Полулях А.Р. ВКБ-32

Условие.

Задача №1620. Субботник

В классе учатся N человек. Классный руководитель получил указание направить на субботник R бригад по С человек в каждой.

Все бригады на субботнике будут заниматься переноской бревен. Каждое бревно одновременно несут все члены одной бригады. При этом бревно нести тем удобнее, чем менее различается рост членов этой бригады.

Числом неудобства бригады будем называть разность между ростом самого высокого и ростом самого низкого членов этой бригады (если в бригаде только один человек, то эта разница равна 0). Классный руководитель решил сформировать бригады так, чтобы максимальное из чисел неудобства сформированных бригад было минимально. Помогите ему в этом!

Рассмотрим следующий пример. Пусть в классе 8 человек, рост которых в сантиметрах равен 170, 205, 225, 190, 260, 130, 225, 160, и необходимо сформировать две бригады по три человека в каждой. Тогда одним из вариантов является такой:

1 бригада: люди с ростом 225, 205, 225

2 бригада: люди с ростом 160, 190, 170

При этом число неудобства первой бригады будет равно 20, а число неудобства второй — 30. Максимальное из чисел неудобств будет 30, и это будет наилучший возможный результат.

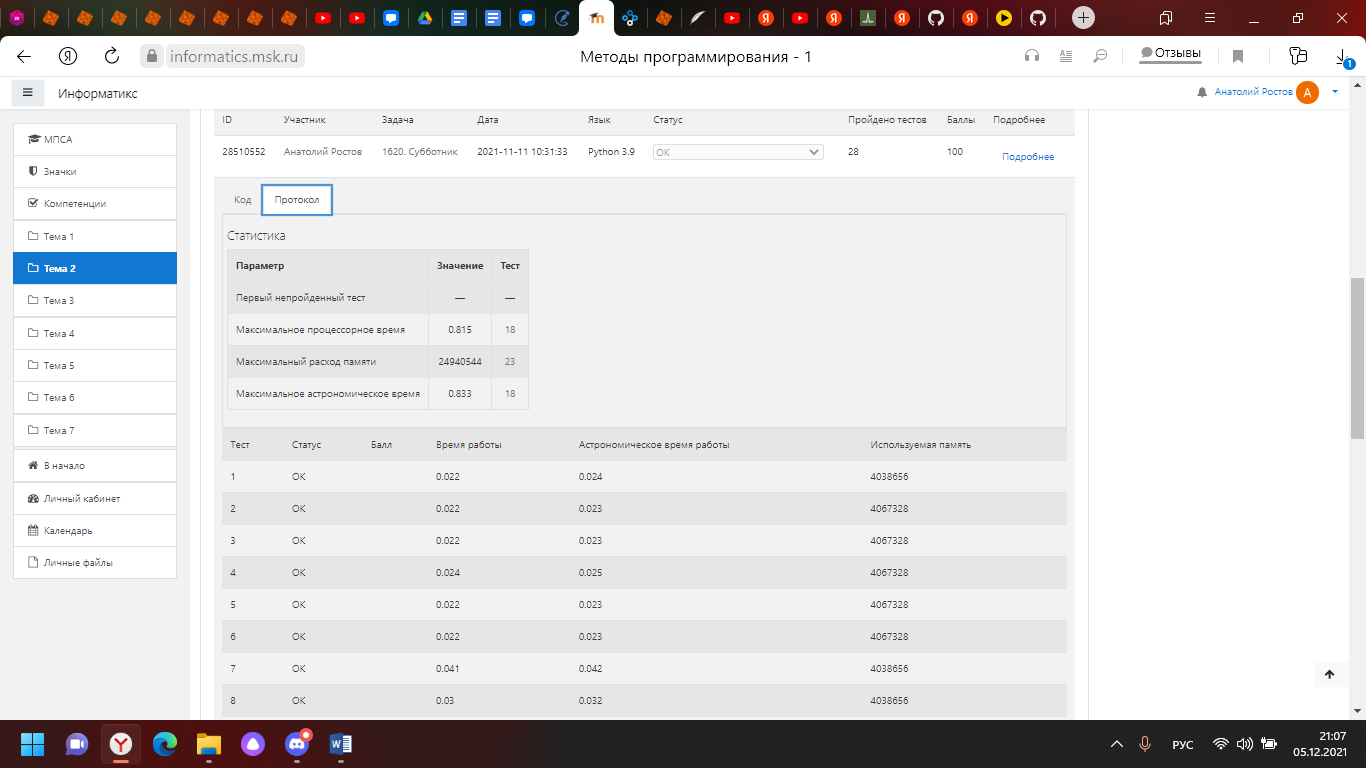
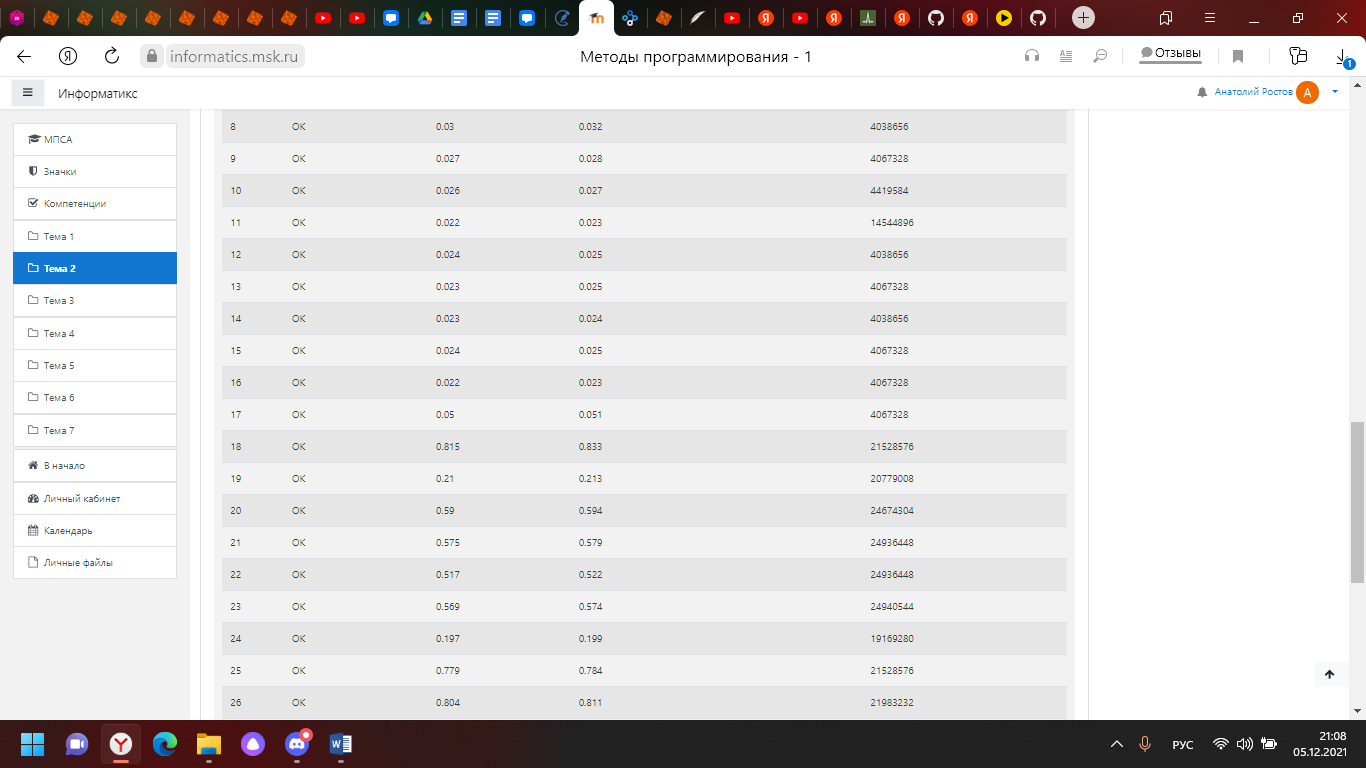
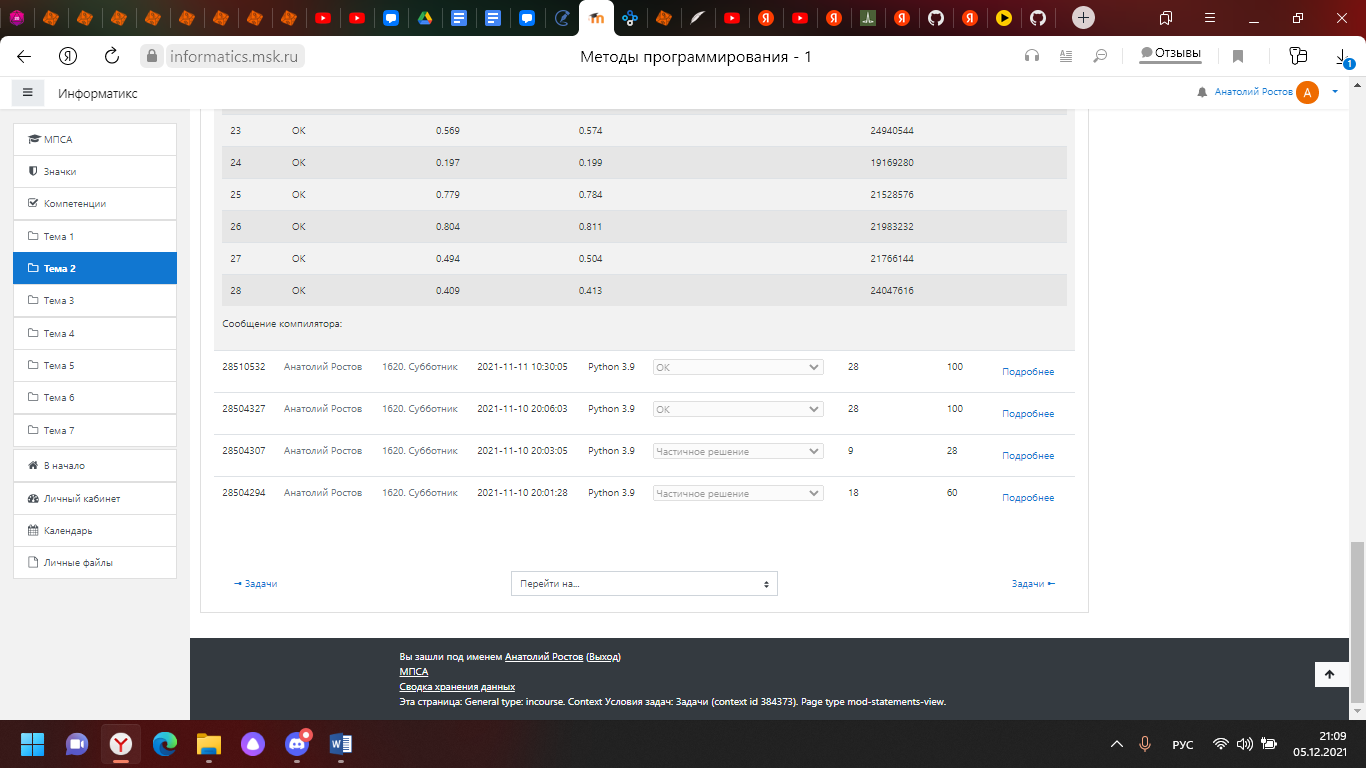
Формат входных данных

Сначала вводятся натуральные числа N, R и C — количество человек в классе, количество бригад и количество человек в каждой бригаде (1 ≤ R∙C ≤ N ≤ 100 000). Далее вводятся N целых чисел — рост каждого из N учеников. Рост ученика — натуральное число, не превышающее 1 000 000 000.

Формат выходных данных

Выведите одно число — наименьше возможное значение максимального числа неудобства сформированных бригад.

Результат теста работы №2.

Код программы:

def bocx(ty):

mt1 = 0

mt2 = 0

for i in range(len(tyu)):

mt2 -= 1

if mt2 < 1 and tyu[i] <= ty:

mt1 += 1

mt2 = soc

if mt1 >= noc:

return True

return False

lng, noc, soc = map(int, input().split())

hgs = sorted([int(input()) for line in range(lng)])

tyu = [hgs[i + soc - 1] - hgs[i] for i in range(len(hgs) - soc + 1)]

pin1 = -1

pin2 = hgs[-1] - hgs[0]

if noc > 1:

while pin1 + 1 != pin2:

mid = (pin1 + pin2) // 2

if bocx(mid):

pin2 = mid

else:

pin1 = mid

print(pin2)

else:

print(min(tyu))