Jeux

```
1 function ALPHA_BETA(s) returns an action
2     v ← MAX_VALUE(s,-∞,+∞)
3     return a ∈ actions(s) with value v
```

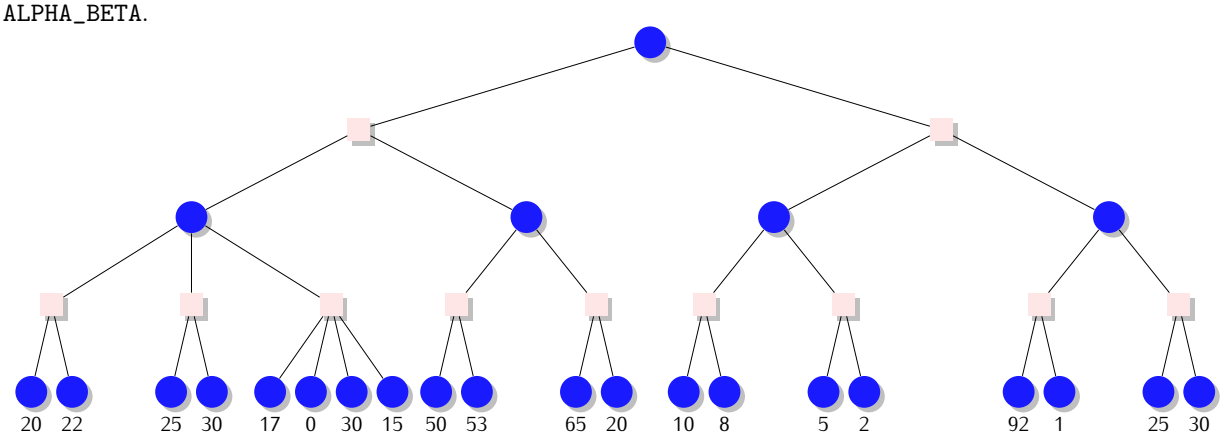
- `result(s,a)` est l'état atteint lorsqu'on prend l'action a dans l'état s
- α valeur de la meilleure (plus haute) valeur jusqu'ici pour `max`
- β valeur de la meilleure (plus basse) valeur jusqu'ici pour `min`

```
1 function MIN_VALUE(s,α,β) returns a utility value
2     if terminal?(s) then return utility(s)
3     v ← +∞
4     for each a ∈ actions(s) do
5         v ← min{v,MAX_VALUE(result(a,s),α,β)}
6         if v ≤ α then return v
7         β ← min{β,v}
8     return v
```

```
1 function MAX_VALUE(s,α,β) returns a utility value
2     if terminal?(s) then return utility(s)
3     v ← −∞
4     for each a ∈ actions(s) do
5         v ← max{v, MIN_VALUE(result(a,s),α,β)}
6         if v ≥ β then return v
7         α ← max{α,v}
8     return v
```

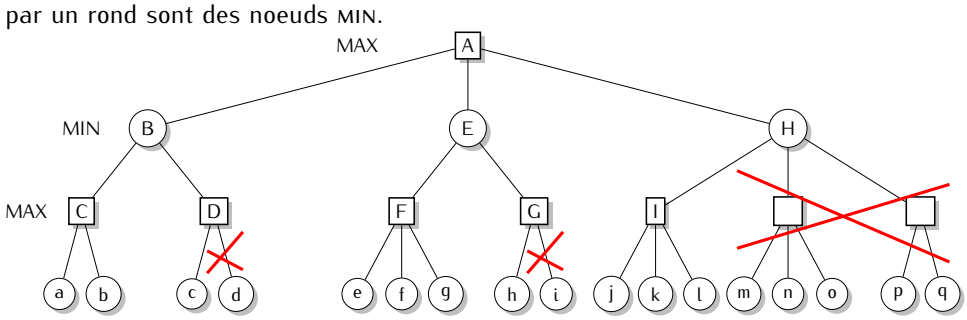
Exercice 1

Appliquez les algorithmes `MINIMAX` et `ALPHA_BETA` à l'arbre de jeu ci-dessous où les cercles sont des noeuds `MAX`, les carrés sont des noeuds `MIN`. Montrez les valeurs de α, β et v pour chaque noeud lors de l'exécution de `ALPHA_BETA`.



Exercice 2

Considérons l'arbre de jeu suivant où les noeuds représentés par un carré sont des noeuds MAX et ceux représentés par un rond sont des noeuds MIN.



- Donnez des valeurs aux feuilles de sorte que l'algorithme α-β coupe *exactement* les branches indiquées.
- Appliquez l'algorithme sur l'arbre avec vos valeurs.