JeuxII

Yue Ma

ma@lisn.fr

LISN - CNRS, Université Paris-Saclay

Exercise

TD1 - Exercice 1 - Question 2 (MinMax)

Négamax

Une réécriture simple de MiniMax

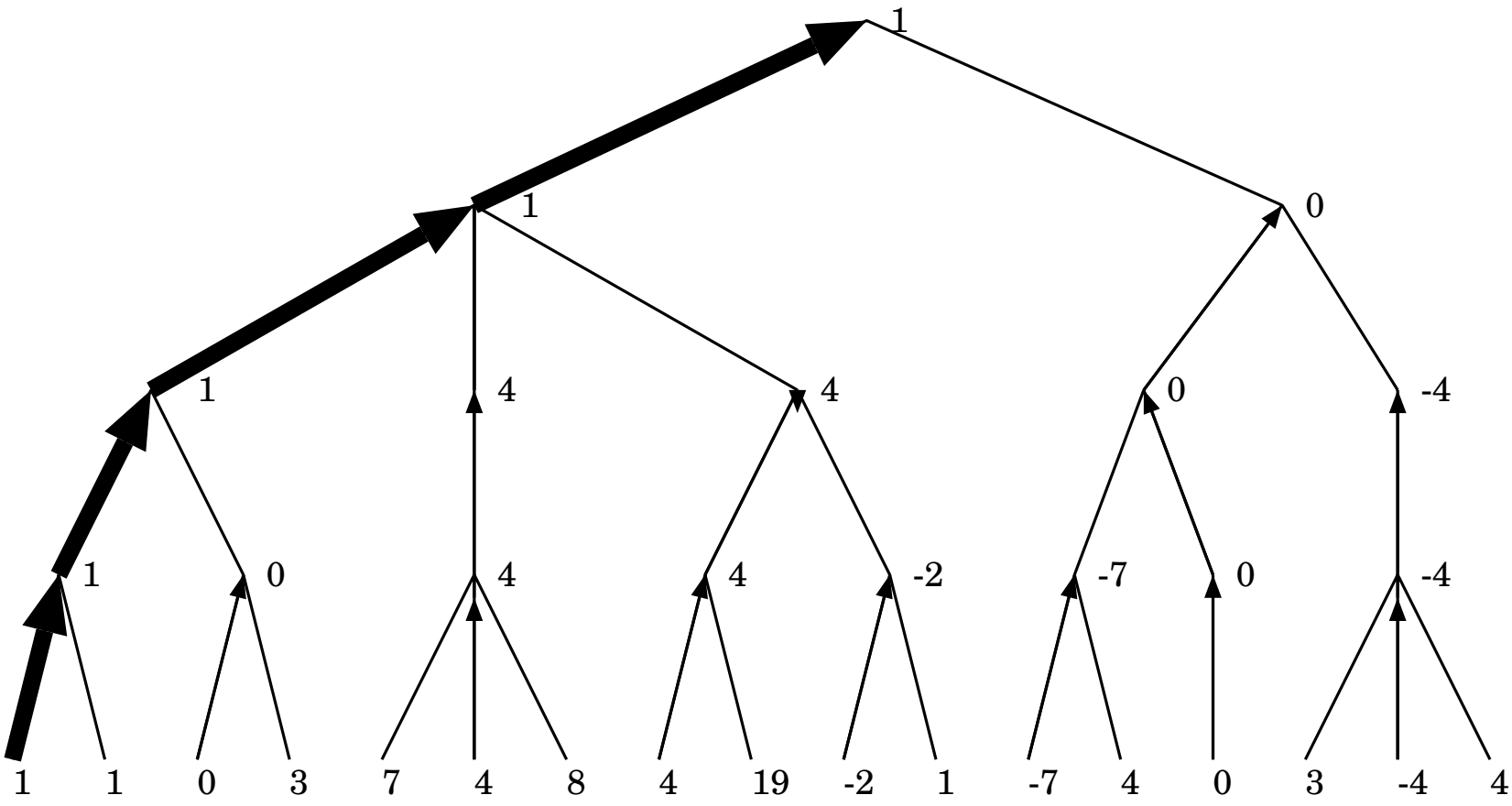
Idée

Au lieu d'alterner deux fonctions Max/Min, on se restreint à une seule fonction en utilisant l'opposé du résultat à chaque niveau.

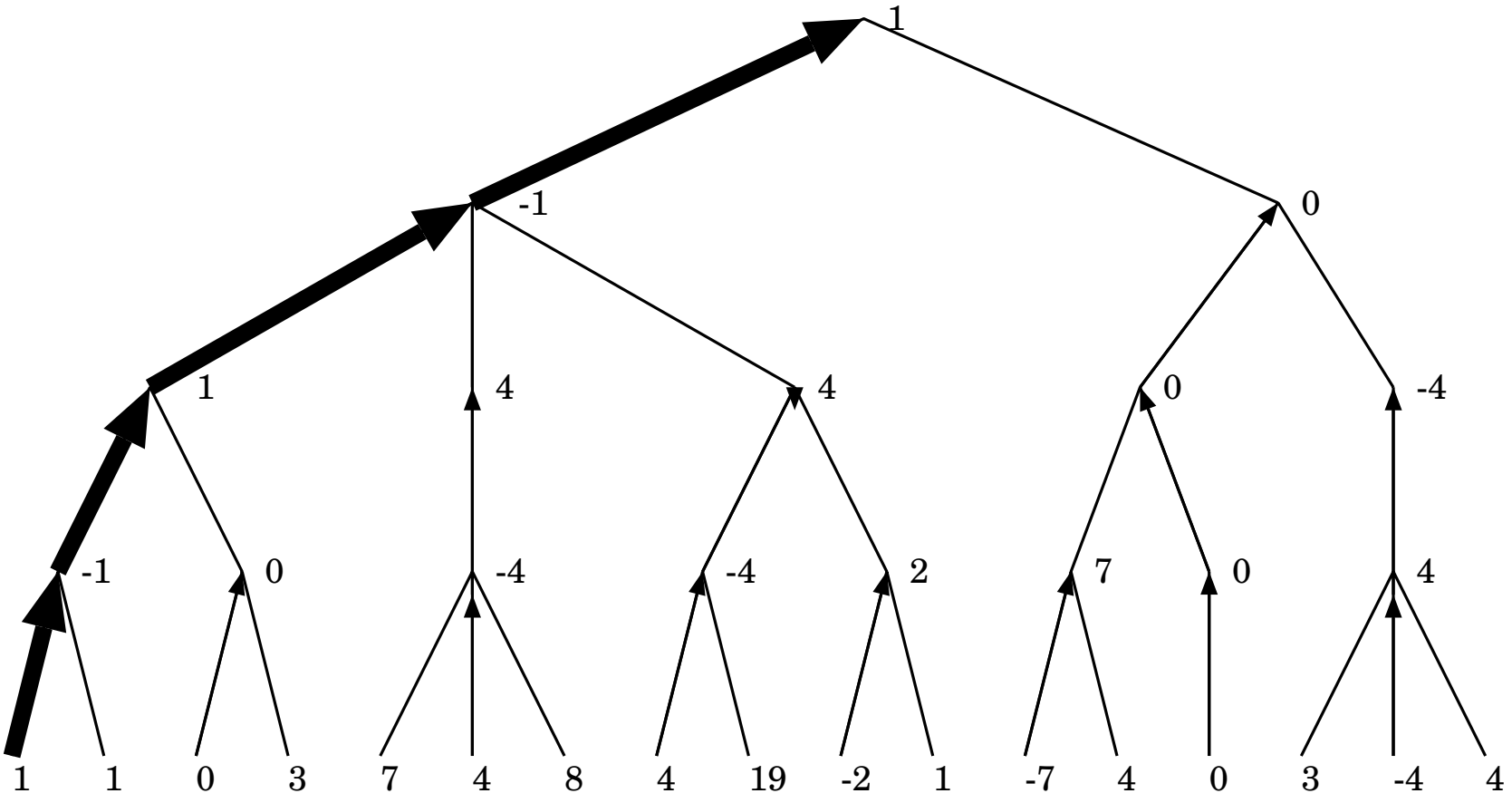
Attention

Cette famille d'algorithmes ne fonctionne que pour les niveaux pairs. Il faut adapter le résultat de la fonction heuristique sinon.

Un exemple : Negamax



Un exemple : Negamax



Négamax

```
1: Fonction NegaMax(etat)                    ▷ Évaluation niveau AMI
2:     etat : Plateau de jeu courant
3:     Meilleur : Évaluation du meilleur coup (localement)

4:     Si EstFeuille(etat) Alors             ▷ Fin de partie ou horizon atteint
5:         Retourner evalue(etat)            ▷ Évaluation heuristique
6:     Fin Si
7:     Meilleur ← −∞
8:     Pour Tout successeur s de etat Faire
9:         val ← −NegaMax(s)
10:        Si val ≥ Meilleur Alors
11:            Meilleur ← val
12:        Fin Si
13:    Fin Pour
14:    Retourner Meilleur
15: Fin Fonction
```

Recherche dans les arbres de jeux

Recherche dans les arbres de jeux

Rappel : L'espace des états atteignables de l'algorithm MinMax peut être gigantesque. Il dépend :

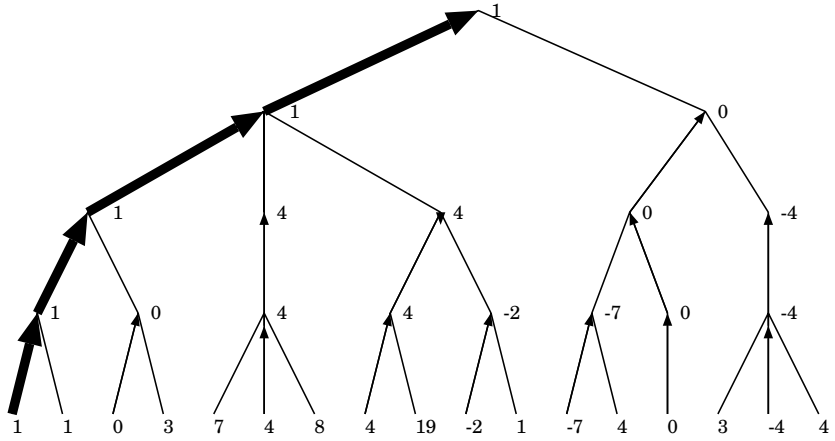
- Facteur de branchement b du jeu
- Nombre de coups n d'une partie

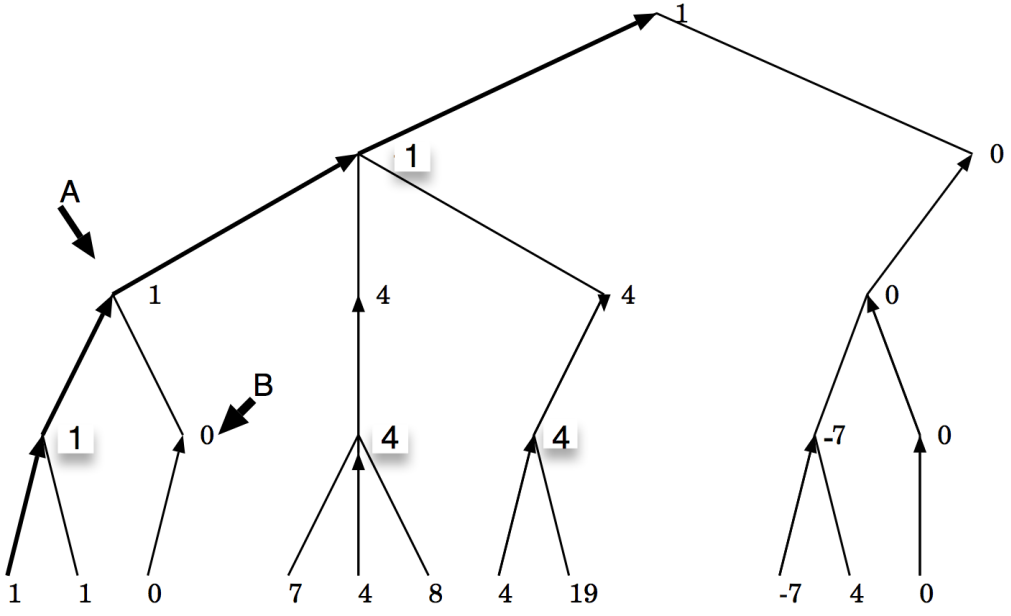
Idée des tailles de certains jeux

- Échecs : $b \tilde{} 35$ et $n \tilde{} 30$ coups au minimum
  |graphe d'états| = $b^n = 35^{30}$
  Soit
  2099139642966190174953146230280399322509765625
- Othello : $5 \leq b \leq 15$
- Go : $b \tilde{} 360$

Il faut retrouver l'idée d'élagage de l'arbre de recherche !!!

Un exemple


1
1
0
1
4
4
0
-4
1
0
4
4
-2
-7
0
-4
1 1 0 3 7 4 8 4 19 -2 1 -7 4 0 3 -4 4


1
1
0
A
1
4
4
0
B
1
0
4
4
-7
0
1 1 0 7 4 8 4 19 -7 4 0

ÉLAGAGE ADMISSIBLE

On veut trouver la même valeur d'évaluation finale du noeud racine sans développer tout l'arbre. Il faut donc élaguer des parties de l'arbre de recherche qui sont sans conséquence sur l'évaluation d'un noeud.

INTUITIVEMENT

Soit un noeud *n* dans l'arbre de recherche, tel que *Joueur* peut jouer en *n*.
S'il existe pour *Joueur* un choix *m* meilleur que *n* (soit à partir du noeud parent de *n*, soit plus haut dans l'arbre), *n* ne sera jamais effectivement joué.

DEUX TYPES DE COUPES : α ET β

- α : le meilleur choix à un instant donné pour Max sur le chemin développé. **La valeur α est croissante**
- β : le meilleur choix à un instant donné pour Min sur le chemin développé. **La valeur β est décroissante**

Initialization : $\alpha = -\inf$ et $\beta = +\inf$.
Les coupes auront lieu dès que α est supérieur à ou égale à β

À bien noter

- Pour évaluer un plateau, on appelle la fonction *MaxValue* avec le plateau racine, $\alpha = -\inf$, et $\beta = +\inf$.
- Les variables α et β sont bien *locales* à chaque fonction.
- Elles sont changées par l'intermédiaire des valeurs de retour de fonction.

Le Principe de Snow White (Blanche-Neige) : Blanche-Neige hérite les valeurs de parents, et sa belle m^eme prend la meilleure valeur des descendants

Algorithme $\alpha\beta$

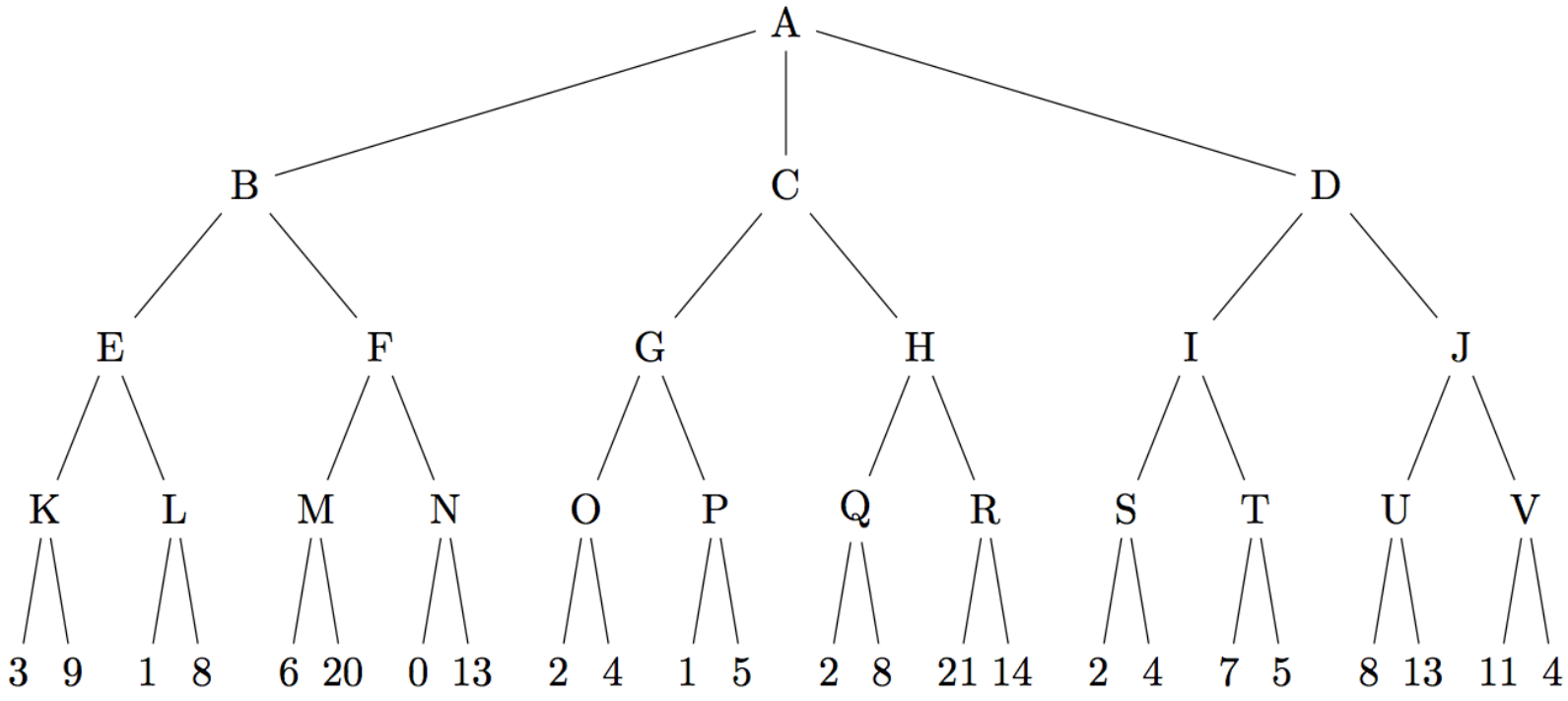
Les deux fonctions ensembles

```
Fonction MaxValue(etat, α, β))                          ▷ Niveau AMI
    Si EstFeuille(etat) Alors
        Retourner evalue(etat)                          ▷ Évaluation heuristique
    Fin Si
    Pour Tout successeur s de etat Faire
        α ← max(α,MinValue(s, α, β))
        Si α ≥ β Alors                                  ▷ Coupe β
            Retourner β
        Fin Si
    Fin Pour
    Retourner α
Fin Fonction
```

```
Fonction MinValue(etat, α, β)
    Si EstFeuille(etat) Alors
        Retourner evalue(etat)                          ▷ Niveau ENNEMI
    Fin Si
    Pour Tout successeur s de etat Faire
        β ← min(β,MaxValue(s, α, β))
        Si α ≥ β Alors                                  ▷ Coupe α
            Retourner α
        Fin Si
    Fin Pour
    Retourner β
Fin Fonction
```

EXEMPLE D'ARBRE DE JEU

TD1 - Exercise 2



Propriétés de $\alpha\beta$

Efficacité théorique

Si on suppose que les fils sont ordonnées idéalement (ou presque), le coût de l'exploration est de $O(b^{d/2})$ au lieu de $O(b^d)$

- Le facteur de branchement théorique passe de b à $\sqrt{b}$
- La recherche peut aller deux fois plus loin dans l'arbre

Preuve : voir l'appendix

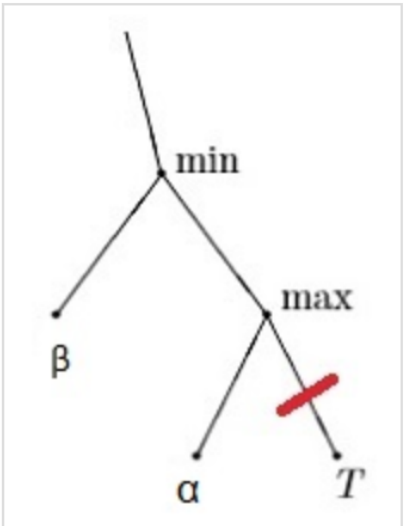
Type d'élagage

Alpha-cutoffs for the min-player, beta-cutoffs occur for the max-play

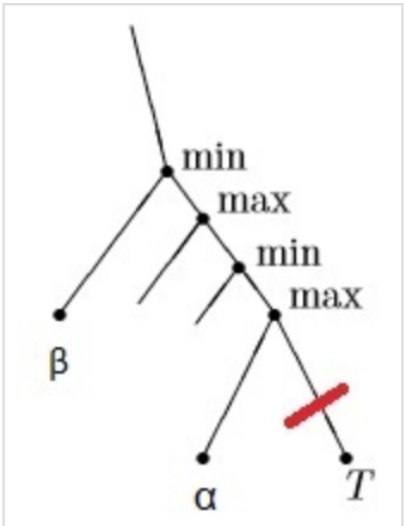
	alpha-cutoffs	beta-cutoffs
deep cutoff shallow cutoff		Example see the figure below

Shallow or Deep

A **shallow** cutoff occurs if the window was narrowed at the parent node, that is, reduced current beta. A **deep** cutoff occurs if the window was narrowed closer to the root with an odd ply-distance of at least three to the current node. Deep cutoffs were sometimes omitted in early chess programs if not passing alpha through the recursive call [2].



Shallow Cutoff if α >= β



Deep Cutoff if α >= β [3]

See to https://chessprogramming.wikispaces.org

Négamax appliqué à $\alpha\beta$

```
1:  Fonction Negαβ(etat, α, β)                    ▷ Évaluation niveau AMI

2:      Si EstFeuille(etat) Alors                 ▷ Fin de partie ou horizon atteint
3:          Retourner evalue(etat)                ▷ Évaluation heuristique
4:      Fin Si
5:      Pour Tout successeur s de etat Faire
6:          val ← −Negαβ(s, −β, −α)
7:          Si val > α Alors
8:              α ← val
9:              Si α > β Alors
10:                 Retourner α                   ▷ Coupe
11:             Fin Si
12:         Fin Si
13:     Fin Pour
14:     Retourner α
15: Fin Fonction
```

Exercise

TD1- Exercise 2 ($\alpha\beta$)

Appendix

Preuve des performances de $\alpha\beta$

Quelles sont les performances de $\alpha\beta$?

Questions

- Combien de noeuds $\alpha\beta$ explore-t-il ?
- Quelle est la différence de performance avec MiniMax ?
- Avec le méme temps donné quelle est la différence de profondeur entre les deux méthodes ?

Definition

On appelle arbre de jeu **uniforme de largeur** *l*, un arbre de jeu où tous les noeuds non terminaux ont exactement *l* fils.

MiniMax explore donc exactement l^p noeuds, où p est la profondeur de recherche.
Combien de noeuds $\alpha\beta$ explore-t-il ?

Types de noeuds visités lors de l'exploration

Tous les noeuds élagués lors de la recherche *Neg*$\alpha\beta$ le sont dés que $\alpha \geq \beta$. L'appel récursif *Neg*$\alpha\beta$ se ferait sur la fenétre $[\alpha, \beta]$ avec $\alpha \geq \beta$.

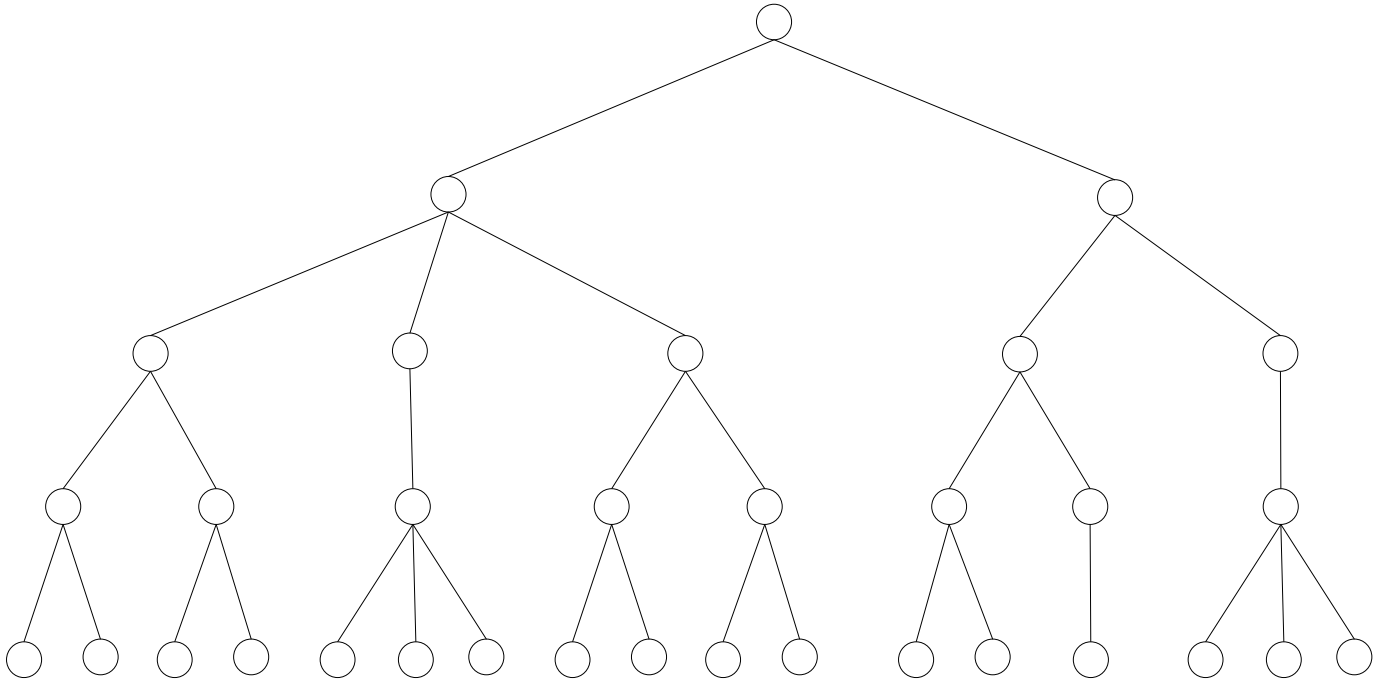
Trois types de noeuds jamais élagués

À chaque niveau de l'arbre, $\alpha\beta$ est appelé avec une certaine fenétre d'appel $[\alpha, \beta]$. Trois types de noeuds ne *peuvent donc* jamais étre élagués.

1. la fenétre d'appel est $[-\infty, +\infty]$
2. la fenétre d'appel est $[-\infty, b]$ avec $b \neq +\infty$
3. la fenétre d'appel est $[a, +\infty]$ avec $a \neq -\infty$

Note 1 : c'est en supposant que ∞ n'est pas une valeur heuristique.
Note 2 : les fils d'un noeud non élagué peuvent bien entendu l'étre.

TYPES DE NOEUDS : EXEMPLE



Type 1 | Type 2 | Type 3 | Inconnu

Types de noeuds visités par $\alpha\beta$

PropriŇ•tŇ• (Visite de l'arbre critique)

L'algorithme $\alpha\beta$ utilisé avec la fenétre $[-\infty, +\infty]$ regarde au moins l'arbre critique*, c'est-ŗ-dire l'ensemble des noeuds de types 1, 2 et 3, et uniquement celui-ci dans le cas où l'arbre est parfaitement ordonné.*

Si l'arbre n'est pas parfaitement ordonné, on visitera plus de noeuds, jusqu'ŗ l'arbre complet (I^p noeuds).
Quelle est la borne inférieur ?

Comptage des différents noeuds

Types de noeuds :

- type 1 : racine et fils ainés des type 1
- type 2 : cadets de type 1 et tous les fils de type 3
- type 3 : fils aÓnés des type 2
- inconnus : cadets des type 2 et les fils des inconnus

Preuve par récurrence
Soit u_n le nombre de feuilles de type 2,
v_n le nombre de feuilles de type 2 pour un arbre critique uniforme de largeur l et de profondeur n.
Suppositions :
$u_0 = 0,\ v_0 = 0$
$\forall n > 0,\ u_n = lv_{n-1} + l - 1,\ v_n = u_{n-1}$
Démonstration :
$\Rightarrow \forall n > 2,\ u_n = lu_{n-2} + l - 1$
$\Rightarrow \forall n > 2,\ u_n + 1 = l(u_{n-2} + 1)$
$$\Rightarrow \forall n > 2,\ u_n = \begin{cases} l^{n/2} - 1 & \text{Si } n \text{ est pair} \\ l^{(n-1)/2} l - 1 & \text{Si } n \text{ est impair} \end{cases}$$
$\Rightarrow \forall n,\ u_n = l^{n/2} - 1$

Comptage des différents noeuds

Types de noeuds :

- type 1 : racine et fils ainés des type 1
- type 2 : cadets de type 1 et tous les fils de type 3
- type 3 : fils aÓnés des type 2
- inconnus : cadets des type 2 et les fils des inconnus

En résumé

On a donc compté

- $l^{n/2} - 1$ noeuds de type 2
- d'où $l^{(n-1)/2} - 1$ noeuds de type 3
- et 1 noeud de type 1

Soit, au mieux pour $\alpha\beta$ de l'ordre de :

$$l^{n/2} - 1 + l^{(n-1)/2} - 1 + 1$$

ou encore

$$l^{n/2}$$

noeuds

Résultats des performances

Encadrement des performances de $\alpha\beta$

Le nombre de feuilles évaluées par $\alpha\beta$ est compris entre $l^{p/2}$ et l^p pour un arbre de largeur l et de profondeur p

Nombre de noeuds visités

$\alpha\beta$ visite au minimum un nombre de noeuds de l'ordre de $l^{p/2}$

Comparaison MiniMax / $\alpha\beta$

Au pire, $\alpha\beta$ explore tous les noeuds de l'arbre. Au mieux, $\alpha\beta$ peut voir r un horizon deux fois plus lointain que MiniMax dans le méme temps.

L'ordre de développement des fils d'un noeud est primordial pour obtenir de bonnes performances !