Team Note of Baka Team

Shine - QioCas - Asamai

Compiled on November 7, 2024

Contents

1 Template

1.1 Template

```cpp
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

#define MASK(k) 1LL << (k)
#define BIT(x, k) ((x) >> (k) & 1)
#define all(x) (x).begin(), (x).end()

template<class T> bool minimize(T& a, const T& b) {
    if(a > b) return a = b, true;
    return false;
}

template<class T> bool maximize(T& a, const T& b) {
    if(a < b) return a = b, true;
    return false;
}

ll fdiv(ll a, ll b) {
    assert(b != 0);
    if(b < 0) a *= -1, b *= -1;
    return a >= 0 ? a / b : (a + 1) / b - 1;
}

ll cdiv(ll a, ll b) {
    assert(b != 0);
    if(b < 0) a *= -1, b *= -1;
    return a <= 0 ? a / b : (a - 1) / b + 1;
}
```

1.2 Debug

```cpp
void debug_utils() {}

template<class T, class ...U> void debug_utils(T a, U... b) {
    cerr << a;
    if(sizeof...(b)) { cerr << ", "; debug_utils(b...);}
}

#define debug(...) { cerr << #__VA_ARGS__ << " = ";
debug_utils(__VA_ARGS__); cerr << "\n"; }

template<class Tp1, class Tp2>
ostream& operator << (ostream& cout, pair<Tp1, Tp2> val) {
    return cout << val.first << " " << val.second << "\n";
}

template<class Data, class Tp =
decltype(declval<Data>().begin())>
typename enable_if<!is_same<Data, string>::value,
ostream&>::type
operator << (ostream& cout, Data val) {
    cout << "[";
    for(auto i = val.begin(); i != val.end(); ++i)
        cout << (i == val.begin() ? "" : " ") << *i;
    return cout << "]";
}

template<class Data, class = decltype(declval<Data>().top())>
ostream& operator << (ostream& cout, Data val) {
    cout << "[";
    for(; val.size(); val.pop())
        cout << val.top() << (val.size() == 1 ? "" : " ");
    return cout << "]";
}

template<class Tp> ostream& operator << (ostream& cout,
queue<Tp> val) {
    cout << "[";
    for(; val.size(); val.pop())
        cout << val.front() << (val.size() == 1 ? "" : " ");
    return cout << "]";
}
```

1.3 Generate

```cpp
mt19937_64
rng(chrono::steady_clock::now().time_since_epoch().count());

ll randint(ll a, ll b) {
    return uniform_int_distribution<ll> (a, b) (rng);
}
```

2 Data Structure

2.1 Mutidimensional Vector

```
template<class Tp, int D = 1>
struct Tvector : public vector<Tvector<Tp, D - 1>> {
    template <class... Args>
    Tvector(int n = 0, Args... args) : vector<Tvector<Tp, D -
    1>>(n, Tvector<Tp, D - 1>(args...)) {}
};

template <class Tp>
struct Tvector<Tp, 1> : public vector<Tp> {
    Tvector(int n = 0, Tp val = Tp()) : vector<Tp>(n, val) {}
};
```

2.2 Rollback

```
vector<pair<int*, int>> event;

void assign(int* u, int v) {
    event.push_back({u, exchange(*u, v)});
}

void rollback(int t) {
    for(; (int) event.size() > t; event.pop_back()) {
        *event.back().first = event.back().second;
    }
}
```

2.3 Wavelet Tree

```
struct wavelet {
    wavelet *left, *right;
    vector<ll> pref;
    int wl, wr;

    wavelet() { }
    wavelet(int tl, int tr, int pL, int pR, vector<ll> &v) {
        wl = tl, wr = tr;
        if (wl == wr || pL > pR) return;

        int mid = (wl + wr) >> 1;

        pref.pb(0);
        for (int i = pL; i <= pR; i++)
            pref.pb(pref.back() + (v[i] <= mid));

        ll piv = stable_partition(v.begin() + pL, v.begin() +
        pR + 1
                    , [&](int x){ return x <= mid; }) -
                    v.begin() - 1;

        left = new wavelet(wl, mid, pL, piv, v);
        right = new wavelet(mid + 1, wr, piv + 1, pR, v);
    }

    ll findKth(int k, int l, int r) {
        if (wl == wr) return wl;
        // cout << wl << " " << wr << " " << k << '\n';

        int amt = pref[r] - pref[l - 1];
        int lBound = pref[l - 1];
        int rBound = pref[r];

        if (amt >= k) return left->findKth(k, lBound + 1,
        rBound);

        return right->findKth(k - amt, l - lBound, r -
        rBound);
    }
};
```

2.4 Sparse lichao tree

```
struct Line {
  ll m, b;
  Line(ll _m = 0, ll _b = INF * 8) : m(_m), b(_b) {}
};

ll F(Line l, ll x) {
  return l.m * x + l.b;
}
struct lichao_t {
  lichao_t *left = nullptr, *right = nullptr;
  Line mn;
  lichao_t(ll tl = 0, ll tr = 0) {}
  void Update(ll tl, ll tr, Line nLine) {
    ll mid = (tl + tr) >> 1;
    bool pLeft = (F(nLine, tl) < F(mn, tl));
    bool pMid = (F(nLine, mid) < F(mn, mid));
    if (pMid)
      swap(mn, nLine);
    if (tl == tr)
      return;
    if (pLeft != pMid) {
      if (left == nullptr)
        left = new lichao_t();
      left->Update(tl, mid, nLine);
    } else {
      if (right == nullptr)
        right = new lichao_t();
      right->Update(mid + 1, tr, nLine);
    }
  }
  ll Query(ll tl, ll tr, ll x) {
    if (tl == tr)
      return F(mn, x);
    ll mid = (tl + tr) >> 1;
    ll res = F(mn, x);
    if (x <= mid) {
      if (left != nullptr)
        minimize(res, left->Query(tl, mid, x));
    } else {
      if (right != nullptr)
        minimize(res, right->Query(mid + 1, tr, x));
    }
    return res;
  }
};
```

2.5 Line Container

```
struct Line {
  mutable ll k, m, p;
  bool operator<(const Line& o) const { return k < o.k; }
  bool operator<(ll x) const { return p < x; }
};

struct LineContainer : multiset<Line, less<>> {
  // (for doubles, use inf = 1/.0, div(a,b) = a/b)
  static const ll inf = LLONG_MAX;
  ll div(ll a, ll b) { // floored division
    return a / b - ((a ^ b) < 0 && a % b); }
  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 = div(y->m - x->m, x->k - y->k);
    return x->p >= y->p;
  }
  void add(ll k, ll 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 query(ll x) {
    assert(!empty());
    auto l = *lower_bound(x);
    return l.k * x + l.m;
  }
};
```

2.6 Ordered Set

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

using namespace __gnu_pbds;

template<class T>
using OrderedTree = tree<T, null_type, less<T>, rb_tree_tag,
    tree_order_statistics_node_update>;
```

2.7 Treap

```
struct Lazy {
    // [...]
};

struct Node {
    // [...]
};

Node operator + (Node u, Node v) {
    // [...]
    return Node{};
}

struct Tnode {
    Tnode *l = NULL, *r = NULL;
    Node node, key;
    Lazy lazy;
    int size = 1, prior = 0;
    Tnode(Node key = Node{}, int prior = randint(-1 << 30, 1
    << 30)): node(key), key(key), prior(prior) {}
};

using Pnode = Tnode*;

Pnode IgnoreNode;

// PreProcess.
void InitIgnoreNode() {
    IgnoreNode = new Tnode{};
    IgnoreNode->size = 0;
}

#define NODE(x) (x ? x : IgnoreNode)

Pnode FIX(Pnode u) {
    if(u) {
        u->size = NODE(u->l)->size + 1 + NODE(u->r)->size;
        u->node = NODE(u->l)->node + NODE(u)->key +
        NODE(u->r)->node;
    }
    return u;
}

void update_node(Pnode u, Lazy val) {
    if(!u) return;
    // [...]
}

void Down(Pnode t) {

}

Pnode merge(Pnode l, Pnode r) {
    if(!l || !r) return (l ? l : r);
    Down(l); Down(r);
    if(l->prior > r->prior) {
        l->r = merge(l->r, r);
        return FIX(l);
    } else {
        r->l = merge(l, r->l);
        return FIX(r);
    }
}

pair<Pnode, Pnode> split(Pnode t, int k) {
    if(!t) return {NULL, NULL};
    else Down(t);
    Pnode l = NULL, r = NULL;
    if(k <= NODE(t->l)->size) tie(l, t->l) = split(t->l, k), r
    = t;
    else                tie(t->r, r) = split(t->r, k - 1 -
    NODE(t->l)->size), l = t;
    FIX(t);
    return {l, r};
}

tuple<Pnode, Pnode, Pnode> split(Pnode t, int u, int v) {
    if(!t) return {NULL, NULL, NULL};
    Pnode l = NULL, m = NULL, r = NULL;
    tie(t, r) = split(t, v + 1);
    tie(l, m) = split(t, u);
    return {l, m, r};
}

void DFS(Pnode t) {
    if(!t) return;
    Down(t);
    DFS(t->l);
    // [...]
    DFS(t->r);
}
```

3 Graphs

3.1 Max Matching (Hopcroft)

```
/*
hopcroft karp for finding maximum matching on bipartite graphs
time complexity : O(E.sqrt(V))
layL[i] is the bfs layer of the ith vertex of left partition
layR[i] is for the ith vertex of the right partition
mtR[i] is the vertex matched with the ith vertex of right
partition, -1 if unmatched
adj[i] is list of neighbours of ith vertex of left partition
*/

struct hopcroft {
    ll nl, nr;

    // adj list of the left partition
    vector<vector<int>> adj;
    vector<int> layL, layR, mtR, cur, nxt;

    vector<bool> vis[2], mark;

    hopcroft(int n, int m) : nl(n), nr(m) {
        adj.assign(n, {});
    }

    bool dfs(int u, int len) {
        if(layL[u] != len) return 0;

        layL[u] = -1;
        for (int v : adj[u]) {
            if (layR[v] == len + 1) {
                layR[v] = 0;
                if (mtR[v] == -1 || dfs(mtR[v], len + 1))
                    return mtR[v] = u, 1;
            }
        }

        return 0;
    }

    ll max_matching() {
        layL.assign(nl, 0);
        layR.assign(nr, 0);
        mtR.assign(nr, -1);

        ll res = 0;

        while (true) {
            fill(all(layL), 0);
        fill(all(layR), 0);
        cur.clear();

        for (int u : mtR)
            if(u != -1) layL[u] = -1;

        for (int i = 0; i < sz(adj); i++)
            if (layL[i] != -1)
                cur.pb(i);

        bool isLast = false;
        for (int lay = 1; ; lay++) {
            nxt.clear();

            for (int u : cur) {
                for (int v : adj[u]) {
                    if (mtR[v] == -1) {
                        layR[v] = lay;
                        isLast = true;
                    }
                    else if (mtR[v] != u && !layR[v]) {
                        layR[v] = lay;
                        nxt.pb(mtR[v]);
                    }
                }
            }

            if (isLast) break;

            if (nxt.empty()) return res;

            for (int u : nxt)
                layL[u] = lay;

            swap(cur, nxt);
        }

        for (int i = 0; i < sz(adj); i++)
            res += dfs(i, 0);
    }
}

void dfs2(int u, int l) {
    vis[l][u] = true;

    if (!l) {
        for (int v : adj[u]) {
            if (!vis[1][v])
                dfs2(v, 1);
        }
    }
    else {
        if (mtR[u] != -1 && !vis[0][mtR[u]])
            dfs2(mtR[u], 0);
    }
}

//edges in matching -> right to left, else left to right
//return {left/right, index} of minimum cover
vector<pll> minCover() {
    vis[0].assign(nl, false);
    vis[1].assign(nr, false);
    mark.assign(nl, false);

    vector<pll> res;
    for (int i = 0; i < nr; i++)
        if (mtR[i] != -1)
            mark[mtR[i]] = true;

    for (int i = 0; i < nl; i++)
        if (!mark[i])
            dfs2(i, 0);

    //unvisited of the left and visited of the right is in
    min cover
    for (int i = 0; i < nl; i++)
        if (!vis[0][i])
            res.pb({0, i});

    for (int i = 0; i < nr; i++)
        if (vis[1][i])
            res.pb({1, i});

    return res;
  }
};
```

3.2 Max Matching (Blossom)

```
/* Complexity: O(E*sqrt(V))
*/
struct Blossom {
    static const int MAXV = 1e3 + 5;
    static const int MAXE = 1e6 + 5;
    int n, E, lst[MAXV], next[MAXE], adj[MAXE];
    int nxt[MAXV], mat[MAXV], dad[MAXV], col[MAXV];
    int que[MAXV], qh, qt;
    int vis[MAXV], act[MAXV];
    int tag, total;

    void init(int n) {
        this->n = n;
        for (int i = 0; i <= n; i++) {
            lst[i] = nxt[i] = mat[i] = vis[i] = 0;
        }
        E = 1, tag = total = 0;
    }
    void add(int u,int v) {
        if (!mat[u] && !mat[v]) mat[u] = v, mat[v] = u,
        total++;
        E++, adj[E] = v, next[E] = lst[u], lst[u] = E;
        E++, adj[E] = u, next[E] = lst[v], lst[v] = E;
    }
    int lca(int u, int v) {
        tag++;
        for(; ; swap(u, v)) {
            if (u) {
                if (vis[u = dad[u]] == tag) {
                    return u;
                }
                vis[u] = tag;
                u = nxt[mat[u]];
            }
        }
    }
    void blossom(int u, int v, int g) {
        while (dad[u] != g) {
            nxt[u] = v;
            if (col[mat[u]] == 2) {
                col[mat[u]] = 1;
                que[++qt] = mat[u];
            }
            if (u == dad[u]) dad[u] = g;
            if (mat[u] == dad[mat[u]]) dad[mat[u]] = g;
            v = mat[u];
            u = nxt[v];
        }
}

    int augument(int s) {
        for (int i = 1; i <= n; i++) {
            col[i] = 0;
            dad[i] = i;
        }
        qh = 0; que[qt = 1] = s; col[s] = 1;
        for (int u, v, i; qh < qt; ) {
            act[u = que[++qh]] = 1;
            for (i = lst[u];i ; i = next[i]) {
                v = adj[i];
                if (col[v] == 0) {
                    nxt[v] = u;
                    col[v] = 2;
                    if (!mat[v]) {
                        for (; v; v = u) {
                            u = mat[nxt[v]];
                            mat[v] = nxt[v];
                            mat[nxt[v]] = v;
                        }
                        return 1;
                    }
                    col[mat[v]] = 1;
                    que[++qt] = mat[v];
                }
                else if (dad[u] != dad[v] && col[v] == 1) {
                    int g = lca(u, v);
                    blossom(u, v, g);
                    blossom(v, u, g);
                    for (int j = 1; j <= n; j++) {
                        dad[j] = dad[dad[j]];
                    }
                }
            }
        }
        return 0;
    }
    int maxmat() {
        for (int i = 1; i <= n; i++) {
            if (!mat[i]) {
                total += augument(i);
            }
        }
        return total;
    }
}
```

3.3 Max Flow (Push Relabel)

```
/**
 * Author: Simon Lindholm
 * Date: 2015-02-24
 * License: CC0
 * Source: Wikipedia, tinyKACTL
 * Description: Push-relabel using the highest label selection
rule and the gap heuristic. Quite fast in practice.
 *  To obtain the actual flow, look at positive values only.
 * Time: $O(V^2\sqrt E)$
 * Status: Tested on Kattis and SPOJ, and stress-tested
 */
#pragma once


struct PushRelabel {
  struct Edge {
    int dest, back;
    ll f, c;
  };
  vector<vector<Edge>> g;
  vector<ll> ec;
  vector<Edge*> cur;
  vector<vi> hs; vi H;
  PushRelabel(int n) : g(n), ec(n), cur(n), hs(2*n), H(n) {}

  void addEdge(int s, int t, ll cap, ll rcap=0) {
    if (s == t) return;
    g[s].push_back({t, sz(g[t]), 0, cap});
    g[t].push_back({s, sz(g[s])-1, 0, rcap});
  }

  void addFlow(Edge& e, ll f) {
    Edge &back = g[e.dest][e.back];
    if (!ec[e.dest] && f) hs[H[e.dest]].push_back(e.dest);
    e.f += f; e.c -= f; ec[e.dest] += f;
    back.f -= f; back.c += f; ec[back.dest] -= f;
  }
  ll calc(int s, int t) {
    int v = sz(g); H[s] = v; ec[t] = 1;
    vi co(2*v); co[0] = v-1;
    rep(i,0,v) cur[i] = g[i].data();
    for (Edge& e : g[s]) addFlow(e, e.c);

    for (int hi = 0;;) {
      while (hs[hi].empty()) if (!hi--) return -ec[s];
      int u = hs[hi].back(); hs[hi].pop_back();
      while (ec[u] > 0)  // discharge u
        if (cur[u] == g[u].data() + sz(g[u])) {
          H[u] = 1e9;
          for (Edge& e : g[u]) if (e.c && H[u] > H[e.dest]+1)
            H[u] = H[e.dest]+1, cur[u] = &e;
          if (++co[H[u]], !--co[hi] && hi < v)
            rep(i,0,v) if (hi < H[i] && H[i] < v)
              --co[H[i]], H[i] = v + 1;
          hi = H[u];
        } else if (cur[u]->c && H[u] == H[cur[u]->dest]+1)
          addFlow(*cur[u], min(ec[u], cur[u]->c));
        else ++cur[u];
    }
  }
  bool leftOfMinCut(int a) { return H[a] >= sz(g); }
};
```

3.4 Gomory Hu

```
const int INF = 1000000000;

struct Edge {
    int a, b, cap, flow;
};

struct MaxFlow {
    int n, s, t;
    vector<int> d, ptr, q;
    vector< Edge > e;
    vector< vector<int> > g;
    MaxFlow() = default;
    MaxFlow(int _n) : n(_n), d(_n), ptr(_n), q(_n), g(_n) {
        e.clear();
        for (int i = 0; i < n; i++) {
            g[i].clear();
            ptr[i] = 0;
        }
    }
    void addEdge(int a, int b, int cap) {
        Edge e1 = { a, b, cap, 0 };
        Edge e2 = { b, a, 0, 0 };
        g[a].push_back( (int) e.size() );
        e.push_back(e1);
        g[b].push_back( (int) e.size() );
        e.push_back(e2);
    }
    int getMaxFlow(int _s, int _t) {
        s = _s; t = _t;
        int flow = 0;
        for (;;) {
            if (!bfs()) break;
            std::fill(ptr.begin(), ptr.end(), 0);
            while (int pushed = dfs(s, INF))
                flow += pushed;
        }
        return flow;
    }
private:
    bool bfs() {
        int qh = 0, qt = 0;
        q[qt++] = s;
        std::fill(d.begin(), d.end(), -1);
        d[s] = 0;
        while (qh < qt && d[t] == -1) {
            int v = q[qh++];
            for (int i = 0; i < (int) g[v].size(); i++) {
                int id = g[v][i], to = e[id].b;
                if (d[to] == -1 && e[id].flow < e[id].cap) {
                    q[qt++] = to;
                    d[to] = d[v] + 1;
                }
            }
        }
        return d[t] != -1;
    }
    int dfs (int v, int flow) {
        if (!flow) return 0;
        if (v == t) return flow;
        for (; ptr[v] < (int)g[v].size(); ++ptr[v]) {
            int id = g[v][ptr[v]],
                to = e[id].b;
            if (d[to] != d[v] + 1) continue;
            int pushed = dfs(to, min(flow, e[id].cap -
            e[id].flow));
            if (pushed) {
                e[id].flow += pushed;
                e[id^1].flow -= pushed;
                return pushed;
            }
        }
        return 0;
    }
};

const int N = 202;
int ok[N], cap[N][N];
int answer[N][N], parent[N];
int n;
MaxFlow flow;

void Init(int vertices) {
    n = vertices;
    flow = MaxFlow(vertices);
    for(int i = 0; i < vertices; ++i) ok[i] = parent[i] = 0;
    for(int i = 0; i < vertices; ++i)
        for(int j = 0; j < vertices; ++j)
            cap[i][j] = 0, answer[i][j] = INF;
}

void bfs(int start) {
    memset(ok,0,sizeof ok);
    queue<int> qu;
    qu.push(start);
    while (!qu.empty()) {
        int u=qu.front(); qu.pop();
        for(int xid = 0; xid < (int) flow.g[u].size(); ++xid)
        {
            int id = flow.g[u][xid];
            int v = flow.e[id].b, F = flow.e[id].flow, C =
            flow.e[id].cap;
            if (!ok[v] && F < C) {
                ok[v]=1;
                qu.push(v);
            }
        }
    }
}
void FindMaxFlow() {
    for(int i = 1; i <= n-1; ++i) {
        flow = MaxFlow(n);
        for(int u = 0; u < n; ++u)
            for(int v = 0; v < n; ++v)
            if (cap[u][v])
                flow.addEdge(u, v, cap[u][v]);

        int f = flow.getMaxFlow(i, parent[i]);

        bfs(i);
        for(int j = i+1; j < n; ++j)
            if (ok[j] && parent[j]==parent[i])
                parent[j]=i;

        answer[i][parent[i]] = answer[parent[i]][i] = f;
        for(int j = 0; j < i; ++j)

            answer[i][j]=answer[j][i]=min(f,answer[parent[i]][j]);
    }
}
```

3.5 Min Cost Max Flow

```
int n, m, k, source, sink;
struct FlowEdge {
    int to, rev, id, flow, cap, cost;
};
vector<FlowEdge> adj[MAX_N];
int dist[MAX_N];
bool inQueue[MAX_N];
pii trc[MAX_N];
queue<int> q;
int ans;

void addEdge(int u, int v, int cost, int cap) {
    int szU = adj[u].size();
    int szV = adj[v].size();
    adj[u].pb({v, szV, szU, 0, cap, cost});
    adj[v].pb({u, szU, szV, 0, cap, cost});
}

bool BellmanFord() {
    for (int i = 1; i <= n; i++) {
        dist[i] = inf;
    }
    dist[source] = 0;
    q.push(source);
    inQueue[source] = true;
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        inQueue[u] = false;
        for (auto e : adj[u]) {
            int v = e.to;
            int c = (e.flow >= 0 ? 1 : -1) * e.cost;
            if (e.flow < e.cap && dist[u] + c < dist[v]) {
                dist[v] = dist[u] + c;
                trc[v] = {u, e.id};
                if (!inQueue[v]) {
                    q.push(v);
                }
            }
        }
    }
    return dist[sink] < inf;
}

void inc() {
    int incFlow = inf;

    for (int i = sink; i != source; i = trc[i].fi) {
        int u = trc[i].fi;
        int id = trc[i].se;
        minimize(incFlow, (adj[u][id].flow >= 0 ?
        adj[u][id].cap - adj[u][id].flow : -adj[u][id].flow));
    }

    minimize(incFlow, k);


    for (int i = sink; i != source; i = trc[i].fi) {
        int u = trc[i].fi;
        int id = trc[i].se;
        adj[u][id].flow += incFlow;
        adj[i][adj[u][id].rev].flow -= incFlow;
    }

    ans += incFlow * dist[sink];
    k -= incFlow;

    if (!k) {
        cout << ans << '\n';
        for (int i = 1; i <= n; i++) {
            for (auto e : adj[i]) {
                if (e.flow > 0) {
                    cout << i << " " << e.to << " " << e.flow
                    << '\n';
                }
            }
        }
        cout << "0 0 0";
        exit(0);
    }
}

signed main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);

    cin >> n >> m >> k >> source >> sink;
    for (int i = 1; i <= m; i++) {
        int u, v, c, d;
        cin >> u >> v >> c >> d;
        addEdge(u, v, c, d);
    }
    while (BellmanFord()) {
        inc();
    }

    cout << -1;

    return 0;
}
```

3.6 Weighted Matching (Hungarian)

```
/**
 * Author: Benjamin Qi, chilli
 * Date: 2020-04-04
 * License: CC0
 * Description: Given a weighted bipartite graph, matches
every node on
 * the left with a node on the right such that no
 * nodes are in two matchings and the sum of the edge weights
is minimal. Takes
 * cost[N][M], where cost[i][j] = cost for L[i] to be matched
with R[j] and
 * returns (min cost, match), where L[i] is matched with
 * R[match[i]]. Negate costs for max cost. Requires N <= M.
 * Time: O(N^2M)
 * Status: Tested on kattis:cordonbleu, stress-tested
 */
#pragma once

pair<int, vi> hungarian(const vector<vi> &a) {
  if (a.empty()) return {0, {}};
  int n = sz(a) + 1, m = sz(a[0]) + 1;
  vi u(n), v(m), p(m), ans(n - 1);
  rep(i,1,n) {
    p[0] = i;
    int j0 = 0; // add "dummy" worker 0
    vi dist(m, INT_MAX), pre(m, -1);
    vector<bool> done(m + 1);
    do { // dijkstra
      done[j0] = true;
      int i0 = p[j0], j1, delta = INT_MAX;
      rep(j,1,m) if (!done[j]) {
        auto cur = a[i0 - 1][j - 1] - u[i0] - v[j];
        if (cur < dist[j]) dist[j] = cur, pre[j] = j0;
        if (dist[j] < delta) delta = dist[j], j1 = j;
      }
      rep(j,0,m) {
        if (done[j]) u[p[j]] += delta, v[j] -= delta;
        else dist[j] -= delta;
      }
      j0 = j1;
    } while (p[j0]);
    while (j0) { // update alternating path
      int j1 = pre[j0];
      p[j0] = p[j1], j0 = j1;
    }
  }
  rep(j,1,m) if (p[j]) ans[p[j] - 1] = j - 1;
  return {-v[0], ans}; // min cost
}
```

3.7 Max Clique

```
template <size_t max_n>
class Clique
{
    using bits = bitset<max_n>;
    bits MASK, ZERO, ans;
    const bits *e;
    int N;
    // int64_t calls;
void bk_init()
    {
        ans = ZERO;
        MASK = ZERO;
        MASK.flip();
        // calls = 0;
    }
    void bk(bits use, bits can_start, bits can_other)
    {
        // ++calls;
        if (can_start.none() && can_other.none())
        {
            if (use.count() > ans.count())
                ans = use;
            return;
        }
        bits r = can_start;
        bool fi = 1;
        for (int i = 0; i < N; ++i)
        {
            if (r[i])
            {
                if (fi)
                {
                    fi = 0;
                    r &= e[i] ^ MASK;
                }
                use[i] = 1;
                bk(use, can_start & e[i], can_other & e[i]);
                use[i] = 0;
                can_start[i] = 0;
                can_other[i] = 1;
            }
        }
    }
    static Clique &get()
    {
        static Clique c;
        return c;
    }

public:
    static bits find_clique(bits const *g, const int &n)
    {
        Clique &c = get();
        c.e = g;
        c.N = n;
        c.bk_init();
        bits me;
        c.bk(me, c.MASK, c.ZERO);
        // cerr << "Calls: " << c.calls << "\n";
        // c.calls = 0;
        return c.ans;
    }
    static bits find_clique(vector<bits> const &g)
    {
        return find_clique(g.data(), g.size());
    }
    static bits find_clique(array<bits, max_n> const &g, const
    int &n)
    {
        return find_clique(g.data(), n);
    }
};
```

3.8 2 SAT

```
//source: https://wiki.vnoi.info/vi/algo/graph-theory/2-SAT

#include <bits/stdc++.h>

using namespace std;

const int maxN = 500500;

int n, m;

// Lu đ th G
vector<int> G[maxN << 1];

// Ly giá tr ph đnh ca x
int NOT(int x) {
    return x + (x <= n ? n : -n); // -x
}
```

```
// Thêm điu kin u OR v
void add_clause(int u, int v) {
    G[NOT(u)].push_back(v); // -u -> v
    G[NOT(v)].push_back(u); // -v -> u
}

// Tìm thành phn liên thông mnh
int id[maxN << 1];
int num[maxN << 1], low[maxN << 1];
int timeDFS = 0, scc = 0;
int st[maxN << 1];

void dfs(int u) {
    num[u] = low[u] = ++timeDFS;
    st[++st[0]] = u;
    for(const int& v : G[u]) {
        if(id[v] != 0) continue;
        if(num[v] == 0) {
            dfs(v);
            low[u] = min(low[u], low[v]);
        } else low[u] = min(low[u], num[v]);
    }

    if(num[u] == low[u]) {
        for(++scc; true; ) {
            int v = st[st[0]--];
            id[v] = scc;
            if(v == u) break;
        }
    }
}

int main() {
    cin.tie(0)->sync_with_stdio(0);

    cin >> n >> m;
    for(int i = 1; i <= m; ++i) {
        int u, v; cin >> u >> v;
        add_clause(u, v);
    }

    // Thut toán Tarjan
    for(int i = 1; i <= 2 * n; ++i) {
        if(!id[i]) dfs(i);
    }

    bool answer = 1;
    for(int i = 1; i <= n; ++i) {
        // Kim tra điu kin tn ti phng án
        if(id[i] == id[NOT(i)]) answer = 0;
    }
    if(!answer) {
        cout << "IMPOSSIBLE"; // Thông báo bài toán vô nghim
        return 0;
    }
    // In tp giá tr a1, a2, ..., an
    for(int i = 1; i <= n; ++i) cout << (id[i] < id[NOT(i)])
    << " ";
    return 0;
}
```

4 DP

4.1 Divide And Conquer Optimization

```
void compute(int l, int r, int optl, int optr) {
    if (l > r)
        return;

    int mid = (l + r) >> 1;
    pair<long long, int> best = {LLONG_MAX, -1};

    for (int k = optl; k <= min(mid, optr); k++) {
        best = min(best, {(k ? dp_before[k - 1] : 0) + C(k,
        mid), k});
    }

    dp_cur[mid] = best.first;
    int opt = best.second;

    compute(l, mid - 1, optl, opt);
    compute(mid + 1, r, opt, optr);
}
```

4.2 Matrix Multiplication Optimization

```
namespace Matrix_Exponentiation {

    const int MAX_ROW = x; // Change Max_row here
    const int MAX_COL = x; // Change Max_col here
    int64_t mod = 1e9 + 7; // Change MOD here
    int64_t mxmod = (int64_t)(7e18 / mod) * mod;

    void change_mod(int _mod) {
        mod = _mod;
        mxmod = (int64_t)(7e18 / mod) * mod;
    }

    int64_t multi(int64_t a, int64_t b) {
        int64_t ret = 0;
        for(int i = 0 ; MASK(i) <= b ; i ++, a = (a + a) %
        mod) {
            if(MASK(i) & b) ret = (ret + a) % mod;
        }
        return ret;
    }

    struct Matrix {
        int r,c;
        int64_t a[MAX_ROW][MAX_COL];
        void Resize(int _r,int _c) {
            for (int i = 0; i < r; i ++) {
                for (int j = 0; j < c; j ++) {
                    a[i][j] = 0;
                }
            }
        }

        auto & operator [] (int i) { return a[i]; }

        const auto & operator[] (int i) const { return a[i]; }

        Matrix operator *(const Matrix& other) {
            Matrix product, tmp;
            product.Resize(r, other.c);
            tmp.Resize(r, other.c);
            for (int i = 0; i < product.r; i ++) {
                for (int j = 0; j < c; j ++) {
                    for (int k = 0; k < product.c; k ++) {
//                        product[i][k] += multi(a[i][j] ,
other.a[j][k]);
                        tmp[i][k] += a[i][j] * other[j][k];
                        if(tmp[i][k] >= mxmod)
                            tmp[i][k] -= mxmod;
                    }
                }
            }
            for (int i = 0; i < product.r; i ++) {
                for (int j = 0; j < product.c; j ++) {
                    product[i][j] = tmp[i][j] % mod;
                }
            }
            return product;
        }

        void operator *= (const Matrix& other) {
            *this = *this * other;
        }

        Matrix operator ^ (const int64_t& b) {
            Matrix ret;
            Matrix m = *this;
            ret.Resize(m.r, m.c);
            for (int i = 0; i < ret.r; i ++) {
                ret[i][i] = 1;
            }
            for(int i = 0 ; MASK(i) <= b ; i ++, m *= m) {
                if (b & MASK(i)) {
```

```
                ret*=m;
            }
        }
        return ret;
    }

    void operator ^= (const int64_t& b) {
        *this = *this ^ b;
    }

    friend ostream& operator << (ostream& os, const
    Matrix& M) {
        for (int i = 0; i < M.r; i ++) {
            for (int j = 0; j < M.c; j ++) {
                os << M.a[i][j] << " \n"[j == M.c - 1];
            }
        }
        return os;
    }
  };
}

using namespace Matrix_Exponentiation;
```

5 Strings

5.1 KMP

```
vector<int> prefix_function(string s) {
    int n = (int)s.length();
    vector<int> pi(n);
    for (int i = 1; i < n; i++) {
        int j = pi[i-1];
        while (j > 0 && s[i] != s[j])
            j = pi[j-1];
        if (s[i] == s[j])
            j++;
        pi[i] = j;
    }
    return pi;
}
```

5.2 Z Function

```
vector<int> zfunction(const string& s) {
    int n = (int) s.size();
    vector<int> z(n);
    for(int i = 1, l = 0, r = 0; i < n; ++i) {
         if(i <= r) z[i] = min(r - i, z[i - l]);
         while(i + z[i] < n && s[z[i]] == s[i + z[i]]) ++z[i];
         if(i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
    } return z;
}
```

5.3 Aho Corasick (static)

```
const int CAP = 1003, ALPHABET = 26;
int cntTrie = 1;
int fail[CAP], to[CAP][ALPHABET];
bool ending[CAP];

void add_string(const string& s) {
    int u = 1;
    for(const char& c : s) {
        int x = c - 'a';
        if(!to[u][x]) {
            to[u][x] = ++cntTrie;
        }
        u = to[u][x];
    }
    ending[u] = true;
}

void aho_corasick() {
    queue<int> q; q.push(1);
    while(q.size()) {
        int u = q.front(); q.pop();
        for(int x = 0; x < ALPHABET; ++x) {
            int& v = to[u][x];
            if(!v) {
                v = u == 1 ? 1 : to[fail[u]][x];
            } else {
                if(!fail[v]) fail[v] = fail[u];
                fail[v] = u == 1 ? 1 : to[fail[v]][x];
                ending[v] |= ending[fail[v]];
                q.push(v);
            }
        }
    }
}
```

5.4 Aho Corasick (vector)

```
struct TrieNode {
  int pi = 0;
  int child[26] = {0};
};
vector<TrieNode> trie;
vector<vector<int>> adj;

int TrieInsert(const string& s) {
  int p = 0;
  for (int i = 0; i < s.size(); i++) {
    if (!trie[p].child[s[i] - 'a']) {
      trie[p].child[s[i] - 'a'] = trie.size();
      trie.pb(TrieNode());
    }
    p = trie[p].child[s[i] - 'a'];
  }
  return p;
}

void AhoCorasickBuild() {
  queue<int> q;
  for (int i = 0; i < 26; i++) {
    if (trie[0].child[i]) {
      q.push(trie[0].child[i]);
    }
  }
  while (!q.empty()) {
    int u = q.front();
    q.pop();
    for (int i = 0; i < 26; i++) {
      if (!trie[u].child[i]) continue;
      int j = trie[u].pi;
      while (!trie[j].child[i]) {
        if (!j) break;
        j = trie[j].pi;
      }
      trie[trie[u].child[i]].pi = trie[j].child[i];
      q.push(trie[u].child[i]);
    }
    adj[trie[u].pi].pb(u);
  }
}

signed main() {
  trie.pb(TrieNode());


  adj.resize(trie.size());
  AhoCorasickBuild();
}
```

6 Math

6.1 Chinese Remainder Theorem

```
struct gcd_t { ll x, y, d; };

gcd_t e_gcd(ll a, ll b) {
    if (b == 0) return {1, 0, a};

    gcd_t res = e_gcd(b, a % b);
    return {res.y, res.x - res.y * (a / b), res.d};
}

pll crt(vector<ll> r, vector<ll> m) {
    //find x such that for (1 <= i <= n): x = r[i] (mod m[i])
    //return {y, z} where x = y (mod z), z = lcm of vector m
    //all solutions are congruent modulo z

    ll y = r[0], z = m[0];
    for (int i = 1; i < sz(r); i++) {
        gcd_t cur = e_gcd(z, m[i]);

        ll x = cur.x, d = cur.d;

        if((r[i] - y) % d != 0) return {-1, -1};

        //ka = kb (mod kc)   =>   a = b (mod c) if (gcd(k, c)
        = 1)
        //add (x * (r[i] - y) / d * z) to result (with moduli
        lcm(z, m[i]))
        ll tmp = (x * (r[i] - y) / d) % (m[i] / d);
        y = y + tmp * z;
        z = z / d * m[i];

        y %= z;
        if (y < 0) y += z;
    }

    return {y, z};
}

ll inverse(ll a, ll m) {
    gcd_t cur = e_gcd(a, m);

    return (cur.x % m + m) % m;
}
```

6.2 Miller Rabin

```
// From
https://github.com/SnapDragon64/ContestLibrary/blob/master/math.h
// which also has specialized versions for 32-bit and 42-bit
//
// Tested:
// - https://oj.vnoi.info/problem/icpc22_national_c (fastest
solution)
// - https://www.spoj.com/problems/PON/

// Rabin miller {{{
inline uint64_t mod_mult64(uint64_t a, uint64_t b, uint64_t m)
{
    return __int128_t(a) * b % m;
}
uint64_t mod_pow64(uint64_t a, uint64_t b, uint64_t m) {
    uint64_t ret = (m > 1);
    for (;;) {
        if (b & 1) ret = mod_mult64(ret, a, m);
        if (!(b >>= 1)) return ret;
        a = mod_mult64(a, a, m);
    }
}

// Works for all primes p < 2^64
bool is_prime(uint64_t n) {
    if (n <= 3) return (n >= 2);
    static const uint64_t small[] = {
        2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43,
        47, 53, 59, 61, 67,
        71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127,
        131, 137, 139,
        149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197,
        199,
    };
    for (size_t i = 0; i < sizeof(small) / sizeof(uint64_t);
    ++i) {
        if (n % small[i] == 0) return n == small[i];
    }
    // Makes use of the known bounds for Miller-Rabin
    pseudoprimes.
    static const uint64_t millerrabin[] = {
        2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37,
    };
    static const uint64_t A014233[] = {  // From OEIS.
        2047LL, 1373653LL, 25326001LL, 3215031751LL,
        2152302898747LL,
        3474749660383LL, 341550071728321LL, 341550071728321LL,
        3825123056546413051LL, 3825123056546413051LL,
        3825123056546413051LL, 0,
    };
    uint64_t s = n-1, r = 0;
    while (s % 2 == 0) {
        s /= 2;
        r++;
    }
    for (size_t i = 0, j; i < sizeof(millerrabin) /
    sizeof(uint64_t); i++) {
        uint64_t md = mod_pow64(millerrabin[i], s, n);
        if (md != 1) {
            for (j = 1; j < r; j++) {
                if (md == n-1) break;
                md = mod_mult64(md, md, n);
            }
            if (md != n-1) return false;
        }
        if (n < A014233[i]) return true;
    }
    return true;
}
// }}}
```

6.3 Discrete Logarithm

```
// Computes x which a ^ x = b mod n.

long long d_log(long long a, long long b, long long n) {
    long long m = ceil(sqrt(n));
    long long aj = 1;
    map<long long, long long> M;
    for (int i = 0; i < m; ++i) {
        if (!M.count(aj))
            M[aj] = i;
        aj = (aj * a) % n;
    }

    long long coef = mod_pow(a, n - 2, n);
    coef = mod_pow(coef, m, n);
    // coef =  a ^ (-m)
    long long gamma = b;
    for (int i = 0; i < m; ++i) {
        if (M.count(gamma)) {
            return i * m + M[gamma];
        } else {
            gamma = (gamma * coef) % n;
        }
    }
    return -1;
}
```

6.4 Fast Fourier Transform

```
typedef complex<double> cmplx;
typedef vector<complex<double> > VC;
const double PI = acos(-1);
struct FFT {
    static void fft(VC &u, int sign) {
        int n = u.size();
        double theta = 2. * PI * sign / n;
        for (int m = n; m >= 2; m >>= 1, theta *= 2.) {
            cmplx w(1, 0), wDelta = polar(1., theta);
            for (int i = 0, mh = m >> 1; i < mh; i++) {
                for (int j = i; j < n; j += m) {
                    int k = j + mh;
                    cmplx temp = u[j] - u[k];
                    u[j] += u[k];
                    u[k] = w * temp;
                }
                w *= wDelta;
            }
        }
        for (int i = 1, j = 0; i < n; i++) {
            for (int k = n >> 1; k > (j ^= k); k >>= 1);
            if (j < i) {
                swap(u[i], u[j]);
            }
        }
    }

    static vector<ll> mul(const vector<int> &a, const
    vector<int> &b) {
        int newSz = a.size() + b.size() - 1;
        int fftSz = 1;
        while (fftSz < newSz) fftSz <<= 1;
        VC aa(fftSz, 0.), bb(fftSz, 0.);
        for (int i = 0; i < a.size(); i++) aa[i] = a[i];
        for (int i = 0; i < b.size(); i++) bb[i] = b[i];
        fft(aa, 1), fft(bb, 1);
        for (int i = 0; i < fftSz; i++) aa[i] *= bb[i];
        fft(aa, -1);
        vector<ll> res(newSz);
        for (int i = 0; i < newSz; i++)
            res[i] = (ll)(aa[i].real() / fftSz + 0.5);
        return res;
    }
};
```

6.5 Berlekamp massey

```
template<typename T> vector<T> berlekampMassey(const vector<T>
&sequence) {
    int n = (int)sequence.size(), len = 0, m = 1;
    vector<T> prevBest(n), coefficients(n);
    T prevDelta = prevBest[0] = coefficients[0] = 1;
    for (int i = 0; i < n; i++, m++) {
        T delta = sequence[i];
        for (int j = 1; j <= len; j++) delta +=
        coefficients[j] * sequence[i - j];
        if ((long long)delta == 0) continue;
        vector<T> temp = coefficients;
        T coef = delta / prevDelta;
        for (int j = m; j < n; j++) coefficients[j] -= coef *
        prevBest[j - m];
        if ((len << 1) <= i)
            len = i + 1 - len, prevBest = temp, prevDelta =
            delta, m = 0;
    }
    coefficients.resize(len + 1);
    coefficients.erase(coefficients.begin());
    for (T &x : coefficients) x = -x;
    return coefficients;
}

template<typename T> T calcKthTerm(
    const vector<T> &coefficients, const vector<T> &sequence,
    long long k
) {
    assert(coefficients.size() <= sequence.size());
    int n = (int)coefficients.size();

    auto mul = [&](const vector<T> &a, const vector<T> &b) {
        vector<T> res(a.size() + b.size() - 1u);
        for (int i = 0; i < (int)a.size(); i++)
            for (int j = 0; j < (int)b.size(); j++)
                res[i + j] += a[i] * b[j];
        for (int i = (int)res.size() - 1; i >= n; i--)
            for (int j = n - 1; j >= 0; j--)
                res[i - j - 1] += res[i] * coefficients[j];
        res.resize(min((int)res.size(), n));
        return res;
    };

    vector<T> a = (n == 1 ? vector<T>{coefficients[0]} :
    vector<T>{0, 1}), x{1};
    for (; k; k >>= 1) {
        if (k & 1) x = mul(x, a);
        a = mul(a, a);
    }
    x.resize(n);
    T res = 0;
    for (int i = 0; i < n; i++) res += x[i] * sequence[i];
    return res;
}

// Usual: cout << calcKthTerm(berlekampMassey(ans), ans, n) <<
"\n";
```

7 Geometry

7.1 Geomtry Point

```
struct Point{
    typedef ll T;
    T x, y;
    Point(T _x = 0, T _y = 0) : x(_x), y(_y) {}

    bool operator < (Point p) const { return tie(x, y) <
    tie(p.x, p.y); }
    bool operator > (Point p) const { return tie(x, y) >
    tie(p.x, p.y); }
    bool operator == (Point p) const { return tie(x, y) ==
    tie(p.x, p.y); }
    bool operator != (Point p) const { return tie(x, y) !=
    tie(p.x, p.y); }

    Point operator + (Point p) const { return Point(x + p.x, y
    + p.y); }
    Point operator - (Point p) const { return Point(x - p.x, y
    - p.y); }
    T operator * (Point p) const { return x * p.x + y * p.y; }
    T operator ^ (Point p) const { return x * p.y - y * p.x; }

    Point operator * (T d) const { return Point(x * d, y * d);
    }
    Point operator / (T d) const { return Point(x / d, y / d);
    }

    T len2() const { return x * x + y * y; }
    double len() const { return sqrt((double)len2()); }
    Point perp() { return Point(-y, x); }
    friend ostream& operator << (ostream &os, const Point &p)
    {
        return os << "(" << p.x << ", " << p.y << ")"; }
};

ll ccw(const Point &P0, const Point &P1, const Point &P2){
    return (P1 - P0) ^ (P2 - P1);
}

ll sgn(const ll &x) { return (x >= 0 ? (x ? 1 : 0) : -1); }
```

7.2 Convex Hull

```
vector<Point> convexHull(vector<Point> dots) {
    sort(dots.begin(), dots.end());
    vector<Point> A(1, dots[0]);
    const int sz = dots.size();
    for(int c = 0; c < 2; reverse(all(dots)), c++)
    for(int i = 1, t = A.size(); i < sz;
    A.emplace_back(dots[i++]))
      while (A.size() > t and ccw(A[A.size()-2], A.back(),
      dots[i]) < 0)
        A.pop_back();
    A.pop_back(); return A;
}
```

7.3 Manhattan MST

```
vector<array<ll, 3>> manhattanMST(vector<Point> ps) {
    vector<int> id(sz(ps));
    iota(all(id), 0);
    vector<array<ll, 3>> edges;
    for (int k = 0; k < 4; k++) {
            sort(all(id), [&](int i, int j) { return (ps[i] -
            ps[j]).x < (ps[j] - ps[i]).y; });
            map<int, int> sweep;
            for (int i : id) {
                for (auto it = sweep.lower_bound(-ps[i].y);
                    it != sweep.end(); sweep.erase(it++)) {
                    int j = it->second;
                    Point d = ps[i] - ps[j];
                    if (d.y > d.x) break;
                    edges.push_back({d.y + d.x, i, j});
                }
                sweep[-ps[i].y] = i;
            }
            for (Point &p : ps)
                if (k & 1)
                    p.x = -p.x;
                else
                    swap(p.x, p.y);
    }
    return edges;
}
```

7.4 Some Common Geometry Operations

```
ll ccw(const Point &P0, const Point &P1, const Point &P2){
    return (P1 - P0) ^ (P2 - P1);
}

bool on_segment(Point &p, Point &p0, Point &p1){
    if((p1 - p0) * (p - p1) > 0) return false;
    if((p0 - p1) * (p - p0) > 0) return false;
    return (ccw(p, p0, p1) == 0);
}

db dist_segment(Point &p, Point &p0, Point &p1){
    if((p1 - p0) * (p - p1) >= 0) return (p - p1).len();
    if((p0 - p1) * (p - p0) >= 0) return (p - p0).len();
    return abs((db)((p1 - p0) ^ (p - p0)) / (p1 - p0).len());
}

bool insideConvex(Point p, vector<Point> &poly){
    // clock wise
    int n = sz(poly);
    if(ccw(poly[0], poly[1], p) >= 0) return false;
    if(ccw(poly[n - 1], poly[0], p) >= 0) return false;

    ll l = 1, r = n-1;
    while(l < r){
        ll mid = (l+r+1)/2;
        if(ccw(poly[0], p, poly[mid]) >= 0) l = mid;
        else r = mid - 1;
    }
    r = l + 1;

    return (ccw(poly[l], p, poly[r]) > 0);
}

ll wn_poly(Point p, vector<Point> &poly){
    // 1 if inside 0 if outside, INF if on boundary
    // counter clock wise
    const ll on_boundary = INF;
    ll wn = 0;

    int n = sz(poly);
    for(int i = 0; i < n; i++){
        if(p == poly[i]) return on_boundary;

        int j = (i + 1 != n ? i + 1 : 0);

        if(poly[i].y == p.y && poly[j].y == p.y){
            if(min(poly[i].x, poly[j].x) <= p.x
                 && p.x <= max(poly[i].x, poly[j].x))
                return on_boundary;
        }
        else{
            bool below = (poly[i].y <= p.y);
            //different sides of horizontal ray
            if (below != (poly[j].y <= p.y)){
                ll orientation = ccw(p, poly[i], poly[j]);
                if (orientation == 0) return on_boundary;
                if (below == (orientation > 0)) wn += (below ?
                1 : -1);
            }
        }
    }

    return wn;
}

bool line_intersect(pii a, pii b, pii c, pii d) {
  if (!ccw(c, a, b) || !ccw(d, a, b) || !ccw(a, c, d) ||
  !ccw(b, c, d)) {
    if (!ccw(c, a, b) && dot_product(a, c, b) <= 0) {
      return true;
    }
    if (!ccw(d, a, b) && dot_product(a, d, b) <= 0) {
      return true;
    }
    if (!ccw(a, c, d) && dot_product(c, a, d) <= 0) {
      return true;
    }
    if (!ccw(b, c, d) && dot_product(c, b, d) <= 0) {
      return true;
    }
    return false;
  }
  return (ccw(a, b, c) * ccw(a, b, d) < 0 && ccw(c, d, a) *
  ccw(c, d, b) < 0);
```

8 Miscellaneous

8.1 Hilber Order for Mo's

```
inline ll gilbertOrder(int x, int y, int pow, int rotate) {
    if (pow == 0) {
        return 0;
    }
    int hpow = 1 << (pow - 1);
    int seg = (x < hpow) ? ((y < hpow) ? 0 : 3) : ((y < hpow)
    ? 1 : 2);
    seg = (seg + rotate) & 3;
    const int rotateDelta[4] = {3, 0, 0, 1};
    int nx = x & (x ^ hpow), ny = y & (y ^ hpow);
    int nrot = (rotate + rotateDelta[seg]) & 3;
    ll subSquareSize = ll(1) << (2 * pow - 2);
    ll ans = seg * subSquareSize;
    ll add = gilbertOrder(nx, ny, pow - 1, nrot);
    ans += (seg == 1 || seg == 2) ? add : (subSquareSize - add
    - 1);
    return ans;
}

struct Query {
    ll l, r, id, ord;
    Query(int _l, int _r, int _id) : l(_l), r(_r), id(_id) {}
    inline void calc() {
        ord = gilbertOrder(l, r, 21, 0);
    }
};

inline bool operator<(const Query &a, const Query &b)
{
    return a.ord < b.ord;
}
```

OEIS link	Name	First elements	Short description
A000010	Euler's totient function φ(n)	1, 1, 2, 2, 4, 2, 6, 4, 6, 4	φ(n) is the number of the positive integers not greater than n that are prime to n
A000027	Natural number	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	The natural numbers
A000032	Lucas number	2, 1, 3, 4, 7, 11, 18, 29, 47, 76	$L(n) = L(n - 1) + L(n - 2)$
A000040	Prime number	2, 3, 5, 7, 11, 13, 17, 19, 23, 29	The prime numbers
A000045	Fibonacci number	0, 1, 1, 2, 3, 5, 8, 13, 21, 34	$F(n) = F(n - 1) + F(n - 2)$ with $F(0) = 0$ and $F(1) = 1$
A000058	Sylvester's sequence	2, 3, 7, 43, 1807, 3263443, 10650056950807, 113423713055421844361000443	$a(n + 1) = a(n)^2 - a(n) + 1$, with $a(0) = 2$
A000073	Tribonacci number	0, 1, 1, 2, 4, 7, 13, 24, 44, 81	$T(n) = T(n - 1) + T(n - 2) + T(n - 3)$ with $T(0) = 0$, $T(1) = T(2) = 1$
A000108	Catalan number	1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862	$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!\,n!} = \prod_{k=2}^{n} \frac{n+k}{k}$ for $n \geq 0$.
A000110	Bell number	1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147	The number of partitions of a set with n elements
A000111	Euler number	1, 1, 1, 2, 5, 16, 61, 272, 1385, 7936	The number of linear extensions of the "zig-zag" poset
A000124	Lazy caterer's sequence	1, 2, 4, 7, 11, 16, 22, 29, 37, 46	The maximal number of pieces formed when slicing a pancake with n cuts
A000129	Pell number	0, 1, 2, 5, 12, 29, 70, 169, 408, 985	$a(0) = 0$, $a(1) = 1$; for $n > 1$, $a(n) = 2a(n - 1) + a(n - 2)$
A000142	Factorial	1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880	$n! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot \ldots \cdot n$
A000217	Triangular number	0, 1, 3, 6, 10, 15, 21, 28, 36, 45	$a(n) = C(n + 1, 2) = n(n + 1)/2 = 0 + 1 + 2 + \ldots + n$
A000292	Tetrahedral number	0, 1, 4, 10, 20, 35, 56, 84, 120, 165	The sum of the first n triangular numbers
A000330	Square pyramidal number	0, 1, 5, 14, 30, 55, 91, 140, 204, 285	$(n(n+1)(2n+1)) / 6$ The number of stacked spheres in a pyramid with a square base
A000396	Perfect number	6, 28, 496, 8128, 33550336, 8589869056, 137438691328, 2305843008139952128	n is equal to the sum of the proper divisors of n
A000668	Mersenne prime	3, 7, 31, 127, 8191, 131071, 524287, 2147483647, 2305843009213693951, 618970019642690137449562111	$2^p - 1$ if p is a prime
A007588	Stella octangula number	0, 1, 14, 51, 124, 245, 426, 679, 1016, 1449, 1990, 2651, 3444, 4381, ...	Stella octangula numbers: $n*(2*n^2 - 1)$.
A000793	Landau's function	1, 1, 2, 3, 4, 6, 6, 12, 15, 20	The largest order of permutation of n elements
A000796	Decimal expansion of Pi	3, 1, 4, 1, 5, 9, 2, 6, 5, 3	
A000931	Padovan sequence	1, 1, 1, 2, 2, 3, 4, 5, 7, 9	$P(0) = P(1) = P(2) = 1$, $P(n) = P(n-2)+P(n-3)$
A000945	Euclid–Mullin sequence	2, 3, 7, 43, 13, 53, 5, 6221671, 38709183810571, 139	$a(1) = 2$, $a(n+1)$ is smallest prime factor of $a(1)a(2)\ldots a(n)+1$.
A000959	Lucky number	1, 3, 7, 9, 13, 15, 21, 25, 31, 33	A natural number in a set that is filtered by a sieve

OEIS link	Name	First elements	Short description
A001006	Motzkin number	1, 1, 2, 4, 9, 21, 51, 127, 323, 835	The number of ways of drawing any number of nonintersecting chords joining *n* (labeled) points on a circle
A001045	Jacobsthal number	0, 1, 1, 3, 5, 11, 21, 43, 85, 171, 341	$a(n) = a(n-1) + 2a(n-2)$, with $a(0) = 0$, $a(1) = 1$
A001065	sequence of Aliquot sums s(n)	0, 1, 1, 3, 1, 6, 1, 7, 4, 8	s(n) is the sum of the proper divisors of the integer n
A001113	Decimal expansion of e (mathematical constant)	2, 7, 1, 8, 2, 8, 1, 8, 2, 8	
A001190	Wedderburn–Etherington number	0, 1, 1, 1, 2, 3, 6, 11, 23, 46	The number of binary rooted trees (every node has out-degree 0 or 2) with n endpoints (and $2n-1$ nodes in all)
A001358	Semiprime	4, 6, 9, 10, 14, 15, 21, 22, 25, 26	Products of two primes
A001462	Golomb sequence	1, 2, 2, 3, 3, 4, 4, 4, 5, 5	$a(n)$ is the number of times n occurs, starting with $a(1) = 1$
A001608	Perrin number	3, 0, 2, 3, 2, 5, 5, 7, 10, 12	$P(0) = 3$, $P(1) = 0$, $P(2) = 2$; $P(n) = P(n-2) + P(n-3)$ for $n > 2$
A001620	Euler–Mascheroni constant	5, 7, 7, 2, 1, 5, 6, 6, 4, 9	$\gamma = \lim_{n\to\infty}\left(\sum_{k=1}^{n}\frac{1}{k} - \ln(n)\right) = \lim_{b\to\infty}\int_1^b\left(\frac{1}{\lfloor x\rfloor} - \frac{1}{x}\right)\,dx.$
A001622	Decimal expansion of the golden ratio	1, 6, 1, 8, 0, 3, 3, 9, 8, 8	$\varphi = \frac{1+\sqrt{5}}{2} = 1.6180339887\ldots.$
A002064	Cullen number	1, 3, 9, 25, 65, 161, 385, 897, 2049, 4609, 10241, 22529, 49153, 106497	$n\,2^n + 1$
A002110	Primorial	1, 2, 6, 30, 210, 2310, 30030, 510510, 9699690, 223092870	The product of first *n* primes
A002113	Palindromic number	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	A number that remains the same when its digits are reversed
A002182	Highly composite number	1, 2, 4, 6, 12, 24, 36, 48, 60, 120	A positive integer with more divisors than any smaller positive integer
A002193	Decimal expansion of square root of 2	1, 4, 1, 4, 2, 1, 3, 5, 6, 2	
A002201	Superior highly composite number	2, 6, 12, 60, 120, 360, 2520, 5040, 55440, 720720	A positive integer n for which there is an $e>0$ such that $d(n)/n^e \geq d(k)/k^e$ for all $k>1$
A002378	Pronic number	0, 2, 6, 12, 20, 30, 42, 56, 72, 90	$n(n+1)$
A002808	Composite number	4, 6, 8, 9, 10, 12, 14, 15, 16, 18	The numbers n of the form xy for $x > 1$ and $y > 1$
A002858	Ulam number	1, 2, 3, 4, 6, 8, 11, 13, 16, 18	$a(1) = 1$; $a(2) = 2$; for $n>2$, $a(n)$ = least number > $a(n-1)$ which is a unique sum of two distinct earlier terms; semiperfect
A002997	Carmichael number	561, 1105, 1729, 2465, 2821, 6601, 8911, 10585, 15841, 29341	Composite numbers n such that $a^{(n-1)} == 1 \pmod{n}$ if a is prime to n
A003261	Woodall number	1, 7, 23, 63, 159, 383, 895, 2047, 4607	$n\,2^n - 1$

OEIS link	Name	First elements	Short description
A003459	Permutable prime	2, 3, 5, 7, 11, 13, 17, 31, 37, 71	The numbers for which every permutation of digits is a prime
A005044	Alcuin's sequence	0, 0, 0, 1, 0, 1, 1, 2, 1, 3, 2, 4, 3, 5, 4, 7, 5, 8, 7, 10, 8, 12, 10, 14	number of triangles with integer sides and perimeter *n*
A005100	Deficient number	1, 2, 3, 4, 5, 7, 8, 9, 10, 11	The numbers *n* such that $\sigma(n) < 2n$
A005101	Abundant number	12, 18, 20, 24, 30, 36, 40, 42, 48, 54	The sum of divisors of *n* exceeds 2*n*
A005150	Look-and-say sequence	1, 11, 21, 1211, 111221, 312211, 13112221, 1113213211, 31131211131221, 13211311123113112211,	A = 'frequency' followed by 'digit'-indication
A005224	Aronson's sequence	1, 4, 11, 16, 24, 29, 33, 35, 39, 45	"t" is the first, fourth, eleventh, ... letter in this sentence, not counting spaces or commas
A005235	Fortunate number	3, 5, 7, 13, 23, 17, 19, 23, 37, 61	The smallest integer m > 1 such that pn# + m is a prime number, where the primorial pn# is the product of the first n prime numbers
A005349	Harshad numbers in base 10	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12	a Harshad number in base 10 is an integer that is divisible by the sum of its digits (when written in base 10)
A005384	Sophie Germain prime	2, 3, 5, 11, 23, 29, 41, 53, 83, 89	A prime number *p* such that $2p+1$ is also prime
A005835	Semiperfect number	6, 12, 18, 20, 24, 28, 30, 36, 40, 42	A natural number *n* that is equal to the sum of all or some of its proper divisors
A006037	Weird number	70, 836, 4030, 5830, 7192, 7912, 9272, 10430, 10570, 10792	A natural number that is abundant but not semiperfect
A006842	Farey sequence numerators	0, 1, 0, 1, 1, 0, 1, 1, 2, 1	
A006843	Farey sequence denominators	1, 1, 1, 2, 1, 1, 3, 2, 3, 1	
A006862	Euclid number	2, 3, 7, 31, 211, 2311, 30031, 510511, 9699691, 223092871	1 + product of first *n* consecutive primes
A006886	Kaprekar number	1, 9, 45, 55, 99, 297, 703, 999, 2223, 2728	$X^2 = Ab^n + B$, where $0 < B < b^n$ $X = A + B$
A007304	Sphenic number	30, 42, 66, 70, 78, 102, 105, 110, 114, 130	Products of 3 distinct primes
A007318	Pascal's triangle	1, 1, 1, 1, 2, 1, 1, 3, 3, 1	Pascal's triangle read by rows
A007770	Happy number	1, 7, 10, 13, 19, 23, 28, 31, 32, 44	The numbers whose trajectory under iteration of sum of squares of digits map includes 1
A010060	Prouhet–Thue–Morse constant	0, 1, 1, 0, 1, 0, 0, 1, 1, 0	$\tau = \sum_{i=0}^{\infty} \frac{t_i}{2^{i+1}}$
A014080	Factorion	1, 2, 145, 40585	A natural number that equals the sum of the factorials of its decimal digits
A014577	Regular paperfolding sequence	1, 1, 0, 1, 1, 0, 0, 1, 1, 1	At each stage an alternating sequence of 1s and 0s is inserted between the terms of the previous sequence
A016114	Circular prime	2, 3, 5, 7, 11, 13, 17, 37, 79, 113	The numbers which remain prime under cyclic shifts of digits
A018226	Magic number (physics)	2, 8, 20, 28, 50, 82, 126	A number of nucleons (either protons or neutrons) such that they are arranged into complete shells within the atomic nucleus.

OEIS link	Name	First elements	Short description
A019279	Superperfect number	2, 4, 16, 64, 4096, 65536, 262144, 1073741824, 1152921504606846976, 309485009821345068724781056	$\sigma^2(n) = \sigma(\sigma(n)) = 2n\,,$
A027641	Bernoulli number	1, -1, 1, 0, -1, 0, 1, 0, -1, 0, 5, 0, -691, 0, 7, 0, -3617, 0, 43867, 0	
A031214	First elements in all OEIS sequences	1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,	One of sequences referring to the OEIS itself
A033307	Decimal expansion of Champernowne constant	1, 2, 3, 4, 5, 6, 7, 8, 9, 1	formed by concatenating the positive integers
A035513	Wythoff array	1, 2, 4, 3, 7, 6, 5, 11, 10, 9	A matrix of integers derived from the Fibonacci sequence
A036262	Gilbreath's conjecture	2, 1, 3, 1, 2, 5, 1, 0, 2, 7	Triangle of numbers arising from Gilbreath's conjecture
A037274	Home prime	1, 2, 3, 211, 5, 23, 7, 3331113965338635107, 311, 773	For $n \geq 2$, $a(n)$ = the prime that is finally reached when you start with n, concatenate its prime factors (A037276) and repeat until a prime is reached; $a(n) = -1$ if no prime is ever reached
A046075	Undulating number	101, 121, 131, 141, 151, 161, 171, 181, 191, 202	A number that has the digit form *ababab*
A050278	Pandigital number	1023456789, 1023456798, 1023456879, 1023456897, 1023456978, 1023456987, 1023457689, 1023457698, 1023457869, 1023457896	Numbers containing the digits 0-9 such that each digit appears exactly once
A052486	Achilles number	72, 108, 200, 288, 392, 432, 500, 648, 675, 800	Powerful but imperfect
A060006	Decimal expansion of Pisot–Vijayaraghavan number	1, 3, 2, 4, 7, 1, 7, 9, 5, 7	real root of x^3-x-1
A076336	Sierpinski number	78557, 271129, 271577, 322523, 327739, 482719, 575041, 603713, 903983, 934909	Odd k for which $\{\,k2^n+1 : n \in \mathbb{N}\,\}$ consists only of composite numbers
A076337	Riesel number	509203, 762701, 777149, 790841, 992077	Odd k for which $\{\,k2^n-1 : n \in \mathbb{N}\,\}$ consists only of composite numbers
A086747	Baum–Sweet sequence	1, 1, 0, 1, 1, 0, 0, 1, 0, 1	$a(n) = 1$ if binary representation of n contains no block of consecutive zeros of odd length; otherwise $a(n) = 0$
A094683	Juggler sequence	0, 1, 1, 5, 2, 11, 2, 18, 2, 27	If n mod 2 = 0 then floor($\sqrt{n}$) else floor($n^{3/2}$)
A097942	Highly totient number	1, 2, 4, 8, 12, 24, 48, 72, 144, 240	Each number k on this list has more solutions to the equation $\varphi(x) = k$ than any preceding k
A100264	Decimal expansion of Chaitin's constant	0, 0, 7, 8, 7, 4, 9, 9, 6, 9	
A104272	Ramanujan prime	2, 11, 17, 29, 41, 47, 59, 67	The nth Ramanujan prime is the least integer R_n for which $\pi(x) - \pi(x/2) \geq n$, for all $x \geq R_n$.
A122045	Euler number	1, 0, −1, 0, 5, 0, −61, 0, 1385, 0	$\frac{1}{\cosh t} = \frac{2}{e^t + e^{-t}} = \sum_{n=0}^{\infty} \frac{E_n}{n!} \cdot t^n$

Theoretical Computer Science Cheat Sheet

Definitions

$f(n) = O(g(n))$	iff $\exists$ positive c, n_0 such that $0 \le f(n) \le cg(n)$ $\forall n \ge n_0$.
$f(n) = \Omega(g(n))$	iff $\exists$ positive c, n_0 such that $f(n) \ge cg(n) \ge 0$ $\forall n \ge n_0$.
$f(n) = \Theta(g(n))$	iff $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.
$f(n) = o(g(n))$	iff $\lim_{n\to\infty} f(n)/g(n) = 0$.
$\lim\limits_{n\to\infty} a_n = a$	iff $\forall \epsilon > 0$, $\exists n_0$ such that $\lvert a_n - a\rvert < \epsilon$, $\forall n \ge n_0$.
$\sup S$	least $b \in \mathbb{R}$ such that $b \ge s$, $\forall s \in S$.
$\inf S$	greatest $b \in \mathbb{R}$ such that $b \le s$, $\forall s \in S$.
$\liminf\limits_{n\to\infty} a_n$	$\lim\limits_{n\to\infty} \inf\{a_i \mid i \ge n, i \in \mathbb{N}\}$.
$\limsup\limits_{n\to\infty} a_n$	$\lim\limits_{n\to\infty} \sup\{a_i \mid i \ge n, i \in \mathbb{N}\}$.
$\binom{n}{k}$	Combinations: Size k subsets of a size n set.
${n \brack k}$	Stirling numbers (1st kind): Arrangements of an n element set into k cycles.
${n \brace k}$	Stirling numbers (2nd kind): Partitions of an n element set into k non-empty sets.
$\left\langle {n \atop k} \right\rangle$	1st order Eulerian numbers: Permutations $\pi_1\pi_2\ldots\pi_n$ on $\{1, 2, \ldots, n\}$ with k ascents.
$\left\langle\!\left\langle {n \atop k} \right\rangle\!\right\rangle$	2nd order Eulerian numbers.
C_n	Catalan Numbers: Binary trees with $n+1$ vertices.

Series

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}, \quad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}.$$

In general:

$$\sum_{i=1}^{n} i^m = \frac{1}{m+1}\left[(n+1)^{m+1} - 1 - \sum_{i=1}^{n}\Big((i+1)^{m+1} - i^{m+1} - (m+1)i^m\Big)\right]$$

$$\sum_{i=1}^{n-1} i^m = \frac{1}{m+1}\sum_{k=0}^{m}\binom{m+1}{k}B_k n^{m+1-k}.$$

Geometric series:

$$\sum_{i=0}^{n} c^i = \frac{c^{n+1}-1}{c-1}, \quad c \ne 1, \quad \sum_{i=0}^{\infty} c^i = \frac{1}{1-c}, \quad \sum_{i=1}^{\infty} c^i = \frac{c}{1-c}, \quad |c| < 1,$$

$$\sum_{i=0}^{n} ic^i = \frac{nc^{n+2} - (n+1)c^{n+1} + c}{(c-1)^2}, \quad c \ne 1, \quad \sum_{i=0}^{\infty} ic^i = \frac{c}{(1-c)^2}, \quad |c| < 1.$$

Harmonic series:

$$H_n = \sum_{i=1}^{n}\frac{1}{i}, \quad \sum_{i=1}^{n} iH_i = \frac{n(n+1)}{2}H_n - \frac{n(n-1)}{4}.$$

$$\sum_{i=1}^{n} H_i = (n+1)H_n - n, \quad \sum_{i=1}^{n}\binom{i}{m}H_i = \binom{n+1}{m+1}\left(H_{n+1} - \frac{1}{m+1}\right).$$

1. $\binom{n}{k} = \frac{n!}{(n-k)!k!}$, **2.** $\sum_{k=0}^{n}\binom{n}{k} = 2^n$, **3.** $\binom{n}{k} = \binom{n}{n-k}$,

4. $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$, **5.** $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$,

6. $\binom{n}{m}\binom{m}{k} = \binom{n}{k}\binom{n-k}{m-k}$, **7.** $\sum_{k=0}^{n}\binom{r+k}{k} = \binom{r+n+1}{n}$,

8. $\sum_{k=0}^{n}\binom{k}{m} = \binom{n+1}{m+1}$, **9.** $\sum_{k=0}^{n}\binom{r}{k}\binom{s}{n-k} = \binom{r+s}{n}$,

10. $\binom{n}{k} = (-1)^k\binom{k-n-1}{k}$, **11.** ${n \brace 1} = {n \brace n} = 1$,

12. ${n \brace 2} = 2^{n-1} - 1$, **13.** ${n \brace k} = k{n-1 \brace k} + {n-1 \brace k-1}$,

14. ${n \brack 1} = (n-1)!$, **15.** ${n \brack 2} = (n-1)!H_{n-1}$, **16.** ${n \brack n} = 1$, **17.** ${n \brack k} \ge {n \brace k}$,

18. ${n \brack k} = (n-1){n-1 \brack k} + {n-1 \brack k-1}$, **19.** ${n \brace n-1} = {n \brack n-1} = \binom{n}{2}$, **20.** $\sum_{k=0}^{n}{n \brack k} = n!$, **21.** $C_n = \frac{1}{n+1}\binom{2n}{n}$,

22. $\left\langle {n \atop 0} \right\rangle = \left\langle {n \atop n-1} \right\rangle = 1$, **23.** $\left\langle {n \atop k} \right\rangle = \left\langle {n \atop n-1-k} \right\rangle$, **24.** $\left\langle {n \atop k} \right\rangle = (k+1)\left\langle {n-1 \atop k} \right\rangle + (n-k)\left\langle {n-1 \atop k-1} \right\rangle$,

25. $\left\langle {0 \atop k} \right\rangle = \begin{cases} 1 & \text{if } k = 0, \\ 0 & \text{otherwise} \end{cases}$ **26.** $\left\langle {n \atop 1} \right\rangle = 2^n - n - 1$, **27.** $\left\langle {n \atop 2} \right\rangle = 3^n - (n+1)2^n + \binom{n+1}{2}$,

28. $x^n = \sum_{k=0}^{n}\left\langle {n \atop k} \right\rangle\binom{x+k}{n}$, **29.** $\left\langle {n \atop m} \right\rangle = \sum_{k=0}^{m}\binom{n+1}{k}(m+1-k)^n(-1)^k$, **30.** $m!{n \brace m} = \sum_{k=0}^{n}\left\langle {n \atop k} \right\rangle\binom{k}{n-m}$,

31. $\left\langle {n \atop m} \right\rangle = \sum_{k=0}^{n}{n \brace k}\binom{n-k}{m}(-1)^{n-k-m}k!$, **32.** $\left\langle\!\left\langle {n \atop 0} \right\rangle\!\right\rangle = 1$, **33.** $\left\langle\!\left\langle {n \atop n} \right\rangle\!\right\rangle = 0$ for $n \ne 0$,

34. $\left\langle\!\left\langle {n \atop k} \right\rangle\!\right\rangle = (k+1)\left\langle\!\left\langle {n-1 \atop k} \right\rangle\!\right\rangle + (2n-1-k)\left\langle\!\left\langle {n-1 \atop k-1} \right\rangle\!\right\rangle$, **35.** $\sum_{k=0}^{n}\left\langle\!\left\langle {n \atop k} \right\rangle\!\right\rangle = \frac{(2n)^{\underline{n}}}{2^n}$,

36. ${x \brace x-n} = \sum_{k=0}^{n}\left\langle\!\left\langle {n \atop k} \right\rangle\!\right\rangle\binom{x+n-1-k}{2n}$, **37.** ${n+1 \brace m+1} = \sum_{k}\binom{n}{k}{k \brace m} = \sum_{k=0}^{n}{k \brace m}(m+1)^{n-k}$,

Theoretical Computer Science Cheat Sheet

Identities Cont.

38. $\left[{n+1 \atop m+1}\right] = \sum_k \left[{n \atop k}\right]\binom{k}{m} = \sum_{k=0}^{n} \left[{k \atop m}\right] n^{\underline{n-k}} = n!\sum_{k=0}^{n} \frac{1}{k!}\left[{k \atop m}\right],$

39. $\left[{x \atop x-n}\right] = \sum_{k=0}^{n} \left\langle\!\!\left\langle {n \atop k} \right\rangle\!\!\right\rangle \binom{x+k}{2n},$

40. $\left\{{n \atop m}\right\} = \sum_k \binom{n}{k}\left\{{k+1 \atop m+1}\right\}(-1)^{n-k},$

41. $\left[{n \atop m}\right] = \sum_k \left[{n+1 \atop k+1}\right]\binom{k}{m}(-1)^{m-k},$

42. $\left\{{m+n+1 \atop m}\right\} = \sum_{k=0}^{m} k\left\{{n+k \atop k}\right\},$

43. $\left[{m+n+1 \atop m}\right] = \sum_{k=0}^{m} k(n+k)\left[{n+k \atop k}\right],$

44. $\binom{n}{m} = \sum_k \left\{{n+1 \atop k+1}\right\}\left[{k \atop m}\right](-1)^{m-k},$

45. $(n-m)!\binom{n}{m} = \sum_k \left[{n+1 \atop k+1}\right]\left\{{k \atop m}\right\}(-1)^{m-k},$ for $n \ge m$,

46. $\left\{{n \atop n-m}\right\} = \sum_k \binom{m-n}{m+k}\binom{m+n}{n+k}\left[{m+k \atop k}\right],$

47. $\left[{n \atop n-m}\right] = \sum_k \binom{m-n}{m+k}\binom{m+n}{n+k}\left\{{m+k \atop k}\right\},$

48. $\left\{{n \atop \ell+m}\right\}\binom{\ell+m}{\ell} = \sum_k \left\{{k \atop \ell}\right\}\left\{{n-k \atop m}\right\}\binom{n}{k},$

49. $\left[{n \atop \ell+m}\right]\binom{\ell+m}{\ell} = \sum_k \left[{k \atop \ell}\right]\left[{n-k \atop m}\right]\binom{n}{k}.$

Trees

Every tree with n vertices has $n-1$ edges.

Kraft inequality: If the depths of the leaves of a binary tree are $d_1, \ldots, d_n$:
$$\sum_{i=1}^{n} 2^{-d_i} \le 1,$$
and equality holds only if every internal node has 2 sons.

Recurrences

Master method:
$$T(n) = aT(n/b) + f(n), \quad a \ge 1, b > 1$$

If $\exists\epsilon > 0$ such that $f(n) = O(n^{\log_b a - \epsilon})$ then
$$T(n) = \Theta(n^{\log_b a}).$$

If $f(n) = \Theta(n^{\log_b a})$ then
$$T(n) = \Theta(n^{\log_b a}\log_2 n).$$

If $\exists\epsilon > 0$ such that $f(n) = \Omega(n^{\log_b a + \epsilon})$, and $\exists c < 1$ such that $af(n/b) \le cf(n)$ for large n, then
$$T(n) = \Theta(f(n)).$$

Substitution (example): Consider the following recurrence
$$T_{i+1} = 2^{2^i} \cdot T_i^2, \quad T_1 = 2.$$

Note that T_i is always a power of two. Let $t_i = \log_2 T_i$. Then we have
$$t_{i+1} = 2^i + 2t_i, \quad t_1 = 1.$$

Let $u_i = t_i/2^i$. Dividing both sides of the previous equation by 2^{i+1} we get
$$\frac{t_{i+1}}{2^{i+1}} = \frac{2^i}{2^{i+1}} + \frac{t_i}{2^i}.$$

Substituting we find
$$u_{i+1} = \tfrac{1}{2} + u_i, \qquad u_1 = \tfrac{1}{2},$$

which is simply $u_i = i/2$. So we find that T_i has the closed form $T_i = 2^{i2^{i-1}}$.
Summing factors (example): Consider the following recurrence
$$T(n) = 3T(n/2) + n, \quad T(1) = 1.$$

Rewrite so that all terms involving T are on the left side
$$T(n) - 3T(n/2) = n.$$

Now expand the recurrence, and choose a factor which makes the left side "telescope"

$$\begin{aligned} 1\big(T(n) - 3T(n/2) &= n\big) \\ 3\big(T(n/2) - 3T(n/4) &= n/2\big) \\ \vdots \quad \vdots \quad &\vdots \\ 3^{\log_2 n - 1}\big(T(2) - 3T(1) &= 2\big) \end{aligned}$$

Let $m = \log_2 n$. Summing the left side we get $T(n) - 3^m T(1) = T(n) - 3^m = T(n) - n^k$ where $k = \log_2 3 \approx 1.58496$. Summing the right side we get
$$\sum_{i=0}^{m-1} \frac{n}{2^i} 3^i = n\sum_{i=0}^{m-1} \left(\tfrac{3}{2}\right)^i.$$

Let $c = \frac{3}{2}$. Then we have
$$\begin{aligned} n\sum_{i=0}^{m-1} c^i &= n\left(\frac{c^m - 1}{c - 1}\right) \\ &= 2n(c^{\log_2 n} - 1) \\ &= 2n(c^{(k-1)\log_c n} - 1) \\ &= 2n^k - 2n, \end{aligned}$$

and so $T(n) = 3n^k - 2n$. Full history recurrences can often be changed to limited history ones (example): Consider
$$T_i = 1 + \sum_{j=0}^{i-1} T_j, \quad T_0 = 1.$$

Note that
$$T_{i+1} = 1 + \sum_{j=0}^{i} T_j.$$

Subtracting we find
$$\begin{aligned} T_{i+1} - T_i &= 1 + \sum_{j=0}^{i} T_j - 1 - \sum_{j=0}^{i-1} T_j \\ &= T_i. \end{aligned}$$

And so $T_{i+1} = 2T_i = 2^{i+1}$.

Generating functions:

1. Multiply both sides of the equation by x^i.
2. Sum both sides over all i for which the equation is valid.
3. Choose a generating function $G(x)$. Usually $G(x) = \sum_{i=0}^{\infty} x^i g_i$.
3. Rewrite the equation in terms of the generating function $G(x)$.
4. Solve for $G(x)$.
5. The coefficient of x^i in $G(x)$ is g_i.

Example:
$$g_{i+1} = 2g_i + 1, \quad g_0 = 0.$$

Multiply and sum:
$$\sum_{i\ge 0} g_{i+1}x^i = \sum_{i\ge 0} 2g_i x^i + \sum_{i\ge 0} x^i.$$

We choose $G(x) = \sum_{i\ge 0} x^i g_i$. Rewrite in terms of $G(x)$:
$$\frac{G(x) - g_0}{x} = 2G(x) + \sum_{i\ge 0} x^i.$$

Simplify:
$$\frac{G(x)}{x} = 2G(x) + \frac{1}{1-x}.$$

Solve for $G(x)$:
$$G(x) = \frac{x}{(1-x)(1-2x)}.$$

Expand this using partial fractions:
$$\begin{aligned} G(x) &= x\left(\frac{2}{1-2x} - \frac{1}{1-x}\right) \\ &= x\left(2\sum_{i\ge 0} 2^i x^i - \sum_{i\ge 0} x^i\right) \\ &= \sum_{i\ge 0} (2^{i+1} - 1)x^{i+1}. \end{aligned}$$

So $g_i = 2^i - 1$.

Theoretical Computer Science Cheat Sheet

$\pi \approx 3.14159$, $e \approx 2.71828$, $\gamma \approx 0.57721$, $\phi = \frac{1+\sqrt{5}}{2} \approx 1.61803$, $\hat{\phi} = \frac{1-\sqrt{5}}{2} \approx -.61803$

i	2^i	p_i
1	2	2
2	4	3
3	8	5
4	16	7
5	32	11
6	64	13
7	128	17
8	256	19
9	512	23
10	1,024	29
11	2,048	31
12	4,096	37
13	8,192	41
14	16,384	43
15	32,768	47
16	65,536	53
17	131,072	59
18	262,144	61
19	524,288	67
20	1,048,576	71
21	2,097,152	73
22	4,194,304	79
23	8,388,608	83
24	16,777,216	89
25	33,554,432	97
26	67,108,864	101
27	134,217,728	103
28	268,435,456	107
29	536,870,912	109
30	1,073,741,824	113
31	2,147,483,648	127
32	4,294,967,296	131

Pascal's Triangle
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1

General

Bernoulli Numbers ($B_i = 0$, odd $i \neq 1$):

$$B_0 = 1,\ B_1 = -\tfrac{1}{2},\ B_2 = \tfrac{1}{6},\ B_4 = -\tfrac{1}{30},$$
$$B_6 = \tfrac{1}{42},\ B_8 = -\tfrac{1}{30},\ B_{10} = \tfrac{5}{66}.$$

Change of base, quadratic formula:

$$\log_b x = \frac{\log_a x}{\log_a b}, \qquad \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Euler's number e:

$$e = 1 + \tfrac{1}{2} + \tfrac{1}{6} + \tfrac{1}{24} + \tfrac{1}{120} + \cdots$$
$$\lim_{n\to\infty}\left(1 + \frac{x}{n}\right)^n = e^x.$$
$$\left(1 + \tfrac{1}{n}\right)^n < e < \left(1 + \tfrac{1}{n}\right)^{n+1}.$$
$$\left(1 + \tfrac{1}{n}\right)^n = e - \frac{e}{2n} + \frac{11e}{24n^2} - O\left(\frac{1}{n^3}\right).$$

Harmonic numbers:

$$1,\ \tfrac{3}{2},\ \tfrac{11}{6},\ \tfrac{25}{12},\ \tfrac{137}{60},\ \tfrac{49}{20},\ \tfrac{363}{140},\ \tfrac{761}{280},\ \tfrac{7129}{2520},\ \ldots$$

$$\ln n < H_n < \ln n + 1,$$
$$H_n = \ln n + \gamma + O\left(\frac{1}{n}\right).$$

Factorial, Stirling's approximation:

$$1,\ 2,\ 6,\ 24,\ 120,\ 720,\ 5040,\ 40320,\ 362880,\ \ldots$$

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1 + \Theta\left(\frac{1}{n}\right)\right).$$

Ackermann's function and inverse:

$$a(i,j) = \begin{cases} 2^j & i = 1 \\ a(i-1, 2) & j = 1 \\ a(i-1, a(i, j-1)) & i, j \geq 2 \end{cases}$$
$$\alpha(i) = \min\{j \mid a(j,j) \geq i\}.$$

Binomial distribution:

$$\Pr[X = k] = \binom{n}{k} p^k q^{n-k}, \qquad q = 1 - p,$$
$$\mathrm{E}[X] = \sum_{k=1}^{n} k \binom{n}{k} p^k q^{n-k} = np.$$

Poisson distribution:

$$\Pr[X = k] = \frac{e^{-\lambda}\lambda^k}{k!}, \quad \mathrm{E}[X] = \lambda.$$

Normal (Gaussian) distribution:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}, \quad \mathrm{E}[X] = \mu.$$

The "coupon collector": We are given a random coupon each day, and there are n different types of coupons. The distribution of coupons is uniform. The expected number of days to pass before we to collect all n types is

$$nH_n.$$

Probability

Continuous distributions: If

$$\Pr[a < X < b] = \int_a^b p(x)\,dx,$$

then p is the probability density function of X. If

$$\Pr[X < a] = P(a),$$

then P is the distribution function of X. If P and p both exist then

$$P(a) = \int_{-\infty}^{a} p(x)\,dx.$$

Expectation: If X is discrete

$$\mathrm{E}[g(X)] = \sum_x g(x)\Pr[X = x].$$

If X continuous then

$$\mathrm{E}[g(X)] = \int_{-\infty}^{\infty} g(x)p(x)\,dx = \int_{-\infty}^{\infty} g(x)\,dP(x).$$

Variance, standard deviation:

$$\mathrm{VAR}[X] = \mathrm{E}[X^2] - \mathrm{E}[X]^2,$$
$$\sigma = \sqrt{\mathrm{VAR}[X]}.$$

For events A and B:

$$\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$$
$$\Pr[A \wedge B] = \Pr[A] \cdot \Pr[B],$$

iff A and B are independent.

$$\Pr[A|B] = \frac{\Pr[A \wedge B]}{\Pr[B]}$$

For random variables X and Y:

$$\mathrm{E}[X \cdot Y] = \mathrm{E}[X] \cdot \mathrm{E}[Y],$$

if X and Y are independent.

$$\mathrm{E}[X + Y] = \mathrm{E}[X] + \mathrm{E}[Y],$$
$$\mathrm{E}[cX] = c\,\mathrm{E}[X].$$

Bayes' theorem:

$$\Pr[A_i|B] = \frac{\Pr[B|A_i]\Pr[A_i]}{\sum_{j=1}^{n}\Pr[A_j]\Pr[B|A_j]}.$$

Inclusion-exclusion:

$$\Pr\left[\bigvee_{i=1}^{n} X_i\right] = \sum_{i=1}^{n}\Pr[X_i] +$$
$$\sum_{k=2}^{n}(-1)^{k+1}\sum_{i_i < \cdots < i_k}\Pr\left[\bigwedge_{j=1}^{k} X_{i_j}\right].$$

Moment inequalities:

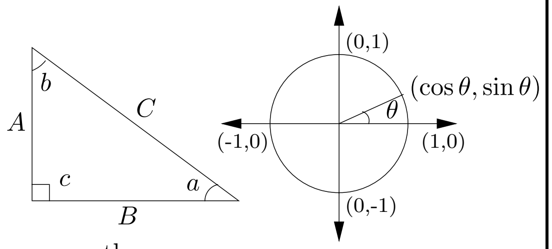
$$\Pr\left[|X| \geq \lambda\,\mathrm{E}[X]\right] \leq \frac{1}{\lambda},$$
$$\Pr\left[\left|X - \mathrm{E}[X]\right| \geq \lambda \cdot \sigma\right] \leq \frac{1}{\lambda^2}.$$

Geometric distribution:

$$\Pr[X = k] = pq^{k-1}, \qquad q = 1 - p,$$
$$\mathrm{E}[X] = \sum_{k=1}^{\infty} kpq^{k-1} = \frac{1}{p}.$$

Theoretical Computer Science Cheat Sheet

Trigonometry



Pythagorean theorem:

$$C^2 = A^2 + B^2.$$

Definitions:

$$\sin a = A/C, \quad \cos a = B/C,$$
$$\csc a = C/A, \quad \sec a = C/B,$$
$$\tan a = \frac{\sin a}{\cos a} = \frac{A}{B}, \quad \cot a = \frac{\cos a}{\sin a} = \frac{B}{A}.$$

Area, radius of inscribed circle:

$$\tfrac{1}{2}AB, \quad \frac{AB}{A+B+C}.$$

Identities:

$$\sin x = \frac{1}{\csc x}, \qquad \cos x = \frac{1}{\sec x},$$
$$\tan x = \frac{1}{\cot x}, \qquad \sin^2 x + \cos^2 x = 1,$$
$$1 + \tan^2 x = \sec^2 x, \qquad 1 + \cot^2 x = \csc^2 x,$$
$$\sin x = \cos\left(\tfrac{\pi}{2} - x\right), \qquad \sin x = \sin(\pi - x),$$
$$\cos x = -\cos(\pi - x), \qquad \tan x = \cot\left(\tfrac{\pi}{2} - x\right),$$
$$\cot x = -\cot(\pi - x), \qquad \csc x = \cot\tfrac{x}{2} - \cot x,$$
$$\sin(x \pm y) = \sin x \cos y \pm \cos x \sin y,$$
$$\cos(x \pm y) = \cos x \cos y \mp \sin x \sin y,$$
$$\tan(x \pm y) = \frac{\tan x \pm \tan y}{1 \mp \tan x \tan y},$$
$$\cot(x \pm y) = \frac{\cot x \cot y \mp 1}{\cot x \pm \cot y},$$
$$\sin 2x = 2 \sin x \cos x, \qquad \sin 2x = \frac{2\tan x}{1 + \tan^2 x},$$
$$\cos 2x = \cos^2 x - \sin^2 x, \qquad \cos 2x = 2\cos^2 x - 1,$$
$$\cos 2x = 1 - 2\sin^2 x, \qquad \cos 2x = \frac{1 - \tan^2 x}{1 + \tan^2 x},$$
$$\tan 2x = \frac{2\tan x}{1 - \tan^2 x}, \qquad \cot 2x = \frac{\cot^2 x - 1}{2\cot x},$$
$$\sin(x+y)\sin(x-y) = \sin^2 x - \sin^2 y,$$
$$\cos(x+y)\cos(x-y) = \cos^2 x - \sin^2 y.$$

Euler's equation:

$$e^{ix} = \cos x + i \sin x, \qquad e^{i\pi} = -1.$$

Matrices

Multiplication:

$$C = A \cdot B, \quad c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}.$$

Determinants: $\det A \neq 0$ iff A is non-singular.

$$\det A \cdot B = \det A \cdot \det B,$$
$$\det A = \sum_{\pi} \prod_{i=1}^{n} \operatorname{sign}(\pi) a_{i,\pi(i)}.$$

2×2 and 3×3 determinant:

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc,$$
$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = g \begin{vmatrix} b & c \\ e & f \end{vmatrix} - h \begin{vmatrix} a & c \\ d & f \end{vmatrix} + i \begin{vmatrix} a & b \\ d & e \end{vmatrix}$$
$$= \begin{matrix} aei + bfg + cdh \\ - ceg - fha - ibd. \end{matrix}$$

Permanents:

$$\operatorname{perm} A = \sum_{\pi} \prod_{i=1}^{n} a_{i,\pi(i)}.$$

Hyperbolic Functions

Definitions:

$$\sinh x = \frac{e^x - e^{-x}}{2}, \qquad \cosh x = \frac{e^x + e^{-x}}{2},$$
$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \qquad \operatorname{csch} x = \frac{1}{\sinh x},$$
$$\operatorname{sech} x = \frac{1}{\cosh x}, \qquad \coth x = \frac{1}{\tanh x}.$$

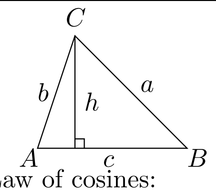
Identities:

$$\cosh^2 x - \sinh^2 x = 1, \qquad \tanh^2 x + \operatorname{sech}^2 x = 1,$$
$$\coth^2 x - \operatorname{csch}^2 x = 1, \qquad \sinh(-x) = -\sinh x,$$
$$\cosh(-x) = \cosh x, \qquad \tanh(-x) = -\tanh x,$$
$$\sinh(x+y) = \sinh x \cosh y + \cosh x \sinh y,$$
$$\cosh(x+y) = \cosh x \cosh y + \sinh x \sinh y,$$
$$\sinh 2x = 2 \sinh x \cosh x,$$
$$\cosh 2x = \cosh^2 x + \sinh^2 x,$$
$$\cosh x + \sinh x = e^x, \qquad \cosh x - \sinh x = e^{-x},$$
$$(\cosh x + \sinh x)^n = \cosh nx + \sinh nx, \quad n \in \mathbb{Z},$$
$$2\sinh^2 \tfrac{x}{2} = \cosh x - 1, \qquad 2\cosh^2 \tfrac{x}{2} = \cosh x + 1.$$

θ	$\sin\theta$	$\cos\theta$	$\tan\theta$
0	0	1	0
$\frac{\pi}{6}$	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{3}}{3}$
$\frac{\pi}{4}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$	1
$\frac{\pi}{3}$	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$	$\sqrt{3}$
$\frac{\pi}{2}$	1	0	∞

… in mathematics you don't understand things, you just get used to them.
– J. von Neumann

More Trig.



Law of cosines:

$$c^2 = a^2 + b^2 - 2ab\cos C.$$

Area:

$$A = \tfrac{1}{2}hc,$$
$$= \tfrac{1}{2}ab\sin C,$$
$$= \frac{c^2 \sin A \sin B}{2\sin C}.$$

Heron's formula:

$$A = \sqrt{s \cdot s_a \cdot s_b \cdot s_c},$$
$$s = \tfrac{1}{2}(a+b+c),$$
$$s_a = s - a,$$
$$s_b = s - b,$$
$$s_c = s - c.$$

More identities:

$$\sin\tfrac{x}{2} = \sqrt{\frac{1 - \cos x}{2}},$$
$$\cos\tfrac{x}{2} = \sqrt{\frac{1 + \cos x}{2}},$$
$$\tan\tfrac{x}{2} = \sqrt{\frac{1 - \cos x}{1 + \cos x}},$$
$$= \frac{1 - \cos x}{\sin x},$$
$$= \frac{\sin x}{1 + \cos x},$$
$$\cot\tfrac{x}{2} = \sqrt{\frac{1 + \cos x}{1 - \cos x}},$$
$$= \frac{1 + \cos x}{\sin x},$$
$$= \frac{\sin x}{1 - \cos x},$$
$$\sin x = \frac{e^{ix} - e^{-ix}}{2i},$$
$$\cos x = \frac{e^{ix} + e^{-ix}}{2},$$
$$\tan x = -i\frac{e^{ix} - e^{-ix}}{e^{ix} + e^{-ix}},$$
$$= -i\frac{e^{2ix} - 1}{e^{2ix} + 1},$$
$$\sin x = \frac{\sinh ix}{i},$$
$$\cos x = \cosh ix,$$
$$\tan x = \frac{\tanh ix}{i}.$$

Theoretical Computer Science Cheat Sheet

Number Theory

The Chinese remainder theorem: There exists a number C such that:

$$\begin{aligned} C &\equiv r_1 \bmod m_1 \\ \vdots \quad &\vdots \quad \vdots \\ C &\equiv r_n \bmod m_n \end{aligned}$$

if m_i and m_j are relatively prime for $i \neq j$.

Euler's function: $\phi(x)$ is the number of positive integers less than x relatively prime to x. If $\prod_{i=1}^n p_i^{e_i}$ is the prime factorization of x then

$$\phi(x) = \prod_{i=1}^n p_i^{e_i - 1}(p_i - 1).$$

Euler's theorem: If a and b are relatively prime then

$$1 \equiv a^{\phi(b)} \bmod b.$$

Fermat's theorem:

$$1 \equiv a^{p-1} \bmod p.$$

The Euclidean algorithm: if $a > b$ are integers then

$$\gcd(a, b) = \gcd(a \bmod b, b).$$

If $\prod_{i=1}^n p_i^{e_i}$ is the prime factorization of x then

$$S(x) = \sum_{d|x} d = \prod_{i=1}^n \frac{p_i^{e_i+1} - 1}{p_i - 1}.$$

Perfect Numbers: x is an even perfect number iff $x = 2^{n-1}(2^n - 1)$ and $2^n - 1$ is prime.

Wilson's theorem: n is a prime iff

$$(n-1)! \equiv -1 \bmod n.$$

Möbius inversion:

$$\mu(i) = \begin{cases} 1 & \text{if } i = 1. \\ 0 & \text{if } i \text{ is not square-free.} \\ (-1)^r & \text{if } i \text{ is the product of} \\ & r \text{ distinct primes.} \end{cases}$$

If

$$G(a) = \sum_{d|a} F(d),$$

then

$$F(a) = \sum_{d|a} \mu(d) G\Big(\frac{a}{d}\Big).$$

Prime numbers:

$$p_n = n \ln n + n \ln\ln n - n + n \frac{\ln\ln n}{\ln n} + O\Big(\frac{n}{\ln n}\Big),$$

$$\pi(n) = \frac{n}{\ln n} + \frac{n}{(\ln n)^2} + \frac{2! n}{(\ln n)^3} + O\Big(\frac{n}{(\ln n)^4}\Big).$$

Graph Theory

Definitions:

Loop	An edge connecting a vertex to itself.
Directed	Each edge has a direction.
Simple	Graph with no loops or multi-edges.
Walk	A sequence $v_0 e_1 v_1 \ldots e_\ell v_\ell$.
Trail	A walk with distinct edges.
Path	A trail with distinct vertices.
Connected	A graph where there exists a path between any two vertices.
Component	A maximal connected subgraph.
Tree	A connected acyclic graph.
Free tree	A tree with no root.
DAG	Directed acyclic graph.
Eulerian	Graph with a trail visiting each edge exactly once.
Hamiltonian	Graph with a cycle visiting each vertex exactly once.
Cut	A set of edges whose removal increases the number of components.
Cut-set	A minimal cut.
Cut edge	A size 1 cut.
k-Connected	A graph connected with the removal of any $k-1$ vertices.
k-Tough	$\forall S \subseteq V, S \neq \emptyset$ we have $k \cdot c(G - S) \leq \lvert S \rvert$.
k-Regular	A graph where all vertices have degree k.
k-Factor	A k-regular spanning subgraph.
Matching	A set of edges, no two of which are adjacent.
Clique	A set of vertices, all of which are adjacent.
Ind. set	A set of vertices, none of which are adjacent.
Vertex cover	A set of vertices which cover all edges.
Planar graph	A graph which can be embeded in the plane.
Plane graph	An embedding of a planar graph.

$$\sum_{v \in V} \deg(v) = 2m.$$

If G is planar then $n - m + f = 2$, so

$$f \leq 2n - 4, \quad m \leq 3n - 6.$$

Any planar graph has a vertex with degree ≤ 5.

Notation:

$E(G)$	Edge set
$V(G)$	Vertex set
$c(G)$	Number of components
$G[S]$	Induced subgraph
$\deg(v)$	Degree of v
$\Delta(G)$	Maximum degree
$\delta(G)$	Minimum degree
$\chi(G)$	Chromatic number
$\chi_E(G)$	Edge chromatic number
G^c	Complement graph
K_n	Complete graph
K_{n_1,n_2}	Complete bipartite graph
$\mathrm{r}(k,\ell)$	Ramsey number

Geometry

Projective coordinates: triples (x, y, z), not all x, y and z zero.

$$(x, y, z) = (cx, cy, cz) \quad \forall c \neq 0.$$

Cartesian	Projective
(x, y)	$(x, y, 1)$
$y = mx + b$	$(m, -1, b)$
$x = c$	$(1, 0, -c)$

Distance formula, L_p and L_∞ metric:

$$\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2},$$

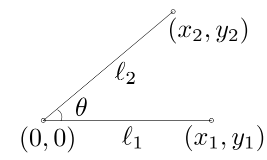
$$\big[\lvert x_1 - x_0\rvert^p + \lvert y_1 - y_0\rvert^p\big]^{1/p},$$

$$\lim_{p \to \infty} \big[\lvert x_1 - x_0\rvert^p + \lvert y_1 - y_0\rvert^p\big]^{1/p}.$$

Area of triangle (x_0, y_0), (x_1, y_1) and (x_2, y_2):

$$\tfrac{1}{2} \operatorname{abs} \begin{vmatrix} x_1 - x_0 & y_1 - y_0 \\ x_2 - x_0 & y_2 - y_0 \end{vmatrix}.$$

Angle formed by three points:



$$\cos\theta = \frac{(x_1, y_1) \cdot (x_2, y_2)}{\ell_1 \ell_2}.$$

Line through two points (x_0, y_0) and (x_1, y_1):

$$\begin{vmatrix} x & y & 1 \\ x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \end{vmatrix} = 0.$$

Area of circle, volume of sphere:

$$A = \pi r^2, \qquad V = \tfrac{4}{3}\pi r^3.$$

If I have seen farther than others, it is because I have stood on the shoulders of giants.
– Issac Newton

Theoretical Computer Science Cheat Sheet

π

Wallis' identity:
$$\pi = 2 \cdot \frac{2 \cdot 2 \cdot 4 \cdot 4 \cdot 6 \cdot 6 \cdots}{1 \cdot 3 \cdot 3 \cdot 5 \cdot 5 \cdot 7 \cdots}$$

Brouncker's continued fraction expansion:
$$\tfrac{\pi}{4} = 1 + \cfrac{1^2}{2 + \cfrac{3^2}{2 + \cfrac{5^2}{2 + \cfrac{7^2}{2 + \cdots}}}}$$

Gregrory's series:
$$\tfrac{\pi}{4} = 1 - \tfrac{1}{3} + \tfrac{1}{5} - \tfrac{1}{7} + \tfrac{1}{9} - \cdots$$

Newton's series:
$$\tfrac{\pi}{6} = \frac{1}{2} + \frac{1}{2 \cdot 3 \cdot 2^3} + \frac{1 \cdot 3}{2 \cdot 4 \cdot 5 \cdot 2^5} + \cdots$$

Sharp's series:
$$\tfrac{\pi}{6} = \frac{1}{\sqrt{3}}\left(1 - \frac{1}{3^1 \cdot 3} + \frac{1}{3^2 \cdot 5} - \frac{1}{3^3 \cdot 7} + \cdots\right)$$

Euler's series:
$$\tfrac{\pi^2}{6} = \tfrac{1}{1^2} + \tfrac{1}{2^2} + \tfrac{1}{3^2} + \tfrac{1}{4^2} + \tfrac{1}{5^2} + \cdots$$
$$\tfrac{\pi^2}{8} = \tfrac{1}{1^2} + \tfrac{1}{3^2} + \tfrac{1}{5^2} + \tfrac{1}{7^2} + \tfrac{1}{9^2} + \cdots$$
$$\tfrac{\pi^2}{12} = \tfrac{1}{1^2} - \tfrac{1}{2^2} + \tfrac{1}{3^2} - \tfrac{1}{4^2} + \tfrac{1}{5^2} - \cdots$$

Partial Fractions

Let $N(x)$ and $D(x)$ be polynomial functions of x. We can break down $N(x)/D(x)$ using partial fraction expansion. First, if the degree of N is greater than or equal to the degree of D, divide N by D, obtaining
$$\frac{N(x)}{D(x)} = Q(x) + \frac{N'(x)}{D(x)},$$
where the degree of N' is less than that of D. Second, factor $D(x)$. Use the following rules: For a non-repeated factor:
$$\frac{N(x)}{(x-a)D(x)} = \frac{A}{x-a} + \frac{N'(x)}{D(x)},$$
where
$$A = \left[\frac{N(x)}{D(x)}\right]_{x=a}.$$
For a repeated factor:
$$\frac{N(x)}{(x-a)^m D(x)} = \sum_{k=0}^{m-1} \frac{A_k}{(x-a)^{m-k}} + \frac{N'(x)}{D(x)},$$
where
$$A_k = \frac{1}{k!}\left[\frac{d^k}{dx^k}\left(\frac{N(x)}{D(x)}\right)\right]_{x=a}.$$

The reasonable man adapts himself to the world; the unreasonable persists in trying to adapt the world to himself. Therefore all progress depends on the unreasonable.
– George Bernard Shaw

Calculus

Derivatives:

1. $\dfrac{d(cu)}{dx} = c\dfrac{du}{dx}$, **2.** $\dfrac{d(u+v)}{dx} = \dfrac{du}{dx} + \dfrac{dv}{dx}$, **3.** $\dfrac{d(uv)}{dx} = u\dfrac{dv}{dx} + v\dfrac{du}{dx}$,

4. $\dfrac{d(u^n)}{dx} = nu^{n-1}\dfrac{du}{dx}$, **5.** $\dfrac{d(u/v)}{dx} = \dfrac{v\left(\frac{du}{dx}\right) - u\left(\frac{dv}{dx}\right)}{v^2}$, **6.** $\dfrac{d(e^{cu})}{dx} = ce^{cu}\dfrac{du}{dx}$,

7. $\dfrac{d(c^u)}{dx} = (\ln c)c^u\dfrac{du}{dx}$, **8.** $\dfrac{d(\ln u)}{dx} = \dfrac{1}{u}\dfrac{du}{dx}$,

9. $\dfrac{d(\sin u)}{dx} = \cos u\dfrac{du}{dx}$, **10.** $\dfrac{d(\cos u)}{dx} = -\sin u\dfrac{du}{dx}$,

11. $\dfrac{d(\tan u)}{dx} = \sec^2 u\dfrac{du}{dx}$, **12.** $\dfrac{d(\cot u)}{dx} = \csc^2 u\dfrac{du}{dx}$,

13. $\dfrac{d(\sec u)}{dx} = \tan u \sec u\dfrac{du}{dx}$, **14.** $\dfrac{d(\csc u)}{dx} = -\cot u \csc u\dfrac{du}{dx}$,

15. $\dfrac{d(\arcsin u)}{dx} = \dfrac{1}{\sqrt{1-u^2}}\dfrac{du}{dx}$, **16.** $\dfrac{d(\arccos u)}{dx} = \dfrac{-1}{\sqrt{1-u^2}}\dfrac{du}{dx}$,

17. $\dfrac{d(\arctan u)}{dx} = \dfrac{1}{1+u^2}\dfrac{du}{dx}$, **18.** $\dfrac{d(\operatorname{arccot} u)}{dx} = \dfrac{-1}{1+u^2}\dfrac{du}{dx}$,

19. $\dfrac{d(\operatorname{arcsec} u)}{dx} = \dfrac{1}{u\sqrt{1-u^2}}\dfrac{du}{dx}$, **20.** $\dfrac{d(\operatorname{arccsc} u)}{dx} = \dfrac{-1}{u\sqrt{1-u^2}}\dfrac{du}{dx}$,

21. $\dfrac{d(\sinh u)}{dx} = \cosh u\dfrac{du}{dx}$, **22.** $\dfrac{d(\cosh u)}{dx} = \sinh u\dfrac{du}{dx}$,

23. $\dfrac{d(\tanh u)}{dx} = \operatorname{sech}^2 u\dfrac{du}{dx}$, **24.** $\dfrac{d(\coth u)}{dx} = -\operatorname{csch}^2 u\dfrac{du}{dx}$,

25. $\dfrac{d(\operatorname{sech} u)}{dx} = -\operatorname{sech} u \tanh u\dfrac{du}{dx}$, **26.** $\dfrac{d(\operatorname{csch} u)}{dx} = -\operatorname{csch} u \coth u\dfrac{du}{dx}$,

27. $\dfrac{d(\operatorname{arcsinh} u)}{dx} = \dfrac{1}{\sqrt{1+u^2}}\dfrac{du}{dx}$, **28.** $\dfrac{d(\operatorname{arccosh} u)}{dx} = \dfrac{1}{\sqrt{u^2-1}}\dfrac{du}{dx}$,

29. $\dfrac{d(\operatorname{arctanh} u)}{dx} = \dfrac{1}{1-u^2}\dfrac{du}{dx}$, **30.** $\dfrac{d(\operatorname{arccoth} u)}{dx} = \dfrac{1}{u^2-1}\dfrac{du}{dx}$,

31. $\dfrac{d(\operatorname{arcsech} u)}{dx} = \dfrac{-1}{u\sqrt{1-u^2}}\dfrac{du}{dx}$, **32.** $\dfrac{d(\operatorname{arccsch} u)}{dx} = \dfrac{-1}{|u|\sqrt{1+u^2}}\dfrac{du}{dx}$.

Integrals:

1. $\displaystyle\int cu\,dx = c\int u\,dx$, **2.** $\displaystyle\int (u+v)\,dx = \int u\,dx + \int v\,dx$,

3. $\displaystyle\int x^n\,dx = \frac{1}{n+1}x^{n+1}, \quad n \neq -1$, **4.** $\displaystyle\int \frac{1}{x}dx = \ln x$, **5.** $\displaystyle\int e^x\,dx = e^x$,

6. $\displaystyle\int \frac{dx}{1+x^2} = \arctan x$, **7.** $\displaystyle\int u\frac{dv}{dx}dx = uv - \int v\frac{du}{dx}dx$,

8. $\displaystyle\int \sin x\,dx = -\cos x$, **9.** $\displaystyle\int \cos x\,dx = \sin x$,

10. $\displaystyle\int \tan x\,dx = -\ln|\cos x|$, **11.** $\displaystyle\int \cot x\,dx = \ln|\cos x|$,

12. $\displaystyle\int \sec x\,dx = \ln|\sec x + \tan x|$, **13.** $\displaystyle\int \csc x\,dx = \ln|\csc x + \cot x|$,

14. $\displaystyle\int \arcsin\tfrac{x}{a}dx = \arcsin\tfrac{x}{a} + \sqrt{a^2 - x^2}, \quad a > 0$,

Calculus Cont.

15. $\int \arccos\frac{x}{a}dx = \arccos\frac{x}{a} - \sqrt{a^2-x^2}, \quad a>0,$

16. $\int \arctan\frac{x}{a}dx = x\arctan\frac{x}{a} - \frac{a}{2}\ln(a^2+x^2), \quad a>0,$

17. $\int \sin^2(ax)dx = \frac{1}{2a}\big(ax - \sin(ax)\cos(ax)\big),$

18. $\int \cos^2(ax)dx = \frac{1}{2a}\big(ax + \sin(ax)\cos(ax)\big),$

19. $\int \sec^2 x\,dx = \tan x,$

20. $\int \csc^2 x\,dx = -\cot x,$

21. $\int \sin^n x\,dx = -\frac{\sin^{n-1}x\cos x}{n} + \frac{n-1}{n}\int \sin^{n-2}x\,dx,$

22. $\int \cos^n x\,dx = \frac{\cos^{n-1}x\sin x}{n} + \frac{n-1}{n}\int \cos^{n-2}x\,dx,$

23. $\int \tan^n x\,dx = \frac{\tan^{n-1}x}{n-1} - \int \tan^{n-2}x\,dx, \quad n\neq 1,$

24. $\int \cot^n x\,dx = -\frac{\cot^{n-1}x}{n-1} - \int \cot^{n-2}x\,dx, \quad n\neq 1,$

25. $\int \sec^n x\,dx = \frac{\tan x\sec^{n-1}x}{n-1} + \frac{n-2}{n-1}\int \sec^{n-2}x\,dx, \quad n\neq 1,$

26. $\int \csc^n x\,dx = -\frac{\cot x\csc^{n-1}x}{n-1} + \frac{n-2}{n-1}\int \csc^{n-2}x\,dx, \quad n\neq 1,$ **27.** $\int \sinh x\,dx = \cosh x,$ **28.** $\int \cosh x\,dx = \sinh x,$

29. $\int \tanh x\,dx = \ln|\cosh x|,$ **30.** $\int \coth x\,dx = \ln|\sinh x|,$ **31.** $\int \operatorname{sech} x\,dx = \arctan\sinh x,$ **32.** $\int \operatorname{csch} x\,dx = \ln\left|\tanh\frac{x}{2}\right|,$

33. $\int \sinh^2 x\,dx = \frac{1}{4}\sinh(2x) - \frac{1}{2}x,$ **34.** $\int \cosh^2 x\,dx = \frac{1}{4}\sinh(2x) + \frac{1}{2}x,$ **35.** $\int \operatorname{sech}^2 x\,dx = \tanh x,$

36. $\int \operatorname{arcsinh}\frac{x}{a}dx = x\operatorname{arcsinh}\frac{x}{a} - \sqrt{x^2+a^2}, \quad a>0,$

37. $\int \operatorname{arctanh}\frac{x}{a}dx = x\operatorname{arctanh}\frac{x}{a} + \frac{a}{2}\ln|a^2-x^2|,$

38. $\int \operatorname{arccosh}\frac{x}{a}dx = \begin{cases} x\operatorname{arccosh}\dfrac{x}{a} - \sqrt{x^2+a^2}, & \text{if } \operatorname{arccosh}\frac{x}{a}>0 \text{ and } a>0, \\ x\operatorname{arccosh}\dfrac{x}{a} + \sqrt{x^2+a^2}, & \text{if } \operatorname{arccosh}\frac{x}{a}<0 \text{ and } a>0, \end{cases}$

39. $\int \frac{dx}{\sqrt{a^2+x^2}} = \ln\left(x+\sqrt{a^2+x^2}\right), \quad a>0,$

40. $\int \frac{dx}{a^2+x^2} = \frac{1}{a}\arctan\frac{x}{a}, \quad a>0,$

41. $\int \sqrt{a^2-x^2}\,dx = \frac{x}{2}\sqrt{a^2-x^2} + \frac{a^2}{2}\arcsin\frac{x}{a}, \quad a>0,$

42. $\int (a^2-x^2)^{3/2}dx = \frac{x}{8}(5a^2-2x^2)\sqrt{a^2-x^2} + \frac{3a^4}{8}\arcsin\frac{x}{a}, \quad a>0,$

43. $\int \frac{dx}{\sqrt{a^2-x^2}} = \arcsin\frac{x}{a}, \quad a>0,$ **44.** $\int \frac{dx}{a^2-x^2} = \frac{1}{2a}\ln\left|\frac{a+x}{a-x}\right|,$ **45.** $\int \frac{dx}{(a^2-x^2)^{3/2}} = \frac{x}{a^2\sqrt{a^2-x^2}},$

46. $\int \sqrt{a^2\pm x^2}\,dx = \frac{x}{2}\sqrt{a^2\pm x^2} \pm \frac{a^2}{2}\ln\left|x+\sqrt{a^2\pm x^2}\right|,$

47. $\int \frac{dx}{\sqrt{x^2-a^2}} = \ln\left|x+\sqrt{x^2-a^2}\right|, \quad a>0,$

48. $\int \frac{dx}{ax^2+bx} = \frac{1}{a}\ln\left|\frac{x}{a+bx}\right|,$

49. $\int x\sqrt{a+bx}\,dx = \frac{2(3bx-2a)(a+bx)^{3/2}}{15b^2},$

50. $\int \frac{\sqrt{a+bx}}{x}\,dx = 2\sqrt{a+bx} + a\int \frac{1}{x\sqrt{a+bx}}dx,$

51. $\int \frac{x}{\sqrt{a+bx}}\,dx = \frac{1}{\sqrt{2}}\ln\left|\frac{\sqrt{a+bx}-\sqrt{a}}{\sqrt{a+bx}+\sqrt{a}}\right|, \quad a>0,$

52. $\int \frac{\sqrt{a^2-x^2}}{x}\,dx = \sqrt{a^2-x^2} - a\ln\left|\frac{a+\sqrt{a^2-x^2}}{x}\right|,$

53. $\int x\sqrt{a^2-x^2}\,dx = -\frac{1}{3}(a^2-x^2)^{3/2},$

54. $\int x^2\sqrt{a^2-x^2}\,dx = \frac{x}{8}(2x^2-a^2)\sqrt{a^2-x^2} + \frac{a^4}{8}\arcsin\frac{x}{a}, \quad a>0,$

55. $\int \frac{dx}{\sqrt{a^2-x^2}} = -\frac{1}{a}\ln\left|\frac{a+\sqrt{a^2-x^2}}{x}\right|,$

56. $\int \frac{x\,dx}{\sqrt{a^2-x^2}} = -\sqrt{a^2-x^2},$

57. $\int \frac{x^2\,dx}{\sqrt{a^2-x^2}} = -\frac{x}{2}\sqrt{a^2-x^2} + \frac{a^2}{2}\arcsin\frac{x}{a,} \quad a>0,$

58. $\int \frac{\sqrt{a^2+x^2}}{x}\,dx = \sqrt{a^2+x^2} - a\ln\left|\frac{a+\sqrt{a^2+x^2}}{x}\right|,$

59. $\int \frac{\sqrt{x^2-a^2}}{x}\,dx = \sqrt{x^2-a^2} - a\arccos\frac{a}{|x|}, \quad a>0,$

60. $\int x\sqrt{x^2\pm a^2}\,dx = \frac{1}{3}(x^2\pm a^2)^{3/2},$

61. $\int \frac{dx}{x\sqrt{x^2+a^2}} = \frac{1}{a}\ln\left|\frac{x}{a+\sqrt{a^2+x^2}}\right|,$

Theoretical Computer Science Cheat Sheet

Calculus Cont.

62. $\displaystyle\int \frac{dx}{x\sqrt{x^2-a^2}} = \frac{1}{a}\arccos\frac{a}{|x|}, \quad a>0,$

63. $\displaystyle\int \frac{dx}{x^2\sqrt{x^2\pm a^2}} = \mp\frac{\sqrt{x^2\pm a^2}}{a^2x},$

64. $\displaystyle\int \frac{x\,dx}{\sqrt{x^2\pm a^2}} = \sqrt{x^2\pm a^2},$

65. $\displaystyle\int \frac{\sqrt{x^2\pm a^2}}{x^4}\,dx = \mp\frac{(x^2+a^2)^{3/2}}{3a^2x^3},$

66. $$\int \frac{dx}{ax^2+bx+c} = \begin{cases} \dfrac{1}{\sqrt{b^2-4ac}}\ln\left|\dfrac{2ax+b-\sqrt{b^2-4ac}}{2ax+b+\sqrt{b^2-4ac}}\right|, & \text{if } b^2>4ac,\\[2ex] \dfrac{2}{\sqrt{4ac-b^2}}\arctan\dfrac{2ax+b}{\sqrt{4ac-b^2}}, & \text{if } b^2<4ac,\end{cases}$$

67. $$\int \frac{dx}{\sqrt{ax^2+bx+c}} = \begin{cases} \dfrac{1}{\sqrt{a}}\ln\left|2ax+b+2\sqrt{a}\sqrt{ax^2+bx+c}\right|, & \text{if } a>0,\\[2ex] \dfrac{1}{\sqrt{-a}}\arcsin\dfrac{-2ax-b}{\sqrt{b^2-4ac}}, & \text{if } a<0,\end{cases}$$

68. $$\int\sqrt{ax^2+bx+c}\,dx = \frac{2ax+b}{4a}\sqrt{ax^2+bx+c}+\frac{4ax-b^2}{8a}\int\frac{dx}{\sqrt{ax^2+bx+c}},$$

69. $$\int\frac{x\,dx}{\sqrt{ax^2+bx+c}} = \frac{\sqrt{ax^2+bx+c}}{a}-\frac{b}{2a}\int\frac{dx}{\sqrt{ax^2+bx+c}},$$

70. $$\int\frac{dx}{x\sqrt{ax^2+bx+c}} = \begin{cases} \dfrac{-1}{\sqrt{c}}\ln\left|\dfrac{2\sqrt{c}\sqrt{ax^2+bx+c}+bx+2c}{x}\right|, & \text{if } c>0,\\[2ex] \dfrac{1}{\sqrt{-c}}\arcsin\dfrac{bx+2c}{|x|\sqrt{b^2-4ac}}, & \text{if } c<0,\end{cases}$$

71. $\displaystyle\int x^3\sqrt{x^2+a^2}\,dx = (\tfrac{1}{3}x^2-\tfrac{2}{15}a^2)(x^2+a^2)^{3/2},$

72. $\displaystyle\int x^n\sin(ax)\,dx = -\tfrac{1}{a}x^n\cos(ax)+\tfrac{n}{a}\int x^{n-1}\cos(ax)\,dx,$

73. $\displaystyle\int x^n\cos(ax)\,dx = \tfrac{1}{a}x^n\sin(ax)-\tfrac{n}{a}\int x^{n-1}\sin(ax)\,dx,$

74. $\displaystyle\int x^ne^{ax}\,dx = \frac{x^ne^{ax}}{a}-\tfrac{n}{a}\int x^{n-1}e^{ax}\,dx,$

75. $\displaystyle\int x^n\ln(ax)\,dx = x^{n+1}\left(\frac{\ln(ax)}{n+1}-\frac{1}{(n+1)^2}\right),$

76. $\displaystyle\int x^n(\ln ax)^m\,dx = \frac{x^{n+1}}{n+1}(\ln ax)^m-\frac{m}{n+1}\int x^n(\ln ax)^{m-1}\,dx.$

$$\begin{array}{lccc}
x^1 = & x^{\underline{1}} & = & x^{\overline{1}}\\
x^2 = & x^{\underline{2}}+x^{\underline{1}} & = & x^{\overline{2}}-x^{\overline{1}}\\
x^3 = & x^{\underline{3}}+3x^{\underline{2}}+x^{\underline{1}} & = & x^{\overline{3}}-3x^{\overline{2}}+x^{\overline{1}}\\
x^4 = & x^{\underline{4}}+6x^{\underline{3}}+7x^{\underline{2}}+x^{\underline{1}} & = & x^{\overline{4}}-6x^{\overline{3}}+7x^{\overline{2}}-x^{\overline{1}}\\
x^5 = & x^{\underline{5}}+15x^{\underline{4}}+25x^{\underline{3}}+10x^{\underline{2}}+x^{\underline{1}} & = & x^{\underline{5}}-15x^{\overline{4}}+25x^{\overline{3}}-10x^{\overline{2}}+x^{\overline{1}}
\end{array}$$

$$\begin{array}{lclc}
x^{\overline{1}} = & x^1 & x^{\underline{1}} = & x^1\\
x^{\overline{2}} = & x^2+x^1 & x^{\underline{2}} = & x^2-x^1\\
x^{\overline{3}} = & x^3+3x^2+2x^1 & x^{\underline{3}} = & x^3-3x^2+2x^1\\
x^{\overline{4}} = & x^4+6x^3+11x^2+6x^1 & x^{\underline{4}} = & x^4-6x^3+11x^2-6x^1\\
x^{\overline{5}} = & x^5+10x^4+35x^3+50x^2+24x^1 & x^{\underline{5}} = & x^5-10x^4+35x^3-50x^2+24x^1
\end{array}$$

Finite Calculus

Difference, shift operators:

$$\Delta f(x) = f(x+1)-f(x),$$
$$\mathrm{E}\,f(x) = f(x+1).$$

Fundamental Theorem:

$$f(x)=\Delta F(x) \Leftrightarrow \sum f(x)\delta x = F(x)+C.$$
$$\sum_a^b f(x)\delta x = \sum_{i=a}^{b-1} f(i).$$

Differences:

$\Delta(cu) = c\Delta u, \qquad \Delta(u+v) = \Delta u+\Delta v,$

$\Delta(uv) = u\Delta v+\mathrm{E}\,v\Delta u,$

$\Delta(x^{\underline{n}}) = nx^{\underline{n-1}},$

$\Delta(H_x) = x^{\underline{-1}}, \qquad \Delta(2^x) = 2^x,$

$\Delta(c^x) = (c-1)c^x, \qquad \Delta\binom{x}{m} = \binom{x}{m-1}.$

Sums:

$\sum cu\,\delta x = c\sum u\,\delta x,$

$\sum(u+v)\,\delta x = \sum u\,\delta x+\sum v\,\delta x,$

$\sum u\Delta v\,\delta x = uv-\sum \mathrm{E}\,v\Delta u\,\delta x,$

$\sum x^{\underline{n}}\,\delta x = \frac{x^{\underline{n+1}}}{m+1}, \qquad \sum x^{\underline{-1}}\,\delta x = H_x,$

$\sum c^x\,\delta x = \frac{c^x}{c-1}, \qquad \sum\binom{x}{m}\,\delta x = \binom{x}{m+1}.$

Falling Factorial Powers:

$x^{\underline{n}} = x(x-1)\cdots(x-n+1), \quad n>0,$

$x^{\underline{0}} = 1,$

$x^{\underline{n}} = \dfrac{1}{(x+1)\cdots(x+|n|)}, \quad n<0,$

$x^{\underline{n+m}} = x^{\underline{m}}(x-m)^{\underline{n}}.$

Rising Factorial Powers:

$x^{\overline{n}} = x(x+1)\cdots(x+n-1), \quad n>0,$

$x^{\overline{0}} = 1,$

$x^{\overline{n}} = \dfrac{1}{(x-1)\cdots(x-|n|)}, \quad n<0,$

$x^{\overline{n+m}} = x^{\overline{m}}(x+m)^{\overline{n}}.$

Conversion:

$x^{\underline{n}} = (-1)^n(-x)^{\overline{n}} = (x-n+1)^{\overline{n}}$
$\quad = 1/(x+1)^{\overline{-n}},$

$x^{\overline{n}} = (-1)^n(-x)^{\underline{n}} = (x+n-1)^{\underline{n}}$
$\quad = 1/(x-1)^{\underline{-n}},$

$$x^n = \sum_{k=1}^n \left\{{n \atop k}\right\} x^{\underline{k}} = \sum_{k=1}^n \left\{{n \atop k}\right\}(-1)^{n-k}x^{\overline{k}},$$

$$x^{\underline{n}} = \sum_{k=1}^n \left[{n \atop k}\right](-1)^{n-k}x^k,$$

$$x^{\overline{n}} = \sum_{k=1}^n \left[{n \atop k}\right]x^k.$$

Theoretical Computer Science Cheat Sheet

Series

Taylor's series:

$$f(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2}f''(a) + \cdots = \sum_{i=0}^{\infty} \frac{(x-a)^i}{i!} f^{(i)}(a).$$

Expansions:

$$\begin{aligned}
\frac{1}{1-x} &= 1 + x + x^2 + x^3 + x^4 + \cdots &&= \sum_{i=0}^{\infty} x^i,\\
\frac{1}{1-cx} &= 1 + cx + c^2x^2 + c^3x^3 + \cdots &&= \sum_{i=0}^{\infty} c^i x^i,\\
\frac{1}{1-x^n} &= 1 + x^n + x^{2n} + x^{3n} + \cdots &&= \sum_{i=0}^{\infty} x^{ni},\\
\frac{x}{(1-x)^2} &= x + 2x^2 + 3x^3 + 4x^4 + \cdots &&= \sum_{i=0}^{\infty} ix^i,\\
x^k \frac{d^n}{dx^n}\left(\frac{1}{1-x}\right) &= x + 2^n x^2 + 3^n x^3 + 4^n x^4 + \cdots &&= \sum_{i=0}^{\infty} i^n x^i,\\
e^x &= 1 + x + \tfrac{1}{2}x^2 + \tfrac{1}{6}x^3 + \cdots &&= \sum_{i=0}^{\infty} \frac{x^i}{i!},\\
\ln(1+x) &= x - \tfrac{1}{2}x^2 + \tfrac{1}{3}x^3 - \tfrac{1}{4}x^4 - \cdots &&= \sum_{i=1}^{\infty} (-1)^{i+1}\frac{x^i}{i},\\
\ln\frac{1}{1-x} &= x + \tfrac{1}{2}x^2 + \tfrac{1}{3}x^3 + \tfrac{1}{4}x^4 + \cdots &&= \sum_{i=1}^{\infty} \frac{x^i}{i},\\
\sin x &= x - \tfrac{1}{3!}x^3 + \tfrac{1}{5!}x^5 - \tfrac{1}{7!}x^7 + \cdots &&= \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)!},\\
\cos x &= 1 - \tfrac{1}{2!}x^2 + \tfrac{1}{4!}x^4 - \tfrac{1}{6!}x^6 + \cdots &&= \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!},\\
\tan^{-1} x &= x - \tfrac{1}{3}x^3 + \tfrac{1}{5}x^5 - \tfrac{1}{7}x^7 + \cdots &&= \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)},\\
(1+x)^n &= 1 + nx + \tfrac{n(n-1)}{2}x^2 + \cdots &&= \sum_{i=0}^{\infty} \binom{n}{i} x^i,\\
\frac{1}{(1-x)^{n+1}} &= 1 + (n+1)x + \tbinom{n+2}{2}x^2 + \cdots &&= \sum_{i=0}^{\infty} \binom{i+n}{i} x^i,\\
\frac{x}{e^x - 1} &= 1 - \tfrac{1}{2}x + \tfrac{1}{12}x^2 - \tfrac{1}{720}x^4 + \cdots &&= \sum_{i=0}^{\infty} \frac{B_i x^i}{i!},\\
\frac{1}{2x}(1 - \sqrt{1-4x}) &= 1 + x + 2x^2 + 5x^3 + \cdots &&= \sum_{i=0}^{\infty} \frac{1}{i+1}\binom{2i}{i} x^i,\\
\frac{1}{\sqrt{1-4x}} &= 1 + x + 2x^2 + 6x^3 + \cdots &&= \sum_{i=0}^{\infty} \binom{2i}{i} x^i,\\
\frac{1}{\sqrt{1-4x}}\left(\frac{1-\sqrt{1-4x}}{2x}\right)^n &= 1 + (2+n)x + \tbinom{4+n}{2}x^2 + \cdots &&= \sum_{i=0}^{\infty} \binom{2i+n}{i} x^i,\\
\frac{1}{1-x}\ln\frac{1}{1-x} &= x + \tfrac{3}{2}x^2 + \tfrac{11}{6}x^3 + \tfrac{25}{12}x^4 + \cdots &&= \sum_{i=1}^{\infty} H_i x^i,\\
\frac{1}{2}\left(\ln\frac{1}{1-x}\right)^2 &= \tfrac{1}{2}x^2 + \tfrac{3}{4}x^3 + \tfrac{11}{24}x^4 + \cdots &&= \sum_{i=2}^{\infty} \frac{H_{i-1}x^i}{i},\\
\frac{x}{1-x-x^2} &= x + x^2 + 2x^3 + 3x^4 + \cdots &&= \sum_{i=0}^{\infty} F_i x^i,\\
\frac{F_n x}{1 - (F_{n-1}+F_{n+1})x - (-1)^n x^2} &= F_n x + F_{2n}x^2 + F_{3n}x^3 + \cdots &&= \sum_{i=0}^{\infty} F_{ni} x^i.
\end{aligned}$$

Ordinary power series:

$$A(x) = \sum_{i=0}^{\infty} a_i x^i.$$

Exponential power series:

$$A(x) = \sum_{i=0}^{\infty} a_i \frac{x^i}{i!}.$$

Dirichlet power series:

$$A(x) = \sum_{i=1}^{\infty} \frac{a_i}{i^x}.$$

Binomial theorem:

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k.$$

Difference of like powers:

$$x^n - y^n = (x-y)\sum_{k=0}^{n-1} x^{n-1-k} y^k.$$

For ordinary power series:

$$\begin{aligned}
\alpha A(x) + \beta B(x) &= \sum_{i=0}^{\infty} (\alpha a_i + \beta b_i) x^i,\\
x^k A(x) &= \sum_{i=k}^{\infty} a_{i-k} x^i,\\
\frac{A(x) - \sum_{i=0}^{k-1} a_i x^i}{x^k} &= \sum_{i=0}^{\infty} a_{i+k} x^i,\\
A(cx) &= \sum_{i=0}^{\infty} c^i a_i x^i,\\
A'(x) &= \sum_{i=0}^{\infty} (i+1) a_{i+1} x^i,\\
xA'(x) &= \sum_{i=1}^{\infty} i a_i x^i,\\
\int A(x)\,dx &= \sum_{i=1}^{\infty} \frac{a_{i-1}}{i} x^i,\\
\frac{A(x) + A(-x)}{2} &= \sum_{i=0}^{\infty} a_{2i} x^{2i},\\
\frac{A(x) - A(-x)}{2} &= \sum_{i=0}^{\infty} a_{2i+1} x^{2i+1}.
\end{aligned}$$

Summation: If $b_i = \sum_{j=0}^{i} a_i$ then

$$B(x) = \frac{1}{1-x} A(x).$$

Convolution:

$$A(x)B(x) = \sum_{i=0}^{\infty}\left(\sum_{j=0}^{i} a_j b_{i-j}\right) x^i.$$

God made the natural numbers;
all the rest is the work of man.
– Leopold Kronecker

Theoretical Computer Science Cheat Sheet

Series

Expansions:

$$\frac{1}{(1-x)^{n+1}} \ln \frac{1}{1-x} = \sum_{i=0}^{\infty} (H_{n+i} - H_n) \binom{n+i}{i} x^i,$$

$$\left(\frac{1}{x}\right)^{\overline{-n}} = \sum_{i=0}^{\infty} \left\{ {i \atop n} \right\} x^i,$$

$$x^{\overline{n}} = \sum_{i=0}^{\infty} \left[{n \atop i} \right] x^i,$$

$$(e^x - 1)^n = \sum_{i=0}^{\infty} \left\{ {i \atop n} \right\} \frac{n! x^i}{i!},$$

$$\left(\ln \frac{1}{1-x}\right)^n = \sum_{i=0}^{\infty} \left[{i \atop n} \right] \frac{n! x^i}{i!},$$

$$x \cot x = \sum_{i=0}^{\infty} \frac{(-4)^i B_{2i} x^{2i}}{(2i)!},$$

$$\tan x = \sum_{i=1}^{\infty} (-1)^{i-1} \frac{2^{2i}(2^{2i}-1) B_{2i} x^{2i-1}}{(2i)!},$$

$$\zeta(x) = \sum_{i=1}^{\infty} \frac{1}{i^x},$$

$$\frac{1}{\zeta(x)} = \sum_{i=1}^{\infty} \frac{\mu(i)}{i^x},$$

$$\frac{\zeta(x-1)}{\zeta(x)} = \sum_{i=1}^{\infty} \frac{\phi(i)}{i^x},$$

$$\zeta(x) = \prod_p \frac{1}{1-p^{-x}},$$

$$\zeta^2(x) = \sum_{i=1}^{\infty} \frac{d(i)}{x^i} \quad \text{where } d(n) = \textstyle\sum_{d|n} 1,$$

$$\zeta(x)\zeta(x-1) = \sum_{i=1}^{\infty} \frac{S(i)}{x^i} \quad \text{where } S(n) = \textstyle\sum_{d|n} d,$$

$$\zeta(2n) = \frac{2^{2n-1}|B_{2n}|}{(2n)!} \pi^{2n}, \quad n \in \mathbb{N},$$

$$\frac{x}{\sin x} = \sum_{i=0}^{\infty} (-1)^{i-1} \frac{(4^i - 2) B_{2i} x^{2i}}{(2i)!},$$

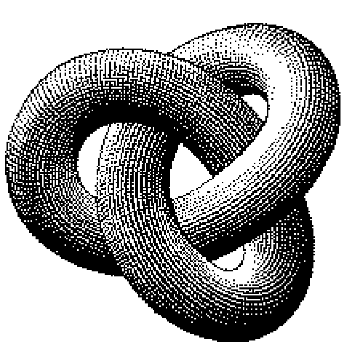
$$\left(\frac{1-\sqrt{1-4x}}{2x}\right)^n = \sum_{i=0}^{\infty} \frac{n(2i+n-1)!}{i!(n+i)!} x^i,$$

$$e^x \sin x = \sum_{i=1}^{\infty} \frac{2^{i/2} \sin \frac{i\pi}{4}}{i!} x^i,$$

$$\sqrt{\frac{1-\sqrt{1-x}}{x}} = \sum_{i=0}^{\infty} \frac{(4i)!}{16^i \sqrt{2}(2i)!(2i+1)!} x^i,$$

$$\left(\frac{\arcsin x}{x}\right)^2 = \sum_{i=0}^{\infty} \frac{4^i i!^2}{(i+1)(2i+1)!} x^{2i}.$$

Escher's Knot



Stieltjes Integration

If G is continuous in the interval $[a,b]$ and F is nondecreasing then

$$\int_a^b G(x)\, dF(x)$$

exists. If $a \le b \le c$ then

$$\int_a^c G(x)\, dF(x) = \int_a^b G(x)\, dF(x) + \int_b^c G(x)\, dF(x).$$

If the integrals involved exist

$$\int_a^b \big(G(x) + H(x)\big)\, dF(x) = \int_a^b G(x)\, dF(x) + \int_a^b H(x)\, dF(x),$$

$$\int_a^b G(x)\, d\big(F(x) + H(x)\big) = \int_a^b G(x)\, dF(x) + \int_a^b G(x)\, dH(x),$$

$$\int_a^b c \cdot G(x)\, dF(x) = \int_a^b G(x)\, d\big(c \cdot F(x)\big) = c \int_a^b G(x)\, dF(x),$$

$$\int_a^b G(x)\, dF(x) = G(b)F(b) - G(a)F(a) - \int_a^b F(x)\, dG(x).$$

If the integrals involved exist, and F possesses a derivative F' at every point in $[a,b]$ then

$$\int_a^b G(x)\, dF(x) = \int_a^b G(x) F'(x)\, dx.$$

Cramer's Rule

If we have equations:

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n &= b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n &= b_2 \\ \vdots \qquad\quad \vdots \qquad\qquad\quad \vdots \;\; & \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,n}x_n &= b_n \end{aligned}$$

Let $A = (a_{i,j})$ and B be the column matrix (b_i). Then there is a unique solution iff $\det A \ne 0$. Let A_i be A with column i replaced by B. Then

$$x_i = \frac{\det A_i}{\det A}.$$

00	47	18	76	29	93	85	34	61	52
86	11	57	28	70	39	94	45	02	63
95	80	22	67	38	71	49	56	13	04
59	96	81	33	07	48	72	60	24	15
73	69	90	82	44	17	58	01	35	26
68	74	09	91	83	55	27	12	46	30
37	08	75	19	92	84	66	23	50	41
14	25	36	40	51	62	03	77	88	99
21	32	43	54	65	06	10	89	97	78
42	53	64	05	16	20	31	98	79	87

Fibonacci Numbers

$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$

Definitions:

$$F_i = F_{i-1} + F_{i-2}, \quad F_0 = F_1 = 1,$$

$$F_{-i} = (-1)^{i-1} F_i,$$

$$F_i = \frac{1}{\sqrt{5}} \left(\phi^i - \hat{\phi}^i\right),$$

Cassini's identity: for $i > 0$:

$$F_{i+1}F_{i-1} - F_i^2 = (-1)^i.$$

Additive rule:

$$F_{n+k} = F_k F_{n+1} + F_{k-1} F_n,$$

$$F_{2n} = F_n F_{n+1} + F_{n-1} F_n.$$

Calculation by matrices:

$$\begin{pmatrix} F_{n-2} & F_{n-1} \\ F_{n-1} & F_n \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n.$$

The Fibonacci number system: Every integer n has a unique representation

$$n = F_{k_1} + F_{k_2} + \cdots + F_{k_m},$$

where $k_i \ge k_{i+1} + 2$ for all i, $1 \le i < m$ and $k_m \ge 2$.

Improvement makes strait roads, but the crooked roads without Improvement, are roads of Genius.
– William Blake (The Marriage of Heaven and Hell)