南京大学 ACM-ICPC 集训队代码模版库

Contents

1 General

1.1 Code library checksum

```
c502 import re, sys, hashlib
427e
b41f def digest_line(s):
d74e     return hashlib.md5(re.sub(r'\s|//.*', '', s)).hexdigest()[-4:]
427e
f7db for line in sys.stdin.read().strip().split("\n"):
f335     print digest_line(line), line
```

1.2 .vimrc

```
914c set nocompatible
733d syntax on
6bbc colorscheme slate
7db5 set number
b0e3 set cursorline
061b set shiftwidth=2
8011 set softtabstop=2
a66d set tabstop=2
d23a set expandtab
5245 set magic
740c set smartindent
bee8 set backspace=indent,eol,start
815d set cmdheight=1
0a40 set laststatus=2
e458 set statusline=\ %<%F[%1*%M%*%n%R%H]%=\ %y\ %0(%{&fileformat}\ %{&encoding}\ %c
       :%l/%L%)\
1c67 set whichwrap=b,s,<,>,[,]
```

1.3 Template

```
302f #include <bits/stdc++.h>
421c using namespace std;
427e
426f #ifdef __LOCAL_DEBUG__
3507 # define _debug(fmt, ...) fprintf(stderr, "\033[94m%s:␣" fmt "\n\033[0m", \
611f     __func__, ##__VA_ARGS__)
a8cb #else
e6b5 # define _debug(...) ((void) 0)
1937 #endif
0d6c #define rep(i, n) for (int i=0; i<(n); i++)
cfe3 #define Rep(i, n) for (int i=1; i<=(n); i++)
8843 #define range(x) (x).begin(), (x).end()
5cad typedef long long LL;
b773 typedef unsigned long long ULL;
427e
5120 template <unsigned p>
87b8 struct Zp{
7797     unsigned x;
ff67     Zp(unsigned x):x(x){}
22e3     operator unsigned(){return x;}
fecc     Zp operator ^ (ULL e) {
4fce         Zp b=x, r=1;
3e90         while (e) {
5421             if (e&1) r=r*b;
2059             b=b*b;
16fc             e>>=1;
95cf         }
547e         return r;
95cf     }
a2f5     Zp operator + (Zp rhs) {return (x+rhs)%p;}
664b     Zp operator - (Zp rhs) {return (x+p-rhs)%p;}
3ec4     Zp operator * (Zp rhs) {return x*rhs%p;}
7cfd     Zp operator / (Zp rhs) {return Zp(x)*(rhs^(p-2));}
329b };
427e
370f typedef Zp<1000000007> zp;
427e
0795 zp operator"" _ (ULL n){return n;}
```

2 Miscellanous Algorithms

2.1 Fast fourier transform

```
4e09 const int NMAX = 1<<20;
427e
3fbf typedef complex<double> cplx;
427e
abd1 const double PI = 2*acos(0.0);
12af struct FFT{
```

```
c47c     int rev[NMAX];
27d7     cplx omega[NMAX], oinv[NMAX];
9827     int K, N;
427e
1442     FFT(int k){
e209         K = k; N = 1 << k;
b393         rep (i, N){
7ba3             rev[i] = (rev[i>>1]>>1) | ((i&1)<<(K-1));
1908             omega[i] = polar(1.0, 2.0 * PI / N * i);
a166             oinv[i] = conj(omega[i]);
95cf         }
95cf     }
427e
b941     void dft(cplx* a, cplx* w){
a215         rep (i, N) if (i < rev[i]) swap(a[i], a[rev[i]]);
ac6e         for (int l = 2; l <= N; l *= 2){
2969             int m = l/2;
b3cf             for (cplx* p = a; p != a + N; p += l)
c24f                 rep (k, m){
fe06                     cplx t = w[N/l*k] * p[k+m];
ecbf                     p[k+m] = p[k] - t; p[k] += t;
95cf                 }
95cf         }
95cf     }
427e
617b     void fft(cplx* a){dft(a, omega);}
a123     void ifft(cplx* a){
3b2f         dft(a, oinv);
57fc         rep (i, N) a[i] /= N;
95cf     }
427e
bdc0     void conv(cplx* a, cplx* b){
6497         fft(a); fft(b);
12a5         rep (i, N) a[i] *= b[i];
f84e         ifft(a);
95cf     }
329b };
```

2.2 2-SAT

```
0f42 const int MAXN = 100005;
03a9 struct twoSAT{
5c83     int n;
8f72     vector<int> G[MAXN*2];
d060     bool mark[MAXN*2];
b42d     int S[MAXN*2], c;
427e
d34f     void init(int n){
b985         this->n = n;
f9ec         for (int i=0; i<n*2; i++) G[i].clear();
0609         memset(mark, 0, sizeof(mark));
95cf     }
427e
3bd5     bool dfs(int x){
bd70         if (mark[x^1]) return false;
c96a         if (mark[x]) return true;
fd23         mark[x] = true;
4bea         S[c++] = x;
1ce6         for (int i=0; i<G[x].size(); i++)
d942             if (!dfs(G[x][i])) return false;
3361         return true;
95cf     }
427e
5894     void add_clause(int x, bool xval, int y, bool yval){
6afe         x = x * 2 + xval;
e680         y = y * 2 + yval;
81cc         G[x^1].push_back(y);
6835         G[y^1].push_back(x);
95cf     }
427e
d0cb     bool solve() {
7c39         for (int i=0; i<n*2; i+=2){
e63f             if (!mark[i] && !mark[i+1]){
88fb                 c = 0;
f4b9                 if (!dfs(i)){
3f03                     while (c > 0) mark[S[--c]] = false;
86c5                     if (!dfs(i+1)) return false;
95cf                 }
95cf             }
95cf         }
3361         return true;
95cf     }
427e
5f0a     inline bool value(unsigned i){return mark[2*i+1];}
329b };
```

2.3 Knuth's optimization

```
5c83  int n;
d77c  int dp[256][256], dc[256][256];
427e
b7ec  template <typename T>
0bc7  void compute(T cost) {
0423    for (int i = 0; i <= n; i++) {
8f5e      dp[i][i] = 0;
9488      dc[i][i] = i;
95cf    }
be8e    rep (i, n) {
95b5      dp[i][i+1] = 0;
aa0f      dc[i][i+1] = i;
95cf    }
ec08    for (int len = 2; len <= n; len++) {
88b8      for (int i = 0; i + len <= n; i++) {
d3da        int j = i + len;
9824        int lbnd = dc[i][j-1], rbnd = dc[i+1][j];
a24a        dp[i][j] = INT_MAX / 2;
f933        int c = cost(i, j);
90d2        for (int k = lbnd; k <= rbnd; k++) {
9bd0          int res = dp[i][k] + dp[k][j] + c;
26b5          if (res < dp[i][j]) {
e6af            dp[i][j] = res;
9c88            dc[i][j] = k;
95cf          }
95cf        }
95cf      }
95cf    }
329b  };
```

3 String

3.1 Knuth-Morris-Pratt algorithm

```
2836  const int SIZE = 10005;
9847  int fail[SIZE];
57b7  int len;
427e
182f  void construct(const char* p) {
aaa1    len = strlen(p);
3dd4    fail[0] = fail[1] = 0;
d8a8    for (int i = 1; i < len; i++) {
147f      int j = fail[i];
3c79      while (j && p[i] != p[j]) j = fail[j];
4643      fail[i + 1] = p[i] == p[j] ? j + 1 : 0;
95cf    }
95cf  }
427e
c464  inline void found(int pos) {
427e    // ! add codes for having found at pos
95cf  }
427e
1932  void match(const char* t, const char* p) {  // must be called after construct
8482    int n = strlen(t);
8fd0    int j = 0;
be8e    rep(i, n) {
4e19      while (j && p[j] != t[i]) j = fail[j];
b5d5      if (p[j] == t[i]) j++;
f024      if (j == len) found(i - len + 1);
95cf    }
95cf  }
```

3.2 Manacher algorithm

```
81d4  struct Manacher {
cd09    int Len;
9255    vector<int> lc;
b301    string s;
427e
ec07    void work() {
c033      lc[1] = 1;
6bef      int k = 1;
427e
491f      for (int i = 2; i <= Len; i++) {
7957        int p = k + lc[k] - 1;
5e04        if (i <= p) {
24a1          lc[i] = min(lc[2 * k - i], p - i + 1);
8e2e        } else {
e0e5          lc[i] = 1;
95cf        }
74ff        while (s[i + lc[i]] == s[i - lc[i]]) lc[i]++;
2b9a        if (i + lc[i] > k + lc[k]) k = i;
95cf      }
```

```
95cf    }
427e
bfd5    void init(const char *tt) {
aaaf      int len = strlen(tt);
f701      s.resize(len * 2 + 10);
7045      lc.resize(len * 2 + 10);
8e13      s[0] = '*';
ae54      s[1] = '#';
1321      for (int i = 0; i < len; i++) {
e995        s[i * 2 + 2] = tt[i];
69fd        s[i * 2 + 1] = '#';
95cf      }
43fd      s[len * 2 + 1] = '#';
75d1      s[len * 2 + 2] = '\0';
61f7      Len = len * 2 + 2;
3e7a      work();
95cf    }
427e
b194    pair<int, int> maxpal(int l, int r) {
901a      int center = l + r + 1;
ffb2      int rad = lc[center] / 2;
ab54      int rmid = (l + r + 1) / 2;
17e4      int rl = rmid - rad, rr = rmid + rad - 1;
3908      if ((r ^ l) & 1) {
69f3      } else rr++;
69dc      return {max(l, rl), min(r, rr)};
95cf    }
329b  };
```

3.3 Aho-corasick automaton

```
a1ad  struct AC : Trie {
9143    int fail[MAXN];
daca    int last[MAXN];
427e
8690    void construct() {
93d2      queue<int> q;
a7a6      fail[0] = 0;
ce3c      rep(c, CHARN) {
b1c6        if (int u = tr[0][c]) {
a506          fail[u] = 0;
3e14          q.push(u);
f689          last[u] = 0;
95cf        }
95cf      }
cc78      while (!q.empty()) {
31f0        int r = q.front();
15dd        q.pop();
ce3c        rep(c, CHARN) {
ab59          int u = tr[r][c];
0ef5          if (!u) {
9d58            tr[r][c] = tr[fail[r]][c];
b333            continue;
95cf          }
3e14          q.push(u);
b3ff          int v = fail[r];
d2ea          while (v && !tr[v][c]) v = fail[v];
c275          fail[u] = tr[v][c];
654c          last[u] = tag[fail[u]] ? fail[u] : last[fail[u]];
95cf        }
95cf      }
95cf    }
427e
7752    void found(int pos, int j) {
043e      if (j) {
427e        // ! add codes for having found word with tag[j]
4a96        found(pos, last[j]);
95cf      }
95cf    }
427e
9785    void find(const char* text) {  // must be called after construct()
80a4      int p = 0, c, len = strlen(text);
9c94      rep(i, len) {
b3db        c = id(text[i]);
f119        p = tr[p][c];
f08e        if (tag[p])
389b          found(i, p);
1e67        else if (last[p])
299e          found(i, last[p]);
95cf      }
95cf    }
329b  };
```

3.4 Trie

```
e6f1  const int MAXN = 12000;
```

```
dd87  const int CHARN = 26;
427e
8ff5  inline int id(char c) { return c - 'a'; }
427e
a281  struct Trie {
5c83    int n;
f4f5    int tr[MAXN][CHARN];  // Trie tree, 0 denotes fail
35a5    int tag[MAXN];
427e
4fee    Trie() {
3ccc      memset(tr[0], 0, sizeof(tr[0]));
4d52      tag[0] = 0;
46bf      n = 1;
95cf    }
427e
427e    // tag should not be 0
30b0    void add(const char* s, int t) {
d50a      int p = 0, c, len = strlen(s);
9c94      rep(i, len) {
3140        c = id(s[i]);
d6c8        if (!tr[p][c]) {
26dd          memset(tr[n], 0, sizeof(tr[n]));
2e5c          tag[n] = 0;
73bb          tr[p][c] = n++;
95cf        }
f119        p = tr[p][c];
95cf      }
35ef      tag[p] = t;
95cf    }
427e
427e    // returns 0 if not found
427e    // AC automaton does not need this function
216c    int search(const char* s) {
d50a      int p = 0, c, len = strlen(s);
9c94      rep(i, len) {
3140        c = id(s[i]);
f339        if (!tr[p][c]) return 0;
f119        p = tr[p][c];
95cf      }
840e      return tag[p];
95cf    }
329b  };
```

4 Graph Theory

4.1 Strongly connected component

```
const int MAXV = 100005;                                    837c
                                                            427e
struct graph{                                               2ea0
    vector<int> adj[MAXV];                                  88e3
    stack<int> s;                                           9cad
    int V; // number of vertices                            3d02
    int pre[MAXV], lnk[MAXV], scc[MAXV];                    8b6c
    int time, sccn;                                         27ee
                                                            427e
    void add_edge(int u, int v){                            bfab
        adj[u].push_back(v);                                c71a
    }                                                       95cf
                                                            427e
    void dfs(int u){                                        d714
        pre[u] = lnk[u] = ++time;                           7e41
        s.push(u);                                          80f6
        for (int v : adj[u]){                               18f6
            if (!pre[v]){                                   173e
                dfs(v);                                     5f3c
                lnk[u] = min(lnk[u], lnk[v]);               002c
            } else if (!scc[v]){                            6068
                lnk[u] = min(lnk[u], pre[v]);               d5df
            }                                               95cf
        }                                                   95cf
        if (lnk[u] == pre[u]){                              8de2
            sccn++;                                         660f
            int x;                                          3c9e
            do {                                            a69f
                x = s.top(); s.pop();                       3834
                scc[x] = sccn;                              b0e9
            } while (x != u);                               6757
        }                                                   95cf
    }                                                       95cf
                                                            427e
    void find_scc(){                                        4c88
        time = sccn = 0;                                    f4a2
        memset(scc, 0, sizeof scc);                         8de7
        memset(pre, 0, sizeof pre);                         8c2f
        Rep (i, V){                                         6901
```

```
56d1             if (!pre[i]) dfs(i);
95cf         }
95cf     }
427e
27ce     vector<int> adjc[MAXV];
364d     void contract(){
1a1e         Rep (i, V)
21a2             rep (j, adj[i].size()){
b730                 if (scc[i] != scc[adj[i][j]])
b46e                     adjc[scc[i]].push_back(scc[adj[i][j]]);
95cf             }
95cf     }
329b };
```

4.2 Vertex biconnected component

```
0f42 const int MAXN = 100005;
2ea0 struct graph {
33ae     int pre[MAXN], iscut[MAXN], bccno[MAXN], dfs_clock, bcc_cnt;
848f     vector<int> adj[MAXN], bcc[MAXN];
6b06     set<pair<int, int>> bcce[MAXN];
427e
76f7     stack<pair<int, int>> s;
427e
bfab     void add_edge(int u, int v) {
c71a         adj[u].push_back(v);
a717         adj[v].push_back(u);
95cf     }
427e
7d3c     int dfs(int u, int fa) {
9fe6         int lowu = pre[u] = ++dfs_clock;
ec14         int child = 0;
18f6         for (int v : adj[u]) {
173e             if (!pre[v]) {
e7f8                 s.push({u, v});
fdcf                 child++;
f851                 int lowv = dfs(v, u);
189c                 lowu = min(lowu, lowv);
b687                 if (lowv >= pre[u]) {
6323                     iscut[u] = 1;
57eb                     bcc[bcc_cnt].clear();
90b8                     bcce[bcc_cnt].clear();
a147                     while (1) {
a6a3                         int xu, xv;
a0c3                         tie(xu, xv) = s.top(); s.pop();
0ef5                         bcce[bcc_cnt].insert({min(xu, xv), max(xu, xv)});
3db2                         if (bccno[xu] != bcc_cnt) {
e0db                             bcc[bcc_cnt].push_back(xu);
d27f                             bccno[xu] = bcc_cnt;
95cf                         }
f357                         if (bccno[xv] != bcc_cnt) {
752b                             bcc[bcc_cnt].push_back(xv);
57c9                             bccno[xv] = bcc_cnt;
95cf                         }
7096                         if (xu == u && xv == v) break;
95cf                     }
03f5                     bcc_cnt++;
95cf                 }
7470             } else if (pre[v] < pre[u] && v != fa) {
e7f8                 s.push({u, v});
f115                 lowu = min(lowu, pre[v]);
95cf             }
95cf         }
e104         if (fa < 0 && child == 1) iscut[u] = 0;
1160         return lowu;
95cf     }
427e
17be     void find_bcc(int n) {
8c2f         memset(pre, 0, sizeof pre);
e2d2         memset(iscut, 0, sizeof iscut);
40d3         memset(bccno, -1, sizeof bccno);
fae2         dfs_clock = bcc_cnt = 0;
5c63         rep (i, n) if (!pre[i]) dfs(i, -1);
95cf     }
329b };
```

4.3 Maximum flow (Dinic)

```
bcf8 struct edge{
60e2     int from, to;
5e6d     LL cap, flow;
329b };
427e
e2cd const int MAXN = 1005;
9062 struct Dinic {
4dbf     int n, m, s, t;
```

```
9f0c    vector<edge> edges;
b891    vector<int> G[MAXN];
bbb6    bool vis[MAXN];
b40a    int d[MAXN];
ddec    int cur[MAXN];
427e
5973    void add_edge(int from, int to, LL cap) {
7b55        edges.push_back(edge{from, to, cap, 0});
1db7        edges.push_back(edge{to, from, 0, 0});
fe77        m = edges.size();
dff5        G[from].push_back(m-2);
8f2d        G[to].push_back(m-1);
95cf    }
427e
1836    bool bfs() {
3b73        memset(vis, 0, sizeof(vis));
93d2        queue<int> q;
5d13        q.push(s);
2cd2        vis[s] = 1;
721d        d[s] = 0;
cc78        while (!q.empty()) {
66ba            int x = q.front(); q.pop();
3b61            for (int i = 0; i < G[x].size(); i++) {
b510                edge& e = edges[G[x][i]];
bba9                if (!vis[e.to] && e.cap > e.flow) {
cd72                    vis[e.to] = 1;
cf26                    d[e.to] = d[x] + 1;
ca93                    q.push(e.to);
95cf                }
95cf            }
95cf        }
b23b        return vis[t];
95cf    }
427e
9252    LL dfs(int x, LL a) {
6904        if (x == t || a == 0) return a;
8bf9        LL flow = 0, f;
f515        for (int& i = cur[x]; i < G[x].size(); i++) {
b510            edge& e = edges[G[x][i]];
2374            if(d[x] + 1 == d[e.to] && (f = dfs(e.to, min(a, e.cap-e.flow))) > 0)
                    {
1cce                e.flow += f;
e16d                edges[G[x][i]^1].flow -= f;
a74d                flow += f;
23e5                a -= f;
97ed                if(a == 0) break;
95cf            }
95cf        }
84fb        return flow;
95cf    }
427e
5bf2    LL max_flow(int s, int t) {
590d        this->s = s; this->t = t;
62e2        LL flow = 0;
ed58        while (bfs()) {
f326            memset(cur, 0, sizeof(cur));
fb3a            flow += dfs(s, LLONG_MAX);
95cf        }
84fb        return flow;
95cf    }
427e
c72e    vector<int> min_cut() { // call this after maxflow
1df9        vector<int> ans;
df9a        for (int i = 0; i < edges.size(); i++) {
56d8            edge& e = edges[i];
46a2            if(vis[e.from] && !vis[e.to] && e.cap > 0) ans.push_back(i);
95cf        }
4206        return ans;
95cf    }
329b};
```

4.4 Maximum cardinality bipartite matching (Hungarian)

```
302f#include <bits/stdc++.h>
421cusing namespace std;
427e
0d6c#define rep(i, n) for (int i = 0; i < (n); i++)
cfe3#define Rep(i, n) for (int i = 1; i <= (n); i++)
8843#define range(x) (x).begin(), (x).end()
5cadtypedef long long LL;
427e
84eestruct Hungarian{
fbf6    int nx, ny;
9ec6    vector<int> mx, my;
9d4c    vector<vector<int> > e;
edec    vector<bool> mark;
427e
```

```
8324     void init(int nx, int ny){
c1d1         this->nx = nx;
f9c1         this->ny = ny;
ac92         mx.resize(nx); my.resize(ny);
3f11         e.clear(); e.resize(nx);
1023         mark.resize(nx);
95cf     }
427e
4589     inline void add(int a, int b){
486c         e[a].push_back(b);
95cf     }
427e
0c2b     bool augment(int i){
207c         if (!mark[i]) {
dae4             mark[i] = true;
6a1e             for (int j : e[i]){
0892                 if (my[j] == -1 || augment(my[j])){
9ca3                     mx[i] = j; my[j] = i;
3361                     return true;
95cf                 }
95cf             }
95cf         }
438e         return false;
95cf     }
427e
3fac     int match(){
5b57         int ret = 0;
b0f1         fill(range(mx), -1);
b957         fill(range(my), -1);
4ed1         rep (i, nx){
13a5             fill(range(mark), false);
cc89             if (augment(i)) ret++;
95cf         }
ee0f         return ret;
95cf     }
329b };
```

4.5 Minimum cost maximum flow

```
bcf8 struct edge{
60e2     int from, to;
d698     int cap, flow;
32cc     LL cost;
};

const LL INF = LLONG_MAX / 2;
const int MAXN = 5005;
struct MCMF {
    int s, t, n, m;
    vector<edge> edges;
    vector<int> G[MAXN];
    bool inq[MAXN]; // queue
    LL d[MAXN];     // distance
    int p[MAXN];    // previous
    int a[MAXN];    // improvement

    void add_edge(int from, int to, int cap, LL cost) {
        edges.push_back(edge{from, to, cap, 0, cost});
        edges.push_back(edge{to, from, 0, 0, -cost});
        m = edges.size();
        G[from].push_back(m-2);
        G[to].push_back(m-1);
    }

    bool spfa(){
        queue<int> q;
        fill(d, d + MAXN, INF); d[s] = 0;
        memset(inq, 0, sizeof(inq));
        q.push(s); inq[s] = true;
        p[s] = 0; a[s] = INT_MAX;
        while (!q.empty()){
            int u = q.front(); q.pop(); inq[u] = false;
            rep (i, G[u].size()){
                edge& e = edges[G[u][i]];
                if (e.cap > e.flow && d[e.to] > d[u] + e.cost){
                    d[e.to] = d[u] + e.cost;
                    p[e.to] = G[u][i];
                    a[e.to] = min(a[u], e.cap - e.flow);
                    if (!inq[e.to]) q.push(e.to), inq[e.to] = true;
                }
            }
        }
        return d[t] != INF;
    }

    void augment(){
        int u = t;
```

```
b19d         while (u != s){
db09             edges[p[u]].flow += a[t];
25a9             edges[p[u]^1].flow -= a[t];
e6c9             u = edges[p[u]].from;
95cf         }
95cf     }
427e
6e20 #ifdef GIVEN_FLOW
5972     bool min_cost(int s, int t, int f, LL& cost) {
590d         this->s = s; this->t = t;
21d4         int flow = 0;
23cb         cost = 0;
22dc         while (spfa()) {
bcdb             augment();
a671             if (flow + a[t] >= f){
9c87                 cost += (f - flow) * a[t]; flow = f;
3361                 return true;
8e2e             } else {
2a83                 flow += a[t]; cost += a[t] * d[t];
95cf             }
95cf         }
438e         return false;
95cf     }
a8cb #else
f9a9     int min_cost(int s, int t, LL& cost) {
590d         this->s = s; this->t = t;
21d4         int flow = 0;
23cb         cost = 0;
22dc         while (spfa()) {
bcdb             augment();
2a83             flow += a[t]; cost += a[t] * d[t];
95cf         }
84fb         return flow;
95cf     }
1937 #endif
329b };
```

4.6 Global minimum cut (Stoer-Wagner)

```
f9d7 typedef vector<LL> VI;
045e typedef vector<VI> VVI;
427e
f012 pair<LL, VI> stoer(VVI &w) {
     int n = w.size();                                                    66f7
     VI used(n), c, bestc;                                                4d98
     LL bestw = -1;                                                       329d
                                                                          427e
     for (int ph = n - 1; ph >= 0; ph--) {                                cd21
         VI wt = w[0], added = used;                                      ec6e
         int prev, last = 0;                                              f20e
         rep (i, ph) {                                                    4b32
             prev = last;                                                 8bfc
             last = -1;                                                   0706
             for (int j = 1; j < n; j++)                                  4942
                 if (!added[j] && (last == -1 || wt[j] > wt[last]))       c4b9
                     last = j;                                            887d
             if (i == ph - 1) {                                           71bc
                 rep (j, n) w[prev][j] += w[last][j];                     9cfa
                 rep (j, n) w[j][prev] = w[prev][j];                      1f25
                 used[last] = true;                                       5613
                 c.push_back(last);                                       8e11
                 if (bestw == -1 || wt[last] < bestw) {                   bb8e
                     bestc = c;                                           bab6
                     bestw = wt[last];                                    372e
                 }                                                        95cf
             } else {                                                     8e2e
                 rep (j, n) wt[j] += w[last][j];                          caeb
                 added[last] = true;                                      8b92
             }                                                            95cf
         }                                                                95cf
     }                                                                    95cf
     return {bestw, bestc};                                               038c
}                                                                         95cf
```

4.7 Heavy-light decomposition

```
const int MAXN = 100005;                                                  0f42
vector<int> adj[MAXN];                                                    0b32
int sz[MAXN], top[MAXN], fa[MAXN], son[MAXN], depth[MAXN], id[MAXN];      42f2
                                                                          427e
void dfs1(int x, int dep, int par){                                       be5c
    depth[x] = dep;                                                       7489
    sz[x] = 1;                                                            2ee7
    fa[x] = par;                                                          adb4
    int maxn = 0, s = 0;                                                  b79d
    for (int c: adj[x]){                                                  c861
```

```
fe45         if (c == par) continue;
fd2f         dfs1(c, dep + 1, x);
b790         sz[x] += sz[c];
f0f1         if (sz[c] > maxn){
c749             maxn = sz[c];
fe19             s = c;
95cf         }
95cf     }
0e08     son[x] = s;
95cf }
427e
ba54 int cid = 0;
3644 void dfs2(int x, int t){
8d96     top[x] = t;
d314     id[x] = ++cid;
c4a1     if (son[x]) dfs2(son[x], t);
c861     for (int c: adj[x]){
9881         if (c == fa[x]) continue;
5518         if (c == son[x]) continue;
13f9         else dfs2(c, c);
95cf     }
95cf }
427e
0f04 void decomp(int root){
9fa4     dfs1(root, 1, 0);
1c88     dfs2(root, root);
95cf }
427e
2c98 void query(int u, int v){
03a1     while (top[u] != top[v]){
45ec         if (depth[top[u]] < depth[top[v]]) swap(u, v);
427e         // id[top[u]] to id[u]
005b         u = fa[top[u]];
95cf     }
6083     if (depth[u] > depth[v]) swap(u, v);
427e     // id[u] to id[v]
95cf }
```

5 Data Structures

5.1 Segment tree

```
LL p;                                                          3942
const int MAXN = 4 * 100006;                                   1ebb
struct segtree {                                               451a
  int l[MAXN], m[MAXN], r[MAXN];                               27be
  LL val[MAXN], tadd[MAXN], tmul[MAXN];                        4510
                                                               427e
#define lson (o<<1)                                            ac35
#define rson (o<<1|1)                                          1294
                                                               427e
  void pull(int o) {                                           1344
    val[o] = (val[lson] + val[rson]) % p;                      bbe9
  }                                                            95cf
                                                               427e
  void push_add(int o, LL x) {                                 e4bc
    val[o] = (val[o] + x * (r[o] - l[o])) % p;                 5dd6
    tadd[o] = (tadd[o] + x) % p;                               6eff
  }                                                            95cf
                                                               427e
  void push_mul(int o, LL x) {                                 d658
    val[o] = val[o] * x % p;                                   b82c
    tadd[o] = tadd[o] * x % p;                                 aa86
    tmul[o] = tmul[o] * x % p;                                 649f
  }                                                            95cf
                                                               427e
  void push(int o) {                                           b149
    if (l[o] == m[o]) return;                                  3159
    if (tmul[o] != 1) {                                        0a90
      push_mul(lson, tmul[o]);                                 0f4a
      push_mul(rson, tmul[o]);                                 045e
      tmul[o] = 1;                                             ac0a
    }                                                          95cf
    if (tadd[o]) {                                             1b82
      push_add(lson, tadd[o]);                                 9547
      push_add(rson, tadd[o]);                                 0e73
      tadd[o] = 0;                                             6234
    }                                                          95cf
  }                                                            95cf
                                                               427e
  void build(int o, int ll, int rr) {                          471c
    int mm = (ll + rr) / 2;                                    0e87
    l[o] = ll; r[o] = rr; m[o] = mm;                           9d27
    tmul[o] = 1;                                               ac0a
    if (ll == mm) {                                            5c92
      scanf("%lld", val + o);                                  001f
```

```
e5b6       val[o] %= p;
8e2e     } else {
7293       build(lson, ll, mm);
5e67       build(rson, mm, rr);
ba26       pull(o);
95cf     }
95cf   }
427e
4406   void add(int o, int ll, int rr, LL x) {
3c16     if (ll <= l[o] && r[o] <= rr) {
db32       push_add(o, x);
8e2e     } else {
c4b0       push(o);
4305       if (m[o] > ll) add(lson, ll, rr, x);
d5a6       if (m[o] < rr) add(rson, ll, rr, x);
ba26       pull(o);
95cf     }
95cf   }
427e
48cd   void mul(int o, int ll, int rr, LL x) {
3c16     if (ll <= l[o] && r[o] <= rr) {
e7d0       push_mul(o, x);
8e2e     } else {
c4b0       push(o);
d1ba       if (ll < m[o]) mul(lson, ll, rr, x);
67f3       if (m[o] < rr) mul(rson, ll, rr, x);
ba26       pull(o);
95cf     }
95cf   }
427e
0f62   LL query(int o, int ll, int rr) {
3c16     if (ll <= l[o] && r[o] <= rr) {
6dfe       return val[o];
8e2e     } else {
f7ff       LL ans = 0;
c4b0       push(o);
c5f8       if (m[o] > ll) ans += query(lson, ll, rr);
ef81       if (m[o] < rr) ans += query(rson, ll, rr);
a420       return ans % p;
95cf     }
95cf   }
4d99 } seg;
```

5.2 Link/cut tree

```
// about 0.13s per 100k ops @luogu.org                                          427e
                                                                                427e
namespace LCT {                                                                 ed4d
  const int MAXN = 300005;                                                      5ece
  int fa[MAXN], ch[MAXN][2], val[MAXN], sum[MAXN];                              6a6d
  bool rev[MAXN];                                                               c6e1
                                                                                427e
  bool isroot(int x) {                                                          7839
    return ch[fa[x]][0] == x || ch[fa[x]][1] == x;                              45a9
  }                                                                             95cf
                                                                                427e
  void pull(int x) {                                                            3bf9
    sum[x] = val[x] ^ sum[ch[x][0]] ^ sum[ch[x][1]];                            6664
  }                                                                             95cf
                                                                                427e
  void reverse(int x) {                                                         3698
    swap(ch[x][0], ch[x][1]);                                                   7850
    rev[x] ^= 1;                                                                52c6
  }                                                                             95cf
                                                                                427e
  void push(int x) {                                                            1a53
    if (rev[x]) {                                                               8f1f
      if (ch[x][0]) reverse(ch[x][0]);                                          ebf3
      if (ch[x][1]) reverse(ch[x][1]);                                          6eb0
      rev[x] = 0;                                                               8fc1
    }                                                                           95cf
  }                                                                             95cf
                                                                                427e
  void rotate(int x) {                                                          425f
    int y = fa[x], z = fa[y], k = ch[y][1] == x, w = ch[x][!k];                 51af
    if (isroot(y)) ch[z][ch[z][1] == y] = x;                                    e1fe
    ch[x][!k] = y; ch[y][k] = w;                                                af46
    if (w) fa[w] = y;                                                           fa6f
    fa[y] = x; fa[x] = z;                                                       3540
    pull(y);                                                                    72ef
  }                                                                             95cf
                                                                                427e
  void pushall(int x) {                                                         bc1b
    if (isroot(x)) pushall(fa[x]);                                              a316
    push(x);                                                                    a97b
  }                                                                             95cf
                                                                                427e
```

```
f69c    void splay(int x) {
d095      int y = x, z = 0;
8ab3      pushall(y);
f244      while (isroot(x)) {
ceef        y = fa[x]; z = fa[y];
4449        if (isroot(y)) rotate((ch[y][0] == x) ^ (ch[z][0] == y) ? x : y);
cf90        rotate(x);
95cf      }
78a0      pull(x);
95cf    }
427e
6229    void access(int x) {
1548      int z = x;
ba78      for (int y = 0; x; x = fa[y = x]) {
8fec        splay(x);
b05d        ch[x][1] = y;
78a0        pull(x);
95cf      }
7afd      splay(z);
95cf    }
427e
502e    void chroot(int x) {
766a      access(x);
cb0d      reverse(x);
95cf    }
427e
471a    void split(int x, int y) {
3015      chroot(x);
29b5      access(y);
95cf    }
427e
d87a    int Root(int x) {
766a      access(x);
874d      while (ch[x][0]) {
a97b        push(x);
b83a        x = ch[x][0];
95cf      }
8fec      splay(x);
d074      return x;
95cf    }
427e
70d3    void Link(int u, int v) {  // assume unconnected before
b8a5      chroot(u);
2448      fa[u] = v;
95cf    }
427e
c2f4    void Cut(int u, int v) {  // assume connected before
e8ce      split(u, v);
fd95      fa[u] = ch[v][0] = 0;
743b      pull(v);
95cf    }
427e
6ca2    int Query(int u, int v) {
e8ce      split(u, v);
a5ba      return sum[v];
95cf    }
427e
eaba    void Update(int u, int x) {
46ce      splay(u);
1d62      val[u] = x;
95cf    }
329b  };
```

5.3 Balanced binary search tree from pb_ds

```
0475  #include <ext/pb_ds/assoc_container.hpp>
332d  using namespace __gnu_pbds;
427e
43a7  tree<int, null_type, less<int>, rb_tree_tag, tree_order_statistics_node_update>
        rkt;
427e  // null_tree_node_update
427e
427e  // SAMPLE USAGE
190e  rkt.insert(x);          // insert element
05d4  rkt.erase(x);           // erase element
add5  rkt.order_of_key(x);    // obtain the number of elements less than x
b064  rkt.find_by_order(i);   // iterator to i-th (numbered from 0) smallest element
c103  rkt.lower_bound(x);
4ff4  rkt.upper_bound(x);
b19b  rkt.join(rkt2);         // merge tree (only if their ranges do not intersect)
cb47  rkt.split(x, rkt2);     // split all elements greater than x to rkt2
```

5.4 Persistent segment tree, range k-th query

```
f1a7  struct node {
```

```
2ff6   static int n, pos;
427e
7cec   int value;
70e2   node *left, *right;
427e
20b0   void* operator new(size_t size);
427e
3dc0   static node* Build(int l, int r) {
b6c5     node* a = new node;
ce96     if (r > l + 1) {
181e       int mid = (l + r) / 2;
3ba2       a->left = Build(l, mid);
8aaf       a->right = Build(mid, r);
8e2e     } else {
bfc4       a->value = 0;
95cf     }
5ffd     return a;
95cf   }
427e
5a45   static node* init(int size) {
2c46     n = size;
7ee3     pos = 0;
be52     return Build(0, n);
95cf   }
427e
93c0   static int Query(node* lt, node *rt, int l, int r, int k) {
d30c     if (r == l + 1) return l;
181e     int mid = (l + r) / 2;
cb5a     if (rt->left->value - lt->left->value < k) {
8edb       k -= rt->left->value - lt->left->value;
2412       return Query(lt->right, rt->right, mid, r, k);
8e2e     } else {
0119       return Query(lt->left, rt->left, l, mid, k);
95cf     }
95cf   }
427e
c9ad   static int query(node* lt, node *rt, int k) {
9e27     return Query(lt, rt, 0, n, k);
95cf   }
427e
b19c   node *Inc(int l, int r, int pos) const {
5794     node* a = new node(*this);
ce96     if (r > l + 1) {
181e       int mid = (l + r) / 2;
203d       if (pos < mid)
f44a         a->left = left->Inc(l, mid, pos);
649a       else
1024         a->right = right->Inc(mid, r, pos);
95cf     }
2b3e     a->value++;
5ffd     return a;
95cf   }
427e
e80f   node *inc(int index) {
c246     return Inc(0, n, index);
95cf   }
865a } nodes[8000000];
427e
99ce int node::n, node::pos;
1987 inline void* node::operator new(size_t size) {
bb3c   return nodes + (pos++);
95cf }
```