International Institute of Information Technology - Hyderabad

lazy three

Mohammed Faisal, Harshvardhan Rana, Tanay Gad

2024-01-25

Contest (1)

template.h

8c379d, 55 lines

```
// #ifdef LOCAL
// #include "include/include.h"
// #else
// #include <bits/stdc++.h>
// #include <ext/pb_ds/assoc_container.hpp>
// #endif
// #pragma GCC target("bmi,bmi2,lzcnt,popcnt")
// #pragma GCC optimize("O2,unroll-loops")
// #pragma GCC target("avx2")

//#pragma GCC optimize("O2")
//#pragma GCC optimize("Ofast")
//#pragma GCC target("avx,avx2,fma")
using namespace std;
using namespace __gnu_pbds;
typedef tree<int, null_type, less<int>, rb_tree_tag,
↳tree_order_statistics_node_update> o_set;
// order_of_key (val): returns the no. of values less than
↳ val
// find_by_order (k): returns the kth largest element.(0-
↳based)
template <typename T>
using minHeap = priority_queue<T, vector<T>, greater<T>>;
template <typename T>
using maxHeap = priority_queue<T>;
#define int long long
#define all(s) s.begin(), s.end()
#define sz(s) (int)s.size()
#define testcases \
  cin >> tt;      \
  for (i = 1; i <= tt; i++)
#define fast                      \
  ios_base::sync_with_stdio(0); \
  cin.tie(NULL);                \
  cout.tie(NULL)
#define deb(a) cerr << #a << " = " << (a) << endl;
#define deb1(a)
↳                                                      \
  cerr << #a << " = [ ";
  ↳                                                    \
  for (auto it = a.begin(); it != a.end(); it++) cerr << *
  ↳it << " "; \
  cerr << "]" << endl;
typedef vector<int> vi;
typedef vector<vector<int>> vvi;
typedef pair<int, int> pii;
typedef vector<pair<int, int>> vpii;
typedef vector<bool> vb;
const int INF = LONG_LONG_MAX;
const int M = 1e9 + 7;

void solve(int tt) {
}

int32_t main() {
  fast;
  int tt = 1;
  int i = 1;
  testcases
      solve(i);
}
```

flags.txt

425523, 4 lines

```
## Put this in end of ~/.bashrc
alias s="g++ -Wall -Wextra -pedantic -std=c++17 -O2 -
↳Wshadow -Wformat=2 -Wfloat-equal -Wconversion -Wlogical-
↳op -Wshift-overflow=2 -Wduplicated-cond -Wcast-qual -
↳Wcast-align -D_GLIBCXX_DEBUG -D_GLIBCXX_DEBUG_PEDANTIC -
↳D_FORTIFY_SOURCE=2 -fsanitize=address -fsanitize=
↳undefined -fno-sanitize-recover -fstack-protector -
↳DLOCAL A.cpp"
```

```
alias f="g++ -std=c++17 -O2 -DLOCAL A.cpp"
alias d="g++ -std=c++17 -g -Wall -Wextra -pedantic -
↳Wshadow -Wno-unused-parameter -Wno-unused-variable -
↳Wformat=2 -Wfloat-equal -Wconversion -Wlogical-op -
↳Wshift-overflow=2 -Wduplicated-cond -Wcast-qual -Wcast-
↳align -D_GLIBCXX_DEBUG -D_GLIBCXX_DEBUG_PEDANTIC -
↳D_FORTIFY_SOURCE=2 -fsanitize=address -fsanitize=
↳undefined -fno-sanitize-recover -fstack-protector -
↳DLOCAL -fdiagnostics-color=always"
```

Mathematics (2)

Data structures (3)

SegmentTree.h

7660bf, 28 lines

```
template <typename T, typename F>
struct SegTree {
  int n;
  vector<T> t;
  const T id;
  F f;
  SegTree(const vector<T> &a, T id, F f) : n(sz(a)), t(2 *
  ↳ n), id(id), f(f) {
    for (int i = 0; i < n; i++) t[n + i] = a[i];
    for (int i = n - 1; i >= 1; i--)
      t[i] = f(t[2 * i], t[2 * i + 1]);
  }
  T query(int l, int r) {
    T resl(id), resr(id);
    for (l += n, r += n; l <= r; l >>= 1, r >>= 1) {
      if (l == r) {
        resl = f(resl, t[l]);
        break;
      }
      if (l & 1) resl = f(resl, t[l++]);
      if (!(r & 1)) resr = f(t[r--], resr);
    }
    return f(resl, resr);
  }
  void update(int v, T value) {
    for (t[v += n] = value; v >>= 1;)
      t[v] = f(t[2 * v], t[2 * v + 1]);
  }
};
```

LazySegTree.h

9376a1, 90 lines

```
template <typename T, typename U>
struct seg_tree_lazy {
  int S, H;
  T zero;
  vector<T> value;
  U noop;
  vector<bool> dirty;
  vector<U> prop;
  seg_tree_lazy(int _S, T _zero = T(), U _noop = U()) {
    zero = _zero, noop = _noop;
    for (S = 1, H = 1; S < _S;) S *= 2, H++;
    value.resize(2 * S, zero);
    dirty.resize(2 * S, false);
    prop.resize(2 * S, noop);
  }
  void set_leaves(vector<T> &leaves) {
    copy(leaves.begin(), leaves.end(), value.begin() + S);
    for (int i = S - 1; i > 0; i--)
      value[i] = value[2 * i] + value[2 * i + 1];
  }
  void apply(int i, U &update) {
    value[i] = update(value[i]);
    if (i < S) {
      prop[i] = prop[i] + update;
      dirty[i] = true;
    }
  }
  void rebuild(int i) {
```

```
    for (int l = i / 2; l; l /= 2) {
      T combined = value[2 * l] + value[2 * l + 1];
      value[l] = prop[l](combined);
    }
  }
  void propagate(int i) {
    for (int h = H; h > 0; h--) {
      int l = i >> h;

      if (dirty[l]) {
        apply(2 * l, prop[l]);
        apply(2 * l + 1, prop[l]);

        prop[l] = noop;
        dirty[l] = false;
      }
    }
  }
  void upd(int i, int j, U update) {
    i += S, j += S;
    propagate(i), propagate(j);
    for (int l = i, r = j; l <= r; l /= 2, r /= 2) {
      if ((l & 1) == 1) apply(l++, update);
      if ((r & 1) == 0) apply(r--, update);
    }
    rebuild(i), rebuild(j);
  }
  T query(int i, int j) {
    i += S, j += S;
    propagate(i), propagate(j);

    T res_left = zero, res_right = zero;
    for (; i <= j; i /= 2, j /= 2) {
      if ((i & 1) == 1) res_left = res_left + value[i++];
      if ((j & 1) == 0) res_right = value[j--] + res_right
      ↳;
    }
    return res_left + res_right;
  }
};
struct node {
  int sum, width;
  node operator+(const node &n) {
    // Change 1
    return {sum + n.sum, width + n.width};
  }
};
struct update {
  bool type;  // 0 for add, 1 for reset
  int value;
  node operator()(const node &n) {  // apply update on n
    // Change 2
    if (type)
      return {n.width * value, n.width};
    else
      return {n.sum + n.width * value, n.width};
  }
  update operator+(const update &u) {  // u is the recent
  ↳update, *this is the older update
    // Change 3
    if (u.type) return u;
    return {type, value + u.value};
  }
};
```

RMQ.h

d15d6d, 16 lines

```
template<class T>
struct RMQ {
  vector<vector<T>> jmp;
  RMQ(const vector<T>& V) : jmp(1, V) {
    for (int pw = 1, k = 1; pw * 2 <= sz(V); pw *= 2, ++k)
    ↳ {
      jmp.emplace_back(sz(V) - pw * 2 + 1);
      rep(j,0,sz(jmp[k]))
```

```cpp
        jmp[k][j] = min(jmp[k - 1][j], jmp[k - 1][j + pw])
        ↳;
      }
    }
    T query(int a, int b) {
      assert(a <= b); // tie(a, b) = minimax(a, b)
      int dep = 31 - __builtin_clz(b-a+1);
      return min(jmp[dep][a], jmp[dep][b - (1 << dep) + 1]);
    }
};
```

FenwickTree.h

6856de, 41 lines

```cpp
template <typename T>
struct Fenwick {
  vector<T> bit;
  vector<T>& original;
  Fenwick(vector<T>& _arr) : bit(_arr.size(), 0LL),
  ↳original(_arr) {
    int n = sz(_arr);
    for (int i = 0; i < n; i++) {
      bit[i] = bit[i] + _arr[i];
      if ((i | (i + 1)) < n) bit[(i | (i + 1))] = bit[(i |
      ↳ (i + 1))] + bit[i];
    }
  }
  // returns smallest index i, st. sum[0..i] >= x, returns
  ↳ -1 if no such i exists
  // returns n if x >= sum of array
  // ASSUMES NON NEGATIVE ENTRIES IN TREE
  int lower_bound(int x) {
    if (x < 0) return -1;
    if (x == 0) return 0;
    int pos = 0;
    for (int pw = 1LL << 20; pw; pw >>= 1)
      if (pw + pos <= sz(bit) and bit[pos + pw - 1] < x)
        pos += pw, x -= bit[pos - 1];
    return pos;
  }
  T query(int r) {
    assert(r < sz(bit));
    int ret = 0;
    for (r++; r > 0; r &= r - 1) ret += bit[r - 1];
    return ret;
  }
  T query(int l, int r) {
    T ret = query(r);
    if (l != 0) ret -= query(l - 1);
    return ret;
  }
  void update(int i, int x) {
    int n = bit.size();
    T diff = x - original[i];
    original[i] = x;
    for (; i < n; i = i | i + 1) bit[i] += diff;
  }
};
```

FenwickTree2d.h

deb44b, 17 lines

```cpp
const int mxn = 1000;
int grid[mxn + 1][mxn + 1];
int bit[mxn + 1][mxn + 1];
void update(int row, int col, int d) {
  grid[row][col] += d;
  for (int i = row; i <= mxn; i += (i & -i))
    for (int j = col; j <= mxn; j += (j & -j))
      bit[i][j] += d;
}
int sum(int row, int col) {
  // calculates sum from [1,1] till [row,col]
  int res = 0;
  for (int i = row; i > 0; i -= (i & -i))
    for (int j = col; j > 0; j -= (j & -j))
      res += bit[i][j];
  return res;
```

```
}
```

DSU.h

524177, 23 lines

```
struct DSU {
  int n;
  vector<int> parent;
  vector<int> size;
  DSU(int _n) : n(_n), parent(n), size(n, 1) { iota(parent
  ↳.begin(), parent.end(), 0); }
  int find_set(int x) {
    if (parent[x] == x) return x;
    return parent[x] = find_set(parent[x]);
  }
  int getSize(int x) { return size[find_set(x)]; }  //
  ↳returns size of component of x
  void union_sets(int x, int y) {
    x = find_set(x);
    y = find_set(y);
    if (x == y) return;
    if (size[x] > size[y]) {
      parent[y] = x;
      size[x] += size[y];
    } else {
      parent[x] = y;
      size[y] += size[x];
    }
  }
};
```

Mos.h

1cb2c0, 34 lines

```
int BLOCK = DO_NOT_FORGET_TO_CHANGE_THIS;
struct Query {
  int l, r, id;
  Query(int _l, int _r, int _id) : l(_l), r(_r), id(_id) {
  ↳}
  bool operator<(Query &o) {
    int mblock = l / BLOCK, oblock = o.l / BLOCK;
    return (mblock < oblock) or
           (mblock == oblock and mblock % 2 == 0 and r < o
           ↳.r) or
           (mblock == oblock and mblock % 2 == 1 and r > o
           ↳.r);
  };
};
// Solve
void solve() {
  vector<Query> queries;
  queries.reserve(q);
  for (int i = 0; i < q; i++) {
    int l, r; cin >> l >> r;
    l--, r--;
    queries.emplace_back(l, r, i);
  }
  sort(all(queries));
  int ans = 0;
  auto add = [&](int v) {};
  auto rem = [&](int v) {};
  vector<int> out(q);  // Change out type if necessary
  int cur_l = 0, cur_r = -1;
  for (auto &[l, r, id] : queries) {
    while (cur_l > l) add(--cur_l);
    while (cur_l < l) rem(cur_l++);
    while (cur_r < r) add(++cur_r);
    while (cur_r > r) rem(cur_r--);
    out[id] = ans;
  }
}
```

Graph (4)

4.1 Network flow

MinCostMaxFlow.h

9a3690, 177 lines

```
template <const int MAX_N, typename flow_t, typename
↳cost_t, flow_t FLOW_INF, cost_t COST_INF, const int
↳SCALE = 16>
struct CostScalingMCMF {
#define sz(a) a.size()
#define zero_stl(v, sz) fill(v.begin(), v.begin() + (sz),
↳0)
  struct Edge {
    int v;
    flow_t c;
    cost_t d;
    int r;
    Edge() = default;
    Edge(int v, flow_t c, cost_t d, int r) : v(v), c(c), d
    ↳(d), r(r) {}
  };
  vector<Edge> g[MAX_N];
  cost_t negativeSelfLoop;
  array<cost_t, MAX_N> pi, excess;
  array<int, MAX_N> level, ptr;
  CostScalingMCMF() { negativeSelfLoop = 0; }
  void clear() {
    negativeSelfLoop = 0;
    for (int i = 0; i < MAX_N; i++) g[i].clear();
  }
  void addEdge(int s, int e, flow_t cap, cost_t cost) {
    if (s == e) {
      if (cost < 0) negativeSelfLoop += cap * cost;
      return;
    }
    g[s].push_back(Edge(e, cap, cost, sz(g[e])));
    g[e].push_back(Edge(s, 0, -cost, sz(g[s]) - 1));
  }
  flow_t getMaxFlow(int V, int S, int T) {
    auto BFS = [&]() {
      zero_stl(level, V);
      queue<int> q;
      q.push(S);
      level[S] = 1;
      for (q.push(S); !q.empty(); q.pop()) {
        int v = q.front();
        for (const auto &e : g[v])
          if (!level[e.v] && e.c) q.push(e.v), level[e.v]
          ↳= level[v] + 1;
      }
      return level[T];
    };
    function<flow_t(int, flow_t)> DFS = [&](int v, flow_t
    ↳fl) {
      if (v == T || fl == 0) return fl;
      for (int &i = ptr[v]; i < (int)g[v].size(); i++) {
        Edge &e = g[v][i];
        if (level[e.v] != level[v] + 1 || !e.c) continue;
        flow_t delta = DFS(e.v, min(fl, e.c));
        if (delta) {
          e.c -= delta;
          g[e.v][e.r].c += delta;
          return delta;
        }
      }
      return flow_t(0);
    };
    flow_t maxFlow = 0, tmp = 0;
    while (BFS()) {
      zero_stl(ptr, V);
      while ((tmp = DFS(S, FLOW_INF))) maxFlow += tmp;
    }
    return maxFlow;
  }
  pair<flow_t, cost_t> maxflow(int N, int S, int T) {
    flow_t maxFlow = 0;
    cost_t eps = 0, minCost = 0;
    stack<int, vector<int>> stk;
    auto c_pi = [&](int v, const Edge &edge) { return edge
```

```
↳.d + pi[v] - pi[edge.v]; };
auto push = [&](int v, Edge &edge, flow_t delta, bool
↳flag) {
  delta = min(delta, edge.c);
  edge.c -= delta;
  g[edge.v][edge.r].c += delta;
  excess[v] -= delta;
  excess[edge.v] += delta;
  if (flag && 0 < excess[edge.v] && excess[edge.v] <=
  ↳delta) stk.push(edge.v);
};
auto relabel = [&](int v, cost_t delta) { pi[v] -=
↳delta + eps; };
auto lookAhead = [&](int v) {
  if (excess[v]) return false;
  cost_t delta = COST_INF;
  for (auto &e : g[v]) {
    if (e.c <= 0) continue;
    cost_t cp = c_pi(v, e);
    if (cp < 0)
      return false;
    else
      delta = min(delta, cp);
  }
  relabel(v, delta);
  return true;
};
auto discharge = [&](int v) {
  cost_t delta = COST_INF;
  for (int i = 0; i < sz(g[v]); i++) {
    Edge &e = g[v][i];
    if (e.c <= 0) continue;
    cost_t cp = c_pi(v, e);
    if (cp < 0) {
      if (lookAhead(e.v)) {
        i--;
        continue;
      }
      push(v, e, excess[v], true);
      if (excess[v] == 0) return;
    } else
      delta = min(delta, cp);
  }
  relabel(v, delta);
  stk.push(v);
};
zero_stl(pi, N);
zero_stl(excess, N);
for (int i = 0; i < N; i++)
  for (auto &e : g[i]) minCost += e.c * e.d, e.d *=
  ↳MAX_N + 1, eps = max(eps, e.d);
maxFlow = getMaxFlow(N, S, T);
while (eps > 1) {
  eps /= SCALE;
  if (eps < 1) eps = 1;
  stk = stack<int, vector<int>>();
  for (int v = 0; v < N; v++)
    for (auto &e : g[v])
      if (c_pi(v, e) < 0 && e.c > 0) push(v, e, e.c,
      ↳false);
  for (int v = 0; v < N; v++)
    if (excess[v] > 0) stk.push(v);
  while (stk.size()) {
    int top = stk.top();
    stk.pop();
    discharge(top);
  }
}
for (int v = 0; v < N; v++)
  for (auto &e : g[v]) e.d /= MAX_N + 1, minCost -= e.
  ↳c * e.d;
minCost = minCost / 2 + negativeSelfLoop;
return {maxFlow, minCost};
}
```

```
};

void solve() {
  CostScalingMCMF<102, int, int, 100, 100> flow;
  int n, m;
  cin >> n >> m;
  int start = 0;
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < m; j++) {
      int inp;
      cin >> inp;
      if (inp) {
        flow.addEdge(i + 1, n + 1 + j, 1, 0);
        start++;
      } else
        flow.addEdge(i + 1, n + 1 + j, 1, 1);
    }
  }
  int counta = 0, countb = 0;
  for (int i = 0; i < n; i++) {
    int inp;
    cin >> inp;
    counta += inp;
    flow.addEdge(0, i + 1, inp, 0);
  }
  for (int i = 0; i < m; i++) {
    int inp;
    cin >> inp;
    countb += inp;
    flow.addEdge(n + i + 1, n + m + 1, inp, 0);
  }
  if (counta != countb) {
    cout << -1 << endl;
    return;
  }
  pii t = flow.maxflow(102, 0, n + m + 1);
  if (t.first != counta) {
    cout << -1 << endl;
    return;
  }
  cout << t.second + start + t.second - counta << endl;
}
```

Dinic.h

Description: Complexity $O(VE\log U)$ where $U = \max|\text{cap}|$. $O(\min(E^{1/2}, V^{2/3})E)$ if $U = 1$; $O(\sqrt{V}E)$ for bipartite matching.

0a13c3, 126 lines

```
template <class T = int>
class Dinic {
 public:
  struct Edge {
    Edge(int a, T b) {
      to = a;
      cap = b;
    }
    int to;
    T cap;
  };

  Dinic(int n) {
    edges.resize(n);
    this->n = n;
  }

  T maxFlow(int src, int sink) {
    T ans = 0;
    while (bfs(src, sink)) {
      T flow;
      pt = vector<int>(n, 0);
      while ((flow = dfs(src, sink))) {
        ans += flow;
      }
    }
    return ans;
  }
```

```
  void addEdge(int from, int to, T cap = 1) {
    edges[from].push_back(list.size());
    list.push_back(Edge(to, cap));
    edges[to].push_back(list.size());
    list.push_back(Edge(from, 0));
  }

private:
  int n;
  vector<vector<int>> edges;
  vector<Edge> list;
  vector<int> h, pt;
  T dfs(int on, int sink, T flow = 1e9) {
    if (flow == 0) {
      return 0;
    }
    if (on == sink) {
      return flow;
    }
    for (; pt[on] < sz(edges[on]); pt[on]++) {
      int cur = edges[on][pt[on]];
      if (h[on] + 1 != h[list[cur].to]) {
        continue;
      }
      T got = dfs(list[cur].to, sink, min(flow, list[cur].
      ↳cap));
      if (got) {
        list[cur].cap -= got;
        list[cur ^ 1].cap += got;
        return got;
      }
    }
    return 0;
  }
  bool bfs(int src, int sink) {
    h = vector<int>(n, n);
    h[src] = 0;
    queue<int> q;
    q.push(src);
    while (!q.empty()) {
      int on = q.front();
      q.pop();
      for (auto a : edges[on]) {
        if (list[a].cap == 0) {
          continue;
        }
        int to = list[a].to;
        if (h[to] > h[on] + 1) {
          h[to] = h[on] + 1;
          q.push(to);
        }
      }
    }
    return h[sink] < n;
  }
};
void solve() {
  int n, m;
  cin >> n >> m;
  vi a(n);
  for (int i = 0; i < n; i++) {
    cin >> a[i];
  }
  Dinic<int> flow(n + 2);
  map<int, map<int, int>> factors;
  for (int i = 0; i < n; i++) {
    for (int j = 2; j * j <= a[i]; j++) {
      while (a[i] % j == 0) {
        factors[j][i + 1]++;
        a[i] /= j;
      }
    }
    if (a[i] > 1) {
```

```
      factors[a[i]][i + 1]++;
    }
  }
  for (int i = 0; i < m; i++) {
    int u, v;
    cin >> u >> v;
    if (u % 2 == 0) {
      swap(u, v);
    }
    flow.addEdge(u, v, 100);
  }
  int ans = 0;
  for (auto t : factors) {
    Dinic<int> tempflow = flow;
    for (auto t1 : t.second) {
      if (t1.first % 2 == 0) {
        tempflow.addEdge(t1.first, n + 1, t1.second);
      } else {
        tempflow.addEdge(0, t1.first, t1.second);
      }
    }
    ans += tempflow.maxFlow(0, n + 1);
  }
  cout << ans << endl;
}
```

4.2 DFS algorithms

SCC.h

cc0b7e, 72 lines

```
struct SCC {
  int n;
  vvi &adjLists, transposeLists;
  vi scc, leader;
  int sccCount = 0;
  vi sccSize;
  SCC(vvi& _adjLists) : n(sz(_adjLists)), adjLists(
  ↳_adjLists), transposeLists(n), scc(n, -1), leader(n,
  ↳-1) {
    for (int u = 0; u < n; u++) {
      for (int v : adjLists[u]) transposeLists[v].
      ↳push_back(u);
    }
    vb visited(n);
    stack<int> topoSort;
    function<void(int)> topoDFS = [&](int from) {
      visited[from] = true;
      for (auto to : adjLists[from]) {
        if (visited[to]) continue;
        topoDFS(to);
      }

      topoSort.push(from);
    };
    for (int i = 0; i < n; i++)
      if (not visited[i]) topoDFS(i);
    visited.assign(n, false);
    int sccPtr = 0;
    sccSize.assign(n, 0);
    function<void(int)> sccDFS = [&](int from) {
      scc[from] = sccPtr;
      sccSize[sccPtr]++;
      visited[from] = true;
      for (auto to : transposeLists[from]) {
        if (visited[to]) continue;
        sccDFS(to);
      }
    };
    while (not empty(topoSort)) {
      int i = topoSort.top();
      topoSort.pop();
      if (visited[i]) continue;
      sccDFS(i);
      leader[sccPtr] = i;
      sccPtr++;
    }
```

```
    sccCount = sccPtr;
  }
  int size(int index) {  // Returns size of scc of index
    return sccSize[scc[index]];
  }
  const int& operator[](int index) {
    return scc[index];
  }
  vi indexInCycle;
  void sccEnumeration() {
    indexInCycle.assign(n, 0);
    vb visited(n);
    int index = 0;
    function<void(int, int)> sccDFS = [&](int from, int sc
    ↳) {
      indexInCycle[from] = index++;
      visited[from] = true;
      for (auto to : adjLists[from]) {
        if (scc[to] != sc) continue;
        if (visited[to]) continue;
        sccDFS(to, sc);
      }
    };
    for (int i = 0; i < sccCount; i++) {
      index = 0;
      sccDFS(leader[i], i);
    }
  }
};
```

bridges.h

1bcaee, 69 lines

```
int n;                    // number of nodes
vector<vector<int>> adj;  // adjacency list of graph

vector<bool> visited;
vector<int> tin, low;
int timer;

void dfs(int v, int p = -1) {
  visited[v] = true;
  tin[v] = low[v] = timer++;
  for (int to : adj[v]) {
    if (to == p) continue;
    if (visited[to]) {
      low[v] = min(low[v], tin[to]);
    } else {
      dfs(to, v);
      low[v] = min(low[v], low[to]);
      if (low[to] > tin[v])
        IS_BRIDGE(v, to);
    }
  }
}

void find_bridges() {
  timer = 0;
  visited.assign(n, false);
  tin.assign(n, -1);
  low.assign(n, -1);
  for (int i = 0; i < n; ++i) {
    if (!visited[i])
      dfs(i);
  }
}

// ARTICULATION POINTS:
int n;
vector<vector<int>> adj;
vector<bool> visited;
vector<int> tin, low;
int timer;
void dfs(int v, int p = -1) {
  visited[v] = true;
```

```
  tin[v] = low[v] = timer++;
  int children = 0;
  for (int to : adj[v]) {
    if (to == p) continue;
    if (visited[to]) {
      low[v] = min(low[v], tin[to]);
    } else {
      dfs(to, v);
      low[v] = min(low[v], low[to]);
      if (low[to] >= tin[v] && p != -1)
        IS_CUTPOINT(v);
      ++children;
    }
  }
  if (p == -1 && children > 1)
    IS_CUTPOINT(v);
}
void find_cutpoints() {
  timer = 0;
  visited.assign(n, false);
  tin.assign(n, -1);
  low.assign(n, -1);
  for (int i = 0; i < n; ++i) {
    if (!visited[i])
      dfs(i);
  }
}
```

4.3 Trees

LCA.h

"../data-structures/RMQ.h" 993da6, 24 lines

```
struct LCA {
  int T = 0;
  vi st, path, ret;
  vi en, d;
  RMQ<int> rmq;
  LCA(vector<vi>& C) : st(sz(C)), en(sz(C)), d(sz(C)), rmq
  ↳((dfs(C, 0, -1), ret)) {}
  void dfs(vvi& adj, int v, int par) {
    st[v] = T++;
    for (auto to : adj[v])
      if (to != par) {
        path.pb(v), ret.pb(st[v]);
        d[to] = d[v] + 1;
        dfs(adj, to, v);
      }
    en[v] = T - 1;
  }
  bool anc(int p, int c) { return st[p] <= st[c] and en[p]
  ↳ >= en[c]; }
  int lca(int a, int b) {
    if (a == b) return a;
    tie(a, b) = minmax(st[a], st[b]);
    return path[rmq.query(a, b - 1)];
  }
  int dist(int a, int b) { return d[a] + d[b] - 2 * d[lca(
  ↳a, b)]; }
};
```

HLD.h

Description: Root must be 0.

c2f6cd, 38 lines

```
struct HLD {
  int n, timer = 0;
  vi top, tin, p, sub;
  HLD(vvi &adj) : n(sz(adj)), top(n), tin(n), p(n, -1),
  ↳sub(n, 1) {
    vi ord(n + 1);
    for (int i = 0, t = 0, v = ord[i]; i < n; v = ord[++i
    ↳])
      for (auto &to : adj[v])
        if (to != p[v]) p[to] = v, ord[++t] = to;
    for (int i = n - 1, v = ord[i]; i > 0; v = ord[--i])
    ↳sub[p[v]] += sub[v];
    for (int v = 0; v < n; v++)
      if (sz(adj[v])) iter_swap(begin(adj[v]), max_element
```

```
↳(all(adj[v]), [&](int a, int b) { return make_pair
↳(a != p[v], sub[a]) < make_pair(b != p[v], sub[b])
↳; }));
  function<void(int)> dfs = [&](int v) {
    tin[v] = timer++;
    for (auto &to : adj[v])
      if (to != p[v]) {
        top[to] = (to == adj[v][0] ? top[v] : to);
        dfs(to);
      }
  };
  dfs(0);
}
int lca(int u, int v) {
  return process(u, v, [](...) {});
}
template <class B>
int process(int a, int b, B op, bool ignore_lca = false)
↳ {
  for (int v;; op(tin[v], tin[b]), b = p[v]) {
    if (tin[a] > tin[b]) swap(a, b);
    if ((v = top[b]) == top[a]) break;
  }
  if (int l = tin[a] + ignore_lca, r = tin[b]; l <= r)
  ↳op(l, r);
  return a;
}
template <class B>
void subtree(int v, B op, bool ignore_lca = false) {
  if (sub[v] > 1 or !ignore_lca) op(tin[v] + ignore_lca,
  ↳ tin[v] + sub[v] - 1);
}
};
```

KthAnc.h

28228d, 55 lines

```
struct LCA {
  int n;
  vvi& adjLists;
  int lg;
  vvi up;
  vi depth;
  LCA(vvi& _adjLists, int root = 0) : n(sz(_adjLists)),
  ↳adjLists(_adjLists) {
    lg = 1;
    int pw = 1;
    while (pw <= n) pw <<= 1, lg++;
    // lg = 20
    up = vvi(n, vi(lg));
    depth.assign(n, -1);
    function<void(int, int)> parentDFS = [&](int from, int
    ↳ parent) {
      depth[from] = depth[parent] + 1;
      up[from][0] = parent;
      for (auto to : adjLists[from]) {
        if (to == parent) continue;
        parentDFS(to, from);
      }
    };
    parentDFS(root, root);
    for (int j = 1; j < lg; j++) {
      for (int i = 0; i < n; i++) {
        up[i][j] = up[up[i][j - 1]][j - 1];
      }
    }
  }
  int kthAnc(int v, int k) {
    int ret = v;
    int pw = 0;
    while (k) {
      if (k & 1) ret = up[ret][pw];
      k >>= 1;
      pw++;
    }
    return ret;
```

```
  }
  int lca(int u, int v) {
    if (depth[u] > depth[v]) swap(u, v);
    v = kthAnc(v, depth[v] - depth[u]);
    if (u == v) return v;
    while (up[u][0] != up[v][0]) {
      int i = 0;
      for (; i < lg - 1; i++) {
        if (up[u][i + 1] == up[v][i + 1]) break;
      }
      u = up[u][i], v = up[v][i];
    }
    return up[u][0];
  };
  int dist(int u, int v) {
    return depth[u] + depth[v] - 2 * depth[lca(u, v)];
  }
};
```

4.4 Math

4.4.1 Number of Spanning Trees

Create an $N \times N$ matrix `mat`, and for each edge $a \to b \in G$, do `mat[a][b]--`, `mat[b][b]++` (and `mat[b][a]--`, `mat[a][a]++` if G is undirected). Remove the ith row and column and take the determinant; this yields the number of directed spanning trees rooted at i (if G is undirected, remove any row/column).

4.4.2 Erdős–Gallai theorem

A simple graph with node degrees $d_1 \geq \cdots \geq d_n$ exists iff $d_1 + \cdots + d_n$ is even and for every $k = 1 \ldots n$,

$$\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min(d_i, k).$$

Number theory (5)

5.1 Modular arithmetic

ModularArithmetic.h 79388d, 20 lines

```
int add(int x, int y, int m = M) {
  int ret = (x + y) % m;
  if (ret < 0) ret += m;
  return ret;
}
int mult(int x, int y, int m = M) {
  int ret = (x * y) % m;
  if (ret < 0) ret += m;
  return ret;
}
int pw(int a, int b, int m = M) {
  int ret = 1;
  int p = a;
  while (b) {
    if (b & 1) ret = mult(ret, p, m);
    b >>= 1;
    p = mult(p, p, m);
  }
  return ret;
}
```

gcdextended.h

Description: Finds two integers x and y, such that $ax + by = \gcd(a, b)$. If a and b are coprime, then x is the inverse of $a \pmod b$.

1ea825, 5 lines

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

5.2 Primality

spf.h

8647ab, 17 lines

```
#define SIEVE_TILL (int)1e6
vector<int> primes;
vector<int> spf;
void sieve() {
  spf = vector<int>(SIEVE_TILL + 1, 0);
  for (int i = 2; i <= SIEVE_TILL; i++) {
    if (spf[i] == 0) primes.push_back(i), spf[i] = i;
    for (int j = 0; j < sz(primes) and i * primes[j] <=
    ↳SIEVE_TILL; j++) {
      spf[i * primes[j]] = primes[j];
      if (spf[i] == primes[j]) break;
    }
  }
}
bool isPrime(int n) {
  if (n <= 1) return false;
  return spf[n] == n;
}
```

MillerRabin.h

Description: Deterministic for numbers up to $7 \cdot 10^{18}$

Time: 7 times the complexity of $a^b \mod c$.

859df8, 13 lines

```
using ull = uint64_t;
bool isPrime(ull n) {
  if (n < 2 || n % 6 % 4 != 1) return (n | 1) == 3;
  ull A[] = {2, 325, 9375, 28178, 450775, 9780504,
  ↳1795265022},
      s = __builtin_ctzll(n - 1), d = n >> s;
  for (ull a : A) {
    ull p = pw(a % n, d, n), i = s;
    while (p != 1 && p != n - 1 && a % n && i--)
      p = mult(p, p, n);
    if (p != n - 1 && i != s) return 0;
  }
  return 1;
}
```

5.2.1 Primes

Primitive roots exist modulo any p^a, except $p = 2, a > 2$, and there are $\phi(\phi(p^a))$.

5.2.2 Estimates

$\sum_{d|n} d = O(n \log \log n)$.

5.2.3 Mobius Function

Mobius Inversion:

$$g(n) = \sum_{d|n} f(d) \Leftrightarrow f(n) = \sum_{d|n} \mu(d) g(n/d)$$

Other useful formulas/forms:

$g(n) = \sum_{n|d} f(d) \Leftrightarrow f(n) = \sum_{n|d} \mu(d/n) g(d)$

$g(n) = \sum_{1 \le m \le n} f(\lfloor \frac{n}{m} \rfloor) \Leftrightarrow f(n) = \sum_{1 \le m \le n} \mu(m) g(\lfloor \frac{n}{m} \rfloor)$

Strings (6)

Manacher.h

Description: p[0][i] = half length of longest even palindrome around pos i, p[1][i] = longest odd (half rounded down).

e7ad79, 13 lines

```
array<vi, 2> manacher(const string& s) {
  int n = sz(s);
  array<vi,2> p = {vi(n+1), vi(n)};
  rep(z,0,2) for (int i=0,l=0,r=0; i < n; i++) {
    int t = r-i+!z;
    if (i<r) p[z][i] = min(t, p[z][l+t]);
    int L = i-p[z][i], R = i+p[z][i]-!z;
    while (L>=1 && R+1<n && s[L-1] == s[R+1])
      p[z][i]++, L--, R++;
    if (R>r) l=L, r=R;
  }
  return p;
}
```

Trie.h

6d8465, 29 lines

```
struct trieobject {
  trieobject() {
    children[0] = NULL;
    children[1] = NULL;
    numelems = 0;
  };

  struct trieobject* children[2];
  int numelems;
};
struct trie {
  trieobject base;
  trie() {
    trieobject base;
  }
  void add(int x) {
    int pow2 = (1ll << 31ll);
    trieobject* temp = &base;
    while (pow2 > 0) {
      if (temp->children[1 && (x & pow2)] == NULL) {
        temp->children[1 && (x & pow2)] = new trieobject;
      }
      temp->children[1 && (x & pow2)]->numelems++;
      temp = temp->children[1 && (x & pow2)];
      pow2 /= 2;
    }
  }
  // ADD FUNCTION BELOW
};
```

Combinatorial (7)

7.1 Permutations

7.1.1 Cycles

Let $g_S(n)$ be the number of n-permutations whose cycle lengths all belong to the set S. Then $\sum_{n=0}^{\infty} g_S(n)\frac{x^n}{n!} = \exp\left(\sum_{n\in S}\frac{x^n}{n}\right)$

7.1.2 Derangements

Permutations of a set such that none of the elements appear in their original position.

$D_n = (n-1)(D_{n-1} + D_{n-2}) = nD_{n-1} + (-1)^n = \left\lfloor \frac{n!}{e} \right\rceil$

7.1.3 Burnside's lemma

Given a group G of symmetries and a set X, the number of elements of X *up to symmetry* equals $\frac{1}{|G|}\sum_{g\in G}|X^g|$, where X^g are the elements fixed by g ($g.x = x$).

If $f(n)$ counts "configurations" (of some sort) of length n, we can ignore rotational symmetry using $G = \mathbb{Z}_n$ to get $g(n) = \frac{1}{n}\sum_{k=0}^{n-1} f(\gcd(n,k)) = \frac{1}{n}\sum_{k|n} f(k)\phi(\frac{n}{k})$.

7.2 Partitions and subsets

7.2.1 Lucas' Theorem

Let n, m be non-negative integers and p a prime. Write $n = n_k p^k + ... + n_1 p + n_0$ and $m = m_k p^k + ... + m_1 p + m_0$. Then $\binom{n}{m} \equiv \prod_{i=0}^{k}\binom{n_i}{m_i} \pmod{p}$.

7.3 General purpose numbers

7.3.1 Stirling numbers (1)

permutations on n items with k cycles.

$$c_{n,k} = c_{n-1,k-1} + (n-1)c_{n-1,k},\ c_{0,0} = 1$$
$$\sum_{k=0}^{n} c_{n,k}x^k = x(x+1)\dots(x+n-1)$$

7.3.2 Eulerian numbers

Number of permutations $\pi \in S_n$ in which exactly k elements are greater than the previous element. k j:s s.t. $\pi(j) > \pi(j+1)$, $k+1$ j:s s.t. $\pi(j) \geq j$, k j:s s.t. $\pi(j) > j$. $E_{n,k} = (n-k)E_{n-1,k-1} + (k+1)E_{n-1,k}$
$E_{n,0} = E_{n,n-1} = 1$ $E_{n,k} = \sum_{j=0}^{k}(-1)^j\binom{n+1}{j}(k+1-j)^n$

7.3.3 Stirling numbers (2)

Partitions of n distinct elements into exactly k groups.
$S_{n,k} = S_{n-1,k-1} + kS_{n-1,k}$ $S_{n,1} = S_{n,n} = 1$
$S_{n,k} = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n$

7.3.4 Bell numbers

Ways to partition of n distinct elements. $B_{p^m+n} \equiv mB_n + B_{n+1} \pmod p$

7.3.5 Catalan numbers

$C_n = \frac{1}{n+1}\binom{2n}{n} = \binom{2n}{n} - \binom{2n}{n+1} = \frac{(2n)!}{(n+1)!n!}$
$C_0 = 1,\ C_{n+1} = \frac{2(2n+1)}{n+2}C_n,\ C_{n+1} = \sum C_iC_{n-i}$ • sub-diagonal monotone paths in an $n \times n$ grid. • strings with n pairs of parenthesis. • binary trees with with $n+1$ leaves (0 or 2 children). • ordered trees with $n+1$ vertices. • ways a convex polygon with $n+2$ sides can be cut into triangles by connecting vertices with straight lines. • permutations of $[n]$ with no 3-term increasing subseq.

Various (8)

8.0.1 Bit hacks

• `x&-x` is the least bit in `x`. • `for(int x=m;x;x=(x-1)&m)` loops over all subset masks of `m`. • `c = x&-x, r = x+c; (((r^x) >> 2)/c) | r` is the next number after `x` with the same number of bits set. • `rep(b,0,K) rep(i,0,(1<<K)) if (i & 1 << b) D[i] += D[i^(1 << b)];` computes all sums of subsets.

Numerical (9)

9.1 Fourier transforms

FastFourierTransform.h

Description: fft(a) computes $\hat{f}(k) = \sum_x a[x]\exp(2\pi i \cdot kx/N)$ for all k. N must be a power of 2. `conv(a, b) = c`, $c[x] = \sum a[i]b[x-i]$. For convolution of complex numbers or more than two vectors: FFT, multiply pointwise, divide by n, reverse(start+1, end), FFT back. Rounding is safe if $(\sum a_i^2 + \sum b_i^2)\log_2 N < 9 \cdot 10^{14}$ (in practice 10^{16}; higher for random inputs).

00ced6, 35 lines

```
typedef complex<double> C;
typedef vector<double> vd;
void fft(vector<C>& a) {
  int n = sz(a), L = 31 - __builtin_clz(n);
  static vector<complex<long double>> R(2, 1);
  static vector<C> rt(2, 1);  // (^ 10% faster if double)
  for (static int k = 2; k < n; k *= 2) {
    R.resize(n); rt.resize(n);
    auto x = polar(1.0L, acos(-1.0L) / k);
    rep(i,k,2*k) rt[i] = R[i] = i&1 ? R[i/2] * x : R[i/2];
  }
  vi rev(n);
  rep(i,0,n) rev[i] = (rev[i / 2] | (i & 1) << L) / 2;
  rep(i,0,n) if (i < rev[i]) swap(a[i], a[rev[i]]);
  for (int k = 1; k < n; k *= 2)
    for (int i = 0; i < n; i += 2 * k) rep(j,0,k) {
      C z = rt[j+k] * a[i+j+k]; // (25% faster if hand-rolled)
      a[i + j + k] = a[i + j] - z;
      a[i + j] += z;
    }
}
vd conv(const vd& a, const vd& b) {
  if (a.empty() || b.empty()) return {};
  vd res(sz(a) + sz(b) - 1);
  int L = 32 - __builtin_clz(sz(res)), n = 1 << L;
```

```
  vector<C> in(n), out(n);
  copy(all(a), begin(in));
  rep(i,0,sz(b)) in[i].imag(b[i]);
  fft(in);
  for (C& x : in) x *= x;
  rep(i,0,n) out[i] = in[-i & (n - 1)] - conj(in[i]);
  fft(out);
  rep(i,0,sz(res)) res[i] = imag(out[i]) / (4 * n);
  return res;
}
```

NTT.h

Description: Can be used for convolutions modulo specific nice primes of the form $2^a b + 1$, where the convolution result has size at most 2^a. (125000001 << 3) + 1 = 1e9 + 7, therefore do not use this for M = 1e9 + 7. For $p < 2^30$ there is also e.g. (5 << 25, 3), (7 << 26, 3), For other primes/integers, use two different primes and combine with CRT. (479 << 21, 3) and (483 << 21, 5). The last two are $> 10^9$. Inputs must be in [0, mod).

f9d990, 46 lines

```
// Requires mod func
const int M = 998244353;
const int root = 3;
// (119 << 23) + 1, root = 3;  // for M = 998244353
void ntt(int* x, int* temp, int* roots, int N, int skip) {
  if (N == 1) return;
  int n2 = N / 2;
  ntt(x, temp, roots, n2, skip * 2);
  ntt(x + skip, temp, roots, n2, skip * 2);
  for (int i = 0; i < N; i++) temp[i] = x[i * skip];
  for (int i = 0; i < n2; i++) {
    int s = temp[2 * i], t = temp[2 * i + 1] * roots[skip * i];
    x[skip * i] = (s + t) % M;
    x[skip * (i + n2)] = (s - t) % M;
  }
}
void ntt(vi& x, bool inv = false) {
  int e = pw(root, (M - 1) / sz(x));
  if (inv) e = pw(e, M - 2);
  vi roots(sz(x), 1), temp = roots;
  for (int i = 1; i < sz(x); i++) roots[i] = roots[i - 1] * e % M;
  ntt(&x[0], &temp[0], &roots[0], sz(x), 1);
}
// Usage: just pass the two coefficients list to get a*b (modulo M)
vi conv(vi a, vi b) {
  int s = sz(a) + sz(b) - 1;
  if (s <= 0) return {};
  int L = s > 1 ? 32 - __builtin_clzll(s - 1) : 0, n = 1 << L;
  if (s <= 200) {  // (factor 10 optimization for |a|,|b| = 10)
    vi c(s);
    for (int i = 0; i < sz(a); i++)
      for (int j = 0; j < sz(b); j++)
        c[i + j] = (c[i + j] + a[i] * b[j]) % M;
    return c;
  }
  a.resize(n);
  ntt(a);
  b.resize(n);
  ntt(b);
  vi c(n);
  int d = pw(n, M - 2);
  for (int i = 0; i < n; i++) c[i] = a[i] * b[i] % M * d % M;
  ntt(c, true);
  c.resize(s);
  return c;
}
```

Geometry (10)

10.1 Geometric primitives

Point.h

Description: Avoid $T = int$

47ec0a, 31 lines

```
template <class T>
int sgn(T x) { return (x > 0) - (x < 0); }
template <class T>
struct Point {
  typedef Point P;
  T x, y;
  explicit Point(T x = 0, T y = 0) : x(x), y(y) {}
  bool operator<(P p) const { return tie(x, y) < tie(p.x,
  ↳p.y); }
  bool operator==(P p) const { return tie(x, y) == tie(p.x
  ↳, p.y); }
  P operator+(P p) const { return P(x + p.x, y + p.y); }
  P operator-(P p) const { return P(x - p.x, y - p.y); }
  P operator*(T d) const { return P(x * d, y * d); }
  P operator/(T d) const { return P(x / d, y / d); }
  T dot(P p) const { return x * p.x + y * p.y; }
  T cross(P p) const { return x * p.y - y * p.x; }
  T cross(P a, P b) const { return (a - *this).cross(b - *
  ↳this); }
  T dist2() const { return x * x + y * y; }
  double dist() const { return sqrt((double)dist2()); }
  // angle to x-axis in interval [-pi, pi]
  double angle() const { return atan2(y, x); }
  P unit() const { return *this / dist(); }  // makes dist
  ↳()=1
  P perp() const { return P(-y, x); }          // rotates
  ↳+90 degrees
  P normal() const { return perp().unit(); }
  // returns point rotated 'a' radians ccw around the
  ↳origin
  P rotate(double a) const {
    return P(x * cos(a) - y * sin(a), x * sin(a) + y * cos
    ↳(a));
  }
  friend ostream& operator<<(ostream& os, P p) {
    return os << "(" << p.x << "," << p.y << ")";
  }
};
```

ConvexHull.h

Description: Returns a vector of the points of the convex hull in counter-clockwise order. Points on the edge of the hull between two other points are not considered part of the hull.

Time: $\mathcal{O}(n \log n)$

310954, 14 lines

```
// Needs point
typedef Point<ll> P;
vector<P> convexHull(vector<P> pts) {
  if (sz(pts) <= 1) return pts;
  sort(all(pts));
  vector<P> h(sz(pts)+1);
  int s = 0, t = 0;
  for (int it = 2; it--; s = --t, reverse(all(pts)))
    for (P p : pts) {
      while (t >= s + 2 && h[t-2].cross(h[t-1], p) <= 0) t
      ↳--;
      h[t++] = p;
    }
  return {h.begin(), h.begin() + t - (t == 2 && h[0] == h
  ↳[1])};
}
```

ClosestPair.h

Description: Finds the closest pair of points.

fb15fd, 18 lines

```
// Requires point
typedef Point<int> P;
pair<P, P> closest(vector<P> v) {
  assert(sz(v) > 1);
  set<P> S;
```

```
  sort(all(v), [](P a, P b) { return a.y < b.y; });
  pair<int, pair<P, P>> ret{LLONG_MAX, {P(), P()}};
  int j = 0;
  for (P p : v) {
    P d{1 + (int)sqrtl(ret.first), 0};
    while (v[j].y <= p.y - d.x) S.erase(v[j++]);
    auto lo = S.lower_bound(p - d), hi = S.upper_bound(p +
    ↳ d);
    for (; lo != hi; ++lo)
      ret = min(ret, {(*lo - p).dist2(), {*lo, p}});
    S.insert(p);
  }
  return ret.second;
}
```