Расселение туристов

В городе Флэшевск есть двусторонняя кольцевая автодорога. Администрация города решила построить на ней отели для заселения в них туристов. Так как туристов очень много, то велика вероятность, что вы окажетесь в Флэшевске в одно время со своими друзьями, а может даже врагами. Администрация решила помочь своим гостям с расселением, а придумать, как искать два наиболее удаленных друг от друга отеля, чтобы заселить в них двух врагов, не смогла. Она просить помочь вас сделать это.

Входные данные

На первой строке дана длина автодороги L (1 <= L <= 10^9, L % 2 = 0) и количество отелей на ней N (1 <= N <= 10^6). Далее следуют N чисел в порядке возрастания — расстояния до каждого отеля от 0-го километра автодороги. Каждое расстояние представлено на отдельной строке. Расстояния лежат в промежутке от 0 до L - 1. Гарантируется, что на одном километре не могут находиться два отеля.

Выходные данные

Выведите одно число — расстояние между наиболее удаленными друг от друга отелями.

Пример

```
24 6
2
7
11
16
20
21
```

Здесь ответом будет расстояние между отелями, расположенными на 7 километре и 20 километре. Расстояние между ними — 11, это максимальное среди всех.

Идея решения

Перефразируем задачу. Дана окружность, на ней отмечены точки. Необходимо найти две наиболее удаленные друг от друга точки, то есть точки с наибольшим расстоянием друг от друга. Так как дорога двусторонняя, то расстояние – минимальное из расстояний по часовой стрелке и против часовой стрелки. Ответ на файл A получить легко – переберем каждую пару отелей, посчитаем расстояния между ними. Выберем максимальное из таких расстояний – это и будет ответом.

```cpp
#include <iostream>
#include <vector>
#include <fstream>

int main() {

    //считываем входные данные
    std::fstream file("tourist_test.txt");
    int L, N;
    file >> L >> N;
    std::vector<int> road(N);
    for (int i = 0; i < N; ++i)
        file >> road[i];

    int max = -1; //здесь будет лежать ответ – максимальное из расстояний
    //рассматриваем каждую пару отелей
    for (int i = 0; i < N; ++i) {
        for (int j = i + 1; j < N; ++j) {
            int right = road[j] - road[i]; //считаем расстояние по часовой стрелке
            int left = L - road[j] + road[i]; //считаем расстояние против часовой стрелки
            int dist = std::min(right, left); //выбираем нужное нам расстояние – минимальное
            max = std::max(max, dist); //обновляем ответ
        }
    }

    //выводим ответ
    std::cout << max;
    return 0;
}
```

Решение файла A

Файл B не получится решить таким переборным алгоритмом. Сначала заметим, что максимальное расстояние между двумя любыми точками на окружности – L / 2, то есть дуга, которая стянута диаметром окружности. Значит отель, куда можно поселить врага, должен находиться возле диаметрально противоположной точки или на ней. Пусть мы нашли ответ для фиксированной точки i (работаем в терминах индексов в массиве). Ответ для это точки – наиболее удаленная от нее точка. Назовем этот ответ точкой j. Очевидно, что для следующей по порядку точки (i

```cpp
#include <iostream>
#include <vector>
#include <fstream>

int main() {

    //считываем входные данные
    std::fstream file("tourist_B.txt");
    int L, N;
    file >> L >> N;
    std::vector<int> road(N);
    for (int i = 0; i < N; ++i)
        file >> road[i];

    int j = 0; //указывает на отель для врага
    int max = -1; //здесь будет находиться ответ
    for (int i = 0; i < N; ++i) {
        while (j < N && road[j] - road[i] < L / 2) ++j; //перебираем пока не перейдем через диаметр или не дойдем до конца
        if (j == N) break; //если мы не нашли подходящего отеля для врага до 0 километра, то выходим из цикла
        int left = 0, right = 0;
        left = L - road[j] + road[i]; //считаем ответ для точки справа от диаметра
        right = road[j - 1] - road[i]; //считаем ответ для точки слева от диаметра
        max = std::max(max, std::max(left, right)); //обновляем ответ
    }

    //выводим ответ
    std::cout << max;
    return 0;
}

```

Решение файла B

+ 1) ответ будет точка k >= j. Для решении задачи воспользуемся методом двух указателей.
Почему мы останавливаем перебор, когда j == N? Потому что если ответ не нашелся до 0 километра, то значит ответ будет дальше него – 0+ километр. Но для первой половины мы уже посчитали ответ, и поэтому считать ответ для диаметрально противоположных точек не имеет смысла.

Генерация тестов

```python
from os import system

from random import randrange
from random import choice
n = randrange(100, 500)
l = randrange(1000, 5000)
if l % 2 != 0: l += 1
a = []
for j in range(n):
    a.append(randrange(0, l - 1))
a = list(set(a))
a.sort()
n = len(a)
with open("tourist_A.txt", "w") as fout:
    print(l, end = ' ', file=fout)
    print(n, file=fout)
    for x in a:
        print(x, file=fout)
```

Генерация файла A

```python
from os import system

from random import randrange
from random import choice
n = 1000000
l = randrange(100000000, 1000000000)
if l % 2 != 0: l += 1
a = []
for j in range(n):
    a.append(randrange(0, l - 1))
a = list(set(a))
a.sort()
n = len(a)
with open("tourist_B.txt", "w") as fout:
    print(l, end = ' ', file=fout)
    print(n, file=fout)
    for x in a:
        print(x, file=fout)
```

Генерация файла B

Файл A и файл B сгенерированы на Python, соблюдая все условия входных данных задачи.

Проверка корректности решения задачи для файла B была написана программа, генерирующая стресс-тесты. Медленное и быстрое решение запускаются на одних и тех же тестах несколько раз, а ответы сравниваются.

```
from os import system

from random import randrange
from random import choice

for i in range(1, 100):
    n = randrange(3, 100)
    l = randrange(20, 100)
    if l < n: continue
    if l % 2 != 0: l += 1
    a = []
    for j in range(n):
        a.append(randrange(0, l - 1))
    a = list(set(a))
    a.sort()
    n = len(a)
    with open("in.txt", "w") as fout:
        print(l, file=fout)
        print(n, file=fout)
        for x in a:
            print(x, file=fout)
    zapusk1 = "./tourist_slow_ < in.txt > out1.txt"
    zapusk2 = "./tourist_fast_ < in.txt > out2.txt"
    system(zapusk1)
    system(zapusk2)
    ans1 = open("out1.txt", "r").readline().strip()
    ans2 = open("out2.txt", "r").readline().strip()
    if (ans1 != ans2):
        print(ans1, ans2)
        exit(0)
```

Генерация стресс-тестов

Подарки

Ася хорошо закончила учебный год, поэтому решила себе купить подарки. Асе без разницы, какие подарки у нее будут в наборе. Самое главное, чтобы их суммарная ценность была максимальной. Когда Ася пришла в магазин, то продавец ей сказал, что она может брать с полки только те подарки, которые стоят последовательно друг за другом. Но так как девочка отличница, то она хочет усложнить себе задачу. Ей хочется, чтобы в ее наборе подарков с ценностью, которая кратна 5, было четное количество. Помогите Асе быстрее справиться с ее задачей, чтобы не задерживать людей в очереди.

Входные данные

На первой строке дано число N (1 <= N <= 10^6) - количество подарков на полке. Далее на N строках перечислена ценность подарков. Ценность подарков - целые числа от -10^9 до 10^9.

Выходные данные

Выведите одно число - суммарную ценность подарков, которые купит Ася.

Пример

```
16
15
6
7
-5
-10
18
2
20
-24
48
-5
-25
13
```

15
21
-105

Здесь ответом является подотрезок чисел с 6 по 15 элемент. Сумма на нем равна 83. Эта сумма максимальна, а количество элементов, кратных 5, равно 4, что четно.

Идея решения

Перефразируем задачу. Необходимо найти подотрезок с максимальной суммой, где четное количество чисел, кратных 5. Переборное решение предельно простое – переберем все отрезки (левая и правая граница), посчитаем сумму на нем и количество элементов, кратных 5. Если количество таких элементов четно, то обновляем ответ. Такое решение позволяет дать ответ на файл A.

```cpp
#include <iostream>
#include <vector>
#include <fstream>

int main() {

    //считываем входные данные
    std::fstream file("gifts_test.txt");
    int N;
    file >> N;
    std::vector<long long> cost(N);
    for (int i = 0; i < N; ++i)
        file >> cost[i];

    long long max_sum = -1e18; //здесь будет ответ - максимальная сумма
    for (int i = 0; i < N; ++i) { //перебираем левую границу отрезка
        for (int j = i; j < N; ++j) { //перебираем правую границу отрезка
            long long cur_sum = 0; //здесь будет сумма на отрезке
            int cnt_5 = 0; //здесь будет количество чисел, кратных 5, на отрезке
            for (int k = i; k <= j; ++k) {
                cur_sum += cost[k]; //считаем сумму на отрезке
                if (cost[k] % 5 == 0)
                    ++cnt_5; //считаем количество чисел, кратных 5, на отрезке
            }
            if (cnt_5 % 2 == 0) //если условие на четное количество выполняется, то обновляем ответ
                max_sum = std::max(max_sum, cur_sum);
        }
    }

    //выводим ответ
    std::cout << max_sum;
    return 0;
}
```

Решение файла A

Файл B не получится решить переборным алгоритмом. Для того, чтобы быстро получать сумму на отрезке, нам необходимо применить идею префиксных сумм. Заведем две переменные – ost0 и ost1, где будем хранить минимальное

значение префиксной суммы на префиксе, где количество элементов, кратных 5, дает остаток 0 или 1 соответственно. ost0 инициализируем значением 0, потому что изначально добавляем информацию о префиксе длины 0, ost1 инициализируем таким значением, которое означает, что у нас нет на текущий момент префикса с нечетным количеством кратных 5. Также нужно хранить информацию о количестве чисел, кратных 5, на текущем префиксе. Перебираем правую границу отрезка – она фиксированная. Для того, чтобы максимизировать сумму, то есть сделать значение pref[r] – pref[l] максимальным, нам нужно минимизировать pref[l], то есть префиксную сумму в левой границе отрезка. Чтобы поддерживать условие четности количества элементов, кратных 5, нам необходимо вычитать ту префиксную сумму, где количество элементов, кратных 5, дает при делении на 2 тот же остаток, что и количество элементов на текущем префиксе, рассматриваемым нами.

```cpp
#include <iostream>
#include <vector>
#include <fstream>

int main() {

    //считываем входные данные
    std::fstream file("gifts_test.txt");
    int N;
    file >> N;
    std::vector<long long> cost(N);
    for (int i = 0; i < N; ++i)
        file >> cost[i];

    long long max_sum = -1e18; // здесь будет лежать ответ – максимальная сумма
    std::vector<long long> pref(N + 1); //создаем массив префиксных сумм
    //считаем массив префиксных сумм
    pref[0] = 0;
    for (int i = 1; i <= N; ++i)
        pref[i] = pref[i - 1] + cost[i - 1];

    long long ost0 = 0, ost1 = -1e18; //значения минимальной префиксной суммы на префиксе, где количество кратных 5 дает остаток 0 или 1
    int cur_5 = 0; //количество чисел, кратных 5, на текущем префиксе
    for (int i = 0; i < N; ++i) {
        if (cost[i] % 5 == 0) //обновляем количество чисел, кратных 5, на префиксе
            cur_5 += 1;
        if (cur_5 % 2 == 1) { //если на префиксе нечетное количество кратных 5
            if (ost1 == -1e18) //если нам встретилось первое число, кратное 5
                ost1 = pref[i + 1]; //записываем префиксную сумму в соответствующую переменную
            else {
                max_sum = std::max(max_sum, pref[i + 1] - ost1); //обновляем максимальную сумму, считая сумму на отрезке
                ost1 = std::min(ost1, pref[i + 1]); //обновляем значение минимальной префиксной суммы на префиксе
            }
        } else {
            max_sum = std::max(max_sum, pref[i + 1] - ost0); //обновляем максимальную сумму, считая сумму на отрезке
            ost0 = std::min(ost0, pref[i + 1]); //обновляем значение минимальной префиксной суммы на префиксе
        }
    }

    //выводим ответ
    std::cout << max_sum;
    return 0;
}
```

Решение файла B

Генерация тестов

```python
from os import system

from random import randrange
from random import choice

n = randrange(20, 100)
a = []
for j in range(n):
    a.append(randrange(-10000, 10000))
with open("gifts_A.txt", "w") as fout:
    print(n, file=fout)
    for x in a:
        print(x, file=fout)
```

Генерация файла A

```python
from os import system

from random import randrange
from random import choice

n = 1000000
a = []
for j in range(n):
    a.append(randrange(-1000000000, 1000000000))
with open("gifts_B.txt", "w") as fout:
    print(n, file=fout)
    for x in a:
        print(x, file=fout)
```

Генерация файла B

```python
from os import system

from random import randrange
from random import choice

for i in range(1, 100):
    n = randrange(3, 100)
    a = []
    for j in range(n):
        a.append(randrange(-1000000000, 1000000000))
    with open("in.txt", "w") as fout:
        print(n, file=fout)
        for x in a:
            print(x, file=fout)
    zapusk1 = "./gifts_slow_ < in.txt > out1.txt"
    zapusk2 = "./gifts_fast_ < in.txt > out2.txt"
    system(zapusk1)
    system(zapusk2)
    ans1 = open("out1.txt", "r").readline().strip()
    ans2 = open("out2.txt", "r").readline().strip()
    if (ans1 != ans2):
        print(ans1, ans2)
        exit(0)
```

Генерация стресс-тестов