

Team Contest Reference

Universität zu Lübeck



Team: No Output

Paul Ketelsen, Mathis Lichtenberger, Malte Skambath

Inhaltsverzeichnis

1 Mathematische Algorithmen

1.1 Primzahlen	2
1.1.1 Sieb des Eratosthenes $\mathcal{O}(n^2)$	2
1.1.2 Primzahlentest	2
1.2 Binomial Koeffizient	2
1.3 Eulersche φ -Funktion	2

2 Mathematisch Formeln und Gesetze

2.1 Catalan	2
2.2 kgV und ggT	2
2.3 modulare Exponentiation	2
2.4 Modulare Arithmetik	3
2.4.1 Erweiterter Euklidischer Algorithmus	3
2.5 Kombinatorik	3

3 Datenstrukturen

3.1 Fenwick Tree (Binary Indexed Tree)	3
3.2 Union-Find	3

4 Graphen

4.1 planare Graphen	3
4.2 Topologische Sortierung	3
4.3 Prim (Minimum Spanning Tree)	4
4.4 Kruskal	4
4.5 Floyd-Warshall ($\mathcal{O}(n^3)$)	4
4.6 Dijkstra	5
4.7 Belman-Ford	5
4.8 MaxFlow	5
4.9 Bipartite Matching	6
4.10 Bitonic TSP	6
4.11 Shortest Cycle	6

5 Geometrie

5.1 Kreuzprodukt, Skalarprodukt	7
5.2 Orthogonale Projektion	7
5.3 Rotation	7
5.4 Geradenschnittpunkt	7
5.5 Zusammenhang Kreuzprodukt & Sinus	7
5.6 Dreiecksfläche	7
5.7 Graham Scan (Convex Hull)	7
5.8 Line Intersection	8
5.9 Punkt in Polygon	8
5.10 Fläche eines Polygons	8

6 2-SAT-Solver

6.1 2-Sat mit SCC	8
6.2 Hilfsalgorithmen	9
6.2.1 Erzeugen eines Graphens	9
6.2.2 Indexumrechnung	9
6.3 Suchen eines Pfades	9
6.4 Algorithmus zum Prüfen der Erfüllbarkeit	9
6.5 Algorithmus zur Belegung einer 2-CNF	9

7 Verschiedenes

7.1 Potenzmenge	10
7.2 LongestCommonSubsequence	10
7.3 LongestCommonSubstring	10
7.4 LongestIncreasingSubsequence	11
7.5 Levenstein-Distanz	11
7.6 Permutation & Sequenzen	11
7.7 Knuth-Morris Pratt	12
7.8 PROPOSAL-Algorithmus (Stable Matching)	12
7.9 CYK-Algorithmus	13

8 Formatierung & Sonstiges

8.1 Ausgabeformatierung mit JAVA - DecimalFormat	13
8.2 Ausgabeformatierung mit printf	13
8.3 C++ Eingabe ohne bekannt Länge	14
8.4 Testskript	14

n	Runtime $100 \cdot 10^6$ in 3s
[10, 11]	$\mathcal{O}(n!)$
< 22	$\mathcal{O}(n2^n)$
≤ 100	$\mathcal{O}(n^4)$
≤ 400	$\mathcal{O}(n^3)$
≤ 2.000	$\mathcal{O}(n^2 \log n)$
≤ 10.000	$\mathcal{O}(n^2)$
$\leq 1.000.000$	$\mathcal{O}(n \log n)$
$\leq 100.000.000$	$\mathcal{O}(n)$

integer (32 Bit, signed):

-2.147.483.648 ... 2.147.483.647

long (64 Bit, signed):

-9.223.372.036.854.775.808 ... 9.223.372.036.854.775.807

MD5: cat <string> | tr -d [:space:] | md5sum

1 Mathematische Algorithmen

1.1 Primzahlen

Für Primzahlen gilt immer (aber nicht nur für Primzahlen)

$$a^p \equiv a \pmod{p} \quad \text{bzw.} \quad a^{p-1} \equiv 1 \pmod{p}.$$

1.1.1 Sieb des Eratosthenes $\mathcal{O}(n^2)$

```
1 static boolean[] sieve(int until) {
2     boolean[] a = new boolean[until + 1];
3     Arrays.fill(a, true); a[1]=false; a[0]=false;
4     for (int i = 2; i < Math.sqrt(a.length); i++) {
5         if (a[i]) {
6             for (int j = i * i; j < a.length; j += i) a[j] =
                false;
7         }
8     }
9     return a; // a[i] == true, iff. i is prime. a[0] is
                ignored
10 }
```

MD5: f2241e45384c9165389a8ef7eaffdb24

1.1.2 Primzahlentest

```
1 static boolean isPrim(int p) {
2     if (p < 2 || p > 2 && p % 2 == 0) return false;
3     for (int i = 3; i <= Math.sqrt(p); i += 2)
4         if (p % i == 0) return false;
5     return true;
6 }
```

MD5: ab672f1e03a3f839b6fb0d9b93dd21d0

1.2 Binomial Koeffizient

```
1 static int[][] mem = new int[MAX_N][(MAX_N + 1) / 2];
2 static int binoCo(int n, int k) {
3     if (k < 0 || k > n) return 0;
4     if (2 * k > n) binoCo(n, n - k);
5     if (mem[n][k] > 0) return mem[n][k];
6     int ret = 1;
7     for (int i = 1; i <= k; i++) {
8         ret *= n - k + i;
9         ret /= i;
10        mem[n][i] = ret;
11    }
12    return ret;
13 }
```

MD5: 3a459246143bbdc49336d77c9b2720e4

1.3 Eulersche φ -Funktion

$$\varphi(n \in \mathbb{N}) := |\{a \in \mathbb{N} | 1 \leq a \leq n \wedge \text{ggT}(a, n) = 1\}|$$

$$\varphi(n \cdot m) = \varphi(n) \cdot \varphi(m)$$

```
1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4 int phi(int);
5 int main(){
6     int n;
```

```
7     while((cin>>n)!=0) cout << phi(n) << endl;
8     return 0;
9 }
10
11 int phi(int n){
12     int coprime = 1;
13     int primes[] = {2,3,5,7,11,13}; //...
14     int primessizes = 6; //anpassen !
15     //zusätzlich Primfaktorzerlegung v. n
16     for(int i =0; i<primessizes; i++){
17         int anz = 0;
18         while(n % primes[i] == 0){
19             n = n / primes[i];
20             anz ++;
21             cout<<"p: " <<primes[i]<<endl;
22         }
23         if(anz>0)
24             coprime *= ((int) pow((double) primes[i],
25                 (double)(anz-1))*(primes[i] -
26 1));
27         if(n==1) break;
28     }
29     if(n != 1){
30         coprime *= (n - 1);
31     }
32     return coprime;
33 }
```

2 Mathematisch Formeln und Gesetze

2.1 Catalan

$$C_n = \frac{1}{n+1} \binom{2n}{n} = \prod_{k=2}^n (n+k)/k$$

$$C_{n+1} = \frac{4n+2}{n+2} C_n = \sum_{k=0}^n C_k C_{n-k}$$

$$C_0 = 1; C_n = \frac{2(2n-1)}{n+1} C_{n-1}$$

number of distinct binary trees with n vertices; number of expressions using n -pairs of correct placed parentheses; number of possible polygon triangulations.

2.2 kgV und ggT

$$\text{ggT}(n, m) \cdot \text{kgV}(m, n) = |m \cdot n|$$

```
int gcd(int a, int b){return (b==0)?a:gcd(b,a %b);}
int lcm(int a, int b){return a*(b/gcd(a,b));}
```

2.3 modulare Exponentiation

$$b^e \equiv c \pmod{m}$$

$$b^e = b^{(\sum_{i=0}^{n-1} a_i 2^i)} = \prod_{i=0}^{n-1} (b^{2^i})^{a_i}$$

```
1 function modular_pow(base, exponent, modulus)
2     result := 1
3     while exponent > 0
4         if (exponent mod 2 == 1):
5             result := (result * base) mod modulus
6             exponent := exponent >> 1
7             base = (base * base) mod modulus
8     return result
```

2.4 Modulare Arithmetik

Bedeutung der größten gemeinsamen Teiler ($[d = ggT(a, b), s, t] := \text{EEA}(a, b)$):

$$d = ggT(a, b) = as + bt.$$

Verwendung zur Berechnung des inversen Elements b^{-1} zu b bezüglich der Basis einer Restklassengruppe $a \in \mathbb{P}$ ($1 \equiv b^{-1}b \pmod{1}$):

$$d = 1 \Rightarrow 1 \equiv t \cdot b \pmod{a} \Rightarrow b^{-1} := t$$

$d \neq 1 \Rightarrow b^{-1}$ existiert nicht bzgl a, b .

2.4.1 Erweiterter Euklidischer Algorithmus

```
1 static int[] eea(int a, int b) {
2     int[] dst = new int[3];
3     if (b == 0) {
4         dst[0] = a;
5         dst[1] = 1;
6         return dst; // a, 1, 0
7     }
8     dst = eea(b, a % b);
9     int tmp = dst[2];
10    dst[2] = dst[1] - ((a / b) * dst[2]);
11    dst[1] = tmp;
12    return dst;
13 }
```

MD5: ec47623482e3cf5297ebe446e8eafd5

2.5 Kombinatorik

	mit ZL	ohne ZL
Variat.	n^k	$\frac{n!}{(n-k)!}$
Kombinat.	$\binom{n}{k} = \binom{n}{n-k} = \frac{n!}{k!(n-k)!}$	$\binom{n+k-1}{k} = \binom{n+k-1}{n-1}$

3 Datenstrukturen

3.1 Fenwick Tree (Binary Indexed Tree)

```
1 class FenwickTree {
2     private int[] values;
3     private int n;
4     public FenwickTree(int n) {
5         this.n = n;
6         values = new int[n];
7     }
8     public int get(int i) { //get value of i
9         int x = values[0];
10        while (i > 0) {
11            x += values[i];
12            i -= i & -i;
13        }
14        return x;
15    }
16    public void add(int i, int x) { // add x to interval [i, n]
17        if (i == 0) values[0] += x;
18        else {
19            while (i < n) {
20                values[i] += x;
21                i += i & -i;
22            }
23        }
24    }
25 }
```

```
22 }
23 }
```

MD5: da8d56a0188958c7d35409b7a6fb7a9c

3.2 Union-Find

```
1 int rank[MAX_N]; //upper bound on the length of the
2   path from the root to a leaf
3 int rep[MAX_N];
4 void makeSet(int x) {
5     rank[x] = 0;
6     rep[x] = x;
7 }
8 int findSet(int x) {
9     if (x != rep[x]) rep[x] = findSet(rep[x]); //path
10    compression
11    return rep[x];
12 }
13 void link(int x, int y) {
14     if (rank[x] > rank[y]) rep[y] = x; //join according
15     to rang
16     else {
17         if (rank[x] == rank[y]) rank[y]++;
18         rep[x] = y;
19     }
20 }
21 void unionSet(int x, int y) {
22     link(findSet(x), findSet(y));
23 }
```

4 Graphen

Graph $G = (V, E)$ mit Kanten E und Knoten V . i.A.: $n = |V(G)|, m = |E|$

Es gilt: $m = n - 1$ gdw. G Baum; $2 \mid \deg(v \in V)$ gdw. ex. Eulerkreis und G (stark, falls gerichtet) zusammenhängend.

4.1 planare Graphen

$|E| \leq 3|V| - 6$ (notwendige Bedingung) oder Eulersche Polyederformel $|V| + |F| - |E| = 2$

4.2 Topologische Sortierung

```
1 static List<Integer> topoSort(Map<Integer, List<
2   Integer>> edges,
3   Map<Integer, List<Integer>> revedges) {
4     Queue<Integer> q = new LinkedList<Integer>();
5     List<Integer> ret = new LinkedList<Integer>();
6     Map<Integer, Integer> indeg = new HashMap<Integer,
7   Integer>();
8     for (int v : revedges.keySet()) {
9         indeg.put(v, revedges.get(v).size());
10        if (revedges.get(v).size() == 0)
11            q.add(v);
12    }
13    while (!q.isEmpty()) {
14        int tmp = q.poll();
15        ret.add(tmp);
16        for (int dest : edges.get(tmp)) {
17            indeg.put(dest, indeg.get(dest) - 1);
18        }
19    }
20 }
```

```

16     if (indeg.get(dest) == 0)
17         q.add(dest);
18     }
19 }
20 return ret;
21 }

```

MD5: f89e486b31561403ed45869c9ca5b180

4.3 Prim (Minimum Spanning Tree)

```

1  #define WHITE 0
2  #define BLACK 1
3  #define INF INT_MAX
4
5  int baum( int **matrix, int N){
6      int i, sum = 0;
7
8      int color[N];
9      int dist[N];
10
11      // markiere alle Knoten ausser 0 als unbesucht
12      color[0] = BLACK;
13      for( i=1; i<N; i++){
14          color[i] = WHITE;
15          dist[i] = INF;
16      }
17
18      // berechne den Rand
19      for( i=1; i<N; i++){
20          if( dist[i] > matrix[i][nextIndex]){
21              dist[i] = matrix[i][nextIndex];
22          } }
23      while( 1){
24          int nextDist = INF, nextIndex = -1;
25
26          /* Den naechsten Knoten waehlen */
27          for(i=0; i<N; i++){
28              if( color[i] != WHITE) continue;
29
30              if( dist[i] < nextDist){
31                  nextDist = dist[i];
32                  nextIndex = i;
33              }
34          }
35
36          /* Abbruchbedingung */
37          if( nextIndex == -1) break;
38
39          /* Knoten in MST aufnehmen */
40          color[nextIndex] = RED;
41          sum += nextDist;
42
43          /* naechste kuerzeste Distanzen berechnen */
44          for( i=0; i<N; i++){
45              if( i == nextIndex || color[i] == BLACK )
46                  continue;
47
48              if( dist[i] > matrix[i][nextIndex]){
49                  dist[i] = matrix[i][nextIndex];
50              }
51          }
52      }
53      return sum;
54 }

```

4.4 Kruskal

```

1 public static LinkedList<Edge> kruskal(LinkedList<Edge>
2     > adjList, int root, int nodeCount) {
3     LinkedList<SortedSet<Integer>> branches = new
4         LinkedList<SortedSet<Integer>>();
5     for (int i = 0; i < nodeCount; i++) {
6         branches.add(new TreeSet<Integer>());
7         branches.get(branches.size() - 1).add(i);
8     }
9
10    PriorityQueue<Edge> edges = new PriorityQueue<Edge>
11        >(1, new Comparator<Edge>() {
12        @Override
13        public int compare(Edge e1, Edge e2) {
14            if (e1.weight <= e2.weight) {
15                return -1;
16            } else {
17                return 1;
18            }
19        });
20    edges.addAll(adjList);
21    LinkedList<Edge> result = new LinkedList<Edge>();
22
23    while (branches.size() > 1) {
24        Edge min = edges.remove();
25
26        SortedSet<Integer> from = null;
27        for (SortedSet<Integer> branchFrom : branches) {
28            if (branchFrom.contains(min.from)) {
29                if (!branchFrom.contains(min.to)) {
30                    from = branchFrom;
31                    break;
32                }
33            }
34        }
35
36        if (from != null) {
37            for (SortedSet<Integer> branchTo : branches) {
38                if (!(from.equals(branchTo))) {
39                    if (branchTo.contains(min.to)) {
40                        from.addAll(branchTo);
41                        branches.remove(branchTo);
42                        result.add(min);
43                        break;
44                    }
45                }
46            }
47        }
48        return result;
49    }

```

4.5 Floyd-Warshal ($\mathcal{O}(n^3)$)

```

1 for(int i = 0; i<n; i++)
2     for(int j = 0; j<n; j++)
3         if((i,j) ∈ E(G)){
4             d[i,j] = w[i,j];
5         }
6         else
7             d[i,j] = ∞
8     for(int k = 0; k<n; k++)
9         for(int i = 0; i<n; i++)
10            for(int j = 0; j<n; j++)
11                d[i,j] = min (d[i,j],d[i,k] + d[k,j]);

```

4.6 Dijkstra

- alle kürzesten Wege von einem Knoten aus in $\mathcal{O}(\#Kanten + \#Knoten)$

- negative Kanten:

- auf alle Kantengewichte $|min| + 1$ (damit 0 nicht entsteht)
- Kantenanzahl zum Ziel mitspeichern

$$\frac{\text{Weglänge}}{\text{Kantenanzahl} \cdot (|min| + 1)}$$

```

1 // look for shortest distance from a to b in adjacency
  matrix
2 // visited nodes for breadth first search
3 bool nodeVisited[26];
4 for (int k=0; k<26; k++) {
5     nodeVisited[k]=false;
6 }
7 queue<int> searchQueue;
8 queue<string> outputQueue;
9 searchQueue.push(aNumber); // start search from a
10 string start="";
11 start += a[0];
12 outputQueue.push(start);
13 string outputString;
14 while (searchQueue.empty()==false && nodeVisited[
    bNumber]==false) {
15     int node=searchQueue.front();
16     searchQueue.pop();
17     string nodeString=outputQueue.front();
18     outputQueue.pop();
19     for (int k=0; k<26; k++) {
20         if (cities[node][k]==true &&
            nodeVisited[k]==false) {
21             searchQueue.push(k);
22             nodeVisited[k]=true;
23             char addToOutput=k+'A';
24             string s=nodeString;
25             s += addToOutput;
26             outputQueue.push(s);
27             if (k==bNumber) {
28                 outputString=s;
29             }
30         }
31     }
32 }
33 cout << outputString << "\n";

```

4.7 Belman-Ford

```

1 procedure BellmanFord(list vertices, list edges,
    vertex source)
2 // This implementation takes in a graph,
    represented as lists of vertices
3 // and edges, and modifies the vertices so that
    their distance and
4 // predecessor attributes store the shortest paths.
5
6 // Step 1: initialize graph
7 for each vertex v in vertices:
8     if v is source then v.distance := 0
9     else v.distance := infinity
10    v.predecessor := null

```

```

// Step 2: relax edges repeatedly
for i from 1 to size(vertices)-1:
    for each edge uv in edges: // uv is the edge
        from u to v
        u := uv.source
        v := uv.destination
        if u.distance + uv.weight < v.distance:
            v.distance := u.distance + uv.weight
            v.predecessor := u

// Step 3: check for negative-weight cycles
for each edge uv in edges:
    u := uv.source
    v := uv.destination
    if u.distance + uv.weight < v.distance:
        error "Graph contains a negative-weight
            cycle"

```

4.8 MaxFlow

```

1 public class Flow {
2     static class Edge {
3         int c;
4         int f = 0;
5         Vertex s;
6         Vertex d;
7         Edge(int cap, Vertex source, Vertex dest) {
8             c = cap;
9             s = source;
10            d = dest;
11        }
12        int res(Vertex v) {
13            if (v == d) return f;
14            else return c - f;
15        }
16    }
17    static class Vertex {
18        List<Edge> lks = new ArrayList<Edge>();
19    }
20    static int maxFlow(Vertex so, Vertex si) {
21        ff: while (true) {
22            HashMap<Vertex, Edge> etp = new HashMap<Vertex,
                Edge>();
23            List<Vertex> fringe = new ArrayList<Vertex>();
24            fringe.add(so);
25            etp.put(so, null);
26            int minRes = Integer.MAX_VALUE;
27            boolean foundrp = false;
28            bfs: while (!fringe.isEmpty()) {
29                List<Vertex> newFringe = new ArrayList<Vertex>
                    >();
30                for (Vertex v : fringe) {
31                    for (Edge e : v.lks) {
32                        Vertex child = (e.d == v) ? e.s : e.d;
33                        if (!etp.containsKey(child) && e.res(v) >
                            0) {
34                            etp.put(child, e);
35                            newFringe.add(child);
36                            minRes = Math.min(minRes, e.res(v));
37                            if (child == si) {
38                                foundrp = true;
39                                break bfs;
40                            }
41                        }
42                    }
43                }
44                fringe = newFringe;
45            }
46            if (!foundrp) break ff;
47        }
48    }
49 }

```

```

44     Vertex nxt = si;
45     while (nxt != so) {
46         Vertex prv = nxt;
47         Edge edge = etp.get(prv);
48         if (edge.s == prv) {
49             edge.f = edge.f - minRes;
50             nxt = edge.d;
51         } else {
52             edge.f = edge.f + minRes;
53             nxt = edge.s;
54         }
55     }
56 }
57 int flow = 0;
58 for (Edge e : so.lks) {
59     flow += e.f;
60 }
61 return flow;
62 }
63 }

```

MD5: a29c73a7d958ca12f3778a65c39a2e3e

4.9 Bipartite Matching

```

1  import java.util.*;
2
3
4  public class BPM {
5      int m, n;
6      boolean[][] graph;
7      boolean seen[];
8      int matchL[]; //What left vertex i is matched
9                  //to (or -1 if unmatched)
10     int matchR[]; //What right vertex j is matched
11                  //to (or -1 if unmatched)
12
13     int maximumMatching() {
14         //Read input and populate graph[][]
15         //Set m to be the size of L, n to be the
16         //size of R
17         Arrays.fill(matchL, -1);
18         Arrays.fill(matchR, -1);
19
20         int count = 0;
21         for (int i = 0; i < m; i++) {
22             Arrays.fill(seen, false);
23             if (bpm(i)) count++;
24         }
25         return count;
26     }
27
28     boolean bpm(int u) {
29         //try to match with all vertices on right
30         //side
31         for (int v = 0; v < n; v++) {
32             if (!graph[u][v] || seen[v]) continue;
33             seen[v] = true;
34             //match u and v, if v is unassigned, or
35             //if v's match on the left side can be
36             //reassigned to another right vertex
37             if (matchR[v] == -1 || bpm(matchR[v])) {
38                 matchL[u] = v;
39                 matchR[v] = u;
40                 return true;
41             }
42         }
43     }
44 }

```

```

37     return false;
38 }
39
40 public void run(){
41
42     Scanner sc = new Scanner(System.in).useLocale(
43         Locale.US);
44     int T = sc.nextInt();
45     while(T-->0){
46         n = sc.nextInt();
47         m = sc.nextInt();
48         int K = sc.nextInt();
49         graph = new boolean [m][n];
50         matchL = new int[m];
51         matchR = new int[n];
52         seen = new boolean[n];
53         while(K-->0){
54             int y = (int)sc.nextDouble();
55             int x = (int)sc.nextDouble();
56             graph[x][y] = true;
57         }
58         System.out.println(maximumMatching());
59     }
60     sc.close();
61 }
62
63 public static void main(String[] args){
64     (new BPM()).run();
65 }
66 }

```

MD5: -----

4.10 Bitonic TSP

All nodes n_i are sorted in x -direction; $d(i, j)$ is the distance:

```

1  public static double bitonic(double[][] d) {
2      int N = d.length;
3      double[][] B = new double[N][N];
4      for (int j = 0; j < N; j++) {
5          for (int i = 0; i <= j; i++) {
6              if (i < j - 1)
7                  B[i][j] = B[i][j - 1] + d[j - 1][j];
8              else {
9                  double min = 0;
10                 for (int k = 0; k < j; k++) {
11                     double r = B[k][i] + d[k][j];
12                     if (min > r || k == 0)
13                         min = r;
14                 }
15                 B[i][j] = min;
16             }
17         }
18     }
19     return B[N-1][N-1];
20 }

```

MD5: 49fca508fb184da171e4c8e18b6ca4c7

4.11 Shortest Cycle

Ln. 22 prevents double edges and taking undirected edges backwards.

```

1  public int minCycle(int n, int m, ArrayList<
2      LinkedList<Integer>> adj){
3      int min = Integer.MAX_VALUE;
4      int[] length = new int[n];
5      int[] prev = new int[n];

```

```

5  for (int start = 0; start < n; start++) {
6      Arrays.fill(length, -1);
7      Arrays.fill(prev, -1);
8      Queue<Integer> queue = new LinkedList<Integer>();
9      queue.add(start);
10     length[start] = 0;
11     while (!queue.isEmpty()) {
12         int u = queue.poll();
13         if (2*length[u] >= min)
14             break;
15         for (int v : adj.get(u)) {
16             if (length[v] < 0) {
17                 length[v] = length[u] + 1;
18                 prev[v] = u;
19                 if (length[v] < min) {
20                     queue.add(v);
21                 }
22             } else if (prev[u] != v && prev[v] != u) {
23                 min = Math.min(length[v] + length[u] + 1,
24                               min);
25             }
26         }
27     }
28     return min;
29 }

```

MD5: bf74ce626179378dcc19a599f6d491d6

5 Geometrie

5.1 Kreuzprodukt, Skalarprodukt

$$\vec{a} \times \vec{b} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{pmatrix}, \langle a, b \rangle = \sum a_i b_i = |\vec{a}| |\vec{b}| \cos(\angle(a, b))$$

5.2 Orthogonale Projektion

r_0 : Ortsvektor; u : Richtungsvektor; n : Normalenvektor

$$P_g(\vec{x}) = \vec{r}_0 + \frac{(\vec{x} - \vec{r}_0) \cdot \vec{u}}{\vec{u} \cdot \vec{u}} \vec{u}$$

$$P_g(\vec{x}) = \vec{x} - \frac{(\vec{x} - \vec{r}_0) \cdot \vec{n}}{\vec{n} \cdot \vec{n}} \vec{n} \text{ (nur 2D bzw. 3D auf Ebene)}$$

5.3 Rotation

```

1  static Point rotate(Point v, double a) {
2      double cos = Math.cos(a);
3      double sin = Math.sin(a);
4      double x = cos * v.x - sin * v.y;
5      double y = sin * v.x + cos * v.y;
6      return new Point(x, y);
7  }

```

5.4 Geradenschnittpunkt

$$g_1 : ax + by = c; g_2 : px + qx = r;$$

$$\Rightarrow \vec{p} = \frac{1}{aq - bp} \begin{pmatrix} x = cq - br \\ y = ar - cp \end{pmatrix}$$

$$g_1 : \vec{p} = \begin{pmatrix} r_x \\ r_y \end{pmatrix} + s \begin{pmatrix} s_x \\ s_y \end{pmatrix} \quad g_2 : \vec{p} = \begin{pmatrix} q_x \\ q_y \end{pmatrix} + t \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

$$w_x = (r_x - q_x), w_y = (r_y - q_y)$$

$$\Rightarrow D = (s_x t_y - t_x s_y), D_s = (t_x w_y - t_y w_x), D_t = (s_y w_x - s_x w_y); s = D_s / D, t = D_t / D$$

5.5 Zusammenhang Kreuzprodukt & Sinus

$$|\vec{a} \times \vec{b}| = |\vec{a}| |\vec{b}| \sin \angle(\vec{a}, \vec{b})$$

5.6 Dreiecksfläche

$$F = \sqrt{s(s-a)(s-b)(s-c)}; s = \frac{a+b+c}{2}$$

5.7 Graham Scan (Convex Hull)

```

1  public static class Point implements Comparable<
2      Point> {
3      double x, y, r;
4      Point p0;
5      public Point(double x, double y) {
6          this.x = x;
7          this.y = y;
8      }
9      public int compareTo(Point p) {
10         double s = ccw(p0, p, this);
11         if (s != 0) return (int) Math.signum(s);
12         else return (int) Math.signum(p.r - r);
13     }
14     public static double dist(Point a, Point b) {
15         double x = a.x - b.x;
16         double y = a.y - b.y;
17         return Math.sqrt(x * x + y * y);
18     }
19     public static double ccw(Point a, Point b, Point c) {
20         {
21             return (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (
22                 c.x - a.x);
23         }
24     }
25     static List<Point> graham(List<Point> P) {
26         Point p0 = P.get(0);
27         for (int i = 1; i < P.size(); i++) {
28             Point p = P.get(i);
29             if (p.y < p0.y || (p.y == p0.y && p.x < p0.x)) {
30                 p0 = p;
31             }
32         }
33         P.remove(p0);
34         for (Point p : P) {
35             p.r = dist(p0, p);
36             p.p0 = p0;
37         }
38         Collections.sort(P);
39         Iterator<Point> I = P.iterator();
40         Point f = I.next();
41         while (I.hasNext()) {
42             Point p = I.next();
43             if (ccw(p0, p, f) == 0) {
44                 I.remove();
45             } else {
46                 f = p;
47             }
48         }
49         LinkedList<Point> S = new LinkedList<Point>();
50         if (P.isEmpty()) {
51             S.add(p0);
52         } else {
53             S.push(p0);
54             S.push(P.get(0));
55         }
56     }

```



```

49     for (int i = 1; i < P.size(); i++) {
50         Point b = S.pop();
51         Point a = S.peek();
52         S.push(b);
53         while (ccw(a, b, P.get(i)) <= 0) {
54             S.pop();
55             b = S.pop();
56             a = S.peek();
57             S.push(b);
58         }
59         S.push(P.get(i));
60     } }
61     return S;
62 }

```

MD5: fa3b15e54ec7447485870a1978f8aac4

5.8 Line Intersection

- **Mehr als 2 Linien:**
- findet nicht alle Intersection Points, aber immer wenn einer existiert, dann angegeben
- $O(n \log n + l \log n)$
- **2 Linien:**
- line intersection (test if possible!)
- Achtung: beide Reihenfolgen testen: if ((checkLines(readLines[j],newLine) == true) && (checkLines(newLine,readLines[j]) == true))

```

1 struct line {
2     int x0;
3     int y0;
4     int x1;
5     int y1;
6 };
7
8 // prueft, ob sich die Linien schneiden koennen
9 bool checkLines(line a, line b) {
10     // Vektor Linie a
11     int x0 = a.x1 - a.x0;
12     int y0 = a.y1 - a.y0;
13     // Vektor zu Startpunkt b
14     int x1 = b.x0 - a.x0;
15     int y1 = b.y0 - a.y0;
16     // Vektor zu Endpunkt b
17     int x2 = b.x1 - a.x0;
18     int y2 = b.y1 - a.y0;
19     // Kreuzprodukte berechnen
20     int crossProduct1 = x0 * y1 + x1 * y0;
21     int crossProduct2 = x0 * y2 + x2 * y0;
22     // Wenn ein Produkt negativ, das andere positiv ist
23     // , koennen sich die Linien schneiden
24     if (crossProduct1 * crossProduct2 < 0) {
25         return true;
26     }
27     return false;
28 }

```

5.9 Punkt in Polygon

KreuzProdTest: -1: $A \rightarrow R$ schneidet BC (ausser unterer Endpunkt); 0: A auf BC ; +1: sonst

PiP: Input: $P[i]$ ($x[i], y[i]$); $P[0]=P[n]$; Output: -1: Q außerhalb Polygon, 0: Q auf Polygon, +1: Q innerhalb des Polygons

```

1 public static int KreuzProdTest(double ax, double ay
2     , double bx, double by,
3     double cx, double cy) {
4     if (ay == by && by == cy) {
5         if ((bx <= ax && ax <= cx) || (cx <= ax && ax <=
6             bx))
7             return 0;
8         else
9             return +1;
10    }
11    if (by > cy) { double tmpx=bx; double tmpy=by; bx=cx; by=
12        cy; cx=tmpx; cy=tmpy; }
13    if (ay == by && ax == bx) return 0;
14    if (ay <= by || ay > cy) return +1;
15    double delta = (bx-ax)*(cy-ay)-(by-ay)*(cx-ax);
16    if (delta > 0) return -1; else if (delta < 0) return +1;
17    else return 0;
18 }
19 public static int PunktInPoly(double[] x, double[] y,
20     double qx, double qy) {
21     int n = x.length - 1;
22     int t = -1;
23     for (int i = 0; i <= n - 1; i++) {
24         t = t * KreuzProdTest(qx, qy, x[i], y[i], x[i +
25             1], y[i + 1]); }
26     return t;
27 }

```

MD5: 38a79d6979334bc6a01381e15eef6e04

5.10 Fläche eines Polygons

Input: Polygon-Koordinaten sortiert im Uhrzeigersinn

```

1 static double area(List<Point> p) {
2     double a = 0;
3     Point q = p.get(p.size() - 1);
4     Point r;
5     for (Point r : p) {
6         a += (q.x + r.x) * (q.y - r.y);
7         q = r;
8     }
9     return a / -2;
10 }

```

MD5: 1f1dbdaaf78726c57e3e0ece63fe1cb3

6 2-SAT-Solver

6.1 2-Sat mit SCC

```

1 public class D_Manha {
2     static class Node {
3         ArrayList<Node> out = new ArrayList<Node>();
4         ArrayList<Node> in = new ArrayList<Node>();
5         int var;
6         boolean explored = false;
7         boolean discovered = false;
8         int CCC;

```



```

9     public Node(int v, String n) {
10         var = v;
11         name = n;
12     }
13 }
14 static void impl(Node x, Node y){
15     x.out.add(y);
16     y.in.add(x);
17 }
18 public static void main(String[] args) {
19     Scanner in = new Scanner(System.in);
20     int n = in.nextInt();
21     while (n-- > 0) {
22         ArrayList<Node> graph; //TODO :
23             implikationsgraph
24         // Kosaraju
25         S = new ArrayList<Node>();
26         for (Node v : graph) {
27             if (!v.explored) {
28                 DFS(v);
29             }
30         }
31         for (Node v : graph) {
32             v.explored = false;
33             v.discovered = false;
34         }
35         int CCCidx = 0;
36         do {
37             ArrayList<Node> CCC = new ArrayList<Node>();
38             DFSTrans(S.get(S.size()-1), CCC, CCCidx++);
39             S.removeAll(CCC);
40         } while (!S.isEmpty());
41
42         boolean possible = true;
43         for (int i = 1; i <= s; i++) {
44             if (st.get(i).CCC == sf.get(i).CCC) {
45                 possible = false;
46             }
47         }
48         for (int i = 1; i <= a; i++) {
49             if (at.get(i).CCC == af.get(i).CCC) {
50                 possible = false;
51             }
52         }
53         if (possible) {
54             System.out.println("Yes");
55         } else {
56             System.out.println("No");
57         }
58     }
59 }
60 static ArrayList<Node> S;
61 public static void DFS(Node v) {
62     v.discovered = true;
63     for (Node u : v.out) {
64         if (!u.discovered) {
65             DFS(u);
66         }
67     }
68     v.explored = true;
69     S.add(v);
70 }
71 public static void DFSTrans(Node v, ArrayList<Node>
72     CCC, int CCCidx) {
73     v.discovered = true;
74     for (Node u : v.in) {
75         if (!u.discovered) {
76             DFSTrans(u, CCC, CCCidx);
77         }
78     }
79     v.explored = true;
80     CCC.add(v);
81     v.CCC = CCCidx;

```

```

75     }}

```

6.2 Hilfsalgorithmen

6.2.1 Erzeugen eines Graphens

```

1 SAT2Graph( $\varphi = (\alpha_1 \vee \beta_1) \wedge \dots \wedge (\alpha_m, \beta_m)$ ) {
2     G: Graph als Adjazenzliste
3     for(int i = 0 < m; i++){
4         jede Klausel liefert zwei Implikationen
5         Fuege Kanten  $(-\alpha_i, \beta_i), (-\beta_i, \alpha_i)$  zu G hinzu.
6     }

```

6.2.2 Indexumrechnung

```

1 /** rechnet den Index fure den Array Zugriff um */
2 idx(int i) := n + i + ((i > 0) ? (-1) : 0)

```

6.3 Suchen eines Pfades

```

1 /**
2 Prueft mithilfe einer Breitensuche ob ein Weg
3 von Knoten x nach -x existiert
4 */
5 boolean BFSSATCheck(SATGraph G, int x) {
6     boolean[] seen = new boolean[2 * n];
7     Queue<Integer> queue;
8     queue.add(x); seen[idx(x)] = true;
9     while (!queue.isEmpty()) {
10         Integer q = queue.poll();
11         for (Integer p : G.get(idx(q))) {
12             if (!seen[idx(p)]) {
13                 queue.add(p);
14                 seen[idx(p)] = true;
15             }
16             if (p == -x) return true;
17         }
18     }
19     return seen[idx(-x)];
20 }

```

6.4 Algorithmus zum Prüfen der Erfüllbarkeit

```

1 /**
2 Prueft ob fuer eine 2-CNF eine Belegung existiert
3 */
4 boolean SAT2Check( $\varphi = (\alpha_1 \vee \beta_1) \wedge \dots \wedge (\alpha_n, \beta_n)$ ) {
5     SAT2Graph G( $\varphi$ ) erzeugen
6     for(int i = 0 < n; i++)
7         if(BFSSATCheck(G,i) && BFSSATCheck(G,-i))
8             return false;
9     //Es gibt einen i → -i und -i → i Weg
10    return true;
11 }

```

6.5 Algorithmus zur Belegung einer 2-CNF

```

1 /**
2 * Ermittelt falls moeglich eine gueltige Belegung fuer
3 eine 2-CNF
4 */
5 Solve2SAT( $\varphi = (\alpha_1 \vee \beta_1) \wedge \dots \wedge (\alpha_n, \beta_n)$ ) {
6     SAT2Graph G( $\varphi$ ) erzeugen

```

```

6  vars = [0,...,0] //(2*n) Variablenbelegung
7  assigned = [false,...,false] //(n+1) Belegung
   zugewiesen?
8
9  for (int x = 1; x < n + 1; x++) {
10     oldVars = vars.clone();oldAssigned = assigned.
        clone();
11     if (assign(vars, assigned, x))
12         continue; //x:=1
13
14     vars = oldVars;assigned = oldAssigned;
15     if (!assign(vars, assigned, -x))
16         return null;//x:=0 liefert auch keine Loesung
17 }
18 return vars; //gueltige Belegung
19 }
20
21 /**
22  * Belegt die Variable x mit 1 und liefert false,
23  * falls dies nicht moeglich ist
24  * WICHTIG: Parameter werden veraendert (Referenzen
   uebergeben!).
25  */
26 boolean assign(ArrayList<Integer> vars,
27     ArrayList<Boolean> assigned, int x) {
28     int xi = (x < 0) ? -x : x;
29     if (assigned[xi]) return (vars[idx(x)] == 1);
30     //Belege x, -x mit 0,1:
31     vars[idx(x)]= 1;vars[idx(-x)]=0;
32     assigned[xi] = true;
33     for (Integer k : G.get(idx(x)))
34         if (!assign(vars, assigned, k)) {
35             //Belegung nicht weiter moeglich
36             assigned[xi] = false; return false;
37         }
38     return true;
39 }

```

7 Verschiedenes

7.1 Potenzmenge

```

1  static <T> Iterator<List<T>> powerSet(final List<T> l)
   {
2  return new Iterator<List<T>>() {
3      int i; // careful: i becomes 2^l.size()
4      public boolean hasNext() {
5          return i < (1 << l.size());
6      }
7      public List<T> next() {
8          Vector<T> temp = new Vector<T>();
9          for (int j = 0; j < l.size(); j++)
10             if (((i >>> j) & 1) == 1)
11                 temp.add(l.get(j));
12             i++;
13             return temp;
14         }
15     public void remove() {}
16 }
17 }

```

7.2 LongestCommonSubsequence

```

1 #include <iostream>

```

```

2 #include <vector>
3 #include <string>
4 #include <sstream>
5 #include <algorithm>
6 #include <iterator>
7 using namespace std;
8 #define MAX(a,b) (a > b) ? a : b
9
10 string X,Y;
11 vector< vector<int> > c(101, vector<int>(101,0));
12 int m,n,ctr;
13
14 int LCS(){
15     m = X.length(),n=Y.length();
16     c.resize(m+1);
17     for(int i = 0; i<n+1; i++) {
18         c[i].resize(n+1);
19         c[i][0] = 0;
20     }
21     int i,j;
22     for (i=0;i<=m;i++)
23         for (j=0;j<=n;j++)
24             c[i][j]=0;
25
26     for (i=1;i<=m;i++)
27         for (j=1;j<=n;j++)
28         {
29             if (X[i-1]==Y[j-1])
30                 c[i][j]=c[i-1][j-1]+1;
31             else
32                 c[i][j]=max(c[i][j-1],c[i-1][j]);
33         }
34     return c[m][n];
35 }
36 /** Print a single LCS */
37 void printLCS(int i,int j){
38     if (i==0 || j==0)
39         return;
40     if (X[i-1]==Y[j-1])
41     {
42         printLCS(i-1,j-1);
43         cout<<X[i-1];
44     }
45     else if (c[i][j]==c[i-1][j])
46         printLCS(i-1,j);
47     else
48         printLCS(i,j-1);
49 }
50
51 int main(){
52     while(cin>>X>>Y) {
53         cout << "Length:_" << LCS() << endl;
54         printLCS(m,n);
55         cout<<endl ;
56     }}

```

7.3 LongestCommonSubstring

```

1 private static List<String> longestCommonSubstring(
   String S1, String S2)
2 {
3     List<String> ret = new ArrayList<String>();
4     List<Integer> idx =new ArrayList<Integer>();
5     int Start = 0;
6     int Max = 0;
7     for (int i = 0; i < S1.length(); i++)
8     {

```

```

9         for (int j = 0; j < S2.length(); j++)
10         {
11             int x = 0;
12             while (S1.charAt(i + x) == S2.charAt(j +
13                 x))
14             {
15                 x++;
16                 if (((i + x) >= S1.length()) || ((j
17                     + x) >= S2.length())) break;
18             }
19             if (x > Max)
20             {
21                 Max = x;
22                 Start = i;
23                 idx.clear();
24                 idx.add(Start);
25             } else if (x == Max) {
26                 Start = i;
27                 idx.add(Start);
28             }
29         }
30         HashSet<String> set = new HashSet<String>(idx.
31             size(), 1f);
32         for (Integer start : idx) {
33             String substr = S1.substring(start, start + Max);
34             if (!set.contains(substr)) {
35                 ret.add(substr);
36                 set.add(substr);
37             }
38         }
39         Collections.sort(ret);
40         //return S1.substring(Start, (Start + Max));
41         return ret;
42     }
43 }

```

7.4 LongestIncreasingSubsequence

```

1 #include <vector>
2 using namespace std;
3
4 /** finde LIS in O(n log k)
5  *a: Sequenz (in)
6  *b: LIS (out)
7  */
8 void find_lis(vector<int> &a, vector<int> &b)
9 {
10     vector<int> p(a.size());
11     int u, v;
12     if (a.empty()) return;
13     b.push_back(0);
14
15     for (size_t i = 1; i < a.size(); i++)
16     {
17         // ist naechstes Element a[i] groesser als
18         // letztes der aktuelle LIS
19         // a[b.back()], fuege es (Index) an "b" an.
20         if (a[b.back()] < a[i]) {
21             p[i] = b.back();
22             b.push_back(i);
23             continue;
24         }
25
26         // finde kleinstes El. in LIS (index in b)
27         // welches gerade groesser als a[i] ist
28         // binaere suche |b|<=k => O(log k)
29         for (u = 0, v = b.size() - 1; u < v;)

```

```

28     {
29         int c = (u + v) / 2;
30         if (a[b[c]] < a[i]) u = c + 1; else v = c;
31     }
32
33     // aktualisiere b falls neuer Wert kleiner als
34     // vorheriger kleinerer Wert
35     if (a[i] < a[b[u]])
36     {
37         if (u > 0) p[i] = b[u - 1];
38         b[u] = i;
39     }
40 }
41
42 for (u = b.size(), v = b.back(); u--; v = p[v]) b[u]
43     = v;
44 }
45
46 #include <cstdio>
47 int main()
48 {
49     int a[] = { 1, 9, 3, 8, 11, 4, 5, 6, 4, 19, 7, 1, 7
50 };
51 vector<int> seq(a, a + sizeof(a) / sizeof(a[0])); // seq
52     : Eingabesequent
53 vector<int> lis; // lis
54     : Index Vektor fuer LIS
55 find_lis(seq, lis);
56 //Sequenz ausgeben:
57 for (size_t i = 0; i < lis.size(); i++)
58     printf("%d_", seq[lis[i]]);
59     printf("\n");
60
61 return 0;
62 }

```

7.5 Levenstein-Distanz

```

1 public static int lev(String a, String b) {
2     int n = a.length(); int m = b.length();
3     int[][] distance = new int[n + 1][m + 1];
4
5     for (int i = 0; i <= n; i++)
6         distance[i][0] = i;
7     for (int j = 1; j <= m; j++)
8         distance[0][j] = j;
9
10    for (int i = 1; i <= n; i++)
11        for (int j = 1; j <= m; j++)
12            distance[i][j] = min(
13                distance[i - 1][j] + 1,
14                distance[i][j - 1] + 1,
15                distance[i - 1][j - 1] + ((a.
16                    charAt(i - 1) == b.charAt(
17                        j - 1)) ? 0 : 1));
18    return distance[n][m];
19 }

```

MD5: 0ce8646f55ffe178751ea0debc9739dd

7.6 Permutation & Sequenzen

```

1 import java.util.Scanner;
2 public class PermsAndSequ {
3     public static void main(String[] args) {
4         Scanner sc = new Scanner(System.in);

```

```

5     int n;
6     while ((n = sc.nextInt()) != 0) {
7         int k = sc.nextInt();
8         Sequences(n, k);
9         Permutations(n);
10    }
11 }

12
13 public static void Sequences(int n, int k) {
14     int[] x = new int[k];
15     for (int i = 0; i < k; i++)
16         x[i] = 1;
17     Print(x);
18     while (true) {
19         boolean lastX = true;
20         for (int i = 0; i < k; i++)
21             if (x[i] != n) {
22                 lastX = false;
23                 break;
24             }
25         if (lastX)
26             break;
27         int p = k - 1;
28         while (!(x[p] < n))
29             p--;
30         x[p] = x[p] + 1;
31         for (int i = p + 1; i < k; i++)
32             x[i] = 1;
33         Print(x);
34     }
35 }

36 public static void Permutations(int n) {
37     int[] x = new int[n];
38     for (int i = 0; i < n; i++)
39         x[i] = i + 1;
40     Print(x);
41     while (true) {
42         boolean lastX = true;
43         for (int i = 0; i < n - 1; i++)
44             if (x[i] < x[i + 1]) {
45                 lastX = false;
46                 break;
47             }
48         if (lastX) break;
49         int k = n - 1 - 1;
50         while (x[k] > x[k + 1]) k--;
51         int t = k + 1;
52         while (t < (n - 1) && x[t + 1] > x[k])
53             t++;
54         int tmp = x[k];
55         x[k] = x[t];
56         x[t] = tmp;
57         // reverse x[k+1] ... x[n-1]
58         for (int i = 0; i <= ((n - 1) - (k + 1)) / 2; i++) {
59             tmp = x[k + 1 + i];
60             x[k + 1 + i] = x[n - 1 - i];
61             x[n - 1 - i] = tmp;
62         }
63         Print(x);
64     }
65 }

66 public static void Print(int[] x) {
67     for (int i = 0; i < x.length; i++)
68         System.out.print(x[i] + " ");
69     System.out.println("");
70 }
71

```

```

72 }

```

7.7 Knuth-Morris Pratt

Finds the first occurrence of the pattern in the text.

```

1 int match(String text, String pattern, int[] jump) {
2     int j = 0;
3     if (text.length() == 0)
4         return -1;
5     for (int i = 0; i < text.length(); i++) {
6         while (j > 0 && pattern.charAt(j) != text.charAt(i))
7             j = jump[j - 1];
8         if (pattern.charAt(j) == text.charAt(i))
9             j++;
10        if (j == pattern.length())
11            return i - pattern.length() + 1;
12    }
13    return -1;
14
15    // Computes the jump function
16    int[] computeJump(String pattern) {
17        int[] jump = new int[pattern.length()];
18        int j = 0;
19        for (int i = 1; i < pattern.length(); i++) {
20            while (j > 0 && pattern.charAt(j) != pattern.
                charAt(i))
21                j = jump[j - 1];
22            if (pattern.charAt(j) == pattern.charAt(i))
23                j++;
24            jump[i] = j;
25        }
26        return jump;
27    }
28 }

```

MD5: b5b9ca67a1df2c7c2913615bf1ed8a5b

7.8 PROPOSAL-Algorithmus (Stable Matching)

Laufzeit: $\Theta(n \log n), \mathcal{O}(n^2)$

Input: Menge $J = \{j_1, j_2, \dots, j_n\}$, $M = \{m_1, m_2, \dots, m_n\}$,
Ordnung $O(x)$ auf J bzw. M je $m \in M$ bzw. $j \in J$:

$O(m_i) : j_{i_1} > j_{i_2} > \dots > j_{i_n}$

Output: Stabiles Matching kein Paar (j_1, j_2) oder (m_1, m_2) vorhanden, sodass Partnertausch für das Paar die Präferenz verbessern würde. Ex. im $K_{n,n}$ immer.

1. Solange $\exists j \in J : j$ ist ungepaart.
2. Wähle zufällig $j \in J$ ohne Partner.
3. Für jedes $m \in M$ bzgl. $O(j)$ absteigend, das noch nicht vorgeschlagen wurde.
4. Lass j m fragen, ob sie einwilligt
5. Falls m hat keinen Partner
6. verheirate (j, m)
7. Sonst
8. Falls j beliebter ist als Partner j_m von m bzgl. $O(m)$ (bessere Wahl):
9. verheirate (j, m)
10. gib j_m frei

7.9 CYK-Algorithmus

Wortproblem für kontextfreie Sprachen mit Grammatik G . Produktionsregeln P z. B. : $S \rightarrow AB|BC$; $A \rightarrow BA|a$

Input: Produktionsregeln P , Startsymbol S ,

Wort $w = w_1 \dots w_n$

Output: $w \in L(G)$ bzw. $w \notin L(G)$ 1. Für $i = 1..n$

2. Für alle $(l \rightarrow r) \in P$

3. Falls $r = w_i$

4. $V_{i,1} := V_{i,1} \cup \{l\}$

5. Für $j = 2..n$

6. Für $i = 1..n - j + 1$

7. Für $k = 1..j - 1$

9. $V_{i,j} := V_{i,j} \cup \{l \mid (BC \rightarrow l) \in P, B \in V_{i,k}, C \in V_{i+k,j-k}\}$

10. Falls $S \in V_{1,n}$

11. $w \in L(G)$

12. Sonst $w \notin L(G)$

8 Formatierung & Sonstiges

8.1 Ausgabeformatierung mit JAVA - DecimalFormat

Symbol	Bedeutung
0	(Ziffer) – unbelegt wird eine Null angezeigt. (0.234=(00.00)=>00.23)
#	(Ziffer) – unbelegt bleibt leer, (keine unnötigen nullen).
.	Dezimaltrenner.
,	Gruppert die Ziffern (eine Gruppe ist so groß wie der Abstand von ",ßu ".).
;	Trennzeichen. Links Muster für pos., rechts für neg. Zahlen
-	Das Standardzeichen für Negativpräfix
%	Prozentwert.
‰	Promille.
X	Alle anderen Zeichen X können ganz normal benutzt werden.
'	Ausmarkieren von speziellen Symbolen im Präfix oder Suffix

8.2 Ausgabeformatierung mit printf

%d %i Decimal signed integer.

%o Octal int.

%x %X Hex int.

%u Unsigned int.

%c Character.

%s String. siehe unten.

%f double

%e %E double.

%g %G double.

- linksbündig.

0 Felder mit 0 ausfüllen
(an Stelle von Leerzeichen).

+ Vorzeichen immer ausgeben.

blank pos. Zahlen mit Leerzeichen beg.

verschiedene Bedeutung:

%#o (Oktal) 0 Präfix wird eingefügt.

%#x (Hex) 0x Präfix bei !=0

%#X (Hex) 0X Präfix bei !=0

%#e Dezimalpunkt immer anzeigen.

%#E Dezimalpunkt immer anzeigen.

%#f Dezimalpunkt immer anzeigen.

%#g

%#G Dezimalpunkt immer anzeigen.

Nullen nach Dzpkt. bleiben

```

1 int i = 123;
2 printf( "%d|_|_|%d\n" , i, -i); // |123| |-123|
3 printf( "%5d|_|%5d\n" , i, -i); // | 123| | -123|
4 printf( "%-5d|_|%-5d\n" , i, -i); // |123| | |-123|
5 printf( "%+-5d|_|%+-5d\n" , i, -i); // |+123| | |-123|
6 printf( "%05d|_|%05d\n\n", i, -i); // |00123| |-0123|
7 printf( "%X|_|%x\n", 0xabc, 0xabc ); // |ABC| |abc|
8 printf( "%08x|_|%#x\n", 0xabc, 0xabc ); // |00000abc| |0xabc|
9 double d = 1234.5678;
10 printf( "%f|_|%f\n" , d, -d); // |1234,567800| |-1234,567800|
11 printf( "%1.2f|_|%1.2f\n" , d, -d); // |1234,57| |-1234,57|
12 printf( "%10f|_|%10f\n" , d, -d); // |1234,567800| |-1234,567800|
13 printf( "%10.2f|_|%10.2f\n" , d, -d); // | 1234,57| | -1234,57|
14 printf( "%010.2f|_|%010.2f\n",d, -d); // |0001234,57| |-001234,57|

```

```

15 String s = "Monsterbacke";
16 printf( "\n%s\n", s );           // |Monsterbacke|
17 printf( "%20s\n", s );           // |           Monsterbacke|
18 printf( "%-20s\n", s );          // |Monsterbacke           |
19 printf( "%7s\n", s );             // |Monsterbacke|
20 printf( "%.7s\n", s );            // |Monster|
21 printf( "%20.7s\n", s );          // |           Monster|

```

8.3 C++ Eingabe ohne bekannt Länge

```

1 #include <iostream>
2 #include <sstream>
3 #include <istream>
4 #include <string>
5 #include <vector>
6 #include <cstdlib>
7
8 using namespace std;
9 int main(){
10     string s;
11     do{
12         getline(cin,s);
13         istringstream* ss;
14         ss = new istringstream( s );
15         while (!ss->eof())
16         {
17             string xs;
18             getline( *ss, xs, '_' ); // try to read the next field into it
19
20             int x = atoi(xs.c_str());
21             cout<<"_"<<xs;
22         }
23         cout<<endl;
24     } while(!cin.eof());
25 }

```

8.4 Testskript

Testscript <ProblemClass>:

```

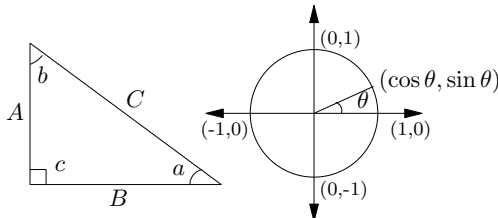
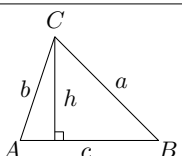
1 #!/bin/bash
2 homefolder="/home/team"
3 folder="$homefolder/test-session"
4 cd $homefolder/workspace/$1/src/
5 javac -encoding UTF-8 -d . "$1.java"
6 cat $folder/$1/sample.in | java -Xrs -Xss8m -Xmx1272864k $1 > $folder/$1/myout.out
7 diff -u $folder/$1/sample.out $folder/$1/myout.out

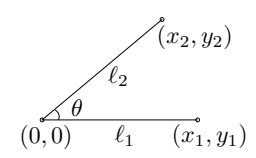
```


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.	$\sum_{i=1}^n H_i = (n+1)H_n - n, \quad \sum_{i=1}^n \binom{i}{m} H_i = \binom{n+1}{m+1} \left(H_{n+1} - \frac{1}{m+1} \right).$
$\left[\begin{smallmatrix} n \\ k \end{smallmatrix} \right]$	Stirling numbers (1st kind): Arrangements of an n element set into k cycles.	1. $\binom{n}{k} = \frac{n!}{(n-k)!k!}, \quad 2. \sum_{k=0}^n \binom{n}{k} = 2^n, \quad 3. \binom{n}{k} = \binom{n}{n-k},$
$\left\{ \begin{smallmatrix} n \\ k \end{smallmatrix} \right\}$	Stirling numbers (2nd kind): Partitions of an n element set into k non-empty sets.	4. $\binom{n}{k} = \frac{n}{k} \binom{n-1}{k-1}, \quad 5. \binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1},$
$\langle \begin{smallmatrix} n \\ k \end{smallmatrix} \rangle$	1st order Eulerian numbers: Permutations $\pi_1 \pi_2 \dots \pi_n$ on $\{1, 2, \dots, n\}$ with k ascents.	6. $\binom{n}{m} \binom{m}{k} = \binom{n}{k} \binom{n-k}{m-k}, \quad 7. \sum_{k=0}^n \binom{r+k}{k} = \binom{r+n+1}{n},$
$\langle\langle \begin{smallmatrix} n \\ k \end{smallmatrix} \rangle\rangle$	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. $\binom{n}{k} = (-1)^k \binom{k-n-1}{k}, \quad 11. \left\{ \begin{smallmatrix} n \\ 1 \end{smallmatrix} \right\} = \left\{ \begin{smallmatrix} n \\ n \end{smallmatrix} \right\} = 1,$
14. $\left[\begin{smallmatrix} n \\ 1 \end{smallmatrix} \right] = (n-1)!,$	15. $\left[\begin{smallmatrix} n \\ 2 \end{smallmatrix} \right] = (n-1)!H_{n-1},$	16. $\left[\begin{smallmatrix} n \\ n \end{smallmatrix} \right] = 1, \quad 17. \left[\begin{smallmatrix} n \\ k \end{smallmatrix} \right] \geq \left\{ \begin{smallmatrix} n \\ k \end{smallmatrix} \right\},$
18. $\left[\begin{smallmatrix} n \\ k \end{smallmatrix} \right] = (n-1) \left[\begin{smallmatrix} n-1 \\ k \end{smallmatrix} \right] + \left[\begin{smallmatrix} n-1 \\ k-1 \end{smallmatrix} \right],$	19. $\left\{ \begin{smallmatrix} n \\ n-1 \end{smallmatrix} \right\} = \left[\begin{smallmatrix} n \\ n-1 \end{smallmatrix} \right] = \binom{n}{2},$	20. $\sum_{k=0}^n \left[\begin{smallmatrix} n \\ k \end{smallmatrix} \right] = n!, \quad 21. C_n = \frac{1}{n+1} \binom{2n}{n},$
22. $\langle \begin{smallmatrix} n \\ 0 \end{smallmatrix} \rangle = \langle \begin{smallmatrix} n \\ n-1 \end{smallmatrix} \rangle = 1,$	23. $\langle \begin{smallmatrix} n \\ k \end{smallmatrix} \rangle = \langle \begin{smallmatrix} n \\ n-1-k \end{smallmatrix} \rangle,$	24. $\langle \begin{smallmatrix} n \\ k \end{smallmatrix} \rangle = (k+1) \langle \begin{smallmatrix} n-1 \\ k \end{smallmatrix} \rangle + (n-k) \langle \begin{smallmatrix} n-1 \\ k-1 \end{smallmatrix} \rangle,$
25. $\langle \begin{smallmatrix} 0 \\ k \end{smallmatrix} \rangle = \begin{cases} 1 & \text{if } k=0, \\ 0 & \text{otherwise} \end{cases}$	26. $\langle \begin{smallmatrix} n \\ 1 \end{smallmatrix} \rangle = 2^n - n - 1,$	27. $\langle \begin{smallmatrix} n \\ 2 \end{smallmatrix} \rangle = 3^n - (n+1)2^n + \binom{n+1}{2},$
28. $x^n = \sum_{k=0}^n \langle \begin{smallmatrix} n \\ k \end{smallmatrix} \rangle \binom{x+k}{n},$	29. $\langle \begin{smallmatrix} n \\ m \end{smallmatrix} \rangle = \sum_{k=0}^m \binom{n+1}{k} (m+1-k)^n (-1)^k,$	30. $m! \left\{ \begin{smallmatrix} n \\ m \end{smallmatrix} \right\} = \sum_{k=0}^n \langle \begin{smallmatrix} n \\ k \end{smallmatrix} \rangle \binom{k}{n-m},$
31. $\langle \begin{smallmatrix} n \\ m \end{smallmatrix} \rangle = \sum_{k=0}^n \left\{ \begin{smallmatrix} n \\ k \end{smallmatrix} \right\} \binom{n-k}{m} (-1)^{n-k-m} k!,$	32. $\langle\langle \begin{smallmatrix} n \\ 0 \end{smallmatrix} \rangle\rangle = 1,$	33. $\langle\langle \begin{smallmatrix} n \\ n \end{smallmatrix} \rangle\rangle = 0 \text{ for } n \neq 0,$
34. $\langle\langle \begin{smallmatrix} n \\ k \end{smallmatrix} \rangle\rangle = (k+1) \langle\langle \begin{smallmatrix} n-1 \\ k \end{smallmatrix} \rangle\rangle + (2n-1-k) \langle\langle \begin{smallmatrix} n-1 \\ k-1 \end{smallmatrix} \rangle\rangle,$	35. $\sum_{k=0}^n \langle\langle \begin{smallmatrix} n \\ k \end{smallmatrix} \rangle\rangle = \frac{(2n)^n}{2^n},$	
36. $\left\{ \begin{smallmatrix} x \\ x-n \end{smallmatrix} \right\} = \sum_{k=0}^n \langle\langle \begin{smallmatrix} n \\ k \end{smallmatrix} \rangle\rangle \binom{x+n-1-k}{2n},$	37. $\left\{ \begin{smallmatrix} n+1 \\ m+1 \end{smallmatrix} \right\} = \sum_k \binom{n}{k} \left\{ \begin{smallmatrix} k \\ m \end{smallmatrix} \right\} = \sum_{k=0}^n \left\{ \begin{smallmatrix} k \\ m \end{smallmatrix} \right\} (m+1)^{n-k},$	

Theoretical Computer Science Cheat Sheet		
Identities Cont.		Trees
<p>38. $\begin{bmatrix} n+1 \\ m+1 \end{bmatrix} = \sum_k \begin{bmatrix} n \\ k \end{bmatrix} \begin{bmatrix} k \\ m \end{bmatrix} = \sum_{k=0}^n \begin{bmatrix} k \\ m \end{bmatrix} n^{n-k} = n! \sum_{k=0}^n \frac{1}{k!} \begin{bmatrix} k \\ m \end{bmatrix},$</p> <p>40. $\left\{ \begin{matrix} n \\ m \end{matrix} \right\} = \sum_k \binom{n}{k} \left\{ \begin{matrix} k+1 \\ m+1 \end{matrix} \right\} (-1)^{n-k},$</p> <p>42. $\left\{ \begin{matrix} m+n+1 \\ m \end{matrix} \right\} = \sum_{k=0}^m k \left\{ \begin{matrix} n+k \\ k \end{matrix} \right\},$</p> <p>44. $\binom{n}{m} = \sum_k \left\{ \begin{matrix} n+1 \\ k+1 \end{matrix} \right\} \begin{bmatrix} k \\ m \end{bmatrix} (-1)^{m-k},$</p> <p>46. $\left\{ \begin{matrix} n \\ n-m \end{matrix} \right\} = \sum_k \binom{m-n}{m+k} \binom{m+n}{n+k} \begin{bmatrix} m+k \\ k \end{bmatrix},$</p> <p>48. $\left\{ \begin{matrix} n \\ \ell+m \end{matrix} \right\} \binom{\ell+m}{\ell} = \sum_k \left\{ \begin{matrix} k \\ \ell \end{matrix} \right\} \left\{ \begin{matrix} n-k \\ m \end{matrix} \right\} \binom{n}{k},$</p>	<p>39. $\begin{bmatrix} x \\ x-n \end{bmatrix} = \sum_{k=0}^n \left\langle \begin{matrix} n \\ k \end{matrix} \right\rangle \begin{bmatrix} x+k \\ 2n \end{bmatrix},$</p> <p>41. $\begin{bmatrix} n \\ m \end{bmatrix} = \sum_k \begin{bmatrix} n+1 \\ k+1 \end{bmatrix} \binom{k}{m} (-1)^{m-k},$</p> <p>43. $\begin{bmatrix} m+n+1 \\ m \end{bmatrix} = \sum_{k=0}^m k(n+k) \begin{bmatrix} n+k \\ k \end{bmatrix},$</p> <p>45. $(n-m)! \binom{n}{m} = \sum_k \begin{bmatrix} n+1 \\ k+1 \end{bmatrix} \left\{ \begin{matrix} k \\ m \end{matrix} \right\} (-1)^{m-k},$ for $n \geq m,$</p> <p>47. $\begin{bmatrix} n \\ n-m \end{bmatrix} = \sum_k \binom{m-n}{m+k} \binom{m+n}{n+k} \left\{ \begin{matrix} m+k \\ k \end{matrix} \right\},$</p> <p>49. $\begin{bmatrix} n \\ \ell+m \end{bmatrix} \binom{\ell+m}{\ell} = \sum_k \begin{bmatrix} k \\ \ell \end{bmatrix} \begin{bmatrix} n-k \\ m \end{bmatrix} \binom{n}{k}.$</p>	<p>Every tree with n vertices has $n-1$ edges.</p> <p>Kraft inequality: If the depths of the leaves of a binary tree are d_1, \dots, d_n:</p> $\sum_{i=1}^n 2^{-d_i} \leq 1,$ <p>and equality holds only if every internal node has 2 sons.</p>
Recurrences		
<p>Master method:</p> $T(n) = aT(n/b) + f(n), \quad a \geq 1, b > 1$ <p>If $\exists \epsilon > 0$ such that $f(n) = O(n^{\log_b a - \epsilon})$ then</p> $T(n) = \Theta(n^{\log_b a}).$ <p>If $f(n) = \Theta(n^{\log_b a})$ then</p> $T(n) = \Theta(n^{\log_b a} \log_2 n).$ <p>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</p> $T(n) = \Theta(f(n)).$ <p>Substitution (example): Consider the following recurrence</p> $T_{i+1} = 2^{2^i} \cdot T_i^2, \quad T_1 = 2.$ <p>Note that T_i is always a power of two. Let $t_i = \log_2 T_i$. Then we have</p> $t_{i+1} = 2^i + 2t_i, \quad t_1 = 1.$ <p>Let $u_i = t_i/2^i$. Dividing both sides of the previous equation by 2^{i+1} we get</p> $\frac{t_{i+1}}{2^{i+1}} = \frac{2^i}{2^{i+1}} + \frac{t_i}{2^i}.$ <p>Substituting we find</p> $u_{i+1} = \frac{1}{2} + u_i, \quad u_1 = \frac{1}{2},$ <p>which is simply $u_i = i/2$. So we find that T_i has the closed form $T_i = 2^{i2^{i-1}}$.</p> <p>Summing factors (example): Consider the following recurrence</p> $T(n) = 3T(n/2) + n, \quad T(1) = 1.$ <p>Rewrite so that all terms involving T are on the left side</p> $T(n) - 3T(n/2) = n.$ <p>Now expand the recurrence, and choose a factor which makes the left side “telescope”</p>	<p>1($T(n) - 3T(n/2) = n$)</p> $3(T(n/2) - 3T(n/4) = n/2)$ $\vdots \quad \vdots \quad \vdots$ $3^{\log_2 n-1} (T(2) - 3T(1) = 2)$ <p>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 $k = \log_2 3 \approx 1.58496$.</p> <p>Summing the right side we get</p> $\sum_{i=0}^{m-1} \frac{n}{2^i} 3^i = n \sum_{i=0}^{m-1} \left(\frac{3}{2}\right)^i.$ <p>Let $c = \frac{3}{2}$. Then we have</p> $n \sum_{i=0}^{m-1} c^i = n \left(\frac{c^m - 1}{c - 1} \right)$ $= 2n(c^{\log_2 n} - 1)$ $= 2n(c^{(k-1)\log_2 n} - 1)$ $= 2n^k - 2n,$ <p>and so $T(n) = 3n^k - 2n$. Full history recurrences can often be changed to limited history ones (example): Consider</p> $T_i = 1 + \sum_{j=0}^{i-1} T_j, \quad T_0 = 1.$ <p>Note that</p> $T_{i+1} = 1 + \sum_{j=0}^i T_j.$ <p>Subtracting we find</p> $T_{i+1} - T_i = 1 + \sum_{j=0}^i T_j - 1 - \sum_{j=0}^{i-1} T_j$ $= T_i.$ <p>And so $T_{i+1} = 2T_i = 2^{i+1}$.</p>	<p>Generating functions:</p> <ol style="list-style-type: none"> 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. <p>Example:</p> $g_{i+1} = 2g_i + 1, \quad g_0 = 0.$ <p>Multiply and sum:</p> $\sum_{i \geq 0} g_{i+1} x^i = \sum_{i \geq 0} 2g_i x^i + \sum_{i \geq 0} x^i.$ <p>We choose $G(x) = \sum_{i \geq 0} x^i g_i$. Rewrite in terms of $G(x)$:</p> $\frac{G(x) - g_0}{x} = 2G(x) + \sum_{i \geq 0} x^i.$ <p>Simplify:</p> $\frac{G(x)}{x} = 2G(x) + \frac{1}{1-x}.$ <p>Solve for $G(x)$:</p> $G(x) = \frac{x}{(1-x)(1-2x)}.$ <p>Expand this using partial fractions:</p> $G(x) = x \left(\frac{2}{1-2x} - \frac{1}{1-x} \right)$ $= x \left(2 \sum_{i \geq 0} 2^i x^i - \sum_{i \geq 0} x^i \right)$ $= \sum_{i \geq 0} (2^{i+1} - 1) x^{i+1}.$ <p>So $g_i = 2^i - 1$.</p>

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 $i \neq 1$): $B_0 = 1, B_1 = -\frac{1}{2}, B_2 = \frac{1}{6}, B_4 = -\frac{1}{30},$ $B_6 = \frac{1}{42}, B_8 = -\frac{1}{30}, B_{10} = \frac{5}{66}.$	Continuous distributions: If $\Pr[a < X < b] = \int_a^b p(x) dx,$
2	4	3		then p is the probability density function of X . If $\Pr[X < a] = P(a),$
3	8	5	Change of base, quadratic formula: $\log_b x = \frac{\log_a x}{\log_a b}, \quad \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$	then P is the distribution function of X . If P and p both exist then $P(a) = \int_{-\infty}^a p(x) dx.$
4	16	7	Euler's number e : $e = 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + \frac{1}{120} + \dots$	Expectation: If X is discrete $\mathbb{E}[g(X)] = \sum_x g(x) \Pr[X = x].$
5	32	11	$\lim_{n \rightarrow \infty} \left(1 + \frac{x}{n}\right)^n = e^x.$	If X continuous then $\mathbb{E}[g(X)] = \int_{-\infty}^{\infty} g(x)p(x) dx = \int_{-\infty}^{\infty} g(x) dP(x).$
6	64	13	$\left(1 + \frac{1}{n}\right)^n < e < \left(1 + \frac{1}{n}\right)^{n+1}.$	Variance, standard deviation: $\text{VAR}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2,$ $\sigma = \sqrt{\text{VAR}[X]}.$
7	128	17	$\left(1 + \frac{1}{n}\right)^n = e - \frac{e}{2n} + \frac{11e}{24n^2} - O\left(\frac{1}{n^3}\right).$	For events A and B : $\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$
8	256	19	Harmonic numbers: $1, \frac{3}{2}, \frac{11}{6}, \frac{25}{12}, \frac{137}{60}, \frac{49}{20}, \frac{363}{140}, \frac{761}{280}, \frac{7129}{2520}, \dots$	$\Pr[A \wedge B] = \Pr[A] \cdot \Pr[B],$
9	512	23	$\ln n < H_n < \ln n + 1,$	iff A and B are independent. $\Pr[A B] = \frac{\Pr[A \wedge B]}{\Pr[B]}$
10	1,024	29	$H_n = \ln n + \gamma + O\left(\frac{1}{n}\right).$	For random variables X and Y : $\mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y],$
11	2,048	31	Factorial, Stirling's approximation: $1, 2, 6, 24, 120, 720, 5040, 40320, 362880, \dots$	if X and Y are independent. $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y],$
12	4,096	37	$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right).$	$\mathbb{E}[cX] = c \mathbb{E}[X].$
13	8,192	41	Ackermann's function and inverse: $a(i, j) = \begin{cases} 2^j & i = 1 \\ a(i-1, 2) & j = 1 \\ a(i-1, a(i, j-1)) & i, j \geq 2 \end{cases}$	Bayes' theorem: $\Pr[A_i B] = \frac{\Pr[B A_i] \Pr[A_i]}{\sum_{j=1}^n \Pr[A_j] \Pr[B A_j]}.$
14	16,384	43	$\alpha(i) = \min\{j \mid a(j, j) \geq i\}.$	Inclusion-exclusion: $\Pr\left[\bigvee_{i=1}^n X_i\right] = \sum_{i=1}^n \Pr[X_i] +$
15	32,768	47	Binomial distribution: $\Pr[X = k] = \binom{n}{k} p^k q^{n-k}, \quad q = 1 - p,$	$\sum_{k=2}^n (-1)^{k+1} \sum_{i_1 < \dots < i_k} \Pr\left[\bigwedge_{j=1}^k X_{i_j}\right].$
16	65,536	53	$\mathbb{E}[X] = \sum_{k=1}^n k \binom{n}{k} p^k q^{n-k} = np.$	Moment inequalities: $\Pr[X \geq \lambda \mathbb{E}[X]] \leq \frac{1}{\lambda},$
17	131,072	59	Poisson distribution: $\Pr[X = k] = \frac{e^{-\lambda} \lambda^k}{k!}, \quad \mathbb{E}[X] = \lambda.$	$\Pr[X - \mathbb{E}[X] \geq \lambda \cdot \sigma] \leq \frac{1}{\lambda^2}.$
18	262,144	61	Normal (Gaussian) distribution: $p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}, \quad \mathbb{E}[X] = \mu.$	Geometric distribution: $\Pr[X = k] = pq^{k-1}, \quad q = 1 - p,$
19	524,288	67	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 $nH_n.$	$\mathbb{E}[X] = \sum_{k=1}^{\infty} kpq^{k-1} = \frac{1}{p}.$
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				

Theoretical Computer Science Cheat Sheet																										
Trigonometry	Matrices	More Trig.																								
<div></div> <p>Pythagorean theorem: $C^2 = A^2 + B^2$.</p> <p>Definitions:</p> $\sin a = A/C, \quad \cos a = B/C,$ $\csc a = C/A, \quad \sec a = C/B,$ $\tan a = \frac{\sin a}{\cos a} = \frac{A}{B}, \quad \cot a = \frac{\cos a}{\sin a} = \frac{B}{A}.$ <p>Area, radius of inscribed circle:</p> $\frac{1}{2}AB, \quad \frac{AB}{A+B+C}.$ <p>Identities:</p> $\sin x = \frac{1}{\csc x}, \quad \cos x = \frac{1}{\sec x},$ $\tan x = \frac{1}{\cot x}, \quad \sin^2 x + \cos^2 x = 1,$ $1 + \tan^2 x = \sec^2 x, \quad 1 + \cot^2 x = \csc^2 x,$ $\sin x = \cos\left(\frac{\pi}{2} - x\right), \quad \sin x = \sin(\pi - x),$ $\cos x = -\cos(\pi - x), \quad \tan x = \cot\left(\frac{\pi}{2} - x\right),$ $\cot x = -\cot(\pi - x), \quad \csc x = \cot \frac{x}{2} - \cot x,$ $\sin(x \pm y) = \sin x \cos y \pm \cos x \sin y,$ $\cos(x \pm y) = \cos x \cos y \mp \sin x \sin y,$ $\tan(x \pm y) = \frac{\tan x \pm \tan y}{1 \mp \tan x \tan y},$ $\cot(x \pm y) = \frac{\cot x \cot y \mp 1}{\cot x \pm \cot y},$ $\sin 2x = 2 \sin x \cos x, \quad \sin 2x = \frac{2 \tan x}{1 + \tan^2 x},$ $\cos 2x = \cos^2 x - \sin^2 x, \quad \cos 2x = 2 \cos^2 x - 1,$ $\cos 2x = 1 - 2 \sin^2 x, \quad \cos 2x = \frac{1 - \tan^2 x}{1 + \tan^2 x},$ $\tan 2x = \frac{2 \tan x}{1 - \tan^2 x}, \quad \cot 2x = \frac{\cot^2 x - 1}{2 \cot x},$ $\sin(x+y) \sin(x-y) = \sin^2 x - \sin^2 y,$ $\cos(x+y) \cos(x-y) = \cos^2 x - \sin^2 y.$ <p>Euler's equation:</p> $e^{ix} = \cos x + i \sin x, \quad e^{i\pi} = -1.$	<p>Multiplication:</p> $C = A \cdot B, \quad c_{i,j} = \sum_{k=1}^n a_{i,k} b_{k,j}.$ <p>Determinants: $\det A \neq 0$ iff A is non-singular.</p> $\det A \cdot B = \det A \cdot \det B,$ $\det A = \sum_{\pi} \prod_{i=1}^n \text{sign}(\pi) a_{i,\pi(i)}.$ <p>2×2 and 3×3 determinant:</p> $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc,$ $\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = g \begin{vmatrix} a & b \\ d & e \end{vmatrix} - h \begin{vmatrix} a & c \\ d & f \end{vmatrix} + i \begin{vmatrix} a & b \\ d & e \end{vmatrix}$ $= aei + bfg + cdh - ceg - fha - ibd.$ <p>Permanents:</p> $\text{perm } A = \sum_{\pi} \prod_{i=1}^n a_{i,\pi(i)}.$	<div></div> <p>Law of cosines: $c^2 = a^2 + b^2 - 2ab \cos C$.</p> <p>Area:</p> $A = \frac{1}{2}hc,$ $= \frac{1}{2}ab \sin C,$ $= \frac{c^2 \sin A \sin B}{2 \sin C}.$ <p>Heron's formula:</p> $A = \sqrt{s \cdot s_a \cdot s_b \cdot s_c},$ $s = \frac{1}{2}(a + b + c),$ $s_a = s - a,$ $s_b = s - b,$ $s_c = s - c.$ <p>More identities:</p> $\sin \frac{x}{2} = \sqrt{\frac{1 - \cos x}{2}},$ $\cos \frac{x}{2} = \sqrt{\frac{1 + \cos x}{2}},$ $\tan \frac{x}{2} = \sqrt{\frac{1 - \cos x}{1 + \cos x}},$ $= \frac{1 - \cos x}{\sin x},$ $= \frac{\sin x}{1 + \cos x},$ $\cot \frac{x}{2} = \sqrt{\frac{1 + \cos x}{1 - \cos x}},$ $= \frac{1 + \cos x}{\sin x},$ $= \frac{\sin x}{1 - \cos x},$ $\sin x = \frac{e^{ix} - e^{-ix}}{2i},$ $\cos x = \frac{e^{ix} + e^{-ix}}{2},$ $\tan x = -i \frac{e^{ix} - e^{-ix}}{e^{ix} + e^{-ix}},$ $= -i \frac{e^{2ix} - 1}{e^{2ix} + 1},$ $\sin x = \frac{\sinh ix}{i},$ $\cos x = \cosh ix,$ $\tan x = \frac{\tanh ix}{i}.$																								
Hyperbolic Functions																										
<p>Definitions:</p> $\sinh x = \frac{e^x - e^{-x}}{2}, \quad \cosh x = \frac{e^x + e^{-x}}{2},$ $\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad \text{csch } x = \frac{1}{\sinh x},$ $\text{sech } x = \frac{1}{\cosh x}, \quad \coth x = \frac{1}{\tanh x}.$ <p>Identities:</p> $\cosh^2 x - \sinh^2 x = 1, \quad \tanh^2 x + \text{sech}^2 x = 1,$ $\coth^2 x - \text{csch}^2 x = 1, \quad \sinh(-x) = -\sinh x,$ $\cosh(-x) = \cosh x, \quad \tanh(-x) = -\tanh x,$ $\sinh(x+y) = \sinh x \cosh y + \cosh x \sinh y,$ $\cosh(x+y) = \cosh x \cosh y + \sinh x \sinh y,$ $\sinh 2x = 2 \sinh x \cosh x,$ $\cosh 2x = \cosh^2 x + \sinh^2 x,$ $\cosh x + \sinh x = e^x, \quad \cosh x - \sinh x = e^{-x},$ $(\cosh x + \sinh x)^n = \cosh nx + \sinh nx, \quad n \in \mathbb{Z},$ $2 \sinh^2 \frac{x}{2} = \cosh x - 1, \quad 2 \cosh^2 \frac{x}{2} = \cosh x + 1.$																										
<table><tr><th>θ</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>		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	∞	<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	∞																							
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 $i \neq j$.</p> <p>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.</p> <p>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 style="text-align: center;">Geometry</p> <p>Projective coordinates: triples (x, y, z), not all x, y and z zero.</p> $(x, y, z) = (cx, cy, cz) \quad \forall c \neq 0.$ <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>  $\cos \theta = \frac{(x_1, y_1) \cdot (x_2, y_2)}{l_1 l_2}.$ <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.</p> <p>– 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"> $\frac{d(cu)}{dx} = c \frac{du}{dx},$ $\frac{d(u+v)}{dx} = \frac{du}{dx} + \frac{dv}{dx},$ $\frac{d(uv)}{dx} = u \frac{dv}{dx} + v \frac{du}{dx},$ $\frac{d(u^n)}{dx} = nu^{n-1} \frac{du}{dx},$ $\frac{d(u/v)}{dx} = \frac{v(\frac{du}{dx}) - u(\frac{dv}{dx})}{v^2},$ $\frac{d(e^{cu})}{dx} = ce^{cu} \frac{du}{dx},$ $\frac{d(c^u)}{dx} = (\ln c)c^u \frac{du}{dx},$ $\frac{d(\ln u)}{dx} = \frac{1}{u} \frac{du}{dx},$ $\frac{d(\sin u)}{dx} = \cos u \frac{du}{dx},$ $\frac{d(\cos u)}{dx} = -\sin u \frac{du}{dx},$ $\frac{d(\tan u)}{dx} = \sec^2 u \frac{du}{dx},$ $\frac{d(\cot u)}{dx} = \csc^2 u \frac{du}{dx},$ $\frac{d(\sec u)}{dx} = \tan u \sec u \frac{du}{dx},$ $\frac{d(\csc u)}{dx} = -\cot u \csc u \frac{du}{dx},$ $\frac{d(\arcsin u)}{dx} = \frac{1}{\sqrt{1-u^2}} \frac{du}{dx},$ $\frac{d(\arccos u)}{dx} = \frac{-1}{\sqrt{1-u^2}} \frac{du}{dx},$ $\frac{d(\arctan u)}{dx} = \frac{1}{1+u^2} \frac{du}{dx},$ $\frac{d(\operatorname{arccot} u)}{dx} = \frac{-1}{1+u^2} \frac{du}{dx},$ $\frac{d(\operatorname{arcsec} u)}{dx} = \frac{1}{u\sqrt{1-u^2}} \frac{du}{dx},$ $\frac{d(\operatorname{arccsc} u)}{dx} = \frac{-1}{u\sqrt{1-u^2}} \frac{du}{dx},$ $\frac{d(\sinh u)}{dx} = \cosh u \frac{du}{dx},$ $\frac{d(\cosh u)}{dx} = \sinh u \frac{du}{dx},$ $\frac{d(\tanh u)}{dx} = \operatorname{sech}^2 u \frac{du}{dx},$ $\frac{d(\coth u)}{dx} = -\operatorname{csch}^2 u \frac{du}{dx},$ $\frac{d(\operatorname{sech} u)}{dx} = -\operatorname{sech} u \tanh u \frac{du}{dx},$ $\frac{d(\operatorname{csch} u)}{dx} = -\operatorname{csch} u \coth u \frac{du}{dx},$ $\frac{d(\operatorname{arcsinh} u)}{dx} = \frac{1}{\sqrt{1+u^2}} \frac{du}{dx},$ $\frac{d(\operatorname{arccosh} u)}{dx} = \frac{1}{\sqrt{u^2-1}} \frac{du}{dx},$ $\frac{d(\operatorname{arctanh} u)}{dx} = \frac{1}{1-u^2} \frac{du}{dx},$ $\frac{d(\operatorname{arcoth} u)}{dx} = \frac{1}{u^2-1} \frac{du}{dx},$ $\frac{d(\operatorname{arcsech} u)}{dx} = \frac{-1}{u\sqrt{1-u^2}} \frac{du}{dx},$ $\frac{d(\operatorname{arccsch} u)}{dx} = \frac{-1}{ u \sqrt{1+u^2}} \frac{du}{dx}.$ <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.		Finite Calculus
62. $\int \frac{dx}{x\sqrt{x^2 - a^2}} = \frac{1}{a} \arccos \frac{a}{ x }, \quad a > 0,$	63. $\int \frac{dx}{x^2\sqrt{x^2 \pm a^2}} = \mp \frac{\sqrt{x^2 \pm a^2}}{a^2 x},$	Difference, shift operators: $\Delta f(x) = f(x+1) - f(x),$ $E f(x) = f(x+1).$
64. $\int \frac{x dx}{\sqrt{x^2 \pm a^2}} = \sqrt{x^2 \pm a^2},$	65. $\int \frac{\sqrt{x^2 \pm a^2}}{x^4} dx = \mp \frac{(x^2 + a^2)^{3/2}}{3a^2 x^3},$	Fundamental Theorem: $f(x) = \Delta F(x) \Leftrightarrow \sum f(x) \delta x = F(x) + C.$
66. $\int \frac{dx}{ax^2 + bx + c} = \begin{cases} \frac{1}{\sqrt{b^2 - 4ac}} \ln \left \frac{2ax + b - \sqrt{b^2 - 4ac}}{2ax + b + \sqrt{b^2 - 4ac}} \right , & \text{if } b^2 > 4ac, \\ \frac{2}{\sqrt{4ac - b^2}} \arctan \frac{2ax + b}{\sqrt{4ac - b^2}}, & \text{if } b^2 < 4ac, \end{cases}$		$\sum_a^b f(x) \delta x = \sum_{i=a}^{b-1} f(i).$
67. $\int \frac{dx}{\sqrt{ax^2 + bx + c}} = \begin{cases} \frac{1}{\sqrt{a}} \ln \left 2ax + b + 2\sqrt{a}\sqrt{ax^2 + bx + c} \right , & \text{if } a > 0, \\ \frac{1}{\sqrt{-a}} \arcsin \frac{-2ax - b}{\sqrt{b^2 - 4ac}}, & \text{if } a < 0, \end{cases}$		Differences: $\Delta(cu) = c\Delta u, \quad \Delta(u+v) = \Delta u + \Delta v,$ $\Delta(uv) = u\Delta v + E v \Delta u,$ $\Delta(x^n) = nx^{n-1},$ $\Delta(H_x) = x^{-1}, \quad \Delta(2^x) = 2^x,$ $\Delta(c^x) = (c-1)c^x, \quad \Delta\binom{x}{m} = \binom{x}{m-1}.$
68. $\int \sqrt{ax^2 + bx + c} dx = \frac{2ax + b}{4a} \sqrt{ax^2 + bx + c} + \frac{4ac - b^2}{8a} \int \frac{dx}{\sqrt{ax^2 + bx + c}},$		Sums: $\sum cu \delta x = c \sum u \delta x,$ $\sum (u+v) \delta x = \sum u \delta x + \sum v \delta x,$ $\sum u \Delta v \delta x = uv - \sum E v \Delta u \delta x,$ $\sum x^n \delta x = \frac{x^{n+1}}{n+1}, \quad \sum x^{-1} \delta x = H_x,$ $\sum c^x \delta x = \frac{c^x}{c-1}, \quad \sum \binom{x}{m} \delta x = \binom{x}{m+1}.$
69. $\int \frac{x dx}{\sqrt{ax^2 + bx + c}} = \frac{\sqrt{ax^2 + bx + c}}{a} - \frac{b}{2a} \int \frac{dx}{\sqrt{ax^2 + bx + c}},$		Falling Factorial Powers: $x^{\overline{n}} = x(x-1) \cdots (x-n+1), \quad n > 0,$ $x^{\overline{0}} = 1,$ $x^{\overline{n}} = \frac{1}{(x+1) \cdots (x+ n)}, \quad n < 0,$ $x^{\overline{n+m}} = x^{\overline{n}}(x-m)^{\overline{n}}.$
70. $\int \frac{dx}{x\sqrt{ax^2 + bx + c}} = \begin{cases} \frac{-1}{\sqrt{c}} \ln \left \frac{2\sqrt{c}\sqrt{ax^2 + bx + c} + bx + 2c}{x} \right , & \text{if } c > 0, \\ \frac{1}{\sqrt{-c}} \arcsin \frac{bx + 2c}{ x \sqrt{b^2 - 4ac}}, & \text{if } c < 0, \end{cases}$		Rising Factorial Powers: $x^{\overline{n}} = x(x+1) \cdots (x+n-1), \quad n > 0,$ $x^{\overline{0}} = 1,$ $x^{\overline{n}} = \frac{1}{(x-1) \cdots (x- n)}, \quad n < 0,$ $x^{\overline{n+m}} = x^{\overline{n}}(x+m)^{\overline{n}}.$
71. $\int x^3 \sqrt{x^2 + a^2} dx = (\frac{1}{3}x^2 - \frac{2}{15}a^2)(x^2 + a^2)^{3/2},$		Conversion: $x^{\overline{n}} = (-1)^n (-x)^{\overline{n}} = (x-n+1)^{\overline{n}}$ $= 1/(x+1)^{\overline{-n}},$ $x^{\overline{n}} = (-1)^n (-x)^{\overline{n}} = (x+n-1)^{\overline{n}}$ $= 1/(x-1)^{\overline{-n}},$ $x^n = \sum_{k=1}^n \left\{ \begin{matrix} n \\ k \end{matrix} \right\} x^{\overline{k}} = \sum_{k=1}^n \left\{ \begin{matrix} n \\ k \end{matrix} \right\} (-1)^{n-k} x^{\overline{k}},$ $x^{\overline{n}} = \sum_{k=1}^n \left[\begin{matrix} n \\ k \end{matrix} \right] (-1)^{n-k} x^k,$ $x^{\overline{n}} = \sum_{k=1}^n \left[\begin{matrix} n \\ k \end{matrix} \right] x^k.$
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. $\int x^n (\ln ax)^m dx = \frac{x^{n+1}}{n+1} (\ln ax)^m - \frac{m}{n+1} \int x^n (\ln ax)^{m-1} dx.$		
$x^{\overline{1}} = x^1$	$x^{\overline{2}} = x^2 + x^1$	$x^{\overline{3}} = x^3 + 3x^2 + x^1$
$x^{\overline{4}} = x^4 + 6x^3 + 7x^2 + x^1$	$x^{\overline{5}} = x^5 + 15x^4 + 25x^3 + 10x^2 + x^1$	
$x^{\overline{1}} = x^1$	$x^{\overline{2}} = x^2 + x^1$	$x^{\overline{3}} = x^3 + 3x^2 + 2x^1$
$x^{\overline{4}} = x^4 + 6x^3 + 11x^2 + 6x^1$	$x^{\overline{5}} = x^5 + 10x^4 + 35x^3 + 50x^2 + 24x^1$	

Theoretical Computer Science Cheat Sheet		
Series		
Taylor's series:		Ordinary power series:
$f(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2}f''(a) + \dots = \sum_{i=0}^{\infty} \frac{(x-a)^i}{i!} f^{(i)}(a).$		$A(x) = \sum_{i=0}^{\infty} a_i x^i.$
Expansions:		Exponential power series:
$\frac{1}{1-x}$	$= 1 + x + x^2 + x^3 + x^4 + \dots$	$= \sum_{i=0}^{\infty} x^i,$
$\frac{1}{1-cx}$	$= 1 + cx + c^2x^2 + c^3x^3 + \dots$	$= \sum_{i=0}^{\infty} c^i x^i,$
$\frac{1}{1-x^n}$	$= 1 + x^n + x^{2n} + x^{3n} + \dots$	$= \sum_{i=0}^{\infty} x^{ni},$
$\frac{x}{(1-x)^2}$	$= x + 2x^2 + 3x^3 + 4x^4 + \dots$	$= \sum_{i=0}^{\infty} i x^i,$
$\sum_{k=0}^n \binom{n}{k} \frac{k! z^k}{(1-z)^{k+1}}$	$= x + 2^n x^2 + 3^n x^3 + 4^n x^4 + \dots$	$= \sum_{i=0}^{\infty} i^n x^i,$
e^x	$= 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \dots$	$= \sum_{i=0}^{\infty} \frac{x^i}{i!},$
$\ln(1+x)$	$= x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 + \dots$	$= \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^i}{i},$
$\ln \frac{1}{1-x}$	$= x + \frac{1}{2}x^2 + \frac{1}{3}x^3 + \frac{1}{4}x^4 + \dots$	$= \sum_{i=1}^{\infty} \frac{x^i}{i},$
$\sin x$	$= x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \frac{1}{7!}x^7 + \dots$	$= \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)!},$
$\cos x$	$= 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \frac{1}{6!}x^6 + \dots$	$= \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!},$
$\tan^{-1} x$	$= x - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + \dots$	$= \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)},$
$(1+x)^n$	$= 1 + nx + \frac{n(n-1)}{2}x^2 + \dots$	$= \sum_{i=0}^{\infty} \binom{n}{i} x^i,$
$\frac{1}{(1-x)^{n+1}}$	$= 1 + (n+1)x + \binom{n+2}{2}x^2 + \dots$	$= \sum_{i=0}^{\infty} \binom{i+n}{i} x^i,$
$\frac{x}{e^x - 1}$	$= 1 - \frac{1}{2}x + \frac{1}{12}x^2 - \frac{1}{720}x^4 + \dots$	$= \sum_{i=0}^{\infty} \frac{B_i x^i}{i!},$
$\frac{1}{2x}(1 - \sqrt{1-4x})$	$= 1 + x + 2x^2 + 5x^3 + \dots$	$= \sum_{i=0}^{\infty} \frac{1}{i+1} \binom{2i}{i} x^i,$
$\frac{1}{\sqrt{1-4x}}$	$= 1 + 2x + 6x^2 + 20x^3 + \dots$	$= \sum_{i=0}^{\infty} \binom{2i}{i} x^i,$
$\frac{1}{\sqrt{1-4x}} \left(\frac{1 - \sqrt{1-4x}}{2x} \right)^n$	$= 1 + (2+n)x + \binom{4+n}{2}x^2 + \dots$	$= \sum_{i=0}^{\infty} \binom{2i+n}{i} x^i,$
$\frac{1}{1-x} \ln \frac{1}{1-x}$	$= x + \frac{3}{2}x^2 + \frac{11}{6}x^3 + \frac{25}{12}x^4 + \dots$	$= \sum_{i=1}^{\infty} H_i x^i,$
$\frac{1}{2} \left(\ln \frac{1}{1-x} \right)^2$	$= \frac{1}{2}x^2 + \frac{3}{4}x^3 + \frac{11}{24}x^4 + \dots$	$= \sum_{i=2}^{\infty} \frac{H_{i-1} x^i}{i},$
$\frac{x}{1-x-x^2}$	$= x + x^2 + 2x^3 + 3x^4 + \dots$	$= \sum_{i=0}^{\infty} F_i x^i,$
$\frac{F_n x}{1 - (F_{n-1} + F_{n+1})x - (-1)^n x^2}$	$= F_n x + F_{2n} x^2 + F_{3n} x^3 + \dots$	$= \sum_{i=0}^{\infty} F_{ni} x^i.$
		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
Expansions:		
$\frac{1}{(1-x)^{n+1}} \ln \frac{1}{1-x}$	$= \sum_{i=0}^{\infty} (H_{n+i} - H_n) \binom{n+i}{i} x^i,$	$\left(\frac{1}{x}\right)^{\overline{-n}} = \sum_{i=0}^{\infty} \left\{ \begin{matrix} i \\ n \end{matrix} \right\} x^i,$
$x^{\overline{n}}$	$= \sum_{i=0}^{\infty} \left[\begin{matrix} n \\ i \end{matrix} \right] x^i,$	$(e^x - 1)^n = \sum_{i=0}^{\infty} \left\{ \begin{matrix} i \\ n \end{matrix} \right\} \frac{n! x^i}{i!},$
$\left(\ln \frac{1}{1-x}\right)^n$	$= \sum_{i=0}^{\infty} \left[\begin{matrix} i \\ n \end{matrix} \right] \frac{n! x^i}{i!},$	$x \cot x = \sum_{i=0}^{\infty} \frac{(-4)^i B_{2i} x^{2i}}{(2i)!},$
$\tan x$	$= \sum_{i=1}^{\infty} (-1)^{i-1} \frac{2^{2i} (2^{2i} - 1) B_{2i} x^{2i-1}}{(2i)!},$	$\zeta(x) = \sum_{i=1}^{\infty} \frac{1}{i^x},$
$\frac{1}{\zeta(x)}$	$= \sum_{i=1}^{\infty} \frac{\mu(i)}{i^x},$	$\frac{\zeta(x-1)}{\zeta(x)} = \sum_{i=1}^{\infty} \frac{\phi(i)}{i^x},$
$\zeta(x)$	$= \prod_p \frac{1}{1-p^{-x}},$	
$\zeta^2(x)$	$= \sum_{i=1}^{\infty} \frac{d(i)}{x^i} \quad \text{where } d(n) = \sum_{d n} 1,$	
$\zeta(x)\zeta(x-1)$	$= \sum_{i=1}^{\infty} \frac{S(i)}{x^i} \quad \text{where } S(n) = \sum_{d n} d,$	
$\zeta(2n)$	$= \frac{2^{2n-1} B_{2n} }{(2n)!} \pi^{2n}, \quad n \in \mathbb{N},$	
$\frac{x}{\sin x}$	$= \sum_{i=0}^{\infty} (-1)^{i-1} \frac{(4^i - 2) B_{2i} x^{2i}}{(2i)!},$	
$\left(\frac{1 - \sqrt{1-4x}}{2x}\right)^n$	$= \sum_{i=0}^{\infty} \frac{n(2i+n-1)!}{i!(n+i)!} x^i,$	
$e^x \sin x$	$= \sum_{i=1}^{\infty} \frac{2^{i/2} \sin \frac{i\pi}{4}}{i!} x^i,$	
$\sqrt{\frac{1 - \sqrt{1-x}}{x}}$	$= \sum_{i=0}^{\infty} \frac{(4i)!}{16^i \sqrt{2} (2i)!(2i+1)!} x^i,$	
$\left(\frac{\arcsin x}{x}\right)^2$	$= \sum_{i=0}^{\infty} \frac{4^i i!^2}{(i+1)(2i+1)!} x^{2i}.$	
Cramer's Rule		Stieltjes Integration
If we have equations:		If G is continuous in the interval $[a, b]$ and F is nondecreasing then
$a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n = b_1$		$\int_a^b G(x) dF(x)$
$a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n = b_2$		exists. If $a \leq b \leq c$ then
\vdots		$\int_a^c G(x) dF(x) = \int_a^b G(x) dF(x) + \int_b^c G(x) dF(x).$
$a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,n}x_n = b_n$		If the integrals involved exist
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		$\int_a^b (G(x) + H(x)) dF(x) = \int_a^b G(x) dF(x) + \int_a^b H(x) dF(x),$
$x_i = \frac{\det A_i}{\det A}.$		$\int_a^b G(x) d(F(x) + H(x)) = \int_a^b G(x) dF(x) + \int_a^b G(x) dH(x),$
		$\int_a^b c \cdot G(x) dF(x) = \int_a^b G(x) d(c \cdot F(x)) = c \int_a^b G(x) dF(x),$
		$\int_a^b G(x) dF(x) = G(b)F(b) - G(a)F(a) - \int_a^b F(x) dG(x).$
		If the integrals involved exist, and F possesses a derivative F' at every point in $[a, b]$ then
		$\int_a^b G(x) dF(x) = \int_a^b G(x) F'(x) dx.$
Improvement makes strait roads, but the crooked roads without Improvement, are roads of Genius. – William Blake (The Marriage of Heaven and Hell)		Fibonacci Numbers
		1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...
		Definitions:
		$F_i = F_{i-1} + F_{i-2}, \quad F_0 = F_1 = 1,$
		$F_{-i} = (-1)^{i-1} F_i,$
		$F_i = \frac{1}{\sqrt{5}} \left(\phi^i - \hat{\phi}^i \right),$
		Cassini's identity: for $i > 0$:
		$F_{i+1}F_{i-1} - F_i^2 = (-1)^i.$
		Additive rule:
		$F_{n+k} = F_k F_{n+1} + F_{k-1} F_n,$
		$F_{2n} = F_n F_{n+1} + F_{n-1} F_n.$
		Calculation by matrices:
		$\begin{pmatrix} F_{n-2} & F_{n-1} \\ F_{n-1} & F_n \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n.$