KMP算法（3461）

Kmp算法神奇在于它的next数组，next[i]保存以第i位结束的后缀和以第一位为开头的前缀的最长公共字符串的结束位置。

a a b a a c a a b a a c a

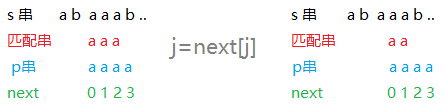
0 1 0 1 2 0 1 2 3 4 5 0 1

每次拿着子串p在母串s中匹配的时候，只要碰到p[j+1]!=s[i+1],则j=next[j];

if(s[i+1]==p[j+1]||j==-1){i++;j++;}

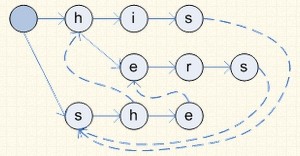
else j=next[j];

这样证明时间复杂度为O（n）,假设s[i+1]==p[j+1],那么匹配串(匹配串为当前子串在母串中匹配的一段区间)右端+1，否则j=next[j]意味着匹配串左端+（j-next[j]），无论左端，右端都单调递增，所以为O（n）



AC 自动机（2222）

AC自动机核心思想与kmp一样，不过由于它是对于多串匹配的，它建立的next数组（即ac自动机的f[]数组）可以从一个子串p1跳到另一个子串p2上（图中的虚线）。



而对于统计答案的时候，我们需要借助last数组，假设子串中同时有p1=abc和p2=ab和p3=bc，我们在走到p1中的b的时候需要找到p2,在走到p1的c的时候需要找到p3

**HDU 2222 Keywords Search**

题意：给定N(N <= 10000)个长度不大于50的模式串，再给定一个长度为L(L <= 106)目标串，求目标串出现了多少个模式串。

题解：AC自动机模板题，在每个trie结点存储一个count值，每次插入一个单词的时候对单词结尾结点的count值进行自增(不能将count值直接置为1，因为有可能模式串中有多个相同的串，它们是要被算作多次的)，然后在询问的时候，每次计数完毕之后，将count值标为-1表示它已经被计算过了。最后输出所有count的累加和即可。

**bzoj 2434 阿狸的打字机**

阿里打字机是这样工作的：

输入小写字母，打字机的一个凹槽中会加入这个字母(这个字母加在凹槽的最后)。

按一下印有'B'的按键，打字机凹槽中最后一个字母会消失。

按一下印有'P'的按键，打字机会在纸上打印出凹槽中现有的所有字母并换行，但凹槽中的字母不会消失。

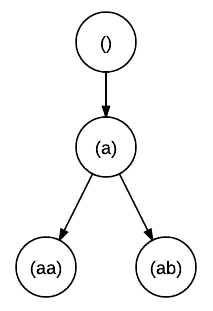
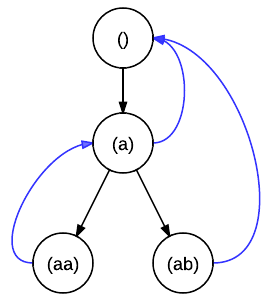
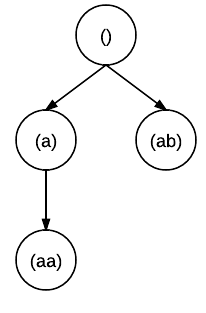
例如，阿狸输入aPaPBbP，纸上被打印的字符如下：a aa ab

我们把纸上打印出来的字符串从1开始顺序编号，一直到n。打字机有一个非常有趣的功能，在打字机中暗藏一个带数字的小键盘，在小键盘上输入两个数(x,y)（其

中1≤x,y≤n），打字机会显示第x个打印的字符串在第y个打印的字符串中出现了多少次。

题解

既然题目中是对各个子串进行匹配，那么我们首先对这些子串建一个Trie树。以样例数据为例，建出的Trie树如下（左图）： 然后再构建其fail指针（中图）：

我们可以发现，如果字符串a可以通过fail指针指向字符串b，那么就说明a串中包含b串。那么对于一个字符串a，如果其中有n个节点的fail指针可以指向字符串b，就说明b串在a串中出现了n次。于是我们可以得到一个基于这个思想的朴素离线算法：枚举每个y字串，维护一个计数器，从根一路遍历到y字串的末尾节点，途中对于每个节点，如果其fail指针指向的是某x串的末尾节点，那么就累加这个串的计数器。但是这个算法的复杂度还是可以达到O(ML)。

我们不妨转换一下思路，对于每个x串，只有能通过fail指针指向它的末尾节点的y串节点才能计数。那么我们不妨把fail指针反向，构建一棵fail树。样例数据的fail树如上面右图。

在串(a)的子树中，属于串(aa)的节点有2个(a)和(aa)，属于串(ab)的节点有1个(a)；在串(aa)的子树中，没有属于串(ab)的节点。

由于在一颗树中，一个节点及其子树在DFS序中是连续的一段，那么我们可以用一个树状数组来维护x串末尾节点及其子树上有多少个属于y串的节点。

那么我们可以得到一个离线算法：对fail树遍历一遍，得到一个DFS序，再维护一个树状数组，对原Trie树进行遍历，每访问一个节点，就修改树状数组，对树状数组中该节点的DFS序起点的位置加上1。每往回走一步，就减去1。如果访问到了一个y字串的末尾节点，枚举询问中每个y串对应的x串，查询树状数组中x串末尾节点从DFS序中的起始位置到结束位置的和，并记录答案。这样，我们就得到了一个时间复杂度为O(N+MlogN)的优美的算法。因为N和M都不超过105，所以这个算法是可行的。

**HDU 5955 掷色子**

题意：掷骰子，n个人，每人预测一个长度为L的序列，直至筛子序列的最后L个数与某个人预测的一致为止游戏结束(每个人预测的序列不一样，且长度均为L)。问每个人的获胜概率。

先用ac自动机建立好状态关系，再列方程，用高斯消元求解。

列方程：对于每个点x，假设可以转移到点y，那么dp[y]的胜率要加上dp[x]\*1/6

对于根节点要注意，相比于其它结点，概率要额外加1。

**HDU 2896 病毒侵袭**

题意：N(N <= 500)个长度不大于200的模式串(保证所有的模式串都不相同)，M(M <= 1000)个长度不大于10000的待匹配串，问待匹配串中有哪几个模式串，题目保证每个待匹配串中最多有三个模式串。

题解：构造trie树和fail指针，由于每个模式串都不同，所以每个代表模式串结尾的trie结点存储模式串对应的编号idx，扫描所有带匹配串，对于每个待匹配串利用失败指针模拟匹配，匹配的模式串个数到达三个的时候放弃扫描该串。

可见字符包括空格，所以读入的时候需要用gets()，子结点个数为128。

**HDU 3065 病毒侵袭持续中**

题意：N(N <= 1000)个长度不大于50的模式串(保证所有的模式串都不相同)，一个长度不大于2000000的待匹配串，求模式串在待匹配串中的出现次数。

题解：由于每个病毒串不会完全相同，对于每个病毒串末尾记录一个编号标记，完全匹配后对编号对应的数组进行累加和计算。

**PKU 1204 Word Puzzles**

题意：给定一个L x C(C <= 1000, L <= 1000)的字母矩阵，再给定W(W <= 1000)个字符串，保证这些字符串都会在字母矩阵中出现(8种方向)，求它们的出现位置和方向。

题解：先缓存所有数据，然后对W个字符串建立字典树和失败指针，再扫描字母矩阵所有8个方向的字符串进行匹配。

**ZJY 3228 Searching the String**

       题意：给定一个长度为N(N <= 105)的目标串，然后再给定M(M <= 105)个长度不大于6的字符串，问这些字符串在目标串的出现次数(分可重叠和不可重叠两种)。

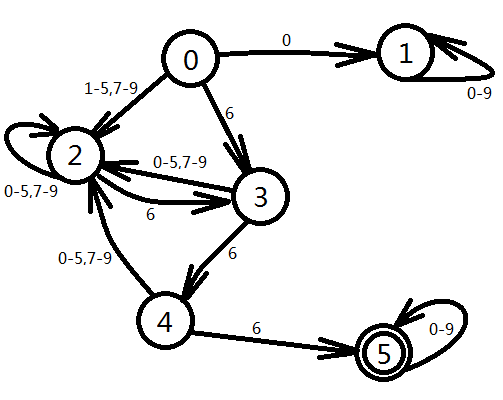
       题解：将M个串作为模式串建立自动机，对于可重叠的情况直接询问即可，类似HDU 3065，不可重叠的情况需要记录每个串的长度Li以及之前这个串匹配到的最大位置Pi，对于当前位置Pos，如果Pi+ Li<= Pos，那么认为和之前的一次匹配没有重叠，计数累加，并且更新Pi=Pos。

为了方便，我把两种计算方式的模式串分别建立了两个自动机。

**PKU 3208 Apocalypse Someday**

题意：求第K(K <= 5\*107)个有连续3个6的数。

题解：建立DFA如下图，其中0为初态，1为非法状态(存在前导0)，2为后缀没有6的状态，3、4、5分别为后缀有1个、2个、3个6的状态，所以5为接收态，因为一旦出现了3个6，那么无论接下来的是什么数都认为是合法数。



既然有了状态转移图就可以轻松地利用状态转移方程通过矩阵乘法求出长度为n有连续3个6的数字的个数，记为P[i],于是我们可以初步确定这是一个几位数字，假设为k位，现在我们要从高位到低位一步一步的把数字求出来。假设第1位为1，那么以1开头的k位数有p[k-1]种答案，拿它与K比较，如果K比p[k-1]大说明第一为1太小了，K-=p[k],继续枚举2,3..

直到第j个数，K<p[k-1]，那么说明第一位就为j.但是对于开头为6的有问题，有可能这个开头的6会与第2位，第3位组成一个666，所以以6为开头的k位数的答案肯定要大于p[k-1]，怎么办，其实我们用p[k]-9\*[k-1]即可。

**PKU 2778 DNA Sequence**

题意：给定m(m <= 10)个DNA片段，每个串长度不超过10。求长度为N(N <= 2\*109)的串中不包括任何给定的DNA片段的串的总数。

题解：利用模式串建立trie图，将trie图转化为矩阵表示，利用二分求幂加速。

为了更加直观，举例说明：

例如，m=2，两个DNA片段分别为A和CAG，可以建立如下AC自动机：

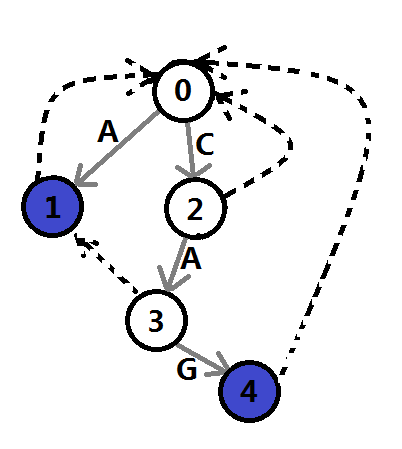


图5

其中，灰色箭头代表树边，虚线代表失败指针，蓝色结点代表终止状态，0为起始状态。然后我们利用它来构造trie图，如图6。

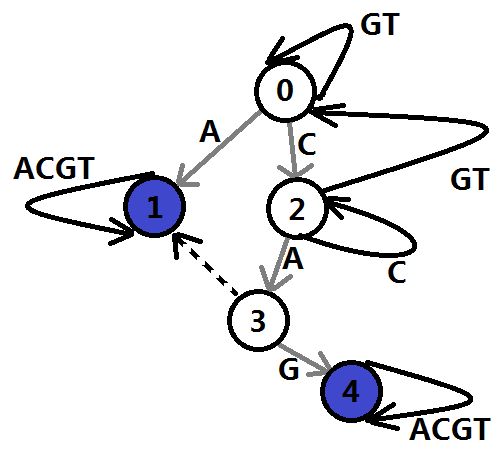


图6

构造方法是利用BFS，依次处理每个状态可以到达哪些状态，建立可达矩阵。

具体步骤如下：

1) 初始状态入队。

2) 每次弹出一个状态结点进行处理，直到队列为空。

a) 对于当前处理的结点P，判断P是否是一个终止结点，如果不是，则判断P的fail指针指向的是否是一个终止结点，一直迭代到fail指针为空，如果迭代过程中找到某个结点为终止结点，那么表示P所在串的某个后缀包含了给定的DNA片段，那么标记P为终止结点，重复2)，否则转b)。

b) 枚举P的所有子结点Q[i](这里的子结点是包含所有字符集的)：

i) 如果Q[i]这个结点不为空，那么DFA[P][Q[i]] ++，Q[i]入队；

ii) 否则沿着P的fail指针一直找，直到找到一个结点S的对应子结点T[i]不为空，那么DFA[P][T[i]]++，如果一直找不到，那么DFA[P][root]++;

当队列为空的时候，有限状态自动机也就构造完毕了，按照这种方式，我们可以发现，除了终止状态，所有状态都有四条出边(A、C、G、T)，但是终止状态并非真正意义上的终止状态，于是我们在终止状态上添加四条回边(指向自己)，表示如果状态进入了终止状态就再也出不去了，这样一来，这个状态机就完整了，任意一个状态只要接收A、C、G、T四个字符中的一个就能进入下一个状态，这样就转化成了一个动态规划问题，假设状态方程DP[i][j]表示长度为i的串在j状态下的字符串个数，那么对于图2的状态机，有如下关系：

DP[i][0] = 2 \* DP[i-1][2] + 2 \* DP[i-1][0];

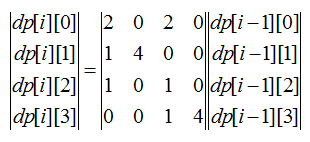
DP[i][1] = DP[i-1][0] + 4 \* DP[i-1][1];

DP[i][2] = DP[i-1][0] + DP[i-1][2];

DP[i][3] = DP[i-1][2] + 4\*DP[i-1][3];

由于在DFA状态处理的时候3号状态为终止状态，所以DP[i][4]其实已经是一个冗余状态了，所以不列入讨论范围。

按照递推方程，DP[N][0] + DP[N][2]就是我们要求的答案，但是N很大，所以可以将DP转移转化成矩阵，即：



然后利用矩阵的二分求幂来加速了。

这题更加直观的理解是：从起点0开始，走N步，经过的路径就是一个DNA串，如果最后到达的是终止状态，那么表示它包含了m个DNA片段中的至少一个。所有路径长度为N，终点非终止状态的路径数目之和就是我们要求的解。

**PKU 1625 Censored!**

题意：给定p(p <= 10)个长度不大于10的模式串，求长度为m(m <= 50)的串中不包含任何模式串的串的种类数。

题解：首先利用模式串建立trie图，用DP[i][j]表示长度为i，状态为j的字符串的种类数，枚举所有字符进行状态转移即可。最后Sum{DP[m][i], i表示非终止状态} 就是答案，这题如果将字符直接进行下标映射，有可能会RE，就是它的字符的ASCII码有可能是在128-255之间的(例如中文)，如果用scanf读入，转换成char就变成了负数，如果映射到下标就RE了，所以在映射之前最好先转成unsigned char。

**PKU 3691 DNA repair**

       题意：给定N(N <= 50)个长度不超过20的模式串，再给定一个长度为M(M <= 1000)的目标串S，求在目标串S上最少改变多少字符，可以使得它不包含任何的模式串（所有串只有ACGT四种字符）。

       题解：利用模式串建立trie图，trie图的每个结点(即下文讲到的状态j)维护三个结构，

Node{

       Node \*next[4];   // 能够到达的四个状态 的结点指针

       int  id;         // 状态ID，用于到数组下标的映射

       int  val;        // 当前状态是否是一个非法状态 （以某些模式串结尾）

}

用DP[i][j]表示长度为i (i <= 1000)，状态为j(j <= 50\*20 + 1)的字符串变成目标串S需要改变的最少字符，设初始状态j = 0，那么DP[0][0] = 0，其它的DP[i][j]均为无穷大。从长度i到i+1进行状态转移，每次转移枚举共四个字符(A、C、G、T)，如果枚举到的字符和S对应位置相同则改变值T=0，否则T=1；那么有状态转移方程 DP[i][j] = Min{ DP[i-1][ fromstate ] + T, fromstate为所有能够到达j的状态 };最后DP[n][j]中的最小值就是答案。

**PKU 1699 Best Sequence**

       题意：给定N(N <= 10)个长度不超过20的模式串，求一个长度最短的串使得它包含所有的模式串。

       题解：利用模式串建立trie图，trie图的每个结点维护一个二进制权值，(val & 2i)不为0表示从根结点到该结点的某条路径上有第i个模式串，用DP[i][j]表示状态为i，模式串的二进制组合为j的最短串的长度，初始化DP[0][0] = 0，然后就转化成了一个在trie图上求(0, 0)到(i, 2n-1)点的最短路问题，最后求出来的DP[i][2n-1] (i < 200, 最多200个结点) 的最小值就是答案。

       注意：本题中的模式串有重复的情况需要特殊处理。

**HDU 2296 Ring**

题意：给定N (N <= 50) 和M(M <= 100)个长度不超过10的字符串以及每个字符串的权值Hi，求一个长度不超过N的字符串使得她包含的权值最大，如果有多个解输出长度最短的，如果还是有多个解，输出字典序最小的。

题解：利用模式串建立trie图，用DP[i][j]表示长度为i，处于j状态下的字符串的最大权值，然后枚举26个字符进行状态转移，转移的过程中需要记录每个状态的前驱，每次进行最大值比较的时候，遇到最大值相等的情况则需要回溯，取字典序最小的。

**HDU 2825 Wireless Password**

       题意：给定m(m <= 10)个长度不大于10的模式串，求长度为n(n <= 25)的至少包含k个模式串的字符串的种数，答案模上20090717。

       题解：类似PKU 1699利用模式串建立trie图，trie图上每个结点表示为一个状态，第i个模式串的权值为2i。用DP[i][j][l]表示长度为i，状态为j，已经有t个模式串的种类数（其中l表示这t个模式串的权值的位或），那么对于每个状态j，输入’a-z’这26个字符后必定能够到达下一个状态，从DP[0][0][0]=1开始迭代计算，最终SUM { DP[n][j][s] , s的二进制表示中1的个数大于等于k}就是答案。

**HDU 3341 Lost 's revenge**

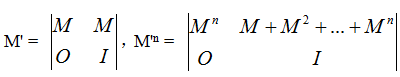
题意：给定N(N <= 50)个长度不超过10的模式串(ACGT串)，再给定一个长度为M(M <= 40)的目标串S，求将目标串重排列，使得它包含最多的模式串，求这个最多的数目。

题解：利用模式串建立trie图，trie图上最多有500个结点( N\*10 )，然后朴素的思想就是用S(i, iA, iC, iG, iT)表示在i状态下，拥有iA个A、iC个C、iG个G、iT个T的串拥有的最多的模式串的个数，但是iA, iC, iG, iT的取值均是[0, 40]，所以我们需要把状态压缩一下，我们知道当四种字符都取10的时候可以让状态数达到最大，即114 = 14641, 所以可以令MaxA、

MaxC、MaxG、MaxT分别表示四种字符出现的个数，那么T字符的权值为1，G字符的权值为(MaxT+ 1)，C字符的权值为(MaxG+ 1) \*(MaxT+ 1)，A字符的权值为(MaxC+ 1) \*(MaxG+ 1) \*(MaxT+ 1)，进行进制压缩之后总的状态数不会超过114,可以用DP[i][j]表示在trie的i号结点时ACGT四个字符个数的压缩状态为j时的字符串包含模式串的最多数目，然后就是进行O(4\*500\*114)的状态转移了。  
 **HDU 2243 考研路茫茫——单词情结**

题意：给定N(N < 6)个长度不超过5的单词，求包含至少一个单词并且长度不超过L(L < 231)的字符串的种数。

题解：利用PKU 2778的方法构造矩阵，由于求的是长度不超过L的种数，即长度为1、2、3...L，假设原有矩阵为M，那么构造一个新的矩阵M'，它由两个原矩阵M，一个零矩阵O和一个单位阵I构成：

该矩阵的右上角的子矩阵就是我们所求的方案矩阵，然后对 M' 二分求幂即可。这里需要总数模264，利用补码的性质，可以直接声明unsigned \_\_int64直接运算即可，不需要用到大数。

**HDU 3247 Resource Archiver**

       题意：给定n(n <= 10)个长度小于等于1000的源字符串以及m(m <= 1000)个病毒串(所有病毒串总长度不超过50000)，求一个串使得它包含所有的源字符串并且不包含任何一个病毒串，求这个字符串的最短长度(所有的串保证都是01串)。

       题解：PKU 1699的加强版。

**PKU 4052 Hrinity**

题意：给定n(n <= 2500)个长度小于等于1100的模式串，求长度不大于5100000的目标串S中包含的模式串的数目，如果包含了模式串A和B，并且B是A的子串，那么只记录A。

题解：建立trie图，每个字符串结尾标记记录模式串编号，进行目标串匹配的时候，利用哈希将所有是目标串子串的模式串标记为1，然后枚举所有标记过的模式串，对他们进行模式匹配，利用同样的方法将模式串的所有模式串子串标记为0，最后统计有多少个模式串的标记为1就是答案了。

**HDU 6086 string**

给你m个01串，问你有多少个长度为2L的01串，满足前半段倒置取反后等于后半段，并且包含所有的m个01串。

考虑单词完全在中线前面或者后面的情况，直接将单词及其倒置取反插入AC自动机，AC自动机每个结点用个tag压位记录单词集合。

对于跨越中线的情况，比如说110010是一个单词，枚举一个中线，

11 | 0010

在前面添加上10，变成10 11 | 0010,

那么其前半部分1011也是一个合法的单词，只不过这种类型的单词只有在dp到长度恰好为L时才能用。

于是AC自动机上每个结点记录两种tag，tag1是原始的单词集合（还有将它倒置再取反的单词），tag2是跨越中线的单词。

f(i,j,S)表示长度为i，到第j个结点，当前集合为S的方案数。第一维滚动。i枚举到n-1时，要用tag2，否则只用tag1