1 Mathematics

1.1 Divisibility

1.1.1 Algorithm

```
// calculates (alternating) k-digitSum for integer number given by M
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;
}
```

1.1.2 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.

D	3	7	9	11	13	17	19	23	37	41
k	1	3	1	2	3	8	9	11	3	5
alt	f	w	f	f	w	w	w	w	f	f

1.2 Binomial Coefficient

1.2.1 Algorithms

```
// binomial coefficient 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;
}
```

Time Complexity: $\mathcal{O}(NK)$

```
// 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;
  }
}
```

Time Complexity: $\mathcal{O}(k)$

1.2.2 Properties

1. $\binom{n}{k} = \dfrac{n!}{k!(n-k)!}$ 2. $\binom{n}{k} = \binom{n}{n-k}$ 3. $\binom{n+1}{k+1} = \binom{n}{k} + \binom{n}{k+1}$ 4. $\sum_{k=0}^{n} \binom{n}{k} = 2^n$

1.3 Combinatorics

- Variations (ordered): k out of n objects (permutations for $k = n$)
  - without repetition:
    $M = \{(x_1, \ldots, x_k) : 1 \le x_i \le n,\ x_i \ne x_j \text{ if } i \ne j\}$, $|M| = \frac{n!}{(n-k)!}$
  - with repetition:
    $M = \{(x_1, \ldots, x_k) : 1 \le x_i \le n\}, |M| = n^k$
- 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): $!n = n! \sum_{k=0}^{n} \frac{(-1)^k}{k!} = \lfloor \frac{n!}{e} + \frac{1}{2} \rfloor$

1.4 Polynomial Interpolation

1.4.1 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: $p(x) = \sum_{i=0}^{n} \gamma_{0,i} \prod_{j=0}^{i-1} (x - x_i)$ 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 $b_0 = p(x)$.

1.4.2 Algorithms

```
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++) {
```

```java
      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));
  }

}
```

```java
// implementation of rational numbers
class rat {

  public long c;
  public long d;

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

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

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

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

  public static rat[] commonDenominator(rat[] c) {
    long kgV = 1;
    for (int i = 0; i < c.length; i++) {
```

```
      kgV = kgV(kgV, c[i].d);
    }
    for (int i = 0; i < c.length; i++) {
      c[i].c *= kgV/c[i].d;
      c[i].d *= kgV/c[i].d;
    }
    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));
  }

}
```

1.5 Fibonacci Sequence

1.5.1 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 $\tilde{\phi} = \frac{1-\sqrt{5}}{2}$.

1.5.2 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 $g_0, g_1 \in \mathbb{N}_0$

1.5.3 Pisano Period

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

k	2	3	4	5	6	7	8	9	10	100	10^n for $n > 2$
$\pi(k)$	3	8	6	20	24	16	12	24	60	300	$15 \cdot 10^{n-1}$