Software Engineering II (PAR)

Training - Concurreny

Überblick	
01 // row(i) = column(i)	34 // Maximize log(s) / log(maxPrimeFactors(s))
02 // Sum of primes	35 // n = a & b = a + b^3
03 // $n^2 + q^2 = P^2 + Q^2$	36 // SPrimeFactors(n) = SPrimeFactors(reverse n)
04 // Honaker & Jud. McCranie Puzzle	37 // Primes ending in 9
05 // Smith Number	38 // Same quantity of prime factors
06 // Gronau Prime Triples	39 // Consecutive primes
07 // 414347	40 // Filling the gap
08 // Edwin Clark Puzzle	41 // Zeisel numbers
09 // Extended Ruth Aaron Pairs	42 // 493009335
10 // Andersen Primes	43 // Grimm's conjecture
11 // Distinct, Increasing, Decreasing Prime Gaps	44 // p = 2 * p1 + 3*p2; p1, p2 are distinct odd primes
12 // Primonacci	45 // Prime gaps
13 // Odd midpoints	46 // Smallest Prime and Perfect Square
14 // Chain of twins	47 // Taxicab – Prime numbers
15 // Dropping-Digits Primes	48 // Travelling intertwined primes
16 // Trotter's Curio	49 // Primes with zeros replaced
17 // Concatenating k consecutive integers	50 // Primes by duplicating some digit
18 // Wagon prime	51 // 2213
19 // First 13 numbers	52 // 10958
20 // Primes reversing n and n^k	53 // Primes and convolution
21 // Queen covering chessboard	54 // Primes between perfect powers
22 // Multiple or divisor	55 // Consecutive integers and prime factors
23 // First n primes in a circle	
24 // 123456789	
25 // Three consecutive primes a prime	
26 // Smallest prime concatenating first k primes	
27 // Smallest set of distinct primes	
28 // Special set of binary numbers	
29 // Cons. integers with prime sum of prime factors	
30 // Primes embedded	
31 // 6 * 7 * 8 * 9 - 1 is prime	
32 // Vector of primes that generates distinct primes	
33 // Prime Tuples	

01 // row(i) = column(i)

Conditions:	(i) row(i) = column(i)	**Example:**	1013
	(ii) sum of the n-1 rows is equal to the row(n)	(n = 4)	0191
	(iii) each row is a prime number		1933
Aufgabe:	Parallelisiert ist eine Lösung für n = 6 und n = 8 zu ermitteln.		3137

02 // Sum of primes

Composite numbers equal to the sum of primes from the least prime factor to the largest prime factor.

Example: 39 = 3 * 13 = 3 + 5 + 7 + 11 + 13

Aufgabe: Parallelisiert sind weitere Lösungen zu ermitteln.

03 // $n^2 + q^2 = P^2 + Q^2$ **BigInteger**

p and P is a reservible pair; q and Q is a reservible pair; p, q, P and Q are primes.

Example: p = 102061, P = 160201, q = 335113, Q = 311533

$102061^2 + 335113^2 = 160201^2 + 311533^2$

Aufgabe: Parallelisiert sind weitere Lösungen zu ermitteln.

04 // Honaker & Jud. McCranie Puzzle

30103 is the only known multi-digit palindromic prime found by averaging the divisors of a composite number. 30103 = average divisors of 149645. The divisors of 149645 are 1, 5, 173, 865, 29929 and 149645. 30103 = (1 + 5 + 173 + 865 + 29929 + 149645) / 6.

Aufgabe: Parallelisiert sind weitere Zahlen mit dieser Eigenschaft zu ermitteln.

05 // Smith Number

Sum of its digits is the sum of its digits of all its prime factors.

Example: 666 = 2 * 3 * 3 * 37 & 6 + 6 + 6 = 2 + 3 + 3 + (3 + 7)

Aufgabe: Parallelisiert sind weitere Lösungen zu ermitteln.

Pollard Rho Algorithm: http://www.sanfoundry.com/java-program-pollard-rho-algorithm/

06 // Gronau Prime Triples **BigInteger**

Triples (p,q,r) of primes, $p < q < r$, such that: $p * q + r = x^2$, $p * r + q = y^2$, $q * r + p = z^2$

Examples: (3,577,10369), (17,19,2593)

Aufgabe: Parallelisiert sind weitere Lösungen zu ermitteln.

Three subsequent primes (41, 43 & 47) in their concatenated form (414347) - and its evenly decomposed sums 414 + 347, 41 + 43 + 47, 4 + 1 + 4 + 3 + 4 + 7 are all primes.

Aufgabe: Parallelisiert sind weitere Lösungen zu ermitteln.

08 // Edwin Clark Puzzle

Call a list (p[1],p[2],...,p[n] of primes p[i] permutable if the concatenations
of all of the n! permutations of the primes in the list are distinct primes.

Examples: n = 2 : [3,7]; 37, 73 are distinct primes.
n = 3 : [7,11,43]; 71143, 74311, 11743, 11437, 43711, 43117 are distinct primes.

Aufgabe: Parallelisiert sind weitere Lösungen für n = 2,3,4 zu ermitteln.

09 // Extended Ruth Aaron Pairs

If the sum of prime factors of two consecutive numbers is also consecutive.

Example: n = 20, prime factors are [2,2,5], sum of divisors is 9.
n + 1 = 21, prime factors are [3,7], sum of divisors is 10.
→ (20,21) is an Extended Ruth Aaron Pair

Aufgabe: Parallelisiert sind weitere Lösungen zu ermitteln.
Pollard Rho Algorithm: http://www.sanfoundry.com/java-program-pollard-rho-algorithm/

10 // Andersen Primes

n is a ppn if the reversal of n is equal to the sum of the reversals of the proper divisors n

Examples: n = 10311; proper divisors: 1, 3, 7, 21, 491, 1473, 3437
11301 = 1 + 3 + 7 + 12 + 194 + 3741 + 7343 ► 10311 is ppn

Aufgabe: Parallelisiert sind weitere Lösungen zu ermitteln.

11 // Distinct, Increasing, Decreasing Prime Gaps

Gap between consecutive primes is defined $g_i = p_{i+1} - p_i$. All the gaps are even numbers.

a) n distinct consecutive gaps
b) n increasing consecutive gaps
c) n decreasing consecutive gaps

Examples: **a** : n = 2 : 2 (1) 3 (2) 5; n = 3 : 17 (2) 19 (4) 23 (6) 29
b : n = 2 : 2 (1) 3 (2) 5; n = 3 : 19 (2) 19 (4) 23 (6) 29
c : n = 2 : 7 (4) 11 (2) 13; n = 3 : 31 (6) 37 (4) 41 (2) 43

Aufgabe: Parallelisiert sind für n = [2..10] Lösungen zu ermitteln.

12 // Primonacci **BigInteger**

Let S be a sequence of prime numbers such that S(1) = a, S(2) = b, S(n+1) = S(n) + S(n-1) + c, where a < b and c = odd > c, for 1 ≤ n ≤ k.

Example: 10331, 67073, 109619, 208907, 350741, 591863, 974819, 1598897, 2605931, 4237043, 6875189, 11144447, 18051851, 20228513, 47312579, 76573307, 123918101, 200523623, 324473939

Aufgabe: Parallelisiert sind weitere Lösungen zu ermitteln.

13 // Odd midpoints

The four consecutive primes 199, 211, 223, and 227 have midpoint 205, 217, 225,
all odd numbers.

Aufgabe: Parallelisiert ist die längste Sequenz zu ermitteln.

14 // Chain of twins

Twin primes that when added their sum is equal to the number 'in the middle' of other two larger primes. (5,7) → (11,13) form a chain of size equal to 2. (5,7); 5 + 7 = 12; 12 is 'in the middle' of 11 and 13. 11 + 13 = 24 and 24 is not in the middle of twin primes (23 is prime but 25 is not prime), so the chain ends here.

Aufgabe: Parallelisiert sind Ketten von Primzahl-Zwillingen zu ermitteln.

15 // Dropping-Digits Primes **BigInteger**

Least prime such that it remains prime after all occurrences of each digit 'd' are dropped at the same time, for each distinct digit of the original prime. No leading zeroes are allowed.

Example: Original prime 1210778071 with k = 5 distinct digits (0, 1, 2, 7 and 8).

0: 12177871 (prime); 1: 2077807 (prime); 2: 110778071; 7: 1210801; 8: 121077071

Aufgabe: Parallelisiert sind für k = [2..6] die Lösungen zu ermitteln.

16 // Trotter's Curio

p is the least prime. n is the count of numbers added. Trotter shows the following property of prime 11: Begin with 11, and continually add the first five powers of 2, but in reverse order (32, 16, 8, 4 and 2). All the sums are primes (43, 59, 67, 71 and 73). (n,p) → (5,11) with least p is the solution.

Aufgabe: Parallelisiert sind für n = [1..7] die Lösungen zu ermitteln.

17 // Concatenating k consecutive integers — BigInteger

Least prime formed by the concatenation of k consecutive 2-digit integers in,
a) ascending order, b) descending order.

Example: a) ascending order, k = 4: 4567; b) descending order, k = 7: 73727170696867

Aufgabe: Parallelisiert sind für k = [4,6,8,10,12,14,16] die Lösungen zu ermitteln.

18 // Wagon prime — BigInteger

Prime p with maximum digits 11, such that every k consecutive digits inside p is a distinct prime p_i.

Example: For k = 2, p = 411379717319 is a prime number, and every two consecutive digits inside p is a distinct prime: 41, 11, 13, 37, 79, 97, 71, 17, 73, 31 and 19.

Aufgabe: Parallelisiert sind für k = [2,3,4] die Lösungen zu ermitteln.

19 // First 13 numbers — BigInteger

Find a solution to $1^p + 2^p + 3^p + 4^p + 5^p + 6^p + 7^p + 8^p + 9^p + 10^p + 11^p + 12^p + 13^p =$
$s^r * (1^q + 2^q + 3^q + 4^q + 5^q + 6^q + 7^q + 8^q + 9^q + 10^q + 11^q + 12^q + 13^q)$
where p, q, r and s are primes.

Example: (p,q,r,s) = (5,3,2,11)

Aufgabe: Parrallelisiert sind weitere Lösungen zu ermitteln.

20 // Primes reversing n and n^k

Find an integer n such that when reverse the digits of n → prime, reverse the digits of n^2 → prime, reverse the digits of n^k → prime.

Example: k = 2; n = 14; 41 → prime; 14^2 = 196 → 691

Aufgabe: Parallelisiert sind für k = [3,4,5] die Lösungen zu ermitteln.

21 // Queen covering chessboard

Determine a path to cover all the cells of chessboard going from the cell c5 to the cell d4 in exactly 14 legal movements of the queen.

Example: c5-f8-c8-h3-b3-g8-g3-b8-b2-g2-a8-a1-h1-h8-d4

Aufgabe: Parallelisiert sind weitere gültige Pfade zu ermitteln.

22 // Multiple or divisor

Find the maximal arrangement in a row of the first n integers in such a way that every integer is a multiple or a divisor of its two immediate neighbours, except the ends which have only one neighbour.

Example: (n = 100) 93-31-62-1-87-29-58-2-46-92-23-69-3-57-19-38-76-4-68-34-17-85-5-55-11-99-33-66-22-44-88-8-32-64-16-80-40-20-60-30-90-45-15-75-25-50-100-10-70-35-7-49-98-14-56-28-84-42-21-63-9-81-27-54-1836-72-12-24-96-48-6-78-39-13-26-52.
(Anzahl der Zahlen: 77)

Aufgabe: Parallelisiert sind weitere Anordnungen zu ermitteln und die Anzahl zu maximieren.

23 // First n primes in a circle

Arrange the first n primes in a circular pattern such that every two adjacent primes and every pair of diagonally opposite primes do not have any common digit.

Example:
```
(n = 28)   53 61 79 103 29 41 59  67 13 47 89 23 107 43
           17  3 11  97 31  2 37 101  7  5 73 19  83 71
```

Aufgabe: Parallelisiert sind für n = [29,30,31,32] die Lösungen zu ermitteln.

24 // 123456789

$123456789 = ((86 + 2 \times 7)^5 - 91) / 3^4$.

This expression:

a) uses the same 9 digits in both parts of the equality, only once in each side.

b) uses all the four arithmetical basic operators and the power operator, plus parentheses.

c) uses four (4) primes: 2, 3, 5, & 7 and three composites 4, 86 & 91.

Aufgabe: Parallelisiert ist für 123456789 ein arithmetischer Ausdruck mit mehr als vier Primzahlen zu ermitteln.

25 // Three consecutive primes a prime

Example: (i) 7 + 11 + 13 = 31 → prime, (ii) 31 + 37 + 41 = 109,
(iii) 109 + 113 + 127 = 349 → prime, (iv) 349 + 353 + 359 = 1061
(v) 1061 + 1063 + 1069 = 3193 → composite (fail).

Aufgabe: Parallelisiert ist ein Pfad mit mindestens fünf erfolgreichen Schritten zu ermitteln.

26 // Smallest prime concatenating first k primes

Examples: k = 1: 2; k = 2: 23; k = 3: 523

Aufgabe: Parallelisiert sind für k = [4..9] die Lösungen zu ermitteln.

27 // Smallest set of distinct primes

Find a set with minimum sum of distinct primes, such that in the set there are exactly one 1, two 2s, three 3s,..., nine 9s.

Example: (5, 7, 29, 47, 59, 61, 67, 79, 83, 89, 269, 463, 467, 487, 569, 599, 859, 883, 887)

Aufgabe: Parallelisiert sind für 10-stellige Primzahlen weitere Lösungen zu ermitteln.

28 // Special set of binary numbers

Let p a prime. Consider the set of numbers sp whose binary representation has p 1's and one zero. This set has p-1 odd numbers.

Example: p = 5; s5 = (111110, 111101, 111011, 110111, 101111) → s5 = (62, 61, 59, 55, 47)
s5 contains three primes.

Aufgabe: Parallelisiert ist bis p = 100000 zu prüfen, ob sp mindestens eine Primzahl enthält.

29 // Consecutive integers with prime sum of prime factors

Example: 197, 198, 199. The factors extended for 198 are 2 * 3 * 3 * 11 with sum 19. 197 and 199 are prime, these three (k = 3) consecutive numbers each have the sum of their prime factors being a prime.

Aufgabe: Parallelisiert sind für k = [4,5,6,7,8,9,10,11] die Lösungen zu ermitteln.
Pollard Rho Algorithm: http://www.sanfoundry.com/java-program-pollard-rho-algorithm/

30 // Primes embedded

Earliest number having k distinct primes embedded.

Example: k = 5: earliest number 137; five distinct primes: 3, 7, 13, 37 and 137.

Aufgabe: Parallelisiert sind für k = [1..10] die Lösungen zu ermitteln.

31 // 6 * 7 * 8 * 9 - 1 is prime

Examples: 6 * 7 * 8 * 9 - 1 = 3023 (prime); 7 * 8 * 9 * 10 - 1 = 5039 (prime);
8 * 9 * 10 * 11 - 1 = 7919 (prime); 9 * 10 * 11 * 12 - 1 = 11879 (composite)

Aufgabe: Parallelisiert ist ein Pfad mit mindestens vier Primzahlen zu ermitteln.

32 // Vector of primes that generates distinct primes

A prime vector of order n is an array of distinct primes P = (p_0, p_1,...,p_{n-1}), such that every sum of an odd number of consecutive elements is also prime.

Example: (239, 131, 109, 181, 83, 43, 41, 223, 53, 233, 271, 103,
(23 elements) 269, 71, 19, 47, 241, 23, 277, 199, 281, 29, 37) = 121 distinct primes

Aufgabe: Parallelisiert ist ein Vektor mit mindestens 24 Elementen zu ermitteln.

33 // Prime Tuples

A prime tuple of order n (odd) with length k is an array of distinct odd primes (p_0, p_1,...,p_{k-1}), such that every term after the n-th term is the sum of the previous n terms. The weight of a prime tuple of order n is the sum of its first n terms. When two tuples of the same order have the same length, then the one with the smaller weight is preferred.

Example: n = 3; primes = 3, 13, 7; k = 7; w = 23; primes generated (k-n) = 4

Aufgabe: Parallelisiert sind für k = [5,7,9,11] die Lösungen zu ermitteln.

34 // Maximize log(s) / log(maxPrimeFactors(s)) **BigInteger**

Maximize log(s) / log(maxPrimeFactors(s)) where s = sigma(p^x), p is prime and x >= 2.

Example: s = sigma(10889273^5) = 153106796458831178584646446634328654 (divisor sum)
factor(sigma(10889273^5)) = $2 * 3^2 * 7 * 11^2 * 19^3 * 53 * 73 * 109 * 163 * 181 * 211 * 223 * 229 * 283 * 307 * 337 * 373$
log(s) / log(maxPrimeFactors(s)) =
log(153106796458831178584646446634328654) / log(373) = 13,6816 (result).

Aufgabe: Parallelisiert ist eine Lösung mit einem maximalen Ergebnis (result) zu ermitteln.

35 // n = a & b = a + b^3

Examples: (i) 568 = 56 + 8^3; (ii) 86044 = 860 + 44^3

Aufgabe: Parallelisiert sind Lösungen zu ermitteln, wobei n eine Primzahl ist.

36 // SPrimeFactors(n) = SPrimeFactors(reverse n)

Example: n = 45 has a sum of prime factors 3 + 3 + 5 = 11; reverse the order of the digits to get reverse n = 54 which also has sum of factors 3 + 3 + 3 +2 = 11.

Aufgabe: Parallelisiert sind Lösungen zu ermitteln, wobei n zu maximieren ist.
Pollard Rho Algorithm: http://www.sanfoundry.com/java-program-pollard-rho-algorithm/

37 // Primes ending in 9

Primes ending in 9 based on n * k + (k-1) for k > 1.

Example: n = 89; 89 * 2 + 1 = 179 (prime), 89 * 3 + 2 = 269 (prime), 89 * 4 + 3 = 359 (prime),
89 * 5 + 4 = 449 (prime), 89 * 6 + 5 = 539 (composite, fail)

Aufgabe: Parallelisiert ist ein möglichst langer Pfad zu ermitteln.

38 // Same quantity of prime factors

Same quantity of prime factors by adding consecutive primes.

Example: three sums, three factors; 3 + 5 = 8 = 2 * 2 * 2, 5 + 7 = 12 = 2 * 2 * 3,
7 + 11 = 18 = 2 * 3 * 3, 11 + 13 = 24 = 2 * 2 * 2 * 3 (fail)

Aufgabe: Parallelisiert sind Lösungen mit einer höheren Anzahl von Summen und Faktoren zu ermitteln.

Pollard Rho Algorithm: http://www.sanfoundry.com/java-program-pollard-rho-algorithm/

39 // Consecutive primes

$p_1 * p_2 + p_1 + p_2$ = prime, p_1 are p_2 consecutive primes.

Example: 11 * 13 + 11 + 13 = 167 (prime), 13 * 17 + 13 + 17 = 251 (prime),
(6 primes) 17 * 19 + 17 + 19 = 359 (prime), 19 * 23 + 19 + 23 = 479 (prime),
23 * 29 + 23 + 29 = 719 (prime), 29 * 31 + 29 + 31 = 959 (composite, fail)

Aufgabe: Parallelisiert ist eine Lösung mit 7 aufeinanderfolgenden Primzahlen zu ermitteln.

40 // Filling the gap

Upper-left cell is filled with the number 1. Any other cell is filled with the sum of numbers already saved in all of its (3, 5, 8) neighbour cells. This step is repeated until the grid is filled. The last cell filled has the largest value (result) for the chosen path. The largest value must be prime.

Example: k = 4

Grid (4x4):

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Grid is filled using the arbitrary path:
1-2-5-9-6-3-4-7-10-13-14-11-8-12-15-16

1	1	7	7
2	6	21	159
2	31	124	304
33	66	525	953

value (result): 953 (prime)

Aufgabe: Parallelisiert sind für k = 5 Lösungen zu ermitteln, das Ergebnis ist zu maximieren.

41 // Zeisel numbers

Zeisel Number, z(n), is a number with distinct prime factors $p_1 * p_2 * \ldots * p_n$,
where n >= 3, $p_0 = 1$, $p_i = a * p_{(i-1)} + b$ for i = 1..n and constant integers a, b.

Example: n = 3; smallest z(n) = 105 = 3 * 5 * 7, a = 1, b = 2

Aufgabe: Parallelisiert ist für n = [4..6] die minimale Zeisel Nummer z(n) zu ermitteln.
Pollard Rho Algorithm: http://www.sanfoundry.com/java-program-pollard-rho-algorithm/

42 // 493009335 **BigInteger**

493009335 is the smallest number n such that, if sum of its prime factors subtracted from it produces the previous prime and if added to the sum of its prime factors produces the next prime.

Example: 493009335 = 3 * 3 * 3 * 3 * 3 * 5 * 7 * 7 * 7 * 7 * 13 * 13: ΣPF = 74
493009335 - 74 = 493009261 (prime); 493009335 + 74 = 493009409 (prime)

Aufgabe: Parallelisiert ist diese Vermutung zu evaluieren.
Pollard Rho Algorithm: http://www.sanfoundry.com/java-program-pollard-rho-algorithm/

43 // Grimm's conjecture

Grimm's conjecture states that to each element of a set of consecutive composite numbers one can assign a distinct prime that divides it.

Example: Range 242 to 250, one can assign distinct primes as follows:
242: 11 243: 3 244: 61 245: 7 246: 41 247: 13 248: 31 249: 83 250: 5

Aufgabe: Parallelisiert sind weitere Wertebereiche zu analysieren.

44 // $p = 2 * p_1 + 3 * p_2$; p_1, p_2 are distinct odd primes

Every prime number p >= 19 can be written as $p = 2 * p_1 + 3 * p_2$
where p_1 and p_2 are distinct odd primes.

Examples: (i) 19 = 2 * 5 + 3 * 3 and (ii) 23 = 2 * 7 + 3 * 3

Aufgabe: Parallelisiert sind weitere Zahlen zu analysieren.

45 // Prime gaps

In the range $2^a * 3^b$ with a,b >= 0 find two such numbers in these three prime gaps.

Example: 7, **8**, **9**, 11, 23, **24**, **27**, 29, 31, **32**, **36**, 37

Aufgabe: Parallelisiert ist zu analysieren, ob eine Lösung mit vier Bereichen existiert.

46 // Smallest Prime and Perfect Square **BigInteger**

Find the smallest prime p such that it is possible to arrange all the primes, 2, 3, 5, 7, 11, 13,..., p, in a row such that the sum of any two adjacent primes is always a perfect square.

Aufgabe: Parallelisiert sind Lösungen zu ermitteln, p ist zu minimieren.

47 // Taxicab – Prime numbers **BigInteger**

Taxicab number is smallest number expressible as the sum of two cubes in two different ways.

Examples: composite: 1729 $9^3 + 10^3 = 1729$ and $1^3 + 12^3 = 1729$
prime: 6058655748 $61^3 + 1823^3 = 1049^3 + 1699^3$

Aufgabe: Parallelisiert sind Lösungen zu ermitteln, wobei die Taxicab Zahl eine Primzahl ist.

48 // Travelling intertwined primes **BigInteger**

Given two primes, intertwine them to get two primes.

Examples: (i) n = 2, 13 & 43 → 1433 & 4133; (ii) n = 3, 139 & 271 → 123791 & 217319

Aufgabe: Parallelisiert sind für n = 2, 3, 4 die Lösungen zu ermitteln.

49 // Primes with zeros replaced

Find the smallest prime containing the digit 0 such that replacing all occurrences of 0 by any other digit also produces a prime.

Example: 60301 → 60301, 61311, 62321, 63331,..., 69391 are all primes.

Aufgabe: Parallelisiert sind Lösungen zu ermitteln.

50 // Primes by duplicating some digit **BigInteger**

Example: 18593 → 18**5**593 → 18559**9**3 → 1**1**855993 → 11855**5**993 → 118555999**3** →
1185559993**3** → 11855599933**3** → 11855599**9**9333 → 118555999933**3**3 →
11855599993333**3**

Aufgabe: Parallelisiert ist eine möglichst lange Sequenz zu ermitteln.

51 // 2213

2213 is the smallest prime p that can be partitioned in three parts a, b and c – not necessarily of the same length – such that (i) p = a&b&c and (ii) $p = a^3 + b^3 + c^3$; $2213 = 2^3 + 2^3 + 13^3$

Aufgabe: Parallelisiert ist diese Vermutung zu evaluieren.

52 // 10958

Way to render every number from 1 to 11111 by starting with either of these ordered strings. 1234567869 (increasing 1 -> 9) and 987654321 (decreasing 9 -> 1) applying any of the following: a) arithmetical operations permitted: addition, subtraction, multiplication, division, and exponentiation; b) string operation permitted: concatenation; c) auxiliary symbols permitted: brackets "(" and ")".

Example: (i) $10957 = (1 + 2)^{(3+4)} * 5 - 67 + 89$; (ii) $11038 = (9 * 8 * 7 + 6^5) * 4/3 - 2 * 1$

Aufgabe: Parallelisiert ist für 10958 eine aufsteigende Sequenz zu ermitteln.

53 // Primes and convolution

Convolution c(n) of an integer n is an operation made over the decimal digits of n,
according to reverse the order of the digits and multiply them in pairs.
Find prime p such that p + c(p) generates another prime q.

Examples: (i) p = 11, p + c(p) = 11 + 2 = 13; (ii) p = 17, p + c(p) = 17 + 14 = 31
{ 11, 17 } → { 13, 31 }

Aufgabe: Parallelisiert ist eine möglichst lange Sequenz von aufeinanderfolgenden Primzahlen zu ermitteln.

54 // Primes between perfect powers

Perfect power is x^a, where x and a are greater than 1. (i) Find the contiguous perfect power y^b to another given x^a, such that $y^b > x^a$. (ii) Find only pairs of contiguous (x^a, y^b) such that $D = y^b - x^a$, for D <= 100 and $x^a < 10^{20}$ and show all these pairs in a table by ascending D value, exhibiting also the quantity of primes between the perfect powers of each pair.

Example:

x, a	y, b	D	number primes
2,3	3,2	1	0
5,2	3,3	2	0
2,2	2,3	4	2

Aufgabe: Parallelisiert sind die Lösungen zu ermitteln.

55 // Consecutive integers and prime factors

Pairs of consecutive numbers n and n+1 have the property that the largest prime factor of n and n+1 is less than the ln(n) and ln(n+1).

Examples: 2400, 2401: (2,3,5), (7) → 5 < ln(2400) and 7 < ln(2401)
4374, 4375: (2,3), (5,7) → 3 < ln(4374) and 7 < ln(4375)

Aufgabe: Parallelisiert sind weitere Lösungen zu ermitteln.
Pollard Rho Algorithm: http://www.sanfoundry.com/java-program-pollard-rho-algorithm/

Zielsetzungen

- Training der Kompetenz bezüglich der Implementierung von anspruchsvollen Algorithmen.
- Wiederholung und Vertiefung des Wissens zu Parallelisierung mit CyclicBarrier.
- Praktische Anwendung des Wissens auf rechenintensive/komplexe Aufgabenstellungen.

Optimierung JVM und Garbage Collection

```
-server -Xms4G -Xmx4G -XX:MaxDirectMemorySize=1024M -XX:NewSize=1G
-XX:MaxNewSize=1G -XX:+UseParNewGC -XX:MaxTenuringThreshold=2
-XX:SurvivorRatio=8 -XX:+UnlockDiagnosticVMOptions
-XX:ParGCCardsPerStrideChunk=32768
```

Aufgabenstellung

- **Implementierung** einer technisch einwandfrei lauffähigen Applikation in Java 8.
  Graphische Benutzeroberfläche in JavaFX.
- Dynamische Ermittlung der Cores (Runtime.getRuntime().availableProcessors())
  und ausbalancierte Verteilung der Berechnung auf die verfügbaren Cores.
- **Test** der Implementierung mit **JUnit** und Gewährleistung der Funktionsweise.
- **Lösungsverfahren**: Brute-Force.
- Nutzung leistungsfähiger Datenstrukturen und Optimierung der Performance mit VisualVM.
- **Maximale Laufzeit der Evaluierung:** 15 Minuten

Wichtige Hinweise:

- **Pro Student** wird **eine Aufgabe** bearbeitet.
- Die **Zuordnung** einer **Aufgabe** zu einem Studierenden erfolgt **mit** einem **Zufallsgenerator**.
- Nutzung der **camelCase-Notation**, um die Lesbarkeit zu vereinfachen.
- Zulässige externe Bibliotheken: junit-jupiter-api.jar und opentest4j.jar.
- Verwendung geeigneter **englische**r Begriffe für **Namen** und **Bezeichnungen**.
- **Erstellung einer vollständigen und verschlüsselten 7-Zip-Datei unter Beachtung des Prozedere für die Abgabe von Prüfungsleistungen und der Namenskonvention.**
- **Zeitansatz:** 10 Stunden
- **Abgabetermin:** Sonntag, 18.02.2018
- **Bewertung**: Testat