

Projet - Puissance 4



Enoncé du projet

Le puissance 4 est un jeu constitué d'une grille de 7 colonnes et 6 lignes. Les deux joueurs qui s'affrontent possèdent chacun une couleur (rouge ou jaune) et font tomber des pions de leur couleur dans les colonnes du jeu. Les pions s'empilent ainsi petit à petit les uns sur les autres, et le but est d'aligner, à l'horizontal, à la verticale ou en diagonal, 4 pions. Le premier joueur à effectuer un alignement remporte la partie. S'il n'y a plus de place dans la grille et que personne ne parvient à faire un alignement, la partie est déclarée nulle. Les joueurs n'ont pas le droit de passer leur tour.

Première partie : jeu à deux joueurs

Dans un premier temps, vous devrez mettre en place les bases du jeu en mode deux joueurs. Vous devrez, dans cette partie :

- Trouver une structure de données pour représenter le grille de jeu, et expliquer comment vous représenterez si une case est libre, ou occupée par un pion d'une des deux joueurs. Même s'il n'est pas demandé, dans ce projet, de proposer des grilles d'une taille différente de celle donnée en introduction, il faudra que, dans votre code, vous proposiez une solution qui puisse facilement s'adapter à différentes dimensions de grille.
- Proposer une fonction permettant d'afficher, dans le terminal, la grille de jeu de manière claire.
- Proposer une fonction permettant de tester, après chaque coup d'un joueur, si ce dernier a gagné ou si la partie se termine par un match nul. Pour vous aider, sachez que, lorsqu'un joueur joue un pion dans la grille, il n'est pas nécessaire de tester toutes les cases de la grille pour les conditions de victoire : seuls les alignements qui contiennent la case qui vient d'être jouée doivent être testés.
- N'oubliez pas de vérifier, quand un joueur joue, que son mouvement soit autorisé (que la colonne n'est pas déjà remplie de pions).
- Proposer un menu permettant de lancer la boucle de jeu, permettant aux deux joueurs de jouer tour à tour, jusqu'à ce qu'un vainqueur soit trouvé ou que la partie de termine sans gagnant.

Seconde partie : jeu à un joueur, l'algorithme minimax

Une fois le jeu à deux joueurs prêt, il vous faudra proposer un mode un joueur contre une IA, qui fonctionnera comme le mode deux joueurs, mis à part que le second joueur sera joué par l'ordinateur. Pour programme une IA compétitive, il vous faudra mettre en place l'algorithme minimax dont on va décrire le fonctionnement ci-après.

L'algorithme minimax, principe de fonctionnement

Imaginons, pour simplifier les illustrations et les explications de l'algorithme, que notre grille de puissance 4 contienne uniquement 2 colonnes et 8 lignes, que le joueur humain soit de couleur rouge, et l'IA soit de couleur bleue. Imaginons également. Imaginons que la grille soit dans l'état représenté à la Figure 1, et que ce soit maintenant au tour de l'IA de jouer. Imaginons également que nous ayons une

fonction permettant d'associer, à chaque état de la grille, un score permettant de savoir à quel point l'état de la grille est avantageuse pour le joueur IA : plus le score est élevé, meilleure est l'état courant du jeu pour l'IA.



FIGURE 1 – L'état de la grille depuis lequel on commence pour l'exemple de l'algorithme minimax.

Algorithme minimax, étape 1/2 - développer les coups

Nous commençons par la configuration actuelle de la grille, et imaginons tous les coups possibles que l'IA peut jouer. On voit que l'IA peut jouer soit dans la colonne 1, soit dans la colonne 2 (à chaque fois, il n'y a que deux coups possibles dans cet exemple, car nous avons volontairement simplifié la grille - dans la réalité, la grille ayant 7 colonnes, il y aura 7 coups possibles). On construit, depuis cette configuration, deux nouvelles configurations représentant les deux coups possibles que l'IA peut jouer (pions bleus), et on obtient le résultat de la Figure 2.

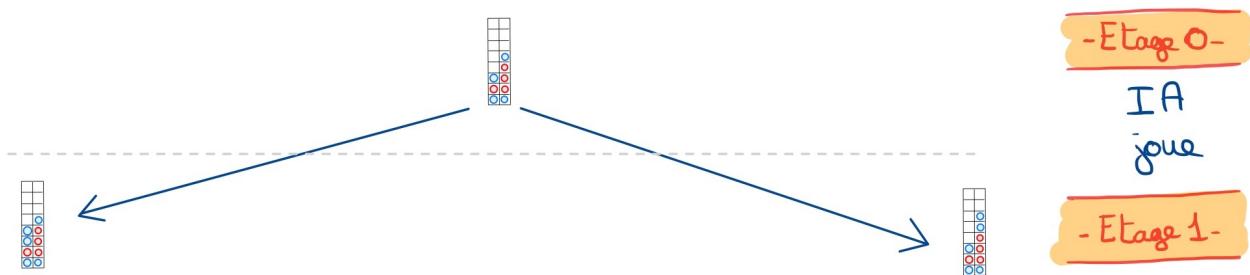


FIGURE 2 – Illustration après avoir développé deux coups jouables par l'IA.

Nous partons alors des deux nouvelles configurations, et développons tous les coups que le joueur humain (pions rouges) pourrait jouer à partir de là, ce qui nous donne la Figure 3.

Quelle est cette structure de données ? On voit sur la Figure 3 apparaître une nouvelle structure de données qui ressemble à une liste où chaque maillon représenterait une configuration de la grille, et aurait plusieurs successeurs. Vous avez peut-être déjà vu cette structures de données dans le cours de Graphes : il s'agit d'un arbre enraciné. Par définition, un arbre est un graphe connexe acyclique. Nous allons introduire un peu de vocabulaire sur les arbres qui nous sera utile pour la suite. Tout d'abord, chaque "maillon" représentant une configuration de grille est appelé un **nœud** ou un **sommel** (nous utiliserons le premier terme ici). Chaque flèche reliant un nœud à un nœud de "niveau" en-dessous est un **arc** (même vocabulaire que pour les graphes). Si une flèche relie un nœud A (placé à l'origine de la flèche) à un nœud B (placé à la pointe de la flèche), alors on dira que B est l'**enfant** de A, et A est le **parent** de B.

Un nœud qui n'a aucun enfant est appelé une **feuille**, tandis que le nœud n'ayant aucun parent est appelé la **racine**. Sur la Figure 3, on peut voir 7 nœuds, dont une seule racine tout en haut et 4 feuilles tout en bas. Les trois sommets qui ne sont pas des feuilles ont deux enfants, et chaque sommet sauf la racine ont un parent.

Sur la Figure 2, les deux feuilles sont les enfants de la racine, tandis que la racine est le parent de chaque feuille.

Nous continuons l'algorithme minimax et développons l'arbre que nous avons commencé à créer. Sur la Figure 3, on considère chaque configuration d'une feuille de l'arbre, et on leur associe comme enfant les configurations correspondant aux coups que l'IA pourrait jouer : on obtient alors la Figure 4. On voit que, petit à petit, l'arbre des configurations s'enrichit de nouveaux noeuds placés à différents étages (ou niveaux) : à l'étage 0, la configuration actuelle de la grille ; à l'étage 1, les coups que l'IA pourrait jouer ; à l'étage 2, les coups que le joueur humain pourrait jouer ; à l'étage 3, les coups que l'IA pourrait jouer ensuite, etc.

On continue de développer l'arbre, en construisant les enfants des feuilles (qui n'en seront plus) correspondant aux coups que le joueur humain peut jouer. On précise deux règles à respecter :

- On ne développe pas un enfant qui correspondrait à un coup non autorisé. Voilà pourquoi, sur la Figure 5, le noeud tout à droite de l'étage 3 n'a qu'un seul enfant : le joueur humain ne peut pas jouer sur la colonne de droite.
- Si un noeud correspond à une victoire d'un des joueurs, on ne développe pas ses enfants, et ce noeud restera une feuille.

On continuera de développer l'arbre sur 5 étages en tout, et on obtiendra une structure comme celle représentée sur la Figure 6. N'oubliez pas qu'en vrai, dans ce projet, chaque noeud aura environ 7 fils, ce qui donnera à peu près 16807 feuilles dans votre arbre.

Algorithmme minimax, étape 2/2 - remonter l'information

Maintenant que nous avons développé l'arbre jusqu'à la hauteur souhaitée (5 dans ce projet), nous utilisons la fonction de score pour évaluer chaque configuration de grille correspondant à une feuille de l'arbre (nous verrons dans la section suivante comment construire cette fonction de score). On commence, comme montré à la Figure 7, par mesurer le score de chaque feuille de l'arbre.

Nous devons ensuite remonter l'information aux noeuds parents de l'arbre. La valeur de score d'un noeud parent sera entièrement évaluée à partir du score de ses enfants, et dépend de qui joue entre la configuration du noeud parent et la configuration des noeud enfants :

- Si entre le parent P et ses enfants, c'est l'IA qui joue, on estime que l'IA ira toujours dans la meilleure configuration pour elle-même : le score associé au noeud parent P sera donc le maximum du score de ses enfants. Cela représente le fait que, si l'IA se retrouve dans la configuration de grille correspondant à P, elle jouera le coup lui permettant de se retrouver dans la situation la plus avantageuse pour elle, donc le coup permettant de maximiser le score de grille.
- Si entre le parent P et ses enfants, c'est l'humain qui joue, on estime que l'humain ira toujours dans la configuration la moins bonne pour l'IA le score associé au noeud parent P sera donc le minimum du score de ses enfants. Cela représente le fait que, si l'humain se retrouve dans la configuration de grille correspondant à P, il jouera le coup lui permettant de se retrouver dans la situation la plus désavantageuse pour l'IA, donc le coup permettant de minimiser le score de grille.

On remonte tout d'abord les score de l'étage 5 à l'étage 4, comme illustré sur la Figure 8 (le passage de l'étage 4 à l'étage 5 représentait le fait que l'IA jouait, donc on utilise la formule du maximum). On continue, et on remonte les score de l'étage 4 à l'étage 3, comme illustré sur la Figure 9 (le passage de l'étage 3 à l'étage 4 représentait le fait que l'humain jouait, donc on utilise la formule du minimum). On continue de remonter jusqu'à la racine, et on obtient la Figure 12.

Le coup que l'IA doit jouer est le coup permettant d'aller de la racine à son enfant ayant le score le plus élevé.. Ensuite, l'humain jouera son coup, et l'IA recommencera l'intégralité de l'algorithme minimax depuis la nouvelle configuration de grille, pour décider du meilleur coup à jouer.

La fonction de score d'une grille

Le dernier élément à aborder dans ce projet est la fonction de score que l'on associera à une grille. C'est cette fonction qui permettra de définir l'intelligence de l'IA : si la fonction n'est pas bonne, l'IA n'évaluera pas bien les configurations vers lesquelles se diriger, et perdra souvent la partie.

Pour définir le score d'une configuration de grille correspondant à une feuille de l'arbre, on parcourra toutes les lignes, colonnes et diagonales, et on regardera chaque suite de 4 cases voisines (soit horizontales, soit verticales, soit diagonales). Dans cette suite :

- Si on a 4 jetons bleus (notre couleur), on ajoute 100 au score.
- Si on a 4 jetons rouges (la couleur de l'adversaire), on retire 95 au score.
- Si on a 3 jetons bleus et un emplacement vide, on ajoute 5 au score.
- Si on a 3 jetons rouges et un emplacement vide, on retire 4 au score.
- Si on a 2 jetons bleus et 2 emplacements vides, on ajoute 2 au score.
- Si on a 2 jetons rouges et 2 emplacements vides, on retire 1 au score.

Ce qui est à rendre

Vous devrez rendre, dans un fichier ZIP, tout le code source de votre programme et, si nécessaire (et que vous en connaissez le fonctionnement), un Makefile.

Vous devrez rédiger un rapport complet sur votre solution, et le rendre en format PDF : description des structures de données adoptées, choix des structures utilisées à différentes étapes de votre programme,

extraits de code qui vous paraissent importants d'expliquer dans le rapport, choix de l'algorithme utilisé, ... Votre rapport doit être un journal de bord permettant d'expliquer chacun de vos choix et chacune des difficultés rencontrées. N'hésitez pas à bien regarder sur Moodle le guide de rédaction d'un rapport, afin de comprendre ce qui est attendu de vous. Votre rapport devra également expliquer comment compiler et utiliser votre programme.

Lisez bien les instructions de rédaction des rapports sur Moodle avant de rédiger le votre.

Vous devrez rendre un fichier ZIP contenant le code source, et un fichier PDF (qui ne devra pas être dans le zip). Tous vos diagrammes devront être intégrés dans le PDF (pas de fichiers à part).

La note de votre projet sera le minimum entre votre note de rapport et votre note de code. Ne sous estimatez donc pas le rapport au profit du code.

Le projet peut être réalisé par groupe de deux au maximum.

Pour information, tous les codes seront comparés pour détecter les plagiats. Vous avez le droit de vous aider entre vous, mais pas de recopier le code des autres... Il est fortement déconseillé de partager son code source avec d'autres.

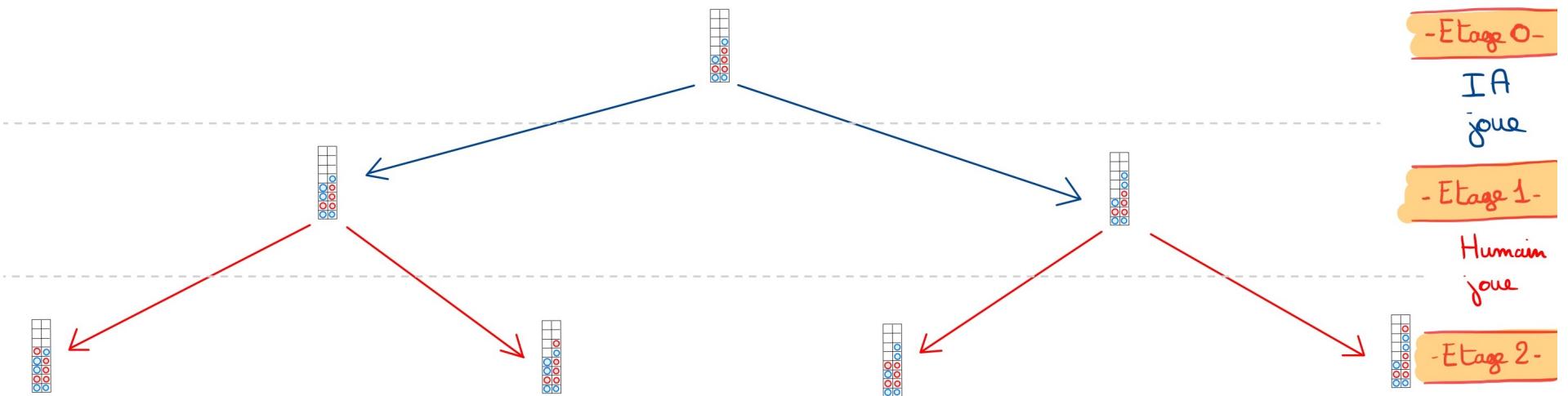


FIGURE 3 – Illustration après avoir développé les coups jouables par le joueur humain.

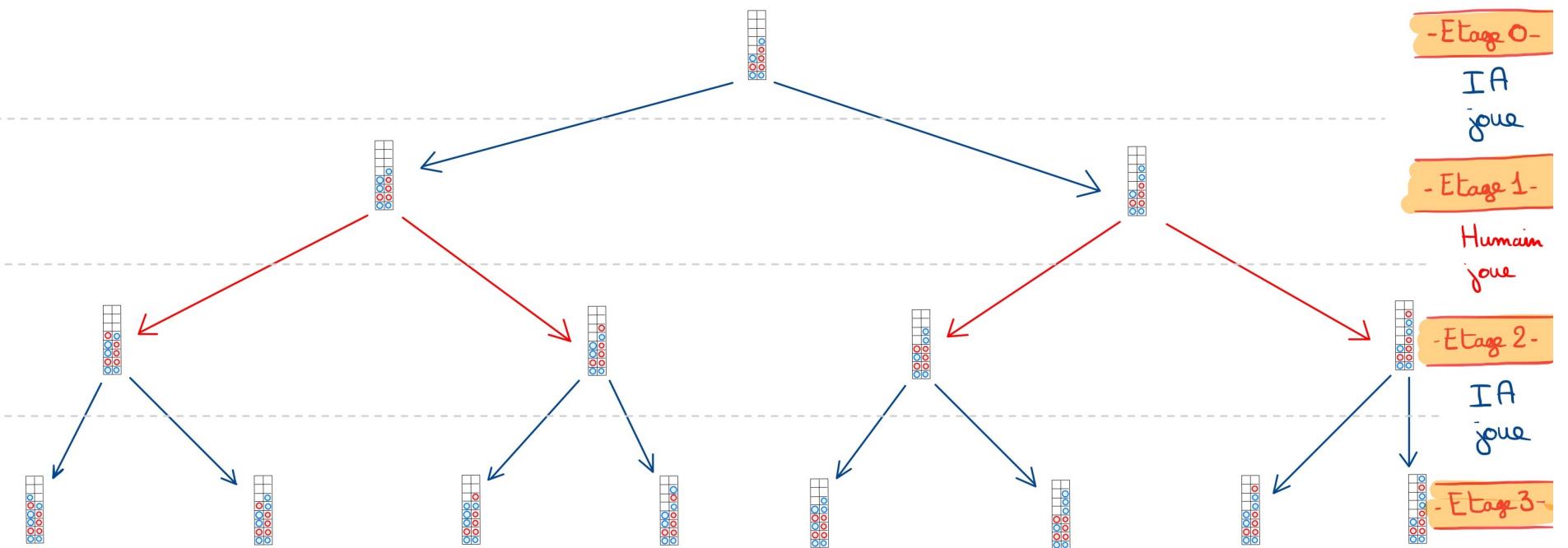


FIGURE 4 – Illustration de l'arbre après avoir développé 3 coups en avance.

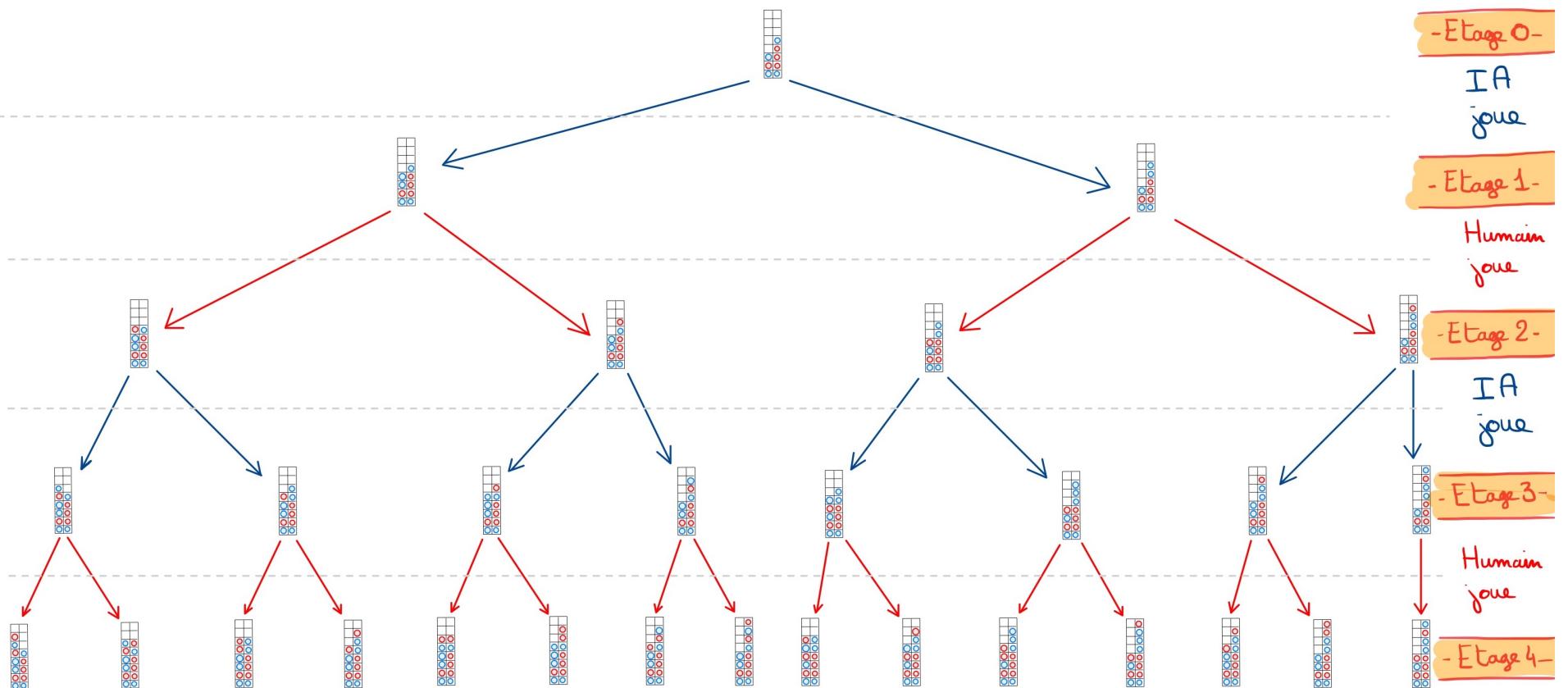


FIGURE 5 – Illustration de l'arbre après avoir développé 4 coups en avance.

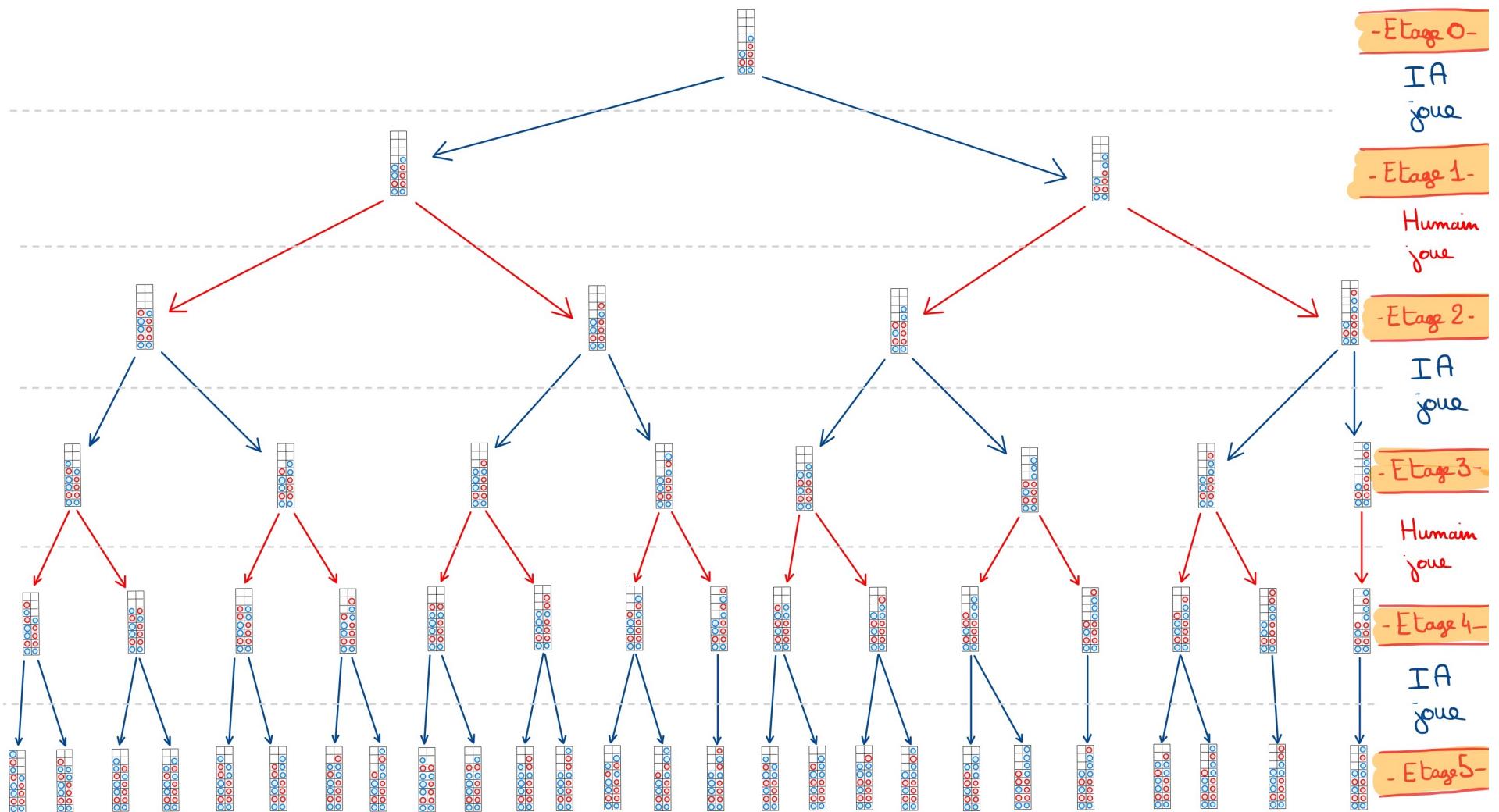


FIGURE 6 – Illustration de l’arbre après avoir développé 5 coups en avance.

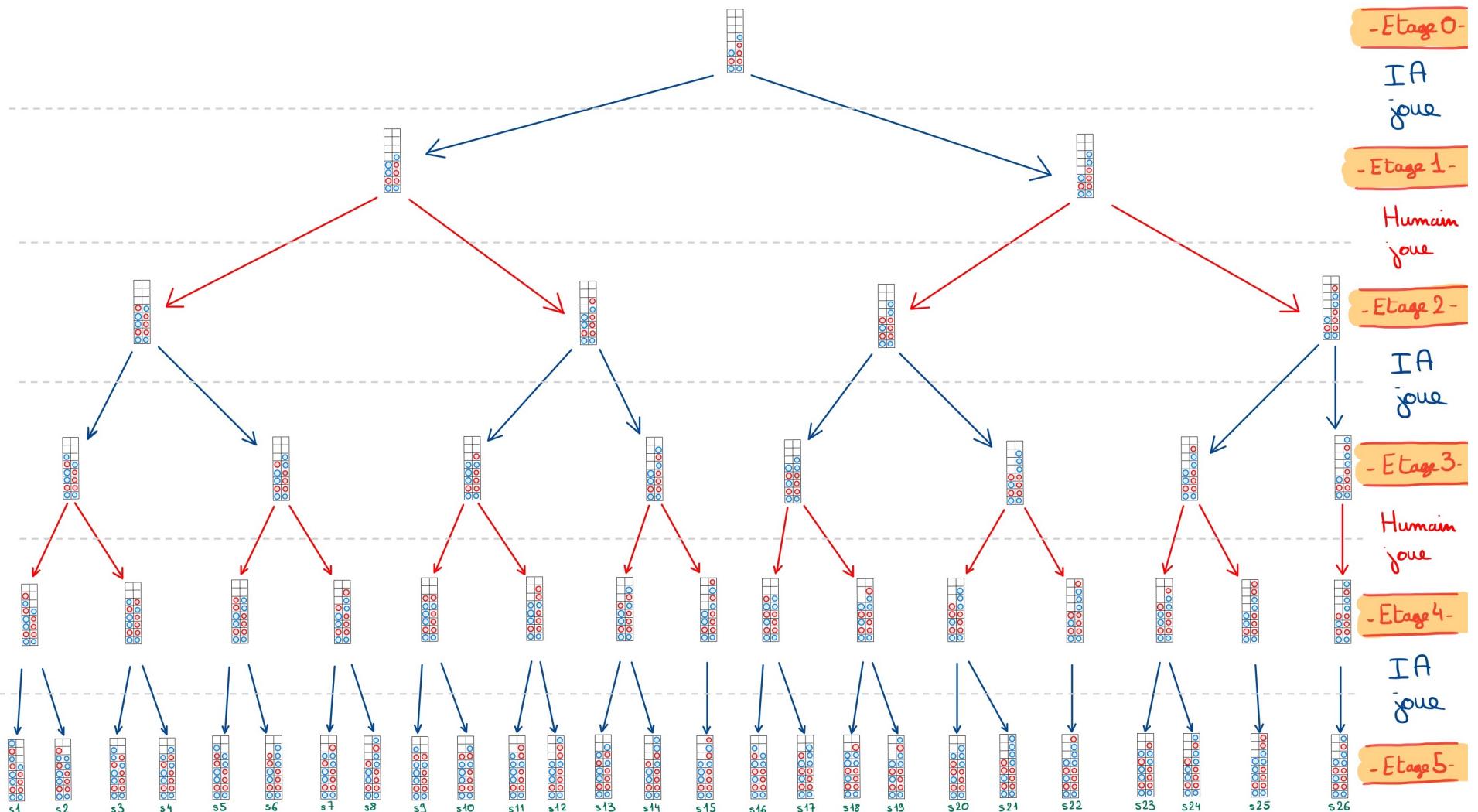


FIGURE 7 – Calcul du score de chaque feuille de l'arbre.

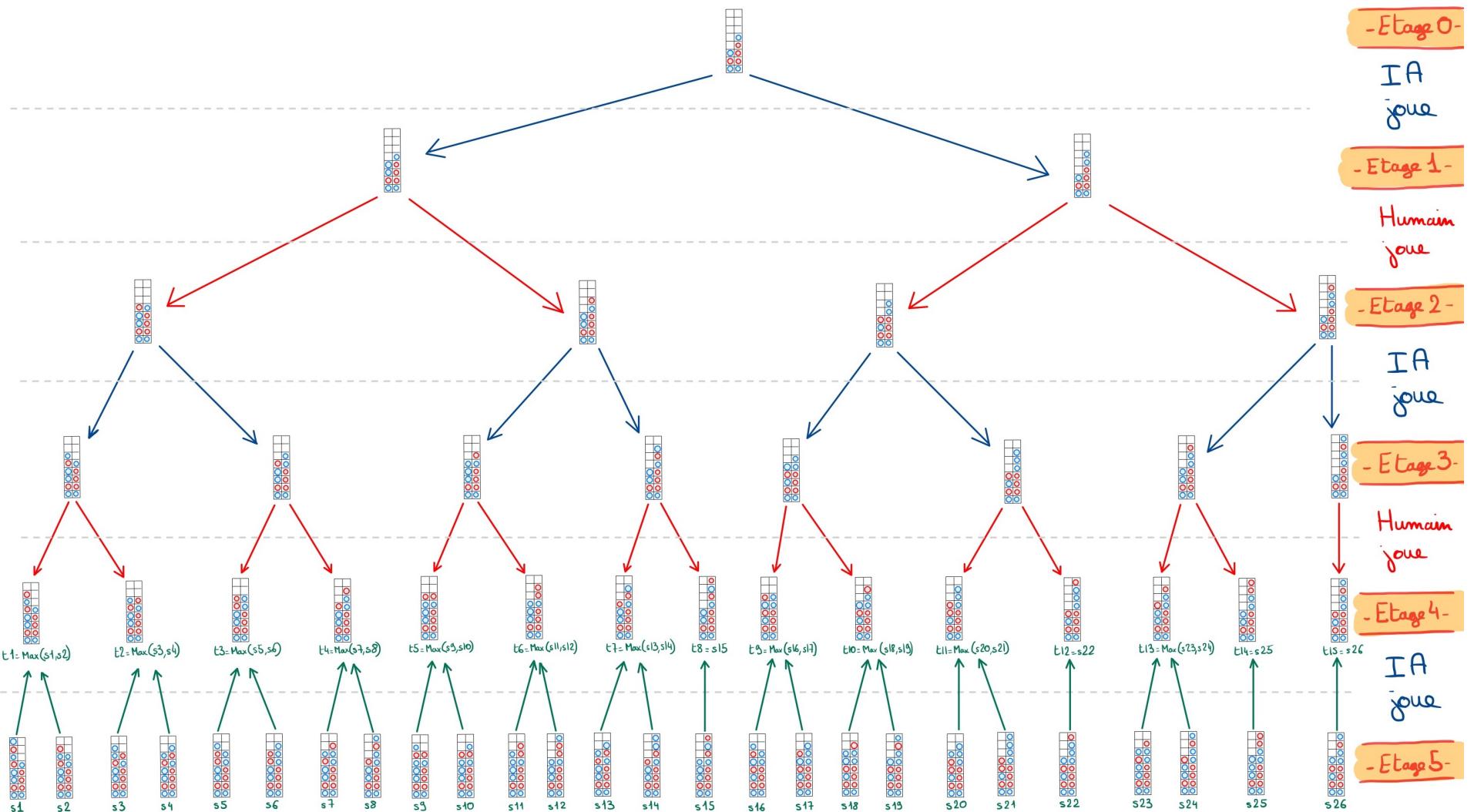


FIGURE 8 – Transmission du score des feuilles de l'étage 5 à l'étage 4.

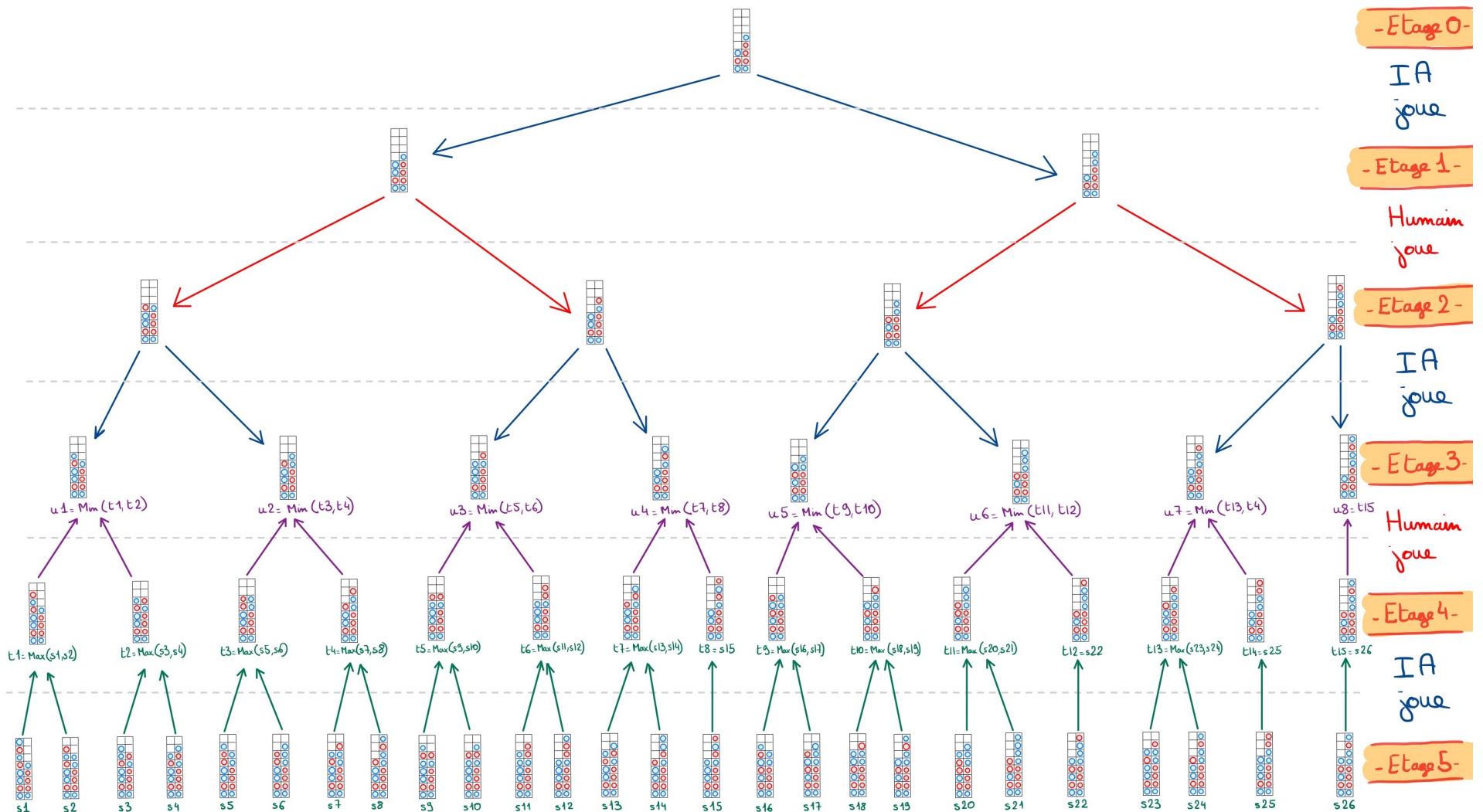


FIGURE 9 – Transmission du score de l'étage 4 à l'étage 3.

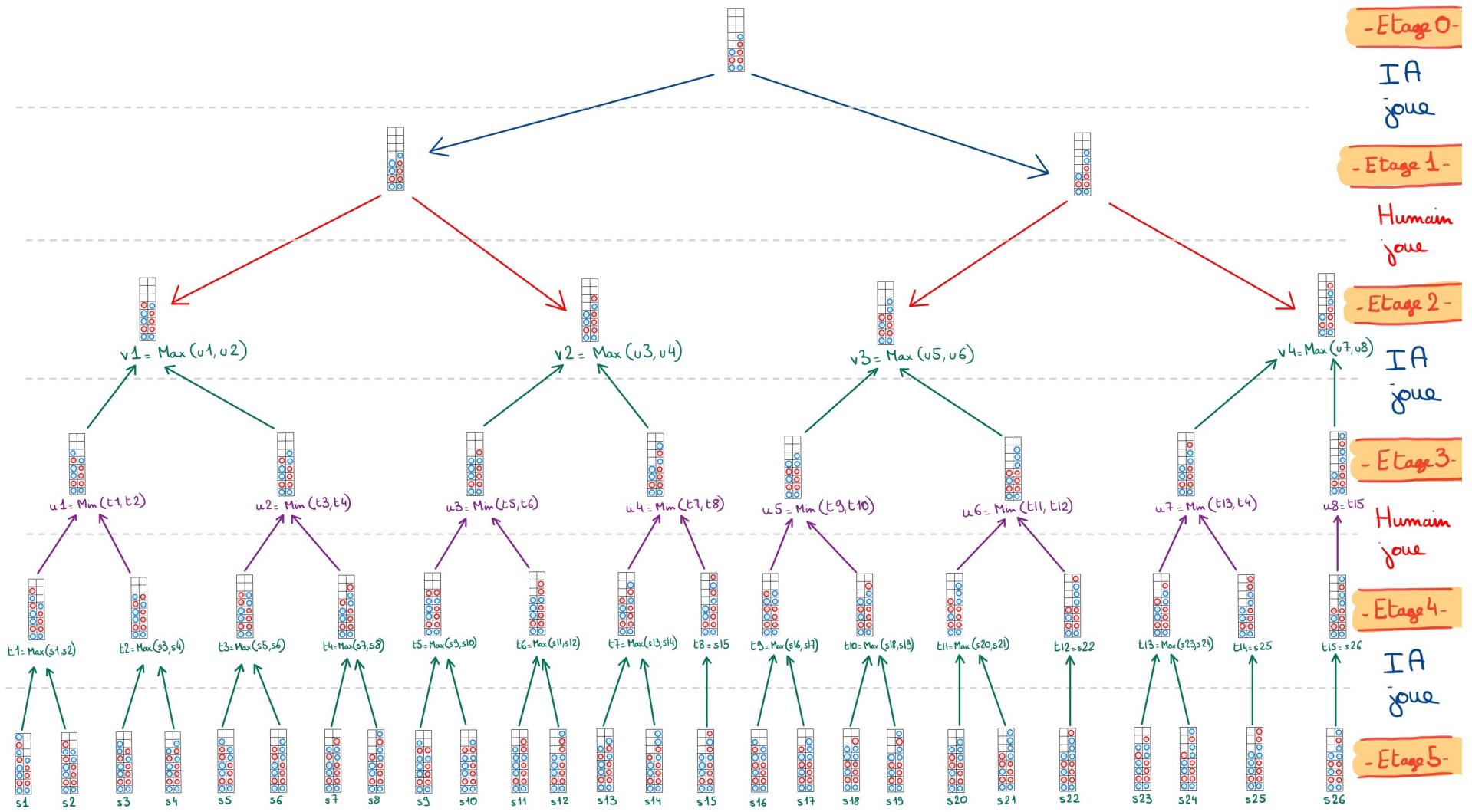


FIGURE 10 – Transmission du score de l'étage 3 à l'étage 2.

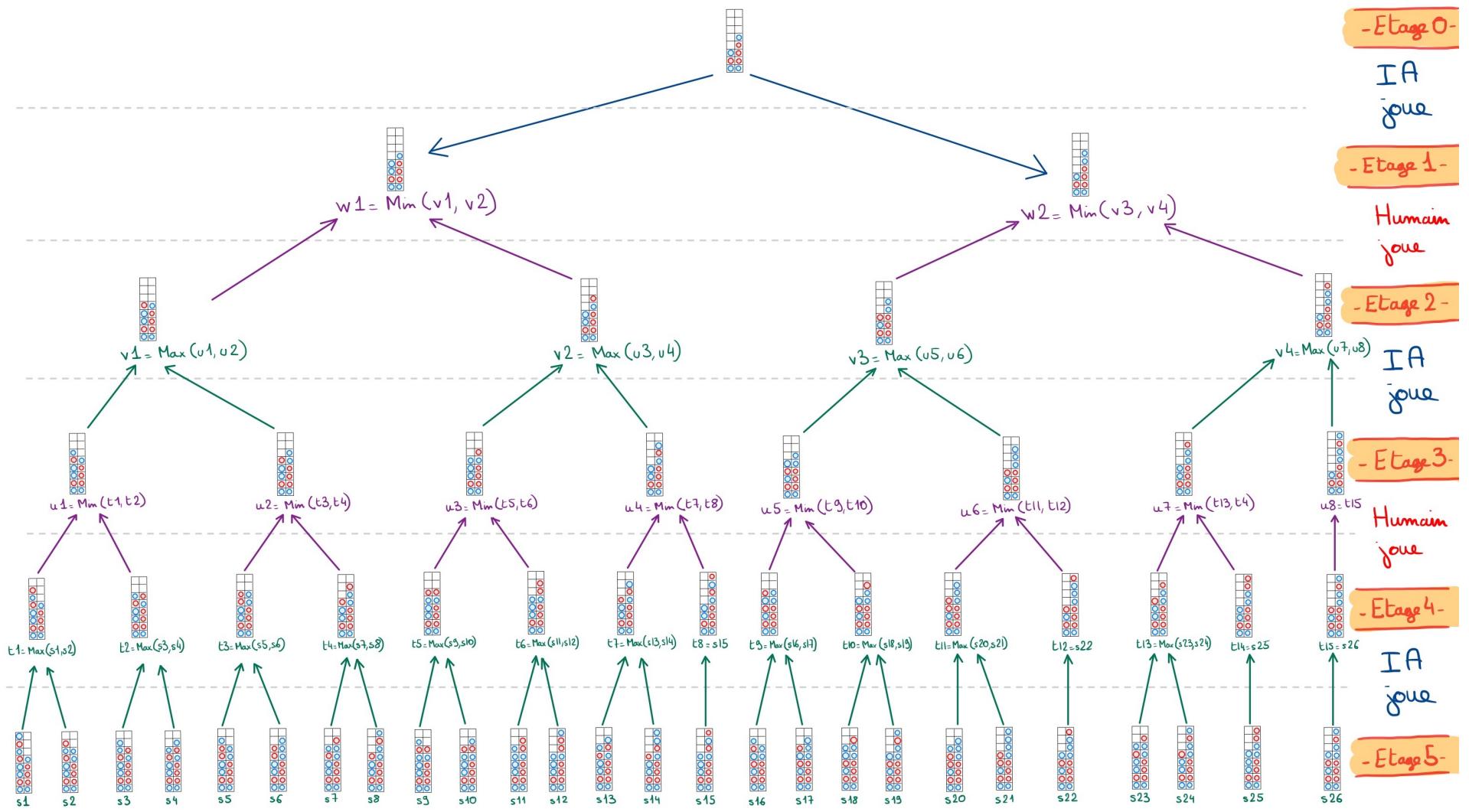


FIGURE 11 – Transmission du score de l'étage 2 à l'étage 1.

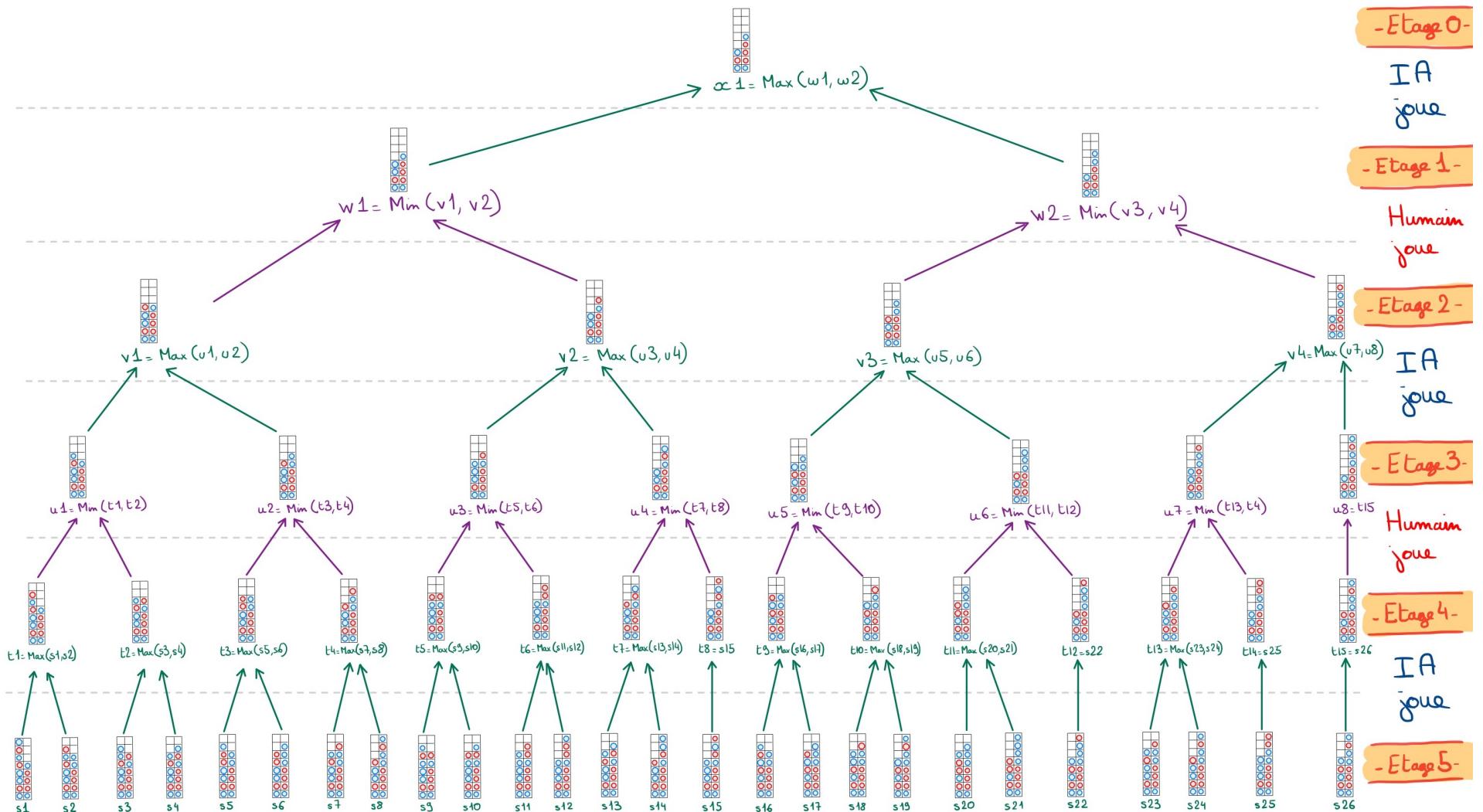


FIGURE 12 – Transmission du score de l'étage 1 à la racine.