字符串与自动机

2015年4月20日 / 97LITTLELEAF11 / 2 COMMENTS

关于字符串的算法，历年的国家集训队论文已经讨论得很详尽了；同时，网络上还涌现出不少逻辑清晰、图文并茂的文章，帮助我们快速理解各类算法。因此，本文不会详细地讨论字符串、自动机的算法，但会提供各类优秀文章的链接，以及BZOJ部分简单题的题解，帮助各位学习。

模式匹配

KMP

- KMP算法详解——Matrix67
- 朱泽园，《多串匹配算法及其启示》，《2004年国家集训队论文集》
- 王鉴浩，《浅谈字符串匹配的几种方法》，《2015年国家集训队论文集》

例题

- 【BZOJ 3670】【NOI 2014】动物园

AC自动机

- AC自动机-DarkRaven
- AC自动机例题
- 王鉴浩，《浅谈字符串匹配的几种方法》，《2015年国家集训队论文集》

1. AC自动机的fail指针指向的节点到根的串，是当前节点到根的串的最长后缀。
2. 常见算法：
   a. 多串匹配。
   b. 与AC自动机性质相关的字符串后缀问题。
   c. 在自动机上预处理两点间走一步的方案数。矩阵快速幂后可以得到任意两点间走n步的方案数。
   d. 自动机上的DP，通常记录一维表示自动机上走到哪个点。还可以与状态压缩相结合，记录走过danger结点的方案。

例题

- 【BZOJ 2938】【POI 2000】病毒：建完AC自动机，如果不走danger点还存在环，那么就存在无限长的串。
- 【BZOJ 3434】【NOI 2011】阿狸的打字机：建完fail树后，每次询问串x在串y中出现的次数，即询问在y在trie中所有点有几个能通过跳fail边跳到x的结尾点。即询问x结尾点这个子树中有几个属于y的点。将询问离线，按y建立邻接表，并记录dfs序，将子树询问转化为前缀和相减。对于每个串，一边在ac自动机上走，一边往树状数组中+1或者-1。跑到串的结尾时，处理所有与该串有关的询问。
- 【BZOJ 3172】【TJOI 2013】单词：在AC自动机上，沿fail边更新每个节点的cnt值。
- 【BZOJ 1030】【JSOI 2007】文本生成器：记$f(i,j)$表示在AC自动机上走j步到i结点的方案数（强制不走danger结点，那么统计出来的答案就是一个串都不包含的方案），答案是用总共可能的方案数（26的串长次方）减去一个串都不包含的方案数。

- 【BZOJ 1195】【HNOI 2006】最短母串：记$f(i,S)$表示在自动机上走到i节点，已经包含了S（2进制压位）子串的最短长度。
- 【BZOJ 2746】【HNOI 2012】旅行问题：询问两串结尾在fail树上的LCA。

回文

Manacher算法

- Manacher算法——ddyyxx
- 徐毅，《浅谈回文子串问题》，《2014年国家集训队论文集》

回文自动机

- 回文自动机

1. 回文自动机在插入节点时候，比AC自动机多一次沿fail指针查找的过程。原因是AC自动机新加入节点是直接连在上一个节点后，而回文自动机在连接时要先沿fail指针判断一遍（判断是否符合回文）。

例题

- 【BZOJ 3676】【APIO 2014】回文串：构出回文自动机，沿fail边更新每个回文串的出现次数。
- 【BZOJ 2160】拉拉队排练：可以使用Manacher，处理出某个位置的p值，来更新一个cnt数组（cnt[i]记录回文串长为i的个数），从大到小统计即可；也可以使用回文自动机，得到每个点往前的最长回文串长度，沿着fail指针来往root更新某个串的出现次数，并统计某个长度的串的个数。
- 【BZOJ 2565】最长双回文串：正反各做一次回文自动机。得到从每个节点开始往前、往后最长的回文串长度。
- 【BZOJ 2342】【SHOI 2011】双倍回文：用Manacher算法构出p数组，每次枚举中心位置'#'（因为要求偶数长度的回文串）i，并找到$\frac{i+p(i)}{2}$这个位置的'#'或这个位置右边的'#'上，记这个位置为k，我们要在k右边找到第一个j，使得$j+p(j)>=i$，意思是在i左侧这一半区域找到一个合法的并且尽量长的回文串，根据对称关系，也就同时找到了i右侧的回文串。对于一个i，如果某个j不能覆盖住

i，那么它更不可能覆盖住之后的点了，所以就抛弃它，这一步可以使用并查集完成（初始时并查集指向自己，一旦废弃该点，则将自己指向右边的'#'，这样询问某一点时，就能快速找到该点右侧第一个可用的'#'了）。注意，以上的操作都是针对'#'字符（Manacher的填充字符），这样才能保证回文串是4的倍数。

后缀自动机与后缀树

- 后缀自动机(FHQ+Neroysq补完)
- 后缀自动机例题
- 陈立杰，《后缀自动机》，NOI2012冬令营讲稿
- 刘研绎，《后缀自动机在字典树上的拓展》，《2015年国家集训队论文集》
- 张天扬，《后缀自动机及其应用》，《2015年国家集训队论文集》

1. 自动机的节点表示一个状态。后缀自动机的接受态节点表示原串的一些后缀；当某个感受态节点代表多类后缀时并需要新加后缀时，转变为中间态节点。因为中间态节点可能代表多个不同后缀，所以在新加入节点时不能直接继承，需要复制一份该节点的father和后继，增加一个新的感受态节点。
2. 一个感受态节点的所有父辈都是感受态节点。
3. 后缀自动机上的father指针（类似AC自动机上fail指针）所指向的节点，是当前节点所表示后缀的后缀。同理可得，一个节点和它的所有father节点拥有相同的后缀。
4. 由前一条定理可得，将一个串逆序构造后缀自动机（等于搞出了原串的前缀自动机），再根据该自动机的father指针构造一棵树，这棵树就是原串的后缀树（所有后缀记在一棵trie上，并压缩了路径），我们需要补齐每条边所代表的序列。原理是：逆序串主链上的每个节点，表示了逆序串的前缀（即原串的后缀）。它们与father相连，表示每个节点与father拥有相同的后缀。即逆序串前缀的后缀等于原串后缀的前缀。所以这棵树就是原串的后缀树了。
5. 后缀树是原串所有后缀的一棵trie树。通常需要路径压缩。
6. 广义后缀自动机是将多串的后缀自动机建在一起，每次插入串的时候从root重新开始，如果某个位置后不存在当前要插入的字符，则直接按照单串后缀自动机的建法新建节点；若已存在这个字符，则看一下这个节点是否代表了多种后缀，如果是则按照

单串的方法将它复制出一个新节点，再接进去。

例题

- 【BZOJ 2946】【Poi 2000】公共串：对第一个串建立后缀自动机，之后的每个串，在后缀自动机上匹配，记录$f(i, j)$表示第i个串匹配到j节点能匹配的最大长度，记得用$f(i, j)$更新$f(i, fail(j))$并统计$dp(j) = \min_{i=1}^{n} f(i, j)$，表示所有串以自动机上$j$节点结尾时的公共子串，答案就是$\max_{j=1}^{m} dp(j)$。
- 【BZOJ 1396&2865】识别子串：后缀自动机的fail树上，叶子节点到叶子节点的父亲所表示的子串就是原串的极小识别子串。我们先把这些串表示的线段拿出来，用线段树完成区间覆盖、单点查询操作。除了这种情况，在极小识别子串的基础上往前扩展的串也是识别子串，例如有一个极小识别子串在原串中是$[l, r]$，那么对于$[\quad, \quad]$这个区间内的点i，都可以用$[i, r]$来作为识别子串，所以$[i, r]$的权值就是$\quad$，我们在线段树上覆盖区间$[\quad, \quad]$，权值为r，每次询问一个点时，将点i的最小覆盖值减去$\quad$即可。
- 【BZOJ 2555】Substring：利用后缀自动机构出fail树，每次加入一个节点，就要让该节点所有父亲的size加1，暴力是这么做的，BZOJ上也确实可以过。正解需要用LCT来维护。
- 【BZOJ 3998】【TJOI 2015】弦论：在后缀自动机上沿fail边更新祖先的出现次数。并统计树上前缀和。
- 【BZOJ 2894】世界线：同3998如出一辙，只不过是多组询问罢了。
- 【BZOJ 3238】【AHOI 2013】差异：逆序后缀自动机构出后缀树，两个后缀的最长公共前缀，即这两个后缀在后缀树上的LCA。问题转化为求一棵树上两两节点的LCA的深度和。
- 【BZOJ 2780】【SPOJ 8093】Sevenk Love Oimaster：建立广义后缀自动机的fail树。询问串s时，先在自动机上跑这个串，得到这个串的位置p，那么后缀树中p的子树节点肯定包含s这个串。问题转化为询问一个子树中有多少种数字，可以离线用树状数组完成。
- 【BZOJ 3277&3473】字符串：我们先用2780的方法处理处后缀树上每个结点在几个原串中出现过，记为cnt_i。由于这是一棵压缩过路径的树，所以如果某个节点的$cnt_i >= k$，则这个节点初始的可行子串的个数是$\quad$。由于广义后缀自动机的一个节点可能是很多串的节点，所以累计子串个数时，我们

不能将答案由儿子累计到父亲而应该将父亲的答案推向儿子，即$v_i += v_{fa(i)}$ 。统计一个原始串的答案就是这个原始串所有在自动机上的节点的v值的和。

- 【BZOJ 3926】【ZJOI 2015】诸神眷顾的幻想乡：询问树上不同子串个数。从叶子节点开始遍历，建立广义后缀自动机。根据后缀自动机的fail树的性质（某个子串的fail是这个子串的最大后缀），因此答案就是

$$\sum^{tot} dep_i - dep_{fa(i)}$$

模板

AC自动机

```
int tot;
struct AC{
    int ch[M][26], danger[M], fail[M], q[M];
    bool vis[M], in[M];

    void insert(char s[]) {
        int len = strlen(s), now = 0;
        for (int i = 0; i < len; i++) {
            int alp = s[i] - '0';
            if (!ch[now][alp]) ch[now][alp] = ++tot;
            now = ch[now][alp];
        }
        danger[now] = 1;
    }
    void build() {
        int u, v, head = 0, rear = 0;
        q[rear++] = 0;
        while (head != rear) {
            u = q[head++];
            for (int i = 0; i < 26; i++) {
                if ((v = ch[u][i])) {
                    if (u) {
                        fail[v] = ch[fail[u]][i];
                        danger[v] |= danger[fail[v]];
                    }
                    q[rear++] = v;
                }
                else if (u) ch[u][i] = ch[fail[u]][i];
            }
        }
    }
}A;
```

回文自动机

```
struct PT{
    int tot, last, fail[MaxN], ch[MaxN][26], len[MaxN], cnt[
    char s[MaxN];

```

```
    void init() {
        tot = 1; last = 0;
        fail[0] = fail[1] = 1;
        len[0] = 0; len[1] = -1;
    }
    int find(int x, int pos) {
        while (s[pos - len[x] - 1] != s[pos]) x = fail[x];
        return x;
    }
    int insert(int pos) {
        int x = find(last, pos), alp = s[pos] - 'a';
        if (!ch[x][alp]) {
            int t = find(fail[x], pos);
            tot++;
            fail[tot] = ch[t][alp];
            len[tot] = len[x] + 2;
            ch[x][alp] = tot;
        }
        last = ch[x][alp];
        cnt[last]++;
        return len[last];
    }
}T;
```

后缀自动机

```
int tot = 1, lastnode = 1;
struct SAM{
    int ch[MaxM][26], dep[MaxM], fa[MaxM];

    inline int newnode(int _dep) {
        dep[++tot] = _dep;
        return tot;
    }
    void insert(int alp) {
        int u, p = newnode(dep[lastnode] + 1);
        for (u = lastnode; u && !ch[u][alp]; u = fa[u])
            ch[u][alp] = p;
        if (!u) fa[p] = 1;
        else {
            int v = ch[u][alp];
            if (dep[v] == dep[u] + 1) fa[p] = v;
            else {
                int nv = newnode(dep[u] + 1);
                fa[nv] = fa[v];
                fa[v] = fa[p] = nv;
                memcpy(ch[nv], ch[v], sizeof(ch[v]));
                for (; u && ch[u][alp] == v; u = fa[u])
                    ch[u][alp] = nv;
            }
        }
        lastnode = p;
    }
}S;
```

广义后缀自动机

```
struct SAM{
    int tot, last, ch[M][26], fa[M], dep[M];

    inline int newnode(int _dep) {
        dep[++tot] = _dep;
        return tot;
    }
    void insert(int alp, int strings) {
        int u, p = ch[last][alp];
        if (p) {
            if (dep[p] == dep[last] + 1) last = p;
            else {
                int np = newnode(dep[last] + 1);
                fa[np] = fa[p];
                fa[p] = np;
                memcpy(ch[np], ch[p], sizeof(ch[p]));
                for (u = last; u && ch[u][alp] == p; u = fa[
                    ch[u][alp] = np;
                last = np;
            }
        } else {
            int np = newnode(dep[last] + 1);
            for (u = last; u && !ch[u][alp]; u = fa[u])
                ch[u][alp] = np;
            if (!u) fa[np] = 1;
            else {
                int v = ch[u][alp];
                if (dep[v] == dep[u] + 1) fa[np] = v;
                else {
                    int nv = newnode(dep[u] + 1);
                    fa[nv] = fa[v];
                    fa[v] = fa[np] = nv;
                    memcpy(ch[nv], ch[v], sizeof(ch[v]));
                    for (; u && ch[u][alp] == v; u = fa[u])
                        ch[u][alp] = nv;
                }
            }
            last = np;
        }
    }
}A;
```

另外有一种更简单的写法

```
struct SAM{
    int tot, last, ch[M][6], fa[M], dep[M];

    inline int newnode(int _dep) {
        dep[++tot] = _dep;
        return tot;
    }
    void insert(int alp, int strings) {
        int u, p = ch[last][alp];
        if (p && dep[p] == dep[last] + 1) last = p;
        else {
            int np = newnode(dep[last] + 1);
            for (u = last; u && !ch[u][alp]; u = fa[u])
                ch[u][alp] = np;
            if (!u) fa[np] = 1;
            else {
                int v = ch[u][alp];
```

```
                if (dep[v] == dep[u] + 1) fa[np] = v;
                else {
                    int nv = newnode(dep[u] + 1);
                    fa[nv] = fa[v];
                    fa[v] = fa[np] = nv;
                    memcpy(ch[nv], ch[v], sizeof(ch[v]));
                    for (; u && ch[u][alp] == v; u = fa[u])
                        ch[u][alp] = nv;
                }
            }
            last = np;
        }
    }
}A;
```

OI

自动机