ACM 算法爆哥 & Buns_out 合订模板

无敌爆哥 & Buns_out

2023 年 11 月 10 日

目录

1 爆哥滴

1.1 拓展卢卡斯

mod 不需要为质数求阶乘部分可以适当的用记忆化

```
ll fac(ll n,int p,int mod){
    if(!n) return 1;
    ll res=1;
    for(int i=2;i<=mod;i++){
        if(i%p) res=res*i%mod;
    }
    res=ksm(res,n/mod,mod);
    for(int i=2;i<=n%mod;i++){
        if(i%p) res=res*i%mod;
    }
    return res*fac(n/p,p,mod)%mod;
}
ll Inv(ll a,ll mod){
    ll x,y;
    exgcd(a,mod,x,y);
    return (x+mod)%mod;
}
ll C(ll n,ll m,int p,int mod){
    ll d=fac(n,p,mod),d1=fac(m,p,mod),d2=fac(n-m,p,mod);
    ll k=0;
    for(ll i=n;i;i/=p){
        k+=i/p;
    }
    for(ll i=m;i;i/=p){
        k-=i/p;
    }
    for(ll i=n-m;i;i/=p){
        k-=i/p;
    }
    return d*Inv(d1,mod)%mod*Inv(d2,mod)%mod*ksm(p,k,mod)%
       mod;
}
```

```
ll exlucas(ll n,ll m,int p){
    if(n<0||m<0||n<m) return 0;
    int x=p;
    tp=0;
    for(int i=2;1ll*i*i<=x;i++){
        if(x%i==0){
            int t=1;
            while(x%i==0){
                x/=i;t*=i;
            }
            ++tp;
            r[tp]=C(n,m,i,t);
            md[tp]=t;
        }
    }
    if(x>1) {
        ++tp;
        r[tp]=C(n,m,x,x),md[tp]=x;
    }
    return CRT();
}
```

1.2 复数

```
1  struct complex_{
2      double a,b;
3      complex_(){a=0;b=0;}
4      complex_(double _a,double _b){a=_a;b=_b;}
5      complex_ operator +(const complex_ &w)const{
6          return complex_(a+w.a,b+w.b);
7      }
8      complex_ operator -(const complex_ &w)const{
9          return complex_(a-w.a,b-w.b);
10     }
11     complex_ operator *(const complex_ &w)const{
12         return complex_(a*w.a-b*w.b,a*w.b+b*w.a);
13     }
14     complex_ operator /(const complex_ &w)const{
15         return complex_(a*w.a+b*w.b,b*w.a-a*w.b)/(w.a*w.a+
               w.b*w.b);
16     }
17     complex_ operator /(const double w){
18         return complex_(a/w,b/w);
19     }
20 };
```

1.3 数据范围对应最大因子与质因子

n<=	10^1	10^2	10^3	10^4	10^5	10^6	10^7	10^8	10^9
$max\{w(n)\}$	2	3	4	5	6	7	8	8	9
$max\{d(n)\}$	4	12	32	64	128	240	448	768	1344
n<=	10^{11}	10^{13}	10^{12}	10^{13}	10^{14}	10^{15}	10^{16}	10^{17}	10^{18}
$max\{w(n)\}$	10	10	11	12	12	13	13	14	15
$max\{d(n)\}$	2304	4032	6720	10752	17280	26880	41472	64512	103680

1.4 三元环计数

```
for(int i=1;i<=m;i++){
        in[p[i].v]++;
        in[p[i].u]++;
    }
    for(int i=1;i<=m;i++){
        int x=p[i].v,y=p[i].u;
        if(in[x]>in[y]) swap(x,y);
        else if(in[x]==in[y]&&x>y) swap(x,y);
        v[x].push_back(y);
    }
    int ans=0;
    for(int i=1;i<=n;i++){
        for(int k:v[i]) up[k]=i;
        for(int j:v[i]){
            for(int k:v[j]){
                ans+=(up[k]==i);
            }
        }
    }
```

2 Buns_out 滴

2.1 Lyndon 分解

```cpp
/*
将S分成若干个部分S=S1S2S3..Sm 每个串都是 Lyndon Word
一个字符串 s 是一个 Lyndon Word 表示 s 是其所有后缀中的最小者。
*/
vector<string> duval(string const &s) {
    int n = s.size(), i = 0;
    vector<string> fac;
    while (i < n) {
        int j = i + 1, k = i;
        while (j < n && s[k] <= s[j]) {
            if (s[k] < s[j]) k = i;
            else k++;
            j++;
        }
        while (i <= k) {
            fac.push_back(s.substr(i, j - k));
            i += j - k;
        }
    }
    return fac;
}
// duval_algorithm
vector<int> duval2(string const &s) {
    int n = s.size(), i = 0;
    vector<int> fac;
    while (i < n) {
        int j = i + 1, k = i;
        while (j < n && s[k] <= s[j]) {
            if (s[k] < s[j])k = i;
            else k++;
            j++;
        }
```

```
33          while (i <= k) {
34              i += j - k;
35              fac.push_back(i);
36          }
37      }
38      return fac;
39  }
40  void solve()
41  {
42      string s;
43      cin >> s;
44      auto v = duval2(s);
45
46      for (auto i : v) {
47          cout << i << "␣";
48      }
49  }
```

2.2 Linux 对拍

```
int main(){
    while(1){
        printf("The␣result␣of␣No.␣%d␣Case␣is:␣␣",++cases);
        system("./data");
        system("./BF");
        system("./test");
        if(system("diff␣BF.out␣test.out")){
            puts("Wrong␣Answer\n");
            return 0;
        }
        puts("Accepted");
    }
    return 0;
}
```

2.3 在线查询回文子串数量

```
1  /*
2  1：在字符串 s 的末尾添加一个字符串
3  2：在字符串 s 的前端添加一个字符串的 反序
4  3：查询字符串 s 的所有非空回文子串的数量
5  */
6
7
8  struct PAM {
9      char s[maxn];
10     int ptrf, ptrb;
11     int node, len[maxn + 5], fail[maxn + 5], ch[maxn +
           5][26];
12     int rlas, llas, dep[maxn + 5];
13
14     PAM() { node = rlas = llas = 1, len[1] = -1, fail[0] =
           1; }
15     void set(int _x,int _y){
16         ptrf=_x;
17         ptrb=_y;
18     }
19
20     inline int push_front( char c ) {
21         s[--ptrf] = c, c -= 'a'; int p = llas;
22         for ( ; s[ptrf + len[p] + 1] != s[ptrf]; p = fail[
               p] );
23         if ( !ch[p][c] ) {
24             len[++node] = len[p] + 2; int q = fail[p];
25             for ( ; s[ptrf + len[q] + 1] != s[ptrf]; q =
                   fail[q] );
26             dep[node] = dep[fail[node] = ch[q][c]] + 1, ch
                   [p][c] = node;
27         }
28         llas = ch[p][c];
29         if ( len[llas] == ptrb - ptrf + 1 ) rlas = llas;
30         return dep[llas];
```

```
31      }
32
33      inline int push_back( char c ) {
34          s[++ptrb] = c, c -= 'a'; int p = rlas;
35          for ( ; s[ptrb - len[p] - 1] != s[ptrb]; p = fail[
                p] );
36          if ( !ch[p][c] ) {
37              len[++node] = len[p] + 2; int q = fail[p];
38              for ( ; s[ptrb - len[q] - 1] != s[ptrb]; q =
                    fail[q] );
39              dep[node] = dep[fail[node] = ch[q][c]] + 1, ch
                    [p][c] = node;
40          }
41          rlas = ch[p][c];
42          if ( len[rlas] == ptrb - ptrf + 1 ) llas = rlas;
43          return dep[rlas];
44      }
45  }pam;
46  string str;
47  ll ans=0;
48  int q;
49  void solve()
50  {
51      pam.set(3e5+1,3e5);
52      cin>>str;
53      for(auto i:str)
54          ans+=pam.push_back(i);
55      cin>>q;
56      while(q--)
57      {
58          int op;
59          cin>>op;
60          if(op==1)
61          {
62              cin>>str;
63              for(auto i:str)
64                  ans+=pam.push_back(i);
```

```
65          }
66          else if(op==2)
67          {
68              cin>>str;
69              for(auto i:str)
70                  ans+=pam.push_front(i);
71          }
72          else cout<<ans<<endl;
73      }
74  }
```