1. kmp

next数组就是第j个元素前缀等于后缀的大小。

完全循环：循环节长度n-nex【n（或者n-1）】，而且满足n%循环节长度==0.

不完全循环：比如abca循环节同完全循环。

1. exkmp

扩展kmp

Nex数组指的是从j开始的后缀与原串的最长公共前缀

void qnxt(char \*c)

{

int len = strlen(c);

int p = 0, k = 1, l; //我们会在后面先逐位比较出 nxt[1] 的值，这里先设 k 为 1

//如果 k = 0，p 就会锁定在 |c| 不会被更改，无法达成算法优化的效果啦

nxt[0] = len; //以 c[0] 开始的后缀就是 c 本身，最长公共前缀自然为 |c|

while(p + 1 < len && c[p] == c[p + 1]) p++;

nxt[1] = p; //先逐位比较出 nxt[1] 的值

for(int i = 2; i < len; i++)

{

p = k + nxt[k] - 1; //定义

l = nxt[i - k]; //定义

if(i + l <= p) nxt[i] = l; //如果灰方框小于初始的绿方框,直接确定 nxt[i] 的值

else

{

int j = max(0, p - i + 1);

while(i + j < len && c[i + j] == c[j]) j++; //否则进行逐位比较

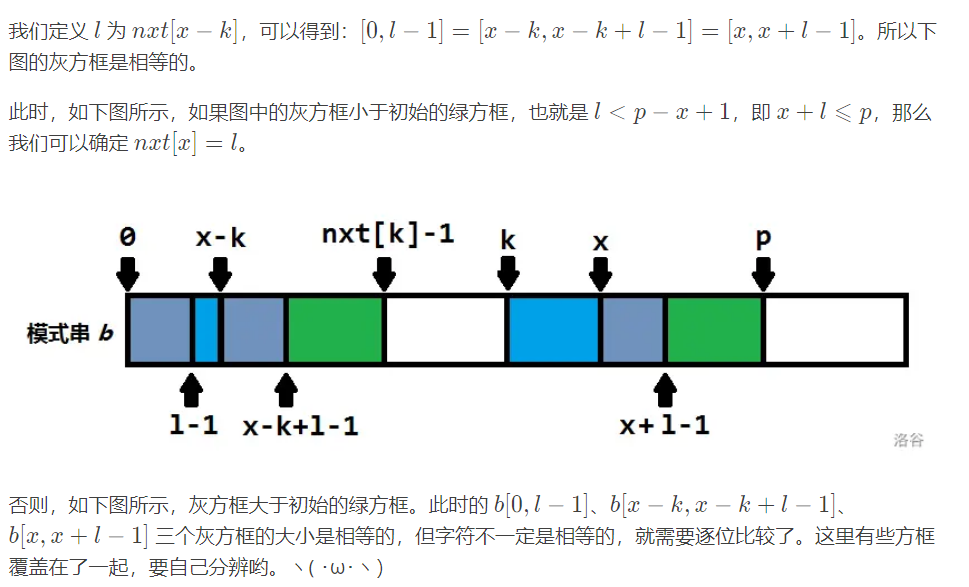
nxt[i] = j;

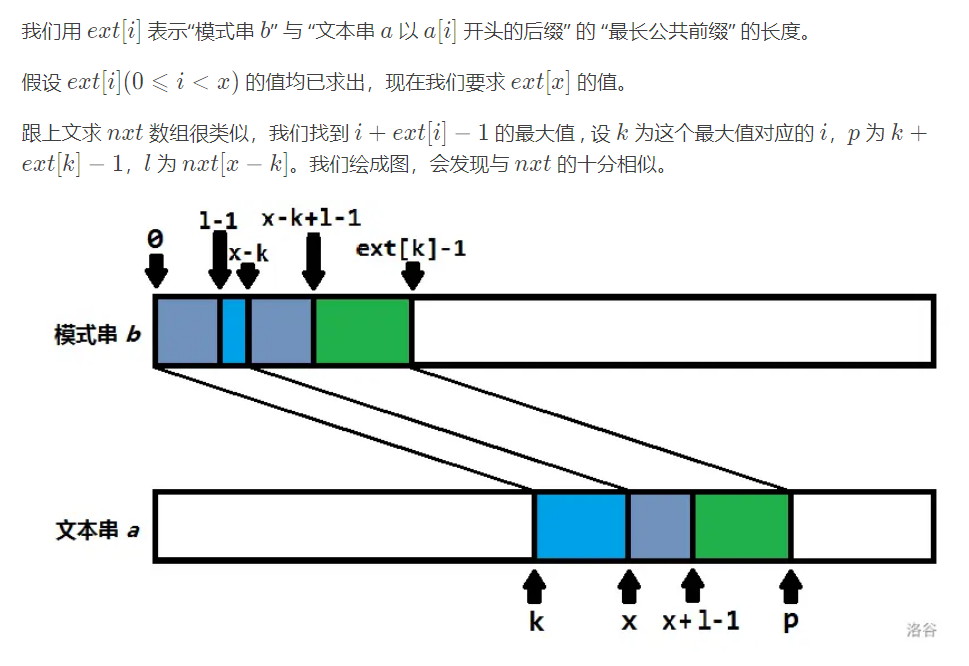
k = i; //此时的 x + nxt[x] - 1 一定刷新了最大值，于是我们要重新赋值 k

}

}

}





void exkmp(char \*a, char \*b)

{

int la = strlen(a), lb = strlen(b);

int p = 0, k = 0, l;

while(p < la && p < lb && a[p] == b[p]) p++; //先算出初值用于递推

ext[0] = p;

for(int i = 1; i < la; i++) //下面都是一样的逻辑啦

{

p = k + ext[k] - 1;

l = nxt[i - k];

if(i + l <= p) ext[i] = l;

else

{

int j = max(0, p - i + 1);

while(i + j < la && j < lb && a[i + j] == b[j]) j++;

ext[i] = j;

k = i;

}

}

}

4.马拉车的思想

5.ac自动机

注意fail树的含义与应用构建

Ac自动机上dp注意fail树子节点代表匹配到的字符串包含了该字符串

Dpij表示长度为i的字符串在自动机上匹配到节点j。更新时选一个已经匹配的点去更新他可以到达的点。

Sam求字典序第k小字串

其中sum表示有多少个子串经过这个节点。（t为0为不同位置相同字串算一个）

struct Suffix{

ll ch[maxn<<1][26],fa[maxn<<1],l[maxn<<1],size[maxn<<1],k[maxn<<1],c[maxn<<1];

ll last,tot,sum[maxn<<1];

void init(){

last=tot=1;

memset(ch[1],0,sizeof ch[1]);

memset(fa,0,sizeof fa);

memset(l,0,sizeof l);

memset(size,0,sizeof size);

memset(k,0,sizeof k);

memset(c,0,sizeof c);

}

void ins(int c,int pos){

int p=last,np=++tot;last=np;l[np]=l[p]+1;

memset(ch[tot],0,sizeof ch[tot]);

for(;p&&!ch[p][c];p=fa[p]){

ch[p][c]=np;

}

if(!p){

fa[np]=1;

}else{

int q=ch[p][c];

if(l[p]+1==l[q]){

fa[np]=q;

}else{

int nq=++tot;l[nq]=l[p]+1;

memcpy(ch[nq],ch[q],sizeof(ch[q]));

fa[nq]=fa[q];fa[q]=fa[np]=nq;

for(;ch[p][c]==q;p=fa[p])ch[p][c]=nq;

}

}

size[np]=1;

}

void build(){

init();

int n=strlen(s+1);

for(int i=1;i<=n;++i){

ins(s[i]-'a',i);

}

for(int i=1;i<=tot;++i){

c[l[i]]++;

}

for(int i=1;i<=tot;++i){

c[i]+=c[i-1];

}

for(int i=1;i<=tot;++i){

k[c[l[i]]--]=i;

}

for(int i=tot;i>=1;--i){

int id=k[i];

size[fa[id]]+=size[id];

}

for(int i=1;i<=tot;++i){

t==0?(sum[i]=size[i]=1):(sum[i]=size[i]);

}

size[1]=sum[1]=0;

for(int i=tot;i>=1;--i){

for(int j=0;j<26;++j){

if(ch[k[i]][j]){

sum[k[i]]+=sum[ch[k[i]][j]];

}

}

}

}

void prin(int x,int y){

if(y<=size[x])return;

y-=size[x];

for(int i=0;i<26;++i){

ll R=ch[x][i];

if(!R){

continue;

}

if(y>=sum[R]){

y-=sum[R];

continue;

}

putchar(i+'a');

prin(R,y);

return;

}

}

void shuchu(){

if(sum[1]<K)printf("-1");

else{

prin(1,K);

}

}

}sam;

最长公共子串：建立n-1个sam用第一个在其他的上面匹配，取min

int p=1,plen=0;//当前点,匹配长度

for (int i=0,c;i<len2;i++){

c=s2[i]-'a';

if (!a[p].t[c]){

while(p!=1&&!a[p].t[c])p=a[p].f;

plen=a[p].len; //如果没有转移边,不断跳fail

}

if (a[p].t[c]){

p=a[p].t[c];plen++; //能匹配,plen++

}slen[i]=plen;

}

int ans=0; for (int i=0;i<len2;i++)ans=max(ans,slen[i]); printf("%d",ans);