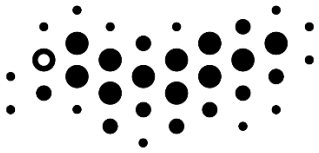
Санкт-Петербургский национальный
исследовательский университет ИТМО

Факультет систем управления и робототехники



ITMO UNIVERSITY

Алгоритмы и структуры данных.
Отчет №1.

Выполнил студент гр. R32362

Лаптев Максим Сергеевич

Санкт-Петербург 2023

Содержание

Цель

Решить данные задачи, написав код, работа которого удовлетворяет условиям

№2025 «Стенка на стенку»

Ограничение времени: 1.0 секунды
Ограничение памяти: 64 МБ

Бокс, каратэ, самбо… Классические боевые единоборства пресытили аудиторию. Поэтому известный спортивный канал запускает новый формат соревнований, основанный на традиционной русской забаве — боях стенка на стенку. В соревновании могут участвовать от двух до k команд, каждая из которых будет соперничать с остальными. Всего в соревновании примут участие n бойцов. Перед началом боя они должны разделиться на команды, каждый боец должен войти ровно в одну команду. За время боя два бойца сразятся, если они состоят в разных командах. Организаторы считают, что популярность соревнований будет тем выше, чем больше будет количество схваток между бойцами. Помогите распределить бойцов по командам так, чтобы максимизировать количество схваток между бойцами, и выведите это количество.

Исходные данные

В первой строке дано количество тестов T ($1 \le T \le 10$). В следующих T строках перечислены тесты. В каждой из них записаны целые числа n и k через пробел ($2 \le k \le n \le 10^4$).

Результат

Для каждого теста в отдельной строке выведите одно целое число — ответ на задачу.

Пример

исходные данные	результат
3 6 3 5 5 4 2	12 10 4

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

Автор: Maxim • Задача: Стенка на стенку

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10163471	15:32:58 7 фев 2023	Maxim	2025. Стенка на стенку	Python 3.8 x64	Accepted		0.078	224 КБ

№1005 «Куча камней»

Ограничение времени: 1.0 секунды
Ограничение памяти: 64 МБ

У вас есть несколько камней известного веса w_1, …, w_n. Напишите программу, которая распределит камни в две кучи так, что разность весов этих двух куч будет минимальной.

Исходные данные

Ввод содержит количество камней n ($1 \le n \le 20$) и веса камней w_1, …, w_n ($1 \le w_i \le 100\ 000$) — целые, разделённые пробельными символами.

Результат

Ваша программа должна вывести одно число — минимальную разность весов двух куч.

Пример

исходные данные	результат
5 5 8 13 27 14	3

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

Автор: Maxim • Задача: Куча камней

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10163545	16:38:52 7 фев 2023	Maxim	1005. Куча камней	PyPy 3.8 x64	Accepted		0.25	2 724 КБ

№1296 «Гиперпереход»

Ограничение времени: 1.0 секунды
Ограничение памяти: 64 МБ

Гиперпереход, открытый ещё в начале XXI-го века, и сейчас остаётся основным способом перемещения на расстояния до сотен тысяч парсеков. Но совсем недавно физиками открыто новое явление. Оказывается, длительностью альфа-фазы перехода можно легко управлять. Корабль, находящийся в альфа-фазе перехода, накапливает гравитационный потенциал. Чем больше накопленный гравитационный потенциал корабля, тем меньше энергии потребуется ему на прыжок сквозь пространство. Ваша цель — написать программу, которая позволит кораблю за счёт выбора времени начала альфа-фазы и её длительности накопить максимальный гравитационный потенциал.

В самой грубой модели грави-интенсивность — это последовательность целых чисел p_i. Будем считать, что если альфа-фаза началась в момент i и закончилась в момент j, то накопленный в течение альфа-фазы потенциал — это сумма всех чисел, стоящих в последовательности на местах от i до j.

Исходные данные

В первой строке входа записано целое число N — длина последовательности, отвечающей за грави-интенсивность ($0 \le N \le 60000$). Далее идут N строк, в каждой записано целое число p_i ($-30000 \le p_i \le 30000$).

Результат

Максимальный гравитационный потенциал, который может накопить корабль в альфа-фазе прыжка. Считается, что потенциал корабля в начальный момент времени равен нулю.

Примеры

исходные данные	результат
10 31 -41 59 26 -53 58 97 -93 -23 84	187
3 -1 -5 -6	0

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:

Автор: Maxim • Задача: Гиперпереход

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10169740	15:55:10 14 фев 2023	Maxim	1296. Гиперпереход	Python 3.8 x64	Accepted		0.218	468 КБ

Вывод

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