Team Reference Document

1. Расширенный алгоритм Евклида
2. Поиск мостов
3. Поиск точек сочленения
4. Топологическая сортировка
5. Компоненты сильной связности
6. Двоичные прыжки
7. LCA
8. Дерево отрезков
9. Дерево Фенвика с двоичным спуском
10. Факторизация с предподсчётом мин. делителей за log(n)
11. Обратные факториалы, биномиальные коэффициенты
12. Быстрое возведение в степень
13. __int128
14. ordered_set
15. Геометрия (Джарвис + вывести объекты на периметре)
16. Хэши
17. Матрицы и их возведение в степень
18. Прогрессии

1) Расширенный евклид: решает a*x+b*y=gcd(a,b) в целых числах. Не проверяет, существует ли решение, не принимает правую часть. emaxx говорит, что работает для отрицательных чисел тоже.

```
int gcd (int a, int b, int & x, int & y) {
	if (a == 0) {
		x = 0; y = 1;
		return b;
	}
	int x1, y1;
	int d = gcd (b%a, a, x1, y1);
	x = y1 - (b / a) * x1;
	y = x1;
	return d;
}
```

2) Поиск мостов: Работает с петлями и кратными ребрами.

```
const int MAXN = КОЛИЧЕСТВО ВЕРШИН;
```

```
bool used[MAXN];
int timer, tin[MAXN], fup[MAXN];
vector<int> adj[MAXN];

void solve() {
   int n, m; cin >> n >> m;
   vb isBridge(1+m);
   map<pair<int, int>, int> mp;
   map<pair<int, int>, int> cnt; // только, если есть кратные ребра
   for(int i = 1; i <= m; ++i) {
      int x, y; cin >> x >> y;
      if(x == y) continue;
      adj[x].push_back(y);
      adj[y].push_back(x);
      if(x > y) swap(x, y);
      mp[{x, y}] = i;
      cnt[{x, y}]++;
   }
   function<void(int, int)> dfs = [&](int v, int p) {
      used[v] = true;
      tin[v] = fup[v] = timer++;
      for(auto u : adj[v]) {
         int to = u;
         if(to == p) continue;
         if(used[to]) {
            fup[v] = min(fup[v], tin[to]);
         }
         else {
            dfs(to, v);
            fup[v] = min(fup[v], fup[to]);
            if(fup[to] > tin[v]) {
               int x = to, y = v;
               if(x > y) swap(x, y);
               if(cnt[{x, y}] == 1) {
                  isBridge[mp[{x, y}]]=true;
               }
            }
         }
```

```
      }
   };
   timer = 1;
   for(int i = 1; i <= n; ++i) {
      used[i] = false;
   }
   for(int i = 1; i <= n; ++i) {
      if(!used[i]) dfs(i, -1);
   }
   for(auto v : isBridge) cout << v << " "; cout << endl;
}
```

3) Поиск точек сочленения.

```
const int MAXN = КОЛИЧЕСТВО ВЕРШИН;
bool used[MAXN];
int timer, tin[MAXN], fup[MAXN];
bool IS_CUTPOINT[MAXN];
vector<int> adj[MAXN];

void solve() {
   int n, m; cin >> n >> m;
   for(int i = 0; i < m; ++i) {
      int x, y; cin >> x >> y;
      adj[x].push_back(y);
      adj[y].push_back(x);
   }
   function<void(int, int)> dfs = [&](int v, int p) {
      used[v] = true;
      tin[v] = fup[v] = timer++;
      int children = 0;
      for(auto u : adj[v]) {
         if(u == p) continue;
         if(used[u]) {
            fup[v] = min(fup[v], tin[u]);
         }
         else {
            dfs(u, v);
            fup[v] = min(fup[v], fup[u]);
```

```
            if(fup[u] >= tin[v] && p != -1) {
               IS_CUTPOINT[v] = true;
            }
            ++children;
         }
      }
      if(p == -1 && children > 1) {
         IS_CUTPOINT[v] = true;
      }
   };
   dfs(1, -1);
   for(int i = 1; i <= n; ++i) cout << IS_CUTPOINT[i] << " "; cout << endl;
}
```

4) Топологическая сортировка

```
const int MAXN = КОЛИЧЕСТВО ВЕРШИН;
vector<int> adj[MAXN];
int cnt[MAXN];

void solve() {
   int n, m; cin >> n >> m;
   for(int i = 0; i < m; ++i) {
      int x, y; cin >> x >> y;
      adj[x].push_back(y);
      cnt[y]++;
   }
   queue<int> q;
   for(int i = 1; i <= n; ++i) {
      if(cnt[i]==0) q.push(i);
   }
   vi ans;
   function<void(int)> dfs = [&](int v) {
      ans.push_back(v);
      for(auto u : adj[v]) {
         cnt[u]--;
         if(cnt[u] == 0) {
            q.push(u);
         }
      }
```

```
    };
    while(!q.empty()) {
        int v = q.front();
        q.pop();
        dfs(v);
    }
    if(ans.size() == n) {
        for(auto v : ans) cout << v << " "; cout << endl;
    }
    else { // Вывести -1, если нельзя сделать топологическую сортировку
        cout << -1 << endl;
    }
}
```

5) Компоненты сильной связности (ребра в adj_scc добавляются повторно, так что я использую set).

```
vector<vector<int>> adj, radj;
vector<bool> used;
vector<int> order, component, roots, root_nodes;

void dfs1(int v) {
    used[v] = true;
    for(auto u : adj[v]) {
        if(!used[u]) {
            dfs1(u);
        }
    }
    order.push_back(v);
}

void dfs2(int v) {
    used[v] = true;
    component.push_back(v);
    for(auto u : radj[v]) {
        if(!used[u]) {
            dfs2(u);
        }
    }
}
```

```
void solve() {
   int n, m; cin >> n >> m;
   adj.resize(1+n);
   radj.resize(1+n);
   used.resize(1+n);
   roots.resize(1+n);
   for(int i = 0; i < m; ++i) {
      int x, y; cin >> x >> y;
      if(x == y) continue;
      adj[x].push_back(y);
      radj[y].push_back(x);
   }
   for(int i = 1; i <= n; ++i) {
      if(!used[i]) dfs1(i);
   }
   vector<bool>(1+n).swap(used); // очистить все значения на false
   reverse(order.begin(), order.end());
   for(auto v : order) {
      if(!used[v]) {
         dfs2(v);
         int root = component.front();
         for(auto u : component) roots[u] = root;
         root_nodes.push_back(root);
         component.clear();
      }
   }
   vector<set<int>> adj_scc(1+n);
   for(int v = 1; v <= n; ++v) {
      for(auto u : adj[v]) {
         int root_v = roots[v];
         int root_u = roots[u];
         if(root_v!=root_u) {
            adj_scc[root_v].insert(root_u);
            adj_scc[root_u].insert(root_v);
         }
      }
   }
```

```
  for(auto v : root_nodes) {
    cout << v << ": ";
    for(auto u : adj_scc[v]) cout << u << " "; cout << endl;
  }
}
```

6) Двоичные прыжки

```
vi nxt;
const int MAX_STEPS = 1e9+10;

class BinaryJumps {
private:
        vvi jumps;
        int log_2;

public:
        BinaryJumps(ll max_steps) {
                log_2 = 0;
                while ((1LL << log_2) <= max_steps) ++log_2;

                int n = nxt.size();
                jumps.assign(log_2, vector<int>(n, 0));

      // Устанавливаем первый прыжок
                for (int v = 0; v < n; ++v) {
                        jumps[0][v] = nxt[v];
                }

                for (int b = 0; b + 1 < log_2; ++b) {
                        for (int v = 0; v < n; ++v) {
           for(int t = 0; t <= 1; ++t) {
                                int p = jumps[b][v];
                                jumps[b + 1][v] = jumps[b][p];
           }
                        }
                }
        }

        int jump(int v, ll steps) {
```

```
            for (int b = log_2; b >= 0; --b) {
                ll cur_step = (1LL << b);
                if (steps >= cur_step) {
                    steps -= cur_step;
                    v = jumps[b][v];
                }
            }
            return v;
        }
};

void solve() {
   int n; cin >> n;
   vi a(n);
   for(auto &v : a) cin >> v;
   nxt.resize(n);
   for(int i = 0; i < n; ++i) {
      int x; cin >> x;
      nxt[i] = x;
   }
   BinaryJumps bj(MAX_STEPS);
   int Q; cin >> Q;
   while(Q--) {
      int v, len; cin >> v >> len;
      int ans = bj.jump(v, len);
      cout << ans << endl;
   }
}
```

7) LCA

```
const int N = 1e5+10;
const int L = 20;

vector<int> adj[N];
int up[N][L], d[N];

void dfs(int v) {
   for(int i = 1; i < L; ++i) {
      up[v][i] = up[up[v][i-1]][i-1];
```

```
    }
    for(auto u : adj[v]) {
        if(u != up[v][0]) {
            up[u][0] = v;
            d[u] = d[v]+1;
            dfs(u);
        }
    }
}

int lca(int x, int y) {
    if(d[x] < d[y]) swap(x, y);
    int t = d[x]-d[y];
    for(int i = 0; i < L; ++i) {
        if(t&(1<<i)) x = up[x][i];
    }
    if(x == y) return x;
    for(int i = L-1; i >= 0; i--) {
        if(up[x][i]!=up[y][i]) {
            x = up[x][i];
            y = up[y][i];
        }
    }
    return up[x][0];
}

void solve() {
    int n, m; cin >> n >> m;
    for(int i = 0; i < m; ++i) {
        int x, y; cin >> x >> y;
        adj[x].push_back(y);
        adj[y].push_back(x);
    }
    dfs(1);
    int Q; cin >> Q;
    while(Q--) {
        int x, y; cin >> x >> y;
        cout << lca(x, y) << endl;
```

```
    }
}
```

8) Дерево отрезков (код с C. Trip to Saint Petersburg) с ленивым проталкиванием

```
typedef long long ll;

const int N = 5e5+10;
const int MAXCOR = 2e5;

vector<pair<int, ll>> rev[N];

int lazy[10*N];
pair<int, int> t[10*N];

void push(int v) {
    t[v*2].first += lazy[v];
    lazy[v*2] += lazy[v];
    t[v*2+1].first += lazy[v];
    lazy[v*2+1] += lazy[v];
    lazy[v] = 0;
}

pair<int, int> merge(pii a, pii b) {
    if(a.first > b.first) return a;
    return b;
}

void update(int v, int tl, int tr, int l, int r, int addend) {
    if(l > r) return;
    if(tl == l && tr == r) {
        t[v].first += addend;
        lazy[v] += addend;
    }
    else {
        push(v);
        int tm = (tl+tr)/2;
        update(v*2, tl, tm, l, min(r, tm), addend);
        update(v*2+1, tm+1, tr, max(l, tm+1), r, addend);
        t[v] = merge(t[v*2], t[v*2+1]);
```

```
    }
}

pii query(int v, int tl, int tr, int l, int r) {
    if(l > r) return {-1e18, 0};
    if(l <= tl && tr <= r) return t[v];
    push(v);
    int tm = (tl+tr)/2;
    return merge(query(v*2, tl, tm, l, min(r, tm)),
        query(v*2+1, tm+1, tr, max(l, tm+1), r));
}

void build(int v, int tl, int tr) {
    if(tl == tr) {
        t[v].second = tl;
        return;
    }
    build(2*v, tl, (tl+tr)/2);
    build(2*v+1, (tl+tr)/2+1, tr);
}

void solve() {
    int n; cin >> n;
    ll k; cin >> k;
    vpii abc;
    for(int i = 0; i < n; ++i) {
        int l, r; cin >> l >> r;
        ll p; cin >> p;
        rev[r].push_back({l, p});
        abc.push_back({l, r});
    }
    ll ans = 0;
    build(1, 1, MAXCOR);
    pair<int, int> opt;
    for(int i = 1; i <= MAXCOR; ++i) {
        update(1, 1, MAXCOR, 1, i, -k);
        if(rev[i].size() == 0) continue;
        sort(all(rev[i]));
```

```
        for(int j = rev[i].size()-1; j >= 0; j--) {
            update(1, 1, MAXCOR, 1, rev[i][j].first, rev[i][j].second);
        }
        pii zz = query(1, 1, MAXCOR, 1, i);
        auto [cost, index] = zz;
        if(ans < cost) {
            ans = cost;
            opt = {index, i};
        }
    }
    if(ans == 0) {
        cout << 0 << endl;
        return;
    }
    cout << ans << " " << opt.first << " " << opt.second << " ";
    vector<int> anss;
    for(int i = 0; i < n; ++i) {
        if(abc[i].first >= opt.first && abc[i].second <= opt.second) {
            anss.push_back(i+1);
        }
    }
    cout << anss.size() << endl;
    for(auto v : anss) cout << v << " "; cout << endl;
}
```

9) Фенвик

```
struct Fenwick {
    vector<int> data;
    Fenwick(int n) : data(n) { }
    void inc(int p, int x) {
        while (p < data.size()) {
            data[p] += x;
            p |= p + 1;
        }
    }
    int sum(int r) const {
        int res = 0;
        while (r >= 0) {
            res += data[r];
```

```
            r = (r & (r+1)) - 1;
        }
        return res;
    }
};
struct Fenwick {
    vector<ll> data;
    Fenwick(int n) : data(n) { }
    void inc(int p, ll x) {
        while (p < data.size()) {
            data[p] = max(data[p], x);
            p |= p + 1;
        }
    }
    ll getMax(int r) const {
        ll res = 0;
        while (r >= 0) {
            res = std::max(res, data[r]);
            r = (r & (r+1)) - 1;
        }
        return res;
    }
};
```

10) Факторизация с предподсчётом мин. делителей за log(n)

```
const int N = 1e6+10;

vi firstDiv(N, -1);
void precalc() {
    for(int i = 2; i < N; ++i) {
        if(firstDiv[i] != -1) continue;
        for(int j = i; j < N; j += i) {
            if(firstDiv[j] == -1) firstDiv[j] = i;
        }
    }
}

vi getCnt(int x) {
    vi cnt;
```

```
   while(x != 1) {
      int curNpm = firstDiv[x];
      cnt.push_back(0);
      while(firstDiv[x] == curNpm) cnt.back()++, x /= curNpm;
   }
   return cnt;
}


void solve() {
   int n; cin >> n;
   vi cnt = getCnt(n);
   for(auto v : cnt) cout << v << " "; cout << endl;
}

int32_t main()
{
   precalc();
   solve();
}
```

11) Обратные факториалы, биномиальные коэффициенты

```
typedef long long ll;
const int MOD = 1e9+7, N = 2e5+10;
int fc[N+10], inv[N+10];

int myp(int x,int t){
        int a=1;
        for(;t;t>>=1,x=(ll)x*x%MOD)if(t&1)a=(ll)a*x%MOD;
        return a;
}

int C(int a, int b) {
   return a<b?0:(ll)fc[a]*inv[b]%MOD*inv[a-b]%MOD;
}

void precalc() {
   fc[0] = 1;
   for(int i = 1; i <= N; ++i) {
```

```
        fc[i] = (ll)fc[i-1]*i%MOD;
    }
    inv[N] = myp(fc[N], MOD-2);
    for(int i = N; i >= 1; i--) {
        inv[i-1] = (ll)inv[i]*i%MOD;
    }
}

void solve() {
    //
}

int32_t main()
{
    precalc();
    solve();
}
```

12) Быстрое возведение в степень

```
template <typename T>
T modpow(T base, T exp) {
  base %= MOD;
  T result = 1;
  while (exp > 0) {
    if (exp & 1) result = (result * base) % MOD;
    base = (base * base) % MOD;
    exp >>= 1;
  }
  return result;
}
```

13) __int128 хранит числа от -2^127 до 2^127-1

```
// пример использования + вывод в консоль:
#include <bits/stdc++.h>
using namespace std;
ostream & operator<<(ostream & os, __int128 a) {
    if (a == 0) return os << "0";
    bool sign = false;
    if (a < 0) {
        sign = true;
```

```
        a = -a;
    }
    if (sign) os << "-";
    __int128 hi = a / (__int128)1e18L;
    __int128 lo = a - (__int128)1e18L * hi;
    if (hi == 0) os << int64_t(lo);
    else {
        os << int64_t(hi); // выводим старшие разряды
        os << setw(18) << setfill('0') << int64_t(lo) << setfill(' ');
    }
    return os;
}

int main() {
    __int128 a = (int64_t)1e18L - 1;
    cout << a * a << endl;
    cout << a * (-a) << endl;
    cout << a << endl;
    cout << -a << endl;
}
```

14) ordered_set

```
typedef pair<int, int> node;
typedef tree<node, null_type, less<node>,
        rb_tree_tag, tree_order_statistics_node_update> ordered_set;

void solve() {
    ordered_set s;
    s.insert(node(1, 4));
    s.insert(node(1, -6));
    s.insert(node(3, 3));
    s.insert(node(2, -8));
    cout << s.order_of_key(node(1, 5)) << endl; //2
    cout << s.order_of_key(node(3, 0)) << endl; //3
    cout << s.order_of_key(node(2, 1)) << endl; //3
}
```

15) Геометрия (Джарвис + вывести объекты на периметре)

a) Джарвис (минимальная оболочка)

```
struct Point
```

```
{
   int x, y;
};

// 0 --> p, q and r are collinear
// 1 --> Clockwise
// 2 --> Counterclockwise
int orientation(Point p, Point q, Point r)
{
   int val = (q.y - p.y) * (r.x - q.x) -
             (q.x - p.x) * (r.y - q.y);

   if (val == 0) return 0;
   return (val > 0)? 1: 2;
}

void convexHull(Point points[], int n)
{
   if (n < 3) return;

   vector<Point> hull;

   int l = 0;
   for (int i = 1; i < n; i++)
      if (points[i].x < points[l].x)
         l = i;

   int p = l, q;
   do
   {

      hull.push_back(points[p]);
      q = (p+1)%n;
      for (int i = 0; i < n; i++)
      {
        if (orientation(points[p], points[i], points[q]) == 2)
           q = i;
      }
```

```
        p = q;

    } while (p != l);

    // Print Result
    for (int i = 0; i < hull.size(); i++)
        cout << "(" << hull[i].x << ", "
             << hull[i].y << ")\n";
}
```

b) Найти и вывести объекты на периметре

```
bool isClockwiseTurn(vector<int>& a, vector<int>& b, vector<int>& c) {
    int xa = a[0], xb = b[0], xc = c[0];
    int ya = a[1], yb = b[1], yc = c[1];
    bool isConvex = (((yc-yb)*(xb-xa))-((yb-ya)*(xc-xb))) >= 0;
    return isConvex;
}

class Solution {
public:
    vector<vector<int>> outerTrees(vector<vector<int>>& trees) {
        int n = trees.size();
        vector<vector<int>> convexHull;

        auto cmp = [&](vector<int>& a, vector<int>& b) {
            if(a[0] < b[0]) return true;
            if(a[0] == b[0] && a[1] < b[1]) return true;
            return false;
        };
        sort(trees.begin(), trees.end(), cmp);

        for(int i = 0; i < n; ++i) {
            while(convexHull.size() > 1 && !isClockwiseTurn(convexHull[convexHull.size()-
2], convexHull[convexHull.size()-1], trees[i])) {
                convexHull.pop_back();
            }
            convexHull.push_back(trees[i]);
        }
```

```
    for(int i = n-1; i >= 0; i--) {
      while(convexHull.size() > 1 && !isClockwiseTurn(convexHull[convexHull.size()-
2], convexHull[convexHull.size()-1], trees[i])) {
        convexHull.pop_back();
      }
      convexHull.push_back(trees[i]);
    }

    convexHull.pop_back();
    sort(convexHull.begin(), convexHull.end(), cmp);
    convexHull.erase(unique(convexHull.begin(), convexHull.end()),
convexHull.end());
    return convexHull;
  }
};
```

16) Хеши
    a) Легкий хеш для задач, в которых нужно подсчитать кол-во различных stack'ов (например задача про бургеры с Rucode 4.0)

```
#define isz(x) (int)(x).size()
using ull = unsigned long long;
const int mod = (int)1e9+33;

struct Hash : public pair<ull, ull>
{
  Hash(ull fi, ull se) : pair<ull, ull>(fi, se) {}

  Hash(ull v = 0) : Hash(v, v) {}

  Hash operator*(Hash h) const {
    return {first * h.first, second * h.second % mod};
  }

  Hash operator+(Hash h) const {
    return {first + h.first, second + h.second % mod};
  }
};

const Hash p = {(int)1e9+7,(int)1e9+9};
```

```
void solve() {
   vector<Hash> h {{0, 0}};
   set<Hash> allHashes;

   auto push = [&](int x) {
      h.push_back(h.back()*p+x); // h = h*p+x
      allHashes.insert(h.back());
   };

   auto pop = [&]() {
      assert(!h.empty());
      h.pop_back();
   };

   //Пример
   push(1), push(2), pop(), push(2), push(3);
   cout << allHashes.size() << endl;

   //push(1), push(2), pop(), push(2), push(3) -> allHashes.size() == 3
   // (0, 1), (0, 1, 2), (0, 1, 2, 3)
}
```

b) Хеш для сравнения строк

```
#define isz(x) (int)(x).size()
using ull = uint64_t;
using hash_type = pair<int,ull>;

const int mod = 2000000011;
int p = (int)1e9+33;
const int NMAX = 1e5+10;

hash_type operator*(hash_type a, hash_type b) {
   return {a.first * 1LL * b.first % mod, a.second * b.second};
}

hash_type operator+(hash_type a, hash_type b) {
   return {(0LL + a.first + b.first) % mod, a.second + b.second};
}
```

```
hash_type operator-(hash_type a, hash_type b) {
   return {(0LL + a.first - b.first + mod) % mod, a.second - b.second};
}

const std::vector<hash_type> power = [](){
   std::vector<hash_type> answer(NMAX,{1,1});
   for (int i = 1; i < NMAX; i++) {
      answer[i] = answer[i-1] * hash_type(p,p);
   }
   return answer;
}();

struct Hash {
   vector<hash_type> pref;

   Hash(const string &s) {

      pref.assign(isz(s)+1,{0,0});
      for (int i = 0; i < isz(s); i++) {
         pref[i+1] = pref[i] * power[1] + hash_type(s[i],s[i]);
      }
   }

   void extend(const string &s) {
      int n = pref.size();
      int m = s.length();
      pref.resize(n+m);
      for(int i = 0; i < m; ++i) {
         pref[n+i] = pref[n+i-1] * power[1] + hash_type{s[i], s[i]};
      }
   }

   hash_type getHash(int i, int j) const {
      if (i > j) return {0, 0};
      return pref[j+1] - pref[i] * power[j-i+1];
   }
};
```

```
void solve() {
   string s = "abaackt";
   string t = "abcktaa";
   Hash hs(s), ht(t);
   //Равные подстроки
   cout << (hs.getHash(0, 1) == ht.getHash(0, 1)) << endl; // "ab"
   cout << (hs.getHash(2, 3) == ht.getHash(5, 6)) << endl; // "aa"
   cout << (hs.getHash(4, 6) == ht.getHash(2, 4)) << endl; // "ckt"
   //Разные подстроки
   cout << (hs.getHash(0, 3) == ht.getHash(3, 6)) << endl;
   cout << (hs.getHash(1, 5) == ht.getHash(2, 6)) << endl;
}
```

13) Матрицы и их возведение в степень (Код для задачи - https://atcoder.jp/contests/dp/tasks/dp_r Дана матрица смежности графа, нужно найти количество различных путей длины k, причем пути начинаются из всех вершин).

```
const int mod = 1e9+7;
const int N = 50;

struct matrix {
   int m[N][N];

   matrix() {
      memset(m, 0, sizeof(m));
   }

   matrix operator * (matrix b) {
      matrix c = matrix();
      for(int i = 0; i < N; ++i) {
         for(int j = 0; j < N; ++j) {
            for(int k = 0; k < N; ++k) {
               long long x = c.m[i][j]+1LL*m[i][k]*b.m[k][j];
               if(x > mod) x %= mod;
               c.m[i][j] = x;
            }
         }
```

```
      }
      return c;
   }
};

matrix unit;
matrix modPow(matrix m, long long n) {
  matrix result;
  for(int i = 0; i < N; ++i) {
   result.m[i][i]=1;
  }
  while (n > 0) {
   if (n & 1) result = result * m;
   m = (m * m);
   n >>= 1;
  }
  return result;
}

void solve() {
   matrix A;
   int n; cin >> n;
   long long k; cin >> k;
   for(int i = 0; i < n; ++i) {
      for(int j = 0; j < n; ++j) {
         cin >> A.m[i][j];
      }
   }
   matrix ans = modPow(A, k);
   long long res = 0;
   for(int i = 0; i < N; ++i) {
      for(int j = 0; j < N; ++j) {
         res = (res + ans.m[i][j])%mod;
      }
   }
   cout << res << endl;
}
```

14) Прогрессии

```
Sg(M q, ll n) {//геометрическая прогрессия (1 + q + q^2 + ...), n - кол-во слагаемых
	M res = 0;
	if(n & 1) {
		--n;
		res += pow(q, n);
	}
	if (n > 0)
		res += (1 + q) * Sg(q*q, n >> 1);
	return res;
}


M Sa(M d, ll n) {//арифметическая прогрессия (1 + d + 2d + ...), n - кол-во слагаемых
	M res = 0;
	if(n & 1) {
		--n;
		res += 1 + n*d;
	}
	if (n > 0)
		res += 2 * Sa(d, n >> 1) + (n >> 1) * (n >> 1) * d;
	return res;
}
```