

Lecture 8: Intelligent Boardgame AI

Search aftermath: Recapitalize intelligent search

We went through all the search problems in the last lectures. Every problem had a specific characteristic and was part of a bigger class of problems. All of them had something in common: they could be solved by using search. If you read the example codes in every lecture, you might have noticed that the basic principle is the same in every problem: start in one position and create all possible following situations. If the following situation is a legal one, create a node out of it and repeat until a given effect appears, for example that a solution has been found or a maximum depth of search has occurred. All of the problems were based on the same code more or less, with changes for every representation. Here's the basic code we used in all problems:

```
1 def generate_tree(node):
2     if recursion_breaking_parameter:
3         return
4     turns = possible_moves(node.position)
5     for turn in turns:
6         turn = Vertex(node, node.depth+1, turn)
7         generate_tree(turn)
```

```
def possible_moves(node):
    return possible_moves
```

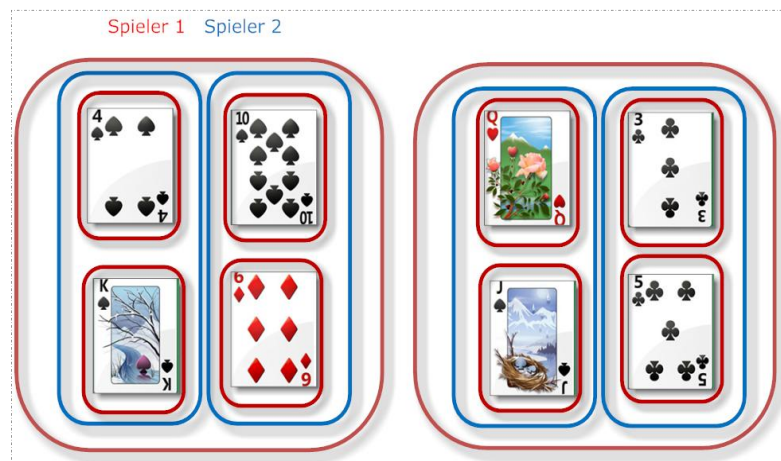
I want you to keep this in mind before moving to intelligent game playing, which is nothing more than another class of problem.

Intelligent game playing

We will now introduce the class of problems called “board games”. Those can be classified by the following question: “How can one find the best possible move in a set position, given the rules of the game?”. This question is sometimes impossible to answer, as we will find out, but we will try to find a method that provides at least an approximation for the best possible move in a given situation.

This lecture will not have any programming assignments, but I want you to play two games. The first one is this:

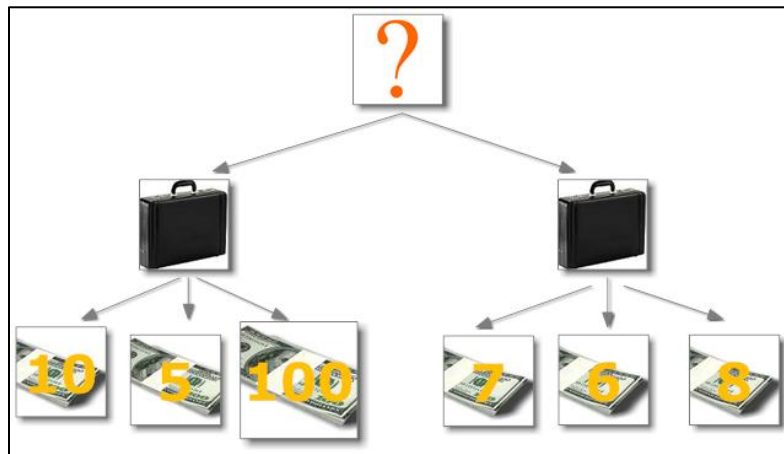
There are 8 cards, separated in two red stacks, which are separated again in two smaller blue stacks, each blue stack consisting of two cards. This is a two player game, and you play first, after which you will alternate rounds. Both players can see the values of all cards and all stack compositions. In every turn you must choose one of the stacks of your color that hasn't been removed from the game. When you choose a stack, its same-sized stack will be removed from the game. The goal is to leave only the highest possible card in the game and remove all others. The rankings are 1-10, Jack, Queen, King, and Ass.



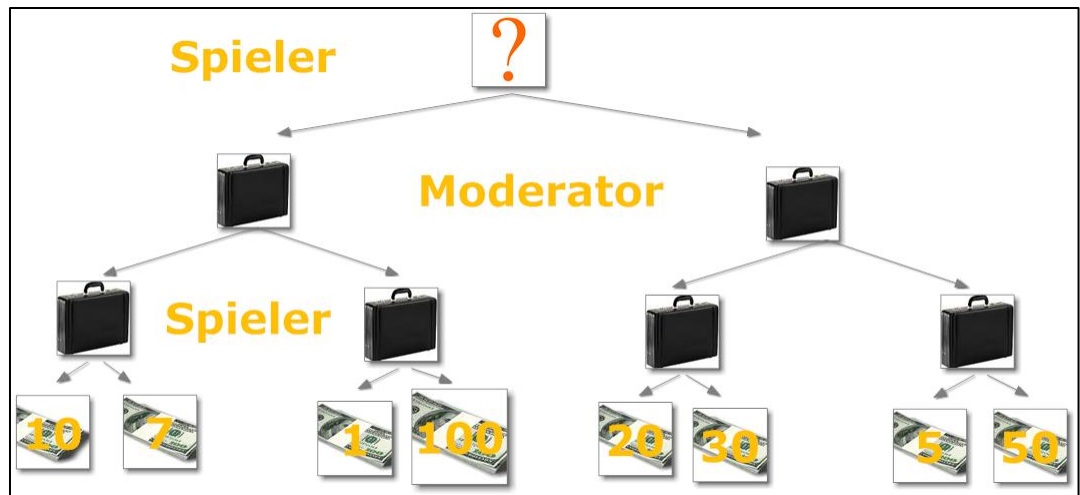
Suppose that the other player plays perfectly (that means, will play in such a manner that will leave you the lowest ranked possible card in the end). The question is what will be your best first move?

Do this with these cards, and then again with randomly shuffled cards in a similar composition. Note how much time you need to solve every composition, and try to improve that time by finding specific techniques for this. Now, when you get bored, get to the next game.

You are a contestant in a game show. There are two suitcases, each of them containing three separated amounts of cash. In your turn you can choose a case, and in the next turn the moderator will choose one of the three amounts of cash to give you. Now, the moderator is known to not give away any money, and knows exactly what he is doing, since he is playing this game since decades. Which case would you choose if you wanted to gain the maximum amount of money?



So you got lucky with this one. The moderator is impressed, and offers you another shot: the rules are the same, but now there are two cases inside of each case, and in those smaller cases are the money stacks. You play first once again. How will you play?



Now this was a bit harder, but you can do better! Expand the game with more cases. Let's make 5 levels out of it, so two suitcases containing two suitcases each, containing two suitcases each, containing two suitcases each, containing two cash stacks each. Those are $2+2^2+2^3+2^4=30$ suitcases and 32 cash stacks. The values of the stacks are random; just type any number between 1 and 100 for each of them. Try that game again. How long do you need to find the best move?