Ivan Franko National University of Lviv

Stallions

Maksym Shcherba, Petro Tarnaksyi, Yarema Stiahar

2023-10-11

Contents

1 Contest 1

2 Convolutions 2

3 Data Structures 5

4 Graphs 9

5 Strings 14

6 Various 17

Contest (1)

template.hpp

<bits/stdc++.h> 4ea3a9, 24 lines

```
using namespace std;

#define FOR(i, a, b) for(int i = (a); i < (b); i
    ++)
#define RFOR(i, b, a) for(int i = (b) - 1; i >= (
    a); i--)
#define SZ(a) (int)a.size()
#define ALL(a) a.begin(), a.end()
#define PB push_back
#define MP make_pair
#define F first
#define S second

typedef long long LL;
typedef vector<int> VI;
typedef pair<int, int> PII;
typedef double db;

int main()
{
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout << fixed << setprecision(15);

  return 0;
}
```

compilation.txt

2 lines

```
g++ -O2 -std=c++17 -Wno-unused-result -Wshadow -
    Wall -o %e %e.cpp
g++ -std=c++17 -Wshadow -Wall -o %e %e.cpp -
    fsanitize=address -fsanitize=undefined -
    D_GLIBCXX_DEBUG -g
```

s.sh

6 lines

```
for((i = 0; ; i++)) do
  echo $i
  ./gen $i > in
  diff -w <(./a < in) <(./brute < in) || break
  [ $? == 0 ] || break
done
```

hash.sh

3 lines

```
# Hashes a file, ignoring all whitespace and
    comments. Use for
# verifying that code was correctly typed.
cpp -dD -P -fpreprocessed | tr -d '[:space:]'|
    md5sum |cut -c-6
```

troubleshoot.txt

52 lines

```
Pre-submit:
Write a few simple test cases if sample is not
    enough.
Are time limits close? If so, generate max cases.
Is the memory usage fine?
Could anything overflow?
Make sure to submit the right file.

Wrong answer:
Print your solution! Print debug output, as well.
Are you clearing all data structures between test
     cases?
Can your algorithm handle the whole range of
    input?
Read the full problem statement again.
Do you handle all corner cases correctly?
Have you understood the problem correctly?
Any uninitialized variables?
Any overflows?
Confusing N and M, i and j, etc.?
Are you sure your algorithm works?
What special cases have you not thought of?
Are you sure the STL functions you use work as
    you think?
Add some assertions, maybe resubmit.
Create some testcases to run your algorithm on.
Go through the algorithm for a simple case.
Go through this list again.
Explain your algorithm to a teammate.
Ask the teammate to look at your code.
Go for a small walk, e.g. to the toilet.
Is your output format correct? (including
    whitespace)
Rewrite your solution from the start or let a
    teammate do it.

Runtime error:
Have you tested all corner cases locally?
Any uninitialized variables?
Are you reading or writing outside the range of
    any vector?
Any assertions that might fail?
Any possible division by 0? (mod 0 for example)
Any possible infinite recursion?
Invalidated pointers or iterators?
Are you using too much memory?
Debug with resubmits (e.g. remapped signals, see
    Various).

Time limit exceeded:
Do you have any possible infinite loops?
What is the complexity of your algorithm?
Are you copying a lot of unnecessary data? (
    References)
How big is the input and output? (consider scanf)
Avoid vector, map. (use arrays/unordered_map)
What do your teammates think about your algorithm
    ?

Memory limit exceeded:
What is the max amount of memory your algorithm
    should need?
Are you clearing all data structures between test
     cases?
```

Convolutions (2)

fft.hpp

4d6300, 87 lines

```
const int mod = 998244353;

int add(int a, int b)
{
  return (a + b < mod) ? (a + b) : (a + b - mod);
}
int sub(int a, int b)
{
  return (a - b < 0) ? (a - b + mod) : (a - b);
}
int mult(int a, int b)
{
  return a * (LL) b % mod;
}
int binpow(int a, int n)
{
  int res = 1;
  while(n)
  {
    if(n & 1)
      res = mult(res, a);
    a = mult(a, a);
    n /= 2;
  }
  return res;
}

const int LEN = 1 << 23;
const int GEN = 31;
const int IGEN = binpow(GEN, mod - 2);

void fft(VI& a, bool inv)
{
  int lg = 0;
  while((1 << lg) < SZ(a)) lg++;
  FOR(i, 0, SZ(a))
  {
    int x = 0;
    FOR(j, 0, lg)
      x |= ((i >> j) & 1) << (lg - j - 1);
    if(i < x)
      swap(a[i], a[x]);
  }
  for(int len = 2; len <= SZ(a); len *= 2)
  {
    int ml = binpow(inv ? IGEN : GEN, LEN / len);
    for(int i = 0; i < SZ(a); i += len)
    {
      int pw = 1;
      FOR(j, 0, len / 2)
      {
        int v = a[i + j];
        int u = mult(a[i + j + len / 2], pw);

        a[i + j] = add(v, u);
        a[i + j + len / 2] = sub(v, u);

        pw = mult(pw, ml);
      }
    }
  }
  if(inv)
  {
    int m = binpow(SZ(a), mod - 2);
    FOR(i, 0, SZ(a))
      a[i] = mult(a[i], m);
  }
}

VI mult(VI a, VI b)
{
  int sz = 0;
  int sum = SZ(a) + SZ(b) - 1;
  while((1 << sz) < sum) sz++;
  a.resize(1 << sz);
  b.resize(1 << sz);

  fft(a, 0);
  fft(b, 0);

  FOR(i, 0, SZ(a))
    a[i] = mult(a[i], b[i]);

  fft(a, 1);
  a.resize(sum);
  return a;
}
```

inverse.hpp

a4673f, 32 lines

```
VI inverse(const VI& a, int k)
{
  assert(SZ(a) == k && a[0] != 0);
  if(k == 1)
    return {binpow(a[0], mod - 2)};

  VI ra = a;
  FOR(i, 0, SZ(ra))
    if(i & 1)
      ra[i] = sub(0, ra[i]);

  int nk = (k + 1) / 2;
  VI t = mult(a, ra);
  t.resize(k);

  FOR(i, 0, nk)
    t[i] = t[2 * i];

  t.resize(nk);
  t = inverse(t, nk);
  t.resize(k);

  RFOR(i, nk, 1)
  {
    t[2 * i] = t[i];
    t[i] = 0;
  }

  VI res = mult(ra, t);
  res.resize(k);
  return res;
}
```

exp-log.hpp

5549eb, 52 lines

```
VI deriv(const VI& a, int k)
{
  VI res(k);
  FOR(i, 0, k)
    if(i + 1 < SZ(a))
      res[i] = mult(a[i + 1], i + 1);
  return res;
}

VI integr(const VI& a, int k)
{
  VI res(k);
  RFOR(i, k, 1)
    res[i] = mult(a[i - 1], inv[i]);
  res[0] = 0;
  return res;
}

VI log(const VI& a, int k)
{
  assert(a[0] == 1);
  VI ml = mult(deriv(a, k), inverse(a, k));
  return integr(ml, k);
```

```
}

VI exp(VI a, int k)
{
  assert(a[0] == 0);

  VI Qk = {1};
  int pw = 1;
  while(pw <= k)
  {
    pw *= 2;

    Qk.resize(pw);
    VI lnQ = log(Qk, pw);

    FOR(i, 0, SZ(lnQ))
    {
      if(i < SZ(a))
        lnQ[i] = sub(a[i], lnQ[i]);
      else
        lnQ[i] = sub(0, lnQ[i]);
    }
    updAdd(lnQ[0], 1);

    Qk = mult(Qk, lnQ);
  }
  Qk.resize(k);
  return Qk;
}
```

modulo.hpp

8b6a95, 34 lines

```
void removeLeadingZeros(VI& a)
{
  while(SZ(a) > 0 && a.back() == 0)
    a.pop_back();
}

pair<VI, VI> modulo(VI a, VI b)
{
  //assert(a.back() != 0 && b.back() != 0);
  int n = SZ(a), m = SZ(b);
  if(m > n)
    return MP(VI{}, a);

  reverse(ALL(a));
  reverse(ALL(b));

  VI d = b;
  d.resize(n - m + 1);
  d = mult(a, inverse(d, n - m + 1));
  d.resize(n - m + 1);

  reverse(ALL(a));
  reverse(ALL(b));
  reverse(ALL(d));

  VI res = mult(b, d);
  res.resize(SZ(a));
  FOR(i, 0, SZ(a))
    res[i] = sub(a[i], res[i]);

  removeLeadingZeros(d);
  removeLeadingZeros(res);
  return MP(d, res);
}
```

multipoint-eval.hpp

8f6f41, 33 lines

```
int x[LEN];
VI P[2 * LEN];

void build(int v, int tl, int tr)
{
  if(tl + 1 == tr)
  {
    P[v] = {sub(0, x[tl]), 1};
    return;
  }
  int tm = (tl + tr) / 2;
  build(2 * v + 1, tl, tm);
  build(2 * v + 2, tm, tr);

  P[v] = mult(P[2 * v + 1], P[2 * v + 2]);
}
int ans[LEN];
void solve(int v, int tl, int tr, const VI& Q)
//Q != Q % P[0] -> wa
{
  if(SZ(Q) == 0)
    return;
  if(tl + 1 == tr)
  {
    ans[tl] = Q[0];
    return;
  }
  int tm = (tl + tr) / 2;
  solve(2 * v + 1, tl, tm,
  modulo(Q, P[2 * v + 1]).S);
  solve(2 * v + 2, tm, tr,
  modulo(Q, P[2 * v + 2]).S);
}
```

newton.hpp

9ffaac, 50 lines

```
VI newton(VI a, int k)
{
  //c_n= a_n + sum(i = 0, n - 1) c_i * c_(n-1-i)
  //Q = A + x * Q * Q
  //F(Q) = Q - x * Q * Q - A
  //F'(Q) = 1 - 2 * x * Q

  VI Qk = {a[0]};
  int pw = 1;
  while(pw <= k)
  {
    assert(SZ(Qk) == pw);
    pw *= 2;

    VI F1(pw);
    F1[0] = 1;
    FOR(i, 0, pw / 2)
      F1[i + 1] = sub(0, mult(2, Qk[i]));
    //F' = 1 - 2 * x * Q

    VI F = mult(Qk, Qk);
    F.resize(pw);
    RFOR(i, pw, 1)
      F[i] = sub(0, F[i - 1]);
    F[0] = 0; // F = -x * Q*Q

    FOR(i, 0, pw / 2)
      F[i] = add(F[i], Qk[i]);
      //F = Q - x * Q * Q
    FOR(i, 0, min(pw, SZ(a)))
      F[i] = sub(F[i], a[i]);
      //F = Q - x * Q * Q - A

    F = mult(F, inverse(F1, pw));
    F.resize(pw);

    FOR(i, 0, pw)
      F[i] = sub(0, F[i]);//-F/F'
    FOR(i, 0, pw / 2)
      F[i] = add(F[i], Qk[i]);//Q - F/F'

    //new Qk = Qk - F(Qk) / F'(Qk) mod(x ^ pw)
    Qk = F;
  }
  Qk.resize(k);
  return Qk;
}
```

berlekamp-massey.hpp

866c28, 36 lines

```
VI berlekampMassey(const VI& a)
{
  VI c = {1}, bp = {1};
  int l = 0, b = 1, x = 1;
  FOR(j, 0, SZ(a))
  {
    assert(SZ(c) == l + 1);
    int d = a[j];
    FOR(i, 1, l + 1)
      updAdd(d, mult(c[i], a[j - i]));
    if (d == 0)
    {
      x++;
      continue;
    }
    VI t = c;
    int coef = mult(d, binPow(b, mod - 2));
    if (SZ(bp) + x > SZ(c))
      c.resize(SZ(bp) + x);
    FOR(i, 0, SZ(bp))
      updSub(c[i + x], mult(coef, bp[i]));
    if (2 * l > j)
    {
      x++;
      continue;
    }
    l = j + 1 - l;
    bp = t;
    b = d;
    x = 1;
  }
  c.erase(c.begin());
  for (int& ci : c)
    ci = mult(ci, mod - 1);
  return c;
}
```

conv-xor.hpp

8ce066, 12 lines

```
void convXor(VI& a, int k)
{
  FOR(i, 0, k)
    FOR(j, 0, 1 << k)
      if((j & (1 << i)) == 0)
      {
        int u = a[j];
        int v = a[j + (1 << i)];
        a[j] = u + v;
        a[j + (1 << i)] = u - v;
      }
}
```

conv-and.hpp

b8d23e, 12 lines

```
void convAnd(VI& a, int k, bool inverse)
{
  FOR(i, 0, k)
    FOR(j, 0, 1 << k)
      if((j & (1 << i)) == 0)
      {
        if(inverse)
          a[j] -= a[j + (1 << i)];
        else
          a[j] += a[j + (1 << i)];
      }
}
```

conv-or.hpp

6ffcf0, 12 lines

```
void convOr(VI& a, int k, bool inverse)
{
  FOR(i, 0, k)
    FOR(j, 0, 1 << k)
      if((j & (1 << i)) == 0)
      {
        if(inverse)
          a[j + (1 << i)] -= a[j];
        else
          a[j + (1 << i)] += a[j];
      }
}
```

Data Structures (3)

dsu.hpp

2de4ff, 34 lines

```
struct DSU
{
  int n;
  VI p;
  VI sz;

  void init(int _n)
  {
    n = _n;
    sz.assign(n, 1);
    p.resize(n);
    iota(ALL(p), 0);
  }

  int find(int v)
  {
    if (v == p[v])
      return v;
    return p[v] = find(p[v]);
  }

  bool unite(int u, int v)
  {
    u = find(u);
    v = find(v);
    if (u == v)
      return false;
    if (sz[u] > sz[v])
      swap(u, v);
    p[u] = v;
    sz[v] += sz[u];
    return true;
  }
};
```

Fenwick.hpp

d4ebda, 43 lines

```
struct Fenwick
{
  int n;
  vector<LL> v;

  void init(int _n)
  {
    n = _n;
    v.assign(n, 0);
  }

  void add(int i, int x)
  {
    for (; i < n; i = (i + 1) | i)
      v[i] += x;
  }

  LL sum(int i)
  {
    LL ans = 0;
    for (; i >= 0; i = (i & (i + 1)) - 1)
      ans += v[i];
    return ans;
  }

  int lower_bound(LL x)
  {
    LL sum = 0;
    int i = -1;
    int lg = 31 - __builtin_clz(n);
    while (lg >= 0)
    {
      int j = i + (1 << lg);
      if (j < n && sum + v[j] < x)
      {
        sum += v[j];
        i = j;
      }
      lg--;
    }
    return i + 1;
  }
};
```

Fenwick.txt

16 lines

```
Minimum on segment:
1) Use two Fenwick trees with n = 2^k. n = 1 << (32 - __builtin_clz(n - 1)
   );
2) One tree for normal array and one for reversed
3) When querying for minimum on the segment only consider
   segments [(i & (i + 1)), i] from trees that are COMPLETELY inside the
       segment

Fenwick tree for adding on segment (prefixes):
1) Use 2 arrays: mult and add
2) upd(int i, int updMult, int updAdd) default Fenwick update.
3) add x on segment [l, r]:
  upd(l, x, -x * (l - 1));
  upd(r, -x, x * r);
4) to calculate sum on prefix r:
```

```
  sumAdd and sumMult - default Fenwick sum
  st - initial value of r
  ans = st * sumMult + sumAdd
```

treap.hpp

1ac2c5, 142 lines

```
mt19937 rng;

struct Node
{
  int l, r;
  int x;
  int y;
  int cnt;
  int par;
  int rev;
  int mn;

  void init(int value)
  {
    l = r = -1;
    x = value;
    y = rng();
    cnt = 1;
    par = -1;
    rev = 0;
    mn = value;
  }
};

struct Treap
{
  Node A[MAX];
  int sz = 0;

  int getCnt(int v)
  {
    if (v == -1)
      return 0;
    return A[v].cnt;
  }
  int getMn(int v)
  {
    if (v == -1)
      return INF;
    return A[v].mn;
  }
  int newNode(int val)
  {
    A[sz].init(val);
    return sz++;
  }
  void upd(int v)
  {
    if (v == -1)
      return;
    A[v].cnt = getCnt(A[v].l) + getCnt(A[v].r) + 1;
    A[v].mn = min(A[v].x, min(getMn(A[v].l), getMn(A[v].r)));
  }
  void reverse(int v)
  {
    if (v == -1)
      return;
    A[v].rev ^= 1;
  }
  void push(int v)
  {
    if (v == -1 || A[v].rev == 0)
      return;
    reverse(A[v].l);
    reverse(A[v].r);
    swap(A[v].l, A[v].r);
    A[v].rev = 0;
  }
  PII split(int v, int cnt)
  {
    if (v == -1)
      return {-1, -1};
    push(v);
    int left = getCnt(A[v].l);
    PII res;
    // if (val <= A[v].x)
    if (cnt <= left)
    {
      if (A[v].l != -1)
        A[A[v].l].par = -1;
      res = split(A[v].l, cnt);
      A[v].l = res.second;
      if (res.second != -1)
        A[res.second].par = v;
      res.second = v;
    }
    else
    {
      if (A[v].r != -1)
        A[A[v].r].par = -1;
      // split(v, val)
      res = split(A[v].r, cnt - left - 1);
      A[v].r = res.first;
      if (res.first != -1)
        A[res.first].par = v;
```

```
      res.first = v;
    }
    upd(v);
    return res;
  }
  int merge(int v, int u)
  {
    if (v == -1) return u;
    if (u == -1) return v;
    int res;
    // if (rng() % (getCnt(v) + getCnt(u)) < getCnt(v))
    if (A[v].y > A[u].y)
    {
      push(v);
      if (A[v].r != -1)
        A[A[v].r].par = -1;
      res = merge(A[v].r, u);
      A[v].r = res;
      if (res != -1)
        A[res].par = v;
      res = v;
    }
    else
    {
      push(u);
      if (A[u].l != -1)
        A[A[u].l].par = -1;
      res = merge(v, A[u].l);
      A[u].l = res;
      if (res != -1)
        A[res].par = u;
      res = u;
    }
    upd(res);
    return res;
  }
  int getIdx(int v, int from = -1)
  {
    if (v == -1)
      return 0;
    int x = getIdx(A[v].par, v);
    if (from == -1 || A[v].r == from)
      x += getCnt(A[v].l) + 1;
    push(v);
    return x;
  }
};
```

ordered-set.hpp

<ext/pb_ds/assoc_container.hpp> 716651, 14 lines

```
using namespace __gnu_pbds;
using namespace std;
typedef tree<int, null_type, less<int>, rb_tree_tag,
    tree_order_statistics_node_update> ordered_set;
typedef tree<int, null_type, less_equal<int>, rb_tree_tag,
    tree_order_statistics_node_update> ordered_multiset;
// example: ordered_set s; s.insert(47);
// s.order_of_key(k); -- returns number of elements less then k
// s.find_by_order(k); - returns iterator to k-th element or s.end()
// s.count() does not exist. Use find in set and upper_bound in multiset.
// *s.end() = 0
// ordered_multiset:
//    1) lower_bound and upper_bound swapped
//      2) find does not work
//    3) you cannot erase by value. Use s.erase(s.upper_bound(x)) instead
//      4) to count x use s.order_of_key(x + 1) - s.order_of_key(x).
```

sparse-table.hpp

d45c6b, 29 lines

```
int lg[MAX + 1];

struct SparseTable
{
  int t[MAX][LOG];

  void init(const VI& v)
  {
    lg[1] = 0;
    FOR (i, 2, MAX + 1) lg[i] = lg[i / 2] + 1;
    FOR (i, 0, MAX) FOR (j, 0, LOG) t[i][j] = INF;
    FOR (i, 0, SZ(v)) t[i][0] = v[i];

    FOR (j, 1, LOG)
    {
      int len = 1 << (j - 1);
      FOR (i, 0, MAX - (1 << j))
      {
        t[i][j] = min(t[i][j - 1], t[i + len][j - 1]);
      }
    }
  }

  int query(int l, int r)
  {
    int i = lg[r - l + 1];
    return min(t[l][i], t[r - (1 << i) + 1][i]);
  }
} st;
```

convex-hull-trick.hpp

06db0c, 70 lines

```
struct Line
{
  LL a, b, xLast;
  Line() {}
  Line(LL _a, LL _b): a(_a), b(_b) {}
  bool operator<(const Line& l) const
  {
    return MP(a, b) < MP(l.a, l.b);
  }
  bool operator<(int x) const
  {
    return xLast < x;
  }
  __int128 getY(__int128 x) const
  {
    return a * x + b;
  }
  LL intersect(const Line& l) const
  {
    assert(a < l.a);
    LL dA = l.a - a, dB = b - l.b, x = dB / dA;
    if (dB < 0 && dB % dA != 0)
      x--;
    return x;
  }
};

struct ConvexHull: set<Line, less<>>
{
  bool needErase(iterator it, const Line& l)
  {
    LL x = it->xLast;
    if (it->getY(x) > l.getY(x))
      return false;
    if (it == begin())
      return it->a >= l.a;
    x = prev(it)->xLast + 1;
    return it->getY(x) < l.getY(x);
  }
  void add(LL a, LL b)
  {
    Line l(a, b);
    auto it = lower_bound(l);
    if (it != end())
    {
      LL x = it == begin() ? -LINF : prev(it)->xLast;
      if ((it == begin() || prev(it)->getY(x) >= l.getY(x))
        && it->getY(x + 1) >= l.getY(x + 1))
        return;
    }
    while (it != end() && needErase(it, l))
      it = erase(it);
    while (it != begin() && needErase(prev(it), l))
      erase(prev(it));
    if (it != begin())
    {
      auto itP = prev(it);
      Line lIt = *itP;
      lIt.xLast = itP->intersect(l);
      erase(itP);
      insert(lIt);
    }
    l.xLast = it == end() ? LINF : l.intersect(*it);
    insert(l);
  }
  LL getMaxY(LL x)
  {
    return lower_bound(x)->getY(x);
  }
};
```

Graphs (4)

centroid.hpp

226f45, 33 lines

```
void build(int cent)
{
  dfsSZ(cent, -1);
  int szAll = sz[cent];
  int pr = cent;
  while (true)
  {
    int v = -1;
    for (auto to : g[cent])
    {
      if (to == pr || usedc[to])
        continue;
      if (sz[to] * 2 > szAll)
      {
        v = to;
        break;
      }
    }
    if (v == -1)
      break;
    pr = cent;
    cent = v;
  }
  usedc[cent] = true;

  for (auto to : g[cent])
  {
    if (!usedc[to])
    {
      build(to);
    }
  }
}
```

HLD.hpp

6b3a13, 66 lines

```
VI g[MAX];
int sz[MAX];
int h[MAX];
int p[MAX];
int top[MAX];
int tin[MAX];
int tout[MAX];
int t = 0;

void dfsSZ(int v, int par = -1, int hei = 0)
{
  sz[v] = 1;
  h[v] = hei;
  p[v] = par;
  for (auto& to : g[v])
  {
    if (to == par)
      continue;
    dfsSZ(to, v, hei + 1);
    sz[v] += sz[to];
    if (g[v][0] == par || sz[g[v][0]] < sz[to])
      swap(g[v][0], to);
  }
}
void dfsHLD(int v, int par = -1, int tp = 0)
{
  tin[v] = t++;
  top[v] = tp;
  FOR (i, 0, SZ(g[v]))
  {
    int to = g[v][i];
    if (to == par)
      continue;
    if (i == 0)
      dfsHLD(to, v, tp);
    else
      dfsHLD(to, v, to);
  }
  tout[v] = t - 1;
}
LL get(int x, int y)
{
  LL res = 0;
  while(true)
  {
    int tx = top[x];
    int ty = top[y];
    if (tx == ty)
    {
      int t1 = tin[x];
      int t2 = tin[y];
      if (t1 > t2)
        swap(t1, t2);
      res += query(t1, t2);
      break;
    }
    if (h[tx] < h[ty])
    {
      swap(tx, ty);
      swap(x, y);
```

```
    }
    res += query(tin[tx], tin[x]);
    x = p[tx];
  }
  return res;
}
```

dinic.hpp

6afa18, 92 lines

```
struct Graph
{
  struct Edge
  {
    int from, to;
    LL cap, flow;
  };

  int _n;
  vector<Edge> edges;
  vector<VI> g;
  VI d, p;

  Graph() : _n(0) {}
  Graph(int n) : _n(n), g(n), d(n), p(n) {}

  void addEdge(int from, int to, LL cap)
  {
    assert(0 <= from && from < _n);
    assert(0 <= to && to < _n);
    assert(0 <= cap);
    g[from].PB(SZ(edges));
    edges.PB({from, to, cap, 0});
    g[to].PB(SZ(edges));
    edges.PB({to, from, 0, 0});
  }

  int bfs(int s, int t)
  {
    fill(ALL(d), -1);
    d[s] = 0;
    queue<int> q;
    q.push(s);
    while (!q.empty())
    {
      int v = q.front();
      q.pop();
      for (int e : g[v])
      {
        int to = edges[e].to;
        if (edges[e].flow < edges[e].cap && d[to] == -1)
        {
          d[to] = d[v] + 1;
          q.push(to);
        }
      }
    }
    return d[t];
  }

  LL dfs(int v, int t, LL flow)
  {
    if (v == t || flow == 0)
      return flow;
    for (; p[v] < SZ(g[v]); p[v]++)
    {
      int e = g[v][p[v]], to = edges[e].to;
      LL c = edges[e].cap, f = edges[e].flow;
      if (f < c && (to == t || d[to] == d[v] + 1))
      {
        LL push = dfs(to, t, min(flow, c - f));
        if (push > 0)
        {
          edges[e].flow += push;
          edges[e ^ 1].flow -= push;
          return push;
        }
      }
    }
    return 0;
  }

  LL flow(int s, int t)
  {
    assert(0 <= s && s < _n);
    assert(0 <= t && t < _n);
    assert(s != t);
    LL flow = 0;
    while (bfs(s, t) != -1)
    {
      fill(ALL(p), 0);
      while (true)
      {
        LL f = dfs(s, t, LINF);
        if (f == 0)
          break;
        flow += f;
      }
    }
    return flow;
  }
};
```

min-cost-flow.hpp

8a8605, 102 lines

```
struct Graph
{
  struct Edge
  {
    int from, to;
    int cap, flow;
    LL cost;
  };

  int _n;
  vector<Edge> edges;
  vector<VI> g;
  vector<LL> d;
  VI p, w;

  Graph(): _n(0) {}
  Graph(int n): _n(n), g(n), d(n), p(n), w(n) {}

  void addEdge(int from, int to, int cap, LL cost)
  {
    assert(0 <= from && from < _n);
    assert(0 <= to && to < _n);
    assert(0 <= cap);
    assert(0 <= cost);
    g[from].PB(SZ(edges));
    edges.PB({from, to, cap, 0, cost});
    g[to].PB(SZ(edges));
    edges.PB({to, from, 0, 0, -cost});
  }

  pair<int, LL> flow(int s, int t)
  {
    assert(0 <= s && s < _n);
    assert(0 <= t && t < _n);
    assert(s != t);
    int flow = 0;
    LL cost = 0;
    while (true)
    {
      fill(ALL(d), LINF);
      fill(ALL(p), -1);
      fill(ALL(w), 0);
      queue<int> q1, q2;
      w[s] = 1;
      d[s] = 0;
      q2.push(s);
      while (!q1.empty() || !q2.empty())
      {
        int v;
        if (!q1.empty())
        {
          v = q1.front();
          q1.pop();
        }
        else
        {
          v = q2.front();
          q2.pop();
        }
        for (int e : g[v])
        {
          if (edges[e].flow == edges[e].cap)
            continue;
          int to = edges[e].to;
          LL newDist = d[v] + edges[e].cost;
          if (newDist < d[to])
          {
            d[to] = newDist;
            p[to] = e;
            if (w[to] == 0)
              q2.push(to);
            else if (w[to] == 2)
              q1.push(to);
            w[to] = 1;
          }
        }
        w[v] = 2;
      }
      if (p[t] == -1)
        break;
      int curFlow = INF;
      LL curCost = 0;
      for (int v = t; v != s;)
      {
        int e = p[v];
        curFlow = min(curFlow, edges[e].cap - edges[e].flow);
        curCost += edges[e].cost;
        v = edges[e].from;
      }
      for (int v = t; v != s;)
      {
        int e = p[v];
        edges[e].flow += curFlow;
        edges[e ^ 1].flow -= curFlow;
        v = edges[e].from;
      }
      flow += curFlow;
      cost += curCost * curFlow;
    }
```

```
    return {flow, cost};
  }
};
```

hungarian.hpp

0baccf, 63 lines

```
LL hungarian(const vector<vector<LL>>& a)
{
  int n = SZ(a), m = SZ(a[0]);
  assert(n <= m);
  vector<LL> u(n + 1), v(m + 1);
  VI p(m + 1, n), way(m + 1);
  FOR(i, 0, n)
  {
    p[m] = i;
    int j0 = m;
    vector<LL> minv(m + 1, LINF);
    vector<int> used(m + 1);
    while (p[j0] != n)
    {
      used[j0] = true;
      int i0 = p[j0], j1 = -1;
      LL delta = LINF;
      FOR(j, 0, m)
      {
        if (!used[j])
        {
          int cur = a[i0][j] - u[i0] - v[j];
          if (cur < minv[j])
          {
            minv[j] = cur;
            way[j] = j0;
          }
          if (minv[j] < delta)
          {
            delta = minv[j];
            j1 = j;
          }
        }
      }
      assert(j1 != -1);
      FOR(j, 0, m + 1)
      {
        if (used[j])
        {
          u[p[j]] += delta;
          v[j] -= delta;
        }
        else
          minv[j] -= delta;
      }
      j0 = j1;
    }
    while (j0 != m)
    {
      int j1 = way[j0];
      p[j0] = p[j1];
      j0 = j1;
    }
  }
  VI ans(n + 1);
  FOR(j, 0, m)
    ans[p[j]] = j;
  LL res = 0;
  FOR(i, 0, n)
    res += a[i][ans[i]];
  assert(res == -v[m]);
  return res;
}
```

edmonds-blossom.hpp

654404, 129 lines

```
struct Graph
{
  int n;
  vector<VI> g;
  VI label, first, mate;

  Graph() {}
  Graph(int _n): n(_n), g(_n + 1), label(_n + 1),
      first(_n + 1), mate(_n + 1) {}

  void addEdge(int u, int v)
  {
    assert(0 <= u && u < n);
    assert(0 <= v && v < n);
    u++;
    v++;
    g[u].PB(v);
    g[v].PB(u);
  }

  void augmentPath(int v, int w)
  {
    int t = mate[v];
    mate[v] = w;
    if(mate[t] != v)
      return;
    if(label[v] <= n)
    {
      mate[t] = label[v];
      augmentPath(label[v], t);
```

```
      return;
    }
    int x = label[v] / (n + 1), y = label[v] % (n + 1);
    augmentPath(x, y);
    augmentPath(y, x);
  }

  int findMaxMatching()
  {
    FOR(i, 0, n + 1)
      assert(mate[i] == 0);
    int mt = 0;
    DSU dsu;
    FOR(u, 1, n + 1)
    {
      if(mate[u] == 0)
      {
        fill(ALL(label), -1);
        iota(ALL(first), 0);
        dsu.init(n + 1);
        label[u] = 0;
        dsu.unite(u, 0);
        queue<int> q;
        q.push(u);
        while(!q.empty())
        {
          int x = q.front();
          q.pop();
          for(int y: g[x])
          {
            if(mate[y] == 0 && y != u)
            {
              mate[y] = x;
              augmentPath(x, y);
              while(!q.empty())
                q.pop();
              mt++;
              break;
            }
            if(label[y] < 0)
            {
              int v = mate[y];
              if(label[v] < 0)
              {
                label[v] = x;
                dsu.unite(v, y);
                q.push(v);
              }
            }
            else
            {
              int r = first[dsu.find(x)], s = first[dsu.find(y)];
              if(r != s)
              {
                int edgeLabel = (n + 1) * x + y;
                label[r] = label[s] = -edgeLabel;
                int join;
                while(true)
                {
                  if(s != 0)
                    swap(r, s);
                  r = first[dsu.find(label[mate[r]])];
                  if(label[r] == -edgeLabel)
                  {
                    join = r;
                    break;
                  }
                  label[r] = -edgeLabel;
                }
                for(int z: {x, y})
                {
                  for(int v = first[dsu.find(z)]; v != join;
                    v = first[dsu.find(label[mate[v]])])
                  {
                    label[v] = edgeLabel;
                    if (dsu.unite(v, join))
                      first[dsu.find(join)] = join;
                    q.push(v);
                  }
                }
              }
            }
          }
        }
      }
    }
    return mt;
  }

  int getMate(int v)
  {
    assert(0 <= v && v < n);
    v++;
    int u = mate[v];
    assert(u == 0 || mate[u] == v);
    u--;
    return u;
  }
};
```

Strings (5)

Aho-Corasick.hpp

b80725, 70 lines

```
const int AL = 26;

struct Node
{
  int p;
  int c;
  int g[AL];
  int nxt[AL];
  int link;

  void init()
  {
    c = -1;
    p = -1;
    fill(g, g + AL, -1);
    fill(nxt, nxt + AL, -1);
    link = -1;
  }
};

struct AC
{
  Node A[MAX];
  int sz;
  void init()
  {
    A[0].init();
    sz = 1;
  }
  int addStr(const string& s)
  {
    int x = 0;
    FOR (i, 0, SZ(s))
    {
      int c = s[i] - 'A'; // change to [0 AL)
      if (A[x].nxt[c] == -1)
      {
        A[x].nxt[c] = sz;
        A[sz].init();
        A[sz].c = c;
        A[sz].p = x;
        sz++;
      }
      x = A[x].nxt[c];
    }
    return x;
  }
  int go(int x, int c)
  {
    if (A[x].g[c] != -1)
      return A[x].g[c];

    if (A[x].nxt[c] != -1)
      A[x].g[c] = A[x].nxt[c];
    else if (x != 0)
      A[x].g[c] = go(getLink(x), c);
    else
      A[x].g[c] = 0;

    return A[x].g[c];
  }
  int getLink(int x)
  {
    if (A[x].link != -1)
      return A[x].link;
    if (x == 0 || A[x].p == 0)
      return 0;
    return A[x].link=go(getLink(A[x].p), A[x].c);
  }
} A;
```

automaton.hpp

2a29a0, 60 lines

```
const int AL = 26;
struct Node
{
  int g[AL];
  int link;
  int len;
  int cnt;
  void init()
  {
    fill(g, g + AL, -1);
    link = -1;
    len = -1;
  }
};
struct Automaton
{
  Node A[MAX * 2];
  int sz;
  int head;
  void init()
  {
    sz = 1;
    head = 0;
    A[0].init();
  }
  void add(char c)
  {
    int ch = c - 'a'; // change to [0 AL)
    int nhead = sz++;
    A[nhead].init();
    A[nhead].len = A[head].len + 1;
    int cur = head;
    head = nhead;
    while (cur != -1 && A[cur].g[ch] == -1)
    {
      A[cur].g[ch] = head;
      cur = A[cur].link;
    }
    if (cur == -1)
    {
      A[head].link = 0;
      return;
    }
    int p = A[cur].g[ch];
    if (A[p].len == A[cur].len + 1)
    {
      A[head].link = p;
      return;
    }
    int q = sz++;
    A[q] = A[p];
    A[q].len = A[cur].len + 1;
    A[p].link = A[head].link = q;
    while (cur != -1 && A[cur].g[ch] == p)
    {
      A[cur].g[ch] = q;
      cur = A[cur].link;
    }
  }
};
```

suffix-array.hpp

d69c2d, 60 lines

```
void countSort(VI& p, const VI& c)
{
  int n = SZ(p);
  VI cnt(n);
  FOR (i, 0, n)
    cnt[c[i]]++;
  VI pos(n);
  FOR (i, 1, n)
    pos[i] = pos[i - 1] + cnt[i - 1];
  VI p2(n);
  for (auto x : p)
  {
```

```
    int i = c[x];
    p2[pos[i]++] = x;
  }
  p = p2;
}

VI suffixArray(const string& t)
{
  string s = t + "$";
  int n = SZ(s);
  VI p(n), c(n);
  FOR (i, 0, n) p[i] = i;
  sort(ALL(p), [&](int i, int j)
  {
    return s[i] < s[j];
  });
  int x = 0;
  c[p[0]] = 0;
  FOR (i, 1, n)
  {
    if (s[p[i]] != s[p[i - 1]])
      x++;
    c[p[i]] = x;
  }
  int k = 0;
  while ((1 << k) < n)
  {
    FOR (i, 0, n)
      p[i] = (p[i] - (1 << k) + n) % n;

    countSort(p, c);

    VI c2(n);
    PII pr = {c[p[0]], c[(p[0] + (1 << k)) % n]};
    FOR (i, 1, n)
    {
      PII nx={c[p[i]], c[(p[i] + (1 << k)) % n]};
      c2[p[i]] = c2[p[i - 1]];
      if (pr != nx)
        c2[p[i]]++;
      pr = nx;
    }
    c = c2;
    k++;
  }
  p.erase(p.begin());
  return p;
}
```

lcp.hpp

72ff1e, 24 lines

```
VI lcpArray(const string& s, const VI& sa)
{
    int n = SZ(s);
    VI rnk(n);
    FOR (i, 0, n)
        rnk[sa[i]] = i;
    VI lcp(n - 1);
    int h = 0;
    FOR (i, 0, n)
    {
        if (h > 0)
      h--;
        if (rnk[i] == 0)
      continue;
        int j = sa[rnk[i] - 1];
        for (; j + h < n && i + h < n; h++)
        {
            if (s[j + h] != s[i + h])
        break;
        }
        lcp[rnk[i] - 1] = h;
    }
    return lcp;
}
```

manacher.hpp

9f7ea5, 39 lines

```
int d1[MAX], d2[MAX];

void manacher(const string& s)
{
    int n = SZ(s);
    int l = -1;
    int r = -1;
    FOR (i, 0, n)
    {
        if (i <= r)
            d1[i] = min(r - i + 1,
              d1[l + (r - i)]);
        while (i + d1[i] < n && i - d1[i] >= 0
          && s[i + d1[i]] == s[i - d1[i]])
            d1[i]++;
        if (i + d1[i] - 1 > r)
        {
            r = i + d1[i] - 1;
            l = i - (d1[i] - 1);
        }
    }
    l = -1;
    r = -1;
    FOR (i, 0, n)
    {
        if (i <= r)
            d2[i] = min(r - i + 1,
              d2[l + (r - i) + 1]);
        while (i + d2[i] < n
          && i - (d2[i] + 1) >= 0
          && s[i + d2[i]] == s[i - (d2[i] + 1)])
            d2[i]++;
        if (i + d2[i] > r)
        {
            r = i + d2[i] - 1;
            l = i - d2[i];
        }
    }
}
```

z.hpp

e27ac7, 23 lines

```
VI zFunction(const string& s)
{
  int n = SZ(s);
  VI z(n);

  int l = 0;
  int r = 0;
  FOR (i, 1, n)
  {
    z[i] = 0;
    if (i <= r)
      z[i] = min(r - i + 1, z[i - l]);

    while(i + z[i] < n && s[i + z[i]] == s[z[i]])
      z[i]++;
    if(i + z[i] - 1 > r)
    {
      r = i + z[i] - 1;
      l = i;
    }
  }
  return z;
}
```

prefix.hpp

500608, 16 lines

```
VI prefixFunction(const string& s)
{
  int n = SZ(s);
  VI p(n);
  p[0] = 0;
```

```
  FOR (i, 1, n)
  {
    int j = p[i - 1];
    while(j != 0 && s[i] != s[j])
      j = p[j - 1];

    if (s[i] == s[j]) j++;
    p[i] = j;
  }
  return p;
}
```

Various (6)

gcd.hpp

e001bc, 16 lines

```
int gcd(int a, int b, int& x, int& y)
{
  x = 1, y = 0;
  int x2 = 0, y2 = 1;
  while (b)
  {
    int k = a / b;
    x -= k * x2;
    y -= k * y2;
    a %= b;
    swap(a, b);
    swap(x, x2);
    swap(y, y2);
  }
  return a;
}
```

mobius.hpp

fba6c5, 19 lines

```
void mobius()
{
  fill(pr, pr + N, 1);
  fill(mu, mu + N, 1);
  pr[1] = false;
  FOR (i, 2, N)
  {
    if (!pr[i])
      continue;
    mu[i] = mod - 1;
    for (int j = 2 * i; j < N; j += i)
    {
      pr[j] = false;
      if (j % (i * i) == 0)
        mu[j] = 0;
      mu[j] = mult(mu[j], mod - 1);
    }
  }
}
```

triangles.hpp

a22d8b, 30 lines

```
int triangles(int n, int m)
{
  FOR (i, 0, m)
  {
    auto [u, v] = edges[i];
        if (MP(deg[u], u) < MP(deg[v], v))
          g[u].PB(v);
        else
      g[v].PB(u);
    }
    int cnt = 0;
    FOR (v, 0, n)
    {
        for (auto u : g[v])
      used[u] = 1;
        for (auto u : g[v])
        {
            for(auto w : g[u])
            {
        if (used[w])
        {
          cnt++;
        }
      }
        }
        for (auto u : g[v])
      used[u] = 0;
    }
    return cnt;
}
```