

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

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
111     friend constexpr bool operator == (MInt lhs, MInt rhs) { return lhs.val() == rhs.val(); }
112
113     friend constexpr bool operator != (MInt lhs, MInt rhs) { return lhs.val() != rhs.val(); }
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();

```