Chapter 5: Mathematics

Michaël Bradet-Legris

Table of Contents

Java BigInteger

- Many useful built-in functions that can save time in a competition
- a.mod(BigInteger b) → $a \bmod b$
- a.gcd(b) → $gcd(a, b)$
- a.modPow(b, c) → $a^b \bmod c$
- a.inverseMod(n) → $a^{-1} \bmod n \ \ (a * a^{-1} = 1 \bmod n)$
- a.toString(b) → $a \text{ in base } b \text{ representation}$
- a.isProbablePrime(int certainty) → a is prime with probability (1- $\frac{1}{2^n}$)
  - If we want to compute k primes with p certainty (total), want to use:

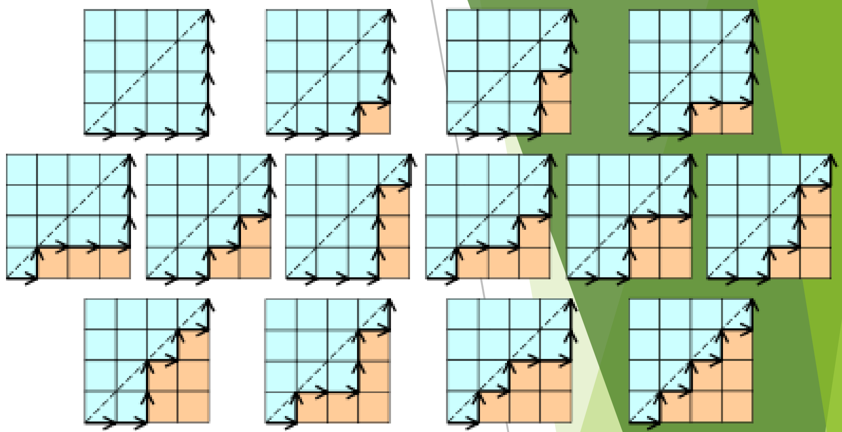
$$n = \mathrm{ceil}\left(-\frac{\ln(1-p^{\frac{1}{k}})}{\ln(2)}\right)$$

Combinatorics: Combinations / Permutations

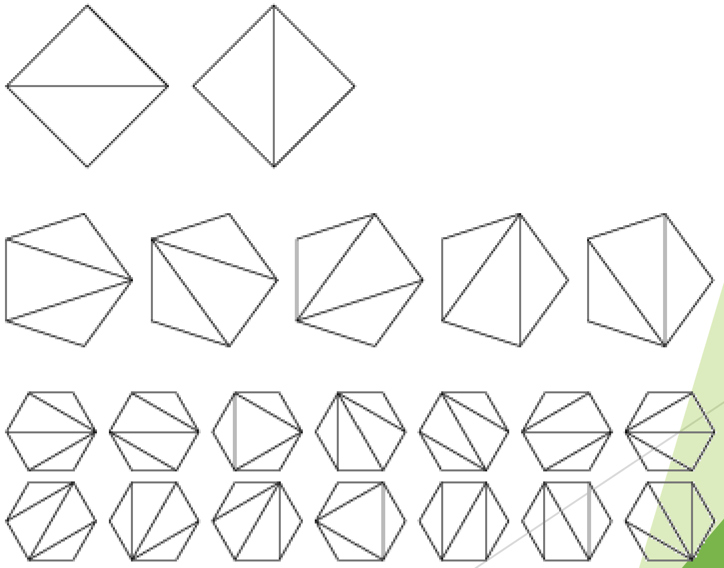
- Ways to choose k elements from n total elements?
  - Order doesn't matter? $\frac{n!}{n-k!}$
  - Order matters? $\frac{n!}{k!(n-k)!}$ = $\binom{n}{k}$

- Duplicate elements? Divide by the factorial of the multiplicity of each (chosen) element. Gets messy if we don't choose all the elements.
- eg// How many 11 letter words can be made with the letters MISSISSIPPI?
  - $\frac{11!}{1!2!4!4!}$

Combinatorics: Catalan Numbers



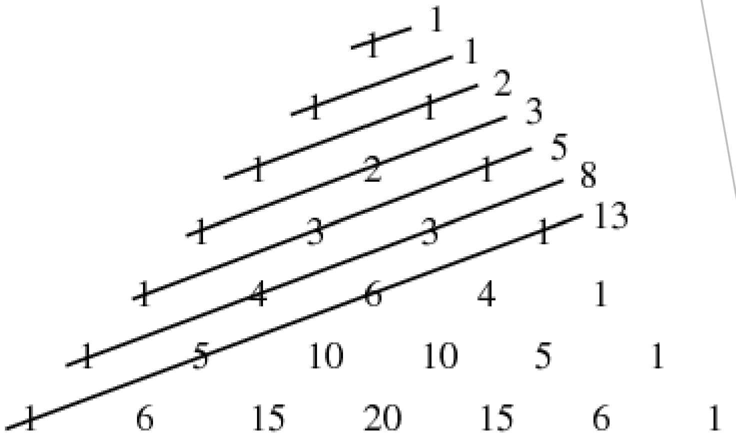
- What is Cat(n)? It's…
  - The number of binary trees with n nodes
  - Number of ways to triangulate convex polygon with n+2 sides
  - Number of monotonic diagonal paths on n x n grid
    (below the diagonal)



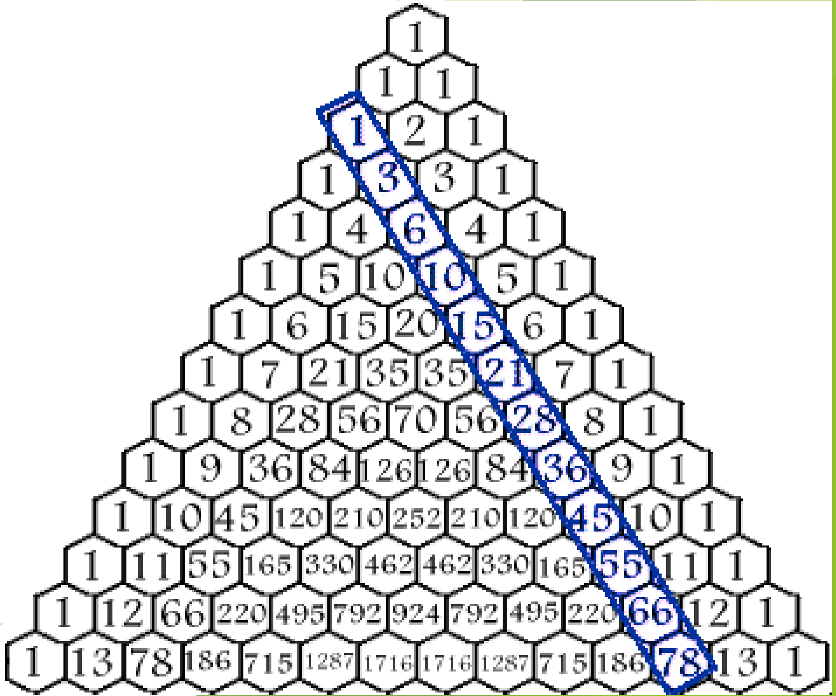
- Two ways to compute:
  - Closed form: $C(n) = \binom{2n}{n} \big/ _{n+1}$
  - Recursive (can use DP): $C(n+1) = \frac{(2n+2)(2n+1)}{(n+2)(n+1)} C(n)$

Combinatorics: Binomial Coefficients

- Given $(x + y)^p$, what is the coefficient of $x^k y^{p-k}$?
  - $\binom{n}{k}$
  - DP approach: Pascal's Triangle



- Given a probability p of an event occurring, what is the probability that this event happens exactly k times out of n?
  - Consider the above equation and let $x = p$, $y = (1 - p)$. We get:
  - $\binom{n}{k} p^k (1 - p)^{p-k}$



Number Theory: GCD / LCD

- GCD : Greatest Common Divisor (Integers)
  - For n < m, gcd(n, m) = gcd(n, m mod n). Can do this recursively.
  - Used for other computations, like Chinese Remainder Thm, Extended Euclidean Algorithm & LCM
  - Can use BigInteger.gcd(), but this is slower.
- LCM : Least Common Multiple (Integers)

```
//n < m
int gcd(int n, int m){
   if (n == 0)
      return m;
   return gcd(m nod n, n);
}
```

```
int lcm (int n, int m){
   return (n*m)/gcd(n,m);
}
```

Number Theory: Primality / Prime Factorization

- How do we check if a number n is prime?
  - Check all the odd numbers (and 2) between 2 and $\sqrt{n}$. If none divide n, then n is prime.
  - If we are computing all primes and storing them, can check all primes between 2 and $\sqrt{n}$ instead.

- Sieve of Eratosthenes : Optimized for generating all primes from 1 to n
  - Initialize an array p[n+1]. Initialize k as 2.
  - Starting at k^2, set $p[k^2] = 1$ (not prime) and keep incrementing by k, setting the $p[k^2+mk]$ as 1 (for $k^2+mk < n$).
  - Walk along the array starting at k until you find a prime number ($p[k'] = 0$).
  - If $(k')^2 < n$, loop again. Otherwise, stop.
  - When finished, may want to loop through all elements once more and put all the primes in an ArrayList.

Number Theory: Primality / Prime Factorization

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

Prime numbers

Number Theory: Linear Diophantine / Extended Euclidian Alg.

- Motivating problems:
  - Given a single equation $ax + by = c$, how do we solve for x and y, given that they are integers? (Linear Diophantine Equation)
  - What is the inverse of a mod b? (Does it exist?)
- Extended Euclid's Algorithm:
  - Given a and b, finds values x,y such that ax + by = gcd(a, b)
  - There are NO solutions for ax + by = c if c is not a multiple of gcd(a, b).
- a inverse mod b : If gcd(a, b) = 1, then can find ax + by = 1. mod this by b to get rid of the second term and you are left with $ax = 1\ (mod\ b)$
- Linear Diophantine: Let c = k*gcd(a, b). Then:
  - Solve for ax + by = gcd(a, b)
  - multiply by k to get : a(xk) + b(yk) = c
  - Can do math-y stuff and show that if $ax_1 + by_1 = c$, all other solutions are of the form $x = x_1 - n\frac{b}{\gcd(a,b)}$ and $y = y_1 - n\frac{a}{\gcd(a,b)}$ for some n.

Number Theory: Linear Diophantine / Extended Euclidian Alg.

STEP 2: EXPRESS 1 AS THE DIFFERENCE BETWEEN MULTIPLES OF 3000 AND 197

$$3000 = 15(197) + 45$$
$$197 = 4(45) + 17$$
$$45 = 2(17) + 11$$
$$17 = 1(11) + 6$$
$$11 = 1(6) + 5$$
$$6 = 1(5) + 1$$

$$\begin{aligned} 1 &= 6 - 1(5) \\ &= 2(6) - 1(11) \\ &= 2(17) - 3(11) \\ &= 8(17) - 3(45) \\ &= 8(197) - 35(45) \\ &= 533(197) - 35(3000) \end{aligned}$$

Number Theory: Chinese Remainder Thm

- Given a set of relatively prime (coprime) numbers $p_1, p_2 \ldots p_n$ (ie, for all $1 \le i,j \le n$, $\gcd(p_i, p_j) = 1$), for any numbers $k_1, k_2 \ldots k_n$, there is exactly one solution to $\begin{cases} x = k_1 \bmod p_1 \\ x = k_2 \bmod p_2 \\ \quad\ldots \\ x = kn \bmod pn \end{cases}$ for $1 \le x \le p_1 p_2 \ldots p_n$

- eg// Since 4 and 5 are coprime, there is a solution to $\begin{cases} x = 1\, mod\, 4 \\ x = 2\, mod\, 5 \end{cases}$

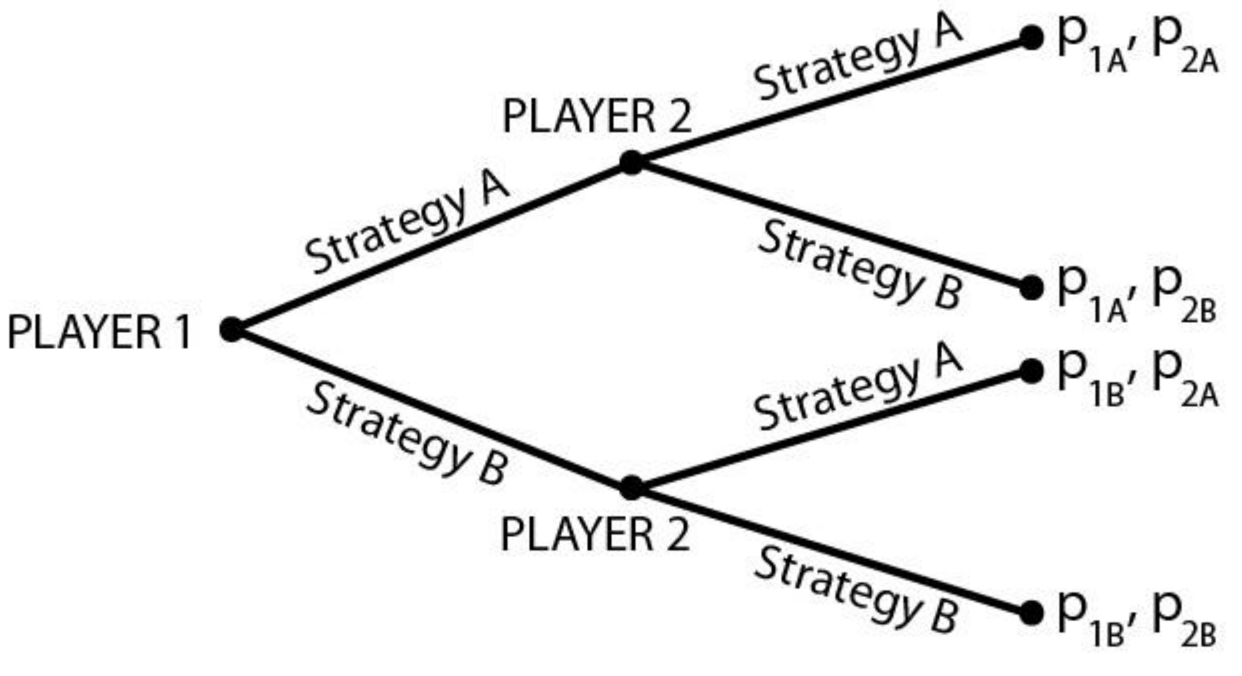
- How do we find a solution to $\begin{cases} x = a\, mod\, p \\ x = b\, mod\, q \end{cases}$?
  - $x = q * (q^{-1} \text{ mod p}) * \text{a} + \text{p} * (p^{-1}\, mod\, q) * \text{b}$
  - To find the inverses, note that since gcd(p, q) = 1, we can use Exteneded Euclid.
  - This generalizes to higher dimensions.

Number Theory: Other useful formulas

- Num Divisors: Given a number n, how many divisors does it have?
  - First, prime factorize it: $n = p^{a1}p^{a2}\ldots p^{an}$
  - Add 1 to each of the multiplicities and multiply them together:
  - n has $(a_1 + 1)(a_2 + 1)\ldots(a_n + 1)$ divisors.
  - eg// $10800 = 2^4 3^3 5^2$ has $(4 + 1)(3 + 1)(2 + 1) = 60$ factors.

- Euler's Totient Function: Counts the number of integers relatively prime to N from 1 to N-1.
  - $\phi(N) = N(\prod_{PF}(1 - \frac{1}{PF}))$, where PF are the prime factors of N.
  - eg// N = 17 has 16 coprime values between 1…N-1 (since it's prime).
  - eg// 36 has $36(1 - \frac{1}{2})\left(1 - \frac{1}{3}\right) = 12$ coprime values between 1..35.

Game Theory



- Most game theory problems can be solved one of three/four ways:
- Dynamic Programming
  - 2 sticks game
- Tree Traversals / Backtracking / Decision Trees
- Pattern Finding (may or may not be some sort of mathematical formula)
- Probability (You make your decisions without knowing your opponent's decisions)
  - Rock-Paper-Scissors-Lizard-Spock

Kattis: Pseudoprime Numbers
Difficulty : 3.8

- Fermat's Little Theorem: for $a \geq 0$, if p is a prime then $a^p \equiv a \pmod{p}$.
- This sometimes works even if p is not a prime. In this case, p is a base-a pseudoprime.
- Input: p, a values where $2 \leq p \leq 1\,000\,000\,000$, $1 < a < p$. Terminated by 0 0.
- Output: "yes" or "no" (if p is a base-a pseudoprime)

Sample Input:	Sample Output:
3 2	no
10 3	no
341 2	yes
341 3	no
1105 2	yes
1105 3	yes
0 0	

```java
public static void main(String[] args) {

  Scanner sc = new Scanner(System.in);

  while (sc.hasNextLine()){
    String line = sc.nextLine();
    line.trim();
    if (line.equals("0 0"))
      break;

    String[] numbers = line.split(" ");

    BigInteger p = new BigInteger(numbers[0]);
    BigInteger a = new BigInteger(numbers[1]);

    if (p.isProbablePrime(20) == true)
      System.out.println("no");
    else if (a.modPow(p,p).equals(a))
      System.out.println("yes");
    else
      System.out.println("no");
  }
}
```

Kattis: Bobby's Bet
Difficulty: 3.5

- Roll an S-sided Y times and want at least X of them to be ≥ R.
- If his return is W times his bet if this happens, should he take the bet or not?
- 1 ≤ N ≤ 10 000 test cases
- Input: R S X Y W, 1 ≤ R ≤ S ≤ 20, 1 ≤ X ≤ Y ≤ 10, 1 ≤ W ≤ 100
- Output: "yes" or "no", if he should take the bet.

Sample Input:
2
5 6 2 3 3
5 6 2 3 4

Sample Output:
no
yes

Kattis: Prime Reduction
Difficulty: 4.2

- Write a function that does these four things:
  - If x is prime, stop.
  - Prime factor x.
  - Call the function again with n = sum of distinct primes of x.
- Input: ≤ 20 000 test cases, each an integer between 2 and 10^9. Terminated by 4.
- Output: The number that was a parameter to the last function call and the number of function calls.

Sample Input:
2
3
5
76
100
2001
4

Sample Output:
2 1
3 1
5 1
23 2
5 5
5 6

Kattis: Divisors
Difficulty: 4.8

- How many divisors does $\binom{n}{k}$ have?
- Input: at most 11 000 lines of $0 \leq k \leq n \leq 431$
- Output: the number of divisors (does not overflow 64 bits)

Sample Input:	Sample Output:
5 1	2
6 3	6
10 4	16

Kattis: Chinese Remainder Difficulty: 4.7

- Input: $1 \leq T \leq 1000$ test cases. Each line : a n b m, $1 \leq n, m \leq 10^9$, $0 < b < m$, $0 < a < n$.
- Output: $x < nm$ and nm, where x satisfies $\begin{cases} x = a \ (mod\ n) \\ x = b \ (mod\ m) \end{cases}$

Sample Input:	Sample Output:
2	5 6
1 2 2 3	31471 217674
151 783 57 278	

```java
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int cases = sc.nextInt();
    for(int i = 0; i < cases; i++){
        int a = sc.nextInt();
        int n = sc.nextInt();
        int b = sc.nextInt();
        int m = sc.nextInt();

        BigInteger M = new BigInteger("" + m);
        BigInteger N = new BigInteger("" + n);
        BigInteger A = new BigInteger("" + a);
        BigInteger B = new BigInteger("" + b);
        BigInteger MN = M.multiply(N);
        BigInteger invM = M.modInverse(N);
        BigInteger invN = N.modInverse(M);

        BigInteger modNPart = M.multiply(invM).multiply(A).mod(MN);
        BigInteger modMPart = N.multiply(invN).multiply(B).mod(MN);

        BigInteger ans = modNPart.add(modMPart).mod(MN);

        System.out.println(ans.toString() + " " + MN.toString());
    }
}
```

Kattis: The Magical 3
Difficulty: 5.4

- Input: ≤ 1000 lines, each with a single positive integer n. Terminates with ‘0’. n fits in a 32-bit int.
- Output: The smallest base b for which n ends with 3 in base b, or “No such base”.

Sample Input:	Sample Output:
11	4
123	4
104	101
2	No such base
3	4
0	

Kattis: List of Mathematics related problems

- Almost Perfect
- Candy Division
- Crypto
- Divisors
- Factovisors
- Fareysums
- Fundamental Neighbors
- Goldbach2
- Industrial Spy
- Catalan Numbers

Kattis: List of Mathematics related problems

- List game
- Magical 3
- Perfect Powers
- Primal
- Prime Path
- Primes
- Primal Presentation
- Prime Reduction
- Primes 2
- Catalan Square

Kattis: List of Mathematics related problems

- Pseudoprime
- Relatives
- Smallest Multiple
- Farey
- CPU
- CPU2
- Happy Prime
- LCM Pair Sum
- Number Set Easy

Kattis: List of Mathematics related problems

- Number Set Hard
- List Game 2
- PXS
- Bakterjie
- Chinese Remainder
- General Chinese Remainder
- Heliocentric
- Radar
- Substitution

References

- Wikipedia
- Competitive Programming 3 – Stephen Halim
- https://www.youtube.com/watch?v=ru7mWZJlRQg
- https://www.youtube.com/watch?v=mgvA3z-vOzc