Team Contest Reference

Universität zu Lübeck

Team: No Output

Inhaltsverzeichnis

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

1 Mathematische Algorithmen

1.1 Primzahlen

Für Primzahlen gilt immer (aber nicht nur für Primzahlen)

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

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

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

MD5: `f2241e45384c9165389a8ef7eaffdb24`

1.1.2 Primzahlentest

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

MD5: `ab672f1e03a3f839b6fb0d9b93dd21d0`

1.2 Binomial Koeffizient

```
static int[][] mem = new int[MAX_N][(MAX_N + 1) / 2];
static int binoCo(int n, int k) {
  if (k < 0 || k > n) return 0;
  if (2 * k > n) binoCo(n, n - k);
  if (mem[n][k] > 0) return mem[n][k];
  int ret = 1;
  for (int i = 1; i <= k; i++) {
    ret *= n - k + i;
    ret /= i;
    mem[n][i] = ret;
  }
  return ret;
}
```

MD5: `3a459246143bbdc49336d77c9b2720e4`

1.3 Eulersche φ-Funktion

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

```
#include <iostream>
#include <cmath>
using namespace std;
int phi(int);
int main(){
  int n;
  while((cin>>n)!=0) cout << phi(n) << endl;
  return 0;
}

int phi(int n){
  int coprime = 1;
  int primes[] = {2,3,5,7,11,13};//...
  int primessizes  = 6; //anpassen !
  //zusaetzlich Primfaktorzerlegung v. n
  for(int i =0; i<primessizes; i++){
    int anz = 0;
    while(n % primes[i] == 0){
      n = n / primes[i];
      anz ++;
      cout<<" p: "<<primes[i]<<endl;
    }
    if(anz>0)
      coprime *= ((int) pow((double) primes[i],
        (double)(anz-1))*(primes[i] -
1));
    if(n==1) break;
  }
  if(n != 1){
    coprime *= (n - 1);
  }
  return coprime;
}
```

2 Mathematisch Formeln und Gesetze

2.1 Catalan

$C_n = \frac{1}{n+1}\binom{2n}{n} = \prod_{k=2}^{n}(n+k)/k$
$C_{n+1} = \frac{4n+2}{n+2} C_n = \sum_{k=0}^{n} C_k C_{n-k}$

2.2 kgV und ggT

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

2.3 modulare Exponentiation

$b^e \equiv c(\mod)m$
$b^e = b^{\left(\sum_{i=0}^{n-1} a_i 2^i\right)} = \prod_{i=0}^{n-1} \left(b^{2^i}\right)^{a_i}$

```
function modular_pow(base, exponent, modulus)
    result := 1
    while exponent > 0
        if (exponent mod 2 == 1):
            result := (result * base) mod modulus
        exponent := exponent >> 1
        base = (base * base) mod modulus
    return result
```

2.4 Modulare Arithmetik

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

$$d = \text{ggT}(a,b) = as + bt.$$

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

mod 1). :
$d = 1 \Rightarrow 1 \equiv t \cdot b(\mod a) \Rightarrow b^{-1} := t$
$d \neq 1 \Rightarrow b^{-1}$ existiert nicht bzgl a, b.

2.4.1 Erweiterter Euklidischer Algorithmus

```
static int[] eea(int a, int b) {
  int[] dst = new int[3];
  if (b == 0) {
    dst[0] = a;
    dst[1] = 1;
    return dst; // a, 1, 0
  }
  dst = eea(b, a % b);
  int tmp = dst[2];
  dst[2] = dst[1] - ((a / b) * dst[2]);
  dst[1] = tmp;
  return dst;
}
```

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 Datenstukturen

3.1 Fenwick Tree (Binary Indexed Tree)

```
class FenwickTree {
  private int[] values;
  private int n;
  public FenwickTree(int n) {
    this.n = n;
    values = new int[n];
  }
  public int get(int i) { //get value of i
    int x = values[0];
    while (i > 0) {
      x += values[i];
      i -= i & -i; }
    return x;
  }
  public void add(int i, int x) { // add x to interval
      [i,n]
    if (i == 0) values[0] += x;
    else {
      while (i < n) {
        values[i] += x;
        i += i & -i; }
    }
  }
}
```

MD5: `da8d56a0188958c7d35409b7a6fb7a9c`

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; $2|\deg(v \in V)$ 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

```
static List<Integer> topoSort(Map<Integer, List<
    Integer>> edges,
    Map<Integer, List<Integer>> revedges) {
  Queue<Integer> q = new LinkedList<Integer>();
  List<Integer> ret = new LinkedList<Integer>();
  Map<Integer, Integer> indeg = new HashMap<Integer,
      Integer>();
  for (int v : revedges.keySet()) {
    indeg.put(v, revedges.get(v).size());
    if (revedges.get(v).size() == 0)
      q.add(v);
  }
  while (!q.isEmpty()) {
    int tmp = q.poll();
    ret.add(tmp);
    for (int dest : edges.get(tmp)) {
      indeg.put(dest, indeg.get(dest) - 1);
      if (indeg.get(dest) == 0)
        q.add(dest);
    }
  }
  return ret;
}
```

MD5: `f89e486b31561403ed45869c9ca5b180`

4.3 Prim (Minimum Spanning Tree)

```
#define WHITE 0
#define BLACK 1
#define INF INT_MAX

int baum( int **matrix, int N){
  int i, sum = 0;

  int color[N];
  int dist[N];

    // markiere alle Knoten ausser 0 als unbesucht
  color[0] = BLACK;
  for( i=1; i<N; i++){
    color[i] = WHITE;
    dist[i] = INF;
  }

    // berechne den Rand
  for( i=1; i<N; i++){
        if( dist[i] > matrix[i][nextIndex]){
            dist[i] = matrix[i][nextIndex];
        }
    }

  while( 1){
    int nextDist = INF, nextIndex = -1;
```

```
    /* Den naechsten Knoten waehlen */
    for(i=0; i<N; i++){
      if( color[i] != WHITE) continue;

      if( dist[i] < nextDist){
        nextDist = dist[i];
        nextIndex = i;
      }
    }

    /* Abbruchbedingung */
    if( nextIndex == -1) break;

    /* Knoten in MST aufnehmen */
    color[nextIndex] = RED;
    sum += nextDist;

    /* naechste kuerzeste Distanzen berechnen */
    for( i=0; i<N; i++){
            if( i == nextIndex || color[i] == BLACK )
                continue;

            if( dist[i] > matrix[i][nextIndex]){
                dist[i] = matrix[i][nextIndex];
            }
    }
  }

  return sum;
}
```

4.4 Kruskal

```
public static LinkedList<Edge> kruskal(LinkedList<Edge
    > adjList, int root, int nodeCount) {
  LinkedList<SortedSet<Integer>> branches = new
      LinkedList<SortedSet<Integer>>();
  for (int i = 0; i < nodeCount; i++) {
    branches.add(new TreeSet<Integer>());
    branches.get(branches.size() - 1).add(i);
  }

  PriorityQueue<Edge> edges = new PriorityQueue<Edge
      >(1, new Comparator<Edge>() {
    @Override
    public int compare(Edge e1, Edge e2) {
      if (e1.weight <= e2.weight) {
        return -1;
      } else {
        return 1;
      }
    }
  });
  edges.addAll(adjList);
  LinkedList<Edge> result = new LinkedList<Edge>();

  while (branches.size() > 1) {
    Edge min = edges.remove();

    SortedSet<Integer> from = null;
    for (SortedSet<Integer> branchFrom : branches) {
      if (branchFrom.contains(min.from)) {
        if (!branchFrom.contains(min.to)) {
          from = branchFrom;
          break;
        }
      }
    }

    if (from != null) {
      for (SortedSet<Integer> branchTo : branches) {
        if (!(from.equals(branchTo))) {
          if (branchTo.contains(min.to)) {
            from.addAll(branchTo);
            branches.remove(branchTo);
            result.add(min);
            break;
          }
        }
      }
    }
  }

  return result;
}
```

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

```
for(int i = 0; i<n; i++)
  for(int j = 0; j<n; j++)
    if((i,j) ∈ E(G)){
      d[i,j] = w[i,j];
    else
      d[i,j] = ∞
for(int k = 0; k<n; k++)
  for(int i = 0; i<n; i++)
    for(int j = 0; j<n; j++)
      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

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

```
// look for shortest distance from a to b in adjacency
    matrix
// visited nodes for breadth first search
bool nodeVisited[26];
for (int k=0; k<26; k++) {
        nodeVisited[k]=false;
}
queue<int> searchQueue;
queue<string> outputQueue;
searchQueue.push(aNumber); // start search from a
string start="";
start += a[0];
outputQueue.push(start);
string outputString;
while (searchQueue.empty()==false && nodeVisited[
    bNumber]==false) {
        int node=searchQueue.front();
        searchQueue.pop();
        string nodeString=outputQueue.front();
        outputQueue.pop();
```

```
        for (int k=0; k<26; k++) {
                if (cities[node][k]==true && nodeVisited[k]==false) {
                        searchQueue.push(k);
                        nodeVisited[k]=true;
                        char addToOutput=k+'A';
                        string s=nodeString;
                        s += addToOutput;
                        outputQueue.push(s);
                        if (k==bNumber) {
                                outputString=s;
                        }
                }
        }
}
cout << outputString << "\n";
```

4.7 Belman-Ford

```
procedure BellmanFord(list vertices, list edges, vertex source)
   // This implementation takes in a graph, represented as lists of vertices
   // and edges, and modifies the vertices so that their distance and
   // predecessor attributes store the shortest paths.

   // Step 1: initialize graph
   for each vertex v in vertices:
       if v is source tn v.distance := 0
       else v.distance := infinity
       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

```
public class Flow {
  static class Edge {
    int c;
    int f = 0;
    Vertex s;
    Vertex d;
    Edge(int cap, Vertex source, Vertex dest) {
      c = cap;
      s = source;
      d = dest;
    }
    int res(Vertex v) {
      if (v == d) return f;
      else return c - f;
    }
  }
  static class Vertex {
    List<Edge> lks = new ArrayList<Edge>();
  }
  static int maxFlow(Vertex so, Vertex si) {
    ff: while (true) {
      HashMap<Vertex, Edge> etp = new HashMap<Vertex, Edge>();
      List<Vertex> fringe = new ArrayList<Vertex>();
      fringe.add(so);
      etp.put(so, null);
      int minRes = Integer.MAX_VALUE;
      boolean foundrp = false;
      bfs: while (!fringe.isEmpty()) {
        List<Vertex> newFringe = new ArrayList<Vertex>();
        for (Vertex v : fringe) {
          for (Edge e : v.lks) {
            Vertex child = (e.d == v) ? e.s : e.d;
            if (!etp.containsKey(child) && e.res(v) > 0) {
              etp.put(child, e);
              newFringe.add(child);
              minRes = Math.min(minRes, e.res(v));
              if (child == si) {
                foundrp = true;
                break bfs;
        } } } }
        fringe = newFringe;
      }
      if (!foundrp) break ff;
      Vertex nxt = si;
      while (nxt != so) {
        Vertex prv = nxt;
        Edge edge = etp.get(prv);
        if (edge.s == prv) {
          edge.f = edge.f - minRes;
          nxt = edge.d;
        } else {
          edge.f = edge.f + minRes;
          nxt = edge.s;
        }
      }
    }
    int flow = 0;
    for (Edge e : so.lks) {
      flow += e.f;
    }
    return flow;
  }
}
```

MD5: a29c73a7d958ca12f3778a65c39a2e3e

4.9 Bipartite Matching

```
import java.util.*;


public class BPM {
   int m, n;
      boolean[][] graph;
      boolean seen[];
      int matchL[];   //What left vertex i is matched to (or -1 if unmatched)
```

```java
    int matchR[];   //What right vertex j is matched to (or -1 if unmatched)

    int maximumMatching() {
        //Read input and populate graph[][]
        //Set m to be the size of L, n to be the size of R
        Arrays.fill(matchL, -1);
        Arrays.fill(matchR, -1);

        int count = 0;
        for (int i = 0; i < m; i++) {
            Arrays.fill(seen, false);
            if (bpm(i)) count++;
        }
        return count;
    }

    boolean bpm(int u) {
        //try to match with all vertices on right side
        for (int v = 0; v < n; v++) {
            if (!graph[u][v] || seen[v]) continue;
            seen[v] = true;
            //match u and v, if v is unassigned, or if v's match on the left side can be reassigned to another right vertex
            if (matchR[v] == -1 || bpm(matchR[v])) {
                matchL[u] = v;
                matchR[v] = u;
                return true;
            }
        }
        return false;
    }

    public void run(){

      Scanner sc = new Scanner(System.in).useLocale(Locale.US);
      int T = sc.nextInt();
      while(T-->0){
        n = sc.nextInt();
        m =  sc.nextInt();
        int K = sc.nextInt();
        graph = new boolean [m][n];
        matchL = new int[m];
        matchR = new int[n];
        seen = new boolean[n];
        while(K-->0){
          int y = (int)sc.nextDouble();
          int x  = (int)sc.nextDouble();
          graph[x][y] = true;

        }
        System.out.println(maximumMatching());
      }
      sc.close();
    }

    public static void main(String[] args){
      (new BPM()).run();
    }
}
```


5 Geometrie

5.1 Kreuzprodukt, Skalarprodukt

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

5.2 Orthogonale Projektion

r_0 : Ortsvektor; u : Richtungsvektor; n : Normalenvektor
$P_g(\vec{x}) = \vec{r}_0 + \frac{(\vec{x}-\vec{r}_0)\cdot\vec{u}}{\vec{u}\cdot\vec{u}}\,\vec{u}$
$P_g(\vec{x}) = \vec{x} - \frac{(\vec{x}-\vec{r}_0)\cdot\vec{n}}{\vec{n}\cdot\vec{n}}\,\vec{n}$(nur 2D bzw. 3D auf Ebene)

5.3 Rotation

```java
  static Point rotate(Point v, double a) {
    double cos = Math.cos(a);
    double sin = Math.sin(a);
    double x = cos * v.x - sin * v.y;
    double y = sin * v.x + cos * v.y;
    return new Point(x, y);
  }
```

5.4 Geradenschnittpunkt

$g_1 : ax + by = c; g_2 : px + qx = r; \Rightarrow \vec{p} = \frac{1}{aq-bp} \begin{pmatrix} x = cq - br \\ y = ar - cp \end{pmatrix}$

$g_1 : \vec{p} = \begin{pmatrix} r_x \\ r_y \end{pmatrix} + s \begin{pmatrix} s_x \\ s_y \end{pmatrix}$ $g_2 : \vec{p} = \begin{pmatrix} q_x \\ q_y \end{pmatrix} + t \begin{pmatrix} t_x \\ t_y \end{pmatrix}$ $w_x = (r_x - q_x), w_y = (r_y - q_y)$
$\Rightarrow D = (s_x t_y - t_x s_y), D_s = (t_x w_y - t_y w_x), D_t = (s_y w_x - s_x w_y); s = D_s/D, t = D_t/D$

5.5 Zusammenhang Kreuzprodukt & Sinus

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

5.6 Dreiecksfläche

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

5.7 Graham Scan (Convex Hull)

```java
  public static class Point implements Comparable<Point> {
    double x, y, r;
    Point p0;
    public Point(double x, double y) {
      this.x = x;
      this.y = y;
    }
    public int compareTo(Point p) {
      double s = ccw(p0, p, this);
      if (s != 0) return (int) Math.signum(s);
      else return (int) Math.signum(p.r - r);
```

```java
  }
  public static double dist(Point a, Point b) {
    double x = a.x - b.x;
    double y = a.y - b.y;
    return Math.sqrt(x * x + y * y);
  }
  public static double ccw(Point a, Point b, Point c)
      {
    return (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (
        c.x - a.x);
  }
  static List<Point> graham(List<Point> P) {
    Point p0 = P.get(0);
    for (int i = 1; i < P.size(); i++) {
      Point p = P.get(i);
      if (p.y < p0.y || (p.y == p0.y && p.x < p0.x)) {
        p0 = p;
    } }
    P.remove(p0);
    for (Point p : P) {
      p.r = dist(p0, p);
      p.p0 = p0;
    }
    Collections.sort(P);
    Iterator<Point> I = P.iterator();
    Point f = I.next();
    while (I.hasNext()) {
      Point p = I.next();
      if (ccw(p0, p, f) == 0) {
        I.remove();
      } else {
        f = p;
    } }
    LinkedList<Point> S = new LinkedList<Point>();
    if (P.isEmpty()) {
      S.add(p0);
    }else{
      S.push(p0);
      S.push(P.get(0));
      for (int i = 1; i < P.size(); i++) {
        Point b = S.pop();
        Point a = S.peek();
        S.push(b);
        while (ccw(a, b, P.get(i)) <= 0) {
          S.pop();
          b = S.pop();
          a = S.peek();
          S.push(b);
        }
        S.push(P.get(i));
    } }
    return S;
  }
```

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))

```cpp
struct line {
  int x0;
  int y0;
  int x1;
  int y1;
};

// prueft, ob sich die Linien schneiden koennen
bool checkLines(line a, line b) {
  // Vektor Linie a
  int x0 = a.x1 - a.x0;
  int y0 = a.y1 - a.y0;
  // Vektor zu Startpunkt b
  int x1 = b.x0 - a.x0;
  int y1 = b.y0 - a.y0;
  // Vektor zu Endpunkt b
  int x2 = b.x1 - a.x0;
  int y2 = b.y1 - a.y0;
  // Kreuzprodukte berechnen
  int crossProduct1 = x0 * y1 + x1 * y0;
  int crossProduct2 = x0 * y2 + x2 * y0;
  // Wenn ein Produkt negativ, das andere positiv ist
      , koennen sich die Linien schneiden
  if (crossProduct1 * crossProduct2 < 0) {
      return true;
  }
  return false;
}
```

5.9 Punkt in Polygon

KreuzProdTest: -1: $A \to R$ 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

```java
  public static int KreuzProdTest(double ax, double ay
      , double bx, double by,
      double cx, double cy) {
    if (ay == by && by == cy) {
      if ((bx <= ax && ax <= cx) || (cx <= ax && ax <=
          bx))
        return 0;
      else
        return +1;
    }
    if(by>cy){double tmpx=bx;double tmpy=by; bx=cx;by=
        cy;cx=tmpx;cy=tmpy;}
    if(ay==by && ax==bx) return 0;
    if(ay<=by || ay>cy) return +1;
    double delta = (bx-ax)*(cy-ay)-(by-ay)*(cx-ax);
    if(delta>0)return -1; else if(delta<0)return +1;
        else return 0;
  }
  public static int PunktInPoly(double[] x,double[] y,
       double qx,double qy){
    int n = x.length - 1;
    int t = -1;
    for (int i = 0; i <= n - 1; i++) {
      t = t * KreuzProdTest(qx, qy, x[i], y[i], x[i +
          1], y[i + 1]);  }
    return t;
```

```
}
```

MD5: `38a79d6979334bc6a01381e15eef6e04`

5.10 Fläche eines Polygons

Input: Polygon-Koordinaten sortiert im Uhrzeigersinn

```java
static double area(List<Point> p) {
  double a = 0;
  Point q = p.get(p.size() - 1);
  Point r;
  for (Point r : p) {
    a += (q.x + r.x) * (q.y - r.y);
    q = r;
  }
  return a / -2;
}
```

MD5: `1f1dbdaaf78726c57e3e0ece63fe1cb3`

6 2-SAT-Solver

6.1 2-Sat mit SCC

```java
public class D_Manha {
  static class Node {
    ArrayList<Node> out = new ArrayList<Node>();
    ArrayList<Node> in = new ArrayList<Node>();
    int var;
    boolean explored = false;
    boolean discovered = false;
    int CCC;
    public Node(int v, String n) {
      var = v;
      name = n;
    }
  }
  static void impl(Node x, Node y){
    x.out.add(y);
    y.in.add(x);
  }
  public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    int n = in.nextInt();
    while (n-- > 0) {
      ArrayList<Node> graph; //TODO : implikationsgraph
      // Kosaraju
      S = new ArrayList<Node>();
      for (Node v : graph) {
        if (!v.explored) {
          DFS(v);
        }
      }
      for (Node v : graph) {
        v.explored = false;
        v.discovered = false;
      }
      int CCCidx = 0;
      do {
        ArrayList<Node> CCC = new ArrayList<Node>();
        DFSTrans(S.get(S.size()-1), CCC, CCCidx++);
        S.removeAll(CCC);
      } while (!S.isEmpty());

      boolean possible = true;
      for (int i = 1; i <= s; i++) {
        if (st.get(i).CCC == sf.get(i).CCC) {
          possible = false;
        }
      }
      for (int i = 1; i <= a; i++) {
        if (at.get(i).CCC == af.get(i).CCC) {
          possible = false;
        }}
      if (possible) {
        System.out.println("Yes");
      } else {
        System.out.println("No");
      }}}
  static ArrayList<Node> S;
  public static void DFS(Node v) {
    v.discovered = true;
    for (Node u : v.out) {
      if (!u.discovered) {
        DFS(u);
      }}
    v.explored = true;
    S.add(v);
  }
  public static void DFSTrans(Node v, ArrayList<Node> CCC, int CCCidx) {
    v.discovered = true;
    for (Node u : v.in) {
      if (!u.discovered) {
        DFSTrans(u, CCC, CCCidx);
      }}
    v.explored = true;
    CCC.add(v);
    v.CCC = CCCidx;
  }}
```

6.2 Hilfsalgorithmen

6.2.1 Erzeugen eines Graphens

```
SAT2Graph(φ = (α_1 ∨ β_1) ∧ ··· ∧ (α_m, β_m)){}
  G: Graph als Adjazenzliste
  for(int i =  0 < m; i++){
    jede Klausel liefert zwei Implikationen
    Fuege Kanten (−α_i, β_i), (−β_i, α_i) zu G hinzu.
}
```

6.2.2 Indexumrechnung

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

6.3 Suchen eines Pfades

```java
/**
Prueft mithilfe einer Breitensuche ob ein Weg
von Knoten x nach -x existiert
 */
 boolean BFSSATCheck(SATGraph G,int x) {
  boolean[] seen = new boolean[2 * n];
  Queue<Integer> queue;
  queue.add(x); seen[idx(x)] = true;
  while (!queue.isEmpty()) {
```

```
    Integer q = queue.poll();
    for (Integer p : G.get(idx(q))) {
      if (!seen[idx(p)]) {
        queue.add(p);
        seen[idx(p)] = true;
      }
      if (p == -x) return true;
    }
  }
  return seen[idx(-x)];
}
```

6.4 Algorithmus zum Prüfen der Erfüllbarkeit

```
/**
 Prueft ob fuer eine 2-CNF eine Belegung existiert
 */
boolean SAT2Check(φ = (α1 ∨ β1) ∧ ··· ∧ (αn, βn)){
  SAT2Graph G(φ) erzeugen
  for(int i = 0 < n; i++)
    if(BFSSATCheck(G,i) &&  BFSSATCheck(G,-i))
      return false;
      //Es gibt einen i → −i und −i → i Weg
  return true;
}
```

6.5 Algorithmus zur Belegung einer 2-CNF

```
/**
 * Ermittelt falls moeglich eine gueltige Belegung fuer
     eine 2-CNF
 */
Solve2SAT(φ = (α1 ∨ β1) ∧ ··· ∧ (αn, βn)){
  SAT2Graph G(φ) erzeugen
  vars = [0,...,0] //(2*n) Variablenbelegung
  assigned = [false,...,false] //(n+1) Belegung
      zugewiesen?

  for (int x = 1; x < n + 1; x++) {
    oldVars =  vars.clone();oldAssigned = assigned.
        clone();
    if (assign(vars, assigned, x))
      continue; //x:=1

    vars =  oldVars;assigned =  oldAssigned;
    if (!assign(vars, assigned, -x))
      return null;//x:=0 liefert auch keine Loesung
  }
  return vars; //gueltige Belegung
}

/**
 * Belegt die Variable x mit 1 und liefert false,
 * falls dies nicht moeglich ist
 * WICHTIG: Parameter werden veraendert (Referenzen
     uebergeben!).
 */
boolean assign(ArrayList<Integer> vars,
    ArrayList<Boolean> assigned, int x) {
  int xi = (x < 0) ? -x : x;
  if (assigned[xi]) return (vars[idx(x)] == 1);
  //Belege x, -x mit 0,1:
  vars[idx(x)]= 1;vars[idx(-x)]=0;
  assigned[xi] = true;
  for (Integer k : G.get(idx(x)))
    if (!assign(vars, assigned, k)) {
      //Belegung nicht weiter moeglich
      assigned[xi] = false; return false;
    }
  return true;
}
```

7 Verschiedenes

7.1 Potenzmenge

```
static <T> Iterator<List<T>> powerSet(final List<T> l)
    {
  return new Iterator<List<T>>() {
    int i; // careful: i becomes 2^l.size()
    public boolean hasNext() {
      return i < (1 << l.size());
    }
    public List<T> next() {
      Vector<T> temp = new Vector<T>();
      for (int j = 0; j < l.size(); j++)
        if (((i >>> j) & 1) == 1)
          temp.add(l.get(j));
      i++;
      return temp;
    }
    public void remove() {}
    };
  }
```

7.2 LongestCommonSubsequence

```
#include <iostream>
#include <vector>
#include <string>
#include <sstream>
#include <algorithm>
#include <iterator>
using namespace std;
#define MAX(a,b) (a > b) ? a : b

string X,Y;
vector< vector<int> > c(101, vector<int>(101,0));
int m,n,ctr;

int LCS(){
     m = X.length(),n=Y.length();
    c.resize(m+1);
  for(int i = 0; i<n+1; i++) {
    c[i].resize(n+1);
    c[i][0] = 0;
  }
     int i,j;
     for (i=0;i<=m;i++)
         for (j=0;j<=n;j++)
             c[i][j]=0;

     for (i=1;i<=m;i++)
         for (j=1;j<=n;j++)
         {
             if (X[i-1]==Y[j-1])
                c[i][j]=c[i-1][j-1]+1;
             else
                 c[i][j]=max(c[i][j-1],c[i-1][j]);
```

```
        }
     return c[m][n];
}
/** Print a songle LCS */
void printLCS(int i,int j){
    if (i==0 || j==0)
       return;
    if (X[i-1]==Y[j-1])
    {
       printLCS(i-1,j-1);
       cout<<X[i-1];
    }
    else if (c[i][j]==c[i-1][j])
         printLCS(i-1,j);
    else
        printLCS(i,j-1);
}

int main(){
    while(cin>>X>>Y)  {
  cout << "Length: " << LCS() << endl;
        printLCS(m,n);
        cout<<endl ;
    }}
```

7.3 LongestCommonSubstring

```
  private static List<String> longestCommonSubstring(
      String S1, String S2)
  {
    List<String> ret = new ArrayList<String>();
    List<Integer> idx  =new ArrayList<Integer>();
      int Start = 0;
      int Max = 0;
      for (int i = 0; i < S1.length(); i++)
      {
          for (int j = 0; j < S2.length(); j++)
          {
              int x = 0;
              while (S1.charAt(i + x) == S2.charAt(j +
                   x))
              {
                  x++;
                  if (((i + x) >= S1.length()) || ((j
                      + x) >= S2.length())) break;
              }
              if (x > Max)
              {
                  Max = x;
                Start = i;
                idx.clear();
                idx.add(Start);
              } else if(x==Max){
                Start = i;
                idx.add(Start);
              }
           }
      }
      HashSet<String> set = new HashSet<String>(idx.
          size(),1f);
      for(Integer start : idx){
        String substr = S1.substring(start,start+Max);
        if(!set.contains(substr)){
          ret.add(substr);
          set.add(substr);
        }
      }
      Collections.sort(ret);
      //return S1.substring(Start, (Start + Max));
      return ret;
  }
```

7.4 LongestIncreasingSubsequence

```
#include <vector>
using namespace std;

/** finde LIS in O(n log k)
 *a: Sequenz (in)
 *b: LIS (out)
 */
void find_lis(vector<int> &a, vector<int> &b)
{
  vector<int> p(a.size());
  int u, v;
  if (a.empty()) return;
  b.push_back(0);

  for (size_t i = 1; i < a.size(); i++)
        {
        // ist naechstes Element a[i] groesser als
            letztes der aktuelle LIS
    // a[b.back()], fuege es (Index) an "b" an.
    if (a[b.back()] < a[i]) {
      p[i] = b.back();
      b.push_back(i);
      continue;
    }

        // finde kleinstes El. in LIS (index in b)
            welches gerade groesser als a[i] ist
        // binaere suche |b|<=k  =>  O(log k)
    for (u = 0, v = b.size()-1; u < v;)
        {
      int c = (u + v) / 2;
      if (a[b[c]] < a[i]) u=c+1; else v=c;
    }

        // aktualisiere b falls neuer Wert kleiner als
             vorheriger kleinerer Wert
    if (a[i] < a[b[u]])
                {
      if (u > 0) p[i] = b[u-1];
      b[u] = i;
    }
  }

  for (u = b.size(), v = b.back(); u--; v = p[v]) b[u]
       = v;
}

#include <cstdio>
int main()
{
  int a[] = { 1, 9, 3, 8, 11, 4, 5, 6, 4, 19, 7, 1, 7
       };
  vector<int> seq(a, a+sizeof(a)/sizeof(a[0])); // seq
       : Eingabesequent
  vector<int> lis;                             // lis
       : Index Vektor fuer LIS
    find_lis(seq, lis);
     //Sequenz ausgeben:
  for (size_t i = 0; i < lis.size(); i++)
    printf("%d ", seq[lis[i]]);
```

```
        printf("\n");

  return 0;
}
```

7.5 Permutation & Sequenzen

```
import java.util.Scanner;
public class PermsAndSequ {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n;
    while ((n = sc.nextInt()) != 0) {
      int k = sc.nextInt();
      Sequences(n, k);
      Permutations(n);
    }
  }

  public static void Sequences(int n, int k) {
    int[] x = new int[k];
    for (int i = 0; i < k; i++)
      x[i] = 1;
    Print(x);
    while (true) {
      boolean lastX = true;
      for (int i = 0; i < k; i++)
        if (x[i] != n) {
          lastX = false;
          break;
        }
      if (lastX)
        break;
      int p = k - 1;
      while (!(x[p] < n))
        p--;
      x[p] = x[p] + 1;
      for (int i = p + 1; i < k; i++)
        x[i] = 1;
      Print(x);
    }
  }
  public static void Permutations(int n) {
    int[] x = new int[n];
    for (int i = 0; i < n; i++)
      x[i] = i + 1;
    Print(x);
    while (true) {
      boolean lastX = true;
      for (int i = 0; i < n - 1; i++)
        if (x[i] < x[i + 1]) {
          lastX = false;
          break;
        }
      if (lastX)  break;
      int k = n - 1 - 1;
      while (x[k] > x[k + 1]) k--;
      int t = k + 1;
      while (t < (n - 1) && x[t + 1] > x[k])
        t++;
      int tmp = x[k];
      x[k] = x[t];
      x[t] = tmp;
      // reverse x[k+1] ... x[n-1]
      for (int i = 0; i <= ((n - 1) - (k + 1)) / 2; i
          ++) {
        tmp = x[k + 1 + i];
        x[k + 1 + i] = x[n - 1 - i];
        x[n - 1 - i] = tmp;
      }
      Print(x);
    }
  }
  public static void Print(int[] x) {
    for (int i = 0; i < x.length; i++)
      System.out.print(x[i] + " ");
    System.out.println("");
  }

}
```

7.6 Knuth-Morris Pratt

Finds the first occurrence of the pattern in the text.

```
int match(String text, String pattern, int[] jump) {
  int j = 0;
  if (text.length() == 0)
    return -1;
  for (int i = 0; i < text.length(); i++) {
    while (j > 0 && pattern.charAt(j) != text.charAt(i
        ))
      j = jump[j - 1];
    if (pattern.charAt(j) == text.charAt(i))
      j++;
    if (j == pattern.length())
      return i - pattern.length() + 1;
  }
  return -1;}

// Computes the jump function
int[] computeJump(String pattern) {
  int[] jump = new int[pattern.length()];
  int j = 0;
  for (int i = 1; i < pattern.length(); i++) {
    while (j > 0 && pattern.charAt(j) != pattern.
        charAt(i))
      j = jump[j - 1];
    if (pattern.charAt(j) == pattern.charAt(i))
      j++;
    jump[i] = j;}
  return jump;}
```

MD5: `b5b9ca67a1df2c7c2913615bf1ed8a5b`

8 Formatierung & Sonstiges

8.1 Ausgabeformatierung mit JAVA - `DecimalFormat`

Symbol	Bedeutung
0	(Ziffer) – unbelegt wird eine Null angezeigt. (0.234=(00.00
#	(Ziffer) – unbelegt bleibt leer, (keine unnötigen nullen).
.	Dezimaltrenner.
,	Gruppiert die Ziffern (eine Gruppe ist so groß wie der Abs
;	Trennzeichen. Links Muster für pos., rechts für neg. Zahlen
-	Das Standardzeichen für Negativpräfix

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 Dzmpkt. bleiben
```

```
int i = 123;
printf( "|%d|   |%d|\n" ,      i, -i);    // |123|  |-123|
printf( "|%5d| |%5d|\n" ,      i, -i);    // |  123| | -123|
printf( "|%-5d| |%-5d|\n" ,    i, -i);    // |123  | |-123 |
printf( "|%+-5d| |%+-5d|\n" , i, -i);     // |+123 | |-123 |
printf( "|%05d| |%05d|\n\n",   i, -i);    // |00123| |-0123|
printf( "|%X| |%x|\n", 0xabc, 0xabc );    // |ABC| |abc|
printf( "|%08x| |%#x|\n\n", 0xabc, 0xabc ); // |00000abc| |0xabc|
double d = 1234.5678;
printf( "|%f| |%f|\n" ,          d, -d);   // |1234,5678
printf( "|%.2f| |%.2f|\n" ,      d, -d);   // |1234,57|
printf( "|%10f| |%10f|\n" ,      d, -d);   // |1234,5678
printf( "|%10.2f| |%10.2f|\n" , d, -d);    // |   1234,5
printf( "|%010.2f| |%010.2f|\n",d, -d);    // |0001234,5
String s = "Monsterbacke";
printf( "\n|%s|\n", s );                   // |Monsterba
printf( "|%20s|\n", s );                   // |        M
printf( "|%-20s|\n", s );                  // |Monsterba
printf( "|%7s|\n", s );                    // |Monsterba
printf( "|%.7s|\n", s );                   // |Monster|
printf( "|%20.7s|\n", s );                 // |
```

8.3 C++ Eingabe ohne bekannt Länge

```cpp
#include <iostream>
#include <sstream>
#include <istream>
#include <string>
#include <vector>
#include <cstdlib>

using namespace std;
int main(){
  string s;
  do{
    getline(cin,s);
    istringstream* ss;
    ss = new istringstream( s );
    while (!ss->eof())
    {
      string xs;
      getline( *ss, xs, '␣' );  // try to read the
          next field into it

      int x = atoi(xs.c_str());
      cout<<"␣"<<xs;
    }
    cout<<endl;
  } while(!cin.eof());
}
```

Theoretical Computer Science Cheat Sheet

Definitions

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

Series

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

In general:

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

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

Geometric series:

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

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

Harmonic series:

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

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

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

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

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

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

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

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

14. $\left[{n \atop 1}\right] = (n-1)!$, **15.** $\left[{n \atop 2}\right] = (n-1)!H_{n-1}$, **16.** $\left[{n \atop n}\right] = 1$, **17.** $\left[{n \atop k}\right] \ge \left\{{n \atop k}\right\}$,

18. $\left[{n \atop k}\right] = (n-1)\left[{n-1 \atop k}\right] + \left[{n-1 \atop k-1}\right]$, **19.** $\left\{{n \atop n-1}\right\} = \left[{n \atop n-1}\right] = \binom{n}{2}$, **20.** $\sum_{k=0}^{n}\left[{n \atop k}\right] = n!$, **21.** $C_n = \frac{1}{n+1}\binom{2n}{n}$,

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

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

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

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

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

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

Theoretical Computer Science Cheat Sheet

Identities Cont.

38. $\begin{bmatrix} n+1 \\ m+1 \end{bmatrix} = \sum_k \begin{bmatrix} n \\ k \end{bmatrix} \binom{k}{m} = \sum_{k=0}^{n} \begin{bmatrix} k \\ m \end{bmatrix} n^{\underline{n-k}} = n! \sum_{k=0}^{n} \frac{1}{k!} \begin{bmatrix} k \\ m \end{bmatrix},$

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

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

41. $\begin{bmatrix} n \\ m \end{bmatrix} = \sum_k \begin{bmatrix} n+1 \\ k+1 \end{bmatrix} \binom{k}{m} (-1)^{m-k},$

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

43. $\begin{bmatrix} m+n+1 \\ m \end{bmatrix} = \sum_{k=0}^{m} k(n+k) \begin{bmatrix} n+k \\ k \end{bmatrix},$

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

45. $(n-m)! \binom{n}{m} = \sum_k \begin{bmatrix} n+1 \\ k+1 \end{bmatrix} \left\{ {k \atop m} \right\} (-1)^{m-k}, \quad \text{for } n \geq m,$

46. $\left\{ {n \atop n-m} \right\} = \sum_k \binom{m-n}{m+k} \binom{m+n}{n+k} \begin{bmatrix} m+k \\ k \end{bmatrix},$

47. $\begin{bmatrix} n \\ n-m \end{bmatrix} = \sum_k \binom{m-n}{m+k} \binom{m+n}{n+k} \left\{ {m+k \atop k} \right\},$

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

49. $\begin{bmatrix} n \\ \ell+m \end{bmatrix} \binom{\ell+m}{\ell} = \sum_k \begin{bmatrix} k \\ \ell \end{bmatrix} \begin{bmatrix} n-k \\ m \end{bmatrix} \binom{n}{k}.$

Trees

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

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

Recurrences

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Generating functions:

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

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

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

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

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

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

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

So $g_i = 2^i - 1$.

Theoretical Computer Science Cheat Sheet

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

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

Pascal's Triangle

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

General

Bernoulli Numbers ($B_i = 0$, odd $i \neq 1$):
$$B_0 = 1,\ B_1 = -\tfrac{1}{2},\ B_2 = \tfrac{1}{6},\ B_4 = -\tfrac{1}{30},\ B_6 = \tfrac{1}{42},\ B_8 = -\tfrac{1}{30},\ B_{10} = \tfrac{5}{66}.$$
Change of base, quadratic formula:
$$\log_b x = \frac{\log_a x}{\log_a b}, \qquad \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$
Euler's number e:
$$e = 1 + \tfrac{1}{2} + \tfrac{1}{6} + \tfrac{1}{24} + \tfrac{1}{120} + \cdots$$
$$\lim_{n\to\infty} \left(1 + \frac{x}{n}\right)^n = e^x.$$
$$\left(1+\tfrac{1}{n}\right)^n < e < \left(1+\tfrac{1}{n}\right)^{n+1}.$$
$$\left(1+\tfrac{1}{n}\right)^n = e - \frac{e}{2n} + \frac{11e}{24n^2} - O\left(\frac{1}{n^3}\right).$$
Harmonic numbers:
$$1,\ \tfrac{3}{2},\ \tfrac{11}{6},\ \tfrac{25}{12},\ \tfrac{137}{60},\ \tfrac{49}{20},\ \tfrac{363}{140},\ \tfrac{761}{280},\ \tfrac{7129}{2520}, \ldots$$
$$\ln n < H_n < \ln n + 1,$$
$$H_n = \ln n + \gamma + O\left(\frac{1}{n}\right).$$
Factorial, Stirling's approximation:
$$1, 2, 6, 24, 120, 720, 5040, 40320, 362880, \ldots$$
$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1 + \Theta\left(\frac{1}{n}\right)\right).$$
Ackermann's function and inverse:
$$a(i,j) = \begin{cases} 2^j & i = 1 \\ a(i-1, 2) & j = 1 \\ a(i-1, a(i, j-1)) & i, j \geq 2 \end{cases}$$
$$\alpha(i) = \min\{j \mid a(j,j) \geq i\}.$$

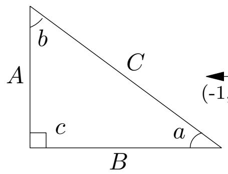
Binomial distribution:
$$\Pr[X = k] = \binom{n}{k} p^k q^{n-k}, \qquad q = 1 - p,$$
$$\mathrm{E}[X] = \sum_{k=1}^{n} k \binom{n}{k} p^k q^{n-k} = np.$$
Poisson distribution:
$$\Pr[X = k] = \frac{e^{-\lambda}\lambda^k}{k!}, \quad \mathrm{E}[X] = \lambda.$$
Normal (Gaussian) distribution:
$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}, \quad \mathrm{E}[X] = \mu.$$
The "coupon collector": We are given a random coupon each day, and there are n different types of coupons. The distribution of coupons is uniform. The expected number of days to pass before we to collect all n types is
$$nH_n.$$

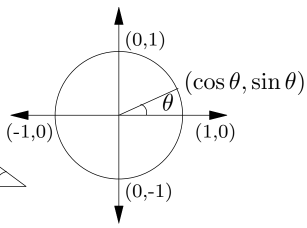
Probability

Continuous distributions: If
$$\Pr[a < X < b] = \int_a^b p(x)\,dx,$$
then p is the probability density function of X. If
$$\Pr[X < a] = P(a),$$
then P is the distribution function of X. If P and p both exist then
$$P(a) = \int_{-\infty}^{a} p(x)\,dx.$$
Expectation: If X is discrete
$$\mathrm{E}[g(X)] = \sum_x g(x)\Pr[X = x].$$
If X continuous then
$$\mathrm{E}[g(X)] = \int_{-\infty}^{\infty} g(x)p(x)\,dx = \int_{-\infty}^{\infty} g(x)\,dP(x).$$
Variance, standard deviation:
$$\mathrm{VAR}[X] = \mathrm{E}[X^2] - \mathrm{E}[X]^2,$$
$$\sigma = \sqrt{\mathrm{VAR}[X]}.$$
For events A and B:
$$\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$$
$$\Pr[A \wedge B] = \Pr[A] \cdot \Pr[B],$$
iff A and B are independent.
$$\Pr[A|B] = \frac{\Pr[A \wedge B]}{\Pr[B]}$$
For random variables X and Y:
$$\mathrm{E}[X \cdot Y] = \mathrm{E}[X] \cdot \mathrm{E}[Y],$$
if X and Y are independent.
$$\mathrm{E}[X + Y] = \mathrm{E}[X] + \mathrm{E}[Y],$$
$$\mathrm{E}[cX] = c\,\mathrm{E}[X].$$
Bayes' theorem:
$$\Pr[A_i|B] = \frac{\Pr[B|A_i]\Pr[A_i]}{\sum_{j=1}^{n}\Pr[A_j]\Pr[B|A_j]}.$$
Inclusion-exclusion:
$$\Pr\left[\bigvee_{i=1}^{n} X_i\right] = \sum_{i=1}^{n}\Pr[X_i] + \sum_{k=2}^{n}(-1)^{k+1}\sum_{i_i < \cdots < i_k}\Pr\left[\bigwedge_{j=1}^{k} X_{i_j}\right].$$
Moment inequalities:
$$\Pr\left[|X| \geq \lambda\,\mathrm{E}[X]\right] \leq \frac{1}{\lambda},$$
$$\Pr\left[\left|X - \mathrm{E}[X]\right| \geq \lambda \cdot \sigma\right] \leq \frac{1}{\lambda^2}.$$
Geometric distribution:
$$\Pr[X = k] = pq^{k-1}, \qquad q = 1 - p,$$
$$\mathrm{E}[X] = \sum_{k=1}^{\infty} kpq^{k-1} = \frac{1}{p}.$$

Theoretical Computer Science Cheat Sheet

Trigonometry





Pythagorean theorem:
$$C^2 = A^2 + B^2.$$

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

Area, radius of inscribed circle:
$$\tfrac{1}{2}AB, \quad \frac{AB}{A+B+C}.$$

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

Euler's equation:
$$e^{ix} = \cos x + i \sin x, \qquad e^{i\pi} = -1.$$

Matrices

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

Determinants: $\det A \neq 0$ iff A is non-singular.
$$\det A \cdot B = \det A \cdot \det B,$$
$$\det A = \sum_{\pi} \prod_{i=1}^{n} \operatorname{sign}(\pi) a_{i,\pi(i)}.$$

2×2 and 3×3 determinant:
$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc,$$
$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = g \begin{vmatrix} b & c \\ e & f \end{vmatrix} - h \begin{vmatrix} a & c \\ d & f \end{vmatrix} + i \begin{vmatrix} a & b \\ d & e \end{vmatrix}$$
$$= \begin{array}{l} aei + bfg + cdh \\ \quad - ceg - fha - ibd. \end{array}$$

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

Hyperbolic Functions

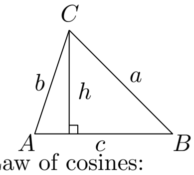
Definitions:
$$\sinh x = \frac{e^x - e^{-x}}{2}, \qquad \cosh x = \frac{e^x + e^{-x}}{2},$$
$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \qquad \operatorname{csch} x = \frac{1}{\sinh x},$$
$$\operatorname{sech} x = \frac{1}{\cosh x}, \qquad \coth x = \frac{1}{\tanh x}.$$

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

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

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

More Trig.



Law of cosines:
$$c^2 = a^2 + b^2 - 2ab \cos C.$$

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

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

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

Theoretical Computer Science Cheat Sheet

Number Theory

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

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

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

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

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

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

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

Fermat's theorem:

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

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

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

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

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

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

Wilson's theorem: n is a prime iff

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

Möbius inversion:

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

If

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

then

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

Prime numbers:

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

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

Graph Theory

Definitions:

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

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

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

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

Any planar graph has a vertex with degree ≤ 5.

Notation:

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

Geometry

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

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

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

Distance formula, L_p and L_∞ metric:

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

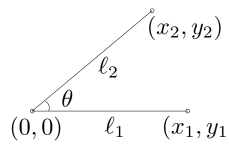
$$\left[\lvert x_1 - x_0\rvert^p + \lvert y_1 - y_0\rvert^p\right]^{1/p},$$

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

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

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

Angle formed by three points:



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

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

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

Area of circle, volume of sphere:

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

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

Theoretical Computer Science Cheat Sheet

π

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

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

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

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

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

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

Partial Fractions

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

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

Calculus

Derivatives:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Integrals:

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

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

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

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

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

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

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

Theoretical Computer Science Cheat Sheet

Calculus Cont.

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

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

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

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

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

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

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

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

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

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

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

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

27. $\int \sinh x\,dx = \cosh x,$

28. $\int \cosh x\,dx = \sinh x,$

29. $\int \tanh x\,dx = \ln|\cosh x|,$

30. $\int \coth x\,dx = \ln|\sinh x|,$

31. $\int \operatorname{sech} x\,dx = \arctan\sinh x,$

32. $\int \operatorname{csch} x\,dx = \ln\left|\tanh\frac{x}{2}\right|,$

33. $\int \sinh^2 x\,dx = \frac{1}{4}\sinh(2x) - \frac{1}{2}x,$

34. $\int \cosh^2 x\,dx = \frac{1}{4}\sinh(2x) + \frac{1}{2}x,$

35. $\int \operatorname{sech}^2 x\,dx = \tanh x,$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Theoretical Computer Science Cheat Sheet

Calculus Cont.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Finite Calculus

Difference, shift operators:

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

Fundamental Theorem:

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

Differences:

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

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

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

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

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

Sums:

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

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

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

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

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

Falling Factorial Powers:

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

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

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

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

Rising Factorial Powers:

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

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

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

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

Conversion:

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

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

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

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

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

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

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

Theoretical Computer Science Cheat Sheet

Series

Taylor's series:

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

Expansions:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{1}{\sqrt{1-4x}} = 1 + 2x + 6x^2 + 20x^3 + \cdots = \sum_{i=0}^{\infty} \binom{2i}{i} x^i,$$

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

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

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

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

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

Ordinary power series:

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

Exponential power series:

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

Dirichlet power series:

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

Binomial theorem:

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

Difference of like powers:

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

For ordinary power series:

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

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

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

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

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

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

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

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

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

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

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

Convolution:

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

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

Theoretical Computer Science Cheat Sheet

Series

Expansions:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Escher's Knot

Stieltjes Integration

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

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

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

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

If the integrals involved exist

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

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

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

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

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

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

Cramer's Rule

If we have equations:

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

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

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

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

Fibonacci Numbers

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

Definitions:

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

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

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

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

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

Additive rule:

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

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

Calculation by matrices:

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

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

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

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

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