ВСЕРОССИЙСКАЯ (С МЕЖДУНАРОДНЫМ УЧАСТИЕМ) СТУДЕНЧЕСКАЯ ОЛИМПИАДА «СПЕКТР» (ИНФОРМАТИКА)

РЕШЕНИЯ ФИНАЛЬНОГО ЭТАПА

1. Дальние путешествия

Вася прошёл очередной этап отбора в межгалактическом турнире. В следующем задании ему надо путешествовать по планетам. Новейший космолёт перемещается в парамерном пространстве как обычный шахматный конь на доске. Парамерное пространство можно отобразить в обычное двумерное пространство, но координаты могут оказаться очень большими.

Шахматный конь ходит на 2 клетки по вертикали или горизонтали, а затем поворачивается на 90 градусов и перемещается ещё на 1 клетку.

Васе требуется определить можно ли попасть из одной клетки доски в другую перемещаясь ходом коня.

Входные данные

Первая строка содержит одно целое число k ($1 \le k \le 10^4$) — количество наборов входных данных.

Первая строка каждого набора входных данных содержит два целых числа n и m ($2 \le n, m \le 10^6$) — размеры поля.

Вторая строка каждого набора входных данных содержит два целых числа x и y ($1 \le x \le n$, $1 \le y \le m$) — координаты начальной клетки.

Третья строка каждого набора входных данных содержит два целых числа v и w ($1 \le v \le n$, $1 \le w \le m$) — координаты конечной клетки.

Выходные данные

Для каждого набора входных данных выведите в одной строке «Yes», если можно достичь конечной клетки доски, и «No» в противном случае. Выводить ответ можно в любом регистре, т.е. ответы «Yes», «yEs» и «YES» будут восприниматься как «Yes».

Ограничения

ограничение по времени на тест	1 секунда
ограничение по памяти на тест	256 мегабайт

Примеры

Входные данные	Выходные данные
3 3 3 1 1 2 3 3 3 1 1 2 2 8 8 2 3 3 3	YES NO YES

РЕШЕНИЕ

В данной задаче необходимо рассмотреть частные случаи. Если использовать типовой подход с перебором всех доступных путей, то время работы для замкнутого маршрута будет $O(N^2)$, что не подходит под ограничения.

Рассмотрим доску размером 3x3. Конь может посетить все клетки, кроме центральной (рисунок 1).

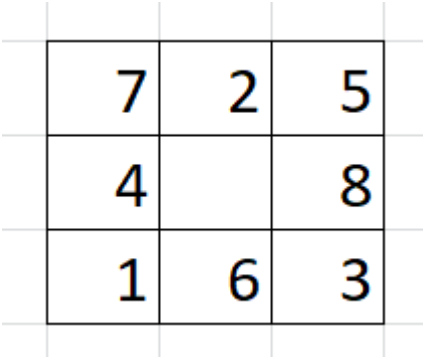


Рисунок 1. Ходы коня на доске 3x3

Если доска имеет больший размер, то конь всегда сможет достичь конечной клетки.

Рассмотрим второй частный случай, когда хотя бы один из размеров равен 2. В этом случае конь сможет посетить только клетки, представленные на рисунке 2. Если имеем 2 горизонтали, то достижимыми будут клетки, расположенные на расстоянии кратном 4 от исходной в той же строки, и клетки, расположенные в другой строке на расстоянии, которое при делении на 4 даёт в остатке 2. Для вертикального расположения будут аналогичные рассуждения.

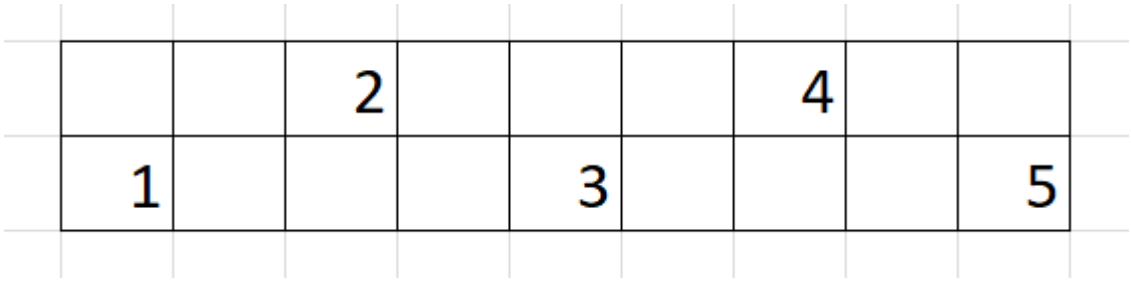


Рисунок 2. Ходы коня на доске 2xN

Один из вариантов решения на языке Python 3.* представлен ниже.

```
k = int(input())
for _ in range(k):
    n, m = map(int, input().split())
    x, y = map(int, input().split())
    v, w = map(int, input().split())

    if (x, y) == (v, w):
        print('Yes')
        # continue

    elif min(n, m) == 2:
        dx, dy = abs(x - v), abs(y - w)
        if m == 2:  # vert
            dx, dy = dy, dx
        if ( (dx == 0 and dy % 4 == 0) or
```

```
            (dx == 1 and dy % 4 == 2) ):
            print('Yes')
        else:
            print('No')
        # continue

    elif (x, y) == (3, 3):
        if (x, y) == (2, 2) or (v, w) == (2, 2):
            print('No')
        else:
            print('Yes')

    else:
        print('Yes')
```

2. Сложная конструкция

Разбирая коллекцию механизмов у дедушки в гараже, Вася нашёл комплект из n шестерёнок. Вася вспомнил, что соединяя шестерни, можно увеличить или уменьшит скорость вращения последней оси. Для каждой пары шестерён с зубьями a и b. Если скорость первой шестерни с a зубами будет равна v, то скорость вращения второй шестерни с b зубами будет равна $w = \frac{a*v}{b}$. Васе необходимо отобрать все или часть шестерёнок (не менее половины) так, чтобы расположив их в ряд можно было бы получить у последней шестерни такую же скорость вращения, что и у первой.

Входные данные

Первая строка содержит одно целое число k ($1 \le k \le 10^4$) — количество наборов входных данных.

Первая строка каждого набора входных данных содержит одно целое число n ($2 \le n \le 200$) — количество найденных шестерён.

Вторая строка каждого набора входных данных содержит n целых чисел, $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 200$) — количество зубьев каждой шестерни.

Выходные данные

Для каждого набора входных данных выведите в одной строке «Yes», если можно выбрать такой набор шестерён из найденных, и «No» в противном случае. Выводить ответ можно в любом регистре, т.е. ответы «Yes», «yes», «yEs» и «YES» будут восприниматься как «Yes».

Ограничения

ограничение по времени на тест	1 секунда
ограничение по памяти на тест	256 мегабайт

Примеры

Входные данные	Выходные данные
5 2 5 5 4 6 3 6 9 2	YES YES NO

2 3 7 30 10 12 10 10 9 18 5 2 4 8 16 32	YES NO

Примечание

В первом примере один из возможных исходов игры может быть следующим:

- На первом ходе Ай выбирает поменять местами a1 и b1. Теперь массивы: a=[3,4,6,1] и b=[1,2,3,7].
- На втором ходе Май выбирает поменять местами a2 и b2. Теперь массивы: a=[3,2,6,1] и b=[1,4,3,7].
- На третьем ходе Ай выбирает пропустить ход.
- На четвертом ходе Май выбирает поменять местами a4 и b4. Теперь массивы: a=[3,2,6,7] и b=[1,4,3,1].

Теперь финальный счет Ай равен 3⊕2⊕6⊕7=0, а финальный счет Май равен 1⊕4⊕3⊕1=7. Поэтому Май выигрывает игру.

Не гарантируется, что вышеуказанное описание является представительным для оптимальной игры.

РЕШЕНИЕ

Рассмотрим некоторую комбинацию шестерён a_1, a_2, …, a_n и будем вращать первую со скоростью v_1. Тогда скорость вращения последней шестерни будет вычисляться по формуле: $v_n = v_1 \times \frac{a_1}{a_2} \times \frac{a_2}{a_3} \times \cdots \frac{a_{n-2}}{n-1} \times \frac{a_{n-1}}{a_n} = v_1 \times \frac{a_1}{a_n}$. Видно, что скорость последней шестерни зависит только от числа зубьев первой и последней шестерён. Для того, чтобы скорости первой и последней шестерён совпадали необходимо и достаточно, чтобы совпадало число зубьев этих шестерён (рисунок 3).

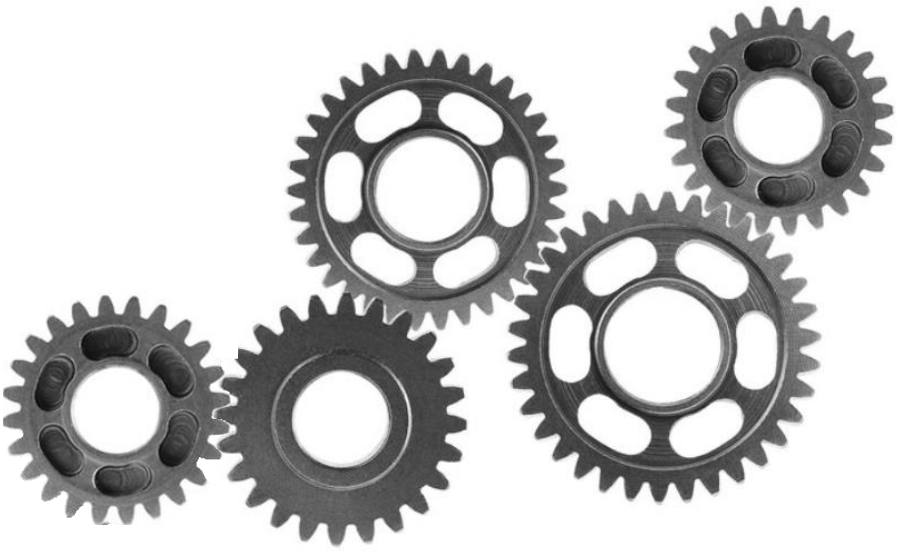


Рисунок 3. Пример правильного зацепления шестерён

Поэтому, если мы хотим получить одинаковую скорость для первой и последней шестерни, нам надо расположить их так, чтобы первая и последняя имели одинаковое число зубьев, а это возможно, только если есть хотя бы одна пара с одинаковым числом зубьев.

Для поиска пары можно подсчитать число различных значений с помощью массива, списка, словаря, счётчика или множества, а можно использовать и простой перебор всех пар, поскольку ограничения очень низкие. Время работы составит O(n), O(n log n) $O(n^2)$.

Один из вариантов решения на языке Python 3.* представлен ниже.

```
k = int(input())
for _ in range(k):
     n = int(input())
    a = list(map(int, input().split()))
    if len(a) > len(set(a)):
        print('Yes')
    else:  # len(a) == len(set(a))
        print('No')
```

3. Ай и Май играют

Ай и Май решили развлечься после сложной технической задачи. Для этого они придумали следующую игру. У Ая и Мая есть 2 массива a и b длины n. Игра продолжается n ходов, причём на нечётных ходах играет Ай, а на чётных Май. В свою очередь хода игрок может поменять местами элементы a_i и b_i или оставить их на своих местах. Например, на первом ходу Ай может поменять местами a_1 и b_1, а Май на втором ходе может поменять a_2 и b_2.

После выполнения n ходов Ай подсчитывает $a_1 \oplus^1 a_2 \oplus \cdots \oplus a_n$, а Май получает счет $b_1 \oplus b_2 \oplus \cdots \oplus b_n$. Игрок с более высоким счетом выигрывает. Если у игроков одинаковый счет, игра заканчивается вничью.

Определите исход игры при оптимальной игре. Более формально, один игрок считается победителем при оптимальной игре, если существует стратегия для него, которая гарантирует победу, независимо от выбора противника. Игра считается ничьей при оптимальной игре, если ни один из игроков не имеет такой стратегии.

Входные данные

Первая строка содержит одно целое число k ($1 \le k \le 10^4$) — количество наборов входных данных.

Первая строка каждого набора входных данных содержит одно целое число n ($1 \le n \le 2 \cdot 10^5$).

Вторая строка каждого набора входных данных содержит n целых чисел, $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^6$).

Третья строка каждого набора входных данных содержит n целых чисел, $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le 10^6$).

Гарантируется, что сумма n по всем наборам входных данных не превышает $2 \cdot 10^5$.

Выходные данные

Для каждого набора входных данных выведите в одной строке «Aji», если Ай выигрывает при оптимальной игре, «Mai», если Май выигрывает при оптимальной игре, или «Tie», если игра заканчивается вничью при оптимальной игре. Выводить ответ можно в любом регистре, т.е. ответы «Aji», «aji» и «AJI» будут восприниматься как «Aji».

Ограничения

ограничение по времени на тест 2 секунды

[1] Побитовое исключающее ИЛИ

ограничение по памяти на тест 256 мегабайт

Примеры

Входные данные	Выходные данные
5 4 1 4 6 1 3 2 3 7 6 20 11 1 7 7 0 14 8 3 6 17 6 4 2 6 3 6 3 4 7 1 5 1 4 5 5 3 6 7 1 2 13 6 9 5 9 17 17 6 1 13 6 13 1 15	Mai Aji Tie Aji Mai

Примечание

В первом примере один из возможных исходов игры может быть следующим:

- На первом ходе Ай выбирает поменять местами a1 и b1. Теперь массивы: a=[3,4,6,1] и b=[1,2,3,7].
- На втором ходе Май выбирает поменять местами a2 и b2. Теперь массивы: a=[3,2,6,1] и b=[1,4,3,7].
- На третьем ходе Ай выбирает пропустить ход.
- На четвертом ходе Май выбирает поменять местами a4 и b4. Теперь массивы: a=[3,2,6,7] и b=[1,4,3,1].

Теперь финальный счет Ай равен $3 \oplus 2 \oplus 6 \oplus 7 = 0$, а финальный счет Май равен $1 \oplus 4 \oplus 3 \oplus 1 = 7$. Поэтому Май выигрывает игру.

Не гарантируется, что вышеуказанное описание является представительным для оптимальной игры.

РЕШЕНИЕ

Рассмотрим XOR всех чисел, данных Ай и Май, $a_1 \oplus a_2 \oplus \cdots \oplus a_n \oplus b_1 \oplus b_2 \oplus \cdots \oplus b_n$. Это будет окончательный счёт в игре между Ай и Май. Разложим каждое из представленных чисел на степени двойки. Подсчитаем, сколько раз встречается каждая степень. Если это число будет чётным, то независимо от того, кто заберёт себе эти степени, на результат игры это не повлияет. Или у каждого игрока будет эта степень, или она будет только у одного игрока, но её вклад в результат игры будет равен 0.

Найдём самую большую степень, которая встречается нечётное число раз. Тот игрок, который сможет забрать её себе, очевидно, сможет выиграть и всю игру, так как младшие степени двойки не смогут достичь её значения. Найдём, в какой позиции последний раз встречается эта степень только в одном из чисел a_i и b_i, для всех i от 1 до n. Если позиция i будет нечётной, то выиграет Ай, а иначе – Май.

Один из вариантов решения на языке Python 3.* представлен ниже.

```
k = int(input())
```

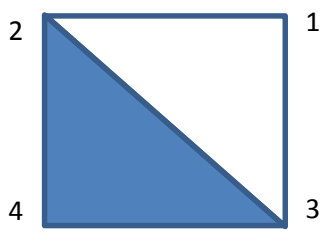
```
for _ in range(k):
    n = int(input())
    a = list(map(int, input().split()))
    b = list(map(int, input().split()))

    pow2 = [0] * 20
    for i in range(n):
        x = a[i] ^ b[i]
        x2 = bin(x)[2:][::-1]
        for j in range(len(x2)):
            if x2[j] == '1':
                pow2[j] = 1 - pow2[j]

    ans = 'Tie'
    if sum(pow2) == 0:
        print(ans)
        continue
    for i in range(19, -1, -1):
        if pow2[i]:
            break
    i_max = i
    for i in range(n - 1, -1, -1):
        x = a[i] ^ b[i]
        x2 = bin(x)[2:][::-1]
        if i_max < len(x2) and x2[i_max] == '1':
            if i % 2:
                ans = 'Mai'
            else:
                ans = 'Aji'
            break
    print(ans)
```

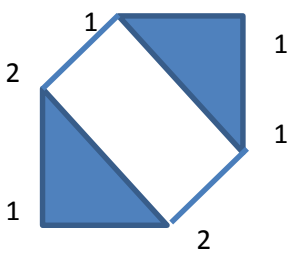
4. Драгоценные треугольники

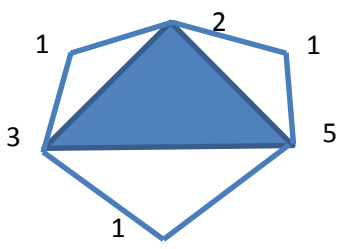
Прогуливаясь по волшебному лесу, Вася обнаружил выпуклый n-угольник с драгоценными камнями в вершинах ($n \geq 3$). Вершины нумеруются по часовой стрелке, начиная с 1. Каждый драгоценный камень в вершине имеет определённую ценность a_i.

Вася решил продать этот n-угольник, но в скупке принимают только треугольники, причём количество монет за треугольник зависит только от произведения ценностей драгоценных камней в его вершинах.

Вася хочет распилить этот n-угольник на треугольники так, чтобы сумма произведений вершин треугольников была как можно больше. Никаких других операций он делать не может. Каждый драгоценный камень может оказаться только в одном треугольнике.







Например, из четырёхугольника с вершинами 2, 1, 3, 4 Вася может вырезать 1 треугольник с вершинами 3, 4 и 2, получив 2 * 3 * 4 = 24 монеты. Для 6-угольника 1, 2, 1, 1, 1, 2 можно получить 2 треугольника: 2, 1, 2 и 1, 1, 1, что даёт 5 монет. Из 6-угольника 1, 2, 1, 5, 1, 3 можно вырезать один треугольник 2, 3, 5, получив 30 монет.

Входные данные

Каждый тест состоит из нескольких наборов входных данных. В первой строке находится одно целое число k ($1 \le k \le 10^4$) — количество наборов входных данных. Далее следует описание наборов входных данных.

Первая строка каждого набора содержит одно целое число n — количество вершин ($3 \le n \le 400$).

Вторая строка каждого набора содержит $a_1, a_2, \ldots, a_n$ — целые числа, написанные на вершинах ($1 \le ai \le 1000$).

Гарантируется, что сумма n^3 по всем наборам входных данных не превышает 400^3.

Выходные данные

Для каждого набора данных выведите в отдельной строке максимальное число монет, которое можно получить нарезав найденный драгоценный n-угольник.

Ограничения

ограничение по времени на тест	4 секунды
ограничение по памяти на тест	512 мегабайт

Примеры

Входные данные	Выходные данные
5 3 1 2 3 4 2 1 3 4 6 2 1 2 1 1 1 6 1 2 1 3 1 5 9 9 9 8 2 4 4 3 5 3	6 24 5 30 732

Примечание

В первом примере вы можете взять только один треугольник. Максимальное число монет равно 6.

Во втором примере вы можете вырезать только один треугольник. Максимальное число монет 24. Смотри рисунок выше.

В третьем примере вы можете вырезать два треугольника. Максимальное число монет 5. Смотри рисунок выше.

В четвертом примере вы можете вырезать два треугольника. Однако вырезание двух треугольников приводит к счету либо 6+5=11, 15+2=17, либо 10+3=13. Максимальный счет 30 достигается, вырезая только один треугольник с камнями 2, 3 и 5.

В пятом примере вы можете нарезать три треугольника. Максимальный счет 732 достигается, разрезая три треугольника следующим образом: 9*9*8 + 2*4*3 + 4*3*5.

РЕШЕНИЕ

В этой задаче есть очевидное решение. Надо перебрать все комбинации точек i, j, k ($1 \le i < j < k \le n$), которые представляют рассматриваемый в данный момент треугольник. Очевидно, что вырезав треугольник с вершинами i, j и k, можно найти стоимость этого треугольника и стоимости оставшихся частей. Тогда искомое значение сумм произведений будет равно $a_i \times a_j \times a_k$ + суммы для каждого интервала (1 … i-1, i+1 … j-1, j+1 … k-1, k-1 … n). Для вычисления значений сумм каждого интервала можно использовать рекурсивную функцию, однако такой подход будет постоянно пересчитывать одни и те же значения, поэтому попробуем решить задачу с помощью динамического программирования (ДП).

Простое следование приведённым выше рассуждениям приводит нас с следующей формуле ДП.

$$dp_{L,R} = \max_{L \le i < j < k \le R} (score(i,j,k) + dp_{L,i-1} + dp_{i+1,j-1} + dp_{j+1,k-1} + dp_{k+1,R})$$

Этот вариант будет работать правильно, но довольно медленно $O(n^5)$. Посмотрим, что здесь можно улучшить.

Сделаем переход от более мелкой подзадачи. Можно заметить, что score(i, j, k) также будет найдено с L=i и R=k позже в одной из более мелких подзадач. Если мы не выбираем треугольник i, j, k, то при L=i и R=k значение будет

$$dp_{L,R} = \max_{L \le i < j < k \le R} (dp_{L,i} + dp_{i+1,j} + dp_{j,R}).$$

Это позволит записать переход

$$dp_{L,R} = max(\max_{L<i<R} \left(score(L,i,R) + dp_{L+1,i-1} + dp_{i+1,R-1}\right), \max_{L \le i < j \le R} (dp_{L,i} + dp_{i+1,j} + dp_{j,R}))$$

Этот вариант позволит избавиться от лишних пересчётов и снизит временную сложность до $O(n^4)$, но этот вариант всё ещё недостаточен для полного решения.

Можно перебирать только одно из рёбер треугольника, поскольку второе ребро появится в одной из подзадач позднее. Это приведёт к следующему переходу

$$dp_{L,R} = max(\max_{L<i<R} \left(score(L,i,R) + dp_{L+1,i-1} + dp_{i+1,R-1}\right), \max_{L \le i < R} (dp_{L,i} + dp_{i+1,R}))$$

Это позволит сократить сложность до $O(n^3)$, что достаточно для получения полного решения.

Пример решения на языке C++17 представлен ниже.

```
#include <iostream>

typedef long long int ll; // 400 / 3 * (ai = 1000)^3 > 2^31
const int MAX_N = 400;


ll calc(int n, int a[]){
    int i, j;
    ll dp[MAX_N][MAX_N];
    for(i = 0; i < n; i++)
        for(j = 0; j < n; j++)
            dp[i][j] = 0;

    for(int le = n - 3; le >= 0; le--){
        dp[le][le + 2] = a[le] * a[le + 1] * a[le + 2];
```

```
        for(int ri = le + 3; ri < n; ri++){
            dp[le][ri] = dp[le + 1][ri];
            for(i = le + 1; i < ri; i++){
                ll   v1   =   dp[le+1][i-1]   +   dp[i+1][ri-1]   +
a[le]*a[i]*a[ri];
                ll v2 = dp[le][i] + dp[i+1][ri];
                if(v1 > dp[le][ri])
                    dp[le][ri] = v1;
                if(v2 > dp[le][ri])
                    dp[le][ri] = v2;
            }
        }
    }

    return dp[0][n - 1];
}


int main(){
    int k, n, a[MAX_N];
    std::cin >> k;
    while(k--){
        std::cin >> n;
        for(int i = 0; i < n; i++)
            std::cin >> a[i];

        std::cout << calc(n, a) << '\n';
    }
    return 0;
}
```