第二章　算法的复杂性分析

2.1 常用的函数和公式

2.2 算法的时间复杂性分析

2.3 最好情况、最坏情况和平均情况分析

2.4 用生成函数求解递归方程

2.5 用特征方程求解递归方程

2.6 用递推方法求解递归方程

2.7 算法的空间复杂性

2.8 最优算法

2.1 常用的函数和公式

- 整数函数
- 对数函数
- 排列、组合和二项式系数
- 级数求和

序列求和的方法

- 数列求和公式

  等差、等比数列与调和级数

$$\sum_{k=1}^{n} a_k = \frac{n(a_1 + a_n)}{2}$$

$$\sum_{k=0}^{n} aq^k = \frac{a(1-q^{n+1})}{1-q}, \quad \sum_{k=0}^{\infty} aq^k = \frac{a}{1-q}(q < 1)$$

$$\sum_{k=1}^{n} \frac{1}{k} = \ln n + O(1)$$

求和的例子

$$\sum_{t=1}^{k} t2^{t-1} = \sum_{t=1}^{k} t(2^t - 2^{t-1})$$ 拆项

$$= \sum_{t=1}^{k} t2^t - \sum_{t=1}^{k} t2^{t-1} = \sum_{t=1}^{k} t2^t - \sum_{t=0}^{k-1} (t+1)\,2^t$$ 变限

$$= \sum_{t=1}^{k} t2^t - \sum_{t=0}^{k-1} t2^t - \sum_{t=0}^{k-1} 2^t$$ 拆项

$$= k2^k - \left(2^k - 1\right) = (k-1)2^k + 1$$

2.2 算法的时间复杂性分析

- 百鸡问题中统计的算法的时间花费，不表示该算法的实际执行时间。
  - 理想的计算模型下统计的；
  - 统计初等操作数目时，忽略了编译程序所产生的辅助操作；
  - 算法在运行时，很多操作是在运行过程中才判断是否执行。

- 希望以一个常数因子得到算法运行时间的阶，估计运行时间与输入规模的关系。
- 算法时间复杂性的分析方法：
  - 循环次数的统计
  - 基本操作频率的统计
  - 计算步的统计

2.2.1 循环次数的统计

算法的运行时间 → 循环次数 → 算法的输入规模

正比　　联系

例2.1：计算多项式：

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

Horner 法则改写：

$$P(x) = \left((a_n x + a_{n-1})x + \cdots + a_1\right)x + a_0$$

输入：存放多项式系数的数组 A[ ], 实数 *x*, 多项式的阶 *n*

输出：多项式的值

```
1. float polynomial(float A[ ],float x,int n)
2. {
3.     int  i;
4.     float value = A[n];
5.     for (i=n-1;i>=0;i--)
6.         value = value * x + A[i];
7.     return value;
8. }
```

循环控制变量赋初值的单位时间数c_1

循环体的单位时间数c_2

$$T(n) = c_1 + c_2 n = \Theta(n)$$

例2.2：冒泡排序算法

```
1. template <class T>
2. void bubble(Type A[],int n)
3. {
4.     int i,k;
5.     for (k=n-1;k>0;k--) {
6.         for (i=0;i<k;i++) {
7.             if (A[i] > A[i+1]) {
8.                 swap(A[i],A[i+1]);
9.             }
10.        }
11.    }
12. }
```

```
13. void swap(Type &x,Type &y)
14. {
15.     Type temp;
16.     temp = x;
17.     x = y;
18.     y = temp;
19. }
```

$\bar{c}$：辅助操作的执行时间　　c：循环体的平均执行时间

算法总的执行时间 T(n) 为

$$T(n) = \big((n-1) + (n-2) + \cdots + 1\big)c + \bar{c} = \frac{c}{2}n(n-1) + \bar{c} = \Theta(n^2)$$

例2.3：选手的竞技淘汰比赛

有$n = 2^k$位选手进行竞技淘汰比赛，最后决出冠军。

用如下的函数：

BOOL comp(Type mem1,Type mem2)

模拟两位选手的比赛，胜则返回 **TRUE**，否则返回 **FALSE**

假定可以在常数时间内完成函数 **comp** 的执行。

输入：选手成员 group[ ], 选手个数 n

输出：冠军的选手

```
1.template<class Type>
2.Type game(Type group[ ],int n)
3. {
4.     int j,i = n;
5.     while (i>1) {
6.         i = i / 2;
7.         for (j=0;j<i;j++)
8.             if (comp(group[ j+i ],group[ j ]);
9.                 group[ j ] = group[ j+i ];
10.    }
11.    return group[0];
12. }
```

当 $n = 2^k$时, while 循环体共执行k次， for 循环体的执行次数分别为n/2，n/4，…，1

- 函数 comp 可以在常数时间内完成
- 算法的执行时间为：

$$
\begin{aligned}
T(n) &= \frac{n}{2} + \frac{n}{4} + \cdots + \frac{n}{n} \\
&= n\left(\frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^k}\right) \\
&= n\left(1 - \frac{1}{2^k}\right) \\
&= n - 1 \\
&= \Theta(n)
\end{aligned}
$$

例2.4：洗牌 :对 n张牌进行 n 次洗牌

规则如下：

在第 k 次洗牌时（ $k = 1,\ldots n$ ），对第 i 张牌（ $i = 1, \ldots n/k$ ）随机地产生一个小于 n 的正整数 d，互换第 i 张牌和第 d 张牌的位置。

```
1. template <class Type>
2. void shuffle(Type A[ ],int n)
3. {
4.     int  i,k,m,d;
5.     random_seed(0);
6.     for (k=1;k<=n;k++) {
7.         m = n / k ;
8.         for (i=1;i<=m;i++) {
9.             d = random(1,n);
10.            swap(A[i],A[d]);
11.        }
12.    }
13. }
```

外循环的循环体共执行 n 次

内循环的循环体分别执行n, $\lfloor n/2 \rfloor, \lfloor n/3 \rfloor, \ldots, \lfloor n/n \rfloor$次

算法的执行时间为内部for循环的循环体的执行次数乘以一个常数时间，因此有：

$$T(n) = \sum_{i=1}^{n} \left\lfloor \frac{n}{i} \right\rfloor$$

因为：$\sum_{i=1}^{n} \left(\frac{n}{i} - 1 \right) \leq \sum_{i=1}^{n} \left\lfloor \frac{n}{i} \right\rfloor \leq \sum_{i=1}^{n} \frac{n}{i}$

由调和级数的性质，有：$\ln(n+1) \leq \sum_{i=1}^{n} \frac{1}{i} \leq lnn + 1$

因为：$\frac{\log(n+1)}{loge} \leq \sum_{i=1}^{n} \frac{1}{i} \leq \frac{logn}{loge} + 1$

所以：$\frac{1}{loge} \mathrm{nlog(n+1)} - \mathrm{n} \leq \sum_{i=1}^{n} \left\lfloor \frac{n}{i} \right\rfloor \leq \frac{1}{loge} nlogn + n$

$$T(n) = \Theta(nlogn)$$

2.2.2 基本操作频率的统计

- 选取算法中的一个初等操作作为基本操作；
- 估计这个操作在算法中的执行频率：
  - ✓ 所选取的基本操作，在算法中的执行频率必须至少和算法中的任何其他操作一样多。

1. 基本操作的定义

定义：算法中的某个初等操作，如果它的最高执行频率，和所有其它初等操作的最高执行频率，相差在一个常数因子之内，就说这个初等操作是一个基本操作。

- 初等操作的执行频率，可正比于任何其它操作的最高执行频率；
- 基本操作的选择，必须反映出该操作随着输入规模的增加而变化的情况。

基本操作与输入规模

- 算法时间复杂度：针对制定基本操作，计算算法所做运算次数；
- 基本操作：比较，加法，乘法，置指针，交换…
- 输入规模：输入串编码长度
  - 通常用下述参数度量：数组元素多少，调度问题的任务个数，图的顶点数与边数等。
- 算法基本操作次数可表示为输入规模的函数；
- 给定问题和基本操作就决定了一个算法类。

输入规模

- 排序：数组中元素个数n
- 检索：被检索数组的元素个数n
- 整数乘法：两个整数的位数m，n
- 矩阵相乘：矩阵的行列数i, j, k
- 图的遍历：图的定点数n，边数m

……

基本操作

- 排序：元素之间的比较
- 检索：被检索元素n与数组元素的比较
- 整数乘法：每位数字相乘（位乘）1次m位和n位整数相乘要做次$m\ n$位乘
- 矩阵相乘：每对元素乘1次

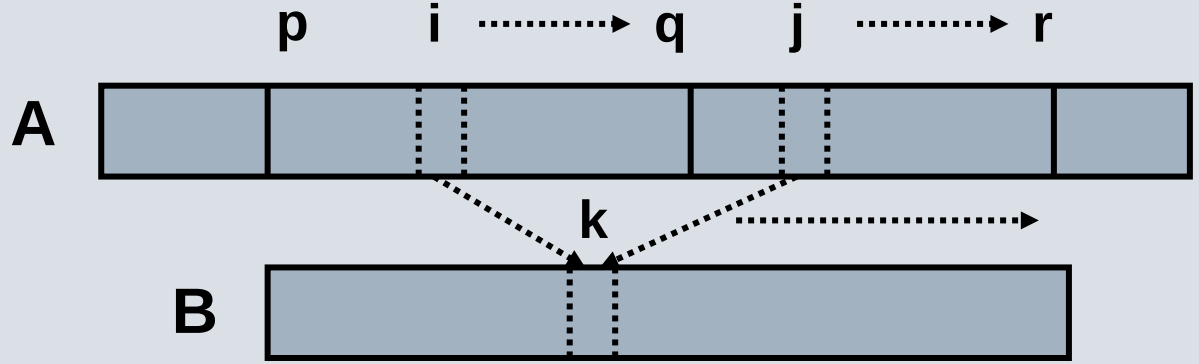
  矩阵$i * j$与$j * k$矩阵相乘要做$i\ j\ k$次乘法
- 图的遍历：置指针

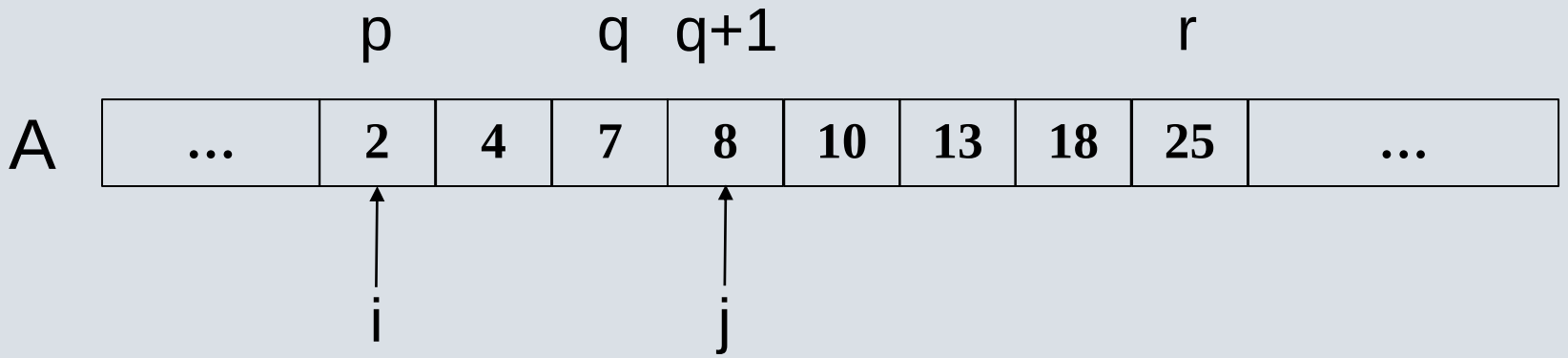
……

2. 用基本操作的执行频率估计算法的时间复杂性

例2.5 合并两个有序的子数组

A 是一个具有 m 个元素的整数数组，给定三个下标：p，q，r， $0 \le p \le q \le r < m$，A[p] ~ A[q]，A[q+1] ~ A[r] 分别是两个以递增顺序排序的子数组。把这两个子数组按递增顺序合并在 A[p] ~ A[r] 中。





2	4	7	8	10	13	18	25

例2.5：合并两个有序的子数组

```
1. void merge(int A[ ],int p,int q,int r)
2. {
3.     int  *bp = new int[r-p+1];  // 分配缓冲区,存放被排序的元素
4.     int  i,j,k;
5.     i = p;   j = q + 1;   k = 0;
6.     while (i<=q && j<=r) {      //  逐一判断两子数组的元素
7.         if (A[ i ] <=A[ j ])          // 按两种情况,把小的元素拷贝到
8.             bp[ k++ ] = A[ i++ ];    // 缓冲区*/
9.         else
10.            bp[ k++ ] = A[ j++ ];
11.    }
```

```
12.    if (i==q+1)                          // 按两种情况,处理其余元素
13.        for (;j<=r;j++)
14.            bp[++] = A[j];           // 把A[ j ]~A[ r ]拷贝到缓冲区
15.    else
16.        for (;i<=q;i++)
17.            bp[++] = A[i];            // 把A[i]~A[q]拷贝到缓冲区
18.    k = 0;
19.    for (i=p;i<=r;i++)                   // 最后,把数组bp的内容拷贝
20.        A[i++] = bp[k++];            // 到 A[p]~A[r]
21.    delete bp;
22. }
```

基本操作的选择：

1. 数组元素的赋值操作作为基本操作，操作频率：$2n$

1）随输入规模的增大而增加；

2）执行频率与其它操作的执行频率相差一个常数因子。

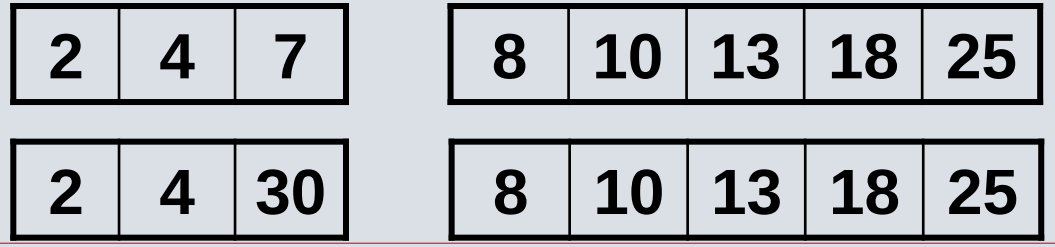
2. 数组元素的比较操作作为基本操作：

令两个子数组的大小分别为 n_1 和 n_2，$n_1 + n_2 = n$

数组元素的比较次数，

最少为 $\min(n_1,n_2)$ 次，最多为 $n-1$ 次。

算法的时间复杂性 $\Theta(n)$



2.2.3 计算步的统计

1. 计算步

定义：计算步是一个语法或语义意义上的程序段，该程序段的执行时间与输入实例无关。

例：flag =(a+b+c==n)&&(5*a+3*b+c/3==n)&&(c%3==0);

a = b;

和输入规模无关。连续 200 个乘法操作可为一个计算步，*n*次加法不能作为一个计算步。

2. 计算步的统计

把全局变量 count 嵌入实现算法的程序中，每执行一个计算步，count 就加1。算法运行结束时， count 的值，就是算法所需执行的计算步数。计算步数随输入实例规模变化而变化。

2.3 最好情况、最坏情况和平均情况分析

一、最好情况、最坏情况和平均情况

1. 影响运行时间的因素

   - ✓ 问题规模的大小
   - ✓ 输入的具体数据（算法性能除外）

2. 算法时间复杂性的三种分析

   最坏情况的分析、平均情况的分析和最好情况的分析

例2.7：插入排序

j:有序序列中的最后一个元素

一般情况的工作过程

i:待插入元素

		0	1	2	3	4	5	j
i	a	3	6	1	7	9	8	
1	6	3	6					1
2	1	1	3	6				2
3	7				7			1
4	9					9		1
5	8					8	9	2

当前待插入元素

比较次数

插入排序

```
insert_sort(A)
1. for i←1 to n-1
2.     a ← A[i]
3.     j ← i-1
4.     while j>=0 and A[j]>a  do
5.         A[ j+1 ] = A[ j ]
6.         j ← j- -
7.     A[ j+1 ] ← a
```

一般情况的工作过程

		0	1	2	3	4	5	j
i	a	3	6	1	7	9	8	
1	6	3	6					1
2	1	1	3	6				2
3	7				7			1
4	9					9		1
5	8					8	9	2

当前待插入元素　　比较次数

插入排序

```
1. void insert_sort(int A[ ],int n)
2. {
3.     int  a,i,j;
4.     for (i=1;i<n;i++) {
5.         a = A[ i ];
6.         j = i – 1;
7.         while (j>=0 && A[j]>a) {
8.             A[ j+1 ] = A[ j ];
9.             j- -;
10.         }
11.         A[ j+1 ] = a;
12.     }
13. }
```

一般情况的工作过程

		0	1	2	3	4	5	j
i	a	3	6	1	7	9	8	
1	6	3	6					1
2	1	1	3	6				2
3	7				7			1
4	9					9		1
5	8					8	9	2

当前待插入元素　　比较次数

□ 插入排序的最坏情况和最好情况分析

最好情况的工作过程

		0	1	2	3	4	5	j
i	a	1	3	6	7	8	9	
1	3	1	3					1
2	6		3	6				1
3	7			6	7			1
4	8				7	8		1
5	9					8	9	1

$\Theta(n)$

最坏情况的工作过程

		0	1	2	3	4	5	j
i	a	9	8	7	6	3	1	
1	8	8	9					1
2	7	7	8	9				2
3	6	6	7	8	9			3
4	3	3	6	7	8	9		4
5	1	1	3	6	7	8	9	5

$$\sum_{i=1}^{n-1} i = \frac{1}{2}n(n-1) = \Theta(n^2)$$

二、最好情况和最坏情况分析

例2.8：线性检索算法

在 n 个已排序的元素的数组 A[ ] 中，用线性检索算法检索元素 x

输入：给定n个元素的数组a[ ]，元素x

输出：若x = A[j],0≤j≤n-1,输出j,否则输出-1

```
1.  int linear_search(int A[],int n,int x);
2.  {   int j = 0;
4.      while (j<n && x!=A[j])
5.          j++;
6.      if (j<n)
7.          return j;
8.      else
9.          return -1;
10. }
```

最好情况：

数组的第一个元素是 x，其运行时间为 $\Theta(1)$

最坏情况：

数组中不存在元素，或元素是数组的最后一个元素，其运行时间为 $O(n)$、$\Omega(n)$、$\Theta(n)$

例2.9：二叉检索算法

在具有*n*个已排序的元素的数组中，用二叉检索算法检索元素 *x*

```
1.  int binary_search(int A[],int n,int x)
2.  {
3.     int mid,low = 0,high = n – 1,j = -1;
4.      while (low<=high && j<0) {
5.          mid = (low + high) / 2;
6.          if (x==A[mid]) j = mid;
7.          else if (x<A[mid]) high = mid –1;
8.          else low = mid + 1;
9.      }
10.     return j;
11. }
```

二叉检索算法分析：

最好情况：数组第 $n/2$ 个元素是 x，执行一次比较操作就可结束算法，算法的时间复杂性是 $\Theta(1)$

最坏情况：数组中不存在元素 x，或元素 x 是数组的第1个元素、或最后一个元素

第 1 次比较时，数组元素个数为 n 个

第 2 次比较时，数组元素个数为 $\lfloor n/2 \rfloor$ 个

第 j 次比较时，数组元素个数为 $\lfloor n/2^{j-1} \rfloor$ 个

设第 j 次比较是最后一次比较，有 $\lfloor n/2^{j-1} \rfloor = 1$

则：$1 \le n/2^{j-1} < 2$　　即：$2^{j-1} \le n \le 2^j$

检索x所需要的最大比较次数是j次，则 $j = \lfloor \log n \rfloor + 1$

算法的时间复杂性是 $\Theta(\log n)$

例2.10：线性检索和二叉检索混合使用

```
1.  int linear_search(int A[ ],int n,int x);
2.  int binary_search(int A[ ],int n,int x);
3.  int serach(int A[ ],int n,int x)
4.  {
5.      if ((n%2)==0)
6.          return linear_search(A,n,x);
7.      else
8.          return binary_search(A,n,x);
9.  }
```

最坏情况：数组中不存在元素 *x*，或元素 *x* 是数组的最后 一个元素，**时间复杂性：*O*(*n*)、Ω(log*n*)**

最好情况：元素 *x* 是数组的第一个元素**时间复杂性： *O*(log*n*)、Ω(1)**

三　平均情况分析

1. 插入排序算法的平均情况分析
2. 冒泡排序算法的下界平均情况分析

1. 插入排序算法的平均情况分析

- 数组中的元素为 $\{x_1, x_2, \ldots\ldots, x_n\}, x_i \neq x_j, 1 \leq i, j \leq n, i \neq j$
- n 个元素共有 $n!$ 种排列，假定每一种排列的概率相同；
- 前面 $i-1$ 个元素已按递增顺序排序，把元素 x_i 插入到合适位置的 i 种可能：
  - $j = 1$： x_i 是序列中最小的，需执行 $i-1$ 次比较；
  - $j = 2$： x_i 是这个序列中第2小的，仍需执行 $i-1$ 次比较；
  - $j = 3$： x_i 是这个序列中第3小的，需执行 $i-2$ 次比较；

    ……
  - $j = i$： x_i 是这个序列中最大的，需执行1次比较。

当 $2 \leq j \leq i$ 时，算法需执行的比较次数为 $i - j + 1$

插入排序算法的平均情况分析（续1）

i 种可能性的概率 $1/i$。元素 x_i 插入到合适位置的平均比较次数 T_i 为：

$$T_i = \frac{i-1}{i} + \sum_{j=2}^{i} \frac{i-j+1}{i}$$

$$= \frac{i-1}{i} + \sum_{j=1}^{i-1} \frac{j}{i} = 1 - \frac{1}{i} + \frac{1}{2i}(i-1)(i-1+1)$$

$$= 1 - \frac{1}{i} + \frac{1}{2}(i-1)$$

$$= \frac{1}{2} + \frac{i}{2} - \frac{1}{i}$$

插入排序算法的平均情况分析（续2）

把 $x_2, \ldots, x_n$ 插入到序列中的合适位置，所需的平均比较总次数 T:

$$T = \sum_{i=2}^{n} T_i = \sum_{i=2}^{n} \left(\frac{1}{2} + \frac{i}{2} - \frac{1}{i}\right) = \frac{1}{2}(n-1) + \frac{1}{2}\sum_{i=2}^{n} i - \sum_{i=1}^{n} \frac{1}{i} + 1$$

$$= \frac{1}{2}(n-1) + \frac{1}{4}(n(n+1) - 2) + 1 - \sum_{i=1}^{n} \frac{1}{i}$$

$$= \frac{1}{4}\left(n^2 + 3n\right) - \sum_{i=1}^{n} \frac{1}{i}$$

因为有 $\ln(n+1) \le \sum_{i=1}^{n} \frac{1}{i} \le \ln n + 1$

$$T \approx \frac{1}{4}\left(n^2 + 3n\right) - \ln n = \Theta(n^2)$$

2. 冒泡排序算法的下界平均情况分析

定义：设 $a_1, a_2, \ldots, a_n$ 是集合 {1,2,…,*n*} 的一个排列，如果 $i < j$ 且 $a_i > a_j$，则对偶 (a_i, a_j) 称为该排列的一个逆序。

- 例：排列 3，4，1，5 有逆序 (3,1) 及 (4,1)。若使元素按序排列，至少必须交换两次。

✓ 交换两个相邻元素，则逆序的总数将增 1 或减 1；

✓ 不断地交换两个相邻元素，使其逆序个数往减少的方向改变，当逆序个数减少为 0 时，就是一个有序的排列了；

✓ 排列中逆序的数目，是算法所执行的元素比较次数的下界；

✓ *n* 个元素共有 *n*! 种排列，所有排列的平均逆序的个数，就是算法所执行的平均比较次数的下界。

冒泡排序算法的下界平均情况分析（续 1）

例：集合 A = {1,2,3} 有如下种 3! 种排列：

排　列	逆序数目k
1 2 3	0
1 3 2	1
2 1 3	1
2 3 1	2
3 1 2	2
3 2 1	3

令 S(*k*) 是逆序个数为 *k* 时的排列数目，则有：

S(0) = 1　　S(1) = 2　　S(2) = 2　　S(3) = 1

具有 3 个元素集合的逆序的平均个数为：

$$\text{mean}(3)=\frac{1}{3!}(S(0)\times 0+S(1)\times 1+S(2)\times 2+S(3)\times 3)=1.5$$

冒泡排序算法的下界平均情况分析（续 2）

考虑 *n* 个元素的集合的所有排列：

最好情况：所有的元素都已顺序排列，该排列的逆序个数为 0；

最坏情况：所有的元素都是逆序排列，该排列的逆序个数为 *n*(*n*-1)/2。

逆序的平均个数为：

$$mean(n) = \frac{1}{n!}\sum_{k=0}^{\frac{n(n-1)}{2}} kS(k) = \sum_{k=1}^{n}\frac{k-1}{2} = \frac{1}{4}n(n-1)$$

约瑟夫环问题

- 问题描述：编号为1~*n*的*n*个士兵围坐在一起形成一个环，从编号为1的士兵开始依次报数，数到*m*的士兵会被杀死出列，之后的士兵再从1开始报数。直到最后剩下1个士兵，求这个士兵的编号。

约瑟夫环问题

- 方法 一：数组（思路简单，编码不简单）

  时间复杂度：$O(n*m)$，空间复杂度为$O(n)$。

- 方法二：环形链表

  时间复杂度：$O(n*m)$，空间复杂度为$O(n)$

- 方法三 ：递归

  $f(n,m)=(f(n-1,m)+m-1)\%n+1$

  时间复杂度：$O(n)$，空间复杂度为$O(n)$

- 算法分析时，采用时间耗费函数表示时间参数；
- 当问题规模充分大时，时间耗费函数的极限表示时间复杂度；
- 一般算法对应的时间耗费函数常用递归方程表示；
- 解递归方程是得到算法复杂性的具体表现。

- 递归：在定义一个过程或函数时，出现调用本过程或本函数的成分，称之为递归。
- 递归方程：使用小的输入值来描述一个函数的方程或不等式。
  - ✓ 用生成函数解递归方程
  - ✓ 用特征方程解递归方程
  - ✓ 用递推方法解递归方程

2.4 用生成函数求解递归方程

1. 生成函数的定义

定义：令$a_0, a_1, a_2 \ldots$是一个实数序列，构造如下的函数：

$$G(z) = a_0 + a_1 z + a_2 z^2 + \cdots = \sum_{k=0}^{\infty} a_k z^k$$

则函数 $G(z)$ 称为序列$a_0, a_1, a_2 \ldots$的生成函数

1. 生成函数的定义

定义：令$a_0, a_1, a_2 \ldots$是一个实数序列，构造如下的函数：

$$G(z) = a_0 + a_1 z + a_2 z^2 + \cdots = \sum_{k=0}^{\infty} a_k z^k$$

则函数 $G(z)$ 称为序列$a_0, a_1, a_2 \ldots$的生成函数

例：函数$(1+x)^n = C_n^0 + C_n^1 x + C_n^2 x^2 + \cdots + C_n^n x^n$

1. 生成函数的定义

定义：令$a_0, a_1, a_2 \ldots$是一个实数序列，构造如下的函数：

$$G(z) = a_0 + a_1 z + a_2 z^2 + \cdots = \sum_{k=0}^{\infty} a_k z^k$$

则函数 $G(z)$ 称为序列$a_0, a_1, a_2 \ldots$的生成函数

例：函数$(1+x)^n = C_n^0 + C_n^1 x + C_n^2 x^2 + \cdots + C_n^n x^n$

则函数$(1+x)^n$ 便是序列$C_n^0, C_n^1, C_n^2, \ldots, C_n^n$ 的生成函数

说明：

- 在用生成函数求解递归问题时，往往不需要求幂级数的和函数，故一般也不考虑幂级数的敛散性；
- 但当涉及到生成函数的性质时，往往要用到幂级数的收敛的性质，如在收敛空间内绝对收敛、逐项求和、可逐项求导和求积以及幂级数的柯西乘积等。

2. 生成函数的性质

1）去掉级数中的奇数项和偶数项

$$\frac{1}{2}\left(G(z)+G(-z)\right)=a_0+a_2z^2+a_4z^4+\cdots$$

$$\frac{1}{2}\left(G(z)-G(-z)\right)=a_1z+a_3z^3+a_5z^5+\cdots$$

2）加法

$G(z)=\sum_{k=0}^{\infty}a_k z^k$ 和$H(z)=\sum_{k=0}^{\infty}b_k z^k$ 分别是序列 $a_0, a_1, a_2 \ldots$ 及$b_0, b_1, b_2 \ldots$的生成函数，则

$$\alpha G(z)+\beta H(z)=\alpha\sum_{k=0}^{\infty}a_k z^k+\beta\sum_{k=0}^{\infty}b_k z^k=\sum_{k=0}^{\infty}(\alpha a_k+\beta b_k)\,z^k$$

是序列 $\alpha a_0+\beta b_0, \alpha a_1+\beta b_1, \alpha a_2+\beta b_2, \ldots$ 的生成函数

3）移位

$G(z)=\sum_{k=0}^{\infty}a_k z^k$ 是序列$a_0, a_1, a_2, \ldots$ 的生成函数，则

$$z^m G(z)=\sum_{k=0}^{\infty}a_k z^{m+k}$$

是序列 0,…,0, $a_0, a_1, a_2, \ldots$的生成函数

4）乘法

$G(z)=\sum_{k=0}^{\infty}a_k z^k$ 和 $H(z)=\sum_{k=0}^{\infty}b_k z^k$ 分别是序列 $a_0, a_1, a_2, \ldots$ 及 $b_0, b_1, b_2, \ldots$ 的生成函数，则

$$G(z)H(z)=\left(a_0+a_1z+a_2z^2+\cdots\right)\left(b_0+b_1z+b_2z^2+\cdots\right)$$

$$=a_0b_0+(a_0b_1+a_1b_0)z+(a_0b_2+a_1b_1+a_2b_0)z^2+\cdots$$

$$=\sum_{k=0}^{\infty}c_k z^k$$

是序列 $c_0, c_1, c_2, \ldots$的生成函数，其中 $c_n=\sum_{k=0}^{n}a_k b_{n-k}$

5）z变换

$G(z)=\sum_{k=0}^{\infty}a_k z^k$ 是序列$a_0, a_1, a_2, \ldots$ 的生成函数，则

$$G(cz)=a_0+a_1cz+a_2(cz)^2+a_3(cz)^3+\cdots$$

$$=a_0+ca_1z+c^2a_2z^2+c^3a_3z^3+\cdots$$

是序列 $a_0, ca_1, c^2a_2, \ldots$ 的生成函数。

- 特别地，当-1<cz<1时，有

$$\frac{1}{1-cz}=\sum_{i=0}^{\infty}(cz)^i=1+cz+\cdots+c^nz^n+\cdots$$

所以，$\frac{1}{1-cz}$是序列 $1, c, c^2, c^3, \ldots$ 的生成函数。

- 若取c=-1，则当 -1<z<1时，有

$$\frac{1}{1+z}=\sum_{i=0}^{\infty}(-z)^i=1-z+z^2-z^3+\cdots+(-1)^nz^n+\cdots$$

所以，$\frac{1}{1+z}$为序列1,-1,1,-1,…的生成函数。

- 若取c=1，有

$$\frac{1}{1-z}=\sum_{i=0}^{\infty}z^i=1+z+z^2+z^3+\cdots+z^n+\cdots$$

所以，$\frac{1}{1-z}$为序列1,1,1,1,…的生成函数.

若G(z) 是序列$a_0, a_1, a_2, \ldots$ 的生成函数，有

$$\frac{1}{1-z}G(z) = a_0 + (a_0 + a_1)z + (a_0 + a_1 + a_2)z^2 + \cdots$$

则$\frac{1}{1-z}G(z)$是序列 $a_0, (a_0 + a_1), (a_0 + a_1 + a_2), \ldots$ 的生成函数。

6）微分和积分

设 $G(z)=\sum_{k=0}^{\infty}a_k z^k$ 是序列$a_0, a_1, a_2, \ldots$ 的生成函数，则逐项求导，有

$$G'(z)=\sum_{k=0}^{\infty}(a_k z^k)'=\sum_{k=0}^{\infty}(k+1)a_{k+1}z^k$$

$$=a_1+2a_2 z+3a_3 z^2+\cdots+(n+1)a_{n+1}z^n+\cdots$$

为序列$a_1, 2a_2, 3a_3, \ldots, (n+1)a_{n+1}, \ldots$的生成函数。

设 $G(z)=\sum_{k=0}^{\infty}a_k z^k$ 是序列$a_0, a_1, a_2, \ldots$的生成函数，逐项积分，有

$$\int_0^z G(t)dt=\sum_{k=0}^{\infty}\int_0^z a_k t^k dt=\sum_{k=0}^{\infty}\frac{a_k}{k+1}z^{k+1}$$

$$=a_0 z+\frac{a_1}{2}z^2+\cdots+\frac{a_{n-1}}{n}z^n+\cdots$$

为序列 $a_0, \frac{a_1}{2}, \ldots, \frac{a_{n-1}}{n}, \ldots$的生成函数。

二　用生成函数求解递归方程

1. 汉诺塔（Hanoi）问题

2. 菲波那契序列问题

2.4.2 用生成函数求解递归方程

- 汉诺塔（Hanoi）问题：

描述：设A，B，C是3个塔座。开始时，在塔座A上有一叠共x个圆盘，这些圆盘自下而上、由大到小地叠在一起。各圆盘从小到大编号为1,2,3,…,n，现在要求将塔座A上的这一叠原圆盘移到塔座C上，并仍按同样顺序叠置。在移动圆盘时应遵守以下移动规则：

规则1：每次只能移动1个圆盘；

规则2：任何时刻都不允许将较大的圆盘压在较小的圆盘之上；

规则3：在满足移动规则1和2的前提下，可将圆盘移至 A,B,C中任一塔座上。

试求：移动的次数函数关于n的表达式。

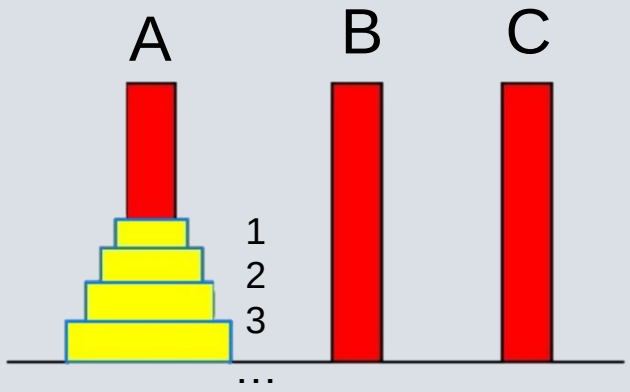
汉诺塔（Hanoi）问题：

输入：串满圆盘的塔座a，目的塔座b，辅助塔座c，圆盘个数n

输出：圆盘移动列表

```
void Hanoi (char a, char b, char c, int n)
{
      if ( n ==1 ) printf (“%c->%c”, a, b);
      else{
          Hanoi( a, c, b, n-1);
          printf( “%c->%c”,  a, b);
          Hanoi(c, b, a, n-1);
       }
}
```

通过分析，移动n个圆盘的移动次数 $h(n)$ 有：



$n = 1 \quad h(1) = 1$

$n = 2 \quad h(2) = 2h(1) + 1$

$n = 3 \quad h(3) = 2h(2) + 1$

得到如下的递归关系式：

$$\begin{cases} h(n) = 2h(n\text{-}1) + 1 \\ h(1) = 1 \end{cases}$$

用 $h(n)$ 作系数，构造生成函数：

$$G(x) = h(1)x + h(2)x^2 + h(3)x^3 + \cdots = \sum_{k=1}^{\infty} h(k)x^k$$

用生成函数求解汉诺塔问题的递归方程

令

- $G(x)-2xG(x)=h(1)x+h(2)x^2+h(3)x^3+\cdots-$

  $2h(1)x^2-2h(2)x^3-\cdots$

  $=h(1)x+\big(h(2)-2h(1)\big)x^2+\big(h(3)-2h(2)\big)x^3+\cdots$

- $(1-2x)G(x)=x+x^2+x^3+\cdots=x(1+x+x^2+\cdots)$

  $$=\frac{x}{1-x}$$

- $G(x)=\frac{x}{(1-x)(1-2x)}$

用生成函数求解汉诺塔问题的递归方程

- $G(x)=\frac{x}{(1-x)(1-2x)}$
- $G(x)=\frac{A}{(1-x)}+\frac{B}{(1-2x)}=\frac{A-2Ax+B-Bx}{(1-x)(1-2x)}$
- $A+B=0$　　$-2A-B=1$　解得：A = -1　B = 1
- $G(x)=\frac{1}{1-2x}-\frac{1}{1-x}=\left(1+2x+2^2x^2+2^3x^3+\cdots\right)-\left(1+x+x^2+x^3\right)=(2-1)x+\left(2^2-1\right)x^2+\left(2^3-1\right)x^3+\cdots=\sum_{k=1}^{\infty}\left(2^k-1\right)x^k$
- 故 $h(n)=2^n-1$

- 菲波那契序列问题：

假设小兔子每隔一个月长成大兔子，大兔子每隔一个月生一只小兔子。第一个月有一只小兔子，求*n*个月后有多少只兔子。

	小兔	大兔	总数
month 1st	1	0	1
month 2nd	0	1	1
month 3rd	1	1	2
month 4th	1	2	3
month 5th	2	3	5
……			
month (*n*-2)th	t	T	
month (*n*-1)th	T	t+T	
month *n* th	t+T	T+t+T	

- 菲波那契序列问题：

$f(n)$： n 个月后兔子的总数目

$t(n)$： n 个月后小兔子的数目

$T(n)$： n 个月后大兔子的数目

有关系： $f(n) = T(n) + t(n)$

$T(n) = T(n\text{-}1) + t(n\text{-}1)$

$t(n) = T(n\text{-}1)$

得递归方程：$\begin{cases} f(n) = f(n\text{-}1) + f(n\text{-}2) \\ f(1) = f(2) = 1 \end{cases}$

无穷数列1，1，2，3，5，8，13，21，34，55，…被称为Fibonacci数列。

用 $f(n)$ 作系数，构造生成函数：

$$F(x)=f(1)x+f(2)x^2+f(3)x^3+\cdots=\sum_{k=1}^{\infty}f(k)x^k$$

令 $F(x)-xF(x)-x^2F(x)$

$$=f(1)x+f(2)x^2+f(3)x^3+\cdots-x(f(1)x+f(2)x^2+\cdots)-x^2(f(1)x+\cdots)$$

$$=f(1)x+(f(2)-f(3)x^2+(f(3)x-f(2)-f(1))x^3+\cdots$$

$$=x$$

$$F(x)=\frac{x}{1-x-x^2}=\frac{-x}{x^2+x+\frac{1}{4}-\frac{5}{4}}=\frac{-x}{\left(x+\frac{1}{2}\right)^2-\left(\frac{1}{2}\sqrt{5}\right)^2}$$

$$=\frac{-x}{\left(x+\frac{1}{2}(1-\sqrt{5})\right)\left(x+\frac{1}{2}(1+\sqrt{5})\right)}$$

令 $F(x)=\dfrac{A}{x+\frac{1}{2}(1-\sqrt{5})}+\dfrac{B}{x+\frac{1}{2}(1+\sqrt{5})}=\dfrac{Ax+\frac{1}{2}(1+\sqrt{5})A+Bx+\frac{1}{2}(1-\sqrt{5})B}{\left(x+\frac{1}{2}(1-\sqrt{5}\right)\left(x+\frac{1}{2}(1+\sqrt{5}\right)}$

有 $A+B=-1,\ (1+\sqrt{5})A+(1-\sqrt{5})B=0$

解得 $A=\dfrac{1}{2\sqrt{5}}(1-\sqrt{5}),\ B=\dfrac{-1}{2\sqrt{5}}(1+\sqrt{5})$

故有 $F(x)=\dfrac{1}{\sqrt{5}}\left(\dfrac{1-\sqrt{5}}{2x+1-\sqrt{5}}-\dfrac{1+\sqrt{5}}{2x+1+\sqrt{5}}\right)=\dfrac{1}{\sqrt{5}}\left(\dfrac{1}{1-\dfrac{2x}{\sqrt{5}-1}}-\dfrac{1}{1-\dfrac{-2x}{\sqrt{5}+1}}\right)$

令 $\alpha=\dfrac{2}{\sqrt{5}-1}=\dfrac{1}{2}(1+\sqrt{5})\qquad\beta=\dfrac{-2}{\sqrt{5}+1}=\dfrac{1}{2}(1-\sqrt{5})$

有 $F(x)=\dfrac{1}{\sqrt{5}}((\alpha-\beta)x+(\alpha^2-\beta^2)x^2+\cdots)$

第n项系数为 $f(n)=\dfrac{1}{\sqrt{5}}(\alpha^n-\beta^n)$

2.5 用特征方程求解递归方程

2.5.1 k 阶常系数线性齐次递归方程

1. K阶常系数线性齐次递归方程及其特征方程

1）递归方程的形式：

$$\begin{cases} f(n) = a_1 f(n-1) + a_2 f(n-2) + \cdots + a_k f(n-k) \\ f(i) = b_i \qquad\qquad\qquad\qquad 0 \le i < k \end{cases}$$

2）递归方程的特征方程

x^n 取代 $f(n)$：$x^n = a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_k x^{n-k}$

两边分别除以x^{n-k}：$x^k = a_1 x^{k-1} + a_2 x^{k-2} + \cdots + a_k$

$$x^k - a_1 x^{k-1} - a_2 x^{k-2} - \cdots - a_k = 0$$

2. k 阶常系数线性齐次递归方程的求解过程

- 求解过程：
  - 求特征方程的根，得到递归方程的通解；
  - 把递归方程的初始条件代入通解，建立联立方程，确定系数

$c_1, c_2, \ldots, c_k$，从而可求出递归方程的解 。

1. 特征方程有k个互不相同的根$q_1, q_2, \ldots, q_k$，通解为：

$$f(n) = c_1 q_1^n + c_2 q_2^n \ldots + c_k q_k^n$$

2. 特征方程的k个根中有r个重根$q_i, q_{i+1}, \ldots, q_{i+r-1}$，通解为：

$$f(n) = c_1 q_1^n + \cdots + c_{i-1} q_{i-1}^n + \left(c_i + c_{i+1} n + \cdots + c_{i+r-1} n^{r-1}\right) q_i^n + \cdots + c_k q_k^n$$

3. 例子

(1) $\begin{cases} f(n) = 6f(n-1) - 11f(n-2) + 6f(n-3) \\ f(0) = 0 \quad f(1) = 2 \quad f(2) = 10 \end{cases}$

特征方程：$x^3 - 6x^2 + 11x - 6 = 0$

$x^3 - 3x^2 - 3x^2 + 9x + 2x - 6 = 0$

$(x-1)(x-2)(x-3) = 0$

特征根：$q_1 = 1 \quad q_2 = 2 \quad q_3 = 3$

通解:$f(n) = c_1 q_1^n + c_2 q_2^n + c_3 q_3^n = c_1 + c_2 2^n + c_3 3^n$

由初始条件得：$f(0) = c_1 + c_2 + c_3 = 0$

$f(1) = c_1 + 2c_2 + 3c_3 = 2$

$f(2) = c_1 + 4c_2 + 9c_3 = 10$

有：$c_1 = 0 \quad c_2 = -2 \quad c_3 = 2$

故 $f(n) = 2(3^n - 2^n)$

(2) $\begin{cases} f(n)=5f(n-1)-7f(n-2)+3f(n-3) \\ f(0)=1 \quad f(1)=2 \quad f(2)=7 \end{cases}$

特征方程：$x^3-5x^2+7x-3=0$

$x^3-3x^2-2x^2+6x+x-3=0$

$(x-1)(x-1)(x-3)=0$

特征根：$q_1=1 \quad q_2=1 \quad q_3=3$

通解:$f(n)=(c_1+c_2n)q_1^n+c_3q_3^n=c_1+c_2n+c_3 3^n$

由初始条件得：$f(0)=c_1+c_3=1$

$f(1)=c_1+c_2+3c_3=2$

$f(2)=c_1+2c_2+9c_3=7$

有：$c_1=0 \quad c_2=-1 \quad c_3=1$

故 $f(n)=3^n-n$

2.5.2 k阶常系数线性非齐次递归方程

1. 非齐次递归方程及其通解的形式
2. 非齐次递归方程特解的求取
3. 非齐次递归方程通解的求取
4. 例子

1. 非齐次递归方程及其通解的形式

1）递归方程的形式：

$$\begin{cases} f(n) = a_1 f(n-1) + a_2 f(n-2) + \cdots + a_k f(n-k) + g(n) \\ f(i) = b_i \qquad\qquad 0 \le i < k \end{cases}$$

2）通解形式：$f(n) = \overline{f(n)} + f^*(n)$

$\overline{f(n)}$：对应齐次递归方程的通解

$f^*(n)$：原非齐次递归方程的特解

2. 非齐次递归方程特解的求取

1）$g(n)$ 是 n 的 m 次多项式：

$$g(n)=b_0n^m+b_1n^{m-1}+\cdots+b_{m-1}n+b_m$$

$b_i, i=0,1,\cdots,m$ 是常数

特解也是n的m次多项式：

$$f^*(n)=A_0n^m+A_1n^{m-1}+\cdots+A_{m-1}n+A_m$$

$A_i, i=0,1,\cdots,m$ 待定系数

2）*g(n)* 是如下形式的指数函数

$$g(n)=(b_0n^m+b_1n^{m-1}+\cdots+b_{m-1}n+b_m)a^n$$

$b_i, i=0,1,\cdots,m$是常数。

① a 不是特征方程的重根，特解为

$$f^*(n)=(A_0n^m+A_1n^{m-1}+\cdots+A_{m-1}n+A_m)a^n$$

② a 是特征方程的 r 重根，特解为

$$f^*(n)=(A_0n^m+A_1n^{m-1}+\cdots+A_{m-1}n+A_m)n^ra^n$$

$A_i, i=0,1,\cdots,m$为待定系数。

3. 非齐次递归方程通解的求取

1）求对应齐次递归方程的通解 $\overline{f(n)}$

2）求非齐次递归方程的特解 $f^*(n)$

- 把带有待定系数的特解代入原非齐次递归方程；
- 比较方程两边系数，列出待定系数的联立方程；
- 解联立方程，求出待定系数。

3）求非齐次递归方程的通解 $f(n)=\overline{f(n)}+f^*(n)$

把初始条件代入通解，列出通解中待定系数的联立方程解联立方程，求出待定系数。

4. 例子

$$\begin{cases} f(n) = 7f(n-1) - 10f(n-2) + 4n^2 \\ f(0) = 1 \quad f(1) = 2 \end{cases}$$

对应的齐次递归方程的特征方程： $x^2 - 7x + 10 = 0$

$$(x-2)(x-5) = 0$$

特征根： $q_1 = 2 \quad q_2 = 5$

对应的齐次递归方程的通解： $\overline{f(n)} = c_1 2^n + c_2 5^n$

令非齐次递归方程的特解： $f^*(n) = A_0 n^2 + A_1 n + A_2$

代入原递归方程，得：

$$A_0 n^2 + A_1 n + A_2 - 7(A_0(n-1)^2 + A_1(n-1) + A_2) +$$
$$10(A_0(n-2)^2 + A_1(n-2) + A_2) = 4n^2$$

例子（续）

$$4A_0n^2+(-26A_0+4A_1)n+33A_0-13A_1+4A_2=4n^2$$

得到联立方程：

$$4A_0=4$$

$$-26A_0+4A_1=0$$

$$33A_0-13A_1+4A_2=0$$

解得：$A_0=1$　$A_1=6\frac{1}{2}$　$A_2=12\frac{7}{8}$

非齐次递归方程的通解：$f(n)=c_1 2^n+c_2 5^n+n^2+6\frac{1}{2}n+12\frac{7}{8}$

代入初始条件：$f(0)=c_1+c_2+12\frac{7}{8}=1$　　$c_1=-13\frac{2}{3}$

$f(1)=2c_1+5c_2+20\frac{3}{8}=2$　　$c_2=1\frac{19}{24}$

通解：$f(n)=-13\frac{2}{3}2^n+1\frac{19}{24}5^n+n^2+6\frac{1}{2}n+12\frac{7}{8}$

k 阶常系数线性齐次递归方程

$$\begin{cases} f(n) = a_1 f(n-1) + a_2 f(n-2) + \cdots + a_k f(n-k) \\ f(i) = b_i \qquad\qquad\qquad\qquad 0 \le i < k \end{cases}$$

1. 求解 递归方程的特征方程的k个根

$$x^k - a_1 x^{k-1} - a_2 x^{k-2} - \cdots - a_k = 0$$

2. 如果没有重根，则递归方程的通解为：

$$f(n) = c_1 q_1^n + c_2 q_2^n \ldots + c_k q_k^n$$

如果有重根，则递归方程的通解为：

$$f(n) = c_1 q_1^n + \cdots + c_{i-1} q_{i-1}^n + \left(c_i + c_{i+1} n + \cdots + c_{i+r-1} n^{r-1}\right) q_i^n + \cdots + c_k q_k^n$$

3. 代入初值，确定待定系数。

$$\begin{cases} f(n) + f(n-1) - 3f(n-2) - 5f(n-3) - 2f(n-4) = 0 \\ f(0) = 1,\ f(1) = 0,\ f(2) = 1,\ f(3) = 2 \end{cases}$$

k 阶常系数线性齐次递归方程

$$\begin{cases} f(n) = a_1 f(n-1) + a_2 f(n-2) + \cdots + a_k f(n-k) + g(n) \\ \quad f(i) = b_i \qquad\qquad\qquad\qquad 0 \le i < k \end{cases}$$

1. 求对应齐次递归方程的通解$\overline{f(n)}$
2. 求非齐次递归方程的特解$f^*(n)$
   - 把带有待定系数的特解代入原非齐次递归方程；
   - 比较方程两边系数，列出待定系数的联立方程；
   - 解联立方程，求出待定系数。
3. 求非齐次递归方程的通解$f(n) = \overline{f(n)} + f^*(n)$

把初始条件代入通解，列出通解中待定系数的联立方程解联立方程，求出待定系数。

2.6 用递推方法求解递归方程

方法：

循环地展开递归方程

把递归方程转化为和式

然后使用求和技术求解

2.6.1 用递推法求解非齐次递归方程

非齐次递归方程：
$$\begin{cases} f(n) = bf(n-1) + g(n) \\ f(0) = c \end{cases}$$

其中，b、c 为常数

$$\begin{aligned} f(n) &= b(b\,f(n-2) + g(n-1)) + g(n) \\ &= b^2 f(n-2) + bg(n-1) + g(n) \\ &= b^2(b\,f(n-3) + g(n-2)) + bg(n-1) + g(n) \\ &= b^3 f(n-3) + b^2 g(n-2) + bg(n-1) + g(n) \\ &= \cdots\cdots \\ &= b^n f(0) + b^{n-1} g(1) + \cdots + b^2 g(n-2) + bg(n-1) + g(n) \\ &= cb^n + \sum_{i=1}^{n} b^{n-i} g(i) \end{aligned}$$

例：汉诺塔问题的递归方程：

$$\begin{cases} h(n) = 2\,h(n-1) + 1 \\ h(1) = 1 \end{cases}$$

- 若直接利用上面结果：$b = 2, c = 1, g(n) = 1$

有：$h(n) = cb^{n-1} + \sum_{i=1}^{n-1} b^{n-1-i} g(i)$

$$= 2^{n-1} + 2^{n-2} + \cdots + 2^2 + 2 + 1$$

$$= 2^n - 1$$

2.6.2 用递推法求解变系数递归方程

变系数齐次递归方程：

$$\begin{cases} f(n) = g(n)f(n-1) \\ f(0) = c \end{cases}$$

其中，c 是常数，$g(n)$ 和 $h(n)$ 是 n 的函数

利用递推可得：

$$f(n) = cg(n)g(n-1)g(n-2)\dots g(1)$$

例：解如下递归函数：

$$\begin{cases} f(n) = n\,f(n-1) \\ f(0) = 1 \end{cases}$$

解：由$f(n) = cg(n)g(n-1)g(n-2)\ldots g(1)$，

有：

$$f(n) = n(n-1)(n-2)\cdots 1 = n!$$

变系数非齐次递归方程：

$$\begin{cases} f(n) = g(n)f(n-1) + h(n) \\ f(0) = c \end{cases}$$

其中，c 是常数，$g(n)$ 和 $h(n)$ 是 n 的函数

利用递推可得：

$$f(n) = g(n)g(n-1)\cdots g(1)\left[f(0) + \sum_{i=1}^{n} \frac{h(i)}{g(i)g(i-1)\cdots g(1)} \right]$$

例子：变系数非齐次递归方程

$$\begin{cases} f(n) = n\,f(n-1) + n! \\ f(0) = 0 \end{cases}$$

$$f(n) = n((n-1)\,f(n-2) + (n-1)!) + n!$$

$$= n(n-1)\,f(n-2) + 2n!$$

$$= \cdots\cdots\cdots$$

$$= n!\,f(0) + nn!$$

$$= nn!$$

若直接利用上面结果：

$$f(n) = n(n-1)\cdots 1 \sum_{i=1}^{n} \frac{i!}{i(i-1)\cdots 1} = nn! \qquad f(0) = 0$$

2.6.3 换名

递归方程：$\begin{cases} f(n) = 2f(n/2) + n/2 - 1 & n = 2^k \\ f(1) = 1 \end{cases}$

把表示成的关系，原递归方程改写为：

$$\begin{cases} f(2^k) = 2f(2^{k-1}) + 2^{k-1} - 1 \\ f(2^0) = 1 \end{cases}$$

令：$g(k) = f(2^k) = f(n)$

原递归方程可写为：

$$\begin{cases} g(k) = 2g(k-1) + 2^{k-1} - 1 \\ g(0) = 1 \end{cases}$$

换名（续 1）

解递归方程：

$$\begin{cases} g(k) = 2g(k-1) + 2^{k-1} - 1 \\ g(0) = 1 \end{cases}$$

$$g(k) = 2(2g(k-2) + 2^{k-2} - 1) + 2^{k-1} - 1$$

$$= 2^2 g(k-2) + 2 \cdot 2^{k-1} - 2 - 1$$

$$= 2^3 g(k-3) + 3 \cdot 2^{k-1} - 2^2 - 2 - 1$$

$$= \cdots\cdots\cdots$$

$$= 2^k g(0) + k \cdot 2^{k-1} - \sum_{i=0}^{k-1} 2^i$$

$$= 2^k \left(1 + \frac{k}{2} - \sum_{i=0}^{k-1} 2^{i-k}\right)$$

换名（续 2）

$$= 2^k(1+\frac{k}{2}-\sum_{i=0}^{k-1}2^{i-k}) = 2^k(1+\frac{k}{2}-\sum_{i=1}^{k}\frac{1}{2^i})$$

$$= 2^k(1+\frac{k}{2}-(1-\frac{1}{2^k}))$$

$$= 2^k(\frac{k}{2}+\frac{1}{2^k})$$

$$= \frac{1}{2}2^k k+1$$

$$= \frac{1}{2}n\log n+1$$

2.7 算法的空间复杂性

- 算法的空间复杂性：该算法所耗费的存储空间，是问题规模n的函数。
- 一个算法在计算机存储器上占用的存储空间，包括：
  - ✓ 存储算法本身所占用的存储空间
  - ✓ 算法的输入输出数据所占用的存储空间
  - ✓ 算法在运行过程中临时占用的存储空间
- 空间复杂度：对一个算法在运行过程中临时占用存储空间大小的量度，用$S(n)$表示

两种处理方法：

1. 算法所需要存储空间，仅是算法所需要的工作空间（线性检索、二叉检索 、合并排序）；

2. 算法所需要存储空间为算法在运行时所占用的内存空间的总和。

算法所需要的存储空间可表示为：

$$S_A=c+S(n)$$

c：程序代码、常数等固定部分

$S(n)$：与输入规模有关的部分

(输出入数据所占用的空间、工作空间、递归栈)

2.8 最优算法（自学）

1. 已知问题的任何算法的运行时间是 $\Omega(f(n))$，则对以时间 $O(f(n))$ 求解问题的任何算法，都认为是最优算法。
2. 运行时间同阶的算法，常数因子小的算法，优于常数因子大的算法。
3. 时间复杂性渐近阶的确定，与n及常数c的选取有关，当规模很小时，复杂性阶低的算法，不一定比复杂性阶高的算法更有效。

本章小结

1. 熟练掌握常用的函数和公式；

2. 理解算法的时间复杂性分析（循环次数的统计、基本操作频率的统计、计算步的统计）；

3. 能分析算法的最好情况、平均情况和最坏情况；

4. 用不同方法求解递归方程（生成函数、特征方程、递推方法）