

§3 Заключительный этап: индивидуальная часть

3.1 Задачи по физике

Можно использовать решение одних подзадач для решения других. В этом случае отсутствие любых формул или рассуждений, которые уже были использованы для решения ранее, ошибкой не является.

Задача 1. Испытания экспериментальных аппаратов

Два экспериментальных аппарата отправили на испытания. Аппараты представляют собой шары, способные двигаться как по суше, так и в воде. Масса аппаратов M .

- 1) (1 балл) Два аппарата, движущиеся по прямой с одинаковыми скоростями V на расстоянии L друг от друга, преодолевают зону на участке маршрута длиной L , где их скорость уменьшается наполовину. Какой суммарный путь пройдет один аппарат относительно другого при прохождении этой зоны?

Решение:

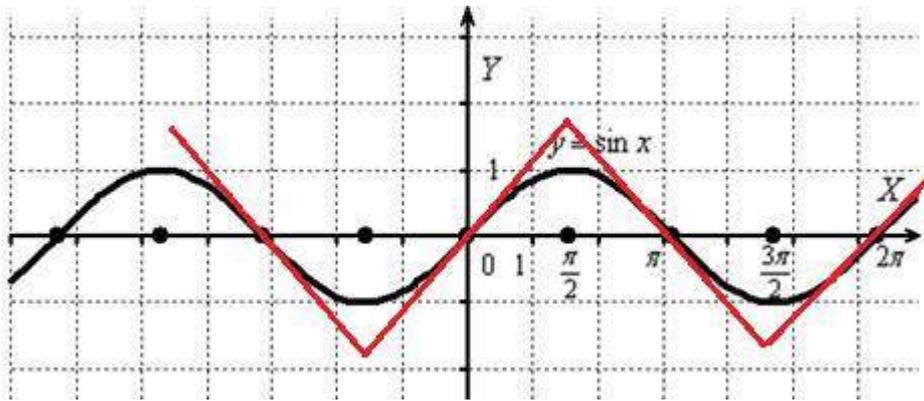
Путь будет складываться из двух участков – участка, где едущий впереди аппарат отстаёт и участка, где едущий позади аппарат проходит тормозную зону и тоже отстаёт. Пусть a – минимальное расстояние между катерами в процессе движения. Тогда $S = 2 \cdot (L - a)$.

$$a = L - (v - v/2) \cdot \frac{L}{v} = \frac{L}{2} \rightarrow S = L. \text{ Ответ: } S = L.$$

- 2) (2 балла) Первый аппарат движется к цели по синусоидальной траектории со скоростью $1.5v$, а второй аппарат по прямой со скоростью v . Обе траектории полностью проходят по поверхности гладкого озера. Какой из аппаратов быстрее доберётся до цели?

Решение:

Длина синусоидальной траектории меньше длины ломаной под углом 45° к горизонтали, так как синусоида находится внутри этой ломаной. Длина такой ломаной в $\sqrt{2}$ раза больше длины прямой. Так как $\sqrt{2} < 1.5$, то аппарат со скоростью $1.5v$ быстрее достигнет конечной цели. Ответ: первый аппарат.



- 3) (5 баллов) Скорость первого аппарата v , скорость второго аппарата равна $2v$, расстояние между ними L . Второй аппарат преследует первый так, что скорость второго аппарата всегда направлена на первый. Как должен двигаться первый аппарат, чтобы второй достиг его за наибольшее время и чему это время равно?

Решение:

Назовём начальную точку первого аппарата A , второго – B). Наибольшее время получается тогда, когда относительная скорость наименьшая, а она наименьшая, когда первый катер удаляется по прямой AB и равна $2v-v$. Тогда $t = L/v$.

- 4) (2 балла) Чему равен период малых колебаний такого сферического аппарата на поверхности воды, если он движется с ускорением a , площадь сечения на уровне ватерлинии равна S , плотность воды равна ρ ?

Решение:

Если дрон сферический, то при наклоне аппарата можно использовать S как площадь сечения дрона на новом уровне воды, возникающем из-за силы инерции. При погружении аппарата на x возникает прибавка к силе Архимеда $\rho \cdot \sqrt{a^2 + g^2} \cdot S \cdot x$, играющая роль возвращающей силы малых колебаний. Тогда $T = \frac{2\pi}{\omega} = 2 \cdot \pi \cdot$

$$\sqrt{\frac{m}{\rho \cdot \sqrt{a^2 + g^2} \cdot S}}.$$

Ответ: $T = 2 \cdot \pi \cdot \sqrt{\frac{m}{\rho \cdot \sqrt{a^2 + g^2} \cdot S}}.$

- 5) (10 баллов) Во время испытаний аппарат должен был погрузиться на дно глубокого водоема (глубина много больше размеров аппарата). При выключенном двигателе аппарат погружается на дно за время t . Двигатель сообщает аппарату силу тяги F , направленную горизонтально. За какое время доплывёт до дна аппарат с включенным двигателем, если сила вязкого сопротивления воды пропорциональна скорости? Волнами и ветром на поверхности можно пренебречь.

Решение:

Так как сила сопротивления зависит от скорости, то по истечении небольшого промежутка времени дрон будет тонуть равномерно со скоростью v . При движении дрона с выключенным двигателем эту скорость можно найти из уравнения:

$$M\vec{g} + \vec{F}_{\text{арх}} + \vec{F}_{\text{сопр}} = \vec{0}$$

Пусть ρ – средняя плотность дрона. Тогда из этого уравнения получим:

$$(\rho - \rho_0) \cdot V \cdot g = \beta \cdot v^2$$

Где ρ_0 — плотность воды, V — объем дрона, β — коэффициент пропорциональности между силой сопротивления и квадратом скорости.

Так как, по условию, море глубокое, можно пренебречь начальным участком движения, на котором скорость дрона изменяется, тогда: $t = \frac{H}{v} = H \cdot \sqrt{\frac{\beta}{(\rho - \rho_0) \cdot V \cdot g}}$.

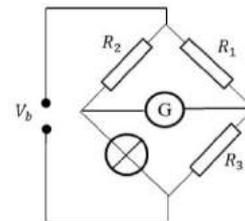
Добавление силы тяги F эквивалентно созданию другого эффективного ускорения свободного падения: $\vec{g}' = \vec{g} - \frac{\vec{F}}{M}$. По модулю оно равно $\sqrt{g^2 + \left(\frac{F}{M}\right)^2}$. Дрон будет двигаться вдоль направления этого \vec{g}' и, соответственно, пройдет путь $L = \frac{H \sqrt{g^2 + \left(\frac{F}{M}\right)^2}}{g}$.

Время прямо пропорционально H и обратно пропорционально \sqrt{g} . Значит:

$$\frac{T}{t} = \sqrt[4]{1 + \left(\frac{F}{Mg}\right)^2} \cdot \text{Ответ: } T = t \sqrt[4]{1 + \left(\frac{F}{Mg}\right)^2}.$$

Задача 2. Вокруг измерительного моста

Источник напряжения связали с лампой накаливания через измерительный мост Уитстона, как показано на рисунке ниже



Гальванометр G позволяет измерять ток, возникающий при дисбалансе измерительного моста. Изменяя внешнее напряжение, можно добиться наиболее эффективного режима работы лампы, рассчитав сопротивления таким образом, чтобы он соответствовал нулевому току, проходящему через гальванометр.

- 1) (1 балл) Чему равно сопротивление схемы, если мост сбалансирован, $R_1=1$ Ом, $R_2=1$ Ом, $R_3=4$ Ом?

Решение:

Условия сбалансированности моста: $R_2=R_3=4$ Ом; Отсюда, общее сопротивление: 2,5 Ом

Ответ: 2.5 Ом

- 2) (2 балла) Чему равно внешнее напряжение источника V_b , выделяемая мощность на лампе, и коэффициент полезного действия схемы, если $R_1=R_2=R_3=4$ Ом, внутреннее сопротивление источника напряжения равно 0.75 Ом, а мост сбалансирован; Вольт-амперная характеристика лампы накаливания задаётся формулой: $V = 3 \cdot I + 7 \cdot I^2$ (формула дана в единицах СИ).

Решение:

Т.к. мост сбалансирован, напряжение на лампе должно быть таким же, как и на сопротивлении 4В, а ток одинаковый в двух ветвях, тогда можно записать, воспользовавшись ВАХ лампы: $4I_l = 3I_l + 7I_l^2$. Общий ток равен сумме одинаковых токов в двух ветвях: $I = 2I_l = 2/7$ А, Общее напряжение может быть вычислено, как произведение общего сопротивления этой цепи на общий ток: $4,75 \text{ Ом} \cdot 0,29 \text{ А} = V_b = 19/14$ В, Мощность, как произведение тока на лампе на напряжение на лампе: $P = 4/49$ Вт, Аналогично вычислив мощность, выделяющуюся во всей цепи, найдем КПД = $4/49 : 19/49 = 4/19$.

Ответ: КПД = 4/19.

- 3) 2 балла В условиях предыдущего пункта напряжение равно 4 В. Чему равна мощность, выделяемая на лампе накаливания?

Решение:

Аналогично предыдущему пункту: $I = 8/7$ А, $P = 32/7$ Вт.

Ответ: $P = 32/7$ Вт.

- 4) (5 баллов) Можно ли, изменяя напряжение схемы в условиях пункта 2, добиться того, чтобы КПД потребления энергии лампой достигнул 25 % ?

Решение:

Рассуждая аналогично пункту 2) - нет, так как $\text{КПД} > \frac{64 \cdot 3}{7 \cdot 125} \approx 0.219429$ (макс. при $I \rightarrow 0$, $x \rightarrow 3$, где $\eta = \frac{16 \cdot x}{(x+4) \cdot (6.75x+11)}$, x – сопротивление лампы при выбранном напряжении, так как $x_{\text{opt}} = \sqrt{176/27} < 3$).

Ответ: нет, нельзя.

- 5) **(10 баллов)** Данную схему в условиях пункта 2 использовали для измерения вольт-амперной характеристики неизвестной лампы, изменяя внешнее напряжение. Данные измерений даны в таблице ниже. Считая, что вольт-амперная характеристика имеет вид $V(I) = \alpha \cdot I + \beta \cdot I^2$, найдите неизвестные коэффициенты α и β .

Напряжение V_b , В	Сила тока на гальванометре I_g , А Положительное направление – влево
6.125	0.25
15.625	0.25

Решение:

Пусть $V_b/I_g = y$, x - отношение напряжения на лампе к силе тока в ней же. Это удобно, так как тогда получается, что: $x = \alpha + \beta \cdot I$

Ток в цепи сначала делится пополам, потом перераспределяется. Через лампу пойдёт ток, равный $4 \cdot I / (4+x)$

Эквивалентное сопротивление схемы равно $2.75 + 4x / (4+x) = (6.75 \cdot x + 11) / (x+4)$

Тогда $I = U_b \cdot (x+4) / (6.75 \cdot x + 11)$

Ток через лампу равен $4 \cdot U_b / (6.75 \cdot x + 11)$

Вычитаем из него ток $0.5 \cdot I$, тогда:

ток через гальванометр равен $U_b \cdot (4-x) / (13.5 \cdot x + 22)$

Отсюда $y = (13.5 \cdot x + 22) / (4-x)$

А ток через лампу соотносится с током через гальванометр по формуле: $I_l = 8 \cdot I_g / (4-x)$

Тогда $y = \frac{13.5 \cdot x + 22}{4-x}$; $x = \frac{4y-22}{y+13.5}$. Далее: $x = \alpha + \frac{8 \cdot I_g}{4-x} \beta$. Составляем систему из двух линейных уравнений, решаем. Получается $\alpha = \beta = 1$.

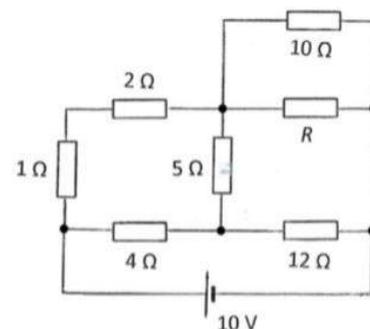
Ответ: $\alpha = \beta = 1$.

Задача 3. Управление электрическими схемами

- 1) **(2 балла)** Предложите электрическую схему, в которой при нажатии первого выключателя загорается первая лампочка, при нажатии второго – первая и вторая загораются вместе, при нажатии третьего – первые три лампочки горят, четвёртого – первая и четвёртая.

Решение:

Подойдёт любой правильный вариант схемы.



- 2) (1 балл) Определите значение сопротивления R схемы, показанной на рисунке справа, при котором тепловая мощность, выделяемая на резисторе 5 Ом, минимальна.

Решение:

Можно добиться нулевой мощности, если заметить, что сопротивление 5 Ом – мостик. Через сбалансированный мостик ток не течет, соответственно, мощность на нем не выделяется. Тогда условие сбалансированности $10R/(R+10) = 9$. Соответственно, $R = 90$ Ом.

Ответ: $R = 90$ Ом.

- 3) (2 балла) Чему равна максимальная мощность, которая может выделяться на этой схеме целиком, если изменять сопротивление R?

Решение:

Максимальная мощность при подключении к постоянному напряжению будет при минимальном сопротивлении всей схемы целиком, а это достигается, когда $R = 0$. Тогда:

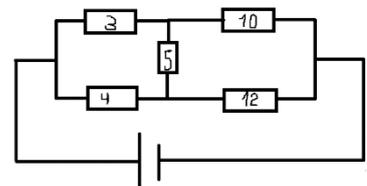
$$R_{ев} = 3 \parallel (4 + 5 \parallel 12) = 3 \parallel \left(4 + \frac{60}{17}\right) = 3 \parallel \frac{128}{17} = \frac{3 \cdot 128}{51 + 128} = \frac{384}{179} \text{ (Ом)}$$

$$\text{Отсюда: } P = \frac{U^2}{R} = 100 \cdot \frac{179}{384} \approx 46.6 \text{ Вт.}$$

Ответ: 46.6 Вт.

- 4) (5 баллов) Чему равна минимальная мощность, которая может выделяться на этой схеме целиком, если изменять сопротивление R?

Минимальная мощность будет при максимальном сопротивлении всей схемы целиком, а это достигается, когда $R \rightarrow \infty$. Нарисуем схему эквивалентного сопротивления:



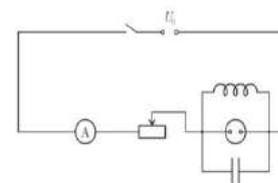
Решение:

Обозначим за ϕ_1 потенциал узла, соединяющего ветви с 3, 5 и 10 Ом, а за ϕ_2 – потенциал узла, соединяющего ветви с 4, 5 и 12 Ом. Алгебраическая сумма токов, входящих в любой узел, равна нулю. Каждый из токов равен отношению разности потенциалов на ветви к её сопротивлению – из этого условия составим уравнения:

$$\begin{cases} \frac{\phi_1 - 10}{3} + \frac{\phi_1 - \phi_2}{5} + \frac{\phi_1}{10} = 0, \\ \frac{\phi_2 - 10}{4} + \frac{\phi_2 - \phi_1}{5} + \frac{\phi_2}{12} = 0; \end{cases} \Rightarrow \begin{cases} \phi_1 = \frac{1025}{134} \\ \phi_2 = \frac{2025}{268} \end{cases} \Rightarrow \begin{cases} I_{10} = \frac{410}{536} \\ I_{12} = \frac{337.5}{536} \end{cases} \Rightarrow \begin{cases} I_{общ} = \frac{1495}{1072} \\ P = U \cdot I_{общ} = \frac{14950}{1072} \approx 13.95 \end{cases}$$

Ответ: 13.95 Вт.

- 5¹¹) (10 баллов) С целью измерения параметров катушки её соединяли параллельно индикаторной неоновой лампе и разными конденсаторами и одной и той же катушкой:



Требуется напряжение $U_n = 80$ В на лампе, чтобы зажечь её.

Лампа погаснет, если напряжение на ней опустится ниже 60 В. Экспериментально замерялся минимальный начальный ток I , при котором лампочка вспыхивает в зависимости от ёмкости включенного в сеть конденсатора. Определите по данным измерений значение индуктивности катушки L и ёмкость C конденсатора.

C , нФ	I , мА
20	58
C	71
40	82

Решение:

Условие зажигания лампочки: $I_0^2 \geq \frac{C \cdot U_n^2}{L}$. Строим график I_0^2 от C и по графику линейной зависимости получаем $C = 30$ нФ. $L = \frac{C \cdot U_n^2}{I^2} = 38.05$ мГн.

Ответ: 38.05 мГн.

Задача 4. Из одной среды в другую

При наблюдении сквозь границу раздела двух прозрачных сред часто наблюдаются искажения изображения, которые необходимо учитывать при настройке фото или видео систем. Абсолютный показатель преломления воды $n = 1.33$.

- 1) (1 балл) Рыбак смотрит на воду вертикально вниз и видит рыбу вблизи поверхности воды. Кажущаяся длина рыбы (параллельно поверхности воды) равна 30 см. Чему равна реальная длина рыбы, если для воды показатель преломления равен $n = 1.33$?

Решение:

30 см, так как в данном случае при взгляде вертикально вниз нет преломления у лучей, движущихся перпендикулярно поверхности воды, и кажущаяся длина рыбы совпадает с реальной её длиной, то есть этот размер не искажается.

Ответ: 30 см.

- 2) (2 балла) Дрон находится под водой на глубине 20 метров, на расстоянии 1000 метров до берега вдоль поверхности воды. На берегу сидит моряк, а на поверхности воды в 500 метрах от берега расположена второй лодка. На какое расстояние нужно передвинуть лодку вдоль поверхности воды, чтобы дрон не смог рассмотреть моряка на берегу? Размерами моряка дрона и лодки можно при решении задачи пренебречь. Все три объекта все время находятся в одной вертикальной плоскости.

Решение:

Если пренебрегать размерами моряка, то луч света должен идти вдоль поверхности воды после преломления. Это значит, что он проходит через угол полного внутреннего отражения: $\beta = \arcsin\left(\frac{1}{n}\right) \approx 48.75^\circ$. Тангенс этого угла равен отношению расстояния до точки выхода луча к глубине погружения первого дрона, следовательно это расстояние равно 22.808 метра. Значит, второй дрон нужно передвинуть на расстояние, равное $500 - 22.808 = 477.192$ метра.
 Ответ: 477.192 метра.

- 3) (2 балла) Под каким углом к вертикали должна быть направлена камера подводного дрона-разведчика, чтобы получить изображение лица рыбака на берегу, если дрон находится под водой на глубине 20 метров, а до берега 40 метров? Рост рыбака 2 метра. Требуется получить приближённый ответ из трёх значащих цифр такой, чтобы его точность была не хуже, чем четверть градуса.

Решение:

Для нахождения угла камеры к вертикали $\phi = \beta$ (как накрест лежащие на рисунке) можно использовать закон преломления и геометрические соображения:

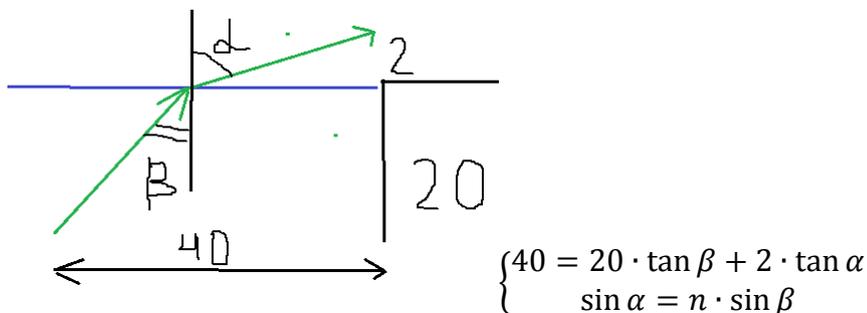
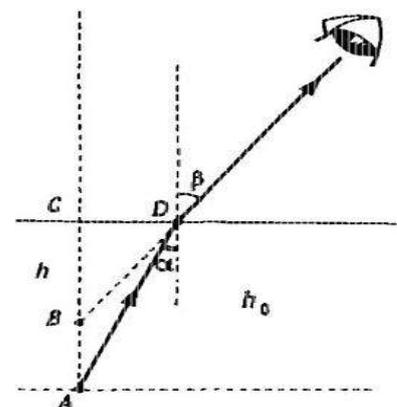


Схема вычислений: $\tan \alpha = 20 - 10 \cdot \tan \beta$; $\sin \alpha = \frac{\tan \alpha}{\sqrt{1 + \tan^2 \alpha}}$; $\beta = \arcsin\left(\frac{\sin \alpha}{n}\right)$

Можно сначала вместо β подставить приближённое значение, затем после прохождения этой схемы 1 раз вычислить более точное и потом опять. Когда два соседних значения будут отличаться меньше, чем на четверть градуса – это будет означать, что ответ можно будет записать с требуемой точностью.

Если в качестве первой итерации подставить угол полного внутреннего отражения, который равен 48.75, то результатом будет 48.50 градуса. Следующей итерацией получаем 48.32 градуса, затем 48.33. Таким образом, оценка 48.3 будет являться достаточно точной. Кроме того, в качестве полностью правильного ответа может быть принято любое число в интервале от 48.1 до 48.5.

- 4) (5 баллов) Рыбак сидит в покоящемся на воде катере, пока его друг аквалангист спускается под воду с этого катера вертикально вниз. Рыбак смотрит в воду сверху вниз и видит, что аквалангист спускается под воду со скоростью 0.5 м/с и находится на глубине 20 метров. Каковы реальная глубина и скорость аквалангиста, если рост моряка 2 метра?



Решение:

Если рыбак смотрит вертикально вниз, то углы α , β стремятся к нулю (показано на рисунке справа):

Тогда истинная глубина h_0 может быть найдена как: $h_0 = h \cdot \frac{\beta}{\alpha} = n \cdot h \approx 26.67 \text{ м}$.
Аналогично скорость тоже в n раз больше, то есть $v_0 = v \cdot n \approx 0.67 \text{ м/с}$.

Ответ: 26.67 м, $v=0.67 \text{ м/с}$.

- 5) (10 баллов) Моряк из предыдущего пункта стоит на берегу океана и видит аквалангиста на поверхности воды на расстоянии $L = 1000$ метров от него. Аквалангист погружается в воду. Моряк видит с берега его находящимся на глубине $h = 20$ метров и опускающимся вниз со скоростью $v = 0.5 \text{ м/с}$. Каковы реальная глубина и скорость аквалангиста? При расчётах считать, что $x + h \ll L$ (угол к горизонту, на котором моряк видит изображение, очень мал).

Решение:

Если угол к горизонту, на котором моряк видит изображение, считать пренебрежимо малым, то истинная глубина H лодки может быть найдена по формуле: $H = \frac{L \cdot h}{x+h} \cdot \sqrt{n^2 - 1} \approx 797 \text{ метров}$ (можно использовать предыдущий рисунок, с той лишь разницей, что углы α и β большие, $\beta \approx 90^\circ$).

Скорость погружения получается, если продифференцировать эту функцию по времени, используя $dh/dt = v$. Тогда: $\frac{dH}{dt} = v \cdot \sqrt{n^2 - 1} \cdot L \cdot \frac{x}{(x+h)^2} \approx 1.81 \frac{\text{м}}{\text{с}}$.

Ответ: 1.81 м/с

3.2. Задачи по информатике

Задача 3.2.1. Траектории (15 баллов)

Уборка - дело важное, вот только никому не нравится ей заниматься. И Сереже тоже! Но ему повезло - недавно его мама подарила ему набор из большого количества роботов, которые могут делать за Сережу работу по дому.

Однако, радость Сережи была не долгой. Он совершенно не представляет, как ими пользоваться. Меню настроек - сложное, а инструкция - на английском.

Серёжа знает, что вы увлекаетесь робототехникой, и обратился к вам за помощью.

Через пару часов началось тестирование: роботы носились по всей квартире. Бесцельно, но весело.

Оказалось, что каждый робот движется по заранее определенной (запрограммированной по умолчанию) траектории. Траектория движения каждого из роботов представляет собой ломаную. А сами роботы имеют цилиндрическую форму и убирают весь мусор, который окажется под ними в какой-либо момент времени.

Для начала Сережа поручил вам определить пересекаются ли траектории двух роботов.

Формат входных данных

В первой строке два целых числа:

- $2 \leq n \leq 500$ - количество точек в траектории первого робота;
- $2 \leq m \leq 500$ - количество точек в траектории второго робота.

В следующих двух строках заданы точки ($2 \cdot n$ целых чисел в первой строке и $2 \cdot m$ целых чисел во второй строке): $x_{i,j} y_{i,j}$ — j -я точка траектории i -го робота ($|x_{i,j}| \leq 10^9$; $|y_{i,j}| \leq 10^9$).

Никакие две последовательные точки никакой из траекторий не совпадают.

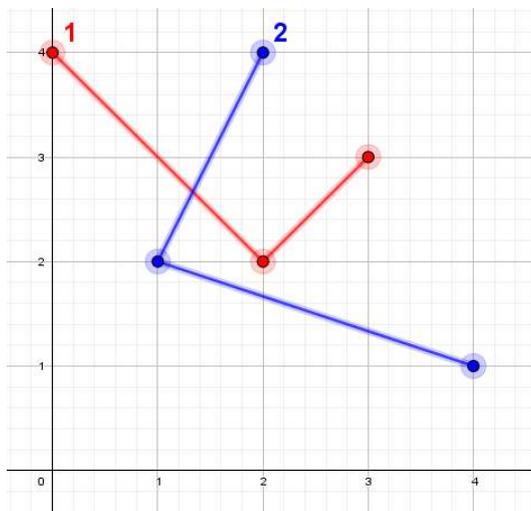


Рис. 1.1: Рисунок к первому примеру

Формат выходных данных

Выведите *Yes* , если траектории пересекаются, иначе - *No*.

Примеры

Пример №1

Стандартный ввод
3 3 0 4 2 2 3 3 2 4 1 2 4 1
Стандартный вывод
Yes

Пример №2

Стандартный ввод
2 2 0 0 1 1000000000 1 999999999 2 999999999
Стандартный вывод
No

Способ оценки работы

За решение задачи начислялось:

- **5 баллов**, если пройдена только первая группа тестов;
- **10 баллов**, если пройдена первая и вторая группы тестов;
- **15 баллов**, если пройдены все тесты.

Для проверки результата использовался следующий код на языке Python3

```
def generate():  
    return []  
  
def check(reply, clue):  
    return reply.strip() == clue.strip()  
  
x = []  
y = []  
  
# triangle area  
def area(a,b,c):  
    s = (x[b] - x[a]) * (y[c] - y[a]) - (y[b] - y[a]) * (x[c] - x[a])  
    return -1 if s < 0 else (1 if s > 0 else 0)  
  
# bounding box  
def box(a, b, c, d):  
    return max(min(a, b), min(c, d)) <= min(max(a, b), max(c, d))
```

```

# пересечение отрезков
def intersect(a, b, c, d):
    return box(x[a],x[b],x[c],x[d]) &
           box(y[a],y[b],y[c],y[d]) &
           (area(a, b, c) * area(a, b, d) <= 0) &
           (area(c, d, a) * area(c, d, b) <= 0)

def solve(dataset):
    ds = dataset.splitlines()

    s = ds[0].split()
    n = int(s[0])
    m = int(s[1])

    s = ds[1].split()
    for i in range(n):
        x.append(int(s[i*2]))
        y.append(int(s[i*2+1]))

    s = ds[2].split()
    for i in range(m):
        x.append(int(s[i * 2]))
        y.append(int(s[i * 2 + 1]))

    inter = False

    for i in range(n - 1):
        for j in range(m - 1):
            inter |= intersect(i, i + 1, n + j, n + j + 1)

    return "Yes" if inter else "No"

```

Решение

В данной задаче достаточно проверить пересекается ли хотя бы один отрезок первой траектории с хотя бы одним отрезком второй траектории.

Асимптотика: $O(n \cdot m)$

Пример программы

Ниже представлено решение на языке Python3

```

1 x = []
2 y = []
3
4 # triangle area
5 def area(a,b,c):
6     s = (x[b] - x[a]) * (y[c] - y[a]) - (y[b] - y[a]) * (x[c] - x[a])
7     return -1 if s < 0 else (1 if s > 0 else 0)
8
9 # bounding box
10 def box(a, b, c, d):
11     return max(min(a, b), min(c, d)) <= min(max(a, b), max(c, d))
12
13
14 # пересечение отрезков

```

```

15 def intersect(a, b, c, d):
16     return box(x[a],x[b],x[c],x[d]) & box(y[a],y[b],y[c],y[d]) \
17     & (area(a, b, c) * area(a, b, d) <= 0) \
18     & (area(c, d, a) * area(c, d, b) <= 0)
19
20 def solve(dataset):
21     ds = dataset.splitlines()
22
23     s = ds[0].split()
24     n = int(s[0])
25     m = int(s[1])
26
27     s = ds[1].split()
28     for i in range(n):
29         x.append(int(s[i * 2]))
30         y.append(int(s[i * 2 + 1]))
31
32     s = ds[2].split()
33     for i in range(m):
34         x.append(int(s[i * 2]))
35         y.append(int(s[i * 2 + 1]))
36
37     inter = False
38
39     for i in range(n - 1):
40         for j in range(m - 1):
41             inter |= intersect(i, i + 1, n + j, n + j + 1)
42
43     return "Yes" if inter else "No"
44
45 print(solve(input() + '\n' + input() + '\n' + input()))

```

Задача 3.2.2. Радиус робота-уборщика (20 баллов)

Пока вы занимались проверкой пересечения траекторий, Серёжа не терял времени даром. Он разработал идеальную (по его мнению) траекторию движения робота уборщика. Учёл (по его мнению) все факторы: расположение мебели, обуви и т. д.

И теперь он просит вас рассчитать, какой должен быть минимальный радиус робота, чтобы он смог убрать весь мусор. При этом робот должен двигаться по траектории, разработанной Серёжей, а мусор - это набор из m материальных точек.

Формат входных данных

В первой строке два целых числа:

- $2 \leq n \leq 500$ - количество точек в траектории робота;
- $1 \leq m \leq 500$ - количество мусор-точек.

В следующей строке заданы точки ($2 \cdot n$ целых чисел): $x_i y_i$ — i -я точка траектории робота ($1 \leq i \leq n$; $|x_i| \leq 10^9$; $|y_i| \leq 10^9$).

В следующей строке заданы точки ($2 \cdot m$ целых чисел): $x_j y_j$ — j -я точка мусора ($1 \leq j \leq m$; $|x_j| \leq 10^9$; $|y_j| \leq 10^9$).

Никакие две последовательные точки траекторий не совпадают.

Формат выходных данных

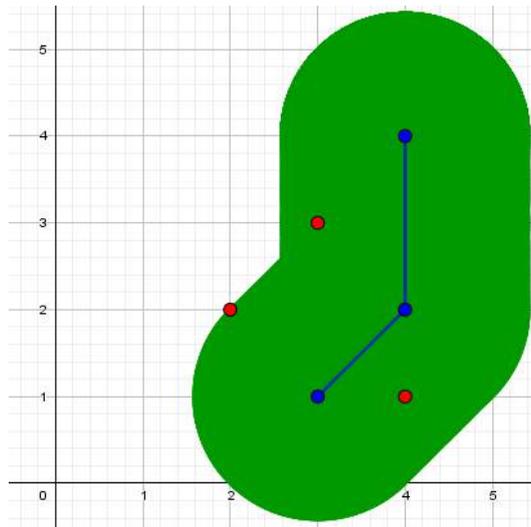


Рис. 1.2: Рисунок к первому примеру

Одно неотрицательное число – минимальный радиус.

Ваш ответ будет засчитан, если его абсолютная или относительная ошибка не превосходит 10^{-6} . Формально, пусть ваш ответ равен a , а ответ жюри равен b . Ваш ответ будет засчитан, если $\frac{|a-b|}{\max(1,b)} \leq 10^{-6}$.

Примеры

Пример №1

Стандартный ввод
3 3 4 4 4 2 3 1 4 1 2 2 3 3
Стандартный вывод
1.4142135623730951

Пример №2

Стандартный ввод
2 1 0 0 1 1 1 1
Стандартный вывод
0.0

Способ оценки работы

За решение задачи начислялось:

- **5 баллов**, если пройдена только первая группа тестов;
- **5 баллов**, если пройдена первая и вторая группы тестов;

- **10 баллов**, если пройдены все тесты.

Для проверки результата использовался следующий код на языке Python3

```
def generate():
    return []

def check(reply, clue):
    a = float(reply)
    b = float(clue)
    return abs(a - b) / max(1.0, b) <= 1e-6

from collections import namedtuple
from math import sqrt

Point = namedtuple("Point", ["x", "y"])

# расстояние между точками a и b
def dist(a, b):
    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y))

# расстояние между точкой c и отрезком ab
def dist2(a, b, c):
    A = a.y - b.y
    B = b.x - a.x
    C = b.y * a.x - a.y * b.x
    cc = B * c.x - A * c.y
    d = A * A + B * B
    p = Point((cc * B - C * A) / d, -(cc * A + C * B) / d)

    return dist(c, p) if dist(a, p) + dist(p, b) - dist(a, b) < 1e-7 else min(dist(c, a), dist(c, b))

def solve(dataset):
    ds = dataset.splitlines()

    s = ds[0].split()
    n = int(s[0])
    m = int(s[1])

    s = ds[1].split()
    p = []
    for i in range(n):
        p.append(Point(int(s[i * 2]), int(s[i * 2 + 1])))

    s = ds[2].split()
    pm = []
    for i in range(m):
        pm.append(Point(int(s[i * 2]), int(s[i * 2 + 1])))

    dist = [1e10] * m

    for i in range(n - 1):
        for j in range(m):
            dist[j] = min(dist[j], dist2(p[i], p[i + 1], pm[j]))

    ans = dist[m-1]
    for i in range(m - 1):
        ans = max(ans, dist[i])

    return str(ans)
```

Решение

В данной задаче достаточно найти для каждой мусор-точки расстояние до ближайшего к ней отрезка, после чего найти максимум среди этих значений – это и будет ответ.

Асимптотика: $O(n \cdot m)$

Пример программы

Ниже представлено решение на языке Python3

```
1 from collections import namedtuple
2 from math import sqrt
3
4 Point = namedtuple("Point", ["x", "y"])
5
6
7 # расстояние между точками a и b
8 def dist(a, b):
9     return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y))
10
11
12 # расстояние между точкой c и отрезком ab
13 def dist2(a, b, c):
14     A = a.y - b.y
15     B = b.x - a.x
16     C = b.y * a.x - a.y * b.x
17     cc = B * c.x - A * c.y
18     d = A * A + B * B
19     p = Point((cc * B - C * A) / d, -(cc * A + C * B) / d)
20
21     return dist(c, p) if dist(a, p) + dist(p, b) - dist(a, b) < 1e-7
22         else min(dist(c, a), dist(c, b))
23
24
25 def solve(dataset):
26     ds = dataset.splitlines()
27
28     s = ds[0].split()
29     n = int(s[0])
30     m = int(s[1])
31
32     s = ds[1].split()
33     p = []
34     for i in range(n):
35         p.append(Point(int(s[i * 2]), int(s[i * 2 + 1])))
36
37     s = ds[2].split()
38     pm = []
39     for i in range(m):
40         pm.append(Point(int(s[i * 2]), int(s[i * 2 + 1])))
41
42     dist = [1e10] * m
43
44     for i in range(n - 1):
45         for j in range(m):
46             dist[j] = min(dist[j], dist2(p[i], p[i + 1], pm[j]))
```

```

47
48     ans = dist[m-1]
49     for i in range(m - 1):
50         ans = max(ans, dist[i])
51
52     return ans
53
54
55 print(solve(input() + '\n' + input() + '\n' + input()))

```

Задача 3.2.3. Радиус роботов-уборщиков (25 баллов)

Не успели вы закончить с предыдущей задачей, как Серёжа уже нарисовал какие-то ломаные на полу и сказал, что роботы должны двигаться по ним, не сталкиваясь между собой и при этом должны быть одинаковых размеров (по каждой из траекторий движется только один робот; касание не считается столкновением).

Рассчитайте максимальный радиус, при котором роботы не столкнутся ни при какой разнице времён запуска роботов.

Формат входных данных

В первой строке два целых числа:

- $2 \leq n \leq 500$ - количество роботов;
- $2 \leq m \leq 500$ - количество точек в траектории каждого робота.

При этом $4 \leq n \cdot m \leq 500$.

В следующих n строках задано по m точек ($2 \cdot m$ целых числа): $x_{i,j}, y_{i,j}$ — j -я точка траектории i -го робота ($1 \leq i \leq n; 1 \leq j \leq m; |x_{i,j}| \leq 10^9; |y_{i,j}| \leq 10^9$).

Никакие две последовательные точки никакой из траекторий не совпадают.

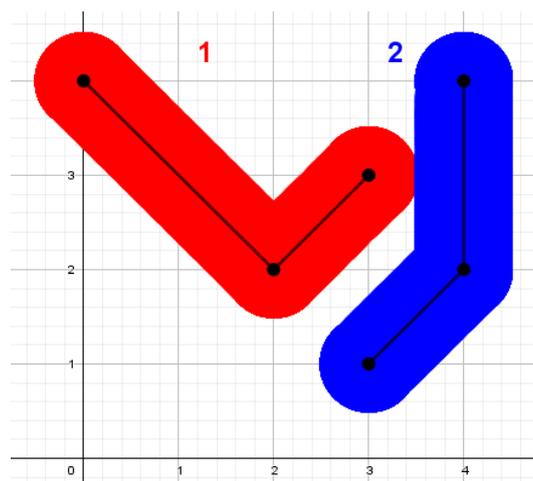


Рис. 1.3: Рисунок к первому примеру

Формат выходных данных

Если есть пересекающиеся траектории, выведите -1 , иначе одно неотрицательное число — максимальный радиус.

Ваш ответ будет засчитан, если его абсолютная или относительная ошибка не превосходит 10^{-6} . Формально, пусть ваш ответ равен a , а ответ жюри равен b . Ваш ответ будет засчитан, если $\frac{|a-b|}{\max(1,b)} \leq 10^{-6}$.

Примеры

Пример №1

Стандартный ввод
2 3 0 4 2 2 3 3 4 4 4 2 3 1
Стандартный вывод
0.5

Способ оценки работы

За решение задачи начислялось:

- **10 баллов**, если пройдена только первая группа тестов;
- **25 баллов**, если пройдены все тесты.

Для проверки результата использовался следующий код на языке Python3

```
def generate():
    return []

def check(reply, clue):
    a = float(reply)
    b = float(clue)
    return abs(a - b) / max(1.0, b) <= 1e-6

from collections import namedtuple
from math import sqrt

Point = namedtuple("Point", ["x", "y"])

# triangle area
def area(a, b, c):
    s = (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (c.x - a.x)
    return -1 if s < 0 else (1 if s > 0 else 0)

# bounding box
def box(a, b, c, d):
    return max(min(a, b), min(c, d)) <= min(max(a, b), max(c, d))

# пересечение отрезков
def intersect(a, b, c, d):
    return box(a.x, b.x, c.x, d.x) & box(a.y, b.y, c.y, d.y) \
        & (area(a, b, c) * area(a, b, d) <= 0) & (area(c, d, a) * area(c, d, b) <= 0)

# расстояние между точками a и b
def dist(a, b):
    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y))
```

```

# расстояние между точкой c и отрезком ab
def dist2(a, b, c):
    A = a.y - b.y
    B = b.x - a.x
    C = b.y * a.x - a.y * b.x
    cc = B * c.x - A * c.y
    d = A * A + B * B
    p = Point((cc * B - C * A) / d, -(cc * A + C * B) / d)

    return dist(c, p) if dist(a, p) + dist(p, b) - dist(a, b) < 1e-7 else min(dist(c, a), dist(c, b))

# расстояние между отрезками ab и cd
def dist3(a, b, c, d):
    return min(min(dist2(c, d, a), dist2(c, d, b)), min(dist2(a, b, c), dist2(a, b, d)))

def solve(dataset):
    ds = dataset.splitlines()

    s = ds[0].split()
    n = int(s[0])
    m = int(s[1])

    p = [[]]
    for i in range(n):
        s = ds[i + 1].split()
        p.append([])
        for j in range(m):
            p[i].append(Point(int(s[j * 2]), int(s[j * 2 + 1])))

    inter = False

    for i in range(n):
        for j in range(i + 1, n):
            for ii in range(m - 1):
                for jj in range(m - 1):
                    inter |= intersect(p[i][ii], p[i][ii + 1], p[j][jj], p[j][jj + 1])

    if inter:
        return str(-1)

    ans = 1e15

    for i in range(n):
        for j in range(i + 1, n):
            for ii in range(m - 1):
                for jj in range(m - 1):
                    ans = min(ans, dist3(p[i][ii], p[i][ii + 1], p[j][jj], p[j][jj + 1]))

    return str(ans / 2.0)

```

Решение

В данной задаче достаточно проверить есть ли пересекающиеся траектории (аналогично тому, как это делается в задаче *A*) и, если такие найдутся, вывести -1 или, если таких траекторий нет, найти два отрезка, которые принадлежат разным траекториям и расстояние между которыми минимально, и вывести поделенное на два расстояние между этими отрезками (тут пригодится решение задачи *B*).

Асимптотика: $O((n \cdot m)^2)$

Пример программы

Ниже представлено решение на языке Python3

```
1 from collections import namedtuple
2 from math import sqrt
3
4 Point = namedtuple("Point", ["x", "y"])
5
6
7 # triangle area
8 def area(a, b, c):
9     s = (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (c.x - a.x)
10    return -1 if s < 0 else (1 if s > 0 else 0)
11
12
13 # bounding box
14 def box(a, b, c, d):
15    return max(min(a, b), min(c, d)) <= min(max(a, b), max(c, d))
16
17
18 # пересечение отрезков
19 def intersect(a, b, c, d):
20    return box(a.x, b.x, c.x, d.x) & box(a.y, b.y, c.y, d.y) \
21           & (area(a, b, c) * area(a, b, d) <= 0) & (area(c, d, a) * area(c, d, b) <= 0)
22
23
24 # расстояние между точками a и b
25 def dist(a, b):
26    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y))
27
28
29 # расстояние между точкой c и отрезком ab
30 def dist2(a, b, c):
31    A = a.y - b.y
32    B = b.x - a.x
33    C = b.y * a.x - a.y * b.x
34    cc = B * c.x - A * c.y
35    d = A * A + B * B
36    p = Point((cc * B - C * A) / d, -(cc * A + C * B) / d)
37
38    return dist(c, p) if dist(a, p) + dist(p, b) - dist(a, b) < 1e-7
39           else min(dist(c, a), dist(c, b))
40
41
42 # расстояние между отрезками ab и cd
43 def dist3(a, b, c, d):
44    return min(min(dist2(c, d, a), dist2(c, d, b)), min(dist2(a, b, c), dist2(a, b, d)))
45
46
47 def solve(dataset):
48    ds = dataset.splitlines()
49
50    s = ds[0].split()
51    n = int(s[0])
52    m = int(s[1])
53
54    p = [[]]
55    for i in range(n):
56        s = ds[i + 1].split()
```

```

57     p.append([])
58     for j in range(m):
59         p[i].append(Point(int(s[j * 2]), int(s[j * 2 + 1])))
60
61     inter = False
62
63     for i in range(n):
64         for j in range(i + 1, n):
65             for ii in range(m - 1):
66                 for jj in range(m - 1):
67                     inter |= intersect(p[i][ii], p[i][ii + 1], p[j][jj], p[j][jj + 1])
68
69     if inter:
70         return -1
71
72     ans = 1e15
73
74     for i in range(n):
75         for j in range(i + 1, n):
76             for ii in range(m - 1):
77                 for jj in range(m - 1):
78                     ans = min(ans, dist3(p[i][ii], p[i][ii + 1], p[j][jj], p[j][jj + 1]))
79
80     return ans / 2.0
81
82
83 pds = input()
84 n = int(pds.split()[0])
85 for i in range(n):
86     pds += '\n'
87     pds += input()
88
89 print(solve(pds))

```

Задача 3.2.4. Уборка (15 баллов)

Определить возможность столкновения - задача довольно сложная, поэтому для начала вы договорились с Серёжей, что перепрограммируете роботов (зададите им новые траектории) и будете запускать их по очереди.

Происходит это следующим образом: очередного робота Серёжа ставит на первую точку траектории и запускает. После достижения конечной точки Серёжа моментально убирает робота и переходит к следующему.

Ваша задача определить каких роботов нужно запустить, чтобы убрать весь мусор. При этом нужно минимизировать количество запущенных роботов.

Допускается погрешность не более 10^{-6} при расчёте расстояния от центра робота в любой момент времени до любой из мусор-точек.

Формат входных данных

В первой строке три целых числа:

- $1 \leq n \leq 20$ - количество роботов;
- $2 \leq m \leq 500$ - количество точек в траектории каждого робота;
- $1 \leq k \leq 200$ - количество мусор-точек.

В следующих n строках по $2 \cdot m + 1$ целых чисел: $1 \leq radius_i \leq 10^9$ - радиус i -го робота и m точек ($2 \cdot m$ целых числа): $x_{i,j}, y_{i,j}$ - j -я точка траектории i -го робота ($1 \leq i \leq n; 1 \leq j \leq m; |x_{i,j}| \leq 10^9; |y_{i,j}| \leq 10^9$).

В последней строке заданы мусор-точки ($2 \cdot k$ целых чисел): x_i, y_i - i -я точка мусора ($1 \leq i \leq k; |x_i| \leq 10^9; |y_i| \leq 10^9$).

Никакие две последовательные точки никакой из траекторий не совпадают.

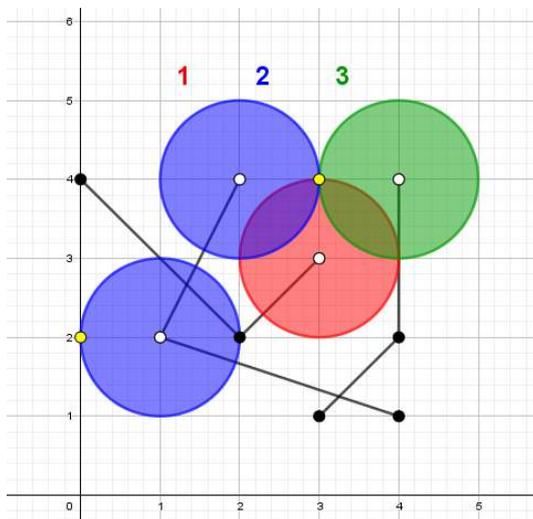


Рис. 1.4: Рисунок к первому примеру

Формат выходных данных

Если весь мусор невозможно убрать, выведите -1 , иначе одно положительное целое число - минимальное необходимое для уборки всего мусора количество роботов.

Примеры

Пример №1

Стандартный ввод
3 3 2
1 0 4 2 2 3 3
1 2 4 1 2 4 1
1 4 4 4 2 3 1
0 2 3 4
Стандартный вывод
1

Способ оценки работы

За решение задачи начислялось:

- **15 баллов**, если пройдены все тесты.

Для проверки результата использовался следующий код на языке Python3

```

def generate():
    return []

def check(reply, clue):
    return reply.strip() == clue.strip()

```

Решение

С помощью функции нахождения расстояния от точки до отрезка можно определить для каждого робота какие мусор-точки он может собрать.

Далее достаточно сделать полный перебор (битмасок) вариантов очереди запуска роботов и найти тот вариант, в котором используется минимум роботов и весь мусор убран.

Асимптотика: $O(n \cdot k \cdot (2^n + m))$

Пример программы

Ниже представлено решение на языке Java

```

1  import java.io.BufferedReader;
2  import java.io.InputStream;
3  import java.io.InputStreamReader;
4  import java.io.IOException;
5  import java.io.PrintWriter;
6  import java.util.StringTokenizer;
7
8  import static java.lang.Math.min;
9  import static java.lang.Math.sqrt;
10
11 public class Main {
12     // Уборка
13     private FastScanner in;
14     private PrintWriter out;
15
16     class Point {
17         double x, y;
18
19         Point(double x, double y) {
20             this.x = x;
21             this.y = y;
22         }
23     }
24
25     private double eps = 1e-7;
26
27     // расстояние между точками a и b
28     private double dist(Point a, Point b) {
29         return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
30     }
31
32     // расстояние между точкой c и отрезком ab
33     private double dist(Point a, Point b, Point c) {
34         double A = a.y - b.y, B = b.x - a.x, C = b.y * a.x - a.y * b.x;
35         double cc = B * c.x - A * c.y, d = A * A + B * B;
36         Point p = new Point((cc * B - C * A) / d, -(cc * A + C * B) / d);

```

```

37
38     return dist(a, p) + dist(p, b) - dist(a, b) < eps ?
39         dist(c, p) :
40         min(dist(c, a), dist(c, b));
41 }
42
43 // траектории и радиусы роботов
44 private int n, m; // количество роботов и количество точек в траектории каждого робота
45 private double[] radius; // радиусы роботов
46 private Point[] [] p; // все точки всех траекторий
47
48 // мусор
49 private int k;
50 private Point[] pk;
51
52 // инициализация
53 private void init() throws IOException {
54     n = in.nextInt();
55     m = in.nextInt();
56     k = in.nextInt();
57
58     radius = new double[n];
59     p = new Point[n][m];
60     pk = new Point[k];
61
62     for (int i = 0; i < n; i++) {
63         radius[i] = in.nextInt();
64
65         for (int j = 0; j < m; j++)
66             p[i][j] = new Point(in.nextInt(), in.nextInt());
67     }
68
69     for (int i = 0; i < k; i++)
70         pk[i] = new Point(in.nextInt(), in.nextInt());
71 }
72
73 // Основа решения
74 private void solve() throws IOException {
75     boolean[] [] can = new boolean[n][k];
76
77     for (int i = 0; i < n; i++)
78         for (int ki = 0; ki < k; ki++)
79             for (int j = 0; j + 1 < m && !can[i][ki]; j++)
80                 can[i][ki] = dist(p[i][j], p[i][j + 1], pk[ki]) - radius[i] < eps;
81
82     int full = 1 << n, ans = n + 1;
83     boolean ok;
84     for (int f = 1; f < full; f++) {
85         ok = true;
86         for (int ki = 0; ki < k && ok; ki++) {
87             ok = false;
88             for (int i = 0; i < n && !ok; i++)
89                 ok = ((1 << i) & f) > 0 && can[i][ki];
90         }
91         if (ok)
92             ans = min(ans, cnt(f));
93     }
94
95     out.println(ans > n ? -1 : ans);
96 }

```

```

97
98     private int cnt(int f) {
99         int cnt = 0;
100        for (char c : Integer.toBinaryString(f).toCharArray())
101            cnt += c == '1' ? 1 : 0;
102        return cnt;
103    }
104
105    class FastScanner {
106        StringTokenizer st;
107        BufferedReader br;
108
109        FastScanner(InputStream s) {
110            br = new BufferedReader(new InputStreamReader(s));
111        }
112
113        String next() throws IOException {
114            while (st == null || !st.hasMoreTokens())
115                st = new StringTokenizer(br.readLine());
116            return st.nextToken();
117        }
118
119        int nextInt() throws IOException {
120            return Integer.parseInt(next());
121        }
122    }
123
124    private void run() throws IOException {
125        in = new FastScanner(System.in);
126        out = new PrintWriter(System.out);
127
128        init();
129        solve();
130
131        out.flush();
132        out.close();
133    }
134
135    public static void main(String[] args) throws IOException {
136        new Main().run();
137    }
138 }

```

Задача 3.2.5. Столкновения (15 баллов)

Поскольку Серёжа не может перепрограммировать роботов, Серёжа решил переставить их так, чтобы они собрали больше мусора, тем самым всё же изменив траектории их движения. Сейчас он готовится запустить всех роботов и посмотреть, сколько мусора они уберут. Но вы-то знаете, что теперь эти роботы не только могут убрать мусор, но и врезаться друг в друга!

Вам нужно срочно указать Серёже какие пары роботов могут столкнуться, если их запустить одновременно.

После достижения конечной точки Серёжа моментально убирает робота.

Касание (расстояние меньше 10^{-6}) считайте столкновением.

Формат входных данных

В первой строке два целых числа:

- $2 \leq n \leq 100$ - количество роботов;
- $2 \leq m \leq 100$ - количество точек в траектории каждого робота.

В следующих n строках по $2 \cdot m + 2$ целых чисел: $1 \leq radius_i \leq 10^9$ - радиус i -го робота, $1 \leq speed_i \leq 10^9$ - скорость i -го робота и m точек $x_{i,j}, y_{i,j}$ - j -я точка траектории i -го робота ($1 \leq i \leq n; 1 \leq j \leq m; |x_{i,j}| \leq 10^9; |y_{i,j}| \leq 10^9$).

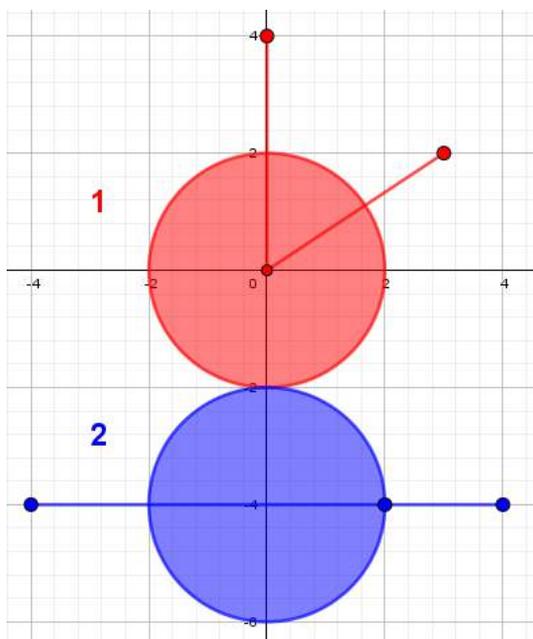


Рис. 1.5: Рисунок к первому примеру. Столкновение происходит примерно через 4 секунды после запуска.

Формат выходных данных

В первой строке одно целое число: k - число пар роботов, которые столкнутся, если запустить одновременно.

В следующих k строках по одной паре целых чисел: $i j$ - i -й робот столкнётся с j -м ($1 \leq i \leq n; 1 \leq j \leq n; i < j$)

Примеры

Пример №1

Стандартный ввод
2 3
2 1 0 4 0 0 3 2
2 1 -4 -4 2 -4 4 -4
Стандартный вывод
1
1 2

Способ оценки работы

За решение задачи начислялось:

- **15 баллов**, если пройдены все тесты.

Для проверки результата использовался следующий код на языке Python3

```
def generate():
    return []

def check(reply, clue):
    return reply.strip() == clue.strip()
```

Решение

Для решения данной задачи нужно написать проверку на столкновение двух одновременно запущенных роботов.

Можно заметить, что в промежутки времени между достижениями роботами точек роботы движутся равномерно и прямолинейно. А это значит, что с помощью тернарного поиска на каждом из таких промежутков времени, мы можем найти минимальное расстояние между центрами роботов, достижимое в некоторый текущий промежуток времени. Если хотя бы одно из этих минимальных расстояний не превышает суммы радиусов роботов, то столкновения не избежать.

Асимптотика: $O(n^2 \cdot m)$

Пример программы

Ниже представлено решение на языке Java

```
1  import java.io.BufferedReader;
2  import java.io.InputStream;
3  import java.io.InputStreamReader;
4  import java.io.IOException;
5  import java.io.PrintWriter;
6  import java.util.StringTokenizer;
7
8  import static java.lang.Math.min;
9  import static java.lang.Math.sqrt;
10
11 public class Main {
12     // Столкновения
13     private FastScanner in;
14     private PrintWriter out;
15
16     class Point {
17         double x, y;
18
19         Point(double x, double y) {
20             this.x = x;
21             this.y = y;
22         }
23     }
24 }
```

```

25 private double eps = 1e-7;
26
27 // расстояние между точками a и b
28 private double dist(Point a, Point b) {
29     return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
30 }
31
32 // точка, в которой находился бы робот, если бы
33 // прошёл расстояние s из точки a в направлении точки b
34 private Point goTo(Point a, Point b, double s) {
35     double S = dist(a, b);
36     return new Point(a.x + (b.x - a.x) / S * s, a.y + (b.y - a.y) / S * s);
37 }
38
39 // траектории и радиусы роботов
40 private int n, m; // количество роботов и количество точек в траектории каждого робота
41 private double[] radius, speed; // радиусы и скорости роботов
42 private Point[][] p; // все точки всех траекторий
43 private double[][] time;
44
45 // инициализация
46 private void init() throws IOException {
47     n = in.nextInt();
48     m = in.nextInt();
49
50     radius = new double[n];
51     speed = new double[n];
52     p = new Point[n][m];
53     time = new double[n][m];
54
55     for (int i = 0; i < n; i++) {
56         radius[i] = in.nextInt();
57         speed[i] = in.nextInt();
58
59         for (int j = 0; j < m; j++)
60             p[i][j] = new Point(in.nextInt(), in.nextInt());
61     }
62 }
63
64 // заполняем таблицу времени
65 private void fillTime() {
66     for (int i = 0; i < n; i++) {
67         time[i][0] = 0.0;
68
69         for (int j = 0; j + 1 < m; j++)
70             time[i][j + 1] = time[i][j] + dist(p[i][j], p[i][j + 1]) / speed[i];
71     }
72 }
73
74 // Основа решения
75 private void solve() throws IOException {
76     StringBuilder ans = new StringBuilder();
77     int cnt = 0;
78
79     for (int i = 0; i < n; i++)
80         for (int j = i + 1; j < n; j++)
81             if (crash(i, j)) {
82                 ans.append(i + 1).append(' ').append(j + 1).append('\n');
83                 cnt++;
84             }

```

```

85
86     out.print(cnt + "\n" + ans);
87 }
88
89 // проверяем на столкновение i-ого и j-ого роботов
90 private boolean crash(int i, int j) {
91     double pt = 0.0, nt;
92     double l, r, t;
93     double tl, tr, dl, dr;
94     Point pil, pir, pjl, pjr;
95
96     for (int ii = 1, jj = 1; ii < m && jj < m; pt = nt) {
97         nt = min(time[i][ii], time[j][jj]);
98
99         if (nt - pt > eps) {
100
101             // тернарный поиск
102             l = pt;
103             r = nt;
104             while (r - l > eps) {
105
106                 tl = l + (r - l) / 3;
107                 tr = tl + (r - l) / 3;
108
109                 pil = goTo(p[i][ii - 1], p[i][ii], (tl - time[i][ii - 1]) * speed[i]);
110                 pir = goTo(p[i][ii - 1], p[i][ii], (tr - time[i][ii - 1]) * speed[i]);
111
112                 pjl = goTo(p[j][jj - 1], p[j][jj], (tl - time[j][jj - 1]) * speed[j]);
113                 pjr = goTo(p[j][jj - 1], p[j][jj], (tr - time[j][jj - 1]) * speed[j]);
114
115                 dl = dist(pil, pjl);
116                 dr = dist(pir, pjr);
117
118                 if (dl > dr)
119                     l = tl;
120                 else
121                     r = tr;
122             }
123
124             t = (l + r) / 2.0;
125             pil = goTo(p[i][ii - 1], p[i][ii], (t - time[i][ii - 1]) * speed[i]);
126             pjl = goTo(p[j][jj - 1], p[j][jj], (t - time[j][jj - 1]) * speed[j]);
127
128             if (dist(pil, pjl) - radius[i] - radius[j] < eps)
129                 return true;
130         }
131
132         if (time[i][ii] - nt < eps)
133             ii++;
134         if (time[j][jj] - nt < eps)
135             jj++;
136     }
137
138     return false;
139 }
140
141 class FastScanner {
142     StringTokenizer st;
143     BufferedReader br;
144

```

```

145     FastScanner(InputStream s) {
146         br = new BufferedReader(new InputStreamReader(s));
147     }
148
149     String next() throws IOException {
150         while (st == null || !st.hasMoreTokens())
151             st = new StringTokenizer(br.readLine());
152         return st.nextToken();
153     }
154
155     int nextInt() throws IOException {
156         return Integer.parseInt(next());
157     }
158 }
159
160 private void run() throws IOException {
161     in = new FastScanner(System.in);
162     out = new PrintWriter(System.out);
163
164     init();
165     fillTime();
166     solve();
167
168     out.flush();
169     out.close();
170 }
171
172 public static void main(String[] args) throws IOException {
173     new Main().run();
174 }
175 }

```

Задача 3.2.6. Complete cleaning (10 баллов)

Возможно вам надоело решать вспомогательные задачи. Обрадуем: теперь вы готовы решить изначальную задачу.

У вас есть n роботов, они двигаются по траекториям состоящим из m точек каждая. Вам нужно убрать как можно больше мусора используя роботов, но вы очень сильно дорожите своими роботами. Поэтому вы не хотите, чтобы во время движения они столкнулись.

Таким образом, вам нужно вывести минимальное количество роботов, при котором можно убрать максимально возможное количество мусора. При этом выбранные роботы запускаются одновременно и не сталкиваются друг с другом.

После достижения конечной точки робот моментально исчезает.

Касание (расстояние меньше 10^{-6}) считайте столкновением.

Формат входных данных

В первой строке три целых числа:

- $1 \leq n \leq 20$ - количество роботов;
- $2 \leq m \leq 500$ - количество точек в траектории каждого робота;
- $1 \leq k \leq 200$ - количество мусор-точек.

В следующих n строках по $2 \cdot m + 2$ целых чисел: $1 \leq radius_i \leq 10^9$ - радиус i -го робота, $1 \leq speed_i \leq 10^9$ - скорость i -го робота и m точек $x_{i,j}, y_{i,j}$ - j -я точка траектории i -го робота ($1 \leq i \leq n; 1 \leq j \leq m; |x_{i,j}| \leq 10^9; |y_{i,j}| \leq 10^9$).

В последней строке заданы мусор-точки ($2 \cdot k$ целых чисел): $x_i y_i$ - i -я точка мусора ($1 \leq i \leq k; |x_i| \leq 10^9; |y_i| \leq 10^9$).

Формат выходных данных

Два неотрицательных числа - максимальное количество убранных мусор-точек и минимальное количество задействованных для этого роботов.

Примеры

Пример №1

Стандартный ввод
2 3 2 2 1 0 4 0 0 3 2 2 1 -4 -4 2 -4 4 -4 3 3 4 -6
Стандартный вывод
1 1

Пример №2

Стандартный ввод
2 3 2 2 1 0 4 0 0 3 2 2 1 -4 -4 2 -4 4 -4 0 -3 4 -6
Стандартный вывод
2 1

Способ оценки работы

За решение задачи начислялось:

- **10 баллов**, если пройдены все тесты.

Для проверки результата использовался следующий код на языке Python3

```
def generate():
    return []

def check(reply, clue):
    return reply.strip() == clue.strip()
```

Решение

Для решения последней задачи нужно оптимизировать фрагменты кода из прошлых задач.

Пример программы

Ниже представлено решение на языке Java

```
1  import java.io.BufferedReader;
2  import java.io.InputStream;
3  import java.io.InputStreamReader;
4  import java.io.IOException;
5  import java.io.PrintWriter;
6  import java.util.ArrayList;
7  import java.util.StringTokenizer;
8
9  import static java.lang.Math.min;
10 import static java.lang.Math.sqrt;
11
12 public class Main {
13     // Complete cleaning
14     private FastScanner in;
15     private PrintWriter out;
16
17     class Point {
18         double x, y;
19
20         Point(double x, double y) {
21             this.x = x;
22             this.y = y;
23         }
24     }
25
26     private double eps = 1e-7;
27
28     // расстояние между точками a и b
29     private double dist(Point a, Point b) {
30         return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
31     }
32
33     // расстояние между точкой c и отрезком ab
34     private double dist(Point a, Point b, Point c) {
35         double A = a.y - b.y, B = b.x - a.x, C = b.y * a.x - a.y * b.x;
36         double cc = B * c.x - A * c.y, d = A * A + B * B;
37         Point p = new Point((cc * B - C * A) / d, -(cc * A + C * B) / d);
38
39         return dist(a, p) + dist(p, b) - dist(a, b) < eps ?
40             dist(c, p) :
41             min(dist(c, a), dist(c, b));
42     }
43
44     // точка, в которой находился бы робот, если бы
45     // прошёл расстояние s из точки a в направлении точки b
46     private Point goTo(Point a, Point b, double s) {
47         double S = dist(a, b);
48         return new Point(a.x + (b.x - a.x) / S * s, a.y + (b.y - a.y) / S * s);
49     }
}
```

```

50
51 // траектории и радиусы роботов
52 private int n, m; // количество роботов и количество точек в траектории каждого робота
53 private double[] radius, speed; // радиусы и скорости роботов
54 private Point[] [] p; // все точки всех траекторий
55 private double[] [] time;
56
57 // мусор
58 private int k;
59 private Point[] pk;
60
61 // инициализация
62 private void init() throws IOException {
63     n = in.nextInt();
64     m = in.nextInt();
65     k = in.nextInt();
66
67     radius = new double[n];
68     speed = new double[n];
69     p = new Point[n][m];
70     pk = new Point[k];
71
72     for (int i = 0; i < n; i++) {
73         radius[i] = in.nextInt();
74         speed[i] = in.nextInt();
75
76         for (int j = 0; j < m; j++)
77             p[i][j] = new Point(in.nextInt(), in.nextInt());
78     }
79
80     for (int i = 0; i < k; i++)
81         pk[i] = new Point(in.nextInt(), in.nextInt());
82 }
83
84 // заполняем таблицу времени
85 private void fillTime() {
86     time = new double[n][m];
87
88     for (int i = 0; i < n; i++) {
89         time[i][0] = 0.0;
90
91         for (int j = 0; j + 1 < m; j++)
92             time[i][j + 1] = time[i][j] + dist(p[i][j], p[i][j + 1]) / speed[i];
93     }
94 }
95
96 // Основа решения
97 private void solve() throws IOException {
98
99     // создаём матрицу столкновений
100    boolean[] [] crash = new boolean[n][n];
101    for (int i = 0; i < n; i++)
102        for (int j = i + 1; j < n; j++)
103            crash[i][j] = crash[j][i] = crash(i, j);
104
105    // создаём матрицу для проверки захвата роботами мусора
106    boolean[] [] can = new boolean[n][k];
107    for (int i = 0; i < n; i++)
108        for (int ki = 0; ki < k; ki++)
109            for (int j = 0; j + 1 < m && !can[i][ki]; j++)

```

```

110         can[i][ki] = dist(p[i][j], p[i][j + 1], pk[ki]) - radius[i] < eps;
111
112     int full = 1 << n, ans = n;
113     boolean ok;
114     ArrayList<Integer> ids = new ArrayList<>();
115
116     int max = 0, min = 0;
117     // делаем полный перебор (битмасок) вариантов запуска роботов
118     for (int f = 1; f < full; f++) {
119         ids.clear();
120
121         // создаём список роботов, которые запускаются
122         for (int i = 0; i < n; i++)
123             if (((1 << i) & f) > 0)
124                 ids.add(i);
125
126         // проверяем роботов из списка на столкновения
127         ok = true;
128         for (int i : ids)
129             for (int j : ids)
130                 ok &= !crash[i][j];
131
132         // в случае столкновения, сразу переходим к следующему варианту
133         if (!ok)
134             continue;
135
136         // считаем количество убираемых мусор-точек
137         int cnt = 0;
138         for (int ki = 0; ki < k; ki++) {
139             ok = false;
140             for (int i : ids)
141                 if (can[i][ki]) {
142                     ok = true;
143                     break;
144                 }
145
146             if (ok)
147                 cnt++;
148         }
149
150         // улучшаем текущий ответ
151         if (cnt > max) {
152             max = cnt;
153             min = ids.size();
154         } else if (cnt == max)
155             if (ids.size() < min)
156                 min = ids.size();
157     }
158
159     // выводим ответ: max - количество мусора, min - кол-во запущенных роботов
160     out.println(max + " " + min);
161 }
162
163 // проверяем на столкновение i-ого и j-ого роботов
164 private boolean crash(int i, int j) {
165     double pt = 0.0, nt;
166     double l, r, t;
167     double tl, tr, dl, dr;
168     Point pil, pir, pj1, pj2;
169

```

```

170     for (int ii = 1, jj = 1; ii < m && jj < m; pt = nt) {
171         nt = min(time[i][ii], time[j][jj]);
172
173         if (nt - pt > eps) {
174
175             // тернарный поиск
176             l = pt;
177             r = nt;
178             while (r - l > eps) {
179
180                 tl = l + (r - l) / 3;
181                 tr = tl + (r - l) / 3;
182
183                 pil = goTo(p[i][ii - 1], p[i][ii], (tl - time[i][ii - 1]) * speed[i]);
184                 pir = goTo(p[i][ii - 1], p[i][ii], (tr - time[i][ii - 1]) * speed[i]);
185
186                 pj1 = goTo(p[j][jj - 1], p[j][jj], (tl - time[j][jj - 1]) * speed[j]);
187                 pj2 = goTo(p[j][jj - 1], p[j][jj], (tr - time[j][jj - 1]) * speed[j]);
188
189                 dl = dist(pil, pj1);
190                 dr = dist(pir, pj2);
191
192                 if (dl > dr)
193                     l = tl;
194                 else
195                     r = tr;
196             }
197
198             t = (l + r) / 2.0;
199             pil = goTo(p[i][ii - 1], p[i][ii], (t - time[i][ii - 1]) * speed[i]);
200             pj1 = goTo(p[j][jj - 1], p[j][jj], (t - time[j][jj - 1]) * speed[j]);
201
202             if (dist(pil, pj1) - radius[i] - radius[j] < eps)
203                 return true;
204         }
205         if (time[i][ii] - nt < eps)
206             ii++;
207         if (time[j][jj] - nt < eps)
208             jj++;
209     }
210
211     return false;
212 }
213
214 class FastScanner {
215     StringTokenizer st;
216     BufferedReader br;
217
218     FastScanner(InputStream s) {
219         br = new BufferedReader(new InputStreamReader(s));
220     }
221
222     String next() throws IOException {
223         while (st == null || !st.hasMoreTokens())
224             st = new StringTokenizer(br.readLine());
225         return st.nextToken();
226     }
227
228     int nextInt() throws IOException {
229         return Integer.parseInt(next());

```