091M4041H - Assignment 2

胡鹏飞 计控学院 201828007329009

1 Money robbing

A robber is planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security system connected and it will automatically contact the police if two adjacent houses were broken into on the same night.

1. Given a list of non-negative integers representing the amount of money of each house, determine the maximum amount of money you can rob tonight without alerting the police.

2. What if all houses are arranged in a circle?

1.1 Optimal substructure and DP equation

首先考虑房子为一排的情况，假设共有 n 间房子，第 i 间房屋的钱财为 v[i],强盗到第 i 间房屋时所盗取的钱财总额为 S[i],由题目可知，为了躲避警察，强盗不能同时盗取相邻两间房屋，即如果决定盗取第 i 间，那么就不能盗取第（i-1）间，如果已经盗取第（i-1）间，那么第 i 间就不能再盗取，如果想要盗取的钱财总额最大，那么就需要比较这两种情况，取其大者即可，表达式为：

$$S[i]=\begin{cases} S[i]=0 & i=0 \\ S[i]=v[1] & i=1 \\ S[i]=\max(S[i-1],S[i-2]+v[i]) & i>=2 \end{cases}$$

1.2 Pseudo-code

```
FindMaxMoeny(v[1…n])
    if n == 0 then
        return 0
    end if
    if n == 1 then
        return v[1]
    end if
    for i from 2 to n do
        S[i] = max(S[i - 1], S[i - 2] + v[i])
    end for
return S[n]
```

1.3 Prove the correctness

首先，如果 n 为 0，即这条街没有房子，很显然盗取金额为 0；如果 n 为 1，即这条街只有一间屋子，如果盗取金额为 S[1]=v[1]，如果不盗取，S[1]=0，很显然，max（v[1],0）=v[1]；如果 n>1，对于第 i 间，首先看第（i-1）间是否盗取，如果第（i-1）间盗取，那么第 i 间不能盗取，S[i]=S[i-1]，如果第（i-1）间未盗取，那么第 i 间可盗取可不盗取，，如盗 S[i]=S[i-2]+v[i],如不盗，S[i]=S[i-2]，很显然，max((S[i-2]+v[i]), S[i-2])=(S[i-2]+v[i]),所以对于第 i 间，S[i] = max(S[i-1], S[i-2] + v[i])，算法无误。

1.4 Analyse the complexity

由伪代码可以看出，只遍历了一次数组，每次遍历的复杂度为 n，所以时间复杂度为：

$$T(\mathrm{n}) = O(\mathrm{n})$$

1.5 All houses are arranged in a circle

相比于刚才的一条直线的情况，如果连成环，需要额外考虑的是起始点和终结点如果是一个点，如果选择盗取，那么就会被发现，所以将刚才的 1-n 的问题分成 1-（n-1）和 2-n 两部分分别考虑即可，伪代码为：

```
FindMaxMoeny_circle(v[1…n])
    v1=v[1…(n-1)]
    v2=v[2…n]
    max1 = FindMaxMoeny(v1)
    max2 = FindMaxMoeny(v2)
    S[n] = max(max1,max2)
return S[n]
```

对于时间复杂度，只是计算了两次直线的情况，所以时间复杂度仍为：

$$T(\mathrm{n}) = O(\mathrm{n})$$

3 Decoding

A message containing letters from A-Z is being encoded to numbers using the following mapping:

A : 1
B : 2
: : :
Z : 26

Given an encoded message containing digits, determine the total number of ways to decode it.

3.1 Optimal substructure and DP equation

假设字符串为 a[0…n]，对于任意 i，a[0…i]的编码种类个数为 s[i]，首先，应该有 s[i]=s[i]+s[i-1]，如果(a[i-1]a[i])组成的数在 10 到 26 之间，那么还有 s[i]=s[i]+s[i-2]，但是还有特殊有 0 的情况，所以综合来看，

（1） 应该首先判断当前字符是否为 0，如为 0，则查看前一位，前一位如果为 1 或 2，则 0 只能和前一位组合，s[i]= s[i-2]，如果前一位大于 2，则无法编码；

（2） 其次，如果当前字符不为 0，那么查看前一个字符，如果前一个字符为 1，那么既可以组合编码也可以独立编码，s[i]=s[i-1]+s[i-2]，如果前一个字符为 2 且当前字符小于 7，同样 s[i]=s[i-1]+s[i-2]，

（3） 除此之外，只能独立编码，s[i]= s[i-1]。

3.2 Pseudo-code

```
Decoding(string a)
    if a is empty or a[0]=='0' then
        return 0;
    end if
    s[0]=1;
    for i from 1 to n do
        s[i]=(a[i-1]=='0'?0:s[i-1]);
        if i>1 and a[i-2]=='1'  or  a[i-2]=='2' and a[i-1]<'7' then do
```

```
            s[i]+=s[i-2];
        end if
        return s[i]
    end for
```

3.3 Prove the correctness

对于当前位置，如果不为 0，首先 s[i]+=s[i-1]，如为 0，不能独立，只能和前一位置组合，算法无误；其次，若当前位置不仅可以独立，还可以和前一位置组合成为 10-26,则 s[i]+=s[i-2]，算法无误。

3.4 Analyse the complexity

算法要遍历一次，即

$$T(\mathrm{n}) = O(\mathrm{n})$$

5 Maximum profit of transactions

You have an array for which the i-th element is the price of a given stock on day i.
Design an algorithm and implement it to find the maximum profit. You may complete at most two transactions.

5.1 Optimal substructure and DP equation

假设 prices[i]表示第 i 天股票的价格，首先查看只能交易一次的话，那么从左到右选取当天和之前的最小值的差与前一天最大利益之间的较大者即为当天的利益，即 left[i] = max(prices[i]-curmin, left[i-1])；由于题目说明只能在第一次出售之后才能购买第二次，所以要计算两次交易的最大值，可以逆序扫描数组，与交易一次类似，从右到左选取之前的最大值和当天的差与前一天最大利益的较大者即为当天的利益最大值，即 right[i] = max(curmax-prices[i], right[i+1])，最后遍历每一天，maxprofit = max(maxprofit, left[i] + right[i])。

5.2 Pseudo-code

```
FindMaxProfit(int prices[n]):
    if len(prices)<=1 then do
        return 0
    end if
    curmin = prices[0]
    for i form 1 to n do
        curmin = min(curmin, prices[i])
        left[i] = max(prices[i]-curmin, left[i-1])
    end for
    curmax = prices[-1]
    for i form n-1 to 0 do
        curmax = max(curmax, prices[i])
        right[i] = max(curmax-prices[i], right[i+1])
    end for
    maxprofit = 0
```

```
    for i form 0 to n do
        maxprofit = max(maxprofit, left[i] + right[i])
    end for
return maxprofit
```

5.3 Prove the correctness

股票在 n 天有 n 个价值，计算这 n 天交易两次获取的最大利益可以转换为求第 i 天的价值，由于交易两次且两次时间上不能有重合，所以选择从不同方向两次遍历数组，前向遍历表示这一天卖出或者在这一天已经卖出获取的最大利益，后续遍历表示在这一天买入或者在这一天之后再买入可以获得最大利益，然后遍历，将每一天的这两个值相加，选取其中和最大的一天即为可以获得的最大利益，算法无误。

5.4 Analyse the complexity

因为遍历了三次数组，所以时间复杂度为：

$$T(\mathrm{n}) = O(\mathrm{n})$$