

Contents

1	Startup templates	2
1.1	template	2
1.2	gvimrc	2
2	Graph Algorithms	3
2.1	Dinic Max-Flow	3
2.2	Hungary Algo	3
2.3	Min Cut	4
2.4	Bridges	4
2.5	Cut Vertices	5
2.6	Min-Cost Max-Flow	5
2.7	Strongly Connected Components	6
2.8	2-SAT	7
3	Linear Algebra	8
3.1	Gauss Elimination	8
3.2	Fast-Fourier Transform	8
3.3	Simplex	9
4	String Algorithms	10
4.1	Suffix Array	10
4.2	Suffix Tree from Suffix Array	10
4.3	Z-function	11
4.4	Suffix Automata	12
4.5	Palindromes	12
4.6	Lyndon decomposition & Duval	12
5	Modular	14
6	Data Structures	16
6.1	Treap	16
6.2	Implicit Treap	16
6.3	Fenwick Tree	16
6.4	Stack with Min	17
6.5	Queue with Min	17
7	Geometry	18
7.1	Point 2D	18
7.2	Point 3D	19

1 Startup templates

1.1 template

```

1 #include <vector>
2 #include <list>
3 #include <map>
4 #include <set>
5 #include <deque>
6 #include <stack>
7 #include <bitset>
8 #include <algorithm>
9 #include <functional>
10 #include <numeric>
11 #include <utility>
12 #include <sstream>
13 #include <iostream>
14 #include <cstdio>
15 #include <cmath>
16 #include <cstdlib>
17 #include <cstring>
18 #include <cassert>
19
20 using namespace std;
21
22 typedef long long ll;
23 typedef pair<int, int> pii;
24
25 template<typename T> int size(T& a){ return (int) a.size(); }
26 template<typename T> T sqr(T a){ return a * a; }
27
28 #define _(a, b) memset((a), (b), sizeof(a))
29 #define fs first
30 #define sc second
31 #define pb push_back
32 #define mp make_pair
33 #define all(a) a.begin(), a.end()
34 #define REP(i, a, b) for(int i = (a); i < (b); ++i)
35 #define REPD(i, a, b) for(int i = (b) - 1; i >= a; --i)
36 #define ve vector

```

1.2 gvimrc

```

1 set autoread
2 set autoindent
3 set autochdir
4 set cindent
5 set number
6 syntax on
7 set shiftwidth =4
8 set tabstop =4
9 colorscheme desert
10 set gfn =Monospace\ 12

```

2 Graph Algorithms

2.1 Dinic Max-Flow

```

1 // need: graph( head, nxt, to, capa, flow ), dist, q
2 bool bfs(int src, int dest) {
3     (dist, -1);
4     dist[src] = 0;
5     int H = 0;
6     q[H++] = src;
7     REP(i, 0, H) {
8         int cur = q[i];
9         for(int e = head[cur]; e != -1; e = nxt[e]) {
10             if (capa[e] > flow[e] && dist[to[e]] == -1) {
11                 dist[to[e]] = dist[cur] + 1;
12                 q[H++] = to[e];
13             }
14         }
15     }
16     return dist[dest] >= 0;
17 }
18
19 int dfs(int cur, int curflow) {
20     if (cur == dest) return curflow;
21     int d;
22     for(int& e = work[cur]; e != -1; e = nxt[e]) {
23         if (capa[e] > flow[e] && (dist[to[e]] == dist[cur] + 1) &&
24             (d = dfs(to[e], min(curflow, capa[e] - flow[e]))) {
25             flow[e] += d;
26             flow[e ^ 1] -= d;
27             return d;
28         }
29     }
30     return 0;
31 }
32
33 int dinic() {
34     int res = 0;
35     while (bfs(src, des)) {
36         int d;
37         memcpy(work, head, sizeof(head));
38         while (true) {
39             d = dfs(src, INF);
40             if (d == 0) break;
41             res += d;
42         }
43     }
44     return res;
45 }

```

2.2 Hungary Algo

```

1 // need: a[n][m], all indices start with 1
2 vector<int> u (n+1), v (m+1), p (m+1), way (m+1);
3 for (int i=1; i<=n; ++i) {
4     p[0] = i;
5     int j0 = 0;
6     vector<int> minv (m+1, INF);
7     vector<char> used (m+1, false);
8     do {
9         used[j0] = true;
10        int i0 = p[j0], delta = INF, j1;
11        for (int j=1; j<=m; ++j)
12            if (!used[j]) {
13                int cur = a[i0][j] - u[i0] - v[j];
14                if (cur < minv[j])
15                    minv[j] = cur, way[j] = j0;
16                if (minv[j] < delta)
17                    delta = minv[j], j1 = j;
18            }
19        for (int j=0; j<=m; ++j)

```

```

20     if (used[j])
21         u[p[j]] += delta, v[j] -= delta;
22     else
23         minv[j] -= delta;
24     j0 = j1;
25 } while (p[j0] != 0);
26 do {
27     int j1 = way[j0];
28     p[j0] = p[j1];
29     j0 = j1;
30 } while (j0);
31 }
32 // restore ans[] — selected column for each row
33 for (int j=1; j<=m; ++j)
34     ans[p[j]] = j;
35 // cost
36 int cost = -v[0];

```

2.3 Min Cut

```

1 pair<int, ve<int>> > GetMinCut(ve< ve<int>> > &weights) {
2     int N = weights.size();
3     ve<int> used(N), cut, best_cut;
4     int best_weight = -1;
5
6     REPD(phase, 0, N) {
7         ve<int> w = weights[0];
8         ve<int> added = used;
9         int prev, last = 0;
10        REP(i, 0, phase) {
11            prev = last;
12            last = -1;
13            REP(j, 1, N)
14                if (!added[j] && (last == -1 || w[j] > w[last])) last = j;
15            if (i == phase-1) {
16                REP(j, 0, N) weights[prev][j] += weights[last][j];
17                REP(j, 0, N) weights[j][prev] = weights[prev][j];
18                used[last] = true;
19                cut.pb(last);
20                if (best_weight == -1 || w[last] < best_weight) {
21                    best_cut = cut;
22                    best_weight = w[last];
23                }
24            } else {
25                REP(j, 0, N)
26                    w[j] += weights[last][j];
27                added[last] = true;
28            }
29        }
30    }
31    return mp(best_weight, best_cut);
32 }

```

2.4 Bridges

```

1 // need: graph( head, to, nxt ), used, tin, fup, timer
2 void dfs (int v, int par = -1) {
3     used[v] = true;
4     tin[v] = fup[v] = timer++;
5     for(int e = head[v]; e != -1; e = nxt[e]) {
6         int u = to[e];
7         if (u == par) continue;
8         if (used[u])
9             fup[v] = min (fup[v], tin[u]);
10        else {
11            dfs (u, v);
12            fup[v] = min (fup[v], fup[u]);
13            if (fup[u] > tin[v])
14                IS_BRIDGE(v, u);
15        }

```

```

16 }
17 }
18
19 void find_bridges() {
20     timer = 0;
21     _ (used, 0)
22     REP(i, 0, N)
23         if (!used[i]) dfs (i);
24 }

```

2.5 Cut Vertices

```

1 // need: graph (head, to, nxt), tin, fup, used, timer
2 void dfs (int v, int par = -1) {
3     used[v] = true;
4     tin[v] = fup[v] = timer++;
5     int children = 0;
6     for(int e = head[v]; e != -1; e = nxt[e]) {
7         int u = to[e];
8         if (u == par) continue;
9         if (used[u])
10             fup[v] = min (fup[v], tin[u]);
11         else {
12             dfs (u, v);
13             fup[v] = min (fup[v], fup[u]);
14             if (fup[u] >= tin[v] && p != -1)
15                 IS_CUTPOINT(v);
16             ++children;
17         }
18     }
19     if (p == -1 && children > 1)
20         IS_CUTPOINT(v);
21 }
22
23 int main() {
24     timer = 0;
25     _ (used, 0);
26     dfs (0);
27 }

```

2.6 Min-Cost Max-Flow

```

1 // need: graph (head, nxt, to, from, capa, cost, flow)
2 // pi, dist, prve
3 void updatePotentials() {
4     memcpy(pi, dist, sizeof(int) * N);
5 }
6
7 bool fordBellman(int src, int dst) {
8     REP(i, 0, N) dist[i] = INF;
9     dist[src] = 0;
10    bool changed;
11
12    REP(phase, 0, N) {
13        changed = false;
14        REP(v, 0, N) {
15            if (dist[v] == INF) continue;
16            for(int e = head[v]; e != -1; e = nxt[e]) {
17                int u = to[e];
18                if (capa[e] > flow[e] && dist[u] > dist[v] + cost[e]) {
19                    dist[u] = dist[v] + cost[e];
20                    prve[u] = e;
21                    changed = true;
22                }
23            }
24        }
25        if (!changed) break;
26    }
27    return !changed;
28 }

```

```

29
30 set< pii > q;
31 bool dijkstra(int src, int dst) {
32     REP(i, 0, N) dist[i] = INF;
33     dist[src] = 0;
34     q.insert( mp(0, 0) );
35
36     while(size(q)) {
37         pii tmp = (*q.begin());
38         int v = tmp.sc, d = tmp.fs;
39         q.erase(q.begin());
40         if(d != dist[v]) continue;
41
42         for(int e = head[v]; e != -1; e = nxt[e]) {
43             int u = to[e];
44             if(capa[e] > flow[e] && dist[u] > dist[v] + cost[e] - pi[v] + pi[u]) {
45                 dist[u] = dist[v] + cost[e] - pi[v] + pi[u];
46                 prve[u] = e;
47                 q.insert( mp(dist[u], u) );
48             }
49         }
50     }
51     return dist[dst] != INF;
52 }
53
54 pii minCostMaxFlow(int src, int dst) {
55     if(!fordBellman(src, dst)) return mp(0, 0);
56     int sumFlow = 0, sumCost = 0;
57
58     do {
59         int curFlow = INF, curCost = 0;
60         int cur = dst;
61         while(cur != src) {
62             int e = prve[cur];
63             curFlow = min(curFlow, capa[e] - flow[e]);
64             curCost += cost[e];
65             cur = from[e];
66         }
67         cur = dst;
68         while(cur != src) {
69             int e = prve[cur];
70             flow[e] += curFlow;
71             flow[e ^ 1] -= curFlow;
72             cur = from[e];
73         }
74         sumCost += curFlow * curCost;
75         updatePotentials();
76     } while(dijkstra(src, dst));
77
78     return mp(sumFlow, sumCost);
79 }

```

2.7 Strongly Connected Components

```

1 // need: graph (head, to, nxt), reverse graph (rhead, rto, rnxt)
2 // used, compID, order
3 void dfs1(int v) {
4     used[v] = true;
5     for(int e = head[v]; e != -1; e = nxt[e]) {
6         if(!used[to[e]]) dfs1(to[e]);
7     }
8     order.pb(v);
9 }
10 void dfs2(int v, int id) {
11     used[v] = true;
12     compID[v] = id;
13     for(int e = rhead[v]; e != -1; e = rnxt[e]) {
14         if(!used[rto[e]]) dfs2(rto[e]);
15     }
16 }
17 void main() {
18     _(used, false);

```

```

19 REP(i, 0, N)
20     if (!used[i]) dfs1(i);
21     _ (used, false);
22     int id = 0;
23     REPD(i, 0, N) {
24         int v = order[i];
25         if (!used[v]) dfs2(v, id++);
26     }
27 }

```

2.8 2-SAT

Problem: $(a \vee c) \& (a \vee !b) \& \dots$

Edges: $(a \vee b)$ is equivalent to $(!a \Rightarrow b) \vee (!b \Rightarrow a)$

Solution: there is no solution iff for some x $compID[x] = compID[!x]$, else see code below

```

1 // need: graph, scc
2 int main() {
3     _ (used, false);
4     REP(i, 0, N)
5         if (!used[i]) dfs1(i);
6
7     _ (compID, -1);
8     int id = 0;
9     REPD(i, 0, N) {
10         int v = order[i];
11         if (comp[v] == -1) dfs2(v, id++);
12     }
13
14     REP(i, 0, N)
15         if (compID[i] == compID[i ^ 1]) {
16             puts ("NO SOLUTION");
17             return 0;
18         }
19     REP(i, 0, N) {
20         int ans = comp[i] > comp[i ^ 1] ? i : i ^ 1;
21         printf ("%d ", ans);
22     }
23 }

```

3 Linear Algebra

3.1 Gauss Elimination

```

1 // Ax = B. RETURN: determinant, A -> A^(-1), B -> solution
2 T GaussJordan(VVT &a, VT &b) {
3     const int n = a.size();
4     ve<int> irow(n), icol(n), ipiv(n);
5     T det = 1;
6
7     REP(i, 0, n) {
8         int pj = -1, pk = -1;
9         REP(j, 0, n) if (!ipiv[j])
10             REP(k, 0, n) if (!ipiv[k])
11                 if (pj == -1 || fabs(a[j][k]) > fabs(a[pj][pk])) { pj = j; pk = k; }
12             if (fabs(a[pj][pk]) < EPS) { cerr << "Matrix is singular." << endl; exit(0); }
13             ipiv[pk]++;
14             swap(a[pj], a[pk]);
15             swap(b[pj], b[pk]);
16             if (pj != pk) det *= -1;
17             irow[i] = pj;
18             icol[i] = pk;
19
20             T c = 1.0 / a[pk][pk];
21             det *= a[pk][pk];
22             a[pk][pk] = 1.0;
23             REP(p, 0, n) a[pk][p] *= c;
24             b[pk] *= c;
25             REP(p, 0, n) if (p != pk) {
26                 c = a[p][pk];
27                 a[p][pk] = 0;
28                 REP(q, 0, n) a[p][q] -= a[pk][q] * c;
29                 b[p] -= b[pk] * c;
30             }
31     }
32
33     REPD(p, 0, n) if (irow[p] != icol[p]) {
34         REP(k, 0, n) swap(a[k][irow[p]], a[k][icol[p]]);
35     }
36
37     return det;
38 }

```

3.2 Fast-Fourier Transform

```

1 typedef complex<double> base;
2
3 void fft (vector<base> &a, bool invert) {
4     int n = (int) a.size();
5
6     for (int i = 1, j = 0; i < n; i++) {
7         int bit = n >> 1;
8         for (; j >= bit; bit >>= 1)
9             j -= bit;
10        j += bit;
11        if (i < j) swap (a[i], a[j]);
12    }
13
14    for (int len = 2; len <= n; len <= 1) {
15        double ang = 2 * PI/len * (invert ? -1 : 1);
16        base wlen (cos(ang), sin(ang));
17        for (int i = 0; i < n; i += len) {
18            base w (1);
19            for (int j = 0; j < len/2; j++) {
20                base u = a[i+j], v = a[i+j+len/2] * w;
21                a[i+j] = u + v;
22                a[i+j+len/2] = u - v;
23                w *= wlen;
24            }
25        }
26    }
27 }

```



```

27     if (invert)
28         REP(i, 0, n) a[i] /= n;
29 }

```

3.3 Simplex

```

1  // maximize c^T x
2  // Ax <= b
3  // x >= 0
4
5  struct LPSolver {
6      int m, n;
7      ve<int> B, N;
8      ve< ve<double>> > D;
9
10     LPSolver(const ve< ve<double>> &A, const ve<double> &b, const ve<double> &c) :
11         m(b.size()), n(c.size()), N(n+1), B(m), D(m+2, ve<double>(n+2)) {
12         REP(i, 0, m) REP(j, 0, n) D[i][j] = A[i][j];
13         REP(i, 0, m) { B[i] = n+i; D[i][n] = -1; D[i][n+1] = b[i]; }
14         REP(j, 0, n) { N[j] = j; D[m][j] = -c[j]; }
15         N[n] = -1; D[m+1][n] = 1;
16     }
17
18     void Pivot(int r, int s) {
19         REP(i, 0, m+2) if (i != r)
20             REP(j, 0, n+2) if (j != s)
21                 D[i][j] -= D[r][j] * D[i][s] / D[r][s];
22         REP(j, 0, n+2) if (j != s) D[r][j] /= D[r][s];
23         REP(i, 0, n+2) if (i != r) D[i][s] /= -D[r][s];
24         D[r][s] = 1.0 / D[r][s];
25         swap(B[r], N[s]);
26     }
27
28     bool Simplex(int phase) {
29         int x = phase == 1 ? m+1 : m;
30         while (true) {
31             int s = -1;
32             REP(j, 0, n+1) {
33                 if (phase == 2 && N[j] == -1) continue;
34                 if (s == -1 || D[x][j] < D[x][s] || D[x][j] == D[x][s] && N[j] < N[s]) s = j;
35             }
36             if (D[x][s] >= -EPS) return true;
37             int r = -1;
38             REP(i, 0, m) {
39                 if (D[i][s] <= 0) continue;
40                 if (r == -1 || D[i][n+1] / D[i][s] < D[r][n+1] / D[r][s] ||
41                     D[i][n+1] / D[i][s] == D[r][n+1] / D[r][s] && B[i] < B[r]) r = i;
42             }
43             if (r == -1) return false;
44             Pivot(r, s);
45         }
46     }
47
48     double Solve(ve<double> &x) {
49         int r = 0;
50         REP(i, 1, m) if (D[i][n+1] < D[r][n+1]) r = i;
51         if (D[r][n+1] <= -EPS) {
52             Pivot(r, n);
53             if (!Simplex(1) || D[m+1][n+1] < -EPS) return numeric_limits<double>::infinity();
54             REP(i, 0, m) if (B[i] == -1) {
55                 int s = -1;
56                 REP(j, 0, n+1) {
57                     if (s == -1 || D[i][j] < D[i][s] || D[i][j] == D[i][s] && N[j] < N[s]) s = j;
58                 }
59                 Pivot(i, s);
60             }
61             if (!Simplex(2)) return numeric_limits<double>::infinity();
62             x = ve<double>(n);
63             REP(i, 0, m) if (B[i] < n) x[B[i]] = D[i][n+1];
64             return D[m][n+1];
65         }
66     };

```

4 String Algorithms

4.1 Suffix Array

```

1 struct entry {
2     int nr[2], p;
3 } L[MAXN], tmp[MAXN];
4 #define eq(a, b) ((a).nr[0] == (b).nr[0] && (a).nr[1] == (b).nr[1])
5 int cnt[MAXN], p[2][MAXN];
6
7 void radixPass(entry * a, int N, int pass, int K, entry * b) {
8     memset(cnt, 0, (K + 1) * sizeof(int));
9     REP(i, 0, N) cnt[a[i].nr[pass]]++;
10    int sum = 0;
11    REP(i, 0, K + 1) {
12        sum += cnt[i];
13        cnt[i] = sum - cnt[i];
14    }
15    REP(i, 0, N) b[cnt[a[i].nr[pass]]++] = a[i];
16 }
17
18 void makeSA(int * s, int N, int * suftab, int * isuftab) {
19     REP(i, 0, N) p[0][i] = s[i];
20     int k = 200;
21     for(int step = 1, len = 1; ; step ^= 1, len <= 1) {
22         for(int i = 0, j = len; i < N; i++, j++) {
23             L[i].nr[0] = p[step ^ 1][i];
24             L[i].nr[1] = j < N ? p[step ^ 1][j] : 0;
25             L[i].p = i;
26         }
27         radixPass(L, N, 1, k, tmp);
28         radixPass(tmp, N, 0, k, L);
29         k = 1;
30         REP(i, 0, N) {
31             p[step][L[i].p] = i > 0 && eq(L[i], L[i - 1]) ?
32             p[step][L[i - 1].p] : k++;
33         }
34         if(k > N) break;
35     }
36     REP(i, 0, N) suftab[i] = L[i].p;
37     REP(i, 0, N) isuftab[suftab[i]] = i;
38 }
39
40 void makeLCP(int * s, int * suftab, int * isuftab, int N, int * lcptab) {
41     int cur = 0;
42     REP(i, 0, N) {
43         if(isuftab[i] == 0) continue;
44         int ii = i + cur, jj = suftab[isuftab[i] - 1] + cur;
45         while(ii < N && jj < N && s[ii] == s[jj]) ii++, jj++, cur++;
46         lcptab[isuftab[i]] = cur--;
47         if(cur < 0) cur = 0;
48     }
49 }

```

4.2 Suffix Tree from Suffix Array

```

1 struct Seg {
2     int lb, rb, lcp;
3     vector<Seg*> childList;
4     void init(int l, int i, int j) {
5         lb = i; rb = j; lcp = l;
6     }
7     void add(Seg * son) {
8         childList.pb(son);
9     }
10 };
11 typedef Seg* pSeg;
12
13 struct Stack {
14     pSeg segs[MAXN << 1];
15     int size;

```

```

16 void push(pSeg seg) {
17     segs[size++] = seg;
18 }
19 pSeg pop() {
20     return segs[--size];
21 }
22 pSeg top() {
23     return segs[size - 1];
24 }
25 } stack;
26
27 pSeg top() { return stack.top(); }
28 void push(pSeg seg) { stack.push(seg); }
29 pSeg pop() { return stack.pop(); }
30
31 pSeg init(int lcp, int lb, int rb) {
32     pSeg ret = new Seg;
33     ret->init(lcp, lb, rb);
34     return ret;
35 }
36
37 pSeg makeTree() {
38     stack.size = 0;
39     pSeg lastInterval = NULL;
40     stack.push(init(0, 0, -1));
41     REP(i, 1, N) {
42         int lb = i - 1;
43         pSeg singleton = init(N - suftab[i - 1] - 1, i - 1, i - 1);
44         //process(singleton);
45
46         while(lcptab[i] < top()->lcp) {
47             if(singleton != NULL) {
48                 top()->add(singleton);
49                 singleton = NULL;
50             }
51             top()->rb = i - 1;
52             lastInterval = pop();
53             //process(lastInterval);
54             lb = lastInterval->lb;
55             if(lcptab[i] <= top()->lcp) {
56                 top()->add(lastInterval);
57                 lastInterval = NULL;
58             }
59         }
60         if(lcptab[i] > top()->lcp) {
61             if(lastInterval != NULL) {
62                 pSeg seg = init(lcptab[i], lb, -1);
63                 seg->add(lastInterval);
64                 push(seg);
65                 lastInterval = NULL;
66             } else push(init(lcptab[i], lb, -1));
67         }
68         if(singleton != NULL) {
69             top()->add(singleton);
70         }
71     }
72     assert(stack.size == 1);
73     //process(top());
74     return top();
75 }

```

4.3 Z-function

```

1 ve<int> z_function (string s) {
2     int n = (int) s.length();
3     ve<int> z (n);
4     for (int i=1, l=0, r=0; i<n; ++i) {
5         if (i <= r)
6             z[i] = min (r-i+1, z[i-l]);
7         while (i+z[i] < n && s[z[i]] == s[i+z[i]])
8             ++z[i];
9         if (i+z[i]-1 > r)

```

```

10     l = i,   r = i+z[i]-1;
11 }
12 return z;
13 }

```

4.4 Suffix Automata

```

1 struct state {
2     int len, link;
3     map<char,int> next;
4 };
5
6 state st[MAXLEN*2];
7 int sz, last;
8
9 void sa_init() {
10     sz = last = 0;
11     st[0].len = 0;
12     st[0].link = -1;
13     ++sz;
14 }
15
16 void sa_extend(char c) {
17     int cur = sz++;
18     st[cur].len = st[last].len + 1;
19     int p;
20     for (p=last; p!=-1 && !st[p].next.count(c); p=st[p].link)
21         st[p].next[c] = cur;
22     if (p == -1)
23         st[cur].link = 0;
24     else {
25         int q = st[p].next[c];
26         if (st[p].len + 1 == st[q].len)
27             st[cur].link = q;
28         else {
29             int clone = sz++;
30             st[clone].len = st[p].len + 1;
31             st[clone].next = st[q].next;
32             st[clone].link = st[q].link;
33             for (; p!=-1 && st[p].next[c]==q; p=st[p].link)
34                 st[p].next[c] = clone;
35             st[q].link = st[cur].link = clone;
36         }
37     }
38     last = cur;
39 }

```

4.5 Palindromes

```

1 for(i = 0; i < n; i++){
2     if(i > r) k = 1;
3     else k = min(d1[l + r - i], r - i);
4
5     while(0 <= i-k && i+k < n && s[i - k] == s[i + k]) k++;
6     d1[i] = k;
7     if(i + k - 1 > r)
8         r = i + k - 1, l = i - k + 1;
9 }
10
11 for(i = 0; i < n; i++){
12     if(i > r) k = 0;
13     else k = min(d2[l + r - i + 1], r - i + 1);
14
15     while(i + k < n && i - k - 1 >= 0 && s[i+k] == s[i - k - 1]) k++;
16     d2[i] = k;
17
18     if(i + k - 1 > r)
19         l = i - k, r = i + k - 1;
20 }

```

4.6 Lyndon decomposition & Duval

```

1 // Lyndon decomposition
2 for(int i = 0; i < n;) {
3     int j=i+1, k=i;
4     while (j < n && s[k] <= s[j]) {
5         if (s[k] < s[j])
6             k = j;
7         else
8             ++k;
9         ++j;
10    }
11    while (i <= k) {
12        cout << s.substr (i, j-k) << ' ';
13        i += j - k;
14    }
15 }
16
17 string min_cyclic_shift (string s) {
18     s += s;
19     int n = (int) s.length();
20     int i=0, ans=0;
21     while (i < n/2) {
22         ans = i;
23         int j=i+1, k=i;
24         while (j < n && s[k] <= s[j]) {
25             if (s[k] < s[j])
26                 k = j;
27             else
28                 ++k;
29             ++j;
30         }
31         while (i <= k)    i += j - k;
32     }
33     return s.substr (ans, n/2);
34 }

```

5 Modular

```

1 // All algorithms described here work on nonnegative integers.
2
3 // return a % b (positive value)
4 int mod(int a, int b) {
5     return ((a%b)+b)%b;
6 }
7
8 // computes gcd(a,b)
9 int gcd(int a, int b) {
10     int tmp;
11     while(b){a%=b; tmp=a; a=b; b=tmp;}
12     return a;
13 }
14
15 // computes lcm(a,b)
16 int lcm(int a, int b) {
17     return a/gcd(a,b)*b;
18 }
19
20 // returns d = gcd(a,b); finds x,y such that d = ax + by
21 int extended_euclid(int a, int b, int &x, int &y) {
22     int xx = y = 0;
23     int yy = x = 1;
24     while (b) {
25         int q = a/b;
26         int t = b; b = a%b; a = t;
27         t = xx; xx = x-q*xx; x = t;
28         t = yy; yy = y-q*yy; y = t;
29     }
30     return a;
31 }
32
33 // finds all solutions to ax = b (mod n)
34 ve<int> modular_linear_equation_solver(int a, int b, int n) {
35     int x, y;
36     ve<int> solutions;
37     int d = extended_euclid(a, n, x, y);
38     if (!(b%d)) {
39         x = mod (x*(b/d), n);
40         for (int i = 0; i < d; i++)
41             solutions.push_back(mod(x + i*(n/d), n));
42     }
43     return solutions;
44 }
45
46 // computes b such that ab = 1 (mod n), returns -1 on failure
47 int mod_inverse(int a, int n) {
48     int x, y;
49     int d = extended_euclid(a, n, x, y);
50     if (d > 1) return -1;
51     return mod(x,n);
52 }
53
54 // find z such that z % x = a, z % y = b. Here, z is unique modulo M = lcm(x,y).
55 // Return (z,M). On failure, M = -1.
56 pii chinese_remainder_theorem(int x, int a, int y, int b) {
57     int s, t;
58     int d = extended_euclid(x, y, s, t);
59     if (a%d != b%d) return make_pair(0, -1);
60     return make_pair(mod(s*b*x+t*a*y,x*y)/d, x*y/d);
61 }
62
63 // Chinese remainder theorem: find z such that
64 // z % x[i] = a[i] for all i. Note that the solution is
65 // unique modulo M = lcm_i (x[i]). Return (z,M). On failure, M = -1.
66 pii chinese_remainder_theorem(const ve<int> &x, const ve<int> &a) {
67     pii ret = make_pair(a[0], x[0]);
68     for (int i = 1; i < x.size(); i++) {
69         ret = chinese_remainder_theorem(ret.second, ret.first, x[i], a[i]);
70         if (ret.second == -1) break;
71     }
72     return ret;

```

```
73 }
74
75 // computes x and y such that ax + by = c; on failure, x = y = -1
76 void linear_diophantine(int a, int b, int c, int &x, int &y) {
77     int d = gcd(a,b);
78     if (c%d) {
79         x = y = -1;
80     } else {
81         x = c/d * mod_inverse(a/d, b/d);
82         y = (c-a*x)/b;
83     }
84 }
```

6 Data Structures

6.1 Treap

```

1 // key(l) < key, key(r) >= key
2 void split (pitem t, int key, pitem & l, pitem & r) {
3     if (!t) l = r = NULL;
4     else if (key < t->key)
5         split (t->l, key, l, t->l), r = t;
6     else
7         split (t->r, key, t->r, r), l = t;
8 }
9
10 void insert (pitem & t, pitem it) {
11     if (!t) t = it;
12     else if (it->prior > t->prior)
13         split (t, it->key, it->l, it->r), t = it;
14     else
15         insert (it->key < t->key ? t->l : t->r, it);
16 }
17
18 void merge (pitem & t, pitem l, pitem r) {
19     if (!l || !r) t = l ? l : r;
20     else if (l->prior > r->prior)
21         merge (l->r, l->r, r), t = l;
22     else
23         merge (r->l, l, r->l), t = r;
24 }

```

6.2 Implicit Treap

```

1 int cnt (pitem t) { return t ? t->cnt : 0; }
2
3 void upd_cnt (pitem t) {
4     if (t) t->cnt = 1 + cnt(t->l) + cnt (t->r);
5 }
6
7 void merge (pitem & t, pitem l, pitem r) {
8     if (!l || !r)
9         t = l ? l : r;
10    else if (l->prior > r->prior)
11        merge (l->r, l->r, r), t = l;
12    else
13        merge (r->l, l, r->l), t = r;
14    upd_cnt (t);
15 }
16 // key(l) < key, key(r) > key
17 void split (pitem t, pitem & l, pitem & r, int key, int add = 0) {
18     if (!t)
19         return void( l = r = 0 );
20     int cur_key = add + cnt(t->l);
21     if (key <= cur_key)
22         split (t->l, l, t->l, key, add), r = t;
23     else
24         split (t->r, t->r, r, key, add + 1 + cnt(t->l)), l = t;
25     upd_cnt (t);
26 }
27
28 pitem build(int *a, int cnt) {
29     if (cnt <= 0) return NULL;
30     int p = cnt/2;
31     pitem root = &buf[buf_sz++];
32     root->val = a[p]; // root->prior ???
33     root->l = build(a, p);
34     root->r = build(a + p + 1, cnt - p - 1);
35     upd_cnt(root);
36     return root;
37 }

```


6.3 Fenwick Tree

```

1 // 1-indexation
2 void set(int x, int v) {
3     while(x <= N) {
4         tree[x] += v;
5         x += (x & -x);
6     }
7 }
8 int get(int x) {
9     int res = 0;
10    while(x) {
11        res += tree[x];
12        x -= (x & -x);
13    }
14    return res;
15 }
16
17 // 0-indexation
18 int sum(int r) {
19     int res = 0;
20     for (; r >= 0; r = (r & (r+1)) - 1)
21         res += t[r];
22     return res;
23 }
24 void inc(int i, int val) {
25     for (; i < n; i = (i | (i+1)))
26         t[i] += val;
27 }

```

6.4 Stack with Min

```

1 // stack[i].second = min { stack[j].first }, j <= i
2
3 // add val
4 int minima = st.empty() ? val : min (val, st.top().second);
5 st.push ( mp(val, minima) );
6
7 // pop
8 int res = st.top().first;
9 st.pop();
10
11 // get minimum
12 int minima = st.top().second;

```

6.5 Queue with Min

```

1 deque<int> q;
2
3 // get minimum
4 current_minimum = q.front();
5
6 // add val
7 while (!q.empty() && q.back() > val)
8     q.pop_back();
9 q.push_back (val);
10
11 // pop val
12 if (!q.empty() && q.front() == val)
13     q.pop_front();

```

7 Geometry

7.1 Point 2D

```

1 // point projection on line (A, B)
2 pt getH(const pt & A, const pt & B) const {
3     pt C = *this;
4     pt v = B - A;
5     pt u = C - A;
6     double k = v ^ u / v.len();
7     v = v.norm(k);
8     pt H = A + v;
9     return H;
10 }
11
12 // Intersection of lines (A, B) & (C, D)
13 int getIntersection(const pt & A, const pt & B, const pt & C, const pt & D, pt & O) {
14     pt v = B - A;
15     double s1 = (C - A) * (D - A);
16     double s2 = (D - B) * (C - B);
17     double s = s1 + s2;
18     if(doubleEqual(s, 0) ) {
19         if(!A.isOnLine(C, D) ) {
20             return 0;
21         }
22         return 2;
23     }
24     v = v / s;
25     v = v * s1;
26     O = A + v;
27     return 1;
28 }
29
30 // Intersection of circles (A, rA) & (B, rB)
31 int getIntersection(const pt & A, double rA, const pt & B, double rB, pt & M, pt & N) {
32     double d = A.distTo(B);
33     if(doubleLess(rA + rB, d) || doubleLess(d, fabs(rA - rB)) ) {
34         return 0;
35     }
36     double a = (sqr(rA) - sqr(rB) + sqr(d) ) / 2 / d;
37     double h = mySqrt(sqr(rA) - sqr(a));
38     pt v = B - A;
39     pt u = v.rotate();
40     v = v.norm(a);
41     u = u.norm(h);
42     pt H = A + v;
43     M = H + u;
44     N = H - u;
45     if(u.isZero()) return 1;
46     return 2;
47 }
48
49 // Intersection of line (A, B) & circle (O, r)
50 int getIntersection(const pt & A, const pt & B, const pt & O, double r, pt & M, pt & N) {
51     double h = O.distTo(A, B);
52     if(doubleLess(r, h) ) {
53         return 0;
54     }
55     pt H = O.getH(A, B);
56     pt v = B - A;
57     double k = mySqrt(sqr(r) - sqr(h) );
58     v = v.norm(k);
59     M = H + v;
60     N = H - v;
61     if(v.isZero() ) return 1;
62     return 2;
63 }
64
65 // Tangent lines through point A to circle (O, r)
66 int getTangent(const pt & A, const pt & O, double r, pt & M, pt & N) {
67     pt v = O - A;
68     double d = v.len();
69     if(doubleLess(d, r) ) return 0;

```

```

70 double alpha = asin(r / d);
71 double L = mySqrt(sqr(d) - sqr(r));
72 v = v.norm(L);
73 M = A + v.rotate(alpha);
74 N = A - v.rotate(alpha);
75 if(doubleEqual(r, d)) return 1;
76 return 2;
77 }
78
79 // Outer tangent lines between circles (A, rA) & (B, rB)
80 void getOutTangent(pt A, double rA, pt B, double rB, pair<pt, pt> & P, pair<pt, pt> & Q) {
81     if(rA > rB) {
82         swap(rA, rB);
83         swap(A, B);
84     }
85     pt u = (A - B).rotate(asin(r / d)).rotate().norm(rA);
86     P.first = A + u;
87     Q.first = A - u;
88     pt T1, T2;
89     getTangent(A, B, rB - rA, T1, T2);
90     P.second = T1 + u;
91     Q.second = T2 - u;
92 }
93
94 // Inner tangent lines between circles (A, rA) & (B, rB)
95 void getInTangent(pt A, double rA, pt B, double rB, pair<pt, pt> & P, pair<pt, pt> & Q) {
96     pt I = (A * rB / (rA + rB)) + (B * rA / (rA + rB));
97     pt M1, N1, M2, N2;
98     getTangent(I, A, rA, M1, N1);
99     getTangent(I, B, rB, M2, N2);
100     if(I.isOnLine(M1, M2)) P = mp(M1, M2), Q = mp(N1, N2);
101     else P = mp(M1, N2), Q = mp(N1, M2);
102 }

```

7.2 Point 3D

```

1 pt operator*(const pt & p) const {
2     return pt( y * p.z - z * p.y,
3               z * p.x - x * p.z,
4               x * p.y - y * p.x );
5 }
6
7 // Projection on line (A, B)
8 pt getH(const pt & A, const pt & B) const {
9     pt C = *this;
10    pt v = B - A;
11    pt u = C - A;
12    double k = v ^ u / v.length();
13    v = v.norm(k);
14    pt H = A + v;
15    return H;
16 }
17
18 // Rotation
19 pt rotate(pt normal) const { return *this * normal; }
20
21 pt rotate(double alpha, const pt & normal) const {
22     return rotate(cos(alpha), sin(alpha), normal);
23 }
24
25 pt rotate(double cosa, double sina, const pt & normal) const {
26     pt v = *this;
27     pt u = v.rotate(normal);
28     pt w = v * cosa + u * sina;
29     return w;
30 }
31
32 // Undirected angle
33 double getAngle(pt u) const {
34     pt v = *this;
35     return atan2((v * u).length(), v ^ u);
36 }

```

```

37
38 bool isOnPlane(const pt & A, const pt & B, const pt & C) const {
39     return doubleEqual( (A - *this) * (B - *this) ^ (C - *this), 0);
40 }
41
42 // Intersection of lines (A, B) & (C, D)
43 int getIntersection(const pt & A, const pt & B, const pt & C, const pt & D, pt & O) {
44     if( !doubleEqual( (B - A) * (C - A) ^ (D - A), 0) ) {
45         throw "It's not plane";
46     }
47     if( doubleEqual( ( (A - B) * (C - D) ).length(), 0) ) {
48         if(A.isOnLine(C, D) ) return 2;
49         return 0;
50     }
51     pt normal = ( (A - B) * (C - B) ).norm();
52     pt v = B - A;
53     double s1 = (C - A) * (D - A) ^ normal;
54     double s2 = (D - B) * (C - B) ^ normal;
55     double s = s1 + s2;
56     v = v / s;
57     v = v * s1;
58     O = A + v;
59     return 1;
60 }
61
62 // Intersection of line (A, B) & plane (C, D, E)
63 int getIntersection(const pt & A, const pt & B, const pt & C, const pt & D, const pt & E, pt & O) {
64     pt v = B - A;
65     double V1 = (C - A) * (D - A) ^ (E - A); // thetetrahedra (A, C, D, E) volume
66     double V2 = (D - B) * (C - B) ^ (E - B);
67     double V = V1 + V2;
68     v = v / V;
69     if(doubleEqual(V, 0) ) {
70         if(A.isOnPlane(C, D, E) ) return 2;
71         return 0;
72     }
73     v = v * V1;
74     O = A + v;
75     return 1;
76 }
77
78 // Intersection of planes (A, nA) & (B, nB)
79 bool getIntersection(const pt & A, const pt & nA, const pt & B, const pt & nB, pt & P, pt & Q) {
80     pt n = nA * nB;
81     if(n.isZero() ) return false;
82     pt v = n * nA;
83     double k = (B - A) ^ nB / (v ^ nB);
84
85     v = v * k;
86     P = A + v;
87     Q = P + n;
88     return true;
89 }

```

Theoretical Computer Science Cheat Sheet

Definitions		Series
$f(n) = O(g(n))$	iff \exists positive c, n_0 such that $0 \leq f(n) \leq cg(n) \forall n \geq n_0$.	$\sum_{i=1}^n i = \frac{n(n+1)}{2}, \quad \sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^n i^3 = \frac{n^2(n+1)^2}{4}.$
$f(n) = \Omega(g(n))$	iff \exists positive c, n_0 such that $f(n) \geq cg(n) \geq 0 \forall n \geq n_0$.	In general:
$f(n) = \Theta(g(n))$	iff $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.	$\sum_{i=1}^n i^m = \frac{1}{m+1} \left[(n+1)^{m+1} - 1 - \sum_{i=1}^n ((i+1)^{m+1} - i^{m+1} - (m+1)i^m) \right]$
$f(n) = o(g(n))$	iff $\lim_{n \rightarrow \infty} f(n)/g(n) = 0$.	$\sum_{i=1}^{n-1} i^m = \frac{1}{m+1} \sum_{k=0}^m \binom{m+1}{k} B_k n^{m+1-k}.$
$\lim_{n \rightarrow \infty} a_n = a$	iff $\forall \epsilon > 0, \exists n_0$ such that $ a_n - a < \epsilon, \forall n \geq n_0$.	Geometric series:
$\sup S$	least $b \in \mathbb{R}$ such that $b \geq s, \forall s \in S$.	$\sum_{i=0}^n c^i = \frac{c^{n+1} - 1}{c - 1}, \quad c \neq 1, \quad \sum_{i=0}^{\infty} c^i = \frac{1}{1-c}, \quad \sum_{i=1}^{\infty} c^i = \frac{c}{1-c}, \quad c < 1,$
$\inf S$	greatest $b \in \mathbb{R}$ such that $b \leq s, \forall s \in S$.	$\sum_{i=0}^n ic^i = \frac{nc^{n+2} - (n+1)c^{n+1} + c}{(c-1)^2}, \quad c \neq 1, \quad \sum_{i=0}^{\infty} ic^i = \frac{c}{(1-c)^2}, \quad c < 1.$
$\liminf_{n \rightarrow \infty} a_n$	$\lim_{n \rightarrow \infty} \inf \{a_i \mid i \geq n, i \in \mathbb{N}\}.$	Harmonic series:
$\limsup_{n \rightarrow \infty} a_n$	$\lim_{n \rightarrow \infty} \sup \{a_i \mid i \geq n, i \in \mathbb{N}\}.$	$H_n = \sum_{i=1}^n \frac{1}{i}, \quad \sum_{i=1}^n iH_i = \frac{n(n+1)}{2}H_n - \frac{n(n-1)}{4}.$
$\binom{n}{k}$	Combinations: Size k sub-sets of a size n set.	$\sum_{i=1}^n H_i = (n+1)H_n - n, \quad \sum_{i=1}^n \binom{i}{m} H_i = \binom{n+1}{m+1} \left(H_{n+1} - \frac{1}{m+1} \right).$
$[n_k]$	Stirling numbers (1st kind): Arrangements of an n element set into k cycles.	1. $\binom{n}{k} = \frac{n!}{(n-k)!k!}, \quad 2. \sum_{k=0}^n \binom{n}{k} = 2^n, \quad 3. \binom{n}{k} = \binom{n}{n-k},$
$\{n_k\}$	Stirling numbers (2nd kind): Partitions of an n element set into k non-empty sets.	4. $\binom{n}{k} = \frac{n}{k} \binom{n-1}{k-1}, \quad 5. \binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1},$
$\langle n_k \rangle$	1st order Eulerian numbers: Permutations $\pi_1 \pi_2 \dots \pi_n$ on $\{1, 2, \dots, n\}$ with k ascents.	6. $\binom{n}{m} \binom{m}{k} = \binom{n}{k} \binom{n-k}{m-k}, \quad 7. \sum_{k=0}^n \binom{r+k}{k} = \binom{r+n+1}{n},$
$\langle\langle n_k \rangle\rangle$	2nd order Eulerian numbers.	8. $\sum_{k=0}^n \binom{k}{m} = \binom{n+1}{m+1}, \quad 9. \sum_{k=0}^n \binom{r}{k} \binom{s}{n-k} = \binom{r+s}{n},$
C_n	Catalan Numbers: Binary trees with $n+1$ vertices.	10. $\binom{n}{k} = (-1)^k \binom{k-n-1}{k}, \quad 11. \left\{ \begin{matrix} n \\ 1 \end{matrix} \right\} = \left\{ \begin{matrix} n \\ n \end{matrix} \right\} = 1,$
14. $\begin{bmatrix} n \\ 1 \end{bmatrix} = (n-1)!,$	15. $\begin{bmatrix} n \\ 2 \end{bmatrix} = (n-1)!H_{n-1},$	16. $\begin{bmatrix} n \\ n \end{bmatrix} = 1, \quad 17. \begin{bmatrix} n \\ k \end{bmatrix} \geq \left\{ \begin{matrix} n \\ k \end{matrix} \right\},$
18. $\begin{bmatrix} n \\ k \end{bmatrix} = (n-1) \begin{bmatrix} n-1 \\ k \end{bmatrix} + \begin{bmatrix} n-1 \\ k-1 \end{bmatrix},$	19. $\left\{ \begin{matrix} n \\ n-1 \end{matrix} \right\} = \begin{bmatrix} n \\ n-1 \end{bmatrix} = \binom{n}{2},$	20. $\sum_{k=0}^n \begin{bmatrix} n \\ k \end{bmatrix} = n!, \quad 21. C_n = \frac{1}{n+1} \binom{2n}{n},$
22. $\langle n_0 \rangle = \langle n_{n-1} \rangle = 1,$	23. $\langle n_k \rangle = \langle n_{n-1-k} \rangle,$	24. $\langle n_k \rangle = (k+1) \langle n_{k-1} \rangle + (n-k) \langle n_{k-1} \rangle,$
25. $\langle n_k \rangle = \begin{cases} 1 & \text{if } k=0, \\ 0 & \text{otherwise} \end{cases}$	26. $\langle n_1 \rangle = 2^n - n - 1,$	27. $\langle n_2 \rangle = 3^n - (n+1)2^n + \binom{n+1}{2},$
28. $x^n = \sum_{k=0}^n \langle n_k \rangle \binom{x+k}{n},$	29. $\langle n_m \rangle = \sum_{k=0}^m \binom{n+1}{k} (m+1-k)^n (-1)^k,$	30. $m! \left\{ \begin{matrix} n \\ m \end{matrix} \right\} = \sum_{k=0}^n \langle n_k \rangle \binom{k}{n-m},$
31. $\langle n_m \rangle = \sum_{k=0}^n \left\{ \begin{matrix} n \\ k \end{matrix} \right\} \binom{n-k}{m} (-1)^{n-k-m} k!,$	32. $\langle\langle n_0 \rangle\rangle = 1,$	33. $\langle\langle n_n \rangle\rangle = 0 \text{ for } n \neq 0,$
34. $\langle\langle n_k \rangle\rangle = (k+1) \langle\langle n_{k-1} \rangle\rangle + (2n-1-k) \langle\langle n_{k-1} \rangle\rangle,$	35. $\sum_{k=0}^n \langle\langle n_k \rangle\rangle = \frac{(2n)^n}{2^n},$	
36. $\left\{ \begin{matrix} x \\ x-n \end{matrix} \right\} = \sum_{k=0}^n \langle\langle n_k \rangle\rangle \binom{x+n-1-k}{2n},$	37. $\left\{ \begin{matrix} n+1 \\ m+1 \end{matrix} \right\} = \sum_k \binom{n}{k} \left\{ \begin{matrix} k \\ m \end{matrix} \right\} = \sum_{k=0}^n \left\{ \begin{matrix} k \\ m \end{matrix} \right\} (m+1)^{n-k},$	

Theoretical Computer Science Cheat Sheet

Identities Cont.

$$\begin{aligned}
38. \quad \left[\begin{matrix} n+1 \\ m+1 \end{matrix} \right] &= \sum_k \left[\begin{matrix} n \\ k \end{matrix} \right] \binom{k}{m} = \sum_{k=0}^n \left[\begin{matrix} k \\ m \end{matrix} \right] n^{n-k} = n! \sum_{k=0}^n \frac{1}{k!} \left[\begin{matrix} k \\ m \end{matrix} \right], & 39. \quad \left[\begin{matrix} x \\ x-n \end{matrix} \right] &= \sum_{k=0}^n \left\langle \begin{matrix} n \\ k \end{matrix} \right\rangle \binom{x+k}{2n}, \\
40. \quad \left\{ \begin{matrix} n \\ m \end{matrix} \right\} &= \sum_k \binom{n}{k} \left\{ \begin{matrix} k+1 \\ m+1 \end{matrix} \right\} (-1)^{n-k}, & 41. \quad \left[\begin{matrix} n \\ m \end{matrix} \right] &= \sum_k \left[\begin{matrix} n+1 \\ k+1 \end{matrix} \right] \binom{k}{m} (-1)^{m-k}, \\
42. \quad \left\{ \begin{matrix} m+n+1 \\ m \end{matrix} \right\} &= \sum_{k=0}^m k \left\{ \begin{matrix} n+k \\ k \end{matrix} \right\}, & 43. \quad \left[\begin{matrix} m+n+1 \\ m \end{matrix} \right] &= \sum_{k=0}^m k(n+k) \left[\begin{matrix} n+k \\ k \end{matrix} \right], \\
44. \quad \binom{n}{m} &= \sum_k \left\{ \begin{matrix} n+1 \\ k+1 \end{matrix} \right\} \left[\begin{matrix} k \\ m \end{matrix} \right] (-1)^{m-k}, & 45. \quad (n-m)! \binom{n}{m} &= \sum_k \left[\begin{matrix} n+1 \\ k+1 \end{matrix} \right] \left\{ \begin{matrix} k \\ m \end{matrix} \right\} (-1)^{m-k}, \quad \text{for } n \geq m, \\
46. \quad \left\{ \begin{matrix} n \\ n-m \end{matrix} \right\} &= \sum_k \binom{m-n}{m+k} \binom{m+n}{n+k} \left[\begin{matrix} m+k \\ k \end{matrix} \right], & 47. \quad \left[\begin{matrix} n \\ n-m \end{matrix} \right] &= \sum_k \binom{m-n}{m+k} \binom{m+n}{n+k} \left\{ \begin{matrix} m+k \\ k \end{matrix} \right\}, \\
48. \quad \left\{ \begin{matrix} n \\ \ell+m \end{matrix} \right\} \binom{\ell+m}{\ell} &= \sum_k \left\{ \begin{matrix} k \\ \ell \end{matrix} \right\} \left\{ \begin{matrix} n-k \\ m \end{matrix} \right\} \binom{n}{k}, & 49. \quad \left[\begin{matrix} n \\ \ell+m \end{matrix} \right] \binom{\ell+m}{\ell} &= \sum_k \left[\begin{matrix} k \\ \ell \end{matrix} \right] \left[\begin{matrix} n-k \\ m \end{matrix} \right] \binom{n}{k}.
\end{aligned}$$

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, \dots, d_n :

$$\sum_{i=1}^n 2^{-d_i} \leq 1,$$

and equality holds only if every internal node has 2 sons.

Recurrences

Master method:

$$T(n) = aT(n/b) + f(n), \quad a \geq 1, b > 1$$

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 $af(n/b) \leq cf(n)$ for large n , then

$$T(n) = \Theta(f(n)).$$

Substitution (example): Consider the following recurrence

$$T_{i+1} = 2^{2^i} \cdot T_i, \quad T_1 = 2.$$

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

$$u_{i+1} = \frac{1}{2} + u_i, \quad u_1 = \frac{1}{2},$$

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”

$$1(T(n) - 3T(n/2)) = n$$

$$3(T(n/2) - 3T(n/4)) = n/2$$

$$\vdots \quad \vdots \quad \vdots$$

$$3^{\log_2 n - 1} (T(2) - 3T(1)) = 2$$

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

$$\sum_{i=0}^{m-1} \frac{n}{2^i} 3^i = n \sum_{i=0}^{m-1} \left(\frac{3}{2}\right)^i.$$

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

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

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

Generating functions:

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

Example:

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

Multiply and sum:

$$\sum_{i \geq 0} g_{i+1} x^i = \sum_{i \geq 0} 2g_i x^i + \sum_{i \geq 0} x^i.$$

We choose $G(x) = \sum_{i \geq 0} x^i g_i$. Rewrite in terms of $G(x)$:

$$\frac{G(x) - g_0}{x} = 2G(x) + \sum_{i \geq 0} x^i.$$

Simplify:

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

Solve for $G(x)$:

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

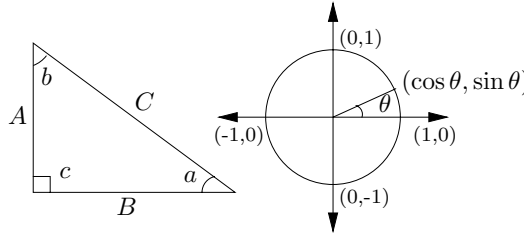
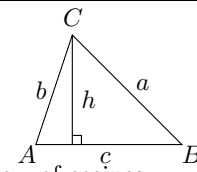
Expand this using partial fractions:

$$\begin{aligned}
G(x) &= x \left(\frac{2}{1-2x} - \frac{1}{1-x} \right) \\
&= x \left(2 \sum_{i \geq 0} 2^i x^i - \sum_{i \geq 0} x^i \right) \\
&= \sum_{i \geq 0} (2^{i+1} - 1) x^{i+1}.
\end{aligned}$$

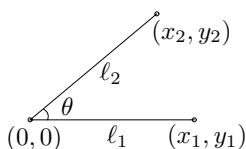
So $g_i = 2^i - 1$.

Theoretical Computer Science Cheat Sheet					
$\pi \approx 3.14159,$		$e \approx 2.71828,$	$\gamma \approx 0.57721,$	$\phi = \frac{1+\sqrt{5}}{2} \approx 1.61803,$	$\hat{\phi} = \frac{1-\sqrt{5}}{2} \approx -.61803$
i	2^i	p_i	General	Probability	
1	2	2	Bernoulli Numbers ($B_i = 0$, odd $i \neq 1$):	Continuous distributions: If	
2	4	3	$B_0 = 1, B_1 = -\frac{1}{2}, B_2 = \frac{1}{6}, B_4 = -\frac{1}{30},$	$\Pr[a < X < b] = \int_a^b p(x) dx,$	
3	8	5	$B_6 = \frac{1}{42}, B_8 = -\frac{1}{30}, B_{10} = \frac{5}{66}.$	then p is the probability density function of	
4	16	7	Change of base, quadratic formula:	X . If	
5	32	11	$\log_b x = \frac{\log_a x}{\log_a b}, \quad \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$	$\Pr[X < a] = P(a),$	
6	64	13	Euler's number e :	then P is the distribution function of X . If	
7	128	17	$e = 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + \frac{1}{120} + \dots$	P and p both exist then	
8	256	19	$\lim_{n \rightarrow \infty} \left(1 + \frac{x}{n}\right)^n = e^x.$	$P(a) = \int_{-\infty}^a p(x) dx.$	
9	512	23	$\left(1 + \frac{1}{n}\right)^n < e < \left(1 + \frac{1}{n}\right)^{n+1}.$	Expectation: If X is discrete	
10	1,024	29	$\left(1 + \frac{1}{n}\right)^n = e - \frac{e}{2n} + \frac{11e}{24n^2} - O\left(\frac{1}{n^3}\right).$	$E[g(X)] = \sum_x g(x) \Pr[X = x].$	
11	2,048	31	Harmonic numbers:	If X continuous then	
12	4,096	37	$1, \frac{3}{2}, \frac{11}{6}, \frac{25}{12}, \frac{137}{60}, \frac{49}{20}, \frac{363}{140}, \frac{761}{280}, \frac{7129}{2520}, \dots$	$E[g(X)] = \int_{-\infty}^{\infty} g(x)p(x) dx = \int_{-\infty}^{\infty} g(x) dP(x).$	
13	8,192	41	$\ln n < H_n < \ln n + 1,$	Variance, standard deviation:	
14	16,384	43	$H_n = \ln n + \gamma + O\left(\frac{1}{n}\right).$	$\text{VAR}[X] = E[X^2] - E[X]^2,$	
15	32,768	47	Factorial, Stirling's approximation:	$\sigma = \sqrt{\text{VAR}[X]}.$	
16	65,536	53	$1, 2, 6, 24, 120, 720, 5040, 40320, 362880, \dots$	For events A and B :	
17	131,072	59	$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right).$	$\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$	
18	262,144	61	Ackermann's function and inverse:	$\Pr[A \wedge B] = \Pr[A] \cdot \Pr[B],$	
19	524,288	67	$a(i, j) = \begin{cases} 2^j & i = 1 \\ a(i-1, 2) & j = 1 \\ a(i-1, a(i, j-1)) & i, j \geq 2 \end{cases}$	iff A and B are independent.	
20	1,048,576	71	$\alpha(i) = \min\{j \mid a(j, j) \geq i\}.$	$\Pr[A B] = \frac{\Pr[A \wedge B]}{\Pr[B]}$	
21	2,097,152	73	Binomial distribution:	For random variables X and Y :	
22	4,194,304	79	$\Pr[X = k] = \binom{n}{k} p^k q^{n-k}, \quad q = 1 - p,$	$E[X \cdot Y] = E[X] \cdot E[Y],$	
23	8,388,608	83	$E[X] = \sum_{k=1}^n k \binom{n}{k} p^k q^{n-k} = np.$	if X and Y are independent.	
24	16,777,216	89	Poisson distribution:	$E[X + Y] = E[X] + E[Y],$	
25	33,554,432	97	$\Pr[X = k] = \frac{e^{-\lambda} \lambda^k}{k!}, \quad E[X] = \lambda.$	$E[cX] = cE[X].$	
26	67,108,864	101	Normal (Gaussian) distribution:	Bayes' theorem:	
27	134,217,728	103	$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}, \quad E[X] = \mu.$	$\Pr[A_i B] = \frac{\Pr[B A_i] \Pr[A_i]}{\sum_{j=1}^n \Pr[A_j] \Pr[B A_j]}.$	
28	268,435,456	107	The "coupon collector": We are given a	Inclusion-exclusion:	
29	536,870,912	109	random coupon each day, and there are n	$\Pr\left[\bigvee_{i=1}^n X_i\right] = \sum_{i=1}^n \Pr[X_i] +$	
30	1,073,741,824	113	different types of coupons. The distribu-	$\sum_{k=2}^n (-1)^{k+1} \sum_{i_1 < \dots < i_k} \Pr\left[\bigwedge_{j=1}^k X_{i_j}\right].$	
31	2,147,483,648	127	tion of coupons is uniform. The expected	Moment inequalities:	
32	4,294,967,296	131	number of days to pass before we to col-	$\Pr[X \geq \lambda E[X]] \leq \frac{1}{\lambda},$	
Pascal's Triangle			lect all n types is	$\Pr[X - E[X] \geq \lambda \cdot \sigma] \leq \frac{1}{\lambda^2}.$	
1			$nH_n.$	Geometric distribution:	
1 1				$\Pr[X = k] = pq^{k-1}, \quad q = 1 - p,$	
1 2 1				$E[X] = \sum_{k=1}^{\infty} k pq^{k-1} = \frac{1}{p}.$	
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					

Theoretical Computer Science Cheat Sheet

Theoretical Computer Science Cheat Sheet																										
Trigonometry	Matrices	More Trig.																								
<div></div> <p>Pythagorean theorem: $C^2 = A^2 + B^2$.</p> <p>Definitions: $\sin a = A/C, \quad \cos a = B/C,$ $\csc a = C/A, \quad \sec a = C/B,$ $\tan a = \frac{\sin a}{\cos a} = \frac{A}{B}, \quad \cot a = \frac{\cos a}{\sin a} = \frac{B}{A}.$</p> <p>Area, radius of inscribed circle: $\frac{1}{2}AB, \quad \frac{AB}{A+B+C}.$</p> <p>Identities:</p> <table><tr><td>$\sin x = \frac{1}{\csc x},$</td><td>$\cos x = \frac{1}{\sec x},$</td></tr><tr><td>$\tan x = \frac{1}{\cot x},$</td><td>$\sin^2 x + \cos^2 x = 1,$</td></tr><tr><td>$1 + \tan^2 x = \sec^2 x,$</td><td>$1 + \cot^2 x = \csc^2 x,$</td></tr><tr><td>$\sin x = \cos\left(\frac{\pi}{2} - x\right),$</td><td>$\sin x = \sin(\pi - x),$</td></tr><tr><td>$\cos x = -\cos(\pi - x),$</td><td>$\tan x = \cot\left(\frac{\pi}{2} - x\right),$</td></tr><tr><td>$\cot x = -\cot(\pi - x),$</td><td>$\csc x = \cot \frac{\pi}{2} - \cot x,$</td></tr></table> <p>$\sin(x \pm y) = \sin x \cos y \pm \cos x \sin y,$ $\cos(x \pm y) = \cos x \cos y \mp \sin x \sin y,$ $\tan(x \pm y) = \frac{\tan x \pm \tan y}{1 \mp \tan x \tan y},$ $\cot(x \pm y) = \frac{\cot x \cot y \mp 1}{\cot x \pm \cot y},$ $\sin 2x = 2 \sin x \cos x, \quad \sin 2x = \frac{2 \tan x}{1 + \tan^2 x},$ $\cos 2x = \cos^2 x - \sin^2 x, \quad \cos 2x = 2 \cos^2 x - 1,$ $\cos 2x = 1 - 2 \sin^2 x, \quad \cos 2x = \frac{1 - \tan^2 x}{1 + \tan^2 x},$ $\tan 2x = \frac{2 \tan x}{1 - \tan^2 x}, \quad \cot 2x = \frac{\cot^2 x - 1}{2 \cot x},$ $\sin(x+y) \sin(x-y) = \sin^2 x - \sin^2 y,$ $\cos(x+y) \cos(x-y) = \cos^2 x - \sin^2 y.$</p> <p>Euler's equation: $e^{ix} = \cos x + i \sin x, \quad e^{i\pi} = -1.$</p>	$\sin x = \frac{1}{\csc x},$	$\cos x = \frac{1}{\sec x},$	$\tan x = \frac{1}{\cot x},$	$\sin^2 x + \cos^2 x = 1,$	$1 + \tan^2 x = \sec^2 x,$	$1 + \cot^2 x = \csc^2 x,$	$\sin x = \cos\left(\frac{\pi}{2} - x\right),$	$\sin x = \sin(\pi - x),$	$\cos x = -\cos(\pi - x),$	$\tan x = \cot\left(\frac{\pi}{2} - x\right),$	$\cot x = -\cot(\pi - x),$	$\csc x = \cot \frac{\pi}{2} - \cot x,$	<p>Multiplication: $C = A \cdot B, \quad c_{i,j} = \sum_{k=1}^n a_{i,k} b_{k,j}.$</p> <p>Determinants: $\det A \neq 0$ iff A is non-singular. $\det A \cdot B = \det A \cdot \det B,$ $\det A = \sum_{\pi} \prod_{i=1}^n \text{sign}(\pi) a_{i,\pi(i)}.$</p> <p>$2 \times 2$ and 3×3 determinant: $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc,$ $\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = g \begin{vmatrix} b & c \\ e & f \end{vmatrix} - h \begin{vmatrix} a & c \\ d & f \end{vmatrix} + i \begin{vmatrix} a & b \\ d & e \end{vmatrix}$ $= aei + bfg + cdh - ceg - fha - ibd.$</p> <p>Permanents: $\text{perm } A = \sum_{\pi} \prod_{i=1}^n a_{i,\pi(i)}.$</p>	<div></div> <p>Law of cosines: $c^2 = a^2 + b^2 - 2ab \cos C.$</p> <p>Area: $A = \frac{1}{2}hc,$ $= \frac{1}{2}ab \sin C,$ $= \frac{c^2 \sin A \sin B}{2 \sin C}.$</p> <p>Heron's formula: $A = \sqrt{s \cdot s_a \cdot s_b \cdot s_c},$ $s = \frac{1}{2}(a + b + c),$ $s_a = s - a,$ $s_b = s - b,$ $s_c = s - c.$</p> <p>More identities: $\sin \frac{x}{2} = \sqrt{\frac{1 - \cos x}{2}},$ $\cos \frac{x}{2} = \sqrt{\frac{1 + \cos x}{2}},$ $\tan \frac{x}{2} = \sqrt{\frac{1 - \cos x}{1 + \cos x}},$ $= \frac{1 - \cos x}{\sin x},$ $= \frac{\sin x}{1 + \cos x},$ $\cot \frac{x}{2} = \sqrt{\frac{1 + \cos x}{1 - \cos x}},$ $= \frac{1 + \cos x}{\sin x},$ $= \frac{\sin x}{1 - \cos x},$ $\sin x = \frac{e^{ix} - e^{-ix}}{2i},$ $\cos x = \frac{e^{ix} + e^{-ix}}{2},$ $\tan x = -i \frac{e^{ix} - e^{-ix}}{e^{ix} + e^{-ix}},$ $= -i \frac{e^{2ix} - 1}{e^{2ix} + 1},$ $\sin x = \frac{\sinh ix}{i},$ $\cos x = \cosh ix,$ $\tan x = \frac{\tanh ix}{i}.$</p>												
$\sin x = \frac{1}{\csc x},$	$\cos x = \frac{1}{\sec x},$																									
$\tan x = \frac{1}{\cot x},$	$\sin^2 x + \cos^2 x = 1,$																									
$1 + \tan^2 x = \sec^2 x,$	$1 + \cot^2 x = \csc^2 x,$																									
$\sin x = \cos\left(\frac{\pi}{2} - x\right),$	$\sin x = \sin(\pi - x),$																									
$\cos x = -\cos(\pi - x),$	$\tan x = \cot\left(\frac{\pi}{2} - x\right),$																									
$\cot x = -\cot(\pi - x),$	$\csc x = \cot \frac{\pi}{2} - \cot x,$																									
	<p>Hyperbolic Functions</p> <p>Definitions: $\sinh x = \frac{e^x - e^{-x}}{2}, \quad \cosh x = \frac{e^x + e^{-x}}{2},$ $\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad \text{csch } x = \frac{1}{\sinh x},$ $\text{sech } x = \frac{1}{\cosh x}, \quad \coth x = \frac{1}{\tanh x}.$</p> <p>Identities: $\cosh^2 x - \sinh^2 x = 1, \quad \tanh^2 x + \text{sech}^2 x = 1,$ $\coth^2 x - \text{csch}^2 x = 1, \quad \sinh(-x) = -\sinh x,$ $\cosh(-x) = \cosh x, \quad \tanh(-x) = -\tanh x,$ $\sinh(x+y) = \sinh x \cosh y + \cosh x \sinh y,$ $\cosh(x+y) = \cosh x \cosh y + \sinh x \sinh y,$ $\sinh 2x = 2 \sinh x \cosh x,$ $\cosh 2x = \cosh^2 x + \sinh^2 x,$ $\cosh x + \sinh x = e^x, \quad \cosh x - \sinh x = e^{-x},$ $(\cosh x + \sinh x)^n = \cosh nx + \sinh nx, \quad n \in \mathbb{Z},$ $2 \sinh^2 \frac{x}{2} = \cosh x - 1, \quad 2 \cosh^2 \frac{x}{2} = \cosh x + 1.$</p> <table><tr><th>$\theta$</th><th>$\sin \theta$</th><th>$\cos \theta$</th><th>$\tan \theta$</th></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td></tr><tr><td>$\frac{\pi}{6}$</td><td>$\frac{1}{2}$</td><td>$\frac{\sqrt{3}}{2}$</td><td>$\frac{\sqrt{3}}{3}$</td></tr><tr><td>$\frac{\pi}{4}$</td><td>$\frac{\sqrt{2}}{2}$</td><td>$\frac{\sqrt{2}}{2}$</td><td>1</td></tr><tr><td>$\frac{\pi}{3}$</td><td>$\frac{\sqrt{3}}{2}$</td><td>$\frac{1}{2}$</td><td>$\sqrt{3}$</td></tr><tr><td>$\frac{\pi}{2}$</td><td>1</td><td>0</td><td>∞</td></tr></table> <p>... in mathematics you don't understand things, you just get used to them. - J. von Neumann</p>	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	∞	
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	∞																							
v2.02 ©1994 by Steve Seiden sseiden@acm.org http://www.csc.lsu.edu/~seiden																										

Theoretical Computer Science Cheat Sheet

Number Theory	Graph Theory	
<p>The Chinese remainder theorem: There exists a number C such that:</p> $C \equiv r_1 \pmod{m_1}$ \vdots $C \equiv r_n \pmod{m_n}$ <p>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</p> $\phi(x) = \prod_{i=1}^n p_i^{e_i-1} (p_i - 1).$ <p>Euler's theorem: If a and b are relatively prime then</p> $1 \equiv a^{\phi(b)} \pmod{b}.$ <p>Fermat's theorem:</p> $1 \equiv a^{p-1} \pmod{p}.$ <p>The Euclidean algorithm: if $a > b$ are integers then</p> $\gcd(a, b) = \gcd(a \bmod b, b).$ <p>If $\prod_{i=1}^n p_i^{e_i}$ is the prime factorization of x then</p> $S(x) = \sum_{d x} d = \prod_{i=1}^n \frac{p_i^{e_i+1} - 1}{p_i - 1}.$ <p>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</p> $(n - 1)! \equiv -1 \pmod{n}.$ <p>Möbius inversion:</p> $\mu(i) = \begin{cases} 1 & \text{if } i = 1. \\ 0 & \text{if } i \text{ is not square-free.} \\ (-1)^r & \text{if } i \text{ is the product of } r \text{ distinct primes.} \end{cases}$ <p>If</p> $G(a) = \sum_{d a} F(d),$ <p>then</p> $F(a) = \sum_{d a} \mu(d) G\left(\frac{a}{d}\right).$ <p>Prime numbers:</p> $p_n = n \ln n + n \ln \ln n - n + n \frac{\ln \ln n}{\ln n} + O\left(\frac{n}{\ln n}\right),$ $\pi(n) = \frac{n}{\ln n} + \frac{n}{(\ln n)^2} + \frac{2!n}{(\ln n)^3} + O\left(\frac{n}{(\ln n)^4}\right).$	<p>Definitions:</p> <p><i>Loop</i> An edge connecting a vertex to itself.</p> <p><i>Directed</i> Each edge has a direction.</p> <p><i>Simple</i> Graph with no loops or multi-edges.</p> <p><i>Walk</i> A sequence $v_0 e_1 v_1 \dots e_\ell v_\ell$.</p> <p><i>Trail</i> A walk with distinct edges.</p> <p><i>Path</i> A trail with distinct vertices.</p> <p><i>Connected</i> A graph where there exists a path between any two vertices.</p> <p><i>Component</i> A maximal connected subgraph.</p> <p><i>Tree</i> A connected acyclic graph.</p> <p><i>Free tree</i> A tree with no root.</p> <p><i>DAG</i> Directed acyclic graph.</p> <p><i>Eulerian</i> Graph with a trail visiting each edge exactly once.</p> <p><i>Hamiltonian</i> Graph with a cycle visiting each vertex exactly once.</p> <p><i>Cut</i> A set of edges whose removal increases the number of components.</p> <p><i>Cut-set</i> A minimal cut.</p> <p><i>Cut edge</i> A size 1 cut.</p> <p><i>k-Connected</i> A graph connected with the removal of any $k - 1$ vertices.</p> <p><i>k-Tough</i> $\forall S \subseteq V, S \neq \emptyset$ we have $k \cdot c(G - S) \leq S$.</p> <p><i>k-Regular</i> A graph where all vertices have degree k.</p> <p><i>k-Factor</i> A k-regular spanning subgraph.</p> <p><i>Matching</i> A set of edges, no two of which are adjacent.</p> <p><i>Clique</i> A set of vertices, all of which are adjacent.</p> <p><i>Ind. set</i> A set of vertices, none of which are adjacent.</p> <p><i>Vertex cover</i> A set of vertices which cover all edges.</p> <p><i>Planar graph</i> A graph which can be embedded in the plane.</p> <p><i>Plane graph</i> An embedding of a planar graph.</p> <hr/> $\sum_{v \in V} \deg(v) = 2m.$ <p>If G is planar then $n - m + f = 2$, so</p> $f \leq 2n - 4, \quad m \leq 3n - 6.$ <p>Any planar graph has a vertex with degree ≤ 5.</p>	<p>Notation:</p> <p>$E(G)$ Edge set</p> <p>$V(G)$ Vertex set</p> <p>$c(G)$ Number of components</p> <p>$G[S]$ Induced subgraph</p> <p>$\deg(v)$ Degree of v</p> <p>$\Delta(G)$ Maximum degree</p> <p>$\delta(G)$ Minimum degree</p> <p>$\chi(G)$ Chromatic number</p> <p>$\chi_E(G)$ Edge chromatic number</p> <p>G^c Complement graph</p> <p>K_n Complete graph</p> <p>K_{n_1, n_2} Complete bipartite graph</p> <p>$r(k, \ell)$ Ramsey number</p> <hr/> <p>Geometry</p> <p>Projective coordinates: triples (x, y, z), not all x, y and z zero. $(x, y, z) = (cx, cy, cz) \quad \forall c \neq 0.$</p> <p>Cartesian Projective</p> <hr/> <p>(x, y) $(x, y, 1)$</p> <p>$y = mx + b$ $(m, -1, b)$</p> <p>$x = c$ $(1, 0, -c)$</p> <p>Distance formula, L_p and L_∞ metric:</p> $\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2},$ $[x_1 - x_0 ^p + y_1 - y_0 ^p]^{1/p},$ $\lim_{p \rightarrow \infty} [x_1 - x_0 ^p + y_1 - y_0 ^p]^{1/p}.$ <p>Area of triangle $(x_0, y_0), (x_1, y_1)$ and (x_2, y_2):</p> $\frac{1}{2} \text{abs} \begin{vmatrix} x_1 - x_0 & y_1 - y_0 \\ x_2 - x_0 & y_2 - y_0 \end{vmatrix}.$ <p>Angle formed by three points:</p>  $\cos \theta = \frac{(x_1, y_1) \cdot (x_2, y_2)}{\ell_1 \ell_2}.$ <p>Line through two points (x_0, y_0) and (x_1, y_1):</p> $\begin{vmatrix} x & y & 1 \\ x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \end{vmatrix} = 0.$ <p>Area of circle, volume of sphere:</p> $A = \pi r^2, \quad V = \frac{4}{3} \pi r^3.$ <hr/> <p>If I have seen farther than others, it is because I have stood on the shoulders of giants. – Issac Newton</p>

Theoretical Computer Science Cheat Sheet

π	Calculus
<p>Wallis' identity:</p> $\pi = 2 \cdot \frac{2 \cdot 2 \cdot 4 \cdot 4 \cdot 6 \cdot 6 \cdots}{1 \cdot 3 \cdot 3 \cdot 5 \cdot 5 \cdot 7 \cdots}$ <p>Brouncker's continued fraction expansion:</p> $\frac{\pi}{4} = 1 + \frac{1^2}{2 + \frac{3^2}{2 + \frac{5^2}{2 + \frac{7^2}{2 + \cdots}}}}$ <p>Gregory's series:</p> $\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots$ <p>Newton's series:</p> $\frac{\pi}{6} = \frac{1}{2} + \frac{1}{2 \cdot 3 \cdot 2^3} + \frac{1 \cdot 3}{2 \cdot 4 \cdot 5 \cdot 2^5} + \cdots$ <p>Sharp's series:</p> $\frac{\pi}{6} = \frac{1}{\sqrt{3}} \left(1 - \frac{1}{3^1 \cdot 3} + \frac{1}{3^2 \cdot 5} - \frac{1}{3^3 \cdot 7} + \cdots \right)$ <p>Euler's series:</p> $\frac{\pi^2}{6} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \cdots$ $\frac{\pi^2}{8} = \frac{1}{1^2} + \frac{1}{3^2} + \frac{1}{5^2} + \frac{1}{7^2} + \frac{1}{9^2} + \cdots$ $\frac{\pi^2}{12} = \frac{1}{1^2} - \frac{1}{2^2} + \frac{1}{3^2} - \frac{1}{4^2} + \frac{1}{5^2} - \cdots$	<p>Derivatives:</p> <ol style="list-style-type: none"> $\frac{d(cu)}{dx} = c \frac{du}{dx},$ $\frac{d(u+v)}{dx} = \frac{du}{dx} + \frac{dv}{dx},$ $\frac{d(uv)}{dx} = u \frac{dv}{dx} + v \frac{du}{dx},$ $\frac{d(u^n)}{dx} = nu^{n-1} \frac{du}{dx},$ $\frac{d(u/v)}{dx} = \frac{v \left(\frac{du}{dx} \right) - u \left(\frac{dv}{dx} \right)}{v^2},$ $\frac{d(e^{cu})}{dx} = ce^{cu} \frac{du}{dx},$ $\frac{d(c^u)}{dx} = (\ln c) c^u \frac{du}{dx},$ $\frac{d(\ln u)}{dx} = \frac{1}{u} \frac{du}{dx},$ $\frac{d(\sin u)}{dx} = \cos u \frac{du}{dx},$ $\frac{d(\cos u)}{dx} = -\sin u \frac{du}{dx},$ $\frac{d(\tan u)}{dx} = \sec^2 u \frac{du}{dx},$ $\frac{d(\cot u)}{dx} = -\csc^2 u \frac{du}{dx},$ $\frac{d(\sec u)}{dx} = \tan u \sec u \frac{du}{dx},$ $\frac{d(\csc u)}{dx} = -\cot u \csc u \frac{du}{dx},$ $\frac{d(\arcsin u)}{dx} = \frac{1}{\sqrt{1-u^2}} \frac{du}{dx},$ $\frac{d(\arccos u)}{dx} = \frac{-1}{\sqrt{1-u^2}} \frac{du}{dx},$ $\frac{d(\arctan u)}{dx} = \frac{1}{1+u^2} \frac{du}{dx},$ $\frac{d(\text{arccot } u)}{dx} = \frac{-1}{1+u^2} \frac{du}{dx},$ $\frac{d(\text{arcsec } u)}{dx} = \frac{1}{u\sqrt{1-u^2}} \frac{du}{dx},$ $\frac{d(\text{arccsc } u)}{dx} = \frac{-1}{u\sqrt{1-u^2}} \frac{du}{dx},$ $\frac{d(\sinh u)}{dx} = \cosh u \frac{du}{dx},$ $\frac{d(\cosh u)}{dx} = \sinh u \frac{du}{dx},$ $\frac{d(\tanh u)}{dx} = \text{sech}^2 u \frac{du}{dx},$ $\frac{d(\coth u)}{dx} = -\text{csch}^2 u \frac{du}{dx},$ $\frac{d(\text{csch } u)}{dx} = -\text{csch } u \coth u \frac{du}{dx},$ $\frac{d(\text{arccosh } u)}{dx} = \frac{1}{\sqrt{u^2-1}} \frac{du}{dx},$ $\frac{d(\text{arccoth } u)}{dx} = \frac{1}{u^2-1} \frac{du}{dx},$ $\frac{d(\text{arcsch } u)}{dx} = \frac{-1}{ u \sqrt{1+u^2}} \frac{du}{dx}.$ <p>Integrals:</p> <ol style="list-style-type: none"> $\int cu \, dx = c \int u \, dx,$ $\int (u+v) \, dx = \int u \, dx + \int v \, dx,$ $\int x^n \, dx = \frac{1}{n+1} x^{n+1}, \quad n \neq -1,$ $\int \frac{1}{x} \, dx = \ln x,$ $\int e^x \, dx = e^x,$ $\int \frac{dx}{1+x^2} = \arctan x,$ $\int u \frac{dv}{dx} \, dx = uv - \int v \frac{du}{dx} \, dx,$ $\int \sin x \, dx = -\cos x,$ $\int \cos x \, dx = \sin x,$ $\int \tan x \, dx = -\ln \cos x ,$ $\int \cot x \, dx = \ln \cos x ,$ $\int \sec x \, dx = \ln \sec x + \tan x ,$ $\int \csc x \, dx = \ln \csc x + \cot x ,$ $\int \arcsin \frac{x}{a} \, dx = \arcsin \frac{x}{a} + \sqrt{a^2 - x^2}, \quad a > 0,$
<p>Partial Fractions</p> <p>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</p> $\frac{N(x)}{D(x)} = Q(x) + \frac{N'(x)}{D(x)},$ <p>where the degree of N' is less than that of D. Second, factor $D(x)$. Use the following rules: For a non-repeated factor:</p> $\frac{N(x)}{(x-a)D(x)} = \frac{A}{x-a} + \frac{N'(x)}{D(x)},$ <p>where</p> $A = \left[\frac{N(x)}{D(x)} \right]_{x=a}.$ <p>For a repeated factor:</p> $\frac{N(x)}{(x-a)^m D(x)} = \sum_{k=0}^{m-1} \frac{A_k}{(x-a)^{m-k}} + \frac{N'(x)}{D(x)},$ <p>where</p> $A_k = \frac{1}{k!} \left[\frac{d^k}{dx^k} \left(\frac{N(x)}{D(x)} \right) \right]_{x=a}.$	
<p>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.</p> <p>– George Bernard Shaw</p>	

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. $\int \sin^2(ax) dx = \frac{1}{2a}(ax - \sin(ax) \cos(ax)),$
18. $\int \cos^2(ax) dx = \frac{1}{2a}(ax + \sin(ax) \cos(ax)),$
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. $\int \operatorname{arccosh} \frac{x}{a} dx = \begin{cases} x \operatorname{arccosh} \frac{x}{a} - \sqrt{x^2 + a^2}, & \text{if } \operatorname{arccosh} \frac{x}{a} > 0 \text{ and } a > 0, \\ x \operatorname{arccosh} \frac{x}{a} + \sqrt{x^2 + a^2}, & \text{if } \operatorname{arccosh} \frac{x}{a} < 0 \text{ and } a > 0, \end{cases}$
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.

$$\begin{aligned}
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^2 x}, \\
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^2 x^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{4ac - 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 &= \left(\frac{1}{3}x^2 - \frac{2}{15}a^2\right)(x^2 + a^2)^{3/2}, \\
72. \int x^n \sin(ax) dx &= -\frac{1}{a}x^n \cos(ax) + \frac{n}{a} \int x^{n-1} \cos(ax) dx, \\
73. \int x^n \cos(ax) dx &= \frac{1}{a}x^n \sin(ax) - \frac{n}{a} \int x^{n-1} \sin(ax) dx, \\
74. \int x^n e^{ax} dx &= \frac{x^n e^{ax}}{a} - \frac{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.
\end{aligned}$$

Finite Calculus

Difference, shift operators:

$$\Delta f(x) = f(x+1) - f(x),$$

$$\mathbf{E} f(x) = f(x+1).$$

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:

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

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

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

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

$$\Delta(2^x) = 2^x,$$

$$\Delta(c^x) = (c-1)c^x,$$

$$\Delta\binom{x}{m} = \binom{x}{m-1}.$$

Sums:

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

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

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

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

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

Falling Factorial Powers:

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

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

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

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

Rising Factorial Powers:

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

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

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

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

Conversion:

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

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

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

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

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

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

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

$x^1 =$	$x^{\underline{1}}$	$=$	$x^{\overline{1}}$
$x^2 =$	$x^{\underline{2}} + x^{\underline{1}}$	$=$	$x^{\overline{2}} - x^{\overline{1}}$
$x^3 =$	$x^{\underline{3}} + 3x^{\underline{2}} + x^{\underline{1}}$	$=$	$x^{\overline{3}} - 3x^{\overline{2}} + x^{\overline{1}}$
$x^4 =$	$x^{\underline{4}} + 6x^{\underline{3}} + 7x^{\underline{2}} + x^{\underline{1}}$	$=$	$x^{\overline{4}} - 6x^{\overline{3}} + 7x^{\overline{2}} - x^{\overline{1}}$
$x^5 =$	$x^{\underline{5}} + 15x^{\underline{4}} + 25x^{\underline{3}} + 10x^{\underline{2}} + x^{\underline{1}}$	$=$	$x^{\overline{5}} - 15x^{\overline{4}} + 25x^{\overline{3}} - 10x^{\overline{2}} + x^{\overline{1}}$
$x^{\overline{1}} =$	x^1	$x^{\underline{1}} =$	x^1
$x^{\overline{2}} =$	$x^2 + x^1$	$x^{\underline{2}} =$	$x^2 - x^1$
$x^{\overline{3}} =$	$x^3 + 3x^2 + 2x^1$	$x^{\underline{3}} =$	$x^3 - 3x^2 + 2x^1$
$x^{\overline{4}} =$	$x^4 + 6x^3 + 11x^2 + 6x^1$	$x^{\underline{4}} =$	$x^4 - 6x^3 + 11x^2 - 6x^1$
$x^{\overline{5}} =$	$x^5 + 10x^4 + 35x^3 + 50x^2 + 24x^1$	$x^{\underline{5}} =$	$x^5 - 10x^4 + 35x^3 - 50x^2 + 24x^1$

Theoretical Computer Science Cheat Sheet

Series

Taylor's series:

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

Expansions:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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:

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

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

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

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

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

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

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

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

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

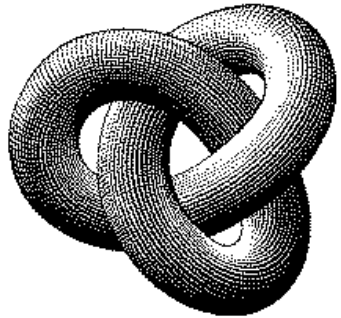
Summation: If $b_i = \sum_{j=0}^i a_j$ then

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

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		Escher's Knot																																																																																																				
<div>Expansions:</div> <div>$\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,$$x^{\overline{n}} = \sum_{i=0}^{\infty} \left[\begin{matrix} n \\ i \end{matrix} \right] x^i,$$\left(\ln \frac{1}{1-x} \right)^n = \sum_{i=0}^{\infty} \left[\begin{matrix} i \\ n \end{matrix} \right] \frac{n! x^i}{i!},$$\tan x = \sum_{i=1}^{\infty} (-1)^{i-1} \frac{2^{2i} (2^{2i} - 1) B_{2i} x^{2i-1}}{(2i)!},$$\frac{1}{\zeta(x)} = \sum_{i=1}^{\infty} \frac{\mu(i)}{i^x},$$\zeta(x) = \prod_p \frac{1}{1 - p^{-x}},$$\zeta^2(x) = \sum_{i=1}^{\infty} \frac{d(i)}{x^i} \quad \text{where } d(n) = \sum_{d n} 1,$$\zeta(x) \zeta(x-1) = \sum_{i=1}^{\infty} \frac{S(i)}{x^i} \quad \text{where } S(n) = \sum_{d n} d,$$\zeta(2n) = \frac{2^{2n-1} B_{2n} }{(2n)!} \pi^{2n}, \quad n \in \mathbb{N},$$\frac{x}{\sin x} = \sum_{i=0}^{\infty} (-1)^{i-1} \frac{(4^i - 2) B_{2i} x^{2i}}{(2i)!},$$\left(\frac{1 - \sqrt{1-4x}}{2x} \right)^n = \sum_{i=0}^{\infty} \frac{n(2i+n-1)!}{i!(n+i)!} x^i,$$e^x \sin x = \sum_{i=1}^{\infty} \frac{2^{i/2} \sin \frac{i\pi}{4}}{i!} x^i,$$\sqrt{\frac{1 - \sqrt{1-x}}{x}} = \sum_{i=0}^{\infty} \frac{(4i)!}{16^i \sqrt{2} (2i)!(2i+1)!} x^i,$$\left(\frac{\arcsin x}{x} \right)^2 = \sum_{i=0}^{\infty} \frac{4^i i!^2}{(i+1)(2i+1)!} x^{2i}.$</div>		<div></div>																																																																																																				
		Stieltjes Integration																																																																																																				
		<div>If G is continuous in the interval $[a, b]$ and F is nondecreasing then</div> <div>$\int_a^b G(x) dF(x)$</div> <div>exists. If $a \leq b \leq c$ then</div> <div>$\int_a^c G(x) dF(x) = \int_a^b G(x) dF(x) + \int_b^c G(x) dF(x).$</div> <div>If the integrals involved exist</div> <div>$\int_a^b (G(x) + H(x)) dF(x) = \int_a^b G(x) dF(x) + \int_a^b H(x) dF(x),$$\int_a^b G(x) d(F(x) + H(x)) = \int_a^b G(x) dF(x) + \int_a^b G(x) dH(x),$$\int_a^b c \cdot G(x) dF(x) = \int_a^b G(x) d(c \cdot F(x)) = c \int_a^b G(x) dF(x),$$\int_a^b G(x) dF(x) = G(b)F(b) - G(a)F(a) - \int_a^b F(x) dG(x).$</div> <div>If the integrals involved exist, and F possesses a derivative F' at every point in $[a, b]$ then</div> <div>$\int_a^b G(x) dF(x) = \int_a^b G(x) F'(x) dx.$</div>																																																																																																				
Cramer's Rule		Fibonacci Numbers																																																																																																				
<div>If we have equations:</div> <div>$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$$a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,n}x_n = b_n$</div> <div>Let $A = (a_{i,j})$ and B be the column matrix (b_i). Then there is a unique solution iff $\det A \neq 0$. Let A_i be A with column i replaced by B. Then</div> <div>$x_i = \frac{\det A_i}{\det A}.$</div>		<div><table><tr><td>00</td><td>47</td><td>18</td><td>76</td><td>29</td><td>93</td><td>85</td><td>34</td><td>61</td><td>52</td></tr><tr><td>86</td><td>11</td><td>57</td><td>28</td><td>70</td><td>39</td><td>94</td><td>45</td><td>02</td><td>63</td></tr><tr><td>95</td><td>80</td><td>22</td><td>67</td><td>38</td><td>71</td><td>49</td><td>56</td><td>13</td><td>04</td></tr><tr><td>59</td><td>96</td><td>81</td><td>33</td><td>07</td><td>48</td><td>72</td><td>60</td><td>24</td><td>15</td></tr><tr><td>73</td><td>69</td><td>90</td><td>82</td><td>44</td><td>17</td><td>58</td><td>01</td><td>35</td><td>26</td></tr><tr><td>68</td><td>74</td><td>09</td><td>91</td><td>83</td><td>55</td><td>27</td><td>12</td><td>46</td><td>30</td></tr><tr><td>37</td><td>08</td><td>75</td><td>19</td><td>92</td><td>84</td><td>66</td><td>23</td><td>50</td><td>41</td></tr><tr><td>14</td><td>25</td><td>36</td><td>40</td><td>51</td><td>62</td><td>03</td><td>77</td><td>88</td><td>99</td></tr><tr><td>21</td><td>32</td><td>43</td><td>54</td><td>65</td><td>06</td><td>10</td><td>89</td><td>97</td><td>78</td></tr><tr><td>42</td><td>53</td><td>64</td><td>05</td><td>16</td><td>20</td><td>31</td><td>98</td><td>79</td><td>87</td></tr></table></div>	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
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																																																																																													
		<div>The Fibonacci number system:</div> <div>Every integer n has a unique representation</div> <div>$n = F_{k_1} + F_{k_2} + \cdots + F_{k_m},$</div> <div>where $k_i \geq k_{i+1} + 2$ for all i, $1 \leq i < m$ and $k_m \geq 2$.</div>																																																																																																				
Improvement makes strait roads, but the crooked roads without Improvement, are roads of Genius. – William Blake (The Marriage of Heaven and Hell)																																																																																																						