Team Notebook CodeWinter Genius

- 1) Source
 - 1.1)
- 2) Деревья
 - 2.1) HLD
 - 2.2) Link-cut
 - 2.3) ETT
- 3) Графы
 - 3.1) Нахождение отрицательного цикла
 - 3.2) mincost maxflow
- 4) TY
 - 4.1) FFT
 - 4.2) Поллард
 - 4.3) Решето Эратосфена за O(n)
- 5) Структуры данных
 - 6.1) ДО снизу
 - 6.2) Персистентное ДД
- 6) Строки
 - 7.1) Суфф. автомат
 - 7.2) Суфф. масс + lcp
- 7) Оптимизация Динамики
 - 9.1) Ли Чао
 - 9.2) Лямбда

```
1) Source
       #define FAST_ALLOCATOR_MEMORY (5e8)
  27
  28
      #ifdef FAST_ALLOCATOR_MEMORY
  29
             int allocator_pos = 0;
             char allocator_memory[(int)FAST_ALLOCATOR_MEMORY];
  30
            inline void * operator new ( size_t n ) {
  31
  32
                   char *res = allocator_memory + allocator_pos;
  33
                   allocator_pos += n;
                   assert(allocator_pos <= (int)FAST_ALLOCATOR_MEMORY);</pre>
  34
                  return (void *)res;
  35
  36
             inline void operator delete ( void * ) noexcept { }
  37
      #endif
       2) Деревья
2.1) HLD
                                      bool cmp(ll a, ll b) {
void prec(ll v, ll p = -1) {
                                         return !(sz[a] < sz[b]);
    sz[v] = 1;
    tin[v] = ttime++;
                                      bool isp(ll a, ll b) {
    for (auto x : g[v]) {
                                       if(tin[a] <= tin[b] && tout[b] <= tout[a])</pre>
                                            return 1;
        if(x == p)
                                         return 0;
           continue:
        prec(x, v);
                                      void update(ll v) {
        sz[v] += sz[x];
                                         if(v == 0)
                                            return;
    tout[v] = ttime++;
                                         tree[v] = tree[v * 2] + tree[v * 2 + 1];
                                         update(v / 2);
ll dfs(ll v, ll p = -1, ll l = -1) { _{11 \text{ get(ll v, ll l, ll r, ll vl, ll vr)}} {
    par[v] = p;
                                         if(vr <= 1 || r <= v1)</pre>
                                             return 0;
    if(1 == -1) {
                                          if(v1 <= 1 && r <= vr) {
        1 = v;
                                            return tree[v];
                                         11 m = (1 + r) / 2:
    gr[v].ft = 1;
                                         return get(v * 2, 1, m, vl, vr) + get(v * 2 + 1, m, r, vl, vr);
    pos[v] = euler.size();
    euler.pb(v);
                                     11 solve(11 v) {
                                        ll ans = 0;
    11 \text{ nxt} = -1;
                                         11 pr = -1;
    for (auto x : g[v]) {
                                         while(v != -1) {
        if(x == p)
                                            if(pr == -1) {
            continue;
                                                ans += get_sz(v) * w[v];
        nxt = x;
                                                if(pos[gr[v].ft] < pos[v]) {</pre>
        gr[v].sc = dfs(x, v, 1);
                                                    ans += get(1, 0, N, pos[gr[v].ft], pos[v]);
        break;
                                            }
    }
                                            else {
    if(nxt == -1) {
                                               if(pos[gr[v].ft] < pos[v]) {
        gr[v].sc = v;
                                                   ans += get(1, 0, N, pos[gr[v].ft], pos[v]);
        rght[v] = euler.size();
        return v;
                                                ans += (get_sz(v) - get_sz(pr)) * w[v];
    }
                                            pr = gr[v].ft;
    for (auto x : g[v]) {
                                            v = par[gr[v].ft];
        if(x == p \mid\mid x == nxt)
                                        }
           continue;
                                         return ans;
        dfs(x, v, -1);
                                    }
    rght[v] = euler.size();
    return gr[v].sc;
}
```

2.2) Link-cut

```
v \rightarrow p \rightarrow p \rightarrow push():
      #include <iostream>
                                                                                        v->p->push();
 2
      #include <cstdio>
                                                                                        v->push();
      #include <cassert>
 3
                                                                                        \begin{array}{c} \text{if } (v \rightarrow p \rightarrow p := \text{None}) \\ \text{if } ((v \rightarrow p \rightarrow 1 := v) \\ \text{rotate} (v \rightarrow p); \end{array}
                                                                                 58
 4
                                                                                 59
                                                                                                                        ( v \rightarrow p \rightarrow p \rightarrow p = v \rightarrow p ) )
      using namespace std;
                                                                                 60
 6
                                                                                 61
      // BEGIN ALGO
                                                                                 62
                                                                                          rotate(v);
 8
                                                                                 64
                                                                                        rotate(v);
 9
      const int MAXN = 110000;
                                                                                 65
10
                                                                                       inline void Splay(node v){
while (v->p != None) bigRotate(v);
                                                                                 66
      typedef struct _node{
11
                                                                                 67
       _node *1, *r, *p, *pp;
int size; bool rev;
12
                                                                                 68
13
                                                                                 69
                                                                                       inline void splitAfter(node v){
                                                                                 70
14
        _node();
                                                                                        v->push();
                                                                                        Splay(v);
        explicit _node(nullptr_t){
                                                                                 71
15
                                                                                 72
                                                                                        v->r->p = None;
        1 = r = p = pp = this;
16
                                                                                       {\tt v-\!\!>\!\!r-\!\!>\!\!pp}\ =\ {\tt v}\,;
                                                                                 73
         size = rev = 0;
17
                                                                                 74
                                                                                        v->r = None:
18
                                                                                 75
                                                                                        v->update();
19
        void push () {
                                                                                 76
        if (rev){
1->rev ^= 1; r->rev ^= 1;
20
                                                                                 77
                                                                                       void expose(int x){
21
                                                                                        node v = v2n[x];
splitAfter(v);
                                                                                 78
           rev = 0; swap(1,r);
22
                                                                                 79
                                                                                        while (v->pp'!= None) {
  assert(v->p == None);
  splitAfter(v->pp);
                                                                                 80
23
       }
                                                                                 81
24
        void update();
25
                                                                                 83
                                                                                         assert(v->pp->r == None);
      }* node;
26
                                                                                 84
                                                                                         assert(v->pp->p == None);
27
      node None = new _node(nullptr);
                                                                                 85
                                                                                         \verb"assert" ( ! v -> pp -> rev") ;
28
      node v2n [MAXN];
                                                                                 86
                                                                                         v \rightarrow pp \rightarrow r = v
      _node :: _node () {
                                                                                 87
                                                                                         v->pp->update();
                                                                                         v = v -> pp;
30
       1 = r = p = pp = None;
                                                                                 88
                                                                                 89
                                                                                         v \rightarrow pp = None;
       size = 1; rev = false;
31
                                                                                 90
32
                                                                                 91
                                                                                        assert(v->p == None);
33
      void _node :: update() {
                                                                                 92
                                                                                        Splay(v2n[x]);
       size = (this != None) + 1->size + r->size;
34
                                                                                 93
       1->p = r->p = this;
35
                                                                                       inline void makeRoot(int x){
                                                                                 94
36
                                                                                 95
                                                                                        expose(x);
37
      void rotate(node v){
                                                                                 96
                                                                                        assert(v2n[x]->p == None);
                                                                                        \begin{array}{lll} \texttt{assert} \, (\, \texttt{v2n} \, [\, \texttt{x}] \! - \! > \! \texttt{pp} & = \! \texttt{None} \, ) \, ; \\ \texttt{assert} \, (\, \texttt{v2n} \, [\, \texttt{x}] \! - \! > \! \texttt{r} & = \! \texttt{None} \, ) \, ; \end{array}
38
       assert(v != None && v->p != None);
                                                                                 97
        assert(!v->rev); assert(!v->p->rev);
                                                                                 98
39
                                                                                 99
                                                                                        v2n[x]->rev
                                                                                                          ^= 1:
40
        node u = v -> p;
                                                                                100
41
        if (v = u \rightarrow 1)
                                                                                       inline void link(int x, int y) {
  makeRoot(x); v2n[x]->pp = v2n[y];
                                                                                101
42
         u->1 = v->r, v->r = u;
43
        else
                                                                                103
        u->r = v->1, v->1 = u;
44
                                                                                104
                                                                                       inline void cut(int x, int y){
                                                                                        expose(x);
Splay(v2n[y]);
45
        \verb"swap" ( \verb"u->p" , \verb"v->p") \; ; \; \; \verb"swap" ( \verb"v->pp" , \verb"u->pp") \; ;
                                                                                105
        if (v->p!= None){
46
                                                                                106
                                                                                        if (v2n[y]->pp != v2n[x]){
                                                                                107
        assert(v->p->1 = u \mid | v->p->r = u);
47
                                                                                        swap(x,y);
48
          if (v->p->r == u) v->p->r = v;
                                                                                109
                                                                                          expose(x)
          else v\rightarrow p\rightarrow 1 = v:
49
                                                                                110
                                                                                         Splay(v2n[y]);
50
                                                                                111
                                                                                         \mathtt{assert}\,(\,\mathtt{v2n}\,[\,\mathtt{y}]->\mathtt{pp}\,=\!\!\!\!-\,\mathtt{v2n}\,[\,\mathtt{x}\,]\,)\;;
51
        u->update(); v->update();
                                                                                112
52
                                                                                113
                                                                                        v2n[y]->pp = None;
53
      void bigRotate(node v){
                                                                                114
                                                                                       inline int get(int x, int y){
                                                                                115
       assert (v->p != None);
54
                                                                                        if (x == y) return 0;
makeRoot(x);
                                                                                116
                                                                                117
                                                                                118
                                                                                        expose(y); expose(x);
                                                                                        Splay(v2n[y]);
if (v2n[y]->pp != v2n[x]) return -1;
return v2n[y]->size;
                                                                                119
                                                                                120
                                                                                121
                                                                                122
                                                                                       // END ALGO
                                                                                123
                                                                                124
                                                                                       _node mem[MAXN];
                                                                                125
                                                                                126
                                                                                127
                                                                                       int main(){
  freopen("linkcut.in","r",stdin);
  freopen("linkcut.out","w",stdout);
                                                                                128
                                                                                129
                                                                                130
                                                                                131
                                                                                       int n,m;
scanf("%d %d",&n,&m);
                                                                                132
                                                                                133
                                                                                134
                                                                                        for (int i = 0; i < n; i++)
                                                                                135
                                                                                136
                                                                                         v2n[i] = &mem[i];
                                                                                137
                                                                                        for (int i = 0; i < m; i++){
                                                                                138
                                                                                139
                                                                                         int a,b;
                                                                                         if (scanf(" link %d %d",&a,&b) == 2)
                                                                                140
                                                                                         link(a-1,b-1);
else if (scanf(" cut %d %d",&a,&b) == 2)
                                                                                141
                                                                                142
                                                                                           cut(a-1,b-1);
                                                                                143
                                                                                          \begin{array}{ll} \text{else if (scanf(" get %d %d",&a,&b) == 2)} \\ \text{printf("%d\n",get(a-1,b-1));} \end{array}
                                                                                144
```

145 146

147

assert (false);

```
2.3) ETT
```

```
1 const int N = 3e5 + 228;
                                              59 pair <treap *, treap *> split(treap *t, int k) {
 2 mt19937 rnd(228);
                                                      if (!t) return {0, 0};
                                              61
                                                      t->par = 0;
 4 struct treap {
                                              62
                                                      if (siz(t->1) >= k) {
       treap *1, *r, *par;
                                                          auto a = split(t->1, k);
                                              63
 6
       int sz;
                                                           set_l(t, a.second);
                                              64
 7
       int y;
                                              65
                                                          return {a.first, t};
 8
       treap() {
                                              66
                                                      } else {
 9
            sz = 1;
                                                           auto a = split(t->r, k - siz(t->l) - 1);
                                              67
            y = rnd();
10
                                              68
                                                           set_r(t, a.first);
11
            par = 0;
                                              69
                                                          return {t, a.second};
12
            1 = r = 0;
                                              70
                                                      }
13
       }
                                              71 }
       treap(treap *p) {
14
                                              72
15
            sz = 1;
                                              73 treap *merge(treap *a, treap *b) {
            y = rnd();
16
                                              74
                                                      if (!a) return b;
17
            par = p;
                                              75
                                                      if (!b) return a;
18
            1 = r = 0;
                                                      a\rightarrow par = 0, b\rightarrow par = 0;
                                              76
19
                                              77
                                                      if (a->y > b->y) {
20 };
                                              78
                                                           set_r(a, merge(a->r, b));
21
                                              79
                                                           return a;
22 int siz(treap *t) {
                                              80
                                                      } else {
23
       if (!t) return 0;
                                              81
                                                           set_l(b, merge(a, b->1));
24
       return t->sz;
                                              82
                                                           return b;
25 }
                                              83
                                                      }
26
                                              84 }
27 void recalc(treap *t) {
       if (!t) return;
                                              85
29
       t->sz = 1 + siz(t->1) + siz(t->r);
                                              86 treap *get_root(treap *t) {
30 }
                                              87
                                                      while (t->par) t = t->par;
31
                                              88
                                                      return t;
32 int get_id(treap *t) {
                                              89 }
33
       int sum = siz(t->1);
                                              90
34
       while (t->par) {
                                              91 set <int> g[N];
           if (t->par->l == t) {
                                              92 map <pair <int, int>, treap*> st;
                t = t->par;
                                              93
            } else {
37
                                              94 pair <int, int> any_edge(int v) {
38
                sum += siz(t->par->1) + 1;
                                              95
                                                      assert(!g[v].empty());
                t = t->par;
39
                                              96
                                                      return make_pair(v, *g[v].begin());
40
                                              97
41
                                              98
42
       return sum:
                                              99 bool is_connected(int a, int b) {
43 }
                                                      if (a == b) {
44
                                              101
                                                          return true;
45 void set_l(treap *t, treap *x) {
                                              102
        if (x)
                                              103
                                                      if (g[a].empty() || g[b].empty()) return false;
47
            x->par = t;
                                              104
                                                      return get_root(st[any_edge(a)]) == get_root(st[any_edge(B)])
        t -> 1 = x;
48
                                              105 }
49
       recalc(t);
50 }
                                              106
                                              107
                                                 void make_first(treap *t) {
51
52 void set_r(treap *t, treap *x) {
                                              108
                                                      int id = get_id(t);
53
       if(x)
                                              109
                                                      auto a = split(get_root(t), id);
           x->par = t;
                                              110
                                                      merge(a.second, a.first);
54
55
       t->r = x;
                                              111 }
56
       recalc(t);
                                              112
57 }
                                              113 void make_last(treap *t) {
58
                                              114
                                                      int id = get_id(t);
                                              115
                                                      auto a = split(get_root(t), id + 1);
                                                      merge(a.second, a.first);
                                              116
                                              117 }
                                              118
```

```
119 void link(int u, int v) {
120
        st[{u, v}] = new treap();
121
        st[{v, u}] = new treap();
122
123
        if (g[v].empty()) swap(u, v);
124
        if (g[u].empty()) {
            if (g[v].empty()) {
125
126
                 merge(st[{u, v}], st[{v, u}]);
127
            } else {
128
                 auto x = any_edge(v);
129
                 make_first(st[x]);
130
                 merge(st[{u, v}], get_root(st[x]));
131
                 merge(get_root(st[x]), st[{v, u}]);
132
            }
133
        } else {
134
            pair <int, int> x, y;
135
136
                 x = any_edge(u);
137
                 swap(x.first, x.second);
138
                 make_last(st[x]);
139
            }
140
            {
141
                 y = any_edge(v);
142
                 make_first(st[y]);
143
144
            merge(get_root(st[x]), st[{u, v}]);
145
            merge(get_root(st[x]), get_root(st[y]));
146
            merge(get_root(st[x]), st[{v, u}]);
147
148
        g[u].insert(v);
149
        g[v].insert(u);
150
    }
151
    void cut(int u, int v) {
152
        g[u].erase(v);
153
        g[v].erase(u);
154
        auto x = make_pair(u, v);
155
        make_first(st[x]);
156
157
        auto a = split(get_root(st[x]), 1);
158
        auto y = make_pair(v, u);
159
        int ret = get_id(st[y]);
        auto b = split(a.second, ret);
160
161
         split(b.second, 1);
162
        delete st[x];
163
        delete st[y];
         st[x] = st[y] = 0;
164
165 }
```

3) Графы

3.1) Нахождение отрицательного цикла

```
struct edge {
      int a, b, cost;
};
int n, m;
vector<edge> e;
const int INF = 1000000000;
void solve() {
      vector<int> d (n);
      vector\langle int \rangle p (n, -1);
      int x;
       for (int i=0; i<n; ++i) {</pre>
             x = -1;
             for (int j=0; j<m; ++j)</pre>
                    if (d[e[j].b] > d[e[j].a] + e[j].cost) {
                            d[e[j].b] = max (-INF, d[e[j].a] +
e[j].cost);
                           p[e[j].b] = e[j].a;
                            x = e[j].b;
       }
       if (x == -1)
             cout << "No negative cycle found.";</pre>
       else {
             int y = x;
              for (int i=0; i<n; ++i)</pre>
                     y = p[y];
             vector<int> path;
             for (int cur=y; ; cur=p[cur]) {
                    path.push back (cur);
                    if (cur == y && path.size() > 1) break;
             reverse (path.begin(), path.end());
             cout << "Negative cycle: ";</pre>
             for (size t i=0; i<path.size(); ++i)</pre>
                    cout << path[i] << ' ';
```

3.2) mincost maxflow

78 **}**

```
8 struct edge {
                                                                     80 int max_flow(int n) {
      int u, c, w, rev, f = 0;
                                                                     81
                                                                              int f = 0;
10
       edge() {}
                                                                              Ford_Bellman(n);
                                                                     82
       edge(int u, int c, int w, int rev): u(u), c(c), w(w), rev(rev) {}
                                                                     83
12 }:
                                                                              while (Dijkstra(n)) {
13
                                                                     84
                                                                                   int pos = t;
14 const int MAX_N = 1007, INF = 1e18;
                                                                                   vector <pair <int, int>> path;
                                                                     85
15 vector <edge> g[MAX_N];
                                                                     86
                                                                                   while (pos !=-1) {
16 int dist[MAX N]:
17 int p[MAX_N];
                                                                     87
                                                                                        if (prevv[pos].first != -1)
18 pair <int, int> prevv[MAX_N];
                                                                     88
                                                                                            path.push_back(prevv[pos]);
19 bool visited[MAX_N];
                                                                     89
                                                                                        pos = prevv[pos].first;
20 int s, t;
                                                                     90
22 void relax(int v, int pos, int f) {
                                                                     91
                                                                                   int cur_f = INF;
     g[v][pos].c = f;
                                                                     92
                                                                                   for (auto elem : path) {
       g[v][pos].f += f;
                                                                     93
                                                                                        cur_f = min(cur_f, g[elem.first][elem.second].c);
25
       int u = g[v][pos].u, pos2 = g[v][pos].rev;
       g[u][pos2].c += f;
                                                                     94
       g[u][pos2].f -= f;
                                                                     95
                                                                                   for (auto elem : path)
                                                                     96
                                                                                       relax(elem.first, elem.second, cur_f);
30 void Ford_Bellman(int n) {
                                                                     97
                                                                                   f += cur_f;
     for (int i = 0; i <= n; i++)
                                                                     98
          dist[i] = INF;
32
                                                                     99
                                                                              return f;
       dist[s] = 0;
                                                                     100 }
       prevv[s] = \{-1, -1\};
       for (int i = 0; i <= n; i++) {
                                                                    102 signed main() {
36
          for (int v = 0; v <= n; v++) {</pre>
                                                                    103
                                                                             int n;
37
              for (int j = 0; j < (int) g[v].size(); j++) {</pre>
                                                                    104
                                                                             cin >> n:
38
                  edge e = g[v][j];
                                                                             s = n + n;
                                                                    105
39
                  if (e.c <= 0)
                                                                    106
                                                                             t = s + 1;
40
                     continue;
                                                                             for (int i = 0; i < n; i++) {
                                                                    107
                  dist[e.u] = min(dist[e.u], dist[v] + e.w);
41
                                                                                 g[s].push_back(edge(i, 1, 0, g[i].size()));
                                                                    108
42
              }
                                                                     109
                                                                                 g[i].push_back(edge(s, 0, 0, (int) g[s].size() - 1));
43
          }
44
                                                                    110
                                                                                 g[i + n].push_back(edge(t, 1, 0, g[t].size()));
45 }
                                                                                 g[t].push_back(edge(i + n, 0, 0, (int) g[i + n].size() - 1));
                                                                    111
47
   bool Dijkstra(int n) {
                                                                    112
       for (int i = 0; i <= n; i++)
                                                                             for (int i = 0; i < n; i++) {
48
                                                                    113
                                                                    114
                                                                                 for (int j = 0; j < n; j++) {</pre>
49
            p[i] = dist[i] + p[i];
                                                                    115
                                                                                     int x;
        for (int i = 0; i <= n; i++)
                                                                    116
                                                                                     cin >> x:
            dist[i] = INF;
51
                                                                                     g[i].push_back(edge(n + j, 1, x, g[n + j].size()));
                                                                    117
        memset(visited, 0, sizeof(visited));
                                                                    118
                                                                                     g[n + j].push_back(edge(i, 0, -x, (int) g[i].size() - 1));
53
        dist[s] = 0;
                                                                                 }
                                                                    119
        prevv[s] = \{-1, -1\};
54
                                                                    120
                                                                             }
        while (1) {
                                                                     121
                                                                             max_flow(t);
            int mini = INF, v = -1;
56
                                                                             vector <pair <int, int>> ans;
57
            for (int i = 0; i <= n; i++) {
                                                                             int cost = 0;
                                                                    123
58
                 if (dist[i] < mini && !visited[i]) {</pre>
                                                                    124
                                                                             for (int i = 0; i < n; i++) {
                     mini = dist[i];
59
                                                                    125
                                                                                 for (edge e : g[i]) {
                                                                    126
                                                                                     if (e.f == 1) {
60
                     v = i;
                                                                                         cost += e.w;
61
                 }
                                                                                         ans.push_back(\{i + 1, e.u - n + 1\});
                                                                    128
            }
62
                                                                    129
                                                                                     }
            if (v == -1)
63
                                                                     130
                                                                                 }
64
                 break;
                                                                             }
            visited[v] = 1;
65
                                                                    132
                                                                             cout << cost << endl;</pre>
             for (int i = 0; i < (int) g[v].size(); i++) {
                                                                    133
                                                                             for (auto elem : ans) {
                 edge e = g[v][i];
67
                                                                                 cout << elem.first << ' ' << elem.second << endl;</pre>
                                                                    134
68
                 if (e.c <= 0 || visited[e.u])</pre>
                                                                    135
69
                     continue;
                                                                    136
                                                                             return 0;
70
                 int w = e.w + p[v] - p[e.u];
                                                                    137 }
                 if (dist[e.u] > dist[v] + w) {
72
                     dist[e.u] = dist[v] + w;
73
                     prevv[e.u] = {v, i};
74
            }
75
76
77
        return (dist[t] < INF);
```

```
4) TY
4.1) FFT
27 int maxlog = 19;
 28 int N = (1 << maxlog);
29 ld pi = acos(-1);
30
31
    vector<complx> fft(vector<complx> a) {
32
        vector<int> rev(N);
 33
        for (int i = 0; i < N; i++) {
            rev[i] = (rev[i / 2] / 2) + ((i & 1) << (maxlog - 1));
 34
 35
        for (int i = 0; i < N; i++) {
 36
 37
            if(i < rev[i])</pre>
 38
                swap(a[i], a[rev[i]]);
 39
 40
        for (int k = 1; k < N; k *= 2) {
            complx w = \{\cos(2 * pi / (2 * k)), \sin(2 * pi / (2 * k))\};
 41
 42
            for (int s = 0; s < N; s += 2 * k) {
 43
                complx now_w = \{1, 0\};
                for (int j = 0; j < k; j++) {
 44
 45
                    complx u = a[s + j];
 46
                    complx v = now_w * a[s + j + k];
 47
                    a[s + j] = u + v;
                    a[s + j + k] = u - v;
 4.0
                    now_w *= w;
 50
                }
 51
            }
        }
 52
        return a;
53
 54 }
55
 56 vector<int> get_ans(vector<complx> a) {
 57
        a = fft(a);
 58
        reverse(a.begin() + 1, a.end());
 59
        vector<int> ans(N, 0);
        for (int i = 0; i < N; i++)
           ans[i] = a[i].real() / N + 0.5;
 61
        return ans;
62
 63 }
4.2) Поллард
20 typedef __int128 bigint;
21
22 vector<int> have;
23 vector<ll> anss;
24 int maxn = 1e4;
25
   mt19937 rnd(228);
26
   bigint pw(bigint a, bigint st, bigint mod) {
27
28
         if(st == 0)
29
              return 1;
         if(st % 2 == 0) {
30
31
             bigint y = pw(a, st / 2, mod);
32
             return (y * y) % mod;
         }
33
         else {
34
35
             bigint y = pw(a, st - 1, mod);
             return (y * a) % mod;
36
37
         }
38
    }
39
40
   bigint gcd(bigint a, bigint b) {
         if(a == 0 || b == 0)
41
```

42

43

44 }

return a + b;

return gcd(b, a % b);

```
void solve(bigint N) {
    bool check(bigint p) {
46
                                                            95
                                                                    if(check(N)) {
        if(p < 1000) {
47
                                                            96
                                                                        anss.pb(N);
             for (int i = 2; i * i <= p; i++) {
48
                                                            97
                                                                        return;
                  if(p % i == 0)
49
                                                            98
50
                      return 0;
                                                                    bool fl = 0;
             }
51
                                                                    for (int i = 2; i < maxn; i++) {
                                                            100
52
             return 1;
                                                            101
                                                                        while(N % i == 0) {
        }
53
                                                            102
                                                                             anss.pb(i);
54
                                                            103
                                                                             fl = 1;
                                                                             N /= i;
55
        bigint f = (p - 1);
                                                                        }
                                                            105
        int cnt = 0;
56
                                                            106
                                                                    }
        while(f % 2 == 0) {
57
                                                           107
                                                                    if(f1) {
58
             cnt++;
                                                            108
                                                                        solve(N);
59
             f /= 2;
                                                            109
                                                                        return;
        }
60
                                                            110
                                                                    }
61
        for (auto a : have) {
                                                            111
             if(pw(a, p - 1, p) != 1)
62
                                                                    while(1) {
                                                           112
                  return 0;
63
                                                           113
                                                                        bigint a = rnd() % N;
             if(gcd(a, p) != 1)
                                                           114
                                                                        vector<bigint> now;
64
                                                           115
                                                                        int v1 = -1, v2 = -1;
                 return 0;
65
                                                           116
                                                                        11 \text{ ans} = -1;
             vector<br/>bigint> now;
66
                                                           117
                                                                        while(1) {
             bigint r = pw(a, f, p);
67
                                                                             now.pb(a);
                                                           118
             for (int i = 0; i <= cnt; i++) {
68
                                                           119
                                                                             a = f(a, N);
                  now.pb(r);
69
                                                           120
                                                                             now.pb(a);
70
                  r = (r * r) % p;
                                                                             v2 += 2;
                                                           121
71
             }
                                                                             v1 += 1;
                                                            122
             for (int i = len(now) - 1; i >= 0; i--) 123
72
                                                                             bigint g = gcd(abss(now[v1] - now[v2]), N);
73
                  if(now[i] != 1) {
                                                            124
                                                                            if(g == N)
                                                                                 break;
                      if(now[i] != (p - 1))
74
                                                            126
                                                                             if(g != 1) {
75
                           return 0;
                                                            127
                                                                                 ans = g;
76
                      break;
                                                            128
                                                                                 break;
77
                  }
                                                                             }
                                                            129
             }
78
                                                            130
                                                                        }
79
        }
                                                            131
                                                                        if(ans != -1) {
80
        return 1;
                                                           132
                                                                             solve(ans);
81
    }
                                                           133
                                                                             solve(N / ans);
                                                           134
                                                                             return;
82
    bigint f(bigint a, bigint mod) {
                                                           135
                                                                        }
83
                                                           136
                                                                    }
        bigint res = (a * a + 1) \% mod;
84
                                                           137 }
85
        return res;
                                                           147
                                                                     for (int i = 2; i < 1e3; i++) {
   }
86
                                                                          have.push_back(i);
                                                           148
87
                                                           149
                                                                          for (int j = 2; j < i; j++) {
88
   bigint abss(bigint x) {
                                                                               if(i \% j == 0) {
        if(x < 0)
                                                           150
89
                                                                                    have.pop_back();
                                                           151
90
             return -x;
                                                                                    break;
                                                           152
91
        return x;
                                                                               }
92 }
                                                           153
                                                           154
                                                                          }
                                                                          if(have.size() >= 30)
                                                           155
                                                                               break;
                                                           156
                                                                     }
                                                           157
                                                           158
                                                                     11 N;
                                                           159
                                                                     cin >> N;
                                                                     solve(N);
                                                           160
```

4.3) Решето Эратосфена за O(n)

8

9

10

11

12

13

15

16

17

18

19

20

21

23

24

25

27

28

22 }

26 }

14 int 0;

```
24
                  vector<int> primes;
        25
                  for (11 i = 2; i < maxn; i++) {
        26
                       if(p[i] == -1) {
        27
                            p[i] = primes.size();
        28
                            primes.push_back(i);
        29
                            11 ic = i;
                      }
        30
                      for (int j = 0; j \le p[i]; j++) {
        31
                            11 x = primes[j];
        32
                            if(x * i > maxn) break;
        33
                            p[primes[j] * i] = j;
        34
                      }
        35
                  }
        36
        0.7
                                                               29 void assign_relax (int i, int v) {
                   5) Структуры данных
                                                                    assign (i, v);
                                                               30
            5.1) ДО снизу
                                                                    i += 0;
                                                               31
                                                                    while (i > 1) {
                                                               32
#define S (1 << 17) // 131072
                                                               33
                                                                      i >>= 1;
#define INF (1e18 + 1)
                                                               34
                                                                      relax (i);
                                                               35
                                                                    }
                                                               36 }
struct node {
  long long ls, rs, s, m;
                                                               37
                                                               38
                                                                 long long eval (int 1, int r) {
T[2 * S];
                                                               39
                                                                    1 += 0, r += 0;
                                                                    long long ls = -INF, rs = -INF, m = -INF;
                                                               40
                                                               41
                                                                    while (1 <= r) {
                                                                      if (1 & 1) {
                                                               42
void relax (int i) {
                                                               43
                                                                        m = max (m, max (T[1].m, ls + T[1].ls));
  int l = 2 * i, r = l + 1;
                                                               44
                                                                       ls = max (T[1].rs, ls + T[1].s);
  T[i].ls = max (T[1].ls, T[1].s + T[r].ls);
                                                               45
                                                                        ++1;
  T[i].rs = max (T[r].rs, T[1].rs + T[r].s);
                                                                      }
                                                               46
  T[i].s = T[1].s + T[r].s;
                                                                      if (!(r & 1)) {
                                                               47
  T[i].m = max (max (T[1].m, T[r].m), T[1].rs + T[r].ls);
                                                                        m = max (m, max (T[r].m, rs + T[r].rs));
                                                               48
                                                                        rs = max (T[r].ls, rs + T[r].s);
                                                               49
                                                               50
                                                                        --r;
void assign (int i, int v) {
                                                                      }
                                                               51
 T[0 + i].ls = T[0 + i].rs = T[0 + i].s = T[0 + i].m = v;
                                                               52
                                                                      1 >>= 1, r >>= 1;
                                                               53
```

54

56

55 }

return max (m, ls + rs);

```
5.2) Персистентное ДД
1. struct treap
2. {
3.
     int val, sz;
     treap *1, *r;
4.
5.
     treap(int val, treap *1, treap *r): val(val), sz(1), l(1), r(r)
7.
8.
    treap()
9.
10.
11. };
12.
13. const int N = 10 * 10000 * 100 + 7;
14. const int MAX_SZ = N - 1e5;
15.
16. treap node[N];
17. int ptr = 0;
19. treap *new treap(int val, treap *1, treap *r)
20.{
21. node[ptr].val = val;
22. node[ptr].l = 1;
23. node[ptr].r = r;
24. node[ptr].sz = 1;
25. assert(ptr < N);
26. return &node[ptr++];
27.}
28.
29.int sz(treap *t)
30.{
31. if (!t)
32. {
33. return 0;
34. }
35. else
36.
37.
       return t->sz;
38. }
39.}
40.
41. void recalc(treap *t)
42. {
43. if (!t)
44. {
45.
     return;
46. }
47. else
48. {
49. t->sz = sz(t->1) + 1 + sz(t->r);
50. }
51.}
53. void zhfs(treap *t, vector <int> &a)
54. {
55. if (!t)
56. {
57.
58.
     return;
59. zhfs(t->1, a);
60. a.push back(t->val);
61. zhfs(t->r, a);
62.}
64.pair <treap*, treap*> split(treap *t, int x)
65.{
66. if (!t)
67. {
68.
       return {0, 0};
```

```
69. }
70. if (sz(t->1) >= x)
71.
72.
      auto a = split(t->1, x);
73.
      treap *l = a.first;
      treap *r = new_treap(t->val, a.second, t->r);
74.
75.
      recalc(r);
76.
77.
      return {1, r};
78. else
79.
80.
      auto a = split(t->r, x - sz(t->1) - 1);
81.
      treap *l = new treap(t->val, t->l, a.first);
82.
      treap *r = a.second;
      recalc(1);
83.
84.
      return {1, r};
85. }
86.}
87.
88. treap *merge(treap *a, treap *b)
89.{
90. if (!a)
91. {
92.
      return b;
93.
94. if (!b)
95. {
96.
      return a;
97. }
98. if (rnd() % (a->sz + b->sz) < a->sz)
99. {
100.
              treap *ret = new treap(a \rightarrow val, a \rightarrow l, merge(a \rightarrow r, b));
101.
              recalc(ret);
102.
             return ret;
103.
104.
            else
105.
             treap *ret = new_treap(b->val, merge(a, b->l), b->r);
106.
107.
              recalc(ret);
108.
              return ret;
109.
            }
110.
          }
111.
112.
          struct q
113.
          int len, from, to;
114.
115.
           q(int len, int from, int to): len(len), from(from), to(to)
116.
117.
118.
           q()
119.
            {
120.
121.
          };
122.
123.
          void print(treap *t)
124.
          if (!t)
125.
126.
           {
127.
            return;
128.
           print(t->1);
129.
130.
           printf("%d ", t->val);
131.
           print(t->r);
132.
133.
134.
          treap *build(const vector <int> &a)
135.
136.
           if (a.empty())
137.
```

```
return 0;
138.
139.
140.
          int mid = (int) a.size() / 2;
141.
           vector <int> 1, r;
142.
           for (int i = 0; i < mid; i++) l.push back(a[i]);</pre>
143.
           for (int i = mid + 1; i < (int) a.size(); i++) r.push back(a[i]);</pre>
144.
           treap *go = new_treap(a[mid], build(l), build(r));
           recalc(go);
145.
146.
           return go;
147.
148.
149.
          treap *cur = 0, *rev = 0;
150.
151.
          treap *get_segm(treap *t, int 1, int r)
152.
            1++, r++;
153.
          1++, r++,

auto a = split(t, r);
154.
155.
           auto b = split(a.first, l - 1);
156.
           return b.second;
157.
158.
159.
          treap *to copy(treap *t, int 1, int r, treap *go)
160.
            1++, r++;
161.
          auto a = split(t, r);
auto b = split(a.first, 1 - 1);
162.
163.
164.
            return merge(b.first, merge(go, a.second));
165.
```

```
6) Строки
  6.1) Суфф. автомат
9 int last = 0, sz = 0;
10
  struct state{
11
12
       int len;
13
14
       int link;
15
       map < char, int > g;
16
17 };
18
   vector < state > st(3e5);
19
20
21 void add(char c) {
22
       int now = sz++;
23
       st[now].len = st[last].len + 1;
25
       int p = last;
26
       last = now;
27
       while (p != -1 \&\& !st[p].g.count(c)) {
28
            st[p].g[c] = now;
29
            p = st[p].link;
       }
30
       if (p == -1) {
32
            st[now].link = 0;
            return;
34
       }
       int q = st[p].g[c];
35
       if (st[p].len + 1 == st[q].len) {
36
37
            st[now].link = q;
38
            return;
39
       }
       int clone = sz++;
40
       st[clone] = st[q];
41
       st[clone].len = st[p].len + 1;
43
       while (p != -1 \&\& st[p].g[c] == q) {
44
            st[p].g[c] = clone;
            p = st[p].link;
45
46
47
       st[q].link = st[now].link = clone;
48
49
   }
50
51
   bool check(string& s) {
52
53
       int now = 0;
54
       int n = s.size();
       for (int i = 0; i < n; i++) {
            if ('A' <= s[i] && s[i] <= 'Z') {</pre>
56
57
                s[i] = 'a' + (s[i] - 'A');
58
            }
            if (!st[now].g.count(s[i])) {
60
                return false;
            }
61
            now = st[now].g[s[i]];
63
64
       return true;
65 }
```

```
6.2) Cyфф. масс + lcp
        vector<int> a(n);
        vector<int> color(n);
95
96
        vector<int> head(n);
97
        vector<int> newa(n);
98
        vector<int> newcolor(n);
        vector<pair<char, int>> fs(n);
        for (int i = 0; i < n; i++) {
100
             fs[i] = {s[i], i};
101
103
        sort(fs.begin(), fs.end());
104
        head[0] = 0;
105
        int cnt = 0;
106
        char pr = fs[0].ft;
107
        for (int i = 0; i < n; i++) {
108
            a[i] = fs[i].sc;
109
             if(fs[i].ft == pr) {
                 color[a[i]] = cnt;
110
            }
111
112
            else {
113
                 cnt++;
114
                 color[a[i]] = cnt;
115
                 head[cnt] = i;
             }
116
            pr = fs[i].ft;
117
118
        }
        int k = 1;
119
        while(k < n) {</pre>
120
             for (int i = 0; i < n; i++) {</pre>
121
                 int st = (a[i] - k + 2 * n) % n;
122
123
                 newa[head[color[st]]] = st;
124
                 head[color[st]]++;
125
            head[0] = 0;
126
            newcolor[newa[0]] = 0;
127
             for (int i = 1; i < n; i++) {
128
                 int st1 = (newa[i - 1] + k) \% n, st2 = (newa[i] + k) \% n;
129
130
                 if(color[newa[i]] == color[newa[i - 1]] && color[st1] == color[st2]) {
131
                     newcolor[newa[i]] = newcolor[newa[i - 1]];
132
                 }
133
                 else {
134
                     newcolor[newa[i]] = newcolor[newa[i - 1]] + 1;
135
                     head[newcolor[newa[i]]] = i;
136
137
            }
138
             a = newa;
139
            color = newcolor;
140
             k *= 2;
141
         vector<int> obr(n);
142
         vector<int> lcp(n);
143
144
         for (int i = 0; i < n; i++) {
145
              obr[a[i]] = i;
146
147
         int maxlcp = 0;
         for (int i = 0; i < n; i++) {
148
149
              int i1 = obr[i];
150
              if(i1 == n - 1) {
151
                   maxlcp = 0;
                   lcp[i1] = 0;
152
                   continue;
153
154
              }
              int i2 = a[i1 + 1];
155
156
              while(s[i + maxlcp] == s[i2 + maxlcp])
157
                   maxlcp++;
158
              lcp[i1] = maxlcp;
159
              maxlcp = max(0, maxlcp - 1);
         }
160
```

7) Оптимизация динамики

```
7.1) Ли Чао
26 struct line {
27
      int k, b;
28
       int get(int x) {
29
           return k * x + b;
30
31
       bool operator < (const line& x) const {</pre>
32
           return k < x.k;</pre>
33
34 };
35
   struct node {
36
37
      node* 1;
38
       node* r;
       line now;
       node(node* _1, node* _r, line _now) : 1(_1), r(_r), now(_now) {}
40
41 };
42
43 int get(node* v, int l, int r, int x) {
       int ans = v->now.get(x);
       int m = (1 + r) / 2;
45
       if(m <= x) {
46
           if(v->r == NULL)
47
48
               return ans;
49
50
               return min(ans, get(v->r, m, r, x));
51
      }
52
      else {
           if(v->1 == NULL)
53
               return ans;
55
           else
               return min(ans, get(v->1, 1, m, x));
56
57
58 }
61 void upd(node* v, int l, int r, line val) {
       int m = (1 + r) / 2;
62
63
       bool lft = val.get(1) < v->now.get(1);
64
       bool mid = val.get(m) < v->now.get(m);
       if(mid)
65
           swap(v->now, val);
       if(r - 1 == 1)
           return;
69
       if(lft == mid) {
           if(v->r == NULL) {
70
               auto neww = new node(NULL, NULL, {0, infx});
71
72
                v->r = neww;
73
            }
74
            upd(v->r, m, r, val);
75
            return;
       }
76
77
       else {
78
           if(v->1 == NULL) {
                auto neww = new node(NULL, NULL, {0, infx});
79
80
                v->1 = neww;
            }
81
            upd(v->1, 1, m, val);
82
83
            return;
84
       }
85 }
```

7.2) Лямбда оптимизация

Рассмотрим следующую задачу: Дан массив a_1, a_2, \dots, a_n . Надо разбить его на k отрезков так, чтобы минимизировать сумму квадратов сумм отрезков, $k \le n$.

Сведем задачу к другой — вместо того, чтобы набирать k отрезков, придумаем разбиение на сколько-нибудь отрезков, но за каждый отрезок будем добавлять к функции ответа штраф, равный λ . Пусть для λ_1 количество отрезков в разбиении равно k_1 . Тогда для $\lambda_2 < \lambda_1 \ k_2 \ge k_1$. Интуиция у этого факта такая — если мы за каждый отрезок платим меньше, то мы можем взять больше отрезков. Понятно, что если данное неравенство не выполняется, то лямбда-оптимизация не работает.

Поскольку зависимость между k и λ монотонная, то можно сделать бинарный поиск по λ и найти такую, при которой $k_\lambda=k$. Новую задачу можно решать при помощи convex hull trick, потому что

$$dp_i = \min_{j=1}^{i-1} dp[j] + \lambda + pref_i^2 - (2 \cdot pref_j) \cdot pref_i + pref_j^2$$

где *pref* — массив префиксных сумм.

```
1 pair<int, int> calc(int X) {
     vector<pair<int, int>> dp(n + 1);
 3
          add(0, 0);
      for (int i = 1; i <= n; i++) {
 4
          int argmin = get(pref[i]);
          dp[i].first = dp[argmin].first + X + pref[i] * pref[i] - 2 * pref[argmin] * pref[i] + pref
 6
           dp[i].second = dp[argmin].second + 1;
 7
 8
           add(-2 * pref[i], dp[i] + pref[i] * pref[i])
 9
10
      return dp[n];
11 }
12 /*
13 .
14 .
15 .
16 */
18 int L = -INF;
19 int R = INF;
20 while (L + 1 < R) {
21
      int mid = (L + R) / 2;
       pair<int, int> X = calc(mid);
22
23
      if (X.second > k) {
24
          R = mid;
25
      }
26
      else {
27
          L = mid;
28
29 }
30 pair<int, int> res = check(R);
31 cout << res.first - k * R;</pre>
```