1 Data-Structure

1.1 fast-set

```
// Can correctly work with numbers in range [0; MAXN]
// Supports all std::set operations in O(1) on random queries / dense arrays, O(log_64(N)) in worst case (sparce array).
// Count operation works in O(1) always.
template<uint MAXN>
class fast_set {
private:
  static const uint PREF = (MAXN <= 64 ? 0 :
              MAXN <= 4096 ? 1 :
              MAXN <= 262144 ? 1 + 64 :
              MAXN <= 16777216 ? 1 + 64 + 4096 :
              MAXN <= 1073741824 ? 1 + 64 + 4096 + 262144 : 227) + 1;
  static constexpr ull lb(int x) {
    if(x == 64) return ULLONG_MAX;
    return (1ULL << x) - 1;
  };
  static const uint SZ = PREF + (MAXN + 63) / 64 + 1;
  ull m[SZ] = {0};
  inline uint left(uint v) const { return (v - 62) * 64; }
  inline uint parent(uint v) const { return v / 64 + 62; }
  inline void setbit(uint v) { m[v >> 6] |= 1ULL << (v & 63); }
  inline void resetbit(uint v) { m[v >> 6] &= ~(1ULL << (v & 63)); }
  inline uint getbit(uint v) const { return m[v >> 6] >> (v & 63) & 1; }
  inline ull childs_value(uint v) const { return m[left(v) >> 6]; }
  inline int left_go(uint x, const uint c) const {
    const ull rem = x & 63;
    uint bt = PREF * 64 + x;
    ull num = m[bt >> 6] & lb(rem + c);
    if(num) return (x ^ rem) | __lg(num);
    for(bt = parent(bt); bt > 62; bt = parent(bt)) {
      const ull rem = bt & 63;
      num = m[bt >> 6] & lb(rem);
      if(num) {
        bt = (bt ^ rem) | __lg(num);
        break;
      }
    }
    if(bt == 62) return -1;
    while(bt < PREF * 64) bt = left(bt) | __lg(m[bt - 62]);
    return bt - PREF * 64;
  }
  inline int right_go(uint x, const uint c) const {
    const ull rem = x & 63;
    uint bt = PREF * 64 + x;
    ull num = m[bt >> 6] & ~lb(rem + c);
    if(num) return (x ^ rem) | __builtin_ctzll(num);
    for(bt = parent(bt); bt > 62; bt = parent(bt)) {
      const ull rem = bt & 63;
      num = m[bt >> 6] & ~lb(rem + 1);
      if(num) {
        bt = (bt ^ rem) | __builtin_ctzll(num);
        break;
      }
    }
    if(bt == 62) return -1;
    while(bt < PREF * 64) bt = left(bt) | __builtin_ctzll(m[bt - 62]);
    return bt - PREF * 64;
  }

public:
  fast_set() { assert(PREF != 228); setbit(62); }
  bool empty() const {return getbit(63);}
  void clear() { fill(m, m + SZ, 0); setbit(62); }
  bool count(uint x) const { return m[PREF + (x >> 6)] >> (x & 63) & 1; }
  void insert(uint x) { for(uint v = PREF * 64 + x; !getbit(v); v = parent(v)) setbit(v); }
  void erase(uint x) {
    if(!getbit(PREF * 64 + x)) return;
    resetbit(PREF * 64 + x);
    for(uint v = parent(PREF * 64 + x); v > 62 && !childs_value(v); v = parent(v)) resetbit(v);
  }
  int find_next(uint x) const { return right_go(x, 0); } // >=
  int find_prev(uint x) const { return left_go(x, 1); } // <=
};
```

1.2 lazysegtree

```
template<class S,
         S (*e)(),
         S (*op)(S, S),
         class F,
         F (*id)(),
         S (*mapping)(F, S),
         F (*composition)(F, F)>
class lazy_segtree {
public:
  lazy_segtree() : lazy_segtree(0) {}
  explicit lazy_segtree(int _n) : lazy_segtree(vector<S>(_n, e())) {}
  explicit lazy_segtree(const vector<S>& v) : n((int) v.size()) {
    log = __lg(2 * n - 1), size = 1 << log;
    d.resize(size * 2, e());
    lz.resize(size, id());
    for(int i = 0; i < n; i++) d[size + i] = v[i];
    for(int i = size - 1; i; i--) update(i);
  }
  void set(int p, S x) {
    assert(0 <= p && p < n);
    p += size;
    for(int i = log; i; --i) push(p >> i);
    d[p] = x;
    for(int i = 1; i <= log; ++i) update(p >> i);
  }
  S get(int p) {
    assert(0 <= p && p < n);
    p += size;
    for(int i = log; i; i--) push(p >> i);
    return d[p];
  }
  S prod(int l, int r) {
    assert(0 <= l && l <= r && r <= n);
    if(l == r) return e();
    l += size;
    r += size;
    for(int i = log; i; i--) {
      if(((l >> i) << i) != l) push(l >> i);
      if(((r >> i) << i) != r) push(r >> i);
    }
    S sml = e(), smr = e();
    while(l < r) {
      if(l & 1) sml = op(sml, d[l++]);
      if(r & 1) smr = op(d[--r], smr);
      l >>= 1;
      r >>= 1;
    }
    return op(sml, smr);
  }
  S all_prod() const { return d[1]; }
  void apply(int p, F f) {
    assert(0 <= p && p < n);
    p += size;
    for(int i = log; i; i--) push(p >> i);
    d[p] = mapping(f, d[p]);
    for(int i = 1; i <= log; i++) update(p >> i);
  }
  void apply(int l, int r, F f) {
    assert(0 <= l && l <= r && r <= n);
    if(l == r) return;
    l += size;
    r += size;
    for(int i = log; i; i--) {
      if(((l >> i) << i) != l) push(l >> i);
      if(((r >> i) << i) != r) push((r - 1) >> i);
    }
    {
      int l2 = l, r2 = r;
      while(l < r) {
        if(l & 1) all_apply(l++, f);
        if(r & 1) all_apply(--r, f);
        l >>= 1;
        r >>= 1;
      }
      l = l2;
      r = r2;
    }
    for(int i = 1; i <= log; i++) {
      if(((l >> i) << i) != l) update(l >> i);
      if(((r >> i) << i) != r) update((r - 1) >> i);
    }
  }
  template<bool (*g)(S)> int max_right(int l) {
    return max_right(l, [](S x) { return g(x); });
  }
  template<class G> int max_right(int l, G g) {
    assert(0 <= l && l <= n && g(e()));
    if(l == n) return n;
    l += size;
    for(int i = log; i; i--) push(l >> i);
    S sm = e();
    do {
      while(!(l & 1)) l >>= 1;
      if(!g(op(sm, d[l]))) {
        while(l < size) {
          push(l);
          l <<= 1;
          if(g(op(sm, d[l]))) sm = op(sm, d[l++]);
        }
        return l - size;
      }
      sm = op(sm, d[l++]);
    } while((l & -l) != l);
    return n;
  }
  template<bool (*g)(S)> int min_left(int r) {
    return min_left(r, [](S x) { return g(x); });
  }
  template<class G> int min_left(int r, G g) {
    assert(0 <= r && r <= n && g(e()));
    if(r == 0) return 0;
    r += size;
    for(int i = log; i >= 1; i--) push((r - 1) >> i);
    S sm = e();
    do {
      r--;
      while(r > 1 && (r & 1)) r >>= 1;
      if(!g(op(d[r], sm))) {
        while(r < size) {
          push(r);
          r = r << 1 | 1;
          if(g(op(d[r], sm))) sm = op(d[r--], sm);
        }
        return r + 1 - size;
      }
      sm = op(d[r], sm);
    } while((r & -r) != r);
    return 0;
  }

private:
  int n, size, log;
```

```cpp
  vector<S> d;
  vector<F> lz;
  void update(int k) { d[k] = op(d[k << 1], d[k << 1 | 1]); }
  void all_apply(int k, F f) {
    d[k] = mapping(f, d[k]);
    if(k < size) lz[k] = composition(f, lz[k]);
  }
  void push(int k) {
    all_apply(k << 1, lz[k]);
    all_apply(k << 1 | 1, lz[k]);
    lz[k] = id();
  }
};
```

1.3 segtree

```cpp
template<class S, S (*e)(), S (*op)(S, S)>
class segtree {
public:
  segtree() : segtree(0) {}
  explicit segtree(int _n) : segtree(vector<S>(_n, e())) {}
  explicit segtree(const vector<S>& a): n(a.size()) {
    log = __lg(2 * n - 1), size = 1 << log;
    st.resize(size << 1, e());
    for(int i = 0; i < n; i++) st[size + i] = a[i];
    for(int i = size - 1; i; i--) update(i);
  }
  void set(int p, S val) {
    assert(0 <= p && p < n);
    st[p += size] = val;
    for(int i = 1; i <= log; ++i) update(p >> i);
  }
  S get(int p) const {
    return st[p + size];
  }
  S prod(int l, int r) const {
    assert(0 <= l && l <= r && r <= n);
    S sml = e(), smr = e();
    l += size, r += size;
    while(l < r) {
      if(l & 1) sml = op(sml, st[l++]);
      if(r & 1) smr = op(st[--r], smr);
      l >>= 1;
      r >>= 1;
    }
    return op(sml, smr);
  }
  S all_prod() const { return st[1]; }
  template<bool (*f)(S)> int max_right(int l) const {
    return max_right(l, [](S x) { return f(x); });
  }
  template<class F> int max_right(int l, F f) const {
    assert(0 <= l && l <= n && f(e()));
    if(l == n) return n;
    l += size;
    S sm = e();
    do {
      while(~l & 1) l >>= 1;
      if(!f(op(sm, st[l]))) {
        while(l < size) {
          l <<= 1;
          if(f(op(sm, st[l]))) sm = op(sm, st[l++]);
        }
        return l - size;
      }
      sm = op(sm, st[l++]);
    } while((l & -l) != l);
    return n;
  }
  template<bool (*f)(S)> int min_left(int r) const {
    return min_left(r, [](S x) { return f(x); });
  }
  template<class F> int min_left(int r, F f) const {
    assert(0 <= r && r <= n && f(e()));
    if(r == 0) return 0;
    r += size;
    S sm = e();
    do {
      r--;
      while(r > 1 && (r & 1)) r >>= 1;
      if(!f(op(st[r], sm))) {
        while(r < size) {
          r = r << 1 | 1;
          if(f(op(st[r], sm))) sm = op(st[r--], sm);
        }
        return r + 1 - size;
      }
      sm = op(st[r], sm);
    } while((r & -r) != r);
    return 0;
  }

private:
  int n, size, log;
  vector<S> st;
  void update(int v) { st[v] = op(st[v << 1], st[v << 1 | 1]); }
};
```

1.4 sparse-table

```cpp
template<class T, T (*op)(T, T)>
class sparse_table {
public:
  sparse_table() {}
  explicit sparse_table(const vector<T>& a) : n(a.size()) {
    int max_log = __lg(n) + 1;
    mat.resize(max_log);
    mat[0] = a;
    for(int j = 1; j < max_log; ++j) {
      mat[j].resize(n - (1 << j) + 1);
      for(int i = 0; i <= n - (1 << j); ++i) {
        mat[j][i] = op(mat[j - 1][i], mat[j - 1][i + (1 << (j - 1))]);
      }
    }
  }
  T prod(int from, int to) const {
    assert(0 <= from && from <= to && to <= n - 1);
    int lg = __lg(to - from + 1);
    return op(mat[lg][from], mat[lg][to - (1 << lg) + 1]);
  }

private:
  int n;
  vector<vector<T>> mat;
};
```

1.5 動態凸包

```cpp
struct line_t {
  mutable ll k, m, p;
  bool operator<(const line_t& o) const { return k < o.k; }
  bool operator<(ll x) const { return p < x; }
};
template<bool MAX>
struct CHT : multiset<line_t, less<>> {
  static const ll INF = numeric_limits<ll>::max();
  bool isect(iterator x, iterator y) {
    if(y == end()) return x->p = INF, 0;
    if(x->k == y->k) {
      x->p = (x->m > y->m ? INF : -INF);
    } else {
      x->p = floor_div(y->m - x->m, x->k - y->k); // see Math
    }
    return x->p >= y->p;
  }
  void add_line(ll k, ll m) {
    if(!MAX) k = -k, m = -m;
    auto z = insert({k, m, 0}), y = z++, x = y;
    while(isect(y, z)) z = erase(z);
    if(x != begin() && isect(--x, y)) isect(x, y = erase(y));
    while((y = x) != begin() && (--x)->p >= y->p) isect(x, erase(y));
  }
  ll get(ll x) {
    assert(!empty());
    auto l = *lower_bound(x);
    return (l.k * x + l.m) * (MAX ? +1 : -1);
  }
};
```

1.6 回滾 DSU

```cpp
struct RollbackDSU {
  void init(int _n) {
    n = _n;
    sz.assign(n, -1);
    tag.clear();
  }
  int leader(int x) {
    while(sz[x] >= 0) x = sz[x];
    return x;
  }
  bool merge(int x, int y) {
    x = leader(x), y = leader(y);
    if(x == y) return false;
    if(-sz[x] < -sz[y]) swap(x, y);
    op.emplace_back(x, sz[x], y, sz[y]);
    sz[x] += sz[y];
    sz[y] = x;
    return true;
  }
  int size(int x) { return -sz[leader(x);] }
  void add_tag() { tag.push_back(op.size()); }
  void rollback() {
    int z = tag.back(); tag.pop_back();
    while(sz(op) > z) {
      auto [x, sx, y, sy] = op.back(); op.pop_back();
      sz[x] = sx;
      sz[y] = sy;
    }
  }
  int n;
  vector<int> sz, tag;
  vector<tuple<int, int, int, int>> op;
};
```

2 Flow-Matching

2.1 Dinic

```cpp
template<class T>
class Dinic {
public:
  struct Edge {
    int from, to;
    T cap;
    Edge(int x, int y, T z) : from(x), to(y), cap(z) {}
  };
  static constexpr T INF = numeric_limits<T>::max();
  int n;
  vector<Edge> edges;
  vector<vector<int>> g;
  vector<int> cur, h; // h : level graph
  Dinic() : n(0) {}
  explicit Dinic(int _n) : n(_n), g(_n) {}
  void add_edge(int u, int v, T c) {
```

```cpp
    assert(0 <= u && u < n);
    assert(0 <= v && v < n);
    g[u].push_back(edges.size());
    edges.emplace_back(u, v, c);
    g[v].push_back(edges.size());
    edges.emplace_back(v, u, 0);
  }
  bool bfs(int s, int t) {
    h.assign(n, -1);
    queue<int> que;
    h[s] = 0;
    que.push(s);
    while(!que.empty()) {
      int u = que.front(); que.pop();
      for(int i : g[u]) {
        const auto& e = edges[i];
        int v = e.to;
        if(e.cap > 0 && h[v] == -1) {
          h[v] = h[u] + 1;
          if(v == t) return true;
          que.push(v);
        }
      }
    }
    return false;
  }
  T dfs(int u, int t, T f) {
    if(u == t) return f;
    T r = f;
    for(int& i = cur[u]; i < (int) g[u].size(); ++i) {
      int j = g[u][i];
      const auto& e = edges[j];
      int v = e.to;
      T c = e.cap;
      if(c > 0 && h[v] == h[u] + 1) {
        T a = dfs(v, t, min(r, c));
        edges[j].cap -= a;
        edges[j ^ 1].cap += a;
        if((r -= a) == 0) return f;
      }
    }
    return f - r;
  }

  T flow(int s, int t, T f = INF) {
    assert(0 <= s && s < n);
    assert(0 <= t && t < n);
    T ans = 0;
    while(f > 0 && bfs(s, t)) {
      cur.assign(n, 0);
      T cur = dfs(s, t, f);
      ans += cur;
      f -= cur;
    }
    return ans;
  }
};
```

2.2 Flow 建模

- Maximum/Minimum flow with lower bound / Circulation problem
  1. Construct super source S and sink T.
  2. For each edge (x, y, l, u), connect $x \to y$ with capacity $u - l$.
  3. For each vertex v, denote by $in(v)$ the difference between the sum of incoming lower bounds and the sum of outgoing lower bounds.
  4. If $in(v) > 0$, connect $S \to v$ with capacity $in(v)$, otherwise, connect $v \to T$ with capacity $-in(v)$.
     - To maximize, connect $t \to s$ with capacity ∞ (skip this in circulation problem), and let f be the maximum flow from S to T. If $f \neq \sum_{v \in V, in(v)>0} in(v)$, there's no solution. Otherwise, the maximum flow from s to t is the answer.
     - To minimize, let f be the maximum flow from S to T. Connect $t \to s$ with capacity ∞ and let the flow from S to T be f'. If $f + f' \neq \sum_{v \in V, in(v)>0} in(v)$, there's no solution. Otherwise, f' is the answer.
  5. The solution of each edge e is $l_e + f_e$, where f_e corresponds to the flow of edge e on the graph.
- Construct minimum vertex cover from maximum matching M on bipartite graph (X, Y)
  1. Redirect every edge: $y \to x$ if $(x, y) \in M$, $x \to y$ otherwise.
  2. DFS from unmatched vertices in X.
  3. $x \in X$ is chosen iff x is unvisited.
  4. $y \in Y$ is chosen iff y is visited.
- Minimum cost cyclic flow
  1. Consruct super source S and sink T
  2. For each edge (x, y, c), connect $x \to y$ with $(cost, cap) = (c, 1)$ if $c > 0$, otherwise connect $y \to x$ with $(cost, cap) = (-c, 1)$
  3. For each edge with $c < 0$, sum these cost as K, then increase $d(y)$ by 1, decrease $d(x)$ by 1
  4. For each vertex v with $d(v) > 0$, connect $S \to v$ with $(cost, cap) = (0, d(v))$
  5. For each vertex v with $d(v) < 0$, connect $v \to T$ with $(cost, cap) = (0, -d(v))$
  6. Flow from S to T, the answer is the cost of the flow $C + K$
- Maximum density induced subgraph
  1. Binary search on answer, suppose we're checking answer T
  2. Construct a max flow model, let K be the sum of all weights
  3. Connect source $s \to v, v \in G$ with capacity K
  4. For each edge (u, v, w) in G, connect $u \to v$ and $v \to u$ with capacity w
  5. For $v \in G$, connect it with sink $v \to t$ with capacity $K + 2T - (\sum_{e \in E(v)} w(e)) - 2w(v)$
  6. T is a valid answer if the maximum flow $f < K|V|$
- Minimum weight edge cover
  1. For each $v \in V$ create a copy v', and connect $u' \to v'$ with weight $w(u, v)$.
  2. Connect $v \to v'$ with weight $2\mu(v)$, where $\mu(v)$ is the cost of the cheapest edge incident to v.
  3. Find the minimum weight perfect matching on G'.
- Project selection problem
  1. If $p_v > 0$, create edge (s, v) with capacity p_v; otherwise, create edge (v, t) with capacity $-p_v$.
  2. Create edge (u, v) with capacity w with w being the cost of choosing u without choosing v.
  3. The mincut is equivalent to the maximum profit of a subset of projects.
- 0/1 quadratic programming

$$\sum_x c_x x + \sum_y c_y \bar{y} + \sum_{xy} c_{xy} x\bar{y} + \sum_{xyx'y'} c_{xyx'y'}(x\bar{y} + x$$

can be minimized by the mincut of the following graph:

  1. Create edge (x, t) with capacity c_x and create edge (s, y) with capacity c_y.
  2. Create edge (x, y) with capacity c_{xy}.
  3. Create edge (x, y) and edge (x', y') with capacity $c_{xyx'y'}$.

2.3 KM

```cpp
template<class T>
struct KM {
  static constexpr T INF = numeric_limits<T>::max();
  int n, ql, qr;
  vector<vector<T>> w;
  vector<T> hl, hr, slk;
  vector<int> fl, fr, pre, qu;
  vector<bool> vl, vr;
  KM(int n) : n(n), w(n, vector<T>(n, -INF)), hl(n), hr(n), slk(n), fl(n), fr(n), pre(n), qu(n), vl(n), vr(n) {}
  void add_edge(int u, int v, int x) { w[u][v] = x; } // 最小值要加負號
  bool check(int x) {
    vl[x] = 1;
    if(fl[x] != -1) return vr[qu[qr++] = fl[x]] = 1;
    while(x != -1) swap(x, fr[fl[x] = pre[x]]);
    return 0;
  }
  void bfs(int s) {
    fill(slk.begin(), slk.end(), INF);
    fill(vl.begin(), vl.end(), 0);
    fill(vr.begin(), vr.end(), 0);
    ql = qr = 0, qu[qr++] = s, vr[s] = 1;
    while(true) {
      T d;
      while(ql < qr) {
        for(int x = 0, y = qu[ql++]; x < n; ++x) {
          if(!vl[x] && slk[x] >= (d = hl[x] + hr[y] - w[x][y])) {
            pre[x] = y;
            if(d) slk[x] = d;
            else if(!check(x)) return;
          }
        }
      }
      d = INF;
      for(int x = 0; x < n; ++x) if(!vl[x] && d > slk[x]) d = slk[x];
      for(int x = 0; x < n; ++x) {
        if(vl[x]) hl[x] += d;
        else slk[x] -= d;
        if(vr[x]) hr[x] -= d;
      }
      for(int x = 0; x < n; ++x) if(!vl[x] && !slk[x] && !check(x)) return;
    }
  }
  T solve() {
    fill(fl.begin(), fl.end(), -1);
    fill(fr.begin(), fr.end(), -1);
    fill(hr.begin(), hr.end(), 0);
    for(int i = 0; i < n; ++i) hl[i] = *max_element(w[i].begin(), w[i].end());
    for(int i = 0; i < n; ++i) bfs(i);
    T ans = 0;
    for(int i = 0; i < n; ++i) ans += w[i][fl[i]]; // i 跟 fl[i] 配對
    return ans;
  }
};
```

2.4 MCMF

```cpp
template<class S, class T>
class MCMF {
public:
  struct Edge {
    int from;
    int to;
    S cap;
    T cost;
    Edge(int u, int v, S x, T y) : from(u), to(v), cap(x), cost(y) {}
  };
  static constexpr S CAP_INF = numeric_limits<S>::max();
  static constexpr T COST_INF = numeric_limits<T>::max();
  int n;
  vector<Edge> edges;
  vector<vector<int>> g;
  vector<T> d;
  vector<bool> inq;
  vector<int> prev_edge;

  MCMF() : n(0) {}
  explicit MCMF(int _n) : n(_n), g(_n), d(_n), inq(_n), prev_edge(_n) {}

  void add_edge(int u, int v, S cap, T cost) {
    assert(0 <= u && u < n);
    assert(0 <= v && v < n);
    g[u].push_back(edges.size());
    edges.emplace_back(u, v, cap, cost);
    g[v].push_back(edges.size());
    edges.emplace_back(v, u, 0, -cost);
  }
```

```cpp

  bool spfa(int s, int t) {
    bool found = false;
    fill(d.begin(), d.end(), COST_INF);
    d[s] = 0;
    inq[s] = true;
    queue<int> que;
    que.push(s);
    while(!que.empty()) {
      int u = que.front(); que.pop();
      if(u == t) found = true;
      inq[u] = false;
      for(auto& id : g[u]) {
        const Edge& e = edges[id];
        if(e.cap > 0 && d[u] + e.cost < d[e.to]) {
          d[e.to] = d[u] + e.cost;
          prev_edge[e.to] = id;
          if(!inq[e.to]) {
            que.push(e.to);
            inq[e.to] = true;
          }
        }
      }
    }
    return found;
  }

  pair<S, T> flow(int s, int t, S f = CAP_INF) {
    assert(0 <= s && s < n);
    assert(0 <= t && t < n);
    S cap = 0;
    T cost = 0;
    while(f > 0 && spfa(s, t)) {
      S send = f;
      int u = t;
      while(u != s) {
        const Edge& e = edges[prev_edge[u]];
        send = min(send, e.cap);
        u = e.from;
      }
      u = t;
      while(u != s) {
        Edge& e = edges[prev_edge[u]];
        e.cap -= send;
        Edge& b = edges[prev_edge[u] ^ 1];
        b.cap += send;
        u = e.from;
      }
      cap += send;
      f -= send;
      cost += send * d[t];
    }
    return make_pair(cap, cost);
  }
};
```

2.5 一般圖最大匹配

```cpp
mt19937 rng(time(0));
struct GeneralMaxMatch {
  int n;
  vector<pair<int, int>> es;
  vector<int> g, vis, mate; // i 與 mate[i] 配對 (mate[i] == -1 代表沒有匹配)
  GeneralMaxMatch(int n) : n(n), g(n, -1), mate(n, -1) {}
  bool dfs(int u) {
    if(vis[u]) return false;
    vis[u] = true;
    for(int ei = g[u]; ei != -1; ) {
      auto [x, y] = es[ei]; ei = y;
      if(mate[x] == -1) {
        mate[mate[u] = x] = u;
        return true;
      }
    }
    for(int ei = g[u]; ei != -1; ) {
      auto [x, y] = es[ei]; ei = y;
      int nu = mate[x];
      mate[mate[u] = x] = u;
      mate[nu] = -1;
      if(dfs(nu)) return true;
      mate[mate[nu] = x] = nu;
      mate[u] = -1;
    }
    return false;
  }
  void add_edge(int a, int b) {
    auto f = [&](int a, int b) {
      es.emplace_back(b, g[a]);
      g[a] = es.size() - 1;
    };
    f(a, b); f(b, a);
  }
  int solve() {
    vector<int> o(n);
    iota(o.begin(), o.end(), 0);
    int ans = 0;
    for(int it = 0; it < 100; ++it) {
      shuffle(o.begin(), o.end(), rng);
      vis.assign(n, false);
      for(auto i : o) if(mate[i] == -1) ans += dfs(i);
    }
    return ans;
  }
};
```

2.6 一般圖最小權完美匹配

```cpp
struct Graph {
  // Minimum General Weighted Matching (Perfect Match) 0-base
  static const int MXN = 105;
  int n, edge[MXN][MXN];
  int match[MXN],dis[MXN],onstk[MXN];
  vector<int> stk;
  void init(int _n) {
    n = _n;
    for(int i=0; i<n; i++)
      for(int j=0; j<n; j++)
        edge[i][j] = 0;
  }
  void add_edge(int u, int v, int w) { edge[u][v] = edge[v][u] = w; }
  bool SPFA(int u){
    if(onstk[u]) return true;
    stk.push_back(u);
    onstk[u] = 1;
    for(int v=0; v<n; v++){
      if(u != v && match[u] != v && !onstk[v]){
        int m = match[v];
        if(dis[m] > dis[u] - edge[v][m] + edge[u][v]){
          dis[m] = dis[u] - edge[v][m] + edge[u][v];
          onstk[v] = 1;
          stk.push_back(v);
          if(SPFA(m)) return true;
          stk.pop_back();
          onstk[v] = 0;
        }
      }
    }
    onstk[u] = 0;
    stk.pop_back();
    return false;
  }
  int solve() {
    for(int i = 0; i < n; i += 2) match[i] = i + 1, match[i+1] = i;
    while(true) {
      int found = 0;
      for(int i=0; i<n; i++) dis[i] = onstk[i] = 0;
      for(int i=0; i<n; i++){
        stk.clear();
        if(!onstk[i] && SPFA(i)){
          found = 1;
          while(stk.size()>=2){
            int u = stk.back(); stk.pop_back();
            int v = stk.back(); stk.pop_back();
            match[u] = v;
            match[v] = u;
          }
        }
      }
      if(!found) break;
    }
    int ans = 0;
    for(int i=0; i<n; i++) ans += edge[i][match[i]];
    return ans / 2;
  }
}graph;
```

2.7 二分圖最大匹配

```cpp
struct bipartite_matching {
  int n, m; // 二分圖左右人數 (0 ~ n-1), (0 ~ m-1)
  vector<vector<int>> g;
  vector<int> lhs, rhs, dist; // i 與 lhs[i] 配對 (lhs[i] == -1 代表沒有配對)
  bipartite_matching(int _n, int _m) : n(_n), m(_m), g(_n), lhs(_n, -1), rhs(_m, -1), dist(_n) {}
  void add_edge(int u, int v) { g[u].push_back(v); }
  void bfs() {
    queue<int> q;
    for(int u = 0; u < n; ++u) {
      if(lhs[u] == -1) {
        q.push(u);
        dist[u] = 0;
      } else {
        dist[u] = -1;
      }
    }
    while(!q.empty()) {
      int u = q.front(); q.pop();
      for(auto v : g[u]) {
        if(rhs[v] != -1 && dist[rhs[v]] == -1) {
          dist[rhs[v]] = dist[u] + 1;
          q.push(rhs[v]);
        }
      }
    }
  }
  bool dfs(int u) {
    for(auto v : g[u]) {
      if(rhs[v] == -1) {
        rhs[lhs[u] = v] = u;
        return true;
      }
    }
    for(auto v : g[u]) {
      if(dist[rhs[v]] == dist[u] + 1 && dfs(rhs[v])) {
        rhs[lhs[u] = v] = u;
        return true;
      }
    }
    return false;
  }
  int solve() {
    int ans = 0;
    while(true) {
      bfs();
      int aug = 0;
      for(int u = 0; u < n; ++u) if(lhs[u] == -1) aug += dfs(u);
      if(aug == 0) break;
      ans += aug;
    }
    return ans;
  }
};
```

3 Math

3.1 BigInt

```cpp
template<typename T>
inline string to_string(const T& x){
  stringstream ss;
  return ss<<x,ss.str();
}
struct bigN:vector<ll>{
  const static int base=1000000000,width=log10(base);
  bool negative;
  bigN(const_iterator a,const_iterator b):vector<ll>(a,b){}
  bigN(string s){
    if(s.empty())return;
    if(s[0]=='-')negative=1,s=s.substr(1);
    else negative=0;
    for(int i=int(s.size())-1;i>=0;i-=width){
      ll t=0;
      for(int j=max(0,i-width+1);j<=i;++j)
        t=t*10+s[j]-'0';
      push_back(t);
    }
    trim();
  }
  template<typename T>
    bigN(const T &x):bigN(to_string(x)){}
  bigN():negative(0){}
  void trim(){
    while(size()&&!back())pop_back();
    if(empty())negative=0;
  }
  void carry(int _base=base){
    for(size_t i=0;i<size();++i){
      if(at(i)>=0&&at(i)<_base)continue;
      if(i+1u==size())push_back(0);
      int r=at(i)%_base;
      if(r<0)r+=_base;
      at(i+1)+=(at(i)-r)/_base,at(i)=r;
    }
  }
  int abscmp(const bigN &b)const{
    if(size()>b.size())return 1;
    if(size()<b.size())return -1;
    for(int i=int(size())-1;i>=0;--i){
      if(at(i)>b[i])return 1;
      if(at(i)<b[i])return -1;
    }
    return 0;
  }
  int cmp(const bigN &b)const{
    if(negative!=b.negative)return negative?-1:1;
    return negative?-abscmp(b):abscmp(b);
  }
  bool operator<(const bigN&b)const{return cmp(b)<0;}
  bool operator>(const bigN&b)const{return cmp(b)>0;}
  bool operator<=(const bigN&b)const{return cmp(b)<=0;}
  bool operator>=(const bigN&b)const{return cmp(b)>=0;}
  bool operator==(const bigN&b)const{return !cmp(b);}
  bool operator!=(const bigN&b)const{return cmp(b)!=0;}
  bigN abs()const{
    bigN res=*this;
    return res.negative=0, res;
  }
  bigN operator-()const{
    bigN res=*this;
    return res.negative=!negative,res.trim(),res;
  }
  bigN operator+(const bigN &b)const{
    if(negative)return -(-(*this)+(-b));
    if(b.negative)return *this-(-b);
    bigN res=*this;
    if(b.size()>size())res.resize(b.size());
    for(size_t i=0;i<b.size();++i)res[i]+=b[i];
    return res.carry(),res.trim(),res;
  }
  bigN operator-(const bigN &b)const{
    if(negative)return -(-(*this)-(-b));
    if(b.negative)return *this+(-b);
    if(abscmp(b)<0)return -(b-(*this));
    bigN res=*this;
    if(b.size()>size())res.resize(b.size());
    for(size_t i=0;i<b.size();++i)res[i]-=b[i];
    return res.carry(),res.trim(),res;
  }
  bigN operator*(const bigN &b)const{
    bigN res;
    res.negative=negative!=b.negative;
    res.resize(size()+b.size());
    for(size_t i=0;i<size();++i)
      for(size_t j=0;j<b.size();++j)
        if((res[i+j]+=at(i)*b[j])>=base){
          res[i+j+1]+=res[i+j]/base;
          res[i+j]%=base;
        }//???k??carry?|????
    return res.trim(),res;
  }
  bigN operator/(const bigN &b)const{
    int norm=base/(b.back()+1);
    bigN x=abs()*norm;
    bigN y=b.abs()*norm;
    bigN q,r;
    q.resize(x.size());
    for(int i=int(x.size())-1;i>=0;--i){
      r=r*base+x[i];
      int s1=r.size()<=y.size()?0:r[y.size()];
      int s2=r.size()<y.size()?0:r[y.size()-1];
      int d=(ll(base)*s1+s2)/y.back();
      r=r-y*d;
      while(r.negative)r=r+y,--d;
      q[i]=d;
    }
    q.negative=negative!=b.negative;
    return q.trim(),q;
  }
  bigN operator%(const bigN &b)const{
    return *this-(*this/b)*b;
  }
  friend istream& operator>>(istream &ss, bigN &b){
    string s;
    return ss>>s, b=s, ss;
  }
  friend ostream& operator<<(ostream &ss, const bigN &b){
    if(b.negative)ss<<'-';
    ss<<(b.empty()?0:b.back());
    for(int i=int(b.size())-2;i>=0;--i)
      ss<<setw(width)<<setfill('0')<<b[i];
    return ss;
  }
  template<typename T>
  operator T(){
    stringstream ss;
    ss<<*this;
    T res;
    return ss>>res,res;
  }
};
```

3.2 Chinese-Remainder

```cpp
// (rem, mod) {0, 0} for no solution
pair<ll, ll> crt(ll r0, ll m0, ll r1, ll m1) {
  r0 = (r0 % m0 + m0) % m0;
  r1 = (r1 % m1 + m1) % m1;
  if(m0 < m1) swap(r0, r1), swap(m0, m1);
  if(m0 % m1 == 0) {
    if(r0 % m1 != r1) return {0, 0};
  }
  ll g, im, qq;
  g = ext_gcd(m0, m1, im, qq);
  ll u1 = (m1 / g);
  if((r1 - r0) % g) return {0, 0};
  ll x = (r1 - r0) / g % u1 * im % u1;
  r0 += x * m0;
  m0 *= u1;
  if(r0 < 0) r0 += m0;
  return {r0, m0};
}
```

3.3 Discrete-Log

```cpp
int discrete_log(int a, int b, int m) {
  assert(b < m);
  if(b == 1 || m == 1) return 0;
  int n = sqrt(m) + 2, e = 1, f = 1, j = 1;
  unordered_map<int, int> A; // becareful!!!
  while(j <= n && (e = f = 1LL * e * a % m) != b) {
    A[1LL * e * b % m] = j++;
  }
  if(e == b) return j;
  if(__gcd(m, e) == __gcd(m, b))  {
    for(int i = 2; i < n + 2; ++i) {
      e = 1LL * e * f % m;
      if(A.find(e) != A.end()) {
        return n * i - A[e];
      }
    }
  }
  return -1;
}
```

3.4 extgcd

```cpp
// ax + by = gcd(a, b)
ll ext_gcd(ll a, ll b, ll& x, ll& y) {
  if(b == 0) {
    x = 1, y = 0;
    return a;
  }
  ll x1, y1;
  ll g = ext_gcd(b, a % b, x1, y1);
  x = y1, y = x1 - (a / b) * y1;
  return g;
}
```

3.5 Floor-Sum

```cpp
// sum_0^{n-1} floor((a * i + b) / m) in log(n + m + a + b)
ll floor_sum(ll n, ll m, ll a, ll b) {
  ll ans = 0;
  if(a >= m) ans += (n - 1) * n * (a / m) / 2, a %= m;
  if(b >= m) ans += n * (b / m), b %= m;
  ll y_max = (a * n + b) / m, x_max = (y_max * m - b);
  if(y_max == 0) return ans;
  ans += (n - (x_max + a - 1) / a) * y_max;
  ans += floor_sum(y_max, a, m, (a - x_max % a) % a);
  return ans;
}
```

3.6 Miller-Rabin

```cpp
bool is_prime(ll n, vector<ll> x) {
  ll d = n - 1;
  d >>= __builtin_ctzll(d);
  for(auto a : x) {
    if(n <= a) break;
    ll t = d;
    ll y = 1, b = t;
    while(b) {
      if(b & 1) y = __int128(y) * a % n;
```

```cpp
      a = __int128(a) * a % n;
      b >>= 1;
    }
    while(t != n - 1 && y != 1 && y != n -
        1) {
      y = __int128(y) * y % n;
      t <<= 1;
    }
    if(y != n - 1 && t % 2 == 0) {
      return false;
    }
  }
  return true;
}

bool is_prime(ll n) {
  if(n <= 1) return false;
  if(n % 2 == 0) return n == 2;
  if(n < (1LL << 30)) return is_prime(n, {2,
      7, 61});
  return is_prime(n, {2, 325, 9375, 28178,
      450775, 9780504, 1795265022});
}
```

3.7 Pollard-Rho

```cpp
void PollardRho(map<ll, int>& mp, ll n) {
  if(n == 1) return;
  if(is_prime(n)) return mp[n]++, void();
  if(n % 2 == 0) {
    mp[2] += 1;
    PollardRho(mp, n / 2);
    return;
  }
  ll x = 2, y = 2, d = 1, p = 1;
  #define f(x, n, p) ((__int128(x) * x % n +
      p) % n)
  while(true) {
    if(d != 1 && d != n) {
      PollardRho(mp, d);
      PollardRho(mp, n / d);
      return;
    }
    p += (d == n);
    x = f(x, n, p), y = f(f(y, n, p), n, p);
    d = __gcd(abs(x - y), n);
  }
  #undef f
}

vector<ll> get_divisors(ll n) {
  if(n == 0) return {};
  map<ll, int> mp;
  PollardRho(mp, n);
  vector<pair<ll, int>> v(mp.begin(), mp.end
      ());
  vector<ll> res;
  auto f = [&](auto f, int i, ll x) -> void
      {
    if(i == (int) v.size()) {
      res.push_back(x);
      return;
    }
    for(int j = v[i].second; ; j--) {
      f(f, i + 1, x);
      if(j == 0) break;
      x *= v[i].first;
    }
  };
  f(f, 0, 1);
  sort(res.begin(), res.end());
  return res;
}
```

3.8 Primes

```cpp
/* 12721 13331 14341 75577 123457 222557
   556679 999983 1097774749 1076767633
   100102021 999997771 1001010013
   1000512343 987654361 999991231 999888733
    98789101 987777733 999991921 1010101333
    1010102101 1000000000039
   1000000000000037 2305843009213693951
   4611686018427387847 9223372036854775783
   18446744073709551557 */
```

3.9 估計值

- Estimation
  - The number of divisors of n is at most around 100 for $n < 5e4$, 500 for $n < 1e7$, 2000 for $n < 1e10$, 200000 for $n < 1e19$.
  - The number of ways of writing n as a sum of positive integers, disregarding the order of the summands. 1, 1, 2, 3, 5, 7, 11, 15, 22, 30 for $n = 0 \sim 9$, 627 for $n = 20$, $\sim 2e5$ for $n = 50$, $\sim 2e8$ for $n = 100$.
  - Total number of partitions of n distinct elements: $B(n) = 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, 678570, 4213597, 27644437, 190899322, \ldots$.

3.10 定理

- Cramer's rule

$$\begin{matrix} ax + by = e \\ cx + dy = f \end{matrix} \Rightarrow \begin{matrix} x = \dfrac{ed - bf}{ad - bc} \\ y = \dfrac{af - ec}{ad - bc} \end{matrix}$$

- Vandermonde's Identity

$$C(n + m, k) = \sum_{i=0}^{k} C(n, i)C(m, k - i)$$

- Kirchhoff's Theorem

  Denote L be a $n \times n$ matrix as the Laplacian matrix of graph G, where $L_{ii} = d(i)$, $L_{ij} = -c$ where c is the number of edge (i, j) in G.
  - The number of undirected spanning in G is $|\det(\tilde{L}_{11})|$.
  - The number of directed spanning tree rooted at r in G is $|\det(\tilde{L}_{rr})|$.

- Tutte's Matrix

  Let D be a $n \times n$ matrix, where $d_{ij} = x_{ij}$ (x_{ij} is chosen uniformly at random) if $i < j$ and $(i, j) \in E$, otherwise $d_{ij} = -d_{ji}$. $\frac{rank(D)}{2}$ is the maximum matching on G.

- Cayley's Formula
  - Given a degree sequence $d_1, d_2, \ldots, d_n$ for each labeled vertices, there are $\frac{(n-2)!}{(d_1-1)!(d_2-1)!\cdots(d_n-1)!}$ spanning trees.
  - Let $T_{n,k}$ be the number of labeled forests on n vertices with k components, such that vertex $1, 2, \ldots, k$ belong to different components. Then $T_{n,k} = kn^{n-k-1}$.

- Erd□s–Gallai theorem

  A sequence of nonnegative integers $d_1 \geq \cdots \geq d_n$ can be represented as the degree sequence of a finite simple graph on n vertices if and only if $d_1 + \cdots + d_n$ is even and $\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min(d_i, k)$ holds for every $1 \leq k \leq n$.

- Gale–Ryser theorem

  A pair of sequences of nonnegative integers $a_1 \geq \cdots \geq a_n$ and $b_1, \ldots, b_n$ is bigraphic if and only if $\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} b_i$ and $\sum_{i=1}^{k} a_i \leq \sum_{i=1}^{n} \min(b_i, k)$ holds for every $1 \leq k \leq n$.

- Fulkerson–Chen–Anstee theorem

  A sequence $(a_1, b_1), \ldots, (a_n, b_n)$ of nonnegative integer pairs with $a_1 \geq \cdots \geq a_n$ is digraphic if and only if $\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} b_i$ and $\sum_{i=1}^{k} a_i \leq \sum_{i=1}^{k} \min(b_i, k-1) + \sum_{i=k+1}^{n} \min(b_i, k)$ holds for every $1 \leq k \leq n$.

- M□bius inversion formula
  - $f(n) = \sum_{d|n} g(d) \Leftrightarrow g(n) = \sum_{d|n} \mu(d) f(\frac{n}{d})$
  - $f(n) = \sum_{n|d} g(d) \Leftrightarrow g(n) = \sum_{n|d} \mu(\frac{d}{n}) f(d)$

- Spherical cap
  - A portion of a sphere cut off by a plane.
  - r: sphere radius, a: radius of the base of the cap, h: height of the cap, θ: $\arcsin(a/r)$.
  - Volume $= \pi h^2(3r - h)/3 = \pi h(3a^2 + h^2)/6 = \pi r^3(2 + \cos\theta)(1 - \cos\theta)^2/3$.
  - Area $= 2\pi rh = \pi(a^2 + h^2) = 2\pi r^2(1 - \cos\theta)$.

3.11 整數除法

```cpp
ll floor_div(ll a, ll b) {
  return a / b - ((a ^ b) < 0 && a % b != 0)
      ;
}
ll ceil_div(ll a, ll b) {
  return a / b + ((a ^ b) > 0 && a % b != 0)
      ;
}
```

3.12 數字

- Bernoulli numbers

  $B_0 - 1, B_1^{\pm} = \pm\frac{1}{2}, B_2 = \frac{1}{6}, B_3 = 0$

  $\sum_{j=0}^{m} \binom{m+1}{j} B_j = 0$, EGF is $B(x) = \frac{x}{e^x - 1} = \sum_{n=0}^{\infty} B_n \frac{x^n}{n!}$.

  $S_m(n) = \sum_{k=1}^{n} k^m = \frac{1}{m+1} \sum_{k=0}^{m} \binom{m+1}{k} B_k^+ n^{m+1-k}$

- Stirling numbers of the second kind Partitions of n distinct elements into exactly k groups.

  $S(n, k) = S(n-1, k-1) + kS(n-1, k)$, $S(n, 1) = S(n, n) = 1$

  $S(n, k) = \frac{1}{k!} \sum_{i=0}^{k} (-1)^{k-i} \binom{k}{i} i^n$

  $x^n = \sum_{i=0}^{n} S(n, i)(x)_i$

- Pentagonal number theorem

  $\prod_{n=1}^{\infty} (1 - x^n) = 1 + \sum_{k=1}^{\infty} (-1)^k \left(x^{k(3k+1)/2} + x^{k(3k-1)/2}\right)$

- Catalan numbers

  $C_n^{(k)} = \frac{1}{(k-1)n+1} \binom{kn}{n}$

  $C^{(k)}(x) = 1 + x[C^{(k)}(x)]^k$

- 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. $\pi(j) > j$.

  $E(n, k) = (n-k)E(n-1, k-1) + (k+1)E(n-1, k)$

  $E(n, 0) = E(n, n-1) = 1$

  $E(n, k) = \sum_{j=0}^{k} (-1)^j \binom{n+1}{j} (k+1-j)^n$

3.13 歐幾里得類算法

- $m = \lfloor \frac{an+b}{c} \rfloor$
- Time complexity: $O(\log n)$

$$f(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor$$
$$= \begin{cases} \lfloor \frac{a}{c} \rfloor \cdot \frac{n(n+1)}{2} + \lfloor \frac{b}{c} \rfloor \cdot (n+1) & \\ +f(a \bmod c, b \bmod c, c, n), & a \geq c \vee b \geq c \\ 0, & n < 0 \vee a = 0 \\ nm - f(c, c-b-1, a, m-1), & \text{otherwise} \end{cases}$$

$$g(a,b,c,n) = \sum_{i=0}^{n} i \lfloor \frac{ai+b}{c} \rfloor$$
$$= \begin{cases} \lfloor \frac{a}{c} \rfloor \cdot \frac{n(n+1)(2n+1)}{6} + \lfloor \frac{b}{c} \rfloor \cdot \frac{n(n+1)}{2} & \\ +g(a \bmod c, b \bmod c, c, n), & a \geq c \vee b \geq c \\ 0, & n < 0 \vee a = 0 \\ \frac{1}{2} \cdot (n(n+1)m - f(c, c-b-1, a, m-1) & \\ -h(c, c-b-1, a, m-1)), & \text{otherwise} \end{cases}$$

$$h(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor^2$$
$$= \begin{cases} \lfloor \frac{a}{c} \rfloor^2 \cdot \frac{n(n+1)(2n+1)}{6} + \lfloor \frac{b}{c} \rfloor^2 \cdot (n+1) & \\ +\lfloor \frac{a}{c} \rfloor \cdot \lfloor \frac{b}{c} \rfloor \cdot n(n+1) & \\ +h(a \bmod c, b \bmod c, c, n) & \\ +2\lfloor \frac{a}{c} \rfloor \cdot g(a \bmod c, b \bmod c, c, n) & \\ +2\lfloor \frac{b}{c} \rfloor \cdot f(a \bmod c, b \bmod c, c, n), & a \geq c \vee b \geq c \\ 0, & n < 0 \vee a = 0 \\ nm(m+1) - 2g(c, c-b-1, a, m-1) & \\ -2f(c, c-b-1, a, m-1) - f(a,b,c,n), & \text{otherwise} \end{cases}$$

3.14 生成函數

- Ordinary Generating Function $A(x) = \sum_{i \geq 0} a_i x^i$
  - $A(rx) \Rightarrow r^n a_n$
  - $A(x) + B(x) \Rightarrow a_n + b_n$
  - $A(x)B(x) \Rightarrow \sum_{i=0}^{n} a_i b_{n-i}$
  - $A(x)^k \Rightarrow \sum_{i_1+i_2+\cdots+i_k=n} a_{i_1} a_{i_2} \ldots a_{i_k}$
  - $xA(x)' \Rightarrow na_n$
  - $\frac{A(x)}{1-x} \Rightarrow \sum_{i=0}^{n} a_i$
- Exponential Generating Function $A(x) = \sum_{i \geq 0} \frac{a_i}{i!} x_i$
  - $A(x) + B(x) \Rightarrow a_n + b_n$
  - $A^{(k)}(x) \Rightarrow a_{n+k}$
  - $A(x)B(x) \Rightarrow \sum_{i=0}^{n} \binom{n}{i} a_i b_{n-i}$
  - $A(x)^k \Rightarrow \sum_{i_1+i_2+\cdots+i_k=n} \binom{n}{i_1, i_2, \ldots, i_k} a_{i_1} a_{i_2} \ldots a_{i_k}$
  - $xA(x) \Rightarrow na_n$
- Special Generating Function
  - $(1+x)^n = \sum_{i \geq 0} \binom{n}{i} x^i$
  - $\frac{1}{(1-x)^n} = \sum_{i \geq 0} \binom{i}{n-1} x^i$

ACM ICPC Team Reference - Angry Crow Takes Flight!

Contents

ACM ICPC Judge Test - Angry Crow Takes Flight!

C++ Resource Test

```cpp
#include <bits/stdc++.h>
using namespace std;

namespace system_test {

const size_t KB = 1024;
const size_t MB = KB * 1024;
const size_t GB = MB * 1024;

size_t block_size, bound;
void stack_size_dfs(size_t depth = 1) {
  if (depth >= bound)
    return;
  int8_t ptr[block_size]; // 若無法編譯將 block_size 改成常數
  memset(ptr, 'a', block_size);
  cout << depth << endl;
  stack_size_dfs(depth + 1);
}

void stack_size_and_runtime_error(size_t block_size, size_t bound = 1024) {
  system_test::block_size = block_size;
  system_test::bound = bound;
  stack_size_dfs();
}

double speed(int iter_num) {
  const int block_size = 1024;
  volatile int A[block_size];
  auto begin = chrono::high_resolution_clock::now();
  while (iter_num--)
    for (int j = 0; j < block_size; ++j)
      A[j] += j;
  auto end = chrono::high_resolution_clock::now();
  chrono::duration<double> diff = end - begin;
  return diff.count();
}

void runtime_error_1() {
  // Segmentation fault
  int *ptr = nullptr;
  *(ptr + 7122) = 7122;
}

void runtime_error_2() {
  // Segmentation fault
  int *ptr = (int *)memset;
  *ptr = 7122;
}

void runtime_error_3() {
  // munmap_chunk(): invalid pointer
  int *ptr = (int *)memset;
  delete ptr;
}

void runtime_error_4() {
  // free(): invalid pointer
  int *ptr = new int[7122];
  ptr += 1;
  delete[] ptr;
}

void runtime_error_5() {
  // maybe illegal instruction
  int a = 7122, b = 0;
  cout << (a / b) << endl;
}

void runtime_error_6() {
  // floating point exception
  volatile int a = 7122, b = 0;
  cout << (a / b) << endl;
}

void runtime_error_7() {
  // call to abort.
  assert(false);
}

} // namespace system_test

#include <sys/resource.h>
void print_stack_limit() { // only work in Linux
  struct rlimit l;
  getrlimit(RLIMIT_STACK, &l);
  cout << "stack_size = " << l.rlim_cur << " byte" << endl;
}
```