Mathematics for computer science

Homework 1

SID-LAKHDAR Riyane

15/10/2015

Résumé

blablbalaba lmds

1 Greedy algorithme

To solve the problem, we have first tried to consider it using a greedy point of view :
Let's consider that solving the problem is equivalent to find, at each step of the game, the best card to pick, regarding only the actual state of the game. This greedy method does not consider the possible evolutions of the game after this step.
In the following greedy algorithm description and analyze, we will first assume that the sister can use any strategy. In a second time, we will determine the complexity and the failure cases of the algorithme when the sister is choosing the biggest card at any step.

1.1 Algorithme descritpion

```
boolean winIsPossible(cards       : array of size N containing cards
                                    guiven by them values
                      sisterStart : boolean)
{
    int min         = 0;    // Lowest index of the deck in the input
        array
    int max         = N-1; // Highest index of the deck
    int sisterScore = 0;
    int myScore     = 0;

    // The variables used with "&" may be modified by the function
    if (sisterStarts) playSister(cards, &min, &max, &sisterScore);
    for (i=0; i< N/2; i++)
    {
        d0 = cards[min] - Math.max(cards[min+1], cards[max]);
        d1 = cards[max] - Math.max(cards[min]   , cards[max-1]);
        if (|d0| < |d1|) {myScore += cards[min]; min++;}
        else             {myScore += cards[max]; max--;}
        if (min != max) playSister(cards, &min, &max, &sisterScore);
    }
}
```

The function p̈laySister ẅill play according to the choosen sister's strategy. It will also update all the game parameters.
Assuming that it is our turn to pick a card, we can pick one card belonging to {cards[0], cards[N-1]}. Given one card of this set, we can compute the points we will earn, and the points the sister can earn. The aim of our algorithm is to chose, at each, step the card that will make the difference between our gain and the sister's gain as big as possible.

This algorithm has many obvious advantages. First of all, it is very easy to implement and to understand. No specific data structure or algorithm are required. Secondly, as we will prove in the next paragraph, this algorithm seems to have an interesting time complexity. However, as we do not consider the whole aspects of the problem, we can wonder if this algorithm will always give an optimal (or at least a working) solution ?

1.2 Time and space complexity of this greedy resolution

The algorithm realises in a N/2 length loop, 3 instruction with an O(1) complexity. Our algorithm complexity can be written as :

$$C(N) = \frac{N}{2} * 3 * O(1) = O(N) \tag{1}$$

Thus, the time complexity of our algorithm is linear.
For all the game steps, our algorithm uses the same set of 4 integers (in addition to the N cards array).
Thus, the space complexity of our algorithm is O(N + 4) = O(N)
You may have noticed that in our complexity analyze, we do not consider the complexity of the sister's decision algorithm. This is done on purpose, as the problem we deal with is to find the an algorithm which determines our decision. However, the next algorithms we have designed have to compute the sister's decision to evaluate our decisions. Thus, to make the comparison between all our algorithm more faire, we will give the complexity of the sister's algorithm in the computation of the time complexity of our greedy algorithm :
Assuming that at each step the sister pick up the biggest card, we can rewrite our time complexity as follows :

$$C(N) = \frac{N}{2} * (3 * O(1) + O(1)) = O(N) \tag{2}$$

1.3 Failure cases

Despite its easy implementation and its interesting complexity, the greedy algorithm we designed is not an acceptable way to solve the given problem.
An easy way to get convinced of this fact is to consider the following counterexample :

Index	0	1	2	3	4	5
Cards	4	4	6	6	12	9

Running our algorithm on this input deck would make us loose the game by choosing the following cards :

Index	0	1	2	3	4	5
Cards	4	4	6	6	12	9
Player ID	Sister 3	You 3	Sister 2	You 2	Sister 1	You 1

However, a solution which could make us win exists (respecting the same player's order).

Index	0	1	2	3	4	5
Cards	4	4	6	6	12	9
Player ID	You 1	Sister 3	You 3	Sister 2	You 2	Sister 1

This counterexample shows the limits of our method : By considering only the current card we can choose at a given step, we may allow the sister to access to a card which is much butter at the next step.
Thus, at a given step of the game, we should be able to consider the implications of the current choice on all stages of the game until the end. And that is the purpose of the improvement made by the next algorithm.