Exercise Session n. 11

Algorithms and Data Structures

We practice a bit with dynamic programming.

Maximal Non-Adjacent Sum

Write an algorithm `max_non_adjacent_sum(A)` that, given an array A of $n \geq 1$ numbers, computes, with worst-case complexity $O(n)$, the maximal sum of zero or more non-adjacent elements in A. A subsequence of non-adjacent elements may include `A[i]` or `A[i+1]` but not both, for all i.

```
>>> max_non_adjacent_sum([2,9,6,2,6,8,5])
20
>>> max_non_adjacent_sum([-2,-1])
0
>>> max_non_adjacent_sum([2,3,2])
4
>>> A = [5,15,30,-9,39,29,40,0,-8,-3,-8,24,-5,2,14,4,28,19,1,23
>>> max_non_adjacent_sum(A)
203
>>> A = [10,3,13,9,13,7,2,9,6,9,15,13,15,2,14,1,5,9,15,
...      10,1,7,9,14,10,6,7,10,8,9,1,8,7,4,8,10,8,4,12,4]
>>> max_non_adjacent_sum(A)
191
>>> A = [14,13,7,6,11,2,19,1,16,8,20,12,18,14,17,12,8,13,15,7,
...      0,7,4,11,8,8,2,16,9,9,8,18,0,5,18,13,13,7,16,13,
...      3,1,12,9,0,20,0,12,6,11,18,9,1,13,16,19,10,17,8,17,
...      0,19,8,16,16,20,2,17,19,8,13,5,15,10,13,18,11,0,0,6,
...      14,19,5,9,2,6,19,-1,6,19,1,1,17,5,10,2,15,4,8,11]
>>> max_non_adjacent_sum(A)
645
```

Best Card Game

Consider the following game: you start with two decks of n playing cards each (shuffled). At each round, you remove one or two cards as follows. If the two cards at the top of the two decks have the same suit or the same numeric value, you may remove both of them

at no cost. If the two cards have different suits and numbers, or if you do not choose to remove both of them, you must remove one of the two cards at a cost corresponding to its numeric value. If one of the decks is empty, you have no choice: you must remove the card on the remaining deck at the cost of its numeric value. The game ends when both decks are empty. You must compute the best score, that is, the minimal cost to clear up both decks.

Write an algorithm `minimal_cost_game(A,B)` that takes an array A and an array B each containing n cards, and returns the minimal cost of clearing both decks. Each card is a tuple (v, s), where $v \in \{1, 2, \ldots, 13\}$ and $s \in \{1, 2, 3, 4\}$ is the suit.

```
>>> A = [(10, 1), (2, 1), (6, 2)]
>>> B = [(1, 1), (10, 1), (6, 1)]
>>> minimal_cost_game(A,B)
3
>>> A = [(9,4),(12,2),(6,2),(6,1),(4,4),(1,4),(5,2),(8,3),(8,1)
>>> B = [(9,4),(3,4),(2,4),(9,1),(9,3),(8,3),(1,1),(1,2),(9,2),
>>> best_game_score(A,B)
59
>>> A = [(2,4),(8,3),(4,2),(3,3),(7,3),(6,3),(6,2),(13,1),(7,4)
...      (13,3),(12,4),(5,3),(3,1),(13,2),(1,4),(10,2),(4,4),(1
>>> B = [(7,4),(11,4),(7,3),(12,4),(4,2),(3,1),(6,3),(8,4),(13,
...      (12,2),(5,3),(12,1),(1,3),(11,1),(9,3),(13,3),(10,1),(
>>> best_game_score(A,B)
78
```