Tutorial 6: Dynamic programming II

1 A card game

Consider the card game where a sequence of n cards with values $v_1, \ldots, v_n$ are placed in a line on the table. Then two players take turns picking either the left-most or the right-most card left on the table.[1] At the end of the game, the player whose cards have the largest total value wins. In the event of a tie, both players are said to win. (In this game, we will always let n be even so that both players end up with the same number of cards.)

(i) The natural greedy strategy for Player 1 is to always take the card with the highest value (out of the left-most and the right-most cards available to be picked at the moment). Prove that this strategy does not guarantee that Player 1 always wins.

(ii) Design an algorithm with time complexity $\Theta(n^2)$ that computes a strategy for Player 1 that maximizes the total value of their cards, assuming that Player 2 always plays optimally. Given the initial sequence of card values $v_1, \ldots, v_n$, the algorithm should precompute some information in time $\Theta(n^2)$ in a way that afterwards, Player 1 can use this precomputed information to decide each move in time $\Theta(1)$.

(iii) Show that the strategy output by your algorithm guarantees that Player 1 will win the game.

[1] I.e., if the cards $i, i+1, \ldots, j$ are left on the table, the player going next can take either card i or card j.