

# ЕГЭ по информатике: 19-21 номер

Теория игр. Смысл игровых стратегий за 5 минут. Решение задач через Python



## Содержание

- Смысл игровых стратегий
- Основные моменты написания программ на языке Python

Решение задач на теорию игр:

1. Номера 19-21 на одну кучу
2. Номера 19-21 на одну кучу с ограничением сверху
3. Номера 19-21 на две кучи (удачный и неудачный ход соперника)

### **Смысл игровых стратегий**

Игровые модели описывают соперничество двух (или более) сторон, каждая из которых стремится выиграть. Выигрыш одной стороны часто означает проигрыш другой. Изучением

таких моделей занимается теория игр – раздел прикладной математики. Её задача состоит в том, чтобы найти такую стратегию (алгоритм), который позволит тому или другому участнику получить наибольший выигрыш, при том, что соперники играют безошибочно. Всего есть два положения игры: выигрышная позиция и проигрышная. Таким образом, сформируем основные понятия.

Стратегия – это алгоритм игры, который позволяет добиться цели в игре, в предположении, что соперники играют безошибочно.

*Безошибочные ходы* – это такие ходы, которые ставят игрока в максимально выгодное положение, то есть ни один из игроков не поддаётся сопернику и ходит с большей выгодой только для себя.

Выигрышная позиция – это такая позиция, в которой игрок, делающий первый ход, может победить при любой игре соперника, если не допустит ошибку.

Проигрышная позиция – это такая позиция, в которой игрок, делающий ход, проиграет со 100% вероятностью, если его соперник не сделает ошибку.

Тогда ваша общая стратегия игры состоит в том, чтобы своим ходом поставить проигрышную позицию сопернику. Примером послужат такие игры, как шахматы, крестики-нолики, карточные игры, морской бой и так далее.

А теперь рассмотрим пару ситуаций в игре крестики-нолики, чтобы наглядно увидеть принцип игровых стратегий.

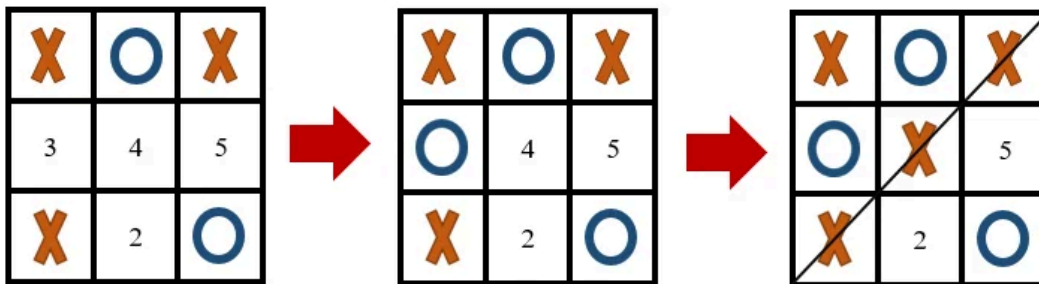
### 1) Ход крестиков.

X	O	X
3	4	5
1	2	O

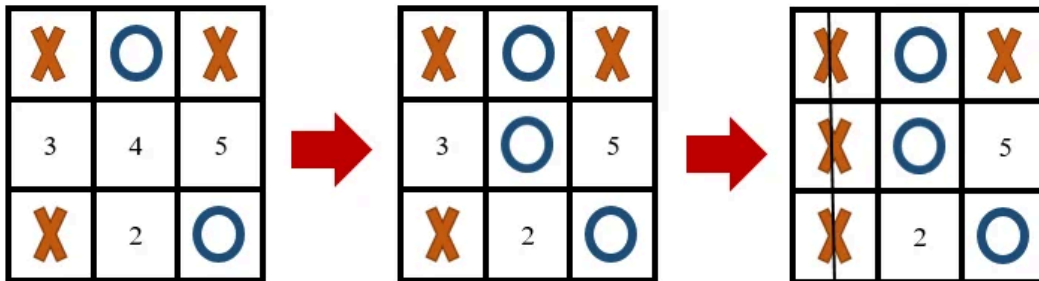
В данном случае у крестиков есть 5 вариантов ходов, 1 из которых 100% выигрышный (рассматриваем только выигрышные при условии, что соперник, то есть “нолики”, ходит без ошибок, но это не всегда может быть так). Походив в поле “1”, при любой игре ноликов, крестики победят. Тогда позиция крестиков – выигрышная, а позиция ноликов после следующего хода крестиков – проигрышная.

### 2) Ход ноликов.

Возможный вариант исхода, если следующими ходят нолики:



Второй возможный исход, если следующими ходят нолики:



## Основные моменты написания программ на Python

Для написания программы нам понадобятся базовые знания Python, которые я подобрала ниже.

Функция def – объявляет функцию. Когда функция вызывается, поток выполнения программы переходит к ее определению и начинает исполнять ее тело.

Оператор return – возврат значений из функции. Программа "забирает" значение, указанное после этой команды, и "уходит" из функции.

Цикл for – (с англ.) “для”, используется для перебора последовательности (то есть списка, кортежа, словаря, набора или строки). С помощью for мы можем выполнить набор операторов, один раз для каждого элемента.

all – (с англ.) “все”

any – (с англ.) “любой”, хотя бы один.

if – с его помощью ставятся все условия. “если...”

print – выводит на экран.

Символ	Смысл	Пример
<code>==</code>	равно	<code>a == b</code>
<code>!=</code>	не равно	<code>a != b</code>
<code>and</code>	конъюнкция (и)	<code>a &gt; 5 and b &lt; 5</code>
<code>or</code>	дизъюнкция (или)	<code>a &gt; 5 or b &lt; 5</code>

Значение range	Сгенерированная последовательность
<code>range(5)</code>	0, 1, 2, 3, 4
<code>range(3, 8)</code>	3, 4, 5, 6, 7
<code>range(10, 0, -1)</code>	10, 9, 8, 7, 6, 5, 4, 3, 2, 1

## Решение задач на теорию игр:

### Номера 19-21 НА ОДНУ КУЧУ

№ 1. Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может *добавить в кучу два камня или увеличить количество камней в куче в два раза*. У каждого игрока, чтобы делать ходы, есть неограниченное количество камней. Игра завершается в тот момент, когда количество камней в куче становится не менее 25. Победителем считается игрок, сделавший последний ход, то есть первым получивший кучу, в которой будет 25 или больше камней. В начальный момент в куче было  $S$  камней,  $1 \leq S \leq 24$ .

Введу обозначения:

Первый ход Пети – 'P1'

Первый ход Вани – 'V1'

Точно так же с последующими ходами.

(Сама программа выделена **синим** цветом, а все объяснения чёрным шрифтом).

**def moves(h):** Это функция, где мы укажем все возможные ходы из условия.

**return h+2, h\*2** Возвращает возможные ходы.

**def game(h):** Принимает количество камней в куче. Тут укажем выигрышную позицию.

**if h >= 25: return 'W'** Возвращаем на экран победу, если  $h \geq 25$  (по условию)

**if any (game(m) == 'W' for m in moves(h)): return 'P1'** Проверяем может ли Петя выиграть за первый ход. Функция "any" ищет *хотя бы один* такой победный ход.

В итоге. Функция game(h) берёт любое значение и подставляет их в moves(h), moves(h) возвращает вместо себя все возможные ходы. Переменная m принимает значения после каждого хода. Проверяем наши условия (if...). Если код выдал 'W', значит у первого игрока есть такой ход, который ведёт к победе.

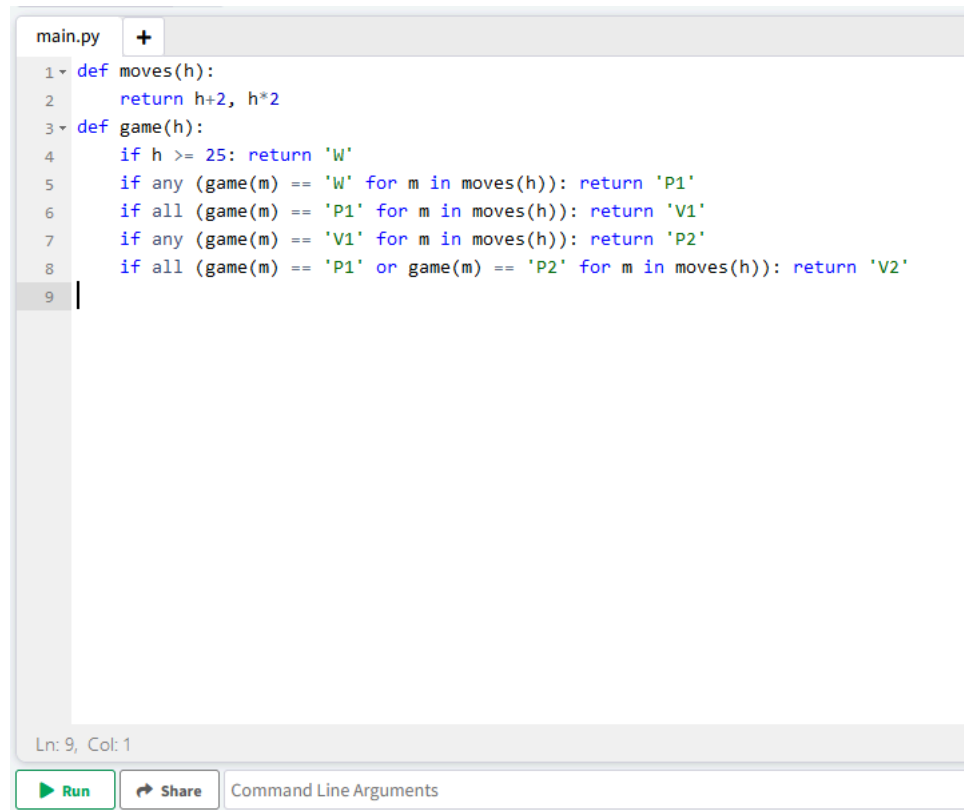
**if all (game(m) == 'P1' for m in moves(h)): return 'V1'** Если из хода P1 все ходы ведут к

выполнению условия для выигрыша, то победа 'V1' (Вани первым ходом). Мы не знаем, каким будет следующий ход Вани, поэтому необходимо написать, что при всех возможных его ходах (all) условие победы будет выполняться.

`if any (game(m) == 'V1' for m in moves(h)): return 'P2'` Если из хода V1 хотя бы один ход P2 ведёт к выполнению условия, то победа 'P2' (Петя вторым ходом).

`if all (game(m) == 'P1' or game(m) == 'P2' for m in moves(h)): return 'V2'` Проверяет, можно ли выиграть Ване после первого или второго хода Пети. Добавляем эту строчку заранее, так как это условие имеется в 21 номере.

В результате получаем:



```
main.py +
1 def moves(h):
2     return h+2, h*2
3 def game(h):
4     if h >= 25: return 'W'
5     if any (game(m) == 'W' for m in moves(h)): return 'P1'
6     if all (game(m) == 'P1' for m in moves(h)): return 'V1'
7     if any (game(m) == 'V1' for m in moves(h)): return 'P2'
8     if all (game(m) == 'P1' or game(m) == 'P2' for m in moves(h)): return 'V2'
9 |

Ln: 9, Col: 1
Run Share Command Line Arguments
```

Теперь осталось ответить на вопросы задачи.

Вопрос 19. Найдите *минимальное значение s*, при котором Ваня выигрывает своим первым ходом при *любой игре* Пети.

В конце нашей программы появятся вот такие строчки:



`for s in range(1,100):` На вход функции подставляем все возможные варианты s, указываем ограничения



`if game(s) == 'V1':` Условие. Если с таким s победит Ваня 1 ходом, то:



`print(s, game(s))` Выводим на экран такое значение s и значение, при котором подбирается s. Получаем:

```
main.py +
1 def moves(h):
2     return h+2, h*2
3 def game(h):
4     if h >= 25: return 'W'
5     if any (game(m) == 'W' for m in moves(h)): return 'P1'
6     if all (game(m) == 'P1' for m in moves(h)): return 'V1'
7     if any (game(m) == 'V1' for m in moves(h)): return 'P2'
8     if all (game(m) == 'P1' or game(m) == 'P2' for m in moves(h)): return 'V2'
9 for s in range(1,100):
10     if game(s) == 'V1':
11         print(s, game(s))
```

Ln: 11, Col: 26

 Run  Share Command Line Arguments

 11 V1  
 12 V1

 \*\* Process exited - Return Code: 0 \*\*  
 Press Enter to exit terminal

Программа вывела все подходящие значения (11 и 12). Мы по условию записываем в ответ минимальное. Итак, наш **ответ: 11**

Вопрос 20. *Сколько существует значений  $S$ , 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.*

В данном случае нам нужно изменить лишь значение, при котором должно выводиться  $s$ . Заменяем 'V1' на 'P2'.

```
main.py +
1 def moves(h):
2     return h+2, h*2
3 def game(h):
4     if h >= 25: return 'W'
5     if any (game(m) == 'W' for m in moves(h)): return 'P1'
6     if all (game(m) == 'P1' for m in moves(h)): return 'V1'
7     if any (game(m) == 'V1' for m in moves(h)): return 'P2'
8     if all (game(m) == 'P1' or game(m) == 'P2' for m in moves(h)): return 'V2'
9 for s in range(1,100):
10     if game(s) == 'P2':
11         print(s, game(s))

Ln: 10, Col: 22
Run Share Command Line Arguments
6 P2
9 P2
10 P2
```

На выходе получаем 3 таких значения (6, 9, 10). В ответ указываем количество.

**Ответ: 3.**

Вопрос 21. Найдите два значения  $s$ , при которых одновременно выполняются два условия: у Вани есть выигрышная стратегия, позволяющая ему *выиграть первым или вторым ходом* при любой игре Пети; у Вани *нет стратегии*, которая позволит ему *гарантированно выиграть первым ходом*. Найденные значения запишите в ответе в порядке возрастания.

Так как Ваня должен выиграть первым или вторым ходом, но необязательно первым, то нам подходят все случаи 'V2'. Снова меняется только одна строчка.

```
main.py +
1 def moves(h):
2     return h+2, h*2
3 def game(h):
4     if h >= 25: return 'W'
5     if any (game(m) == 'W' for m in moves(h)): return 'P1'
6     if all (game(m) == 'P1' for m in moves(h)): return 'V1'
7     if any (game(m) == 'V1' for m in moves(h)): return 'P2'
8     if all (game(m) == 'P1' or game(m) == 'P2' for m in moves(h)): return 'V2'
9 for s in range(1,100):
10     if game(s) == 'V2':
11         print(s, game(s))
```

Ln: 11, Col: 26

Run Share Command Line Arguments

7 V2  
8 V2

**Ответ: 7 8**

Благодаря удобной программе мы можем сразу ответить на 19, 20 и 21 вопрос, практически ничего не меняя в самой программе. Сейчас мы рассмотрели стандартный тип заданий. Это базовый алгоритм решения на 1 кучу камней.

## Номера 19-21 НА ОДНУ КУЧУ С ОГРАНИЧЕНИЕМ СВЕРХУ

№2. Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может *добавить в кучу один камень; увеличить количество камней в куче в два раза; увеличить количество камней в куче в три раза*. Игра завершается в тот момент, когда количество камней в куче становится не менее 36. Если при этом в куче оказалось *не более 60 камней*, то победителем считается игрок, сделавший последний ход. В противном случае победителем становится его противник. В начальный момент в куче было  $S$  камней,  $1 \leq S \leq 35$ .

Это значит, что если количество камней выйдет за верхний предел (60), то победа перейдёт сопернику. Для выигрыша нужно попасть в определённый интервал количества камней в куче.

Теперь в нашей программе появляются новые строчки.

`from functools import lru_cache` Мы используем библиотеку. Данная функция помогает программе работать быстрее, так как программирование рекурсивное.



`@lru_cache(None)` Здесь обозначаем лимит нашей рекурсии None, то есть бесконечность.

Программа будет кешировать столько, столько нужно.

Две предыдущие строчки лучше всего запомнить, дабы избежать того, что рабочая программа просто не выведет ответ, так как истечет время ожидания.

`If 36<=h<=60: return 'W'` Указываем условия победы.  $h$  – количество камней в куче.

`If h > 60: return 'P1'` Не забываем про дополнительную строчку. Она говорит о том, что если количество камней выйдет за предел, то мы автоматически отдаём победу сопернику. Если первый игрок походил таким образом, что камней в куче стало больше 60-ти, то побеждает второй игрок. А данная строчка проверяет камни перед игрой. Грубо говоря, если в куче изначально было камней больше 60, то победит первый игрок – Петя, так как он ещё не делал свой ход.

Вопрос 19. Найдите минимальное значение  $S$ , при котором Ваня выигрывает своим первым ходом при любой игре Пети.

```
main.py +
1 from functools import lru_cache
2 def moves(h):
3     return h+1, h*2, h*3
4 @lru_cache(None)
5 def game(h):
6     if 36<=h<=60: return 'W'
7     if h > 60: return 'P1'
8     if any (game(m) == 'W' for m in moves(h)): return 'P1'
9     if all (game(m) == 'P1' for m in moves(h)): return 'V1'
10    if any (game(m) == 'V1' for m in moves(h)): return 'P2'
11    if all (game(m) == 'P1' or game(m) == 'P2' for m in moves(h)): return 'V2'
12 for s in range(1,100):
13     if game(s) == 'V1':
14         print(s, game(s))
```

Ln: 14, Col: 26

Run Share Command Line Arguments

34 V1

**Ответ: 34**

Вопрос 20. Сколько существует значений  $S$ , 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.

Нам остаётся поменять только одну строчку. Заменяем 'V1' на 'P2', так как Петя должен победить за второй ход.

```
main.py +
1 from functools import lru_cache
2 def moves(h):
3     return h+1, h*2, h*3
4 @lru_cache(None)
5 def game(h):
6     if 36<=h<=60: return 'W'
7     if h > 60: return 'P1'
8     if any (game(m) == 'W' for m in moves(h)): return 'P1'
9     if all (game(m) == 'P1' for m in moves(h)): return 'V1'
10    if any (game(m) == 'V1' for m in moves(h)): return 'P2'
11    if all (game(m) == 'P1' or game(m) == 'P2' for m in moves(h)): return 'V2'
12 for s in range(1,100):
13     if game(s) == 'P2':
14         print(s, game(s))
```

Ln: 13, Col: 22

Run Share Command Line Arguments

33 P2

Программа вывела только одно значение (33).

**Ответ: 1.**

Вопрос 21. Найдите *минимальное и максимальное значения  $S$* , при которых одновременно выполняются два условия: у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Пети; у Вани нет стратегии, которая позволит ему гарантированно выиграть первым ходом. Найденные значения запишите в ответе в порядке возрастания.

```
main.py +
1 from functools import lru_cache
2 def moves(h):
3     return h+1, h*2, h*3
4 @lru_cache(None)
5 def game(h):
6     if 36<=h<=60: return 'W'
7     if h > 60: return 'P1'
8     if any (game(m) == 'W' for m in moves(h)): return 'P1'
9     if all (game(m) == 'P1' for m in moves(h)): return 'V1'
10    if any (game(m) == 'V1' for m in moves(h)): return 'P2'
11    if all (game(m) == 'P1' or game(m) == 'P2' for m in moves(h)): return 'V2'
12 for s in range(1,100):
13     if game(s) == 'V2':
14         print(s, game(s))
```

Ln: 13, Col: 21

Run Share Command Line Arguments

11 V2  
32 V2

Так как Ваня не может гарантировано победить первым ходом, но может вторым, то нам подходит 'V2'.

На выходе получаем всего два значения. Одно из них является минимальным, соответственно, второе – максимальным.

**Ответ: 11 32**

### Номера 19-21 НА ДВЕ КУЧИ (неудачный и удачный ход соперника)

№3. 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 один камень 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 83. Победителем считается игрок, сделавший последний ход, т. е. первым получивший позицию, в которой в кучах будет 83 или больше камней. В начальный момент в первой куче было 9 камней, во второй куче –  $S$  камней,  $1 \leq S \leq 83$ . Будем говорить, что игрок имеет выигрышную стратегию, если он может выиграть при любых ходах противника.

Вопрос 19. Известно, что *Ваня выиграл своим первым ходом после неудачного первого хода Пети*. Назовите минимальное значение  $S$ , при котором это возможно.

Синим цветом помечу универсальную часть программы. Красным цветом отмечу модернизированную (переделанную) часть, а чёрным — пояснения.

```
from functools import lru_cache
```

```
def moves(h):
```

**a, b = h** Так как кучи теперь две, то все возможные ходы (за один раз) можно применить либо к первой куче, либо ко второй. Распаковываем  $h$  на две кучи.

```
return (a+1, b), (a*2, b), (a, b+1), (a, b*2)
```

 Возможных ходов стало в два раза больше, так как теперь у нас есть куча “a” и куча “b”. Прописываем все ходы. Например, в первой скобке мы прибавили один камень к первой куче, а вторую оставили без изменений. И так далее.

```
@lru_cache(None)
```

```
def game(h):
```

**a, b = h** Снова показываем, что ходы распространяются на две кучи

```
if a+b >= 83: return 'W'
```

 Указываем условие из задачи. Для победы сумма камней обеих куч должна быть равна или больше 83.

```
if any (game(m) == 'W' for m in moves(h)): return 'P1'
```

```
if any (game(m) == 'P1' for m in moves(h)): return 'V1'
```

 Ранее вместо **any** мы писали **all**, так как у

нас было условие, что один из игроков должен победить при любых ходах соперника, то есть при всех его возможных ходах – выгодных и не очень выгодных. А в данном номере 19 сказано, что Ваня должен победить первым ходом после **неудачного хода Пети**. Как будто Петя поддался Ване и позволил выиграть, походив в плохую позицию. То есть Ваня победит своим первым ходом, если после хода Пети будет *хоть одна выигрышная позиция*, которая позволит увеличить количество камней до 83 или более.

```
if any (game(m) == 'V1' for m in moves(h)): return 'P2'
```

```
if all (game(m) == 'P1' or game(m) == 'P2' for m in moves(h)): return 'V2'
```

```
for s in range(1, 100):
```

**h = 9, s** По условию в первой куче (например, в “a”) 9 камней, а во второй какое-то неизвестное значение s. Разбиваем h на две кучи, на 9 и s.

```
if game(h) == 'V1':
```

```
print(s, game(h))
```

```
main.py +
1 from functools import lru_cache
2 def moves(h):
3     a, b = h
4     return (a+1, b), (a*2, b), (a, b+1), (a, b*2)
5 @lru_cache(None)
6 def game(h):
7     a, b = h
8     if a+b >= 83: return 'W'
9     if any (game(m) == 'W' for m in moves(h)): return 'P1'
10    if any (game(m) == 'P1' for m in moves(h)): return 'V1'
11    if any (game(m) == 'V1' for m in moves(h)): return 'P2'
12    if all (game(m) == 'P1' or game(m) == 'P2' for m in moves(h)): return 'V2'
13 for s in range(1,100):
14     h = 9, s
15     if game(h) == 'V1':
16         print(s, game(h))
```

Ln: 16, Col: 26

Run Share Command Line Arguments

```
19 V1
20 V1
21 V1
22 V1
23 V1
24 V1
25 V1
26 V1
```

На экран выводится множество чисел. По условию нам нужно минимальное.

**Ответ: 19**

Вопрос 20. Найдите два таких значения S, при которых 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. Найденные значения запишите в ответе в порядке возрастания.

Возвращаем в ту же строчку all вместо any, так как в условии сказано, что Петя может выиграть при всех возможных ходах.

```
main.py +
1 from functools import lru_cache
2 def moves(h):
3     a, b = h
4     return (a+1, b), (a*2, b), (a, b+1), (a, b*2)
5 @lru_cache(None)
6 def game(h):
7     a, b = h
8     if a+b >= 83: return 'W'
9     if any (game(m) == 'W' for m in moves(h)): return 'P1'
10    if all (game(m) == 'P1' for m in moves(h)): return 'V1'
11    if any (game(m) == 'V1' for m in moves(h)): return 'P2'
12    if all (game(m) == 'P1' or game(m) == 'P2' for m in moves(h)): return 'V2'
13 for s in range(1,100):
14     h = 9, s
15     if game(h) == 'P2':
16         print(s, game(h))
```

Ln: 10, Col: 11

Run Share Command Line Arguments

32 P2  
36 P2

**Ответ: 32 36**

Вопрос 21. Укажите минимальное значение S, при котором 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.

Меняем только одну строчку, победу отдаём Ване за второй ход.

Спасибо за отведённое время;)

Доброго времени суток

This site was made on Tilda — a website builder that helps to create a website without any code

[Create a website](#)

[How to remove this block?](#)

[About platform](#)

[Submit a complaint](#)