GCC扩展

builtin系列

`__builtin_clz` 返回高位 `0` 个数。

`__builtin_ctz` 返回低位 `0` 个数。

`__builtin_popcount` 返回 `1` 个数。

注：类型为 `long long` 时需要在后面加 `ll`。

bitset

`_Find_first` 返回第一个 `1` 的下标。

`_Find_next(p)` 返回第 `p` 位之后第个 `1` 的下标。

```
#define ffirst(s) (s)._Find_first()
#define fnext(s, i) (s)._Find_next(i)
#define for_(i, n, s) for (int i = ffirst((s)); i < n; i = fnext((s), i))
```

具体用法可参考图论-最大独立集。

PBDS

优先队列

```
//  Paring heap
#include <ext/pb_ds/priority_queue.hpp>
typedef __gnu_pbds::priority_queue<int, less<int>> heap;
typedef heap::point_iterator iter;
//  Example:
heap h, h2;
iter it = h.push(2);// h.top() == 2;
h.push(1);          // h.top() == 1;
h.modify(it, 0);    // h.top() == 0;
h.join(h2);
```

红黑树

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
typedef tree<int, null_type, less<int>, rb_tree_tag,
tree_order_statistics_node_update> rbt;
```

`order_of_key` 返回小于 `v` 的元素个数。

`find_by_order` 返回第 `i` 个元素，下标从 `0` 开始。

不支持多重元素，请考虑使用 `pair<int,int>` 之类的东西。

哈希表

```cpp
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
typedef gp_hash_table<int, int, myhash, myequ> hashtable;
typedef cc_hash_table<int, int, myhash, myequ> hashtable;
```

分治

三维偏序

第一维排序后分治，每次统计前一半对后一半的贡献，就变成了一个二维偏序问题。

```
int a[N], b[N], c[N], p[N];

ll solve(int l, int r) {
    if (l + 1 == r) return 0;
    int m = (l + r) >> 1;
    ll res = solve(l, m) + solve(m, r);
    inplace_merge(p + l, p + m, p + r, [&](int i, int j) { return b[i] < b[j];
});
    for (int i = l; i < r; ++i) {
        int u = p[i];
        if (a[u] < m) BIT::modify(c[u], 1);
        else res += BIT::query(c[u]);
    }
    for (int i = l; i < r; ++i) {
        int u = p[i];
        if (a[u] < m) BIT::modify(c[u], -1);
    }
    return res;
}

ll solve(int n) {
    iota(p + 1, p + n + 1, 1);
    sort(p + 1, p + n + 1, [&](int i, int j) { return a[i] < a[j]; });
    return solve(1, n + 1);
}
```

注：代码中未考虑等号！

最小平均圈

O(nm)

```
typedef double dbl;
struct edge { int v; dbl w; };
vector<edge> g[N];

dbl dp[N][N];
dbl minmwc(int n) {
    for (int i = 1; i <= n; ++i)
        g[n + 1].push_back({ i, 0 });
    n++;
    for (int i = 1; i <= n; ++i)
        dp[i][0] = DBL_MAX;
    dp[n][0] = 0;
    for (int k = 0; k < n; ++k) {
        for (int u = 0; u <= n; ++u)
            dp[u][k + 1] = DBL_MAX;
        for (int u = 0; u <= n; ++u)
            if (dp[u][k] != DBL_MAX)
                for (edge e : g[u])
                    dp[e.v][k + 1] = min(dp[e.v][k + 1], dp[u][k] + e.w);
    }
    dbl ans = DBL_MAX;
    for (int u = 1; u <= n; ++u) {
        if (dp[u][n] == DBL_MAX) continue;
        dbl res = -DBL_MAX;
        for (int k = 0; k < n; ++k)
            if (dp[u][k] != DBL_MAX)
                res = max(res, (dp[u][n] - dp[u][k]) / (n - k));
        ans = min(ans, res);
    }
    return ans;
}
```

染色数

```
#define W 23
#define N 1<<23
typedef bitset<W> bs;
typedef long long ll;
bs g[W];
int w[N]; ll a[N], b[N];

int chromatic_number(int n) {
    fill_n(a, 1 << n, 1); fill_n(b, 1 << n, 1);
    for (int i = 0; i != (1 << n); ++i)
        w[i] = w[i >> 1] + (i & 1);
    for (int i = 0; i != (1 << n); ++i)
        for (int j = 0; j != n; ++j)
            if ((i & (1 << j)) && (g[j] & bs(i)).any())
                a[i] = 0;
    a[0] = 0;
    for (int i = 0; i != n; ++i)
        for (int j = 0; j != (1 << n); ++j)
            if (j & (1 << i)) a[j] += a[j ^ (1 << i)];
    int ans = 0;
    for (int k = 1; !ans; ++k) {
        for (int i = 0; i != (1 << n); ++i) b[i] *= a[i];
        ll s = 0;
        for (int i = 0; i != (1 << n); ++i)
            s += (w[i] & 1) ? -b[i] : b[i];
        if (s) ans = k;
    }
    return ans;
}
```

高斯消元与逆矩阵

逆矩阵

模质数

```
mat inv(mat a) {
    int n = a.size();
    for (int i = 0; i < n; ++i) {
        a[i].resize(2 * n, 0);
        a[i][n + i] = 1;
    }
    for (int i = 0; i < n; ++i) {
        for (int j = i + 1; j < n && !a[i][i]; ++j)
            if (a[j][i]) swap(a[i], a[j]);
        if (!a[i][i]) return {};
        for (int j = i, t = inv(a[i][i]); j < 2 * n; ++j)
            a[i][j] = mul(a[i][j], t);
        for (int j = 0; j < n; ++j) if(j != i)
            for (int k = i, t = a[j][i]; k < 2 * n; ++k)
                a[j][k] = sub(a[j][k], mul(t, a[i][k]));
    }
    for (int i = 0; i < n; ++i) {
```

```
        copy_n(a[i].begin() + n, n, a[i].begin());
        a[i].resize(n);
    }
    return a;
}
```

实数

```
mat inv(mat a) {
    const int n = a.size();
    for (int i = 0; i < n; ++i) {
        a[i].resize(2 * n, 0);
        a[i][n + i] = 1;
    }
    for (int i = 0; i < n; ++i) {
        for (int j = i + 1; j < n; ++j)
            if (abs(a[j][i]) > abs(a[i][i]))
                swap(a[i], a[j]);
        if (!sgn(a[i][i])) return {};
        for (int k = 2 * n - 1; k >= i; --k)
            a[i][k] /= a[i][i];
        for (int j = 0; j < n; ++j)
            for (int k = 2 * n - 1; j != i && k >= i; --k)
                a[j][k] -= a[j][i] * a[i][k];
    }
    for (int i = 0; i < n; ++i) {
        copy_n(a[i].begin() + n, n, a[i].begin());
        a[i].resize(n);
    }
    return a;
}
```

求伴随阵

```
int rk(mat a, int& f) {
    const int n = a.size(), m = a[0].size();
    int i = 0;
    for (int j = 0; i != n && j != m; j++) {
        for (int k = j + 1; !a[i][j] && k < n; ++k)
            if (a[k][j]) swap(a[k], a[j]);
        if (!a[i][j]) { f = i; continue; }
        for (int k = i + 1, t = inv(a[i][j]); k < n; ++k)
            for (int l = j, w = mul(a[k][j], t); l < m; ++l)
                a[k][l] = sub(a[k][l], mul(w, a[i][l]));
        i++;
    }
    return i;
}

bool adj0(mat& b) {
    const int n = b.size(); int d = 1;
    mat a(n, vec(2 * n, 0));
    for (int i = 0; i < n; ++i) {
        copy_n(b[i].begin(), n, a[i].begin());
        a[i][n + i] = 1;
    }
    for (int i = 0; i < n; ++i) {
        for (int j = i + 1; j < n && !a[i][i]; ++j)
            if (a[j][i]) swap(a[i], a[j]), d = sub(0, d);
        d = mul(d, a[i][i]);
        for (int j = i, t = inv(a[i][i]); j < 2 * n; ++j)
            a[i][j] = mul(a[i][j], t);
        for (int j = 0; j < n; ++j) if(j != i)
            for (int k = i, t = a[j][i]; k < 2 * n; ++k)
                a[j][k] = sub(a[j][k], mul(t, a[i][k]));
    }
    if (!d) return false;
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            b[i][j] = mul(a[i][n + j], d);
    return true;
}

mat adj(mat a) {
    int n = a.size(), fi, fj;
    if (n == 1)
        return mat(1, vec(1, a[0][0] ? 1 : 0));
    int r = rk(a, fj);
    if (r == n) { adj0(a); return a; }
    if (r <= n - 2) return mat(n, vec(n, 0));
    uniform_int_distribution<int> uid(0, P - 1);
    mt19937_64 mt(chrono::system_clock::now().time_since_epoch().count());
    mat b(n, vec(n, 0)), c = a;
    do for (int i = 0; i < n; ++i) c[i][fj] = uid(mt);
    while (!adj0(c));
    for (int i = 0; i < n; ++i) b[fj][i] = c[fj][i];
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
```

```
            c[i][j] = a[j][i];
    rk(c, fi);
    do for (int j = 0; j < n; ++j) c[j][fi] = uid(mt);
    while (!adj0(c));
    for (int j = 0; j < n; ++j) b[j][fi] = c[fi][j];
    int w = inv(b[fj][fi]);
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            if (i != fj && j != fi)
                b[i][j] = mul(mul(b[fj][j], b[i][fi]), w);
    return b;
}
```

行列式

定义:

$$\det A = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{i\sigma_i}$$

模质数

高斯消元，$O(n^3)$。

```
typedef vector<int> vec;
typedef vector<vec> mat;

int det(mat a) {
    const int n = a.size(); int res = 1;
    for (int i = 0; i != n; ++i) {
        for (int j = i + 1; j != n && !a[i][i]; ++j)
            if (a[j][i]) swap(a[i], a[j]), res = sub(0, res);
        if (!a[i][i]) return 0; else res = mul(res, a[i][i]);
        for (int j = i + 1, t = inv(a[i][i]); j != n; ++j)
            for (int k = i, w = mul(a[j][i], t); k != n; ++k)
                a[j][k] = sub(a[j][k], mul(w, a[i][k]));
    }
    return res;
}
```

模合数

高斯消元+辗转相除，$O(n^3 \log U)$。其中U是模数。

```
typedef vector<int> vec;
typedef vector<vec> mat;

int det(mat a) {
    const int n = a.size(); int res = 1, f = 0;
    for (int i = 0; i != n; ++i) {
        for (int j = i + 1; j != n && !a[i][i]; ++j)
            if (a[j][i]) swap(a[i], a[j]), f ^= 1;
        if (!a[i][i]) return 0;
        for (int j = i + 1; j != n; swap(a[i], a[j]), f ^= 1, ++j)
            for (; a[i][i]; swap(a[i], a[j]), f ^= 1)
                for (int k = i, q = a[j][i] / a[i][i]; k != n; ++k)
                    a[j][k] = sub(a[j][k], mul(q, a[i][k]));
        res = mul(res, a[i][i]);
    }
    return f ? sub(0, res) : res;
}
```

实数

高斯消元，O(n^3)

```
typedef double dbl;
const dbl eps = 1e-10;
```

```cpp
int sgn(dbl f) { return f < -eps ? -1 : f > eps; }

typedef vector<dbl> vec;
typedef vector<vec> mat;

dbl det(mat a) {
    const int n = a.size(); dbl res = 1;
    for (int i = 0; i != n; ++i) {
        for (int j = i + 1; j != n; ++j)
            if (abs(a[j][i]) > abs(a[i][i]))
                swap(a[i], a[j]), res = -res;
        if (!sgn(a[i][i])) return 0; else res *= a[i][i];
        for (int j = i + 1; j != n; ++j) {
            if (!sgn(a[j][i])) continue;
            double w = a[j][i] / a[i][i];
            for (int k = i; k != n; ++k)
                a[j][k] -= w * a[i][k];
        }
    }
    return res;
}
```

求范德华矩阵逆

```
mat inverse_of_vandermonde(vec x) {
    const int n = x.size();
    vec c(n + 1, 0); c[0] = 1;
    for (int i = 1; i <= n; ++i)
        for (int j = i; j >= 1; --j)
            c[j] = add(c[j], mul(x[i - 1], c[j - 1]));
    mat a(n, c);
    for (int i = 0; i < n; ++i)
        for (int j = 1; j <= n; ++j)
            a[i][j] = sub(a[i][j], mul(x[i], a[i][j - 1]));
    vec b = x;
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            if (j != i) b[i] = mul(b[i], sub(x[j], x[i]));
    for (int i = 0; i < n; ++i)
        b[i] = inv(b[i]);
    mat r(n, vec(n, 0));
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            r[i][j] = (j & 1 ? sub : add)(0, mul(b[i], a[i][n - j - 1]));
    return r;
}
```

BM算法

问题描述

给一个序列$\{a_0, a_1, \ldots, a_n\}$

要求最短的序列$\{f_0, f_1, \ldots, f_m\}$

使得对于所有$i > m$有

$$a_i = \sum_{k=0}^{m} f_k * a_{i-1-k}$$

算法流程

主要思想是依次考虑每个a_i

不断修改$\{f\}$

使得其在保证正确的同时尽量短

一开始$\{f\}$为空

对每个a_i判断当前递推式是否满足条件

如果满足就直接判断下一个

否则需要修改

如果当前$\{f\}$为空

说明a_i之前数全是0

而$a_i \neq 0$

所以是不可能用之前的数推出a_i的

这种情况下直接把$f_0, f_1, \ldots f_i$记为0

这样$i \leq m$就不用推了

否则说明之前已经有一个错误的$\{f\}$

为了方便记为$\{F\}$

我们希望能通过$\{F\}$在i位推出一个不为0的值

然后把这次的错误抵消掉

如果$\{F\}$出错的位置是p且多了Δ

这个时候a_p一定不等于$\sum_{k=0}^{m} F_k * a_{i-k}$

就可以对$\{F\}$稍作修改

在a_i的位置上递推出一个$-\Delta$

具体来说

把$\{F\}$全部变为相反数再在最前面补1

得到的新的$\{F\}$可以满足在$p+1$位置推出一个$-\Delta$来

再在$\{F\}$最前面补$i-p-1$个0

$-\Delta$就跑到i位置来了

把现在得到的$\{F\}$除以Δ再乘上这次的差

加上$\{f\}$就可以把这次的差抵消掉

因为要求递推式最短

我们希望每次能得到最优的$\{F\}$

考虑每次修改时$\{f\}$的长度变化

变成了$max(len(f),len(F)+i-p)$

所以只要记录$len(F)-p$最短的$\{F\}$即可

还有Berlekamp-Massey返回的递推式的长度最好要在输入的一半以内

不然还是再多打点表吧

为什么?

感谢LHJ神仙指教

这样多出来的一半就可以列出至少$len(F)$个方程组

确定了递推式的唯一性

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
const int p=1e9+7;
inline int add(int a,int b){return (a+=b)>=p?a-p:a;}
inline int sub(int a,int b){return (a-=b)<0?a+p:a;}
inline ll qpow(ll a,ll b){
    ll ans=1;
    for(;b;b>>=1,(a*=a)%=p)if(b&1)(ans*=a)%=p;
    return ans;
}
namespace BerlekampMassey{
    typedef vector<int> poly;
    #define len(A) A.size()
    inline poly BM(const poly& A){
        poly F,F0;
        int d0,p0;
```

```
        for(int i=0;i<len(A);++i){
            int d=0;
            for(int j=0;j<len(F);++j){
                d=add(d,(ll)F[j]*A[i-j-1]%p);
            }
            d=sub(d,A[i]);
            if(!d)continue;
            if(!len(F)){
                F.resize(i+1);
                d0=d;
                p0=i;
                continue;
            }
            ll t=qpow(d0,p-2)*d%p;
            poly G(i-p0-1);
            G.push_back(t);
            t=sub(0,t);
            for(int j=0;j<len(F0);++j){
                G.push_back(F0[j]*t%p);
            }
            if(len(G)<len(F))G.resize(len(F));
            for(int j=0;j<len(F);++j){
                G[j]=add(G[j],F[j]);
            }
            if(i-p0+len(F0)>=len(F)){
                //注意这里不要把i移项,vector的size是unsigned类型,减成负的就凉了
                F0=F;
                d0=d;
                p0=i;
            }
            F=G;
        }
        return F;
    }
}
using namespace BerlekampMassey;
int F[]={1,2,4,9,20,40,90};
int main(){
    poly G(F,F+((sizeof F)>>2));
    G=BM(G);
    printf("%d\n",G.size());
    for(int i=0;i<G.size();++i)printf("%d ",G[i]);
}
```