ADA-2024: Homework-2

Deadline: 11th February, 2024; Full Marks: 25

Guidelines for submission: If your algorithm is based on dynamic programming, then you must write your solution using the following format in that order.

- Subproblem definition.
- Recurrence of the subproblem.
- The specific subproblem(s) that solves the actual problem.
- Algorithm description.
- Explanation of the running time of your algorithm.

Problem: Farmers Boggis, Bunce and Beans have set-up an obstacle course for Mr. Fox. The course consists of a long row of booths, each with a number written at the door of the booth. Formally, Mr. Fox is given an array $A[1, 2, \ldots, n]$ where $A[k]$ denotes the number written at the door of the k-th booth. Each number $A[k]$ can be positive, negative or zero. Everybody agrees to the following rules.

- At each booth, Mr. Fox must say either RING or DING.
- If Mr. Fox says RING at the k-th booth, then he earns a reward of $A[k]$ chickens if $A[k] > 0$ and pays a penalty of $-A[k]$ chickens if $A[k] < 0$.
- If Mr. Fox says DING at the k-th booth, then he earns a reward of $A[k]$ chickens if $A[k] < 0$ and pays a penalty of $-A[k]$ chickens if $A[k] > 0$.
- Mr. Fox is forbidden to say the same word more than three times a row. What it means is that if Mr. Fox says RING in booths 5, 6, and 7, then Mr. Fox must say DING at booth 8. Conversely if Mr. Fox says DING in booths 6, 7, 8, then he must say RING at booth 9.
- All accounts will be settled at the end after Mr. Fox visits every booth in the order given and the umpire calls "Hot Box". Mr. Fox will not have to carry chickens through the obstacle course.
- Finally, if Mr. Fox violates any of the rules, or if he ends the obstacle course owing the farmers chickens, the farmers will shoot him.

Design an efficient algorithm that computes the largest number of chickens that Mr. Fox earns by running the obstacle course given the array $A[1, 2, \ldots, n]$ of numbers as the input. The running time of your algorithm must be polynomial in n.

Solution. In this problem, no prepossessing is needed. Neither RING nor DING can be said by Mr. Fox consecutively in booths i, $i+1$, $i+2$, and $i+3$. Observe that at any i-th booth, the reward/penalty earned by Mr. Fox after saying RING or DING can be determined by $A[i]$ or $-A[i]$ respectively.

- **Subproblem Definition: (4 marks)** Here we define two subproblems,
  1. $DP(i,0)$ denotes the maximum profit can be earned in the booths $\{1,2,\ldots,i\}$ after saying DING at i-th booth where $i \in \{1,2,\cdots,n\}$.
  2. $DP(i,1)$ denotes the maximum profit can be earned in the booths $\{1,2,\ldots,i\}$ after saying RING at i-th booth where $i \in \{1,2,\cdots,n\}$.
- **Recurrence Relation: (8 marks)** For every $i \geq 4$.
  1. $DP(i,0) = \max \begin{cases} DP(i-1,1) - A[i] \\ DP(i-2,1) - A[i-1] - A[i] \\ DP(i-3,1) - A[i-2] - A[i-1] - A[i] \end{cases}$
  2. $DP(i,1) = \max \begin{cases} DP(i-1,0) + A[i] \\ DP(i-2,0) + A[i-1] + A[i] \\ DP(i-3,0) + A[i-2] + A[i-1] + A[i] \end{cases}$
- **Base Cases: (3 marks)** Here, we need to define three base cases for $i = 1,2,3$ as Mr. Fox can say any of the words three times consecutively.

  [i = 1]: $DP(i,0) = -A[1]$
  $DP(i,1) = A[1]$

  [i = 2]: $DP(i,0) = \max\{(A[1] - A[2]), (-A[1] - A[2])\}$
  $DP(i,1) = \max\{(A[1] + A[2]), (-A[1] + A[2])\}$

  [i = 3]: $DP(i,0) = \max\{DP(i-1,0) - A[3], DP(i-1,1) - A[3]\}$
  $DP(i,1) = \max\{DP(i-1,0) + A[3], DP(i-1,1) + A[3]\}$
- **Subproblem that solves the actual problem: (2 marks)** $\max\{DP(n,0), DP(n,1)\}$.
- **Algorithm Description: (6 marks)**
  Consider that $ArrRI[n]$ and $ArrDI[n]$ are two arrays each of size n, maintaining the profit for the word RING and DING. Here, we consider the array index to start from 1. Following is the algorithm.
  1. Initialize
     - $ArrRI[1] = A[1]$;
       $ArrDI[1] = -A[1]$
     - $ArrRI[2] = \max\{A[1] + A[2], -A[1] + A[2]\}$;
       $ArrDI[2] = \max\{A[1] - A[2], -A[1] - A[2]\}$
     - $ArrRI[3] = \max\{ArrRI[2] + A[3], ArrDI[2] + A[3]\}$;
       $ArrDI[3] = \max\{ArrRI[2] - A[2], ArrDI[2] - A[3]\}$
  2. FOR $i = 4,\cdots,n$ do

  - $ArrRI[i] = \max\{ArrDI[i-1] + A[i], ArrDI[i-2] + A[i-1] + A[i], ArrDI[i-3] + A[i-2] + A[i-1] + A[i]\}$
  - $ArrDI[i] = \max\{ArrRI[i-1] - A[i], ArrRI[i-2] - A[i-1] - A[i], ArrRI[i-3] - A[i-2] - A[i-1] - A[i]\}$

3. Output
   - $\max\{ArrRI[n], ArrDI[n]\}$

- **Running Time: (2 marks)** Initialization in step 1 takes $O(1)$ time. In step 2, the for loop runs $O(n)$ time, and each statement inside the loop takes $O(1)$ time to execute because we are directly accessing the array and finding the maximum of three elements. Finally, step 3 takes $O(1)$. hence the algorithm has $O(n)$ running time.