Ch4

1.假设输入整数数组为 $A[1:n]$。用函数 $f(i)$表示以第 i 个数结尾的子数组的最大和，需要求解的是则我们需要求出 $max[f(i)]$，其中 $0\leq i<n$。对于 $f(i)$，若$f(i-1)\leq 0$，则$A[i]$就是第 i 个数结尾的子数组的最大和；否则$f(i-1)+A[\mathrm{i}]$是第 i 个数结尾的子数组的最大和。因此有如下递推式:

$$f(i)=\begin{cases}A[i] & i=1\text{ 或}f(i-1)\leq 0\\ f(i-1)+A[\mathrm{i}] & i\neq 1\text{ 且}f(i-1)>0\end{cases}$$

算法 **FindMaxSubseq**(A)

输入 :整数 A [1:n]

输出 :A 中连续子数组最大的和

For i=1 **TO** n **DO**
If i==1 $f(i)=A[i]$
Else if $f(i-1)\leq 0$ $f(i)=A[i]$
Else $f(i)=f(i-1)+A[i]$
return $max(f(i))$

求解$f(n)$的操作次数为 $O(n)$，查找$f(n)$中最大值的操作次数为 $O(n-1)$，所以算法总的时间复杂度为 $O(n)$。

2.用$f(i)$表示爬 i 阶楼梯的方法数，对于$f(i)$，$f(i+1)$有两种情况：下次走 1 步，则$f(i+1)=f(i)$；下次走 2 步，则 $f(i+1)=f(i-1)$，恰好满足斐波那契数列的性质。因此有如下递推式:

$$f(i)=\begin{cases}1 & i=1\\ 2 & i=2\\ f(i-1)+f(i-2) & i\geq 3\end{cases}$$

算法 **FindStairsNum**(n)

输入 :正整数 n

输出 :爬 n 阶楼梯的方法数，每次可以爬 1/2 个台阶

For i=1 **TO** n **DO**
If i==1 $f(i)=1$
Else if i==2 $f(i)=2$
Else $f(i)=f(i-1)+f(i-2)$
return $f(n)$

求解$f(n)$的操作次数为 $O(n)$，假定 n 不定长，则输入规模为 $\log n$，所以算法总的时间复杂度为 $O(2^{\log n})$。

3.以“ababa”为例，如果已知“bab”是回文，则仅需判定左右字母是否相同即可。假设输入字符串数组为 $A[1:n]$，用布尔数组 $f[i,j]$表示第 i 到 j 位是否为回文串，得到如下递推式:

$$f[i,\ j]=\begin{cases}true & i=j\\ true & j=i+1\text{且}A[i]=A[j]\\ f[i+1,j-1] & A[i]=A[j]\end{cases}$$

算法 **FindPalindromSeq**(A)

输入 :字符数组 A [1:n]

输出 :字符数组 A 的最长回文子串

For i=1 **TO** n-1 **DO**
$f[i, i]$=true, $j = i$
while $f[i, j]$==true

If $A[i+1]==A[i]$ $j=i+1$//初始回文序列为两个字符
If $((i-1)>0)\&\&((j+1)<n))$ $f[i-1,j+1]=f[i,j]$, $i=i-1$, $j=j+1$
return $f[i,j]$中为 true，且 j 和 i 相差最大的回文子串 $A[i:j]$

求解 $f[i,j]$的操作次数至多为 $O(n^2)$，遍历 $f[i,j]$中为 true 的节点、找出 j 和 i 相差最大的回文子串 $A[i:j]$的次数为 $O(n)$，又输入规模为 n，所以算法总的时间复杂度为 $O(n^2)$。

4.假定 $A[m,n]$表示第 m 行 n 列网格中的数值；$f[m,n]$表示从左上角到第 m 行 n 列网格的最小路径和。对于第 i 行第 j 列的网格，它只可能是它上方或左方的网格遍历而来，这样选择 $f[i-1,j]$和 $f[i,j-1]$中较小的即可。计算路径时，可通过右下角 $f[m,n]$逆推，减去 $A[m,n]$，观察其值等于左方还是上方网格的 f 值。因此有如下递推式:

$$f[i,j]=\begin{cases} A[i,j] & i=1 \text{ 且 } j=1 \\ min\{f[i,j-1],\ f[i-1,j]\}+A[i,j] & \text{其他} \end{cases}$$

算法 **FindMinSumPath**(A)

输入 :$m*n$ 的网格 A

输出 :从网格左上角到右下角总和最小的路径

根据 $f[1,1]$更新第一行和第一列的 f 值
For i=2 **TO** m **DO**
For j=2 **TO** n **DO**
If $f[i,j-1]<f[i-1,j]$ $f[i,j]=f[i,j-1]+A[i,j]$
Else $f[i,j]=f[i-1,j]+A[i,j]$
If $((i-1)>0)\&\&((j+1)<n))$ $f[i-1,j+1]=f[i,j]$, $i=i-1$, $j=j+1$
//打印路径
$i=m$, $j=n$
While $i!=1\&\&j!=1$
Print(i,j)
If $f[i,j]-A[i,j]==f[i-1,j]$
$i=i-1$, Print("↑")
Else $j=j-1$, Print("←")

最终输出的是从右下角到左上角的逆序路径，反着看就是题目要求的结果。求解 $f[i,j]$的操作次数为 $O(n^2)$，输出路径的复杂度至多为 $O(m+n)$，所以算法总的时间复杂度为 $O(n^2)$。

5.假设 $f[i,j]$表示第 i 个时刻，在第 j 个位置上最多接到的馅饼数。则 $f(0,5)$就是 Tom 最多接到的馅饼数，有如下递推式:

$$f[i,j]=\begin{cases} P[i,j] & i=\mathrm{T} \\ max\{f[i+1,j-1],\ f[i+1,j],f[i+1,j+1]\}+P[i,j] & \text{其他} \end{cases}$$

算法 **FindMaxPieNum**(P)

输入 :第 i 秒在 j 位置掉落的馅饼数量 $P(i,j)$

输出 :Tom 初始站在位置 5，经过 T 时刻最多接到的馅饼数

For j=0 **TO** 10 **DO**
$f[T,j]=P(T,j)$
For i=T-1 **TO** 1 **DO**
For j=0 **TO** 10 **DO**
If j=0 $f[i,j]=max\{f[i+1,j],f[i+1,j+1]\}+P[i,j]$

Else if j=10 $f[i,j]=max\{f[i+1,j-1],f[i+1,j]\}+P[i,j]$

Else $f[i,j]=max\{f[i+1,j-1],\ f[i+1,j],f[i+1,j+1]\}+P[i,j]$

return $f[0,5]$

赋初值的复杂度为 $O(j)$，动态规划计算的复杂度为 $O(T*j)$，返回结果的复杂度为 $O(1)$，所以算法总的时间复杂度为 $O(T*j)$。