康托展开 与 字符串hash

By Helang

1.康托展开

排列与组合公式

1.从n个不同学生中，选m(m≤n）个学生出来排列成一队，有多少种不同的排列方法？叫做从n个不同元素中取出m个元素的一个排列；

$$A_n^m = n! / (n-m)!$$

2.从n个不同学生中，选m(m≤n）个学生出来打扫清洁，有多少种不同的选法？叫做从n个不同元素中取出m个元素的一个组合；

$$C_n^m = A_n^m / m! = n! / ((n-m)!*m!)$$

$$C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$$

康托展开

{1,2,3} 三个数的排列按从小到大一共6个：123，132，213，231，312，321

可以用十进制的数字编号把他们对应起来：　1，　2，　3，　4，　5，　6

上述排列与编号之间的关系可以用"**康托展开**"来计算

如我想知道321是{1, 2, 3}排列中第几个大的数，可以这样考虑：
假设比321小的数字为Num，我们讨论Num个数：

1. 讨论Num**左起第一位，如果第一位的数小于3**，那构成排列一定小于321如123, 213。 小于3的数有1和2两个， 所以满足条件1的Num有2*2!个=4个。也就是讨论1XX和2XX的个数

2. 如果Num**左起第一位为3**，再讨论左起第二位，如果**左起第二位小于2**，那构成的排列一定小于321，如312。小于2的数只有一个就是1, 所以满足条件2的Num有1*1!=1个，也就是讨论31X的个数

3. 如果Num**左起第一位为3，第二位是2**，再讨论**左起第三位，比1小**的数没有，满足条件3的Num有0个。

因此可以得出，在{1, 2, 3}的排列中，比321小的数有4+1+0=5个，那么321是编号为6的数。

康托展开

练习：35142在{1, 2, 3, 4, 5}构成的全排列中，排号第几？

【讨论左一】**左一为3**：比3小的有2个，因此得到2*4！　也就是1XXXX和2XXXX的个数

【讨论左二】**左一为3，左二为5**：比5小的有4个，除去位于左一的3，剩下3个。因此得到3*3！也就是讨论34XXX　32XXX　31XXX的个数

【讨论左三】**左一为3，左二为5，左三为1**：比1小的有0个，因此得到0*2！也就是350XX 的个数

【讨论左四】**左一为3，左二为5，左三为1，左四为4**：比4小的有3个，除去位于左一的3和位于左三的1，剩下1个，因此得到1*1！也就是3512X 的个数

【讨论左五】**左一为3，左二为5，左三为1，左四为4，左五为2**：比2小的有1个，除去位于左一到左四的数，剩下0个，因此得到0*0！也就是35142 的个数

由上述计算可以得出结论：

比35142小的数有2*4！ + 3*3！ + 0*2！ + 1*1！ +0*0！ =67个。

35142在{1, 2, 3, 4, 5}的全排列中排名第68

康托展开的代码

3*8! +4*7! +2*6! +4*5! +4*4! +3*3! +0*2! +1*1! +0*0! =143155

```
int s[10]={0,3,5,2,7,8,6,0,4,1}; //将待展开的"352780641"存入数组
int p[10] = { 1, 1, 2, 6, 24, 120,720,5040,40320,362880};
//p[x]记录x!,p数组依次是0!,1!,2!,3!,4!,5!,6!,7!,8!,9!
int KangTuo()                    //将s[ ]数组表示的352786041转换成编号
{
    int i,j,temp,num;
    num=s[1]*p[9-1];     //处理左起第一个数s1*(8!),这里首位可以是0
    for(i=2;i<=9;i++)    //依次处理左起第2到第9个数
    {
        temp=0;
        for(j=1;j<i;j++)
            if(s[j]<s[i])temp++;        //记录不能使用的数字的个数
        num=num+(s[i]-temp)*p[9-i];
    }
    return num+1;
}
```

康托展开的逆运算

{1,2,3,4,5}的全排列中，排名第96的是多少？

第一步，用96-1得到95，用95去除4! 得到3余23。有3个数比它小的数是4，所以第一位是4

第二步，用23去除3! 得到3余5。有3个数比它小的数是4但4已经在之前出现过了所以第二位是5（4在之前出现过，所以实际比5小的数是3个）

第三步，用5去除2!得到2余1。有2个数比它小的数是3，第三位是3

第四步，用1去除1!得到1余0。有1个数比它小的数是2，第四位是2

最后一个数只能是1。所以这个数是**45321**

练习：{1,2,3,4,5}的全排列已经从小到大排序，要找出第16个数

练习：{1,2,3,4,5}的全排列已经从小到大排序，要找出第16个数：

1. 首先用16-1得到15
2. 用15去除4! 得到0余15
3. 用15去除3! 得到2余3
4. 用3去除2! 得到1余1
5. 用1去除1! 得到1余0

有0个数比它小的数是1，所以第一位是1

有2个数比它小的数是3，但1已经在之前出现过了所以第二位是4

有1个数比它小的数是2，但1已经在之前出现过了所以第三位是3

有1个数比它小的数是2，但1,3,4都出现过了所以第四位是5

最后一个数只能是2

所以这个数是**14352**

康托展开的逆运算代码

```
int p[10] = { 1, 1, 2, 6, 24, 120};     //p[i]表示i的阶乘
void NiKangTuo(int n, int k, int s[ ]) //n表示全排列的位数，k为排名，s存排名为k的结果
{
    int i, j, t, flag[9]={0};  //flag[i]用于标记数字i是否已被使用过了
    k--;
    for(i=1;i<=n;i++)
    {
        t=k/p[n-i];
        for(j=1;j<=n;j++)
            if(!flag[j])
            {
                if(t==0)break;
                t--;
            }
        s[i]=j;
        flag[j]=1;
        k%=p[n-i];
    }
}
```

益智玩具 魔板

八数码问题NKOJ2271(POJ1077)

八数码问题是人工智能中的经典问题

有一个3*3的棋盘，其中有0-8共9个数字，0表示空格子，每一步0可以与它相邻的数字交换位置。求由初始状态

8 2 3
4 1 6
5 7 0

到达目标状态

1 2 3
4 5 6
7 8 0

的步数最少的解？

初始状态

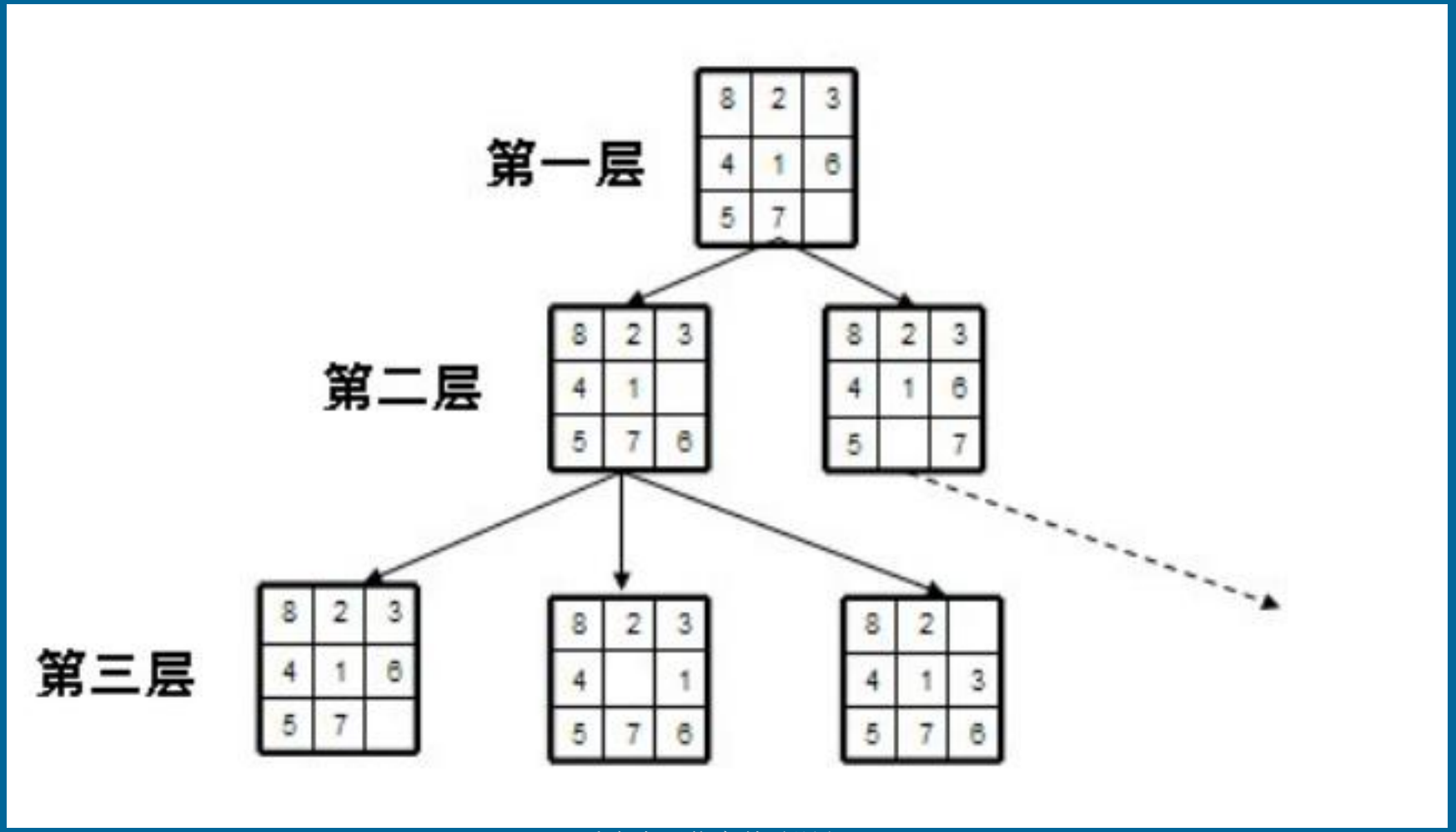
8	2	3
4	1	6
5	7	0

目标状态

1	2	3
4	5	6
7	8	0

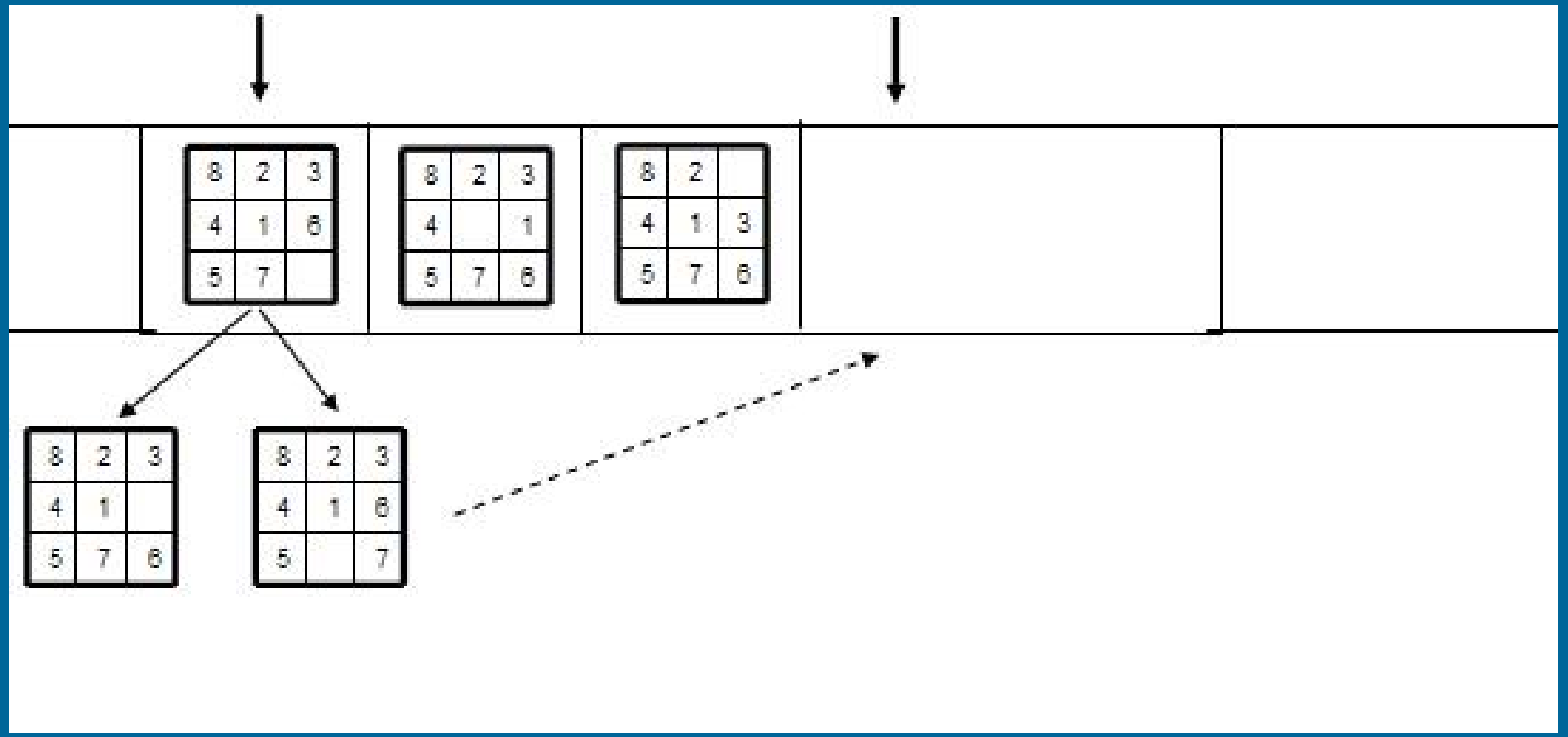
求最小步数，我们可以考虑通过广搜解决。
本题，初始状态和目标状态明确，也可以考虑双向广搜。

- 广度优先搜索（bfs）
  - 优先扩展浅层节点，逐渐深入



广度优先搜索

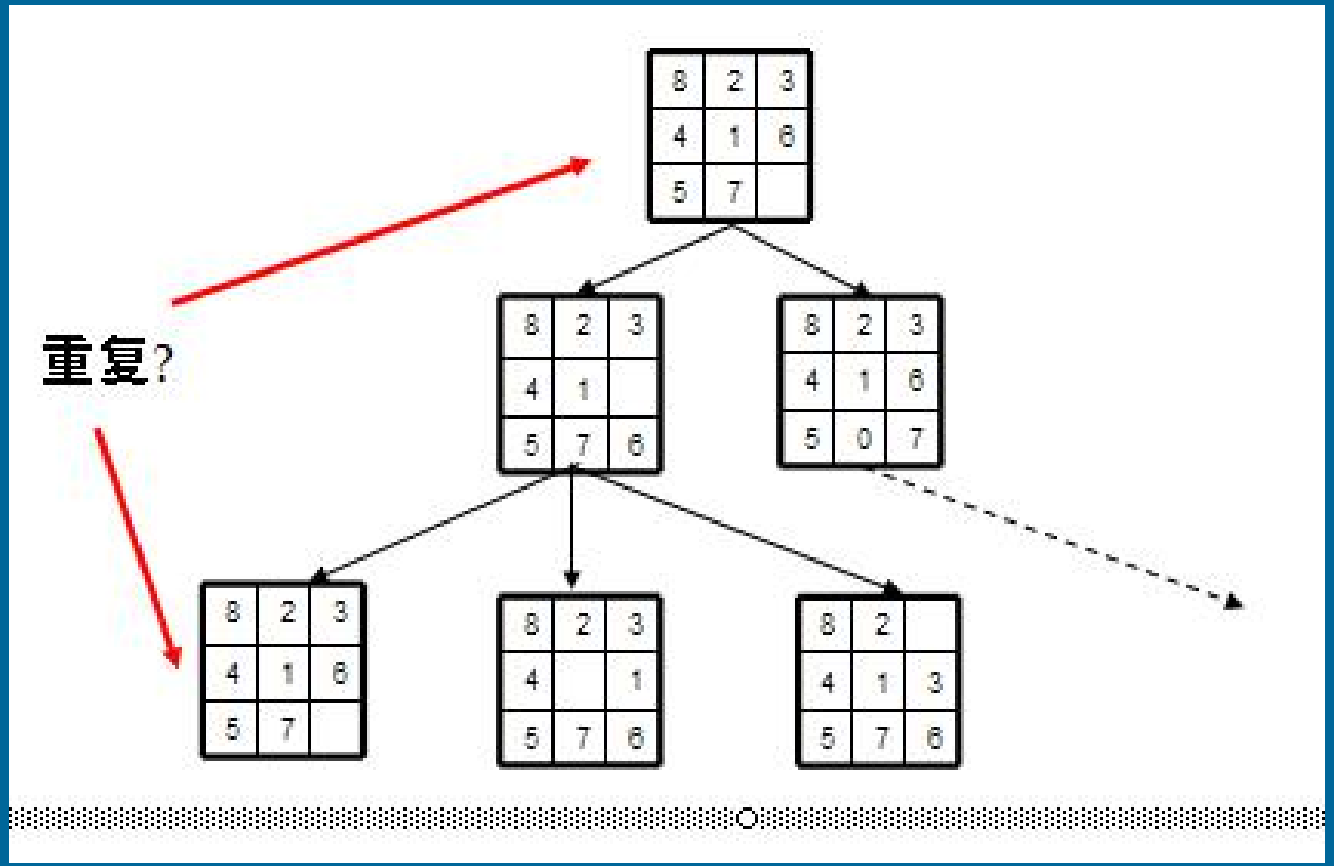
- 用队列保存待扩展的节点，从队首队取出节点，扩展出的新节点放入队尾，直到找到目标节点（问题的解）



判重

新扩展出的节点如果和以前扩展出的节点相同，则这个新节点就不必再考虑

如何判重？



宽搜？怎么记录状态？

3	5	2
7	8	6
0	4	1

采用类似位运算状态压缩的方法：
每个格子有0到8九种情况，总共9个格子，那么可以用一个9位的九进制数来表示当前棋盘的状态。
比如：左边的棋盘用352786041来表示。
Mark[352786041]=true，表示这种布局的棋盘出现过了

但这样需要申明一个大小为9^9的数组:
bool Mark[387420489];
显然会爆空间。

采用其他的记录状态的方法：**康托展开**

康托展开的作用

搜索判重

3	5	2
7	8	6
	4	1

把352786041 看成0到8的全排列中的一种情况，0到8的全排列总共有9！种情况，9！=362880，用bool数组f[362881]来标记即可判重。

(352786041)

康托展开

3*8！ +4*7！ +2*6！ +4*5！ +4*4！ +3*3！ +0*2！ +1*1！ +0*0！ =143155

编号143156 f[143156]=true

练习：

2272 数字排列

1032 NOIP 2004 火星人

1829 usaco 3.2.5 Magic Squares 魔板

补充一点全排列知识
permutation

```cpp
#include<cstdio>
#include<algorithm>
using namespace std;
int n,k,a[1000];
int main()
{
	scanf("%d%d",&n,&k);
	for(int i=1;i<=n;i++) scanf("%d",&a[i]);

	for(int i=0;i<k;i++)
	{
		next_permutation(a+1,a+1+n);
		 for(int i=1;i<=n;i++) printf("%d ",a[i]);
		printf("\n");
	}

	for(int i=1;i<k;i++) prev_permutation(a+1,a+1+n);
	for(int i=1;i<=n;i++) printf("%d ",a[i]);

}
//next_permutation(a+1,a+1+n);求出a的下一个排列
//prev_permutation(a+1,a+1+n);求出a的上一个排列
```

2.字符串hash

八数码问题的搜索，怎么记录状态？

3	5	2
7	8	6
0	4	1

我们可以把该九宫格看做**字符串**“352786041”，
并标记该字符串已出现过。
怎样标记该字符串呢？

hash表

HASH表(散列表)

它通过把关键码值映射到表中一个位置来访问记录，以加快查找的速度。这个映射函数叫做散列函数(hsah函数)，存放记录的数组叫做散列表。——百度百科

3	5	2
7	8	6
0	4	1

我们可以把该九宫格看做字符串“352786041”，并标记该字符串已出现过。

对于字符串“352786041”，如果我们能构造一个hash函数，将其转换成一个合理范围内的数字x，将mark[x]=true,就完成了对该字符串进行了标记。

如果另一个字符串，通过该hash函数转换后得到的结果也是x，就说明该字符串已出现过。

BKDRhash

字符串hash模板，BKDR_Hash

```
bool mark[524288];
unsigned int BKDRHash(string str)  //处理字符串
{
    unsigned int seed = 131;    // hash种子，一般取质数：31,131,1313,13131,131313 等
    unsigned int hash = 0, i=0, Len=str.length();
    while (i<Len) hash = hash * seed + str[i++];   //计算hash值
    return (hash & 0x7FFFF);        //0x7FFFF为十六进制，==524287=2^19-1
                                    //保证最后的hash值在0到524287之间
    //如果想要最后的hash值在1000000之间，就return(hash % 1000000)
}

int main()      //输入10000个学生的名字，统计其中不同的名字的个数
{
      int i,x,cnt=0;
      string a;
      for(i=1;i<=10000;i++)
      {
          cin>>a;
          x=BKDRHash(a);
          if(mark[x]==false){cnt++; mark[x]=true;}
      }  cout<<cnt;
}
```

字符数组hash模板，BKDR_Hash

```
char str[1000];
unsigned int BKDRHash(char *str)  //处理字符数组
{
    unsigned int seed = 131; // 31 131 1313 13131 131313 等等
    unsigned int hash = 0;
    while (*str)
    {
        hash = hash * seed + (*str++);
    }
    return (hash & 0x7FFFF);  //0x7FFFF==(11111111111111111)2
}
```

其他各种hash

```
#define M  249997
unsigned int RSHash(char *str)
{
    unsigned int b=378551 ;
    unsigned int a=63689 ;
    unsigned int hash=0 ;

    while(*str)
    {
        hash=hash*a+(*str++);
        a*=b ;
    }
    return(hash % M);
}
```

```
#define M  249997
unsigned int JSHash(char*str)
{
    unsigned int
hash=1315423911 ;

    while(*str)
    {
        hash^=((hash<<5)+(*str++)
                +(hash>>2));
    }

    return(hash % M);
}
```

```
#define M  249997
unsigned int SDBMHash(char*str)
{
   unsigned int hash=0 ;

    while(*str)
    {
        hash=(*str++)+(hash<<6)
                +(hash<<16)-hash ;
    }

    return(hash % M);
}
```

```
#define M  249997
unsigned int DJBHash(char*str)
{
    unsigned int hash=5381 ;

    while(*str)
    {
        hash+=(hash<<5)+(*str++);
    }

    return(hash % M);
}
```

```
#define M  249997
unsigned int ELFHash(char*str)
{
    unsigned int hash=0 ;
    unsigned int x=0 ;
    while(*str)
    {
        hash=(hash<<4)+(*str++);
        if((x=hash&0xF0000000L)!=0)
        {
            hash^=(x>>24);
            hash&=~x ;
        }
    }
    return(hash % M);
}
```



```
#define M  249997
unsigned int PJWHash(char*str)
{
    unsigned int BitsInUnignedInt=(unsigned
int)(sizeof(unsigned int)*8);
    unsigned int ThreeQuarters=(unsigned
int)((BitsInUnignedInt*3)/4);
    unsigned int OneEighth=(unsigned
int)(BitsInUnignedInt/8);
    unsigned int HighBits=(unsigned
int)(0xFFFFFFFF)<<(BitsInUnignedInt-OneEighth);
    unsigned int hash=0 ;
    unsigned int test=0 ;
    while(*str)
    {
        hash=(hash<<OneEighth)+(*str++);
        if((test=hash&HighBits)!=0)
        {

hash=((hash^(test>>ThreeQuarters))&(~HighBits));
        }
    }
    return(hash % M);
}
```

字符串hash效率对比详见：《字符串hash函数》
课后阅读2：《hash表》

字符串hash习题：1809，3708

整数hash习题：3704,3705,3706