Оглавление

[1. Анализ информационных моделей 2](#_Toc162014077)

[2. Построение таблиц истинности логических выражений 2](#_Toc162014078)

[3. Поиск информации в реляционных базах данных 2](#_Toc162014079)

[4. Кодирование и декодирование информации 2](#_Toc162014080)

[5. Анализ и построение алгоритмов для исполнителей 3](#_Toc162014081)

[6. Определение результатов работы простейших алгоритмов 6](#_Toc162014082)

[7. Кодирование и декодирование информации. Передача информации 9](#_Toc162014083)

[8. Перебор слов и системы счисления 9](#_Toc162014084)

[9. Работа с таблицами 11](#_Toc162014085)

[10. Поиск символов в текстовом редакторе 11](#_Toc162014086)

[11. Вычисление количества информации 12](#_Toc162014087)

[12. Выполнение алгоритмов для исполнителей 13](#_Toc162014088)

[13. Организация компьютерных сетей. Адресация 15](#_Toc162014089)

[14. Кодирование чисел. Системы счисления 16](#_Toc162014090)

[15. Преобразование логических выражений 18](#_Toc162014091)

[16. Рекурсивные алгоритмы 20](#_Toc162014092)

[17. Обработки числовой последовательности 24](#_Toc162014093)

[18. Робот-сборщик монет 26](#_Toc162014094)

[19. Выигрышная стратегия. Задание 1 27](#_Toc162014095)

[20. Выигрышная стратегия. Задание 2 30](#_Toc162014096)

[21. Выигрышная стратегия. Задание 3 33](#_Toc162014097)

[22. Многопроцессорные системы 36](#_Toc162014098)

[23. Оператор присваивания и ветвления. Перебор вариантов, построение дерева 36](#_Toc162014099)

[24. Обработка символьных строк 38](#_Toc162014100)

[25. Обработка целочисленной информации 41](#_Toc162014101)

[26. Обработка целочисленной информации 43](#_Toc162014102)

[27. Программирование 43](#_Toc162014103)

# Анализ информационных моделей

# Построение таблиц истинности логических выражений

print("x w z y")

# x y w z переменные из задачи

for x in range(0, 2):

for y in range(0, 2):

for z in range(0, 2):

for w in range(0, 2):

if (not(not(x) or y) or (x == z) or w) == False:

# В данной формуле нужно менять условие в соответствии с условием задачи

print(x, w, z, y)

# отрицание - not

# конъюнкция(и, \*) - and

# дизъюнкция(или, +) - or

# следование(импликация) - (not(x) or y) = (x -> y)

# тождество - ==

# Поиск информации в реляционных базах данных

# Кодирование и декодирование информации

# Анализ и построение алгоритмов для исполнителей

|  |  |
| --- | --- |
| for x in range(1,30): #перебираем х  N=bin(x)[2:] #получаем двоичную запись, убираем первые 2 символа 0b  if N[-1]== '0': # преобразование 2 шага  N = N + '0'  else:  N = N + '1'  if N.count('1')%3 == 0: # преобразование 3 шага  R=N.replace(N[:2],'11', 1)  else:  R=N.replace(N[:2],'10', 1)  print(x, int(R,2)) # переводим R в десятичную систему счисления | for x in range(100,1000):  a= x//100  b = (x//10)%10  c = x % 10  p = a \* b \* c  s = a + b + c  if p == 240 and s == 19:  c,b,a = sorted((a,b,c))  print(a\*100 + b\*10 + c)  break |

|  |  |
| --- | --- |
| for x in range(1,50): #перебираем х  N=bin(x)[2:] #получаем двоичную запись, убираем первые 2 символа 0b  N = N.replace('0','00')  N = N.replace('1','11')  R = int(N,2)# переводим R в десятичную систему счисления  print(x, R) | for x in range(10,35): #перебираем х  N = x - x % 8  N = N + N % 2  N=bin(N)[2:] #получаем двоичную запись, убираем первые 2 символа 0b  if N.count('1')%2 == 0:  N = N + '0'  else:  N = N + '1'  if N.count('1')%2 == 0:  R = N + '0'  else:  R = N + '1'  print(x, int(R,2)) # переводим R в десятичную систему счисления |

|  |  |
| --- | --- |
| for x in range(1,30): #перебираем х  N=bin(x)[2:] #получаем двоичную запись, убираем первые 2 символа 0b  if len(N)%2 == 0:  R = N[:len(N)//2] + '1' +N[len(N)//2:]  else:  R = N  print(x, int(R,2)) # переводим R в десятичную систему счисления | for x in range(100,1000):  a= x//100  b = (x//10)%10  c = x % 10  if (a \*\* 2 + b\*\*2 == 74 and b\*\*2 + c\*\*2 == 34) or (a \*\* 2 + b\*\*2 == 34 and b\*\*2 + c\*\*2 == 74):  c, a = sorted((a, c))  print(a\*100 +b\*10+c)  break |

# Определение результатов работы простейших алгоритмов

|  |  |
| --- | --- |
| from turtle import \*  koff\_mash = 20  povtor = 2  correction = 0  color("green")  left(90)  begin\_fill()  for i in range(povtor-correction):  forward(17\*koff\_mash)  left(90)  forward(10\*koff\_mash)  left(90)  penup()  end\_fill()  backward(4\*koff\_mash)  right(90)  backward(3\*koff\_mash)  left(90)  pendown()  begin\_fill()  for i in range(povtor-correction):  forward(40\*koff\_mash)  right(90)  forward(10\*koff\_mash)  right(90)  end\_fill()  penup()  cnt = 0  canvas = getcanvas()  for x in range(-100\*koff\_mash, 100\*koff\_mash, koff\_mash):  for y in range(-100\*koff\_mash, 100\*koff\_mash, koff\_mash):  s = canvas.find\_overlapping(x,y,x,y)  if len(s) > 0:  cnt += 1  print(cnt) |  |

|  |  |
| --- | --- |
| Исполнитель Черепаха действует на плоскости с декартовой системой координат. В начальный момент Черепаха находится в начале координат, её голова направлена вдоль положительного направления оси ординат, хвост опущен. При опущенном хвосте Черепаха оставляет на поле след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя существует две команды: **Вперёд n**, вызывающая передвижение Черепахи на n единиц в том направлении, куда указывает её голова, и **Направо m**, вызывающая изменение направления движения на m градусов по часовой стрелке. Черепахе был дан для исполнения следующий алгоритм:  **Повтори 4 [Вперёд 8 Направо 90]**  **Повтори 3 [Вперёд 12 Направо 120]**  Определите, сколько точек с целочисленными координатами будут находиться внутри области, ограниченной линией, заданной данным алгоритмом: **Повтори 4 [Вперёд 8 Направо 90]** и находиться вне области, ограниченной линией, заданной данным алгоритмом: **Повтори 3 [Вперёд 12 Направо 120].** Точки на линии учитывать не следует.  # Если в задании просят посчитать кол-во точек входящих в одну фигуру и не входящих в другую,  # то надо считать разность между количеством точек внутри одной фигуры и другой без учета границ  from turtle import \*  koff\_mash = 10  povtor = 4  correction = 1  color("green")  left(90)  begin\_fill()  for i in range(povtor):  forward(9\*koff\_mash)  right(90)  print(position())  end\_fill()  penup()  cnt = 0  canvas = getcanvas()  for x in range(-100\*koff\_mash,100\*koff\_mash,koff\_mash):  for y in range(-100\*koff\_mash,100\*koff\_mash,koff\_mash):  s = canvas.find\_overlapping(x,y,x,y)  if len(s) == 1 :  cnt+=1  print(cnt)  canvas.delete('all')  begin\_fill()  color("red")  for i in range((povtor- correction)):  forward(9\*koff\_mash)  right(120)  print(position())  end\_fill()  penup()  cnt = 0  canvas = getcanvas()  for x in range(-100\*koff\_mash,100\*koff\_mash,koff\_mash):  for y in range(-100\*koff\_mash,100\*koff\_mash,koff\_mash):  s = canvas.find\_overlapping(x,y,x,y)  if len(s) == 1:  cnt+=1  print(cnt) | Исполнитель Цапля действует на плоскости с декартовой системой координат. В начальный момент Цапля находится в начале координат, её клюв направлен вдоль положительного направления оси ординат, клюв опущен. При опущенном клюве Цапля оставляет на поле след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя существует три команды: **Вперёд n**, вызывающая передвижение Цапли на n единиц в том направлении, куда указывает её клюв; **Направо m**, вызывающая изменение направления движения на m градусов по часовой стрелке; **Дуга r, a, b, α**, вызывающая передвижение Цапли из текущей точки с координатами (x, y) по дуге окружности с центром в точке с координатами и радиусом r, градусная мера дуги равна α, движение по дуге идёт по часовой стрелке. Цапле был дан для исполнения следующий алгоритм:  **Повтори 5 [Дуга 5, 0, 5, 180 Дуга 5, 5, 0, 180 Дуга 5, 0, -5, 180 Дуга 5, -5, 0, 180].**  Определите, сколько точек с целочисленными координатами будут находиться внутри области, ограниченной линией, заданной данным алгоритмом. Точки на линии учитывать не следует.  from turtle import \*  koff\_mash = 10  povtor = 5  correction = 4  color("green")  left(90)  begin\_fill()  for i in range(povtor - correction):  seth(0)  circle(5\*koff\_mash,180)  seth(90)  circle(5\*koff\_mash,180)  seth(180)  circle(5\*koff\_mash,180)  seth(270)  circle(5\*koff\_mash,180)  end\_fill()  penup()  cnt = 0  canvas = getcanvas()  for x in range(-100\*koff\_mash,100\*koff\_mash,koff\_mash):  for y in range(-100\*koff\_mash,100\*koff\_mash,koff\_mash):  s = canvas.find\_overlapping(x,y,x,y)  if len(s) == 1 and s[0] == 5:  cnt+=1  print(cnt) |

# Кодирование и декодирование информации. Передача информации

# Перебор слов и системы счисления

|  |  |
| --- | --- |
| num = 0  count = 0  for n1 in "АВИОРТФ":  for n2 in "АВИОРТФ":  for n3 in "АВИОРТФ":  for n4 in "АВИОРТФ":  for n5 in "АВИОРТФ":  for n6 in "АВИОРТФ":  num += 1  col\_n = 0  s = n1 + n2 + n3 + n4 + n5 + n6  for i in s:  if i == "Р":  col\_n += 1  if (n1 != "О" and num % 2 == 0 and col\_n == 2):  count += 1  print(num, n1, n2, n3, n4, n5, n6, col\_n)  print(count) | def convert\_to(number, base, razr):  # digits = '0123456789abcdefghijklmnopqrstuvwxyz'  st = "К"\*razr  digits = "КНОРСЯ"  if base > len(digits): return None  result = ''  while number > 0:  result = digits[number % base] + result  number //= base  len\_res = len(result)  st = st[len\_res:] + result  return st  for i in range(1000000):  s = convert\_to(i, 6,6)  if s.count("Я") == 2 and s.count("К") <= 3:  print(s, i+1)  break |

|  |  |
| --- | --- |
| count = 0  for n1 in range(1,5):  for n2 in range(5):  for n3 in range(5):  if not(n1 < n2 or n2 < n3):  count += 1  print(count) | num = 0  for n1 in range(1,8):  for n2 in range(8):  for n3 in range(8):  for n4 in range(8):  if (n1 == n2 and n1 != n3 and n1 != n4 and n3 != n4) \  or (n2 == n3 and n2 != n4 and n3 != n1 and n1 != n4) \  or (n3 == n4 and n3 != n2 and n1 != n3 and n1 != n2):  num += 1  print(n1, n2, n3, n4, num)  print(num)  print(5\*5\*5\*5\*5)  # 1 - 12345, 2,3,4,5-23456 |

# Работа с таблицами

|  |  |  |
| --- | --- | --- |
| f = open("C:\\Users\\user\\ege\_inf\\EGE-inf\\EGE\_2024\_Informatika\_20var\_variant-01\_Faily\\9\\9 вариант 1.csv")  counter = 0  for s in f:  s\_list = list(map(int, s.split(";")))  s\_repeat = []  pair = 0  for i in s\_list:  if s\_list.count(i) == 3 and not(i in s\_repeat):  s\_repeat.append(i)  if s\_list.count(i) == 2:  pair += 1  break  if len(s\_repeat) == 2 and pair == 0 and not(max(s\_list) in s\_repeat):  counter += 1  print(s\_list)  print(counter) |  |  |

# Поиск символов в текстовом редакторе

# Вычисление количества информации

Объем сообщения - это произведение части сообщения на колличество его частей

V = Vn\*n

N = 2^i

N - максимально возморжное количество символов, которое можно закодировать с помощью i бит

i - количество бит с помощью которого можно закодировать N символолв

# Выполнение алгоритмов для исполнителей

|  |  |
| --- | --- |
|  | for i in range(4, 10000):  s = "1" + "2"\*i  count = 0  while ("12" in s) or ("3322" in s) or ("2222" in s):  if ("12" in s):  s = s.replace("12", "33", 1)  elif ("2222" in s):  s = s.replace("2222", "1", 1)  elif ("3322" in s):  s = s.replace("3322", "21", 1)  for j in s:  count += int(j)  if count == 218:  print(i)  break |

|  |  |
| --- | --- |
|  | for i in range(0, 10000):  s = ">" + "1"\*11 + "2"\*i + "3"\*11  count = 0  prost\_mr = 0  while (">1" in s) or (">2" in s) or (">3" in s):  if (">1" in s):  s = s.replace(">1", "222>", 1)  elif (">2" in s):  s = s.replace(">2", "3>", 1)  elif (">3" in s):  s = s.replace(">3", "1>", 1)  for j in s:  if j != '>':  count += int(j)  for n in range(2,count):  if count % n == 0:  prost\_mr += 1  if prost\_mr == 0:  print(i)  break |

|  |  |
| --- | --- |
|  | s='1' + '5'\*25  count = 0  while ('15' in s) or ('1' in s):  if ('15' in s):  s = s.replace("15", '5551', 1)  elif ('1' in s):  s=s.replace ('1','5',1)  for j in s:  if j == "5":  count += 1  print(count) |

# Организация компьютерных сетей. Адресация

# Кодирование чисел. Системы счисления

|  |  |
| --- | --- |
| for x in range(23):  n1 = 1 \* 23\*\*8 + x \* 23\*\*7 + 1 \* 23\*\*6 + x \* 23\*\*5 + 1 \* 23\*\*4 + x \* 23\*\*3 + 1 \* 23\*\*2 + x \* 23\*\*1 + 1 \* 23\*\*0  n2 = 2 \* 23\*\*4 + 0 \* 23\*\*3 + x \* 23\*\*2 + 2 \* 23\*\*1 + 4 \* 23\*\*0  n3 = 1 \* 23\*\*4 + x \* 23\*\*3 + 2 \* 23\*\*2 + 3 \* 23\*\*1 + 5 \* 23\*\*0  if (n1 + n2 + n3) % 22 == 0:  print((n1 + n2 + n3) // 22, x)  break | x = 4\*25\*\*2022 - 2 \*5\*\*2000 + 125\*\*1011 - 3 \* 5\*\*100 - 660  k = 0  while x > 0:  if x % 5 == 4:  k += 1  x = x // 5  print(k) |
| x = 1331\*\*650 - 55 \*121\*\*610 + 77\*11\*\*510 - 3 \* 11\*\*100 - 221  k = 0  while x > 0:  if x % 11 == 10:  k += 1  x = x // 11  print(k)  # 0 1 2 3 4 5 6 7 8 9 A | for x in '0123456789ABCDEFG':  a = '135'+ str(x) + '9'  b = '9' + str(x) + '531'  c = int(a,17)+ int(b,17)  if c % 9 == 0:  print(c//9) |

|  |  |
| --- | --- |
| for x in '0123456789ABCDEFGHIJKLMNOP':  ok = True  for y in '0123456789ABCDEFGHIJKLMNOP':  a = '13'+ y + x + '5'  b = '24' + y + '13'  if (int(a,26)+ int(b,26)) % 8 != 0:  ok = False  if ok:  a = '132' + x + '5'  b = '24213'  print((int(a,26)+int(b,26))//8) |  |

# Преобразование логических выражений

|  |  |
| --- | --- |
| for A in range(1, 1\_000):  ok = True  for x in range(1\_000):  for y in range(1\_000):  f = (4\*x + y < A) or (x < y) or (22 <= x)  if not f:  ok = False  break  if ok:  print(A)  break | for A in range(1, 1\_000):  ok = True  for x in range(1\_000):  for y in range(1\_000):  f = (x < 4) or (x >= 20) or (y >= 3\*x + A) or (y < 100)  if not f:  ok = False  break  if ok:  print(A) |
| def d(n, m):  if n % m == 0: #проверка на делимость  return 1  for A in range (1,1000):  OK = True  for x in range (1,1000):  OK = OK\*( not(d(x,13)) or not(d(x,21)) or (x + A) >= 500 )  if OK: # истинность для любого х  print (A)  break | def treug(n, m, k):  if n < m + k and m < k + n and k < n + m:  return 1  else: return 0  for A in range (1,100):  OK = True  for x in range (1,100):  OK\*= (not((treug(x, 11, 18) == (not(max(x, 5) > 15))) and treug(x, A, 5)))  if OK: # истинность для любого х  print (A) |

|  |  |
| --- | --- |
| B = [i for i in range(10,16)]  C = [i for i in range(20,28)]  A = []  for x in range(100):  if not( ((x in B) or (x in C)) <= (x in A) ):  A.append(x)# добавляем в массив А все х, для которых выражение ложно  print(max(A) - min(A)) #находим разницу между макс и мин значением элементов массива | B = [i for i in range(4,19)]  C = [i for i in range(12,41)]  A = []  for x in range(1,100):  if not(not(x in A)) <= ((x in B) == (x in C)) : # внимание, скобки!  A.append(x)  print(max(A)- min(A)) |
| for A in range (0,100):  OK = True  for x in range (1,100):  for y in range (1, 100):  OK \*= not( (x < A) and (y < A) and (x\*y > 1200))  if OK: # истинность для любого х  print (A, OK) | На числовой прямой даны два отрезка: **P = [5, 30]** и **Q = [14, 23].** Укажите наибольшую возможную длину промежутка A, для которого формула  **((x ∈ P) ≡ (x ∈ Q)) → ¬(x ∈ A)**  тождественно истинна, то есть принимает значение 1 при любом значении переменной х.  m = 0  P = [i for i in range(5, 31)]  Q = [i for i in range(14, 24)]  for Amin in range(1, 101):  for Amax in range(Amin + 1, 101):  check = 1  A = [i for i in range(Amin, Amax)]  for x in range(1, 101):  f = ((x in P) == (x in Q)) <= (not(x in A))  if not f:  check = 0  break  if check == 1:  print(m, Amin, Amax)  m = max(m, Amax - Amin)  print(m) |

# Рекурсивные алгоритмы

для больших чисел

Нужно разложить число по формуле для представленных чисел

после сократить если есть возможность

|  |  |
| --- | --- |
| для больших чисел  Нужно разложить число по формуле для представленных чисел  после сократить если есть возможность  Например для задачи 16 вар 3:  F(n) = 1 при n = 1  F(n) = 2 при n = 2  F(n) = F(n - 1) + F(n - 2) + n\*(n - 1) если n > 2  Чему равно значение выражения F(2024) - F(2022) - 2\*F(2021) - F(2020)  Решение:  F(2024) - F(2022) - 2\*F(2021) - F(2020) =  = F(2024 - 1) + F(2024 - 2) + 2024\*(2024 - 1) - F(2022) - 2\*F(2021) - F(2020) =  = F(2023) + F(2022) + 2024\*2023 - F(2022) - 2\*F(2021) - F(2020) =  = F(2023) + 2024\*2023 - 2\*F(2021) - F(2020) =  = F(2022) + F(2021) + 2023\*2022 + 2024\*2023 - 2\*F(2021) - F(2020) =  = F(2022) + 2023\*2022 + 2024\*2023 - F(2021) - F(2020) =  = F(2021) + F(2020) + 2022\*2021 + 2023\*2022 + 2024\*2023 - F(2021) - F(2020) =  = 2022\*2021 + 2023\*2022 + 2024\*2023 =  = 12271520 | import sys  sys.setrecursionlimit(3000) #регулирует глубину рекурсии  def Func(n):  if n == 1:  res = 5  elif n > 1:  res = Func(n - 1) + 2\*n + 1  return res  s1 = Func(2024)  s2 = Func(2022)  print(s1 - s2) |

для больших чисел

Нужно разложить число по формуле для представленных чисел после сократить если есть возможность

|  |  |
| --- | --- |
| import sys  sys.setrecursionlimit(3000) #регулирует глубину рекурсии  def Func(n):  if n < 3:  res = n  elif n > 2 and n % 2 == 0:  res = 2 \* (n - 1) + Func(n - 1) + 2  elif n > 2 and n % 2 != 0:  res = 2 \* (n + 1) + Func(n - 2) - 5  return res  s1 = Func(data['n1'])  print(s1) | import sys  sys.setrecursionlimit(3000) #регулирует глубину рекурсии  def Func(n):  if n == 1:  res = 5  elif n > 1:  res = Func(n - 1) + 2\*n + 1  return res  s1 = Func(2024)  s2 = Func(2022)  print(s1 - s2) |

для больших чисел

Нужно разложить число по формуле для представленных чисел после сократить если есть возможность

|  |  |
| --- | --- |
| import sys  sys.setrecursionlimit(3000) #регулирует глубину рекурсии  def Func(n):  if n < 3:  res = n  elif n > 2 and n % 2 == 0:  res = 2 \* (n - 1) + Func(n - 1) + 2  elif n > 2 and n % 2 != 0:  res = 2 \* (n + 1) + Func(n - 2) - 5  return res  s1 = Func(32)  print(s1) | import sys  sys.setrecursionlimit(3000) #регулирует глубину рекурсии  def Func(n):  if n < 3:  res = 1  elif n > 2 and n % 2 != 0:  res = Func(n - 1) + Func(n - 2)  elif n > 2 and n % 2 == 0:  res = sum(Func(i) for i in range (1,n))  return res  s1 = Func(24)  print(s1) |

|  |  |
| --- | --- |
| from functools import lru\_cache # Декоратор @lru\_cache() модуля functools #оборачивает функцию с переданными в нее аргументами и запоминает #возвращаемый результат. Экономит время.  @lru\_cache  def F(n):  if n < 3: return 1  if n > 2 and n % 2 == 1: return F(n - 1) - F(n - 2)  if n > 2 and n % 2 == 0: return sum(F(i) for i in range (1,n))  print(F(39)) |  |

# Обработки числовой последовательности

|  |  |
| --- | --- |
| В файле содержится последовательность натуральных чисел. Элементы последовательности могут принимать целые значения от 1 до 100 000 включительно. Определите количество троек элементов последовательности, в которых **только одно** из чисел является **двузначным**, а сумма элементов тройки меньше **минимального** элемента последовательности, оканчивающегося **на 25**. В ответе запишите количество найденных троек чисел, затем **максимальную** из сумм элементов таких троек.  f = open('17var01.txt') p = [int(i) for i in f] f.close() m = float('inf') count = 0 m3 = 0 for i in p:     if i % 100 == 25:         m = min (m, i) for i in range(len(p) - 2):     s = p[i] + p[i+1] + p[i + 2]     if (10<= p[i] < 100) + (10 <= p[i+1] < 100) + (10 <= p[i+2] < 100) == 1 and s < m:         count += 1         m3 = max(m3, s ) print(count, m3) | В файле содержится последовательность целых чисел. Элементы последовательности могут принимать целые значения от -100 000 до 100 000 включительно. Определите количество **троек** элементов последовательности, в которых **хотя бы одно из чисел** является **четырехзначным**, а сумма элементов тройки больше **максимального** элемента последовательности, оканчивающегося **на 90**. В ответе запишите количество найденных троек чисел, затем **минимальную** из сумм элементов таких троек.  f = open('17var03.txt') p = [int(i) for i in f] f.close()  m = -100000 m3 = 300000 count = 0  for i in p:     if abs(i) % 100 == 90:         m = max (m, i)  for i in range(len(p) - 2):     s = p[i] + p[i+1] + p[i + 2]     if ((999 < abs(p[i]) <= 9999) + (999 < abs(p[i+1]) <= 9999)  + (999 < abs(p[i+2]) <= 9999) > 0 )  and s > m:          count += 1         m3 = min(m3, s  )  print(count, m3) |

|  |  |
| --- | --- |
| В файле содержится последовательность натуральных чисел. Элементы последовательности могут принимать целые значения от 1 до 100 000 включительно. Определите **количество** пар последовательности, в которых **сумма чисел пары равна максимальному элементу** последовательности. В ответе запишите **количество найденных пар**, затем **максимальную из сумм квадратов** элементов таких пар.  f= open('17var05.txt') p=[int(i) for i in f] f.close() k = 0 PP = 0 M = max(p)  for i in range(1,len(p)):     if p[i-1] + p[i] == M:         k+=1         PP = max(PP, p[i-1]\*\*2 + p[i]\*\*2)  print(k,PP) | В файле содержится последовательность натуральных чисел. Элементы последовательности могут принимать целые значения от 1 до 100 000 включительно. Определите **количество** **троек** последовательности, в которых **только одно число оканчивается на 0**, а сумма чисел тройки меньше **максимального элемента**последовательности. В ответе запишите **количество найденных троек**, затем **максимальную из сумм элементов**таких троек.  f= open('17var07.txt') p=[int(i) for i in f] f.close() k = 0 PP = 0 M = max(p)  for i in range(2,len(p)):     if  (p[i-2] + p[i-1] + p[i] < M) and ((p[i-2]%10 == 0) + ( p[i-1]%10 == 0) + (p[i]%10 == 0) == 1) :         k+=1         PP = max(PP, p[i-2] + p[i-1] + p[i])  print(k,PP) |
| В файле содержится последовательность целых чисел. Элементы последовательности могут принимать целые значения от -10 000 до 10 000 включительно. Определите и запишите в ответе сначала **количество пар** элементов последовательности, в которых **хотя бы одно число является полным квадратом некоторого натурального числа**, затем **максимальную** из сумм элементов таких пар. **Например**, для **последовательности 6, 2, 9, -3, 1** ответом будет **3  11.**  f=open("17var09.txt") p= [int(i) for i in f] f.close()  k=0 PP = -20000 for i in range(len(p)-1):    if p[i]>0 and p[i]\*\*0.5==int(p[i]\*\*0.5) or p[i+1]>0 and [i+1]\*\*0.5==int(p[i+1]\*\*0.5):         k+=1         PP = max(p[i]+p[i+1], PP) print(k, PP) | В файле содержится последовательность целых чисел. Элементы последовательности могут принимать целые значения от**-10 000 до 10 000** включительно. Определите и запишите в ответе сначала **количество пар** элементов последовательности, в которых **хотя бы один элемент пары превосходит 700, затем максимальную из сумм квадратов элементов таких пар**. **Например**, для **последовательности из пяти элементов: 60; 2; 90; -30; 100**ответом будут числа**2   120.**  f =open("17var18.txt") p = [int(i) for i in f] f.close()  k = 0 PP = -float('inf')  for i in range(1,len(p)):     if p[i-1] > 700 or  p[i] > 700:         k += 1         PP = max(p[i-1] \*\*2 + p[i]\*\*2, PP)  print(k, PP) |

# Робот-сборщик монет

|  |  |
| --- | --- |
|  | vert = [все координаты в которые нельзя попасть по вертикали]  gor = [все координаты в которые нельзя попасть по горизонтали]  iskl = [все координаты в которые нельзя попасть]  def max\_way():  res = [] # двумерный список для столбцов  for i in range(len(tabl)):  res\_st = [] # одномерный список для строк  for j in range(len(tabl[i])):  if [i,j] == [0,0]:  res\_st.append(tabl[i][j])  elif [i, j] in vert: # вертикальные исключения  res\_st.append(tabl[i][j]+max(res[i-1][j], 0))  elif [i, j] in gor: # горизонтальные исключения  res\_st.append(tabl[i][j]+max(0,res\_st[j-1]))  else:  res\_st.append(tabl[i][j]+max(res[i-1][j],res\_st[j-1]))  res.append(res\_st)  return res  def min\_way():  res = []  for i in range(len(tabl)):  res\_st = []  for j in range(len(tabl[i])):  if [i,j] == [0,0]:  res\_st.append(tabl[i][j])  elif [i, j] in vert: # вертикальные исключения  res\_st.append(tabl[i][j]+min(res[i-1][j], 10\*\*10))  elif [i, j] in gor: # горизонтальные исключения  res\_st.append(tabl[i][j]+min(10\*\*10,res\_st[j-1]))  else:  res\_st.append(tabl[i][j]+min(res[i-1][j],res\_st[j-1]))  res.append(res\_st)  return res  file = open("C:\\Users\\user\\ege\_inf\\EGE-inf\\18\\18var01.csv")  tabl = []  for s in file:  st = list(map(int, s.split(",")))  tabl.append(st)  print(st)  file.close()  num\_str = len(tabl) - 1  num\_column = len(tabl[0]) - 1  print(max\_way()[num\_str][num\_column], min\_way()[num\_str][num\_column],min\_way()[19][7], min\_way()[11][9]) |

# Выигрышная стратегия. Задание 1

|  |  |
| --- | --- |
|  | def game(col\_rocks, pl\_position, v\_hod1, v\_hod2, v\_hod3, winner):  #р - позиция игрока. если четная, то ход Вани, иначе - ход Пети  if pl\_position == 3 and col\_rocks >= winner:  s = 1  elif pl\_position == 3 and col\_rocks < winner:  s = 0  elif col\_rocks >= winner and pl\_position < 3:  s = 0  else:  if pl\_position % 2 == 0: # стратегия победителя  s1 = game(col\_rocks + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s2 = game(col\_rocks + v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s3 = game(col\_rocks \* v\_hod3, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s = s1 or s2 or s3  else: # стратегия проигравшего  s1 = game(col\_rocks + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s2 = game(col\_rocks + v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s3 = game(col\_rocks \* v\_hod3, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s = s1 and s2 and s3  return s  path = "/home/user/Documents/EGE inf/19/19/19\_type1.json"  data = read\_json(path)  for s in range(1, 202):  if game(s, 1,1,4,3,202) == 1:  print(s)  break |

|  |  |
| --- | --- |
|  | def game(col\_rocks, start\_heap\_1, pl\_position, v\_hod1, v\_hod2, winner): #р - позиция игрока. если четная, то значит, ход Вани, если нет - ход Пети  if pl\_position == 3 and col\_rocks + start\_heap\_1 >= winner:  s = 1  elif pl\_position == 3 and col\_rocks + start\_heap\_1 < winner:  s = 0  elif col\_rocks + start\_heap\_1 >= winner and pl\_position < 3:  s = 0  else:  if pl\_position % 2 == 0: # стратегия победителя  s1 = game(col\_rocks + v\_hod1, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s2 = game(col\_rocks \* v\_hod2, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s3 = game(col\_rocks, start\_heap\_1 + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s4 = game(col\_rocks, start\_heap\_1 \* v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, winner)  s = s1 or s2 or s3 or s4  else: # стратегия проигравшего  s1 = game(col\_rocks + v\_hod1, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s2 = game(col\_rocks \* v\_hod2, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s3 = game(col\_rocks, start\_heap\_1 + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s4 = game(col\_rocks, start\_heap\_1 \* v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, winner)  s = s1 or s2 or s3 or s4  return s  for s in range(1, 139):  if game(s, 2, 1, 2, 2, 142) == 1:  print(s)  break |

|  |  |
| --- | --- |
|  | def game(col\_rocks, start\_heap\_1, pl\_position, v\_hod1, v\_hod2, winner): #р - позиция игрока. если четная, то значит, ход Вани, если нет - ход Пети  if pl\_position == 3 and col\_rocks \* start\_heap\_1 >= winner:  s = 1  elif pl\_position == 3 and col\_rocks \* start\_heap\_1 < winner:  s = 0  elif col\_rocks + start\_heap\_1 >= winner and pl\_position < 3:  s = 0  else:  if pl\_position % 2 == 0: # стратегия победителя  s1 = game(col\_rocks + v\_hod1, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s2 = game(col\_rocks \* v\_hod2, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s3 = game(col\_rocks, start\_heap\_1 + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s4 = game(col\_rocks, start\_heap\_1 \* v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, winner)  s = s1 or s2 or s3 or s4  else: # стратегия проигравшего  s1 = game(col\_rocks + v\_hod1, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s2 = game(col\_rocks \* v\_hod2, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s3 = game(col\_rocks, start\_heap\_1 + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s4 = game(col\_rocks, start\_heap\_1 \* v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, winner)  s = s1 or s2 or s3 or s4  return s  for s in range(1,142):  if game(s, 1, 1,1, 2,144) == 1:  print(s)  break |

# Выигрышная стратегия. Задание 2

|  |  |
| --- | --- |
|  | def game(col\_rocks, pl\_position, v\_hod1, v\_hod2, v\_hod3, winner):  #р - позиция игрока. если четная, то значит, ход Вани, если нет - ход Пети  if **pl\_position == 4** and col\_rocks >= winner:  s = 1  elif **pl\_position == 4** and col\_rocks < winner:  s = 0  elif col\_rocks >= winner and **pl\_position < 4**:  s = 0  else:  if pl\_position % 2 **!=** 0: # стратегия победителя  s1 = game(col\_rocks + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s2 = game(col\_rocks + v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s3 = game(col\_rocks \* v\_hod3, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s = s1 or s2 or s3  else: # стратегия проигравшего  s1 = game(col\_rocks + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s2 = game(col\_rocks + v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s3 = game(col\_rocks \* v\_hod3, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s = s1 **and** s2 **and** s3  return s  for s in range(1, 202):  if game(s, 1,1,4,3,202) == 1:  print(s) |

|  |  |
| --- | --- |
|  | def game(col\_rocks, start\_heap\_1, pl\_position, v\_hod1, v\_hod2, winner):  #р - позиция игрока. если четная, то значит, ход Вани, если нет - ход Пети  if **pl\_position == 4** and col\_rocks >= winner:  s = 1  elif **pl\_position == 4** and col\_rocks < winner:  s = 0  elif col\_rocks >= winner and **pl\_position < 4**:  s = 0  else:  if pl\_position % 2 **!=** 0: # стратегия победителя  s1 = game(col\_rocks + v\_hod1, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s2 = game(col\_rocks \* v\_hod2, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s3 = game(col\_rocks, start\_heap\_1 + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s4 = game(col\_rocks, start\_heap\_1 \* v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, winner)  s = s1 or s2 or s3 or s4  else: # стратегия проигравшего  s1 = game(col\_rocks + v\_hod1, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s2 = game(col\_rocks \* v\_hod2, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s3 = game(col\_rocks, start\_heap\_1 + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s4 = game(col\_rocks, start\_heap\_1 \* v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, winner)  s = s1 **and** s2 **and** s3 **and** s4  return s  for s in range(1, 139):  if game(s, 2, 1, 2, 2, 142) == 1:  print(s) |

|  |  |
| --- | --- |
|  | def game(col\_rocks, start\_heap\_1, pl\_position, v\_hod1, v\_hod2, winner): #р - позиция игрока. если четная, то значит, ход Вани, если нет - ход Пети  if **pl\_position == 4** and col\_rocks \* start\_heap\_1 >= winner:  s = 1  elif **pl\_position == 4** and col\_rocks \* start\_heap\_1 < winner:  s = 0  elif col\_rocks \* start\_heap\_1 >= winner and **pl\_position < 4**:  s = 0  else:  if pl\_position % 2 **!=** 0: # стратегия победителя  s1 = game(col\_rocks + v\_hod1, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s2 = game(col\_rocks \* v\_hod2, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s3 = game(col\_rocks, start\_heap\_1 + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s4 = game(col\_rocks, start\_heap\_1 \* v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, winner)  s = s1 or s2 or s3 or s4  else: # стратегия проигравшего  s1 = game(col\_rocks + v\_hod1, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s2 = game(col\_rocks \* v\_hod2, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s3 = game(col\_rocks, start\_heap\_1 + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s4 = game(col\_rocks, start\_heap\_1 \* v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, winner)  s = s1 **and** s2 **and** s3 **and** s4  return s  for s in range(1,142):  if game(s, 1, 1,1, 2,144) == 1:  print(s) |

# Выигрышная стратегия. Задание 3

|  |  |
| --- | --- |
|  | def game(col\_rocks, pl\_position, v\_hod1, v\_hod2, v\_hod3, winner):  #р - позиция игрока. если четная, то значит, ход Вани, если нет - ход Пети  if **(pl\_position == 5 or pl\_position == 3**) and col\_rocks >= winner:  s = 1  elif **pl\_position == 5** and col\_rocks < winner:  s = 0  elif col\_rocks >= winner and **pl\_position < 5**:  s = 0  else:  if pl\_position % 2 == 0: # стратегия победителя  s1 = game(col\_rocks + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s2 = game(col\_rocks + v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s3 = game(col\_rocks \* v\_hod3, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s = s1 or s2 or s3  else: # стратегия проигравшего  s1 = game(col\_rocks + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s2 = game(col\_rocks + v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s3 = game(col\_rocks \* v\_hod3, pl\_position + 1, v\_hod1, v\_hod2, v\_hod3, winner)  s = s1 **and** s2 **and** s3  return s  for s in range(1, 202):  if game(s, 1,1,4,3,202) == 1:  print(s)  **break** |

|  |  |
| --- | --- |
|  | def game(col\_rocks, start\_heap\_1, pl\_position, v\_hod1, v\_hod2, winner):  #р - позиция игрока. если четная, то значит, ход Вани, если нет - ход Пети  if **(pl\_position == 5 or pl\_position == 3**) and col\_rocks >= winner:  s = 1  elif **pl\_position == 5** and col\_rocks < winner:  s = 0  elif col\_rocks >= winner and **pl\_position < 5**:  s = 0  else:  if pl\_position % 2 **==** 0: # стратегия победителя  s1 = game(col\_rocks + v\_hod1, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s2 = game(col\_rocks \* v\_hod2, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s3 = game(col\_rocks, start\_heap\_1 + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s4 = game(col\_rocks, start\_heap\_1 \* v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, winner)  s = s1 or s2 or s3 or s4  else: # стратегия проигравшего  s1 = game(col\_rocks + v\_hod1, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s2 = game(col\_rocks \* v\_hod2, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s3 = game(col\_rocks, start\_heap\_1 + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s4 = game(col\_rocks, start\_heap\_1 \* v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, winner)  s = s1 **and** s2 **and** s3 **and** s4  return s  for s in range(1, 139):  if game(s, 2, 1, 2, 2, 142) == 1:  print(s)  **break** |

|  |  |
| --- | --- |
|  | def game(col\_rocks, start\_heap\_1, pl\_position, v\_hod1, v\_hod2, winner): #р - позиция игрока. если четная, то значит, ход Вани, если нет - ход Пети  if **(pl\_position == 5 or pl\_position == 3**) and col\_rocks >= winner:  s = 1  elif **pl\_position == 5** and col\_rocks < winner:  s = 0  elif col\_rocks >= winner and **pl\_position < 5**:  s = 0  else:  if pl\_position % 2 **!=** 0: # стратегия победителя  s1 = game(col\_rocks + v\_hod1, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s2 = game(col\_rocks \* v\_hod2, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s3 = game(col\_rocks, start\_heap\_1 + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s4 = game(col\_rocks, start\_heap\_1 \* v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, winner)  s = s1 or s2 or s3 or s4  else: # стратегия проигравшего  s1 = game(col\_rocks + v\_hod1, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s2 = game(col\_rocks \* v\_hod2, start\_heap\_1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s3 = game(col\_rocks, start\_heap\_1 + v\_hod1, pl\_position + 1, v\_hod1, v\_hod2, winner)  s4 = game(col\_rocks, start\_heap\_1 \* v\_hod2, pl\_position + 1, v\_hod1, v\_hod2, winner)  s = s1 **and** s2 **and** s3 **and** s4  return s  for s in range(1,142):  if game(s, 1, 1,1, 2,144) == 1:  print(s)  **break** |

# Многопроцессорные системы

# Оператор присваивания и ветвления. Перебор вариантов, построение дерева

|  |  |
| --- | --- |
| def f(x, y):  if x == y:  return 1  elif x > y or x == 18:  return 0  else:  return f(x+1,y) + f(x+4,y) + f(x\*2,y)  print (f(4,11) \* f(11,28)) | def f(x, y):  if x == y:  return 1  elif x < y:  return 0  else:  return f(x-1, y) + f(x//2, y)  print(f(50,20) \* f(20,1)) |

|  |  |
| --- | --- |
| def f(x,y):  if x == y:  return 1  elif x > y:  return 0  else:  return f(x+2,y) + f(x+7,y)  print(f(5,49)) | def f(x, y):  if x == y:  return 1  elif x > y:  return 0  else:  return f(x+2,y) + f(x\*2,y) + f(x\*3,y)  print (f(1,6) \* f(6,24)) |

# Обработка символьных строк

|  |  |
| --- | --- |
| f=open('C:\\Users\\user\\ege\_inf\\EGE-inf\\24\\24\\24var01.txt')  file\_list = f.readline()  f.close()  index\_list = [] # массив для индексов А  min\_posl = 10\*\*10  for i in range(len(file\_list)):# в файле ищем А и их индексы добавляем в cписок  if file\_list[i] == 'A':  index\_list.append(i)  for i in range (len(index\_list)- 2023): # ищем последовательности длиной 2024 символа  #print(index\_list[i])  min\_posl = min(min\_posl, (index\_list[i + 2023] - index\_list[i]) + 1)  # 2024 = 2023 - 0 + 1 нумерация с нуля  print(min\_posl) | Текстовый файл состоит из символов A, B, C, D, E, F и G.  Определите в прилагаемом файле **максимальное** количество идущих подряд символов (длину непрерывной подпоследовательности), среди которых **символ А встречается не более 350** раз.  f=open('C:\\Users\\user\\ege\_inf\\EGE-inf\\24\\24\\24var02.txt')  file\_list = f.readline()  f.close()  index\_list = [] # массив для индексов букв А  max\_posl = 0  for i in range(len(file\_list)):# в файле ищем буквы А и их индексы добавляем в cписок  if file\_list[i] == 'A':  index\_list.append(i)  for i in range (len(index\_list)- 349): # ищем подпоследовательности длиной 350 символов  max\_posl = max(max\_posl, (index\_list[i + 349] - index\_list[i]) + 1)  # 2024 = 2023 - 0 + 1 нумерация с нуля  print(max\_posl) |

|  |  |
| --- | --- |
| Текстовый файл состоит из символов A, B, C, D, E, F и G.  Определите в прилагаемом файле **максимальное** количество идущих подряд символов (длину непрерывной подпоследовательности), среди которых символ **А встречается не более 700 раз**, а символ **Е не встречается** совсем  f=open('C:\\Users\\user\\ege\_inf\\EGE-inf\\24\\24\\24var04.txt')  file\_list = f.readline()  f.close()  file\_list = file\_list.replace("E", "")  #print(file\_list)  index\_list = [] # массив для индексов букв А  max\_posl = 0  for i in range(len(file\_list)):# в файле ищем буквы А и их индексы добавляем в cписок  if file\_list[i] == 'A':  index\_list.append(i)  for i in range (len(index\_list)- 700): # в ищем подпоследовательности длиной 700 символа  max\_posl = max(max\_posl, (index\_list[i + 700] - index\_list[i]) + 1) # 700 = 699 - 0 + 1 нумерация с нуля  print(max\_posl) | Текстовый файл состоит из символов **арабских цифр (0, 1, 2, 3, ....9).** Определите **максимальное** количество идущих подряд символов в прилагаемом файле, среди которых **нет символов 0**, стоящих рядом.  f=open('C:\\Users\\user\\ege\_inf\\EGE-inf\\24\\24\\24var09-12.txt')  file\_list = f.readline()  f.close()  file\_list = file\_list.replace('00', '0 0')  file\_list = file\_list.split(' ')  print( max(map(len, file\_list))) |

|  |  |
| --- | --- |
| Текстовый файл состоит не более чем из **106 символов X, Y и Z**. Определите **максимальное** количество идущих подряд символов, среди которых символ **Z встречается не более двух раз**.  f=open('C:\\Users\\user\\ege\_inf\\EGE-inf\\24\\24\\24var13-17.txt')  file\_list = f.readline()  f.close()  file\_list = file\_list.split('Z')  file\_list = list(map(len, file\_list))  M = 0  for i in range(len(file\_list)-2):  M = max(M, sum(file\_list[i:i+3]) + 2) # +2 так как Z не считается из-за split  print(M) | Текстовый файл состоит не более чем из **106** символов **арабских цифр (0, 1, 2, ..., 9)**. Определите **максимальное** количество идущих подряд цифр, расположенных в **строго возрастающем порядке**.  f=open('C:\\Users\\user\\ege\_inf\\EGE-inf\\24\\24\\24var18-20.txt')  file = f.readline()  f.close()  max\_len = 0  counter = 1  for i in range(1,len(file)):  if file[i-1] > file[i]:  counter +=1  else:  max\_len = max(counter,max\_len)  counter=1  print(max\_len) |

# Обработка целочисленной информации

|  |  |
| --- | --- |
| from fnmatch import \* # fnmatch сравнивает имя файла с заданным шаблоном и возвращает True, если они совпадают, иначе False.  for x in range(0, 10\*\*10, 12007):  if fnmatch(str(x), '9\*?001?1'): # Проверяем полученное число на соответствие  print(x, x // 12007) | counter =0  x = 799999  while counter < 5:  M = 0  for i in range (2, int(x\*\*0.5)+1):  if x % i == 0: # i мин натуральный делитель x  M = x//i - i # x // i макс натуральный делитель x  break  if M % 17 == 0 and M != 0:  counter += 1  print(x, M)  x -= 1 |

|  |  |
| --- | --- |
| counter = 0  x = 850001  while counter < 6:  F = 0  for i in range (2, int(x\*\*0.5)+1):  if x % i == 0:  F = x // i - i  break  if F % 7 == 0 and F != 0:  counter += 1  print(x, F)  x += 1 | def pr(D): # функция, которая проверяет простое ли число  res = False  for i in range(2, int(D\*\*0.5)+1):  if D % i == 0:  res = True  return res  counter = 0  x = 550001  while counter < 6:  for i in range (2, int(x\*\*0.5)+1): # делители числа симметричны  if x % i == 0:  D = x//i # макс дел числа это разность числа и мин дел  if (pr(D)):  counter += 1  print (x, D)  break  x += 1 |

# Обработка целочисленной информации

# Программирование