ACM模板(2017)

By CZWin32768

目录

[1. 字符串 3](#_Toc490482394)

[1.1 KMP 3](#_Toc490482395)

[1.2 AC自动机 3](#_Toc490482396)

[2. 图论 5](#_Toc490482397)

[3. 数据的表示与计算 5](#_Toc490482398)

[3.1 矩阵表示 5](#_Toc490482399)

[4. 结论与黑科技 6](#_Toc490482400)

[4.1 快速读入 6](#_Toc490482401)

[4.2 树的哈希 7](#_Toc490482402)

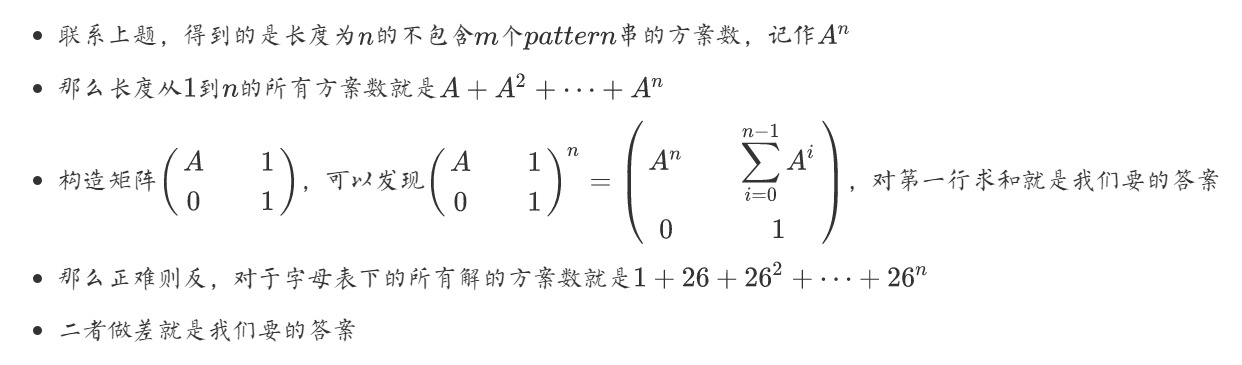
# 字符串

## KMP

## 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**：

**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。

1. #include<bits/stdc++.h>
2. using namespace std;
3. //hdu 2222 经典的问题。找有几个模式串在文本串中出现
4. const int maxn = 500005, Z = 26, N = 10005;
5. using BS = bitset<maxn>;
6. struct AC {
7. int Next[maxn][Z], fail[maxn], end[maxn];
8. int root, L, offset = int('a');
9. BS bs;//用于HDU2222标记使用
10. int newnode() {
11. for(int i = 0; i < Z; i++)
12. Next[L][i] = -1;
13. end[L++] = 0;
14. return L-1;
15. }
16. void init() {
17. L = 0;
18. root = newnode();
19. }
20. void insert(char buf[], int pos) {
21. // c = 'a' -> buf[i]-'a'
22. int len = strlen(buf);
23. int now = root;
24. for(int i = 0; i < len; i++) {
25. if(Next[now][buf[i]-offset] == -1)
26. Next[now][buf[i]-offset] = newnode();
27. now = Next[now][buf[i]-offset];
28. }
29. end[now]++; //如果只记录是否出现过标1
30. }
31. void build() {
32. queue<int> Q;
33. fail[root] = root;
34. for(int i = 0; i < Z; i++)
35. if(Next[root][i] == -1)
36. Next[root][i] = root;
37. else {
38. fail[Next[root][i]] = root;
39. Q.push(Next[root][i]);
40. }
41. while(!Q.empty()) {
42. int now = Q.front(); // if(il[fail[now]]) il[now] = 1;
43. Q.pop();
44. for(int i = 0; i < Z; i++)
45. if(Next[now][i] == -1)
46. Next[now][i] = Next[fail[now]][i];
47. else {
48. fail[Next[now][i]] = Next[fail[now]][i];
49. Q.push(Next[now][i]);
50. }
51. }
52. }
53. int query(char buf[]) {
54. bs.reset();
55. int len = strlen(buf);
56. int now = root;
57. for(int i = 0; i < len; i++) {
58. now = Next[now][buf[i]-offset];
59. int temp = now;
60. while(temp != root) {
61. bs.set(temp); //标记哪些出现过 通常使用 res+=end[temp];
62. temp = fail[temp];
63. }
64. }
65. int res = 0;
66. for(int i = 0; i < L; i++) {
67. if(bs.test(i)) res += end[i];
68. }
69. return res;
70. }
71. };
72. const int textn = 1000005;
73. char text[textn], pat[55];
74. AC ac;
75. int main() {
76. int T;
77. scanf("%d",&T);
78. while(T--) {
79. ac.init();
80. int n;
81. scanf("%d",&n);
82. while(n--) {
83. scanf("%s",pat);
84. ac.insert(pat, n);
85. }
86. ac.build();
87. scanf("%s",text);
88. printf("%d\n",ac.query(text));
89. }
90. }

# 图论

# 数据的表示与计算

## 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. //任意大小矩阵：
33. typedef unsigned long long ull;
34. typedef long long ll;
35. typedef vector<ll> vec;
36. typedef vector<vec> Mat;
37. Mat mul(Mat &A, Mat &B)
38. {
39. Mat C(A.size(),vec(B[0].size()));
40. for(int i=0;i<A.size();i++)
41. for(int k=0;k<B.size();k++)
42. for(int j=0;j<B[0].size();j++)
43. C[i][j] = ( C[i][j] + A[i][k] \* B[k][j] ) % MOD;
44. return move(C);
45. }
46. Mat pow(Mat A, ll n)
47. {
48. Mat I(A.size(),vec(A.size()));
49. for(int i=0;i<A.size();i++) I[i][i] = 1;
50. while(n > 0)
51. {
52. if(n & 1) I = mul(I, A);
53. A = mul(A, A);
54. n >>= 1;
55. }
56. return move(I);
57. }
58. // new Mat -> Mat mat(N, vec(N))

# 结论与黑科技

## 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. };

## 树的哈希

（p为质数，d为当前节点的深度）