



Team Contest Reference

Team:

System.out.println(42);

Roland Haase
Thore Tiemann
Marcel Wienöbst

Contents

| | | |
|----------|--|-----------|
| 1 | Algorithms | 2 |
| 1.1 | Binary Search | 2 |
| 1.2 | Bitonic TSP | 2 |
| 1.3 | Held Karp | 2 |
| 1.4 | Knuth-Morris-Pratt | 3 |
| 1.5 | Levenshtein Distance | 3 |
| 1.6 | Longest Common Subsequence | 3 |
| 1.7 | LongestIncreasingSubsequence | 4 |
| 1.8 | LongestIncreasingSubsequence | 4 |
| 1.9 | NextPermutation | 4 |
| 1.10 | Solve 2SAT | 5 |
| 2 | Graphs | 5 |
| 2.1 | BellmanFord | 5 |
| 2.2 | Bipartite Graph Check | 5 |
| 2.3 | Maximum Bipartite Matching | 6 |
| 2.4 | Depth First Search | 6 |
| 2.5 | Dijkstra | 6 |
| 2.6 | EdmondsKarp | 7 |
| 2.7 | FenwickTree | 7 |
| 2.8 | FloydWarshall | 7 |
| 2.9 | BFS AdjMtrx Iterativ | 8 |
| 2.10 | Kruskal | 8 |
| 2.11 | MinCut | 9 |
| 2.12 | Path-Based SCCs | 9 |
| 2.13 | Suurballe | 10 |
| 2.14 | Topological Sort | 10 |
| 3 | Math | 10 |
| 3.1 | Binomial Coefficient | 10 |
| 3.2 | Binomial Matrix | 11 |
| 3.3 | Graham Scan | 11 |
| 3.4 | Divisability | 11 |
| 3.5 | Iterative EEA | 12 |
| 3.6 | Polynomial Interpolation | 12 |
| 3.7 | Sieve of Eratosthenes | 13 |
| 4 | tcr-roland | 13 |
| 5 | Math Roland | 13 |
| 5.1 | Divisability Explanation | 13 |
| 5.2 | Combinatorics | 13 |
| 5.3 | Polynomial Interpolation | 13 |
| 5.3.1 | Theory | 13 |
| 5.4 | Fibonacci Sequence | 13 |
| 5.4.1 | Binet's formula | 13 |


```

30         if(tmp < min) {
31             min = tmp;
32             minprev = m;
33         }
34     }
35     C[S][k] = min;
36     p[S][k] = minprev;
37 }
38 }
39 }
40
41 //find shortest tour length
42 int min = Integer.MAX_VALUE;
43 int minprev = -1;
44 for(int k = 1; k < n; k++) {
45     //Set of all nodes except for the first + cost
46     //from 0 to k
47     int tmp = C[(1<<n) - 2][k] + graph[0][k];
48     if(tmp < min) {
49         min = tmp;
50         minprev = k;
51     }
52 }
53 //Note that the tour has not been tested yet, only
54 //the correctness of the min-tour-value
55 //backtrack tour
56 int[] tour = new int[n+1];
57 tour[n] = 0;
58 tour[n-1] = minprev;
59 int bits = (1<<n)-2;
60 for(int k = n-2; k>0; k--) {
61     tour[k] = p[bits][tour[k+1]];
62     bits = bits ^ (1<<tour[k+1]);
63 }
64 tour[0] = 0;
65 return tour;
66 }

```

MD5: 233d98980b1f4dae50ac892d7112dafb | $\mathcal{O}(2^n n^2)$

1.4 Knuth-Morris-Pratt

Input: String s to be searched, String w to search for.

Output: Array with all starting positions of matches

```

1 public static ArrayList<Integer> kmp(String s, String
2     w) {
3     ArrayList<Integer> ret = new ArrayList<>();
4     //Build prefix table
5     int[] N = new int[w.length()+1];
6     int i=0; int j=-1; N[0]=-1;
7     while (i<w.length()) {
8         while (j>=0 && w.charAt(j) != w.charAt(i))
9             j = N[j];
10        i++; j++; N[i]=j;
11    }
12    //Search string
13    i=0; j=0;
14    while (i<s.length()) {
15        while (j>=0 && s.charAt(i) != w.charAt(j))
16            j = N[j];
17        if (j==w.length()) { //match found
18            ret.add(i-w.length()); //add its start index
19            j = N[j];
20        }

```

```

21    }
22    return ret;
23 }

```

MD5: 3cb03964744db3b14b9bff265751c84b | $\mathcal{O}(n + m)$

1.5 Levenshtein Distance

Calculates the Levenshtein distance for two strings (minimum number of insertions, deletions, or substitutions).

Input: A string a and a string b .

Output: An integer holding the distance.

```

1 public static int levenshteinDistance(String a, String
2     b) {
3     a = a.toLowerCase();
4     b = b.toLowerCase();
5
6     int[] costs = new int[b.length() + 1];
7
8     for (int j = 0; j < costs.length; j++) {
9         costs[j] = j;
10    }
11
12    for (int i = 1; i <= a.length(); i++) {
13        costs[0] = i;
14        int nw = i - 1;
15        for (int j = 1; j <= b.length(); j++) {
16            int cj = Math.min(1 + Math.min(costs[j], costs[j
17                - 1]),
18                a.charAt(i - 1) == b.charAt(j - 1) ? nw : nw
19                    + 1);
20            nw = costs[j];
21            costs[j] = cj;
22        }
23    }
24    return costs[b.length()];
25 }

```

MD5: d9a487365717a996fbc91b2276fb0636 | $\mathcal{O}(|a| \cdot |b|)$

1.6 Longest Common Subsequence

Finds the longest common subsequence of two strings.

Input: Two strings $string1$ and $string2$.

Output: The LCS as a string.

```

1 public static String longestCommonSubsequence(String
2     string1, String string2) {
3
4     char[] s1 = string1.toCharArray();
5     char[] s2 = string2.toCharArray();
6
7     int[][] num = new int[s1.length + 1][s2.length + 1];
8
9     // Actual algorithm
10    for (int i = 1; i <= s1.length; i++)
11        for (int j = 1; j <= s2.length; j++)
12            if (s1[i - 1] == s2[j - 1])
13                num[i][j] = 1 + num[i - 1][j - 1];
14            else
15                num[i][j] = Math.max(num[i - 1][j], num[i][j -
16                    1]);

```

```

15 // System.out.println("length of LCS = " + num[s1.
16     length][s2.length]);
17
18 int s1position = s1.length, s2position = s2.length;
19 List<Character> result = new LinkedList<Character>()
20     ;
21 while (s1position != 0 && s2position != 0) {
22     if (s1[s1position - 1] == s2[s2position - 1]) {
23         result.add(s1[s1position - 1]);
24         s1position--;
25         s2position--;
26     } else if (num[s1position][s2position - 1] >= num[
27         s1position][s2position]) {
28         s2position--;
29     } else {
30         s1position--;
31     }
32 }
33 Collections.reverse(result);
34
35 char[] resultString = new char[result.size()];
36 int i = 0;
37 for (Character c : result) {
38     resultString[i] = c;
39     i++;
40 }
41
42 return new String(resultString);
43 }

```

MD5: c228e9d0a77d837f10900bc174cd3759 | $\mathcal{O}(n \cdot m)$

1.7 LongestIncreasingSubsequence

Computes the longest increasing subsequence and is easy to be adapted.

```

1 //This has not been tested yet (adapted from tested C
2 ++ Murcia Code)
3 public static int longestInc(int[] array, int N) {
4     int[] m = new int[N];
5     for (int i = N - 1; i >= 0; i--) {
6         m[i] = 1;
7         for (int j = i + 1; j < N; j++) {
8             if (array[j] > array[i]) {
9                 if (m[i] < m[j] + 1) {
10                     m[i] = m[j] + 1;
11                 }
12             }
13         }
14     }
15     int longest = 0;
16     for (int i = 0; i < N; i++) {
17         if (m[i] > longest) {
18             longest = m[i];
19         }
20     }
21     return longest;
22 }

```

MD5: 7ee618a580f2736226054b5e106d5635 | $\mathcal{O}(n^2)$

1.8 LongestIncreasingSubsequence

Computes the longest increasing subsequence using binary search.

```

1 public static int[] LongestIncreasingSubsequencenlogn(
2     int[] a, int[] p) {
3     int[] m = new int[a.length+1];
4     int l = 0;
5     for(int i = 0; i < a.length; i++) {
6         int lo = 1;
7         int hi = l;
8         while(lo <= hi) {
9             int mid = (int) (((lo + hi) / 2.0) + 0.6);
10            if(a[m[mid]] < a[i]) {
11                lo = mid+1;
12            } else {
13                hi = mid-1;
14            }
15        }
16        int newL = lo;
17        p[i] = m[newL-1];
18        m[newL] = i;
19        if(newL > l) {
20            l = newL;
21        }
22    }
23    int[] s = new int[l];
24    int k = m[l];
25    for(int i = l-1; i >= 0; i--) {
26        s[i] = a[k];
27        k = p[k];
28    }
29    return s;
30 }

```

MD5: e4b7591a2e204809f3e105521a616f70 | $\mathcal{O}(n \log n)$

1.9 NextPermutation

n Returns true if there is another permutation. Can also be used to compute the nextPermutation of an array.

```

1 public static boolean nextPermutation(char[] a) {
2     int i = a.length - 1;
3     while(i > 0 && a[i-1] >= a[i]) {
4         i--;
5     }
6     if(i <= 0) {
7         return false;
8     }
9     int j = a.length - 1;
10    while (a[j] <= a[i-1]) {
11        j--;
12    }
13    char tmp = a[i - 1];
14    a[i - 1] = a[j];
15    a[j] = tmp;
16
17    j = a.length - 1;
18    while(i < j) {
19        tmp = a[i];
20        a[i] = a[j];
21        a[j] = tmp;
22        i++;
23        j--;
24    }
25    return true;
26 }

```

```
26 }
MD5: ca6266722db16f2dc8eae5a6cc5fcacf | O(?)
```

1.10 Solve 2SAT

Allocate a graph with $|V| = 2 \cdot n$ for $x_{1..n}$. Add clauses, for example for $(x_1 \vee x_2) \wedge (\neg x_3 \vee x_4)$:

```
addClause(G,1,2); addClause(G,-3,4); int[]
b = solve2Sat(G);
returns a satisfying mapping for the  $x_i, i > 0$ , or null.
```

```
1 public static void addClause(Vertex[] G, int a, int b)
2 {
3     int nega = a<0 ? 0 : 1; int negb = b<0 ? 0 : 1;
4     a = Math.abs(a)-1; b = Math.abs(b)-1;
5     int Xa = (a<<1)+nega; int Xb = (b<<1)+negb;
6     G[Xa^1].next.add(Xb);
7     G[Xb^1].next.add(Xa);
8 }
9 public static int[] solve2Sat(Vertex[] G) {
10     Integer[] color = scc(G);
11     for (int i=0; i<G.length; i+=2)
12         if (color[i] == color[i+1])
13             return null; //contradiction!
14
15     HashSet<Integer>[] sccV = new HashSet[G.length];
16     HashSet<Integer>[] sccEn = new HashSet[G.length];
17     HashSet<Integer>[] sccEp = new HashSet[G.length];
18     Integer[] vals = new Integer[G.length];
19     for (int i=0; i<G.length; i++) {
20         sccV[i] = new HashSet<Integer>();
21         sccEn[i] = new HashSet<Integer>();
22         sccEp[i] = new HashSet<Integer>();
23     }
24     //create reverse SCC DAG
25     for (int i=0; i<G.length; i++)
26         if (G[i]!=null) {
27             sccV[color[i]].add(i);
28             for (int j : G[i].next)
29                 if (color[i] != color[j]) {
30                     sccEn[color[i]].add(color[j]);
31                     sccEp[color[j]].add(color[i]);
32                 }
33         }
34     //go in rev topo order and set vars
35     Stack<Integer> tail = new Stack<Integer>();
36     for (int i=0; i<G.length; i++)
37         if (!sccV[i].isEmpty() && sccEn[i].isEmpty())
38             tail.push(i);
39     while (!tail.isEmpty()) {
40         int curr = tail.pop();
41         for (int i : sccV[curr]) {
42             if (vals[i]!=null)
43                 break;
44             vals[i] = 1;
45             vals[i^1] = 0;
46         }
47         for (int i : sccEp[curr]) {
48             sccEn[i].remove(curr);
49             if (sccEn[i].isEmpty())
50                 tail.push(i);
51         }
52     }
53     int[] ret = new int[G.length/2+1];
```

```
54 for (int i=0; i<G.length; i+=2)
55     if (vals[i+1]==1)
56         ret[i/2+1] = 1;
57     return ret;
58 }
```

MD5: 60fb0af11d8fc325eb0efb71031ca312 | $O(|E| + |V|)$

2 Graphs

2.1 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[] vertices) {
2     //source is 0
3     vertices[0].mindistance = 0;
4     //calc distances
5     for(int i = 0; i < vertices.length-1; i++) {
6         for(int j = 0; j < vertices.length; j++) {
7             for(Edge e: vertices[j].adjacencies) {
8                 if(vertices[j].mindistance != Integer.
9                     MAX_VALUE
10                    && e.target.mindistance > vertices[j].
11                        mindistance + e.distance) {
12                     e.target.mindistance = vertices[j].
13                         mindistance + e.distance;
14                 }
15             }
16         }
17     }
18     //check for negative-length cycle
19     for(int i = 0; i < vertices.length; i++) {
20         for(Edge e: vertices[i].adjacencies) {
21             if(vertices[i].mindistance != Integer.
22                 MAX_VALUE && e.target.mindistance >
23                 vertices[i].mindistance + e.distance) {
24                 return true;
25             }
26         }
27     }
28     return false;
29 }
```

MD5: 36561a7913a81baf7b7c79b606683819 | $O(|V| \cdot |E|)$

2.2 Bipartite Graph Check

Checks a graph represented as adjList for being bipartite.

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

OUTPUT: true if graph is bipartite, false otherwise

```
1 public static boolean bipartiteGraphCheck(
2     ArrayList<ArrayList<Integer>> graph, int N) {
3     int[] color = new int[N];
4     for(int i = 0; i < N; i++) color[i] = -1;
5     // use bfs for coloring each node
6     color[0] = 1;
7     // FIFO-Queue
8     Queue<Integer> q = new LinkedList<Integer>();
9     q.add(0);
10    while(!q.isEmpty()) {
11        int u = q.poll();
12        for(int i : graph.get(u)) {
```

```

13 // if node i not yet visited,
14 // give opposite color of parent node u
15 if(color[i] == -1) {
16     color[i] = 1-color[u];
17     q.add(i);
18 // if node i has same color as parent node u
19 // the graph is not bipartite
20 } else if(color[u] == color[i])
21     return false;
22 // if node i has different color
23 // than parent node u keep going
24 }
25 }
26 return true;
27 }

```

MD5: 248cb70cd02d89421b8f4f6a8d551add | $\mathcal{O}(|V| + |E|)$

2.3 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 | $\mathcal{O}(M \cdot N)$

2.4 Depth First Search

Searches for a path between two vertices in a graph per DFS.

Input: A source vertex `s`, a target vertex `t`, an adjacency matrix `G` and two new (empty) lists `path` and `list` (for recursion).

Output: A boolean, indicating whether a path exists or not. If a path exists, a possible path is stored in `path`.

```

1 public static boolean DFS(int s, int t, int[][] G,
2 List<Integer> path, List<Integer> list) {
3     if (path.size() == 0) {
4         path.add(s);
5     }
6     if (s == t) {
7         return true;
8     }
9
10    for (int i = 0; i < G.length; i++) {
11        if (G[s][i] > 0 && !list.contains(i)) {
12            path.add(i);
13            list.add(i);
14            if (DFS(i, t, G, path, list)) {
15                return true;
16            } else {
17                path.remove(path.size() - 1);
18            }
19        }
20    }
21    return false;
22 }

```

MD5: 596c08e2603bb329abbc92058f0386dd | $\mathcal{O}(|V|^2)$

2.5 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[] vertices, int src
2 ) {

```

```

2 vertices[src].mindistance = 0;
3 PriorityQueue<Vertex> queue = new PriorityQueue<
  Vertex>();
4 queue.add(vertices[src]);
5 while(!queue.isEmpty()) {
6     Vertex u = queue.poll();
7     if(u.visited)
8         continue;
9     u.visited = true;
10    for(Edge e : u.adjacencies) {
11        Vertex v = e.target;
12        if(v.mindistance > u.mindistance + e.distance
13            ) {
14            v.mindistance = u.mindistance + e.distance
15                ;
16            queue.add(v);
17        }
18    }
19 }
20
21 class Vertex implements Comparable<Vertex> {
22     public int id;
23     public int mindistance = Integer.MAX_VALUE;
24     public LinkedList<Edge> adjacencies = new
25         LinkedList<Edge>();
26     public boolean visited = false;
27
28     public int compareTo(Vertex other) {
29         return Integer.compare(this.mindistance, other.
30             mindistance);
31     }
32 }
33
34 class Edge {
35     public Vertex target;
36     public int distance;
37
38     public Edge (Vertex target, int distance) {
39         this.target = target;
40         this.distance = distance;
41     }
42 }

```

MD5: d6882162849418a2541cfc7f6c3ddc58 | $\mathcal{O}(|E| \log |V|)$

2.6 EdmondsKarp

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

```

1 public static boolean BFS(int[][] graph, int s, int t,
2     int[] parent) {
3     int N = graph.length;
4     boolean[] visited = new boolean[N];
5     for(int i = 0; i < N; i++) {
6         visited[i] = false;
7     }
8     Queue<Integer> queue = new LinkedList<Integer>();
9     queue.add(s);
10    visited[s] = true;
11    parent[s] = -1;
12    while(!queue.isEmpty()) {
13        int u = queue.poll();
14        if(u == t) return true;
15        for(int v = 0; v < N; v++) {
16            if(visited[v] == false && graph[u][v] > 0) {
17                queue.add(v);
18                parent[v] = u;
19                visited[v] = true;
20            }
21        }
22    }
23    return false;
24 }

```

```

19     }
20 }
21
22 return (visited[t]);
23 }
24
25 public static int fordFulkerson(int[][] graph, int s,
26     int t) {
27     int N = graph.length;
28     int[][] rgraph = new int[graph.length][graph.length
29         ];
30     for(int u = 0; u < graph.length; u++) {
31         for(int v = 0; v < graph.length; v++) {
32             rgraph[u][v] = graph[u][v];
33         }
34     }
35     int[] parent = new int[N];
36     int maxflow = 0;
37     while(BFS(rgraph, s, t, parent)) {
38         int pathflow = Integer.MAX_VALUE;
39         for(int v = t; v != s; v = parent[v]) {
40             int u = parent[v];
41             pathflow = Math.min(pathflow, rgraph[u][v]);
42         }
43         for(int v = t; v != s; v = parent[v]) {
44             int u = parent[v];
45             rgraph[u][v] -= pathflow;
46             rgraph[v][u] += pathflow;
47         }
48         maxflow += pathflow;
49     }
50     return maxflow;
51 }

```

MD5: 8d85785d45794f20303d9b9f920e80dd | $\mathcal{O}(|V|^2 \cdot |E|)$

2.7 FenwickTree

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
11 public static int[] update(int index, int addValue,
12     int[] fenwickTree) {
13     while (index <= fenwickTree.length - 1) {
14         fenwickTree[index] += addValue;
15         index += (index & -index);
16     }
17     return fenwickTree;
18 }

```

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

2.8 FloydWarshall

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

```

1 public static void floydWarshall(int[][] graph, int
2     [][] next, int[][] ans) {

```



```

2   for(int i = 0; i < ans.length; i++) {
3       for(int j = 0; j < ans.length; j++) {
4           ans[i][j] = graph[i][j];
5       }
6   }
7   for (int k = 0; k < ans.length; k++) {
8       for (int i = 0; i < ans.length; i++) {
9           for (int j = 0; j < ans.length; j++) {
10              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 | $\mathcal{O}(|V|^3)$

2.9 BFS AdjMtrx Iterativ

Iterative BFS on adjacency matrix. Returns true or false, depending on whether there is a connection between s and g or not.

```

1   public static boolean BFSWithoutPathForAdjMatr(int s,
2       int g, int[][] graph) {
3       //s being the start and g the goal
4       boolean[] visited = new boolean[graph.length];
5       for(int i = 0; i < visited.length; i++)
6           visited[i] = false;
7       Queue<Integer> queue = new LinkedList<Integer>();
8       queue.add(s);
9       visited[s] = true;
10      while(!queue.isEmpty()) {
11          int node = queue.poll();
12          if(node == g)
13              return true;
14          for(int i = 0; i < graph.length; i++) {
15              if(graph[node][i] > 0 && !visited[i]) {
16                  queue.add(i);
17                  visited[i] = true;
18              }
19          }
20      }
21      return false;

```

MD5: 754e7dfa0a691a2511464e16104b8880 | $\mathcal{O}(|V| + |E|)$

2.10 Kruskal

Computes a minimum spanning tree for a weighted undirected graph.

```

1   public class Freckles {
2       public static void main(String[] args) {
3           Scanner s = new Scanner(System.in);
4           int t = s.nextInt();
5           for(int i = 0; i < t; i++) {
6               int n = s.nextInt();
7               double[] x = new double[n];
8               double[] y = new double[n];
9               for(int j = 0; j < n; j++) {
10                  x[j] = s.nextDouble();

```

```

        y[j] = s.nextDouble();
    }
    Edge1[] edge = new Edge1[n*n];
    for(int j = 0; j < n; j++) {
        for(int l = 0; l < n; l++) {
            double distance = Math.sqrt((x[l]-x[j])
                * (x[l] - x[j]) + (y[l]-y[j]) * (y
                    [l] - y[j]));
            edge[j * n + l] = new Edge1(distance, j
                , l);
        }
    }
    Arrays.sort(edge);
    UnionFind uf = new UnionFind(n);
    double sum = 0;
    int cnt = 0;
    for(int j = 0; j < n*n; j++) {
        if(cnt == n-1)
            break;
        if(uf.union(edge[j].start, edge[j].end)) {
            sum += edge[j].distance;
            cnt++;
        }
    }
    System.out.printf("%.2f
", sum);
    if(i < t-1)
        System.out.println();
    }
}

class UnionFind {
    private int[] p = null;
    private int[] r = null;
    private int count = 0;

    public int count() {
        return count;
    } // number of sets

    public UnionFind(int n) {
        count = n; // every node is its own set
        r = new int[n]; // every node is its own tree
                        with height 0
        p = new int[n];
        for (int i = 0; i < n; i++)
            p[i] = -1; // no parent = -1
    }

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

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

```



```

74     if (px == py)
75         return false; // same set -> reject edge
76     if (r[px] < r[py]) { // swap so that always h[px]
77         ]>=h[py]
78         int tmp = px;
79         px = py;
80         py = tmp;
81     }
82     p[py] = px; // hang flatter tree as child of
83     higher tree
84     r[px] = Math.max(r[px], r[py] + 1); // update (
85     worst-case) height
86     count--;
87     return true;
88 }
89
90 class Edge1 implements Comparable<Edge1> {
91     double distance;
92     int start;
93     int end;
94
95     public Edge1(double distance, int start, int end) {
96         this.distance = distance;
97         this.start = start;
98         this.end = end;
99     }
100
101     public int compareTo(Edge1 arg0) {
102         return Double.compare(this.distance, arg0.
distance);
103     }
104 }

```

MD5: 5d75c90ca7d6a6d3a041079a766a99fe | $\mathcal{O}(|E| + \log |V|)$

2.11 MinCut

Calculates the min-cut of a graph (represented as adjMtrx).

```

1 public static void MinCut(int s, int[][] graph,
2     LinkedList<Integer> S, LinkedList<Integer> T) {
3     boolean[] visited = new boolean[graph.length];
4     for(int i = 0; i < visited.length; i++)
5         visited[i] = false;
6     Queue<Integer> queue = new LinkedList<Integer>();
7     queue.add(s);
8     S.add(s);
9     visited[s] = true;
10    while(!queue.isEmpty()) {
11        int node = queue.poll();
12        for(int i = 0; i < graph.length; i++) {
13            if(graph[node][i] > 0 && !visited[i]) {
14                queue.add(i);
15                if(!S.contains(i))
16                    S.add(i);
17                visited[i] = true;
18            }
19        }
20    }
21    for(int i = 0; i < graph.length; i++) {
22        if(!S.contains(i)) {
23            T.add(i);
24        }
25    }
26    for(int i = 0; i < graph.length; i++) {
27        for(int j = 0; j < graph.length; j++) {

```

```

27         if((graph[i][j] > 0 || graph[j][i] > 0) && S.
28             contains(i) && T.contains(j)) {
29             System.out.println((i+1) + " " + (j+1));
30         }
31     }
32 }

```

MD5: 57afc679d5d50ed15f504244aad43bc8 | $\mathcal{O}(?)$

2.12 Path-Based SCCs

Finds the strongly connected components in given directed graph.

```

1 public static Integer[] scc(Vertex[] G) {
2     Stack<Integer> call = new Stack<>();
3
4     Stack<Integer> reps = new Stack<>();
5     Stack<Integer> open = new Stack<>();
6     Integer[] order = new Integer[G.length];
7     int count = 0;
8
9     Integer[] sccs = new Integer[G.length];
10    int sccnum = 0;
11
12    for (int i=0; i<G.length; i++) {
13        if (G[i]==null) //no such vertex
14            continue;
15
16        if (sccs[i]==null) {
17            call.push(i);
18            while (!call.isEmpty()) {
19                int v = call.peek();
20                if (order[v]==null) { //first entered
21                    order[v] = count++;
22                    reps.push(v);
23                    open.push(v);
24
25                    for (int w : G[v].next) { //process edges
26                        if (order[w]==null) {
27                            call.push(w);
28                        } else if (sccs[w]==null) {
29                            while (order[reps.peek()]>order[w])
30                                reps.pop();
31                        }
32                    }
33
34                    } else { //returned from recursion
35                        //is still rep. -> completed SCC
36                        if (reps.peek()==v) {
37                            int tmp = 0;
38                            do {
39                                tmp = open.pop();
40                                sccs[tmp] = sccnum;
41                            } while (tmp != v);
42                            sccnum++;
43                            reps.pop();
44                        }
45
46                        call.pop(); //node done
47                    }
48                }
49            }
50        }
51        return sccs;
52    }

```

MD5: a88a646c1ef6c1a60d9eb122ea1b6c4b | $\mathcal{O}(|E| + |V|)$

2.13 Suurballe

Finds two edge-disjoint paths from s to t with minimal sum length. depends on Dijkstra. Add to Vertex class 2 HashMaps backupNext and resultSuurballe. For also vertex-disjoint paths split vertices in in- and outgoing vertices connected with zero-valued edges.

```

1 public static int suurballe(int s, int t, Vertex[] G)
2 {
3     dijkstra(s, G); //find a shortest path
4     ArrayList<Integer> path = new ArrayList<Integer>();
5     int id = t;
6     while (G[id].pred != id) {
7         path.add(0, id);
8         id = G[id].pred;
9     }
10    path.add(0, id);
11
12    //modify weights
13    for (int i=0; i<G.length; i++) {
14        Vertex u = G[i];
15        if (u==null) continue;
16        u.backupNext = new HashMap<Integer,Integer>(u.next
17            ); //copy old values
18        for (Integer j : u.backupNext.keySet()) {
19            Vertex v = G[j];
20            int weight = u.next.get(j);
21            u.next.put(j, weight - v.dist + u.dist);
22        }
23    }
24    //reverse edges on shortest path
25    id = s;
26    for (int i=0; i<path.size()-1; i++) {
27        G[path.get(i)].next.remove(path.get(i+1));
28        G[path.get(i+1)].next.put(path.get(i), 0);
29    }
30    //remove edges to s
31    for (int i=0; i<G.length; i++) {
32        if (G[i]==null) continue;
33        if (G[i].next.containsKey(s))
34            G[i].next.remove(s);
35    }
36    dijkstra(s, G);
37    ArrayList<Integer> path2 = new ArrayList<Integer>();
38    id = t;
39    if (G[id].pred == -1)
40        return -1; //no 2nd path!
41
42    while (G[id].pred != id) {
43        path2.add(0, id);
44        id = G[id].pred;
45    }
46    path2.add(0, id);
47
48    int totalpath = 0;
49
50    //disregard 0-cycles and edges not on both paths
51    id = s;
52    //add edges on first shortest path
53    for (int i=0; i<path.size()-1; i++) {
54        int u = path.get(i);
55        int v = path.get(i+1);

```

```

56        G[u].suurballeResult.put(v, G[u].backupNext.get(v))
57        ;
58        totalpath += G[u].suurballeResult.get(v);
59    }
60    //add second path, remove cycles
61    for (int i=0; i<path2.size()-1; i++) {
62        int u = path2.get(i);
63        int v = path2.get(i+1);
64
65        if (G[v].suurballeResult.containsKey(u)) {
66            totalpath -= G[v].suurballeResult.get(u);
67            G[v].suurballeResult.remove(u);
68        } else {
69            G[u].suurballeResult.put(v, G[u].backupNext.get(v)
70                );
71            totalpath += G[u].suurballeResult.get(v);
72        }
73    }
74    return totalpath;

```

MD5: b57c5d377ec0af5e1145a05d471a0437 | $\mathcal{O}(|E| + |V| \log |V|)$

2.14 Topological Sort

Sorts a graph (represented as adjMtrx) topologically

```

1 // l enthaelt alle Knoten topologisch sortiert (Start:
2 // 0, Ende= n)
3 int[] l = new int[n];
4 int idx = 0;
5 // s enthaelt alle Knoten, die keine eingehende Kante
6 // haben
7 ArrayList<Integer> s = new ArrayList<Integer>();
8 // initialisiere s
9 for (int i = 0; i < n; i++) {
10     if (edgesIn[i] == 0) {
11         s.add(i);
12     }
13 }
14 // Algo Beginn
15 while (!s.isEmpty()) {
16     int node = s.remove(0);
17     l[idx++] = node;
18     for (int i = 0; i < n; i++) {
19         if (adjMtrx[node][i]) {
20             adjMtrx[node][i] = false;
21             edgesIn[i] -= 1;
22             if (edgesIn[i] == 0) {
23                 s.add(i);
24             }
25         }
26     }
27 }

```

MD5: 01974f4bab4e48916ecdc48531a79c84 | $\mathcal{O}(|V| + |E|)$

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 Graham Scan

GrahamScan finds convex hull. Still has collinear point problem-atic at the last diagonal.

```

1 public static int ccw(Point src, Point q1, Point q2) {
2     return (q1.x - src.x) * (q2.y - src.y) - (q2.x -
3         src.x) * (q1.y - src.y);
4 }
5 public static boolean isColl(Point a, Point b, Point c) {
6     if ((b.y - a.y) * (c.x - b.x) == (c.y - b.y) * (b.x
7         - a.x)) {
8         return true;
9     } else {
10        return false;
11    }
12 }
13 public static double calcDist(Point src, Point target) {
14     return Math.sqrt((src.x + target.x) * (src.x +
15         target.x) + (src.y + target.y) * (src.y +
16         target.y));
17 }
18 //Expects a array sorted with PolarComp as Comparator
19 //IMPORTANT! before sorting put lowest, and if two are
20 //the same leftmost, element at position 0 in array
21 public static void grahamScan(Point[] points) {

```

```

20 int m = 1;
21 for(int i = 2; i < points.length; i++) {
22     while(ccw(points[m-1], points[m], points[i]) <
23         0) {
24         if(m > 1) m--;
25         else if(i == points.length) break;
26         else i++;
27     }
28     m++;
29     Point tmp = points[i];
30     points[i] = points[m];
31     points[m] = tmp;
32 }

```

```

33 class Point {
34     int x;
35     int y;
36     public Point(int x, int y) {
37         this.x = x;
38         this.y = y;
39     }
40 }
41 class PolarComp implements Comparator<Point> {
42     Point src;
43     public PolarComp(Point source) {
44         src = source;
45     }
46     public double calcDist(Point q1, Point q2) {
47         return Math.sqrt((q1.x - q2.x) * (q1.x - q2.x) +
48             (q1.y - q2.y) * (q1.y - q2.y));
49     }
50     public int ccw(Point q1, Point q2) {
51         return (q1.x - src.x) * (q2.y - src.y) - (q2.x -
52             src.x) * (q1.y - src.y);
53     }
54     public int compare(Point q1, Point q2) {
55         int res = ccw(q1, q2);
56         double dist1 = calcDist(src, q1);
57         double dist2 = calcDist(src, q2);
58         if(res > 0) return -1;
59         else if(res < 0) return 1;
60         else if(res == 0 && dist1 < dist2) return 1;
61         else if(res == 0 && dist1 > dist2) return -1;
62         else return 0;
63     }
64 }

```

MD5: 97ad3ab5efa1cbfa7374a86aa2db7f62 | $\mathcal{O}(n \log n)$

3.4 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;

```

```

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

```

MD5: 33b3094ebf431e1e71cd8e8db3c9cdd6 | $\mathcal{O}(?)$

3.5 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
17    long[] erg = { gcd, x, y };
18    return erg;
19 }

```

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

3.6 Polynomial Interpolation

```

1 public class interpol {
2
3     // divided differences for points given by vectors x
    // and y
4     public static rat[] divDiff(rat[] x, rat[] y) {
5         rat[] temp = y.clone();
6         int n = x.length;
7         rat[] res = new rat[n];
8         res[0] = temp[0];
9         for (int i=1; i < n; i++) {
10            for (int j = 0; j < n-i; j++) {
11                temp[j] = (temp[j+1].sub(temp[j])).div(x[j+i].
                    sub(x[j]));
12            }
13            res[i] = temp[0];
14        }
15        return res;
16    }
17
18    // evaluates interpolating polynomial p at t for
    // given

```

```

// x-coordinates and divided differences
public static rat p(rat t, rat[] x, rat[] dD) {
    int n = x.length;
    rat p = new rat(0);
    for (int i = n-1; i > 0; i--) {
        p = (p.add(dD[i])).mult(t.sub(x[i-1]));
    }
    p = p.add(dD[0]);
    return p;
}

public static void main(String[] args) {

    rat[] test = {new rat(4,5), new rat(7,10), new rat
        (3,4)};
    test = rat.commonDenominator(test);
    for (int i = 0; i < test.length; i++) {
        System.out.println(test[i].toString());
    }

    rat[] x = {new rat(0),new rat(1), new rat(2), new
        rat(3), new rat(4), new rat(5)};
    rat[] y = {new rat(-10), new rat(9), new rat(0),
        new rat(1), new rat(1,2), new rat(1,80)};
    rat[] dD = divDiff(x,y);
    System.out.println("p("+7+")_u=u"+p(new rat(7), x,
        dD));
}

// implementation of rational numbers
class rat {

    public long c;
    public long d;

    public rat (long c, long d) {
        this.c = c;
        this.d = d;
        this.shorten();
    }

    public rat (long c) {
        this.c = c;
        this.d = 1;
    }

    public static long ggT(long a, long b) {
        while (b != 0) {
            long h = a%b;
            a = b;
            b = h;
        }
        return a;
    }

    public static long kgV(long a, long b) {
        return a*b/ggT(a,b);
    }

    public static rat[] commonDenominator(rat[] c) {
        long kgV = 1;
        for (int i = 0; i < c.length; i++) {
            kgV = kgV(kgV, c[i].d);
        }
        for (int i = 0; i < c.length; i++) {
            c[i].c *= kgV/c[i].d;
            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
120 }

```

MD5: d98bd247b95395d8596ff1d5785ee06b | O(?)

3.7 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 tcr-roland

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

$$g_n = \frac{1}{\sqrt{5}}(g_0(\phi^{n-1} - \tilde{\phi}^{n-1}) + g_1(\phi^n - \tilde{\phi}^n)) = g_0f_{n-1} + g_1f_n$$

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

```
1 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. $\left[\begin{matrix} n \\ 2 \end{matrix} \right] = (n-1)!H_{n-1},$ | 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. $\sum_{k=0}^n \left[\begin{matrix} n \\ k \end{matrix} \right] = n!,$ | 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 \begin{bmatrix} n+1 \\ m+1 \end{bmatrix} &= \sum_k \begin{bmatrix} n \\ k \end{bmatrix} \begin{bmatrix} k \\ m \end{bmatrix} = \sum_{k=0}^n \begin{bmatrix} k \\ m \end{bmatrix} n^{\overline{n-k}} = n! \sum_{k=0}^n \frac{1}{k!} \begin{bmatrix} k \\ m \end{bmatrix}, & 39. \quad \begin{bmatrix} x \\ x-n \end{bmatrix} &= \sum_{k=0}^n \left\langle \begin{matrix} n \\ k \end{matrix} \right\rangle \begin{bmatrix} x+k \\ 2n \end{bmatrix}, \\
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 \begin{bmatrix} n \\ m \end{bmatrix} &= \sum_k \begin{bmatrix} n+1 \\ k+1 \end{bmatrix} \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 \begin{bmatrix} m+n+1 \\ m \end{bmatrix} &= \sum_{k=0}^m k(n+k) \begin{bmatrix} n+k \\ k \end{bmatrix}, \\
44. \quad \binom{n}{m} &= \sum_k \left\{ \begin{matrix} n+1 \\ k+1 \end{matrix} \right\} \begin{bmatrix} k \\ m \end{bmatrix} (-1)^{m-k}, & 45. \quad (n-m)! \binom{n}{m} &= \sum_k \begin{bmatrix} n+1 \\ k+1 \end{bmatrix} \left\{ \begin{matrix} k \\ m \end{matrix} \right\} (-1)^{m-k}, \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} \begin{bmatrix} m+k \\ k \end{bmatrix}, & 47. \quad \begin{bmatrix} n \\ n-m \end{bmatrix} &= \sum_k \binom{m-n}{m+k} \binom{m+n}{n+k} \left\{ \begin{matrix} m+k \\ k \end{matrix} \right\}, \\
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 \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}.
\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:

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

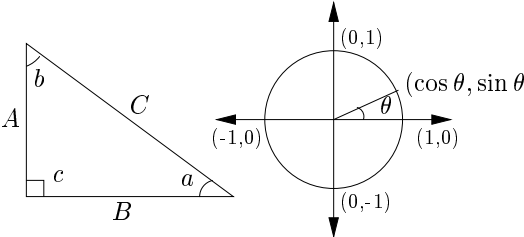
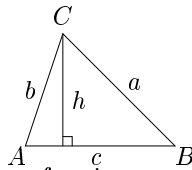
So $g_i = 2^i - 1$.

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 $i \neq 1$): | Continuous distributions: If |
| 2 | 4 | 3 | $B_0 = 1, B_1 = -\frac{1}{2}, B_2 = \frac{1}{6}, B_4 = -\frac{1}{30},$ | $\Pr[a < X < b] = \int_a^b p(x) dx,$ |
| 3 | 8 | 5 | $B_6 = \frac{1}{42}, B_8 = -\frac{1}{30}, B_{10} = \frac{5}{66}.$ | then p is the probability density function of X . If |
| 4 | 16 | 7 | Change of base, quadratic formula: | $\Pr[X < a] = P(a),$ |
| 5 | 32 | 11 | $\log_b x = \frac{\log_a x}{\log_a b}, \quad \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$ | then P is the distribution function of X . If P and p both exist then |
| 6 | 64 | 13 | Euler's number e : | $P(a) = \int_{-\infty}^a p(x) dx.$ |
| 7 | 128 | 17 | $e = 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + \frac{1}{120} + \dots$ | Expectation: If X is discrete |
| 8 | 256 | 19 | $\lim_{n \rightarrow \infty} \left(1 + \frac{x}{n}\right)^n = e^x.$ | $E[g(X)] = \sum_x g(x) \Pr[X = x].$ |
| 9 | 512 | 23 | $\left(1 + \frac{1}{n}\right)^n < e < \left(1 + \frac{1}{n}\right)^{n+1}.$ | If X continuous then |
| 10 | 1,024 | 29 | $\left(1 + \frac{1}{n}\right)^n = e - \frac{e}{2n} + \frac{11e}{24n^2} - O\left(\frac{1}{n^3}\right).$ | $E[g(X)] = \int_{-\infty}^{\infty} g(x)p(x) dx = \int_{-\infty}^{\infty} g(x) dP(x).$ |
| 11 | 2,048 | 31 | Harmonic numbers: | Variance, standard deviation: |
| 12 | 4,096 | 37 | $1, \frac{3}{2}, \frac{11}{6}, \frac{25}{12}, \frac{137}{60}, \frac{49}{20}, \frac{363}{140}, \frac{761}{280}, \frac{7129}{2520}, \dots$ | $\text{VAR}[X] = E[X^2] - E[X]^2,$ |
| 13 | 8,192 | 41 | $\ln n < H_n < \ln n + 1,$ | $\sigma = \sqrt{\text{VAR}[X]}.$ |
| 14 | 16,384 | 43 | $H_n = \ln n + \gamma + O\left(\frac{1}{n}\right).$ | For events A and B : |
| 15 | 32,768 | 47 | Factorial, Stirling's approximation: | $\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$ |
| 16 | 65,536 | 53 | $1, 2, 6, 24, 120, 720, 5040, 40320, 362880, \dots$ | $\Pr[A \wedge B] = \Pr[A] \cdot \Pr[B],$ |
| 17 | 131,072 | 59 | $n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right).$ | iff A and B are independent. |
| 18 | 262,144 | 61 | Ackermann's function and inverse: | $\Pr[A B] = \frac{\Pr[A \wedge B]}{\Pr[B]}$ |
| 19 | 524,288 | 67 | $a(i, j) = \begin{cases} 2^j & i = 1 \\ a(i-1, 2) & j = 1 \\ a(i-1, a(i, j-1)) & i, j \geq 2 \end{cases}$ | For random variables X and Y : |
| 20 | 1,048,576 | 71 | $\alpha(i) = \min\{j \mid a(j, j) \geq i\}.$ | $E[X \cdot Y] = E[X] \cdot E[Y],$ |
| 21 | 2,097,152 | 73 | Binomial distribution: | if X and Y are independent. |
| 22 | 4,194,304 | 79 | $\Pr[X = k] = \binom{n}{k} p^k q^{n-k}, \quad q = 1 - p,$ | $E[X + Y] = E[X] + E[Y],$ |
| 23 | 8,388,608 | 83 | $E[X] = \sum_{k=1}^n k \binom{n}{k} p^k q^{n-k} = np.$ | $E[cX] = c E[X].$ |
| 24 | 16,777,216 | 89 | Poisson distribution: | Bayes' theorem: |
| 25 | 33,554,432 | 97 | $\Pr[X = k] = \frac{e^{-\lambda} \lambda^k}{k!}, \quad E[X] = \lambda.$ | $\Pr[A_i B] = \frac{\Pr[B A_i] \Pr[A_i]}{\sum_{j=1}^n \Pr[A_j] \Pr[B A_j]}.$ |
| 26 | 67,108,864 | 101 | Normal (Gaussian) distribution: | Inclusion-exclusion: |
| 27 | 134,217,728 | 103 | $p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}, \quad E[X] = \mu.$ | $\Pr\left[\bigvee_{i=1}^n X_i\right] = \sum_{i=1}^n \Pr[X_i] +$ |
| 28 | 268,435,456 | 107 | 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 | $\sum_{k=2}^n (-1)^{k+1} \sum_{i_1 < \dots < i_k} \Pr\left[\bigwedge_{j=1}^k X_{i_j}\right].$ |
| 29 | 536,870,912 | 109 | $nH_n.$ | Moment inequalities: |
| 30 | 1,073,741,824 | 113 | | $\Pr[X \geq \lambda E[X]] \leq \frac{1}{\lambda},$ |
| 31 | 2,147,483,648 | 127 | | $\Pr[X - E[X] \geq \lambda \cdot \sigma] \leq \frac{1}{\lambda^2}.$ |
| 32 | 4,294,967,296 | 131 | | Geometric distribution: |
| Pascal's Triangle | | | | $\Pr[X = k] = pq^{k-1}, \quad q = 1 - p,$ |
| 1 | | | | $E[X] = \sum_{k=1}^{\infty} kpq^{k-1} = \frac{1}{p}.$ |
| 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>v2.01 ©1994 by Steve Seiden sseiden@acm.org http://www.csc.lsu.edu/~seiden</p> | <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 | ∞ | | | | | | | | | | | | | | | | | | | | | | | |

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:

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

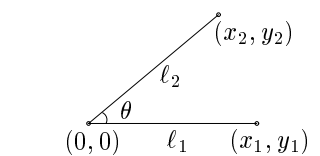
$$[|x_1 - x_0|^p + |y_1 - y_0|^p]^{1/p},$$

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

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:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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:

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

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

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

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

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

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

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

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

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:

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

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

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

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

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

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

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

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

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

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

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

Convolution:

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

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

Theoretical Computer Science Cheat Sheet

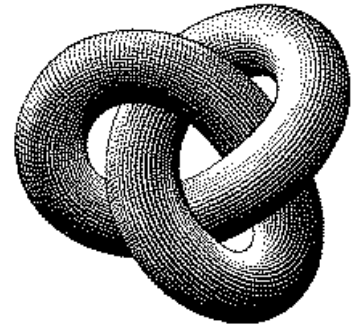
Series

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:

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