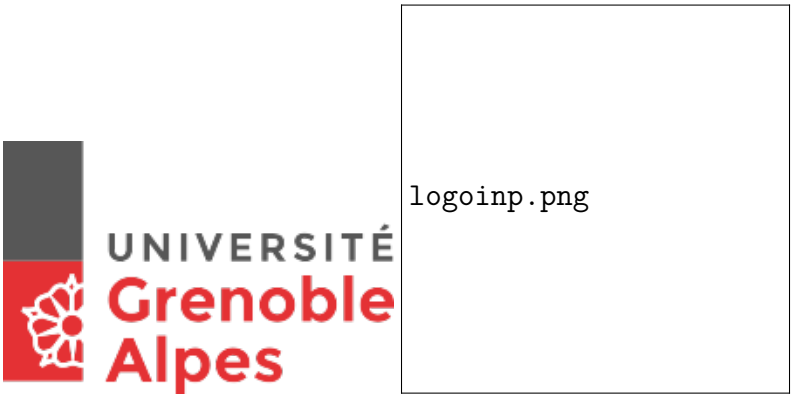
locale = FR



Master M1 MOSIG – UGA & Grenoble INP

Algorithmic Problem Solving

APP2: Hold'em for n00bs

Authors :
Habib SLIM
Eslam MOHAMMED
Archit YADAV
Albert STRÜMPLER
Manuel TREUTLEIN
Sofiane TANJI

Teacher :
Ms. Malin RAU

Last Version
October 11, 2019

Contents

1 Introduction and modelling

In this APP we have to deal with a card game played by two persons. One player is referred to as sister, the other player is referred to as strategist. A series of n cards lying on the table face up in a line. We are modelling the card deck through a double ended linked list, short deque. The elements of the deque are integers in the range of [2, 14], whereas the value 2 represents the card 2 and the value 14 represents the value of an ace. All values in between are assigned appropriately, this means in particular for the face cards that the jack is modeled by 11, the queen by 12 and the king by 13. The deque allows according to the game rules to take only the rightmost or the leftmost card. The players take turns while playing the game and always decide to take the leftmost or rightmost card. This is modeled through popleft() to take leftmost card and through pop() to take the rightmost card. Be careful, the naming of pop() for taking the rightmost card can be considered not consistent with popleft(), but we stick to the denotation of the phyton standard library. The player with the highest score in the end wins. Even though we will apply different algorithms to solve the problem, the input and ouput of the algorithm stays the same. The input is a list of cards represented as deque and the choice (of the sister) who starts the game represented as boolean. The output is 0 if the sister wins or the sum of the players are equal. The output is 1 if the player strategist wins.

In this APP we decided to change to python code for representing algorithms. The reason for this is that python code has a simple structure and resembles pseudo-code in a way. But furthermore it allows us to represent the developed algorithms more detailed.

The sister will always play the so called greedy algorithm. Therefore we first consider an algorithm applying this method in chapter 2. Because of some limitations of the greedy algorithm we will then consider a complete solution space exploration in chapter 3. This gives us an optimal solution, but with an unacceptable runtime. For this reason we will introduce a dynamic algorithm in chapter 4, resulting in an optimal solution with acceptable runtime.

2 The greedy algorithm

– Section 2 basically

2.1 Pseudocode

– write down the python pseudocode

2.2 Complexity

– short chapter about the complexity, maybe not necessary.

2.3 Limitations and advantages

– maybe make a table

3 Complete solution space exploration

– Section 3 basically

3.1 Pseudocode

– write down the python pseudocode

3.2 Complexity

– short chapter about the complexity.

4 Solution with dynamic programming

– Section 4 basically

4.1 Pseudocode

For the algorithm we make some assumptions to make the code easier to understand. This leads to the necessary to adapt the code under after assumptions. We will consider these later. The assumptions

- The player strategist always starts.
- There can never occur the situation that the sister chooses greedy between two cards with the same value.

– write down the python pseudocode

4.2 Complexity

– short chapter about the complexity.

4.3 The algorithm under different assumptions

– add the consideration of different assumptions for the dynamic programming.

5 Conclusion and feedback

– Write something in general