

Team Contest Reference

Ballmer Peak

Universität zu Lübeck

19. November 2013

Inhaltsverzeichnis

1	Mathematische Algorithmen	2
1.1	Primzahlen	2
1.1.1	Sieb des Eratosthenes	2
1.1.2	Primzahlentest	2
1.2	Binomial Koeffizient	2
1.3	Modulare Arithmetik	2
1.3.1	Erweiterter Euklidischer Algorithmus	2
1.4	Matrixmultiplikation	2
2	Datenstrukturen	2
2.1	Fenwick Tree (Binary Indexed Tree)	2
3	Graphenalgorithmen	2
3.1	Topologische Sortierung	2
3.2	Minimum Spanning Tree	3
3.2.1	Prim's Algorithm	3
3.2.2	Union and Find: Kruskal's Algorithm	3
3.3	Maximaler Fluss (Ford-Fulkerson)	3
3.4	Floyd-Warshall	4
3.5	Dijkstra	4
3.6	Bellmann-Ford	5
3.7	Starke Zusammenhangskomponenten (Kosaraju)	5
4	Geometrische Algorithmen	5
4.1	Rotate a Point	5
4.2	Graham Scan (Convex Hull)	5
4.3	Maximum Distance in a Point Set	6
4.4	Area of a Polygon	6
4.5	Punkt in Polygon	6
4.6	Line Intersection	6
5	Textsuche	7
5.1	Knuth-Morris-Pratt	7
6	Verschiedenes	7
6.1	Potenzmenge	7
6.2	Longest Common Subsequence	7
6.3	Longest Increasing Subsequence	7
6.4	CYK-Algorithmus	8
7	Eine kleine C-Referenz	8

1 Mathematische Algorithmen

1.1 Primzahlen

Für Primzahlen gilt immer (aber nicht nur für Primzahlen)

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

Ein paar Primzahlen für den Hausgebrauch:
1000003, 2147483647 (2^{31}), 4294967291 (2^{32})

1.1.1 Sieb des Eratosthenes

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

1.1.2 Primzahlentest

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

1.2 Binomial Koeffizient

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

1.3 Modulare Arithmetik

Bedeutung der größten gemeinsamen Teiler:

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

Verwendung zu Berechnung des inversen Elements b zu a bezüglich einer Restklassengruppe n (a und n müssen teilerfremd sein):

$$ab \equiv 1 \pmod{n} \Leftrightarrow s \equiv b \pmod{n} \text{ für } 1 = \text{ggT}(a, n)$$

1.3.1 Erweiterter Euklidischer Algorithmus

```
1 static int[] eea(int a, int b) {
2     int[] dst = new int[3];
3     if (b == 0) {
4         dst[0] = a;
5         dst[1] = 1;
```

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

Zur Berechnung des Inversen von n im Restklassenring p gilt: $d = \text{eea}(p, n)$.

1.4 Matrixmultiplikation

Strassen-Algorithmus: $C = AB \quad A, B, C \in R^{2^n \times 2^n}$

$$\begin{aligned} C_{1,1} &= A_{1,1}B_{1,1} + A_{1,2}B_{2,1} \\ C_{1,2} &= A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ C_{2,1} &= A_{2,1}B_{1,1} + A_{2,2}B_{2,1} \\ C_{2,2} &= A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{aligned}$$

2 Datenstrukturen

2.1 Fenwick Tree (Binary Indexed Tree)

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

3 Graphenalgorithmen

3.1 Topologische Sortierung

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

```

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

```

3.2 Minimum Spanning Tree

3.2.1 Prim's Algorithm

```

1  #define WHITE 0
2  #define BLACK 1
3  #define INF INT_MAX
4
5  int baum( int **matrix, int N){
6      int i, sum = 0;
7
8      int color[N];
9      int dist[N];
10
11     // markiere alle Knoten ausser 0 als unbesucht
12     color[0] = BLACK;
13     for( i=1; i<N; i++){
14         color[i] = WHITE;
15         dist[i] = INF;
16     }
17
18     // berechne den Rand
19     for( i=1; i<N; i++){
20         if( dist[i] > matrix[i][nextIndex]){
21             dist[i] = matrix[i][nextIndex];
22         }
23     }
24
25     while( 1){
26         int nextDist = INF, nextIndex = -1;
27
28         /* Den naechsten Knoten waehlen */
29         for(i=0; i<N; i++){
30             if( color[i] != WHITE) continue;
31
32             if( dist[i] < nextDist){
33                 nextDist = dist[i];
34                 nextIndex = i;
35             }
36         }
37
38         /* Abbruchbedingung*/
39         if( nextIndex == -1) break;
40
41         /* Knoten in MST aufnehmen */
42         color[nextIndex] = RED;
43         sum += nextDist;
44
45         /* naechste kuerzeste Distanzen berechnen */
46         for( i=0; i<N; i++){
47             if( i == nextIndex || color[i] == BLACK )▼
48                 continue;
49
50             if( dist[i] > matrix[i][nextIndex]){
51                 dist[i] = matrix[i][nextIndex];
52             }
53         }
54
55     return sum;
56 }

```

3.2.2 Union and Find: Kruskal's Algorithm

Amortized time per operation is $O(\alpha(n))$.

```

1 // Only the tree root is stored. The edges must be ▼
   stored separately.
2 // Path compression and union by rank
3

```

```

4 int *par = (int *) malloc(n * sizeof(int));
5 int *rank = (int *) malloc(n * sizeof(int));
6
7 // Create new forest of n vertices
8 void init(int n, int *par, int *rank) {
9     int i;
10    for (i = 1; i <= n; i++) {
11        par[i] = i; // every vertex is its own root
12        rank[i] = 0;
13    }
14 }
15
16 // Union two trees which contain x and y ▼
   respectively, returns new root
17 int union(int n, int *par, int *rank, int x, int y)▼
18 {
19     y = find(n, par, y);
20     x = find(n, par, x);
21     if (rank[x] > rank[y]) return par[y] = x;
22     if (rank[x] < rank[y]) return par[x] = y;
23     rank[x]++; // rank[x] == rank[y]
24     return par[y] = x;
25 }
26
27 // Find the tree root of x
28 int find(int n, int *par, int x) {
29     // if parent is not a tree root
30     if (par[x] != par[par[x]]) par[x] = find(n, par, ▼
31         par[x]);
32     return par[x];
33 }

```

3.3 Maximaler Fluss (Ford-Fulkerson)

```

1 /* die folgende Zeile anpassen! */
2
3 #define N_MAX 30*30+30
4
5 /* hier drunter nichts anfassen! */
6 /* ----- */
7 #define SIZE_MAX (N_MAX+2)
8 #define SIZE (N+2)
9 #define QUELLE (N)
10 #define SENKE (N+1)
11 extern int capacity[SIZE_MAX][SIZE_MAX];
12 extern int N;
13
14 int maxFlow();
15 void reset();
16
17 #include <stdio.h>
18 #include <limits.h>
19 #include <string.h>
20 #include "flow.h"
21
22 #define NONE -1
23 #define INF INT_MAX/2
24
25 int N;
26 int capacity[SIZE_MAX][SIZE_MAX];
27 int flow[SIZE_MAX][SIZE_MAX];
28 int queue[SIZE_MAX], *head, *tail;
29 int state[SIZE_MAX];
30 int pred[SIZE_MAX];
31
32 enum { UNVISITED, WAITING, PROCESSED };
33
34 void enqueue( int x){
35     *tail++ = x;
36     state[x] = WAITING;
37 }
38
39 int dequeue(){
40     int x = *head++;
41     state[x] = PROCESSED;
42 }

```

```

26     return x;
27 }
28
29 void reset(){
30     int i, j;
31     for(i=0; i<SIZE;i++){
32         memset( capacity[i], 0, sizeof(int)*SIZE );
33     }
34 }
35
36 int bfs( int start, int target){
37     int u, v;
38     for( u=0; u< SIZE; u++){
39         state[u] = UNVISITED;
40     }
41     head = tail = queue;
42     pred[start] = NONE;
43
44     enqueue(start);
45
46     while( head < tail){
47         u = dequeue();
48
49         for( v= 0; v< SIZE; v++){
50             if( state[v] == UNVISITED &&
51                 capacity[u][v] - flow[u][v] > 0){
52
53                 enqueue(v);
54                 pred[v] = u;
55             }
56         }
57     }
58
59     return state[target] == PROCESSED;
60 }
61
62 int maxFlow(){
63     int max_flow = 0;
64     int u;
65
66     int i, j;
67     for(i=0; i<SIZE;i++){
68         memset( flow[i], 0, sizeof(int)*SIZE );
69     }
70
71     while( bfs( QUELLE, SENKE)){
72         int increment = INF, temp;
73
74         for( u= SENKE; pred[u] != NONE; u = pred[u])▼
75             {
76                 temp = capacity[pred[u]][u] - flow[pred[u]
77 ][u];
78                 if( temp < increment){
79                     increment = temp;
80                 }
81             }
82
83             for( u= SENKE; pred[u] != NONE; u = pred[u])▼
84                 {
85                     flow[pred[u]][u] += increment;
86                     flow[u][pred[u]] -= increment;
87                 }
88
89             max_flow += increment;
90     }
91
92     return max_flow;
93 }
94
95 /**
96  * Ford Fulkerson
97  * @param s source
98  * @param d destination
99  * @param c capacity
100  * @param f flow, init with 0
101  * @return
102 */

```

```

9 static int ff(int s, int d, int[][] c, int[][] f) {
10     List<Integer> path = dfs(s, d, c, f, new boolean[▼
11         c.length]); // find path
12     if (path.size() < 2) {
13         int flow = 0;
14         for (int i = 0; i < f[s].length; i++) { // ▼
15             leaving flow of source
16             flow += f[s][i];
17         }
18         return flow;
19     }
20     int cap = Integer.MAX_VALUE; // capacity of ▼
21     current path
22     for (int i = 0; i < path.size() - 1; i++) {
23         int a = path.get(i), b = path.get(i + 1);
24         cap = Math.min(cap, c[a][b] - f[a][b]);
25     }
26     for (int i = 0; i < path.size() - 1; i++) { //▼
27         update flow
28         int a = path.get(i), b = path.get(i + 1);
29         f[a][b] += cap;
30         f[b][a] -= cap;
31     }
32     return ff(s, d, c, f); // tail recursion
33 }
34
35 /**
36  * depth first search in flow network
37  * @param s source
38  * @param d destination
39  * @param c capacity
40  * @param f flow
41  * @param v visited, init with false
42  * @return
43 */
44 static List<Integer> dfs(int s, int d, int[][] c, ▼
45     int[][] f, boolean[] v) {
46     v[s] = true;
47     if (s == d) { // destination found
48         LinkedList<Integer> path = new LinkedList<▼
49             Integer>();
50         path.add(d);
51         return path;
52     }
53     for (int i = 0; i < c[s].length; i++) {
54         if (!v[i] && c[s][i] - f[s][i] > 0) {
55             List<Integer> path = dfs(i, d, c, f, v);
56             if (path.size() > 0) {
57                 ((LinkedList<Integer>) path).addFirst(s);
58                 return path;
59             }
60         }
61     }
62     return ((List<Integer>) Collections.EMPTY_LIST);
63 }

```

3.4 Floyd-Warshall

```

1 static int n;
2 static int[][] path = new int[n][n];
3 static int[][] next = new int[n][n];
4 static void floyd(int[][] ad) {
5     for (int i = 0; i < n; i++)
6         path[i] = Arrays.copyOf(ad[i], n);
7     for (int i = 0; i < n; i++)
8         for (int j = 0; j < n; j++)
9             for (int k = 0; k < n; k++)
10                 if (path[i][k] + path[k][j] < path[i][j]) {
11                     path[i][j] = path[i][k] + path[k][j];
12                     next[i][j] = k;
13                 }
14     // there is a negative circle iff. there is a i ▼
15     such that path[i][i] < 0

```

3.5 Dijkstra

```

1 HashMap<Integer, List<Edge>> graph = new HashMap<▼
    Integer, List<Edge>>();
2 for (int i = 0; i < n; i++) graph.put(i, new ▼
    ArrayList<Edge>());
3 int dist[] = new int[n];
4 Arrays.fill(dist, Integer.MAX_VALUE);
5 int shortest = dijkstra(source, dest, graph, dist);
6
7 static int dijkstra(int s, int d, HashMap<Integer, ▼
    List<Edge>> graph, final int[] dist) {
8     dist[s] = 0;
9     TreeSet<Integer> queue = new TreeSet<Integer>(
10         new Comparator<Integer>() {
11             public int compare(Integer o1, Integer o2) {
12                 if (dist[o1] == dist[o2]) return o1.▼
                    compareTo(o2);
13                 return ((Integer) o1).compareTo(o2);
14             } });
15     queue.add(s);
16     while (queue.size() > 0) { // || queue.first() !=▼
        d) {
17         int c = queue.pollFirst();
18         for (Edge e : graph.get(c)) {
19             if (dist[e.to] > dist[c] + e.val) {
20                 queue.remove(e.to);
21                 dist[e.to] = dist[c] + e.val;
22                 queue.add(e.to);
23             } } }
24     return dist[d];
25 }
26
27 class Edge {
28     int from, to, val;
29     public Edge(int from, int to, int val) {
30         this.from = from;
31         this.to = to;
32         this.val = val;
33     } }

```

3.6 Bellmann-Ford

Single source all paths, negative weights.

```

1 // returns true iff negative-weight cycle reachable
2 private static boolean bellmannford(Node start, int▼
    n, List<Edge> edges) {
3     start.dist = 0; // others: dist = Integer.▼
        MAX_VALUE
4     while (n-- > 0) { // number of nodes --> for all ▼
        vertices
5         for (Edge edge : edges) { // --> for all edges
6             if (edge.from.dist < Integer.MAX_VALUE
7                 && edge.from.dist + edge.w < edge.to.dist)
8                 edge.to.dist = edge.from.dist + edge.w; // ▼
                    update predecessor
9         } }
10    for (Edge edge : edges) {
11        if (edge.from.dist < Integer.MAX_VALUE
12            && edge.from.dist + edge.w < edge.to.dist)
13            return true;
14    }
15    return false;
16 }
17 class Node {}
18 class Edge {
19     Node from, to;
20     int w;
21     public Edge(Node from, Node to, int w) {
22         this.from = from; this.to = to; this.w = w;
23     }
24 }

```

3.7 Starke Zusammenhangskomponenten (Kosaraju)

```

1 #define POS(X,Y) ((X)+size*(Y))
2 #define M(X,Y) (M[POS((X),(Y))])
3
4 int *top;
5 int *color;
6
7 void Kosaraju( int *M, int size);
8 void DFS( int *M, int u, int size);
9 void RDFS( int *M, int u, int size, int colorN);
10
11 void Kosaraju( int *M, int size){
12     int i;
13     int *stack = malloc( size * sizeof(int));
14     top = stack;
15
16     for(i=0;i<size;i++)
17         color[i] = 0;
18
19     for(i=0;i<size;i++){
20         if(color[i] != 0) continue;
21
22         DFS(M,i,size);
23     }
24
25     for(i=0;i<size;i++)
26         color[i] = 0;
27
28     int colorN = 1;
29
30     while( top > stack ){
31         int v = *(--top);
32         if( color[v] != 0 ) continue;
33         RDFS( M, v, size, colorN++);
34     }
35
36     free( stack);
37 }
38
39 void DFS( int *M, int u, int size){
40     int v;
41     color[u] = 1;
42     for(v=0;v<size;v++){
43         if( M(u,v) && color[v] == 0){
44             DFS( M, v, size);
45         }
46     }
47
48     *top++ = u;
49 }
50
51 void RDFS( int *M, int u, int size, int colorN){
52     int v;
53     color[u] = colorN;
54     for(v=0;v<size;v++){
55         if( M(u,v) && color[v] == 0){
56             RDFS( M, v, size, colorN);
57         }
58     }
59 }

```

4 Geometrische Algorithmen

4.1 Rotate a Point

```

1 static P rotate(P origin, P p, double ccw) {
2     double x = (p.x - origin.x) * Math.cos(ccw) - (p.▼
        y - origin.y) Math.sin(ccw);
3     double y = (p.x - origin.x) * Math.sin(ccw) + (p.▼
        y - origin.y) Math.cos(ccw);
4     return new P(x, y);
5 }

```

4.2 Graham Scan (Convex Hull)

```

1 class P {

```

```

2 double x, y;
3
4 P(double x, double y) {
5     this.x = x;
6     this.y = y;
7 }
8 // polar coordinates (not used in graham scan)
9 double r() { return Math.sqrt(x * x + y * y); }
10 double d() { return Math.atan2(y, x); }
11 }
12
13 // turn is counter-clockwise if > 0; collinear if = 0; clockwise else
14 static double ccw(P p1, P p2, P p3) {
15     return (p2.x - p1.x) * (p3.y - p1.y) - (p2.y - p1.y) * (p3.x - p1.x);
16 }
17
18 static List<P> graham(List<P> l) {
19     if (l.size() < 3)
20         return l;
21     P temp = l.get(0);
22     for (P p : l)
23         if (temp.y > p.y || temp.y == p.y && temp.x > p.x)
24             temp = p;
25     final P start = temp; // min y (then leftmost)
26
27     Collections.sort(l, new Comparator<P>() {
28         public int compare(P o1, P o2) {
29             if (new Double(Math.atan2(o1.y - start.y, o1.x - start.x)) // same angle
30                 .compareTo(Math.atan2(o2.y - start.y, o2.x - start.x)) == 0)
31                 return new Double((o1.x - start.x) * (o1.x - start.x)
32                     + (o1.y - start.y) * (o1.y - start.y))
33                     .compareTo((o2.x - start.x) * (o2.x - start.x)
34                     + (o2.y - start.y) * (o2.y - start.y)); // use distance
35             return new Double(Math.atan2(o1.y - start.y, o1.x - start.x))
36                 .compareTo(Math.atan2(o2.y - start.y, o2.x - start.x));
37         }
38     });
39     Stack<P> s = new Stack<P>();
40     s.add(start);
41     s.add(l.get(1));
42     for (int i = 2; i < l.size(); i++) {
43         while (s.size() >= 2
44             && ccw(s.get(s.size() - 2), s.get(s.size() - 1), l.get(i)) <= 0)
45             s.pop();
46         s.push(l.get(i));
47     }
48     return s;
49 }

```

4.3 Maximum Distance in a Point Set

```

1 List<P> hull = graham(list);
2 maxDist(hull);
3
4 static double dist(P p1, P p2) {
5     return Math.sqrt((p1.x - p2.x) * (p1.x - p2.x)
6         + (p1.y - p2.y) * (p1.y - p2.y));
7 }
8
9 static double maxDist(List<P> hull) {
10     double max = 0, tmp = 0;
11     int j = 0, n = hull.size();
12     for (P p : hull) {
13         for (P q : hull) {
14             if (p == q) continue;

```

```

15         tmp = dist(p, q);
16         max = Math.max(max, tmp);
17     }
18 }
19 return max;
20 }

```

4.4 Area of a Polygon

```

1 // area of a polygon, e.g. area(gham(list))
2 static double area(List<P> l) {
3     double sum = 0;
4     // points must be in ccw order, otherwise negative area returned
5     for (int i = 0; i < l.size(); i++) {
6         sum += l.get(i).x * l.get((i + 1) % l.size()).y;
7         sum -= l.get(i).y * l.get((i + 1) % l.size()).x;
8     }
9     return sum / 2;
10 }

```

4.5 Punkt in Polygon

```

1 /**
2  * -1: A liegt links von BC (ausser unterer Endpunkt)
3  * 0: A auf BC
4  * +1: sonst
5  */
6 public static int KreuzProdTest(double ax, double ay, double bx, double by, double cx, double cy) {
7     if (ay == by && by == cy) {
8         if ((bx <= ax && ax <= cx) || (cx <= ax && ax <= bx)) return 0;
9         else return +1;
10    }
11    if (by > cy) {
12        double tmpx = bx, tmpy = by;
13        bx = cx;
14        by = cy;
15        cx = tmpx;
16        cy = tmpy;
17    }
18    if (ay == by && ax == bx) return 0;
19    if (ay <= by || ay > cy) return +1;
20    double delta = (bx - ax) * (cy - ay) - (by - ay) * (cx - ax);
21    if (delta > 0) return -1;
22    else if (delta < 0) return +1;
23    else return 0;
24 }
25
26 /**
27  * Input: P[i] (x[i],y[i]); P[0]:=P[n]
28  * -1: Q ausserhalb Polygon
29  * 0: Q auf Polygon
30  * +1: Q innerhalb des Polygons
31  */
32
33 public static int PunktInPoly(double[] x, double[] y, double qx, double qy) {
34     int t = -1;
35     for (int i = 0; i < x.length - 1; i++)
36         t = t * KreuzProdTest(qx, qy, x[i], y[i], x[i + 1], y[i + 1]);
37     return t;
38 }

```

4.6 Line Intersection

```

1 // intersection of p0-p1 and p2-p3.
2 static P intersect(P p0, P p1, P p2, P p3) {
3     double a_x, a_y, b_x, b_y, r, s, t;
4     a_x = p1.x - p0.x;
5     a_y = p1.y - p0.y;

```

```

6   b_x = p3.x - p2.x;
7   b_y = p3.y - p2.y;
8
9   r = (-b_x * a_y + a_x * b_y); // lines are ▽
      parallel if r == 0
10  s = (-a_y * (p0.x - p2.x) + a_x * (p0.y - p2.y)) ▽
      / r;
11  t = ( b_x * (p0.y - p2.y) - b_y * (p0.x - p2.x)) ▽
      / r;
12
13  // remove this condition when looking at lines ▽
      and not only segments
14  if (s >= 0 && s <= 1 && t >= 0 && t <= 1)
15      return new P(p0.x + (t * a_x), p0.y + (t * ▽
          a_y));
16
17  return null;
18 }

```

5 Textsuche

5.1 Knuth-Morris-Pratt

```

1 // Finds the first occurrence of the pattern in the ▽
   text.
2 int match(String text, String pattern, int[] jump) ▽
   {
3   int j = 0;
4   if (text.length() == 0)
5       return -1;
6   for (int i = 0; i < text.length(); i++) {
7       while (j > 0 && pattern.charAt(j) != text.charAt ▽
          (i))
8           j = jump[j - 1];
9       if (pattern.charAt(j) == text.charAt(i))
10          j++;
11       if (j == pattern.length())
12           return i - pattern.length() + 1;
13   }
14   return -1;
15 }
16
17 // Computes the jump function
18 int[] computeJump(String pattern) {
19     int[] jump = new int[pattern.length()];
20     int j = 0;
21     for (int i = 1; i < pattern.length(); i++) {
22         while (j > 0 && pattern.charAt(j) != pattern. ▽
            charAt(i))
23             j = jump[j - 1];
24         if (pattern.charAt(j) == pattern.charAt(i))
25             j++;
26         jump[i] = j;
27     }
28     return jump;
29 }

```

6 Verschiedenes

6.1 Potenzmenge

```

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

```

```

13         return temp;
14     }
15     public void remove() {}
16 };
17 }

```

6.2 Longest Common Subsequence

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5
6 int lcs( char *a, char *b){
7     int len = strlen( a);
8     int lenb =strlen(b);
9
10    int *zeile = malloc( (len+1) * sizeof(int)), * ▽
        temp,
11        *neue = malloc( (len+1) * sizeof(int)), i, j ▽
        ;
12
13    for(i=0; i<len+1; i++){
14        zeile[i] = neue[i] = 0;
15    }
16
17    for(j=0; j<lenb; j++){
18        for(i=0; i<len; i++){
19            if( a[i] == b[j]){
20                neue[i+1] = zeile[i] + 1;
21            } else {
22                neue[i+1] = neue[i] > zeile[i+1] ? ▽
                    neue[i] : zeile[i+1];
23            }
24        }
25        temp = zeile;
26        zeile = neue;
27        neue = temp;
28    }
29
30    int res = zeile[len];
31    free( zeile);
32    free( neue);
33    return res;
34 }

```

6.3 Longest Increasing Subsequence

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int lis( int *list, int n){
5     int *sorted = malloc( n*sizeof(int)), sorted_n;
6     int i, *lower, *upper, *mid, *pos;
7
8     if( n == 0) return 0;
9
10    sorted[0] = list[0];
11    sorted_n = 1;
12
13    for( i=1; i<n; i++){
14        /* binaere Suche */
15        lower = list;
16        upper = list + sorted_n;
17        mid = list + sorted_n / 2;
18
19        while( lower < upper-1){
20            if( list[i] < *mid){
21                upper = mid;
22            } else {
23                lower = mid;
24            }
25
26            mid = lower + (upper-lower) / 2;
27        }
28    }

```

```

29
30  if( mid == list + sorted_n -1 && *mid < list[i]){
31      *mid = list[i];
32      sorted_n++;
33  }
34
35      if( list[i] < *mid){
36          *mid = list[i];
37      }
38  }
39
40  free( sorted);
41
42  return sorted_n;
43 }

```

6.4 CYK-Algorithmus

```

1 let the input be a string S consisting of n ▼
  characters: a1 ... an.
2 let the grammar contain r nonterminal symbols R1 ▼
  ... Rr.

```

```

3 This grammar contains the subset Rs which is the ▼
  set of start symbols.
4 let P[n,n,r] be an array of booleans. Initialize ▼
  all elements of P to false.
5 for each i = 1 to n
6   for each unit production Rj -> ai
7     set P[i,1,j] = true
8 for each i = 2 to n -- Length of span
9   for each j = 1 to n-i+1 -- Start of span
10    for each k = 1 to i-1 -- Partition of span
11      for each production RA -> RB RC
12        if P[j,k,B] and P[j+k,i-k,C] then set P[j,i,▼
          A] = true
13 if any of P[1,n,x] is true (x is iterated over the ▼
  set s, where s are all the indices for Rs) ▼
  then
14   S is member of language
15 else
16   S is not member of language

```

7 Eine kleine C-Referenz

C Reference Card (ANSI)

Program Structure/Functions

```
type func(type1,...)
type name
main() {
    declarations
    statements
}
type func(arg1,...) {
    declarations
    statements
    return value;
}
/* */
main(int argc, char *argv[])
exit(arg)
```

C Preprocessor

```
#include <filename>
#include "filename"
#define name text
#define name(var) text
Example. #define max(A,B) ((A)>(B) ? (A) : (B))
#undef name
#
#
concatenate args and rescan
conditional execution
is name defined, not defined?
name defined?
line continuation char
\
```

Data Types/Declarations

```
character (1 byte)
integer
float (single precision)
float (double precision)
short (16 bit integer)
long (32 bit integer)
positive and negative
only positive
pointer to int, float,...
enumeration constant
constant (unchanging) value
declare external variable
register variable
local to source file
no value
structure
create name by data type
size of an object (type is size_t)
size of a data type (type is size_t)
```

Initialization

```
initialize variable
initialize array
initialize char string
type name=value
type name[]={value1,...}
char name[]="string"
```

Constants

```
long (suffix)
float (suffix)
exponential form
octal (prefix zero)
hexadecimal (prefix zero-ex)
character constant (char, octal, hex)
newline, cr, tab, backspace
special characters
string constant (ends with '\0')
```

Pointers, Arrays & Structures

```
declare pointer to type
declare function returning pointer to type type *f()
declare pointer to function returning type type (*pf)()
generic pointer type
void *
NULL
object pointed to by pointer
address of object name
array
multi-dim array
name[dim1][dim2]...
 Structures
struct tag {
    declarations
};
create structure
member of structure from template
member of pointed to structure
Example. (*p).x and p->x are the same
single value, multiple type structure
union
member : b
```

Operators (grouped by precedence)

structure member operator	<i>name.member</i>
structure pointer	<i>pointer->member</i>
increment, decrement	++, --
plus, minus, logical not, bitwise not	+, -, !, ~
indirection via pointer, address of object	* <i>pointer</i> , & <i>name</i>
cast expression to type	(<i>type</i>) <i>expr</i>
size of an object	sizeof
multiply, divide, modulus (remainder)	*, /, %
add, subtract	+, -
left, right shift [bit ops]	<<, >>
comparisons	>, >=, <, <=
comparisons	==, !=
bitwise and	&
bitwise exclusive or	^
bitwise or (incl)	
logical and	&&
logical or	
conditional expression	<i>expr</i> ₁ ? <i>expr</i> ₂ : <i>expr</i> ₃
assignment operators	+=, -=, *=, ...
expression evaluation separator	,

Unary operators, conditional expression and assignment operators group right to left; all others group left to right.

Flow of Control

```
statement terminator
block delimiters
exit from switch, while, do, for
next iteration of while, do, for
goto label
label:
return expr
Flow Constructions
if statement
if (expr) statement
else if (expr) statement
else statement
while (expr)
    statement
for (expr1; expr2; expr3)
    statement
do statement
while(expr);
switch statement
    switch (expr) {
        case const1: statement1 break;
        case const2: statement2 break;
        default: statement
    }
```

ANSI Standard Libraries

```
<assert.h> <ctype.h> <errno.h> <float.h> <limits.h>
<locale.h> <math.h> <setjmp.h> <signal.h> <stdarg.h>
<stddef.h> <stdio.h> <stdlib.h> <string.h> <time.h>
```

Character Class Tests <ctype.h>

```
alphanumeric?
alphabetic?
control character?
decimal digit?
lower case letter?
printing character (not incl space)?
printing character (incl space)?
printing char except space, letter, digit?
space, formfeed, newline, cr, tab, vtab?
upper case letter?
hexadecimal digit?
convert to lower case?
convert to upper case?
toupper(c)
```

String Operations <string.h>

```
s, t are strings, cs, ct are constant strings
length of s
strcpy(s, ct)
strncpy(s, ct, n)
strcat(s, ct)
strncat(s, ct, n)
strcmp(cs, ct)
strncmp(cs, ct, n)
strchr(cs, c)
strrchr(cs, c)
memcpy(s, ct, n)
memmove(s, ct, n)
memcmp(cs, ct, n)
memchr(cs, c, n)
memset(s, c, n)
```

C Reference Card (ANSI)

Input/Output <stdio.h>

Standard I/O

standard input stream
standard output stream
standard error stream
end of file
get a character
print a character
print formatted data
print to string *s*
read formatted data
read from string *s*
read line to string *s* (< max chars)
print string *s*
File I/O
declare file pointer
pointer to named file
modes: *r* (read), *w* (write), *a* (append)
get a character
write a character
write to file
read from file
close file
non-zero if error
non-zero if EOF
read line to string *s* (< max chars)
write string *s*
Codes for Formatted I/O: "%-+ 0w.pmic"
- left justify
+ print with sign
space print space if no sign
0 pad with leading zeros
w min field width
p precision
m conversion character:
 h short, *l* long, *L* long double
c conversion character:
 d,i integer
 u unsigned
 s char string
 e,E exponential
 o octal
 x,X hexadecimal
 p pointer
 n number of chars written
g,G same as *f* or *e,E* depending on exponent

Variable Argument Lists <stdarg.h>

declaration of pointer to arguments *va_list name*;
initialization of argument pointer *va_start(name, lastarg)*
lastarg is last named parameter of the function
access next unnamed arg, update pointer *va_arg(name, type)*
call before exiting function *va_end(name)*

4

Standard Utility Functions <stdlib.h>

absolute value of int *n*
absolute value of long *n*
quotient and remainder of ints *n,d*
return structure with *div_t.quot* and *div_t.rem*
quotient and remainder of longs *n,d*
returns structure with *ldiv_t.quot* and *ldiv_t.rem*
pseudo-random integer [0, RAND_MAX]
rand()
set random seed to *n*
terminate program execution
pass string *s* to system for execution
Conversions
convert string *s* to double
convert string *s* to integer
convert string *s* to long
convert string *s* to long
convert prefix of *s* to double
convert prefix of *s* (base *b*) to long
same, but unsigned long
strtoul(*s*, endp, *b*)

Storage Allocation

allocate storage
change size of object
deallocate space
Array Functions
search array for key
sort array ascending order
malloc(*size*), calloc(*nobj*, *size*)
realloc(*pts*, *size*)
free(*ptr*)

Time and Date Functions <time.h>

processor time used by program
Example: clock()/CLOCKS_PER_SEC is time in seconds
current calendar time
time2-time1 in seconds (double)
arithmetic types representing times
structure type for calendar time comps
tm_sec seconds after minute
tm_min minutes after hour
tm_hour hours since midnight
tm_mday day of month
tm_mon months since January
tm_year years since 1900
tm_wday days since Sunday
tm_yday days since January 1
tm_isdst Daylight Savings Time flag
convert local time to calendar time
convert time in *tp* to string
convert calendar time in *tp* to local time
convert calendar time to GMT
convert calendar time to local time
format date and time info
tp is a pointer to a structure of type tm

Mathematical Functions <math.h>

Arguments and returned values are double
trig functions
inverse trig functions
atan(*y/x*)
hyperbolic trig functions
exponentials & logs
exponentials & logs (2 power)
division & remainder
powers
rounding

5

Integer Type Limits <limits.h>

The numbers given in parentheses are typical values for the constants on a 32-bit Unix system.

CHAR_BIT bits in char (8)
CHAR_MAX max value of char (127 or 255)
CHAR_MIN min value of char (-128 or 0)
INT_MAX max value of int (+32,767)
INT_MIN min value of int (-32,768)
LONG_MAX max value of long (+2,147,483,647)
LONG_MIN min value of long (-2,147,483,648)
SHAR_MAX max value of signed char (+127)
SHAR_MIN min value of signed char (-128)
SHRT_MAX max value of short (+32,767)
SHRT_MIN min value of short (-32,768)
UCHAR_MAX max value of unsigned char (255)
UINT_MAX max value of unsigned int (65,535)
ULONG_MAX max value of unsigned long (4,294,967,295)
USHRT_MAX max value of unsigned short (65,536)

Float Type Limits <float.h>

FLT_RADIX radix of exponent rep (2)
FLT_ROUNDS floating point rounding mode (6)
FLT_DIG decimal digits of precision (10-5)
FLT_EPSILON smallest *x* so $1.0 + x \neq 1.0$
FLT_MANT_DIG number of digits in mantissa (10³⁷)
FLT_MAX maximum floating point number (10³⁷)
FLT_MAX_EXP maximum exponent (10-37)
FLT_MIN minimum floating point number (10-37)
FLT_MIN_EXP minimum exponent (10)
DBL_DIG decimal digits of precision (10-9)
DBL_EPSILON smallest *x* so $1.0 + x \neq 1.0$
DBL_MANT_DIG number of digits in mantissa (10³⁷)
DBL_MAX max double floating point number (10³⁷)
DBL_MAX_EXP maximum exponent (10-37)
DBL_MIN min double floating point number (10-37)
DBL_MIN_EXP minimum exponent

May 1999 v1.3. Copyright © 1999 Joseph H. Silverman

Permission is granted to make and distribute copies of this card provided the copyright notice and this permission notice are preserved on all copies.

Send comments and corrections to J.H. Silverman, Math. Dept., Brown Univ., Providence, RI 02912 USA. (jhs@math.brown.edu)

6

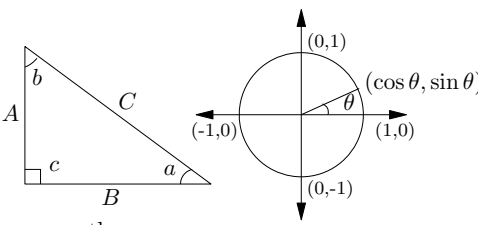
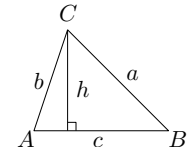
Theoretical Computer Science Cheat Sheet

Definitions		Series	
$f(n) = O(g(n))$	iff \exists positive c, n_0 such that $0 \leq f(n) \leq cg(n) \forall n \geq n_0$.	$\sum_{i=1}^n i = \frac{n(n+1)}{2}, \quad \sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^n i^3 = \frac{n^2(n+1)^2}{4}.$	
$f(n) = \Omega(g(n))$	iff \exists positive c, n_0 such that $f(n) \geq cg(n) \geq 0 \forall n \geq n_0$.	In general:	
$f(n) = \Theta(g(n))$	iff $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.	$\sum_{i=1}^n i^m = \frac{1}{m+1} \left[(n+1)^{m+1} - 1 - \sum_{i=1}^n ((i+1)^{m+1} - i^{m+1} - (m+1)i^m) \right]$	
$f(n) = o(g(n))$	iff $\lim_{n \rightarrow \infty} f(n)/g(n) = 0$.	$\sum_{i=1}^{n-1} i^m = \frac{1}{m+1} \sum_{k=0}^m \binom{m+1}{k} B_k n^{m+1-k}.$	
$\lim_{n \rightarrow \infty} a_n = a$	iff $\forall \epsilon > 0, \exists n_0$ such that $ a_n - a < \epsilon, \forall n \geq n_0$.	Geometric series:	
$\sup S$	least $b \in \mathbb{R}$ such that $b \geq s, \forall s \in S$.	$\sum_{i=0}^n c^i = \frac{c^{n+1} - 1}{c - 1}, \quad c \neq 1, \quad \sum_{i=0}^{\infty} c^i = \frac{1}{1 - c}, \quad \sum_{i=1}^{\infty} c^i = \frac{c}{1 - c}, \quad c < 1,$	
$\inf S$	greatest $b \in \mathbb{R}$ such that $b \leq s, \forall s \in S$.	$\sum_{i=0}^n ic^i = \frac{nc^{n+2} - (n+1)c^{n+1} + c}{(c-1)^2}, \quad c \neq 1, \quad \sum_{i=0}^{\infty} ic^i = \frac{c}{(1-c)^2}, \quad c < 1.$	
$\liminf_{n \rightarrow \infty} a_n$	$\lim_{n \rightarrow \infty} \inf \{a_i \mid i \geq n, i \in \mathbb{N}\}.$	Harmonic series:	
$\limsup_{n \rightarrow \infty} a_n$	$\lim_{n \rightarrow \infty} \sup \{a_i \mid i \geq n, i \in \mathbb{N}\}.$	$H_n = \sum_{i=1}^n \frac{1}{i}, \quad \sum_{i=1}^n iH_i = \frac{n(n+1)}{2} H_n - \frac{n(n-1)}{4}.$	
$\binom{n}{k}$	Combinations: Size k sub-sets of a size n set.	$\sum_{i=1}^n H_i = (n+1)H_n - n, \quad \sum_{i=1}^n \binom{i}{m} H_i = \binom{n+1}{m+1} \left(H_{n+1} - \frac{1}{m+1} \right).$	
$[n_k]$	Stirling numbers (1st kind): Arrangements of an n element set into k cycles.	1. $\binom{n}{k} = \frac{n!}{(n-k)!k!}, \quad 2. \sum_{k=0}^n \binom{n}{k} = 2^n, \quad 3. \binom{n}{k} = \binom{n}{n-k},$	
$\{n_k\}$	Stirling numbers (2nd kind): Partitions of an n element set into k non-empty sets.	4. $\binom{n}{k} = \frac{n}{k} \binom{n-1}{k-1}, \quad 5. \binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1},$	
$\langle n_k \rangle$	1st order Eulerian numbers: Permutations $\pi_1 \pi_2 \dots \pi_n$ on $\{1, 2, \dots, n\}$ with k ascents.	6. $\binom{n}{m} \binom{m}{k} = \binom{n}{k} \binom{n-k}{m-k}, \quad 7. \sum_{k=0}^n \binom{r+k}{k} = \binom{r+n+1}{n},$	
$\langle\langle n_k \rangle\rangle$	2nd order Eulerian numbers.	8. $\sum_{k=0}^n \binom{k}{m} = \binom{n+1}{m+1}, \quad 9. \sum_{k=0}^n \binom{r}{k} \binom{s}{n-k} = \binom{r+s}{n},$	
C_n	Catalan Numbers: Binary trees with $n+1$ vertices.	10. $\binom{n}{k} = (-1)^k \binom{k-n-1}{k}, \quad 11. \left\{ \begin{matrix} n \\ 1 \end{matrix} \right\} = \left\{ \begin{matrix} n \\ n \end{matrix} \right\} = 1,$	
14. $\begin{bmatrix} n \\ 1 \end{bmatrix} = (n-1)!,$	15. $\begin{bmatrix} n \\ 2 \end{bmatrix} = (n-1)!H_{n-1},$	16. $\begin{bmatrix} n \\ n \end{bmatrix} = 1,$	17. $\begin{bmatrix} n \\ k \end{bmatrix} \geq \left\{ \begin{matrix} n \\ k \end{matrix} \right\},$
18. $\begin{bmatrix} n \\ k \end{bmatrix} = (n-1) \begin{bmatrix} n-1 \\ k \end{bmatrix} + \begin{bmatrix} n-1 \\ k-1 \end{bmatrix},$	19. $\left\{ \begin{matrix} n \\ n-1 \end{matrix} \right\} = \begin{bmatrix} n \\ n-1 \end{bmatrix} = \binom{n}{2},$	20. $\sum_{k=0}^n \begin{bmatrix} n \\ k \end{bmatrix} = n!,$	21. $C_n = \frac{1}{n+1} \binom{2n}{n},$
22. $\langle n_0 \rangle = \langle n_{n-1} \rangle = 1,$	23. $\langle n_k \rangle = \langle n_{n-1-k} \rangle,$	24. $\langle n_k \rangle = (k+1) \langle n_{k-1} \rangle + (n-k) \langle n_{k-1} \rangle,$	
25. $\langle k_0 \rangle = \begin{cases} 1 & \text{if } k=0, \\ 0 & \text{otherwise} \end{cases}$	26. $\langle n_1 \rangle = 2^n - n - 1,$	27. $\langle n_2 \rangle = 3^n - (n+1)2^n + \binom{n+1}{2},$	
28. $x^n = \sum_{k=0}^n \langle n_k \rangle \binom{x+k}{n},$	29. $\langle n_m \rangle = \sum_{k=0}^m \binom{n+1}{k} (m+1-k)^n (-1)^k,$	30. $m! \left\{ \begin{matrix} n \\ m \end{matrix} \right\} = \sum_{k=0}^n \langle n_k \rangle \binom{k}{n-m},$	
31. $\langle n_m \rangle = \sum_{k=0}^n \left\{ \begin{matrix} n \\ k \end{matrix} \right\} \binom{n-k}{m} (-1)^{n-k-m} k!,$	32. $\langle\langle n_0 \rangle\rangle = 1,$	33. $\langle\langle n_n \rangle\rangle = 0 \quad \text{for } n \neq 0,$	
34. $\langle\langle n_k \rangle\rangle = (k+1) \langle\langle n_{k-1} \rangle\rangle + (2n-1-k) \langle\langle n_{k-1} \rangle\rangle,$	35. $\sum_{k=0}^n \langle\langle n_k \rangle\rangle = \frac{(2n)^n}{2^n},$		
36. $\left\{ \begin{matrix} x \\ x-n \end{matrix} \right\} = \sum_{k=0}^n \langle\langle n_k \rangle\rangle \binom{x+n-1-k}{2n},$	37. $\left\{ \begin{matrix} n+1 \\ m+1 \end{matrix} \right\} = \sum_k \binom{n}{k} \left\{ \begin{matrix} k \\ m \end{matrix} \right\} = \sum_{k=0}^n \left\{ \begin{matrix} k \\ m \end{matrix} \right\} (m+1)^{n-k},$		

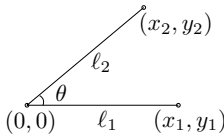
Theoretical Computer Science Cheat Sheet

Identities Cont.		Trees
<p>38. $\begin{bmatrix} n+1 \\ m+1 \end{bmatrix} = \sum_k \begin{bmatrix} n \\ k \end{bmatrix} \begin{bmatrix} k \\ m \end{bmatrix} = \sum_{k=0}^n \begin{bmatrix} k \\ m \end{bmatrix} n^{\overline{n-k}} = n! \sum_{k=0}^n \frac{1}{k!} \begin{bmatrix} k \\ m \end{bmatrix},$</p> <p>40. $\left\{ \begin{matrix} n \\ m \end{matrix} \right\} = \sum_k \binom{n}{k} \left\{ \begin{matrix} k+1 \\ m+1 \end{matrix} \right\} (-1)^{n-k},$</p> <p>42. $\left\{ \begin{matrix} m+n+1 \\ m \end{matrix} \right\} = \sum_{k=0}^m k \left\{ \begin{matrix} n+k \\ k \end{matrix} \right\},$</p> <p>44. $\binom{n}{m} = \sum_k \left\{ \begin{matrix} n+1 \\ k+1 \end{matrix} \right\} \begin{bmatrix} k \\ m \end{bmatrix} (-1)^{m-k},$</p> <p>46. $\left\{ \begin{matrix} n \\ n-m \end{matrix} \right\} = \sum_k \binom{m-n}{m+k} \binom{m+n}{n+k} \begin{bmatrix} m+k \\ k \end{bmatrix},$</p> <p>48. $\left\{ \begin{matrix} n \\ \ell+m \end{matrix} \right\} \binom{\ell+m}{\ell} = \sum_k \left\{ \begin{matrix} k \\ \ell \end{matrix} \right\} \left\{ \begin{matrix} n-k \\ m \end{matrix} \right\} \binom{n}{k},$</p>	<p>39. $\begin{bmatrix} x \\ x-n \end{bmatrix} = \sum_{k=0}^n \left\langle \begin{matrix} n \\ k \end{matrix} \right\rangle \begin{bmatrix} x+k \\ 2n \end{bmatrix},$</p> <p>41. $\begin{bmatrix} n \\ m \end{bmatrix} = \sum_k \begin{bmatrix} n+1 \\ k+1 \end{bmatrix} \binom{k}{m} (-1)^{m-k},$</p> <p>43. $\begin{bmatrix} m+n+1 \\ m \end{bmatrix} = \sum_{k=0}^m k(n+k) \begin{bmatrix} n+k \\ k \end{bmatrix},$</p> <p>45. $(n-m)! \binom{n}{m} = \sum_k \begin{bmatrix} n+1 \\ k+1 \end{bmatrix} \left\{ \begin{matrix} k \\ m \end{matrix} \right\} (-1)^{m-k}, \text{ for } n \geq m,$</p> <p>47. $\begin{bmatrix} n \\ n-m \end{bmatrix} = \sum_k \binom{m-n}{m+k} \binom{m+n}{n+k} \left\{ \begin{matrix} m+k \\ k \end{matrix} \right\},$</p> <p>49. $\begin{bmatrix} n \\ \ell+m \end{bmatrix} \binom{\ell+m}{\ell} = \sum_k \begin{bmatrix} k \\ \ell \end{bmatrix} \begin{bmatrix} n-k \\ m \end{bmatrix} \binom{n}{k}.$</p>	<p>Every tree with n vertices has $n-1$ edges.</p> <p>Kraft inequality: If the depths of the leaves of a binary tree are d_1, \dots, d_n:</p> $\sum_{i=1}^n 2^{-d_i} \leq 1,$ <p>and equality holds only if every internal node has 2 sons.</p>
Recurrences		
<p>Master method:</p> $T(n) = aT(n/b) + f(n), \quad a \geq 1, b > 1$ <p>If $\exists \epsilon > 0$ such that $f(n) = O(n^{\log_b a - \epsilon})$ then</p> $T(n) = \Theta(n^{\log_b a}).$ <p>If $f(n) = \Theta(n^{\log_b a})$ then</p> $T(n) = \Theta(n^{\log_b a} \log_2 n).$ <p>If $\exists \epsilon > 0$ such that $f(n) = \Omega(n^{\log_b a + \epsilon})$, and $\exists c < 1$ such that $af(n/b) \leq cf(n)$ for large n, then</p> $T(n) = \Theta(f(n)).$ <p>Substitution (example): Consider the following recurrence</p> $T_{i+1} = 2^{2^i} \cdot T_i^2, \quad T_1 = 2.$ <p>Note that T_i is always a power of two. Let $t_i = \log_2 T_i$. Then we have</p> $t_{i+1} = 2^i + 2t_i, \quad t_1 = 1.$ <p>Let $u_i = t_i/2^i$. Dividing both sides of the previous equation by 2^{i+1} we get</p> $\frac{t_{i+1}}{2^{i+1}} = \frac{2^i}{2^{i+1}} + \frac{t_i}{2^i}.$ <p>Substituting we find</p> $u_{i+1} = \frac{1}{2} + u_i, \quad u_1 = \frac{1}{2},$ <p>which is simply $u_i = i/2$. So we find that T_i has the closed form $T_i = 2^{i2^{i-1}}$.</p> <p>Summing factors (example): Consider the following recurrence</p> $T(n) = 3T(n/2) + n, \quad T(1) = 1.$ <p>Rewrite so that all terms involving T are on the left side</p> $T(n) - 3T(n/2) = n.$ <p>Now expand the recurrence, and choose a factor which makes the left side “telescope”</p>	<p>1($T(n) - 3T(n/2) = n$)</p> $3(T(n/2) - 3T(n/4) = n/2)$ \vdots $3^{\log_2 n - 1} (T(2) - 3T(1) = 2)$ <p>Let $m = \log_2 n$. Summing the left side we get $T(n) - 3^m T(1) = T(n) - 3^m = T(n) - n^k$ where $k = \log_2 3 \approx 1.58496$. Summing the right side we get</p> $\sum_{i=0}^{m-1} \frac{n}{2^i} 3^i = n \sum_{i=0}^{m-1} \left(\frac{3}{2}\right)^i.$ <p>Let $c = \frac{3}{2}$. Then we have</p> $n \sum_{i=0}^{m-1} c^i = n \left(\frac{c^m - 1}{c - 1} \right)$ $= 2n(c^{\log_2 n} - 1)$ $= 2n(c^{(\log_2 n) \log_2 c} - 1)$ $= 2n^k - 2n,$ <p>and so $T(n) = 3n^k - 2n$. Full history recurrences can often be changed to limited history ones (example): Consider</p> $T_i = 1 + \sum_{j=0}^{i-1} T_j, \quad T_0 = 1.$ <p>Note that</p> $T_{i+1} = 1 + \sum_{j=0}^i T_j.$ <p>Subtracting we find</p> $T_{i+1} - T_i = 1 + \sum_{j=0}^i T_j - 1 - \sum_{j=0}^{i-1} T_j$ $= T_i.$ <p>And so $T_{i+1} = 2T_i = 2^{i+1}$.</p>	<p>Generating functions:</p> <ol style="list-style-type: none"> 1. Multiply both sides of the equation by x^i. 2. Sum both sides over all i for which the equation is valid. 3. Choose a generating function $G(x)$. Usually $G(x) = \sum_{i=0}^{\infty} x^i g_i$. 3. Rewrite the equation in terms of the generating function $G(x)$. 4. Solve for $G(x)$. 5. The coefficient of x^i in $G(x)$ is g_i. <p>Example:</p> $g_{i+1} = 2g_i + 1, \quad g_0 = 0.$ <p>Multiply and sum:</p> $\sum_{i \geq 0} g_{i+1} x^i = \sum_{i \geq 0} 2g_i x^i + \sum_{i \geq 0} x^i.$ <p>We choose $G(x) = \sum_{i \geq 0} x^i g_i$. Rewrite in terms of $G(x)$:</p> $\frac{G(x) - g_0}{x} = 2G(x) + \sum_{i \geq 0} x^i.$ <p>Simplify:</p> $\frac{G(x)}{x} = 2G(x) + \frac{1}{1-x}.$ <p>Solve for $G(x)$:</p> $G(x) = \frac{x}{(1-x)(1-2x)}.$ <p>Expand this using partial fractions:</p> $G(x) = x \left(\frac{2}{1-2x} - \frac{1}{1-x} \right)$ $= x \left(2 \sum_{i \geq 0} 2^i x^i - \sum_{i \geq 0} x^i \right)$ $= \sum_{i \geq 0} (2^{i+1} - 1) x^{i+1}.$ <p>So $g_i = 2^i - 1$.</p>

Theoretical Computer Science Cheat Sheet				
$\pi \approx 3.14159,$ $e \approx 2.71828,$ $\gamma \approx 0.57721,$ $\phi = \frac{1+\sqrt{5}}{2} \approx 1.61803,$ $\hat{\phi} = \frac{1-\sqrt{5}}{2} \approx -0.61803$				
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		if X and Y are independent.
22	4,194,304	79		$E[X + Y] = E[X] + E[Y],$
23	8,388,608	83		$E[cX] = cE[X].$
24	16,777,216	89	Binomial distribution:	Bayes' theorem:
25	33,554,432	97	$\Pr[X = k] = \binom{n}{k} p^k q^{n-k}, \quad q = 1 - p,$	$\Pr[A_i B] = \frac{\Pr[B A_i] \Pr[A_i]}{\sum_{j=1}^n \Pr[A_j] \Pr[B A_j]}.$
26	67,108,864	101	$E[X] = \sum_{k=1}^n k \binom{n}{k} p^k q^{n-k} = np.$	Inclusion-exclusion:
27	134,217,728	103	Poisson distribution:	$\Pr\left[\bigvee_{i=1}^n X_i\right] = \sum_{i=1}^n \Pr[X_i] +$
28	268,435,456	107	$\Pr[X = k] = \frac{e^{-\lambda} \lambda^k}{k!}, \quad E[X] = \lambda.$	$\sum_{k=2}^n (-1)^{k+1} \sum_{i_1 < \dots < i_k} \Pr\left[\bigwedge_{j=1}^k X_{i_j}\right].$
29	536,870,912	109	Normal (Gaussian) distribution:	Moment inequalities:
30	1,073,741,824	113	$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}, \quad E[X] = \mu.$	$\Pr[X \geq \lambda E[X]] \leq \frac{1}{\lambda},$
31	2,147,483,648	127	The "coupon collector": We are given a random coupon each day, and there are n different types of coupons. The distribution of coupons is uniform. The expected number of days to pass before we to collect all n types is	$\Pr\left[X - E[X] \geq \lambda \cdot \sigma\right] \leq \frac{1}{\lambda^2}.$
32	4,294,967,296	131	$nH_n.$	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: $\sin a = A/C, \quad \cos a = B/C,$ $\csc a = C/A, \quad \sec a = C/B,$ $\tan a = \frac{\sin a}{\cos a} = \frac{A}{B}, \quad \cot a = \frac{\cos a}{\sin a} = \frac{B}{A}.$</p> <p>Area, radius of inscribed circle: $\frac{1}{2}AB, \quad \frac{AB}{A+B+C}.$</p> <p>Identities: $\sin x = \frac{1}{\csc x}, \quad \cos x = \frac{1}{\sec x},$ $\tan x = \frac{1}{\cot x}, \quad \sin^2 x + \cos^2 x = 1,$ $1 + \tan^2 x = \sec^2 x, \quad 1 + \cot^2 x = \csc^2 x,$ $\sin x = \cos\left(\frac{\pi}{2} - x\right), \quad \sin x = \sin(\pi - x),$ $\cos x = -\cos(\pi - x), \quad \tan x = \cot\left(\frac{\pi}{2} - x\right),$ $\cot x = -\cot(\pi - x), \quad \csc x = \cot \frac{\pi}{2} - \cot x,$ $\sin(x \pm y) = \sin x \cos y \pm \cos x \sin y,$ $\cos(x \pm y) = \cos x \cos y \mp \sin x \sin y,$ $\tan(x \pm y) = \frac{\tan x \pm \tan y}{1 \mp \tan x \tan y},$ $\cot(x \pm y) = \frac{\cot x \cot y \mp 1}{\cot x \pm \cot y},$ $\sin 2x = 2 \sin x \cos x, \quad \sin 2x = \frac{2 \tan x}{1 + \tan^2 x},$ $\cos 2x = \cos^2 x - \sin^2 x, \quad \cos 2x = 2 \cos^2 x - 1,$ $\cos 2x = 1 - 2 \sin^2 x, \quad \cos 2x = \frac{1 - \tan^2 x}{1 + \tan^2 x},$ $\tan 2x = \frac{2 \tan x}{1 - \tan^2 x}, \quad \cot 2x = \frac{\cot^2 x - 1}{2 \cot x},$ $\sin(x + y) \sin(x - y) = \sin^2 x - \sin^2 y,$ $\cos(x + y) \cos(x - y) = \cos^2 x - \sin^2 y.$</p> <p>Euler's equation: $e^{ix} = \cos x + i \sin x, \quad e^{i\pi} = -1.$</p>	<p>Multiplication: $C = A \cdot B, \quad c_{i,j} = \sum_{k=1}^n a_{i,k} b_{k,j}.$</p> <p>Determinants: $\det A \neq 0$ iff A is non-singular. $\det A \cdot B = \det A \cdot \det B,$ $\det A = \sum_{\pi} \prod_{i=1}^n \text{sign}(\pi) a_{i,\pi(i)}.$</p> <p>$2 \times 2$ and 3×3 determinant: $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc,$ $\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = g \begin{vmatrix} b & c \\ e & f \end{vmatrix} - h \begin{vmatrix} a & c \\ d & f \end{vmatrix} + i \begin{vmatrix} a & b \\ d & e \end{vmatrix}$ $= aei + bfg + cdh - ceg - fha - ibd.$</p> <p>Permanents: $\text{perm } A = \sum_{\pi} \prod_{i=1}^n a_{i,\pi(i)}.$</p>	<div></div> <p>Law of cosines: $c^2 = a^2 + b^2 - 2ab \cos C.$</p> <p>Area: $A = \frac{1}{2}hc,$ $= \frac{1}{2}ab \sin C,$ $= \frac{c^2 \sin A \sin B}{2 \sin C}.$</p> <p>Heron's formula: $A = \sqrt{s \cdot s_a \cdot s_b \cdot s_c},$ $s = \frac{1}{2}(a + b + c),$ $s_a = s - a,$ $s_b = s - b,$ $s_c = s - c.$</p> <p>More identities: $\sin \frac{x}{2} = \sqrt{\frac{1 - \cos x}{2}},$ $\cos \frac{x}{2} = \sqrt{\frac{1 + \cos x}{2}},$ $\tan \frac{x}{2} = \sqrt{\frac{1 - \cos x}{1 + \cos x}},$ $= \frac{1 - \cos x}{\sin x},$ $= \frac{\sin x}{1 + \cos x},$ $\cot \frac{x}{2} = \sqrt{\frac{1 + \cos x}{1 - \cos x}},$ $= \frac{1 + \cos x}{\sin x},$ $= \frac{\sin x}{1 - \cos x},$ $\sin x = \frac{e^{ix} - e^{-ix}}{2i},$ $\cos x = \frac{e^{ix} + e^{-ix}}{2},$ $\tan x = -i \frac{e^{ix} - e^{-ix}}{e^{ix} + e^{-ix}},$ $= -i \frac{e^{2ix} - 1}{e^{2ix} + 1},$ $\sin x = \frac{\sinh ix}{i},$ $\cos x = \cosh ix,$ $\tan x = \frac{\tanh ix}{i}.$</p>																								
Hyperbolic Functions																										
<p>Definitions: $\sinh x = \frac{e^x - e^{-x}}{2}, \quad \cosh x = \frac{e^x + e^{-x}}{2},$ $\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad \text{csch } x = \frac{1}{\sinh x},$ $\text{sech } x = \frac{1}{\cosh x}, \quad \coth x = \frac{1}{\tanh x}.$</p> <p>Identities: $\cosh^2 x - \sinh^2 x = 1, \quad \tanh^2 x + \text{sech}^2 x = 1,$ $\coth^2 x - \text{csch}^2 x = 1, \quad \sinh(-x) = -\sinh x,$ $\cosh(-x) = \cosh x, \quad \tanh(-x) = -\tanh x,$ $\sinh(x + y) = \sinh x \cosh y + \cosh x \sinh y,$ $\cosh(x + y) = \cosh x \cosh y + \sinh x \sinh y,$ $\sinh 2x = 2 \sinh x \cosh x,$ $\cosh 2x = \cosh^2 x + \sinh^2 x,$ $\cosh x + \sinh x = e^x, \quad \cosh x - \sinh x = e^{-x},$ $(\cosh x + \sinh x)^n = \cosh nx + \sinh nx, \quad n \in \mathbb{Z},$ $2 \sinh^2 \frac{x}{2} = \cosh x - 1, \quad 2 \cosh^2 \frac{x}{2} = \cosh x + 1.$</p>																										
<table><tr><th>θ</th><th>$\sin \theta$</th><th>$\cos \theta$</th><th>$\tan \theta$</th></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td></tr><tr><td>$\frac{\pi}{6}$</td><td>$\frac{1}{2}$</td><td>$\frac{\sqrt{3}}{2}$</td><td>$\frac{\sqrt{3}}{3}$</td></tr><tr><td>$\frac{\pi}{4}$</td><td>$\frac{\sqrt{2}}{2}$</td><td>$\frac{\sqrt{2}}{2}$</td><td>1</td></tr><tr><td>$\frac{\pi}{3}$</td><td>$\frac{\sqrt{3}}{2}$</td><td>$\frac{1}{2}$</td><td>$\sqrt{3}$</td></tr><tr><td>$\frac{\pi}{2}$</td><td>1</td><td>0</td><td>∞</td></tr></table>		θ	$\sin \theta$	$\cos \theta$	$\tan \theta$	0	0	1	0	$\frac{\pi}{6}$	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{3}}{3}$	$\frac{\pi}{4}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$	1	$\frac{\pi}{3}$	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$	$\sqrt{3}$	$\frac{\pi}{2}$	1	0	∞	<p>... in mathematics you don't understand things, you just get used to them. - J. von Neumann</p>
θ	$\sin \theta$	$\cos \theta$	$\tan \theta$																							
0	0	1	0																							
$\frac{\pi}{6}$	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{3}}{3}$																							
$\frac{\pi}{4}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$	1																							
$\frac{\pi}{3}$	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$	$\sqrt{3}$																							
$\frac{\pi}{2}$	1	0	∞																							
v2.02 ©1994 by Steve Seiden sseiden@acm.org http://www.csc.lsu.edu/~seiden																										

Theoretical Computer Science Cheat Sheet

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

Theoretical Computer Science Cheat Sheet

π	Calculus
<p>Wallis' identity:</p> $\pi = 2 \cdot \frac{2 \cdot 2 \cdot 4 \cdot 4 \cdot 6 \cdot 6 \cdots}{1 \cdot 3 \cdot 3 \cdot 5 \cdot 5 \cdot 7 \cdots}$ <p>Brouncker's continued fraction expansion:</p> $\frac{\pi}{4} = 1 + \frac{1^2}{2 + \frac{3^2}{2 + \frac{5^2}{2 + \frac{7^2}{2 + \cdots}}}}$ <p>Gregory's series:</p> $\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots$ <p>Newton's series:</p> $\frac{\pi}{6} = \frac{1}{2} + \frac{1}{2 \cdot 3 \cdot 2^3} + \frac{1 \cdot 3}{2 \cdot 4 \cdot 5 \cdot 2^5} + \cdots$ <p>Sharp's series:</p> $\frac{\pi}{6} = \frac{1}{\sqrt{3}} \left(1 - \frac{1}{3 \cdot 3} + \frac{1}{3^2 \cdot 5} - \frac{1}{3^3 \cdot 7} + \cdots \right)$ <p>Euler's series:</p> $\frac{\pi^2}{6} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \cdots$ $\frac{\pi^2}{8} = \frac{1}{1^2} + \frac{1}{3^2} + \frac{1}{5^2} + \frac{1}{7^2} + \frac{1}{9^2} + \cdots$ $\frac{\pi^2}{12} = \frac{1}{1^2} - \frac{1}{2^2} + \frac{1}{3^2} - \frac{1}{4^2} + \frac{1}{5^2} - \cdots$	<p>Derivatives:</p> <ol style="list-style-type: none"> $\frac{d(cu)}{dx} = c \frac{du}{dx},$ $\frac{d(u+v)}{dx} = \frac{du}{dx} + \frac{dv}{dx},$ $\frac{d(uv)}{dx} = u \frac{dv}{dx} + v \frac{du}{dx},$ $\frac{d(u^n)}{dx} = nu^{n-1} \frac{du}{dx},$ $\frac{d(u/v)}{dx} = \frac{v(\frac{du}{dx}) - u(\frac{dv}{dx})}{v^2},$ $\frac{d(e^{cu})}{dx} = ce^{cu} \frac{du}{dx},$ $\frac{d(c^u)}{dx} = (\ln c)c^u \frac{du}{dx},$ $\frac{d(\ln u)}{dx} = \frac{1}{u} \frac{du}{dx},$ $\frac{d(\sin u)}{dx} = \cos u \frac{du}{dx},$ $\frac{d(\cos u)}{dx} = -\sin u \frac{du}{dx},$ $\frac{d(\tan u)}{dx} = \sec^2 u \frac{du}{dx},$ $\frac{d(\cot u)}{dx} = -\csc^2 u \frac{du}{dx},$ $\frac{d(\sec u)}{dx} = \tan u \sec u \frac{du}{dx},$ $\frac{d(\csc u)}{dx} = -\cot u \csc u \frac{du}{dx},$ $\frac{d(\arcsin u)}{dx} = \frac{1}{\sqrt{1-u^2}} \frac{du}{dx},$ $\frac{d(\arccos u)}{dx} = \frac{-1}{\sqrt{1-u^2}} \frac{du}{dx},$ $\frac{d(\arctan u)}{dx} = \frac{1}{1+u^2} \frac{du}{dx},$ $\frac{d(\operatorname{arccot} u)}{dx} = \frac{-1}{1+u^2} \frac{du}{dx},$ $\frac{d(\operatorname{arcsec} u)}{dx} = \frac{1}{u\sqrt{1-u^2}} \frac{du}{dx},$ $\frac{d(\operatorname{arccsc} u)}{dx} = \frac{-1}{u\sqrt{1-u^2}} \frac{du}{dx},$ $\frac{d(\sinh u)}{dx} = \cosh u \frac{du}{dx},$ $\frac{d(\cosh u)}{dx} = \sinh u \frac{du}{dx},$ $\frac{d(\tanh u)}{dx} = \operatorname{sech}^2 u \frac{du}{dx},$ $\frac{d(\coth u)}{dx} = -\operatorname{csch}^2 u \frac{du}{dx},$ $\frac{d(\operatorname{sech} u)}{dx} = -\operatorname{sech} u \tanh u \frac{du}{dx},$ $\frac{d(\operatorname{csch} u)}{dx} = -\operatorname{csch} u \coth u \frac{du}{dx},$ $\frac{d(\operatorname{arcsinh} u)}{dx} = \frac{1}{\sqrt{1+u^2}} \frac{du}{dx},$ $\frac{d(\operatorname{arccosh} u)}{dx} = \frac{1}{\sqrt{u^2-1}} \frac{du}{dx},$ $\frac{d(\operatorname{arctanh} u)}{dx} = \frac{1}{1-u^2} \frac{du}{dx},$ $\frac{d(\operatorname{arccoth} u)}{dx} = \frac{1}{u^2-1} \frac{du}{dx},$ $\frac{d(\operatorname{arcsech} u)}{dx} = \frac{-1}{u\sqrt{1-u^2}} \frac{du}{dx},$ $\frac{d(\operatorname{arccsch} u)}{dx} = \frac{-1}{ u \sqrt{1+u^2}} \frac{du}{dx}.$ <p>Integrals:</p> <ol style="list-style-type: none"> $\int cu \, dx = c \int u \, dx,$ $\int (u+v) \, dx = \int u \, dx + \int v \, dx,$ $\int x^n \, dx = \frac{1}{n+1} x^{n+1}, \quad n \neq -1,$ $\int \frac{1}{x} \, dx = \ln x,$ $\int e^x \, dx = e^x,$ $\int \frac{dx}{1+x^2} = \arctan x,$ $\int u \frac{dv}{dx} \, dx = uv - \int v \frac{du}{dx} \, dx,$ $\int \sin x \, dx = -\cos x,$ $\int \cos x \, dx = \sin x,$ $\int \tan x \, dx = -\ln \cos x ,$ $\int \cot x \, dx = \ln \cos x ,$ $\int \sec x \, dx = \ln \sec x + \tan x ,$ $\int \csc x \, dx = \ln \csc x + \cot x ,$ $\int \arcsin \frac{x}{a} \, dx = \arcsin \frac{x}{a} + \sqrt{a^2 - x^2}, \quad a > 0,$
<p>Partial Fractions</p> <p>Let $N(x)$ and $D(x)$ be polynomial functions of x. We can break down $N(x)/D(x)$ using partial fraction expansion. First, if the degree of N is greater than or equal to the degree of D, divide N by D, obtaining</p> $\frac{N(x)}{D(x)} = Q(x) + \frac{N'(x)}{D(x)},$ <p>where the degree of N' is less than that of D. Second, factor $D(x)$. Use the following rules: For a non-repeated factor:</p> $\frac{N(x)}{(x-a)D(x)} = \frac{A}{x-a} + \frac{N'(x)}{D(x)},$ <p>where</p> $A = \left[\frac{N(x)}{D(x)} \right]_{x=a}.$ <p>For a repeated factor:</p> $\frac{N(x)}{(x-a)^m D(x)} = \sum_{k=0}^{m-1} \frac{A_k}{(x-a)^{m-k}} + \frac{N'(x)}{D(x)},$ <p>where</p> $A_k = \frac{1}{k!} \left[\frac{d^k}{dx^k} \left(\frac{N(x)}{D(x)} \right) \right]_{x=a}.$	
<p>The reasonable man adapts himself to the world; the unreasonable persists in trying to adapt the world to himself. Therefore all progress depends on the unreasonable.</p> <p>– George Bernard Shaw</p>	

Theoretical Computer Science Cheat Sheet

Calculus Cont.

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

Theoretical Computer Science Cheat Sheet

Calculus Cont.

$$\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:

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

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

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

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

Rising Factorial Powers:

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

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

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

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

Conversion:

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

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

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

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

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

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

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

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

Theoretical Computer Science Cheat Sheet

Series

Taylor's series:

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

Expansions:

$$\begin{aligned} \frac{1}{1-x} &= 1 + x + x^2 + x^3 + x^4 + \dots = \sum_{i=0}^{\infty} x^i, \\ \frac{1}{1-cx} &= 1 + cx + c^2x^2 + c^3x^3 + \dots = \sum_{i=0}^{\infty} c^i x^i, \\ \frac{1}{1-x^n} &= 1 + x^n + x^{2n} + x^{3n} + \dots = \sum_{i=0}^{\infty} x^{ni}, \\ \frac{x}{(1-x)^2} &= x + 2x^2 + 3x^3 + 4x^4 + \dots = \sum_{i=0}^{\infty} i x^i, \\ \sum_{k=0}^n \binom{n}{k} \frac{k! z^k}{(1-z)^{k+1}} &= x + 2^n x^2 + 3^n x^3 + 4^n x^4 + \dots = \sum_{i=0}^{\infty} i^n x^i, \\ e^x &= 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \dots = \sum_{i=0}^{\infty} \frac{x^i}{i!}, \\ \ln(1+x) &= x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 + \dots = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^i}{i}, \\ \ln \frac{1}{1-x} &= x + \frac{1}{2}x^2 + \frac{1}{3}x^3 + \frac{1}{4}x^4 + \dots = \sum_{i=1}^{\infty} \frac{x^i}{i}, \\ \sin x &= x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \frac{1}{7!}x^7 + \dots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)!}, \\ \cos x &= 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \frac{1}{6!}x^6 + \dots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!}, \\ \tan^{-1} x &= x - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + \dots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)}, \\ (1+x)^n &= 1 + nx + \frac{n(n-1)}{2}x^2 + \dots = \sum_{i=0}^{\infty} \binom{n}{i} x^i, \\ \frac{1}{(1-x)^{n+1}} &= 1 + (n+1)x + \binom{n+1}{2}x^2 + \dots = \sum_{i=0}^{\infty} \binom{i+n}{i} x^i, \\ \frac{x}{e^x - 1} &= 1 - \frac{1}{2}x + \frac{1}{12}x^2 - \frac{1}{720}x^4 + \dots = \sum_{i=0}^{\infty} \frac{B_i x^i}{i!}, \\ \frac{1}{2x}(1 - \sqrt{1-4x}) &= 1 + x + 2x^2 + 5x^3 + \dots = \sum_{i=0}^{\infty} \frac{1}{i+1} \binom{2i}{i} x^i, \\ \frac{1}{\sqrt{1-4x}} &= 1 + 2x + 6x^2 + 20x^3 + \dots = \sum_{i=0}^{\infty} \binom{2i}{i} x^i, \\ \frac{1}{\sqrt{1-4x}} \left(\frac{1 - \sqrt{1-4x}}{2x} \right)^n &= 1 + (2+n)x + \binom{4+n}{2}x^2 + \dots = \sum_{i=0}^{\infty} \binom{2i+n}{i} x^i, \\ \frac{1}{1-x} \ln \frac{1}{1-x} &= x + \frac{3}{2}x^2 + \frac{11}{6}x^3 + \frac{25}{12}x^4 + \dots = \sum_{i=1}^{\infty} H_i x^i, \\ \frac{1}{2} \left(\ln \frac{1}{1-x} \right)^2 &= \frac{1}{2}x^2 + \frac{3}{4}x^3 + \frac{11}{24}x^4 + \dots = \sum_{i=2}^{\infty} \frac{H_{i-1} x^i}{i}, \\ \frac{x}{1-x-x^2} &= x + x^2 + 2x^3 + 3x^4 + \dots = \sum_{i=0}^{\infty} F_i x^i, \\ \frac{F_n x}{1 - (F_{n-1} + F_{n+1})x - (-1)^n x^2} &= F_n x + F_{2n} x^2 + F_{3n} x^3 + \dots = \sum_{i=0}^{\infty} F_{ni} x^i. \end{aligned}$$

Ordinary power series:

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

Exponential power series:

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

Dirichlet power series:

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

Binomial theorem:

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

Difference of like powers:

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

For ordinary power series:

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

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

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

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

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

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

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

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

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

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

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

Convolution:

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

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

Theoretical Computer Science Cheat Sheet																																																																																																						
Series		Escher's Knot																																																																																																				
<div>Expansions:</div> <div>$\frac{1}{(1-x)^{n+1}} \ln \frac{1}{1-x} = \sum_{i=0}^{\infty} (H_{n+i} - H_n) \binom{n+i}{i} x^i,$$x^{\overline{n}} = \sum_{i=0}^{\infty} \left[\begin{matrix} n \\ i \end{matrix} \right] x^i,$$\left(\ln \frac{1}{1-x} \right)^n = \sum_{i=0}^{\infty} \left[\begin{matrix} i \\ n \end{matrix} \right] \frac{n! x^i}{i!},$$\tan x = \sum_{i=1}^{\infty} (-1)^{i-1} \frac{2^{2i} (2^{2i} - 1) B_{2i} x^{2i-1}}{(2i)!},$$\frac{1}{\zeta(x)} = \sum_{i=1}^{\infty} \frac{\mu(i)}{i^x},$$\zeta(x) = \prod_p \frac{1}{1 - p^{-x}},$$\zeta^2(x) = \sum_{i=1}^{\infty} \frac{d(i)}{x^i} \quad \text{where } d(n) = \sum_{d n} 1,$$\zeta(x)\zeta(x-1) = \sum_{i=1}^{\infty} \frac{S(i)}{x^i} \quad \text{where } S(n) = \sum_{d n} d,$$\zeta(2n) = \frac{2^{2n-1} B_{2n} }{(2n)!} \pi^{2n}, \quad n \in \mathbb{N},$$\frac{x}{\sin x} = \sum_{i=0}^{\infty} (-1)^{i-1} \frac{(4^i - 2) B_{2i} x^{2i}}{(2i)!},$$\left(\frac{1 - \sqrt{1-4x}}{2x} \right)^n = \sum_{i=0}^{\infty} \frac{n(2i+n-1)!}{i!(n+i)!} x^i,$$e^x \sin x = \sum_{i=1}^{\infty} \frac{2^{i/2} \sin \frac{i\pi}{4}}{i!} x^i,$$\sqrt{\frac{1 - \sqrt{1-x}}{x}} = \sum_{i=0}^{\infty} \frac{(4i)!}{16^i \sqrt{2} (2i)!(2i+1)!} x^i,$$\left(\frac{\arcsin x}{x} \right)^2 = \sum_{i=0}^{\infty} \frac{4^i i!^2}{(i+1)(2i+1)!} x^{2i}.$</div>		<div>$\left(\frac{1}{x} \right)^{-n} = \sum_{i=0}^{\infty} \left\{ \begin{matrix} i \\ n \end{matrix} \right\} x^i,$$(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},$$\frac{\zeta(x-1)}{\zeta(x)} = \sum_{i=1}^{\infty} \frac{\phi(i)}{i^x},$</div>																																																																																																				
		<div>Stieltjes Integration</div> <div>If G is continuous in the interval $[a, b]$ and F is nondecreasing then$\int_a^b G(x) dF(x)$exists. If $a \leq b \leq c$ then$\int_a^c G(x) dF(x) = \int_a^b G(x) dF(x) + \int_b^c G(x) dF(x).$If the integrals involved exist$\int_a^b (G(x) + H(x)) dF(x) = \int_a^b G(x) dF(x) + \int_a^b H(x) dF(x),$$\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$\int_a^b G(x) dF(x) = \int_a^b G(x) F'(x) dx.$</div>																																																																																																				
<div>Cramer's Rule</div> <div>If we have equations:$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n &= b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n &= b_2 \\ &\vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,n}x_n &= b_n \end{aligned}$Let $A = (a_{i,j})$ and B be the column matrix (b_i). Then there is a unique solution iff $\det A \neq 0$. Let A_i be A with column i replaced by B. Then$x_i = \frac{\det A_i}{\det A}.$</div>		<div><table><tr><td>00</td><td>47</td><td>18</td><td>76</td><td>29</td><td>93</td><td>85</td><td>34</td><td>61</td><td>52</td></tr><tr><td>86</td><td>11</td><td>57</td><td>28</td><td>70</td><td>39</td><td>94</td><td>45</td><td>02</td><td>63</td></tr><tr><td>95</td><td>80</td><td>22</td><td>67</td><td>38</td><td>71</td><td>49</td><td>56</td><td>13</td><td>04</td></tr><tr><td>59</td><td>96</td><td>81</td><td>33</td><td>07</td><td>48</td><td>72</td><td>60</td><td>24</td><td>15</td></tr><tr><td>73</td><td>69</td><td>90</td><td>82</td><td>44</td><td>17</td><td>58</td><td>01</td><td>35</td><td>26</td></tr><tr><td>68</td><td>74</td><td>09</td><td>91</td><td>83</td><td>55</td><td>27</td><td>12</td><td>46</td><td>30</td></tr><tr><td>37</td><td>08</td><td>75</td><td>19</td><td>92</td><td>84</td><td>66</td><td>23</td><td>50</td><td>41</td></tr><tr><td>14</td><td>25</td><td>36</td><td>40</td><td>51</td><td>62</td><td>03</td><td>77</td><td>88</td><td>99</td></tr><tr><td>21</td><td>32</td><td>43</td><td>54</td><td>65</td><td>06</td><td>10</td><td>89</td><td>97</td><td>78</td></tr><tr><td>42</td><td>53</td><td>64</td><td>05</td><td>16</td><td>20</td><td>31</td><td>98</td><td>79</td><td>87</td></tr></table></div> <div>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 $k_m \geq 2$.</div>	00	47	18	76	29	93	85	34	61	52	86	11	57	28	70	39	94	45	02	63	95	80	22	67	38	71	49	56	13	04	59	96	81	33	07	48	72	60	24	15	73	69	90	82	44	17	58	01	35	26	68	74	09	91	83	55	27	12	46	30	37	08	75	19	92	84	66	23	50	41	14	25	36	40	51	62	03	77	88	99	21	32	43	54	65	06	10	89	97	78	42	53	64	05	16	20	31	98	79	87
00	47	18	76	29	93	85	34	61	52																																																																																													
86	11	57	28	70	39	94	45	02	63																																																																																													
95	80	22	67	38	71	49	56	13	04																																																																																													
59	96	81	33	07	48	72	60	24	15																																																																																													
73	69	90	82	44	17	58	01	35	26																																																																																													
68	74	09	91	83	55	27	12	46	30																																																																																													
37	08	75	19	92	84	66	23	50	41																																																																																													
14	25	36	40	51	62	03	77	88	99																																																																																													
21	32	43	54	65	06	10	89	97	78																																																																																													
42	53	64	05	16	20	31	98	79	87																																																																																													
<div>Improvement makes strait roads, but the crooked roads without Improvement, are roads of Genius. – William Blake (The Marriage of Heaven and Hell)</div>		<div>Fibonacci Numbers</div> <div>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.$</div>																																																																																																				