最小表示法

最小表示法是用于解决字符串最小表示问题的方法（废话

字符串的最小表示

循环同构

当字符串 S中可以选定一个位置 i满足

$$S[i \cdots n] + S[1 \cdots n-1] = T$$

则称 S与 T循环同构

最小表示

字符串 S的最小表示为与 S循环同构的所有字符串中字典序最小的字符串

simple的暴力

我们直接比较与 S同构的所有字符串，共 n个。

每次保留当前字典序最小的字符串与剩余的字符串比较。

```
int k=0,i=0,j=1;
for(;j<n;j++)
```

```
{
    if(sec[(i+k)%n]==sec[(j+k)%n])
    {
        k++;
    }
    else
    {
        if(sec[(i+k)%n]>sec[(j+k)%n]%n)
        {
            i=j;
        }
        k=0;
    }
}
```

随机数据下表现良好，但是可以构造特殊数据卡掉。

例如：对于 $aaa\cdots aaa$,不难发现这个算法的复杂度退化为 $O(n^2)$

我们发现，当字符串中出现多个连续重复子串时，此算法效率降低，我们考虑优化这个过程。

最小表示法

算法核心

考虑对于一对字符串 A, B,它们在原字符串 S中的起始位置分别为 i, j,且它们的前 k 个字符均相同，即

$$A[i\cdots i+k-1]=B[j\cdots j+k-1]$$

不妨先考虑 $A[i+k]>B[j+k]$的情况，我们发现起始位置下标 l满足 $i\le l\le i+k$的字符串均不能成为答案。因为对于任意一个字符串 S_{i+p}（表示以 $i+p$为起始位置的字符串）一定存在字符串 S_{j+p}比它更优。

所以我们比较时可以跳过下标 $l\in[i,i+k]$,直接比较 S_{i+k+1}

这样，我们就完成了对于上文暴力的优化。

时间复杂度

$O(n)$

~~证明：显然~~

算法流程

1. 初始化指针 i为 0，j为 1；初始化匹配长度 k为 0
2. 比较第 k位的大小，根据比较结果跳转相应指针。若跳转后两个指针相同，则随意选一个加一以保证比较的两个字符串不同
3. 重复上述过程，直到比较结束
4. 答案为 i, j中较小的一个

代码

```
int k=0,i=0,j=1;
while(k<n&&i<n&&j<n)
{
    if(sec[(i+k)%n]==sec[(j+k)%n])
    {
        k++;
    }
    else
    {
        sec[(i+k)%n]>sec[(j+k)%n]?i=i+k+1:j=j+k+1;
        if(i==j) i++;
        k=0;
    }
}
i=min(i,j);
```