We want to design an algorithm for a cryptocurrency startup. We're given two arrays of positive number $N = \{N_1, \dots, N_T\}$ and $W = \{W_1, \dots, W_T\}$ which represent the profits from NorthCoins and WestCoins respectively. Each type of coins requires its own software which takes 1 unit of time to load.

1. First we define several subproblems. Let ProfitN(t) be the subproblem where the computer mines optimally for time intervals up to $t-1$, and mines NorthCoins at time t. Likewise define ProfitW(t) and ProfitL(t) so that the computer mines WestCoins or loads software at time t, respectively. We order subproblems by increasing t.

2. The base cases are as follows: $\text{ProfitN}(1) = N_1$, $\text{ProfitW}(1) = W_1$, and $\text{ProfitL}(1) = 0$.

3. The recurrence relations are as follows:

$$\text{ProfitN}(t+1) = N_{t+1} + \max(\text{ProfitN}(t), \text{ProfitL}(t)),$$

$$\text{ProfitW}(t+1) = W_{t+1} + \max(\text{ProfitW}(t), \text{ProfitL}(t)),$$

$$\text{ProfitL}(t+1) = \max(\text{ProfitN}(t), \text{ProfitW}(t)).$$

   In words, ProfitN($t+1$) is the maximum profit such that the computer *must* mine a NorthCoin at time $t+1$ (hence the N_{t+1} summand) and mines optimally before that (at time t, it either mined a NorthCoin or loaded software to be able to mine NorthCoins, whichever is better, hence the second summand). ProfitW($t+1$) is analogous.

   ProfitL($t+1$) is the maximum profit such that the computer *must* load software at time $t+1$ and mines optimally before that. Since it makes no sense for the computer to load software for two consecutive time intervals, the computer must have either mined NorthCoins or WestCoins at time t, and therefore we say that it did whichever gives a higher profit.

4. We prove that the recurrence relations are correct.

   Proof. We claim that the recurrence relation for ProfitN(t) is correct. Consider ProfitN(i) for some i. By assumption, the machine is mining optimally for time 1 to $i-1$ and must mine NorthCoins at time i. We have two cases (these are the only two cases, since these are the only possibilities which allow for feasible solutions where our algorithm solution can mine NorthCoins at time i):

   Case 1. Suppose the algorithm's machine mined NorthCoins at time $i-1$. Then in the optimal solution, the machine also mined NorthCoins at time $i-1$. It follows that in the optimal solution, we should mine NorthCoins at time i (loading software won't improve our profit, and mining WestCoins is impossible because we have the wrong software). Thus, we get that

$$\text{Opt}(i) = N_i + \text{ProfitN}(i-1).$$

   Case 2. Suppose the algorithm's machine loaded NorthCoin software at time $i-1$. Then in the optimal solution, the machine also loaded NorthCoin software at time $i-1$. It follows that in the optimal solution, we should mine NorthCoins at time i (loading software again won't improve our profit, and we don't have the right software to mine WestCoins). Thus, we get that

$$\text{Opt}(i) = N_i + \text{ProfitL}(i-1).$$

These are all possible cases. In each case, we've shown what the optimal solution must be. Combining cases, we find that in general,

$$\text{Opt}(i) = N_i + \max(\text{ProfitN}(i-1), \text{ProfitL}(i-1)).$$

Rewriting indices, we observe that this is exactly the recurrence relation for ProfitN(t):

$$\text{ProfitN}(t+1) = N_{t+1} + \max(\text{ProfitN}(t), \text{ProfitL}(t)).$$

The recurrence relation is therefore correct for ProfitN(t). □

Proof. We claim that the recurrence relation for ProfitW(t) is correct. The argument is exactly the same as for the recurrence relation for ProfitN(t). □

Proof. We claim that the recurrence relation for ProfitL(t) is correct. Consider ProfitL(i) for some i. By assumption, the machine is mining optimally for time 1 to $i-1$ and must load software at time i. Noting that it doesn't make sense for the machine to load software for two consecutive time intervals, then the algorithm machine must have mined either NorthCoins or WestCoins at time $i-1$. But we assumed that the algorithm machine was mining coins optimally at time $i-1$. So the algorithm machine mined NorthCoins if $\text{ProfitN}(i-1) > \text{ProfitW}(i-1)$ and WestCoins otherwise. If at some point k, the optimal solution loads software, it follows that at time $k-1$, the optimal machine mined NorthCoins if $\text{ProfitN}(k-1) > \text{ProfitW}(k-1)$ and WestCoins otherwise. Observing that these relationships are the same, it follows that the recurrence relation

$$\text{ProfitL}(t+1) = \max(\text{ProfitN}(t), \text{ProfitW}(t))$$

is correct. □

5. An algorithm for finding a solution to the subproblem is as follows: for given time t, calculate ProfitN(t), ProfitW(t), and ProfitL(t). The overall optimal profit is

$$\max(\text{ProfitN}(t), \text{ProfitW}(t), \text{ProfitL}(t)).$$

6. An algorithm for finding the optimal solution to the original problem is as follows:

   If $T = 1$, then the overall maximum profit is just $\max(N_1, W_1)$.

   Otherwise, create arrays ProfitN, ProfitW, and ProfitL. Initialize each according to the base cases from (2): let $\text{ProfitN}[0] = N_1$, $\text{ProfitW}[0] = W_1$, and $\text{ProfitL}[0] = 0$. For each time interval, fill in the next element in each of these arrays according to the recurrence relations in (3). The overall maximum profit is $\max(\text{ProfitN}(T), \text{ProfitW}(T))$. In pseudocode,

```
if size of north is 1, return max(north[0], west[0])

PN = [north[0]]
PW = [west[0]]
PL = [0]

for i = 0 to T-1:
PN[i+1] = north[i+1]+max(PN[i], PL[i])
PW[i+1] = west[i+1]+max(PW[i], PL[i])
PL[i+1] = max(PN[i], PW[i])

return max(PN[T], PW[T])
```

7. The `for` loop runs T times, and the operations inside the loop take $O(1)$ time to complete. The time complexity is therefore $O(T)$, linear time.