2 НОМЕР ЕГЭ ПО ИНФОРМАТИКЕ

Кубанский государственный университет
Физико-технический факультет

Теоретическая часть

- Таблицы истинности и порядок выполнения логических операций
- **Для логических операций приняты следующие обозначения:**

операция	пояснение	**в программировании**
1)¬ A, A	не A (отрицание, инверсия)	**not(A)**
2)A ∧ B, A · B	A и B (логическое умножение, конъюнкция)	**A and B**
3)A ∨ B, A + B	A или B (логическое сложение, дизъюнкция)	**A or B**
4)A ⟶ B	импликация (следование)	**A <= B**
5)A ↔ B, A ≡ B, A ∼ B	эквиваленция (эквивалентность, равносильность)	**A==B (python)**
6)A ⊕ B	строгая дизъюнкция	**A != B (python)** **A <> B (pascal)**

Логические переменные		Логические операции					
		отрицание	конъюнкция	дизъюнкция	исключающая дизъюнкция	следование	эквивалентность
A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \oplus B$	$A \rightarrow B$	$A \sim B$
1	0	0	0	1	1	0	0
0	1	1	0	1	1	1	0
0	0	1	0	0	0	1	1
1	1	0	1	1	0	1	1

Что значит 1 и 0?

- 1 – Истина
- 0 – Ложь

Условие по которому строится таблица

2 *№ 12670 (Уровень: Базовый)*

(С. Чайкин) Логическая функция F задаётся выражением $((x \wedge \neg y) \to (z \wedge w)) \wedge ((y \to z) \vee (w \to x))$. На рисунке приведён частично заполненный фрагмент таблицы истинности функции F, содержащий неповторяющиеся строки. Определите, какому столбцу таблицы истинности соответствует каждая из переменных w, x, y, z.

Переменные

?	?	?	?	F
		1	1	0
1		1		0
	0		1	0

Таблица

В ответе напишите буквы w, x, y, z в том порядке, в котором идут соответствующие им столбцы. Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

ЧТО СОДЕРЖИТ В СЕБЕ №2

Типы заданий №2

- 1)Полностью заполненная таблица
- 2)Таблица с пропусками

- Варианты решений
- №1 – вложенный цикл
- №2 – продвинутый уровень

Вложенный цикл

- Этот код не выдает сразу ответ, вам нужно ещё понять, как расставить правильно буквы
- Рассмотрим на примере номера с сайта КЕГЭ

2 *№ 14651 Открытый курс "Слово пацана" (Уровень: Базовый)*

(М. Попков) Логическая функция F задаётся выражением $((z \to x) \equiv (w \to y) \vee (x \wedge w))$. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z, w.

?	?	?	?	F
			1	0
		1	1	0
	1	1	1	0

В ответе напишите буквы x, y, z, w в том порядке, в котором идут соответствующие им столбцы.

```
1  print("x y z w")
2  for x in range(2):
3      for y in range(2):
4          for z in range(2):
5              for w in range(2):
6                  if (((z <= x) == (w <= y)) or (x and w)) == 0:
7                      print(x, y, z, w)
8
```

```
x y z w
0 0 0 1
0 0 1 0
0 1 1 0
0 1 1 1
```

Разберём код

- 1 – Вывод переменных в консоль в кавычках через пробел
- 2 - 5 – 4 вложенных цикла, в которых идёт перебор 0 и 1
- 6 – Условие по которому строится таблица == значению F
- 7 - Вывод переменных в консоль (в том же порядке, как и в первой строчке, но теперь переменные записаны через запятую)

Разберем вывод

- Его нужно сопоставить с таблицей из условия
- Отсюда видно, что первая строчка не подходит
- Сопоставляя столбцы мы видим , что первый столбец = x. Второй = w. Третий = y. Четвертый = z

```
x y z w
0 0 0 1
0 0 1 0
0 1 1 0
0 1 1 1
```

?	?	?	?	F
			1	0
		1	1	0
	1	1	1	0

Продвинутый
Тип первый

- Рассмотрим на примере номера с сайта КЕГЭ

2 *№ 1523 (Уровень: Базовый)*

Логическая функция F задаётся выражением $\neg w \wedge z \wedge (y \to x)$. На рисунке приведён фрагмент таблицы истинности функции F, содержащий **все наборы аргументов**, при которых функция F истинна.

?	?	?	?	F
1	0	0	0	1
1	0	1	0	1
1	0	1	1	1

Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z, w.

```
from itertools import *


1 usage
def f(x, y, z, w):
    return not (w) and z and (y <= x)


table = [(1, 0, 0, 0), (1, 0, 1, 0), (1, 0, 1, 1)]

for p in permutations("xyzw"):
    if [f(**dict(zip(p, r))) for r in table] == [1, 1, 1]:
        print(p)

```

```
('z', 'w', 'x', 'y')
```

Разберем код

- 1 – Использование библиотеки itertools (библиотека для создания перестановок)
- 2 – Функция def f(переменные через запятую):return Условия по которому строится таблица
- 3 – переменная tabl = [(таблица строка 1),(таблица строка 2),(таблица строка n)] – пишем строчки через запятую создавая двумерный массив значений
- 4 – for p in permutations(переменные кавычках): идет по перестановкам переменных
  - if [f(**dict(zip(p, r))) for r in table] == [Значения F в таблице через запятую]: - создаем массив проверок, для этого берём один массив из значений таблицы и переменной, а затем создаем словарь на основе этого массива и открываем его, после чего отправляем в функцию f
  - print(p) – получаем ответ на задание

Тип второй

Рассмотрим на примере номера с сайта КЕГЭ

2 *№ 1525 (Уровень: Базовый)*

Логическая функция F задаётся выражением $((x \to y) \wedge (y \to w)) \vee (z \equiv (x \vee y))$. На рисунке приведён частично заполненный фрагмент таблицы истинности функции F, содержащий **неповторяющиеся строки**. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z, w.

?	?	?	?	F
1			1	0
1				0
	1		1	0

В ответе напишите буквы x, y, z, w в том порядке, в котором идут соответствующие им столбцы. Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

```python
from itertools import *


1 usage
def f(x, y, z, w):
    return ((x <= y) and (y <= w)) or (z == (x or y))


for x1, x2, x3, x4, x5, x6, x7 in product([0, 1], repeat=7):
    table = [(1, x1, x2, 1), (1, x3, x4, x5), (x6, 1, x7, 1)]
    if len(table) == len(set(table)):
        for p in permutations("xyzw"):
            if [f(**dict(zip(p, r))) for r in table] == [0, 0, 0]:
                print(p)
```

```
('y', 'w', 'z', 'x')
```

Разберем код

- 1 – Использование библиотеки itertools (библиотека для создания перестановок)
- 2 – Функция def f(переменные через запятую):return Условия по которому строится таблица
- 3 – for создаем столько переменных, сколько пропусков ([0,1], repeat = количеству пропусков):
  - Создаём переменную tabl и заполняем её по очереди переменными [().(),()]
- 4 – Проверяем, чтобы tabl содержала только уникальные строки
  длина(tabl) == длина(без_повторений(tabl))
- 5 – for p in permutations(переменные кавычках): идет по перестановкам переменных
  - if [f(**dict(zip(p, r))) for r in tabl] == [Значения F в таблице через запятую]: - создаем массив проверок, для этого берём один массив из значений таблицы и переменной, а затем создаем словарь на основе этого массива и открываем его, после чего отправляем в функцию f
  - print(p) – получаем ответ на задание

14 НОМЕР ЕГЭ ПО ИНФОРМАТИКЕ

Кубанский государственный университет
Физико-технический факультет

Теория

- **Что такое системы счисления**
- Системой счисления называется система записи чисел с помощью знаков по определенным правилам.
- Символы, с помощью которых записываются числовые значения, обычно называют цифрами, а все вместе знаки системы счисления образуют алфавит. Количество знаков, используемых для обозначения чисел, называется основанием системы счисления.
- Приведем примеры чисел систем счисления с различным основанием.
- Основная десятичная система, привычная и общеупотребимая, имеет десять символов для обозначения всех чисел, то есть ее основание равно 10. Символы 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 представляют собой цифры. После цифры 9 в числовом ряду идет двузначное 10. При этом происходит сдвиг разрядной сетки числа влево на один разряд.

Позиционные системы счисления

- Рассмотренные системы счисления относятся к классу позиционных систем. В них числовое значение каждой цифры зависит от положения в числе. Например, в десятичном числе 126 единица означает сотню, а в числе 216 единица уже на другом месте и обозначает десять.
- Каждое число позиционной системы счисления можно представить как в свернутом виде, например, 126, так и в развернутом: $1*10^2 + 2*10^1 + 6*10^0$, то есть 100 + 20 +6 =126.
- Аналогично, двоичное число 111001 = $1*2^5 + 1*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0$
- Восьмеричное число: 247 = $2*8^2 + 4*8^1 + 7*8^0$
- Шестнадцатеричное число: 2A5F = $2*16^3 + A*16^2 + 5*16^1 + F*16^0$
- Используя развернутую форму, можно переводить числа из любой системы счисления в десятичную систему.

Рассмотрим номер 14 в ЕГЭ

- Существует два вида номера 14

1. Задание с подбором x для ответа
2. Подсчёт сложной суммы

Тип первый

- Рассмотрим типовое решение номер в качестве примера будет взят номер на сайте КЕГЭ

14 *№ 13243 Открытый курс "Слово пацана" (Уровень: Базовый)*

(М. Попков) Операнды арифметического выражения записаны в системе счисления с основанием 18.

$1x253_{18} + 32x8_{18}$

В записи чисел переменной x обозначена неизвестная цифра из алфавита 18-ричной системы счисления. Определите наименьшее значение x, при котором значение данного арифметического выражения кратно 7. Для найденного значения x вычислите частное от деления значения арифметического выражения на 7 и укажите его в ответе в десятичной системе счисления. Основание системы счисления в ответе указывать не нужно.

```
for x in range(18):
    a = list(reversed([1,x,2,5,3]))
    b = list(reversed([3,2,x,8]))
    for i in range(len(a)):
        a[i] = a[i]*18**i
    for i in range(len(b)):
        b[i] = b[i]*18**i
    a = sum(a)
    b = sum(b)
    if (a+b)%7==0:
        print((a+b)//7)
```

Выходные данные:

```
19367
25217
31067
```

Разберём код

1. for переменная in range(система исчисления):
2. a, b – это числа, которые мы записываем в лист и переворачиваем его (записываем слева на право через запятую)
3. Переводим из любой системы счисления в десятичную с помощью позиционного способа
4. Получаем список, в котором хранятся члены суммы. Чтобы найти число в десятичной системе счисления, необходимо посчитать сумму массива, можем записать как a = sum(a)
5. Дальше пишем условие, проверяя на кратность. Чтобы проверить, что число кратно, используется запись: число%(кратное число) == 0 , то есть (21%7 == 0) – это True(правда)

◦Выходные данные:

- Берем то значение, которое нужно по условию: если минимальное, то берем первое, если максимальное - последнее

Тип второй

Рассмотрим типовое решение номер в качестве примера будет взят номер на сайте КЕГЭ

14 *№ 12923 PRO100 ЕГЭ 26.01.24 (Уровень: Базовый)*

Значение арифметического выражения $3 \times 3125^9 + 2 \times 625^8 - 4 \times 625^7 + 3 \times 125^6 - 2 \times 25^5 - 2024$ записали в системе счисления с основанием 25. Сколько значащих нулей содержится в этой записи?

```
x = 3 * 3125 ** 9 + 2 * 625 ** 8 - 4 * 625 ** 7 + 3 * 125 ** 6 - 2 * 25 ** 5 - 2024
a = []
while x > 0:
    a += [x % 25]
    x = x // 25
print(a.count(0))
```

Разберём код

1. Запишем в переменную x арифметическое выражение (на языке Python значок степени обозначается , как двойное умножение **)
2. Создадим пустой массив
3. Пока x > 0, в массив мы записываем x%(система счисления в которую надо переводить)
4. Уменьшаем x = x//(система счисления в которую надо переводить)
5. Выводим a.count(то значение, которое надо)

- count – количество элементов, которые надо найти в списке или в строчке

15 НОМЕР ЕГЭ ПО ИНФОРМАТИКЕ

Кубанский государственный университет

Физико-технический факультет

Теоретическая часть

операция	пояснение	в программировании
1)¬ A, A	не A (отрицание, инверсия)	not(A)
2)A ∧ B, A · B	A и B (логическое умножение, конъюнкция)	A and B
3)A ∨ B, A + B	A или B (логическое сложение, дизъюнкция)	A or B
4)A ⟶ B	импликация (следование)	A <= B
5)A ↔ B, A ≡ B, A ~ B	эквиваленция (эквивалентность, равносильность)	A==B (python)
6)A ⊕ B	строгая дизъюнкция	A != B (python)

- Таблицы истинности и порядок выполнения логических операций
- **Для логических операций приняты следующие обозначения:**

РАЗНОВИДНОСТИ 15 НОМЕРА

1) Функции
2) Дороги
3) Гибрид

Тип 1 функции

- Рассмотрим на примере номера с сайта КЕГЭ

15 *№ 11665 (Уровень: Базовый)*

(Л. Шастин) Обозначим через $\mathrm{mod}(m, n)$ остаток от деления m на n. Для какого наименьшего натурального числа A выражение

$$(A + x > 700 - A) \wedge (\mathrm{mod}(A, 100) + \mathrm{mod}(100, x) > 50)$$

тождественно истинно, т.е. принимает значение 1 при любом натуральном значении переменной x?

```
1 usage
def f(x, A):
    return ((A + x > 700 - A) and ((A % 100) + (100 % x) > 50)) == 1


for a in range(1, 1000):
    if all(f(x, a) for x in range(1, 1000)):
        print(a)
        break
```

Разберём код

- Создаём функцию f, которая будет возвращать(return) значения, либо True, либо False самой функции
- Пишем цикл для переменной a, чтобы подобрать нужное значение
- Если все x для a удовлетворяют условию, то выводим a и заканчиваем цикл

- Рассмотрим на примере номера с сайта КЕГЭ

№ 7846 Danov2304 (Уровень: Базовый)

(А.Богданов) На числовой прямой даны два отрезка: P = [13; 19] и Q = [17; 23]. Укажите наибольшую возможную длину такого отрезка A, что формула

$\neg(\neg(x \in P) \rightarrow (x \in Q)) \rightarrow ((x \in A) \rightarrow (\neg(x \in Q) \rightarrow (x \in P)))$

тождественно истинна, то есть принимает значение 1 при любых x.

Показать ответ

Разбор

10

```
1 usage
def f(x,a):
    p = list(range(13, 20))
    q = list(range(17, 24))
    return ((not (not (x in p) <= (x in q))) <= ((x == a) <= ((not (x in q)) <= (x in p)))) == 1

ar = []
for a in range(1,100):
    if all(f(x,a) for x in range(1,100)):
        ar +=[a]
print(max(ar)-min(ar))
```

Разберём код

- Создаём функцию в которой содержатся отрезки и функция, которая возвращает значение True в случае выполнения условия
- Создаём пустой список ar
- Идём по всем возможным значениям a, они должны быть больше, чем промежутки
- Если все x для a удовлетворяют условию, то добавляем a в список ar
- Выводим **максимальную** длину списка, отняв максимальное значение от минимального

Тип 3 гибрид

- Рассмотрим на примере номера с сайта КЕГЭ

№ 8159 /dev/inf 05.23 (Уровень: Базовый)

(А. Рогов) На числовой прямой дан отрезок B = [50; 70]. Для какого **наименьшего** целого неотрицательного числа A логическое выражение

$$(2x + y \neq 150) \vee (x \notin B) \vee (A > y)$$

тождественно истинно (т.е. принимает значение 1) при любых целых неотрицательных x и y?

Показать ответ

Разбор

51

```
def f(a,x,y):
    return ((2*x+y)!=150)or(not(50<=x<=70))or(a>y)

for a in range(0,1000):
    if all(f(a,x,y) for x in range(0,1000) for y in range(0,1000)):
        print(a)
        break
```

Разберём код

- Создаём функцию и возвращаем значение логического выражения
- Пишем цикл для a
- Проверяем: если все x и y подходят, то выводим a

16 НОМЕР ЕГЭ
ИНФОРМАТИКА

Кубанский
государственный
университет

Физико-технический
факультет

- **Что такое рекурсия в программировании**
- Рекурсия — это функция, которая вызывает саму себя.
- Представим, что есть функция А, которая выполняет определённое действие, — например, перемножает два значения. Внутри этой функции А в качестве одного из значений для умножения возьмём ту же самую функцию А. Получается, что функция умножает число на саму себя и внутри ещё раз умножает число на саму себя и так далее, до бесконечности или выполнения определённого условия.

Функции

- **Функция** — это фрагмент программного кода, который решает какую-либо задачу.
- Ключевое слово def в начале функции сообщает интерпретатору о том, что следующий за ним код — есть её определение. Всё вместе — это объявление функции.
- объявим функцию my_function()
- def my_function():
- тело функции программного кода, который решает какую-либо задачу.
- Функция может вызывать себя образуя рекурсию, так рекурсии бывают двух видов рекурсия бесконечная и с условием выхода

Рассмотрим номер 16

- Существует два типа номеров:

1. С одной функцией
2. С двумя функциями

- Также, в номерах лучше указывать глубину рекурсии с использованием модуля sys ඞ

Тип 1

- Рассмотрим на примере номера с сайта КЕГЭ

16 *№ 12481 (Уровень: Базовый)*

(Л. Шастин) Алгоритм вычисления значения функции $F(n)$, где n – целое число, задан следующими соотношениями:

$F(n) = 4$, если $n < 5$

$F(n) = 4 \times F(n-4)$, если $n > 4$

Тело функции

Чему равно значение выражения $F(4444)/F(4400)$?

```
import sys

sys.setrecursionlimit(5000)



3 usages
def F(n):
    if n < 5:
        return 4
    if n > 4:
        return 4 * F(n - 4)


print(F(4444) / F(4400))
```

Разберём код

1. Импортируем модуль sys, чтобы установить глубину рекурсии, иначе возникнет ошибка

```
RecursionError: maximum recursion depth exceeded in comparison
```

2. Устанавливаем глубину рекурсии [sys.setrecursionlimit(1000000)]
3. Пишем условие, заменяя "F(n) =" на return, а также прописывая условия рекурсии: если n > 4, то возвращаем значение 4*F(n-4)
4. Выводим нужные значения, вызывая функцию с определённым значением внутри, и записываем ответ

Тип 2

Рассмотрим на примере номера с сайта КЕГЭ

16 *№ 13297 Открытый курс "Слово пацана" (Уровень: Базовый)*

(М. Попков) Алгоритм вычисления функций $F(n)$ и $G(n)$ задан следующими соотношениями:

$F(n) = G(n) = 1$ при $n = 3$

$F(n) = 5 \times F(n-1) + 6 \times G(n-1) - 3n + 8$ при $n > 3$

$G(n) = 6 \times F(n-1) + 5 \times G(n-1) + 3$ при $n > 3$

Определите число, которое получится, если в обе функции передать аргумент $n = 9$ и сложить получившиеся значения.

```python
import sys
sys.setrecursionlimit(100000)
3 usages
def F(n):
    if n == 3:
        return 1
    if n > 3:
        return 5 * F(n - 1) + 6 * G(n - 1) - 3 * n + 8
3 usages
def G(n):
    if n == 3:
        return 1
    if n > 3:
        return 6 * F(n - 1) + 5 * G(n - 1) + 3

print(F(9) + G(9))
```

Разберём код

1. Импортируем модуль sys, чтобы установить глубину рекурсии, иначе возникнет ошибка. Устанавливаем глубину рекурсии sys.setrecursionlimit(1000000)
2. Описываем первую функцию(def F(n)), как в условии, заменяя "F(n) =" на return. После, вторую функцию (def G(n)), как в условии, заменяя "G(n) =" на return
3. Выводим нужные значения, вызывая функцию с определённым значением внутри, и записываем ответ

17 НОМЕР ЕГЭ
ИНФОРМАТИКА

2024

КУБГУ

Кубанский государственный университет

Физико-технический факультет

Теория

- Файл — это всего лишь набор данных, сохраненный в виде последовательности битов на компьютере. Информация хранится в куче данных (структура данных) и имеет название «имя файла» (filename).
- Текстовые файлы
- Это файлы с читаемым содержимым. В них хранятся последовательности символов, которые понимает человек. Блокнот и другие стандартные редакторы умеют читать и редактировать этот тип файлов.

- Текст может храниться в двух форматах: (.txt) — простой текст и (.rtf) — «формат обогащенного текста».

- Открытие файла
- Метод open()
- В Python есть встроенная функция open(). С ее помощью можно открыть любой файл на компьютере. Технически Python создает на его основе объект.
- Синтаксис следующий:
- f = open(file_name, access_mode)
- Где,file_name = имя открываемого файла
- access_mode = режим открытия файла. Он может быть: для чтения, записи и т. д. По умолчанию используется режим чтения (r), если другое не указано. Далее полный список режимов открытия файла

- Полный путь можно узнать так:

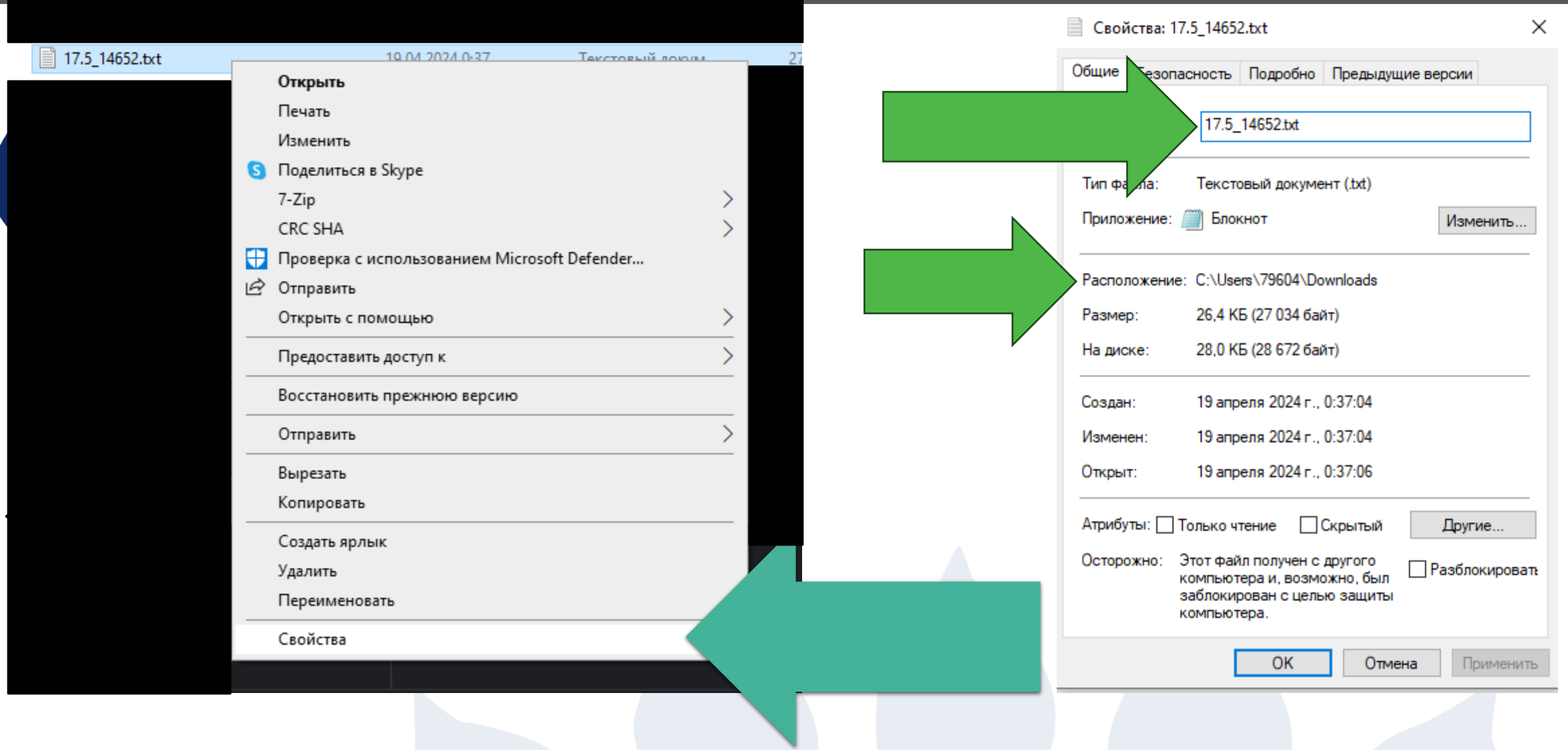
1) Нажав правой кнопкой мыши на файл

2) Открыть свойство

3) Расположение

4) Добавить имя файла поставив перед ним \

- Таким образом получим(например)
- C:\Users\79604\Downloads\kubgu.pptx



Таким образом полный путь будет выглядеть так:
C:\Users\79604\Downloads\17.5_14652.txt

Рассмотрим решение №1

- Рассмотрим решение на примере номера с сайта КЕГЭ

17 *№ 14652 Открытый курс "Слово пацана" (Уровень: Базовый)*

(М. Попков) 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 квадратом натурального числа, затем максимальную из сумм элементов таких пар. В данной задаче под парой подразумевается два идущих подряд элемента последовательности.

Файлы к заданию: 17.5.txt

```
f = open("C:/Users/79604/Downloads/17.5_14652.txt")
s = [int(x) for x in f.readlines()]
ar = []
for i in range(len(s) - 1):
    if (s[i] > 0 and (s[i] ** 0.5) == int((s[i] ** 0.5))) or (
            s[i + 1] > 0 and (s[i + 1] ** 0.5) == int((s[i + 1] ** 0.5))):
        ar += [s[i] + s[i + 1]]
print(len(ar), max(ar))
```

Разберём код

1. Открываем файл (пишем полный путь меняя \ на / или \\ , потому что \ является спец. символом на языке Python)
2. Создаём список значений, который строится по чтению линий из файла
3. Создаём пустой список, в который потом будут помещены подходящие значения для решения
4. Берем цикл длинною в столько шагов, сколько у нас значений в массиве, и отнимаем от него 1, если по задачи нужно взять пару чисел, отнимаем 2, если нужно взять тройку чисел, отнимаем 3, если нужно взять четверку чисел и т.д.
5. Пишем условие проверки, которое указано в варианте, например: в данной задаче мы берем корень числа и сравниваем его с целой частью от корня
6. Записываем сумму значений в пустой список
7. Выводим нужные значения из списка, например: len(имя_списка)- длина списка(сколько всего значений),
   max(имя_списка) – максимальное значение списка, min(имя_списка) – минимальное значение.

Рассмотрим решение №2

- Рассмотрим решение на примере номера с сайта КЕГЭ

17

№ 14653 Открытый курс "Слово пацана" (Уровень: Сложный)

(М. Попков) В файле содержится последовательность целых чисел. Элементы последовательности могут принимать целые значения от -10 000 до 10 000 включительно. Определите количество четверок последовательности, в которых только два элемента являются трехзначными числами, ровно один элемент делится на 18, а сумма элементов делится на сумму двух минимальных положительных элементов последовательности, кратных 17, и произведение элементов не превосходит квадрат максимального элемента последовательности, оканчивающегося на 69. В ответе запишите количество найденных четверок чисел, затем минимальный из квадратов сумм элементов таких четверок. В данной задаче под четверкой подразумевается четыре идущих подряд элемента последовательности.

Файлы к заданию: 17.16.txt

```python
f = open("C:/Users/79604/Downloads/17.16_14653.txt")
s = [int(x) for x in f.readlines()]
ar = []

min_17 = []
max_69 = []

for i in range(len(s)):
    if s[i] % 17 == 0 and s[i] > 0:
        min_17 += [s[i]]
    if str(s[i])[-2:] == "69":
        max_69 += [s[i]]

min_17.sort()
min_17 = min_17[0] + min_17[1]
max_69.sort()
max_69 = max_69[-1]

for i in range(len(s) - 3):
    if ((len(str(abs(s[i]))) == 3) + (len(str(abs(s[i + 1]))) == 3) + (len(str(abs(s[i + 2]))) == 3) + (len(str(abs(s[i + 3]))) == 3)
        if ((s[i] % 18 == 0) + (s[i + 1] % 18 == 0) + (s[i + 2] % 18 == 0) + (s[i + 3] % 18 == 0)) == 1:
            if (s[i] + s[i + 1] + s[i + 2] + s[i + 3]) % min_17 == 0:
                if (s[i] * s[i + 1] * s[i + 2] * s[i + 3]) <= max_69 ** 2:
                    ar += [(s[i] + s[i + 1] + s[i + 2] + s[i + 3]) ** 2]

print(len(ar), min(ar))
```

Разберём код

1. Открываем файл (пишем полный путь меняя \ на / или \\ , потому что \ является спец символом на языке Python)
2. Создаём список значений, который строится по чтению линий из файла
3. Создаём пустой список, в который потом будут помещены подходящие значения для решения
4. Создаём списки в которые мы будем записывать числа, которые делятся на 17 и числа, которые заканчиваются на 69

5. Идём по каждому элементу файла
    1) добавляем все элементы , которые делятся на 17
    2) добавляем все элементы , которые оканчиваются на 69
6. Сортируем два получившихся списка
7. Определяем переменные с нужными значениями

    1) минимальная сумма двух положительных первых элементов, которые делятся на 17

    2) берём максимальный элемент из списка чисел, которые оканчиваются на 69, то есть последний

1. Берем цикл длинною в столько шагов, сколько у нас значений в массиве, и отнимаем от него 3, потому что надо брать четвёрку чисел
8. Проверяем, чтобы длина только двух элементов была равна 3 с помощью сложения True или False, где True = 1 False = 0
9. Проверяем, что кратность только одного элемента равна 18 с помощью сложения True или False (не забываем скобки, так как операция процента имеет не самый высокий приоритет)

11. Проверяем, чтобы сумма четырёх элементов была кратна сумме минимальных значений
12. Проверяем, чтобы произведение четырёх элементов было меньше, либо равно максимальному числу, которое заканчивается на 69 в квадрате
13. Записываем в пустой список квадрат суммы
14. Выводим длину получившегося списка и его минимальный элемент

19-21 НОМЕРА ЕГЭ ПО ИНФОРМАТИКЕ

Кубанский государственный университет
Физико-технический факультет

Теоретическая часть

- 19-21 задачи - это теория игр, где чаще всего встречается игры с кучами камней
- Играют 2 игрока, Петя и Ваня (Первый и Второй)
- У каждого игрока есть возможность сделать ход (Добавить n камней в кучу, умножить кол-во камней и др.)
- Чаще всего, условие победы - набрать больше всего камней в сумме

Задачи с двумя кучами
Аналитическое решение

19
20
21

(№ 2412) Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежат две кучи камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может **добавить** в одну из куч **два камня** или **увеличить** количество камней в куче **в два раза**. Чтобы делать ходы, у каждого игрока есть неограниченное количество камней. Игра завершается в тот момент, когда суммарное количество камней в кучах становится не менее 73. Победителем считается игрок, сделавший последний ход, т. е. первым получивший позицию, в которой в кучах будет 73 или больше камней.

В начальный момент в первой куче было 9 камней, во второй куче – S камней, $1 \le S \le 63$. Будем говорить, что игрок имеет выигрышную стратегию, если он может выиграть при любых ходах противника.

Ответьте на следующие вопросы:

Вопрос 1. Известно, что Ваня выиграл своим первым ходом после неудачного первого хода Пети. Назовите минимальное значение S, при котором это возможно.

Вопрос 2. Укажите минимальное значение S, при котором у Пети есть выигрышная стратегия, причём Петя не может выиграть первым ходом, но может выиграть своим вторым ходом независимо от того, как будет ходить Ваня.

Вопрос 3. Найдите два значения S, при которых у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Пети, и при этом у Вани нет стратегии, которая позволит ему гарантированно выиграть первым ходом. Найденные значения запишите в ответе в порядке возрастания.

1. Определим "выигрышный промежуток"

- Назовём выигрышным промежутком некоторое множество значений S (количества камней в куче), сделав из которых свой самый сильный ход, игрок сможет выиграть
- Для игроков доступны ходы:
  +2 в любую в кучу
  *2 любую кучу
- Сильным ходом для нас будет умножение на 2
- Изначально у нас была позиция: (9, S)
- Чтобы выиграть, необходимо в сумме набрать 73 или более камней
- Если взять S = 32, то умножив вторую кучу на 2, мы получим позицию (9, 64), она в сумме даст 73, тем самым игрок сможет выиграть
- Итого, выигрышный промежуток, то есть значения S, при котором сделав самый сильный ход можно выиграть, это [32, 63]

2. Неудачный ход Пети

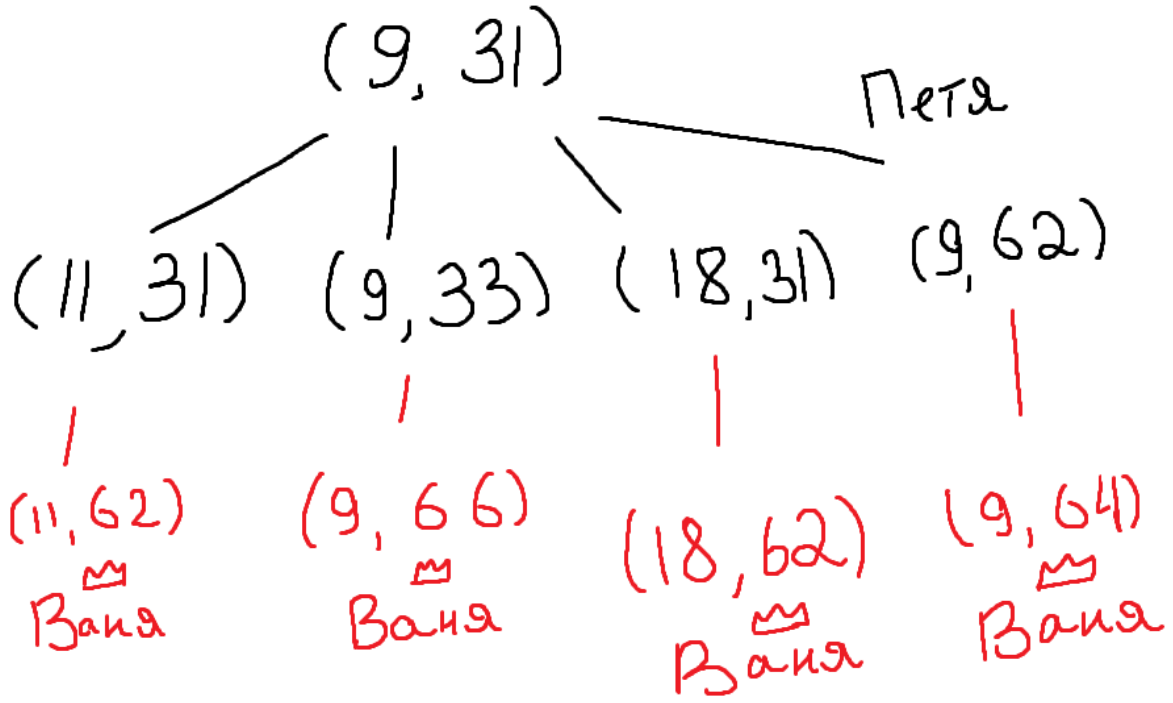
- Неудачный ход - это такой ход, сделав который, мы позволяем противнику выиграть
- Необходимо определить минимальное значение S, при котором Ваня выиграет, учитывая, что Петя сыграет неудачно
- Заметим, что если Петя своим ходом сделает так, что во 2 куче будет 32 камня, то Ваня, сделав свой первый сильный ход *2, выиграет, получив позицию (9, 64)
- Значит, если изначально S = 16, то из позиции (9, 16) неудачным ходом Пети будет считаться умножение 2-ой кучи на 2: (9, 16) -> (9, 32)
- **№19. Ответ: 16**

3. Выигрышная стратегия для Пети

- №20 довольно близко связан с "выигрышным промежутком"
- Чтобы Петя мог выиграть своим вторым ходом, он должен "подставить" Ваню, то есть поставить в такую позицию, из которой его самый сильный ход не давал ему победы, а самый слабый - не позволял Пете проиграть
- Чаще всего, значение S можно подобрать сделав "обратный ход" от минимального значения выигрышного промежутка, например S=32-2=30
- Также, нам необходимо построить дерево решений для наглядного анализа ходов
- Дерево решений - это модель, позволяющая рассмотреть все игровые случаи
- Для примера, построим дерево решений при S = 31

Дерево решений для S = 31

Как мы видим, при любом ходе Пети, Ваня может выиграть, что нас не устраивает!

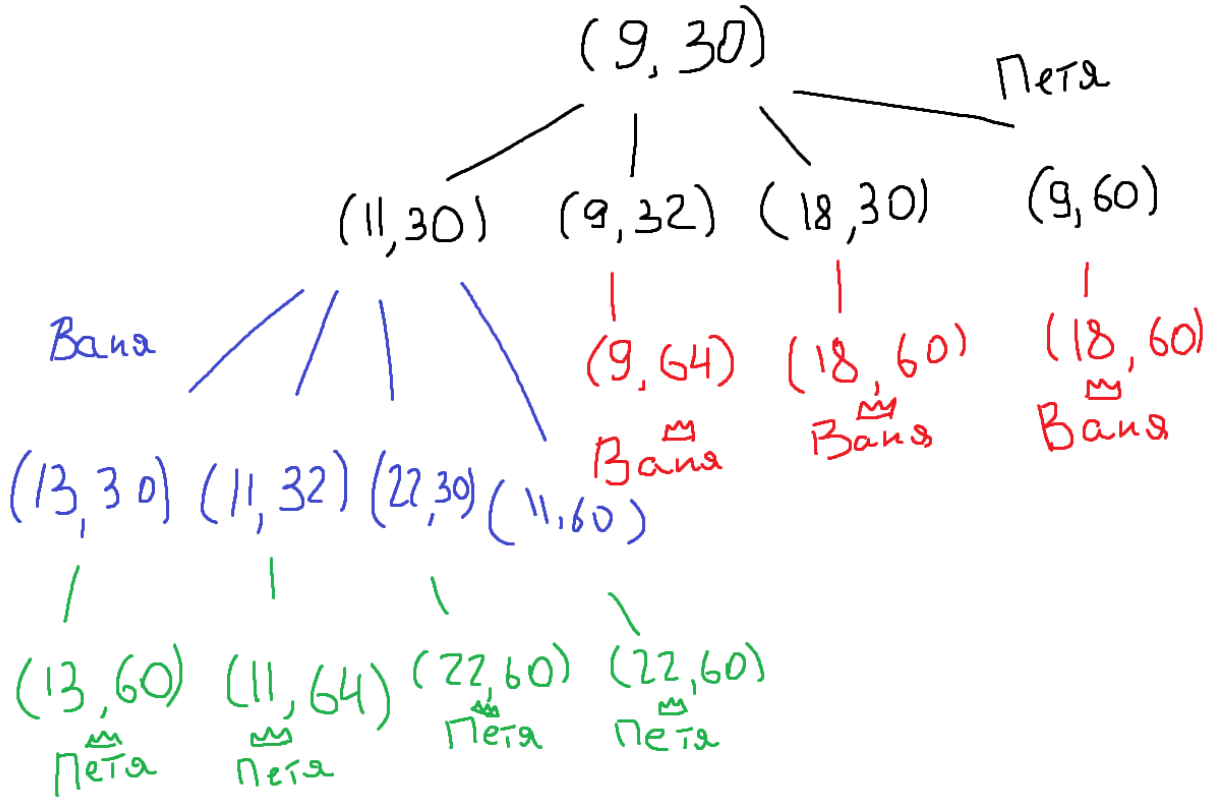


Дерево решений для S = 30

Если во второй куче изначально было 30 камней, то у Пети есть выигрышная стратегия, при которой он выигрывает 2 ходом

Позиция (11,30) будет неудачной для противника, так как из неё нет победной стратегии.

ОДНАКО, от нас требуют минимальное значение S!



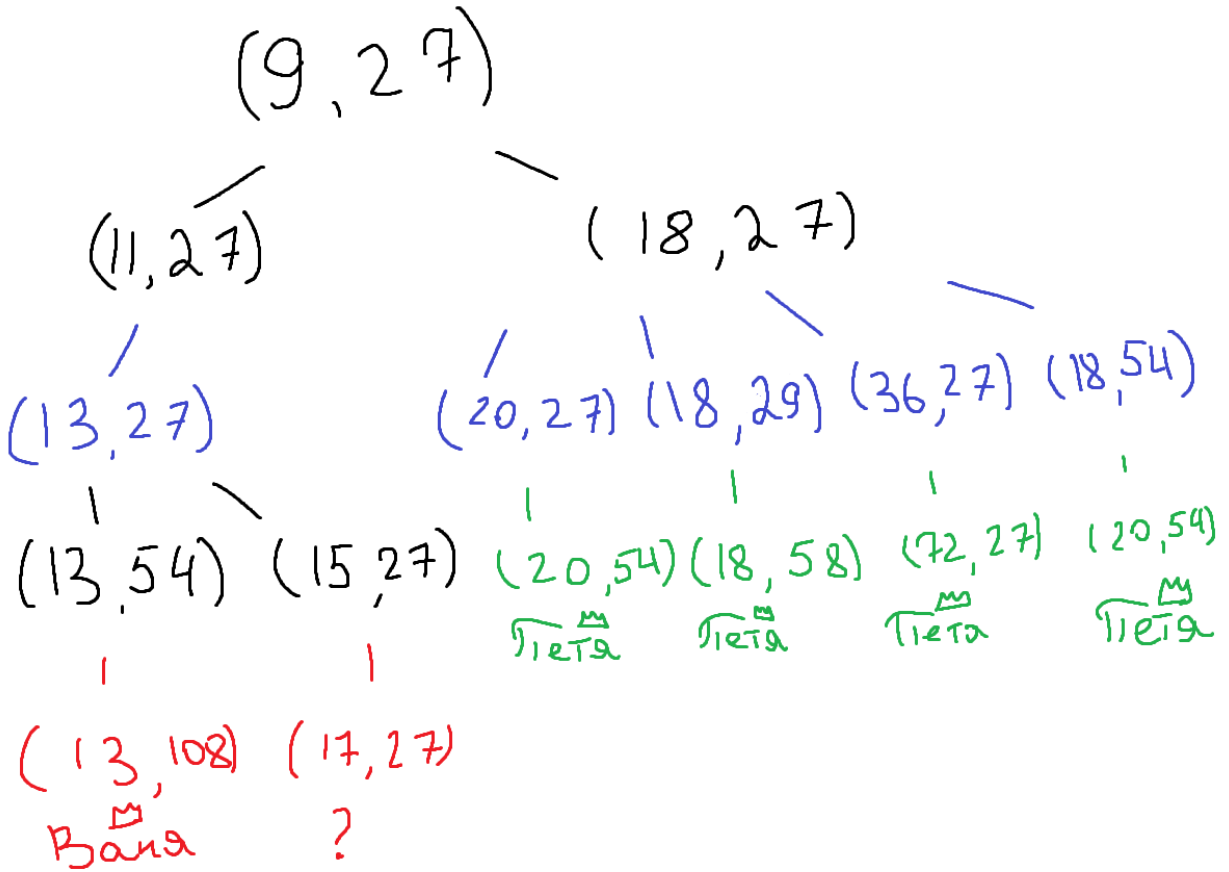
Дерево решений для S = 27

Заметим, что в предыдущем решении Петя может сделать ход +2 в 1-ую кучу.

Попробуем найти S, при котором выгодным решением будет сделать ход *2 для первой кучи

Еще одним оптимальным вариантом будет S=29, но минимальным считается именно S=27

Так же заметим, что позиция (18,27) считается "неудачной", то есть из неё НЕВОЗМОЖНО выиграть!



№20. Ответ: 27

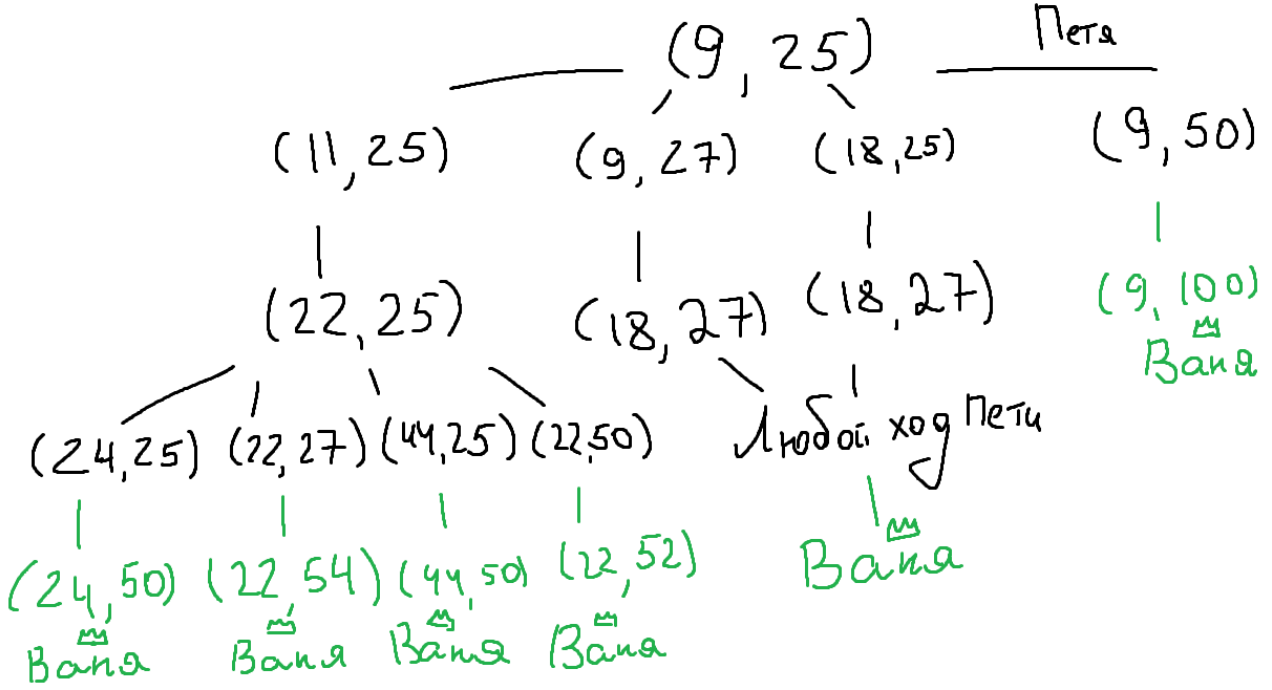
Решение №21

Ответ на №21 напрямую зависит от ответа, который был найден в №20

Чтобы найти ответ, чаще всего нам необходимо сделать обратный слабый ход

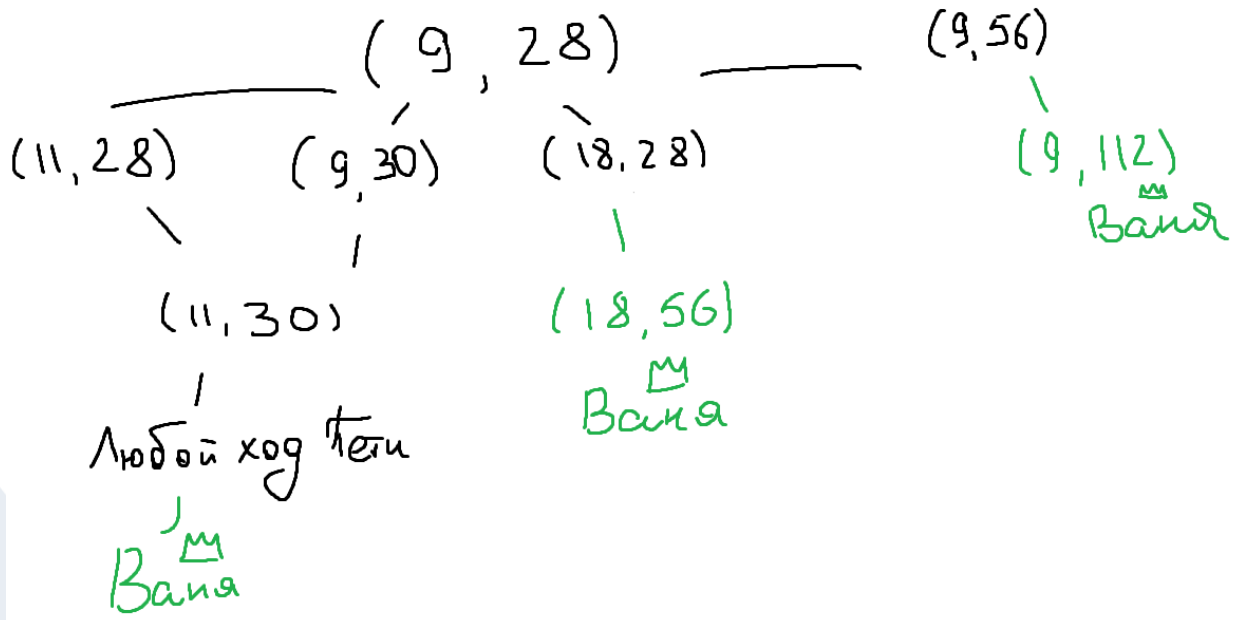
Так, относительно S=27, обратным слабым ходом будет -2, то есть S=25

Давайте рассмотрим дерево решений для S=25



Решение №21

По аналогии,
рассмотрим
S = 30 – 2 = 28



№21. Ответ: 25, 28

Задачи с двумя кучами
Алгоритмическое решение

19
20
21

(№ 6292) Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежат две кучи камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в большую кучу любое количество камней от одного до трёх или удвоить количество камней в меньшей куче. Если кучи содержат равное количество камней, можно добавить в любую из них от одного до трёх
камней, удвоение в этой ситуации запрещено.
Игра завершается, когда общее количество камней в двух кучах становится больше или равно 60. Победителем считается игрок, сделавший последний ход, то есть первым получивший 60 или больше камней в двух кучах.
Ответьте на следующие вопросы:

Вопрос 1. Известно, что Петя смог выиграть первым ходом. Какое наименьшее число камней могло быть суммарно в двух кучах?

Вопрос 2. Известно, что в первой куче 12 камней, а во второй – S камней ($1 \le S \le 47$). Найдите наименьшее и наибольшее значения S, при которых у Пети есть выигрышная стратегия, причём одновременно выполняются два условия:
– Петя не может выиграть за один ход;
– Петя может выиграть своим вторым ходом независимо от того, как будет ходить Ваня.
Найденные значения запишите в ответе в порядке возрастания.

Решение в Python

Для начала добавим рекурсивную функцию, с помощью которой мы будем перебирать все возможные ходы

```python
def f(a, b, m):
    if a + b >= 60:
        return m % 2 == 0
    if m==0:
        return 0
    h = []
    if a>b:
        h = [f(a+1,b,m-1), f(a+2,b,m-1), f(a+3,b,m-1), f(a,b*2,m-1)]
    elif b>a:
        h = [f(a,b+1,m-1), f(a,b+2,m-1), f(a,b+3,m-1), f(a*2,b,m-1)]
    else:
        h = [f(a+1,b,m-1), f(a+2,b,m-1), f(a+3,b,m-1),
             f(a,b+1,m-1), f(a,b+2,m-1), f(a,b+3,m-1)]
    return any(h) if m%2!=0 else all(h)
```

Решение в Python

Затем, будем вызывать рекурсивную функцию в зависимости от того, что нам нужно

```
16  sol_19 = []
17  for a in range(1, 60):
18      a_sol = [a+b for b in range(1, 60) if f(a,b,1)]
19      if a_sol:
20          sol_19 += [min(a_sol)]
21  print('19: ', min(sol_19))
22  print('20: ', [s for s in range(1, 48) if not f(12,s,1) and f(12,s,3)])
23  print('21: ', [s for s in range(1, 35) if not f(25,s,2) and f(25,s,4)])
```