



Team Contest Reference

Team:

System.out.println(42);

*Roland Haase
Thore Tiemann
Marcel Wienöbst*

Contents

1	DataStructures	2
1.1	Fenwick-Tree	2
1.2	Range Maximum Query	2
1.3	Union-Find	2
2	Graph	3
2.1	Breadth First Search	3
2.2	BellmanFord	3
2.3	Bipartite Graph Check	3
2.4	Maximum Bipartite Matching	4
2.5	Single-source shortest paths in dag	4
2.6	Dijkstra	4
2.7	EdmondsKarp	5
2.8	Reference for Edge classes	5
2.9	FloydWarshall	5
2.10	terative DFS	6
2.11	Johnsons Algorithm	6
2.12	Kruskal	6
2.13	Prim	6
2.14	Recursive Depth First Search	7
2.15	Strongly Connected Components	7
2.16	Topological Sort	8
2.17	Reference for Vertex classes	8
3	Math	8
3.1	Binomial Coefficient	8
3.2	Binomial Matrix	8
3.3	Divisability	9
3.4	Iterative EEA	9
3.5	Polynomial Interpolation	9
3.6	Sieve of Eratosthenes	10
4	Misc	10
4.1	Binary Search	10
4.2	Next number with n bits set	10
5	Math Roland	11
5.1	Divisability Explanation	11
5.2	Combinatorics	11
5.3	Polynomial Interpolation	11
5.3.1	Theory	11
5.4	Fibonacci Sequence	11
5.4.1	Binet's formula	11
5.4.2	Generalization	11
5.4.3	Pisano Period	11

6 Java Knowhow

11

6.1	System.out.printf() und String.format()	11
6.2	Modulo: Avoiding negative Integers	11
6.3	Speed up IO	11

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

byte (8 Bit, signed): -128 ... 127

short (16 Bit, signed): -32.768 ... 23.767

integer (32 Bit, signed): -2.147.483.648 ... 2.147.483.647

long (64 Bit, signed): $-2^{63} \dots 2^{63} - 1$

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

1 DataStructures

1.1 Fenwick-Tree

Can be used for computing prefix sums.

```

1 int[] fwktree = new int[m + n + 1];
2 public static int read(int index, int[] fenwickTree) {
3     int sum = 0;
4     while (index > 0) {
5         sum += fenwickTree[index];
6         index -= (index & -index);
7     }
8     return sum;
9 }
10 public static int[] update(int index, int addValue,
11     int[] fenwickTree) {
12     while (index <= fenwickTree.length - 1) {
13         fenwickTree[index] += addValue;
14         index += (index & -index);
15     }
16     return fenwickTree;
17 }
```

MD5: 97fd176a403e68cb76a82196191d5f19 | $\mathcal{O}(\log n)$

1.2 Range Maximum Query

process processes an array A of length N in $\mathcal{O}(N \log N)$ such that *query* can compute the maximum value of A in interval $[i, j]$. Therefore $M[a, b]$ stores the maximum value of interval $[a, a + 2^b - 1]$.

Input: dynamic table M , array to search A , length N of A , start index i and end index j

Output: filled dynamic table M or the maximum value of A in interval $[i, j]$

```

1 public static void process(int[][] M, int[] A, int N)
2 {
3     for(int i = 0; i < N; i++)
4         M[i][0] = i;
5     // filling table M
6     // M[i][j] = max(M[i][j-1], M[i+(1<<(j-1))][j-1]),
7     // cause interval of length 2^j can be partitioned
8     // into two intervals of length 2^(j-1)
9     for(int j = 1; 1 <= j <= N; j++) {
10         for(int i = 0; i + (1 << j) - 1 < N; i++) {
11             if(A[M[i][j-1]] >= A[M[i+(1 << (j-1))][j-1]])
12                 M[i][j] = M[i][j-1];
13             else
14                 M[i][j] = M[i + (1 << (j-1))][j-1];
15         }
16     }
17 }
18 public static int query(int[][] M, int[] A, int N,
19     int i, int j) {
20     // k = |_ log_2(j-i+1) _|
21     int k = (int) (Math.log(j - i + 1) / Math.log(2));
22     if(A[M[i][k]] >= A[M[j - (1 << k) + 1][k]])
23         return M[i][k];
24     else
25         return M[j - (1 << k) + 1][k];
26 }
```

MD5: db0999fa40037985ff27dd1a43c53b80 | $\mathcal{O}(N \log N, 1)$

1.3 Union-Find

Union-Find is a data structure that keeps track of a set of elements partitioned into a number of disjoint subsets. *UnionFind* creates n disjoint sets each containing one element. *union* joins the sets x and y are contained in. *find* returns the representative of the set x is contained in.

Input: number of elements n , element x , element y

Output: the representative of element x or a boolean indicating whether sets got merged.

```

1 class UnionFind {
2     private int[] p = null;
3     private int[] r = null;
4     private int count = 0;
5
6     public int count() {
7         return count;
8     } // number of sets
9
10    public UnionFind(int n) {
11        count = n; // every node is its own set
12        r = new int[n]; // every node is its own tree
13        // with height 0
14        p = new int[n];
15        for (int i = 0; i < n; i++)
```

```

15     p[i] = -1; // no parent = -1
16 }
17
18 public int find(int x) {
19     int root = x;
20     while (p[root] >= 0) { // find root
21         root = p[root];
22     }
23     while (p[x] >= 0) { // path compression
24         int tmp = p[x];
25         p[x] = root;
26         x = tmp;
27     }
28     return root;
29 }
30
31 // return true, if sets merged and false, if
32 // already from same set
33 public boolean union(int x, int y) {
34     int px = find(x);
35     int py = find(y);
36     if (px == py)
37         return false; // same set -> reject edge
38     if (r[px] < r[py]) { // swap so that always h[px]
39         // >= h[py]
40         int tmp = px;
41         px = py;
42         py = tmp;
43     }
44     p[py] = px; // hang flatter tree as child of
45     // higher tree
46     r[px] = Math.max(r[px], r[py] + 1); // update (
47     // worst-case) height
48     count--;
49     return true;
50 }
51 }

```

MD5: 5c507168e1ffd9ead25babf7b3769cfd | $\mathcal{O}(\alpha(n))$

2 Graph

2.1 Breadth First Search

Iterative BFS. Needs testing. Uses ref Vertex class, no Edge class needed. In this version we look for a shortest path from s to t though we could also find the BFS-tree by leaving out t . *Input:* IDs of start and goal vertex and graph as AdjList *Output:* true if there is a connection between s and g , false otherwise

```

1 public static boolean BFS(Vertex[] G, int s, int t) {
2
3     //make sure that all Vertices vis values are false
4     //etc
5
6     Queue<Vertex> q = new LinkedList<Vertex>();
7
8     G[s].vis = true;
9     G[s].dist = 0;
10    G[s].pre = -1;
11    q.add(G[s]);
12
13    //expand frontier between undiscovered and
14    //discovered vertices
15    while(!q.isEmpty()) {

```

```

14    Vertex u = q.poll();
15    //when reaching the goal, return true
16    //if we want to construct a BFS-tree delete this
17    //line
18    if(u.id == t) return true;
19    //else add adj vertices if not visited
20    for(Vertex v : u.adj) {
21        if(!v.vis) {
22            v.vis = true;
23            v.dist = u.dist + 1;
24            v.pre = u.id;
25            q.add(v);
26        }
27    }
28 }
29 }

```

MD5: 01c4dadba37bb0e95625e8522e3f6362 | $\mathcal{O}(|V| + |E|)$

2.2 BellmanFord

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

```

1 public static boolean bellmanFord(Vertex[] G) {
2     //source is 0
3     G[0].dist = 0;
4     //calc distances
5     //the path has max length |V|-1
6     for(int i = 0; i < G.length-1; i++) {
7         //each iteration relax all edges
8         for(int j = 0; j < G.length; j++) {
9             for(Edge e : G[j].adj) {
10                if(G[j].dist != Integer.MAX_VALUE
11                    && e.t.dist > G[j].dist + e.w) {
12                    e.t.dist = G[j].dist + e.w;
13                }
14            }
15        }
16    }
17    //check for negative-length cycle
18    for(int i = 0; i < G.length; i++) {
19        for(Edge e : G[i].adj) {
20            if(G[i].dist != Integer.MAX_VALUE && e.t.dist
21                > G[i].dist + e.w) {
22                return true;
23            }
24        }
25    }
26    return false;
27 }

```

MD5: d101e6b6915f012b3f0c02dc79e1fc6f | $\mathcal{O}(|V| \cdot |E|)$

2.3 Bipartite Graph Check

Checks a graph represented as adjList for being bipartite. Needs a little adaption, if the graph is not connected.

Input: graph as adjList, amount of nodes N as int

Output: true if graph is bipartite, false otherwise

```

1 public static boolean bipartiteGraphCheck(Vertex[] G)
2 {

```

```

3 // use bfs for coloring each node
4 G[0].color = 1;
5 Queue<Vertex> q = new LinkedList<Vertex>();
6 q.add(G[0]);
7 while(!q.isEmpty()) {
8     Vertex u = q.poll();
9     for(Vertex v : u.adj) {
10         // if node i not yet visited,
11         // give opposite color of parent node u
12         if(v.color == -1) {
13             v.color = 1-u.color;
14             q.add(v);
15             // if node i has same color as parent node u
16             // the graph is not bipartite
17             } else if(u.color == v.color)
18                 return false;
19             // if node i has different color
20             // than parent node u keep going
21         }
22     }
23     return true;
24 }

```

MD5: e93d242522e5b4085494c86f0d218dd4 | $\mathcal{O}(|V| + |E|)$

2.4 Maximum Bipartite Matching

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

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

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

```

1 // A DFS based recursive function that returns true
2 // if a matching for vertex u is possible
3 boolean bpm(boolean bpGraph[][], int u,
4             boolean seen[], int matchR[]) {
5     // Try every job one by one
6     for (int v = 0; v < N; v++) {
7         // If applicant u is interested in job v and v
8         // is not visited
9         if (bpGraph[u][v] && !seen[v]) {
10             seen[v] = true; // Mark v as visited
11
12             // If job v is not assigned to an applicant OR
13             // previously assigned applicant for job v (which
14             // is matchR[v]) has an alternate job available.
15             // Since v is marked as visited in the above line,
16             // matchR[v] in the following recursive call will
17             // not get job v again
18             if (matchR[v] < 0 ||
19                 bpm(bpGraph, matchR[v], seen, matchR)) {
20                 matchR[v] = u;
21                 return true;
22             }
23         }
24     }
25     return false;
26 }

```

```

27
28 // Returns maximum number of matching from M to N
29 int maxBPM(boolean bpGraph[][]) {
30     // An array to keep track of the applicants assigned
31     // to jobs. The value of matchR[i] is the applicant
32     // number assigned to job i, the value -1 indicates

```

```

33 // nobody is assigned.
34 int matchR[] = new int[N];
35
36 // Initially all jobs are available
37 for(int i = 0; i < N; ++i)
38     matchR[i] = -1;
39 // Count of jobs assigned to applicants
40 int result = 0;
41 for (int u = 0; u < M; u++) {
42     // Mark all jobs as not seen for next applicant.
43     boolean seen[] = new boolean[N];
44     for(int i = 0; i < N; ++i)
45         seen[i] = false;
46
47     // Find if the applicant u can get a job
48     if (bpm(bpGraph, u, seen, matchR))
49         result++;
50 }
51 return result;
52 }

```

MD5: e559cef1fc0d34e0ba49b7568cfd480d | $\mathcal{O}(M \cdot N)$

2.5 Single-source shortest paths in dag

```

1 public static void dagSSP(Vertex[] G, int s) {
2     //calls topological sort method
3     LinkedList<Integer> sorting = TS(G);
4
5     G[s].dist = 0;
6
7     //go through vertices in ts order
8     for(int u : sorting) {
9         for(Edge e : G[u].adj) {
10             Vertex v = e.t;
11             if(v.dist > u.d + e.w) {
12                 v.dist = u.d + e.w;
13                 v.pre = u.id;
14             }
15         }
16     }
17 }

```

MD5: 3fc829298eb1489b255acd3427d89d1a | $\mathcal{O}(|V| + |E|)$

2.6 Dijkstra

Finds the shortest paths from one vertex to every other vertex in the graph (SSSP).

For negative weights, add $|\min|+1$ to each edge, later subtract from result.

To get a different shortest path when edges are ints, add an $\epsilon = \frac{1}{k+1}$ on each edge of the shortest path of length k , run again.

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

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

```

1 public static void dijkstra(Vertex[] G, int s) {
2     G[s].dist = 0;
3
4     //Tuple class can be found at Prim's Alg, maybe we
5     //should give this class its own space
6     Tuple st = new Tuple(s, 0);

```

```

6
7     PriorityQueue<Tuple> q = new PriorityQueue<Tuple>
8         >();
9     q.add(G[s]);
10
11     while(!q.isEmpty()) {
12         Tuple sm = q.poll();
13         Vertex u = G[sm.id];
14
15         if(u.vis) continue;
16         if(sm.dist > u.dist) continue;
17         u.vis = true;
18         for(Edge e : u.adj) {
19             Vertex v = e.t;
20             if(!v.vis && v.dist > u.dist + e.w) {
21                 v.pre = u.id;
22                 v.dist = u.dist + e.w;
23                 Tuple nt = new Tuple(v.id, v.dist);
24                 queue.add(nt);
25             }
26         }
27     }

```

MD5: 15598cf27ada41bf8cdf83dd5d3301bf | $\mathcal{O}(|E| \log |V|)$

2.7 EdmondsKarp

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

```

1 public static boolean BFS(Vertex[] G, int s, int t) {
2     int N = G.length;
3     for(int i = 0; i < N; i++) {
4         G[i].vis = false;
5     }
6
7     Queue<Vertex> q = new LinkedList<Vertex>();
8     G[s].vis = true;
9     G[s].pre = -1;
10    queue.add(G[s]);
11
12    while(!q.isEmpty()) {
13        Vertex u = queue.poll();
14        if(u.id == t) return true;
15        for(int i : u.adj.keySet()) {
16            Edge e = u.adj.get(i);
17            Vertex v = e.t;
18            if(!v.vis) {
19                v.vis = true;
20                v.pre = u.id;
21                q.add(v);
22            }
23        }
24    }
25    return (G[t].vis);
26 }
27 //We store the edges in the graph in a hashmap
28 public static int fordFulkerson(Vertex[] G, int s, int
29     t) {
30     int maxflow = 0;
31     while(BFS(rgraph, s, t)) {
32         int pathflow = Integer.MAX_VALUE;
33         for(int v = t; v != s; v = v.pre) {
34             int u = v.pre;
35             pathflow = Math.min(pathflow, G[u].adj.get(v).rw);
36         }

```

```

37
38         for(int v = t; v != s; v = v.pre) {
39             int u = v.pre;
40             G[u].adj.get(v).rw -= pathflow;
41             G[v].adj.get(u).rw += pathflow;
42         }
43
44         maxflow += pathflow;
45     }
46     return maxflow;
47 }

```

MD5: b5e1ff020addc8138cde5398ec518985 | $\mathcal{O}(|V|^2 \cdot |E|)$

2.8 Reference for Edge classes

Used for example in Dijkstra algorithm, implements edges with weight. Needs testing.

```

1 //for Kruskal we need to sort edges, use:
2 class Edge implements Comparable<Edge> {}
3
4 class Edge {
5
6     //for Kruskal it is helpful to store the start as
7     //well
8     //moreover we might not need the vertex class
9     int s;
10    int t;
11
12    public Edge(int s, int t, int w) {...}
13    public int compareTo(Edge other) {
14        return Integer.compare(this.w, other.w);
15    }
16
17    //for EKarp we also want to store residual weights
18    int rw;
19
20    Vertex t;
21    int w;
22
23    public Edge(Vertex t, int w) {
24        this.t = t;
25        this.w = w;
26        this.rw = w;
27    }
28 }

```

MD5: fd4ed227f042ee49ef9dac031ad2d5a0 | $\mathcal{O}(?)$

2.9 FloydWarshall

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

```

1 public static void floydWarshall(int[][] graph, int
2     [][] next, int[][] ans) {
3     for(int i = 0; i < ans.length; i++) {
4         for(int j = 0; j < ans.length; j++) {
5             ans[i][j] = graph[i][j];
6         }
7     }
8     for (int k = 0; k < ans.length; k++) {
9         for (int i = 0; i < ans.length; i++) {
10            for (int j = 0; j < ans.length; j++) {
11                if (ans[i][k] + ans[k][j] < ans[i][j])

```

```

11         && ans[i][k] < Integer.MAX_VALUE && ans[k
12             ][j] < Integer.MAX_VALUE) {
13             ans[i][j] = ans[i][k] + ans[k][j];
14             next[i][j] = next[i][k];
15         }
16     }
17 }
18 }

```

MD5: 4faf8c41a9070f106e68864cc131706d | $\mathcal{O}(|V|^3)$

2.10 Iterative DFS

Simple iterative DFS, the recursive variant is a bit fancier. Not tested.

```

1 //if we want to start the DFS for different connected
  components, there is such a method
2 //in the recursive variant of DFS
3
4 public static boolean ItDFS(Vertex[] G, int s, int t)
5 {
6     //take care that all the nodes are not visited at
7     the beginning
8
9     Stack<Integer> S = new Stack<Integer>();
10    s.push(s);
11    while(!S.isEmpty()) {
12        int u = S.pop();
13        if(u.id == t) return true;
14        if(!G[u].vis) {
15            G[u].vis = true;
16            for(Vertex v : G[u].adj) {
17                if(!v.vis) S.push(v.id);
18            }
19        }
20    }
21    return false;
22 }

```

MD5: 1f83d8077e6252b6894eb5711298d79c | $\mathcal{O}(|V| + |E|)$

2.11 Johnsons Algorithm

```

1 public static int[][] johnson(Vertex[] G) {
2
3     Vertex[] Gd = new Vertex[G.length+1];
4     int s = G.length;
5     for(int i = 0; i < G.length; i++) {
6         Gd[i] = G[i];
7     }
8     //init new vertex with zero-weight-edges to each
9     vertex
10    Vertex S = new Vertex(G.length);
11    for(int i = 0; i < G.length) {
12        S.adj.add(new Edge(Gd[i], 0));
13    }
14
15    //bellman-ford to check for neg-weight-cycles and
16    to adapt edges to enable running dijkstra
17    if(!bellmanFord(G, s)) {
18        System.out.println("False");
19    }
20 }

```

```

17 return;
18 }
19 //change weights
20 for(int i = 0; i < G.length; i++) {
21     for(Edge e : Gd[i].adj) {
22         e.w = e.w + Gd[i].dist - e.t.dist;
23     }
24 }
25 //store distances to invert this step later
26 int[] h = new int[G.length];
27 for(int i = 0; i < G.length; i++) {
28     h[i] = G[i].dist;
29 }
30
31 //create shortest path matrix
32 int[][] apsp = new int[G.length][G.length];
33
34 //now use original graph G
35 //start a dijkstra for each vertex
36 for(int i = 0; i < G.length; i++) {
37     //reset weights, maybe we should put that in the
38     dijkstra
39     for(int j = 0; j < G.length; j++) {
40         G[j].vis = false;
41         G[j].dist = Integer.MAX_VALUE;
42     }
43     dijkstra(G, i);
44     for(int j = 0; j < G.length; j++) {
45         apsp[i][j] = G[j].dist + h[j] - h[i];
46     }
47 }
48 return apsp;
49 }

```

MD5: 6bce8e864871064f450e0115a9ab77df | $\mathcal{O}(|V|^2 \log V + VE)$

2.12 Kruskal

Computes a minimum spanning tree for a weighted undirected graph.

```

1 public static int kruskal(Edge[] edges, int n) {
2     Arrays.sort(edges);
3     //n is the number of vertices
4     UnionFind uf = new UnionFind(n);
5     //we will only compute the sum of the MST, one
6     could of course also store the edges
7
8     int sum = 0;
9     int cnt = 0;
10    for(int i = 0; i < edges.length; i++) {
11        if(cnt == n-1) break;
12        if(uf.union(edges[i].s, edges[i].t)) {
13            sum += edges[i].w;
14            cnt++;
15        }
16    }
17    return sum;
18 }

```

MD5: aa6cc91ea8a00f6b38aa0433130d1be9 | $\mathcal{O}(|E| + \log |V|)$

2.13 Prim

```

1 //s is the startpoint of the algorithm, in general not
  too important

```

```

2 //we assume that graph is connected
3 public static int prim(Vertex[] G, int s) {
4
5     //make sure dists are maxint
6     G[s].dist = 0;
7     Tuple st = new Tuple(s, 0);
8
9     PriorityQueue q = new PriorityQueue
10         ();
11     q.add(st);
12
13     //we will store the sum and each nodes predecessor
14     int sum = 0;
15
16     while(!q.isEmpty()) {
17         Tuple sm = q.poll();
18         Vertex u = G[sm.id];
19         //u has been visited already
20         if(u.vis) continue;
21         //this is not the latest version of u
22         if(sm.dist > u.dist) continue;
23         u.vis = true;
24         //u is part of the new tree and u.dist the cost of
25         //adding it
26         sum += u.dist;
27         for(Edge e : u.adj) {
28             Vertex v = e.t;
29             if(!v.vis && v.dist > e.w) {
30                 v.pre = u.id;
31                 v.dist = e.w;
32                 Tuple nt = new Tuple(v.id, e.w);
33                 q.add(nt);
34             }
35         }
36     }
37     return sum;
38 }
39
40 class Tuple implements Comparable {
41
42     int id;
43     int dist;
44
45     public Tuple(int id, int dist) {
46         this.id = id;
47         this.dist = dist;
48     }
49
50     public int compareTo(Tuple other) {
51         return Integer.compare(this.dist, other.dist);
52     }
53 }

```

MD5: 1c35fcc2a3f44ab7c1658d2716805ee1 | $\mathcal{O}()$

2.14 Recursive Depth First Search

Recursive DFS with different options (storing times, connected/unconnected graph). Needs testing. *Input:* A source vertex s , a target vertex t , and adjlist G and the time (0 at the start) *Output:* Indicates if there is connection between s and t .

```

1 //if we want to visit the whole graph, even if it is
  not connected we might use this

```

```

2 public static void DFS(Vertex[] G) {
3
4     //make sure all vertices vis value is false etc
5
6     int time = 0;
7     for(int i = 0; i < G.length; i++) {
8         if(!G[i].vis) {
9             //note that we leave out t so this does not work
10             //with the below function
11             //adaption will not be too difficult though
12             //fix time
13             recDFS(i, G, 0);
14         }
15     }
16
17     //first call with time = 0
18     public static boolean recDFS(int s, int t, Vertex[] G,
19         int time){
20
21         //it might be necessary to store the time of
22         //discovery
23         time = time + 1;
24         G[s].dtime = time;
25
26         G[s].vis = true; //new vertex has been discovered
27         //when reaching the target return true
28         //not necessary when calculating the DFS-tree
29         if(s == t) return true;
30         for(Vertex v : G[s].adj) {
31             //exploring a new edge
32             if(!v.vis) {
33                 v.pre = u.id;
34                 if(recDFS(v.id, t, G)) return true;
35             }
36         }
37
38         //storing finishing time
39         time = time + 1;
40         G[s].ftime = time;
41
42         return false;
43     }
44 }

```

MD5: e11b8416945db1004b13346a22341c87 | $\mathcal{O}(|V| + |E|)$

2.15 Strongly Connected Components

```

1 public static void fDFS(Vertex u, LinkedList<Integer>
2     sorting) {
3     //compare with TS
4     u.vis = true;
5     for(Vertex v : u.out) {
6         if(!v.vis)
7             fDFS(v, sorting);
8     }
9     sorting.addFirst(u.id);
10    return sorting;
11 }
12
13 public static void sDFS(Vertex u, int cnt) {
14     //basic DFS, all visited vertices get cnt
15     u.vis = true;
16     u.comp = cnt;
17     for(Vertex v : u.in) {
18         if(!v.vis)
19             sDFS(v, cnt);
20     }
21 }

```



```

19 }
20 }
21
22 public static void doubleDFS(Vertex[] G) {
23     //first calc a topological sort by first DFS
24     LinkedList<Integer> sorting = new LinkedList<Integer>
25         >();
26     for(int i = 0; i < G.length; i++) {
27         if(!G[i].vis)
28             fDFS(G[i], sorting);
29     }
30     for(int i = 0; i < G.length; i++){
31         G[i].vis = false;
32     }
33     //then go through the sort and do another DFS on G^T
34     //each tree is a component and gets a unique number
35     int cnt = 0;
36     for(int i : sorting) {
37         if(!G[i].vis)
38             sDFS(G[i], cnt++);
39     }

```

MD5: 67ac4aac19ee3ce07f23dd8ed9877b23 | $\mathcal{O}(|V| + |E|)$

2.16 Topological Sort

```

1 public static LinkedList<Integer> TS(Vertex[] G) {
2     LinkedList<Integer> sorting = new LinkedList<Integer>
3         >();
4     for(int i = 0; i < G.length; i++) {
5         if(!G[i].vis)
6             recTS(G[i], sorting);
7     }
8     //check sorting for a -1 if the graph is not
9     //necessarily dag
10    //maybe checking if there are too many values in
11    //sorting is easier?!
12    return sorting;
13 }
14
15 public static LinkedList<Integer> recTS(Vertex u,
16     LinkedList<Integer> sorting) {
17     u.vis = true;
18     for(Vertex v : u.adj) {
19         if(v.vis)
20             //the -1 indicates that it will not be
21             //possible to find an TS
22             //there might be a much faster and elegant way
23             //(flag?!)
24             sorting.addFirst(-1);
25     }
26     else
27         recTS(v, sorting);
28     sorting.addFirst(u.id);
29     return sorting;
30 }

```

MD5: b4fb592469cf03dcb788aba03b98263e | $\mathcal{O}(|V| + |E|)$

2.17 Reference for Vertex classes

Used in many graph algorithms, implements a vertex with its edges. Needs testing.

```

1 class Vertex {
2
3     int id;
4     boolean vis = false;
5     int pre = -1;
6
7     //for dijkstra and prim
8     int dist = Integer.MAX_VALUE;
9
10    //for SCC store number indicating the dedicated
11    //component
12    int comp = -1;
13
14    //for DFS we could store the start and finishing
15    //times
16    int dtime = -1;
17    int ftime = -1;
18
19    //use an ArrayList of Edges if those information
20    //are needed
21    ArrayList<Edge> adj = new ArrayList<Edge>();
22    //use an ArrayList of Vertices else
23    ArrayList<Vertex> adj = new ArrayList<Vertex>();
24    //use two ArrayLists for SCC
25    ArrayList<Vertex> in = new ArrayList<Vertex>();
26    ArrayList<Vertex> out = new ArrayList<Vertex>();
27
28    //for EdmondsKarp we need a HashMap to store Edges
29    HashMap<Integer, Edge> adj = new HashMap<Integer,
30        Edge>();
31
32    //for bipartite graph check
33    int color = -1;
34
35    //we store as key the target
36    public Vertex(int id) {
37        this.id = id;
38    }
39 }

```

MD5: f41108043e72983fc088f5851de6b932 | $\mathcal{O}(?)$

3 Math

3.1 Binomial Coefficient

Gives binomial coefficient (n choose k)

```

1 public static long bin(int n, int k) {
2     if (k == 0) {
3         return 1;
4     } else if (k > n/2) {
5         return bin(n, n-k);
6     } else {
7         return n*bin(n-1, k-1)/k;
8     }
9 }

```

MD5: ceca2cc881a9da6269c143a41f89cc12 | $\mathcal{O}(k)$

3.2 Binomial Matrix

Gives binomial coefficients for all $K \leq N$.


```

1 public static long[][] binomial_matrix(int N, int K) {
2     long[][] B = new long[N+1][K+1];
3     for (int k = 1; k <= K; k++) {
4         B[0][k] = 0;
5     }
6     for (int m = 0; m <= N; m++) {
7         B[m][0] = 1;
8     }
9     for (int m = 1; m <= N; m++) {
10        for (int k = 1; k <= K; k++) {
11            B[m][k] = B[m-1][k-1] + B[m-1][k];
12        }
13    }
14    return B;
15 }

```

MD5: 0754f4e27d08a1d1f5e6c0cf4ef636df | $\mathcal{O}(N \cdot K)$

3.3 Divisability

Calculates (alternating) k-digitSum for integer number given by M.

```

1 public static long digit_sum(String M, int k, boolean
2     alt) {
3     long dig_sum = 0;
4     int vz = 1;
5     while (M.length() > k) {
6         if (alt) vz *= -1;
7         dig_sum += vz * Integer.parseInt(M.substring(M.
8             length()-k));
9         M = M.substring(0, M.length()-k);
10    }
11    if (alt) vz *= -1;
12    dig_sum += vz * Integer.parseInt(M);
13    return dig_sum;
14 }
15 // example: divisibility of M by 13
16 public static boolean divisible13(String M) {
17     return digit_sum(M, 3, true)%13 == 0;
18 }

```

MD5: 33b3094ebf431e1e71cd8e8db3c9cd6 | $\mathcal{O}(?)$

3.4 Iterative EEA

Berechnet den ggT zweier Zahlen a und b und deren modulare Inverse $x = a^{-1} \bmod b$ und $y = b^{-1} \bmod a$.

```

1 // Extended Euclidean Algorithm - iterativ
2 public static long[] eea(long a, long b) {
3     if (b > a) {
4         long tmp = a;
5         a = b;
6         b = tmp;
7     }
8     long x = 0, y = 1, u = 1, v = 0;
9     while (a != 0) {
10        long q = b / a, r = b % a;
11        long m = x - u * q, n = y - v * q;
12        b = a; a = r; x = u; y = v; u = m; v = n;
13    }
14    long gcd = b;
15    // x = a^-1 % b, y = b^-1 % a
16    // ax + by = gcd

```

```

long[] erg = { gcd, x, y };
return erg;
}

```

MD5: 81fe8cd4adab21329dcbe1ce0499ee75 | $\mathcal{O}(\log a + \log b)$

3.5 Polynomial Interpolation

```

1 public class interpol {
2
3     // divided differences for points given by vectors x
4     // and y
5     public static rat[] divDiff(rat[] x, rat[] y) {
6         rat[] temp = y.clone();
7         int n = x.length;
8         rat[] res = new rat[n];
9         res[0] = temp[0];
10        for (int i=1; i < n; i++) {
11            for (int j = 0; j < n-i; j++) {
12                temp[j] = (temp[j+1].sub(temp[j])).div(x[j+i].
13                    sub(x[j]));
14            }
15            res[i] = temp[0];
16        }
17        return res;
18    }
19
20    // evaluates interpolating polynomial p at t for
21    // given
22    // x-coordinates and divided differences
23    public static rat p(rat t, rat[] x, rat[] dD) {
24        int n = x.length;
25        rat p = new rat(0);
26        for (int i = n-1; i > 0; i--) {
27            p = (p.add(dD[i])).mult(t.sub(x[i-1]));
28        }
29        p = p.add(dD[0]);
30        return p;
31    }
32
33    public static void main(String[] args) {
34
35        rat[] test = {new rat(4,5), new rat(7,10), new rat
36            (3,4)};
37        test = rat.commonDenominator(test);
38        for (int i = 0; i < test.length; i++) {
39            System.out.println(test[i].toString());
40        }
41
42        rat[] x = {new rat(0),new rat(1), new rat(2), new
43            rat(3), new rat(4), new rat(5)};
44        rat[] y = {new rat(-10), new rat(9), new rat(0),
45            new rat(1), new rat(1,2), new rat(1,80)};
46        rat[] dD = divDiff(x,y);
47        System.out.println("p("+7+")="+p(new rat(7), x,
48            dD));
49    }
50 }
51
52 // implementation of rational numbers
53 class rat {
54
55     public long c;
56     public long d;
57
58     public rat (long c, long d) {
59         this.c = c;

```

```

53     this.d = d;
54     this.shorten();
55 }
56
57 public rat (long c) {
58     this.c = c;
59     this.d = 1;
60 }
61
62 public static long ggT(long a, long b) {
63     while (b != 0) {
64         long h = a%b;
65         a = b;
66         b = h;
67     }
68     return a;
69 }
70
71 public static long kgV(long a, long b) {
72     return a*b/ggT(a,b);
73 }
74
75 public static rat[] commonDenominator(rat[] c) {
76     long kgV = 1;
77     for (int i = 0; i < c.length; i++) {
78         kgV = kgV(kgV, c[i].d);
79     }
80     for (int i = 0; i < c.length; i++) {
81         c[i].c *= kgV/c[i].d;
82         c[i].d *= kgV/c[i].d;
83     }
84     return c;
85 }
86
87 public void shorten() {
88     long ggT = ggT(this.c, this.d);
89     this.c = this.c / ggT;
90     this.d = this.d / ggT;
91     if (d < 0) {
92         this.d *= -1;
93         this.c *= -1;
94     }
95 }
96
97 public String toString() {
98     if (this.d == 1) return ""+c;
99     return ""+c+"/"+d;
100 }
101
102 public rat mult(rat b) {
103     return new rat(this.c*b.c, this.d*b.d);
104 }
105
106 public rat div(rat b) {
107     return new rat(this.c*b.d, this.d*b.c);
108 }
109
110 public rat add(rat b) {
111     long new_d = kgV(this.d, b.d);
112     long new_c = this.c*(new_d/this.d) + b.c*(new_d/b.d);
113     return new rat(new_c, new_d);
114 }
115
116 public rat sub(rat b) {
117     return this.add(new rat(-b.c, b.d));
118 }
119

```

}

MD5: d98bd247b95395d8596ff1d5785ee06b | $\mathcal{O}(?)$

3.6 Sieve of Eratosthenes

Calculates Sieve of Eratosthenes.

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

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

```

1 public static boolean[] sieveOfEratosthenes(int N) {
2     boolean[] isPrime = new boolean[N+1];
3     for (int i=2; i<=N; i++) isPrime[i] = true;
4     for (int i = 2; i*i <= N; i++)
5         if (isPrime[i])
6             for (int j = i*i; j <= N; j+=i)
7                 isPrime[j] = false;
8     return isPrime;
9 }

```

MD5: 95704ae7c1fe03e91adeb8d695b2f5bb | $\mathcal{O}(n)$

4 Misc

4.1 Binary Search

Binary searches for an element in a sorted array.

Input: sorted *array* to search in, amount N of elements in *array*, element to search for a

Output: returns the index of a in *array* or -1 if *array* does not contain a

```

1 public static int BinarySearch(int[] array,
2                               int N, int a) {
3     int lo = 0;
4     int hi = N-1;
5     // a might be in interval [lo,hi] while lo <= hi
6     while(lo <= hi) {
7         int mid = (lo + hi) / 2;
8         // if a > elem in mid of interval,
9         // search the right subinterval
10        if(array[mid] < a)
11            lo = mid+1;
12        // else if a < elem in mid of interval,
13        // search the left subinterval
14        else if(array[mid] > a)
15            hi = mid-1;
16        // else a is found
17        else
18            return mid;
19    }
20    // array does not contain a
21    return -1;
22 }

```

MD5: 203da61f7a381564ce3515f674fa82a4 | $\mathcal{O}(\log n)$

4.2 Next number with n bits set

From x the smallest number greater than x with the same amount of bits set is computed. Little changes have to be made, if the calculated number has to have length less than 32 bits.

Input: number x with n bits set ($x = (1 \ll n) - 1$)

Output: the smallest number greater than x with n bits set

```

1 public static int nextNumber(int x) {
2     //break when larger than limit here
3     if(x == 0) return 0;
4     int smallest = x & -x;
5     int ripple = x + smallest;
6     int new_smallest = ripple & -ripple;
7     int ones = ((new_smallest/smallest) >> 1) - 1;
8     return ripple | ones;
9 }

```

MD5: 2d8a79cb551648e67fc3f2f611a4f63c | $\mathcal{O}(1)$

5 Math Roland

5.1 Divisability Explanation

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

5.2 Combinatorics

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

5.3 Polynomial Interpolation

5.3.1 Theory

Problem: for $\{(x_0, y_0), \dots, (x_n, y_n)\}$ find $p \in \Pi_n$ with $p(x_i) = y_i$ for all $i = 0, \dots, n$.

Solution: $p(x) = \sum_{i=0}^n \gamma_{0,i} \prod_{j=0}^{i-1} (x - x_j)$ where $\gamma_{j,k} = y_j$ for $k = 0$ and $\gamma_{j,k} = \frac{\gamma_{j+1,k-1} - \gamma_{j,k-1}}{x_{j+k} - x_j}$ otherwise.

Efficient evaluation of $p(x)$: $b_n = \gamma_{0,n}$, $b_i = b_{i+1}(x - x_i) + \gamma_{0,i}$ for $i = n-1, \dots, 0$ with $b_0 = p(x)$.

5.4 Fibonacci Sequence

5.4.1 Binet's formula

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

5.4.2 Generalization

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

5.4.3 Pisano Period

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

6 Java Knowhow

6.1 System.out.printf() und String.format()

Syntax: %[flags][width][.precision][conv]

flags:

- left-justify (default: right)
- + always output number sign
- 0 zero-pad numbers
- (space) space instead of minus for pos. numbers
- , group triplets of digits with ,

width specifies output width

precision is for floating point precision

conv:

- d byte, short, int, long
- f float, double
- c char (use C for uppercase)
- s String (use S for all uppercase)

6.2 Modulo: Avoiding negative Integers

```
1 int mod = (((nums[j] % D) + D) % D);
```

6.3 Speed up IO

Use

```
1 BufferedReader br = new BufferedReader(new
2 InputStreamReader(System.in));
```

Use

```
Double.parseDouble(Scanner.next());
```

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 $b \geq s, \forall s \in S$.	$\sum_{i=0}^{\infty} c^i = \frac{c^{n+1} - 1}{c - 1}, \quad c \neq 1, \quad \sum_{i=0}^{\infty} c^i = \frac{1}{1-c}, \quad \sum_{i=1}^{\infty} c^i = \frac{c}{1-c}, \quad c < 1,$
$\inf S$	greatest $b \in \mathbb{R}$ such that $b \leq s, \forall s \in S$.	$\sum_{i=0}^{\infty} ic^i = \frac{nc^{n+2} - (n+1)c^{n+1} + c}{(c-1)^2}, \quad c \neq 1, \quad \sum_{i=0}^{\infty} ic^i = \frac{c}{(1-c)^2}, \quad c < 1.$
$\liminf_{n \rightarrow \infty} a_n$	$\lim_{n \rightarrow \infty} \inf \{a_i \mid i \geq n, i \in \mathbb{N}\}.$	Harmonic series:
$\limsup_{n \rightarrow \infty} a_n$	$\lim_{n \rightarrow \infty} \sup \{a_i \mid i \geq n, i \in \mathbb{N}\}.$	$H_n = \sum_{i=1}^n \frac{1}{i}, \quad \sum_{i=1}^n iH_i = \frac{n(n+1)}{2}H_n - \frac{n(n-1)}{4}.$
$\binom{n}{k}$	Combinations: Size k sub-sets of a size n set.	$\sum_{i=1}^n H_i = (n+1)H_n - n, \quad \sum_{i=1}^n \binom{i}{m} H_i = \binom{n+1}{m+1} \left(H_{n+1} - \frac{1}{m+1} \right).$
$[n]_k$	Stirling numbers (1st kind): Arrangements of an n element set into k cycles.	1. $\binom{n}{k} = \frac{n!}{(n-k)!k!}, \quad 2. \sum_{k=0}^n \binom{n}{k} = 2^n, \quad 3. \binom{n}{k} = \binom{n}{n-k},$
$\{n\}_k$	Stirling numbers (2nd kind): Partitions of an n element set into k non-empty sets.	4. $\binom{n}{k} = \frac{n}{k} \binom{n-1}{k-1}, \quad 5. \binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1},$
$\langle n \rangle_k$	1st order Eulerian numbers: Permutations $\pi_1 \pi_2 \dots \pi_n$ on $\{1, 2, \dots, n\}$ with k ascents.	6. $\binom{n}{m} \binom{m}{k} = \binom{n}{k} \binom{n-k}{m-k}, \quad 7. \sum_{k=0}^n \binom{r+k}{k} = \binom{r+n+1}{n},$
$\langle\langle n \rangle\rangle_k$	2nd order Eulerian numbers.	8. $\sum_{k=0}^n \binom{k}{m} = \binom{n+1}{m+1}, \quad 9. \sum_{k=0}^n \binom{r}{k} \binom{s}{n-k} = \binom{r+s}{n},$
C_n	Catalan Numbers: Binary trees with $n+1$ vertices.	10. $\binom{n}{k} = (-1)^k \binom{k-n-1}{k}, \quad 11. \left\{ \begin{matrix} n \\ 1 \end{matrix} \right\} = \left\{ \begin{matrix} n \\ n \end{matrix} \right\} = 1,$
14. $\left[\begin{matrix} n \\ 1 \end{matrix} \right] = (n-1)!,$	15. $\left[\begin{matrix} n \\ 2 \end{matrix} \right] = (n-1)!H_{n-1},$	12. $\left\{ \begin{matrix} n \\ 2 \end{matrix} \right\} = 2^{n-1} - 1, \quad 13. \left\{ \begin{matrix} n \\ k \end{matrix} \right\} = k \left\{ \begin{matrix} n-1 \\ k \end{matrix} \right\} + \left\{ \begin{matrix} n-1 \\ k-1 \end{matrix} \right\},$
16. $\left[\begin{matrix} n \\ n \end{matrix} \right] = 1,$	17. $\left[\begin{matrix} n \\ k \end{matrix} \right] \geq \left\{ \begin{matrix} n \\ k \end{matrix} \right\},$	
18. $\left[\begin{matrix} n \\ k \end{matrix} \right] = (n-1) \left[\begin{matrix} n-1 \\ k \end{matrix} \right] + \left[\begin{matrix} n-1 \\ k-1 \end{matrix} \right],$	19. $\left\{ \begin{matrix} n \\ n-1 \end{matrix} \right\} = \left[\begin{matrix} n \\ n-1 \end{matrix} \right] = \binom{n}{2},$	20. $\sum_{k=0}^n \left[\begin{matrix} n \\ k \end{matrix} \right] = n!, \quad 21. C_n = \frac{1}{n+1} \binom{2n}{n},$
22. $\left\langle \begin{matrix} n \\ 0 \end{matrix} \right\rangle = \left\langle \begin{matrix} n \\ n-1 \end{matrix} \right\rangle = 1,$	23. $\left\langle \begin{matrix} n \\ k \end{matrix} \right\rangle = \left\langle \begin{matrix} n \\ n-1-k \end{matrix} \right\rangle,$	24. $\left\langle \begin{matrix} n \\ k \end{matrix} \right\rangle = (k+1) \left\langle \begin{matrix} n-1 \\ k \end{matrix} \right\rangle + (n-k) \left\langle \begin{matrix} n-1 \\ k-1 \end{matrix} \right\rangle,$
25. $\left\langle \begin{matrix} 0 \\ k \end{matrix} \right\rangle = \begin{cases} 1 & \text{if } k=0, \\ 0 & \text{otherwise} \end{cases}$	26. $\left\langle \begin{matrix} n \\ 1 \end{matrix} \right\rangle = 2^n - n - 1,$	27. $\left\langle \begin{matrix} n \\ 2 \end{matrix} \right\rangle = 3^n - (n+1)2^n + \binom{n+1}{2},$
28. $x^n = \sum_{k=0}^n \left\langle \begin{matrix} n \\ k \end{matrix} \right\rangle \binom{x+k}{n},$	29. $\left\langle \begin{matrix} n \\ m \end{matrix} \right\rangle = \sum_{k=0}^m \binom{n+1}{k} (m+1-k)^n (-1)^k,$	30. $m! \left\{ \begin{matrix} n \\ m \end{matrix} \right\} = \sum_{k=0}^n \left\langle \begin{matrix} n \\ k \end{matrix} \right\rangle \binom{k}{n-m},$
31. $\left\langle \begin{matrix} n \\ m \end{matrix} \right\rangle = \sum_{k=0}^n \left\{ \begin{matrix} n \\ k \end{matrix} \right\} \binom{n-k}{m} (-1)^{n-k-m} k!,$	32. $\langle\langle \begin{matrix} n \\ 0 \end{matrix} \rangle\rangle = 1,$	33. $\langle\langle \begin{matrix} n \\ n \end{matrix} \rangle\rangle = 0 \quad \text{for } n \neq 0,$
34. $\langle\langle \begin{matrix} n \\ k \end{matrix} \rangle\rangle = (k+1) \langle\langle \begin{matrix} n-1 \\ k \end{matrix} \rangle\rangle + (2n-1-k) \langle\langle \begin{matrix} n-1 \\ k-1 \end{matrix} \rangle\rangle,$	35. $\sum_{k=0}^n \langle\langle \begin{matrix} n \\ k \end{matrix} \rangle\rangle = \frac{(2n)^n}{2^n},$	
36. $\left\{ \begin{matrix} x \\ x-n \end{matrix} \right\} = \sum_{k=0}^n \langle\langle \begin{matrix} n \\ k \end{matrix} \rangle\rangle \binom{x+n-1-k}{2n},$	37. $\left\{ \begin{matrix} n+1 \\ m+1 \end{matrix} \right\} = \sum_k \binom{n}{k} \left\{ \begin{matrix} k \\ m \end{matrix} \right\} = \sum_{k=0}^n \left\{ \begin{matrix} k \\ m \end{matrix} \right\} (m+1)^{n-k},$	

Theoretical Computer Science Cheat Sheet

Identities Cont.

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

Trees

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

Kraft inequality: If the depths of the leaves of a binary tree are d_1, \dots, d_n :

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

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

Recurrences

Master method:

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

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

$$T(n) = \Theta(n^{\log_b a}).$$

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

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

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

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

Substitution (example): Consider the following recurrence

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

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

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

which is simply $u_i = i/2$. So we find that T_i has the closed form $T_i = 2^{i2^{i-1}}$.

Summing factors (example): Consider the following recurrence

$$T(n) = 3T(n/2) + n, \quad T(1) = 1.$$

Rewrite so that all terms involving T are on the left side

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

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

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

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

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

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

Let $m = \log_2 n$. Summing the left side we get $T(n) - 3^m T(1) = T(n) - 3^m = T(n) - n^k$ where $k = \log_2 3 \approx 1.58496$.

Summing the right side we get

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

Let $c = \frac{3}{2}$. Then we have

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

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

$$= 2n(c^{(k-1)\log_2 n} - 1)$$

$$= 2n^k - 2n,$$

and so $T(n) = 3n^k - 2n$. Full history recurrences can often be changed to limited history ones (example): Consider

$$T_i = 1 + \sum_{j=0}^{i-1} T_j, \quad T_0 = 1.$$

Note that

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

Subtracting we find

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

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

Generating functions:

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

Example:

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

Multiply and sum:

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

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

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

Simplify:

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

Solve for $G(x)$:

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

Expand this using partial fractions:

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

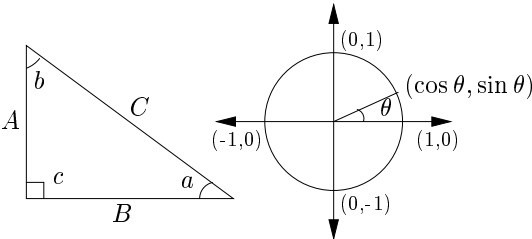
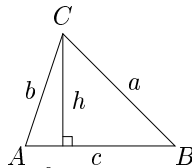
So $g_i = 2^i - 1$.

Theoretical Computer Science Cheat Sheet

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

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
3	8	5	Change of base, quadratic formula: $\log_b x = \frac{\log_a x}{\log_a b}, \quad \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$	$\Pr[X < a] = P(a),$
4	16	7	Euler's number e :	then P is the distribution function of X . If P and p both exist then
5	32	11	$e = 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + \frac{1}{120} + \dots$	$P(a) = \int_{-\infty}^a p(x) dx.$
6	64	13	$\lim_{n \rightarrow \infty} \left(1 + \frac{x}{n}\right)^n = e^x.$	Expectation: If X is discrete
7	128	17	$\left(1 + \frac{1}{n}\right)^n < e < \left(1 + \frac{1}{n}\right)^{n+1}.$	$E[g(X)] = \sum_x g(x) \Pr[X = x].$
8	256	19	$\left(1 + \frac{1}{n}\right)^n = e - \frac{e}{2n} + \frac{11e}{24n^2} - O\left(\frac{1}{n^3}\right).$	If X continuous then
9	512	23	Harmonic numbers:	$E[g(X)] = \int_{-\infty}^{\infty} g(x)p(x) dx = \int_{-\infty}^{\infty} g(x) dP(x).$
10	1,024	29	$1, \frac{3}{2}, \frac{11}{6}, \frac{25}{12}, \frac{137}{60}, \frac{49}{20}, \frac{363}{140}, \frac{761}{280}, \frac{7129}{2520}, \dots$	Variance, standard deviation:
11	2,048	31	$\ln n < H_n < \ln n + 1,$	$\text{VAR}[X] = E[X^2] - E[X]^2,$
12	4,096	37	$H_n = \ln n + \gamma + O\left(\frac{1}{n}\right).$	$\sigma = \sqrt{\text{VAR}[X]}.$
13	8,192	41	Factorial, Stirling's approximation:	For events A and B :
14	16,384	43	$1, 2, 6, 24, 120, 720, 5040, 40320, 362880, \dots$	$\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$
15	32,768	47	$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right).$	$\Pr[A \wedge B] = \Pr[A] \cdot \Pr[B],$
16	65,536	53	Ackermann's function and inverse:	iff A and B are independent.
17	131,072	59	$a(i, j) = \begin{cases} 2^j & i = 1 \\ a(i-1, 2) & j = 1 \\ a(i-1, a(i, j-1)) & i, j \geq 2 \end{cases}$	$\Pr[A B] = \frac{\Pr[A \wedge B]}{\Pr[B]}$
18	262,144	61	$\alpha(i) = \min\{j \mid a(j, j) \geq i\}.$	For random variables X and Y :
19	524,288	67	Binomial distribution:	$E[X \cdot Y] = E[X] \cdot E[Y],$
20	1,048,576	71	$\Pr[X = k] = \binom{n}{k} p^k q^{n-k}, \quad q = 1 - p,$	if X and Y are independent.
21	2,097,152	73	$E[X] = \sum_{k=1}^n k \binom{n}{k} p^k q^{n-k} = np.$	$E[X + Y] = E[X] + E[Y],$
22	4,194,304	79	Poisson distribution:	$E[cX] = c E[X].$
23	8,388,608	83	$\Pr[X = k] = \frac{e^{-\lambda} \lambda^k}{k!}, \quad E[X] = \lambda.$	Bayes' theorem:
24	16,777,216	89	Normal (Gaussian) distribution:	$\Pr[A_i B] = \frac{\Pr[B A_i] \Pr[A_i]}{\sum_{j=1}^n \Pr[A_j] \Pr[B A_j]}.$
25	33,554,432	97	$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}, \quad E[X] = \mu.$	Inclusion-exclusion:
26	67,108,864	101	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	$\Pr\left[\bigvee_{i=1}^n X_i\right] = \sum_{i=1}^n \Pr[X_i] +$
27	134,217,728	103	$nH_n.$	$\sum_{k=2}^n (-1)^{k+1} \sum_{i_1 < \dots < i_k} \Pr\left[\bigwedge_{j=1}^k X_{i_j}\right].$
28	268,435,456	107		Moment inequalities:
29	536,870,912	109		$\Pr[X \geq \lambda E[X]] \leq \frac{1}{\lambda},$
30	1,073,741,824	113		$\Pr[X - E[X] \geq \lambda \cdot \sigma] \leq \frac{1}{\lambda^2}.$
31	2,147,483,648	127		Geometric distribution:
32	4,294,967,296	131		$\Pr[X = k] = pq^{k-1}, \quad q = 1 - p,$
Pascal's Triangle				$E[X] = \sum_{k=1}^{\infty} kpq^{k-1} = \frac{1}{p}.$
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:</p> $c^2 = a^2 + b^2 - 2ab \cos C.$ <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> $\sin \frac{x}{2} = \sqrt{\frac{1 - \cos x}{2}},$ $\cos \frac{x}{2} = \sqrt{\frac{1 + \cos x}{2}},$ $\tan \frac{x}{2} = \sqrt{\frac{1 - \cos x}{1 + \cos x}},$ $= \frac{1 - \cos x}{\sin x},$ $= \frac{\sin x}{1 + \cos x},$ $\cot \frac{x}{2} = \sqrt{\frac{1 + \cos x}{1 - \cos x}},$ $= \frac{1 + \cos x}{\sin x},$ $= \frac{\sin x}{1 - \cos x},$ $\sin x = \frac{e^{ix} - e^{-ix}}{2i},$ $\cos x = \frac{e^{ix} + e^{-ix}}{2},$ $\tan x = -i \frac{e^{ix} - e^{-ix}}{e^{ix} + e^{-ix}},$ $= -i \frac{e^{2ix} - 1}{e^{2ix} + 1},$ $\sin x = \frac{\sinh ix}{i},$ $\cos x = \cosh ix,$ $\tan x = \frac{\tanh ix}{i}.$																								
	<p>Hyperbolic Functions</p> <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> <p>... in mathematics you don't under- stand things, you just get used to them. - J. von Neumann</p>	θ	$\sin \theta$	$\cos \theta$	$\tan \theta$	0	0	1	0	$\frac{\pi}{6}$	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{3}}{3}$	$\frac{\pi}{4}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$	1	$\frac{\pi}{3}$	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$	$\sqrt{3}$	$\frac{\pi}{2}$	1	0	∞	
θ	$\sin \theta$	$\cos \theta$	$\tan \theta$																							
0	0	1	0																							
$\frac{\pi}{6}$	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{3}}{3}$																							
$\frac{\pi}{4}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$	1																							
$\frac{\pi}{3}$	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$	$\sqrt{3}$																							
$\frac{\pi}{2}$	1	0	∞																							
v2.01 ©1994 by Steve Seiden sseiden@acm.org http://www.csc.lsu.edu/~seiden																										

Theoretical Computer Science Cheat Sheet

Number Theory

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

$$C \equiv r_1 \pmod{m_1}$$

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

$$C \equiv r_n \pmod{m_n}$$

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

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

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

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

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

Fermat's theorem:

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

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

$$(n-1)! \equiv -1 \pmod{n}.$$

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:

$$\begin{aligned} p_n &= n \ln n + n \ln \ln n - n + n \frac{\ln \ln n}{\ln n} \\ &\quad + O\left(\frac{n}{\ln n}\right), \\ \pi(n) &= \frac{n}{\ln n} + \frac{n}{(\ln n)^2} + \frac{2!n}{(\ln n)^3} \\ &\quad + O\left(\frac{n}{(\ln n)^4}\right). \end{aligned}$$

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 $v_0 e_1 v_1 \dots e_\ell v_\ell$.

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 $k \cdot c(G-S) \leq |S|$.

k-Regular A graph where all vertices have degree k .

k-Factor A k -regular spanning subgraph.

Matching A set of edges, no two of which are adjacent.

Clique A set of vertices, all of which are adjacent.

Ind. set A set of vertices, none of which are adjacent.

Vertex cover A set of vertices which cover all edges.

Planar graph A graph which can be embedded in the plane.

Plane graph An embedding of a planar graph.

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

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

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

Any planar graph has a vertex with degree ≤ 5 .

Notation:

$E(G)$ Edge set

$V(G)$ Vertex set

$c(G)$ Number of components

$G[S]$ Induced subgraph

$\deg(v)$ Degree of v

$\Delta(G)$ Maximum degree

$\delta(G)$ Minimum degree

$\chi(G)$ Chromatic number

$\chi_E(G)$ Edge chromatic number

G^c Complement graph

K_n Complete graph

K_{n_1, n_2} Complete bipartite graph

$r(k, \ell)$ 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

$$(x, y) \quad (x, y, 1)$$

$$y = mx + b \quad (m, -1, b)$$

$$x = c \quad (1, 0, -c)$$

Distance formula, L_p and L_∞ metric:

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

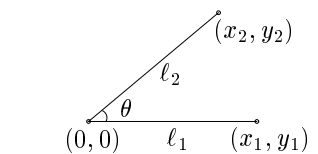
$$[|x_1 - x_0|^p + |y_1 - y_0|^p]^{1/p},$$

$$\lim_{p \rightarrow \infty} [|x_1 - x_0|^p + |y_1 - y_0|^p]^{1/p}.$$

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

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

Angle formed by three points:



$$\cos \theta = \frac{(x_1, y_1) \cdot (x_2, y_2)}{l_1 l_2}.$$

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:

$$A = \pi r^2, \quad V = \frac{4}{3} \pi r^3.$$

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:

$$\frac{\pi}{4} = 1 + \frac{1^2}{2 + \frac{3^2}{2 + \frac{5^2}{2 + \frac{7^2}{2 + \cdots}}}}$$

Gregory's series:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots$$

Newton's series:

$$\frac{\pi}{6} = \frac{1}{2} + \frac{1}{2 \cdot 3 \cdot 2^3} + \frac{1 \cdot 3}{2 \cdot 4 \cdot 5 \cdot 2^5} + \cdots$$

Sharp's series:

$$\frac{\pi}{6} = \frac{1}{\sqrt{3}} \left(1 - \frac{1}{3^1 \cdot 3} + \frac{1}{3^2 \cdot 5} - \frac{1}{3^3 \cdot 7} + \cdots \right)$$

Euler's series:

$$\frac{\pi^2}{6} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \cdots$$

$$\frac{\pi^2}{8} = \frac{1}{1^2} + \frac{1}{3^2} + \frac{1}{5^2} + \frac{1}{7^2} + \frac{1}{9^2} + \cdots$$

$$\frac{\pi^2}{12} = \frac{1}{1^2} - \frac{1}{2^2} + \frac{1}{3^2} - \frac{1}{4^2} + \frac{1}{5^2} - \cdots$$

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:

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

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

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

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

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

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

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

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

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

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

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

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

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

$$29. \frac{d(\operatorname{arctanh} u)}{dx} = \frac{1}{1-u^2} \frac{du}{dx}, \quad 30. \frac{d(\operatorname{arcoth} u)}{dx} = \frac{1}{u^2-1} \frac{du}{dx},$$

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

Integrals:

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

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

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

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

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

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

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

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. $\int \sin^2(ax) dx = \frac{1}{2a}(ax - \sin(ax) \cos(ax)),$
18. $\int \cos^2(ax) dx = \frac{1}{2a}(ax + \sin(ax) \cos(ax)),$
19. $\int \sec^2 x dx = \tan x,$
20. $\int \csc^2 x dx = -\cot x,$
21. $\int \sin^n x dx = -\frac{\sin^{n-1} x \cos x}{n} + \frac{n-1}{n} \int \sin^{n-2} x dx,$
22. $\int \cos^n x dx = \frac{\cos^{n-1} x \sin x}{n} + \frac{n-1}{n} \int \cos^{n-2} x dx,$
23. $\int \tan^n x dx = \frac{\tan^{n-1} x}{n-1} - \int \tan^{n-2} x dx, \quad n \neq 1,$
24. $\int \cot^n x dx = -\frac{\cot^{n-1} x}{n-1} - \int \cot^{n-2} x dx, \quad n \neq 1,$
25. $\int \sec^n x dx = \frac{\tan x \sec^{n-1} x}{n-1} + \frac{n-2}{n-1} \int \sec^{n-2} x dx, \quad n \neq 1,$
26. $\int \csc^n x dx = -\frac{\cot x \csc^{n-1} x}{n-1} + \frac{n-2}{n-1} \int \csc^{n-2} x dx, \quad n \neq 1,$
27. $\int \sinh x dx = \cosh x,$
28. $\int \cosh x dx = \sinh x,$
29. $\int \tanh x dx = \ln |\cosh x|,$
30. $\int \coth x dx = \ln |\sinh x|,$
31. $\int \operatorname{sech} x dx = \arctan \sinh x,$
32. $\int \operatorname{csch} x dx = \ln \left| \tanh \frac{x}{2} \right|,$
33. $\int \sinh^2 x dx = \frac{1}{4} \sinh(2x) - \frac{1}{2} x,$
34. $\int \cosh^2 x dx = \frac{1}{4} \sinh(2x) + \frac{1}{2} x,$
35. $\int \operatorname{sech}^2 x dx = \tanh x,$
36. $\int \operatorname{arcsinh} \frac{x}{a} dx = x \operatorname{arcsinh} \frac{x}{a} - \sqrt{x^2 + a^2}, \quad a > 0,$
37. $\int \operatorname{arctanh} \frac{x}{a} dx = x \operatorname{arctanh} \frac{x}{a} + \frac{a}{2} \ln |a^2 - x^2|,$
38. $\int \operatorname{arccosh} \frac{x}{a} dx = \begin{cases} x \operatorname{arccosh} \frac{x}{a} - \sqrt{x^2 + a^2}, & \text{if } \operatorname{arccosh} \frac{x}{a} > 0 \text{ and } a > 0, \\ x \operatorname{arccosh} \frac{x}{a} + \sqrt{x^2 + a^2}, & \text{if } \operatorname{arccosh} \frac{x}{a} < 0 \text{ and } a > 0, \end{cases}$
39. $\int \frac{dx}{\sqrt{a^2 + x^2}} = \ln \left(x + \sqrt{a^2 + x^2} \right), \quad a > 0,$
40. $\int \frac{dx}{a^2 + x^2} = \frac{1}{a} \arctan \frac{x}{a}, \quad a > 0,$
41. $\int \sqrt{a^2 - x^2} dx = \frac{x}{2} \sqrt{a^2 - x^2} + \frac{a^2}{2} \arcsin \frac{x}{a}, \quad a > 0,$
42. $\int (a^2 - x^2)^{3/2} dx = \frac{x}{8} (5a^2 - 2x^2) \sqrt{a^2 - x^2} + \frac{3a^4}{8} \arcsin \frac{x}{a}, \quad a > 0,$
43. $\int \frac{dx}{\sqrt{a^2 - x^2}} = \arcsin \frac{x}{a}, \quad a > 0,$
44. $\int \frac{dx}{a^2 - x^2} = \frac{1}{2a} \ln \left| \frac{a+x}{a-x} \right|,$
45. $\int \frac{dx}{(a^2 - x^2)^{3/2}} = \frac{x}{a^2 \sqrt{a^2 - x^2}},$
46. $\int \sqrt{a^2 \pm x^2} dx = \frac{x}{2} \sqrt{a^2 \pm x^2} \pm \frac{a^2}{2} \ln \left| x + \sqrt{a^2 \pm x^2} \right|,$
47. $\int \frac{dx}{\sqrt{x^2 - a^2}} = \ln \left| x + \sqrt{x^2 - a^2} \right|, \quad a > 0,$
48. $\int \frac{dx}{ax^2 + bx} = \frac{1}{a} \ln \left| \frac{x}{a+bx} \right|,$
49. $\int x \sqrt{a+bx} dx = \frac{2(3bx-2a)(a+bx)^{3/2}}{15b^2},$
50. $\int \frac{\sqrt{a+bx}}{x} dx = 2\sqrt{a+bx} + a \int \frac{1}{x\sqrt{a+bx}} dx,$
51. $\int \frac{x}{\sqrt{a+bx}} dx = \frac{1}{\sqrt{2}} \ln \left| \frac{\sqrt{a+bx} - \sqrt{a}}{\sqrt{a+bx} + \sqrt{a}} \right|, \quad a > 0,$
52. $\int \frac{\sqrt{a^2 - x^2}}{x} dx = \sqrt{a^2 - x^2} - a \ln \left| \frac{a + \sqrt{a^2 - x^2}}{x} \right|,$
53. $\int x \sqrt{a^2 - x^2} dx = -\frac{1}{3} (a^2 - x^2)^{3/2},$
54. $\int x^2 \sqrt{a^2 - x^2} dx = \frac{x}{8} (2x^2 - a^2) \sqrt{a^2 - x^2} + \frac{a^4}{8} \arcsin \frac{x}{a}, \quad a > 0,$
55. $\int \frac{dx}{\sqrt{a^2 - x^2}} = -\frac{1}{a} \ln \left| \frac{a + \sqrt{a^2 - x^2}}{x} \right|,$
56. $\int \frac{x dx}{\sqrt{a^2 - x^2}} = -\sqrt{a^2 - x^2},$
57. $\int \frac{x^2 dx}{\sqrt{a^2 - x^2}} = -\frac{x}{2} \sqrt{a^2 - x^2} + \frac{a^2}{2} \arcsin \frac{x}{a}, \quad a > 0,$
58. $\int \frac{\sqrt{a^2 + x^2}}{x} dx = \sqrt{a^2 + x^2} - a \ln \left| \frac{a + \sqrt{a^2 + x^2}}{x} \right|,$
59. $\int \frac{\sqrt{x^2 - a^2}}{x} dx = \sqrt{x^2 - a^2} - a \arccos \frac{a}{|x|}, \quad a > 0,$
60. $\int x \sqrt{x^2 \pm a^2} dx = \frac{1}{3} (x^2 \pm a^2)^{3/2},$
61. $\int \frac{dx}{x \sqrt{x^2 + a^2}} = \frac{1}{a} \ln \left| \frac{x}{a + \sqrt{a^2 + x^2}} \right|,$

Theoretical Computer Science Cheat Sheet

Calculus Cont.

$$\begin{aligned}
62. \int \frac{dx}{x\sqrt{x^2 - a^2}} &= \frac{1}{a} \arccos \frac{a}{|x|}, \quad a > 0, & 63. \int \frac{dx}{x^2\sqrt{x^2 \pm a^2}} &= \mp \frac{\sqrt{x^2 \pm a^2}}{a^2 x}, \\
64. \int \frac{x dx}{\sqrt{x^2 \pm a^2}} &= \sqrt{x^2 \pm a^2}, & 65. \int \frac{\sqrt{x^2 \pm a^2}}{x^4} dx &= \mp \frac{(x^2 + a^2)^{3/2}}{3a^2 x^3}, \\
66. \int \frac{dx}{ax^2 + bx + c} &= \begin{cases} \frac{1}{\sqrt{b^2 - 4ac}} \ln \left| \frac{2ax + b - \sqrt{b^2 - 4ac}}{2ax + b + \sqrt{b^2 - 4ac}} \right|, & \text{if } b^2 > 4ac, \\ \frac{2}{\sqrt{4ac - b^2}} \arctan \frac{2ax + b}{\sqrt{4ac - b^2}}, & \text{if } b^2 < 4ac, \end{cases} \\
67. \int \frac{dx}{\sqrt{ax^2 + bx + c}} &= \begin{cases} \frac{1}{\sqrt{a}} \ln \left| 2ax + b + 2\sqrt{a}\sqrt{ax^2 + bx + c} \right|, & \text{if } a > 0, \\ \frac{1}{\sqrt{-a}} \arcsin \frac{-2ax - b}{\sqrt{b^2 - 4ac}}, & \text{if } a < 0, \end{cases} \\
68. \int \sqrt{ax^2 + bx + c} dx &= \frac{2ax + b}{4a} \sqrt{ax^2 + bx + c} + \frac{4ac - b^2}{8a} \int \frac{dx}{\sqrt{ax^2 + bx + c}}, \\
69. \int \frac{x dx}{\sqrt{ax^2 + bx + c}} &= \frac{\sqrt{ax^2 + bx + c}}{a} - \frac{b}{2a} \int \frac{dx}{\sqrt{ax^2 + bx + c}}, \\
70. \int \frac{dx}{x\sqrt{ax^2 + bx + c}} &= \begin{cases} \frac{-1}{\sqrt{c}} \ln \left| \frac{2\sqrt{c}\sqrt{ax^2 + bx + c} + bx + 2c}{x} \right|, & \text{if } c > 0, \\ \frac{1}{\sqrt{-c}} \arcsin \frac{bx + 2c}{|x|\sqrt{b^2 - 4ac}}, & \text{if } c < 0, \end{cases} \\
71. \int x^3 \sqrt{x^2 + a^2} dx &= \left(\frac{1}{3}x^2 - \frac{2}{15}a^2\right)(x^2 + a^2)^{3/2}, \\
72. \int x^n \sin(ax) dx &= -\frac{1}{a}x^n \cos(ax) + \frac{n}{a} \int x^{n-1} \cos(ax) dx, \\
73. \int x^n \cos(ax) dx &= \frac{1}{a}x^n \sin(ax) - \frac{n}{a} \int x^{n-1} \sin(ax) dx, \\
74. \int x^n e^{ax} dx &= \frac{x^n e^{ax}}{a} - \frac{n}{a} \int x^{n-1} e^{ax} dx, \\
75. \int x^n \ln(ax) dx &= x^{n+1} \left(\frac{\ln(ax)}{n+1} - \frac{1}{(n+1)^2} \right), \\
76. \int x^n (\ln ax)^m dx &= \frac{x^{n+1}}{n+1} (\ln ax)^m - \frac{m}{n+1} \int x^n (\ln ax)^{m-1} dx.
\end{aligned}$$

Finite Calculus

Difference, shift operators:

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

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

Fundamental Theorem:

$$f(x) = \Delta F(x) \Leftrightarrow \sum f(x) \delta x = F(x) + C.$$

$$\sum_a^b f(x) \delta x = \sum_{i=a}^{b-1} f(i).$$

Differences:

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

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

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

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

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

Sums:

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

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

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

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

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

Falling Factorial Powers:

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

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

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

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

Rising Factorial Powers:

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

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

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

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

Conversion:

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

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

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

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

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

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

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

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

Theoretical Computer Science Cheat Sheet

Series

Taylor's series:

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

Expansions:

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

Ordinary power series:

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

Exponential power series:

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

Dirichlet power series:

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

Binomial theorem:

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

Difference of like powers:

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

For ordinary power series:

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

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

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

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

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

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

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

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

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

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

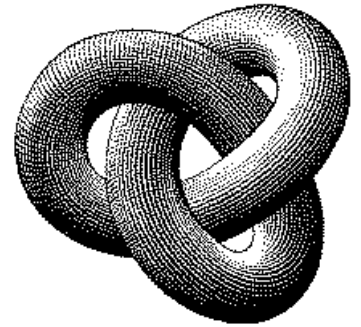
Series

Expansions:

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

$$\begin{aligned} \left(\frac{1}{x} \right)^{\overline{-n}} &= \sum_{i=0}^{\infty} \left\{ \begin{matrix} i \\ n \end{matrix} \right\} x^i, \\ (e^x - 1)^n &= \sum_{i=0}^{\infty} \left\{ \begin{matrix} i \\ n \end{matrix} \right\} \frac{n! x^i}{i!}, \\ x \cot x &= \sum_{i=0}^{\infty} \frac{(-4)^i B_{2i} x^{2i}}{(2i)!}, \\ \zeta(x) &= \sum_{i=1}^{\infty} \frac{1}{i^x}, \\ \frac{\zeta(x-1)}{\zeta(x)} &= \sum_{i=1}^{\infty} \frac{\phi(i)}{i^x}, \end{aligned}$$

Escher's Knot



Stieltjes Integration

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

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

exists. If $a \leq b \leq c$ 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

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

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

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

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

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:

$$a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n = b_1$$

$$a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n = b_2$$

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

$$a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,n}x_n = b_n$$

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

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

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

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

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

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

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

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:

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