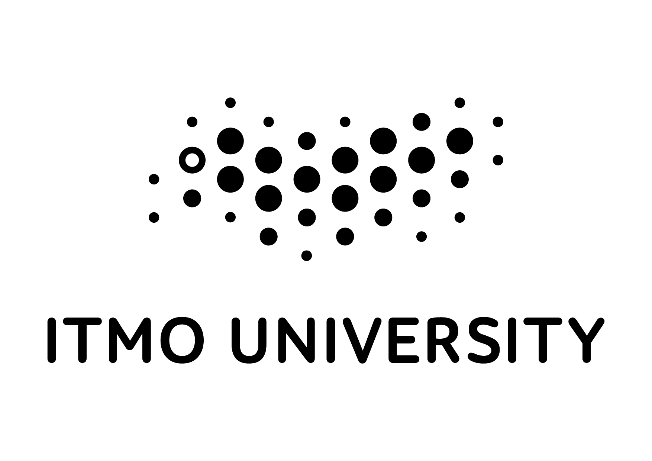
Санкт-Петербургский национальный исследовательский университет ИТМО

Факультет систем управления и робототехники



Алгоритмы и структуры данных.

Отчет №1.

Выполнил студент гр. R32362

Лаптев Максим Сергеевич

Санкт-Петербург 2023

**Содержание**

1. Цель
2. Задача №2025 – Стенка на стенку
3. Задача №1005 – Куча камней
4. Задача №1296 – Гиперпереход
5. Вывод

**Цель**

Решить данные задачи, написав код, работа которого удовлетворяет условиям

**№2025 «Стенка на стенку»**

Изображение выглядит как текст

Автоматически созданное описание

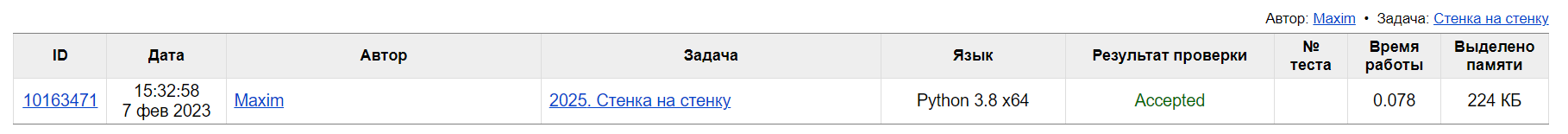
Исходный код:

t = int(input()) # ввод количества тестов  
for \_ in range(t):  
 n, k = map(int, input().split()) # ввод данных  
 fights = k \* (k-1) // 2  
 sostav = n // k  
 ostatok = n % k  
 result = sostav \* sostav \* fights # количество боев не учитывая остаток  
 result += ostatok \* sostav \* (k-1) + ostatok \* (ostatok – 1) // 2  
 print(result) # вывод ответа

Определив базовый алгоритм разбиения на команды и получения максимального количества боев на листочке, была выведена формула.

Язык программирования: python

Результат



**№1005** **«Куча камней»**

Изображение выглядит как текст

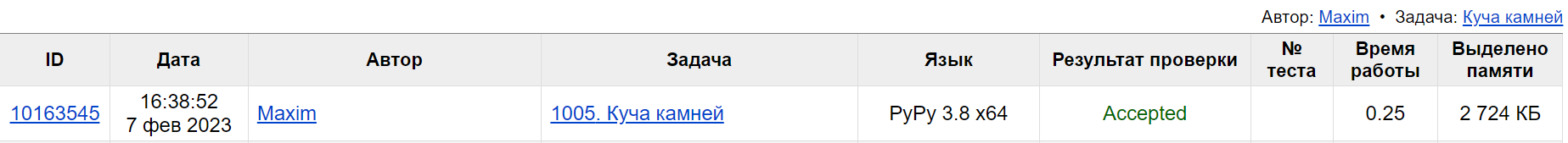
Автоматически созданное описание

Исходный код:

from itertools import product  
  
# ввод данных  
n = int(input())  
w = [int(i) for i in input().split()]  
# оптимизация и подготовка алгоритма  
s1 = max(w)  
sall = sum(w)  
w.remove(s1)  
result = 100000  
# перебор всевозможных сумм  
for el in product(range(2), repeat=n-1):  
 s = s1  
 for i in range(n-1):  
 s += el[i] \* w[i]  
 result = min(result, abs(sall - s - s)) # получение минимальной разности между кучами  
print(result) # вывод ответа

Так как ограничения на количество камней не такое уж и большое (<=20) решено было использовать алгоритм перебора всевозможных сумм, основанный на операции перемножения на побитовые маски

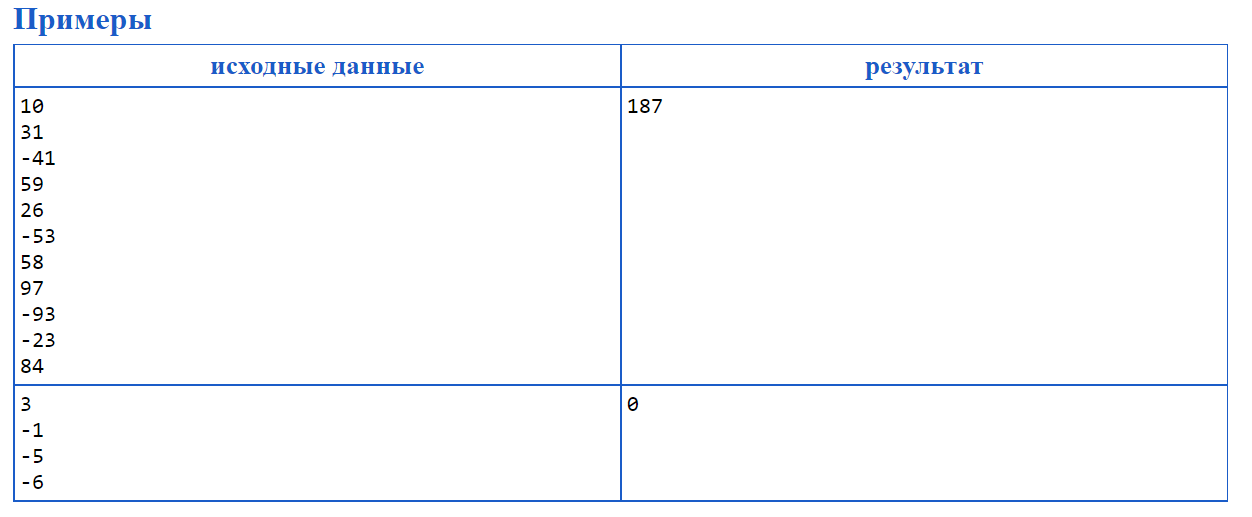
Язык программирования: python

Результат

**№1296 «Гиперпереход»**

Изображение выглядит как текст

Автоматически созданное описание



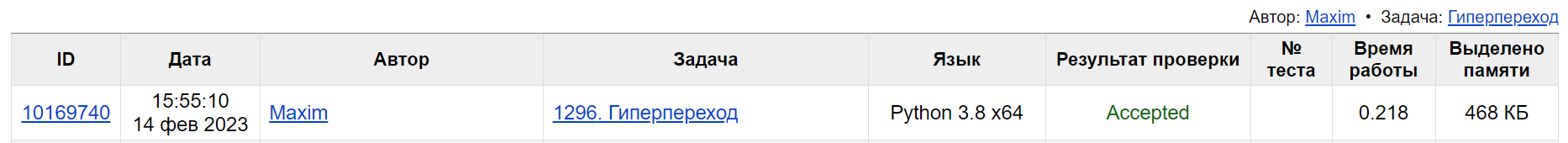
Исходный код:

n = int(input()) # ввод длины последовательности  
s1, s2 = 0, 0  
res = 0  
for i in range(n):  
 q = int(input())  
 s1 += q  
 if i != 0:  
 s2 += q # s1 хранит первый элемент последовательности, s2 - второй  
 res = max(res, max(s1, s2)) # определение максимальной суммы, сравнение  
 if s1 < 0:  
 s1 = 0 # обнуление в случае отрицательной суммы последовательности  
 if s2 < 0:  
 s2 = 0  
print(res) # вывод ответа

По сути, необходимо было найти максимальную сумму подпоследовательности (и 0). Для решения хватает двух переменных (s1 и s2), которые хранят в себе максимальную сумму и изменяющиеся по ходу ввода чисел. К тому же в конце каждой итерации происходит проверка – если в результате сумма последовательности оказывается меньше 0, то необходимо ее обнулить, чтобы начать накапливать потенциал заново.

Язык программирования: python

Результат:



**Вывод**

В результате проделанной работы были успешно решены 3 задачи, используя язык программирования python. Работа программы удовлетворяет условиям задачи (проходит по времени и памяти).