§3. Заключительный этап: индивидуальная часть

Заключительный этап олимпиады состоит из двух частей: индивидуальное решение задач по предметам (физика, информатика) и командное решение инженерное задачи. На индивидуальное решение задач дается по 2 часа на один предмет. Решение каждой задачи дает определенное количество баллов (см. критерии оценки). По физике за каждую задачу можно получить от 0 до указанного количества баллов. Баллы по информатике зачисляются в полном объеме за правильное решение задачи. Решение задачи по информатике подразумевает написание программы на языке C/C#, Java, Python или C++. Участники получают оценку за решение задач в совокупности по всем предметам данного профиля (физика и информатика) — суммарно от 0 до 150 баллов.

3.1 Задачи по физике (120 мин.)

На выполнение всех задач отводится 2 часа. Задачи имеют разную сложность и, в целом, их число избыточно для такого времени, однако достаточно подробные критерии оценки дают возможность дифференцировано оценить работы участников.

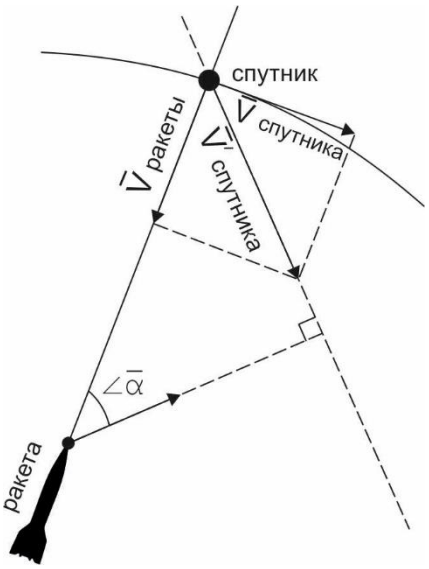
3.2 Задачи по физике (9 класс)

Задача 3.2.1 (20 баллов)

Условие:

Спутник движется по геостационарной орбите. Ракета, летящая с Земли к спутнику, находится на расстоянии в 20 километров от спутника. При этом сама ракета находится на прямой соединяющей спутник и центр Земли, а ее скорость направлена точно на спутник и равна 1.77 км/с. На каком минимальном расстоянии пройдет ракета от спутника, если скорость ее больше не меняется. *Можно считать, что ускорение ракеты - это итоговое ускорение, которое создает равнодействующая сила. Высота геостационарной орбиты 35 786 км над уровнем моря.*

Решение:



Рассчитаем линейную скорость спутника на геостационарной орбите.

$$V = \frac{2\pi R}{T} = 3{,}06\text{км/с}.$$

За время полета ракеты спутник пройдет расстояние порядка десятков километров, что много меньше радиуса его орбиты. Таким образом на интересующем нас масштабе движение спутника можно считать прямолинейными.

Тогда для решения задачи достаточно перейти в систему отсчета, связанную с ракетой. Минимальное расстояние между ракетой и спутником будет равно длине перпендикуляра, опущенного на прямую, по которой движется спутник в этой системе отсчёта.

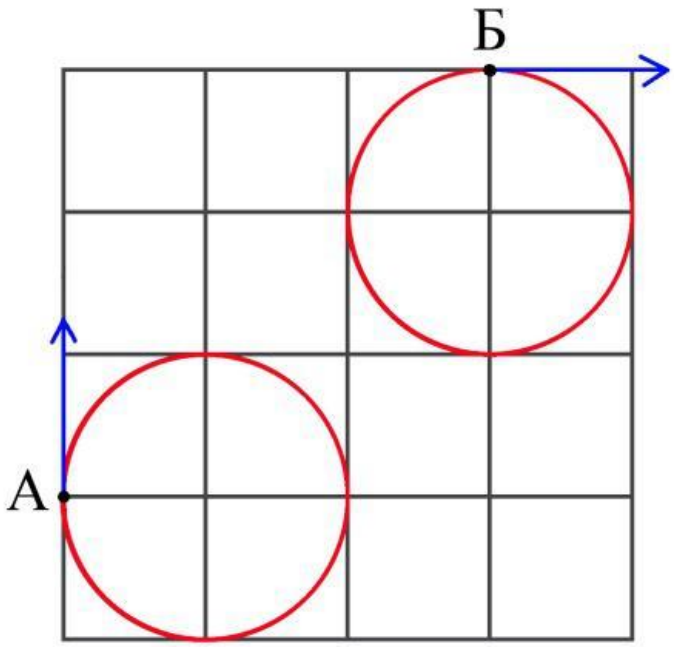
$S = H\cos\alpha = H\frac{V_c}{(\sqrt{V_c^2 + V_p^2})} \approx$

Ответ:

$S = H\cos\alpha = H\frac{V_c}{(\sqrt{V_c^2 + V_p^2})} \approx$ 15.5км

Критерии:

- Сделано предположениеи о прямолинейном приближении для решения задачи - 4 балла.
- Найдена скорость движения спутника - 4 балла.
- Сделано предположение о рациональном выборе системы отсчета для решения задачи - 4 балла.
- Получено выражение для расстояния - 4 балла.
- Получен правильный ответ - 4 балла.
- Решения удовлетворяющего критериям, приведенным выше, нет, но приведены разумные рассуждения, направленные на решение задачи - 2 балла.



Задача 3.2.2 (32 балла)

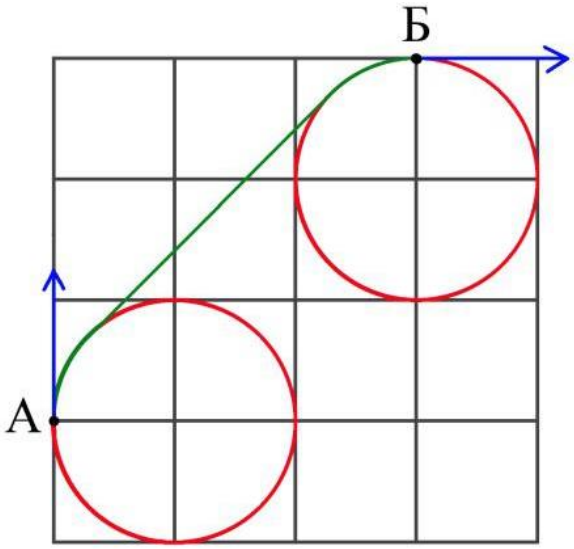
Условие:

Испытания марсохода на Земле показали, что минимальное время разворота по дуге окружности радиуса R_0 = 10м при движении по твердому грунту составляет T = 12с. За какое минимальное время марсоход сможет объехать систему кратеров на Марсе, двигаясь с неизменяющейся по модулю во время движения скоростью, начав двигаться из положения А в направлении на север (вверх на рисунке) и закончив в точке Б в положении на восток (вправо на рисунке)? Грунт в районе кратеров практически такой же, как при испытаниях на Земле. Средняя плотность Марса 0.714 плотности Земли, радиус Марса $R_м$ = 3400км. Радиус каждого из кратеров R = 9м.

Решение:

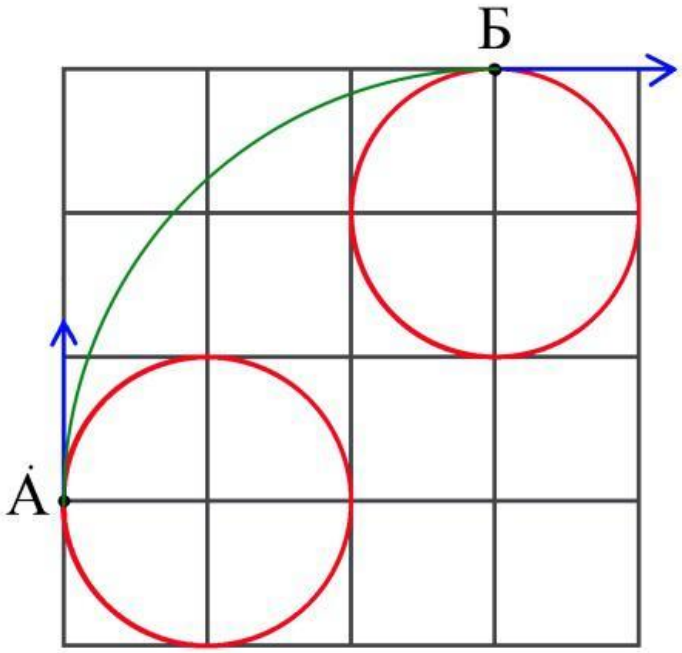
Для того, чтобы объехать кратеры марсоходу нужно повернуть на 90˚. Для поворота марсоходу необходимо будет двигаться по дуге окружности и именно это будет лимитировать максимальную скорость движения. Из второго закона Ньютона следует, что максимальная скорость движения по окружности определяется соотношением:

$V_{макс} = \sqrt{\mu g R}$, соответственно, чем больше радиус окружности, тем больше возможная



скорость. При этом растет и время прохождения дуги окружности: $\tau = \frac{2\pi R}{\sqrt{\mu g R}} = 2\pi\sqrt{\frac{R}{\mu g}}$. Кратчайшее расстояние, по которому можно пройти, повернув при этом на 90˚ равно $L_1 = 2R+\pi R/2$ Это движение соответствует траектории, изображенной на рисунке. Остальные траектории, содержащие участок прямой будут иметь большую длину либо не позволят повернуть на 90˚ оказавшись в заданных точках (т.к. увеличится либо радиус одной из дуг либо длина прямолинейной части).

В этом случае время буедт равно $\tau_1 = \frac{2R + \frac{\pi}{2}R}{\sqrt{\mu g R}} = 3.57\sqrt{\frac{R}{\mu g}}$



Минимальным будет время, при движении по окружности радиусом 3R. $\tau_2 = \frac{\pi \cdot 3R}{2\sqrt{\mu g \cdot 3R}} = \frac{\sqrt{3}\pi}{2}\sqrt{\frac{R}{\mu g}} = 2.72\sqrt{\frac{R}{\mu g}}$. Движение по другой траектории потребует использовать дугу окружности с меньшим радиусом, что приведет к снижению общей скорости, при возрастании расстояния.

Ускорение свободного падения на Марсе можно выразить через ускорение свободного падения на Земле: $g = g_0 \frac{\rho_м}{\rho_з} \cdot \frac{R_м}{R_з} = 3{,}72 м/с^2$

Осталось найти μ. Это можно сделать из условия испытания на Земле, считая, что одинаковые грунты имеют одинаковый коэффициент трения.

$\mu = \frac{4\pi^2 R_0}{g_0 T^2}$ =0.279, где g_0 - ускорение свободного падения на Земле.

Тогда искомое время равно $\tau_2 = 2.72\sqrt{\frac{R}{0.279g}} =$ 8с.

Ответ:

$\tau_2 = 2.72\sqrt{\frac{R}{0.279g}} =$ 8с.

Критерии:

- Сделано предположение о зависимости максимальной скорости от радиуса поворота. - 4 балла
- Сделано предположение об оптимальном маршруте - 4 балла.
- Приведено обоснование оптимальности маршрута - 12 баллов.
- Найден коэффициент трения (в том числе и неявно, без прямого вычисления) - 4 балла

- Найдено ускорение свободного падения на Марсе (в том числе и неявно, без прямого вычисления) - 4 балла.
- Найдено минимальное время поворота - 4 балла.
- Если минимальное время не найдено, но предложен какой-либо маршрут и для него рассчитано время - 4 балла.
- Решения удовлетворяющего критериям, приведенным выше, нет, но приведены разумные рассуждения, направленные на решение задачи - 2 балла.

Задача 3.2.3 (12 баллов)

Условие:

В процессе исследования электросхемы автономного аппарата проводились стресс тесты на длительную эксплуатацию. К аккумуляторной батарее аппарата было параллельно подключено два дублирующихся (одинаковых) прибора. Амперметр, стоящий перед батареей первоначально показал, что сила тока через батарею, при включении любого из приборов была расчетной I_0, но через продолжительное время при включении одного из приборов изменилась до 16,66% от I_0, а при переключении на второй оставалась первоначальной I_0. Инженерами было предположено, что такое изменение силы тока возможно из-за возникновения паразитного сопротивления в контактах одного из приборов. Оцените величину этого паразитного сопротивления, если сопротивление прибора R = 10 Ом, а источник питания идеальный.

Решение:

Поскольку источник идеальный, его сопротивление будем считать равным нулю. Запишем расчетную силу тока: $I_0 = \frac{U}{R}$. Поскольку сила тока изменилась только при включении одного из приборов, можно считать, что сопротивление второго осталось неизменным. Тогда ток $I_1 = \frac{U}{R + R_п}$, разделив вторую величину на первую, избавимся от сопротивления. Тогда: $\frac{I_0}{I_1} = \frac{R + R_п}{R}$, откуда $R_п = R(\frac{I_0}{I_1} - 1) =$ 50 Ом.

Ответ:

$R_п = R(\frac{I_0}{I_1} - 1) =$ 50 Ом

Критерии:

- Сделано предположение об устройстве электрической схемы, описывающей задачу, или нарисована схема - 2 балла.
- Правильно записаны законы Ома для токов в первом и втором случаях - 4 балла.
- Получена формула для правильного ответа – 4 балла.
- Получен правильный ответ - 2 балла.
- Решения удовлетворяющего критериям, приведенным выше, нет, но приведены разумные рассуждения, направленные на решение задачи - 2 балла.

Задача 3.2.4 (36 баллов)

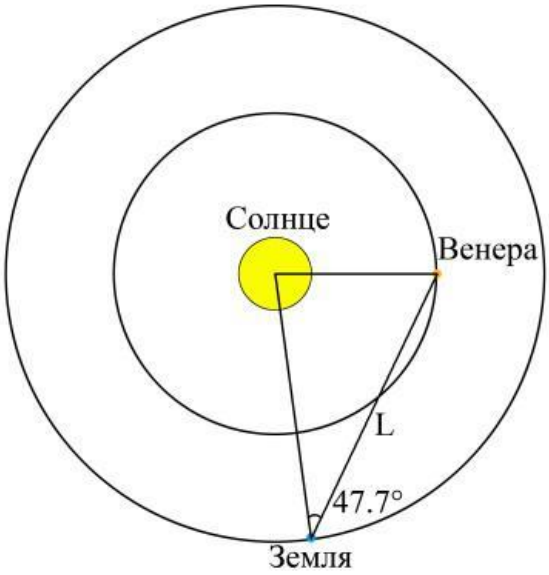
Условие:

Венера лучше всего видна на небе в моменты, когда угловое расстояние между ней и Солнцем максимально и составляет примерно 47,7˚ (моменты элонгации). Ее видимая звездная величина при этом составляет $m_в$ = -4,8. В этот же момент звездная величина изучаемого спутника составила $m_с$ = 0.8. Оцените эффективную площадь отражения спутника, если орбита спутника 50км.

Известно, что по разности видимых звездных величин объектов можно определить отношение освещенностей от этих объектов, т.е. энергий приходящих от объекта на единичную площадку. $\frac{L_2}{L_1} = 2.512^{m_1 - m_2}$.

Радиус орбиты Земли $r_з$ = 147.18 млн км, радиус орбиты в момент наблюдения Венеры r_v = 108.86 млн км., радиус Венеры - R_v = 6051км, сферическое альбедо Венеры 0,9. Сферическое альбедо – отношение светового потока отраженного во все стороны, к падающему на тело световому потоку.

Решение:



Мощность освещения падающего на площадку зависит от того сколько света от Солнца отразит объект и сколько от этого света дойдет до площадки.

Можно записать, что $L = \frac{\alpha P * S_{объекта}}{4\pi R^2} \cdot \frac{1}{4\pi d^2}$, где P - полная мощность излучения Солнца, R - расстояние от Солнца до отражающего объекта, d - расстояние от отражающего объекта до площадки, α -альбедо объекта.

Тогда отношение освещенностей от Венеры и от спутника можно записать, как:

$$\frac{L_v}{Ls} = \frac{\frac{\alpha P * 4\pi r_v^2}{4\pi R_v^2} \cdot \frac{1}{4\pi L^2}}{\frac{P * S_{спутника}}{4\pi R_{земли}^2} \cdot \frac{1}{4\pi h^2}} = \frac{4\pi\alpha \cdot R_{земли}^2 r_v^2 h^2}{R_v^2 S_{спутника} L^2}$$

$$= 2.512^{0.8+4.8} = 2.512^{5,6} = 173$$

Осталось найти L - расстояние между Солнцем и Венерой. Из рисунка видно, что для этого достаточно воспользоваться теоремой косинусов.

$R_в^2 = R_{земли}^2 + L^2 - 2LR_{земли}\cos\varphi$, решив квадратное уравнение, получим L. Вообще говоря, возможны два значения: L_1=99.56 млн км и L_2=98.42 млн км, но для дальнешего рассуждения можно взять среднее значение L = 99 млн км.

Тогда искомая площадь:

$$S_{спутника} = \frac{4\pi\alpha \cdot R_{земли}^2 r_в^2 h^2}{2^5 R_в^2 L^2} = 4{,}85 м^2$$

Ответ:

$$S_{спутника} = \frac{4\pi\alpha \cdot R_{земли}^2 r_в^2 h^2}{2^5 R_в^2 L^2} = 4{,}85 м^2$$

(от 4.79 до 4.89 $м^2$, если подставлять полученные значения для L)

Критерии:

- Записано выражение для мощности света отраженного объектом - 4 балла
- Записано выражение для освещенности площадки отраженным светом - 8 баллов.
- Учтено альбедо для Венеры - 4 балла.
- Поставлена геометрическая задача для поиска расстояние между Землей и Венерой – 4 балла.
- Найдено расстояние между Землей и Венерой - 4 балла.
- Получено два ответа для расстояния – 3 балла.
- Результат интерпретирован и правильно выбран один из ответов – 1 балл.
- Получен правильный ответ - 8 баллов.
- Решения удовлетворяющего критериям, приведенным выше, нет, но приведены разумные рассуждения, направленные на решение задачи - 2 балла.

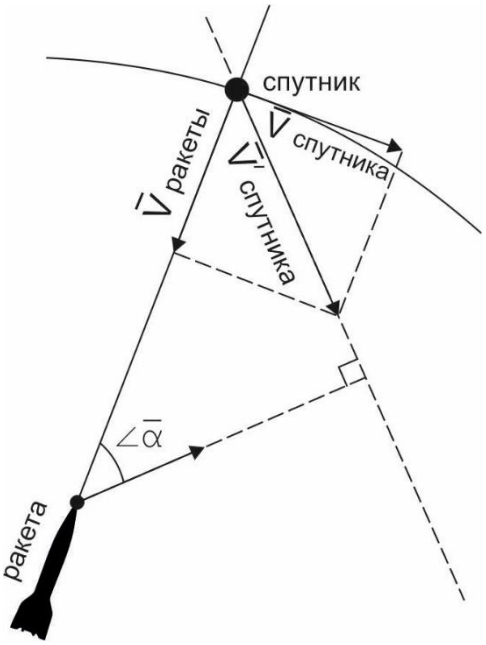
3.3 Задачи по физике 11 класс

Задача 3.3.1 (20 баллов)

Условие:

Спутник движется по геостационарной орбите. Ракета, летящая с Земли к спутнику, находится на расстоянии в 20 километров от спутника. При этом сама ракета находится на прямой соединяющей спутник и центр Земли, а ее скорость направлена точно на спутник и равна 1.77 км/с. На каком минимальном расстоянии пройдет ракета от спутника, если скорость ее больше не меняется. *Высота геостационарной орбиты 35 786 км над уровнем моря.*

Решение:



Рассчитаем линейную скорость спутника на геостационарной орбите. $V = \frac{2\pi R}{T}$ = 3,06км/с.
За время полета ракеты спутник пройдет расстояние порядка десятков километров, что много меньше радиуса его орбиты. Таким образом на интересующем нас масштабе движение спутника можно считать прямолинейными.

Тогда для решения задачи достаточно перейти в систему отсчета, связанную с ракетой. Минимальное расстояние между ракетой и спутником будет равно длине перпендикуляра, опущенного на прямую, по которой движется спутник в этой системе отсчёта.

$S = H\cos\alpha = H\frac{V_c}{(\sqrt{V_c^2 + V_p^2})} \approx$ 15.5км

Ответ:

$S = H\cos\alpha = H\frac{V_c}{(\sqrt{V_c^2 + V_p^2})} \approx$ 15.5км

Критерии:

- Сделано предположение о прямолинейном приближении для решения задачи - 4 балла.
- Найдена скорость движения спутника - 4 балла.
- Сделано предположение о рациональном выборе системы отсчета для решения задачи - 4 балла.
- Получено выражение для расстояния - 4 балла.
- Получен правильный ответ - 4 балла.
- Решения удовлетворяющего критериям, приведенным выше, нет, но приведены разумные рассуждения, направленные на решение задачи - 2 балла.

Задача 3.3.2 (36 баллов)

Условие:

Тело, размерами которого можно пренебречь, съехало с горки и попало на мертвую петлю в нижней ее точке, после чего прошло по дуге расстояние L = 11.78м и сорвалось. В какой точки дуги приземлилось тело, если радиус мертвой петли R = 5м?

Решение:

При внимательном анализе условия можно увидеть, что L/R=2.356=3/4π, т.е. тело срывается в точке находящейся под углом α = 45˚ от центра окружности.

Запишем, с помощью второго закона Ньютона величину скорости в момент отрыва:

$\frac{mV^2}{R} = mg\sin\alpha$, откуда $V = \sqrt{gR\sin\alpha}$. Направлена эта скорость по касательной к окружности, т.е. под углом 45˚ к горизонту.

Введем систему координат с центром в центре окружности. Тело срывается в точке $(R\frac{\sqrt{2}}{2}, R\frac{\sqrt{2}}{2})$

Уравнение параболы для полета тела будет выглядеть следующим образом:

$$y(x) = -\frac{\sqrt{2}}{R}x^2 - x + \frac{\sqrt{2}}{2}R$$

При этом уравнение мертвой петли:

$$y^2 = R^2 - x^2$$

Формальным решением будет решение уравнения 4 степени для нахождения координат пересечения параболы и окружности.

$$\left(-\frac{\sqrt{2}}{R}x^2 - x + \frac{\sqrt{2}}{2}R\right)^2 = R^2 - x^2$$

$$\frac{2}{R^2}x^4 + \frac{2\sqrt{2}}{R}x^3 - x^2 - \sqrt{2}Rx + \frac{R^2}{2} = R^2 - x^2$$

$$\frac{2}{R^2}x^4 + \frac{2\sqrt{2}}{R}x^3 - \sqrt{2}Rx - \frac{R^2}{2} = 0$$

$$x^4 + \sqrt{2}Rx^3 - \frac{\sqrt{2}}{2}R^3x - \frac{R^4}{4} = 0$$

Просто так решить такое уравнение достаточно сложно, однако, есть два важных соображения.

Во-первых, можно заметить, что уравнение не содержит квадратичного члена и симметрично относительно него.

Во-вторых, один из корней этого уравнения мы знаем: $x_1 = \frac{\sqrt{2}}{2}R$. Можно увидеть, что

$x_2 = -\frac{\sqrt{2}}{2}R$ так же является корнем этого уравнения.

Подставив найденный корень в уравнение параболы, найдем и вторую координату пересечения: $y_2 = -\frac{\sqrt{2}}{2}R$.

Довольно интересно, что это диаметрально противоположная точка окружности. При этом ответ не зависит от ускорения свободного падения. Необходимо только, чтобы это ускорение было и было однородным.

Ответ:

Диаметрально противоположная точка, к точке отрыва, расположенная под углом 45˚ внизу мертвой петли.

Критерии:

- Найдена точка отрыва - 6 баллов.
- Получено уравнение для скорости в точке отрыва - 4 балла.
- Получено уравнение траектории тела после отрыва - 6 баллов.
- Написано условие столкновения тела с мертвой петлей после отрыва - 8 баллов.
- Из этого условия найдено решение – 8 баллов.
- Приведен и интерпретирован правильный ответ – 4 балла
- Решения удовлетворяющего критериям, приведенным выше, нет, но приведены разумные рассуждения направленные на решение задачи - 2 балла.

Задача 3.3.3 (20 баллов)

Условие:

В процессе исследования электросхемы марсохода проводились стресс тесты на длительную эксплуатацию. К неидеальной аккумуляторной батарее марсохода было параллельно подключено два дублирующихся (одинаковых) прибора. Амперметр, стоящий перед батареей первоначально показал, что сила тока через батарею, при включении любого прибора была расчетной, но через продолжительное время при включении одного из приборов изменилась до 18% от расчетной, а при переключении на второй осталась исходной. При включении одновременно двух приборов сила тока стала превышать расчетную (для одного прибора) на 14.9% . Инженерами было предположено, что такое изменение силы тока возможно из-за возникновения паразитного сопротивления в контактах одного из приборов. Оцените величину этого паразитного сопротивления, если сопротивление прибора R = 10 Ом.

Решение:

Запишем расчетную силу тока: $I_0 = \frac{U}{R + r}$, где r - сопротивление аккумуляторной батареи.

Запишем силу тока при включении неисправной цепи: $I_1 = \frac{U}{R + R_{пр} + r}$

Запишем силу тока при включении сразу двух приборов: $I_2 = \frac{U}{\frac{(R + R_{пр})R}{(2R + R_{пр})} + r}$

Преобразуя эти уравнения запишем:

$$\begin{cases} \frac{U}{I_2} - \frac{U}{I_0} = \frac{R(R + R_{пр})}{R_{пр} + 2R} - R \\ \frac{U}{I_1} - \frac{U}{I_0} = R_{пр} \end{cases}$$

Разделив первое уравнение на второе и обозначив как $\alpha = \frac{\frac{I_0}{I_2} - 1}{\frac{I_0}{I_1} - 1}$, получим квадратное уравнение на искомую величину:

$R = -R + R\sqrt{1 - \frac{2}{\alpha}} =$ 50 Ом. В данном случае, отрицательный корень мы отбросили, как не имеющий физического смысла.

Ответ:

$R = -R + R\sqrt{1 - \frac{2}{\alpha}} =$ 50 Ом

Критерии:

- Сделано предположение об устройстве электрической схемы, описывающей задачу или нарисована схема - 4 балла.
- Правильно записаны законы Ома для токов - 4 балла.
- Получено выражение описывающее правильный ответ - 8 баллов.
- Получен правильный числовой ответ – 4 балла
- Решения удовлетворяющего критериям, приведенным выше, нет, но приведены разумные рассуждения, направленные на решение задачи - 2 балла.

Задача 3.3.4 (24 балла)

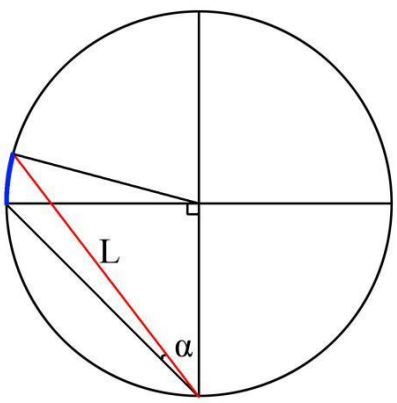
Условие:

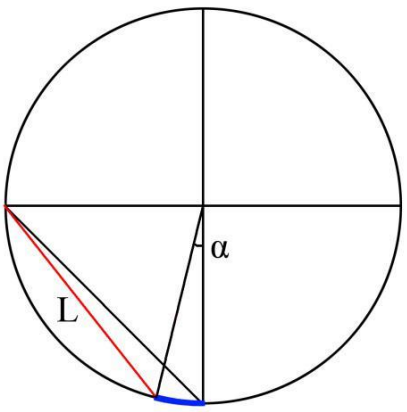
Вокруг Земли по одной геостационарной орбите движутся четыре спутника так, что они всегда образуют вершины квадрата. Один из спутников одновременно пересылает два одинаковых сигнала на другой спутник, расположенный по диагонали от него: первый сигнал через спутник, находящийся по направлению орбитального движения, а второй через спутник находящийся против направления орбитального движения. Спутники передают принятый сигнал без изменений. Оцените задержку времени между приемами этих сигналов, пренебрегая временем пересылки сигнала спутниками. Нужно ли учитывать эту задержку в расчетах или можно считать, что сигналы приходят одновременно? *Высота геостационарной орбиты 35 786 км над уровнем моря.*

Решение:

Задержка времени возникает из-за движения спутников вокруг Земли.
Для оценки будем считать, что расстояние, которое пройдут спутники за время передачи сигнала много меньше радиуса орбиты.

Обозначим угол $\alpha = \frac{Vt}{R}$





Воспользовавшись теоремой синусов для треугольников приведенных на рисунке, получим $\frac{R}{\sin(\frac{\pi}{4}+\frac{\alpha}{2})}=\frac{L_1}{\sin(\frac{\pi}{2}-\alpha)}$ для одного луча и $\frac{R}{\sin(\frac{\pi}{4}-\frac{\alpha}{2})}=\frac{L_2}{\sin(\frac{\pi}{2}+\alpha)}$ для второго.

Тогда, учитывая, что угол $\alpha=\frac{Vt}{R}$ << 1, можно упросить:

$\begin{cases}\frac{\sqrt{2}R}{(1+\frac{\alpha}{2})}=L_1\\ \frac{\sqrt{2}R}{(1-\frac{\alpha}{2})}=L_2\end{cases}$, что в свою очередь можно приблизить как:

$\begin{cases}L_1=\sqrt{2}R(1+\frac{\alpha}{2})\\ L_2=\sqrt{2}R(1-\frac{\alpha}{2})\end{cases}$, теперь учитывая, что $\alpha=\frac{Vt}{R}$ (нужно учесть, что в каждом случае t= $\frac{L}{c}$ разное, получим:

$\begin{cases}L_1=\sqrt{2}R(1+\frac{VL_1}{2cR})\\ L_2=\sqrt{2}R(1-\frac{VL_2}{2cR})\end{cases}$, после преобразования:

$\begin{cases}L_1=\frac{\sqrt{2}R}{(1-\frac{V}{2c})}\approx\sqrt{2}R(1+\frac{V}{2c})\\ L_2=\frac{\sqrt{2}R}{(1+\frac{V}{2c})}\approx\sqrt{2}R(1-\frac{V}{2c})\end{cases}$

Откуда:

$\Delta L=\sqrt{2}R\frac{V}{c}$, теперь учтем, что каждый луч проходит такое расстояние дважды, соответственно искомое время задержки:

$\Delta\tau=2\sqrt{2}R\frac{V}{c^2}$

Скорость спутника на геостационарной орбите легко оценить, считая период обращения равным периоду обращения Земли:

$V=\frac{2\pi R}{T}=$ 3.1км/с

Тогда искомое время: $\Delta\tau=\frac{4\pi\sqrt{2}R^2}{c^2T}\approx4\cdot10^{-8}c$, легко можно увидеть, что по сравнению со временем, за которое дойдет сигнал: $\tau=\frac{2\sqrt{2}R}{c}\approx0{,}04c$ эта величина крайне мала, однако даже она может иметь значение в особенно чувствительных экспериментах.

Ответ:

$\Delta\tau=\frac{4\pi\sqrt{2}R}{c^2T}\approx4\cdot10^{-8}c$, задержка составляет одну десятитысячную процента от времени, за которое проходит сигнал и может играть роль только в достаточно точных экспериментах.

Критерии:

- Приведена физическая причина возникновения разницы в задержке времени прохождения сигнала – 4 балла.
- Записана формулировка геометрической задачи, решение которой необходимо для решения задачи – 4 баллов.
-
- Сделано предположение о наличии малого параметра - 4 балла.
-
- Задача решена с разложением до первой степени по малому параметру - 8 баллов.
-
- Разница во времени соотнесена со временем задержки и приведено разумное объяснение того важно ли учитывать эту задержку - 4 балла.
-
- Решения удовлетворяющего критериям, приведенным выше, нет, но приведены разумные рассуждения направленные на решение задачи - 2 балла.

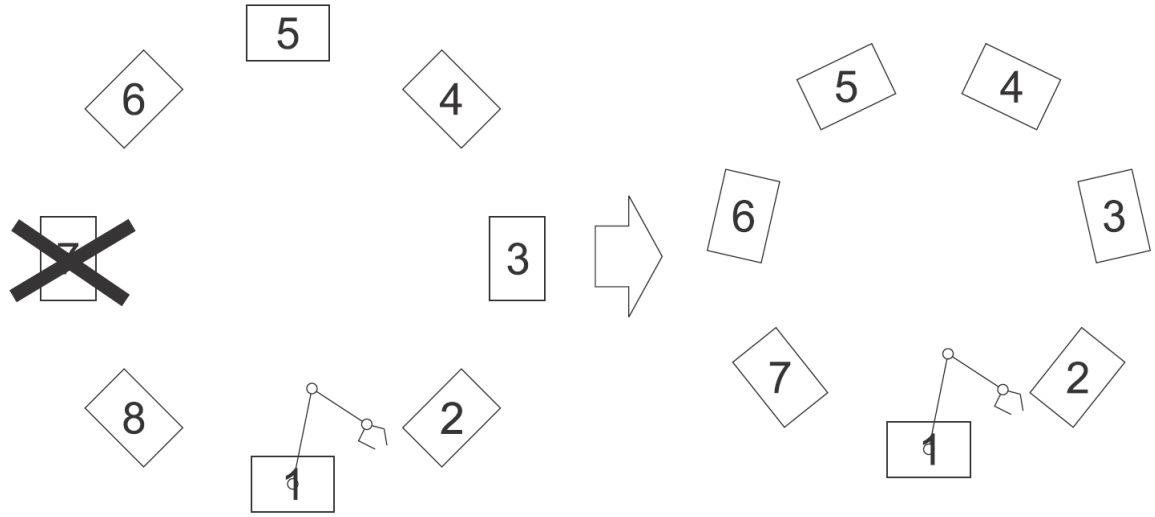
3.4 Задачи по информатике (120 мин.)

Задача 3.4.1 (6 баллов)

Условие:

В цехе по упаковке конечной продукции на заводе по производству мебели робот-манипулятор подготавливает товар к отгрузке. Контейнеры с элементами, из которых упаковывается та или иная единица мебели, расположены так, что робот вместе с ними образует круг. Каждая позиция, где может быть расположен контейнер, пронумерована против часовой стрелки. Манипулятор установлен в позиции 1, справа от него находится контейнер с позицией 2, слева — контейнер с позицией n.

В каждом i-ом контейнере находится ci определенного типа элементов. Во время операций по отгрузке робот может доставать из контейнеров по одному элементу. Длины звеньев манипулятора хватает только, чтобы достичь k-го контейнера по правую или по левую сторону от робота. Если во время операций по отгрузке какой-то контейнер становится пустым, он отправляется в производственных цех, а круг контейнеров сужается.



Операция извлечения элемента и перемещения его в зону упаковки манипулятором занимает 1 минуту.

В какой-то момент времени в цех по упаковке пришел запрос из производственного цеха освободить контейнер с определенным типом элементов. Необходимо определить,

какое минимальное количество времени необходимо манипулятору, чтобы освободить запрошенный контейнер, находящийся на текущий момент в позиции под номером mm

Формат входных данных:

В первой строке входных файлов содержится три разделенных пробелом целых числа n, k, m ($2 \le n \le 25$, $1 \le k < n$, $2 \le m \le n$).

Во второй строке содержится n разделенных пробелом чисел, где на i-й позиции стоит hihi ($1 \le hi \le 10000$) — количество элементов в i-ом контейнере.

Формат входных данных:

Выведите одно число — минимальное количество времени, необходимое для освобождения контейнера под номером m.

Замечание:

В первом примере манипулятор не может сразу достигнуть до контейнера, поэтому для начала ему нужно потратить 2 минуты, чтобы освободить контейнер под номером 3. После этого манипулятор уже может выгрузить нужный контейнер, потратив 5 минут.

Во втором примере манипулятор сразу может за 6 минут выгрузить контейнер.

```
Sample Input 1:
  6 2 4
  10 3 2 5 4 4

Sample Output 1:
  7
Sample Input 2:
  5 1 5
  1 4 8 2 6

Sample Output 2:
  6
```

Способ оценки работы:

Для генерации тестов и проверки результата используется следующий код на языке Python:

```
import random
from heapq import heappush, heappop

def generate():
    return ["6 2 4\n10 3 2 5 4 4\n", "5 1 5\n1 4 8 2 6\n"]


def solve(dataset):
    dataset = dataset.splitlines()
    n, k, m = map(int, dataset[0].split())
    m = m - 1
    health = list(map(int, dataset[1].split()))

    heap = []
    r_ans = 0
    last = k
    for i in range(1, last + 1):
        heappush(heap, health[i])
    while last < m:
        r_ans += heappop(heap)
        last = last + 1
        heappush(heap, health[last])

    heap = []
```

```
    l_ans = 0
    last = n - k
    for i in range(last, n):
        heappush(heap, health[i])
    while last > m:
        l_ans += heappop(heap)
        last = last - 1
        heappush(heap, health[last])

    return str(min(l_ans, r_ans) + health[m])

def check(reply, clue):
    return int(reply) == int(clue)
```

Решение на языке C++:

```
#include <iostream>
#include <vector>
#include <set>
#include <cmath>
#include <cstdio>

using namespace std;

int main() {
    int n, k, m;
    cin >> n >> k >> m;
    vector <int> health(n + 1);
    for (int i = 1; i <= n; i++) {
        scanf("%d ", &health[i]);
    }

    multiset <int> heap;
    int r_ans = 0;
    int last = k + 1;
    for (int i = 2; i <= last; i++) {
        heap.insert(health[i]);
    }
    while (last < m) {
        r_ans += *(heap.begin());
        heap.erase(heap.begin());
        heap.insert(health[++last]);
    }

    heap.clear();
    int l_ans = 0;
    last = n - k + 1;
    for (int i = n; i >= last; i--) {
        heap.insert(health[i]);
    }
    while (last > m) {
        l_ans += *(heap.begin());
        heap.erase(heap.begin());
        heap.insert(health[--last]);
    }
```

```
    cout << min(l_ans, r_ans) + health[m] << endl;

}
```

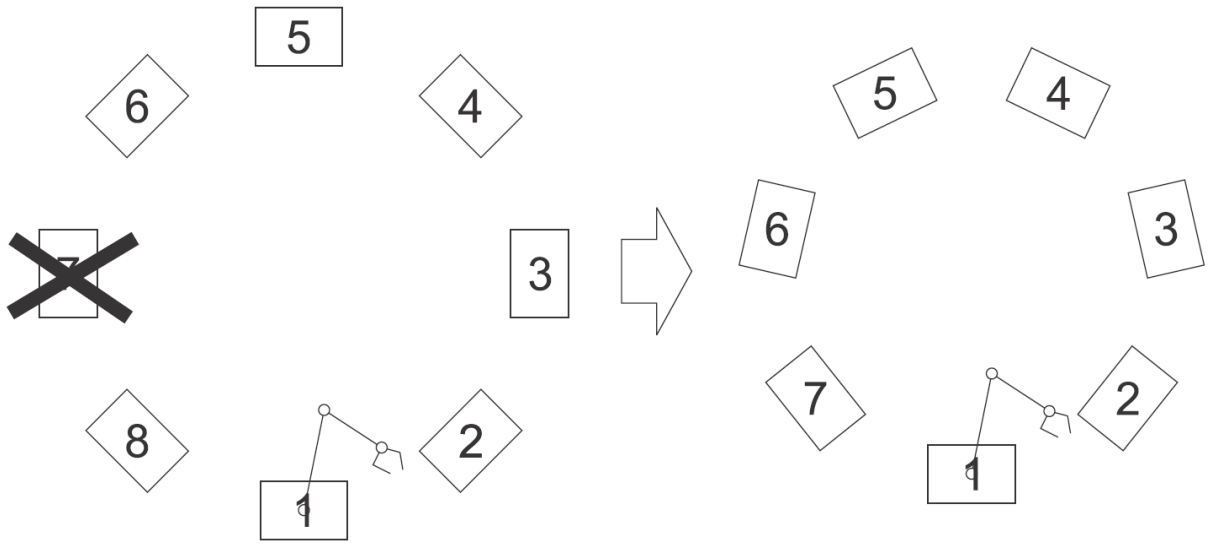
Задача 3.4.2 (9 баллов)

Условие:

Данный шаг отличается от предыдущего формулировкой ограничений на входные данные. Перед проверкой решения задачи с формулировкой ограничений из этого шага, убедитесь, что это решение проходит и на предыдущем шаге.

В цехе по упаковке конечной продукции на заводе по производству мебели робот-манипулятор подготавливает товар к отгрузке. Контейнеры с элементами, из которых упаковывается та или иная единица мебели, расположены так, что робот вместе с ними образует круг. Каждая позиция, где может быть расположен контейнер, пронумерована против часовой стрелки. Манипулятор установлен в позиции 1, справа от него находится контейнер с позицией 2, слева — контейнер с позицией n.

В каждом i-ом контейнере находится ci определенного типа элементов. Во время операций по отгрузке робот может доставать из контейнеров по одному элементу. Длины звеньев манипулятора хватает только, чтобы достичь k-го контейнера по правую или по левую сторону от робота. Если во время операций по отгрузке какой-то контейнер становится пустым, он отправляется в производственных цех, а круг контейнеров сужается.



Операция извлечения элемента и перемещения его в зону упаковки манипулятором занимает 1 минуту.

В какой-то момент времени в цех по упаковке пришел запрос из производственного цеха освободить контейнер с определенным типом элементов. Необходимо определить, какое минимальное количество времени необходимо манипулятору, чтобы освободить запрошенный контейнер, находящийся на текущий момент в позиции под номером m

Формат входных данных:

В первой строке входных файлов содержится три разделенных пробелом целых числа nn, kk, mm ($2 \le n \le 500$, $1 \le k < n$, $2 \le m \le n$).

Во второй строке содержится n разделенных пробелом чисел, где на i-й позиции стоит hihi ($1 \le hi \le 10000$) — количество элементов в i-ом контейнере.

Формат входных данных:

Выведите одно число — минимальное количество времени, необходимое для освобождения контейнера под номером m.

Замечание:

В первом примере манипулятор не может сразу достигнуть до контейнера, поэтому для начала ему нужно потратить 2 минуты, чтобы освободить контейнер под номером 3. После этого манипулятор уже может выгрузить нужный контейнер, потратив 5 минут.

Во втором примере манипулятор сразу может за 6 минут выгрузить контейнер.

```
Sample Input 1:
  6 2 4
  10 3 2 5 4 4

Sample Output 1:
  7
Sample Input 2:
  5 1 5
  1 4 8 2 6

Sample Output 2:
  6
```

Способ оценки работы:

Для генерации тестов и проверки результата используется следующий код на языке Python:

```
import random
from heapq import heappush, heappop

def generate():
    return ["6 2 4\n10 3 2 5 4 4\n", "5 1 5\n1 4 8 2 6\n"]


def solve(dataset):
    dataset = dataset.splitlines()
    n, k, m = map(int, dataset[0].split())
    m = m - 1
    health = list(map(int, dataset[1].split()))

    heap = []
    r_ans = 0
    last = k
    for i in range(1, last + 1):
        heappush(heap, health[i])
    while last < m:
        r_ans += heappop(heap)
        last = last + 1
        heappush(heap, health[last])

    heap = []
    l_ans = 0
    last = n - k
    for i in range(last, n):
        heappush(heap, health[i])
    while last > m:
        l_ans += heappop(heap)
        last = last - 1
        heappush(heap, health[last])

    return str(min(l_ans, r_ans) + health[m])
```

```
def check(reply, clue):
    return int(reply) == int(clue)
```

Решение на языке C++:

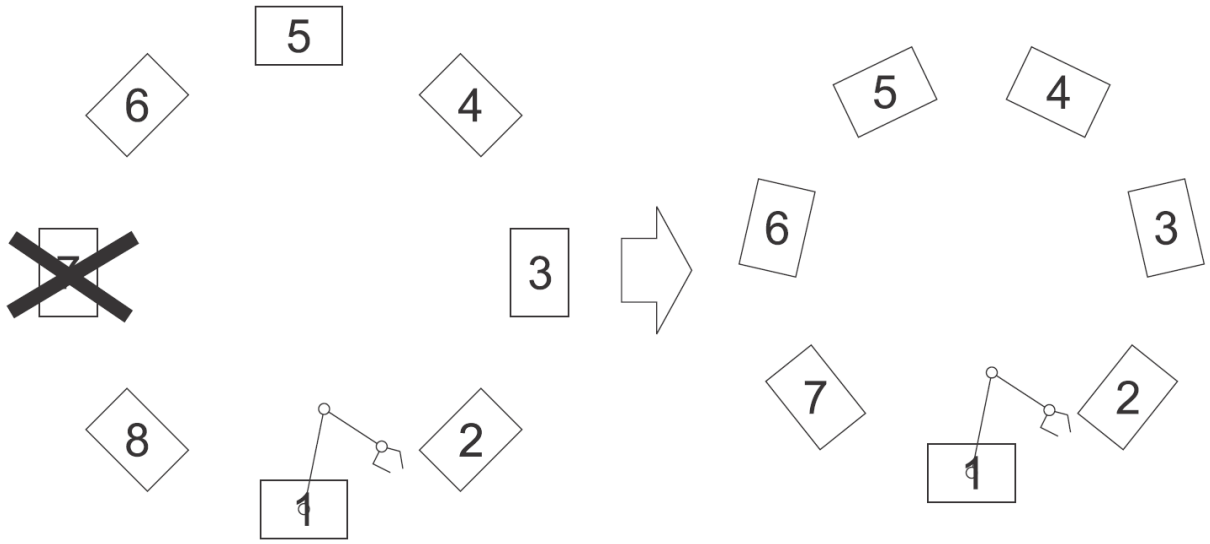
```
from heapq import heappush, heappop

n, k, m = map(int, input().split())
m = m - 1
health = list(map(int, input().split()))

heap = []
r_ans = 0
last = k
for i in range(1, last + 1):
    heappush(heap, health[i])
while last < m:
    r_ans += heappop(heap)
    last = last + 1
    heappush(heap, health[last])

heap = []
l_ans = 0
last = n - k
for i in range(last, n):
    heappush(heap, health[i])
while last > m:
    l_ans += heappop(heap)
    last = last - 1
    heappush(heap, health[last])

print(min(l_ans, r_ans) + health[m])
```

Задача 3.4.3 (15 баллов)

Данный шаг отличается от предыдущего формулировкой ограничений на входные данные. Перед проверкой решения задачи с формулировкой ограничений из этого шага, убедитесь, что это решение проходит и на предыдущем шаге.

В цехе по упаковке конечной продукции на заводе по производству мебели робот-манипулятор подготавливает товар к отгрузке. Контейнеры с элементами, из которых упаковывается та или иная единица мебели, расположены так, что робот вместе с ними образует круг. Каждая позиция, где может быть расположен контейнер, пронумерована против часовой стрелки. Манипулятор установлен в позиции 1, справа от него находится контейнер с позицией 2, слева — контейнер с позицией n.

В каждом i-ом контейнере находится ci определенного типа элементов. Во время операций по отгрузке робот может доставать из контейнеров по одному элементу. Длины звеньев манипулятора хватает только, чтобы достичь k-го контейнера по правую или по левую сторону от робота. Если во время операций по отгрузке какой-то контейнер становится пустым, он отправляется в производственных цех, а круг контейнеров сужается.



Операция извлечения элемента и перемещения его в зону упаковки манипулятором занимает 1 минуту.

В какой-то момент времени в цех по упаковке пришел запрос из производственного цеха освободить контейнер с определенным типом элементов. Необходимо определить, какое минимальное количество времени необходимо манипулятору, чтобы освободить запрошенный контейнер, находящийся на текущий момент в позиции под номером m

Формат входных данных:

В первой строке входных файлов содержится три разделенных пробелом целых числа nn, kk, mm ($2 \le n \le 100000$, $1 \le k < n$, $2 \le m \le n$).

Во второй строке содержится n разделенных пробелом чисел, где на i-й позиции стоит hihi ($1 \le hi \le 10000$) — количество элементов в ii-ом контейнере.

Формат входных данных:

Выведите одно число — минимальное количество времени, необходимое для освобождения контейнера под номером m.

Замечание:

В первом примере манипулятор не может сразу достигнуть до контейнера, поэтому для начала ему нужно потратить 2 минуты, чтобы освободить контейнер под номером 3. После этого манипулятор уже может выгрузить нужный контейнер, потратив 5 минут.

Во втором примере манипулятор сразу может за 6 минут выгрузить контейнер.

```
Sample Input 1:
  6 2 4
  10 3 2 5 4 4

Sample Output 1:
  7
Sample Input 2:
  5 1 5
  1 4 8 2 6

Sample Output 2:
6
```

Способ оценки работы:

Для генерации тестов и проверки результата используется следующий код на языке Python:

```
import random
from heapq import heappush, heappop

def generate():
    return ["6 2 4\n10 3 2 5 4 4\n", "5 1 5\n1 4 8 2 6\n"]


def solve(dataset):
    dataset = dataset.splitlines()
    n, k, m = map(int, dataset[0].split())
    m = m - 1
    health = list(map(int, dataset[1].split()))

    heap = []
    r_ans = 0
    last = k
    for i in range(1, last + 1):
        heappush(heap, health[i])
    while last < m:
        r_ans += heappop(heap)
        last = last + 1
        heappush(heap, health[last])

    heap = []
    l_ans = 0
    last = n - k
    for i in range(last, n):
        heappush(heap, health[i])
    while last > m:
        l_ans += heappop(heap)
        last = last - 1
        heappush(heap, health[last])

    return str(min(l_ans, r_ans) + health[m])

def check(reply, clue):
    return int(reply) == int(clue)
```

Решение на языке C++:

```
#include <iostream>
#include <vector>
#include <set>
#include <cmath>
#include <cstdio>

using namespace std;

int main() {
    int n, k, m;
    cin >> n >> k >> m;
    vector <int> health(n + 1);
```

```
    for (int i = 1; i <= n; i++) {
        scanf("%d ", &health[i]);
    }

    multiset <int> heap;
    int r_ans = 0;
    int last = k + 1;
    for (int i = 2; i <= last; i++) {
        heap.insert(health[i]);
    }
    while (last < m) {
        r_ans += *(heap.begin());
        heap.erase(heap.begin());
        heap.insert(health[++last]);
    }

    heap.clear();
    int l_ans = 0;
    last = n - k + 1;
    for (int i = n; i >= last; i--) {
        heap.insert(health[i]);
    }
    while (last > m) {
        l_ans += *(heap.begin());
        heap.erase(heap.begin());
        heap.insert(health[--last]);
    }

    cout << min(l_ans, r_ans) + health[m] << endl;

}
```

Задача 3.4.4 (10 баллов)

Условие:

Передача информации от Центра Планирования Миссии (ЦПМ) до робота на Марсе происходит пакетами. Пакеты передаются в виде массива, каждый пакет представляет из себя целое положительное число. Пакеты доходят с разной скоростью, и из-за этого существует одна проблема — они могут поменяться местами. Но мы живём в будущем, и не всё уж так плохо: гарантируется, что пакет мог поменяться только с одним из соседних пакетов (после чего эти 2 пакета не могли дальше поменяться местами с новыми соседями).

Главе ЦПМ надо приготовить отчёт, поэтому ему надо узнать, какой может быть максимальное значение суммы, вычисленной по следующей формуле: a1 xor a2 + a3 xor a4 + ... + an−1xor an, где xor - побитовое исключающее ИЛИ. Именно вам предстоит вычислить это значение.

Формат входных данных:

В первой строке дано целое положительное число nn (2⩽n⩽20) — количество элементов массива. Гарантируется, что оно чётное.

Следующая строка содержит n чисел — изначально заданный массив.

Все числа положительные и не превосходят 10^5.

Формат входных данных:

Выведите одно число — максимальное значение суммы, вычисленный вышеописанным способом.

Замечание:

Рассмотрим примеры.

Массив в первом пример, при котором достигается максимальная сумма: 1216
Во втором примере: 567814

```
Sample Input 1:
  4
  1 1 2 6

Sample Output 1:
  10
Sample Input 2:
  6
  5 7 6 1 8 4

Sample Output 2:
  23
```

Способ оценки работы:

Для генерации тестов и проверки результата используется следующий код на языке Python:

```
import random

def generate():
    return ["4\n1 1 2 6\n", "6\n5 7 6 1 8 4\n"]


def solve(dataset):
    dataset = dataset.splitlines()
    n = int(dataset[0])
    a = dataset[1].split()

    for i in range(0, n):
        a[i] = int(a[i])

    a = [a[0]] + a + [a[n - 1]]
    n = n + 2

    dp = [[0 for x in range(2)] for y in range(n)]

    for i in range(3, n, 2):
        dp[i][0] = max(dp[i - 2][0] + (a[i - 2] ^ a[i - 1]), dp[i -
2][1] + (a[i - 3] ^ a[i - 1]))
        dp[i][1] = max(dp[i - 2][0] + (a[i - 2] ^ a[i]), dp[i - 2][1]
+ (a[i - 3] ^ a[i]))

    return str(max(dp[n - 1][0], dp[n - 1][1]))



def check(reply, clue):
    return int(reply) == int(clue)
```

Решение на языке C++:

```
perm = list()


def permutations(arr, i, swapped=False):
    #print(i)
    #print(len(arr))
    arr1 = list()
    if i < len(arr) - 1:
        if not swapped:
            arr1 = arr[0:i]
            arr1.append(arr[i + 1])
            arr1.append(arr[i])
            arr1.extend(arr[i + 2:len(arr)])
            permutations(arr1, i + 1, True)
        permutations(arr, i + 1, False)
    else:
        perm.append(arr)

def xor_paired_sum(arr):
    sum = 0
    for i in range(0, len(arr) - 1, 2):
        sum += arr[i] ^ arr[i+1]
    return sum

n = int(input())
data = [int(x) for x in input().split(' ')]

permutations(data, 0)

max_xor_paired_sum = 0
for i in range(len(perm)):
    c = xor_paired_sum(perm[i])
    if c > max_xor_paired_sum:
        max_xor_paired_sum = c

print(max_xor_paired_sum)
```

Задача 3.4.5 (20 баллов)

Условие:

Данный шаг отличается от предыдущего формулировкой ограничений на входные данные. Перед проверкой решения задачи с формулировкой ограничений из этого шага, убедитесь, что это решение проходит и на предыдущем шаге.

Формат входных данных:

В первой строке дано целое положительное число nn (2⩽n⩽100000) — количество элементов массива. Гарантируется, что оно чётное. Следующая строка содержит n чисел — изначально заданный массив. Все числа положительные и не превосходят 10^5.

Способ оценки работы:

Для генерации тестов и проверки результата используется следующий код на языке Python:

```
import random
```

```
def generate():
    return ["4\n1 1 2 6\n", "6\n5 7 6 1 8 4\n"]


def solve(dataset):
    dataset = dataset.splitlines()
    n = int(dataset[0])
    a = dataset[1].split()

    for i in range(0, n):
        a[i] = int(a[i])

    a = [a[0]] + a + [a[n - 1]]
    n = n + 2

    dp = [[0 for x in range(2)] for y in range(n)]

    for i in range(3, n, 2):
        dp[i][0] = max(dp[i - 2][0] + (a[i - 2] ^ a[i - 1]), dp[i -
2][1] + (a[i - 3] ^ a[i - 1]))
        dp[i][1] = max(dp[i - 2][0] + (a[i - 2] ^ a[i]), dp[i - 2][1]
+ (a[i - 3] ^ a[i]))

    return str(max(dp[n - 1][0], dp[n - 1][1]))



def check(reply, clue):
    return int(reply) == int(clue)
```

Решение на языке C++:

```
n = int(input())
a = input().split()

for i in range(0, n):
    a[i] = int(a[i])

a = [a[0]] + a + [a[n - 1]]
n = n + 2

dp = [[0 for x in range(2)] for y in range(n)]

for i in range(3, n, 2):
    dp[i][0] = max(dp[i - 2][0] + (a[i - 2] ^ a[i - 1]), dp[i - 2][1]
+ (a[i - 3] ^ a[i - 1]))
    dp[i][1] = max(dp[i - 2][0] + (a[i - 2] ^ a[i]), dp[i - 2][1] +
(a[i - 3] ^ a[i]))

print(max(dp[n - 1][0], dp[n - 1][1]))
```

Задача 3.4.6 (15 баллов)

Условие:

В одном из государств было решено создать сеть из n городов, между которыми бы курсировали беспилотные автомобили, но остается проблема возникновения внештатных

ситуаций. Для решения этого вопроса было решено поставить маяки, которые бы управляли автомобилями, в случаях возникновения опасности. В каждом из n городов хотят поставить по одному маяку так, чтобы если какой-то один маяк выйдет из строя, то оставшиеся города тоже представляли из себя единую сеть, что из всех оставшихся существовал путь между любыми двумя городами.

Так как государство не хочет сильно тратиться, то решили произвести одинаковые маяки с наименьшим радиусом действия, но выполняющие все условия безопасности, которые были описаны ранее.

В нашем случае представим, что города — точки на плоскости, а дороги, как кратчайшее расстояние между точками. Заметим, что из города a в город b можно попасть и через другие города. В итоге мы хотим выбрать минимальный радиус маяка, чтобы получить такую сеть, что из любого города можно добраться до любого другого и, если какой-то маяк выйдет из строя, то оставшиеся города тоже представляли из себя единую сеть.

Формат входных данных:

Первая строка содержит одно целое число n (0⩽n⩽100) — количество городов.
Следующие n строк содержат по два числа — координаты городов соответственно.
Все координаты точек неотрицательные и не превосходят 10^9.

Формат выходных данных:

Выведите одно число — минимальный радиус действия для всех маяков, удовлетворяющий всем вышеописанным условиям. Число выведите с точностью не менее 10^−6.

```
Sample Input 1:
  4
  3 0
  0 4
  3 5
  6 4

Sample Output 1:
  2.500000
Sample Input 2:
  6
  2 1
  3 5
  8 2
  8 4
  12 1#This is sample Code Challenge
  import random
  import math

  t = 0


  def isclose(a, b, rel_tol=1e-06, abs_tol=0.0):
      return abs(a-b) <= max(rel_tol * max(abs(a), abs(b)), abs_tol)


  def generate():
      return ["4\n3 0\n0 4\n3 5\n6 4\n", "6\n2 1\n3 5\n8 2\n8 4\n12
  1\n12 5\n"]


  def dfs(v, p, edges, used, tin, up):
```

```
    global t
    used[v] = 1
    up[v] = t
    tin[v] = t
    t = t + 1
    flag = 0
    was = 0
    child = 0
    for i in range(len(edges[v])):
        to = i
        if to == p or edges[v][i] == 0:
            continue
        if used[to] == 0:
            child = child + 1
            was = max(was, dfs(to, v, edges, used, tin, up))
            up[v] = min(up[v], up[to])
            if up[i] >= tin[v]:
                flag = 1
        else:
            up[v] = min(up[v], tin[to])
    if p == -1:
        flag = 0
    if flag == 1 or (p == -1 and child > 1):
        return 1
    else:
        return was


def solve(dataset):
    global t
    dataset = dataset.splitlines()
    n = int(dataset[0])
    a = [list(map(int, dataset[i + 1].split())) for i in range(n)]

    t = 0
    l = -1
    r = 10 ** 18
    while r - l > 1:
        edges = [[0 for i in range(n)] for j in range(n)]
        used = [0] * n
        up = [0] * n
        tin = [0] * n
        t = 0
        m = (l + r) // 2
        for i in range(n):
            for j in range(n):
                dist = (a[i][0]  - a[j][0]) ** 2 + (a[i][1] -
a[j][1]) ** 2
                if dist <= m:
                    edges[i][j] = 1
        res = dfs(0, -1, edges, used, tin, up)
        flag = 0
        for i in range(n):
            if used[i] == 0:
                flag = 1
        if flag == 1 or res == 1:
            l = m
        else:
```

```
                r = m
        return str('%.6f' % (math.sqrt(r) / 2))


    def check(reply, clue):
        return isclose(float(reply), float(clue))

    12 5

Sample Output 2:
    3.041381
```

Способ оценки работы:

Для генерации тестов и проверки результата используется следующий код на языке Python:

```
import random
import math

t = 0

def isclose(a, b, rel_tol=1e-06, abs_tol=0.0):
    return abs(a-b) <= max(rel_tol * max(abs(a), abs(b)), abs_tol)

def generate():
    return ["4\n3 0\n0 4\n3 5\n6 4\n", "6\n2 1\n3 5\n8 2\n8 4\n12 1\n12
5\n"]

def dfs(v, p, edges, used, tin, up):
    global t
    used[v] = 1
    up[v] = t
    tin[v] = t
    t = t + 1
    flag = 0
    was = 0
    child = 0
    for i in range(len(edges[v])):
        to = i
        if to == p or edges[v][i] == 0:
            continue
        if used[to] == 0:
            child = child + 1
            was = max(was, dfs(to, v, edges, used, tin, up))
            up[v] = min(up[v], up[to])
            if up[i] >= tin[v]:
                flag = 1
        else:
            up[v] = min(up[v], tin[to])
    if p == -1:
        flag = 0
    if flag == 1 or (p == -1 and child > 1):
        return 1
    else:
        return was
```

```
def solve(dataset):
   global t
   dataset = dataset.splitlines()
   n = int(dataset[0])
   a = [list(map(int, dataset[i + 1].split())) for i in range(n)]

   t = 0
   l = -1
   r = 10 ** 18
   while r - l > 1:
       edges = [[0 for i in range(n)] for j in range(n)]
       used = [0] * n
       up = [0] * n
       tin = [0] * n
       t = 0
       m = (l + r) // 2
       for i in range(n):
           for j in range(n):
               dist = (a[i][0]  - a[j][0]) ** 2 + (a[i][1] - a[j][1])
** 2
               if dist <= m:
                   edges[i][j] = 1
       res = dfs(0, -1, edges, used, tin, up)
       flag = 0
       for i in range(n):
           if used[i] == 0:
               flag = 1
       if flag == 1 or res == 1:
           l = m
       else:
           r = m
   return str('%.6f' % (math.sqrt(r) / 2))

def check(reply, clue):
   return isclose(float(reply), float(clue))
```

Решение на языке C++:

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include <cstdio>
#include <iomanip>
#include <vector>
#include <algorithm>


using namespace std;

typedef long long ll;



bool intersection(pair <int, int> &p1, pair <int, int> &p2, ll &r) {
     ll dist = ((ll)p1.first - p2.first) * (p1.first - p2.first) +
     ((ll)p1.second - p2.second) * (p1.second - p2.second);
```

```
      //cerr << dist << ' ' << r << '\n';
      return dist <= r;
}

int t;

bool dfs(int v, int p, vector < vector <int> > &edges, vector <bool>
&used, vector <int> &in, vector <int> &up) {
      //cerr << "in " << v << '\n';
      used[v] = true;
      in[v] = up[v] = ++t;
      int child = 0;
      bool flag = false, was = false;
      for(auto i : edges[v]) {
            if(i == p) continue;
            if(!used[i]) {
                  child++;
                  was = max(was, dfs(i, v, edges, used, in, up));
                  up[v] = min(up[v], up[i]);
                  if(up[i] >= in[v]) flag = true;
            } else {
                  up[v] = min(up[v], in[i]);
            }
      }
      //cerr << "out " << v << '\n';
      if(p == -1) flag = false;
      if(flag || (p == -1 && child > 1))
            return true;
      else
            return was;
}



int main() {
    int n;
    cin >> n;
    vector < pair <int, int> > v(n);
    for(auto &i : v) {
     cin >> i.first >> i.second;
    }
     ll l = -1, r = 2 * 1e18 + 1;
    while(r - l > 1) {
     ll m = (l + r) / 2;
     //cerr << "m: " << m << '\n';
     vector < vector <int> > edges(n);
     vector <bool> used(n, false);
     vector <int> in(n), up(n);
     t = 0;
     for(size_t i = 0; i < v.size(); i++) {
            for(size_t j = 0; j < i; j++) {
                  if(intersection(v[i], v[j], m)) {
                        edges[i].push_back(j);
                        edges[j].push_back(i);
                        //cerr << i << ' ' << j << '\n';
                  }
            }
      }
```

```
      bool res = dfs(0, -1, edges, used, in, up);
      bool flag = false;
      for(auto i : used) {
           if(!i) flag = true;
      }
      //cerr << m << ' ' << flag << ' ' << res << endl;
      if(flag || res)
           l = m;
      else
           r = m;
      //cerr << "l r: " << l << ' ' << r << '\n';
    }
    cout << fixed << setprecision(15) << sqrt((double)r) / 2;
      return 0;
}
```

Задача 3.4.7 (25 баллов)

Условие:

Данный шаг отличается от предыдущего формулировкой ограничений на входные данные. Перед проверкой решения задачи с формулировкой ограничений из этого шага, убедитесь, что это решение проходит и на предыдущем шаге.

Формат входных данных:

Первая строка содержит одно целое число nn (0⩽n⩽300) — количество городов. Следующие n строк содержат по два числа — координаты городов соответственно. Все координаты точек неотрицательные и не превосходят 10^9.

Способ оценки работы:

Для генерации тестов и проверки результата используется следующий код на языке Python:

```
import random
import math
t = 0
def isclose(a, b, rel_tol=1e-06, abs_tol=0.0):
    return abs(a-b) <= max(rel_tol * max(abs(a), abs(b)), abs_tol)
def generate():
    return ["4\n3 0\n0 4\n3 5\n6 4\n", "6\n2 1\n3 5\n8 2\n8 4\n12 1\n12
5\n"]
def dfs(v, p, edges, used, tin, up):
    global t
    used[v] = 1
    up[v] = t
    tin[v] = t
    t = t + 1
    flag = 0
    was = 0
    child = 0
    for i in range(len(edges[v])):
        to = i
        if to == p or edges[v][i] == 0:
            continue
        if used[to] == 0:
            child = child + 1
            was = max(was, dfs(to, v, edges, used, tin, up))
            up[v] = min(up[v], up[to])
```

```
            if up[i] >= tin[v]:
                flag = 1
        else:
            up[v] = min(up[v], tin[to])
    if p == -1:
        flag = 0
    if flag == 1 or (p == -1 and child > 1):
        return 1
    else:
        return was
  def solve(dataset):
    global t
    dataset = dataset.splitlines()
    n = int(dataset[0])
    a = [list(map(int, dataset[i + 1].split())) for i in range(n)]

    t = 0
    l = -1
    r = 10 ** 18
    while r - l > 1:
        edges = [[0 for i in range(n)] for j in range(n)]
        used = [0] * n
        up = [0] * n
        tin = [0] * n
        t = 0
        m = (l + r) // 2
        for i in range(n):
            for j in range(n):
                dist = (a[i][0]  - a[j][0]) ** 2 + (a[i][1] - a[j][1])
** 2
                if dist <= m:
                    edges[i][j] = 1
        res = dfs(0, -1, edges, used, tin, up)
        flag = 0
        for i in range(n):
            if used[i] == 0:
                flag = 1
        if flag == 1 or res == 1:
            l = m
        else:
            r = m
    return str('%.6f' % (math.sqrt(r) / 2))


def check(reply, clue):
    return isclose(float(reply), float(clue))
```

Решение на языке C++:

```
import math

n = int(input())
a = [list(map(int, input().split())) for i in range(n)]

t = 0
```

```
def dfs(v, p, edges, used, tin, up):
      global t
      used[v] = 1
      up[v] = t
      tin[v] = t
      t = t + 1
      flag = 0
      was = 0
      child = 0
      for i in range(len(edges[v])):
            to = i
            if to == p or edges[v][i] == 0:
                  continue
            if used[to] == 0:
                  child = child + 1
                  was = max(was, dfs(to, v, edges, used, tin, up))
                  up[v] = min(up[v], up[to])
                  if up[i] >= tin[v]:
                        flag = 1
            else:
                  up[v] = min(up[v], tin[to])
      if p == -1:
            flag = 0
      if flag == 1 or (p == -1 and child > 1):
            return 1
      else:
            return was

l = -1
r = 10 ** 18
while r - l > 1:
      edges = [[0 for i in range(n)] for j in range(n)]
      used = [0] * n
      up = [0] * n
      tin = [0] * n
      t = 0
      m = (l + r) // 2
      for i in range(n):
            for j in range(n):
                  dist = (a[i][0]  - a[j][0]) ** 2 + (a[i][1] -
a[j][1]) ** 2
                  if dist <= m:
                        edges[i][j] = 1
      res = dfs(0, -1, edges, used, tin, up)
      flag = 0
      for i in range(n):
            if used[i] == 0:
                  flag = 1
      if flag == 1 or res == 1:
            l = m
      else:
            r = m
print(math.sqrt(r) / 2)
```