串

字符串?

一类跟串有关的问题

签到？难题？思维题？数据结构？

抛砖引玉

- kmp/扩展kmp
- trie树/01字典树
- ac自动机/ac自动机上的动态规划
- 后缀数组
- 后缀自动机
- 马拉车算法
- 回文自动机
- ...

本节

- 马拉车算法
- 字符hash

什么是回文串？中心对称

如何求一个串最长回文子串？n^{2}暴力？

马拉车算法

首先用一个非常巧妙的方式，将所有可能的奇数/偶数长度的回文子串都转换成了奇数长度：在每个字符的两边都插入一个特殊的符号。比如 abba 变成 #a#b#b#a#， aba变成 #a#b#a#。 为了进一步减少编码的复杂度，可以在字符串的开始加入另一个特殊字符，这样就不用特殊处理越界问题，比如$#a#b#a#

以字符串12212321为例，经过上一步，变成了 S[] = "$#1#2#2#1#2#3#2#1#";

然后用一个数组 P[i] 来记录以字符S[i]为中心的最长回文子串向左/右扩张的长度

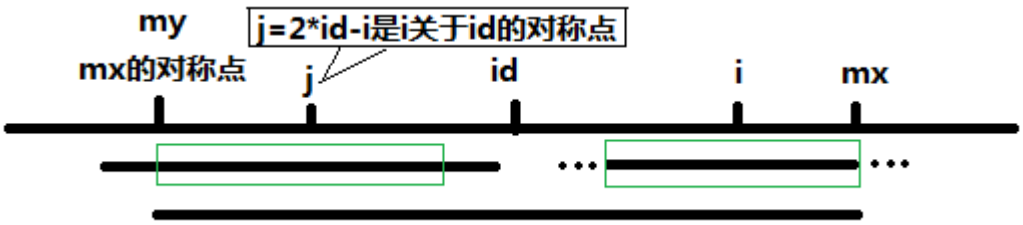
那么怎么计算P[i]呢？该算法增加两个辅助变量（其实一个就够了，两个更清晰）id和mx，其中 id 为已知的 {右边界最大} 的回文子串的中心，mx则为id+P[id]，也就是这个子串的右边界。

如果mx > i，那么P[i] >= MIN(P[2 * id - i], mx - i)。

```
//记j = 2 * id - i，也就是说 j 是 i 关于 id 的对称点(j = id - (i - id))
if (mx - i > P[j])
    P[i] = P[j];
else /* P[j] >= mx - i */
    P[i] = mx - i; // P[i] >= mx - i，取最小值，之后再匹配更新。
```

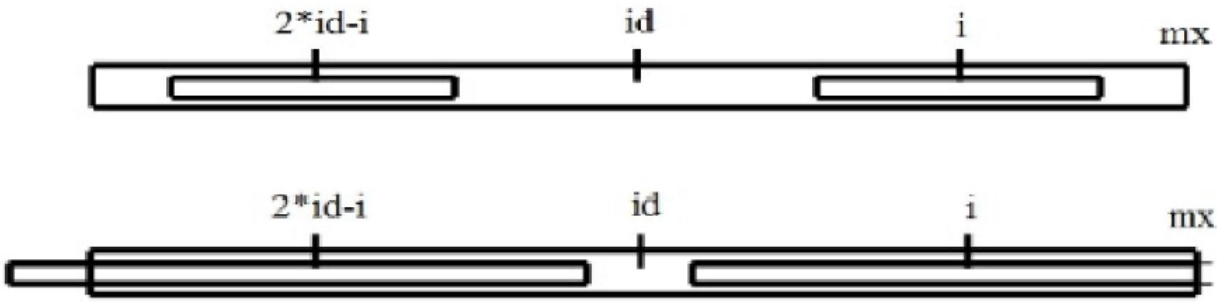
当 mx - i > P[j] 的时候，以S[j]为中心的回文子串包含在以S[id]为中心的回文子串中，由于 i 和 j 对称，以S[i]为中心的回文子串必然包含在以S[id]为中心的回文子串中，所以必有 P[i] = P[j]，见下图。

当 P[j] >= mx - i 的时候，以S[j]为中心的回文子串不一定完全包含于以S[id]为中心的回文子串中，但是基于对称性可知，下图中两个绿框所包围的部分是相同的，也就是说以S[i]为中心的回文子串，其向右至少会扩张到mx的位置，也就是说 P[i] >= mx - i。至于mx之后的部分是否对称，就只能老老实实去匹配了。




```
int p[1000], mx = 0, id = 0;
memset(p, 0, sizeof(p));
for (i = 1; s[i] != '\0'; i++) {
    p[i] = mx > i ? min(p[2*id-i], mx-i) : 1;
    while (s[i + p[i]] == s[i - p[i]]) p[i]++;
    if (i + p[i] > mx) {
        mx = i + p[i];
        id = i;
    }
}
```

一个性质



如果mx不更新，就不会出现本质不同的回文子串，因为前面已经出现过了；而每扩展一次mx，最多新出现一个本质不同的回文子串。 于是得到性质：#一个字符串最多只有n个本质不同的回文子串#。这个性质很重要，有些题会用到，需要这个性质去分析。

字符hash

什么是hash？ 为什么需要？将子串快速映射成一个数值，比较两个串是否相同，只需要O(1)比较hash值是否相同

字符串Hash函数把一个任意长度的字符串映射成一个非负整数，并且其冲突概率**几乎**为零 取一固定值P，把字符串看作P进制数，并分配一个大于0的数值，代表每种字符。 一般来说，我们分配的数值都远小于P。例如，对于小写字母构成的字符串，可以令a=1,b=2,...,z=26。a=1,b=2,...,z=26。 取一固定值M，求出该P进制数对M的余数，作为该字符串的Hash值。

一般来说，我们取P=131或P=13331，此时Hash值产生冲突的概率极低，只要Hash值相同，我们就可以认为原字符串是相等的。通常我们取$M=2^{64}$，即直接使用unsigned long long类型存储这个Hash值，在计算时不处理算术溢出问题，产生溢出时相当于自动对2^{64}取模，这样可以避免低效的取模运算。

对字符串的各种操作，都可以直接对P进制数进行算数运算反映到Hash值上。

如果我们已知字符串S的Hash值为H(S)，那么在S后添加一个字符c构成的新字符串S+c的Hash值就是$H(S+c)=(H(s)*P+value[c])\%M$。其中乘P就相当于P进制下的左移运算，value[c]是我们的为c选定的代表数值。

如果我们已知字符串S的Hash值为H(S)，字符串S+T的Hash值为H(S+T)，那么字符串T的Hash值$H(T)=(H(S+T)-H(S)*P^{length(T)})\%M$。这就相当于通过P进制下在S后边补0的方式，把S左移到与S+T的左端对其，然后二者相减就得到了H(T)。

> 例如，S= “abc” ，c= “d” ，T= “xyz” ，则:
> S表示为P进制数: 1 2 3
> $H(S) = 1 * P^2 + 2 * P + 3$
> $H(S + c) = 1 * P^3 + 2 * P^2 + 3 * P + 4 = H(S) * P + 4$
> S+T表示为P进制数: 1 2 3 24 25 26
> $H(S + T) = 1 * P^5 + 2 * P^4 + 3 * P^3 + 24 * P^2 + 25 * P + 26$
> S在P进制下左移length(T) 位: 1 2 3 0 0 0
> 二者相减就是T表示为P进制数: 24 25 26
> $H(T) = H(S + T) - (1 * P^2 + 2 * P + 3) * P^3 = 24 * P^2 + 25 * P + 26$

不懂？用就完事!

```
//字符串从1开始
typedef unsigned long long ull;
const ull base = 131;
struct My_Hash
{
    ull p[maxn],hs[maxn];
    void Insert( char s[] )
    {
        int len = strlen(s+1);
        p[0] = 1,hs[0] = 0;
        for ( int i=1 ; i<=len ; i++ )
            p[i] = p[i-1]*base,hs[i] = hs[i-1]*base+(ull)s[i];
    }
    ull GetHash( int l , int r )
    {
        return (ull)hs[r]-p[r-l+1]*hs[l-1];
    }
}S;
```

双模 两个模数，更小的碰撞概率!

```
typedef unsigned long long ull;
const ull mod1 = 1e9+7;
const ull mod2 = 1e9+9;
const ull base1 = 131;
const ull base2 = 233;

struct My_Hash
{
    ull hs1[maxn],p1[maxn];
    ull hs2[maxn],p2[maxn];
    void Insert( char s[] )
    {
        int len = strlen(s+1);
        hs1[0] = 0,p1[0] = 1;
        hs2[0] = 0,p2[0] = 1;
        for ( int i=1 ; i<=len ; i++ )
        {
            p1[i] = p1[i-1]*base1%mod1;
            p2[i] = p2[i-1]*base2%mod2;
            hs1[i] = ( hs1[i-1]*base1%mod1+(ull)s[i] )%mod1;
            hs2[i] = ( hs2[i-1]*base2%mod2+(ull)s[i] )%mod2;
        }
    }
```

```
    pair<ull,ull> GetHash( int l , int r )
    {
        ull tmp1 = hs1[r];
        ull tmp2 = hs2[r];
        tmp1 = ( tmp1-p1[r-l+1]*hs1[l-1]%mod1+mod1 )%mod1;
        tmp2 = ( tmp2-p2[r-l+1]*hs2[l-1]%mod2+mod2 )%mod2;
        return make_pair( tmp1 , tmp2 );
    }
}s;
```

例题

一个主串，一个模式串，问模式串在主串中出现了几次 处理两个hash，在主串中枚举子串，O(1)比较hash值是否相等

多个串，询问某两个子串的最长公共前缀

哈希表

- map
- 链式（类似建图）
