CS CSI

2016-10-20 Thu

The major credit card company DoctorCard can either check a transaction to see if it's fraudulent or skip it. Each transaction comes with an estimated value for fraud checking (based on the size and suspiciousness of the transaction). Skipping a transaction provides no value. Checking if it's fraudulent provides the estimated value. However, because of computational costs, DoctorCard **must** skip checking the next two transactions if they check the current one.

We're interested in maximizing the total fraud value checked given such an array of estimated values A.

So, for example, the optimal transactions to check are underlined in the following transaction record: $[4, \underline{9}, 1, 10, 1, \underline{7}, 2, 2]$. Note that each checked (underlined) transaction must be followed by at least two unchecked transactions and that therefore the last two entries must always be unchecked.

Design an efficient algorithm to find the maximum total fraud value. Your algorithm need not find the actual transactions to check, just the value.[1] Your algorithm may use linear memory and time.

HINT: Start by designing a recurrence $F(n)$ that describes the optimal fraud value achievable for transactions $1, \ldots, n$ in terms of the optimal value(s) for smaller stretches of the array. Then, either convert that into a recursive solution and memoize it or convert that into a dynamic programming solution.

[1]If you want a justification of that: DoctorCard is comparing the value achieved by their online algorithm—i.e., algorithm that makes choices as transactions arrive—against the optimal total achievable.

1 CSI Finds the Culprit

Adapt your original algorithm so that it produces the optimal set of indexes to check for fraud rather than the maximum total value.

2 Forgetful CSI

Instead adapt your original algorithm so that it produces only the maximum total value but does it using constant memory.