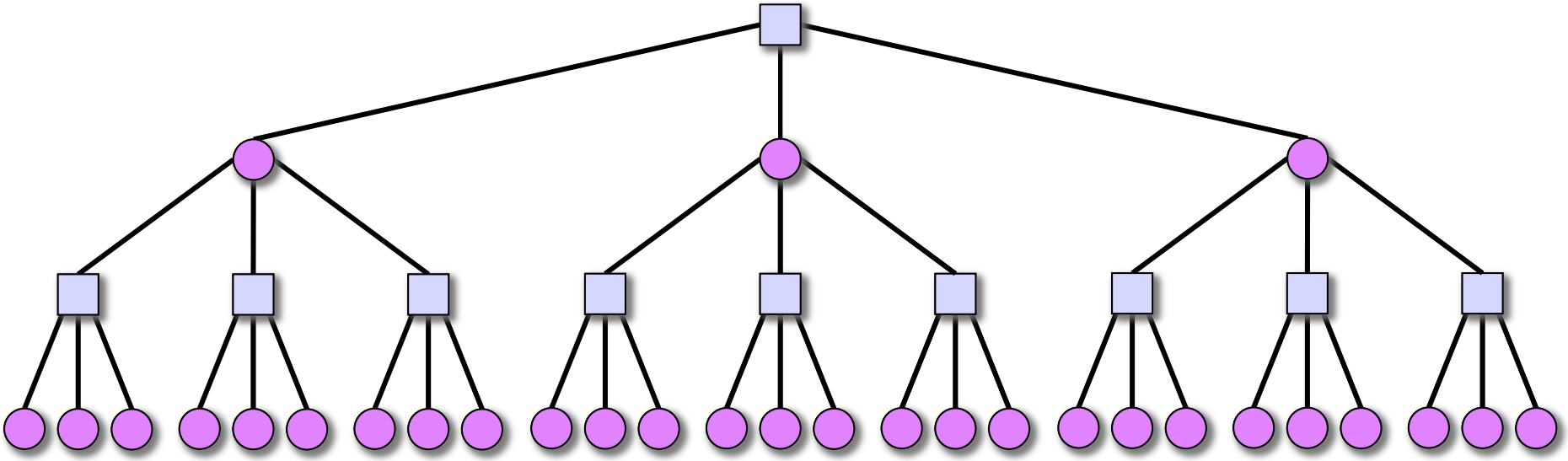
MinMax et Alpha-Beta

Min-Max



- Jeu à 2 joueurs: MAX ■ and MIN ●.
- L'arbre de jeux représente tous les coups possibles jusqu'à une certaine profondeur à partir de la position courante.
- Chaque feuille est évaluée avec une fonction d'évaluation.
- Un nœud MAX choisit un coup de score maximal parmi ses fils.
- Un nœud MIN choisit un coup de score minimal parmi ses fils.

Algorithme Min-Max
Version deux fonctions

Fonction ami

```
function ami (noeud : SITUATION ; niveau : integer ) : integer ;
var n : SITUATION ;
    l : LISTE_DE_COUPS ;
    c : COUP ;
    eval : integer ;
begin
    if niveau = NIVMAX then eval := FONCTION_D_EVALUATION (noeud) ;
    else begin
         l := LISTE_DES_COUPS_POSSIBLES (noeud, niveau) ;
         eval := -INFINI ;
         while COUP_POSSIBLE (l) do
         begin
             c := CHOISIR_COUP (l) ;
             n := ENGENDRER_SITUATION (noeud, c) ;
             eval := MAX (eval, ennemi (n,niveau + 1)) ;
             l := RETIRER_COUP (l, c) ;
         end
    end ;
    ami := eval
    end ;
```

Fonction ennemi

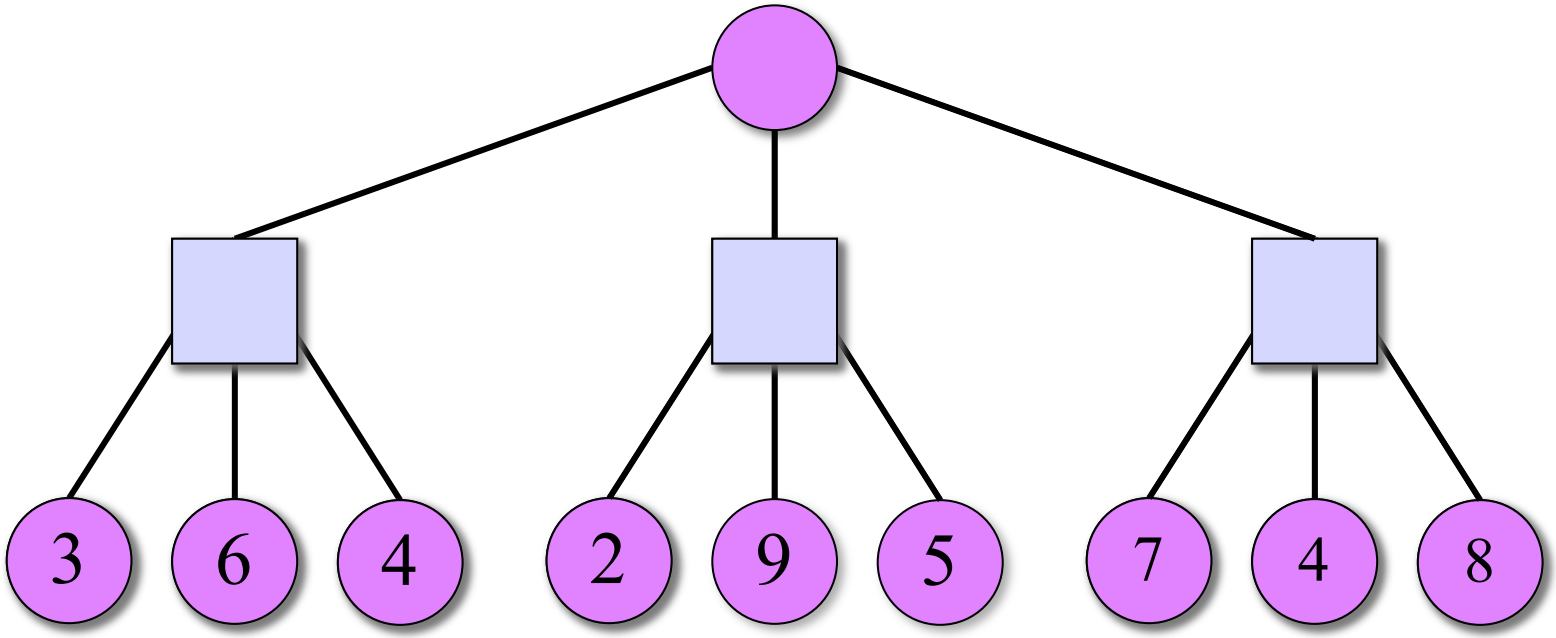
```
function ennemi (noeud : SITUATION ; niveau : integer ) : integer  ;
var n : SITUATION ;
    l : LISTE_DE_COUPS ;
    c : COUP ;
    eval : integer ;
begin
    if niveau = NIVMAX then eval := FONCTION_D_EVALUATION (noeud) ;
        else
        begin
           l := LISTE_DES_COUPS_POSSIBLES (noeud, niveau) ;
           eval := +INFINI ;
           while COUP_POSSIBLE (l) do
           begin
              c := CHOISIR_COUP (l) ;
              n := ENGENDRER_SITUATION (noeud, c) ;
              eval := MIN (eval, ami (n,niveau + 1)) ;
              l := RETIRER_COUP (l, c) ;
           end
        end ;
     ennemi := eval
end ;
```

Algorithme Min-Max

```
function evaluer (noeud:SITUATION ; niveau:integer ; i:integer ):integer ;
var n : SITUATION ;
    l : LISTE_DE_COUPS ;
    c : COUP ;
    eval : integer ;
    i : integer ;
begin
    if niveau = NIVMAX then eval := FONCTION_D_EVALUATION (noeud) ;
    else begin
       l := LISTE_DES_COUPS_POSSIBLES (noeud, niveau) ;
       eval := i * INFINI ;
       while COUP_POSSIBLE (l) do begin
           c := CHOISIR_COUP (l) ;
           n := ENGENDRER_SITUATION (noeud, c) ;
           eval := i * MIN (i * eval, i * evaluer (n,niveau + 1,-i)) ;
           l := RETIRER_COUP (l, c) ;
       end
    end ;
    evaluer := eval
end ;

begin
    evaluer (RACINE, 0, -1)
end .
```

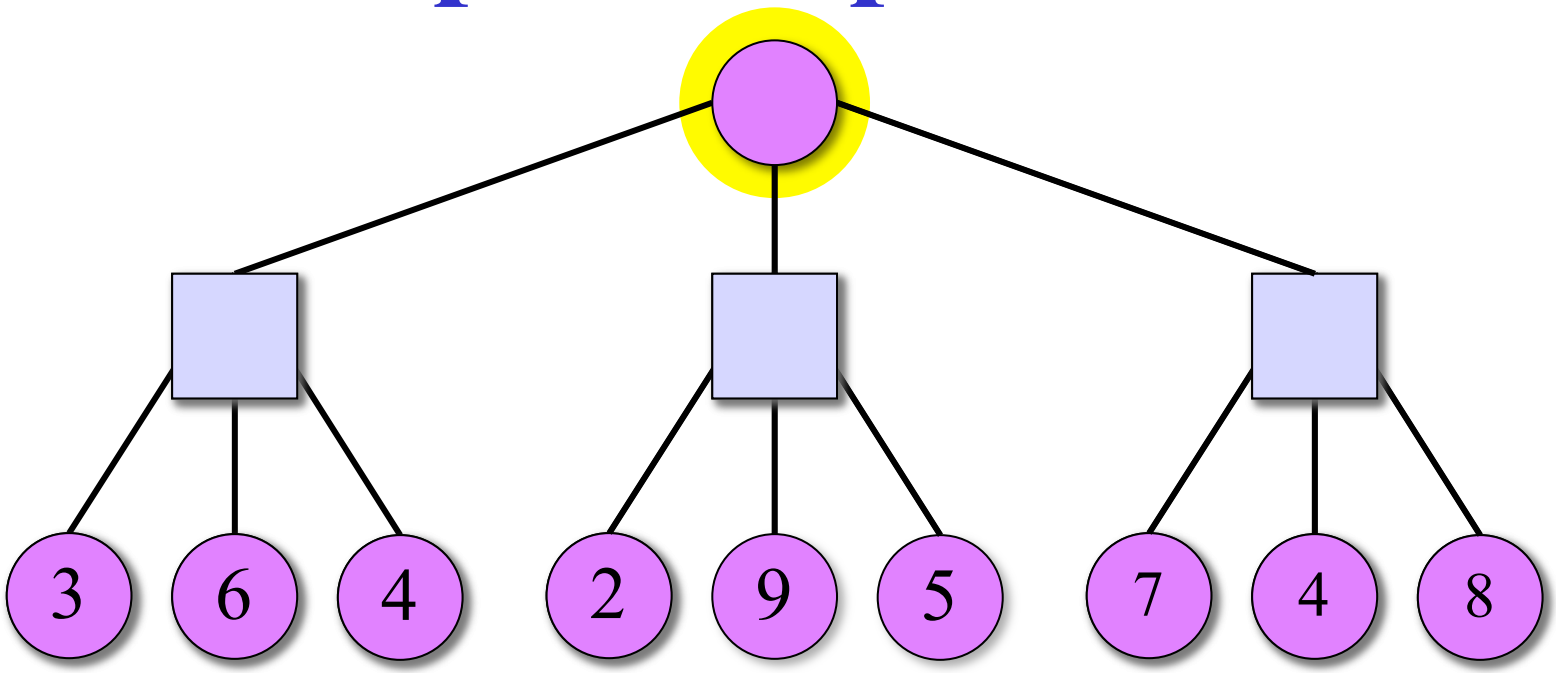
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

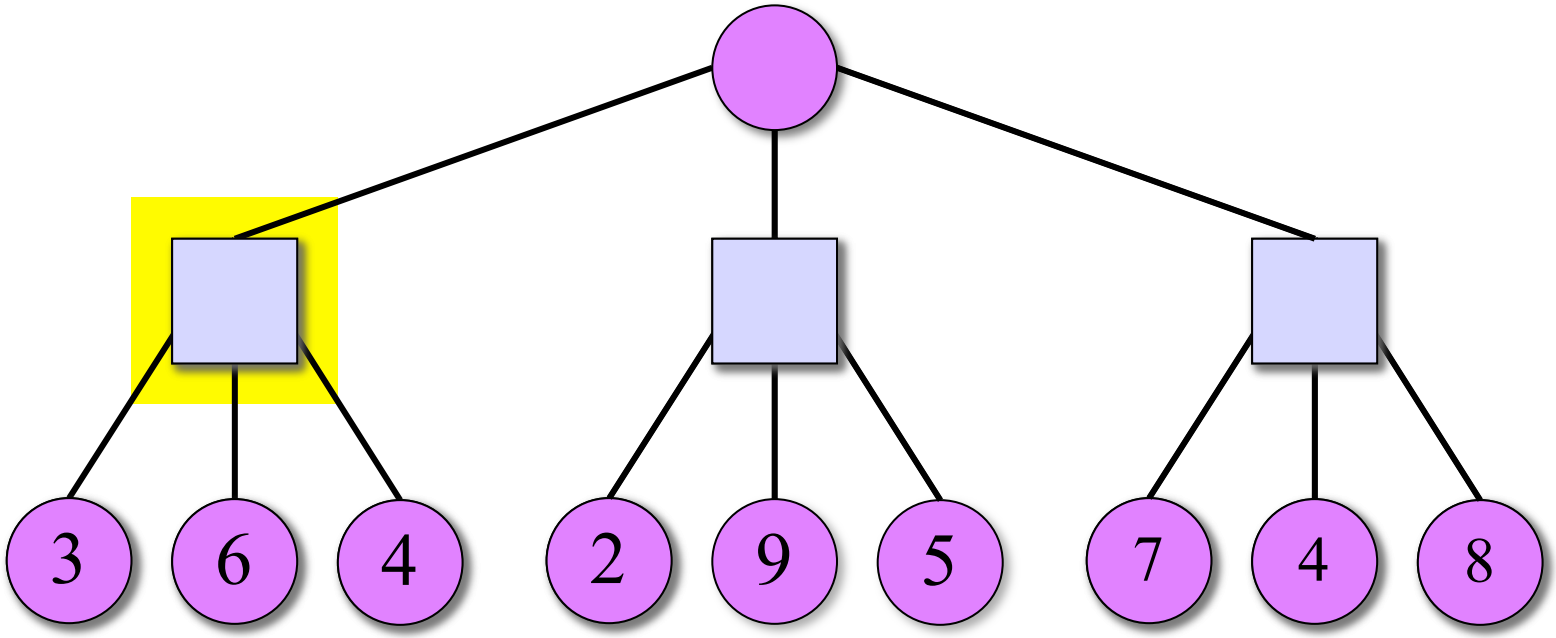
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

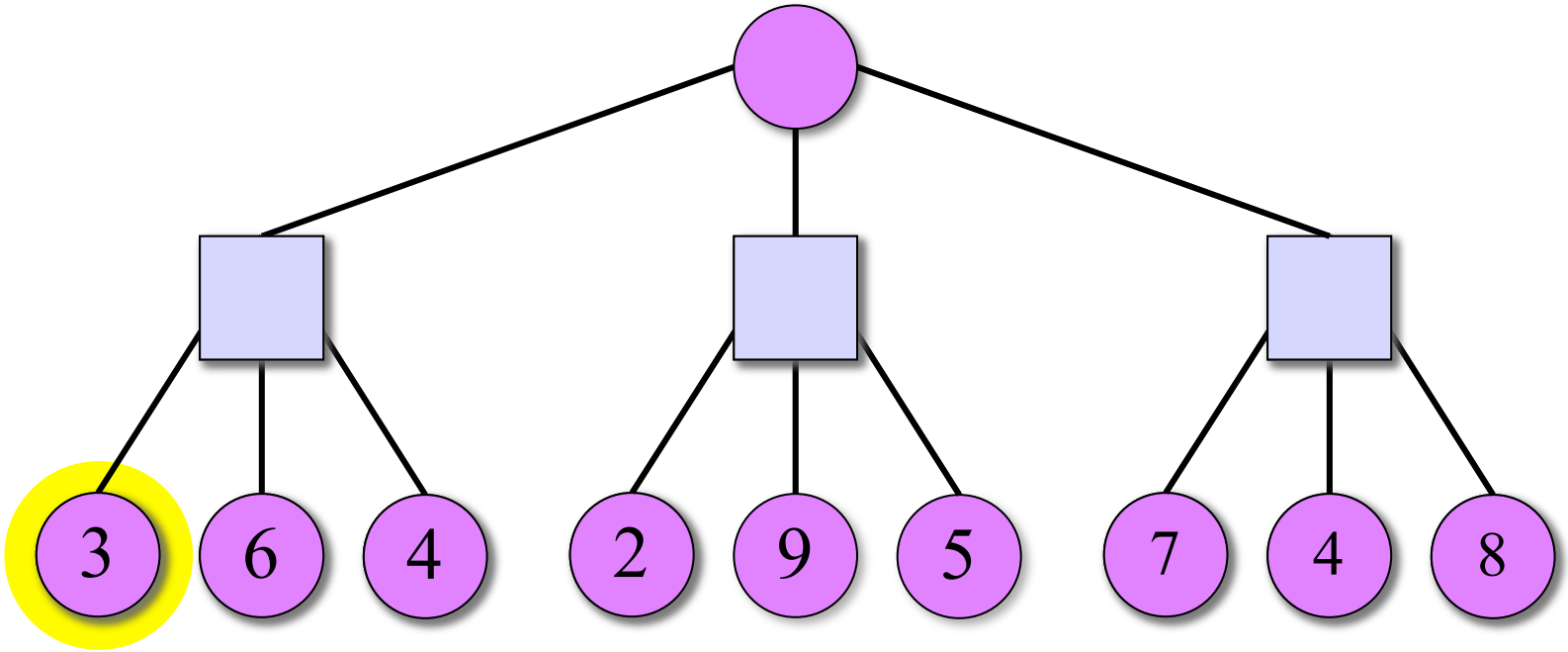
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

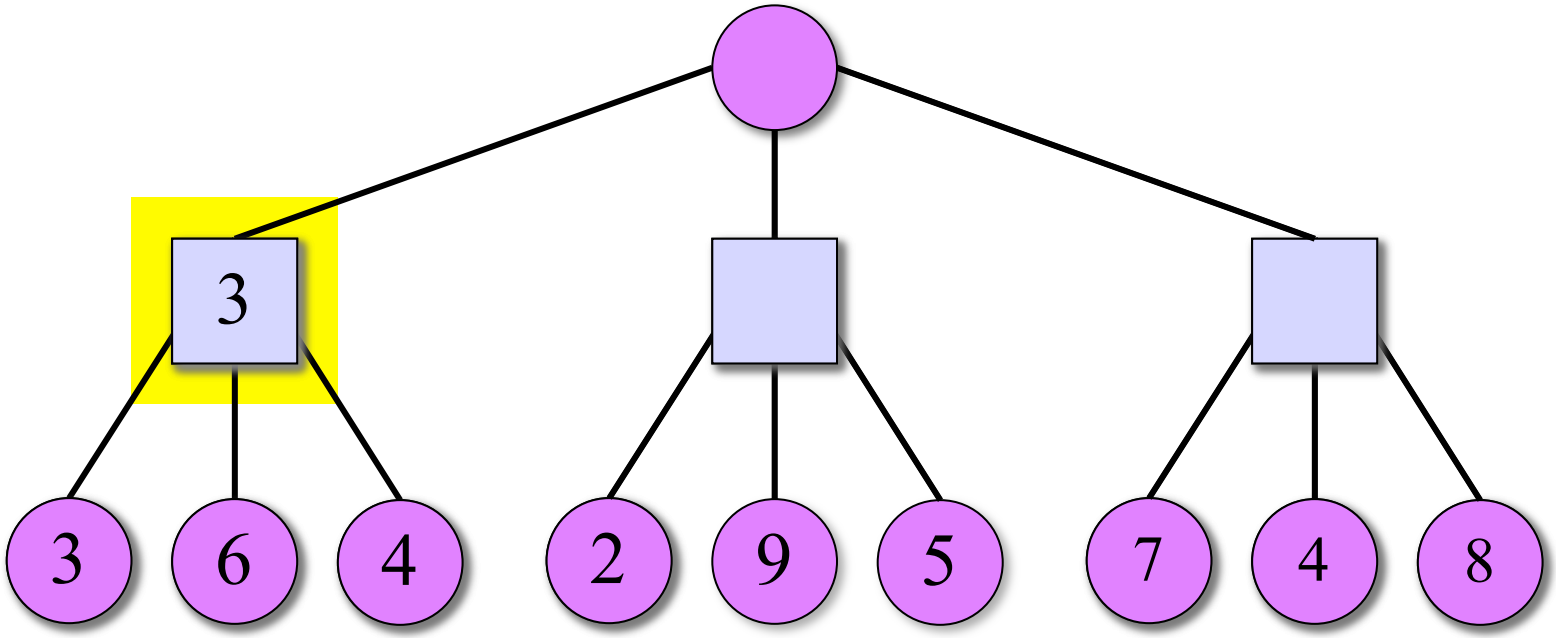
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

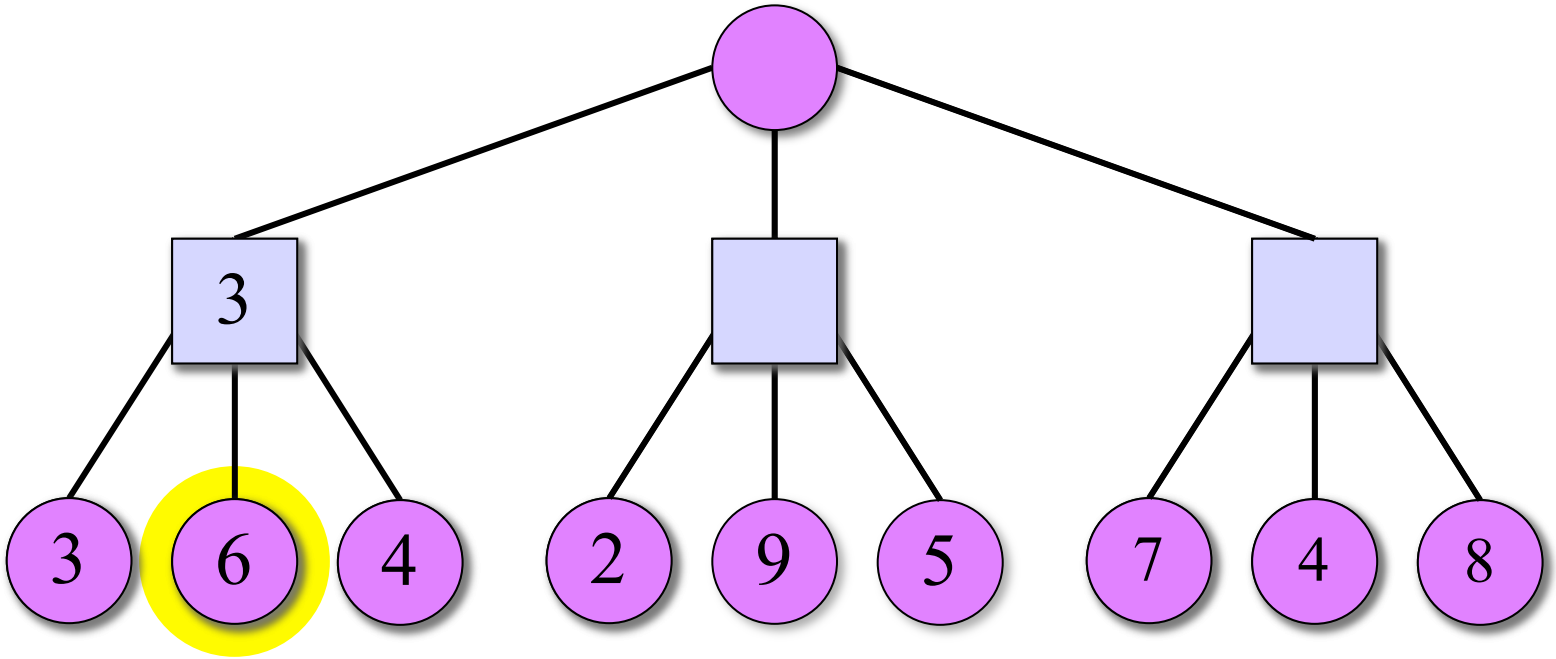
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

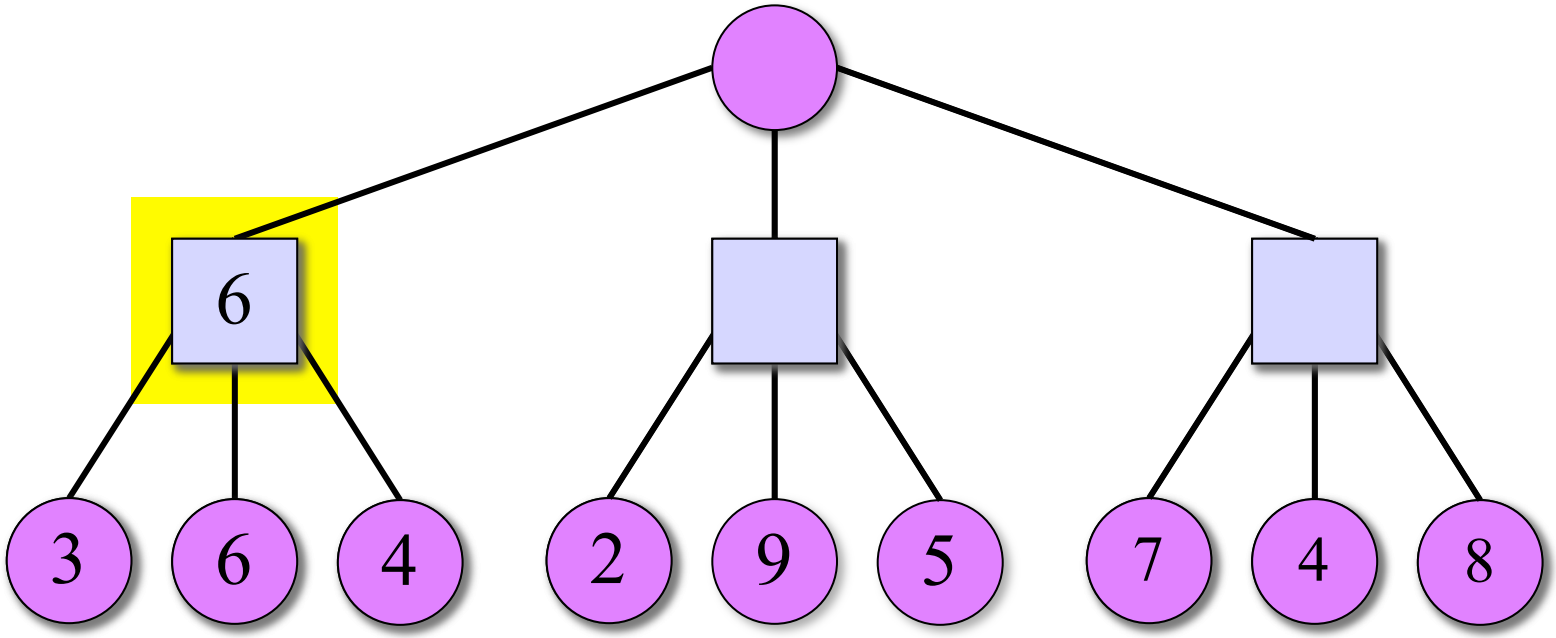
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

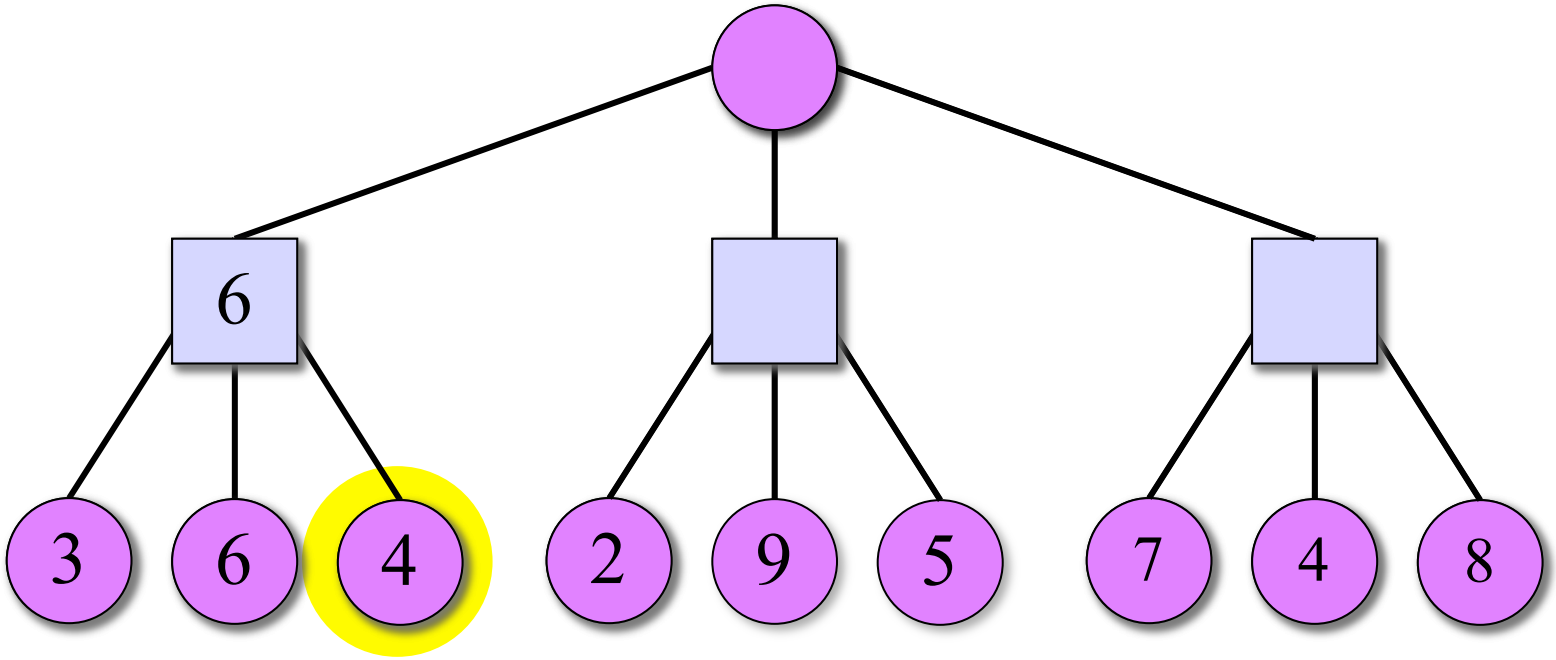
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

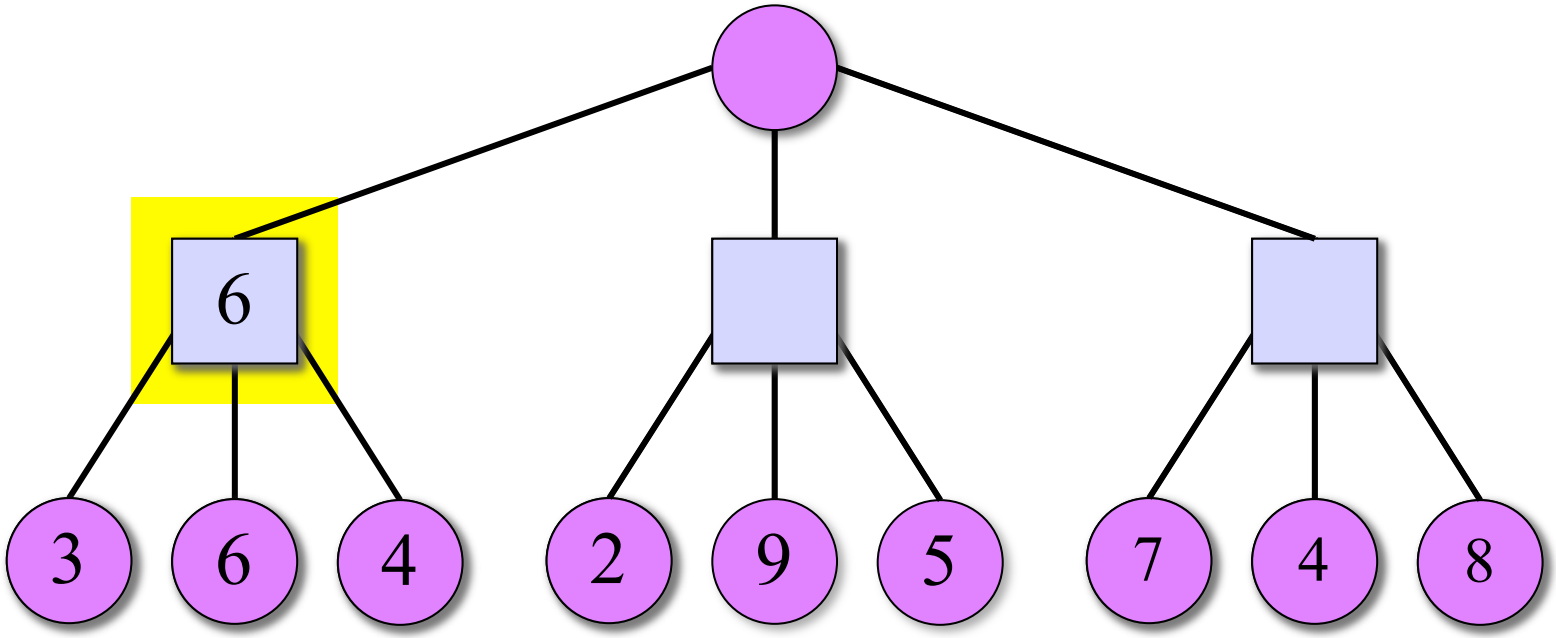
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

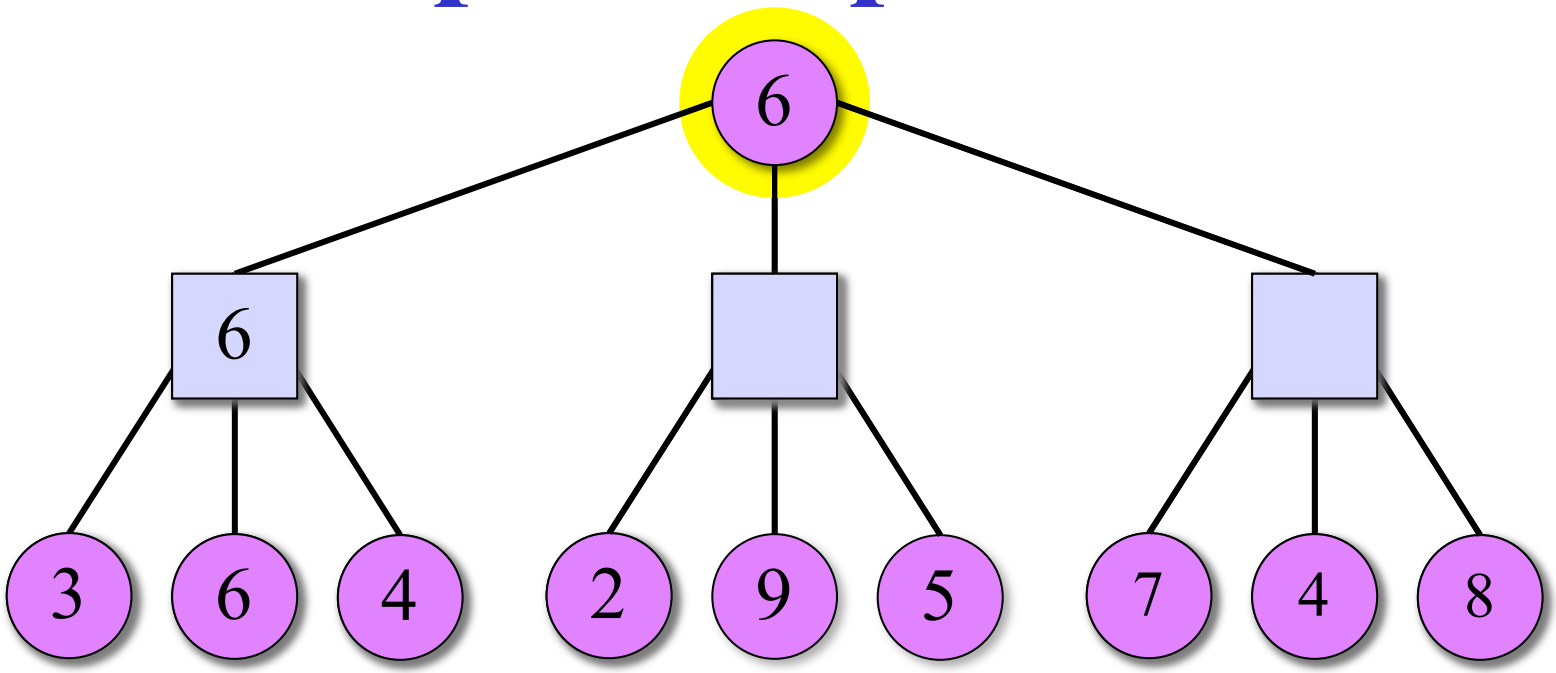
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

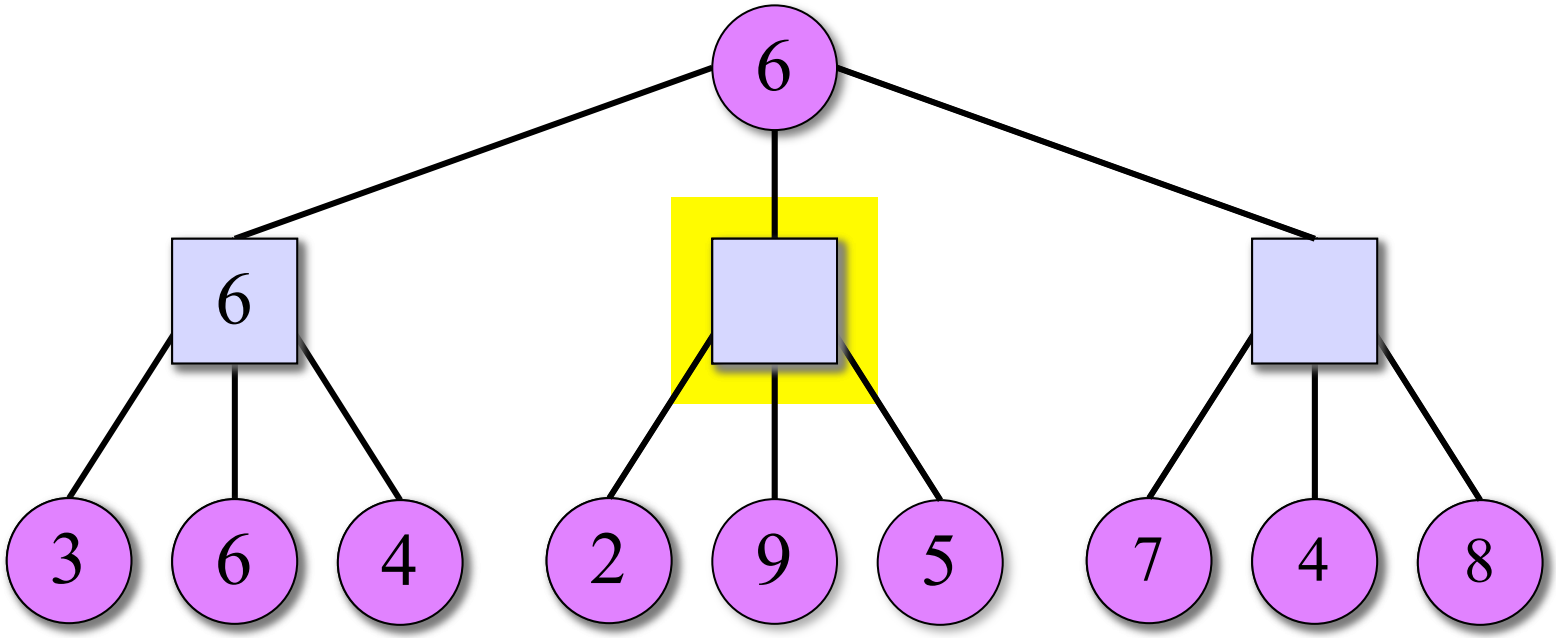
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

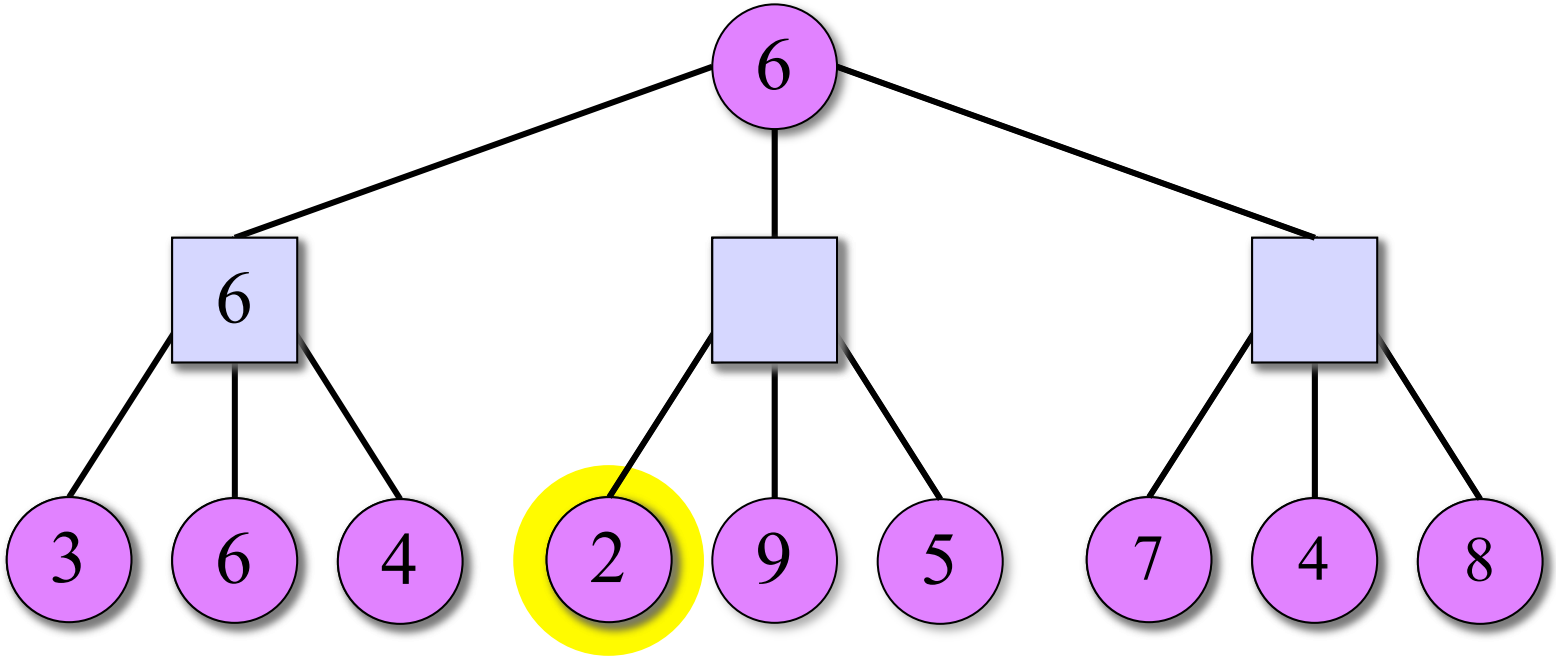
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

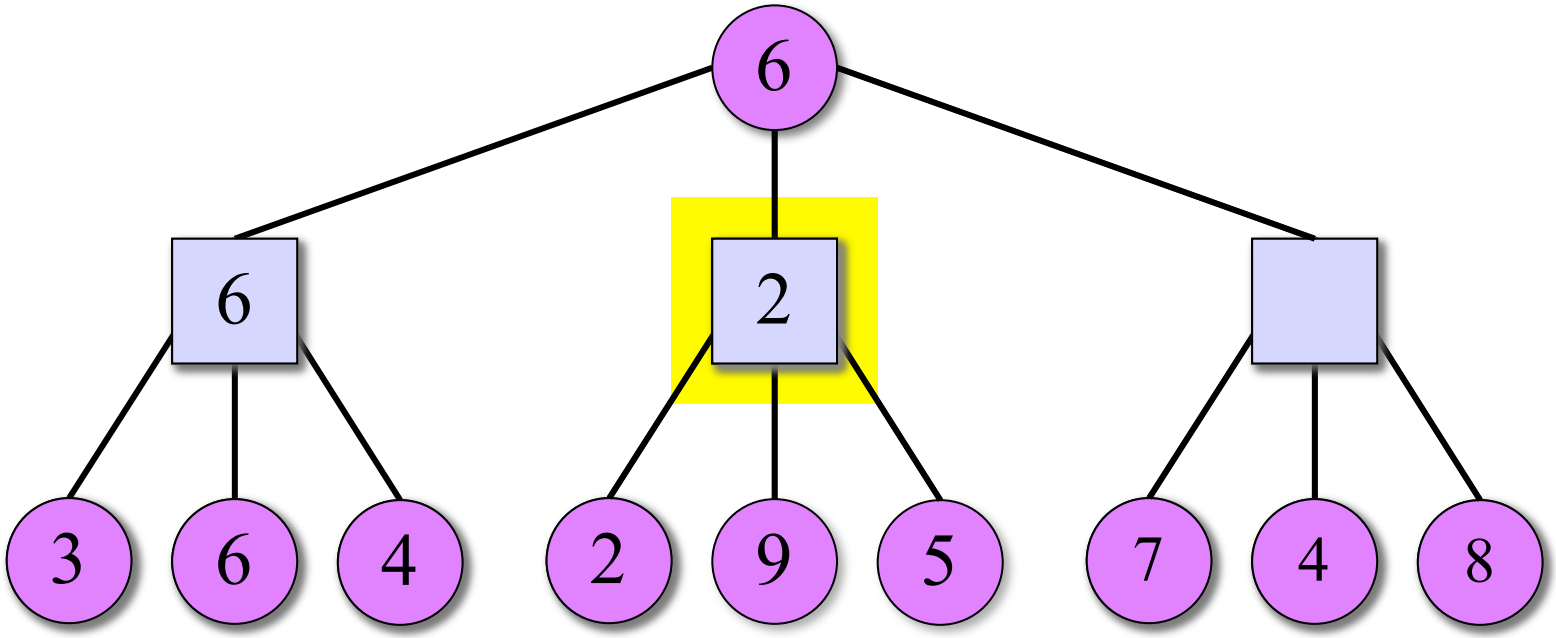
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

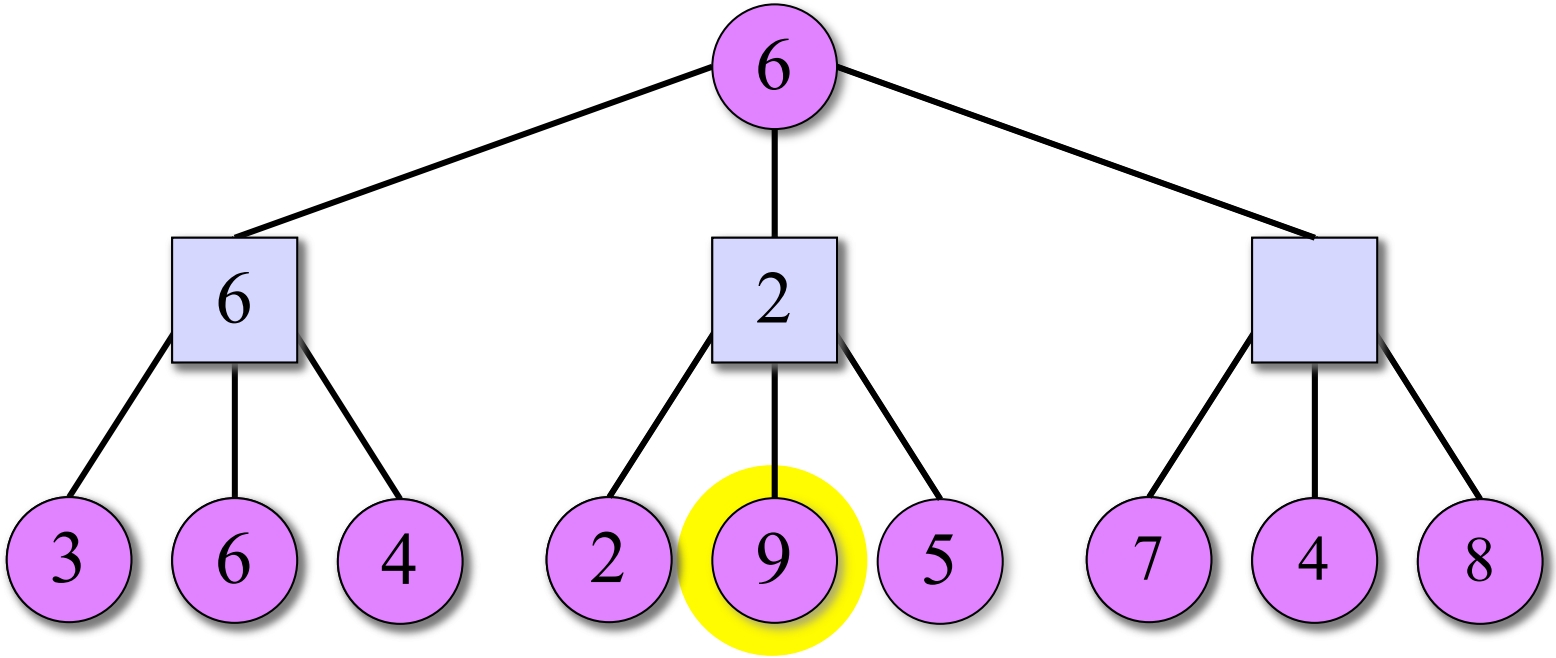
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

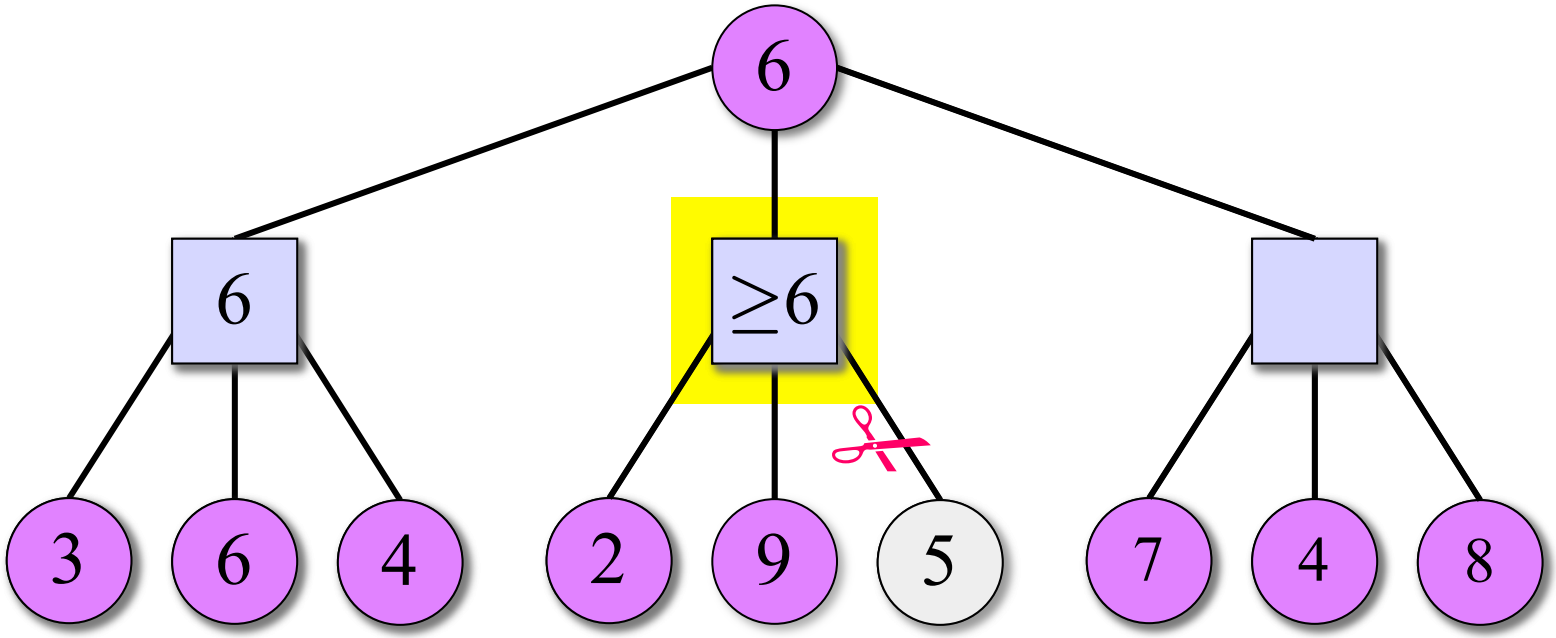
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

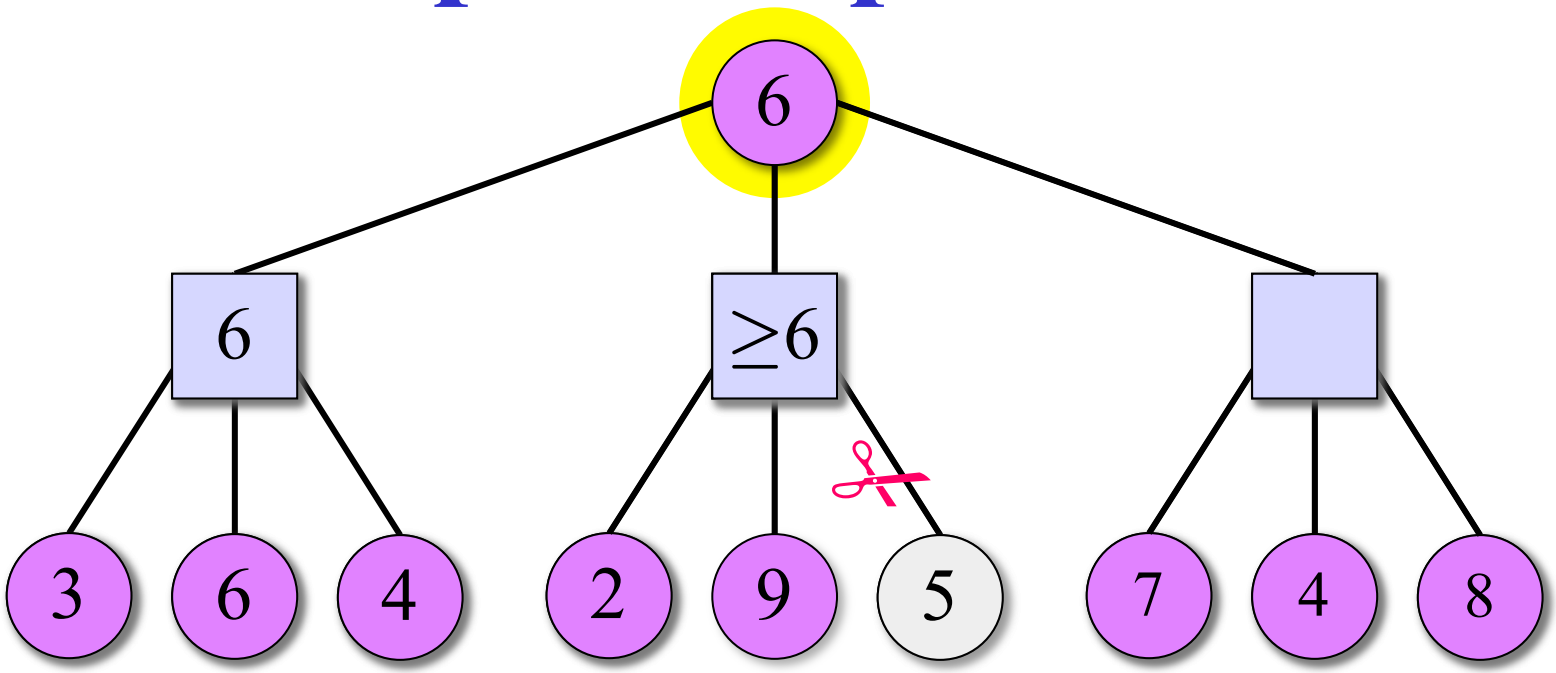
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

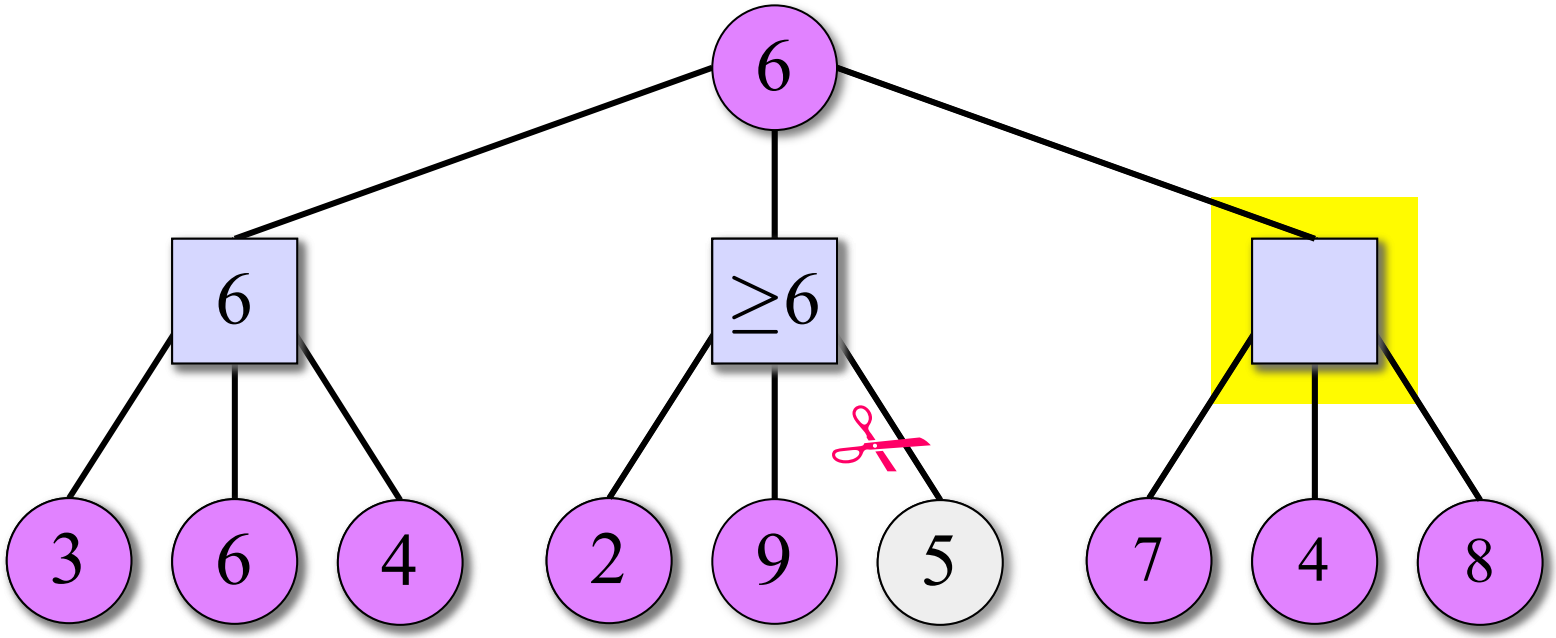
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

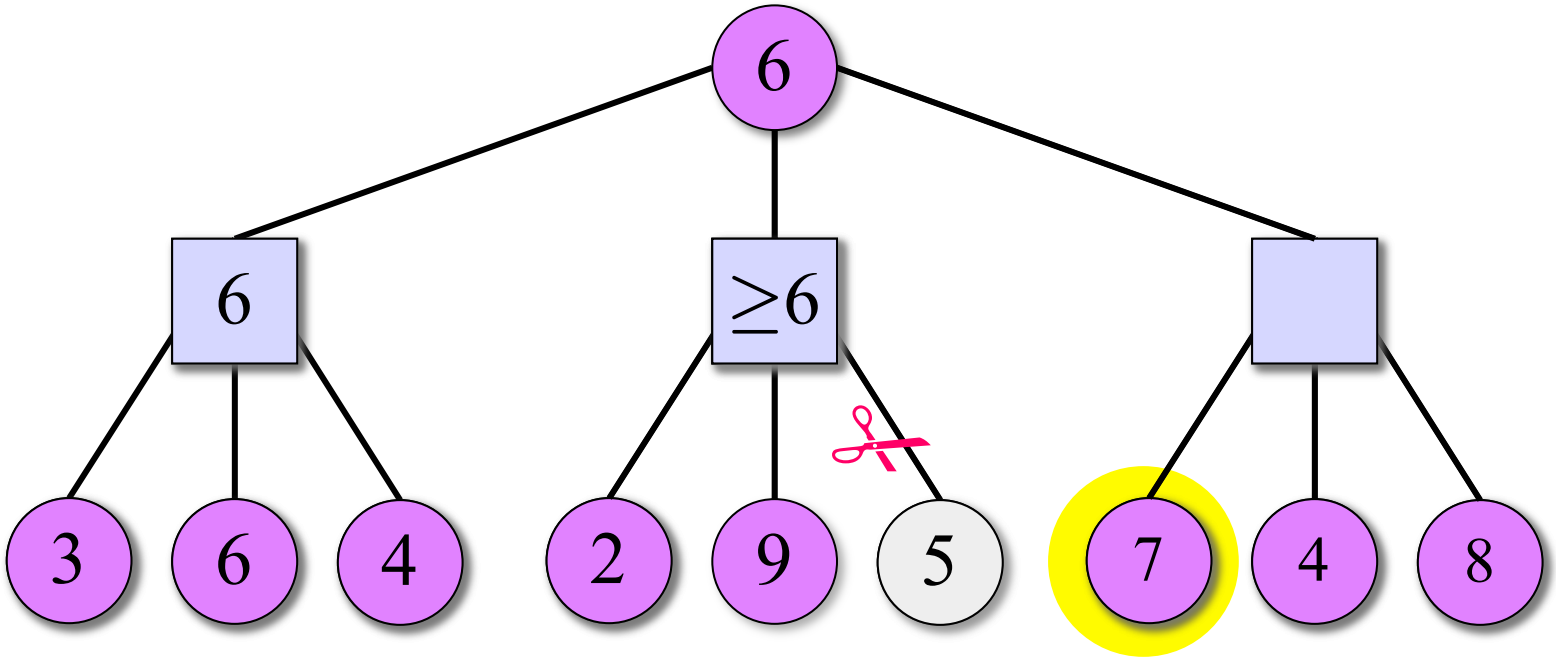
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

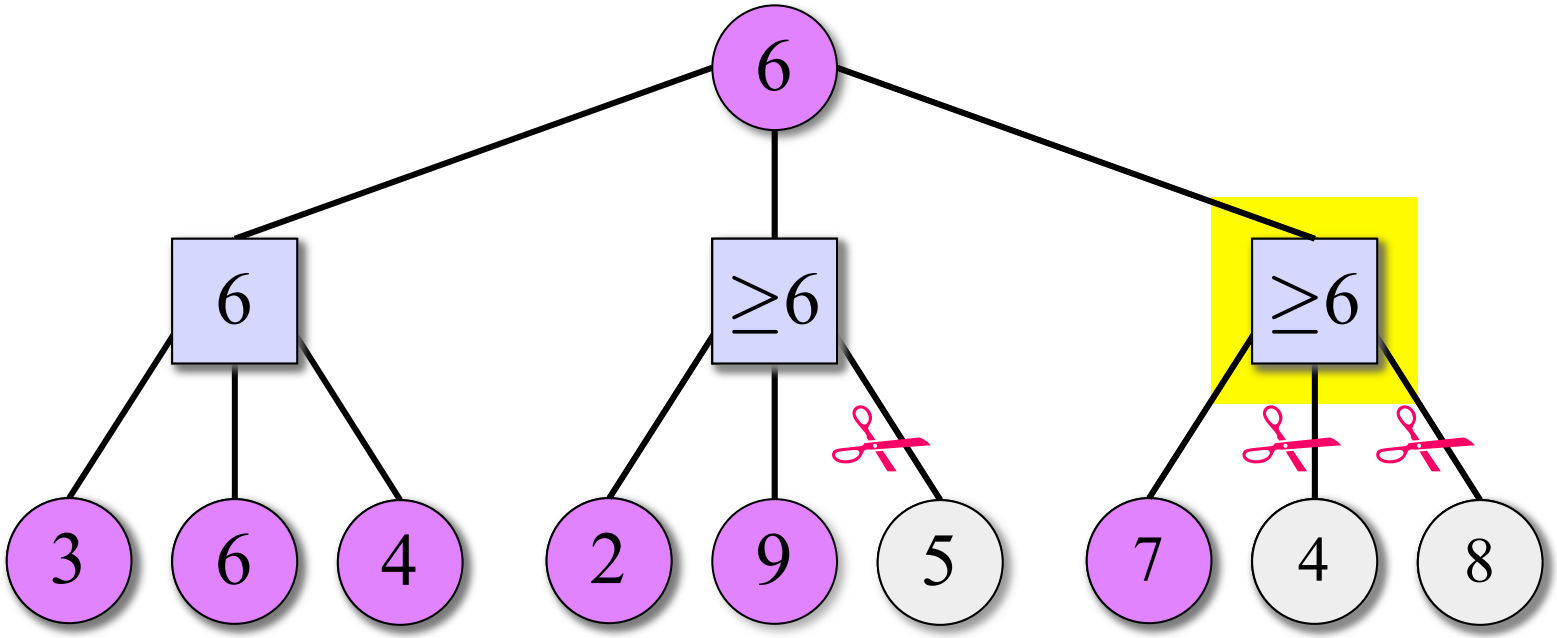
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

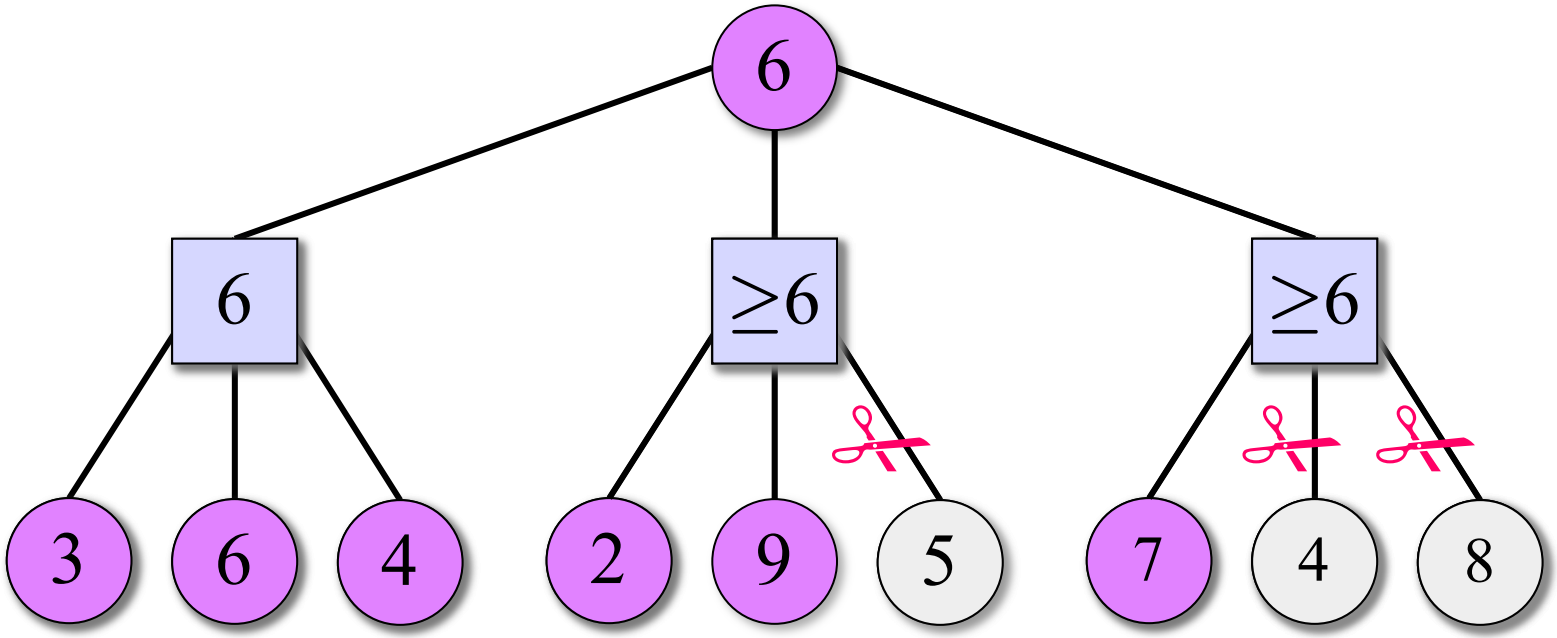
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

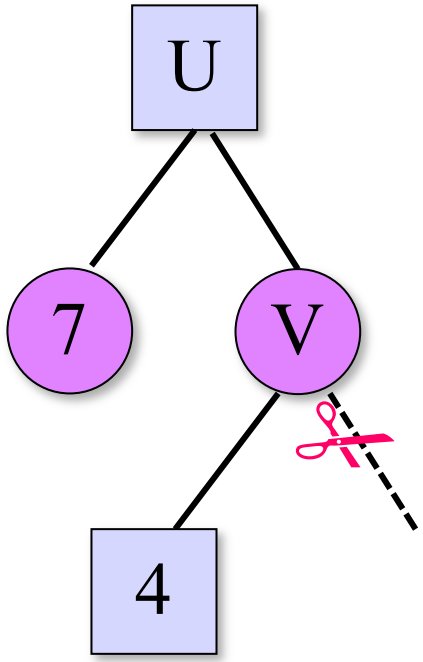
Coupure Alpha-Beta



Principe: si la valeur d'un fils *f* d'un nœud MAX ■ est supérieure à la valeur courante d'un nœud MIN ● ancêtre, alors les frères de *f* n'ont pas besoin d'être explorés :

coupure beta (***beta cutoff***).

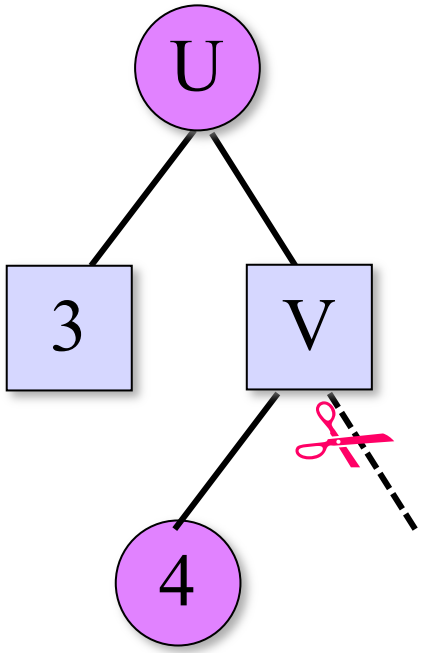
Coupure Alpha



Coupe α

Élagage fils d'un noeud MIN ●

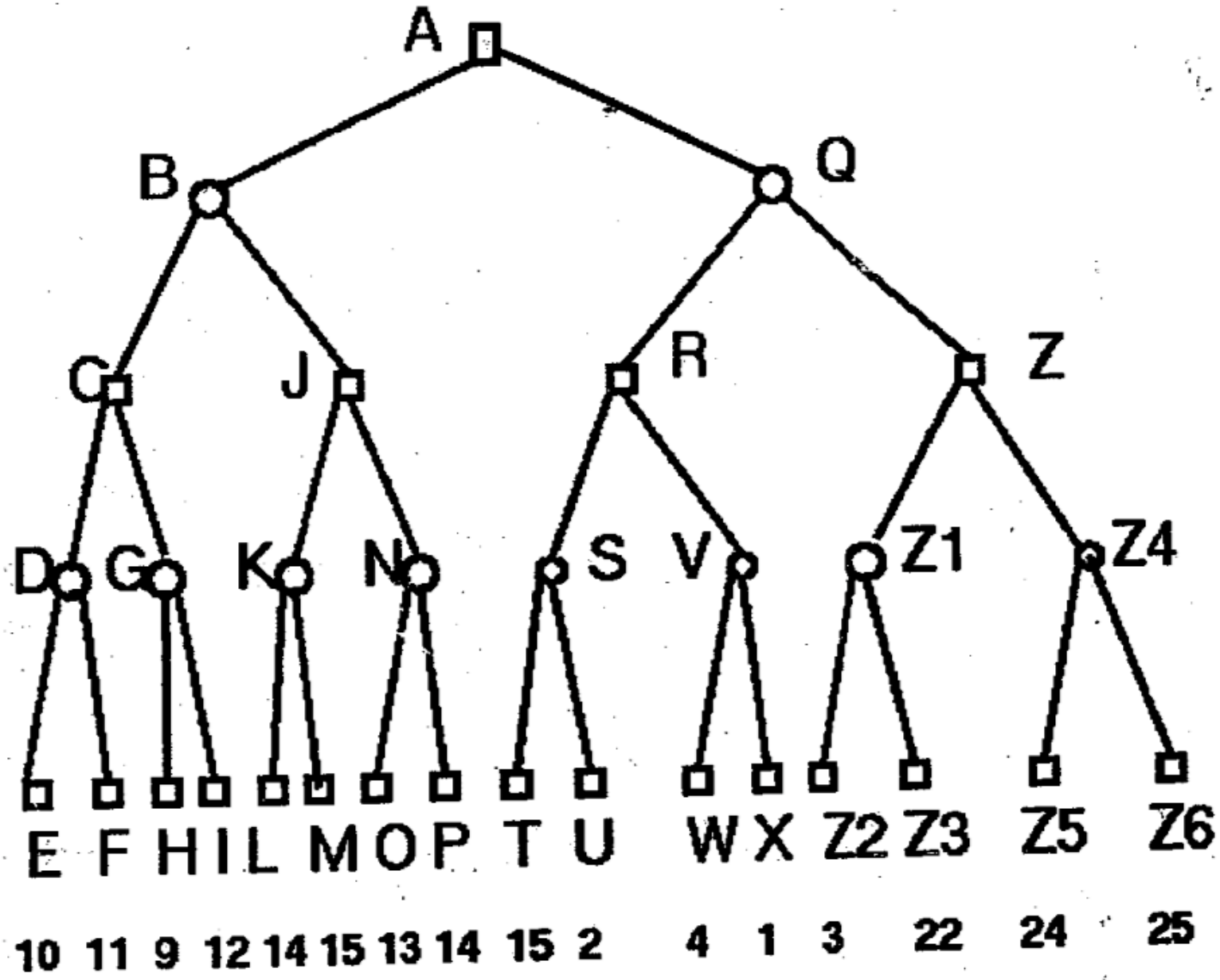
Coupure Beta

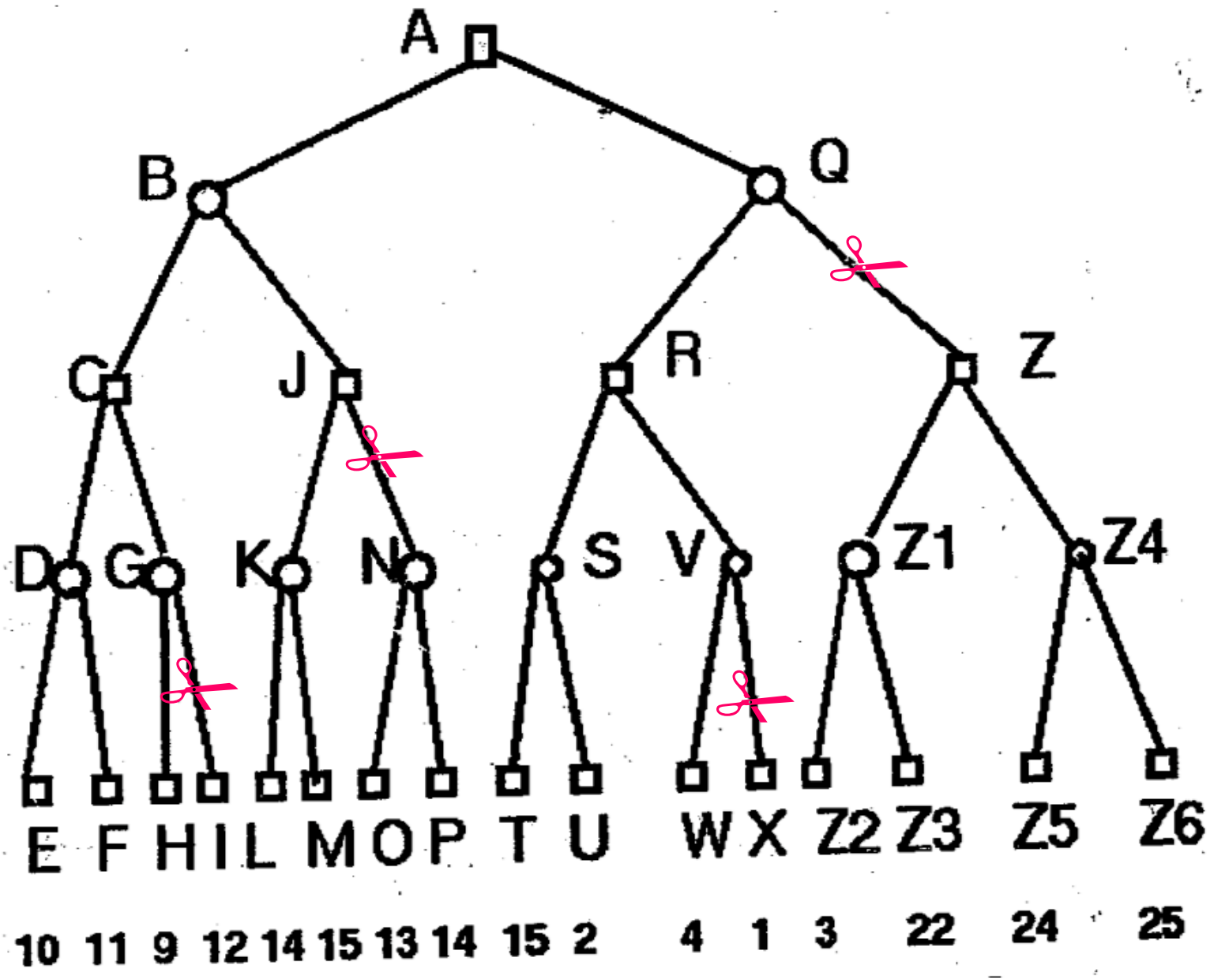


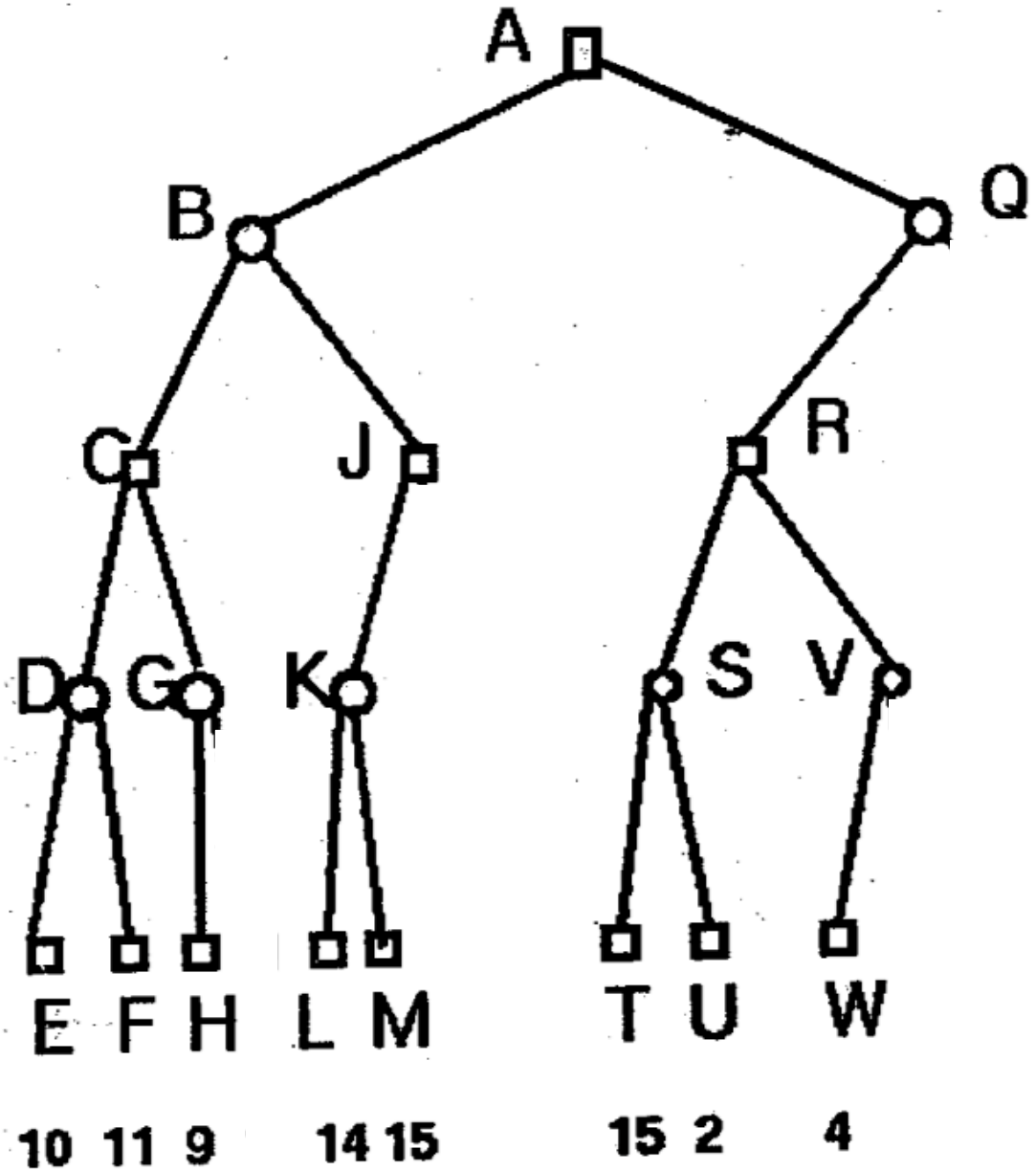
Coupe β

Élagage fils d'un noeud MAX ■

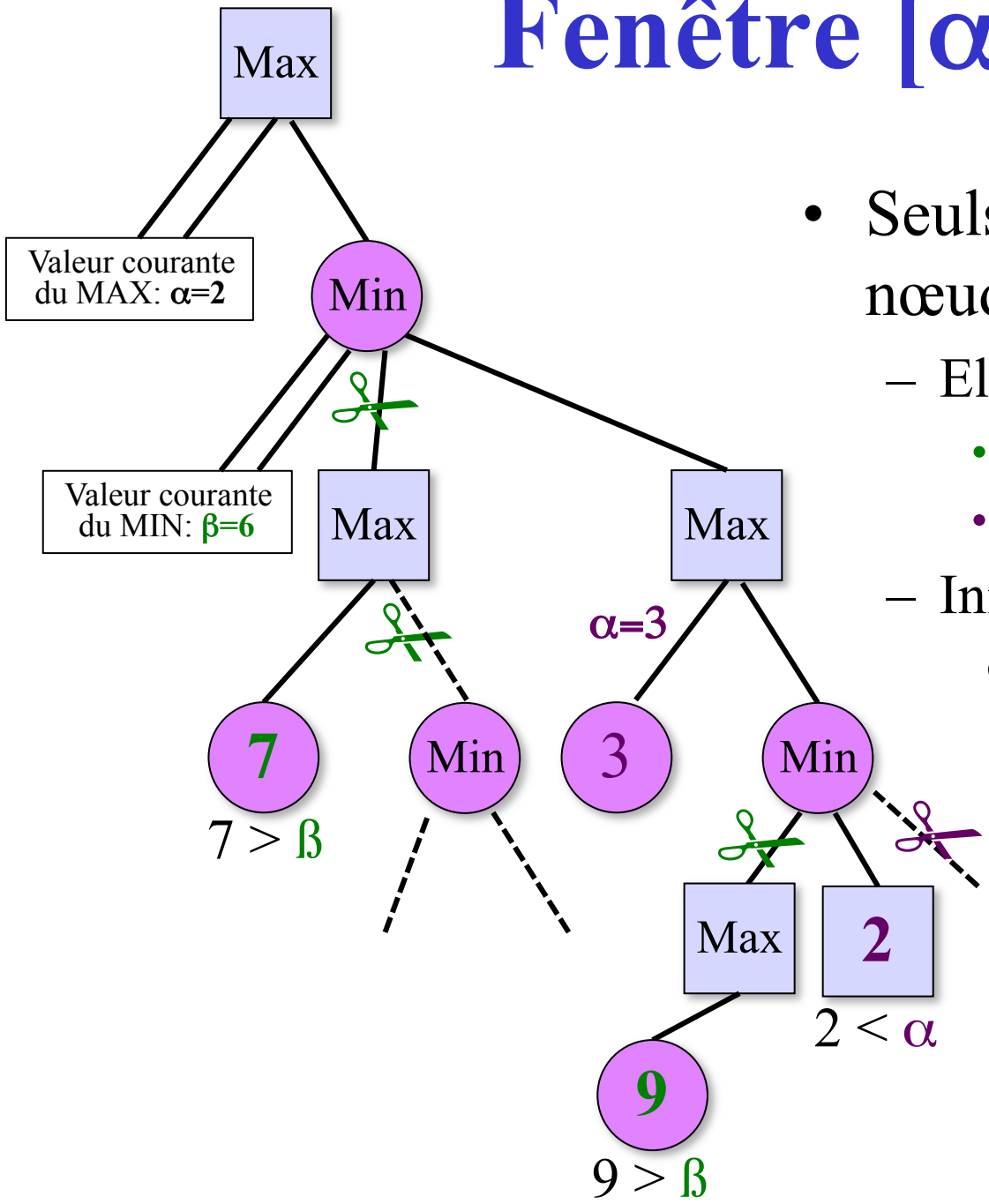
NB Les coupes peuvent être pronfondes.


A
B
Q
C
J
R
Z
D
G
K
N
S
V
Z1
Z4
E F H I L M O P T U W X Z2 Z3 Z5 Z6
10 11 9 12 14 15 13 14 15 2 4 1 3 22 24 25


A
B
Q
C
J
R
Z
D
G
K
N
S
V
Z1
Z4
E F H I L M O P T U W X Z2 Z3 Z5 Z6
10 11 9 12 14 15 13 14 15 2 4 1 3 22 24 25


A
B
Q
C
J
R
D
G
K
S
V
E
F
H
L
M
T
U
W
10
11
9
14
15
15
2
4

Fenêtre [α,β]



- Seuls sont considérés les nœuds de valeur dans [α,β] :
  - Elagage des nœuds
    - ✂ **MAX si valcour > β**
    - ✂ **MIN si valcour < α.**
  - Initialisation: α=-∞, β=+∞ ou alors *iterative deepening*

Algorithme alpha-beta

```
function MINIMAX-AB(N, A, B) is ;; A is always less than B
begin
   if N is a leaf then
      return the estimated score of this leaf;
   Set Alpha to A and Beta to B;
   if N is a Min node then
     For each successor Ni of N loop
       Set Beta to
         Min{Beta, MINIMAX-AB(Ni, Alpha, Beta)};
       When Alpha >= Beta then
         Return Alpha endloop;
     Return Beta;
   else
     For each successor Ni of N loop
       Set Alpha to
         Max{Alpha, MINIMAX-AB(Ni, Alpha, Beta)};
       When Alpha >= Beta then
         Return Beta endloop;
     Return Alpha;
end MINIMAX-AB;
```

Arbre ordonné optimal

- Alpha-beta est très séquentiel
- Pour maximiser les coupures, l'idéal serait que les fils d'un nœud MAX soient triés par ordre décroissant
  - A priori, l'arbre optimal est inconnu: tri « a priori »
- Lors de l'évaluation par alpha beta de l'arbre optimal:
  - Si le premier fils d'un MAX est coupé par le père MIN: le nœud MAX est coupé; les autres fils ne sont pas évalués
  - Sinon, tous les fils sont évalués (et on garde « le vrai » MAX)

Alpha-beta parallèle

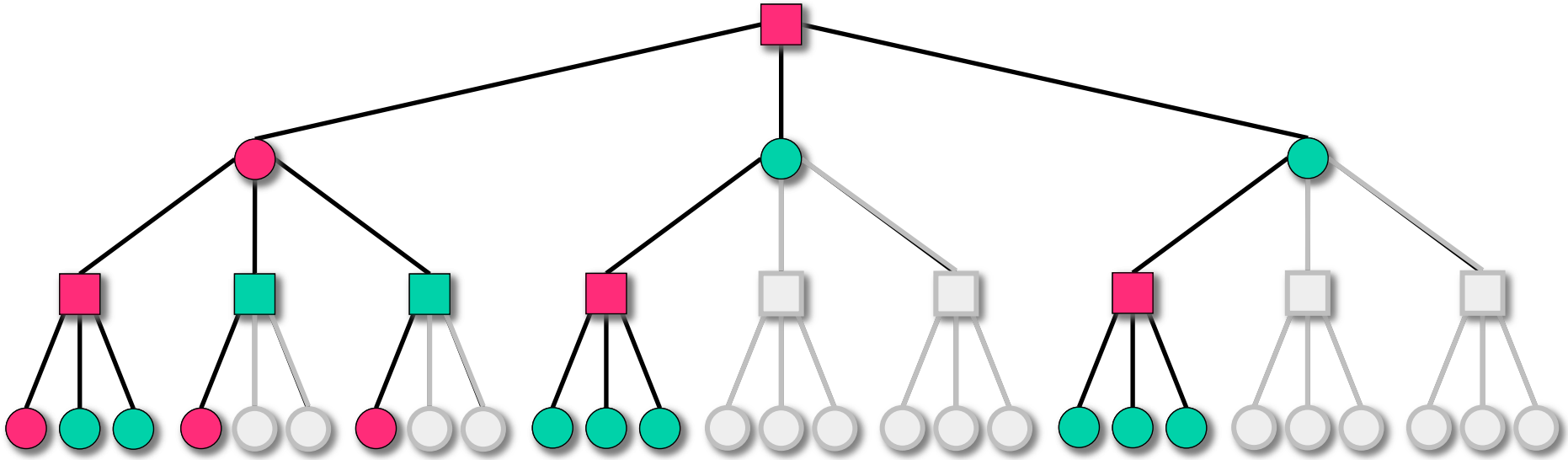
- *Rappel MIN-MAX et alpha-beta séquentiel*
- *Arbre optimal pour les coupures alpha-beta*
- *Parallélisation d'alpha-beta*
- *Implantation du tri des fils*

- *Parallélisation:*
  - *Utilisation de Intel-TBB*

- Ref: [Multithreaded programming in Cilk] Charles Leiserson, Avril 2007
  Intel Thread Building Block (version limitée à la parallélisation de for)

Alpha-Beta Parallèle

OBSERVATION: dans un arbre optimal, le degré de tout nœud interne est soit 1 soit ***maximal***.



Heuristique de la parallélisation: [Feldman-Mysliwietz-Monien 91]
Si le premier fils *f* d'un nœud ne permet pas de le couper, alors on ***spécule*** que tous les coups frères de *f* peuvent être explorés en **parallèle**,
en espérant ne pas explorer trop de nœuds supplémentaires par rapport à un alpha-beta séquentiel.

Parallélisation de alpha-beta

- *Réduction parallèle (produit itéré)*

```
 ParABEval (n) {
1      Si (profondeur limite atteinte)
2            return Eval(n);
3      Soit f[0, .., m-1] les m fils de n ;
4      res = ParABEval( f[0] ) ;
5      si ( n n'est pas élagué ) alors // EN PARALLELE
6        res = MAX/MIN (res, ParABEval(f[i]), i=1.. m-1) ;
7      return res ;
```

- Parallélisation de la boucle 6: de type « produit itéré »
- L'arbre est dynamique et d'arité variable
  - *Découpe récursive de la boucle (1 .. m-1) jusqu'à un « certain » seuil (granularité)*
  - *Ordonnancement de type « glouton » / work-stealing*

Comment trier les fils

Stocker les coups possibles dans un tableau/vector,

triés par ordre de priorité:
plusieurs choix possibles (de difficulté croissante):

- **simple**: utilisation de la fonction d'évaluation des fils
- utiliser une évaluation plus précise (mini alpha-beta)
- coupler avec iterative deepening …
- …

Remarque: On peut aussi arrêter l'alpha-beta parallèle à une profondeur donnée, où le nœud est évalué avec un Alpha-beta séquentiel.

Implantation de la parallélisation

- *Au choix:*
  - *Soit « à la main » avec threads Posix*
    - *par exemple: découpe statique de l'arbre en sous-arbre;*
    - *ou découpe récursive dynamique de la boucle 6 [cf SEPC]*
  - *Soit en utilisant une **bibliothèque** qui intégre un ordonnancement par **work-stealing** en **C++**:*
    - *Cilk Arts (... en 2008 ...)*
    - *INRIA Kaapi: en contexte distribué*
      cluster / grille avec nœuds hétérogènes SMP/multicore
      http://kaapi.gforge.inria.fr/
    - Le plus simple: ***Intel TBB*** http://threadingbuildingblocks.org/
      - contexte SMP/multicore

Parallélisation boucle 6 en TBB

- *Intel TBB : librairie C++ qui fournit un interface parallèle intégrant la plupart des fonctions de la STL ainsi que du parallélisme « fonctionnel »*

- *Ici: boucle 6 = produit itéré = « parallel_reduce »*

  - *Impléménté par Intel TBB : cf doc section 3.3*
  - le tutorial sur TBB est sur le kiosk (p.21/68)
  - et aussi accessible http://threadingbuildingblocks.org
  - http://pages.cs.wisc.edu/~gibson/tbbTutorial.html
- *Attention: ne pas installer Intel TBB*
  - *la librairie a dû être modifiée pour ensibm*