Team Contest Reference

Team:

System.out.println(42);

Roland Haase
Thore Tiemann
Marcel Wienöbst

Contents

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

`byte` (8 Bit, signed): -128 …127
`short` (16 Bit, signed): -32.768 …23.767
`integer` (32 Bit, signed): -2.147.483.648 …2.147.483.647
`long` (64 Bit, signed): $-2^{63} \ldots 2^{63} - 1$

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

1 Algorithms

1.1 Binary Search

Binary searchs for an element in a sorted array.

```java
public static boolean BinarySearch(int[] array, int N,
    int a) {
  int lo = 0;
  int hi = N-1;
  while(lo <= hi) {
    int mid = (int) (((lo + hi) / 2.0) + 0.6);
    if(array[mid] < a) {
      lo = mid+1;
    } else {
      hi = mid-1;
    }
  }
  if(lo < N  && array[lo] == a) {
    return true;
  } else {
    return false;
  }
}
```

MD5: `bb87f09a50f05e688706641c26759706` | $\mathcal{O}(\log n)$

1.2 Bitonic TSP

Input: Distance matrix d with vertices sorted in x-axis direction.
Output: Shortest bitonic tour length

```java
public static double bitonic(double[][] d) {
  int N = d.length;
  double[][] B = new double[N][N];
  for (int j = 0; j < N; j++) {
    for (int i = 0; i <= j; i++) {
      if (i < j - 1)
        B[i][j] = B[i][j - 1] + d[j - 1][j];
      else {
        double min = 0;
        for (int k = 0; k < j; k++) {
          double r = B[k][i] + d[k][j];
          if (min > r || k == 0)
            min = r;
        }
        B[i][j] = min;
      }
    }
  }
  return B[N-1][N-1];
}
```

MD5: `49fca508fb184da171e4c8e18b6ca4c7` | $\mathcal{O}(?)$

1.3 Held Karp

Algorithm for TSP

```java
public static int[] tsp(int[][] graph) {
  int n = graph.length;
  if(n == 1) return new int[]{0};
  //C stores the shortest distance to node of the second dimension
  //first dimension is the bitstring of included nodes on the way
  int[][] C = new int[1<<n][n];
  int[][] p = new int[1<<n][n];
  //initialize
  for(int k = 1; k < n; k++) {
     C[1<<k][k] = graph[0][k];
  }
  for(int s = 2; s < n; s++) {
     for(int S = 1; S < (1<<n); S++) {
        if(Integer.bitCount(S)!=s || (S&1) == 1)
           continue;
        for(int k = 1; k < n; k++) {
                if((S & (1 << k)) == 0)
                   continue;

           //Smk is the set of nodes without k
           int Smk = S ^ (1<<k);

           int min = Integer.MAX_VALUE;
           int minprev = 0;
           for(int  m=1; m<n; m++) {
              if((Smk & (1<<m)) == 0)
                 continue;
              //distance to m with the nodes in Smk + connection from m to k
              int tmp = C[Smk][m] +graph[m][k];
```

```java
                if(tmp < min) {
                    min = tmp;
                    minprev = m;
                }
            }
            C[S][k] = min;
            p[S][k] = minprev;
        }
    }
  }

  //find shortest tour length
  int min = Integer.MAX_VALUE;
  int minprev = -1;
  for(int k = 1; k < n; k++) {
    //Set of all nodes except for the first + cost from 0 to k
    int tmp = C[(1<<n) - 2][k] + graph[0][k];
    if(tmp < min) {
      min = tmp;
      minprev = k;
    }
  }

  //Note that the tour has not been tested yet, only the correctness of the min-tour-value
  //backtrack tour
  int[] tour = new int[n+1];
  tour[n] = 0;
  tour[n-1] = minprev;
  int bits = (1<<n)-2;
  for(int k = n-2; k>0; k--) {
    tour[k] = p[bits][tour[k+1]];
    bits = bits ^ (1<<tour[k+1]);
  }
  tour[0] = 0;
  return tour;
}
```

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

```java
public static ArrayList<Integer> kmp(String s, String w) {
  ArrayList<Integer> ret = new ArrayList<>();
  //Build prefix table
  int[] N = new int[w.length()+1];
  int i=0; int j =-1; N[0]=-1;
  while (i<w.length()) {
    while (j>=0 && w.charAt(j) != w.charAt(i))
      j = N[j];
    i++; j++; N[i]=j;
  }
  //Search string
  i=0; j=0;
  while (i<s.length()) {
    while (j>=0 && s.charAt(i) != w.charAt(j))
      j = N[j];
    i++; j++;
    if (j==w.length()) { //match found
      ret.add(i-w.length()); //add its start index
      j = N[j];
    }
  }
  return ret;
}
```

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.

```java
public static int levenshteinDistance(String a, String b) {

  a = a.toLowerCase();
  b = b.toLowerCase();

  int[] costs = new int[b.length() + 1];

  for (int j = 0; j < costs.length; j++) {
    costs[j] = j;
  }

  for (int i = 1; i <= a.length(); i++) {
    costs[0] = i;
    int nw = i - 1;
    for (int j = 1; j <= b.length(); j++) {
      int cj = Math.min(1 + Math.min(costs[j], costs[j - 1]),
          a.charAt(i - 1) == b.charAt(j - 1) ? nw : nw + 1);
      nw = costs[j];
      costs[j] = cj;
    }
  }

  return costs[b.length()];
}
```

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.

```java
public static String longestCommonSubsequence(String string1, String string2) {

  char[] s1 = string1.toCharArray();
  char[] s2 = string2.toCharArray();

  int[][] num = new int[s1.length + 1][s2.length + 1];

  // Actual algorithm
  for (int i = 1; i <= s1.length; i++)
    for (int j = 1; j <= s2.length; j++)
      if (s1[i - 1] == s2[j - 1])
        num[i][j] = 1 + num[i - 1][j - 1];
      else
        num[i][j] = Math.max(num[i - 1][j], num[i][j - 1]);
```

```java

  // System.out.println("length of LCS = " + num[s1.length][s2.length]);

  int s1position = s1.length, s2position = s2.length;
  List<Character> result = new LinkedList<Character>();

  while (s1position != 0 && s2position != 0) {
    if (s1[s1position - 1] == s2[s2position - 1]) {
      result.add(s1[s1position - 1]);
      s1position--;
      s2position--;
    } else if (num[s1position][s2position - 1] >= num[s1position][s2position]) {
      s2position--;
    } else {
      s1position--;
    }
  }
  Collections.reverse(result);

  char[] resultString = new char[result.size()];
  int i = 0;

  for (Character c : result) {
    resultString[i] = c;
    i++;
  }

  return new String(resultString);
}
```

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

1.7 LongestIncreasingSubsequence

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

```java
//This has not been tested yet (adapted from tested C++ Murcia Code)
public static int longestInc(int[] array, int N) {
   int[] m = new int[N];
   for (int i = N - 1; i >= 0; i--) {
      m[i] = 1;
      for (int j = i + 1; j < N; j++) {
         if (array[j] > array[i]) {
            if (m[i] < m[j] + 1) {
               m[i] = m[j] + 1;
            }
         }
      }
   }
   int longest = 0;
   for (int i = 0; i < N; i++) {
      if (m[i] > longest) {
         longest = m[i];
      }
   }
   return longest;
}
```

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

1.8 LongestIncreasingSubsequence

Computes the longest increasing subsequence using binary search.

```java
public static int[] LongestIncreasingSubsequencenlogn(int[] a, int[] p) {
   int[] m = new int[a.length+1];
   int l = 0;
   for(int i = 0; i < a.length; i++) {
      int lo = 1;
      int hi = l;
      while(lo <= hi) {
         int mid = (int) (((lo + hi) / 2.0) + 0.6);
         if(a[m[mid]] < a[i]) {
            lo = mid+1;
         } else {
            hi = mid-1;
         }
      }
      int newL = lo;
      p[i] = m[newL-1];
      m[newL] = i;
      if(newL > l) {
         l = newL;
      }
   }
   int[] s = new int[l];
   int k = m[l];
   for(int i= l-1; i>= 0; i--) {
      s[i] = a[k];
      k = p[k];
   }
   return s;
}
```

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.

```java
public static boolean nextPermutation(char[] a) {
   int i = a.length - 1;
   while(i > 0 && a[i-1] >= a[i]) {
      i--;
   }
   if(i <= 0) {
      return false;
   }
   int j = a.length - 1;
   while (a[j] <=  a[i-1]) {
      j--;
   }
   char tmp = a[i - 1];
   a[i - 1] = a[j];
   a[j] = tmp;

   j = a.length - 1;
   while(i < j) {
      tmp = a[i];
      a[i] = a[j];
      a[j] = tmp;
      i++;
      j--;
   }
   return true;
```

```
}
```

MD5: `ca6266722db16f2dc8eae5a6cc5fcacf` | $\mathcal{O}(?)$

1.10 Solve 2SAT

Allocate a graph with $|V| = 2 \cdot n$ for $x_{1 \ldots n}$. 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`.

```java
public static void addClause(Vertex[] G, int a, int b)
    {
  int nega = a<0 ? 0 : 1; int negb = b<0 ? 0 : 1;
  a = Math.abs(a)-1; b = Math.abs(b)-1;
  int Xa = (a<<1)+nega; int Xb = (b<<1)+negb;
  G[Xa^1].next.add(Xb);
  G[Xb^1].next.add(Xa);
}
public static int[] solve2Sat(Vertex[] G) {
  Integer[] color = scc(G);
  for (int i=0; i<G.length; i+=2)
    if (color[i] == color[i+1])
      return null; //contradiction!

  HashSet<Integer>[] sccV = new HashSet[G.length];
  HashSet<Integer>[] sccEn = new HashSet[G.length];
  HashSet<Integer>[] sccEp = new HashSet[G.length];
  Integer[] vals = new Integer[G.length];
  for (int i=0; i<G.length; i++) {
    sccV[i] = new HashSet<Integer>();
    sccEn[i] = new HashSet<Integer>();
    sccEp[i] = new HashSet<Integer>();
  }
  //create reverse SCC DAG
  for (int i=0; i<G.length; i++)
    if (G[i]!=null) {
      sccV[color[i]].add(i);
      for (int j : G[i].next)
        if (color[i] != color[j]) {
          sccEn[color[i]].add(color[j]);
          sccEp[color[j]].add(color[i]);
        }
    }
  //go in rev topo order and set vars
  Stack<Integer> tail = new Stack<Integer>();
  for (int i=0; i<G.length; i++)
    if (!sccV[i].isEmpty() && sccEn[i].isEmpty())
      tail.push(i);
  while (!tail.isEmpty()) {
    int curr = tail.pop();
    for (int i : sccV[curr]) {
      if (vals[i]!=null)
        break;
      vals[i] = 1;
      vals[i^1] = 0;
    }
    for (int i : sccEp[curr]) {
      sccEn[i].remove(curr);
      if (sccEn[i].isEmpty())
        tail.push(i);
    }
  }

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

MD5: `60fb0af11d8fc325eb0efb71031ca312` | $\mathcal{O}(|E| + |V|)$

2 Graphs

2.1 BellmanFord

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

```java
public static boolean bellmanFord(Vertex[] vertices) {
   //source is 0
   vertices[0].mindistance = 0;
   //calc distances
   for(int i = 0; i < vertices.length-1; i++) {
      for(int j = 0; j < vertices.length; j++) {
         for(Edge e: vertices[j].adjacencies) {
            if(vertices[j].mindistance != Integer.
                MAX_VALUE
               && e.target.mindistance > vertices[j].
                   mindistance + e.distance) {
               e.target.mindistance = vertices[j].
                   mindistance + e.distance;
            }
         }
      }
   }
   //check for negative-length cycle
   for(int i = 0; i < vertices.length; i++) {
      for(Edge e: vertices[i].adjacencies) {
         if(vertices[i].mindistance != Integer.
             MAX_VALUE && e.target.mindistance >
             vertices[i].mindistance + e.distance) {
            return true;
         }
      }
   }
   return false;
}
```

MD5: `36561a7913a81baf7b7c79b606683819` | $\mathcal{O}(|V| \cdot |E|)$

2.2 Bipartite Graph Check

Checks a graph represented as adjList for being bipartite. Needs a little adaption, if the graph is not connected.
Input: $graph$ as adjList, amount of nodes N as int
Output: `true` if graph is bipartite, `false` otherwise

```java
public static boolean bipartiteGraphCheck(
    ArrayList<ArrayList<Integer>> graph, int N) {
  int[] color = new int[N];
  for(int i = 0; i < N; i++) color[i] = -1;
  // use bfs for coloring each node
  color[0] = 1;
  // FIFO-Queue
  Queue<Integer> q = new LinkedList<Integer>();
  q.add(0);
  while(!q.isEmpty()) {
```

```java
    int u = q.poll();
    for(int i : graph.get(u)) {
      // if node i not yet visited,
      // give opposite color of parent node u
      if(color[i] == -1) {
        color[i] = 1-color[u];
        q.add(i);
      // if node i has same color as parent node u
      // the graph is not bipartite
      } else if(color[u] == color[i])
        return false;
      // if node i has different color
      // than parent node u keep going
    }
  }
  return true;
}
```

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.)

```java
// A DFS based recursive function that returns true
// if a matching for vertex u is possible
boolean bpm(boolean bpGraph[][], int u,
            boolean seen[], int matchR[]) {
// Try every job one by one
  for (int v = 0; v < N; v++) {
// If applicant u is interested in job v and v
// is not visited
    if (bpGraph[u][v] && !seen[v]) {
      seen[v] = true; // Mark v as visited

// If job v is not assigned to an applicant OR
// previously assigned applicant for job v (which
// is matchR[v]) has an alternate job available.
// Since v is marked as visited in the above line,
// matchR[v] in the following recursive call will
// not get job v again
      if (matchR[v] < 0 ||
          bpm(bpGraph, matchR[v], seen, matchR)) {
        matchR[v] = u;
        return true;
      }
    }
  }
  return false;
}

// Returns maximum number of matching from M to N
int maxBPM(boolean bpGraph[][]) {
// An array to keep track of the applicants assigned
// to jobs. The value of matchR[i] is the applicant
// number assigned to job i, the value -1 indicates
// nobody is assigned.
  int matchR[] = new int[N];

// Initially all jobs are available
  for(int i = 0; i < N; ++i)
    matchR[i] = -1;
// Count of jobs assigned to applicants
  int result = 0;
  for (int u = 0; u < M; u++) {
// Mark all jobs as not seen for next applicant.
    boolean seen[] = new boolean[N];
    for(int i = 0; i < N; ++i)
      seen[i] = false;

// Find if the applicant u can get a job
    if (bpm(bpGraph, u, seen, matchR))
      result++;
  }
  return result;
}
```

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

2.4 Depth First Search

Searches for a path between two vertices and 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 $path$.

```java
public static boolean DFS(int s, int t, int[][] G,
    List<Integer> path, List<Integer> list) {
  // needed for start of recursion
  if (path.size() == 0)
    path.add(s);
  // return true if target reached
  if (s == t)
    return true;
  // otherwise recursively search neighbour
  for (int i = 0; i < G.length; i++) {
    // if node reachable but not yet visible
    if (G[s][i] > 0 && !list.contains(i)) {
      path.add(i); // i is on path from s to t
      list.add(i); // mark i as visited
      // if path from i to t found
      // return true
      if (DFS(i, t, G, path, list))
        return true;
      // else i is not on path from s to t
      // search next neighbour
      else
        path.remove(path.size() - 1);
    }
  }
  return false;
}
```

MD5: `59fee23ddc452534f3712142186e59cc` | $\mathcal{O}(|V|^2)$

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 predcessor vertices set.

```java
public static void dijkstra(Vertex[] vertices, int src
    ) {
   vertices[src].mindistance = 0;
   PriorityQueue<Vertex> queue = new PriorityQueue<
       Vertex>();
   queue.add(vertices[src]);
   while(!queue.isEmpty()) {
      Vertex u = queue.poll();
      if(u.visited)
         continue;
      u.visited = true;
      for(Edge e : u.adjacencies) {
         Vertex v = e.target;
         if(v.mindistance > u.mindistance + e.distance
             ) {
            v.mindistance = u.mindistance + e.distance
               ;
            queue.add(v);
         }
      }
   }
}
class Vertex implements Comparable<Vertex> {
   public int id;
   public int mindistance = Integer.MAX_VALUE;
   public LinkedList<Edge> adjacencies = new
       LinkedList<Edge>();
   public boolean visited = false;

   public int compareTo(Vertex other) {
      return Integer.compare(this.mindistance, other.
          mindistance);
   }
}
class Edge {
   public Vertex target;
   public int distance;

   public Edge (Vertex target, int distance) {
      this.target = target;
      this.distance = distance;
   }
}
```

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

2.6 EdmondsKarp

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

```java
public static boolean BFS(int[][] graph, int s, int t,
     int[] parent) {
   int N = graph.length;
   boolean[] visited = new boolean[N];
   for(int i = 0; i < N; i++) {
      visited[i] = false;
   }
   Queue<Integer> queue = new LinkedList<Integer>();
   queue.add(s);
   visited[s] = true;
   parent[s] = -1;
   while(!queue.isEmpty()) {
      int u = queue.poll();
      if(u == t) return true;
      for(int v= 0; v < N; v++) {
         if(visited[v] == false && graph[u][v] > 0) {
            queue.add(v);
            parent[v] = u;
            visited[v] = true;
         }
      }
   }
   return (visited[t]);
}
public static int fordFulkerson(int[][] graph, int s,
     int t) {
   int N = graph.length;
   int[][] rgraph = new int[graph.length][graph.length
      ];
   for(int u = 0; u < graph.length; u++) {
      for(int v = 0; v < graph.length; v++) {
         rgraph[u][v] = graph[u][v];
      }
   }
   int[] parent = new int[N];
   int maxflow = 0;
   while(BFS(rgraph, s, t, parent)) {
      int pathflow = Integer.MAX_VALUE;
      for(int v = t; v!= s; v = parent[v]) {
         int u = parent[v];
         pathflow = Math.min(pathflow, rgraph[u][v]);
      }

      for(int v = t; v != s; v = parent[v]) {
         int u = parent[v];
         rgraph[u][v] -= pathflow;
         rgraph[v][u] += pathflow;
      }

      maxflow += pathflow;
   }
   return maxflow;
}
```

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

2.7 FenwickTree

Can be used for computing prefix sums.

```java
int[] fwktree = new int[m + n + 1];
public static int read(int index, int[] fenwickTree) {
   int sum = 0;
   while (index > 0) {
      sum += fenwickTree[index];
      index -= (index & -index);
   }
   return sum;
}
public static int[] update(int index, int addValue,
     int[] fenwickTree) {
   while (index <= fenwickTree.length - 1) {
      fenwickTree[index] += addValue;
      index += (index & -index);
   }
   return fenwickTree;
}
```

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

2.8 FloydWarshall

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

```
public static void floydWarshall(int[][] graph, int
    [][] next, int[][] ans) {
  for(int i = 0; i < ans.length; i++) {
    for(int j = 0; j < ans.length; j++) {
      ans[i][j] = graph[i][j];
    }
  }
  for (int k = 0; k < ans.length; k++) {
    for (int i = 0; i < ans.length; i++) {
      for (int j = 0; j < ans.length; j++) {
        if (ans[i][k] + ans[k][j] < ans[i][j]
        && ans[i][k] < Integer.MAX_VALUE && ans[k
            ][j] < Integer.MAX_VALUE) {
          ans[i][j] = ans[i][k] + ans[k][j];
          next[i][j] = next[i][k];
        }
      }
    }
  }
}
```

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

2.9 Breadth First Search AdjMtrx Iterative

Iterative BFS on adjacency matrix. Needs a little adaption, if graph is not connected.

Input: nodes s and g as int and $graph$ as adjMatrix

Output: `true` if there is a connection between s and g, `false` otherwise

```
public static boolean BFSWithoutPathForAdjMatr(int s,
    int g, int[][] graph) {
  //s being the start and g the goal
  boolean[] visited = new boolean[graph.length];
  for(int i = 0; i < visited.length; i++)
    visited[i] = false;
  // FIFO-Queue
  Queue<Integer> queue = new LinkedList<Integer>();
  queue.add(s);
  visited[s] = true;
  // search all nodes reachable from s
  while(!queue.isEmpty()) {
    int node = queue.poll();
    // if goal reached, return true
    if(node == g)
      return true;
    // else add all neighbours to queue
    // if not yet visited
    for(int i = 0; i < graph.length; i++) {
      if(graph[node][i] > 0 && !visited[i]) {
        queue.add(i);
        visited[i] = true;
      }
    }
  }
  return false;
}
```

MD5: `63fa4882cc8ab028b97d432b725c7f89` | $\mathcal{O}(|V| + |E|)$

2.10 Kruskal

Computes a minimum spanning tree for a weighted undirected graph.

```
public class Freckles {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int t = s.nextInt();
    for(int i = 0; i < t; i++) {
      int n = s.nextInt();
      double[] x = new double[n];
      double[] y = new double[n];
      for(int j = 0; j < n; j++) {
        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
```

```java
            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);
        if (px == py)
            return false; // same set -> reject edge
        if (r[px] < r[py]) { // swap so that always h[px
            ]>=h[py]
            int tmp = px;
            px = py;
            py = tmp;
        }
        p[py] = px; // hang flatter tree as child of
            higher tree
        r[px] = Math.max(r[px], r[py] + 1); // update (
            worst-case) height
        count--;
        return true;
    }
}

class Edge1 implements Comparable<Edge1> {
    double distance;
    int start;
    int end;

    public Edge1(double distance, int start, int end) {
        this.distance = distance;
        this.start = start;
        this.end = end;
    }

    public int compareTo(Edge1 arg0) {
        return Double.compare(this.distance, arg0.
            distance);
    }
}
```

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

2.11 MinCut

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

```java
public static void MinCut(int s, int[][] graph,
    LinkedList<Integer> S, LinkedList<Integer> T) {
    boolean[] visited = new boolean[graph.length];
    for(int i = 0; i < visited.length; i++)
        visited[i] = false;
    Queue<Integer> queue = new LinkedList<Integer>();
    queue.add(s);
    S.add(s);
    visited[s] = true;
    while(!queue.isEmpty()) {
        int node = queue.poll();
        for(int i = 0; i < graph.length; i++) {
            if(graph[node][i] > 0 && !visited[i]) {
                queue.add(i);
                if(!S.contains(i))
                    S.add(i);
                visited[i] = true;
            }
        }
    }
    for(int i = 0; i < graph.length; i++) {
        if(!S.contains(i)) {
            T.add(i);
        }
    }
    for(int i = 0; i < graph.length; i++) {
        for(int j = 0; j < graph.length; j++) {
            if((graph[i][j] > 0 || graph[j][i] > 0) && S.
                contains(i) && T.contains(j)) {
                System.out.println((i+1) + "␣" + (j+1));
            }
        }
    }
}
```

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

2.12 Path-Based SCCs

Finds the strongly connected components in given directed graph.

```java
public static Integer[] scc(Vertex[] G) {
  Stack<Integer> call = new Stack<>();

  Stack<Integer> reps = new Stack<>();
  Stack<Integer> open = new Stack<>();
  Integer[] order = new Integer[G.length];
  int count = 0;

  Integer[] sccs = new Integer[G.length];
  int sccnum = 0;

  for (int i=0; i<G.length; i++) {
    if (G[i]==null) //no such vertex
      continue;

    if (sccs[i]==null) {
      call.push(i);
      while (!call.isEmpty()) {
        int v = call.peek();
        if (order[v]==null) { //first entered
          order[v] = count++;
          reps.push(v);
          open.push(v);

          for (int w : G[v].next) { //process edges
            if (order[w]==null) {
              call.push(w);
            } else if (sccs[w]==null) {
              while (order[reps.peek()]>order[w])
                reps.pop();
            }
          }

        } else { //returned from recursion
          //is still rep. -> completed SCC
          if (reps.peek()==v) {
            int tmp = 0;
            do {
```

```
              tmp = open.pop();
              sccs[tmp] = sccnum;
            } while (tmp != v);
            sccnum++;
            reps.pop();
          }

          call.pop(); //node done
        }
      }
    }
  }
  return sccs;
}
```

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.

```
public static int suurballe(int s, int t, Vertex[] G)
    {
  dijkstra(s, G); //find a shortest path
  ArrayList<Integer> path = new ArrayList<Integer>();
  int id = t;
  while (G[id].pred != id) {
    path.add(0, id);
    id = G[id].pred;
  }
  path.add(0, id);

  //modify weights
  for (int i=0; i<G.length; i++) {
    Vertex u = G[i];
    if (u==null) continue;
    u.backupNext = new HashMap<Integer,Integer>(u.next
        ); //copy old values
    for (Integer j : u.backupNext.keySet()) {
      Vertex v = G[j];
      int weight = u.next.get(j);
      u.next.put(j, weight - v.dist + u.dist);
    }
  }
  //reverse edges on shortest path
  id = s;
  for (int i=0; i<path.size()-1; i++) {
    G[path.get(i)].next.remove(path.get(i+1));
    G[path.get(i+1)].next.put(path.get(i), 0);
  }
  //remove edges to s
  for (int i=0; i<G.length; i++) {
    if (G[i]==null) continue;
    if (G[i].next.containsKey(s))
      G[i].next.remove(s);
  }

  dijkstra(s, G);
  ArrayList<Integer> path2 = new ArrayList<Integer>();
  id = t;
  if (G[id].pred == -1)
    return -1; //no 2nd path!

  while (G[id].pred != id) {
    path2.add(0, id);
    id = G[id].pred;
  }
  path2.add(0, id);

  int totalpath = 0;

  //disregard 0-cycles and edges not on both paths
  id = s;
  //add edges on first shortest path
  for (int i=0; i<path.size()-1; i++) {
    int u = path.get(i);
    int v = path.get(i+1);

    G[u].suurbaleResult.put(v, G[u].backupNext.get(v))
        ;
    totalpath += G[u].suurbaleResult.get(v);
  }
  //add second path, remove cycles
  for (int i=0; i<path2.size()-1; i++) {
    int u = path2.get(i);
    int v = path2.get(i+1);

    if (G[v].suurbaleResult.containsKey(u)) {
      totalpath -= G[v].suurbaleResult.get(u);
      G[v].suurbaleResult.remove(u);
    } else {
      G[u].suurbaleResult.put(v, G[u].backupNext.get(v
          ));
      totalpath += G[u].suurbaleResult.get(v);
    }
  }

  return totalpath;
}
```

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

2.14 Topological Sort

Sorts a graph (represented as adjMtrx) topologically

```
// l enthaelt alle Knoten topologisch sortiert (Start:
     0, Ende= n)
int[] l = new int[n];
int idx = 0;
// s enthaelt alle Knoten, die keine eingehende Kante
    haben
ArrayList<Integer> s = new ArrayList<Integer>();
// initialisiere s
for (int i = 0; i < n; i++) {
if (edgesIn[i] == 0) {
s.add(i);
}
}
// Algo Beginn
while (!s.isEmpty()) {
   int node = s.remove(0);
   l[idx++] = node;
   for (int i = 0; i < n; i++) {
      if (adjMtrx[node][i]) {
         adjMtrx[node][i] = false;
         edgesIn[i] -= 1;
         if (edgesIn[i] == 0) {
            s.add(i);
         }
```

```
    }
  }
}
```

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

3 Math

3.1 Binomial Coefficient

Gives binomial coefficient (n choose k)

```
public static long bin(int n, int k) {
  if (k == 0) {
    return 1;
  } else if (k > n/2) {
    return bin(n, n-k);
  } else {
    return n*bin(n-1, k-1)/k;
  }
}
```

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

3.2 Binomial Matrix

Gives binomial coefficients for all K <= N.

```
public static long[][] binomial_matrix(int N, int K) {
  long[][] B = new long[N+1][K+1];
  for (int k = 1; k <= K; k++) {
    B[0][k] = 0;
  }
  for (int m = 0; m <= N; m++) {
    B[m][0] = 1;
  }
  for (int m = 1; m <= N; m++) {
    for (int k = 1; k <= K; k++) {
      B[m][k] = B[m-1][k-1] + B[m-1][k];
    }
  }
  return B;
}
```

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

3.3 Graham Scan

GrahamScan finds convex hull. Still has collinear point problematic at the last diagonal.

```
public static int ccw(Point src, Point q1, Point q2) {
   return (q1.x - src.x) * (q2.y - src.y) - (q2.x - src.x) * (q1.y - src.y);
}

public static boolean isColl(Point a, Point b, Point c) {
   if((b.y - a.y) * (c.x - b.x) == (c.y - b.y) * (b.x - a.x)) {
      return true;
   } else {
      return false;
   }
}

public static double calcDist(Point src, Point target) {
   return Math.sqrt((src.x + target.x) * (src.x + target.x) + (src.y + target.y) * (src.y * target.y));
}

//Expects a array sorted with PolarComp as Comparator
//IMPORTANT! before sorting put lowest, and if two are the same leftmost, element at position 0 in array
public static void grahamScan(Point[] points) {
   int m = 1;
   for(int i = 2; i < points.length; i++) {
      while(ccw(points[m-1], points[m], points[i]) < 0) {
         if(m > 1) m--;
         else if(i == points.length) break;
         else i++;
      }
      m++;
      Point tmp = points[i];
      points[i] = points[m];
      points[m] = tmp;
   }
}

class Point {
   int x;
   int y;
   public Point(int x, int y) {
      this.x = x;
      this.y = y;
   }
}

class PolarComp implements Comparator<Point> {
   Point src;

   public PolarComp(Point source) {
      src = source;
   }

   public double calcDist(Point q1, Point q2) {
      return Math.sqrt((q1.x - q2.x) * (q1.x - q2.x) + (q1.y - q2.y) * (q1.y - q2.y));
   }

   public int ccw(Point q1, Point q2) {
      return (q1.x - src.x) * (q2.y - src.y) - (q2.x - src.x) * (q1.y - src.y);
   }

   public int compare(Point q1, Point q2) {
      int res = ccw(q1, q2);
      double dist1 = calcDist(src, q1);
      double dist2 = calcDist(src, q2);
      if(res > 0) return -1;
      else if(res < 0) return 1;
      else if(res == 0 && dist1 < dist2) return 1;
      else if(res == 0 && dist1 > dist2) return -1;
      else  return 0;
   }
}
```

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

3.4 Divisability

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

```java
public static long digit_sum(String M, int k, boolean alt) {
  long dig_sum = 0;
  int vz = 1;
  while (M.length() > k) {
    if (alt) vz *= -1;
    dig_sum += vz*Integer.parseInt(M.substring(M.length()-k));
    M = M.substring(0, M.length()-k);
  }
  if (alt) vz *= -1;
  dig_sum += vz*Integer.parseInt(M);
  return dig_sum;
}
// example: divisibility of M by 13
public static boolean divisible13(String M) {
  return digit_sum(M, 3, true)%13 == 0;
}
```

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

3.5 Iterative EEA

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

```java
// Extended Euclidean Algorithm - iterativ
public static long[] eea(long a, long b) {
  if (b > a) {
    long tmp = a;
    a = b;
    b = tmp;
  }
  long x = 0, y = 1, u = 1, v = 0;
  while (a != 0) {
    long q = b / a, r = b % a;
    long m = x - u * q, n = y - v * q;
    b = a; a = r; x = u; y = v; u = m; v = n;
  }
  long gcd = b;
  // x = a^-1 % b, y = b^-1 % a
  // ax + by = gcd
  long[] erg = { gcd, x, y };
  return erg;
}
```

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

3.6 Polynomial Interpolation

```java
public class interpol {

  // divided differences for points given by vectors x and y
  public static rat[] divDiff(rat[] x, rat[] y) {
    rat[] temp = y.clone();
    int n = x.length;
    rat[] res = new rat[n];
    res[0] = temp[0];
    for (int i=1; i < n; i++) {
      for (int j = 0; j < n-i; j++) {
        temp[j] = (temp[j+1].sub(temp[j])).div(x[j+i].sub(x[j]));
      }
      res[i] = temp[0];
    }
    return res;
  }

  // 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+") = "+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);
```

```java
    }

    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;
      }
      return c;
    }

    public void shorten() {
      long ggT = ggT(this.c, this.d);
      this.c = this.c / ggT;
      this.d = this.d / ggT;
      if (d < 0) {
        this.d *= -1;
        this.c *= -1;
      }
    }

    public String toString() {
      if (this.d == 1) return ""+c;
      return ""+c+"/"+d;
    }

    public rat mult(rat b) {
      return new rat(this.c*b.c, this.d*b.d);
    }

    public rat div(rat b) {
      return new rat(this.c*b.d, this.d*b.c);
    }

    public rat add(rat b) {
      long new_d = kgV(this.d, b.d);
      long new_c = this.c*(new_d/this.d) + b.c*(new_d/b.
          d);
      return new rat(new_c, new_d);
    }

    public rat sub(rat b) {
      return this.add(new rat(-b.c, b.d));
    }

}
```

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

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.

```java
public static boolean[] sieveOfEratosthenes(int N) {
  boolean[] isPrime = new boolean[N+1];
    for (int i=2; i<=N; i++) isPrime[i] = true;
    for (int i = 2; i*i <= N; i++)
        if (isPrime[i])
            for (int j = i*i; j <= N; j+=i)
                isPrime[j] = false;
    return isPrime;
}
```

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

4 tcr-roland

5 Math Roland

5.1 Divisability Explanation

$D \mid M \Leftrightarrow D \mid \texttt{digit\_sum}(\texttt{M}, \texttt{k}, \texttt{alt})$, 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:
    $M = \{(x_1, \ldots, x_k) : 1 \leq x_i \leq n,\ x_i \neq x_j \text{ if } i \neq j\}$,
    $|M| = \frac{n!}{(n-k)!}$
  - with repetition:
    $M = \{(x_1, \ldots, x_k) : 1 \leq x_i \leq n\}, |M| = n^k$
- Combinations (unordered): k out of n objects
  - without repetition: $M = \{(x_1, \ldots, x_n) : x_i \in \{0,1\},\ x_1 + \ldots + x_n = k\}, |M| = \binom{n}{k}$
  - with repetition: $M = \{(x_1, \ldots, x_n) : x_i \in \{0, 1, \ldots, k\},\ x_1 + \ldots + x_n = k\}, |M| = \binom{n+k-1}{k}$
- Ordered partition of numbers: $x_1 + \ldots + x_k = n$ (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: $x_1 + \ldots + x_k = n$ (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 $\{(x_0, y_0), \ldots, (x_n, y_n)\}$ find $p \in \Pi_n$ with $p(x_i) = y_i$ for all $i = 0, \ldots, n$.
Solution: $p(x) = \sum\limits_{i=0}^{n} \gamma_{0,i} \prod\limits_{j=0}^{i-1} (x - x_i)$ 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 $i = n-1, \ldots, 0$ with $b_0 = p(x)$.

5.4 Fibonacci Sequence

5.4.1 Binet's formula

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

5.4.2 Generalization

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

5.4.3 Pisano Period

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

6 Java Knowhow

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

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

flags:

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

width specifies output width

precision is for floating point precision

conv:

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

6.2 Modulo: Avoiding negative Integers

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

6.3 Speed up IO

Use

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

Use

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

Theoretical Computer Science Cheat Sheet

Definitions

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

Series

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

In general:

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

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

Geometric series:

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

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

Harmonic series:

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

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

Identities

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Theoretical Computer Science Cheat Sheet

Identities Cont.

38. $\displaystyle {n+1 \brack m+1} = \sum_k {n \brack k}\binom{k}{m} = \sum_{k=0}^{n} {k \brack m} n^{\underline{n-k}} = n! \sum_{k=0}^{n} \frac{1}{k!}{k \brack m},$ **39.** $\displaystyle {x \brack x-n} = \sum_{k=0}^{n} \left\langle\!\!\left\langle {n \atop k} \right\rangle\!\!\right\rangle \binom{x+k}{2n},$

40. $\displaystyle {n \brace m} = \sum_k \binom{n}{k}{k+1 \brace m+1}(-1)^{n-k},$ **41.** $\displaystyle {n \brack m} = \sum_k {n+1 \brack k+1}\binom{k}{m}(-1)^{m-k},$

42. $\displaystyle {m+n+1 \brace m} = \sum_{k=0}^{m} k{n+k \brace k},$ **43.** $\displaystyle {m+n+1 \brack m} = \sum_{k=0}^{m} k(n+k){n+k \brack k},$

44. $\displaystyle \binom{n}{m} = \sum_k {n+1 \brace k+1}{k \brack m}(-1)^{m-k},$ **45.** $\displaystyle (n-m)!\binom{n}{m} = \sum_k {n+1 \brack k+1}{k \brace m}(-1)^{m-k},$ for $n \geq m$,

46. $\displaystyle {n \brace n-m} = \sum_k \binom{m-n}{m+k}\binom{m+n}{n+k}{m+k \brack k},$ **47.** $\displaystyle {n \brack n-m} = \sum_k \binom{m-n}{m+k}\binom{m+n}{n+k}{m+k \brace k},$

48. $\displaystyle {n \brace \ell+m}\binom{\ell+m}{\ell} = \sum_k {k \brace \ell}{n-k \brace m}\binom{n}{k},$ **49.** $\displaystyle {n \brack \ell+m}\binom{\ell+m}{\ell} = \sum_k {k \brack \ell}{n-k \brack m}\binom{n}{k}.$

Trees

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

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

Recurrences

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

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

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

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

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

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

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

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

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

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

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

$$\begin{aligned} 1\big(T(n) - 3T(n/2) &= n\big) \\ 3\big(T(n/2) - 3T(n/4) &= n/2\big) \\ \vdots \quad \vdots \quad & \vdots \\ 3^{\log_2 n - 1}\big(T(2) - 3T(1) &= 2\big) \end{aligned}$$

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

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

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

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

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

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

Generating functions:

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

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

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

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

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

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

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

So $g_i = 2^i - 1$.

Theoretical Computer Science Cheat Sheet

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

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

Pascal's Triangle

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

General

Bernoulli Numbers ($B_i = 0$, odd $i \neq 1$):

$$B_0 = 1,\ B_1 = -\tfrac{1}{2},\ B_2 = \tfrac{1}{6},\ B_4 = -\tfrac{1}{30},$$
$$B_6 = \tfrac{1}{42},\ B_8 = -\tfrac{1}{30},\ B_{10} = \tfrac{5}{66}.$$

Change of base, quadratic formula:

$$\log_b x = \frac{\log_a x}{\log_a b}, \qquad \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Euler's number e:

$$e = 1 + \tfrac{1}{2} + \tfrac{1}{6} + \tfrac{1}{24} + \tfrac{1}{120} + \cdots$$
$$\lim_{n\to\infty}\left(1 + \frac{x}{n}\right)^n = e^x.$$
$$\left(1 + \tfrac{1}{n}\right)^n < e < \left(1 + \tfrac{1}{n}\right)^{n+1}.$$
$$\left(1 + \tfrac{1}{n}\right)^n = e - \frac{e}{2n} + \frac{11e}{24n^2} - O\left(\frac{1}{n^3}\right).$$

Harmonic numbers:

$$1,\ \tfrac{3}{2},\ \tfrac{11}{6},\ \tfrac{25}{12},\ \tfrac{137}{60},\ \tfrac{49}{20},\ \tfrac{363}{140},\ \tfrac{761}{280},\ \tfrac{7129}{2520}, \ldots$$

$$\ln n < H_n < \ln n + 1,$$
$$H_n = \ln n + \gamma + O\left(\frac{1}{n}\right).$$

Factorial, Stirling's approximation:

$$1,\ 2,\ 6,\ 24,\ 120,\ 720,\ 5040,\ 40320,\ 362880, \ldots$$

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right).$$

Ackermann's function and inverse:

$$a(i,j) = \begin{cases} 2^j & i = 1 \\ a(i-1, 2) & j = 1 \\ a(i-1, a(i, j-1)) & i,j \geq 2 \end{cases}$$
$$\alpha(i) = \min\{j \mid a(j,j) \geq i\}.$$

Binomial distribution:

$$\Pr[X = k] = \binom{n}{k} p^k q^{n-k}, \qquad q = 1 - p,$$
$$\mathrm{E}[X] = \sum_{k=1}^{n} k \binom{n}{k} p^k q^{n-k} = np.$$

Poisson distribution:

$$\Pr[X = k] = \frac{e^{-\lambda}\lambda^k}{k!}, \quad \mathrm{E}[X] = \lambda.$$

Normal (Gaussian) distribution:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}, \quad \mathrm{E}[X] = \mu.$$

The "coupon collector": We are given a random coupon each day, and there are n different types of coupons. The distribution of coupons is uniform. The expected number of days to pass before we to collect all n types is

$$nH_n.$$

Probability

Continuous distributions: If

$$\Pr[a < X < b] = \int_a^b p(x)\,dx,$$

then p is the probability density function of X. If

$$\Pr[X < a] = P(a),$$

then P is the distribution function of X. If P and p both exist then

$$P(a) = \int_{-\infty}^{a} p(x)\,dx.$$

Expectation: If X is discrete

$$\mathrm{E}[g(X)] = \sum_x g(x)\Pr[X = x].$$

If X continuous then

$$\mathrm{E}[g(X)] = \int_{-\infty}^{\infty} g(x)p(x)\,dx = \int_{-\infty}^{\infty} g(x)\,dP(x).$$

Variance, standard deviation:

$$\mathrm{VAR}[X] = \mathrm{E}[X^2] - \mathrm{E}[X]^2,$$
$$\sigma = \sqrt{\mathrm{VAR}[X]}.$$

For events A and B:

$$\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$$
$$\Pr[A \wedge B] = \Pr[A] \cdot \Pr[B],$$

iff A and B are independent.

$$\Pr[A|B] = \frac{\Pr[A \wedge B]}{\Pr[B]}$$

For random variables X and Y:

$$\mathrm{E}[X \cdot Y] = \mathrm{E}[X] \cdot \mathrm{E}[Y],$$

if X and Y are independent.

$$\mathrm{E}[X + Y] = \mathrm{E}[X] + \mathrm{E}[Y],$$
$$\mathrm{E}[cX] = c\,\mathrm{E}[X].$$

Bayes' theorem:

$$\Pr[A_i|B] = \frac{\Pr[B|A_i]\Pr[A_i]}{\sum_{j=1}^{n}\Pr[A_j]\Pr[B|A_j]}.$$

Inclusion-exclusion:

$$\Pr\left[\bigvee_{i=1}^{n} X_i\right] = \sum_{i=1}^{n}\Pr[X_i] + \sum_{k=2}^{n}(-1)^{k+1}\sum_{i_i < \cdots < i_k}\Pr\left[\bigwedge_{j=1}^{k} X_{i_j}\right].$$

Moment inequalities:

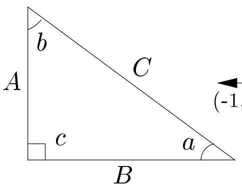
$$\Pr\left[|X| \geq \lambda\,\mathrm{E}[X]\right] \leq \frac{1}{\lambda},$$
$$\Pr\left[\left|X - \mathrm{E}[X]\right| \geq \lambda \cdot \sigma\right] \leq \frac{1}{\lambda^2}.$$

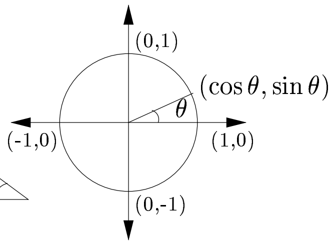
Geometric distribution:

$$\Pr[X = k] = pq^{k-1}, \qquad q = 1 - p,$$
$$\mathrm{E}[X] = \sum_{k=1}^{\infty} kpq^{k-1} = \frac{1}{p}.$$

Theoretical Computer Science Cheat Sheet

Trigonometry





Pythagorean theorem:

$$C^2 = A^2 + B^2.$$

Definitions:

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

Area, radius of inscribed circle:

$$\tfrac{1}{2}AB, \quad \frac{AB}{A+B+C}.$$

Identities:

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

Euler's equation:

$$e^{ix} = \cos x + i \sin x, \qquad e^{i\pi} = -1.$$

`sseiden@acm.org`
`http://www.csc.lsu.edu/~seiden`

Matrices

Multiplication:

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

Determinants: $\det A \neq 0$ iff A is non-singular.

$$\det A \cdot B = \det A \cdot \det B,$$
$$\det A = \sum_{\pi} \prod_{i=1}^{n} \operatorname{sign}(\pi) a_{i,\pi(i)}.$$

2×2 and 3×3 determinant:

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc,$$
$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = g \begin{vmatrix} b & c \\ e & f \end{vmatrix} - h \begin{vmatrix} a & c \\ d & f \end{vmatrix} + i \begin{vmatrix} a & b \\ d & e \end{vmatrix}$$
$$= \begin{matrix} aei + bfg + cdh \\ -\, ceg - fha - ibd. \end{matrix}$$

Permanents:

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

Hyperbolic Functions

Definitions:

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

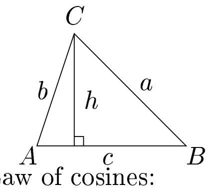
Identities:

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

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

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

More Trig.



Law of cosines:

$$c^2 = a^2 + b^2 - 2ab\cos C.$$

Area:

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

Heron's formula:

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

More identities:

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

Theoretical Computer Science Cheat Sheet

Number Theory

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

$$C \equiv r_1 \bmod m_1$$
$$\vdots \quad \vdots \quad \vdots$$
$$C \equiv r_n \bmod m_n$$

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

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

Fermat's theorem:

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

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

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

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

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

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

Wilson's theorem: n is a prime iff

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

Möbius inversion:

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

If

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

then

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

Prime numbers:

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

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

Graph Theory

Definitions:

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

Geometry

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

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

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

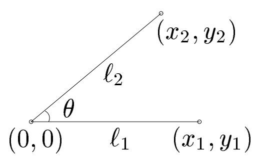
Distance formula, L_p and L_∞ metric:

$$\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2},$$
$$\left[\lvert x_1 - x_0 \rvert^p + \lvert y_1 - y_0 \rvert^p\right]^{1/p},$$
$$\lim_{p \to \infty} \left[\lvert x_1 - x_0 \rvert^p + \lvert y_1 - y_0 \rvert^p\right]^{1/p}.$$

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

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

Angle formed by three points:



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

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

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

Area of circle, volume of sphere:

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

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

Theoretical Computer Science Cheat Sheet

π

Wallis' identity:

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

Brouncker's continued fraction expansion:

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

Gregrory's series:

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

Newton's series:

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

Sharp's series:

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

Euler's series:

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

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

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

Partial Fractions

Let $N(x)$ and $D(x)$ be polynomial functions of x. We can break down $N(x)/D(x)$ using partial fraction expansion. First, if the degree of N is greater than or equal to the degree of D, divide N by D, obtaining

$$\frac{N(x)}{D(x)} = Q(x) + \frac{N'(x)}{D(x)},$$

where the degree of N' is less than that of D. Second, factor $D(x)$. Use the following rules: For a non-repeated factor:

$$\frac{N(x)}{(x-a)D(x)} = \frac{A}{x-a} + \frac{N'(x)}{D(x)},$$

where

$$A = \left[\frac{N(x)}{D(x)}\right]_{x=a}.$$

For a repeated factor:

$$\frac{N(x)}{(x-a)^m D(x)} = \sum_{k=0}^{m-1} \frac{A_k}{(x-a)^{m-k}} + \frac{N'(x)}{D(x)},$$

where

$$A_k = \frac{1}{k!}\left[\frac{d^k}{dx^k}\left(\frac{N(x)}{D(x)}\right)\right]_{x=a}.$$

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

Calculus

Derivatives:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Integrals:

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

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

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

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

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

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

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

Theoretical Computer Science Cheat Sheet

Calculus Cont.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Theoretical Computer Science Cheat Sheet

Calculus Cont.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Finite Calculus

Difference, shift operators:

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

Fundamental Theorem:

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

Differences:

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

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

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

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

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

Sums:

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

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

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

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

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

Falling Factorial Powers:

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

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

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

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

Rising Factorial Powers:

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

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

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

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

Conversion:

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

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

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

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

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

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

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

Theoretical Computer Science Cheat Sheet

Series

Taylor's series:

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

Expansions:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Ordinary power series:

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

Exponential power series:

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

Dirichlet power series:

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

Binomial theorem:

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

Difference of like powers:

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

For ordinary power series:

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

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

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

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

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

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

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

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

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

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

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

Convolution:

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

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

Theoretical Computer Science Cheat Sheet

Series

Expansions:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

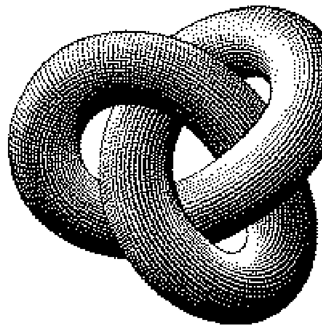
$$\left(\frac{1-\sqrt{1-4x}}{2x}\right)^n = \sum_{i=0}^{\infty} \frac{n(2i+n-1)!}{i!(n+i)!} x^i,$$

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

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

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

Escher's Knot



Stieltjes Integration

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

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

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

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

If the integrals involved exist

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

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

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

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

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

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

Cramer's Rule

If we have equations:

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

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

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

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

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

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

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

Fibonacci Numbers

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

Definitions:

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

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

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

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:

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

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