ACM 模板(2017)

By CZWin32768

目录

1. 字符串

1.1 KMP

1.2 AC 自动机

POJ2778 计数：长度为 n 不包含模式串的串的个数：
矩阵 mat[i][j]代表从 i 到 j 走一步的走法（不能走到非法状态，非法状态为 end 和 fail 为 end 的所有状态），那么 mat^n 为走 n 步 i 到 j 的走法，Sigma(mat[0][i])就是长度为 n 并且没有出现过模式串（即非法状态的串的个数）

HDU2243 计数：至少包含一个模式串，长度<=n 的串的个数：

- 联系上题，得到的是长度为n的不包含m个$pattern$串的方案数，记作A^n
- 那么长度从1到n的所有方案数就是$A+A^2+\cdots+A^n$
- 构造矩阵$\begin{pmatrix} A & 1 \\ 0 & 1 \end{pmatrix}$，可以发现$\begin{pmatrix} A & 1 \\ 0 & 1 \end{pmatrix}^n = \begin{pmatrix} A^n & \sum_{i=0}^{n-1} A^i \\ 0 & 1 \end{pmatrix}$，对第一行求和就是我们要的答案
- 那么正难则反，对于字母表下的所有解的方案数就是$1+26+26^2+\cdots+26^n$
- 二者做差就是我们要的答案

HDU2825 计数：计算长度为 n 至少包含 k 个模式串的串的个数：
dp[i][j][mask] 表示长度为 i 的字符串，走到 AC 自动机上第 j 个节点时，状态为 mask 的方案数。转移时我们将状态转移给下一个 AC 自动机节点，设该节点为 u，取了编号为 k 的单词，那么：dp[i + 1][u][mask | k] += dp[i][j][k]
end[i]表示走到 i 节点时包含的模式串的情况（用掩码表示） end[now] |= end[fail[now]];
在弹出队列是执行次操作，这样保证了对任意 Trie 树上的节点的 end 都与其 fail 边连向的节点的 end 值进行过逻辑或操作，这样当前 now 节点只用与 1 级 fail 边连向的 end 进行或操作，而不需要级联。

```
#include<bits/stdc++.h>
using namespace std;

//hdu 2222 经典的问题。找有几个模式串在文本串中出现
const int maxn = 500005, Z = 26, N = 10005;
using BS = bitset<maxn>;

struct AC {
    int Next[maxn][Z], fail[maxn], end[maxn];
    int root, L, offset = int('a');
    BS bs;//用于 HDU2222 标记使用
    int newnode() {
        for(int i = 0; i < Z; i++)
            Next[L][i] = -1;
        end[L++] = 0;
```

```
16.             return L-1;
17.         }
18.         void init() {
19.             L = 0;
20.             root = newnode();
21.         }
22.         void insert(char buf[], int pos) {
23.             // c = 'a' -> buf[i]-'a'
24.             int len = strlen(buf);
25.             int now = root;
26.             for(int i = 0; i < len; i++) {
27.                 if(Next[now][buf[i]-offset] == -1)
28.                     Next[now][buf[i]-offset] = newnode();
29.                 now = Next[now][buf[i]-offset];
30.             }
31.             end[now]++; //如果只记录是否出现过标 1
32.         }
33.         void build() {
34.             queue<int> Q;
35.             fail[root] = root;
36.             for(int i = 0; i < Z; i++)
37.                 if(Next[root][i] == -1)
38.                     Next[root][i] = root;
39.                 else {
40.                     fail[Next[root][i]] = root;
41.                     Q.push(Next[root][i]);
42.                 }
43.             while(!Q.empty()) {
44.                 int now = Q.front(); // if(il[fail[now]]) il[now] =
        1;
45.                 Q.pop();
46.                 for(int i = 0; i < Z; i++)
47.                     if(Next[now][i] == -1)
48.                         Next[now][i] = Next[fail[now]][i];
49.                     else {
50.                         fail[Next[now][i]] = Next[fail[now]][i];
51.                         Q.push(Next[now][i]);
52.                     }
53.             }
54.         }
55.         int query(char buf[]) {
56.             bs.reset();
57.             int len = strlen(buf);
58.             int now = root;
59.             for(int i = 0; i < len; i++) {
60.                 now = Next[now][buf[i]-offset];
61.                 int temp = now;
62.                 while(temp != root) {
63.                     bs.set(temp); //标记哪些出现过 通常使用 res+=end[temp];
64.                     temp = fail[temp];
65.                 }
66.             }
67.             int res = 0;
68.             for(int i = 0; i < L; i++) {
69.                 if(bs.test(i)) res += end[i];
70.             }
71.             return res;
72.         }
73.     };
74.
75.     const int textn = 1000005;
76.     char text[textn], pat[55];
77.     AC ac;
78.
79.     int main() {
80.         int T;
81.         scanf("%d",&T);
82.         while(T--) {
83.             ac.init();
```

```
84.             int n;
85.             scanf("%d",&n);
86.             while(n--) {
87.                 scanf("%s",pat);
88.                 ac.insert(pat, n);
89.             }
90.             ac.build();
91.             scanf("%s",text);
92.             printf("%d\n",ac.query(text));
93.         }
94.     }
95.
```

2. 图论

3. 数据的表示与计算

3.1 矩阵表示

```
1.      //方阵
2.      const int Mat_N = 105;
3.      struct Matrix {
4.          int mat[Mat_N][Mat_N];
5.          int n;
6.          Matrix operator*(const Matrix& b) {
7.              Matrix ret(n);
8.              for(int i = 0; i < n; i++)
9.                  for(int j = 0; j < n; j++)
10.                     for(int k = 0; k < n; k++) {
11.                         ret.mat[i][j] = (ret.mat[i][j] + mat[i][k] *
        b.mat[k][j]) % MOD;
12.                     }
13.             return move(ret);
14.         }
15.         Matrix(int _n, bool isI = false) {
16.             n = _n;
17.             memset(mat, 0, sizeof(mat));
18.             if(isI) {
19.                 for(int i = 0; i < n; i++) mat[i][i] = 1;
20.             }
21.         }
22.         Matrix operator^(ll k) {
23.             Matrix ans(n, 1), a = *this;
24.             while(k) {
25.                 if(k & 1) ans = ans * a;
26.                 k >>= 1;
27.                 a = a * a;
28.             }
29.             return ans;
30.         }
31.     };
32.
33.     //任意大小矩阵：
34.     typedef unsigned long long ull;
35.     typedef long long ll;
36.     typedef vector<ll> vec;
37.     typedef vector<vec> Mat;
38.
39.     Mat mul(Mat &A, Mat &B)
```

```
40.     {
41.         Mat C(A.size(),vec(B[0].size()));
42.         for(int i=0;i<A.size();i++)
43.             for(int k=0;k<B.size();k++)
44.                 for(int j=0;j<B[0].size();j++)
45.                     C[i][j] = ( C[i][j] + A[i][k] * B[k][j] ) % MOD;
46.         return move(C);
47.     }
48.
49.     Mat pow(Mat A, ll n)
50.     {
51.         Mat I(A.size(),vec(A.size()));
52.         for(int i=0;i<A.size();i++) I[i][i] = 1;
53.         while(n > 0)
54.         {
55.             if(n & 1) I = mul(I, A);
56.             A = mul(A, A);
57.             n >>= 1;
58.         }
59.         return move(I);
60.     }
61.     // new Mat -> Mat mat(N, vec(N))
```

4. 结论与黑科技

4.1 快速读入

```
1.      namespace fastIO {
2.          #define BUF_SIZE 100000
3.          //fread -> read
4.          bool IOerror = 0;
5.          inline char nc() {
6.              static char buf[BUF_SIZE], *p1 = buf + BUF_SIZE, *pend =
        buf + BUF_SIZE;
7.              if(p1 == pend) {
8.                  p1 = buf;
9.                  pend = buf + fread(buf, 1, BUF_SIZE, stdin);
10.                 if(pend == p1) {
11.                     IOerror = 1;
12.                     return -1;
13.                 }
14.             }
15.             return *p1++;
16.         }
17.         inline bool blank(char ch) {
18.             return ch == ' ' || ch == '\n' || ch == '\r' || ch ==
        '\t';
19.         }
20.         inline void read(int &x) {
21.             char ch;
22.             while(blank(ch = nc()));
23.             if(IOerror)
24.                 return;
25.             for(x = ch - '0'; (ch = nc()) >= '0' && ch <= '9'; x = x *
        10 + ch - '0');
26.         }
27.         #undef BUF_SIZE
28.     };
```

4.2 树的哈希

$D_h = p_1 p_2^d$ （p 为质数，d 为当前节点的深度）

$$\text{Hash (u)} = \left(D_h(d) + \sum_{v.fa=u} Hash(v) D_h(d) \right)^2$$