

Algorithm Library

`magic::team.getname()`

South China Normal University

August 29, 2024

Contents

| | |
|-----------------------------|----------|
| 头文件 | 2 |
| DEBUG 头 | 2 |
| __int128 输出流 | 2 |
| 常用数学函数 | 2 |
| 数学 | 3 |
| 欧拉筛 | 3 |
| 取模类 (MInt) | 3 |
| 数据结构 | 5 |
| 并查集 (启发式合并 + 带撤销) | 5 |
| 状压 RMQ | 5 |
| 树状数组 | 6 |
| 线段树 | 7 |
| 字符串 | 9 |
| 字符串哈希 (随机模数) | 9 |
| KMP | 10 |
| Z 函数 | 10 |
| AC 自动机 | 10 |
| 后缀数组 | 11 |
| (广义) 后缀自动机 | 12 |
| Manacher | 13 |
| 回文自动机 | 14 |

头文件

DEBUG 头

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  using i64=long long;
4  using i128=__int128;
5
6  namespace DBG
7  {
8      template <class T>
9      void _dbg(const char *f,T t) { cerr<<f<<'\n'; }
10
11     template <class A,class... B>
12     void _dbg(const char *f,A a,B... b)
13     {
14         while (*f!=',') cerr<<*f++;
15         cerr<<'\n';
16         _dbg(f+1,b...);
17     }
18
19     template <class T>
20     ostream& operator << (ostream& os,const vector<T> &v)
21     {
22         os<<"[ ";
23         for (const auto &x:v) os<<x<<", ";
24         os<<"]";
25         return os;
26     }
27
28     #define dbg(...) _dbg(#__VA_ARGS__, __VA_ARGS__)
29 }
30
31 using namespace DBG;
```

__int128 输出流

```
1  ostream &operator << (ostream &os,i128 n)
2  {
3      string s;
4      bool neg=n<0;
5      if (neg) n=-n;
6      while (n)
7      {
8          s+='0'+n%10;
9          n/=10;
10     }
11     if (neg) s+='-';
12     reverse(s.begin(),s.end());
13     if (s.empty()) s+='0';
14     return os<<s;
15 }
```

常用数学函数

```
1  i64 ceilDiv(i64 n,i64 m)
2  {
3      if (n>=0) return (n+m-1)/m;
4      else return n/m;
5  }
6
7  i64 floorDiv(i64 n,i64 m)
8  {
9      if (n>=0) return n/m;
10     else return (n-m+1)/m;
11 }
12
13 i128 gcd(i128 a,i128 b)
14 {
```

```

15     return b?gcd(b,a%b):a;
16 }

```

数学

欧拉筛

```

1  vector<int> minp,primes;
2
3  void sieve(int n)
4  {
5      minp.assign(n+1,0);
6      primes.clear();
7      for (int i=2;i<=n;i++)
8      {
9          if (!minp[i])
10             {
11                 minp[i]=i;
12                 primes.push_back(i);
13             }
14             for (auto p:primes)
15             {
16                 if (i*p>n) break;
17                 minp[i*p]=p;
18                 if (p==minp[i]) break;
19             }
20     }
21 }

```

取模类 (MInt)

```

1  template <class T>
2  constexpr T power(T a,i64 b)
3  {
4      T res=1;
5      for (;b>>=1,a*=a)
6          if (b&1) res*=a;
7      return res;
8  }
9
10 template <int P>
11 struct MInt
12 {
13     int x;
14     constexpr MInt():x{} {}
15     constexpr MInt(i64 x):x{norm(x%getMod())} {}
16
17     static int Mod;
18     constexpr static int getMod()
19     {
20         if (P>0) return P;
21         else return Mod;
22     }
23
24     constexpr static void setMod(int Mod_) { Mod=Mod_; }
25
26     constexpr int norm(int x) const
27     {
28         if (x<0) x+=getMod();
29         if (x>=getMod()) x-=getMod();
30         return x;
31     }
32
33     constexpr int val() const { return x; }
34
35     explicit constexpr operator int () const { return x; }
36
37     constexpr MInt operator - () const
38     {

```

```

39     MInt res;
40     res.x=norm(getMod()-x);
41     return res;
42 }
43
44 constexpr MInt inv() const
45 {
46     assert(x!=0);
47     return power(*this,getMod()-2);
48 }
49
50 constexpr MInt &operator *= (MInt rhs) &
51 {
52     x=1ll*x*rhs.x%getMod();
53     return *this;
54 }
55
56 constexpr MInt &operator += (MInt rhs) &
57 {
58     x=norm(x+rhs.x);
59     return *this;
60 }
61
62 constexpr MInt &operator -= (MInt rhs) &
63 {
64     x=norm(x-rhs.x);
65     return *this;
66 }
67
68 constexpr MInt &operator /= (MInt rhs) &
69 {
70     return *this*=rhs.inv();
71 }
72
73 friend constexpr MInt operator * (MInt lhs,MInt rhs)
74 {
75     MInt res=lhs;
76     res*=rhs;
77     return res;
78 }
79
80 friend constexpr MInt operator + (MInt lhs,MInt rhs)
81 {
82     MInt res=lhs;
83     res+=rhs;
84     return res;
85 }
86
87 friend constexpr MInt operator - (MInt lhs,MInt rhs)
88 {
89     MInt res=lhs;
90     res-=rhs;
91     return res;
92 }
93
94 friend constexpr MInt operator / (MInt lhs,MInt rhs)
95 {
96     MInt res=lhs;
97     res/=rhs;
98     return res;
99 }
100
101 friend constexpr istream &operator >> (istream &is,MInt &a)
102 {
103     i64 v;
104     is>>v;
105     a=MInt(v);
106     return is;
107 }
108
109 friend constexpr ostream &operator << (ostream &os,const MInt &a) { return os<<a.val(); }

```

```

110     friend constexpr bool operator == (MInt lhs,MInt rhs) { return lhs.val()==rhs.val(); }
111
112     friend constexpr bool operator != (MInt lhs,MInt rhs) { return lhs.val()!=rhs.val(); }
113 };
114
115
116 template<>
117 int MInt<0>::Mod=1;
118
119 template<int V,int P>
120 constexpr MInt<P> CInv=MInt<P>(V).inv();

```

数据结构

并查集（启发式合并 + 带撤销）

```

1 struct DSU
2 {
3     int n=0;
4     vector<int> fa,siz;
5     stack<int> s;
6
7     DSU(int n) { init(n); }
8
9     void init(int n)
10    {
11        fa.resize(n);
12        iota(fa.begin(),fa.end(),0);
13        siz.assign(n,1);
14        while (!s.empty()) s.pop();
15    }
16
17    int get(int x) { return fa[x]==x?x:get(fa[x]); }
18
19    void merge(int x,int y)
20    {
21        x=get(x),y=get(y);
22        if (x==y) return;
23        if (siz[x]<siz[y]) swap(x,y);
24        s.push(y),fa[y]=x,siz[x]+=siz[y];
25    }
26
27    void undo()
28    {
29        if (s.empty()) return;
30        int y=s.top();
31        s.pop();
32        siz[fa[y]]-=siz[y];
33        fa[y]=y;
34    }
35
36    void back(int t=0) { while (s.size()>t) undo(); }
37 };

```

状压 RMQ

```

1 template <class T,class Cmp=less<T>>
2 struct RMQ
3 {
4     const Cmp cmp=Cmp();
5     static constexpr unsigned B=64;
6     using u64=unsigned long long;
7     int n;
8     vector<vector<T>> a;
9     vector<T> pre,suf,ini;
10    vector<u64> stk;
11
12    RMQ() {}
13    RMQ(const vector<T> &v) { init(v); }

```

```

14
15 void init(const vector<T> &v)
16 {
17     n=v.size();
18     pre=suf=ini=v;
19     stk.resize(n);
20     if (!n) return;
21     const int M=(n-1)/B+1;
22     const int lg=__lg(M);
23     a.assign(lg+1,vector<T>(M));
24     for (int i=0;i<M;i++)
25     {
26         a[0][i]=v[i*B];
27         for (int j=1;j<B&& i*B+j<n;j++)
28             a[0][i]=min(a[0][i],v[i*B+j],cmp);
29     }
30     for (int i=1;i<n;i++)
31         if (i%B) pre[i]=min(pre[i],pre[i-1],cmp);
32     for (int i=n-2;i>=0;i--)
33         if (i%B!=B-1) suf[i]=min(suf[i],suf[i+1],cmp);
34     for (int j=0;j<lg;j++)
35         for (int i=0;i+(2<<j)<=M;i++)
36             a[j+1][i]=min(a[j][i],a[j][i+(1<<j)],cmp);
37     for (int i=0;i<M;i++)
38     {
39         const int l=i*B;
40         const int r=min(1U*n,l+B);
41         u64 s=0;
42         for (int j=l;j<r;j++)
43         {
44             while (s&&cmp(v[j],v[__lg(s)+l])) s^=1ULL<<__lg(s);
45             s|=1ULL<<(j-l);
46             stk[j]=s;
47         }
48     }
49 }
50
51 //查询区间 [l,r) 的 RMQ
52 T operator()(int l,int r)
53 {
54     if (l/B!=(r-1)/B)
55     {
56         T ans=min(suf[l],pre[r-1],cmp);
57         l=l/B+1,r=r/B;
58         if (l<r)
59         {
60             int k=__lg(r-l);
61             ans=min({ans,a[k][l],a[k][r-(1<<k)]},cmp);
62         }
63         return ans;
64     }
65     else
66     {
67         int x=B*(l/B);
68         return ini[__builtin_ctzll(stk[r-1]>>(l-x))+1];
69     }
70 }
71 };

```

树状数组

```

1 template <class T>
2 struct BIT
3 {
4     int n;
5     vector<T> a;
6
7     BIT(int n_=0) { init(n_); }
8
9     void init(int n_)
10    {

```

```

11     n=n_;
12     a.assign(n,T{});
13 }
14
15 void add(int x,const T &v)
16 {
17     for (int i=x+1;i<=n;i+=i&-i)
18         a[i-1]=a[i-1]+v;
19 }
20
21 //查询区间 [0,x)
22 T sum(int x)
23 {
24     T ans{};
25     for (int i=x;i>0;i-=i&-i)
26         ans=ans+a[i-1];
27     return ans;
28 }
29
30 //查询区间 [l,r)
31 T rangeSum(int l,int r) { return sum(r)-sum(l); }
32
33 int select(const T &k)
34 {
35     int x=0;
36     T cur{};
37     for (int i=1<<__lg(n);i>=1)
38     {
39         if (x+i<=n&&cur+a[x+i-1]<=k)
40         {
41             x+=i;
42             cur=cur+a[x-1];
43         }
44     }
45     return x;
46 }
47 };

```

线段树

```

1  template <class Info,class Tag>
2  struct SGT
3  {
4      int n;
5      vector<Info> info;
6      vector<Tag> tag;
7
8      SGT():n(0) {}
9      SGT(int n_,Info v_=Info()) { init(n_,v_); }
10
11     template <class T>
12     SGT(vector<T> init_) { init(init_); }
13
14     void init(int n_,Info v_=Info()) { init(vector(n_,v_)); }
15
16     template <class T>
17     void init(vector<T> init_)
18     {
19         n=init_.size();
20         info.assign(4<<__lg(n),Info());
21         tag.assign(4<<__lg(n),Tag());
22         function<void(int,int,int)> build=[&](int p,int l,int r)
23         {
24             if (r-l==1)
25             {
26                 info[p]=init_[l];
27                 return;
28             }
29             int m=(l+r)>>1;
30             build(p<<1,l,m);
31             build(p<<1|1,m,r);

```



```

32         pushup(p);
33     };
34     build(1,0,n);
35 }
36
37 void pushup(int p) { info[p]=info[p<<1]+info[p<<1|1]; }
38
39 void apply(int p,const Tag &v)
40 {
41     info[p].apply(v);
42     tag[p].apply(v);
43 }
44
45 void pushdown(int p)
46 {
47     apply(p<<1,tag[p]);
48     apply(p<<1|1,tag[p]);
49     tag[p]=tag();
50 }
51
52 void modify(int p,int l,int r,int x,const Info &v)
53 {
54     if (r-l==1)
55     {
56         info[p]=v;
57         return;
58     }
59     int m=(l+r)>>1;
60     pushup(p);
61     if (x<m) modify(p<<1,l,m,x,v);
62     else modify(p<<1|1,m,r,x,v);
63 }
64
65 //O(log n) 单点修改
66 void modify(int p,const Info &v) { modify(1,0,n,p,v); }
67
68 Info rangeQuery(int p,int l,int r,int x,int y)
69 {
70     if (l>=y||r<=x) return Info();
71     if (l>=x&&r<=y) return info[p];
72     int m=(l+r)>>1;
73     pushdown(p);
74     return rangeQuery(p<<1,l,m,x,y)+rangeQuery(p<<1|1,m,r,x,y);
75 }
76
77 //O(log n) 区间查询 [l,r)
78 Info rangeQuery(int l,int r) { rangeQuery(1,0,n,l,r); }
79
80 void rangeApply(int p,int l,int r,int x,int y,const Tag &v)
81 {
82     if (l>=y||r<=x) return;
83     if (l>=x&&r<=y)
84     {
85         apply(p,v);
86         return;
87     }
88     int m=(l+r)>>1;
89     pushdown(p);
90     rangeApply(p<<1,l,m,x,y,v);
91     rangeApply(p<<1|1,m,r,x,y,v);
92     pushup(p);
93 }
94
95 //O(log n) 区间操作 [l,r)
96 void rangeApply(int l,int r,const Tag &v) { rangeApply(1,0,n,l,r,v); }
97
98 //O(log n) 区间 [l,r) 内查找第一个合法位置
99 template <class F>
100 int findFirst(int p,int l,int r,int x,int y,F pred)
101 {
102     if (l>=y||r<=x||!pred(info[p])) return -1;

```

```

103         if (r-l==1) return l;
104         int m=(l+r)>>1;
105         pushdown(p);
106         int res=findFirst(p<<1,l,m,x,y,pred);
107         if (res==-1) res=findFirst(p<<1|1,m,r,x,y,pred);
108         return res;
109     }
110
111     template <class F>
112     int findFirst(int l,int r,F pred) { return findFirst(1,0,n,l,r,pred); }
113
114     template <class F>
115     int findLast(int p,int l,int r,int x,int y,F pred)
116     {
117         if (l>=y||r<=x||!pred(info[p])) return -1;
118         if (r-l==1) return l;
119         int m=(l+r)>>1;
120         pushdown(p);
121         int res=findFirst(p<<1|1,m,r,x,y,pred);
122         if (res==-1) res=findFirst(p<<1,l,m,x,y,pred);
123         return res;
124     }
125
126     template <class F>
127     int findLast(int l,int r,F pred) { return findLast(1,0,n,l,r,pred); }
128 };
129
130 //这里默认乘法优先 (x*a+b)*c+d=x*(a*c)+(b*c+d)
131 struct Tag
132 {
133     i64 a=1,b=0;
134     void apply(Tag t)
135     {
136         a*=t.a;
137         b=b*t.a+t.b;
138     }
139 };
140
141 struct Info
142 {
143     i64 x=0,l=0,r=0;
144     void apply(Tag t)
145     {
146         int len=r-l+1;
147         x=x*t.a+len*t.b;
148     }
149 };
150
151 Info operator + (Info a,Info b)
152 {
153     return {a.x+b.x,min(a.l,b.l),max(a.r,b.r)};
154 }

```

字符串

字符串哈希（随机模数）

```

1 bool isPrime(int n)
2 {
3     if (n<=1) return 0;
4     for (int i=2;i*i<=n;i++)
5         if (n%i==0) return 0;
6     return 1;
7 }
8
9 int findPrime(int n)
10 {
11     while (!isPrime(n)) n++;
12     return n;
13 }

```

```

14 mt19937 rng(time(0));
15 const int P=findPrime(rng()%900000000+100000000);
16 struct StrHash
17 {
18     int n;
19     vector<int> h,p;
20
21     StrHash(const string &s){ init(s); }
22
23     void init(const string &s)
24     {
25         n=s.size();
26         h.resize(n+1);
27         p.resize(n+1);
28         p[0]=1;
29         for (int i=0;i<n;i++) h[i+1]=(10ll*h[i]+s[i]-'a')%P;
30         for (int i=0;i<n;i++) p[i+1]=10ll*p[i]%P;
31     }
32
33     //查询 [l,r) 的区间哈希
34     int get(int l,int r) { return (h[r]+1ll*(P-h[l])*p[r-l])%P; }
35 };
36

```

KMP

```

1 vector<int> KMP(const string &s)
2 {
3     int now=0;
4     vector<int> pre(s.size(),0);
5     for (int i=1;i<s.size();i++)
6     {
7         while (now&&s[i]!=s[now]) now=pre[now-1];
8         if (s[i]==s[now]) now++;
9         pre[i]=now;
10    }
11    return pre;
12 }

```

Z函数

```

1 vector<int> zFunction(string s)
2 {
3     int n=s.size();
4     vector<int> z(n);
5     z[0]=n;
6     for (int i=1,j=1;i<n;i++)
7     {
8         z[i]=max(0,min(j+z[j]-i,z[i-j]));
9         while (i+z[i]<n&&s[z[i]]==s[i+z[i]]) z[i]++;
10        if (i+z[i]>j+z[j]) j=i;
11    }
12    return z;
13 }

```

AC自动机

```

1 struct ACAM
2 {
3     static constexpr int ALPHABET=26;
4     struct Node
5     {
6         int len;
7         int link;
8         array<int,ALPHABET> next;
9         Node():len{0},link{0},next{{{}}}
10    };
11
12    vector<Node> t;

```

```

13
14 ACAM() { init(); }
15
16 void init()
17 {
18     t.assign(2,Node());
19     t[0].next.fill(1);
20     t[0].len=-1;
21 }
22
23 int newNode()
24 {
25     t.emplace_back();
26     return t.size()-1;
27 }
28
29 int add(const string &a)
30 {
31     int p=1;
32     for (auto c:a)
33     {
34         int x=c-'a';
35         if (t[p].next[x]==0)
36         {
37             t[p].next[x]=newNode();
38             t[t[p].next[x]].len=t[p].len+1;
39         }
40         p=t[p].next[x];
41     }
42     return p;
43 }
44
45 void work()
46 {
47     queue<int> q;
48     q.push(1);
49     while (!q.empty())
50     {
51         int x=q.front();
52         q.pop();
53         for (int i=0;i<ALPHABET;i++)
54         {
55             if (t[x].next[i]==0) t[x].next[i]=t[t[x].link].next[i];
56             else
57             {
58                 t[t[x].next[i]].link=t[t[x].link].next[i];
59                 q.push(t[x].next[i]);
60             }
61         }
62     }
63 }
64
65 int next(int p,int x) { return t[p].next[x]; }
66
67 int link(int p) { return t[p].link; }
68
69 int size() { return t.size(); }
70 };

```

后缀数组

```

1 struct SA
2 {
3     int n;
4     vector<int> sa,rk,lc;
5     SA(const string &s)
6     {
7         n=s.length();
8         sa.resize(n);
9         rk.resize(n);
10        lc.resize(n-1);

```

```

11     iota(sa.begin(),sa.end(),0);
12     sort(sa.begin(),sa.end(),[&](int a,int b){ return s[a]<s[b]; });
13     rk[sa[0]]=0;
14     for (int i=1;i<n;i++) rk[sa[i]]=rk[sa[i-1]]+(s[sa[i]]!=s[sa[i-1]]);
15     int k=1;
16     vector<int> tmp,cnt(n);
17     tmp.reserve(n);
18     while (rk[sa[n-1]]<n-1)
19     {
20         tmp.clear();
21         for (int i=0;i<k;i++) tmp.push_back(n-k+i);
22         for (auto i:sa)
23             if (i>=k) tmp.push_back(i-k);
24         fill(cnt.begin(),cnt.end(),0);
25         for (int i=0;i<n;i++) cnt[rk[i]]++;
26         for (int i=1;i<n;i++) cnt[i]+=cnt[i-1];
27         for (int i=n-1;i>=0;i--) sa[--cnt[rk[tmp[i]]]]=tmp[i];
28         swap(rk,tmp);
29         rk[sa[0]]=0;
30         for (int i=1;i<n;i++)
31             rk[sa[i]]=rk[sa[i-1]]+(tmp[sa[i-1]]<tmp[sa[i]] || sa[i-1]+k==n || tmp[sa[i-1]+k]<tmp[sa[i]+k]);
32         k<=1;
33     }
34     for (int i=0,j=0;i<n;i++)
35     {
36         if (rk[i]==0) j=0;
37         else
38         {
39             for (j-=j>0;i+j<n&&sa[rk[i]-1]+j<n&&s[i+j]==s[sa[rk[i]-1]+j]); j++;
40             lc[rk[i]-1]=j;
41             //lc[i]:lcp(sa[i],sa[i+1]),lcp(sa[i],sa[j])=min{lc[i...j-1]}
42         }
43     }
44 };

```

(广义) 后缀自动机

```

1  struct SAM
2  {
3      static constexpr int ALPHABET=26;
4      struct Node
5      {
6          int len;
7          int link;
8          array<int,ALPHABET> next;
9          Node():len{},link{},next{} {}
10     };
11
12     vector<Node> t;
13
14     SAM() { init(); }
15
16     void init()
17     {
18         t.assign(2,Node());
19         t[0].next.fill(1);
20         t[0].len=-1;
21     }
22
23     int newNode()
24     {
25         t.emplace_back();
26         return t.size()-1;
27     }
28
29     int extend(int lst,int c)
30     {
31         if (t[lst].next[c]&&t[t[lst].next[c]].len==t[lst].len+1)
32             return t[lst].next[c];
33         int p=lst,np=newNode(),flag=0;
34         t[np].len=t[p].len+1;

```

```

35     while (!t[p].next[c])
36     {
37         t[p].next[c]=np;
38         p=t[p].link;
39     }
40     if (!p)
41     {
42         t[np].link=1;
43         return np;
44     }
45     int q=t[p].next[c];
46     if (t[q].len==t[p].len+1)
47     {
48         t[np].link=q;
49         return np;
50     }
51     if (p==lst) flag=1,np=0,t.pop_back();
52     int nq=newNode();
53     t[nq].link=t[q].link;
54     t[nq].next=t[q].next;
55     t[nq].len=t[p].len+1;
56     t[q].link=t[np].link=nq;
57     while (p&& t[p].next[c]==q)
58     {
59         t[p].next[c]=nq;
60         p=t[p].link;
61     }
62     return flag?nq:np;
63 }
64
65 int add(const string &a)
66 {
67     int p=1;
68     for (auto c:a) p=extend(p,c-'a');
69     return p;
70 }
71
72 int next(int p,int x) { return t[p].next[x]; }
73
74 int link(int p) { return t[p].link; }
75
76 int len(int p) { return t[p].len; }
77
78 int size() { return t.size(); }
79 };

```

Manacher

```

1  vector<int> manacher(vector<int> s)
2  {
3      vector<int> t{0};
4      for (auto c:s)
5      {
6          t.push_back(c);
7          t.push_back(0);
8      }
9      int n=t.size();
10     vector<int> r(n);
11     for (int i=0,j=0;i<n;i++)
12     {
13         if (j*2-i>=0&&j+r[j]>i) r[i]=min(r[j*2-i],j+r[j]-i);
14         while (i-r[i]>=0&&i+r[i]<n&&t[i-r[i]]==t[i+r[i]]) r[i]++;
15         if (i+r[i]>j+r[j]) j=i;
16     }
17     return r;
18 }

```

回文自动机

```
1  struct PAM
2  {
3      static constexpr int ALPHABET_SIZE=28;
4      struct Node
5      {
6          int len,link,cnt;
7          array<int,ALPHABET_SIZE> next;
8          Node():len{},link{},cnt{},next{}{}
9      };
10     vector<Node> t;
11     int suff;
12     string s;
13
14     PAM() { init(); }
15
16     void init()
17     {
18         t.assign(2,Node());
19         t[0].len=-1;
20         suff=1;
21         s.clear();
22     }
23
24     int newNode()
25     {
26         t.emplace_back();
27         return t.size()-1;
28     }
29
30     bool add(char c,char offset='a')
31     {
32         int pos=s.size();
33         s+=c;
34         int let=c-offset;
35         int cur=suff,curlen=0;
36         while (1)
37         {
38             curlen=t[cur].len;
39             if (pos-curlen-1>=0&&s[pos-curlen-1]==s[pos]) break;
40             cur=t[cur].link;
41         }
42         if (t[cur].next[let])
43         {
44             suff=t[cur].next[let];
45             return 0;
46         }
47         int num=newNode();
48         suff=num;
49         t[num].len=t[cur].len+2;
50         t[cur].next[let]=num;
51         if (t[num].len==1)
52         {
53             t[num].link=t[num].cnt=1;
54             return 1;
55         }
56         while (1)
57         {
58             cur=t[cur].link;
59             curlen=t[cur].len;
60             if (pos-curlen-1>=0&&s[pos-curlen-1]==s[pos])
61             {
62                 t[num].link=t[cur].next[let];
63                 break;
64             }
65         }
66         t[num].cnt=t[t[num].link].cnt+1;
67         return 1;
68     }
69 };
```