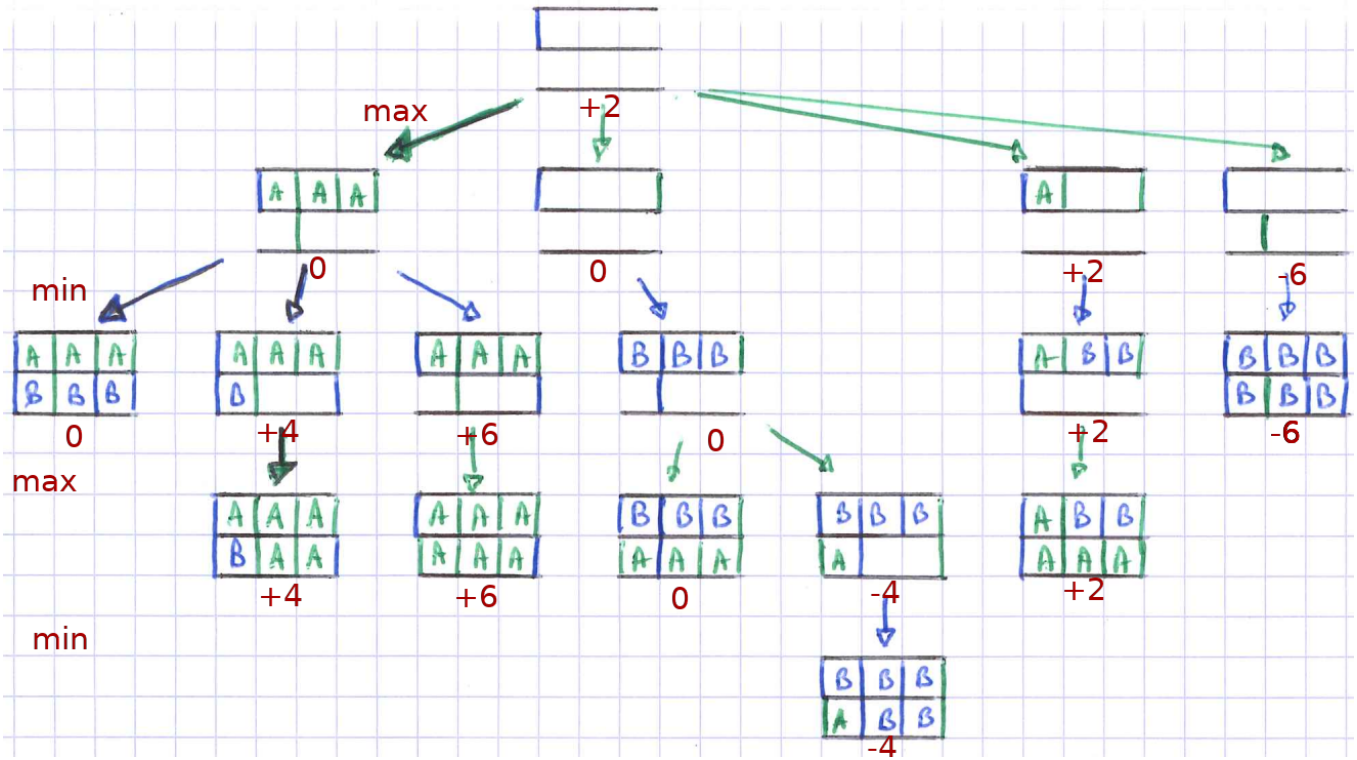
3 - Arbres implicites



3.1 - Backtrack – retour sur trace

Solveur de sudoku

Comment résoudre un sudoku ?

Méthode explicite

Essayer de coder les règles que l'on appliquerai *à la main*

- un seul élément possible dans une case
- une seule case possible pour un élément
- généralisation de ces arguments...

		8			6			3
6			9	8				
		9	1				5	
			8	3				7
4	3							
			7	2				6
		6	4				7	
9			3	5				
		5			2			1

Solveur de sudoku

Comment résoudre un sudoku ?

Méthode explicite

Essayer de coder les règles que l'on appliquerai *à la main*

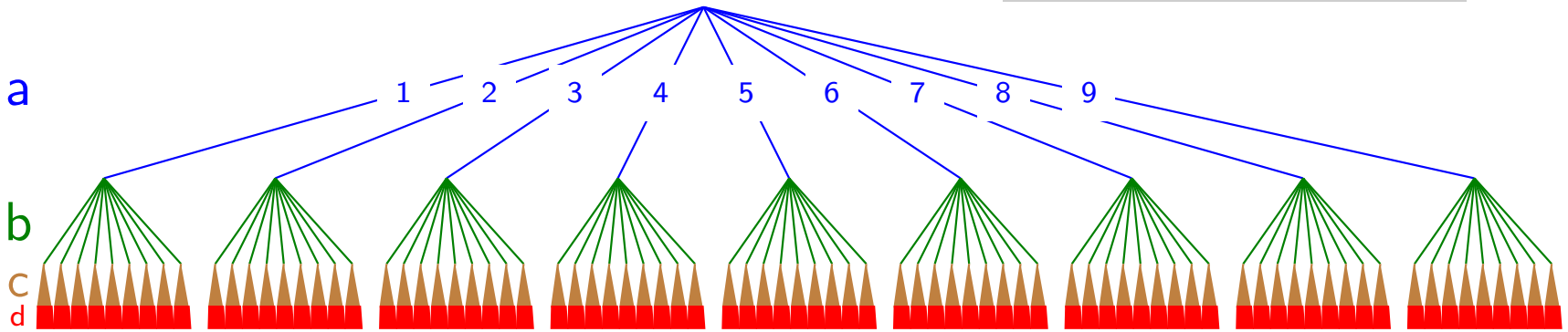
- un seul élément possible dans une case
- une seule case possible pour un élément
- généralisation de ces arguments...
- pas toujours suffisant

		8			6			3
6			9	8				
		9	1				5	
			8	3				7
4	3							
			7	2				6
		6	4				7	
9			3	5				
		5			2			1

Solveur de sudoku

Un sudoku est un arbre...

a	b	8	c	d	6			3
6			9	8				
		9	1				5	
			8	3				7
4	3							
			7	2				6
		6	4				7	
9			3	5				
		5			2			1

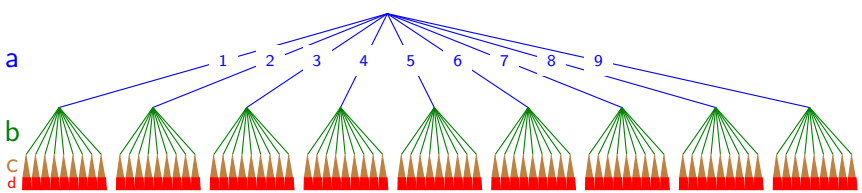


- Chaque case blanche $\longrightarrow$ un étage
- chaque choix $\longrightarrow$ un nœud
- un grand arbre... (mais pas plus de 9^{81} feuilles)

Solveur de sudoku

Retour sur trace – backtrack

a	b	8	c	d	6			3
6			9	8				
		9	1				5	
			8	3				7
4	3							
			7	2				6
		6	4				7	
9			3	5				
		5			2			1



La grille a-t-elle une solution ?

- si la grille est totalement remplie, je teste sa validité
- sinon, pour une case vide :
  1. placer la prochaine valeur disponible
  2. tester sur le reste (appel récursif)
  3. si aucune solution trouvée, passer à la valeur suivante, sinon, valider.
- toutes les valeurs testées : retourner *échec*

Retour sur trace – backtrack

À quoi ressemblerait le code ?

```
boolean solve(Sudoku sudoku){
    if(gameOver(sudoku))
        return isValid(sudoku);
    else {
        Square choice = firstEmptySquare(sudoku);
        for(int i = 1; i <= 9; i++){
            sudoku.set(choice, i);
            if(solve(sudoku)){
                return true;
            }
        }
        sudoku.unset(choice);
        return false;
    }
}
```

- Profondeur
- *unset*
- État du sudoku à la fin de l'exécution

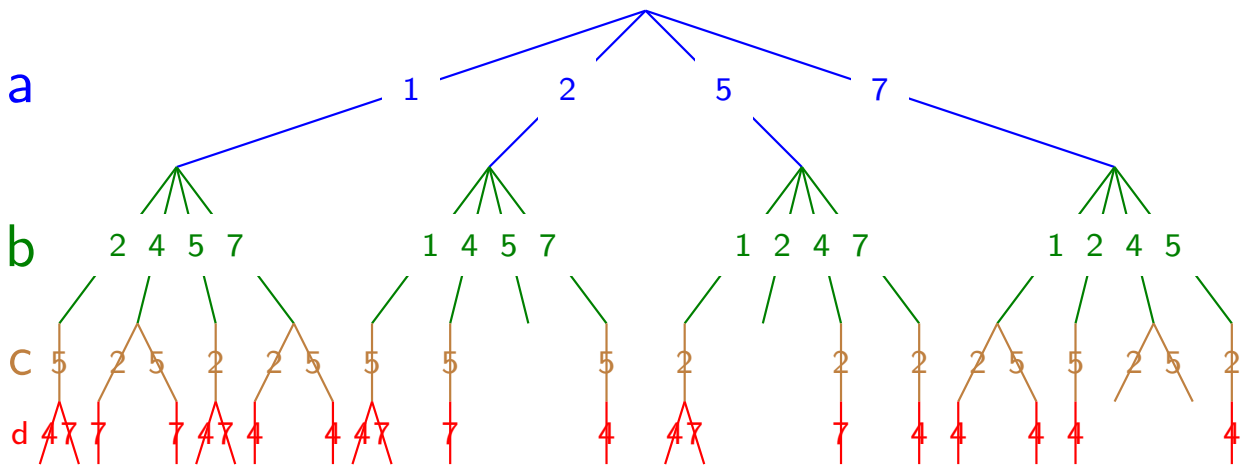
a	b	8	c	d	6			3
6			9	8				
		9	1				5	
			8	3				7
4	3							
			7	2				6
		6	4				7	
9			3	5				
		5			2			1

Problème

Arbre trop grand

Amélioration :

- Ne pas s'encombrer des branches perdues d'avance



- Ordre des cases à traiter

Retour sur le principe

Retour sur trace

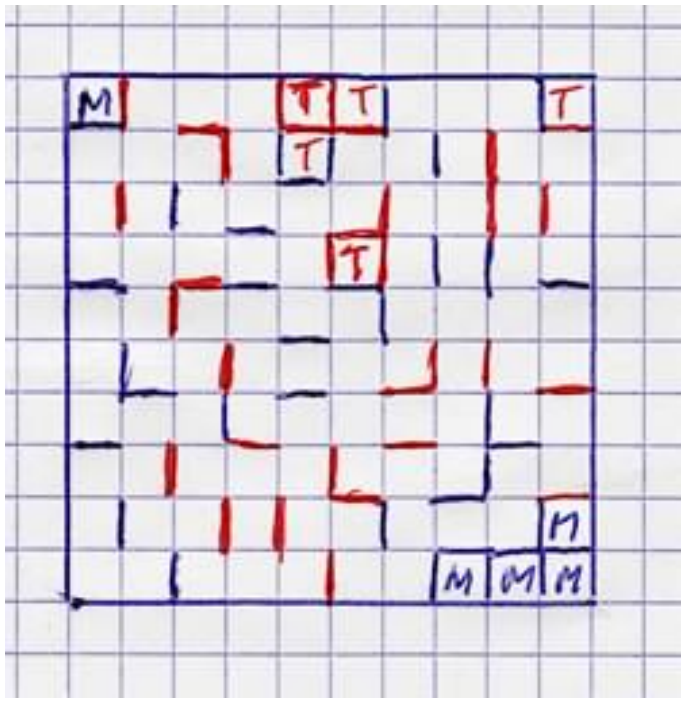
- Un arbre implicite
- Exploration d'une branche en réalisant une action
  (souvent par effet de bord)
- Annulation du dernier mouvement avant de rendre la main
- Très pratique, mais vite coûteux
- Parfois pas de meilleures solutions exactes...

3.2 - Jeux à deux joueurs

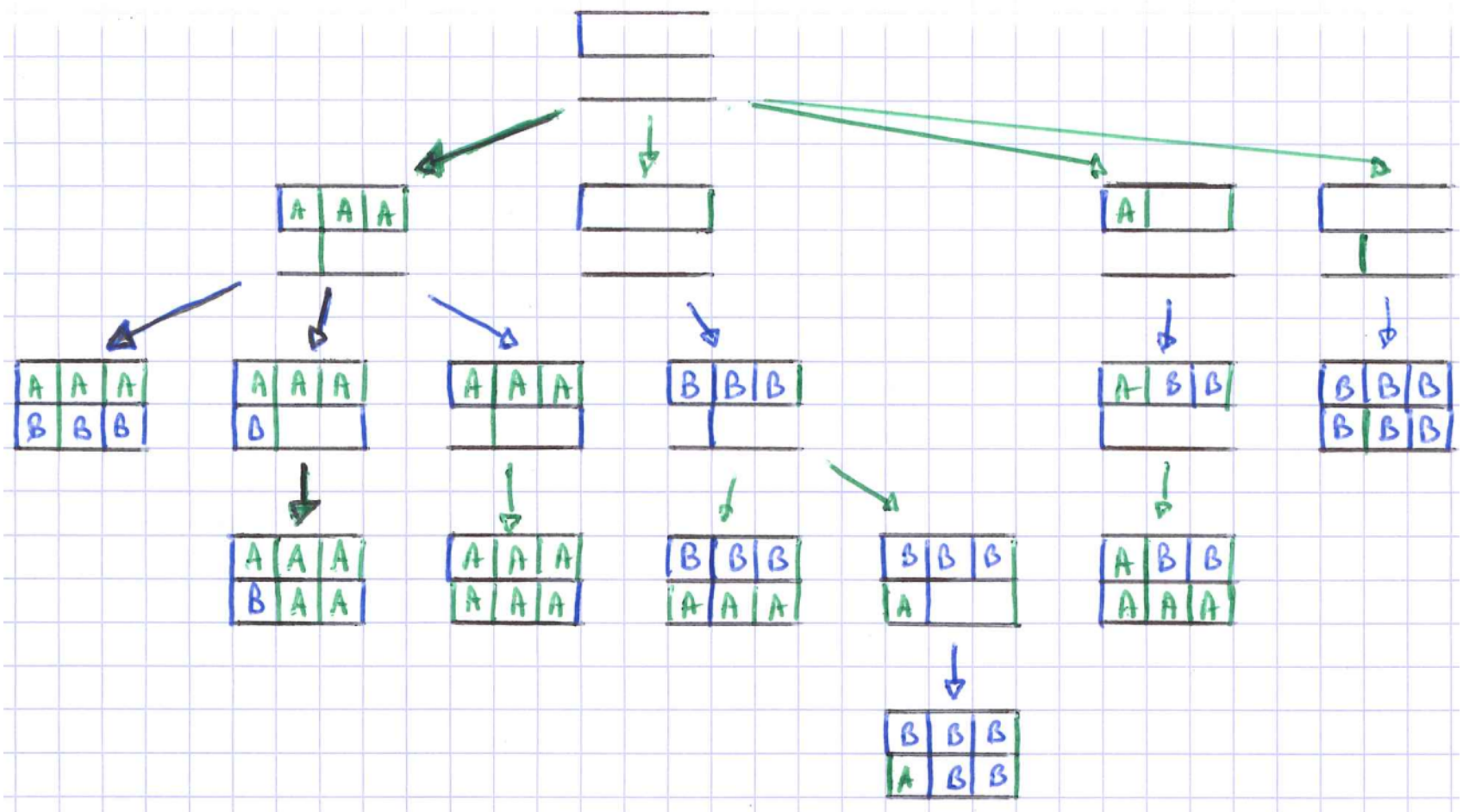
La Pipopipette (ou jeu des petits carrés)

Chacun son tour :

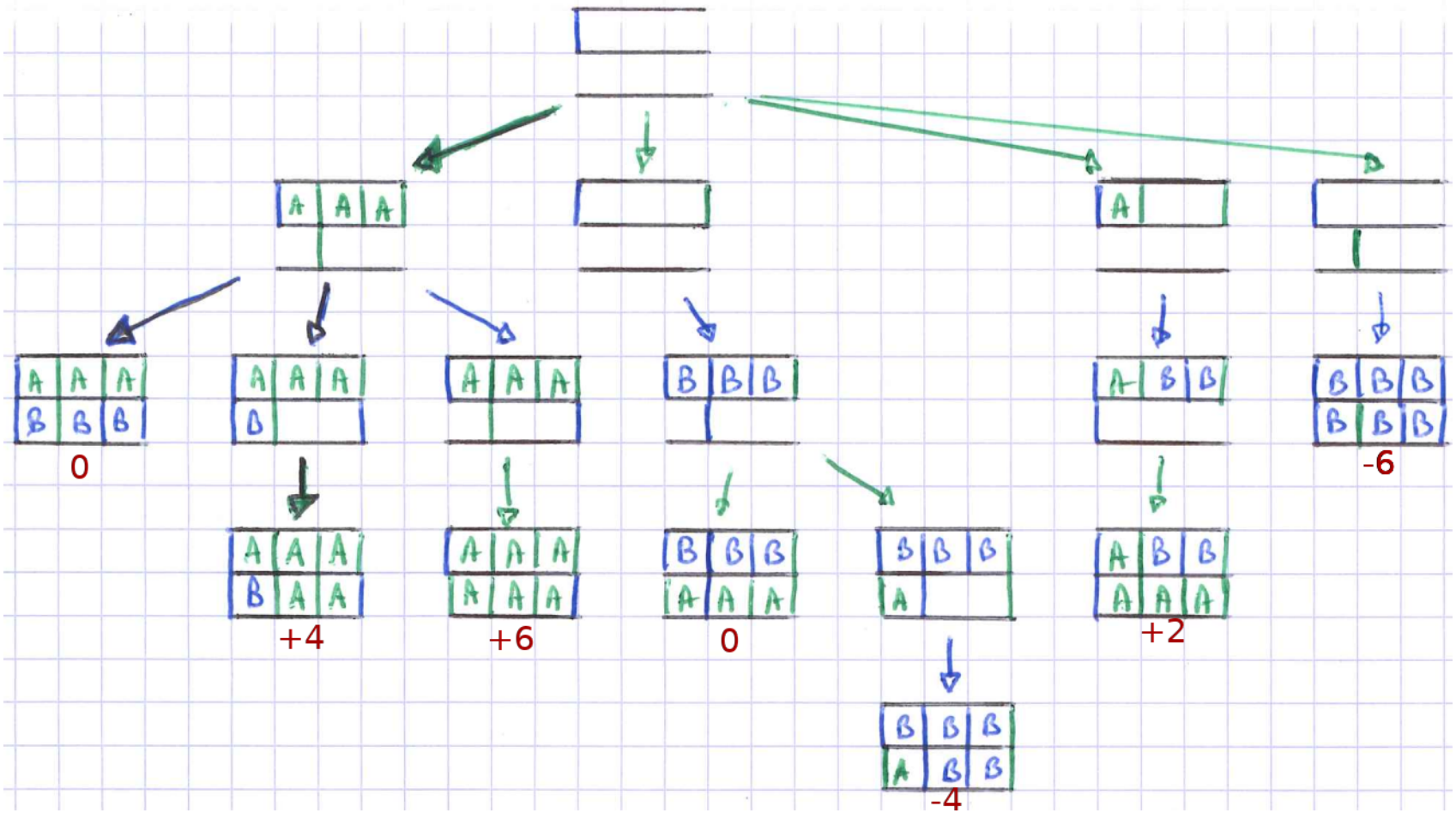
- tracer un coté d'une case
- si on ferme une case, +1pt et on rejoue.



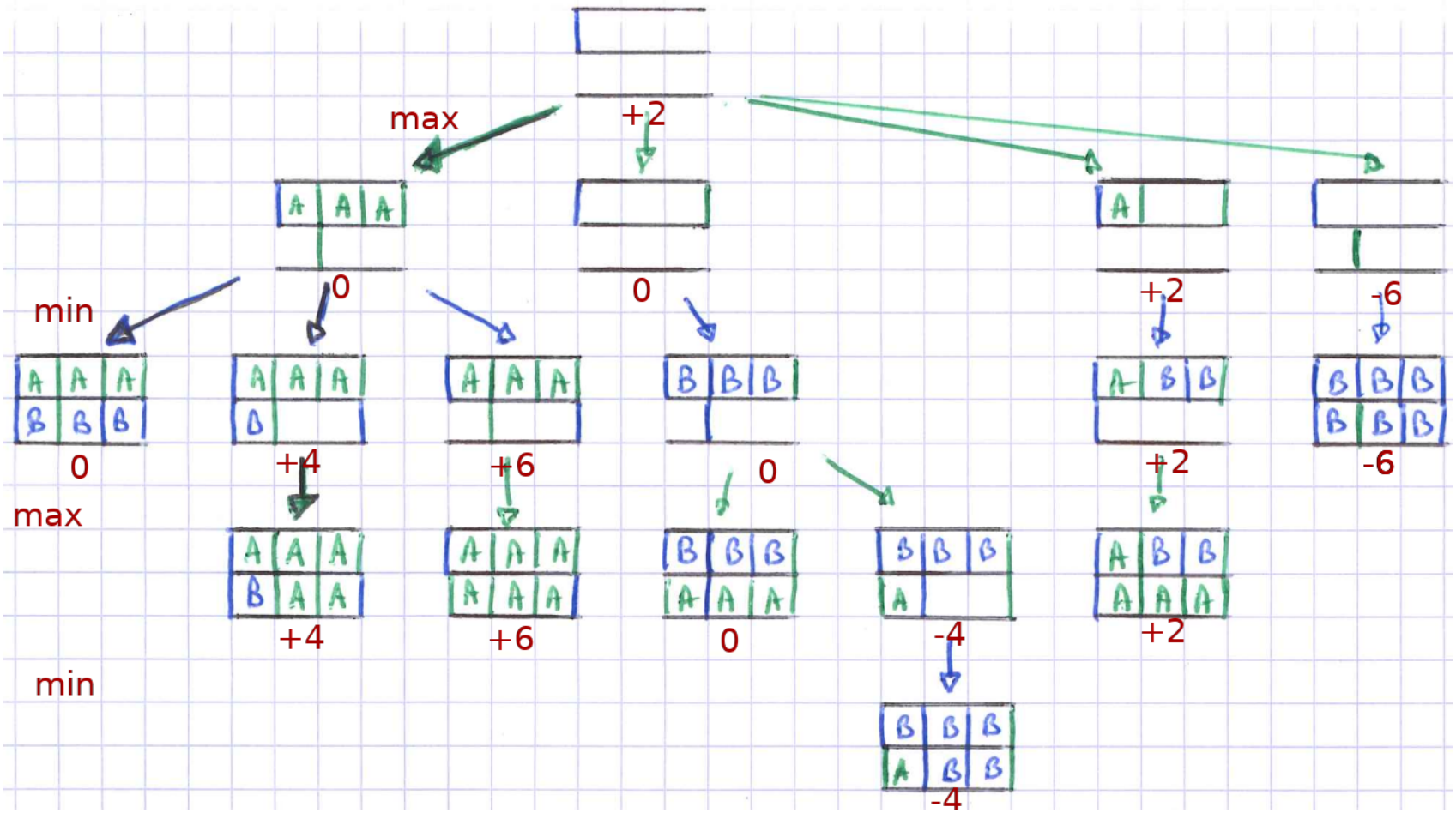
Arbre de stratégie



Arbre de stratégie



Arbre de stratégie



Méthode

Stratégie

- Pour Ann, choisir le coup qui maximise le score.
- Pour Bob, choisir le coup qui minimise le score.

Comment traiter cela ?

Solution 1 : Récursivité croisée

```
boolean scoreAnn(DotsAndBoxes game){
     // scoreBob(DotsAndBoxes game){
  if(game.isOver())
    return game.getScore();
  else {
    Iterator<Move> iter = game.availableMoves();
    game.play(iter.next());
    best = scoreBob(game);        // scoreAnn(game)
    game.cancelLast();
    while(iter.hasNext()){
      game.play(iter.next())
      best = max(best,            //  best = min(best,
             scoreBob(game)); //          scoreAnn(game));
      game.cancelLast():
      }
    return best;
  }
}
```

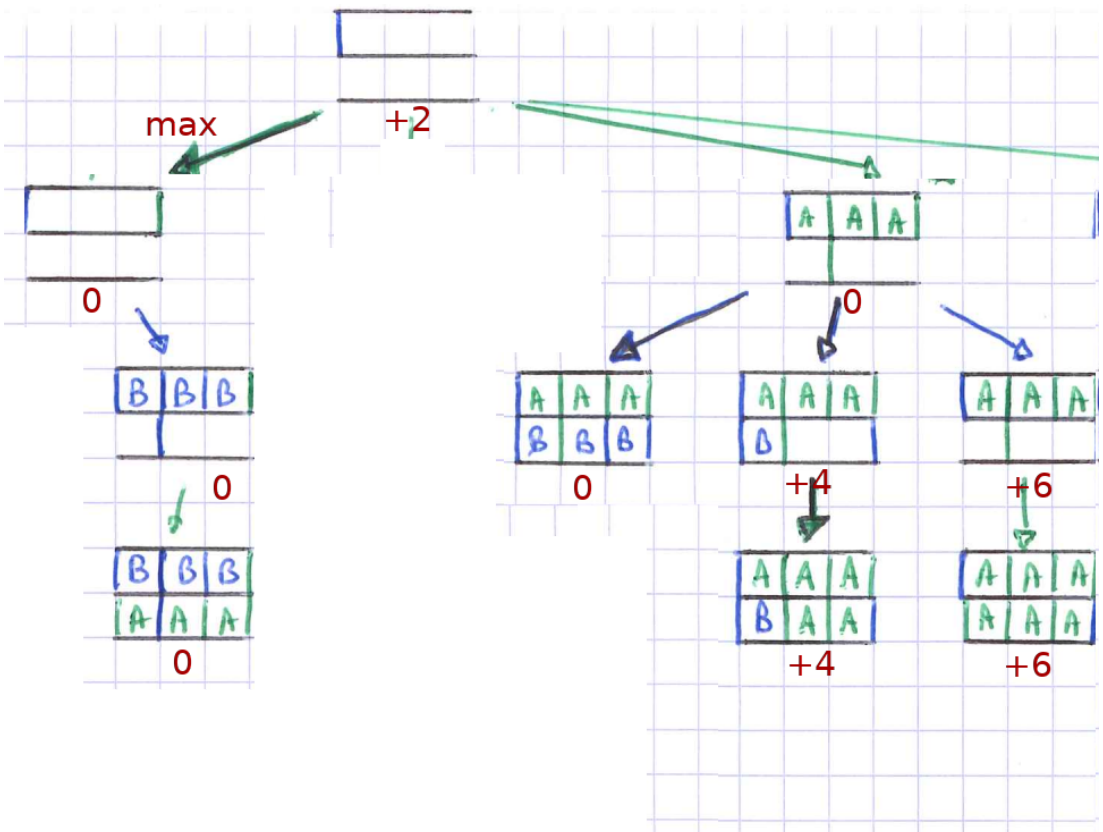
Solution 2 : Par inversion

```
boolean scoreNext(DotsAndBoxes game){
  if(game.isOver())
    return game.getScore();
  else {
    Iterator<Move> iter = game.availableMoves();
    game.play(iter.next());
    best = -scoreNext(game);
    game.cancelLast();
    while(iter.hasNext()){
      game.play(iter.next());
      best = max(best, -scoreNext(game));
      game.cancelLast();
      }
    return best;
  }
}
```

Couper des branches

Algorithme Alpha-beta

- Parfois inutile d'aller plus loin.



Autres arbres implicites

Autres arbres implicites :

XML

```xml
<?xml version="1.1" encoding="UTF-8"?>
<!DOCTYPE bibliotheque SYSTEM "oc.dtd">
<bibliotheque>
    <livre genre="aventure">
        <titre>Le tour du monde en 80 jours</titre>
        <auteur>Jules Verne</auteur>
        <edition>Flammarion</edition>
    </livre>
    <livre genre="policier">
        <titre>Le crime de l'orient express</titre>
        <auteur>Agatha Christie</auteur>
        <collection>Le masque</collection>
    </livre>
    <livre genre="theatre">
        <titre>Le Cid</titre>
        <auteur>Corneille</auteur>
        <edition>Hachette éducation</edition>
    </livre>
    <livre genre="aventure">
        <titre>Le hussard sur le toit</titre>
        <auteur>Jean Giono</auteur>
        <memeauteur>
            <livre genre="roman">
                <titre>Colline</titre>
                <auteur>Jean Giono</auteur>
            </livre>
        </memeauteur>
    </livre>
</bibliotheque>
```

Règles :

- En-tête (facultative)
- Une balise englobante.
- toute balise ouverte est fermée.
- Imbrication propre de balises (pas d'entrelacement)

→ arbre sémantique

Autres arbres implicites :

Grammaire

$$3 * 5 + 4 + 7 * (2 + 3)$$

- Ordre de priorité dans les calculs
- représentation sous forme d'un arbre.