Assignment2

卫卓虹

2018/10/31

1 Money Robbing

1.1 Solution

We define that $m[i]$ represent the money he can get at house i, and $a[i]$ represent the money amount he gets when he arrives house i.

Since he can't rob adjacent houses, he can either rob house i-1 or house i,the best amount he can get is either $a[i-1]$ or $a[i-2]+m[i]$. Considering the margin, the largest money he can get at house 1 is m[1].The general form of sub-problem is to get largest amount of money when the robber arrives at house i.

DP Equation: $a[i] = max(a[i-1], a[i-2]+m[i])$

```
Input: money list
Output: the largest amount of money
function MONEY ROBBING($i$)
    if $i == 0$ then
        return 0
    end if
    if $i == 1$ then
        return m[1]
    end if
    $ans \leftarrow max($MONEY ROBBING$(i-2) + m[i],$ MONEY ROBBING$(i-1))$
    return $ans$
end function
```

1.2 Correctness Validation

Suppose for house i,there is a better value $aa[i]$,then if $a[i]$ choose to rob at house i-1,then $a[i-1] > a[i-2]+m[i]$. Since $aa[i]$ is a different choice,$aa[i]$ should be different from $a[i]$,which makes $a[i-2]+m[i]$ bigger than $a[i-1]$, a contradiction.

1.3 Complexity Analysis

For every recursion step, there is only constant level complexity.Thus T(n) = O(n)

2 Decoding

2.1 Solution

Suppose $n[i]$ is the ith number and $a[i]$ is the total number of decode ways from beginning to the ith place.Consider the i_{th} number in message, the total number of ways is the sum of $a[i-1]$ and $a[i-2]$ (if $n[i-1]$ and $n[i-2]$ combines a decodable number).

Now consider '0'. Since '0x' is illegal in this problem,it is essential to check if it can be combined with former number as '10' or '20'.

Suppose CHECK is a function to check whether n[i] or n[i],n[i-1] combined is decodable.

DP Equation: $a[i] = CHECK(n[i]) * a[i-1] + CHECK2(n[i], n[i-1]) * a[i-2]$

```
function CHECK($char\ one$)
    if $one ==' 0'$ then
        return 0
    else
        return 1
    end if
end function

function CHECK2($char\ one, char\ two$)
    if $one ==' 1' || (two ==' 2'$ and $one <=' 6')$ then
        return 1
    else
        return 0
    end if
end function

function DECODE WAYS($string\ s$)
    if $s.size() == 0 || s[0] ==' 0'$ then
        return 0
    end if
    if $s.size() == 1$ then
        return CHECK(s[0])
    end if
    $fn \leftarrow 0$
    $fn_1 \leftarrow$ CHECK($s$[0])) $*$ CHECK($s$[1]) $+$ CHECK2($s$[0], $s$[1])
    $fn_2 \leftarrow 1$
    for $i = 2$ to $s.size() - 1$ do
        $fn \leftarrow$ CHECK($s[i]$) $* fn_1 +$ CHECK2($s[i], s[i-1]$) $* fn_2$
    end for
    if $fn == 0$ then
        return 0
    end if
    $fn_2 \leftarrow fn_1$
    $fn_1 \leftarrow fn$
    $fn \leftarrow 0$
    **return** $fn_1$
**end function**
```

2.2 Correctness Validation

With the technique of loop-variant,we can prove it in following steps:
Initialization: When the size of string is 1,the number of ways to decode it is 1 when message is not '0'.
Maintenance: Since the total number of ways is only affected by two numbers before the i_{th} number,the sum of the two ensures the sum of ways.
Termination: At termination,decode ways of the last number represent the total decode ways.

2.3 Complexity Analysis

For every recursion step, there is only constant level complexity.Thus T(n) = O(n)

3 Maximum profit of transanctions

3.1 Solution

Let prices be the stock price array with length n, i denote the i_{th} day (i will go from 0 to $n-1$), k denote the maximum number of transactions allowed to complete, $T[i][k][0]$ denotes the maximum profit at the end of the i_{th} day with at most k transactions and with 0 stock in our hand after taking the action, while $T[i][k][1]$ denotes the maximum profit at the end of the i_{th} day with at most k transactions and with 1 stock in our hand after taking the action. The number of stocks held in our hand is the hidden factor mentioned above that will affect the action on the i-th day and thus affect the maximum profit. Apparently we have base cases: $T[-1][k][0] = T[i][0][0] = 0$, that is, no stock or no transaction yield no profit (note the first day has $i = 0$ so $i = -1$ means no stock). And $T[-1][k][1] = T[i][0][1] = -Infinity$ .It emphasizes the fact that it is impossible for us to have 1 stock in hand if there is no stock available or no transactions are allowed.

We have recursion relation as belows:

DP Equation:

$T[i][2][0] = max(T[i-1][2][0], T[i-1][2][1] + prices[i])$
$T[i][2][1] = max(T[i-1][2][1], T[i-1][1][0] - prices[i])$
$T[i][1][0] = max(T[i-1][1][0], T[i-1][1][1] + prices[i])$
$T[i][1][1] = max(T[i-1][1][1], -prices[i])$

Input: *price list*
Output: *max profit*

```
function MAX PROFIT($Array$)
    for $i = 0$ to $Array.size()$ do
        $T[i][1][0] \leftarrow 0$
        $T[i][1][1] \leftarrow -inf$
        $T[i][2][0] \leftarrow 0$
        $T[i][2][1] \leftarrow -inf$
    end for
    for $i = 2$ to $Array.size() - 1$ do
        $T[i][2][0] \leftarrow max(T[i][2][0], T[i][2][1] + Array[i])$
        $T[i][2][1] \leftarrow max(T[i][2][1], T[i][1][0] - Array[i])$
        $T[i][1][0] \leftarrow max(T[i][1][0], T[i][1][1] + Array[i])$
        $T[i][1][1] \leftarrow max(T[i][1][1], -Array[i])$
    end for
    return $T[Array.size() - 1][2][0]$
end function
```

3.2 Correctness Validation

For $T[i][k][0]$ in the recurrence relations, the actions taken on the i_{th} day can only be rest and sell, since we have 0 stock in our hand at the end of the day. T[i-1][k][0] is the maximum profit if action rest is taken, while T[i-1][k][1] + prices[i] is the maximum profit if action sell is taken.

Suppose for equation $T[i][2][0] = max(T[i-1][2][0], T[i-1][2][1] + prices[i])$,there is a better value $TT[i][2][0]$,then if $T[i][2][0]$ choose to take rest, $T[i-1][2][0] > T[i-1][2][1]+prices[i]$. Since $TT[i][2][0]$ is a different choice,$TT[i][2][0]$ should be different from $T[i][2][0]$,which makes $T[i-1][2][1]+prices[i]$ bigger than $T[i-1][2][0]$, a contradiction.

$T[i][k][1]$ is the same.

3.3 Complexity Analysis

For every recursion step, there is only constant level complexity.Thus T(n) = O(n)