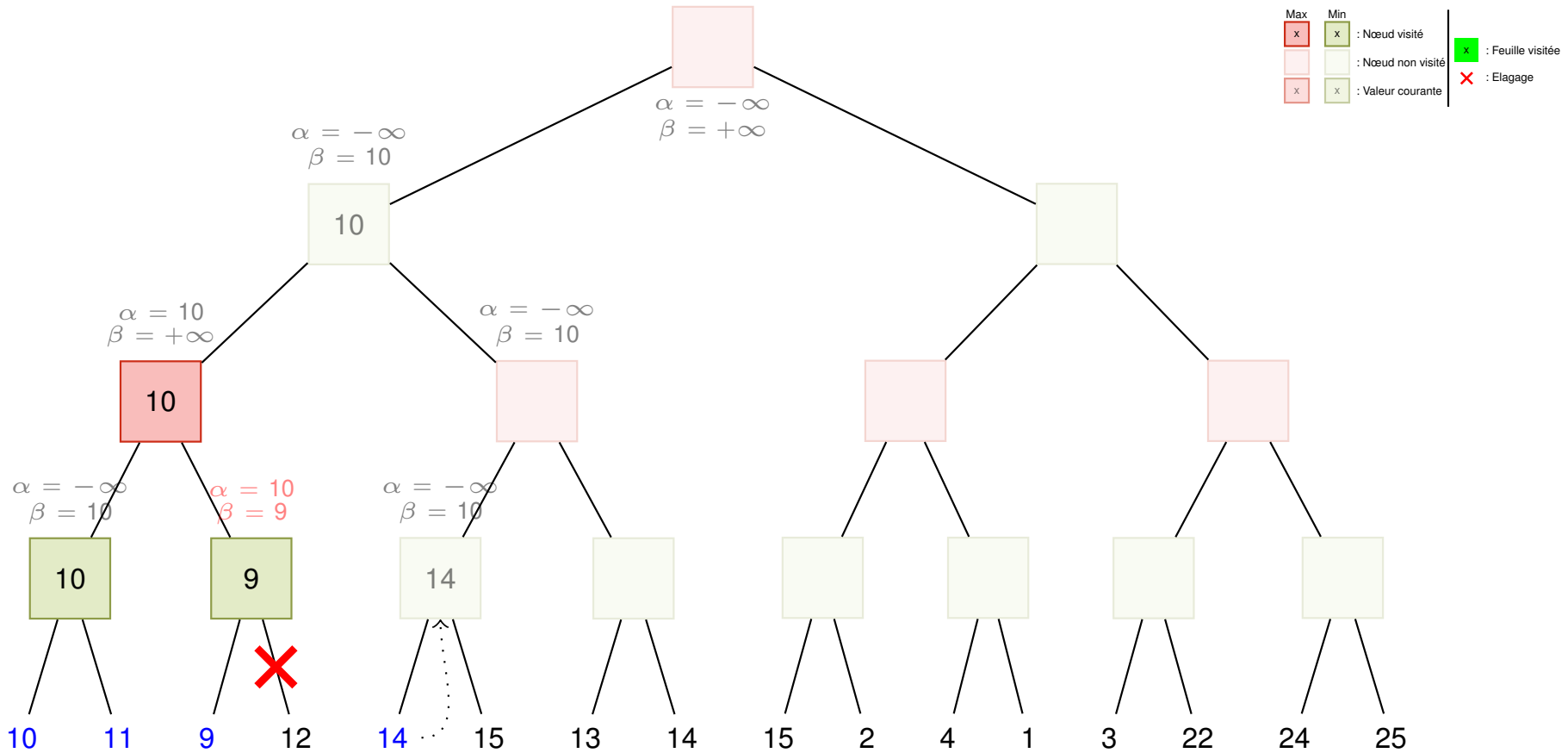
Exemples

Exemple 1



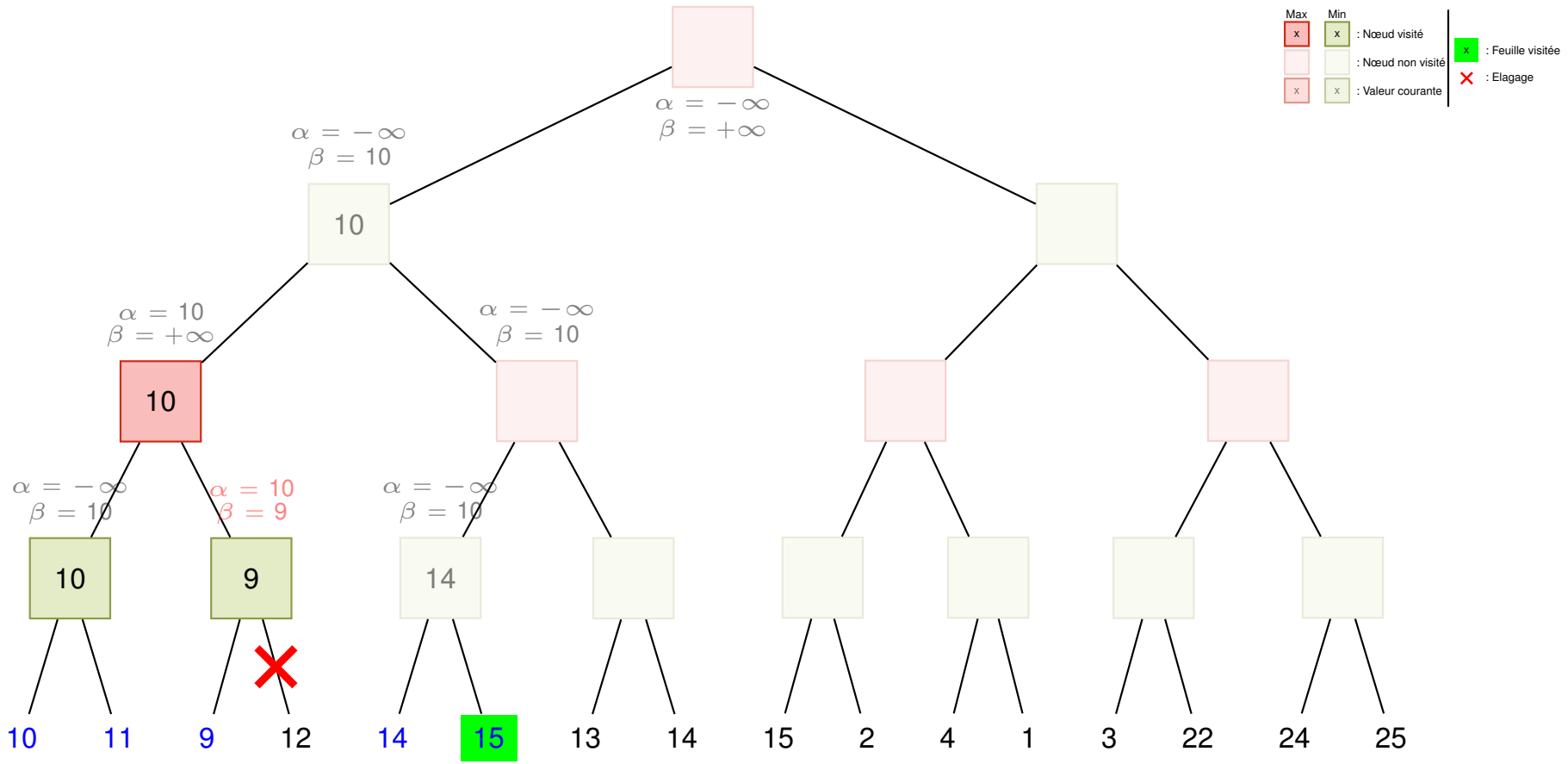
Exemples

Exemple 1



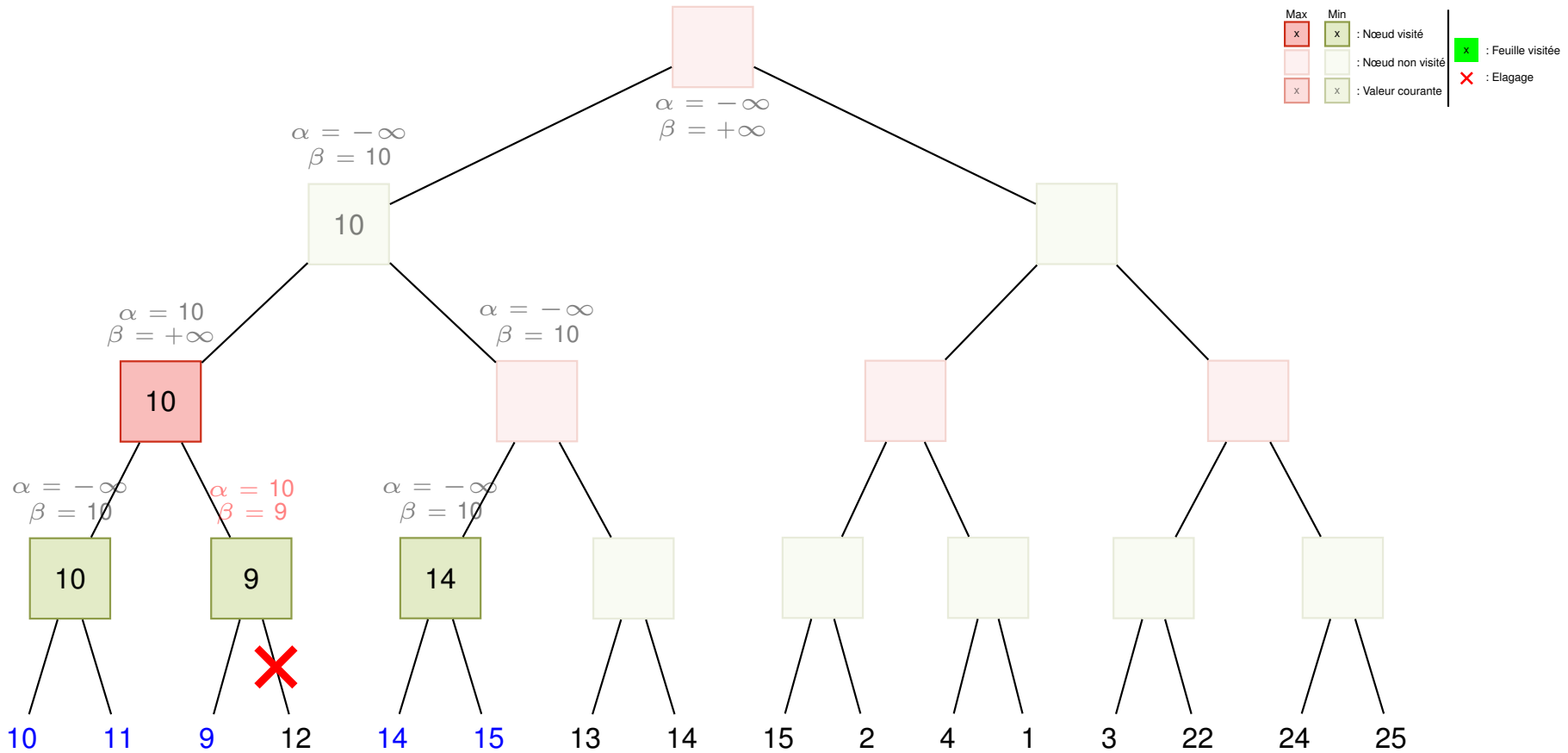
Exemples

Exemple 1



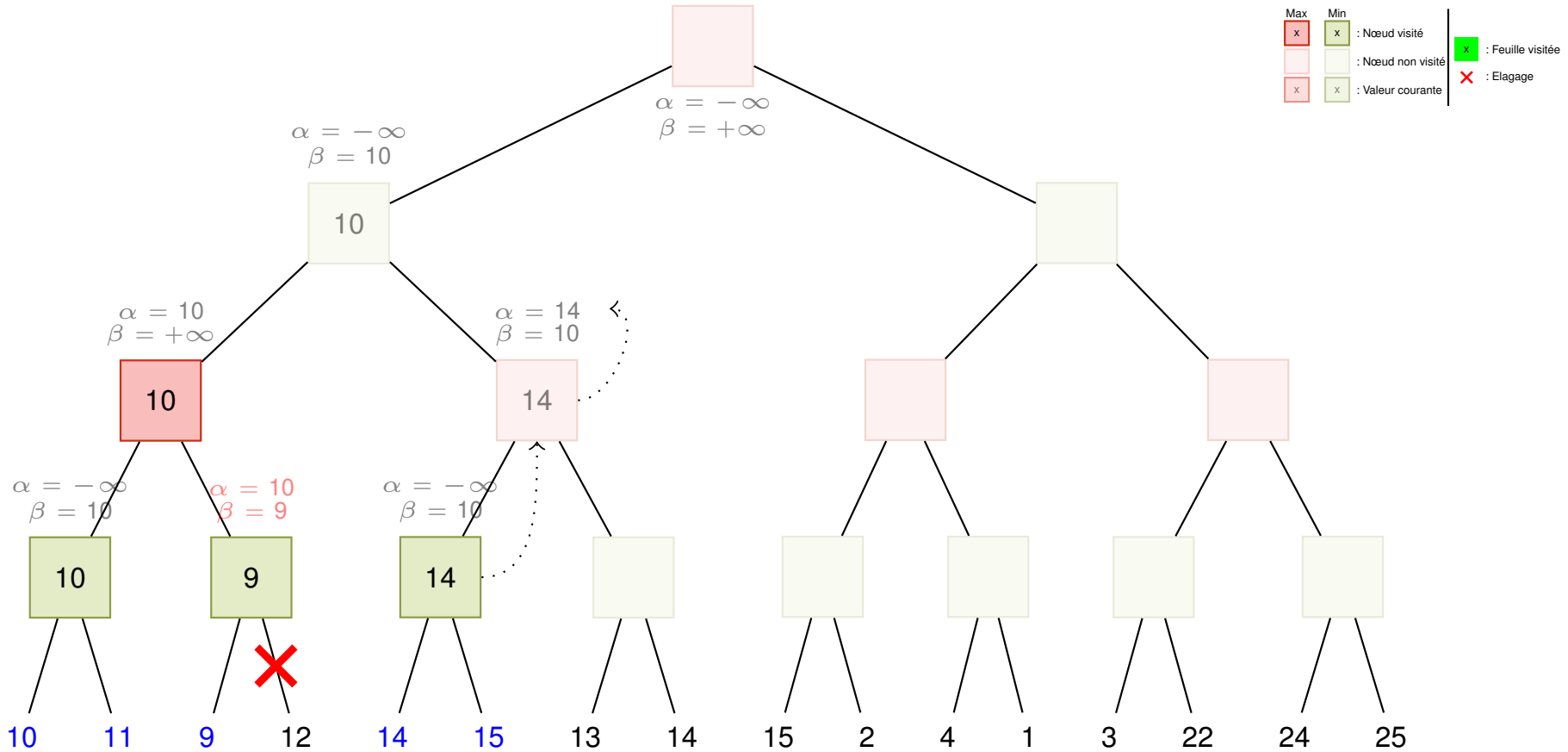
Exemples

Exemple 1



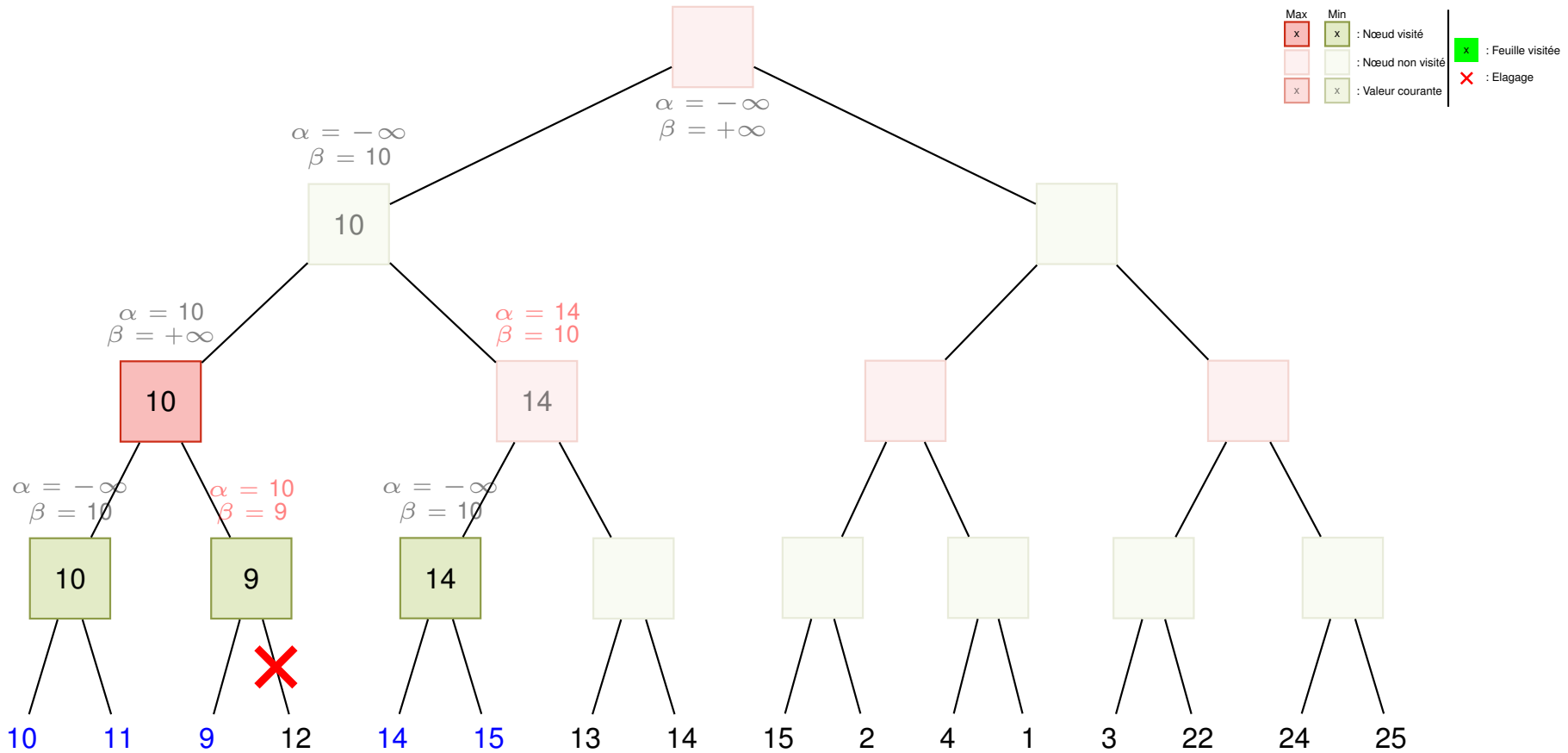
Exemples

Exemple 1



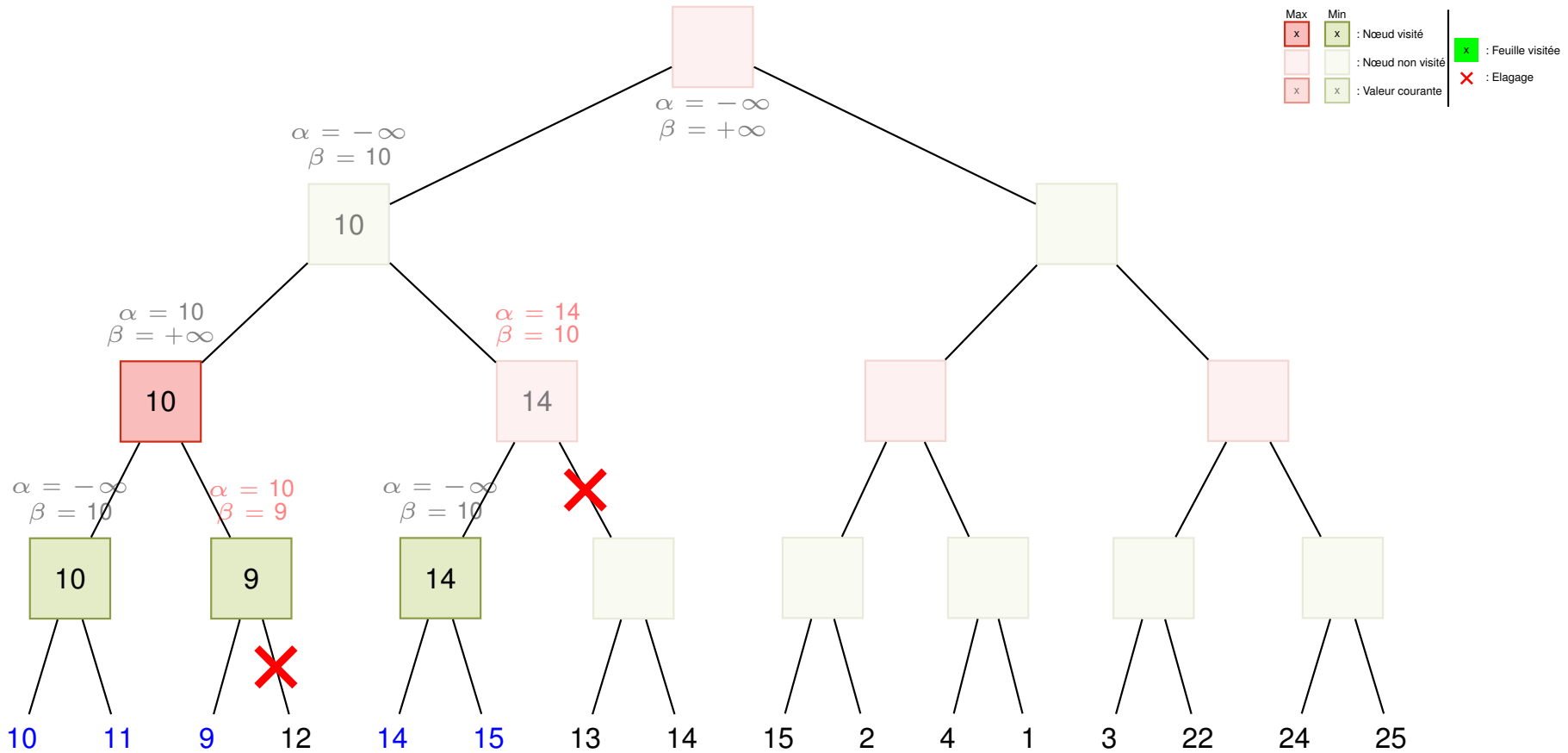
Exemples

Exemple 1



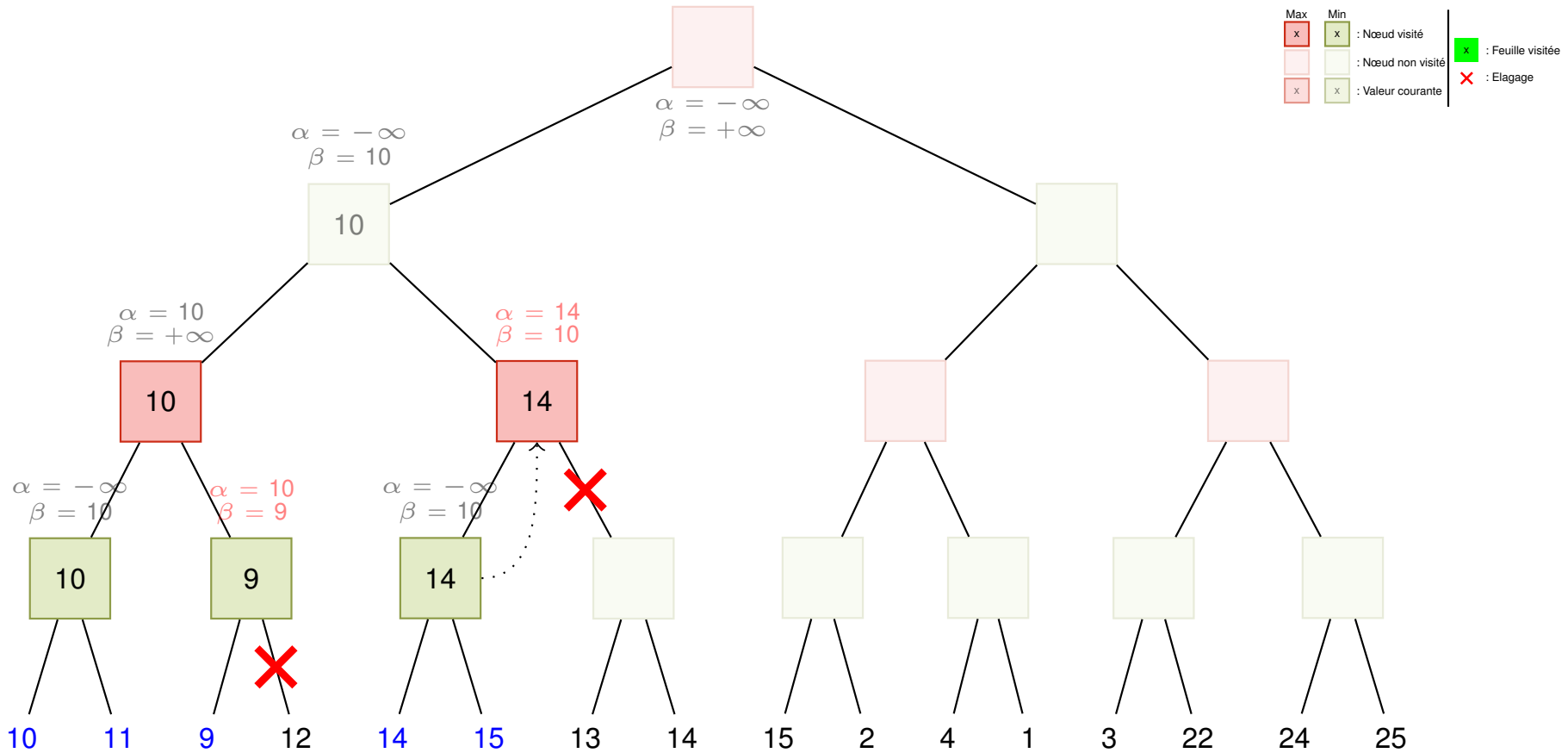
Exemples

Exemple 1



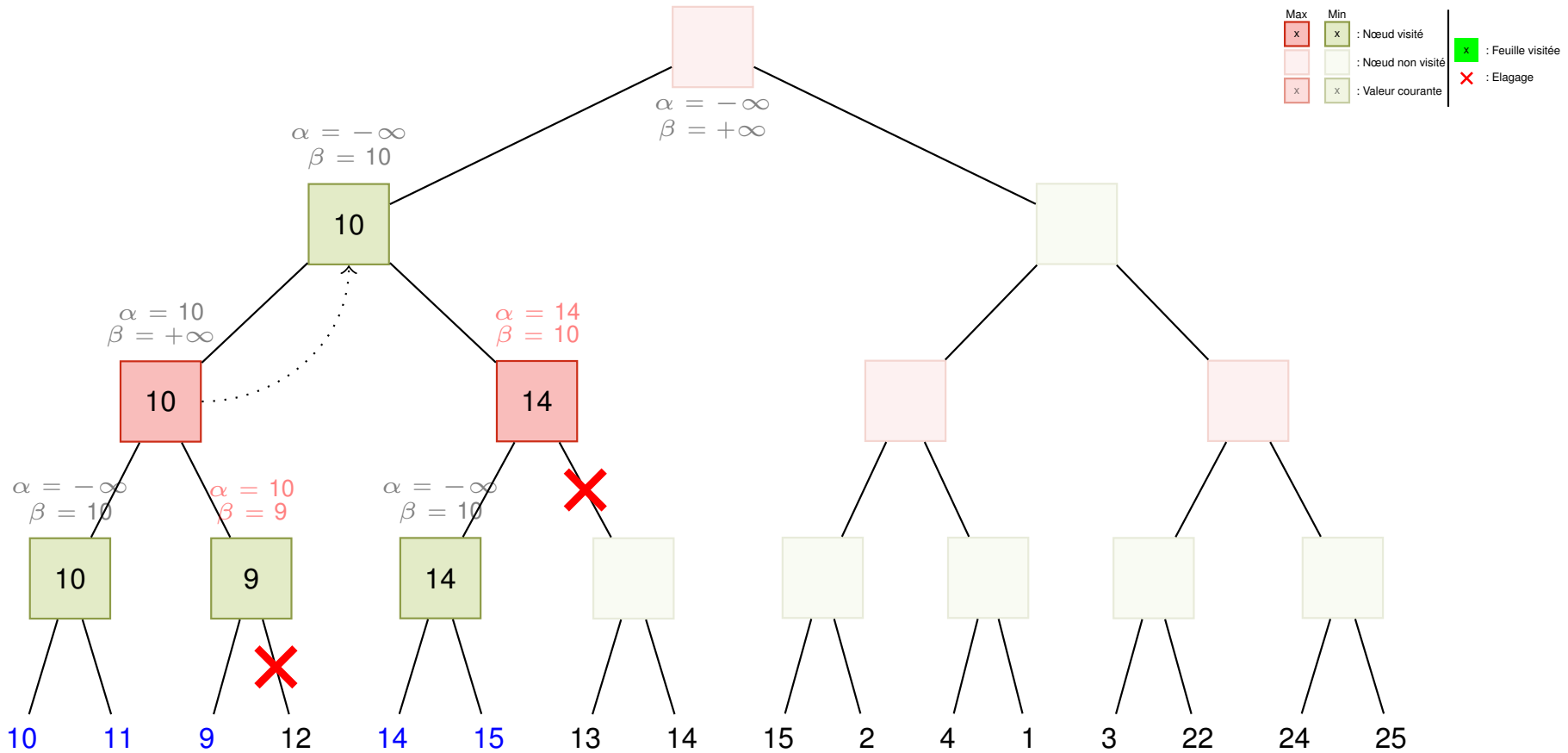
Exemples

Exemple 1



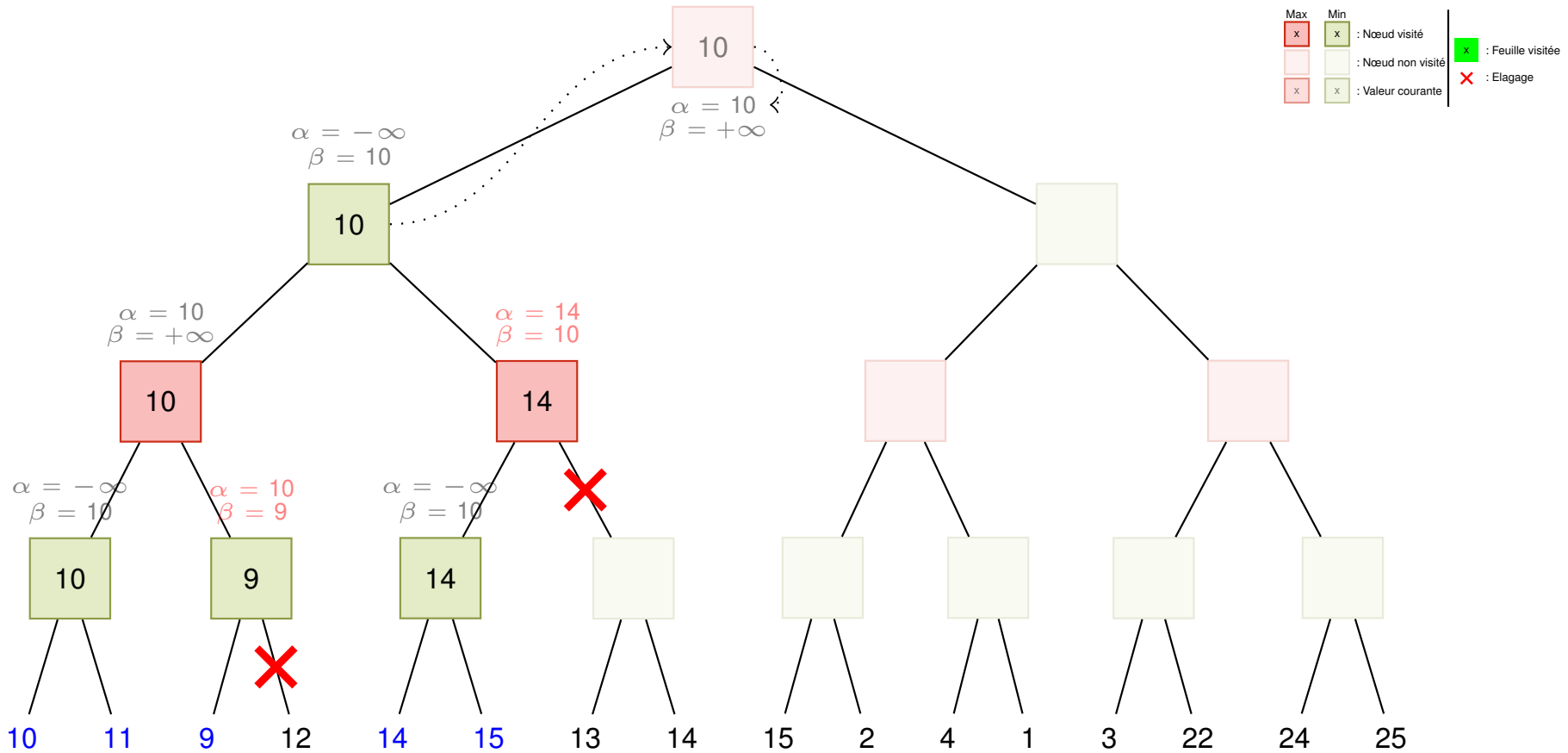
Exemples

Exemple 1



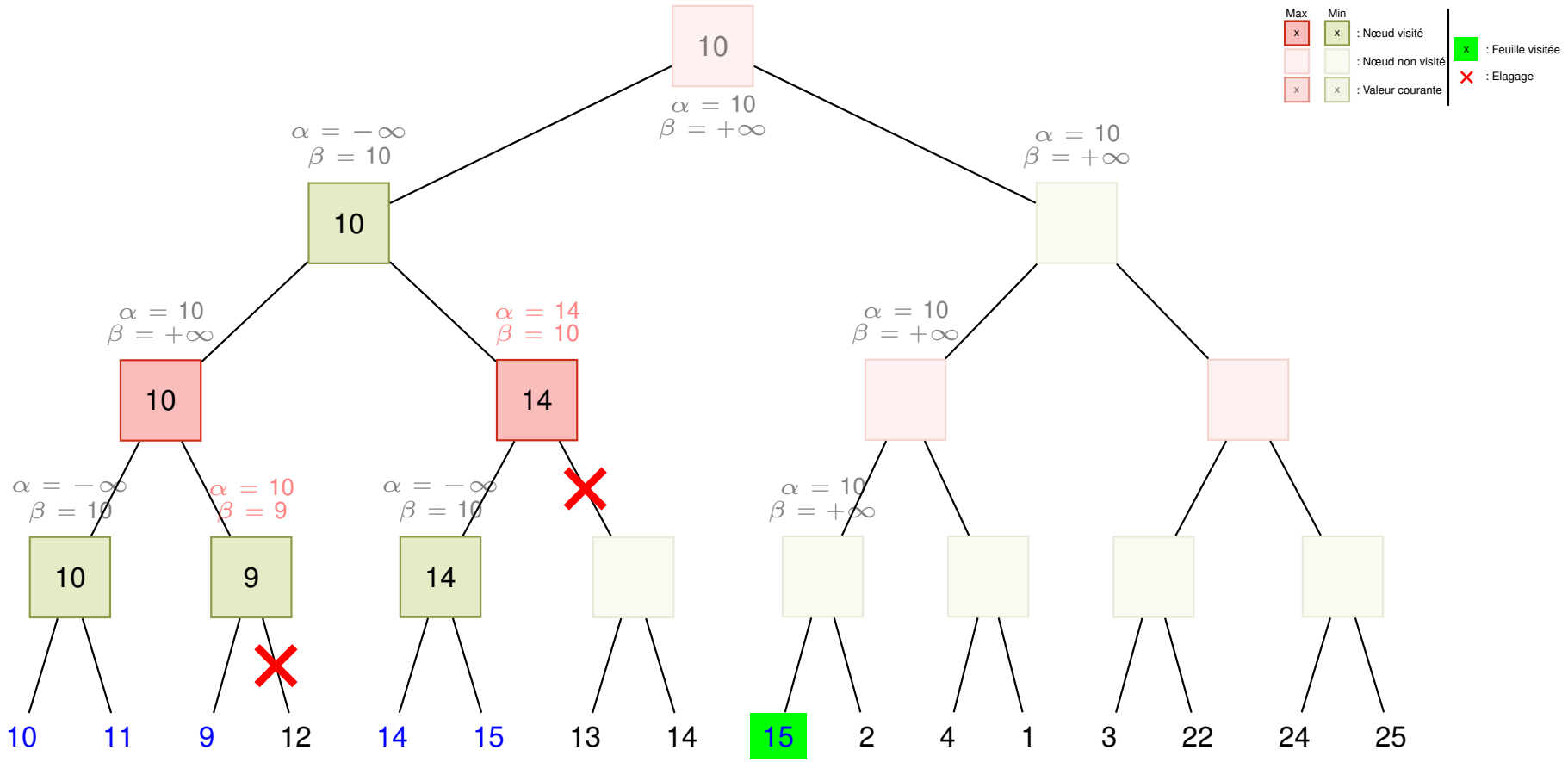
Exemples

Exemple 1



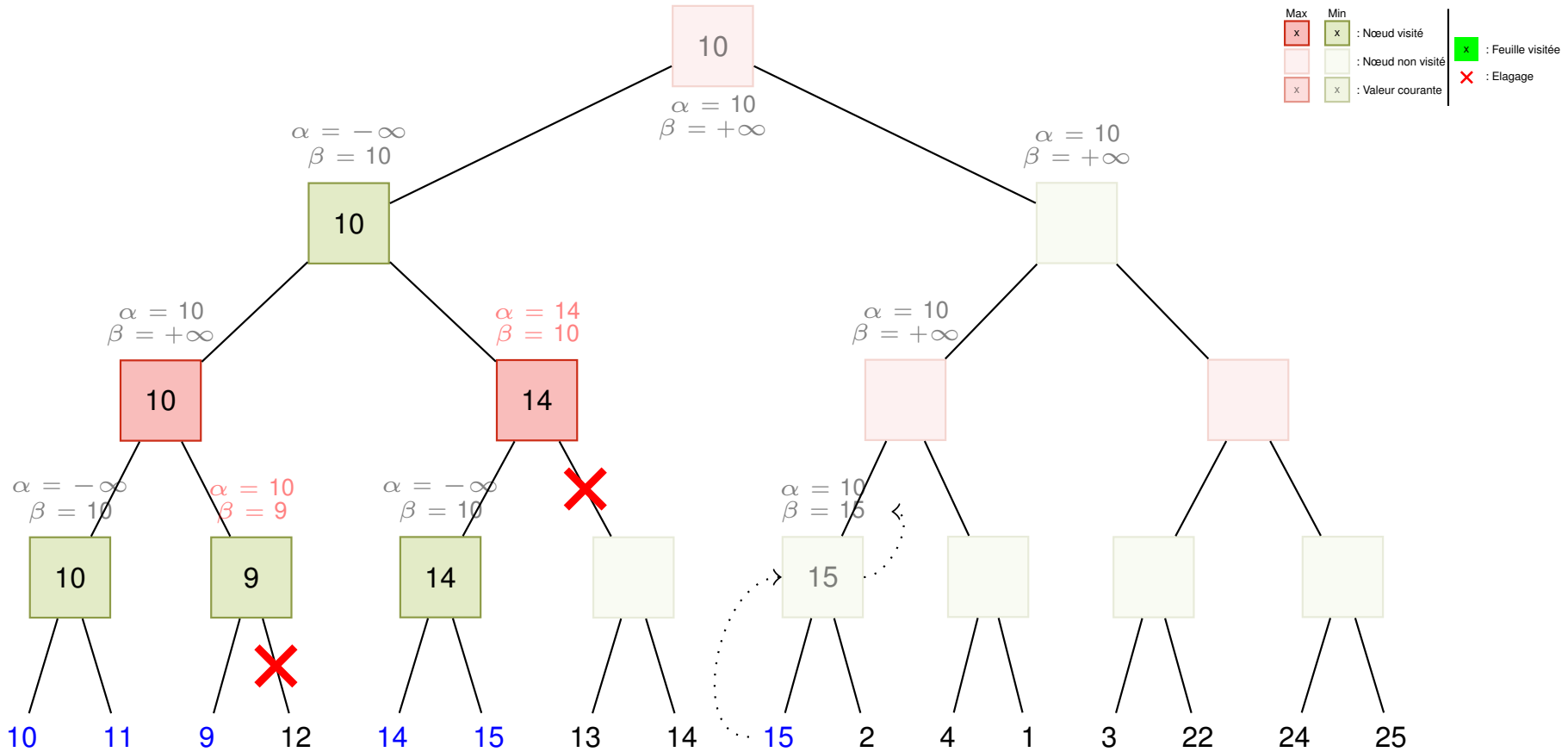
Exemples

Exemple 1



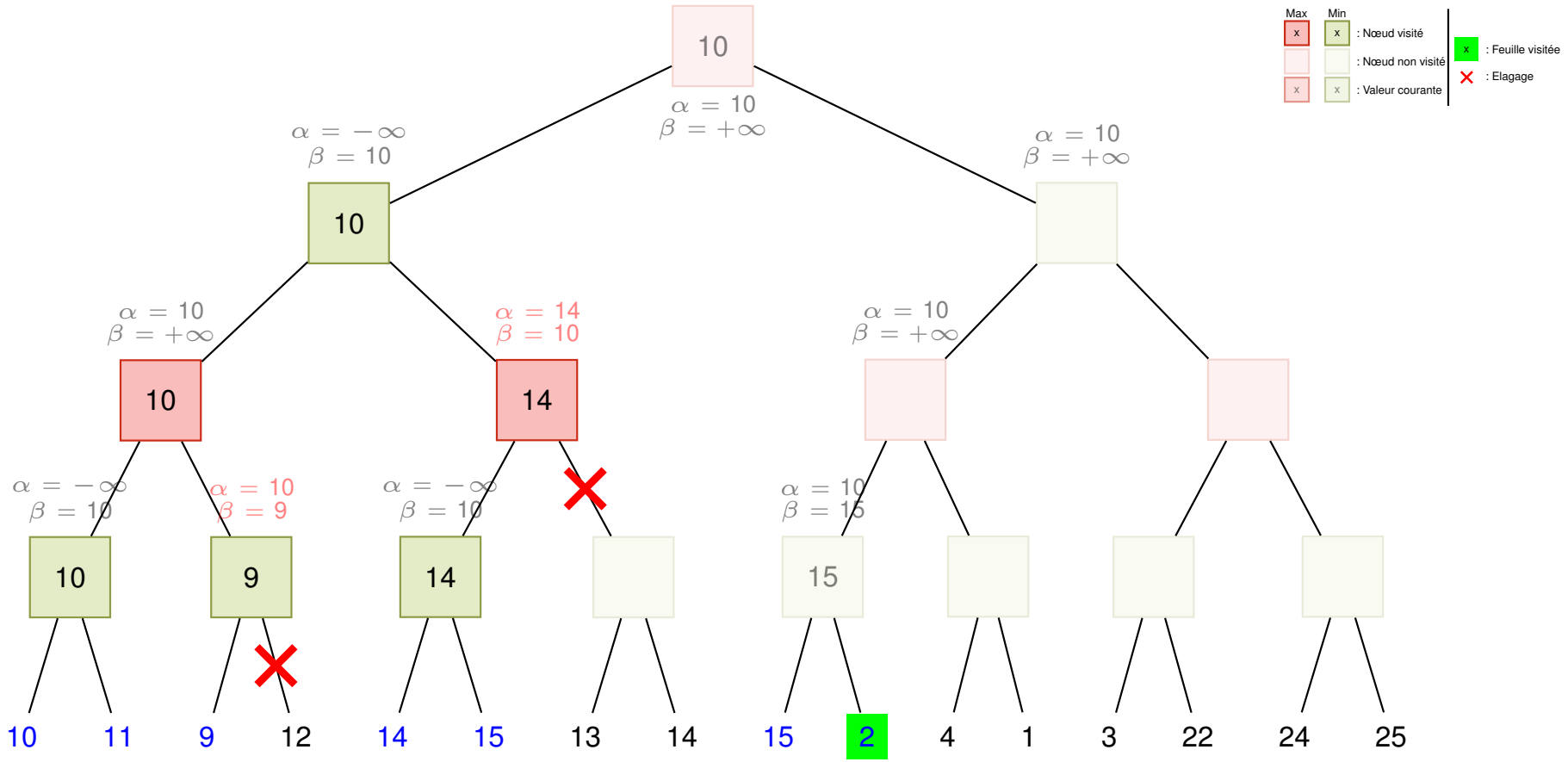
Exemples

Exemple 1



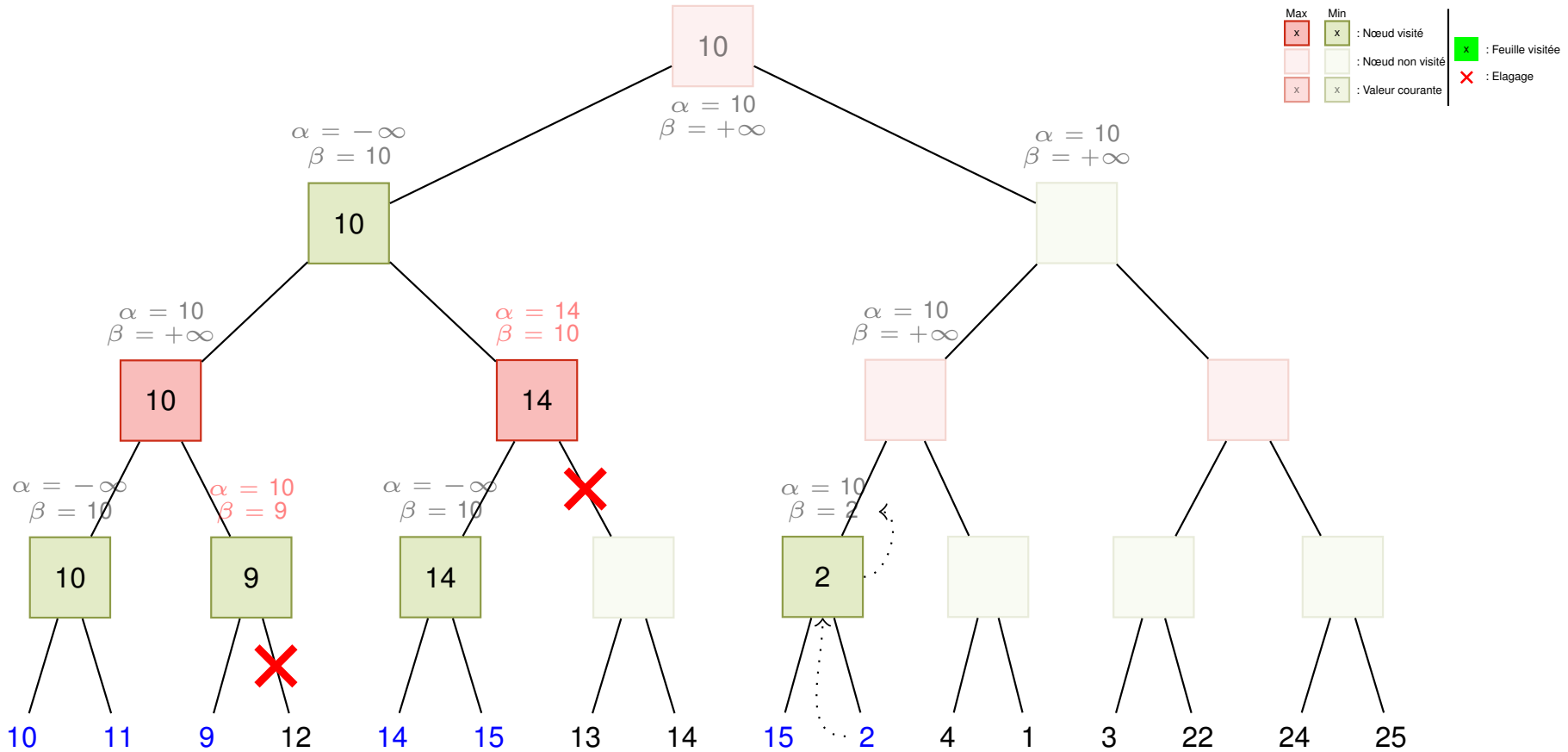
Exemples

Exemple 1



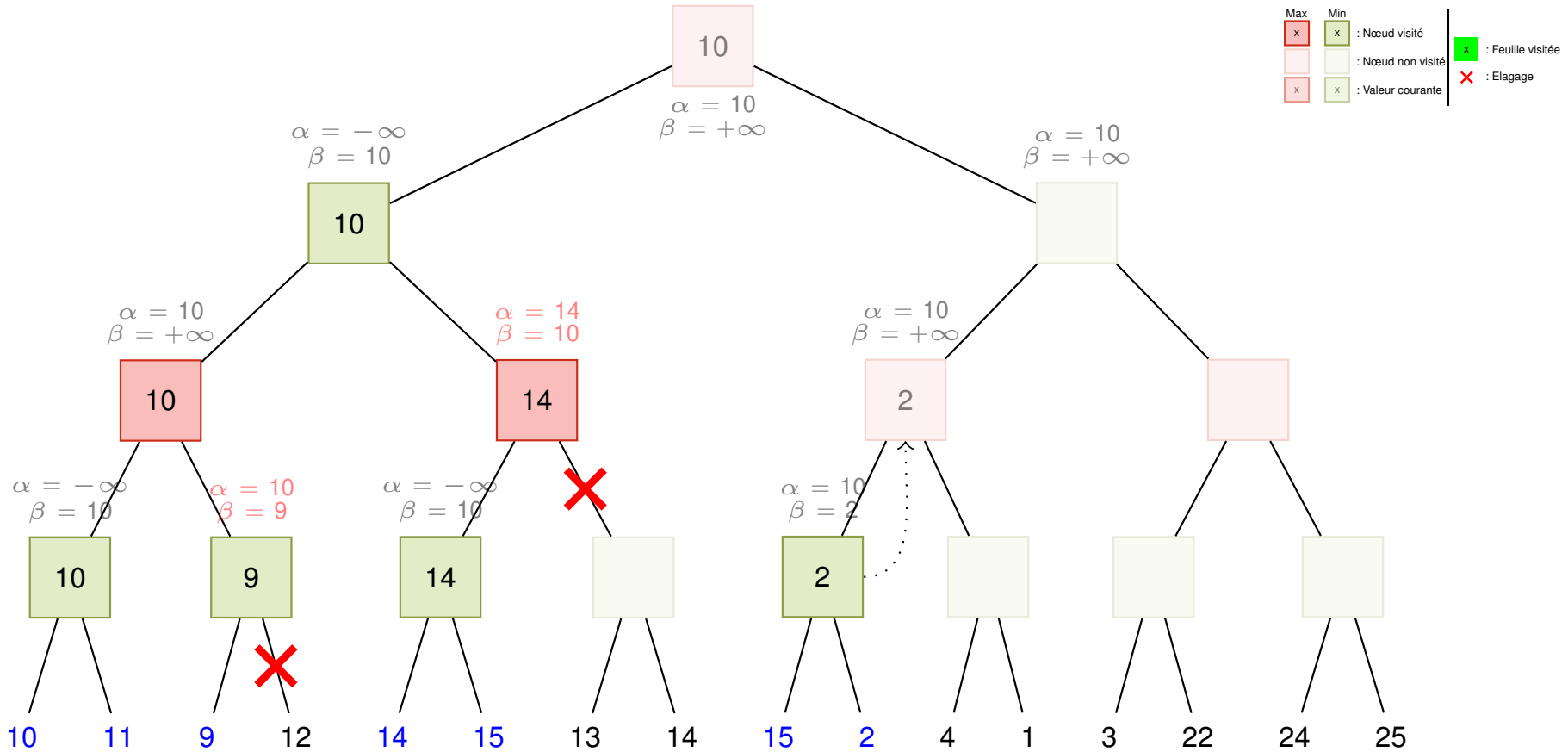
Exemples

Exemple 1



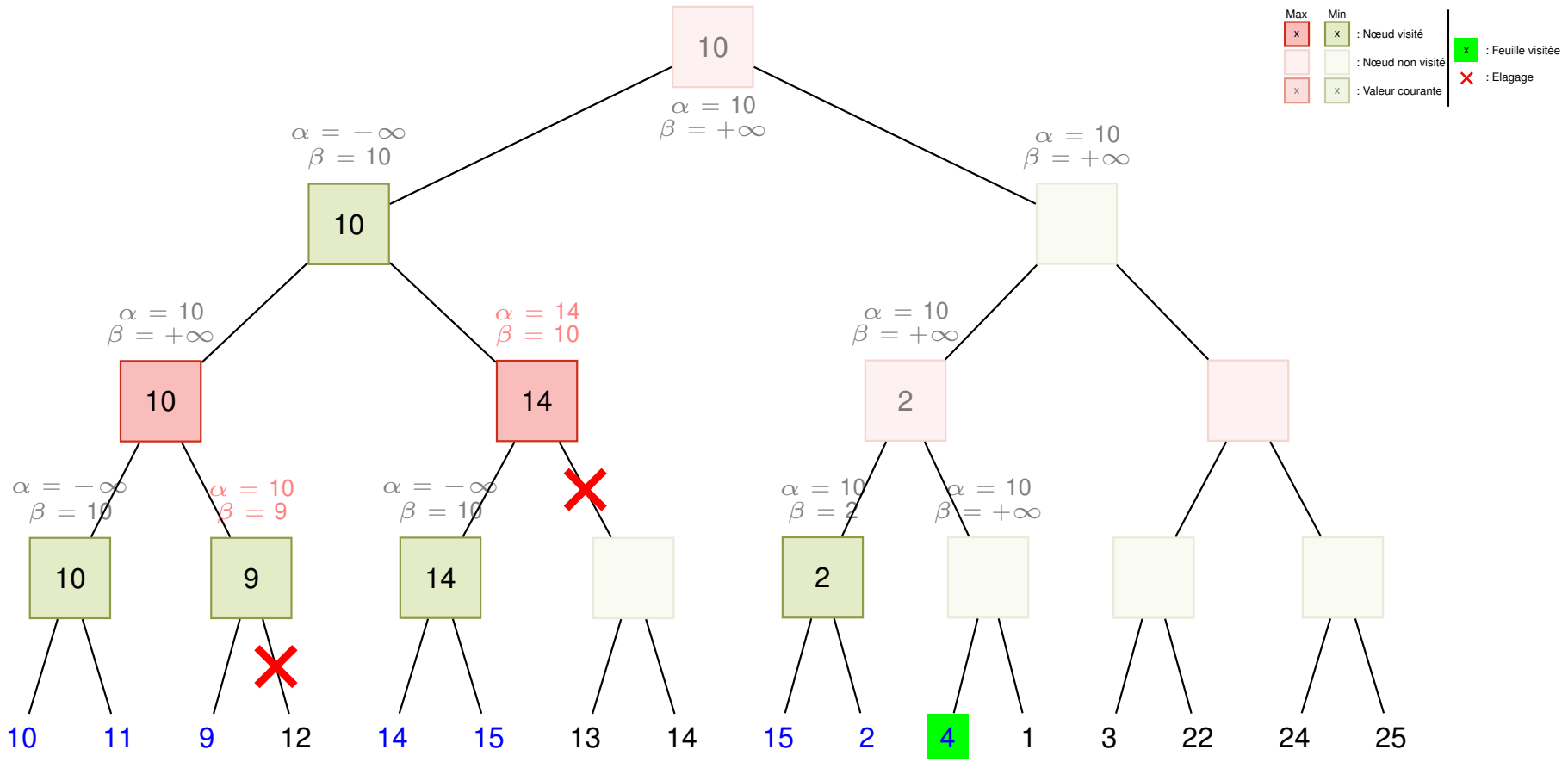
Exemples

Exemple 1



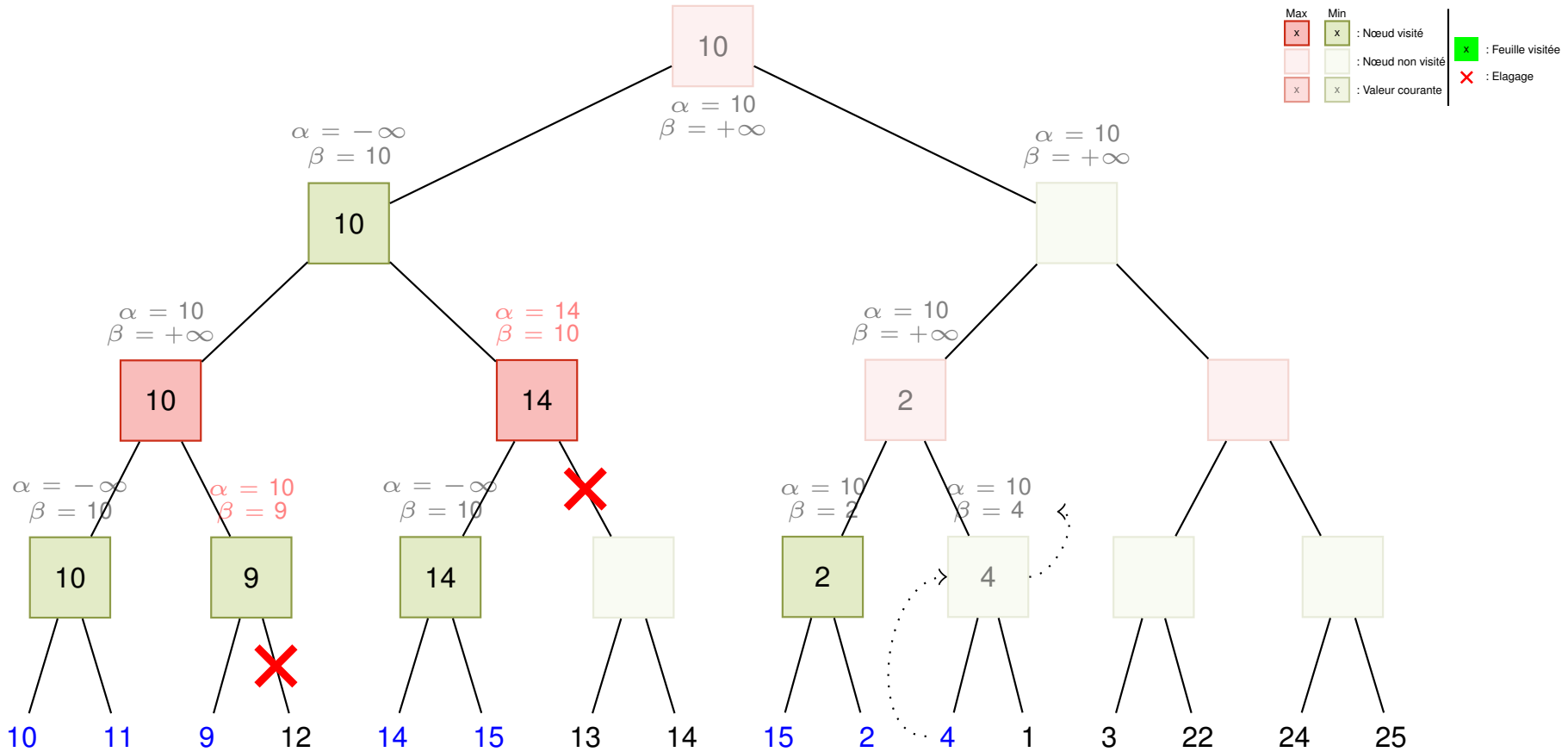
Exemples

Exemple 1



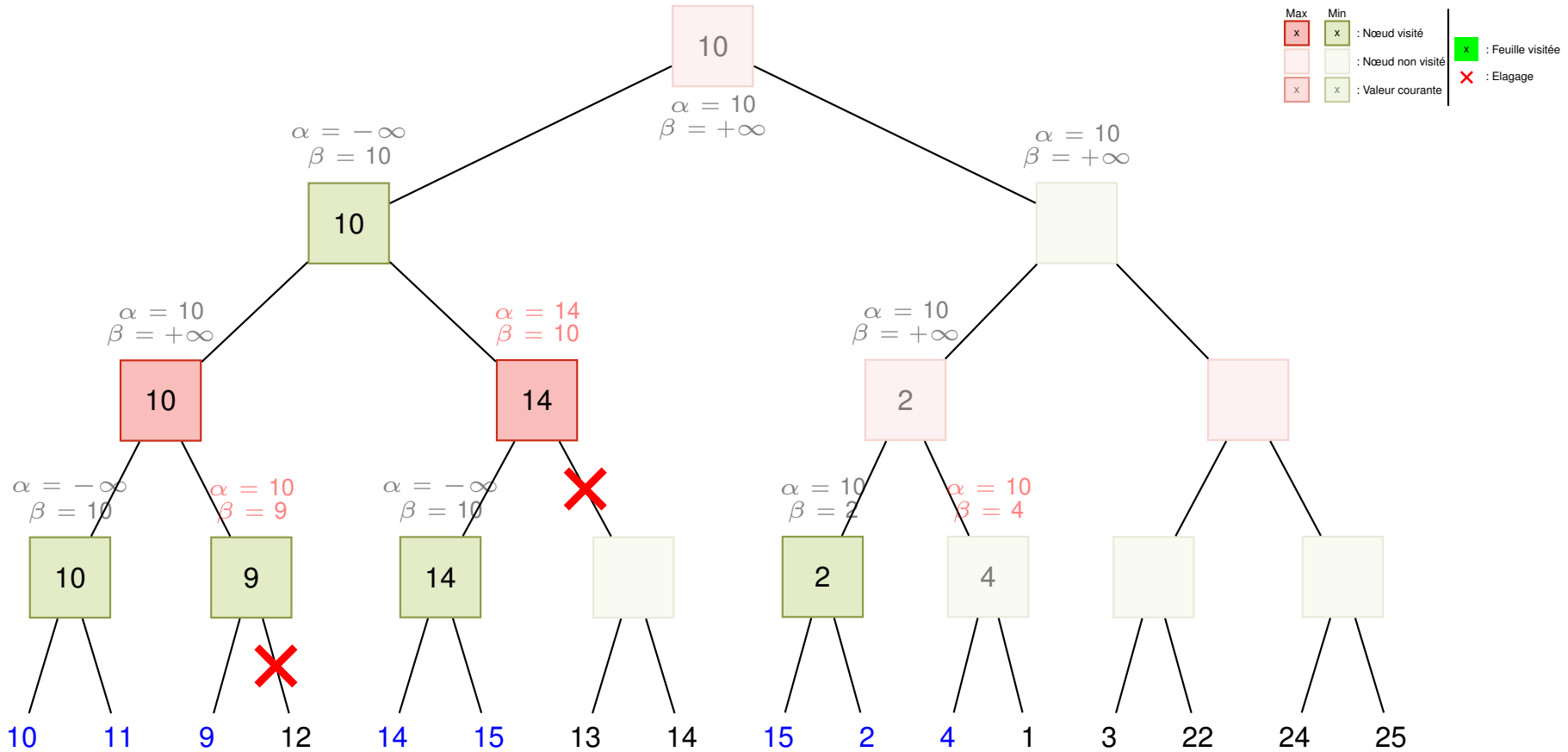
Exemples

Exemple 1



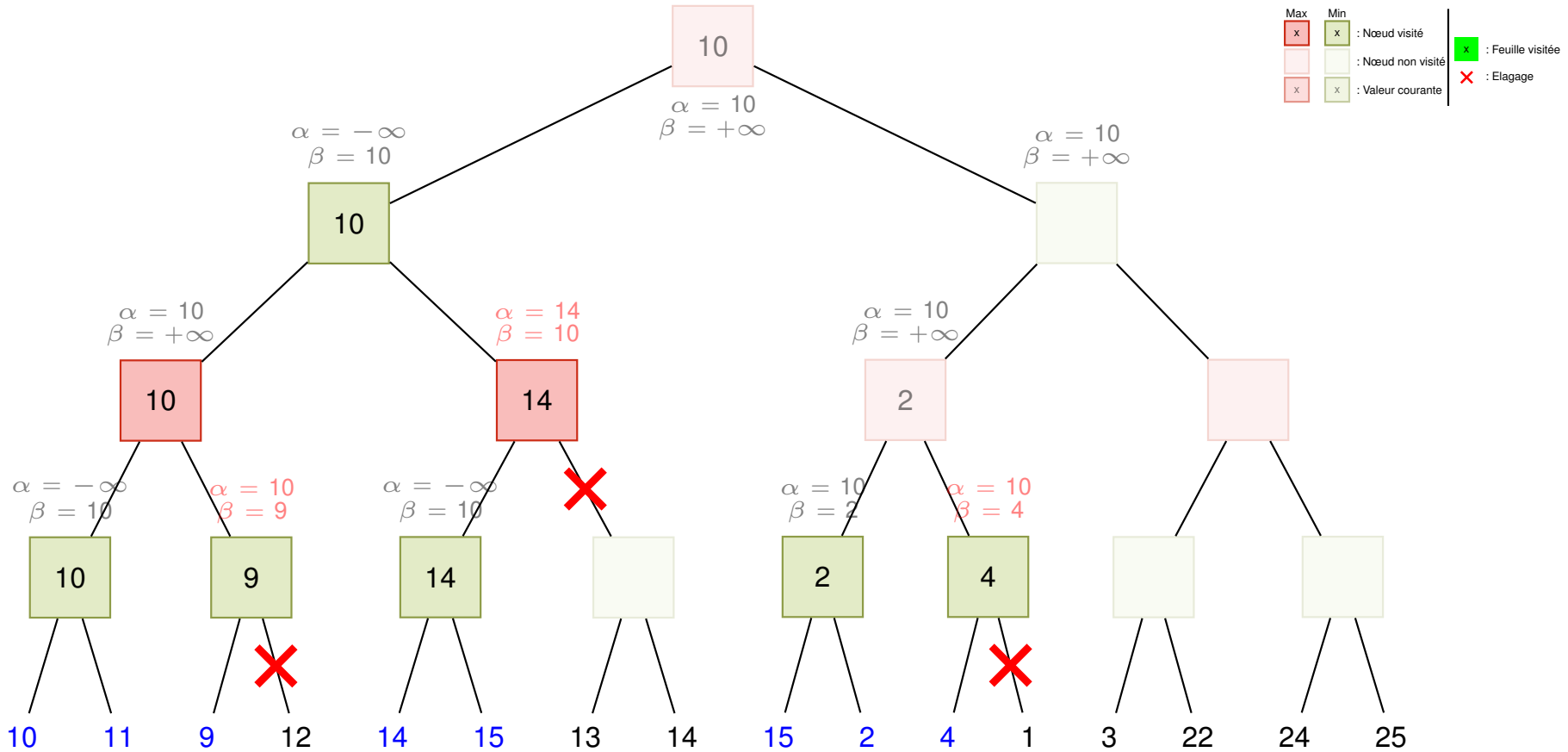
Exemples

Exemple 1



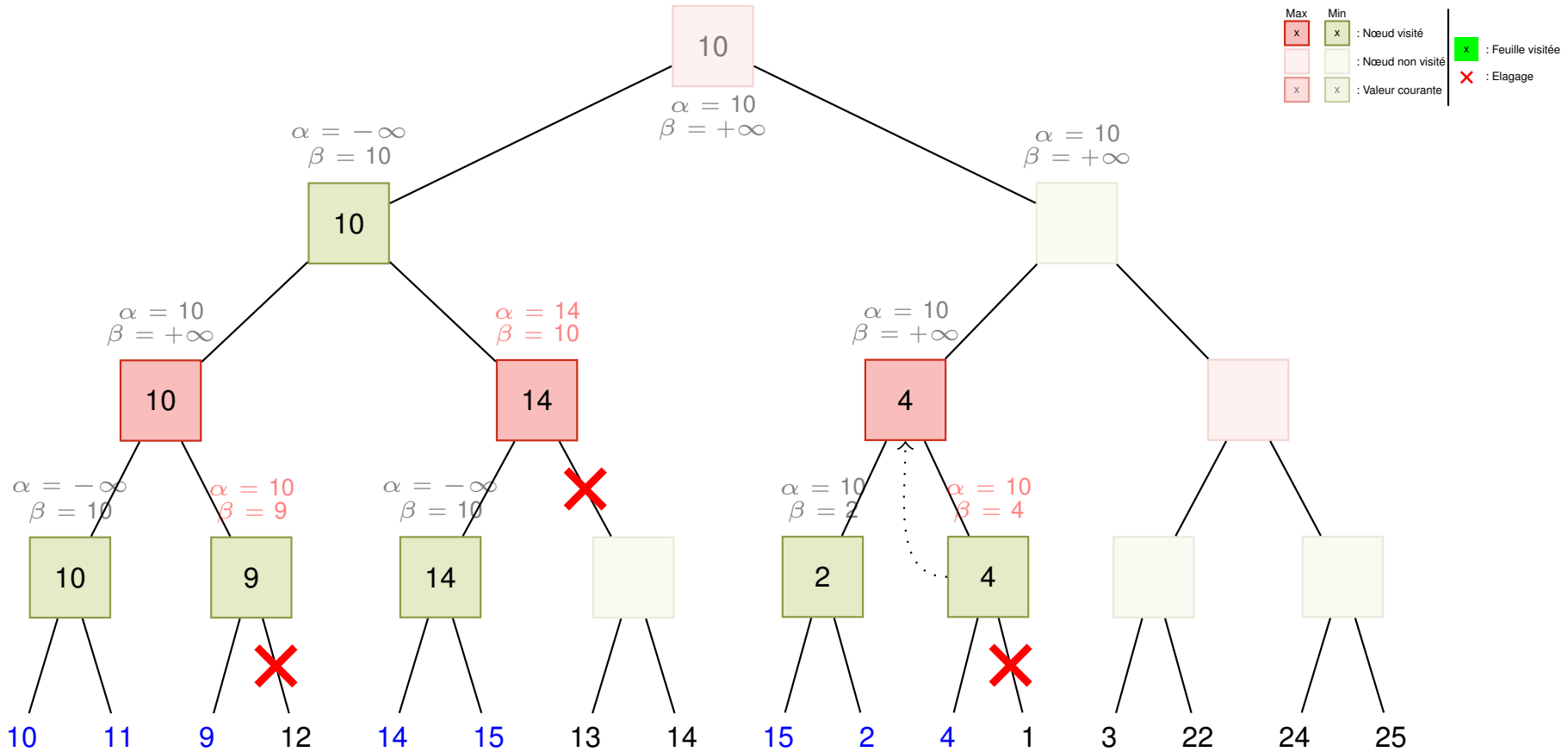
Exemples

Exemple 1



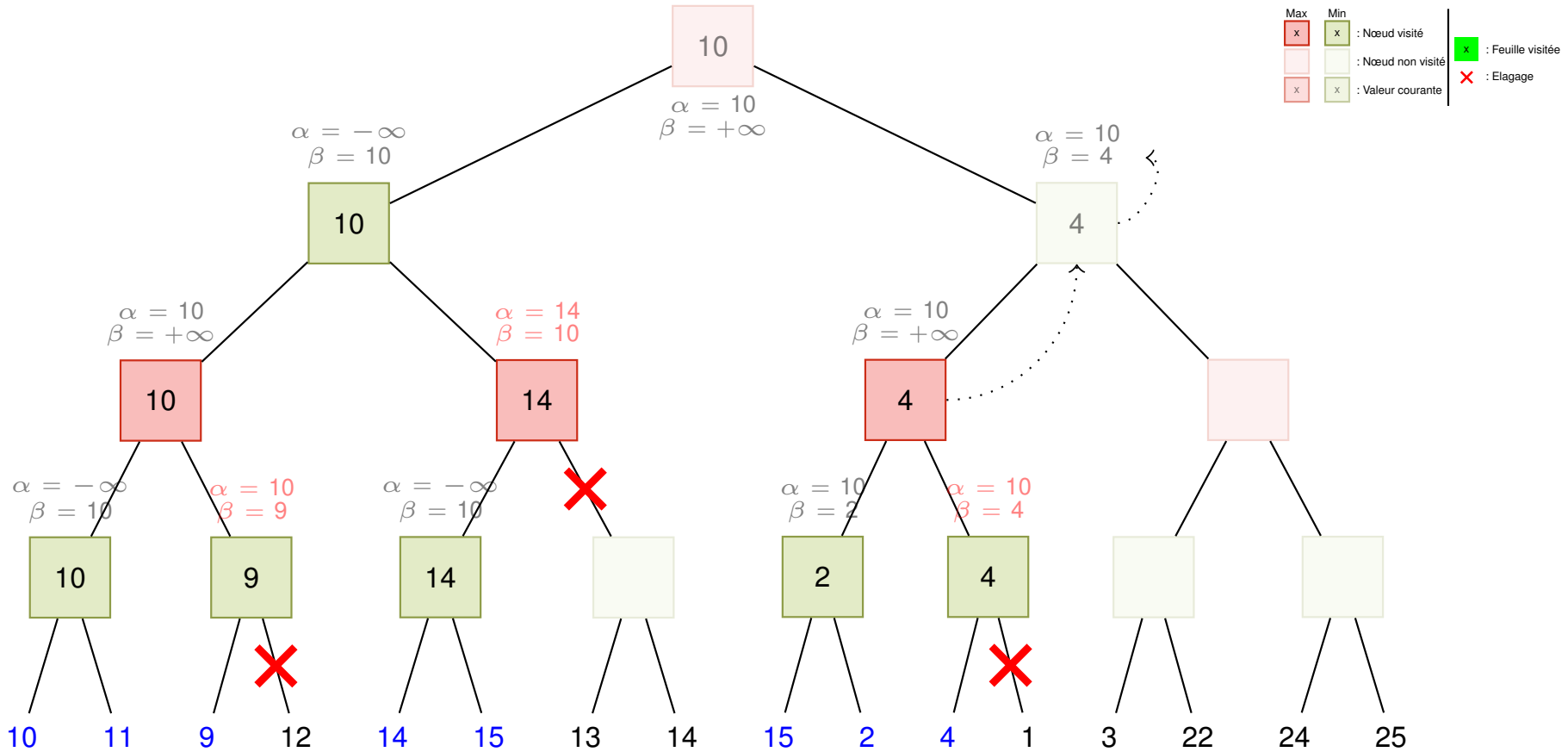
Exemples

Exemple 1



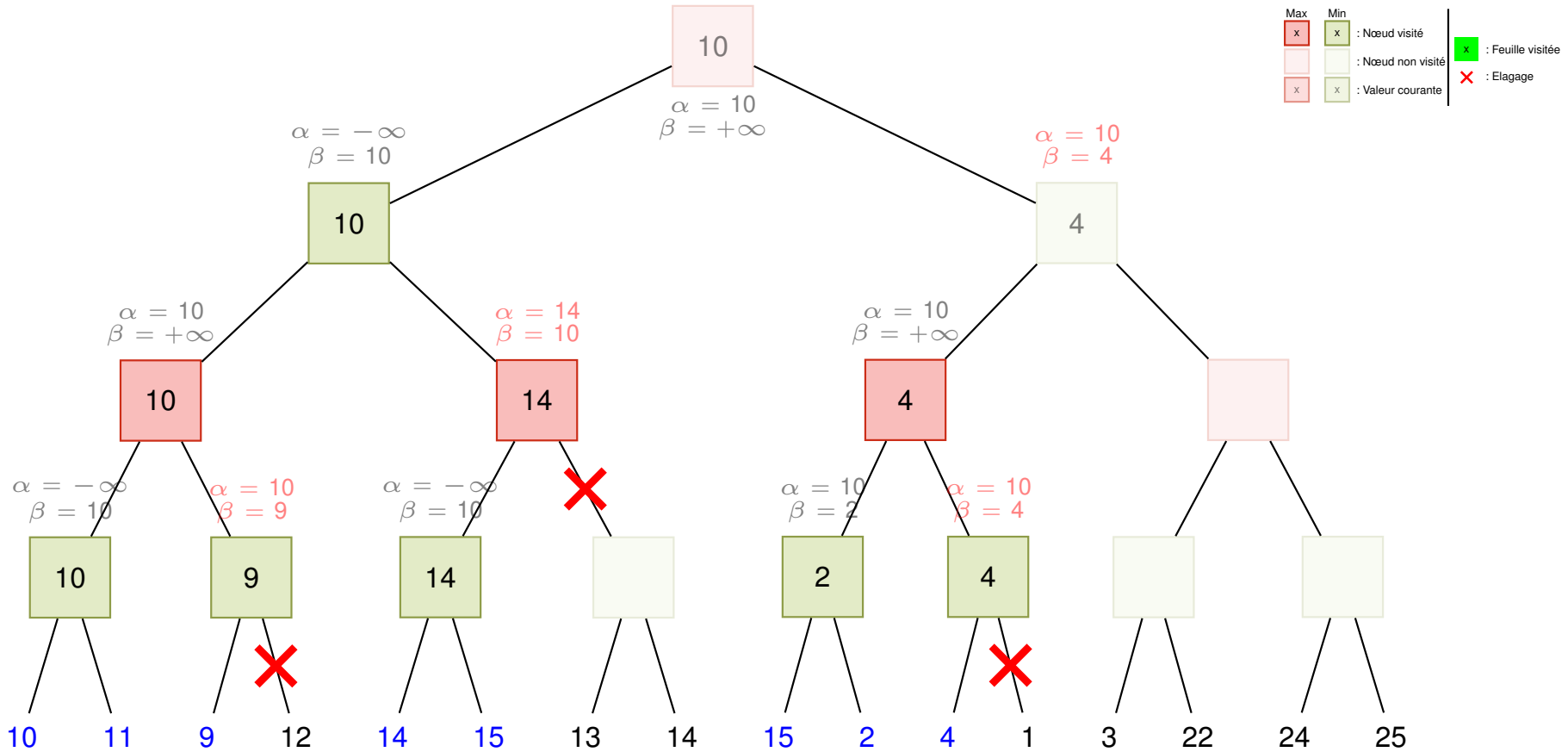
Exemples

Exemple 1



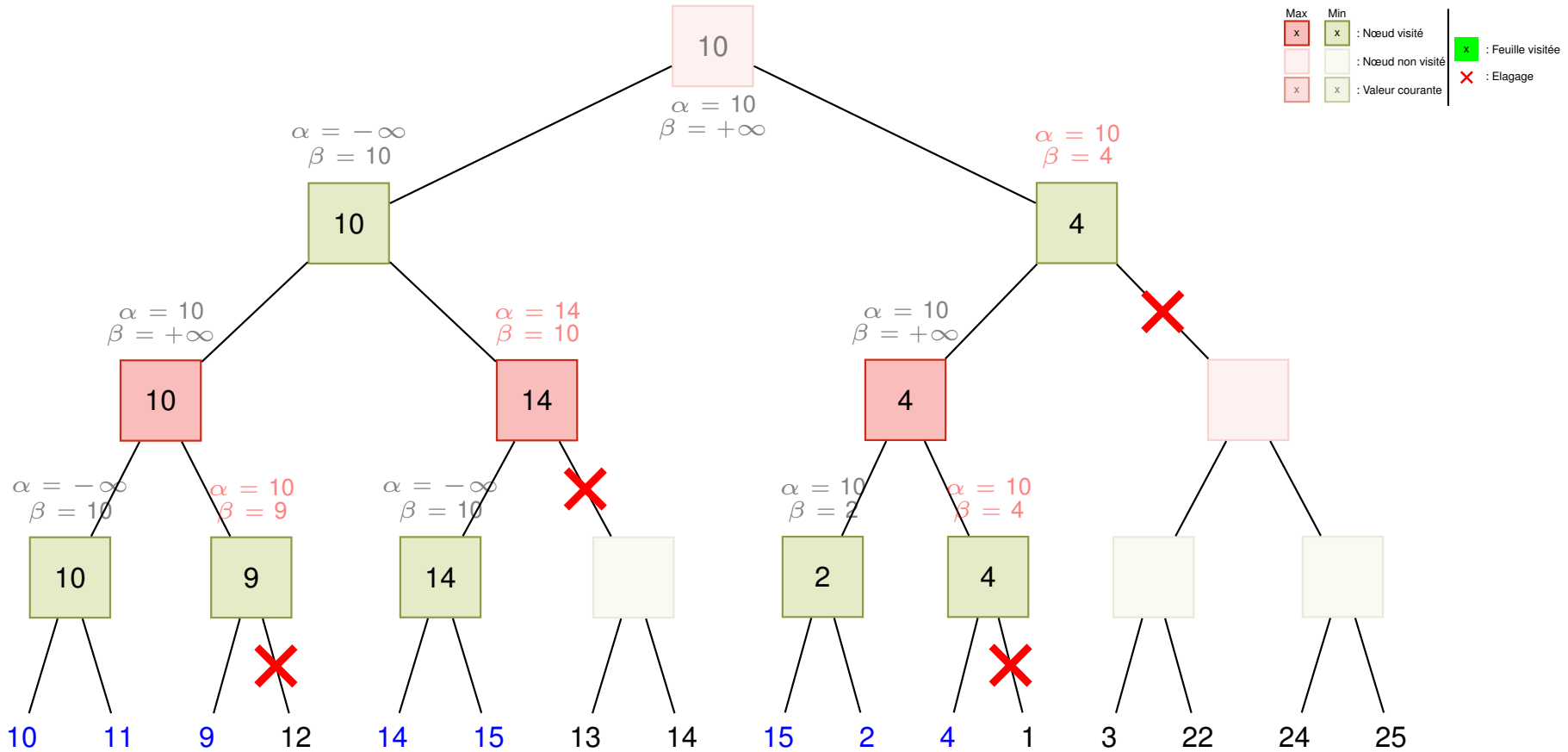
Exemples

Exemple 1



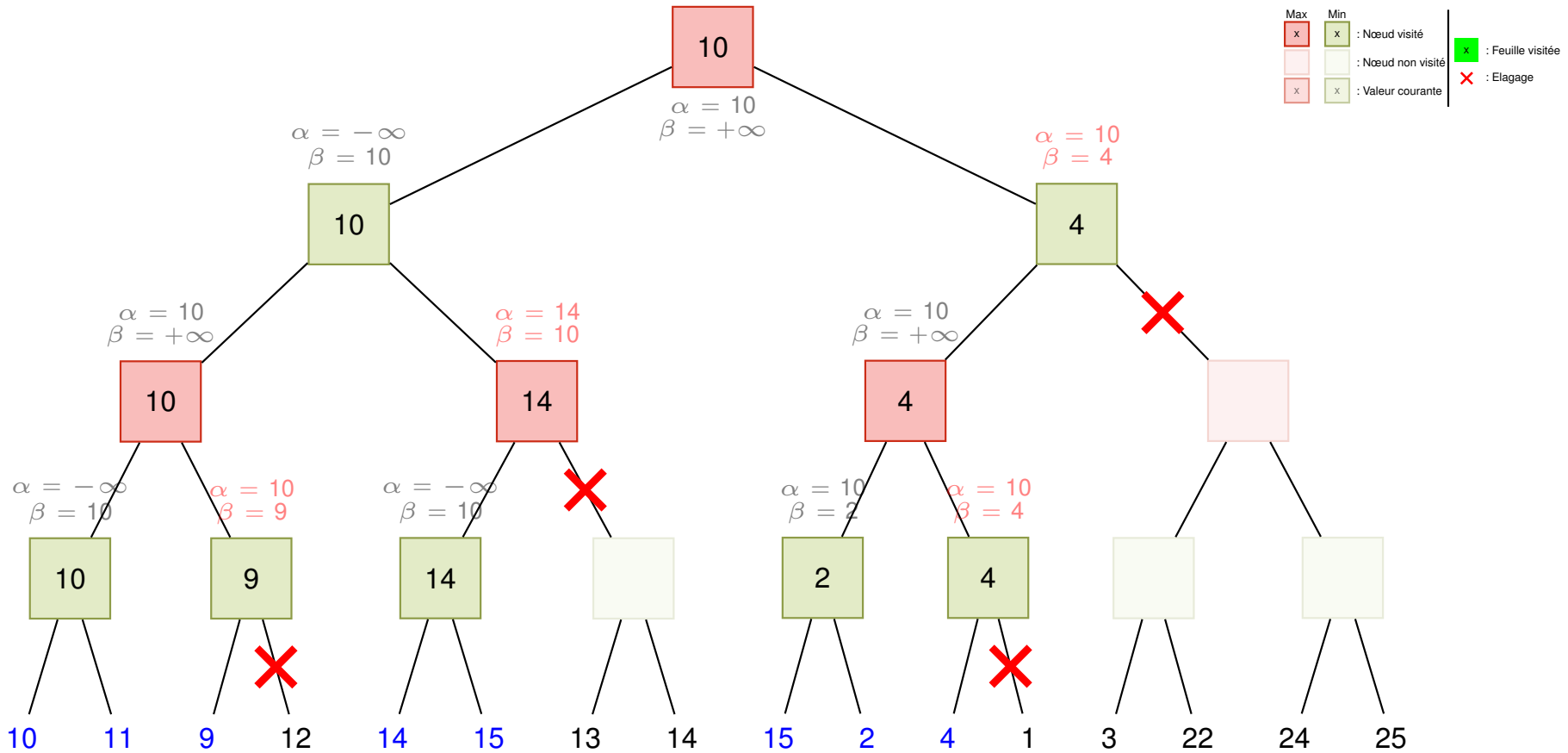
Exemples

Exemple 1



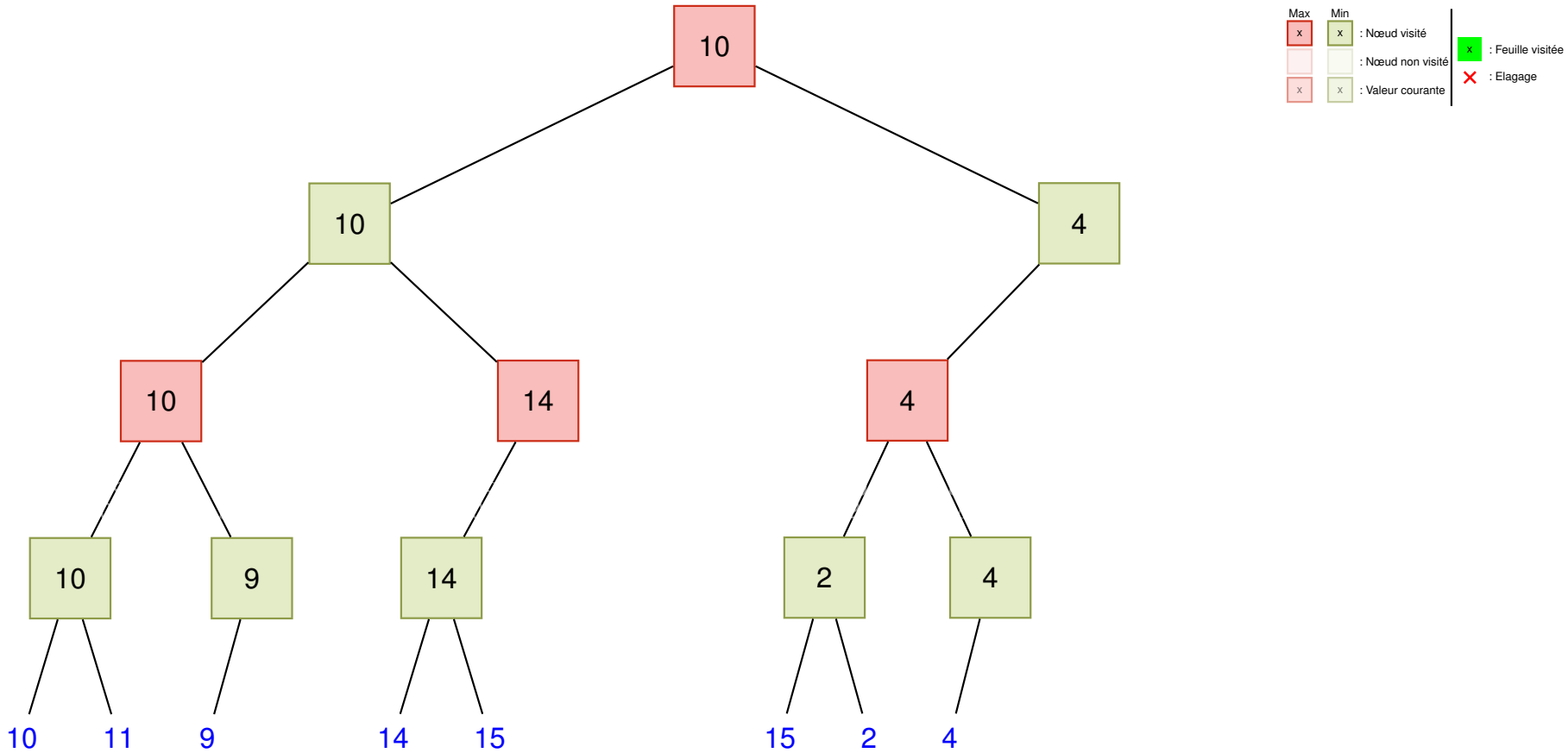
Exemples

Exemple 1



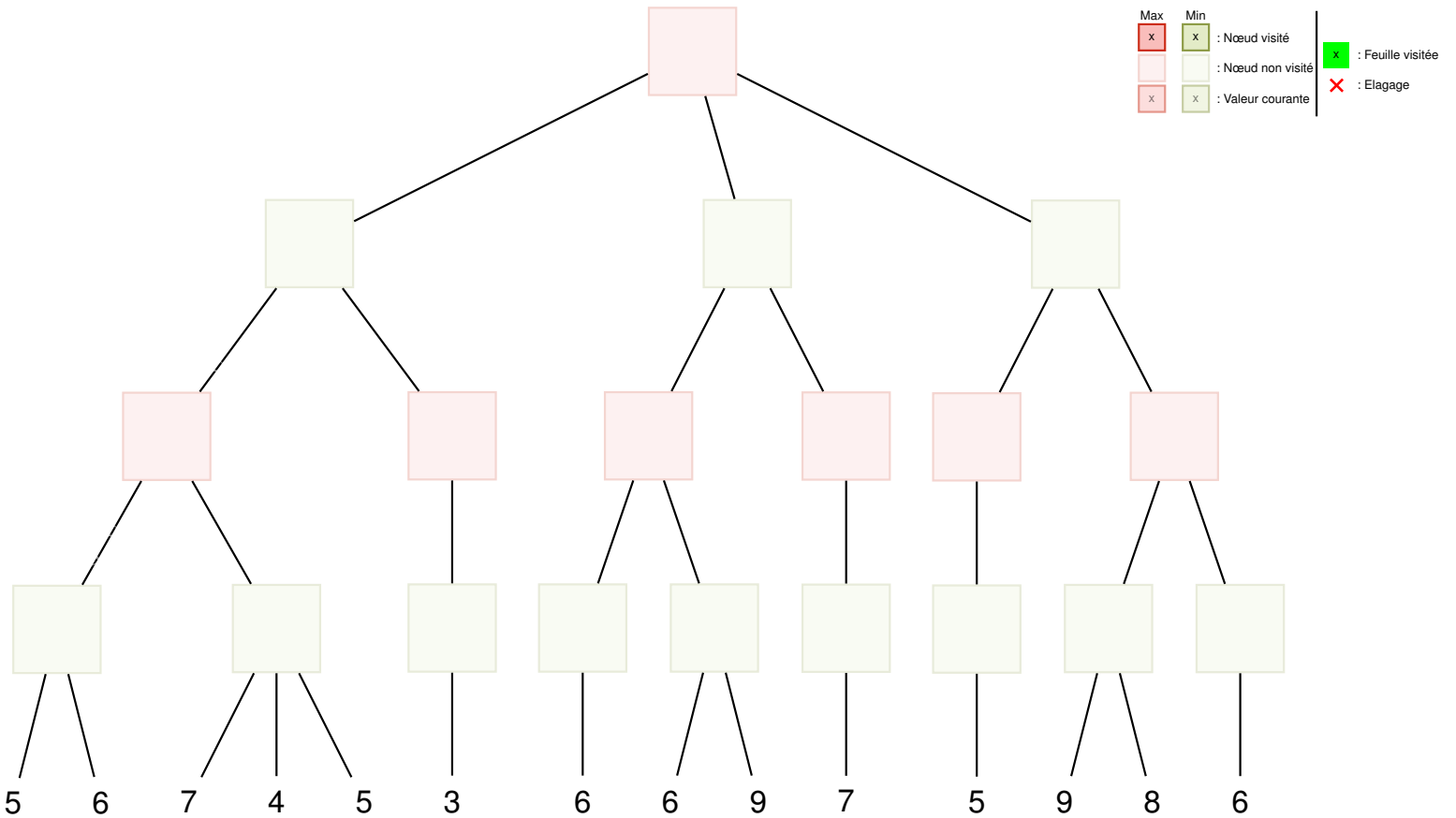
Exemples

Exemple 1



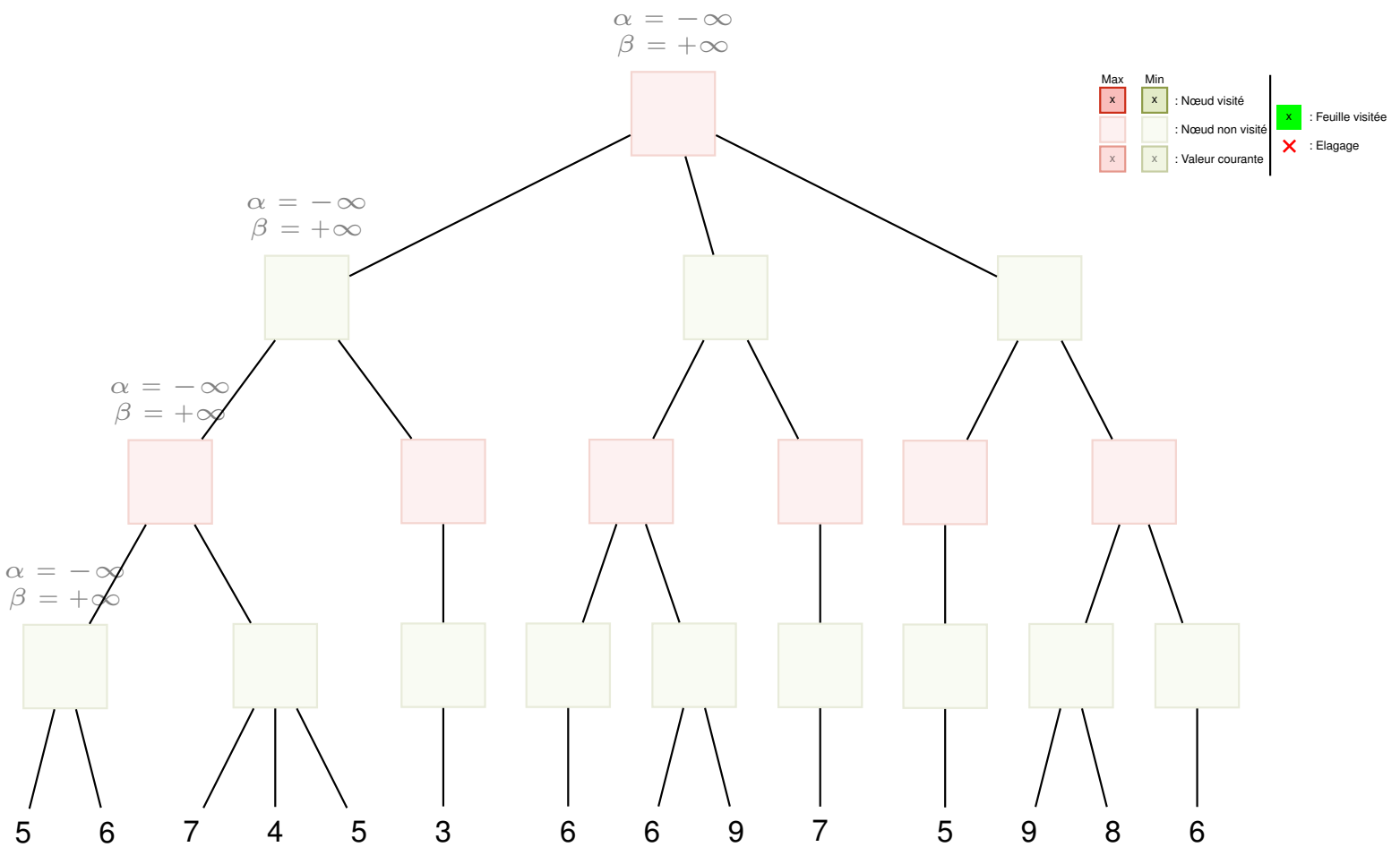
Exemples

Exemple 2



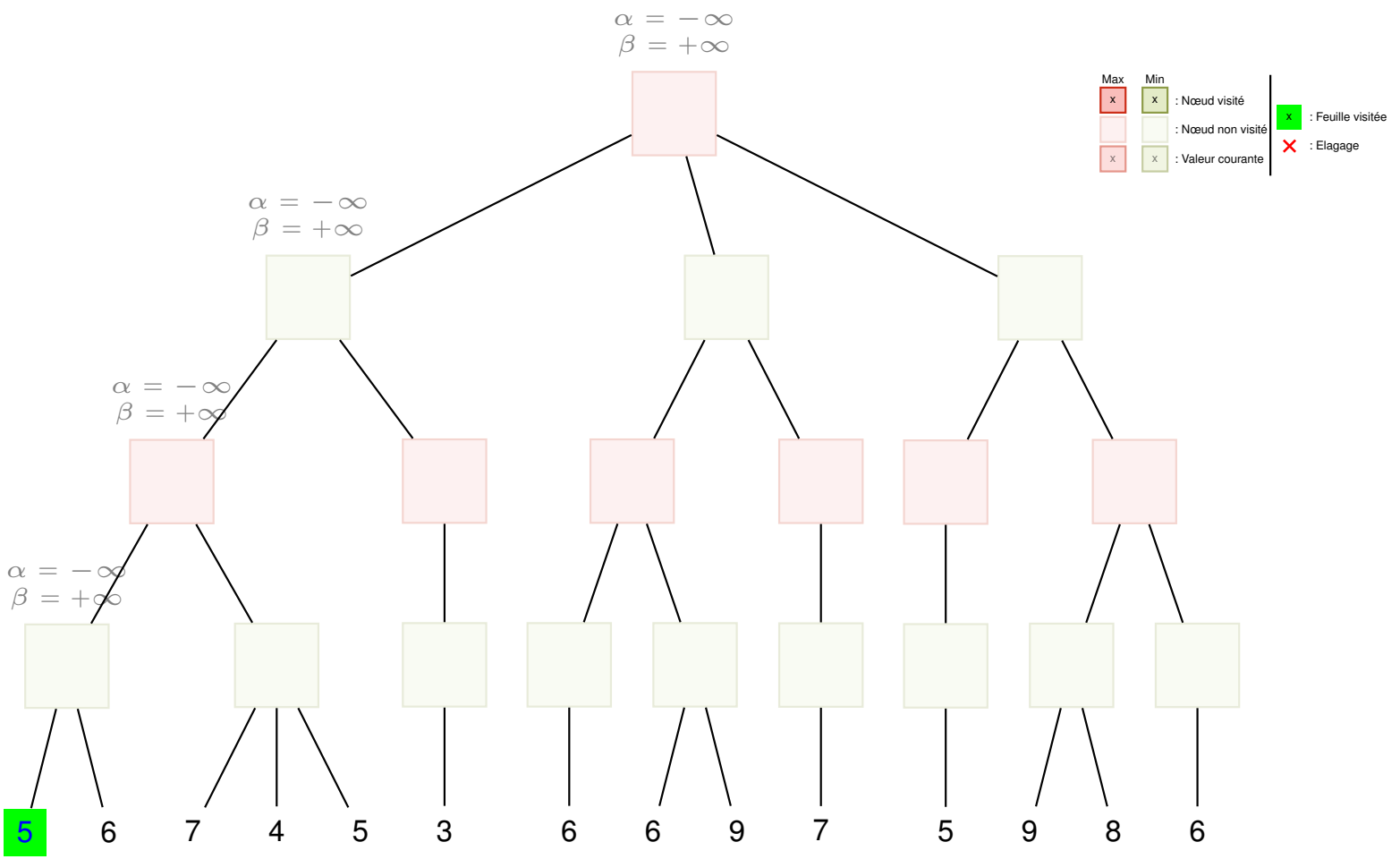
Exemples

Exemple 2



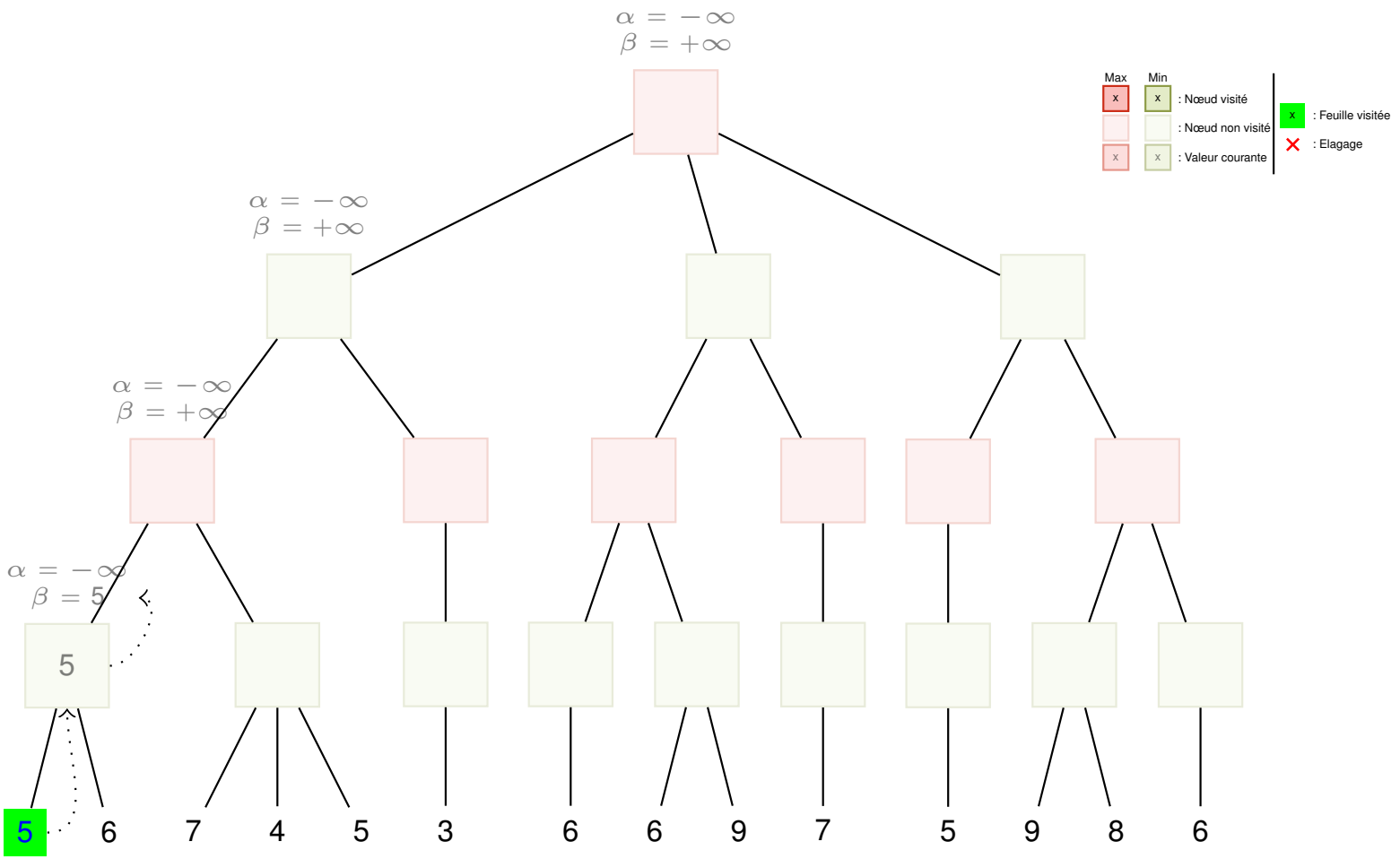
Exemples

Exemple 2



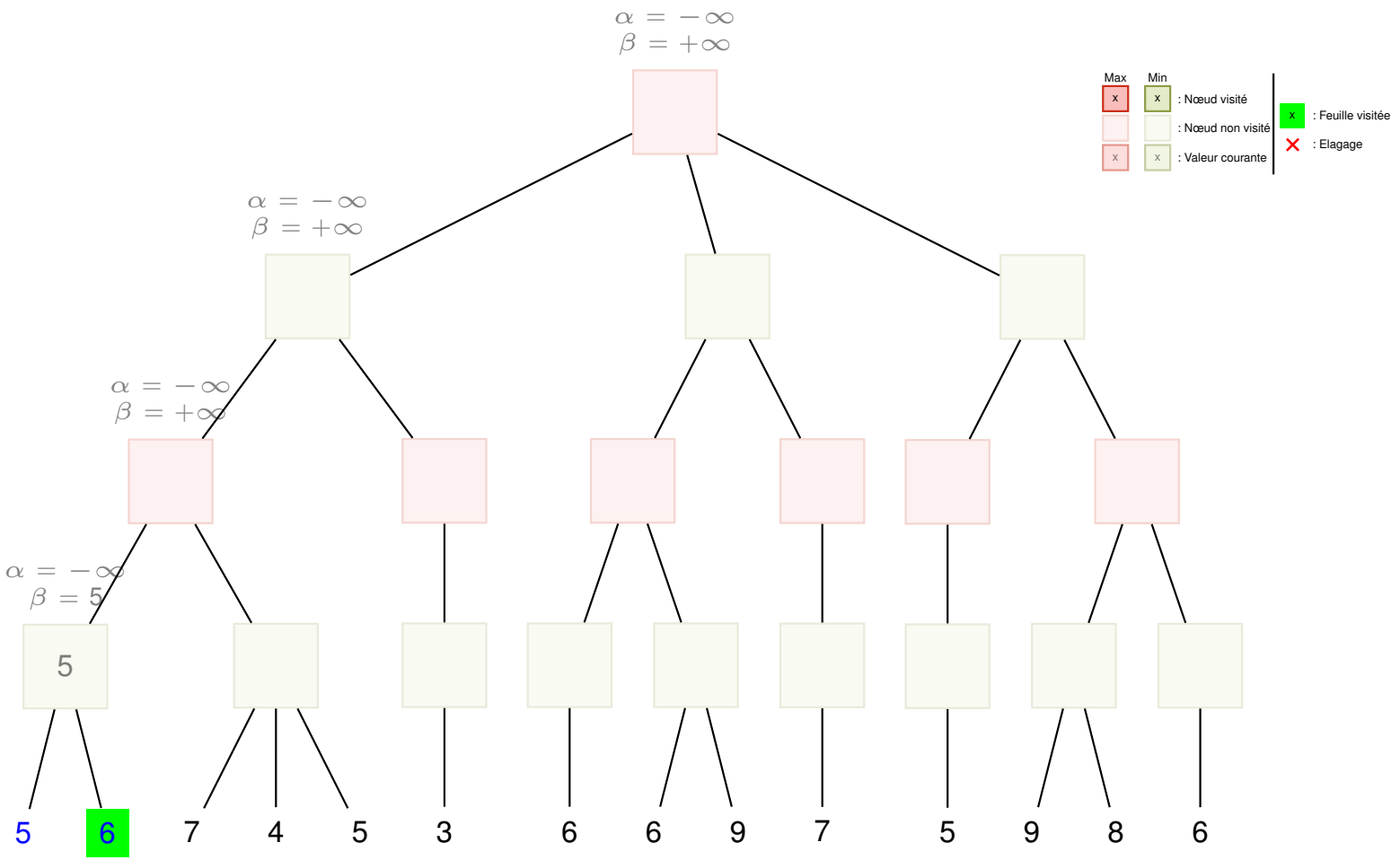
Exemples

Exemple 2



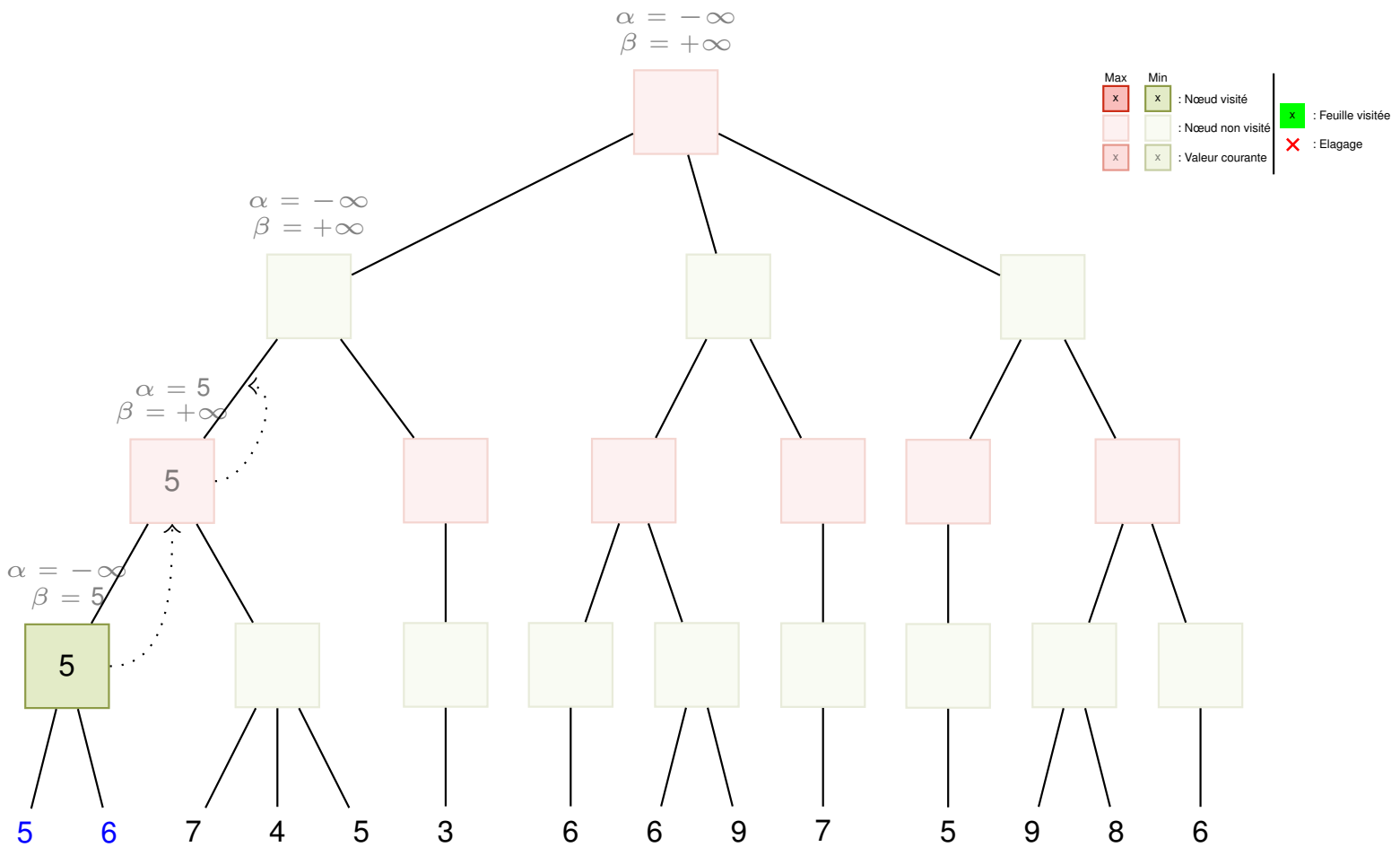
Exemples

Exemple 2



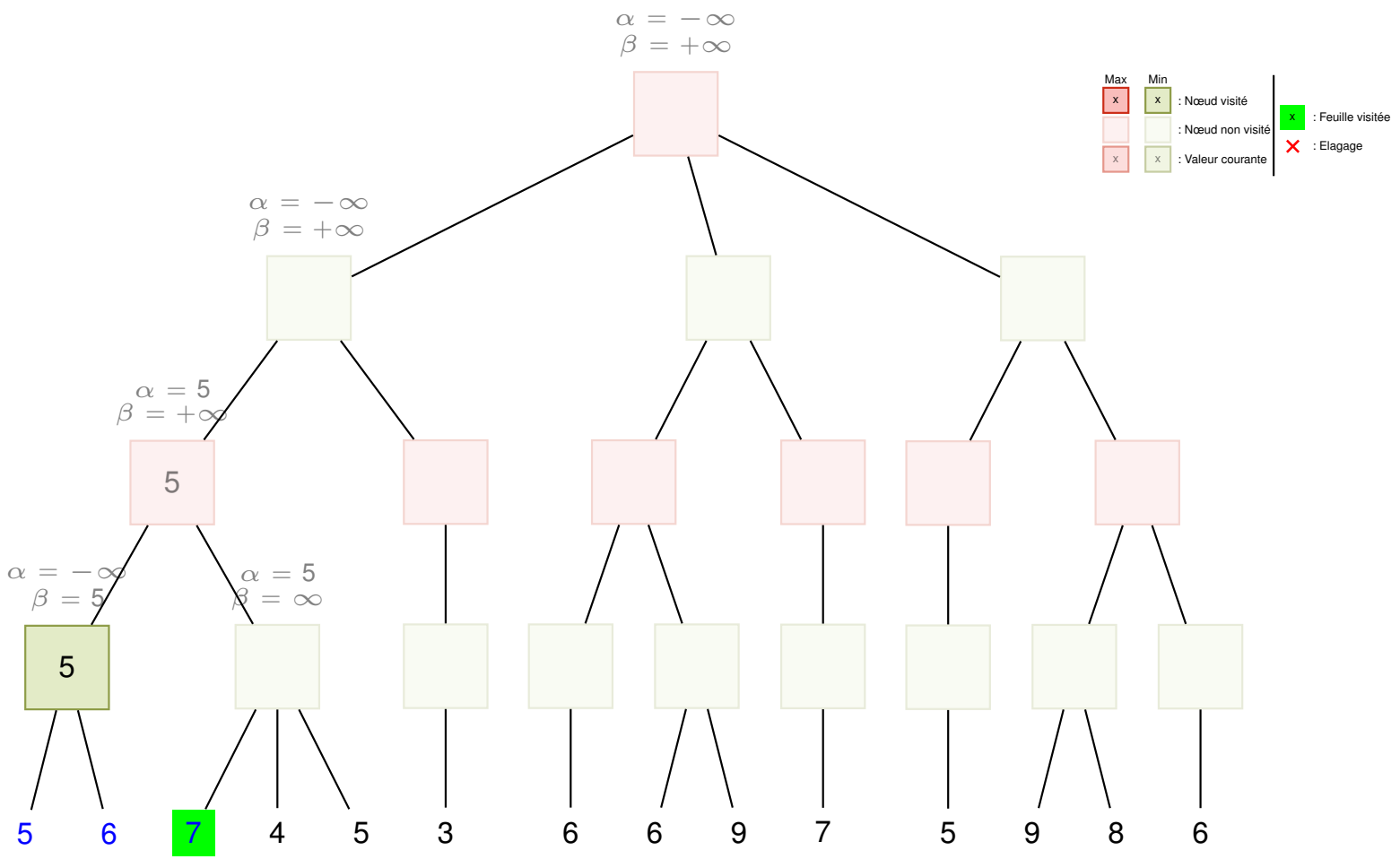
Exemples

Exemple 2



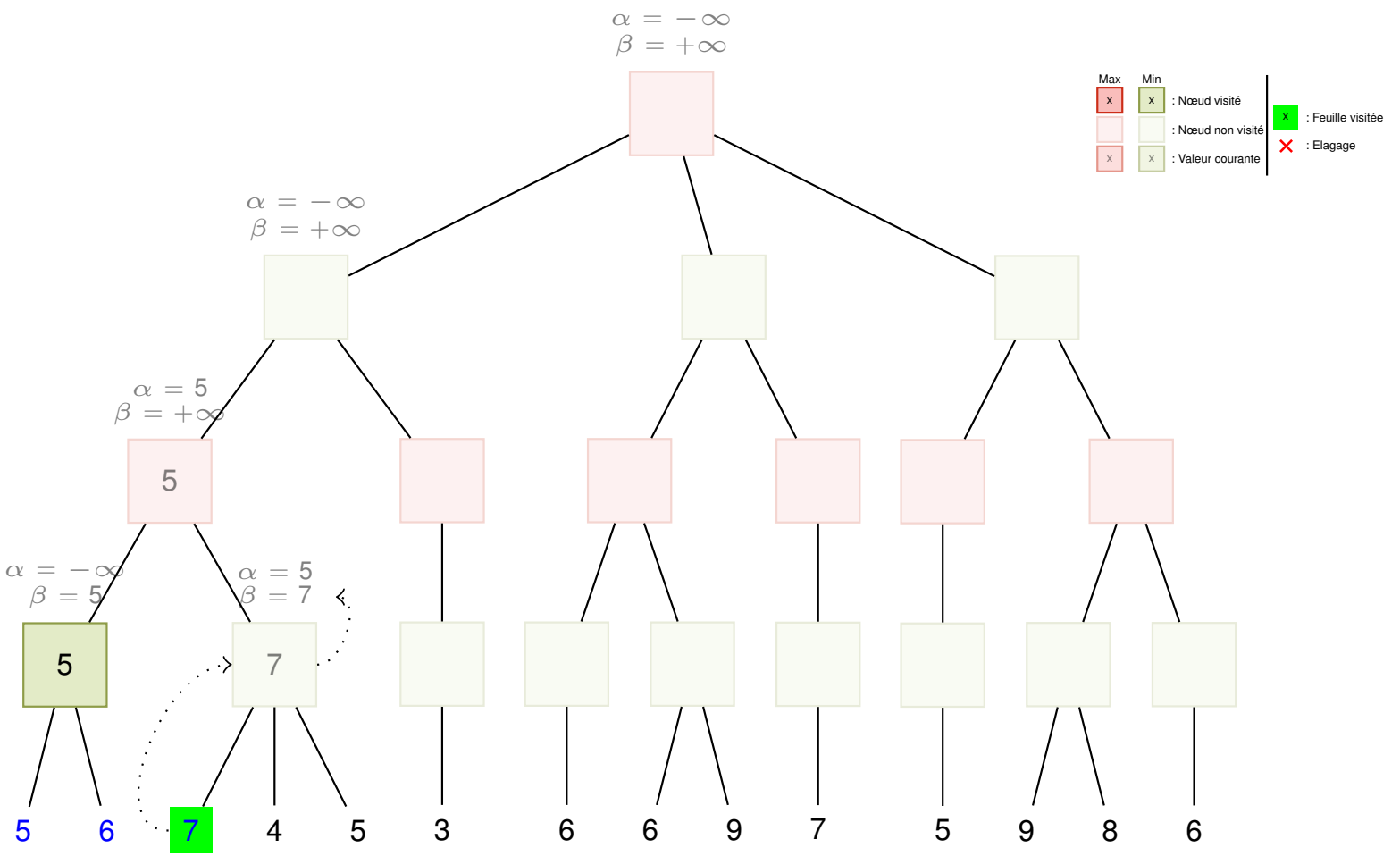
Exemples

Exemple 2



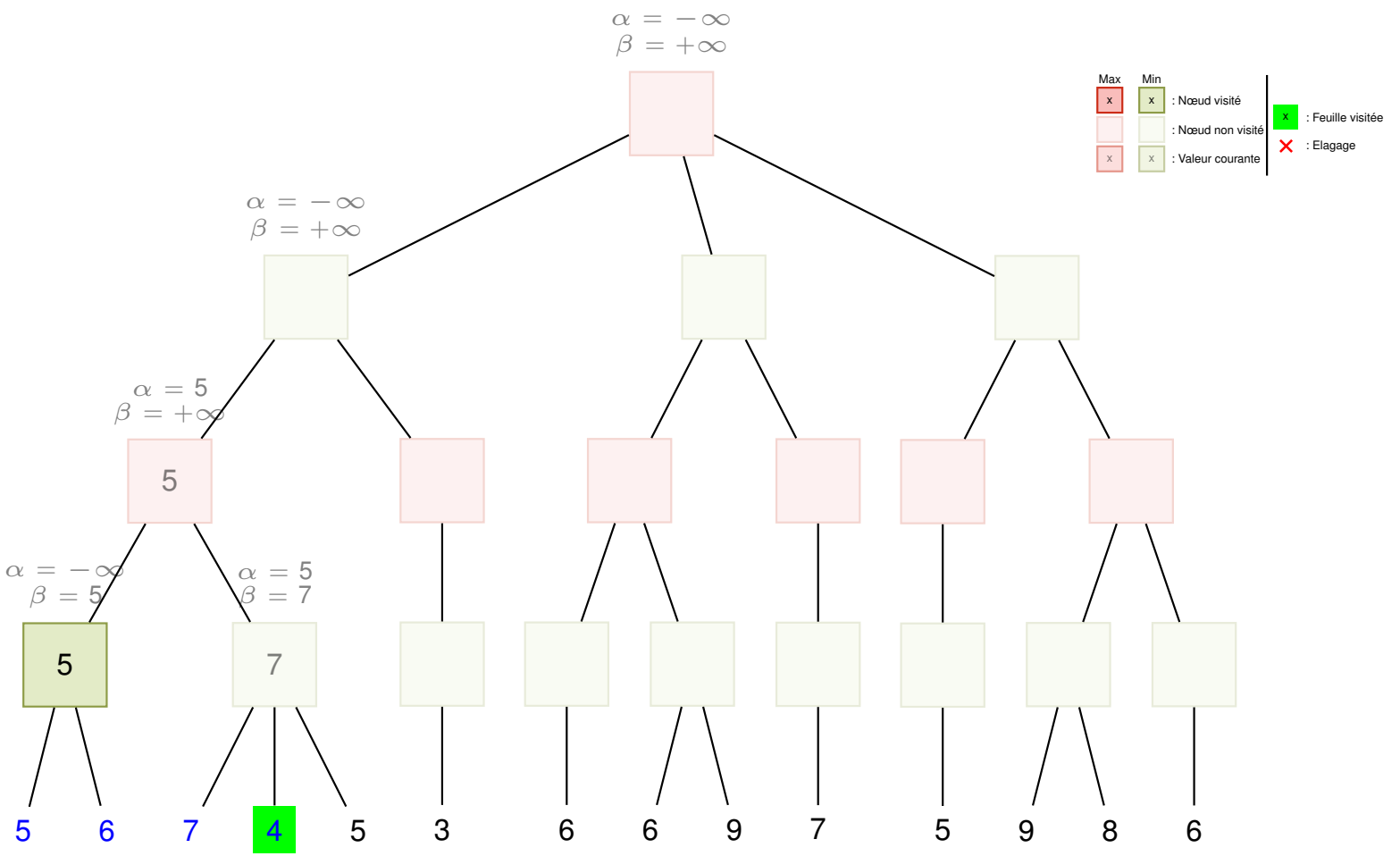
Exemples

Exemple 2



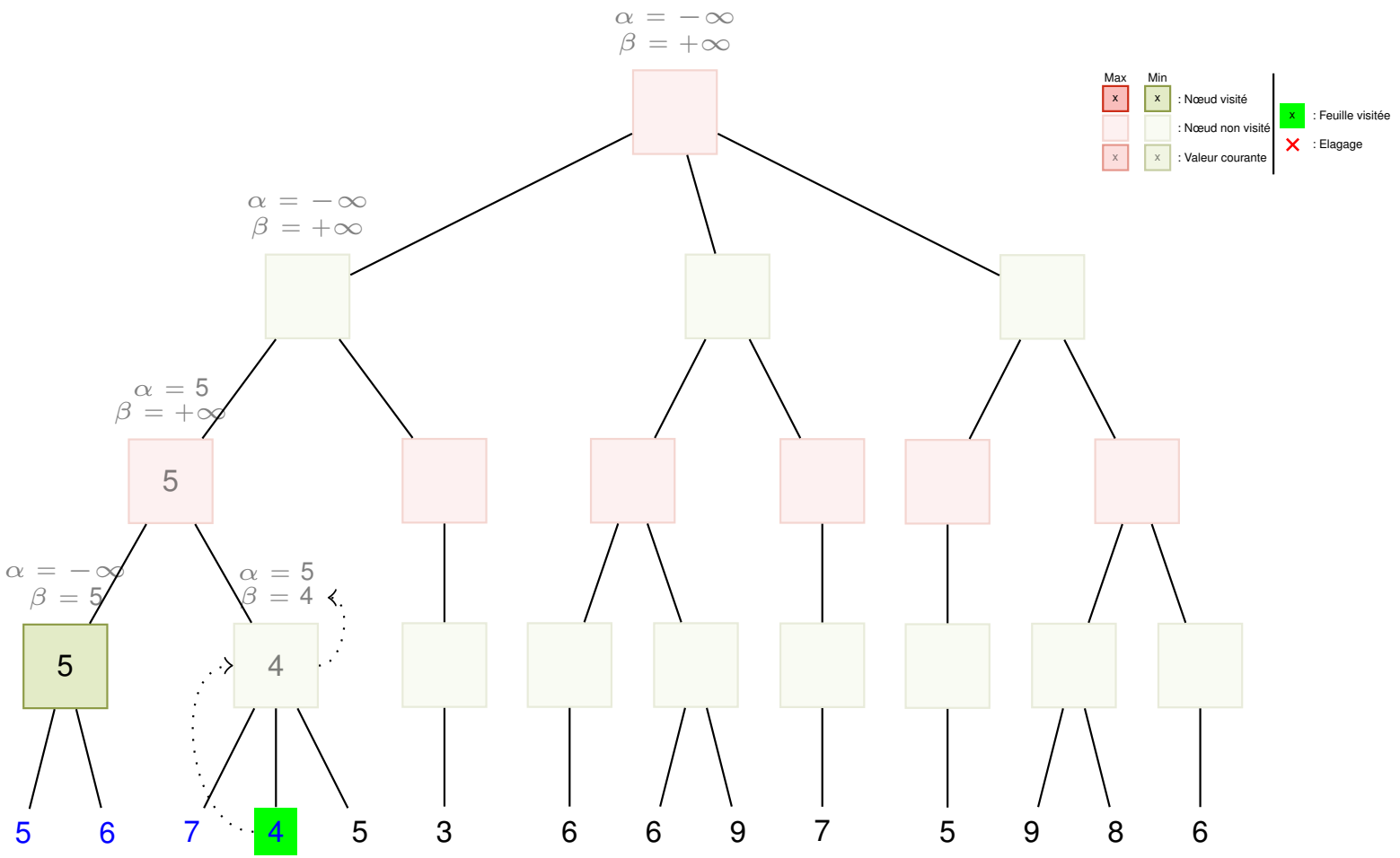
Exemples

Exemple 2



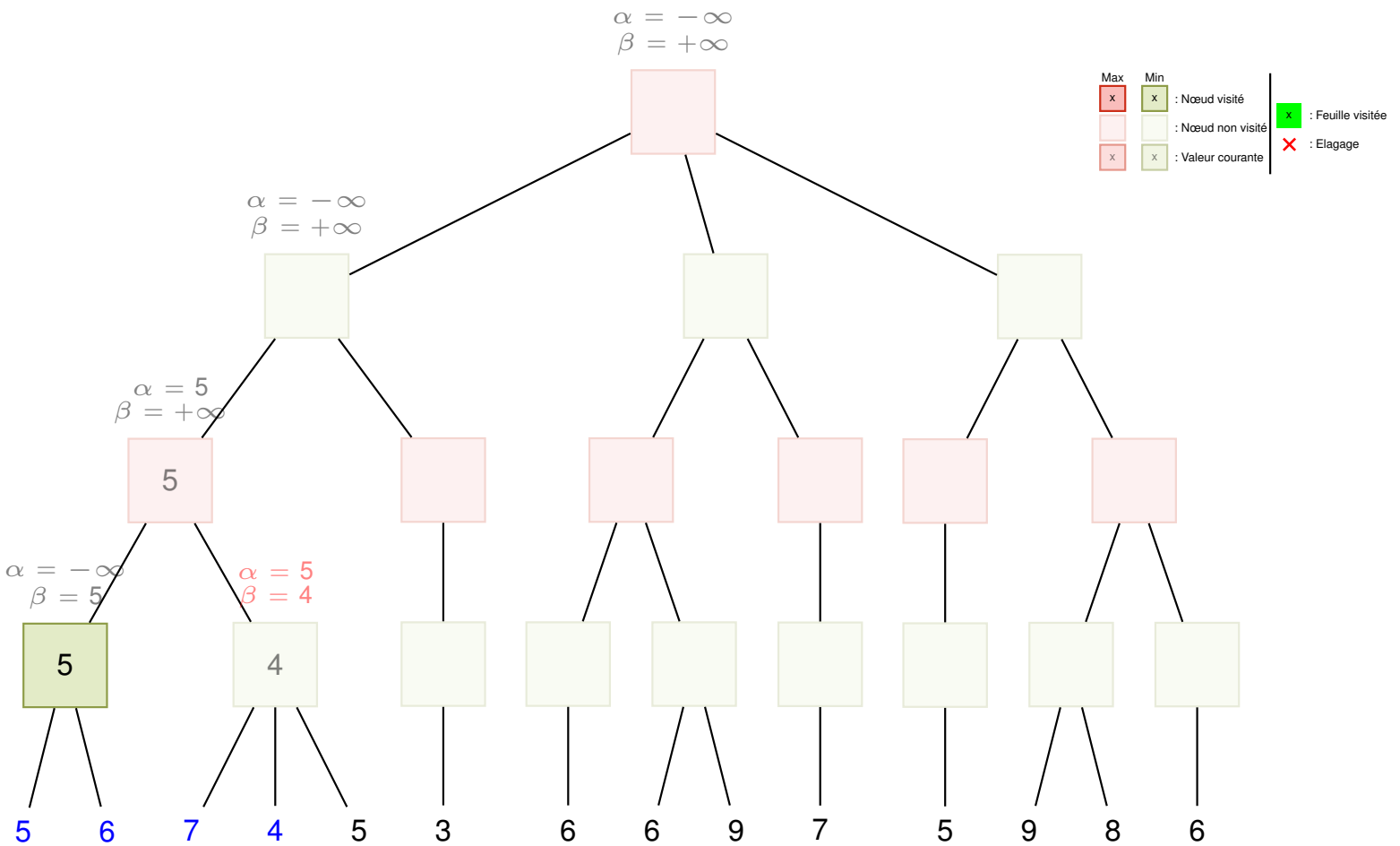
Exemples

Exemple 2



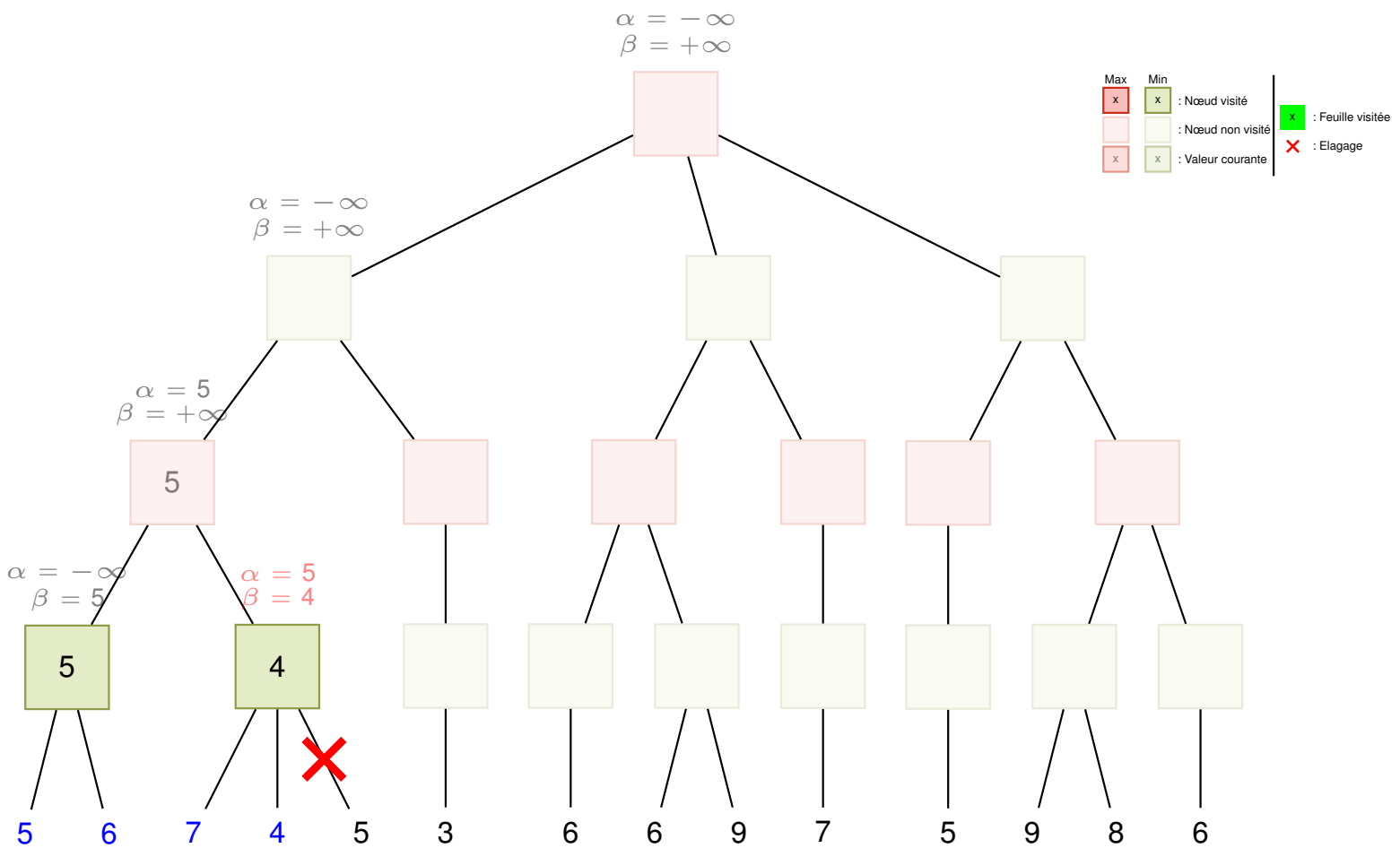
Exemples

Exemple 2



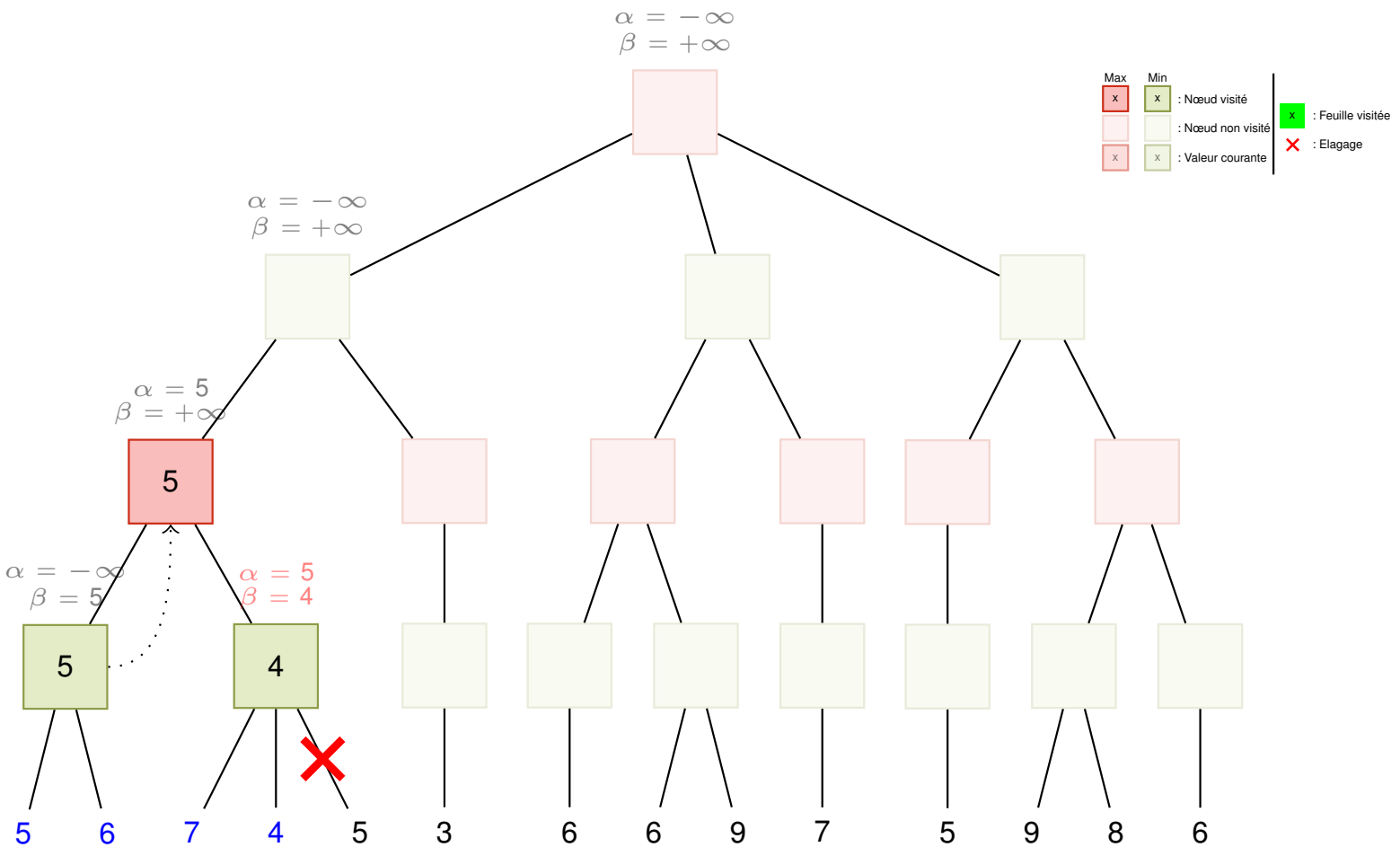
Exemples

Exemple 2



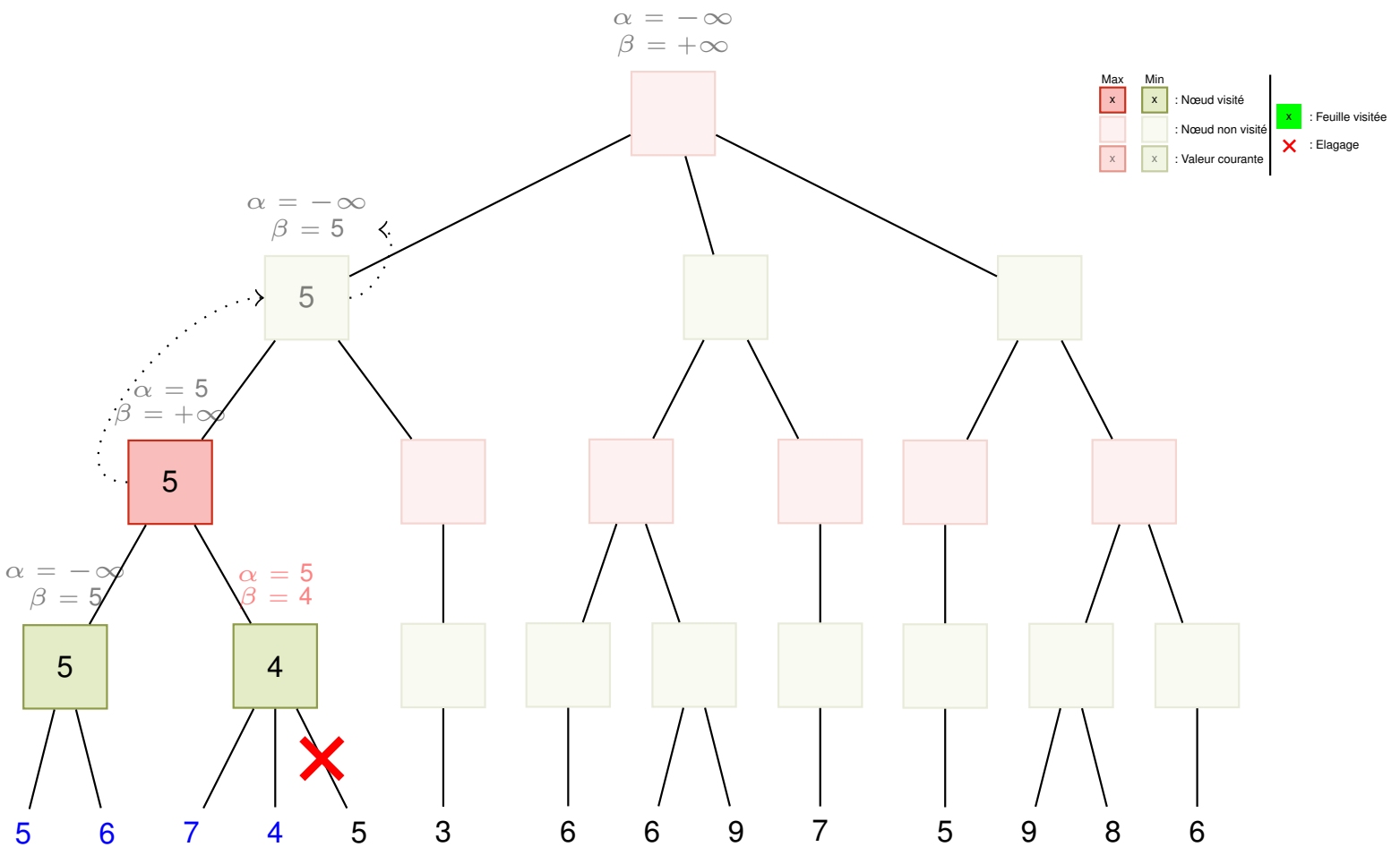
Exemples

Exemple 2



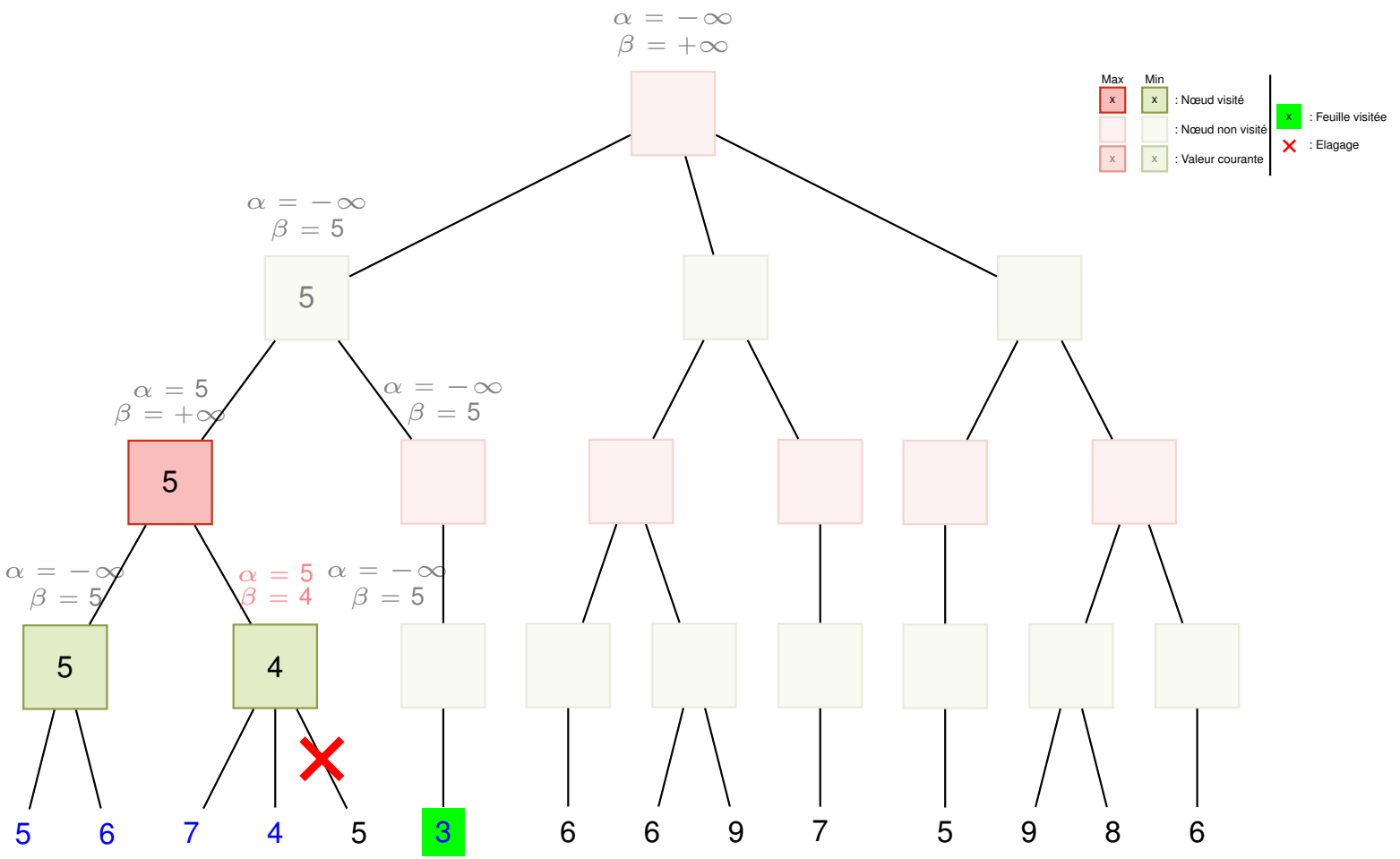
Exemples

Exemple 2



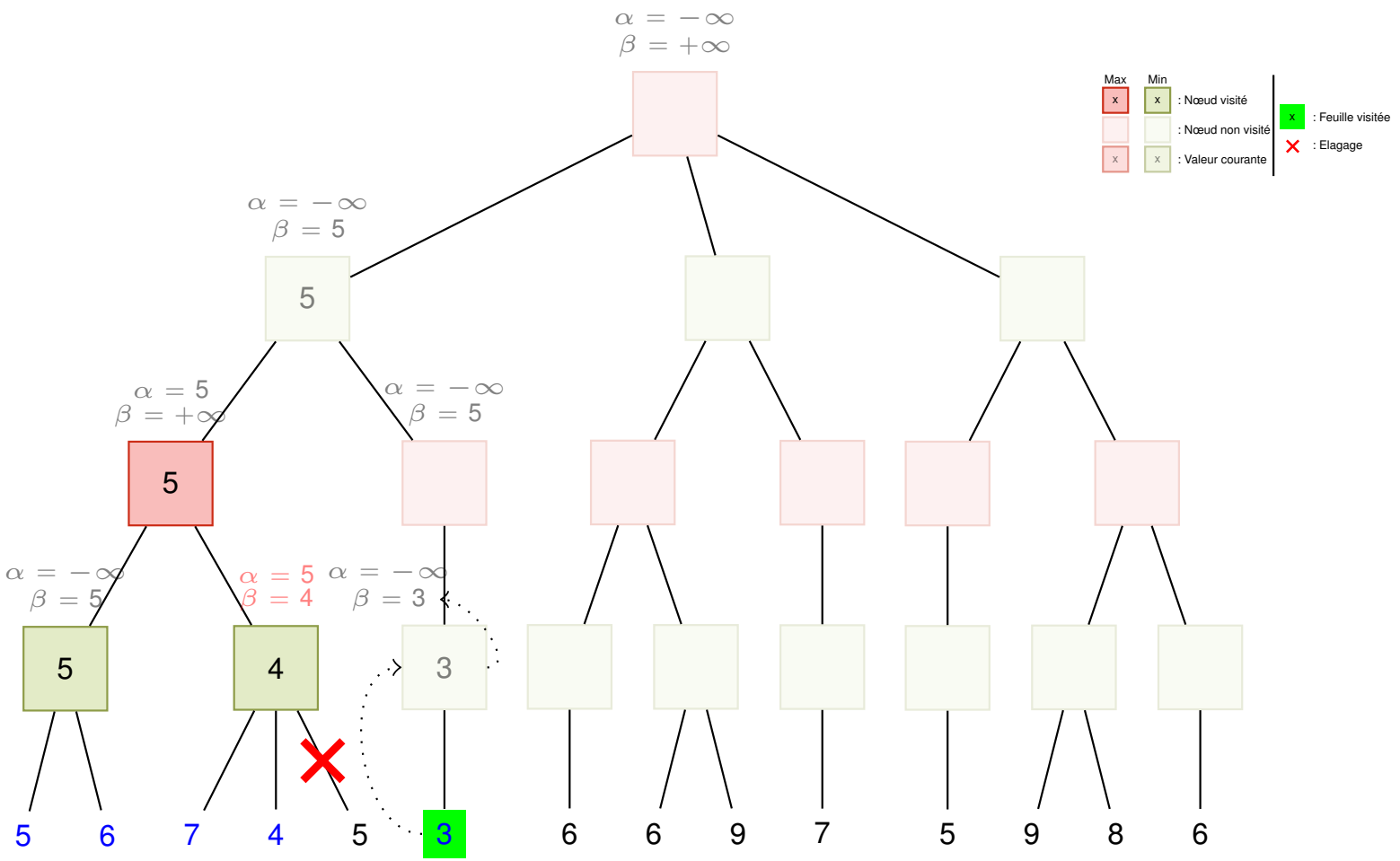
Exemples

Exemple 2



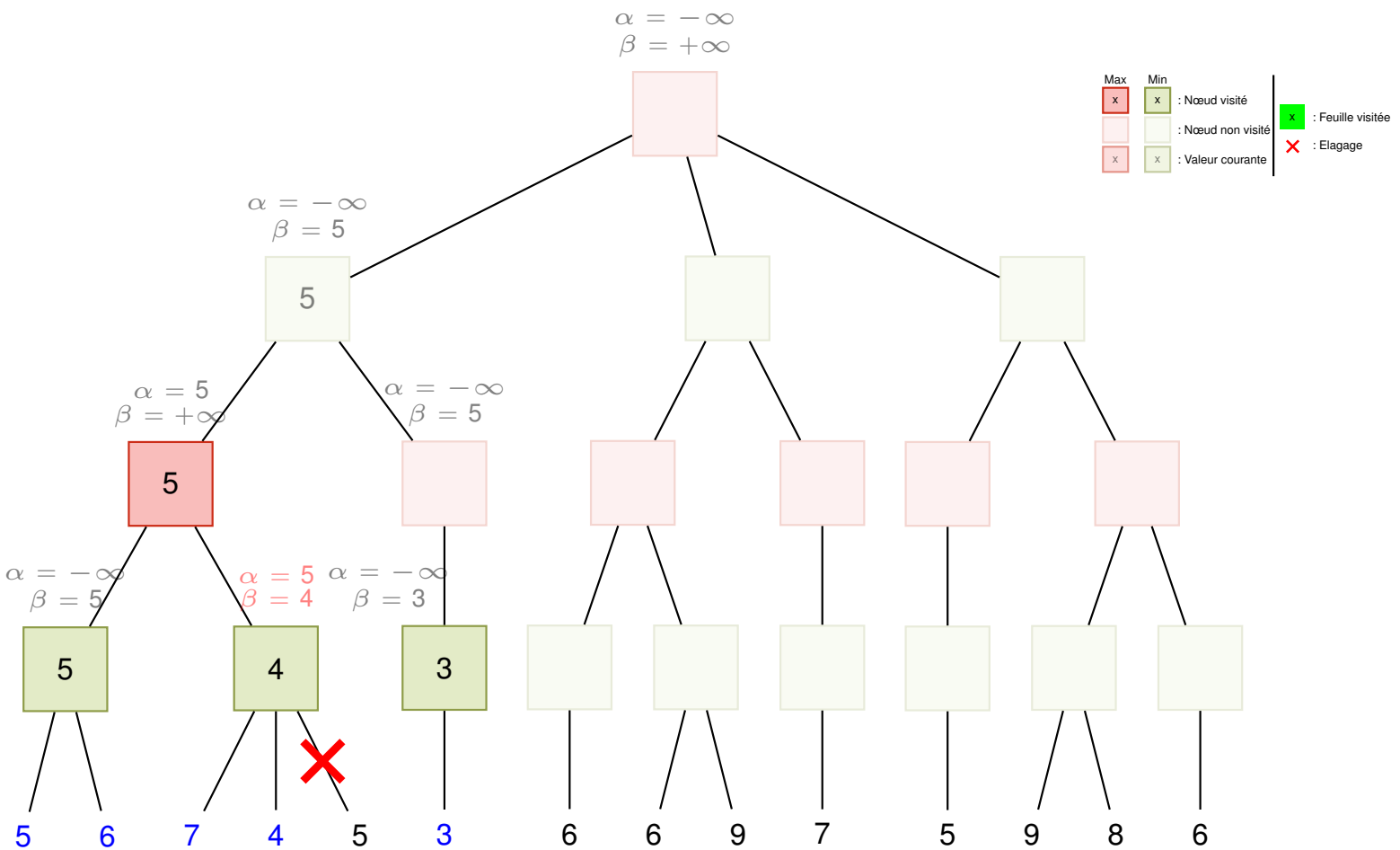
Exemples

Exemple 2



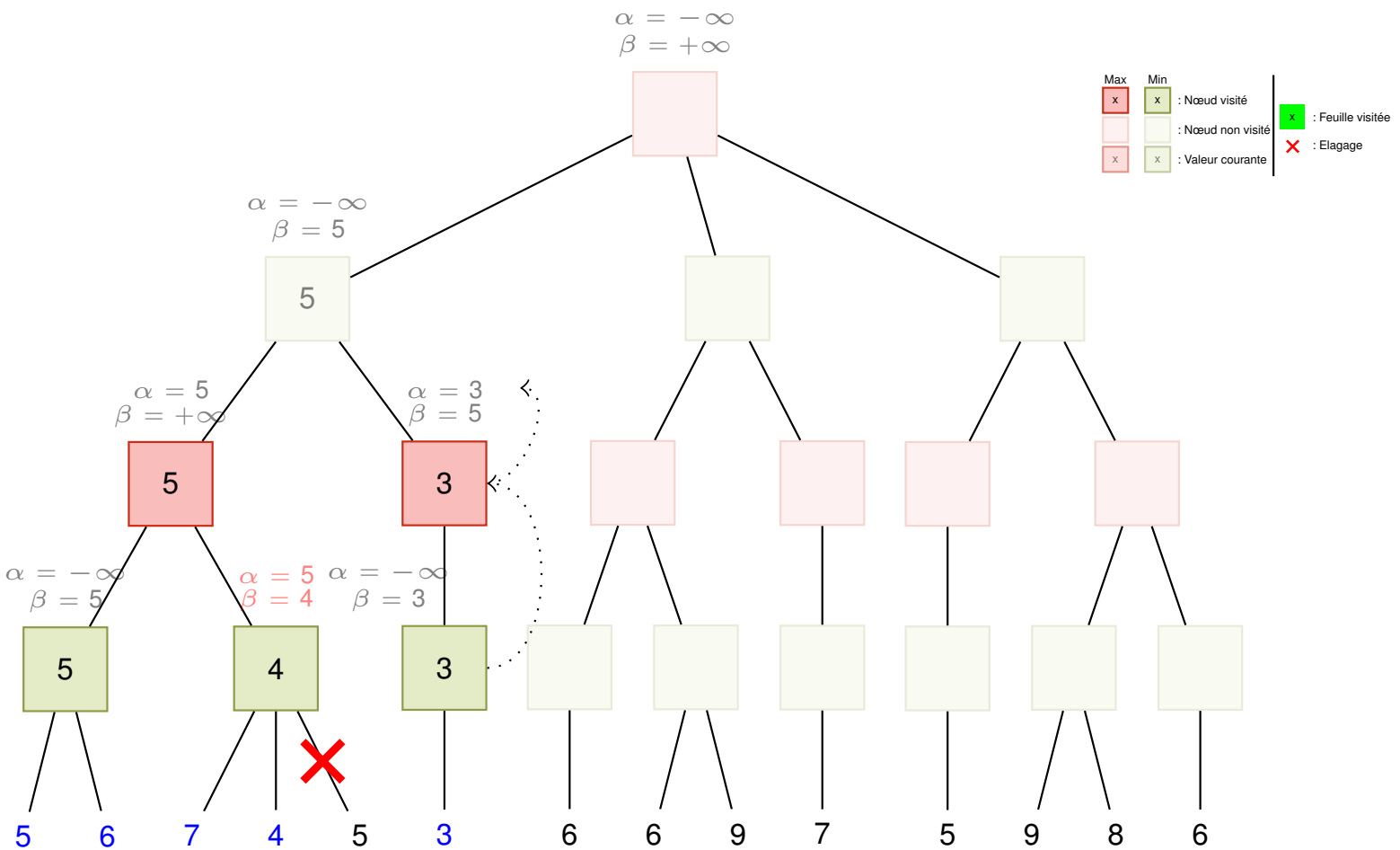
Exemples

Exemple 2



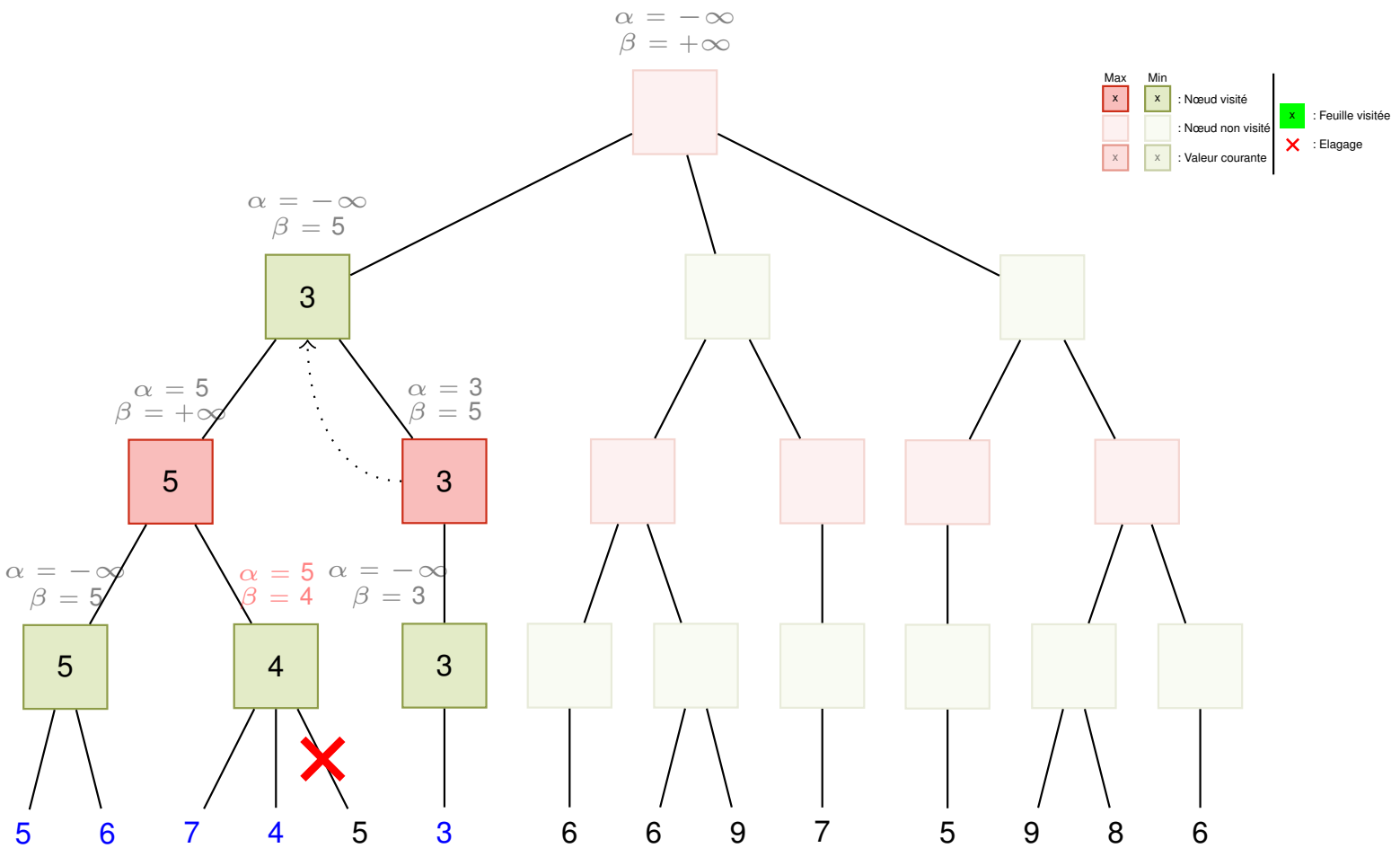
Exemples

Exemple 2



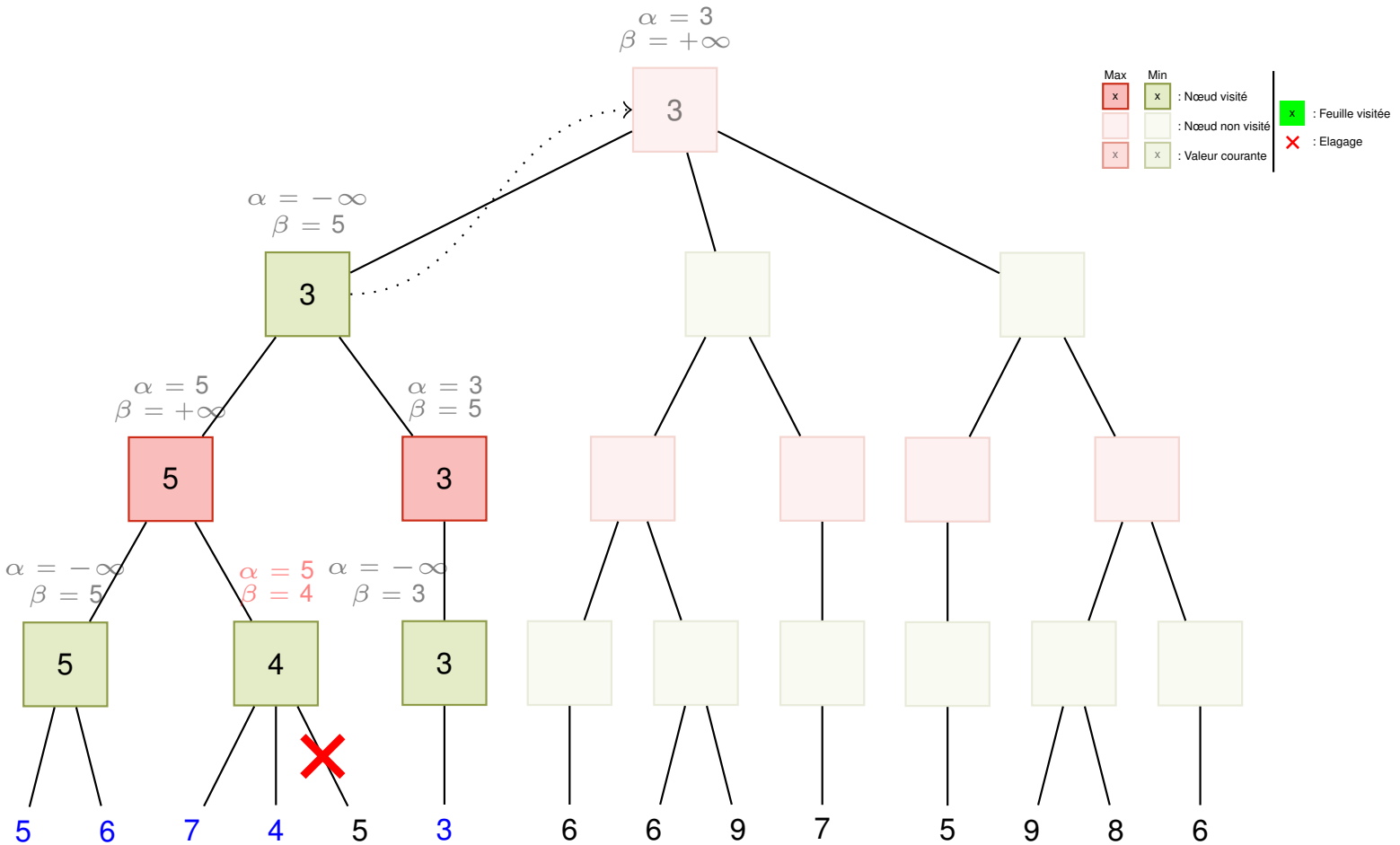
Exemples

Exemple 2



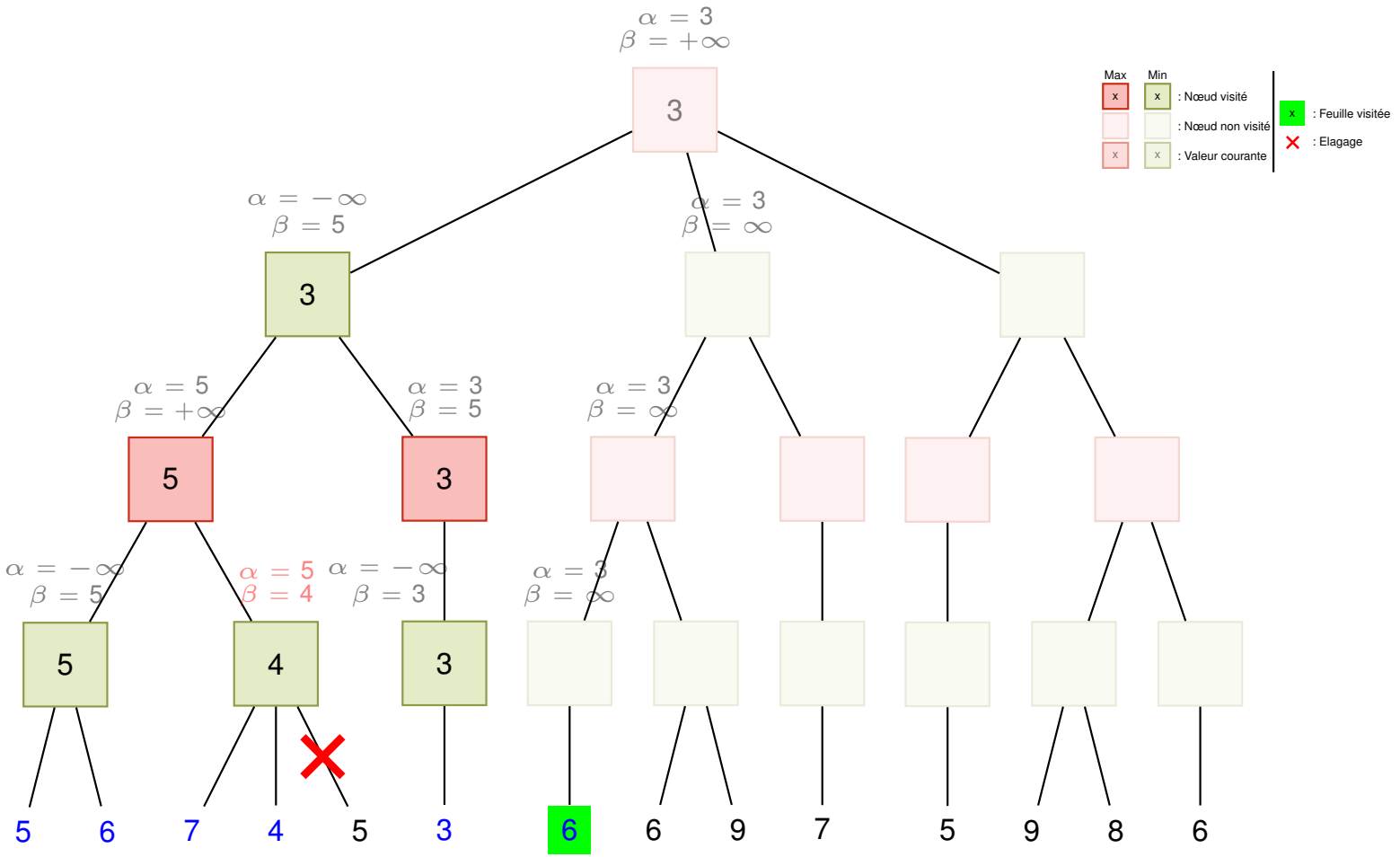
Exemples

Exemple 2



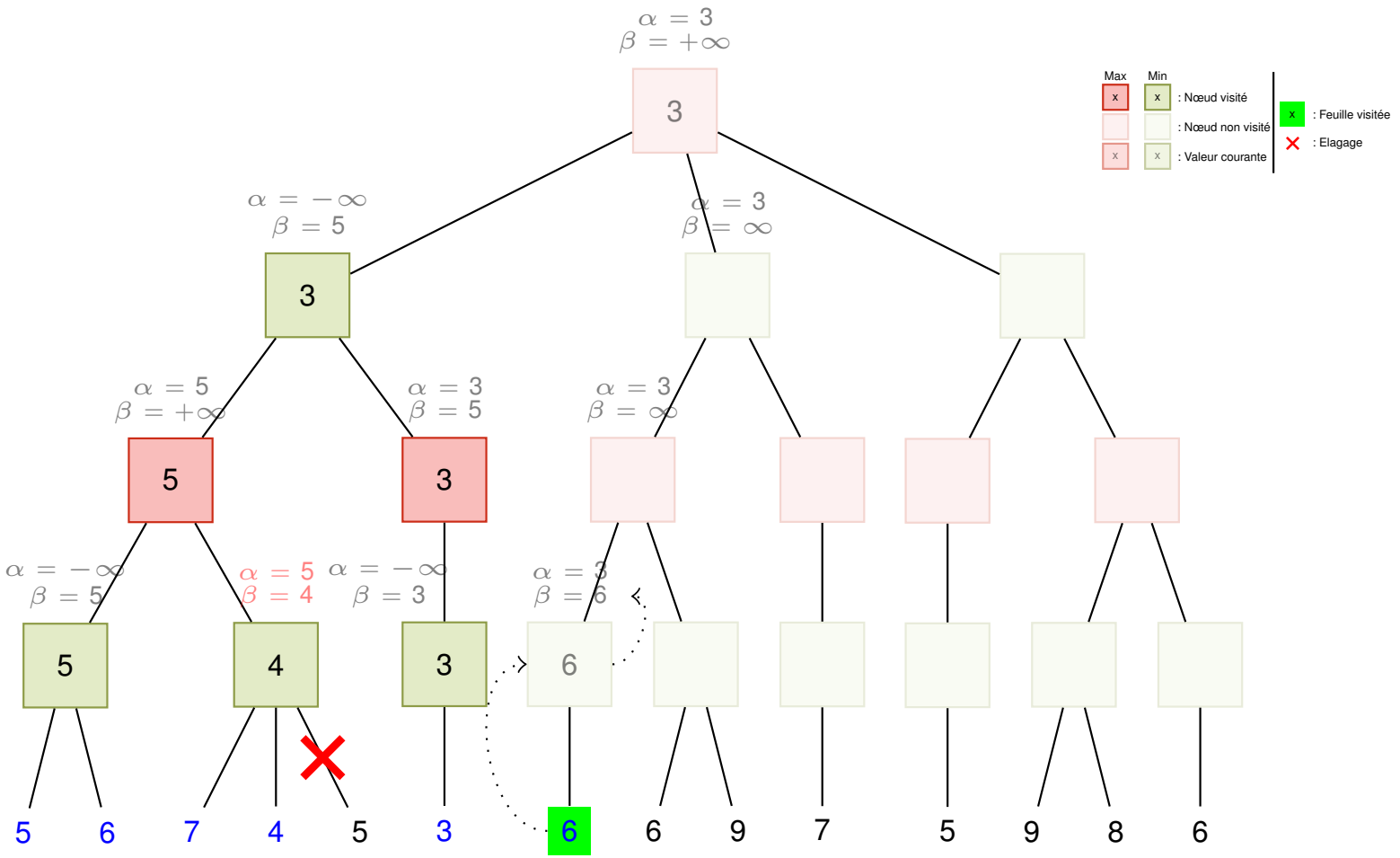
Exemples

Exemple 2



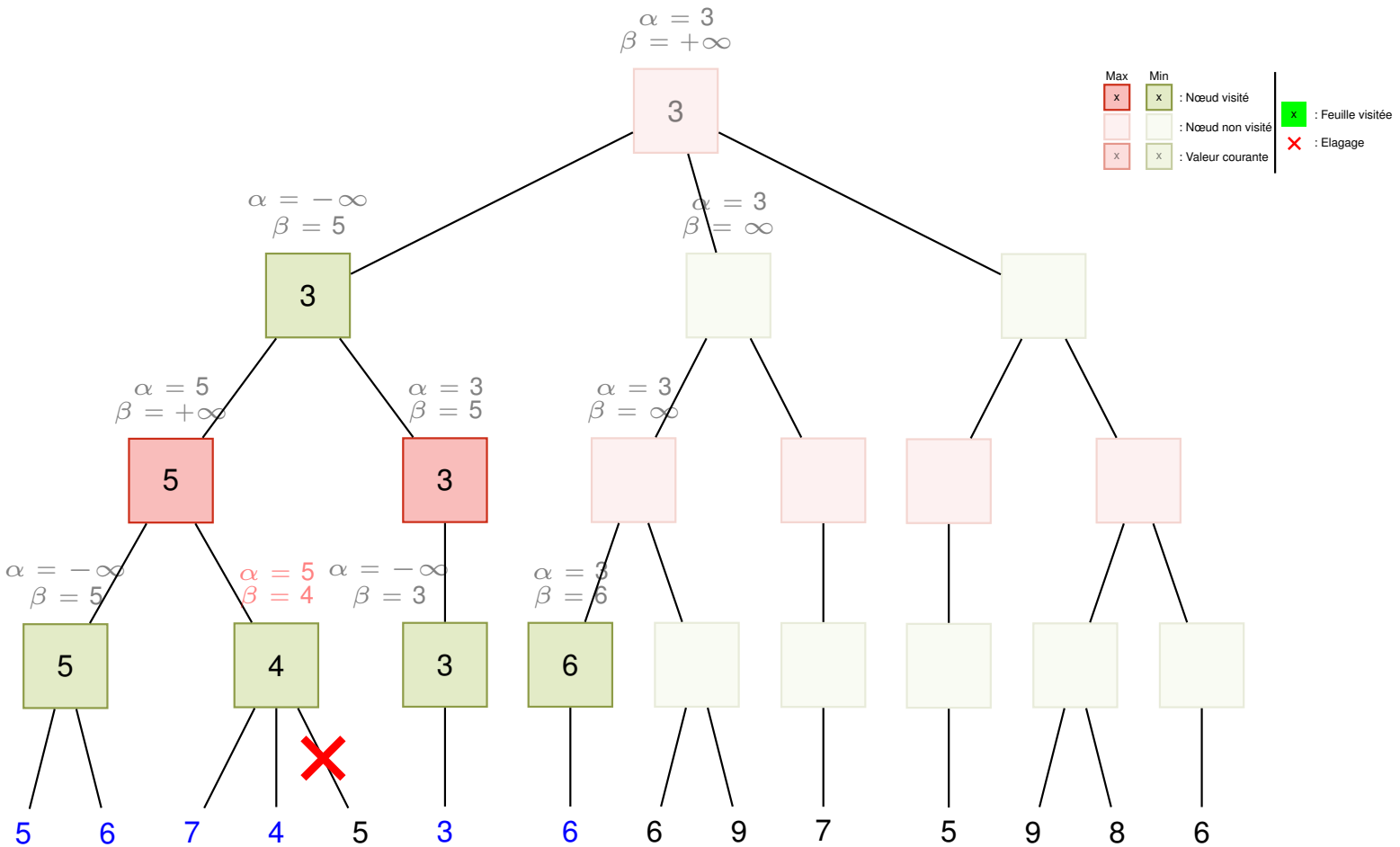
Exemples

Exemple 2



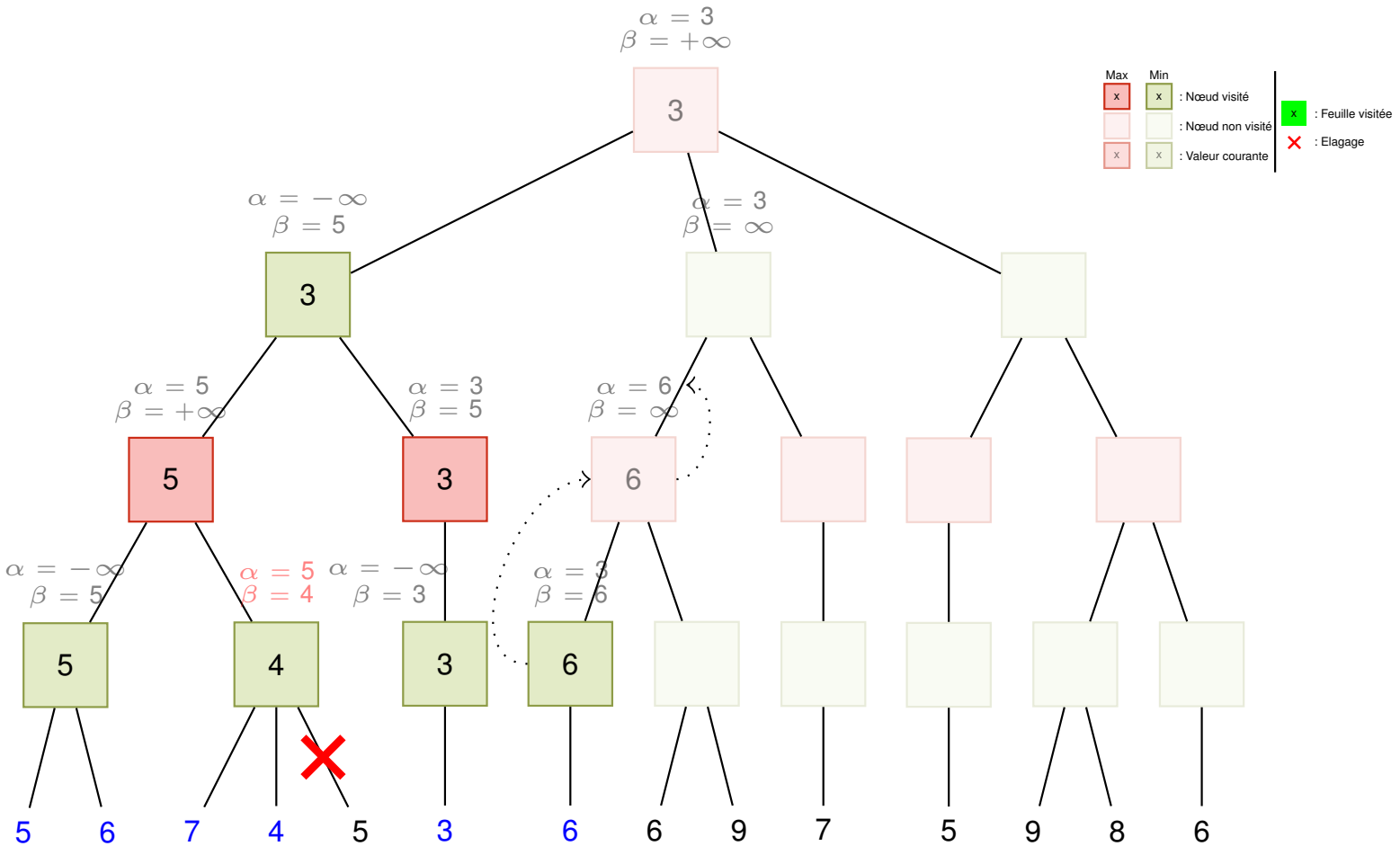
Exemples

Exemple 2



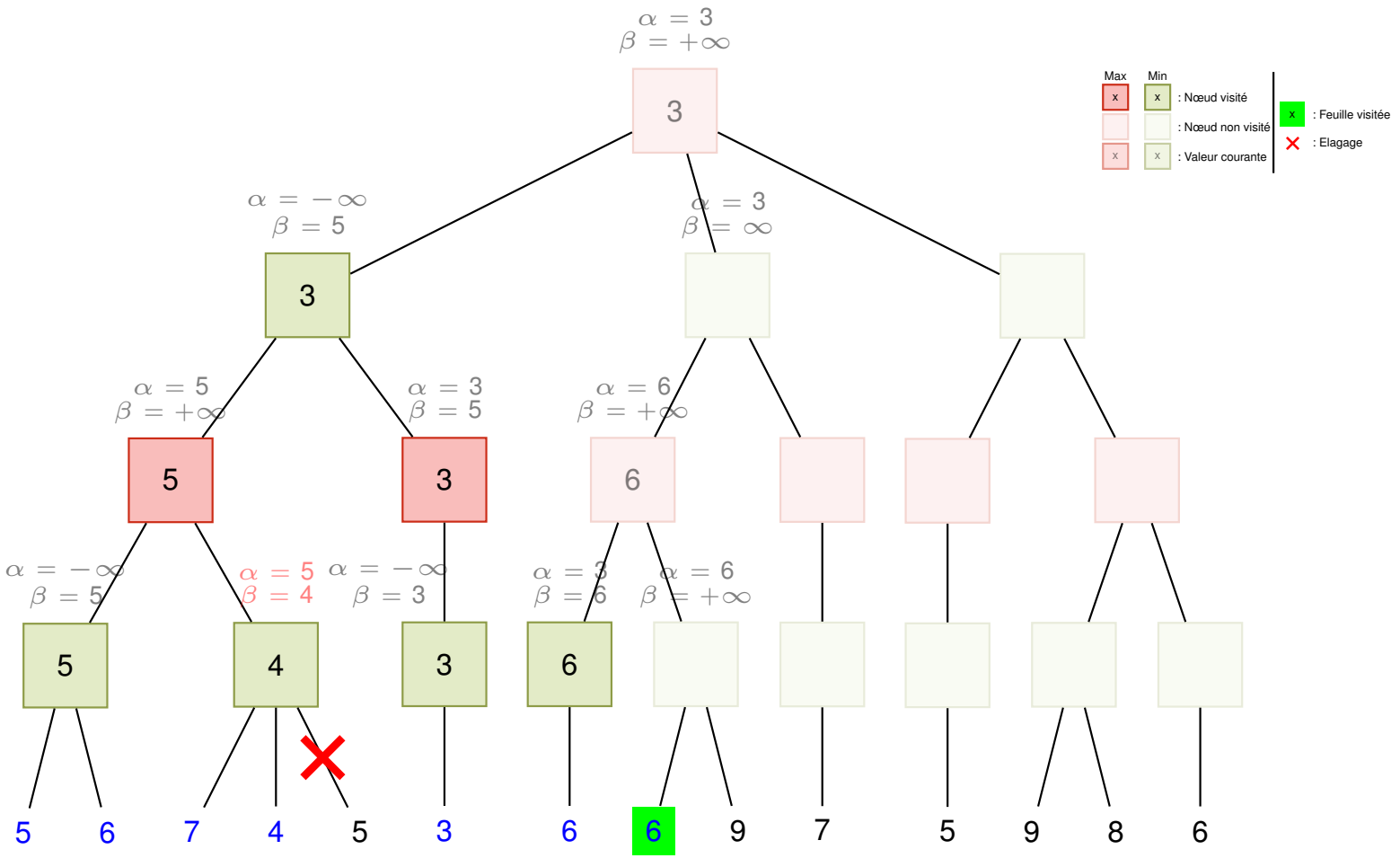
Exemples

Exemple 2



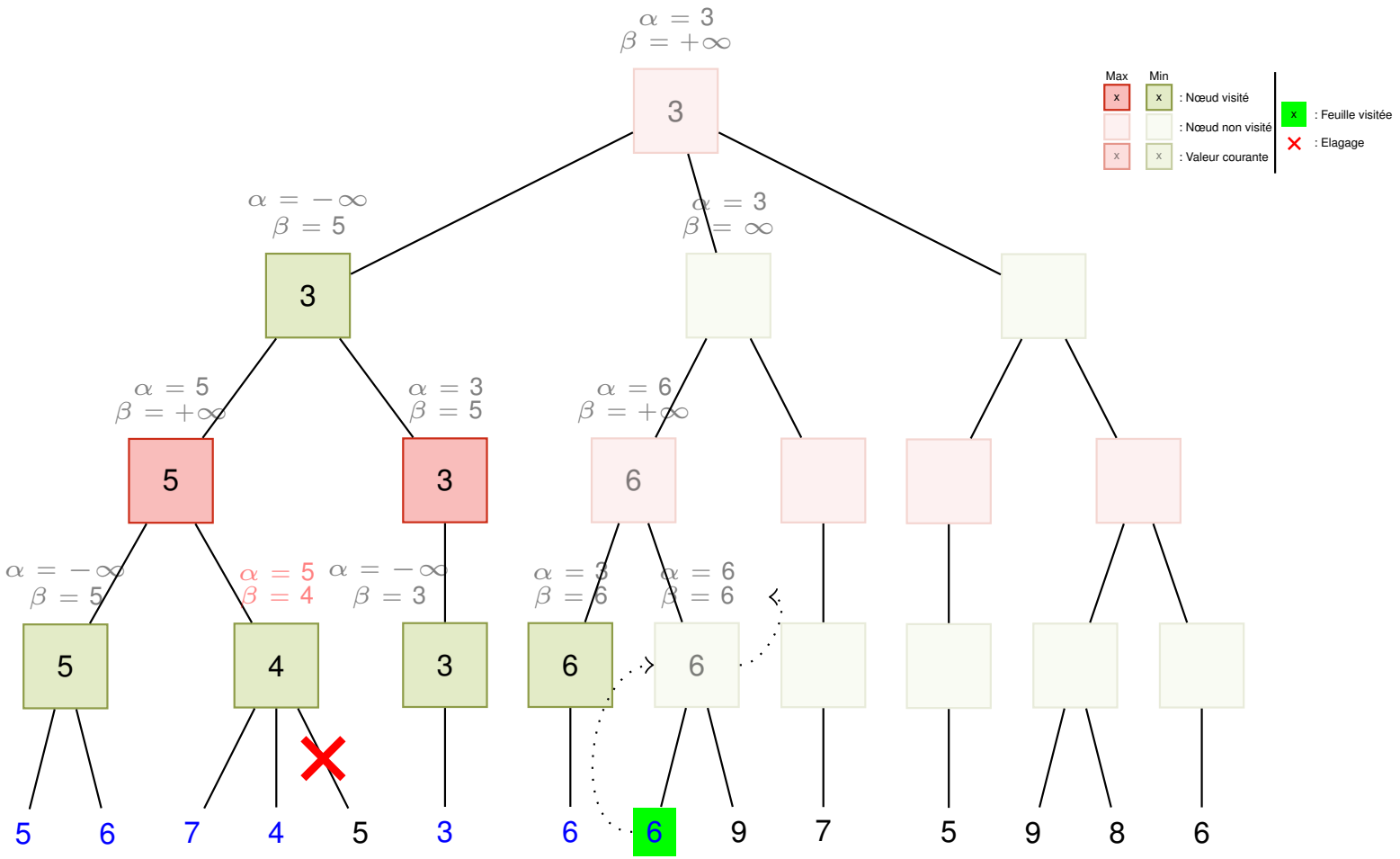
Exemples

Exemple 2



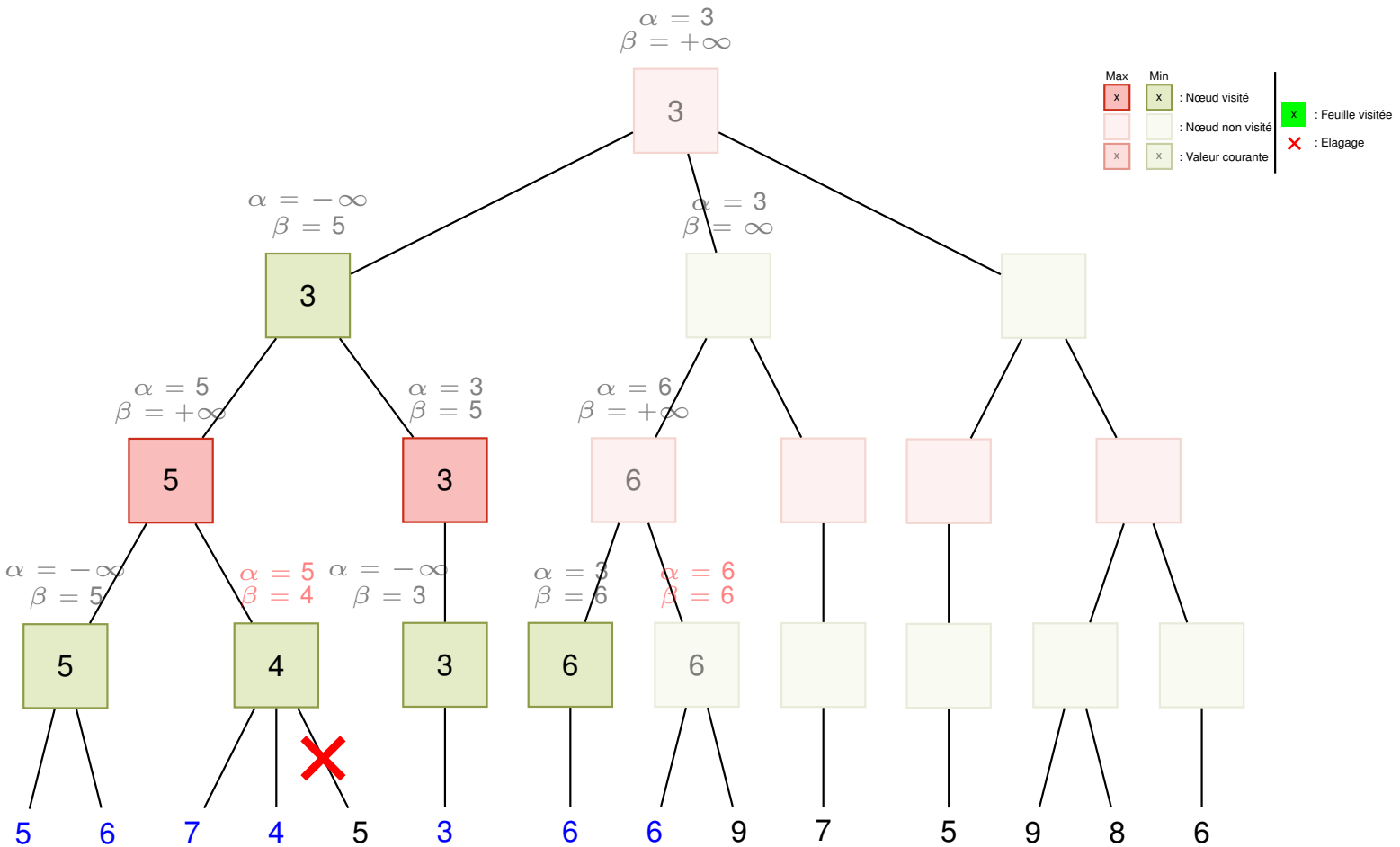
Exemples

Exemple 2



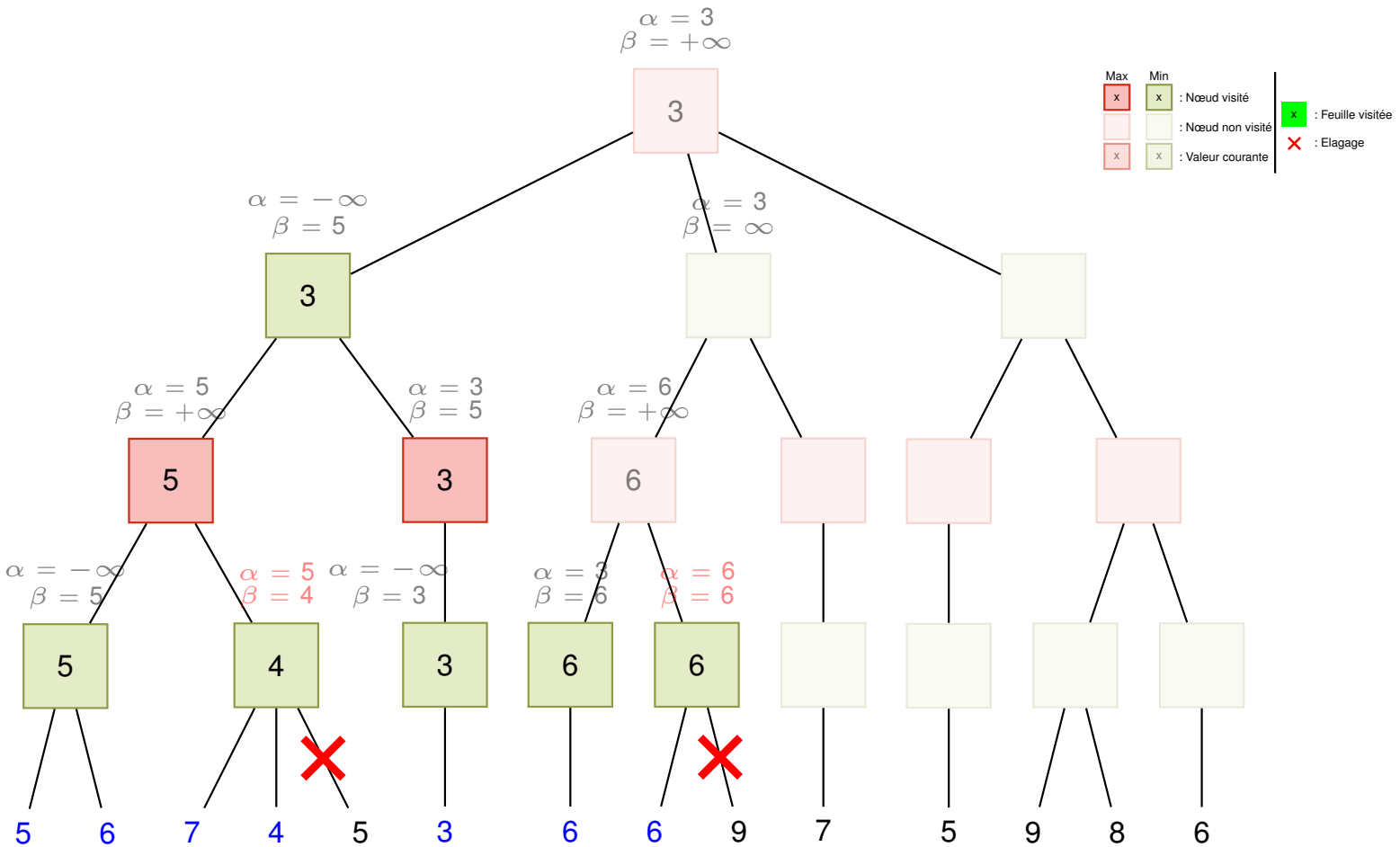
Exemples

Exemple 2



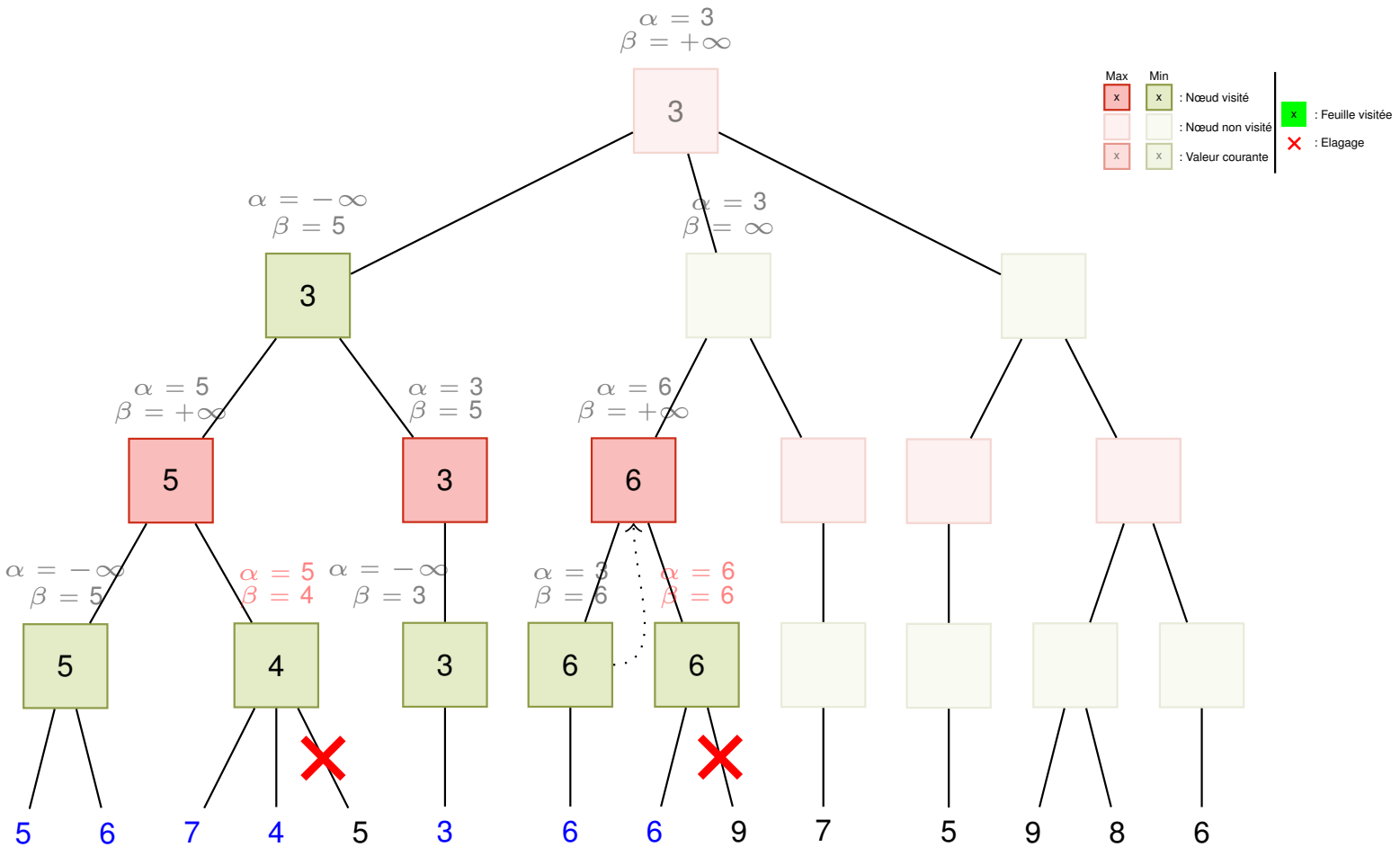
Exemples

Exemple 2



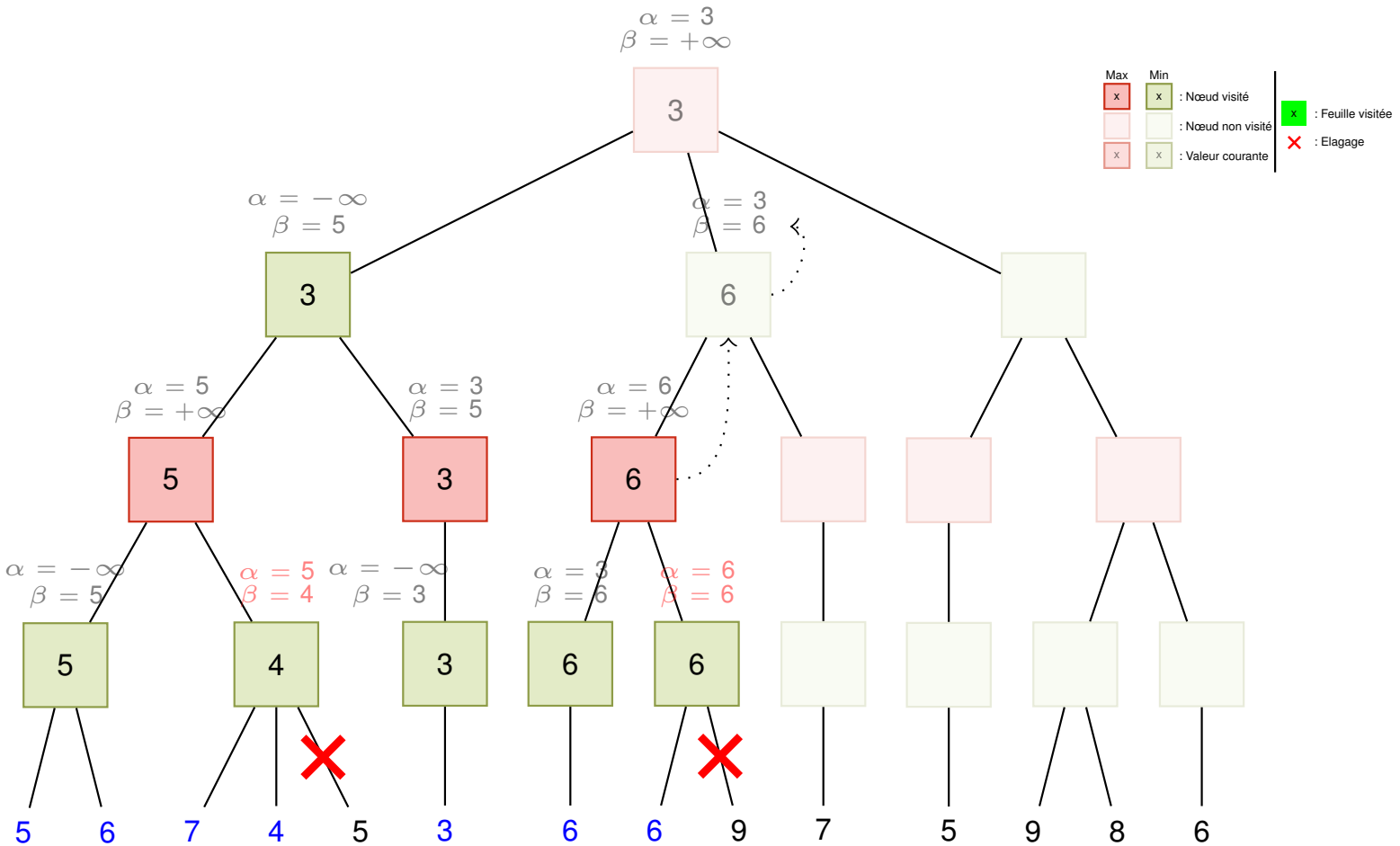
Exemples

Exemple 2



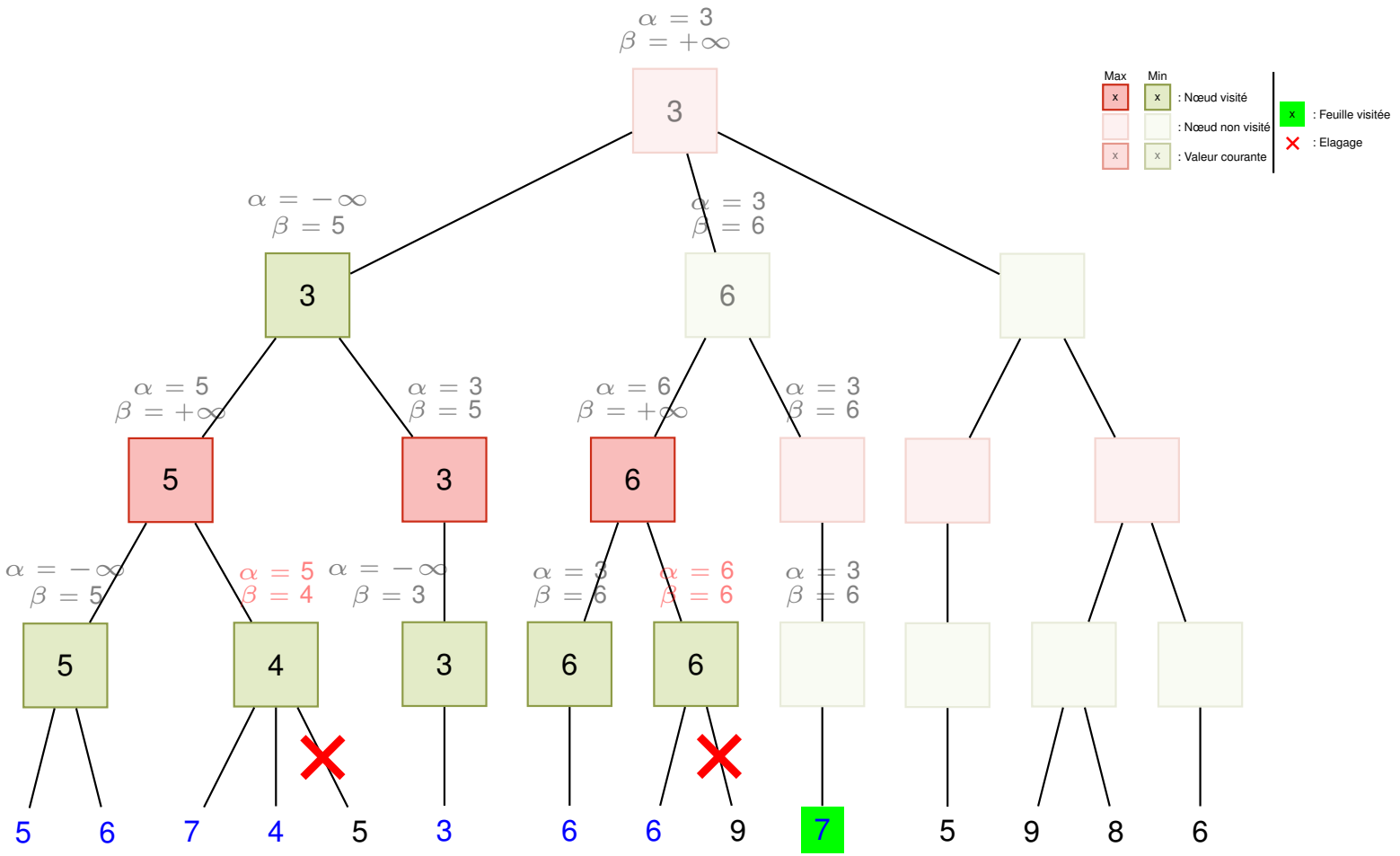
Exemples

Exemple 2



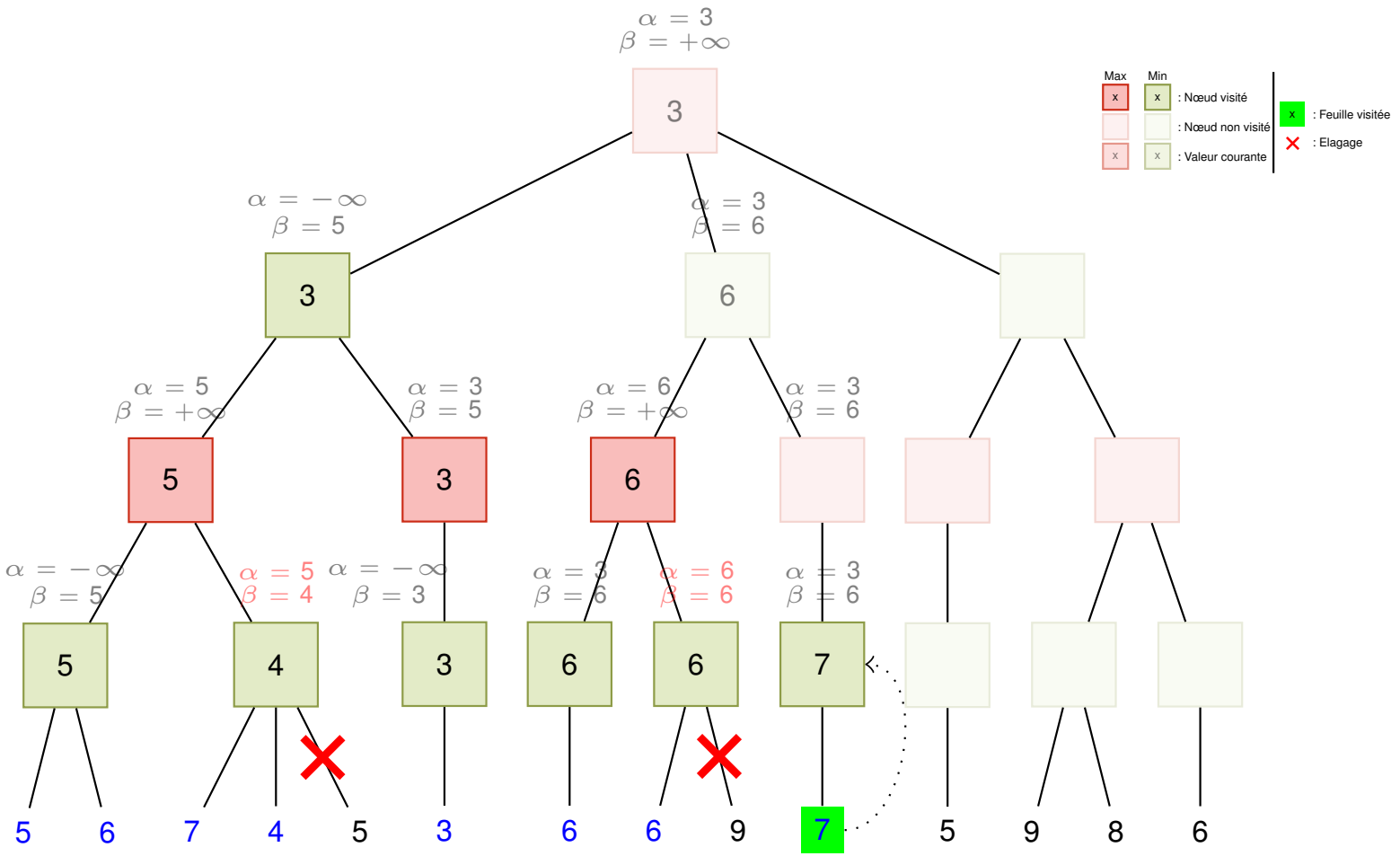
Exemples

Exemple 2



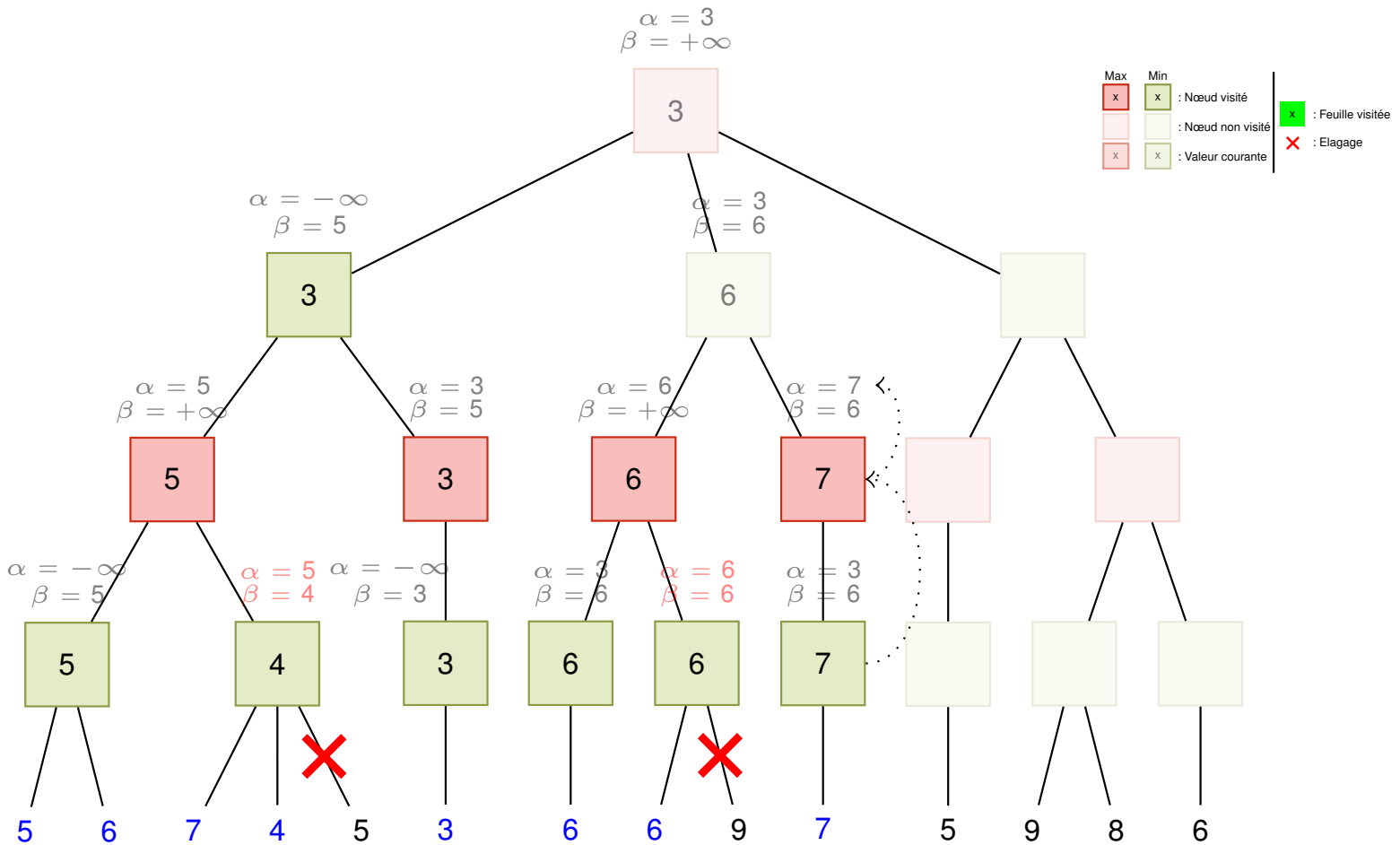
Exemples

Exemple 2



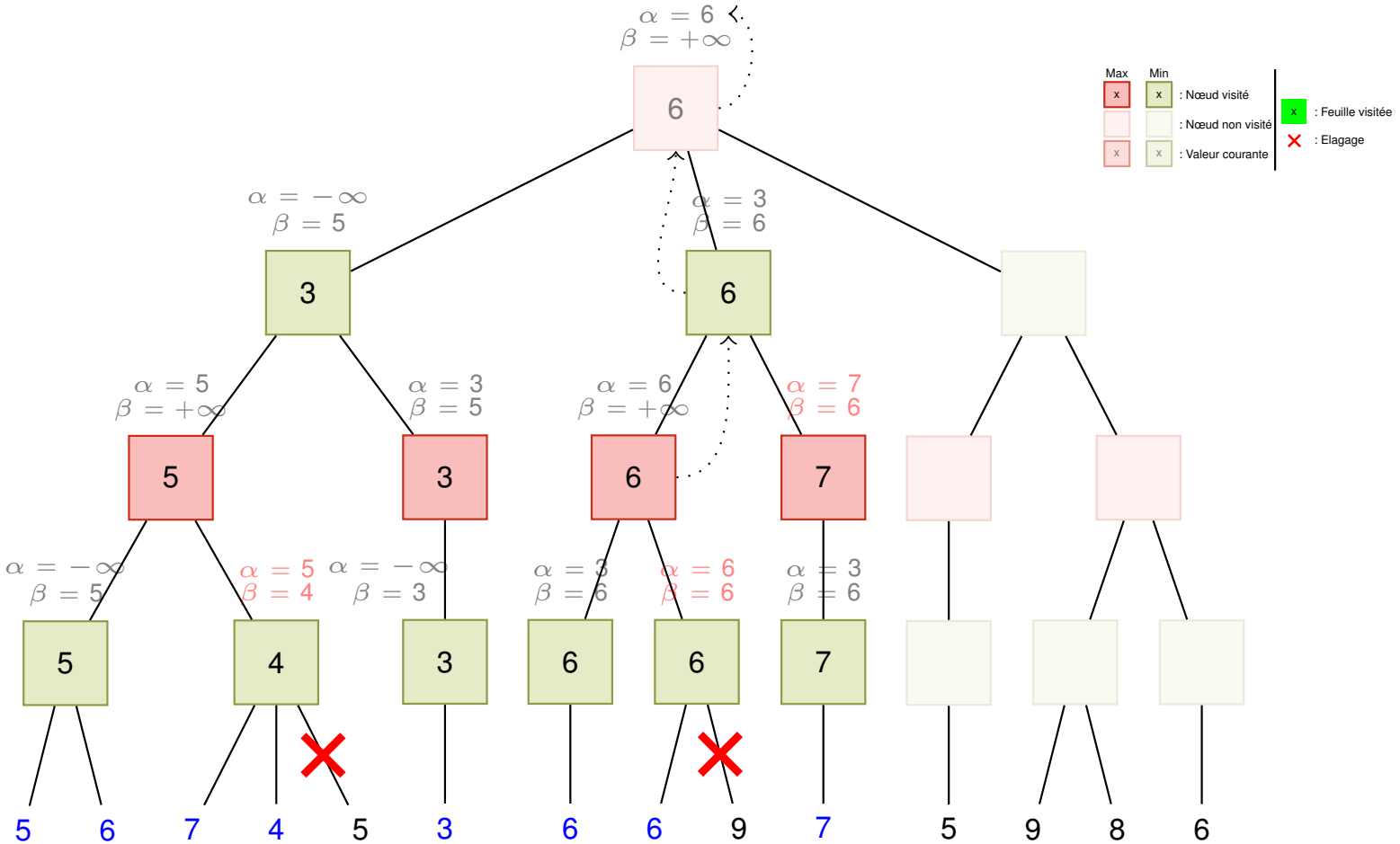
Exemples

Exemple 2



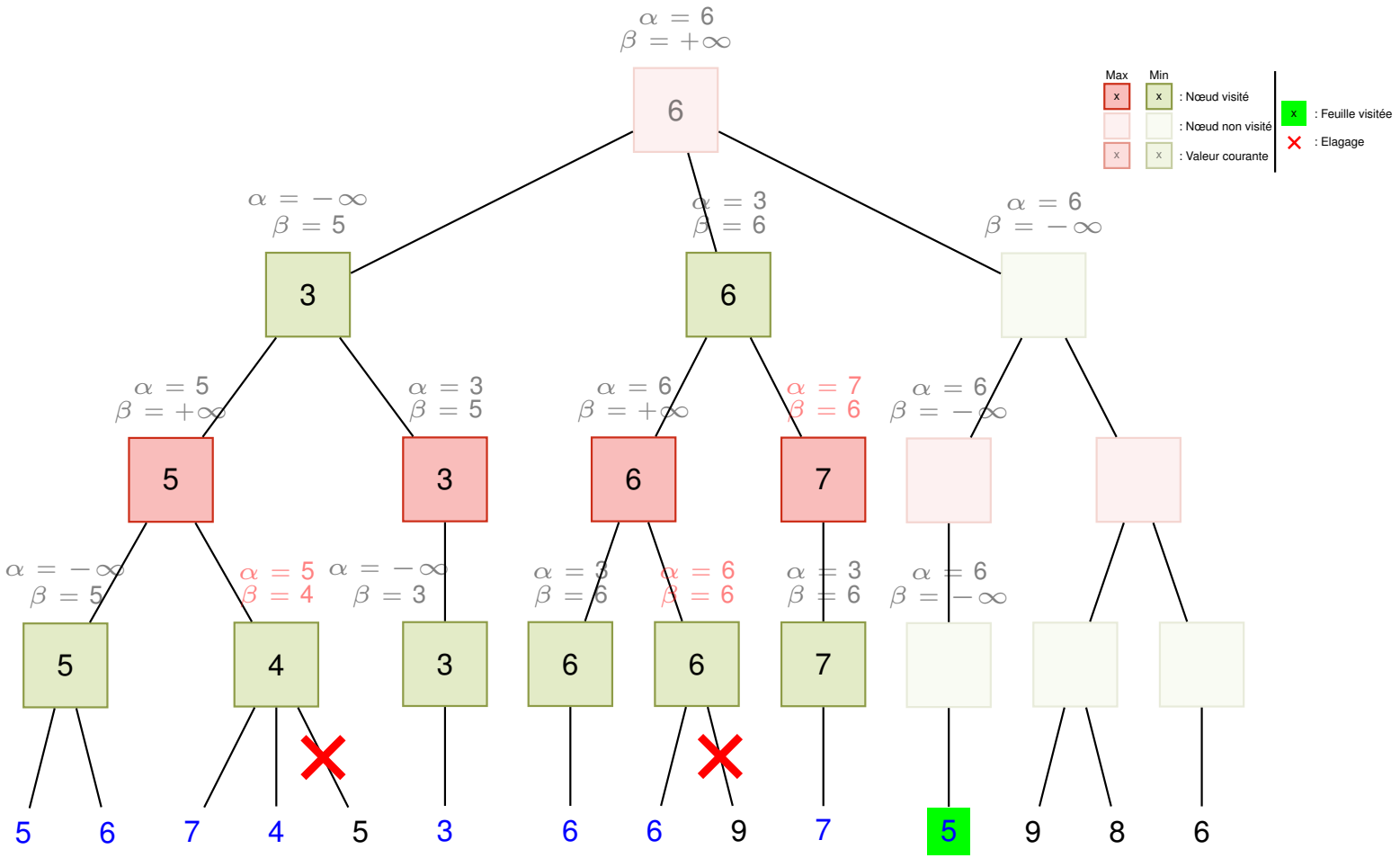
Exemples

Exemple 2



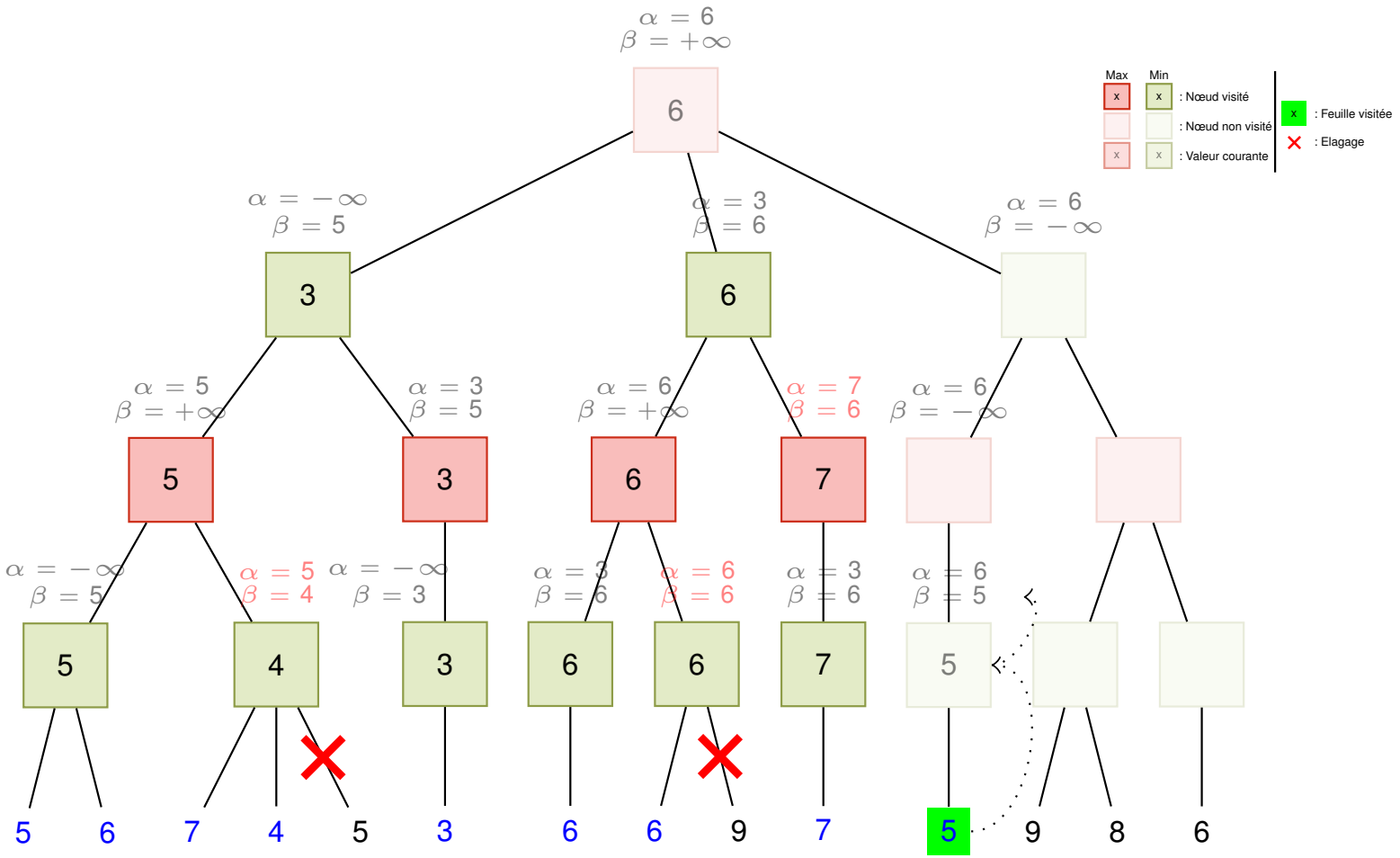
Exemples

Exemple 2



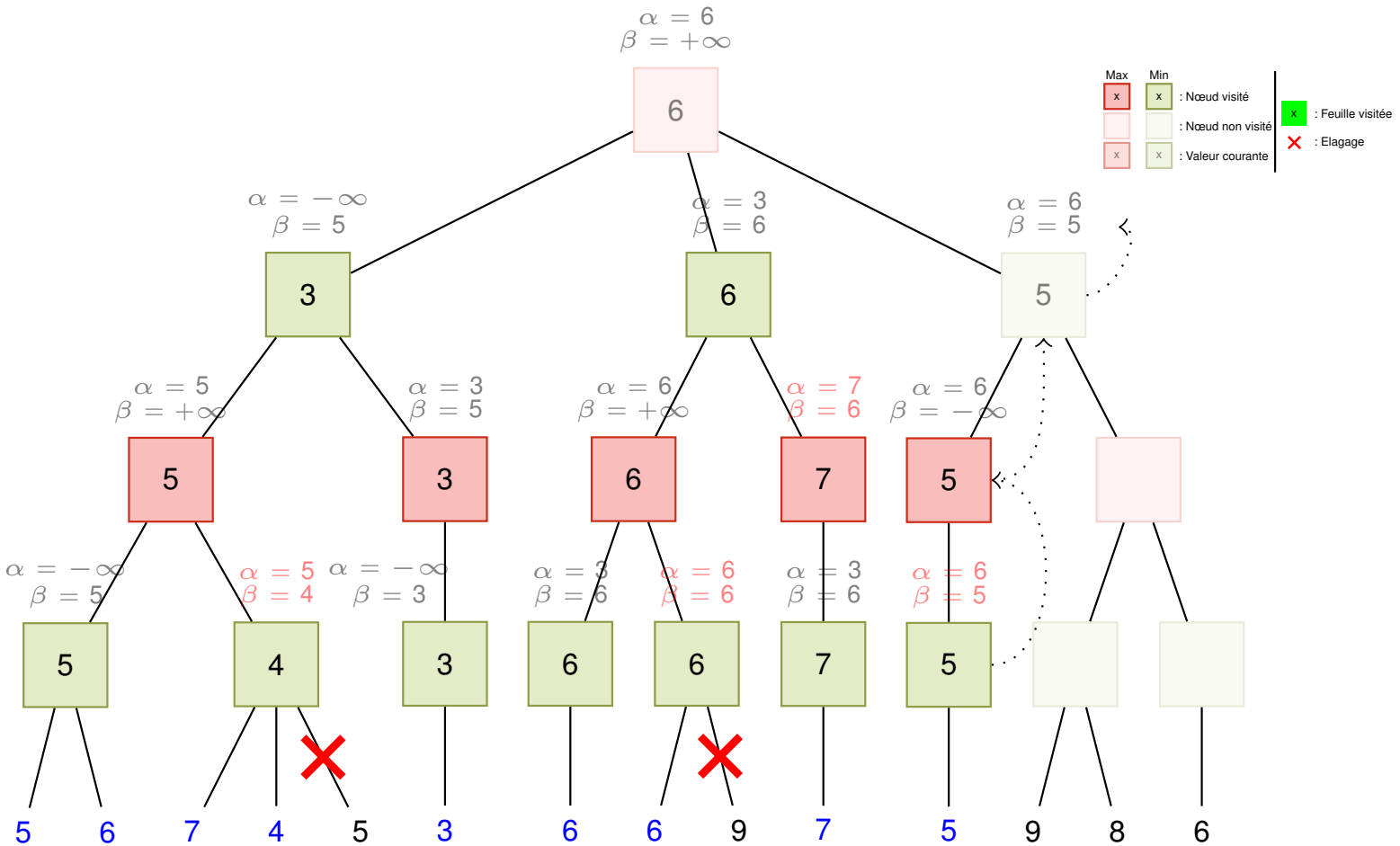
Exemples

Exemple 2



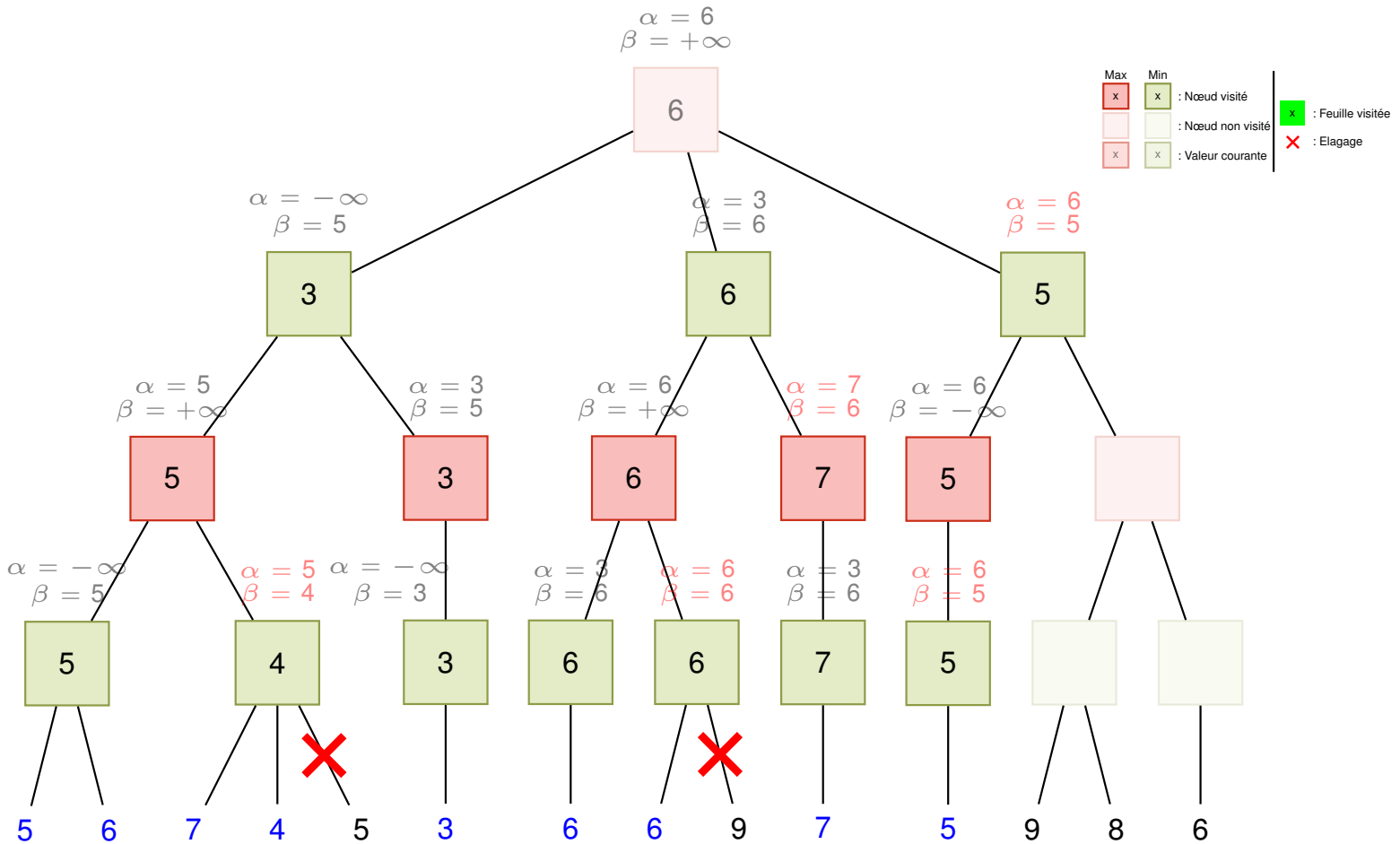
Exemples

Exemple 2



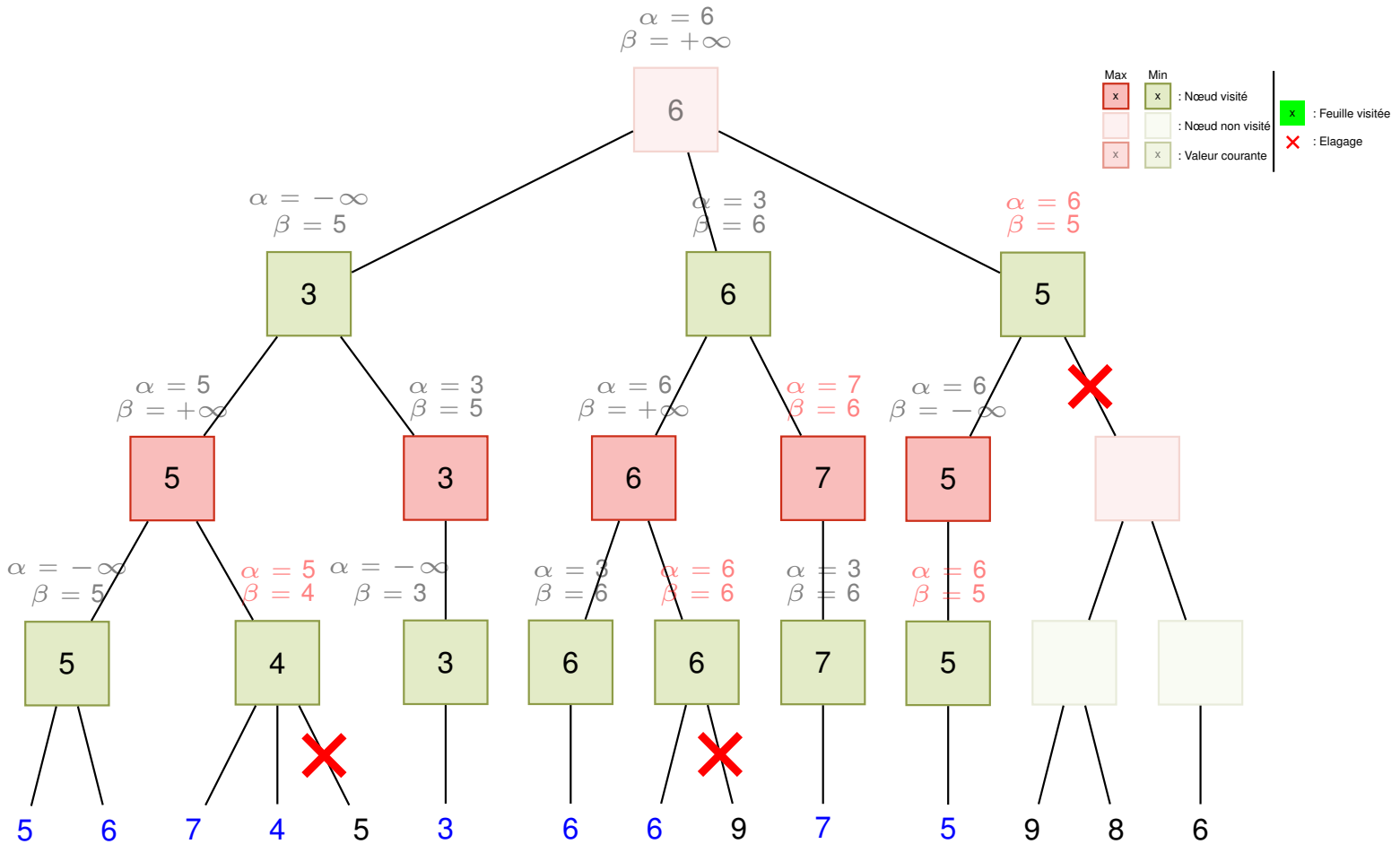
Exemples

Exemple 2



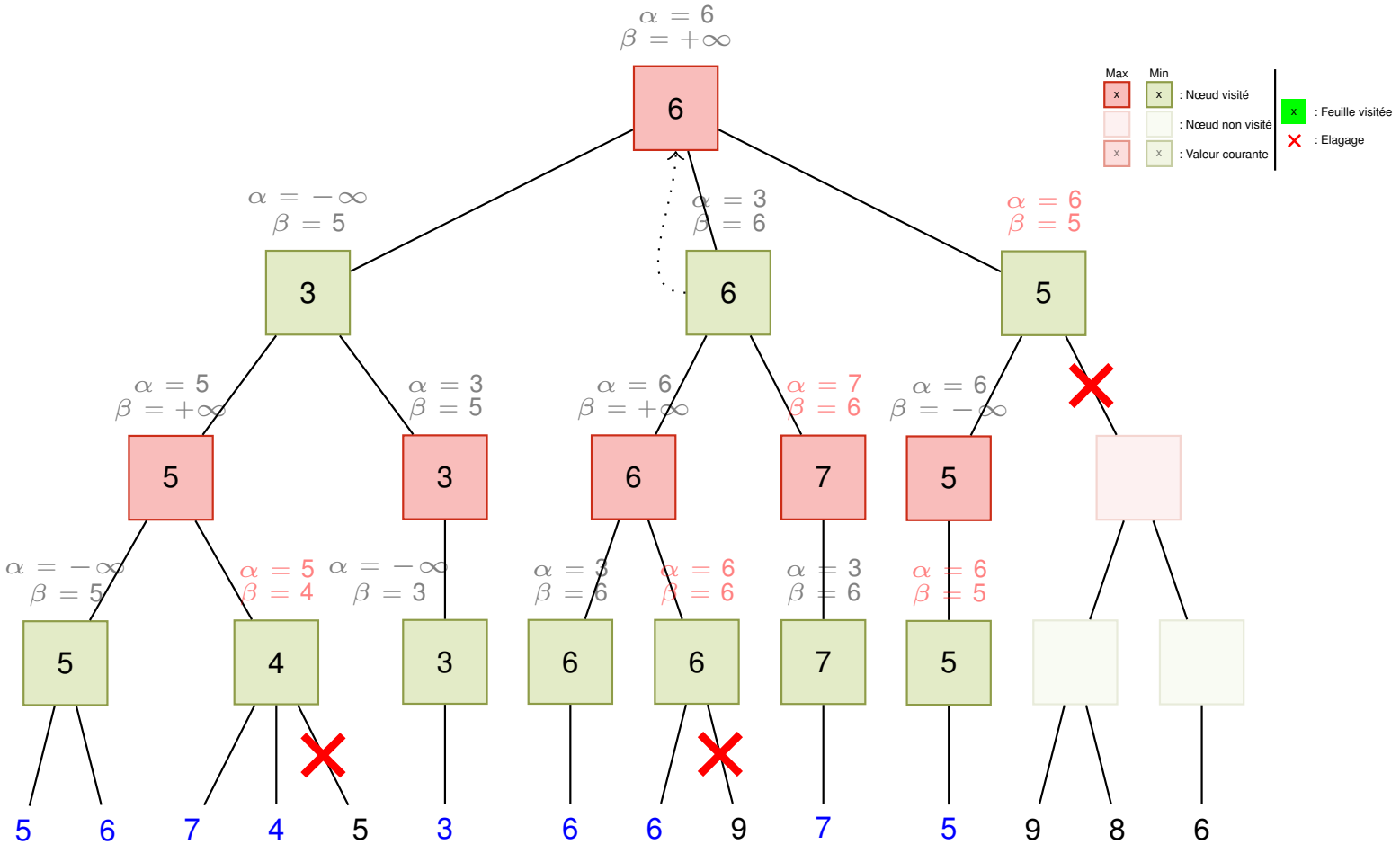
Exemples

Exemple 2



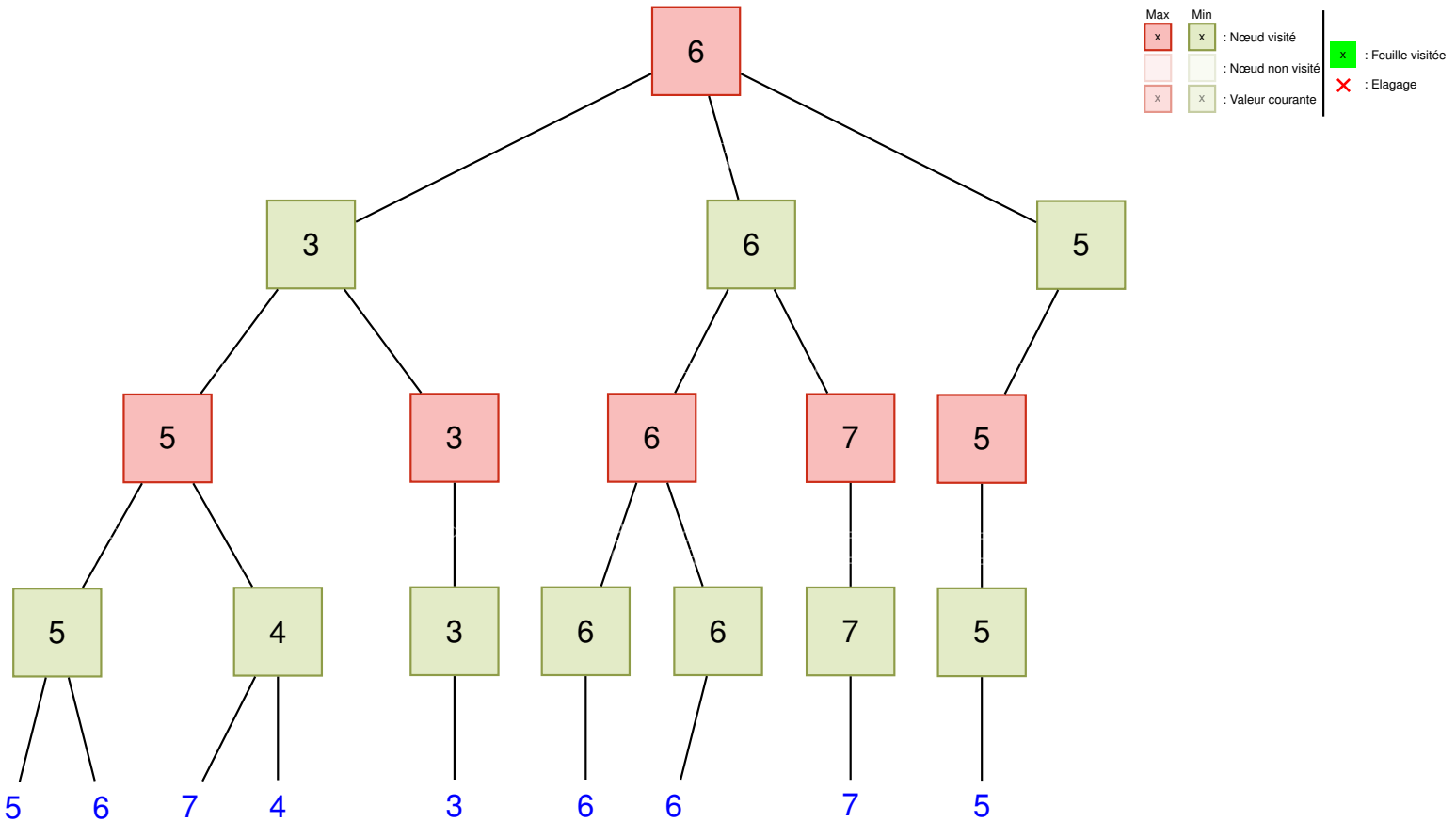
Exemples

Exemple 2



Exemples

Exemple 2



Optimisations d'Alpha-Beta

Les améliorations de l'algorithme alpha–beta sont de trois types :

- **Trier :** améliorer l'ordre d'examen des noeuds. Ce qui permet de faire beaucoup plus de coupes
- **Réduire :** plus la fenêtre de recherche $(\beta - \alpha)$ est petite, plus il y a de coupes
- **Réutiliser :** sauvegarde des résultats pour le cas où ils ré-apparaîtraient