Games and Game Trees

One of the most common games is Tictactoe, also called Noughts and Crosses.

TICTACTOE. *There is a 3-by-3 grid. Players alternate marking their symbol: the first to get three in a row is the winner.*

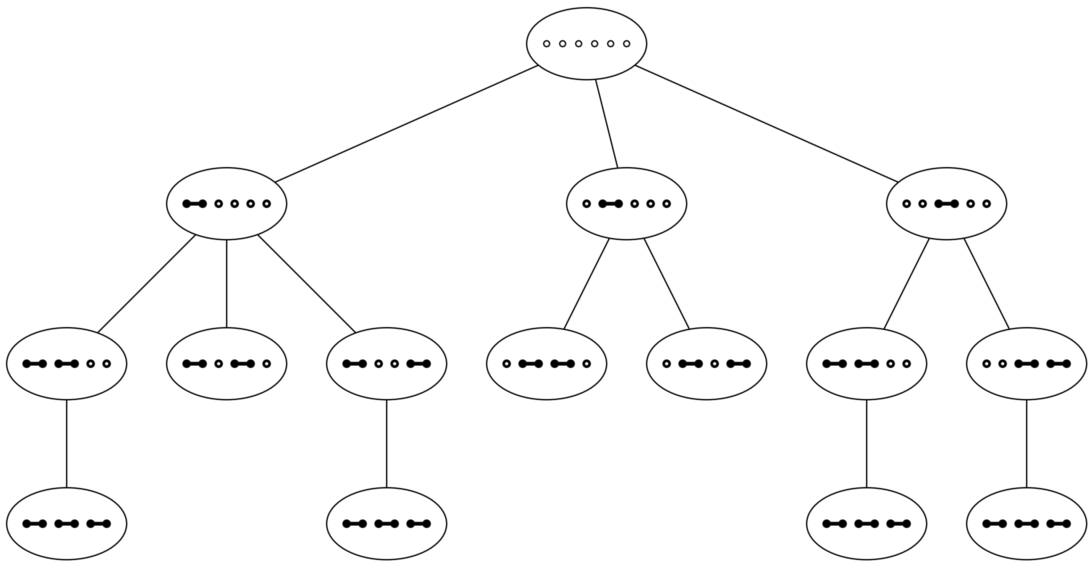
A similar game is the following: there are the numbers 1 through 9. Players take turns picking a number (no repeats). The first player to have a triple which sums to 15 is the winner.

The first type of game we look at is where the players alternate but there is no randomness or hidden information. This genre include Chess, Checkers, Tictactoe and Othello/Reversi. There is a natural way to represent the game.

2.1 Game Trees

We consider games where two adversaries alternate. One choice leads to another and a tree of choices results—the ***game tree***. The nodes of the tree represent board positions. For each node, the children are the next positions. The game ends with certain positions being reached. Since this is a tree, these positions are called leaves.

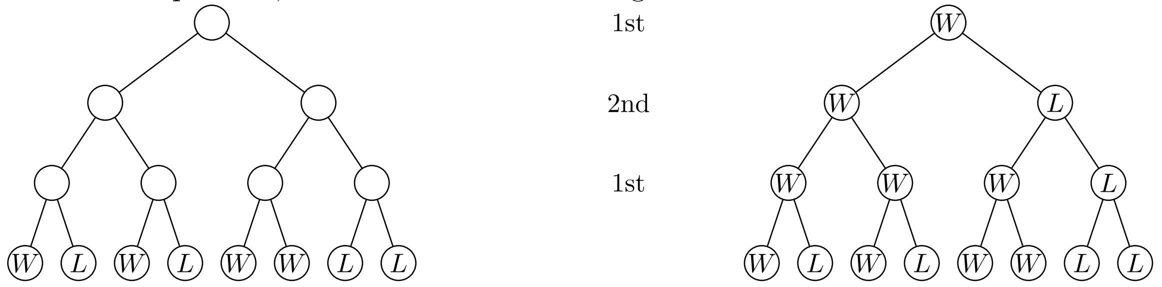
DAWSON'S KAYLES. *In Dawson's Kayles, some number of dots are drawn on a line. Two players alternate by picking two consecutive unused dots and joining them. The last person to make a move is the winner. The picture below gives the game tree for the game with 6 dots (to fit in the page, we can assume that the first person moves in the left half).*



In the tree for Dawson's Kayles above, the first player can force a win by taking the two middle dots. In all other cases, the second player can take the last pair.

If the tree is small enough, it can be searched exhaustively, moving from the leaves upwards. We start by marking all final positions as to whether the first player wins or loses. We then consider a position one move from the end. If every move that a player can make causes him to lose, then that position is a loss. On the other hand, if some move causes him to win, then that position can be viewed as a win. *Assuming perfect play*, we can then label each position one move from the end. And repeat the same argument with positions two moves from the end, and so up the tree.

To simplify the writing, we view the game *from the first player's perspective.* So we label a node as a Win if the first player wins, and as a Loss if the first player loses. Here is a simple tree, and the resultant labeling.



A ***strategy*** then is a complete set of rules. Meaning, for every position where it is that person's turn to move, the strategy specifies one move. Sometimes we can write down the strategy succinctly; sometimes we can do little better than stating which is the correct move in each position.

2.2 A Guarantee

Note that this process assigns a label to the root node of the tree: the game has a definite ***value/result***. If a draw is impossible, then the game is labeled as either first player win or second player win. This means that

> *either there is a strategy that guarantees the first player a win, or there is a strategy that guarantees the second player a win.*

The same idea extends to Chess. There is one of the following: (a) a strategy that guarantees the first player wins; (b) a strategy that guarantees the second player wins; (c) a pair of strategies that guarantee that neither player can lose (and so the game ends in a draw).

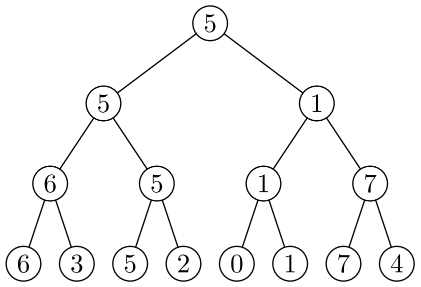
Of course, we still don't know which of the three is true. But if we did, it might make Chess uninteresting. But then again, that doesn't stop people enjoying Tictactoe, which is a draw with optimal strategies by both players.

2.3 MiniMax

We can generalize this labeling to handle games where there are more than two outcomes (for example win, loss and draw). This uses the MiniMax procedure. We still view the game from the perspective of the first player. We assign the value of 1 to a win for the first player, 0 a loss for the first player and $\frac{1}{2}$ for a draw. Then the first player tries to maximize the value of a position (preferring a 1 to a $\frac{1}{2}$, and a $\frac{1}{2}$ to a 0). This player is thus called the ***maximizer***. The second player the ***minimizer***. We use recursion.

```
MiniMax
 IF position is final, THEN RETURN value
 IF position is Minimizer's move, THEN
    call MiniMax on children
    RETURN minimum of values of children
 IF position is Maximizer's move, THEN
    call MiniMax on children
    RETURN maximum of values of children
```

An example is given in the following figure. The values at the leaves propagate upwards.



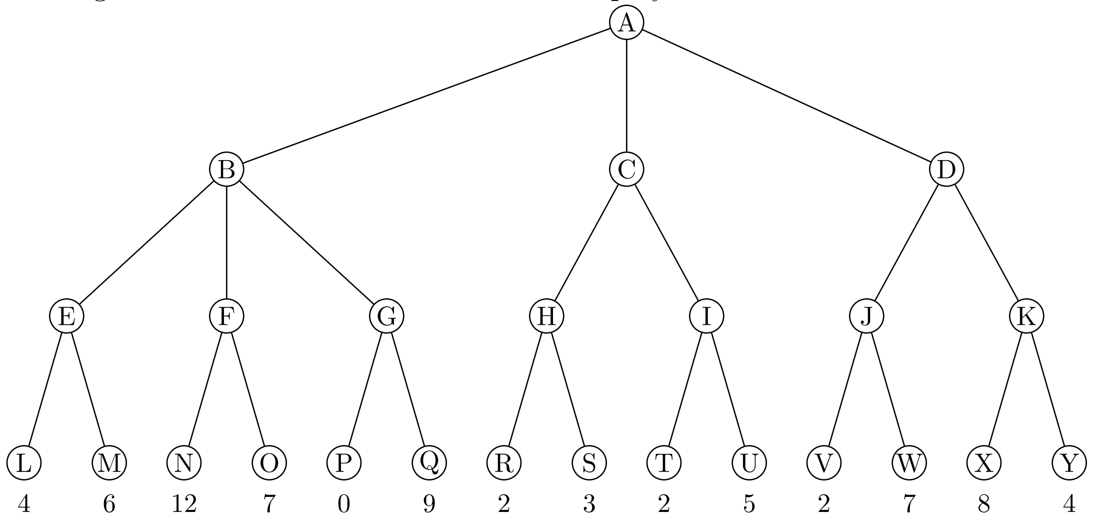
Unfortunately, for many interesting games, the tree is far too large to construct, let alone to search.

Exercises

2.1. Calculate who wins Dawson's Kayles on up to 7 dots.

2.2. Code up the MiniMax procedure, and determine the winner and their strategy for the game of Finger.

2.3. For the following tree (in which the maximizer moves first), what is the value of the game and what move should the first player make?


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
4
6
12
7
0
9
2
3
2
5
2
7
8
4