Team Contest Reference
Team: stoptryharding

Florian Marwitz
Ian Pösse
Marcel Wienöbst

Contents

1 ds

1.1 Fenwick-Tree

Can be used for computing prefix sums.

```
//note that 0 can not be used
//globally create array
int fwktree[1000001];
int read(int index) {
   int sum = 0;
   while (index > 0) {
      sum += fwktree[index];
      index -= (index & -index);
   }
   return sum;
}
// n is the actual size of the tree (e.g. the array is
     used from 1 to n-1)
void update(int index, int addValue, int n) {
   while (index <= n - 1) {
      fwktree[index] += addValue;
      index += (index & -index);
   }
}
```

MD5: e4a8636bf315ee63c280f209bb98ed0d | $\mathcal{O}(logn)$

1.2 Range maximum query

finds maximum in range [i,j] in O(1) preprocessing takes O(n log n)

```
// create A globally, contains the input
int A[10000];
// M is the DP table has size N*log N
int M[10000][20];
// N is the input size
void process(int N) {
        for(int i = 0; i < N; i++)
                M[i][0] = i;
        // filling table M
        // M[i][j] = max(M[i][j-1], M[i+(1<<(j-1))][j
            -1]),
        // cause interval of length 2^j can be
            partitioned
        // into two intervals of length 2^(j-1)
        for(int j = 1; 1 << j <= N; j++) {
                for(int i = 0; i + (1 << j) - 1 < N; i
                    ++) {
                        if(A[M[i][j-1]] >= A[M[i+(1 <<
                             (j-1))][j-1]])
                                M[i][j] = M[i][j-1];
                        else
                                M[i][j] = M[i + (1 <<
                                    (j-1))][j-1];
                }
        }
}
// range is [i,j], returns index of max
int query(int N, int i, int j) {
        // k = |_ log_2(j-i+1) _|
        int k = (int) (log(j - i + 1) / log(2));
        if(A[M[i][k]] >= A[M[j- (1 << k) + 1][k]])
                return M[i][k];
        else
                return M[j - (1 << k) + 1][k];
}
```

MD5: eae61471981a55989f42aa6631bb2f13 | $\mathcal{O}(?)$

1.3 Suffix array

```
vector<int> sa, pos, tmp, lcp;
string s;
int N, gap;

bool sufCmp(int i, int j) {
  if(pos[i] != pos[j])
    return pos[i] < pos[j];
  i += gap;
  j += gap;
  return (i < N && j < N) ? pos[i] < pos[j] : i > j;
}

void buildSA()
{
  N = s.size();
```

```
  for(int i = 0; i < N; ++i) {
    sa.push_back(i);
    pos.push_back(s[i]);
  }
  tmp.resize(N);
  for(gap = 1;;gap *= 2) {
    sort(sa.begin(), sa.end(), sufCmp);
    for(int i = 0; i < N - 1; ++i) {
      tmp[i+1] = tmp[i] + sufCmp(sa[i], sa[i+1]);
    }
    for(int i = 0; i < N; ++i) {
      pos[sa[i]] = tmp[i];
    }
    if(tmp[N-1] == N-1) break;
  }
}

void buildLCP()
{
  lcp.resize(N);
  for(int i = 0, k = 0; i < N; ++i) {
    if(pos[i] != N - 1) {
      for(int j = sa[pos[i] + 1]; s[i + k] == s[j + k
          ];) {
        ++k;
      }
      lcp[pos[i]] = k;
      if (k) --k;
    }
  }
}

int main()
{
  string r, t;
  cin >> r >> t;
  s = r + "§" + t;
  buildSA();
  buildLCP();
  for(int i = 0; i < N; ++i) {
    cout << sa[i] << " " << lcp[i] << endl;
  }
  //suffix arrays can be used for various things:
  //for example: finding lcs between to strings
}
```

MD5: 47eb870ecfe9cb548eb96a15c077fab7 | $\mathcal{O}(?)$

1.4 trie

source: github -> SuprDewd

```
template <class T>
struct trie {
  struct node {
    map<T, node*> children;
    int prefixes, words;
    node() { prefixes = words = 0; } };
  node* root;
  trie() : root(new node()) {  }
  template <class I>
  void insert(I begin, I end) {
    node* cur = root;
    while (true) {
      cur->prefixes++;
      if (begin == end) { cur->words++; break; }
      else {
        T head = *begin;
        typename map<T, node*>::const_iterator it;
        it = cur->children.find(head);
        if (it == cur->children.end()) {
          pair<T, node*> nw(head, new node());
          it = cur->children.insert(nw).first;
        } begin++, cur = it->second; } } }
  template<class I>
  int countMatches(I begin, I end) {
    node* cur = root;
    while (true) {
      if (begin == end) return cur->words;
      else {
        T head = *begin;
        typename map<T, node*>::const_iterator it;
        it = cur->children.find(head);
        if (it == cur->children.end()) return 0;
        begin++, cur = it->second; } } }
  template<class I>
  int countPrefixes(I begin, I end) {
    node* cur = root;
    while (true) {
      if (begin == end) return cur->prefixes;
      else {
        T head = *begin;
        typename map<T, node*>::const_iterator it;
        it = cur->children.find(head);
        if (it == cur->children.end()) return 0;
begin++, cur = it->second; } } } };

// use as follows
trie<char> t;
string s = "aaa";
t.insert("aaa");
```

MD5: 4410c62ce77f58cf564ac6881096d200 | $\mathcal{O}(?)$

1.5 Union-Find

union joins the sets x and y are contained in. *find* returns the representative of the set x is contained in.
Input: number of elements n, element x, element y
Output: the representative of element x or a boolean indicating whether sets got merged.

```
// globally create arrays
int p[100000];
int r[100000];

int find(int x) {
    int root = x;
    while (p[root] >= 0) { // find root
        root = p[root];
    }
    while (p[x] >= 0) { // path compression
        int tmp = p[x];
        p[x] = root;
        x = tmp;
    }
    return root;
}

// return true, if sets merged and false, if already
    from same set
bool union(int x, int y) {
    int px = find(x);
```

```
    int py = find(y);
    if (px == py)
        return false; // same set -> reject edge
    if (r[px] < r[py]) { // swap so that always h[px]>=h[py]
        int tmp = px;
        px = py;
        py = tmp;
    }
    p[py] = px; // hang flatter tree as child of higher tree
    r[px] = max(r[px], r[py] + 1); // update (worst-case) height
    // if needed use count here
    // count--;
    return true;
}

int main() {
    // init count to number of nodes if it is needed
    // int count = n;

    for(int i = 0; i < n; ++i) {
        p[i] = -1;
    }
    // do something
}
```

MD5: `ad8e7972aec615641842e4e82858cb97` | $\mathcal{O}(\alpha(n))$

2 graph

2.1 2SAT

```
// create implication graph
// do SCC
// check if var and its negation are in the same component
```

MD5: `a2e8b2ae500366ce942af79e0a3f4283` | $\mathcal{O}(|V| + |E|)$

2.2 BellmanFord

Finds shortest pathes from a single source. Negative edge weights are allowed. Can be used for finding negative cycles.

```
// globally create arrays and graph
vector<vector<pair<int, int>>> g;
int dist[n];
int MAX_VALUE = (1 << 30);

bool bellmanFord() {
    //source is 0
    dist[0] = 0;
    //calc distances
    //the path has max length |V|-1
    for(int i = 0; i < n-1; i++) {
        //each iteration relax all edges
        for(int j = 0; j < n; j++) {
            for(int k = 0; k < g[j].size(); ++k) {
                pair<int, int> e = g[j][k];
                if(dist[j] != MAX_VALUE
                   && dist[e.first] > dist[j] + e.second) {
                    dist[e.first] = dist[j] + e.second;
                }
            }
        }
    }
    //check for negative-weight cycle
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < g[i].size(); ++j) {
            if(dist[i] != Integer.MAX_VALUE
               && dist[e.first] > dist[i] + e.second) {
                return true;
            }
        }
    }
    return false;
}
```

MD5: `0dfb4089a47db73dbaaad5add58fd2a0` | $\mathcal{O}(|V| \cdot |E|)$

2.3 bipartite graph check

```
// traverse through graph with bfs
// assign labels 0 and 1
// if child is unexplored it gets different label from parent and put in the queue
// if already visited check if labels are different
```

MD5: `0b64ac42e8b846e97c338eaeb7d73575` | $\mathcal{O}(|V| + |E|)$

2.4 Maximum Bipartite Matching

Finds the maximum bipartite matching in an unweighted graph using DFS.

Input: An unweighted adjacency matrix boolean[M][N] with M nodes being matched to N nodes.

Output: The maximum matching. (For getting the actual matching, little changes have to be made.)

```
// globally create graph array
// adjacency matrix but smaller as only edges between M and N exist
bool bpGraph[M][N];

// A DFS based recursive function
// that returns true if a matching
// for vertex u is possible
bool bpm(int u,  vector<bool> &seen, vector<int> &matchR)
{
    // Try every job one by one
    for (int v = 0; v < N; v++)
    {
        // If applicant u is interested in
        // job v and v is not visited
        if (bpGraph[u][v] && !seen[v])
        {
            // Mark v as visited
            seen[v] = true;

            // If job v is not assigned to an
            // applicant OR previously assigned
            // applicant for job v (which is matchR[v])
```

```
            // has an alternate job available.
            // Since v is marked as visited in
            // the above line, matchR[v] in the following
            // recursive call will not get job v again
            if (matchR[v] < 0 || bpm(bpGraph, matchR[v],
                                     seen, matchR))
            {
                matchR[v] = u;
                return true;
            }
        }
    }
    return false;
}

// Returns maximum number
// of matching from M to N
int maxBPM()
{
    // An array to keep track of the
    // applicants assigned to jobs.
    // The value of matchR[i] is the
    // applicant number assigned to job i,
    // the value -1 indicates nobody is
    // assigned.
    vector<int> matchR (N);

    // Initially all jobs are available
    for(int i = 0; i < N; ++i) {
        matchR[i] = -1;
    }

    // Count of jobs assigned to applicants
    int result = 0;
    for (int u = 0; u < M; u++)
    {
        // Mark all jobs as not seen
        // for next applicant.
        vector<int> seen (N);

        // Find if the applicant u can get a job
        if (bpm(bpGraph, u, seen, matchR))
            result++;
    }
    return result;
}
```

MD5: `035f3ecf4735d724aad793ac4c1417c3` | $\mathcal{O}(M \cdot N)$

2.5 shortest path for dags

can also be applied to longest path problem in dags

```
// calc topological sorting
// go through nodes in ts order
// relaxate its neighbours
```

MD5: `337da9f825b3decf382ab7a8278b025c` | $\mathcal{O}(|V| + |E|)$

2.6 Recursive Depth First Search

Recursive DFS with different options (storing times, connected/unconnected graph). this is very much pseudocode, needs a lot of problem adaption anyway

Input: A source vertex s, a target vertex t, and adjlist G and the time (0 at the start)
Output: Indicates if there is connection between s and t.

```
// globally create adj list etc
vector<vector<int>> g;
int dtime[n];
int ftime[n];
int vis[n];
int pre[n];
//first call with time = 0
void rec_dfs(int u, int time){
    //it might be necessary to store the time of discovery
    time = time + 1;
    dtime[u] = time;

    vis[u] = 1; //new vertex has been discovered
    //For cycle check vis should be int and 0 are not vis nodes
    //1 are vis nodes which havent been finished and 2 are finished nodes
    //cycle exists iff edge to node with vis=1
    //when reaching the target return true
    //not necessary when calculating the DFS-tree
    for(int i = 0; i < g[u].size(); ++i) {
        int v = g[u][i];
        //exploring a new edge
        if(!vis[v]) {
            pre[v] = u;
            if(rec_dfs(v, time)) return true;
        }
    }
    //storing finishing time
    time = time + 1;
    ftime[s] = time;
    vis[s] = 2;
    return false;
}

//if we want to visit the whole graph, even if it is not connected we might use this
//make sure all vertices vis value is false etc
int time = 0;
for(int i = 0; i < n; i++) {
    if(vis[i] == 0) {
        //note that we leave out t so this does not work with the below function
        //adaption will not be too difficult though
        //time should not always start at zero, change if needed
        rec_dfs(i, 0);
    }
 }
}
```

MD5: `56aad25fb3082bfd389df98e5371262e` | $\mathcal{O}(|V| + |E|)$

2.7 Dijkstra

Finds the shortest paths from one vertex to every other vertex in the graph (SSSP).
For negative weights, add |min|+1 to each edge, later subtract from result.
To get a different shortest path when edges are ints, add an $\varepsilon = \frac{1}{k+1}$ on each edge of the shortest path of length k, run again.

Input: A source vertex s and an adjacency list G.

Output: Modified adj. list with distances from s and predcessor vertices set.

```
int mxi = (1 << 25);

bool cmp(pair<int, int> a, pair<int, int> b)
{
    //make sure this is the right way around!
    return (a.second > b.second);
}

int dijkstra(vector<vector<pair<int, int>>> &g, int N)
{
    priority_queue<pair<int, int>, vector<pair<int,
        int>>, decltype(cmp) *> pq(cmp);
    vector<int> dist (N, mxi);
    dist[0] = 0;
    pq.push({0, 0});
    while(!pq.empty()) {
        int u = pq.top().first;
        int d = pq.top().second;
        pq.pop();
        if(d > dist[u]) continue;
        if(u == N-1) return d;
        for(auto it = g[u].begin(); it != g[u].end();
            ++it) {
            int v = it -> first;
            int w = it -> second;
            if(w + dist[u] < dist[v]) {
                dist[v] = w + dist[u];
                pq.push({v, dist[v]});
            }
        }
    }
    return dist[N-1];
}
```

MD5: ab999003aab97fd17ec9ce012dc8b0a9 | $\mathcal{O}(|E| \log |V|)$

2.8 FloydWarshall

Finds all shortest paths. Paths in array next, distances in ans.

```
int MAX_VALUE = (1 << 30);

void floydWarshall(int[][] graph,
                   int[][] next, int[][] ans, int n) {
    for(int i = 0; i < n; i++)
        for(int j = 0; j < n; j++)
            ans[i][j] = graph[i][j];

    for (int k = 0; k < n; k++)
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                if (ans[i][k] + ans[k][j] < ans[i][j]
                    && ans[i][k] < MAX_VALUE
                    && ans[k][j] < MAX_VALUE) {
                    ans[i][j] = ans[i][k] + ans[k][j];
                    next[i][j] = next[i][k];
                }
}
```

MD5: d93432a80b6b67952eedde97a4e7df79 | $\mathcal{O}(|V|^3)$

2.9 Hungarian algorithm

Finds the minimal matching (sum of edge weights should be minimal) in a bipartite graph with weights

```
const int MAXN = 105;
const int INF = 1000 * 1000 * 1000;

int n;
int a[MAXN][MAXN];
int u[MAXN], v[MAXN], link[MAXN], par[MAXN], used[MAXN
    ], minval[MAXN];

int main() {
    //freopen("input.txt","r",stdin);
    //freopen("output.txt","w",stdout);
    // input is the matrix a (jobs are rows, columns
        are workers)
    scanf("%d", &n);
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            scanf("%d", &a[i][j]);

    for (int i = 1; i <= n; i++) {
        for (int j = 0; j < MAXN; j++) {
            used[j] = false;
            minval[j] = INF;
        }
        int j_cur = 0;
        par[j_cur] = i;
        do {
            used[j_cur] = true;
            int j_next, delta = INF, i_cur = par[j_cur
                ];
            for (int j = 0; j <= n; j++)
                if (!used[j]) {
                    int cur = a[i_cur][j] - u[i_cur] -
                        v[j];
                    if (cur < minval[j]) {
                        minval[j] = cur; link[j] =
                            j_cur;
                    }
                    if (minval[j] < delta) {
                        delta = minval[j]; j_next = j;
                    }
                }
            for (int j = 0; j <= n; j++)
                if (used[j]) {
                    u[par[j]] += delta; v[j] -= delta;
                }
                else {
                    minval[j] -= delta;
                }
            j_cur = j_next;
        } while (par[j_cur]);
        do {
            int j_prev = link[j_cur];
            par[j_cur] = par[j_prev];
            j_cur = j_prev;
        } while (j_cur > 0);
    }

    printf("%d", -v[0]);
    return 0;
}
```

MD5: 7affea02e98658db60c58072dfa85775 | $\mathcal{O}(n^3)$

2.10 kruskal algorithm

finds the minimum spanning tree

```
// sort edges by increasing weight
// init union find (the nodes are the sets)
// go through the sorted edges and check if the
    corresponding nodes
// are in the same set, if yes skip the edge, if no
    the edge is part
// of the minimum spanning tree -> unite nodes
```

MD5: 82c91537f2425cfed1809d2f685dafcd | $\mathcal{O}(|E|\log(|E|))$

2.11 Edmonds-Karp

Finds the greatest flow in a graph. Capacities must be positive.

```
#include<iostream>
#include<vector>
#include<queue>
#include<unordered_map>
#include<cmath>

using namespace std;

bool bfs(vector<unordered_map<int, long long>> &g, int
     s, int t, vector<int> &pre)
{
    int n = g.size();
    for(int i = 0; i < n; ++i) {
        pre[i] = -1;
    }
    vector<bool> vis (n);
    queue<int> q;
    vis[s] = true;
    q.push(s);
    while(!q.empty()) {
        int u = q.front();
        q.pop();
        if(u == t) return true;
        for(auto v = g[u].begin(); v != g[u].end(); ++
            v) {
            if(!vis[v->first] && (v->second) > 0) {
                vis[v->first] = true;
                pre[v->first] = u;
                q.push(v->first);
            }
        }
    }
    return vis[t];
}

long long ed_karp(vector<unordered_map<int, long long
    >> &g, int s, int t)
{
    long long mxf = 0;
    int n = g.size();
    vector<int> pre (n);
    while(bfs(g, s, t, pre)) {
        long long pf = (1L << 58);
        for(int v = t; v != s; v = pre[v]) {
            int u = pre[v];
            pf = min(pf, g[u][v]);
        }
        for(int v = t; v != s; v = pre[v]) {
            int u = pre[v];
            g[u][v] -= pf;
            g[v][u] += pf;
        }
        mxf += pf;
    }
    return mxf;
}
```

MD5: 7ea28f5038311710693958817l692efe | $\mathcal{O}(|V|^2 \cdot |E|)$

2.12 find min cut edges

```
// do a maxflow
// go through residual graph with dfs or bfs
    traversing edges with residual cap > 0 and
// back edges with flow > 0, mark all visited nodes
// then output all edges from a marked to an unmarked
    node (maybe another BFS or something)
```

MD5: fb27cd04a3f1ab0ea7e494c40be18fbe | $\mathcal{O}(|V|+|E|)$

2.13 strongly connected components

```
// use two DFSs
// 1. DFS: topological sort produces list l
// 2. DFS: go through sorting and for transposed graph
     (edges are flipped) do the DFS, all reached nodes
     get the same label (are in the same component),
    of course BFS could also be used
```

MD5: 8ba4235a4fe35b79c0c3d4a86341c525 | $\mathcal{O}(|E|+|V|)$

2.14 topological sort

```
//two options:
//1. remove nodes with in-degree 0
//2. do DFS and prepend nodes to list when they are
    done
// (so all the nodes they depend on have already been
    prepended as they already finished)
```

MD5: db8519c36fbafe6a952fa5c808a5932e | $\mathcal{O}(|E|+|V|)$

3 math

3.1 binomial coefficient

gives binomial coefficient (n choose k)

```
// note that if we have to calculate the bin coeff
    modulo some prime
// we cannot divide, but have to multiply by the
    inverse of k
// that can be easily computed as k^p-2 % p with
    modular exponentiation (use successive squaring)
// another approach would be to just calculate n! / ((
    n-k)!*k!) (again invert denominator and use mod in
     all steps)
long long bin(int n, int k) {
  if (k == 0)
    return 1;
  else if (k > n/2)
    return bin(n, n-k);
  else
```

```
    return n*bin(n-1, k-1)/k;
}
```

MD5: `610ff61f07eef70ca116e75e1b15cf7c` | $\mathcal{O}(k)$

3.2 Iterative EEA

Calculates the gcd of a and b and their modular inverse $x = a^{-1} \mod b$ and $y = b^{-1} \mod a$.

```
// extended euclidean algorithm - iterativ
if (b > a) {
    long tmp = a;
    a = b;
    b = tmp;
 }
long x = 0, y = 1, u = 1, v = 0;
while (a != 0) {
    long q = b / a, r = b % a;
    long m = x - u * q, n = y - v * q;
    b = a; a = r; x = u; y = v; u = m; v = n;
 }
long gcd = b;
// x = a^-1 % b, y = b^-1 % a
// ax + by = gcd
```

MD5: `737c57d8f09d748f54c57851ea1e759d` | $\mathcal{O}(\log a + \log b)$

3.3 Fourier transform

calculates the fourier transform for a given vector here used for polynom multiplication in O(n log n)

```
// pol is the vector that should be transformed
// fft is the resulting vector (note the complex numbers)
// n is the size of pol and fft which has to be of the form 2^k (just fill up with zeros and choose big enough size)
// if inv = true the inverse transform is calculated (here too the result can be found in fft!)
void iterativefft(const vector<long long> &pol, vector<complex<double>> &fft, int n, bool inv)
{
    //copy pol into fft
    if(!inv) {
        for(int i = 0; i < n; ++i) {
            complex<double> cp (pol[i], 0);
            fft[i] = cp;
        }
    }
    //swap positions accordingly
    for(int i = 0, j = 0; i < n; ++i) {
        if(i < j) swap(fft[i], fft[j]);
        int m = n >> 1;
        while(1 <= m && m <= j) j -= m, m >>= 1;
        j += m;
    }
    for(int m = 1; m <= n; m <<= 1) { //<= or <
        double theta = (inv ? -1 : 1) * 2 * M_PI / m;
        complex<double> wm(cos(theta), sin(theta));
        for(int k = 0; k < n; k += m) {
            complex<double> w = 1;
            for(int j = 0; j < m/2; ++j) {
                complex<double> t = w * fft[k + j + m/2];
                complex<double> u = fft[k + j];
                fft[k + j] = u + t;
                fft[k + j + m/2] = u - t;
                w = w*wm;
            }
        }
    }
    if(inv) {
        for(int i = 0; i < n; ++i) {
            fft[i] /= complex<double> (n);
        }
    }
}
// the polynom pol gets squared, the result is put in res
vector<complex<double>> fft (n);
iterativefft(pol, fft, n, false);
for(int i = 0; i < n; ++i) {
    fft[i] *= fft[i];
 }
iterativefft(pol, fft, n, true);
vector<long long> res(n);
for(int i = 0; i < n; ++i) {
    res[i] = round(fft[i].real());
 }
```

MD5: `9dd418b1bc3d7685c5c55b287cc8555e` | $\mathcal{O}(n \log n)$

3.4 Greatest Common Divisor

Calculates the gcd of two numbers a and b or of an array of numbers $input$.

Input: Numbers a and b or array of numbers $input$

Output: Greatest common divisor of the input

```
long long gcd(long long a, long long b) {
    while (b > 0) {
        long long temp = b;
        b = a % b; // % is remainder
        a = temp;
    }
    return a;
}

long long gcd(vector<long long> &input) {
    long long result = input[0];
    for(int i = 1; i < input.size(); i++)
    result = gcd(result, input[i]);
    return result;
}
```

MD5: `27f69f32d6e1f59d16b9c8ea0028a9fb` | $\mathcal{O}(\log a + \log b)$

3.5 geometry lib

```
// this library has been copied from https://github.com/SuprDewd/T-414-AFLV
#include <complex>
using namespace std;
#define P(p) const point &p
#define L(p0, p1) P(p0), P(p1)
#define C(p0, r) P(p0), double r
#define PP(pp) pair<point,point> &pp
typedef complex<double> point;
const double pi = acos(-1.0);
```

```
const double EPS = 1e-9;
double dot(P(a), P(b)) {
    return real(conj(a) * b);
}
double cross(P(a), P(b)) {
    return imag(conj(a) * b);
}
point rotate(P(p), double radians = pi / 2, P(about) =
     point(0,0)) {
    return (p - about) * exp(point(0, radians)) +
        about;
}
point proj(P(u), P(v)) {
    return dot(u, v) / dot(u, u) * u;
}
point normalize(P(p), double k = 1.0) {
    return abs(p) == 0 ? point(0,0) : p / abs(p) * k;
}
bool parallel(L(a, b), L(p, q)) {
    return abs(cross(b - a, q - p)) < EPS;
}
double ccw(P(a), P(b), P(c)) {
    return cross(b - a, c - b);
}
bool collinear(P(a), P(b), P(c)) { return abs(ccw(a, b
    , c)) < EPS; }
double angle(P(a), P(b), P(c)) {
    return acos(dot(b - a, c - b) / abs(b - a) / abs(c
         - b));
}
bool intersect(L(a, b), L(p, q), point &res, bool
    segment = false) {
    // NOTE: check for parallel/collinear lines before
         calling this function
    point r = b - a, s = q - p;
    double c = cross(r, s), t = cross(p - a, s) / c, u
         = cross(p - a, r) / c;
    if (segment && (t < 0-EPS || t > 1+EPS || u < 0-
        EPS || u > 1+EPS))
        return false;
    res = a + t * r;
    return true;
}
point closest_point(L(a, b), P(c), bool segment =
    false) {
    if (segment) {
        if (dot(b - a, c - b) > 0) return b;
        if (dot(a - b, c - a) > 0) return a;
    }
    double t = dot(c - a, b - a) / norm(b - a);
    return a + t * (b - a);
}

typedef vector<point> polygon;
#define MAXN 1000
point hull[MAXN];
bool cmp(const point &a, const point &b) {
    return abs(real(a) - real(b)) > EPS ?
        real(a) < real(b) : imag(a) < imag(b); }
// note that this might fail some weird edge cases
    like a small case of three colinear points!
// also not totally clear what this returns, possibly
    something like the point in the lower left corner?
int convex_hull(vector<point> p) {
    int n = p.size(), l = 0;
    sort(p.begin(), p.end(), cmp);
    for (int i = 0; i < n; i++) {
        if (i > 0 && p[i] == p[i - 1])
            continue;
        while (l >= 2 && ccw(hull[l - 2], hull[l - 1],
             p[i]) >= 0)
            l--;
        hull[l++] = p[i];
    }
    int r = l;
    for (int i = n - 2; i >= 0; i--) {
        if (p[i] == p[i + 1])
            continue;
        while (r - l >= 1 && ccw(hull[r - 2], hull[r -
             1], p[i]) >= 0)
            r--;
        hull[r++] = p[i];
    }
    return l == 1 ? 1 : r - 1;
}
```

MD5: 5606cce22e2a4fa50e8ee45227bb1d40 | $\mathcal{O}(?)$

3.6 Least Common Multiple

Calculates the lcm of two numbers a and b or of an array of numbers $input$.

Input: Numbers a and b or array of numbers $input$

Output: Least common multiple of the input

```
long long lcm(long long a, long long b) {
    return a * (b / gcd(a, b));
}

long long lcm(vector<long long> &input) {
  long result = input[0];
  for(int i = 1; i < input.size(); i++)
    result = lcm(result, input[i]);
  return result;
}
```

MD5: f9b4919c74ef3ca9c1e0e2964d59fd7b | $\mathcal{O}(\log a + \log b)$

3.7 phi function calculator

```
int phi(int n)
{
    double result = n;
    for(int p = 2; p * p <= n; ++p) {
        if(n % p == 0) {
            while(n % p == 0) n /= p;
            result *= (1.0 - (1.0 / (double) p));
        }
    }
    if(n > 1) result *= (1.0 - (1.0 / (double) n));
    return round(result);
}
```

MD5: 2ec930cc10935f1638700bb74e3439d9 | $\mathcal{O}(\sqrt{(n)})$

3.8 Sieve of Eratosthenes

Calculates Sieve of Eratosthenes.

Input: A integer N indicating the size of the sieve.

Output: A boolean array, which is true at an index i iff i is prime.

```
vector<boolean> is_prime (n+1);
for (int i = 2; i <= n; i++) is_prime[i] = true;
for (int i = 2; i*i <= n; i++)
   if (is_prime[i])
  for (int j = i*i; j <= n; j+=i)
      is_prime[j] = false;
```

MD5: `2b965443a98027ed7f531d5360e00b48` | $\mathcal{O}(n)$

3.9 Simplex algorithm

source: github -> SuprDewd

```
typedef long double DOUBLE;
typedef vector<DOUBLE> VD;
typedef vector<VD> VVD;
typedef vector<int> VI;
const DOUBLE EPS = 1e-9;
struct LPSolver {
 int m, n;
 VI B, N;
 VVD D;
 LPSolver(const VVD &A, const VD &b, const VD &c) :
  m(b.size()), n(c.size()),
  N(n + 1), B(m), D(m + 2, VD(n + 2)) {
  for (int i = 0; i < m; i++) for (int j = 0; j < n; j++)
    D[i][j] = A[i][j];
  for (int i = 0; i < m; i++) { B[i] = n + i; D[i][n] = -1;
    D[i][n + 1] = b[i]; }
  for (int j = 0; j < n; j++) { N[j] = j; D[m][j] = -c[j]; }
  N[n] = -1; D[m + 1][n] = 1; }
 void Pivot(int r, int s) {
  double inv = 1.0 / D[r][s];
  for (int i = 0; i < m + 2; i++) if (i != r)
   for (int j = 0; j < n + 2; j++) if (j != s)
    D[i][j] -= D[r][j] * D[i][s] * inv;
  for (int j = 0; j < n + 2; j++) if (j != s) D[r][j] *= inv;
  for (int i = 0; i < m + 2; i++) if (i != r) D[i][s] *= -inv;
  D[r][s] = inv;
  swap(B[r], N[s]); }
 bool Simplex(int phase) {
  int x = phase == 1 ? m + 1 : m;
  while (true) {
   int s = -1;
   for (int j = 0; j <= n; j++) {
    if (phase == 2 && N[j] == -1) continue;
    if (s == -1 || D[x][j] < D[x][s] ||
        D[x][j] == D[x][s] && N[j] < N[s]) s = j; }
   if (D[x][s] > -EPS) return true;
   int r = -1;
   for (int i = 0; i < m; i++) {
    if (D[i][s] < EPS) continue;
    if (r == -1 || D[i][n + 1] / D[i][s] < D[r][n + 1] /
        D[r][s] || (D[i][n + 1] / D[i][s]) == (D[r][n + 1] /
        D[r][s]) && B[i] < B[r]) r = i; }
   if (r == -1) return false;
   Pivot(r, s); } }
 DOUBLE Solve(VD &x) {
  int r = 0;
  for (int i = 1; i < m; i++) if (D[i][n + 1] < D[r][n + 1])
    r = i;
  if (D[r][n + 1] < -EPS) {
   Pivot(r, n);
   if (!Simplex(1) || D[m + 1][n + 1] < -EPS)
     return -numeric_limits<DOUBLE>::infinity();
   for (int i = 0; i < m; i++) if (B[i] == -1) {
    int s = -1;
    for (int j = 0; j <= n; j++)
     if (s == -1 || D[i][j] < D[i][s] ||
         D[i][j] == D[i][s] && N[j] < N[s])
       s = j;
    Pivot(i, s); } }
  if (!Simplex(2)) return numeric_limits<DOUBLE>::infinity();
  x = VD(n);
  for (int i = 0; i < m; i++) if (B[i] < n)
    x[B[i]] = D[i][n + 1];
  return D[m][n + 1]; } };
// Two-phase simplex algorithm for solving linear programs
// of the form
//     maximize     c^T x
//     subject to   Ax <= b
//                  x >= 0
// INPUT: A -- an m x n matrix
//        b -- an m-dimensional vector
//        c -- an n-dimensional vector
//        x -- a vector where the optimal solution will be
//                stored
// OUTPUT: value of the optimal solution (infinity if
//                   unbounded above, nan if infeasible)
// To use this code, create an LPSolver object with A, b,
// and c as arguments.  Then, call Solve(x).
// #include <iostream>
// #include <iomanip>
// #include <vector>
// #include <cmath>
// #include <limits>
// using namespace std;
// int main() {
//   const int m = 4;
//   const int n = 3;
//   DOUBLE _A[m][n] = {
//     { 6, -1, 0 },
//     { -1, -5, 0 },
//     { 1, 5, 1 },
//     { -1, -5, -1 }
//   };
//   DOUBLE _b[m] = { 10, -4, 5, -5 };
//   DOUBLE _c[n] = { 1, -1, 0 };
//   VVD A(m);
//   VD b(_b, _b + m);
//   VD c(_c, _c + n);
//   for (int i = 0; i < m; i++) A[i] = VD(_A[i], _A[i] + n);
//   LPSolver solver(A, b, c);
//   VD x;
//   DOUBLE value = solver.Solve(x);
//   cerr << ”VALUE: ” << value << endl; // VALUE: 1.29032
//   cerr << ”SOLUTION:”; // SOLUTION: 1.74194 0.451613 1
//   for (size_t i = 0; i < x.size(); i++) cerr << ” ”
```

```
     << x[i];
//   cerr << endl;
//   return 0;
// }
```

MD5: `8f3e8ef20d60f94de287b4d9ff903e9e` | $\mathcal{O}(?)$

3.10 successive squaring

calculates g^L here shown for matrix mult, but can be applied in other cases

```
void mult(int a[][nos], int b[][nos], int N)
{
    int res[nos][nos] = {0};
    for(int i = 0; i < N; i++) {
        for(int j = 0; j < N; j++) {
            for(int k = 0; k < N; k++) {
                res[i][j] = (res[i][j] + a[i][k]*b[k][
                    j]) % 10000;
            }
        }
    }
    for(int i = 0; i < N; i++) {
        for(int j = 0; j < N; j++) {
            a[i][j] = res[i][j];
        }
    }
}
// res stores the result and is initialized to the
    identity matrix
int res[nos][nos] = {0};
for(int i = 0; i < N; i++) {
    for(int j = 0; j < N; j++) {
        if(i == j) res[i][j] = 1;
    }
 }
for(int i = 0; (1 << i) <= L; i++) {
    if(((1 << i) & L) == (1 << i)) {
        mult(res, g, N);
    }
    mult(g, g, N);
 }
```

MD5: `f86c0e996e5eec0aedce9308951f2ddc` | $\mathcal{O}(?)$

4 misc

4.1 Binary Search

Binary searchs for an element in a sorted array.
Input: sorted $array$ to search in, amount N of elements in $array$, element to search for a
Output: returns the index of a in $array$ or -1 if $array$ does not contain a

```
int lo = 0;
int hi = N-1;
// a might be in interval [lo,hi] while lo <= hi
while(lo <= hi) {
    int mid = (lo + hi) / 2;
    // if a > elem in mid of interval,
    // search the right subinterval
    if(array[mid] < a)
        lo = mid+1;
    // else if a < elem in mid of interval,
    // search the left subinterval
    else if(array[mid] > a)
        hi = mid-1;
    // else a is found
    else
        return mid;
 }
// array does not contain a
return -1;
```

MD5: `2049104cd8aaced6ba8de166e9bd2abe` | $\mathcal{O}(\log n)$

4.2 comparator in C++

```
bool myfunction (int i, int j) {return (i<j); }

int main() {
    vector<int> vec;
    sort(vec.begin(), vec.end(), myfunction);
    priority_queue<int, vector<int>, decltype(
        myfunction) *> pq(myfunction);
}
```

MD5: `f4beb6e197be08977fd4f74b2537ae09` | $\mathcal{O}(?)$

4.3 Comparator for a class in java

```
class bid implements Comparable<bid> {
    String person;
    BigInteger bd;

    public bid(String person, BigInteger bd) {
        this.person = person;
        this.bd = bd;
    }

    public int compareTo(bid other) {
        return this.bd.compareTo(other.bd);
    }
}
```

MD5: `2362bb9e94fe52807365ff02c48f9a15` | $\mathcal{O}(?)$

4.4 Divide and Conquer Optimization (DP)

Anwendbar, wenn T[i][j] = min (für k < j) (T[i-1][k] + C[k][j]) und A[i][j] <= A[i][j+1], wobei A[i][j] das kleinste k, dass die optimale Antwort gibt)

```
void optdp(int i, int jleft, int jright, int kleft,
    int kright)
{
    if(jleft > jright) return;
    int jmid = (jleft + jright) / 2;
    T[i][jmid] = (1LL << 62);
    int bestk = -1;
    for(int k = kleft; k <= kright && k < jmid; ++k) {
        if(T[i-1][k] + C[k+1][jmid] < T[i][jmid]) {
            T[i][jmid] = T[i-1][k] + C[k+1][jmid];
            bestk = k;
        }
```

```
    }
    optdp(i, jleft, jmid-1, kleft, bestk);
    optdp(i, jmid+1, jright, bestk, kright);
}

int main()
{
    for(int j = 1; j <= n; ++j) {
        T[1][j] = C[1][j];
    }
    for(int i = 2; i <= g; ++i) {
        optdp(i, 1, n, 1, n);
    }
    cout << T[g][n] << endl;
}
```

MD5: 7d9dc859bc5e4c541718273a12b8e764 | $\mathcal{O}(kn/logn)$

4.5 hashing pair in C++

```
struct pairhash {
public:
  template <typename T, typename U>
  std::size_t operator()(const std::pair<T, U> &x) const
  {
    return std::hash<T>()(x.first) ^ std::hash<U>()(x.second);
  }
};

int main() {
    unordered_map<pair<unsigned int, char>, double, pairhash> T;
}
```

MD5: 49bde857f5a8078349cf97308bd8144c | $\mathcal{O}(?)$

4.6 knuth-morris-pratt

finds pattern in a string

```
//---------------------------
// Returns a vector containing the zero based index of
//  the start of each match of the string K in S.
//  Matches may overlap
// source: wikipedia
//----------------------------
vector<int> KMP(string S, string K)
{
    vector<int> T(K.size() + 1, -1);
    vector<int> matches;

    if (K.size() == 0) {
        matches.push_back(0);
        return matches;
    }

    for (int i = 1; i <= K.size(); i++) {
        int pos = T[i - 1];
        while (pos != -1 && K[pos] != K[i - 1])
            pos = T[pos];
        T[i] = pos + 1;
    }

    int sp = 0;
    int kp = 0;
    while (sp < S.size()) {
        while (kp != -1 && (kp == K.size() || K[kp] != S[sp]))
            kp = T[kp];
        kp++;
        sp++;
        if (kp == K.size())
            matches.push_back(sp - K.size());
    }

    return matches;
}
```

MD5: 856843d59319d4adac8e62968cc7ccf0 | $\mathcal{O}(|K| + |S|)$

4.7 Knuth optimization (DP)

```
//s - length(size) of substring
for (int s = 0; s<=k; s++) {
    for (int i = 0; i+s<=k; i++) {
        int j = i + s;
        if (s < 2) {
            res[i][j] = 0;
            // at the start mid is equal to left border
            mid[i][j] = l;
            continue;
        }
        int mleft = mid[i][j-1];
        int mright = mid[i+1][j];
        res[l][r] = 1000000000000000000LL;
         //iterating for m in the bounds only
        for (int m = mleft; m<=mright; m++) {
            int tres = res[i][m] + res[m][j] + (x[j]-x[i]);
            if (res[i][j] > tres) {
                res[i][j] = tres;
                mid[i][j] = m;
            }
        }
    }
 }
```

MD5: c80f9b880ef8fabba88f533294ad4a8e | $\mathcal{O}(n^2)$

4.8 LongestIncreasingSubsequence

Input: array arr containing a sequence and empty array p of length $arr.length$ for storing indices of the LIS
Output: array s containing the longest increasing subsequence

```
// p[k] stores index of the predecessor of arr[k]
// in the LIS ending at arr[k]
// m[j] stores index k of smallest value arr[k]
// so there is a LIS of length j ending at arr[k]
int m[n+1];
int l = 0;
for(int i = 0; i < n; i++) {
    // bin search for the largest positive j <= l
    // with arr[m[j]] < arr[i]
    int lo = 1;
    int hi = l;
    while(lo <= hi) {
        int mid = (int) (((lo + hi) / 2.0) + 0.6);
```

```
        if(arr[m[mid]] <= arr[i])
            lo = mid+1;
        else
            hi = mid-1;
    }
    // lo is 1 greater than length of the
    // longest prefix of arr[i]
    int newL = lo;
    p[i] = m[newL-1];
    m[newL] = i;
    // if LIS found is longer than the ones
    // found before, then update l
    if(newL > l)
        l = newL;
}
// reconstruct the LIS
vector<int> s (l);
int k = m[l];
for(int i= l-1; i>= 0; i--) {
    s[i] = arr[k];
    k = p[k];
}
//s is the resulting seq
```

MD5: `8eb64842ea26475286a264c3557c355d` | $\mathcal{O}(n \log n)$

4.9 Mo's algorithm

Works for queries on intervals. Idea: Sort queries. Add and remove on borders has to work in O(1). Thus only usable when this is possible for the task.

```
// sort the querys [L,R] as follows: if L is in the
    same block (blocks have size sqrt n), sort by
    increasing R else sort by L
bool cmp(const pair<pair<int, int>, int> &i, const
    pair<pair<int, int>, int> &j) {
    if(i.first.first / BLOCK_SIZE != j.first.first /
        BLOCK_SIZE) {
        return i.first.first < j.first.first;
    }
    return i.first.second < j.first.second;
}

int main() {
    BLOCK_SIZE = static_cast<int>(sqrt(N));
    // store original index in queries
    vector<pair<pair<int, int>, int>> queries(M);
    vector<int> answers(M);
    //sort the queries into buckets
    sort(queries.begin(), queries.end(), cmp);
    //this is the essential part
    //for each querie we shift the previous borders
        one by one
    //careful analysis shows that the runtime is
        something like n*sqrtn + m*sqrt n (n elements
        and m queries)
    int mo_left = 0, mo_right = -1;
    for(int i = 0; i < M; ++i) {
        int left = queries[i].first.first;
        int right = queries[i].first.second;
        while(mo_right < right) {
            ++mo_right;
            // add can be any function as long as it
                is O(1)
            add(lmen[mo_right], lwomen[mo_right]);
        }
        while(mo_right > right) {
            // remove can be any function as long as
                it is O(1)
            remove(lmen[mo_right], lwomen[mo_right]);
            --mo_right;
        }
        while(mo_left < left) {
            remove(lmen[mo_left], lwomen[mo_left]);
            ++mo_left;
        }
        while(mo_left > left) {
            --mo_left;
            add(lmen[mo_left], lwomen[mo_left]);
        }
        answers[queries[i].second] = cur_answer;
    }
}
```

MD5: `3819261a7ee35c7d05e57ea167e0a27a` | $\mathcal{O}(?)$

4.10 Next number with n bits set

From x the smallest number greater than x with the same amount of bits set is computed. Little changes have to be made, if the calculated number has to have length less than 32 bits.
Input: number x with n bits set ($x = (1 << n) - 1$)
Output: the smallest number greater than x with n bits set

```
int nextNumber(int x) {
  //break when larger than limit here
  if(x == 0) return 0;
  int smallest = x & -x;
  int ripple = x + smallest;
  int new_smallest = ripple & -ripple;
  int ones  = ((new_smallest/smallest) >> 1) - 1;
  return ripple | ones;
}
```

MD5: `a70e3ab92156018533fa25fea2297214` | $\mathcal{O}(1)$

5 more math

5.1 Tree

Diameter: BFS from any node, then BFS from last visited node. Max dist is then the diameter. Center: Middle vertex in second step from above.

5.2 Divisability Explanation

$D \mid M \Leftrightarrow D \mid \texttt{digit\_sum}(M, k, alt)$, refer to table for values of D, k, alt.

5.3 Combinatorics

- Variations (ordered): k out of n objects (permutations for $k = n$)
  - without repetition:
    $M = \{(x_1, \ldots, x_k) : 1 \leq x_i \leq n,\ x_i \neq x_j \text{ if } i \neq j\}$,
    $|M| = \frac{n!}{(n-k)!}$
  - with repetition:
    $M = \{(x_1, \ldots, x_k) : 1 \leq x_i \leq n\}, |M| = n^k$

- Combinations (unordered): k out of n objects
  - without repetition: $M = \{(x_1, \ldots, x_n) : x_i \in \{0,1\},\ x_1 + \ldots + x_n = k\}$, $|M| = \binom{n}{k}$
  - with repetition: $M = \{(x_1, \ldots, x_n) : x_i \in \{0,1,\ldots,k\},\ x_1 + \ldots + x_n = k\}$, $|M| = \binom{n+k-1}{k}$
- Ordered partition of numbers: $x_1 + \ldots + x_k = n$ (i.e. 1+3 = 3+1 = 4 are counted as 2 solutions)
  - #Solutions for $x_i \in \mathbb{N}_0$: $\binom{n+k-1}{k-1}$
  - #Solutions for $x_i \in \mathbb{N}$: $\binom{n-1}{k-1}$
- Unordered partition of numbers: $x_1 + \ldots + x_k = n$ (i.e. 1+3 = 3+1 = 4 are counted as 1 solution)
  - #Solutions for $x_i \in \mathbb{N}$: $P_{n,k} = P_{n-k,k} + P_{n-1,k-1}$ where $P_{n,1} = P_{n,n} = 1$
- Derangements (permutations without fixed points): $!n = n! \sum_{k=0}^{n} \frac{(-1)^k}{k!} = \lfloor \frac{n!}{e} + \frac{1}{2} \rfloor$

5.4 Polynomial Interpolation

5.4.1 Theory

Problem: for $\{(x_0, y_0), \ldots, (x_n, y_n)\}$ find $p \in \Pi_n$ with $p(x_i) = y_i$ for all $i = 0, \ldots, n$.
Solution: $p(x) = \sum\limits_{i=0}^{n} \gamma_{0,i} \prod\limits_{j=0}^{i-1} (x - x_i)$ where $\gamma_{j,k} = y_j$ for $k = 0$ and $\gamma_{j,k} = \frac{\gamma_{j+1,k-1} - \gamma_{j,k-1}}{x_{j+k} - x_j}$ otherwise.
Efficient evaluation of $p(x)$: $b_n = \gamma_{0,n}$, $b_i = b_{i+1}(x - x_i) + \gamma_{0,i}$ for $i = n-1, \ldots, 0$ with $b_0 = p(x)$.

5.5 Fibonacci Sequence

5.5.1 Binet's formula

$\begin{pmatrix} f_n \\ f_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \begin{pmatrix} 0 \\ 1 \end{pmatrix} \Rightarrow f_n = \frac{1}{\sqrt{5}}(\phi^n - \tilde{\phi}^n)$ where $\phi = \frac{1+\sqrt{5}}{2}$ and $\tilde{\phi} = \frac{1-\sqrt{5}}{2}$.

5.5.2 Generalization

$g_n = \frac{1}{\sqrt{5}}(g_0(\phi^{n-1} - \tilde{\phi}^{n-1}) + g_1(\phi^n - \tilde{\phi}^n)) = g_0 f_{n-1} + g_1 f_n$ for all $g_0, g_1 \in \mathbb{N}_0$

5.5.3 Pisano Period

Both $(f_n \mod k)_{n \in \mathbb{N}_0}$ and $(g_n \mod k)_{n \in \mathbb{N}_0}$ are periodic.

5.6 Series

$\sum\limits_{i=1}^{n} i = \frac{n(n+1)}{2}, \sum\limits_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}, \sum\limits_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}$
$\sum\limits_{i=0}^{n} c^i = \frac{c^{n+1}-1}{c-1}, c \neq 1, \sum\limits_{i=0}^{\infty} c^i = \frac{1}{1-c}, \sum\limits_{i=1}^{n} c^i = \frac{c}{1-c}, |c| < 1$
$\sum\limits_{i=0}^{n} ic^i = \frac{nc^{n+2} - (n+1)c^{n+1} + c}{(c-1)^2}, c \neq 1, \sum\limits_{i=0}^{\infty} ic^i = \frac{c}{(1-c)^2}, |c| < 1$

5.7 Binomial coefficients

$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$, $\binom{n}{m}\binom{m}{k} = \binom{n}{k}\binom{n-k}{m-k}$, $\binom{m+n}{r} = \sum_{k=0}^{r} \binom{m}{k}\binom{n}{r-k}$ and in general, $n_1 + \cdots + n_p = \sum_{k_1 + \cdots + k_p = m} \binom{n_1}{k_1} \cdots \binom{n_p}{k_p}$

5.8 Catalan numbers

$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$
$C_0 = 1, C_{n+1} = \sum\limits_{k=0}^{n} C_k C_{n-k}, C_{n+1} = \frac{4n+2}{n+2} C_n$

5.9 Geometry

Area of a polygon: $A = \frac{1}{2}(x_1 y_2 - x_2 y_1 + x_2 y_3 - x_3 y_2 + \cdots + x_{n-1} y_n - x_n y_{n-1} + x_n y_1 - x_1 y_n)$

5.10 Number Theory

Chinese Remainder Theorem: There exists a number C, such that:
$C \equiv a_1 \mod n_1, \cdots, C \equiv a_k \mod n_k, \mathrm{ggt}(n_i, n_j) = 1, i \neq j$
Case $k = 2$: $m_1 n_1 + m_2 n_2 = 1$ with EEA.
Solution is $x = a_1 m_2 n_2 + a_2 m_1 n_1$.
General case: iterative application of $k = 2$
Euler's φ-Funktion: $\varphi(n) = n \prod_{p|n}(1 - \frac{1}{p}), p$ prime
$\varphi(p) = p - 1, \varphi(pq) = \varphi(p)\varphi(q), p, q$ prime
$\varphi(p^k) = p^k - p^{k-1}, p, q$ prime, $k \geq 1$
Eulers Theorem: $a^{\varphi(n)} \equiv 1 \mod n$
Fermats Theorem: $a^p \equiv a \mod p, p$ prime

5.11 Convolution

$(f * g)(n) = \sum\limits_{m=-\infty}^{\infty} f(m)g(n-m) = \sum\limits_{m=-\infty}^{\infty} f(n-m)g(m)$

5.12 DP Optimization

- Convex Hull Optimization:

$$T[i] = \min_{j<i}(T[j] + b[j] \cdot a[i])$$

  with the constraints $b[j] \geq b[j+1]$ and $a[j] \leq a[j+1]$. Solution is convex and thus the optimal j for i will always be smaller than the one for $i+1$. So we can use a pointer which we increment as long as the solution gets better. Running time is $\mathcal{O}(n)$ as the pointer visits each element no more than once.

- Divide and Conquer Optimization:

$$T[i][j] = \min_{k<j}(T[i-1][k] + C[k][j])$$

  with the constraint $A[i][j] \leq A[i][j+1]$ with $A[i][j]$ giving the smallest optimal k. Is dealt with (including code) in misc chapter above.

- Knuth Optimization:

$$T[i][j] = \min_{i<k<j}(T[i][k] + T[k][j]) + C[i][j]$$

with the constraint $A[i][j-1] \leq A[i][j] \leq A[i+1][j]$ which is apparently equal to the following two constraints:

$$C[a][c] + C[b][d] \leq C[a][d] + C[b][c],\ a \leq b \leq c \leq d$$
$$C[b][c] \leq C[a][d],\ a \leq b \leq c \leq d$$

With above constraint we get good bounds on k by going calculating T with increasing $j - i$. Also see the code in misc.

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 $b \in$ such that $b \ge s$, $\forall s \in S$.
$\inf S$	greatest $b \in$ such that $b \le s$, $\forall s \in S$.
$\liminf\limits_{n\to\infty} a_n$	$\lim\limits_{n\to\infty} \inf\{a_i \mid i \ge n, i \in \}$.
$\limsup\limits_{n\to\infty} a_n$	$\lim\limits_{n\to\infty} \sup\{a_i \mid i \ge n, i \in \}$.
$\binom{n}{k}$	Combinations: Size k subsets of a size n set.
$\begin{bmatrix} n \\ k \end{bmatrix}$	Stirling numbers (1st kind): Arrangements of an n element set into k cycles.
$\begin{Bmatrix} n \\ k \end{Bmatrix}$	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

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

In general:

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

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

Geometric series:

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

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

Harmonic series:

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

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

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

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

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

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

10. $\displaystyle\binom{n}{k} = (-1)^k\binom{k-n-1}{k}$, **11.** $\displaystyle\begin{Bmatrix} n \\ 1 \end{Bmatrix} = \begin{Bmatrix} n \\ n \end{Bmatrix} = 1$,

12. $\displaystyle\begin{Bmatrix} n \\ 2 \end{Bmatrix} = 2^{n-1} - 1$, **13.** $\displaystyle\begin{Bmatrix} n \\ k \end{Bmatrix} = k\begin{Bmatrix} n-1 \\ k \end{Bmatrix} + \begin{Bmatrix} n-1 \\ k-1 \end{Bmatrix}$,

14. $\displaystyle\begin{bmatrix} n \\ 1 \end{bmatrix} = (n-1)!$, **15.** $\displaystyle\begin{bmatrix} n \\ 2 \end{bmatrix} = (n-1)!H_{n-1}$, **16.** $\displaystyle\begin{bmatrix} n \\ n \end{bmatrix} = 1$, **17.** $\displaystyle\begin{bmatrix} n \\ k \end{bmatrix} \ge \begin{Bmatrix} n \\ k \end{Bmatrix}$,

18. $\displaystyle\begin{bmatrix} n \\ k \end{bmatrix} = (n-1)\begin{bmatrix} n-1 \\ k \end{bmatrix} + \begin{bmatrix} n-1 \\ k-1 \end{bmatrix}$, **19.** $\displaystyle\begin{Bmatrix} n \\ n-1 \end{Bmatrix} = \begin{bmatrix} n \\ n-1 \end{bmatrix} = \binom{n}{2}$, **20.** $\displaystyle\sum_{k=0}^{n}\begin{bmatrix} n \\ k \end{bmatrix} = n!$, **21.** $\displaystyle C_n = \frac{1}{n+1}\binom{2n}{n}$,

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

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

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

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

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

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

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"
$$1\big(T(n) - 3T(n/2) = n\big)$$
$$3\big(T(n/2) - 3T(n/4) = n/2\big)$$
$$\vdots \qquad \vdots \qquad \vdots$$
$$3^{\log_2 n - 1}\big(T(2) - 3T(1) = 2\big)$$

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
$$n \sum_{i=0}^{m-1} c^i = n\left(\frac{c^m - 1}{c - 1}\right)$$
$$= 2n(c^{\log_2 n} - 1)$$
$$= 2n(c^{(k-1)\log_c n} - 1)$$
$$= 2n^k - 2n,$$

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
$$T_{i+1} - T_i = 1 + \sum_{j=0}^{i} T_j - 1 - \sum_{j=0}^{i-1} T_j$$
$$= T_i.$$

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:
$$G(x) = x\left(\frac{2}{1-2x} - \frac{1}{1-x}\right)$$
$$= x\left(2\sum_{i \ge 0} 2^i x^i - \sum_{i \ge 0} x^i\right)$$
$$= \sum_{i \ge 0} (2^{i+1} - 1)x^{i+1}.$$

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, ...

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

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:

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

Moment inequalities:

$$\Pr\big[|X| \geq \lambda\,\mathrm{E}[X]\big] \leq \frac{1}{\lambda},$$

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

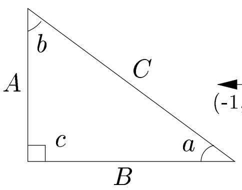
Geometric distribution:

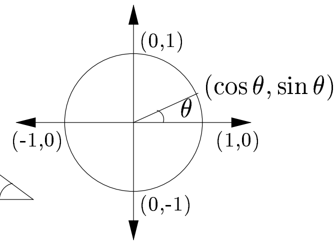
$$\Pr[X = k] = pq^{k-1}, \qquad q = 1 - p,$$

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

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:

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

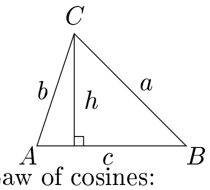
Identities:

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

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:

$$C \equiv r_1 \bmod m_1$$
$$\vdots \quad \vdots \quad \vdots$$
$$C \equiv r_n \bmod m_n$$

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

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

Prime numbers:

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

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

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

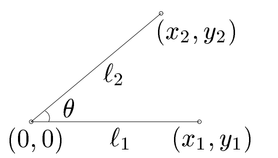
Distance formula, L_p and L_∞ metric:

$$\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2},$$
$$\left[|x_1 - x_0|^p + |y_1 - y_0|^p\right]^{1/p},$$
$$\lim_{p \to \infty} \left[|x_1 - x_0|^p + |y_1 - y_0|^p\right]^{1/p}.$$

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

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

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

Theoretical Computer Science Cheat Sheet

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. $\int \frac{dx}{x\sqrt{x^2-a^2}} = \frac{1}{a}\arccos\frac{a}{|x|}, \quad a>0,$ **63.** $\int \frac{dx}{x^2\sqrt{x^2\pm a^2}} = \mp\frac{\sqrt{x^2\pm a^2}}{a^2x},$

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

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

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

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

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

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

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

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

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

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

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

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

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:

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

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

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

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

Rising Factorial Powers:

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

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

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

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

Conversion:

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

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

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

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

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

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:

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + x^4 + \cdots = \sum_{i=0}^{\infty} x^i,$$

$$\frac{1}{1-cx} = 1 + cx + c^2x^2 + c^3x^3 + \cdots = \sum_{i=0}^{\infty} c^i x^i,$$

$$\frac{1}{1-x^n} = 1 + x^n + x^{2n} + x^{3n} + \cdots = \sum_{i=0}^{\infty} x^{ni},$$

$$\frac{x}{(1-x)^2} = x + 2x^2 + 3x^3 + 4x^4 + \cdots = \sum_{i=0}^{\infty} i x^i,$$

$$x^k \frac{d^n}{dx^n}\left(\frac{1}{1-x}\right) = x + 2^n x^2 + 3^n x^3 + 4^n x^4 + \cdots = \sum_{i=0}^{\infty} i^n x^i,$$

$$e^x = 1 + x + \tfrac{1}{2}x^2 + \tfrac{1}{6}x^3 + \cdots = \sum_{i=0}^{\infty} \frac{x^i}{i!},$$

$$\ln(1+x) = x - \tfrac{1}{2}x^2 + \tfrac{1}{3}x^3 - \tfrac{1}{4}x^4 - \cdots = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^i}{i},$$

$$\ln\frac{1}{1-x} = x + \tfrac{1}{2}x^2 + \tfrac{1}{3}x^3 + \tfrac{1}{4}x^4 + \cdots = \sum_{i=1}^{\infty} \frac{x^i}{i},$$

$$\sin x = x - \tfrac{1}{3!}x^3 + \tfrac{1}{5!}x^5 - \tfrac{1}{7!}x^7 + \cdots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)!},$$

$$\cos x = 1 - \tfrac{1}{2!}x^2 + \tfrac{1}{4!}x^4 - \tfrac{1}{6!}x^6 + \cdots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!},$$

$$\tan^{-1} x = x - \tfrac{1}{3}x^3 + \tfrac{1}{5}x^5 - \tfrac{1}{7}x^7 + \cdots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)},$$

$$(1+x)^n = 1 + nx + \tfrac{n(n-1)}{2}x^2 + \cdots = \sum_{i=0}^{\infty} \binom{n}{i} x^i,$$

$$\frac{1}{(1-x)^{n+1}} = 1 + (n+1)x + \tbinom{n+2}{2}x^2 + \cdots = \sum_{i=0}^{\infty} \binom{i+n}{i} x^i,$$

$$\frac{x}{e^x - 1} = 1 - \tfrac{1}{2}x + \tfrac{1}{12}x^2 - \tfrac{1}{720}x^4 + \cdots = \sum_{i=0}^{\infty} \frac{B_i x^i}{i!},$$

$$\frac{1}{2x}(1 - \sqrt{1-4x}) = 1 + x + 2x^2 + 5x^3 + \cdots = \sum_{i=0}^{\infty} \frac{1}{i+1}\binom{2i}{i} x^i,$$

$$\frac{1}{\sqrt{1-4x}} = 1 + x + 2x^2 + 6x^3 + \cdots = \sum_{i=0}^{\infty} \binom{2i}{i} x^i,$$

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

$$\frac{1}{1-x}\ln\frac{1}{1-x} = x + \tfrac{3}{2}x^2 + \tfrac{11}{6}x^3 + \tfrac{25}{12}x^4 + \cdots = \sum_{i=1}^{\infty} H_i x^i,$$

$$\frac{1}{2}\left(\ln\frac{1}{1-x}\right)^2 = \tfrac{1}{2}x^2 + \tfrac{3}{4}x^3 + \tfrac{11}{24}x^4 + \cdots = \sum_{i=2}^{\infty} \frac{H_{i-1}x^i}{i},$$

$$\frac{x}{1-x-x^2} = x + x^2 + 2x^3 + 3x^4 + \cdots = \sum_{i=0}^{\infty} F_i x^i,$$

$$\frac{F_n x}{1 - (F_{n-1} + F_{n+1})x - (-1)^n x^2} = F_n x + F_{2n}x^2 + F_{3n}x^3 + \cdots = \sum_{i=0}^{\infty} F_{ni} x^i.$$

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:

$$\alpha A(x) + \beta B(x) = \sum_{i=0}^{\infty} (\alpha a_i + \beta b_i) x^i,$$

$$x^k A(x) = \sum_{i=k}^{\infty} a_{i-k} x^i,$$

$$\frac{A(x) - \sum_{i=0}^{k-1} a_i x^i}{x^k} = \sum_{i=0}^{\infty} a_{i+k} x^i,$$

$$A(cx) = \sum_{i=0}^{\infty} c^i a_i x^i,$$

$$A'(x) = \sum_{i=0}^{\infty} (i+1) a_{i+1} x^i,$$

$$xA'(x) = \sum_{i=1}^{\infty} i a_i x^i,$$

$$\int A(x)\,dx = \sum_{i=1}^{\infty} \frac{a_{i-1}}{i} x^i,$$

$$\frac{A(x) + A(-x)}{2} = \sum_{i=0}^{\infty} a_{2i} x^{2i},$$

$$\frac{A(x) - A(-x)}{2} = \sum_{i=0}^{\infty} a_{2i+1} x^{2i+1}.$$

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

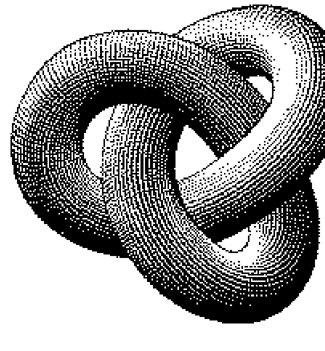
Series

Expansions:

$$
\begin{aligned}
\frac{1}{(1-x)^{n+1}}\ln\frac{1}{1-x} &= \sum_{i=0}^{\infty}(H_{n+i}-H_n)\binom{n+i}{i}x^i, & \left(\frac{1}{x}\right)^{\overline{-n}} &= \sum_{i=0}^{\infty}\left\{ {i \atop n} \right\}x^i,\\
x^{\overline{n}} &= \sum_{i=0}^{\infty}\left[{n \atop i} \right]x^i, & (e^x-1)^n &= \sum_{i=0}^{\infty}\left\{ {i \atop n} \right\}\frac{n!x^i}{i!},\\
\left(\ln\frac{1}{1-x}\right)^n &= \sum_{i=0}^{\infty}\left[{i \atop n} \right]\frac{n!x^i}{i!}, & x\cot x &= \sum_{i=0}^{\infty}\frac{(-4)^iB_{2i}x^{2i}}{(2i)!},\\
\tan x &= \sum_{i=1}^{\infty}(-1)^{i-1}\frac{2^{2i}(2^{2i}-1)B_{2i}x^{2i-1}}{(2i)!}, & \zeta(x) &= \sum_{i=1}^{\infty}\frac{1}{i^x},\\
\frac{1}{\zeta(x)} &= \sum_{i=1}^{\infty}\frac{\mu(i)}{i^x}, & \frac{\zeta(x-1)}{\zeta(x)} &= \sum_{i=1}^{\infty}\frac{\phi(i)}{i^x},
\end{aligned}
$$

$$
\begin{aligned}
\zeta(x) &= \prod_p \frac{1}{1-p^{-x}},\\
\zeta^2(x) &= \sum_{i=1}^{\infty}\frac{d(i)}{x^i} \quad \text{where } d(n)=\textstyle\sum_{d|n}1,\\
\zeta(x)\zeta(x-1) &= \sum_{i=1}^{\infty}\frac{S(i)}{x^i} \quad \text{where } S(n)=\textstyle\sum_{d|n}d,\\
\zeta(2n) &= \frac{2^{2n-1}|B_{2n}|}{(2n)!}\pi^{2n}, \quad n\in\mathbb{N},\\
\frac{x}{\sin x} &= \sum_{i=0}^{\infty}(-1)^{i-1}\frac{(4^i-2)B_{2i}x^{2i}}{(2i)!},\\
\left(\frac{1-\sqrt{1-4x}}{2x}\right)^n &= \sum_{i=0}^{\infty}\frac{n(2i+n-1)!}{i!(n+i)!}x^i,\\
e^x\sin x &= \sum_{i=1}^{\infty}\frac{2^{i/2}\sin\frac{i\pi}{4}}{i!}x^i,\\
\sqrt{\frac{1-\sqrt{1-x}}{x}} &= \sum_{i=0}^{\infty}\frac{(4i)!}{16^i\sqrt{2}(2i)!(2i+1)!}x^i,\\
\left(\frac{\arcsin x}{x}\right)^2 &= \sum_{i=0}^{\infty}\frac{4^ii!^2}{(i+1)(2i+1)!}x^{2i}.
\end{aligned}
$$

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

$$
\begin{aligned}
\int_a^b \big(G(x)+H(x)\big)\,dF(x) &= \int_a^b G(x)\,dF(x) + \int_a^b H(x)\,dF(x),\\
\int_a^b G(x)\,d\big(F(x)+H(x)\big) &= \int_a^b G(x)\,dF(x) + \int_a^b G(x)\,dH(x),\\
\int_a^b c\cdot G(x)\,dF(x) &= \int_a^b G(x)\,d\big(c\cdot F(x)\big) = c\int_a^b G(x)\,dF(x),\\
\int_a^b G(x)\,dF(x) &= G(b)F(b) - G(a)F(a) - \int_a^b F(x)\,dG(x).
\end{aligned}
$$

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

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 $k_m \ge 2$.

Fibonacci Numbers

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

Definitions:

$$F_i = F_{i-1}+F_{i-2}, \quad F_0 = F_1 = 1,$$
$$F_{-i} = (-1)^{i-1}F_i,$$
$$F_i = \tfrac{1}{\sqrt{5}}\left(\phi^i - \hat{\phi}^i\right),$$

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

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

Additive rule:

$$F_{n+k} = F_kF_{n+1} + F_{k-1}F_n,$$
$$F_{2n} = F_nF_{n+1} + F_{n-1}F_n.$$

Calculation by matrices:

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

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