Умный город

2022/23 учебный год

Второй отборочный этап

Задача IV.1. Вывеска НТО (8 баллов)

Темы: постоянный ток, электротехника.

Условие

Какой фестиваль проходит без неоновых вывесок? На рисунке 1 представление электрическая цепь вывески НТО. Она состоит из 15 неоновых ламп. Буква «Н» состоит из фиолетовых ламп, буква «Т» — из синих и буква «О» — из голубых ламп. Сопротивление каждой фиолетовой лампы равно 1 кОм, синей — 2 кОм и голубой 3 кОм. Питание производится от идеального источника напряжения, который выдает напряжение $E = 10$ В. Каково эквивалентное сопротивление данной цепи? Найдите значение наибольшей и наименьшей мощности, выделяющейся на одном резисторе. Ответы округлить до сотых.

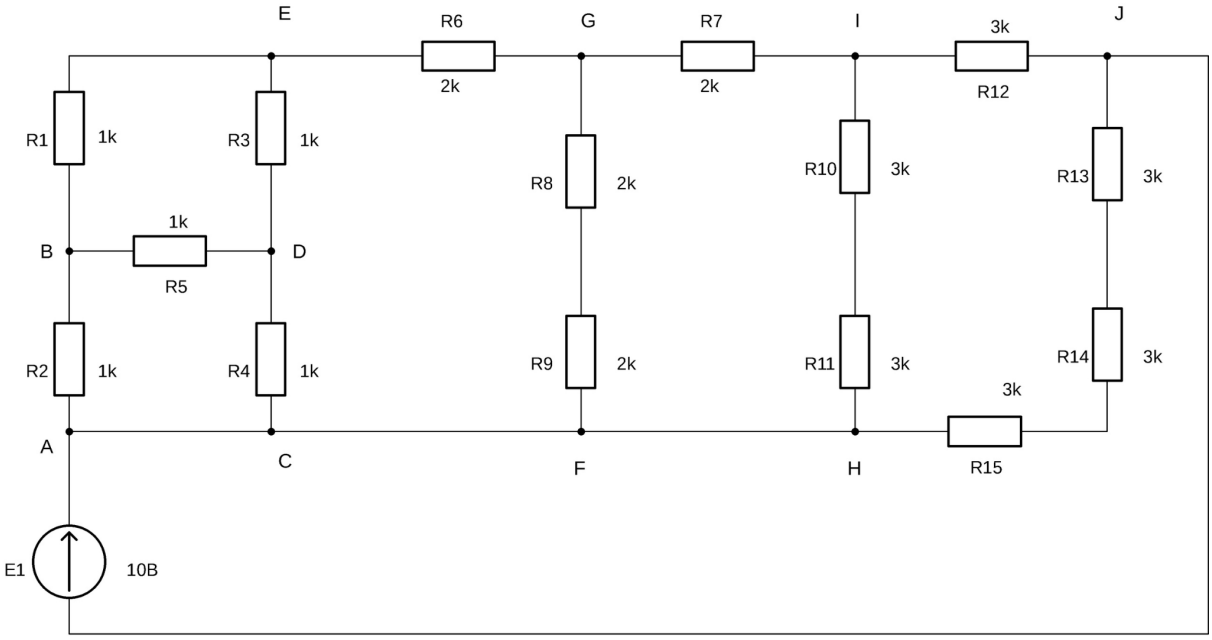
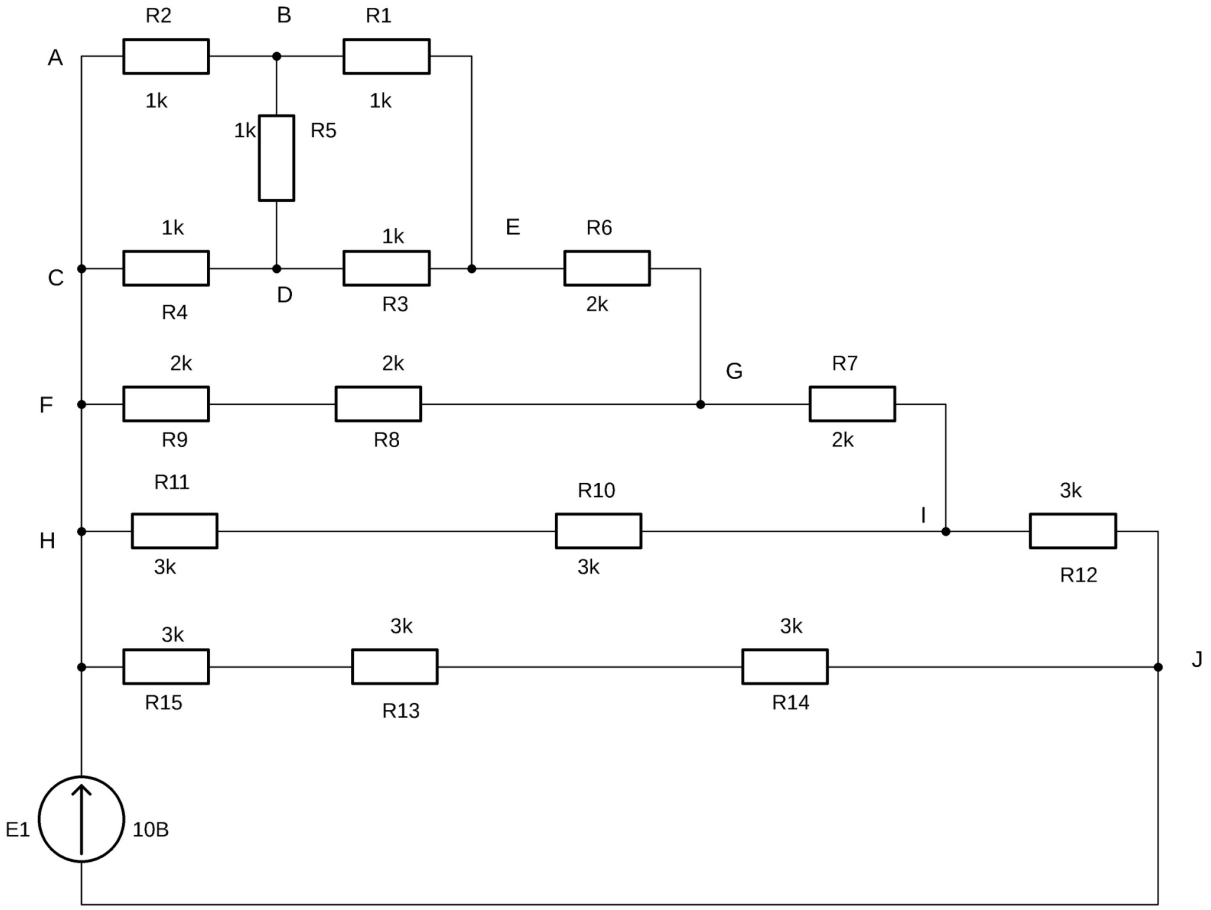


Рис. 1. Электрическая цепь вывески

Решение

Чтобы найти эквивалентное сопротивление цепи нарисуем эквивалентную схему.



Точки B и D находятся симметрично между точками A и E, следовательно, они эквипотенциальны. Ток по закону Ома для участка цепи будет равен:

$$I = \frac{U}{R}.$$

Так как падение напряжения на участке BD равно нулю, то и ток через резистор $R5$ будет равен нулю. Можно сделать вывод, что резистором $R5$ в дальнейших расчетах можно пренебречь. При последовательном соединении резисторов их общее сопротивление можно найти по формуле:

$$R = R_1 + R_2 + ... + R_n.$$

Найдем сопротивления некоторых участков:

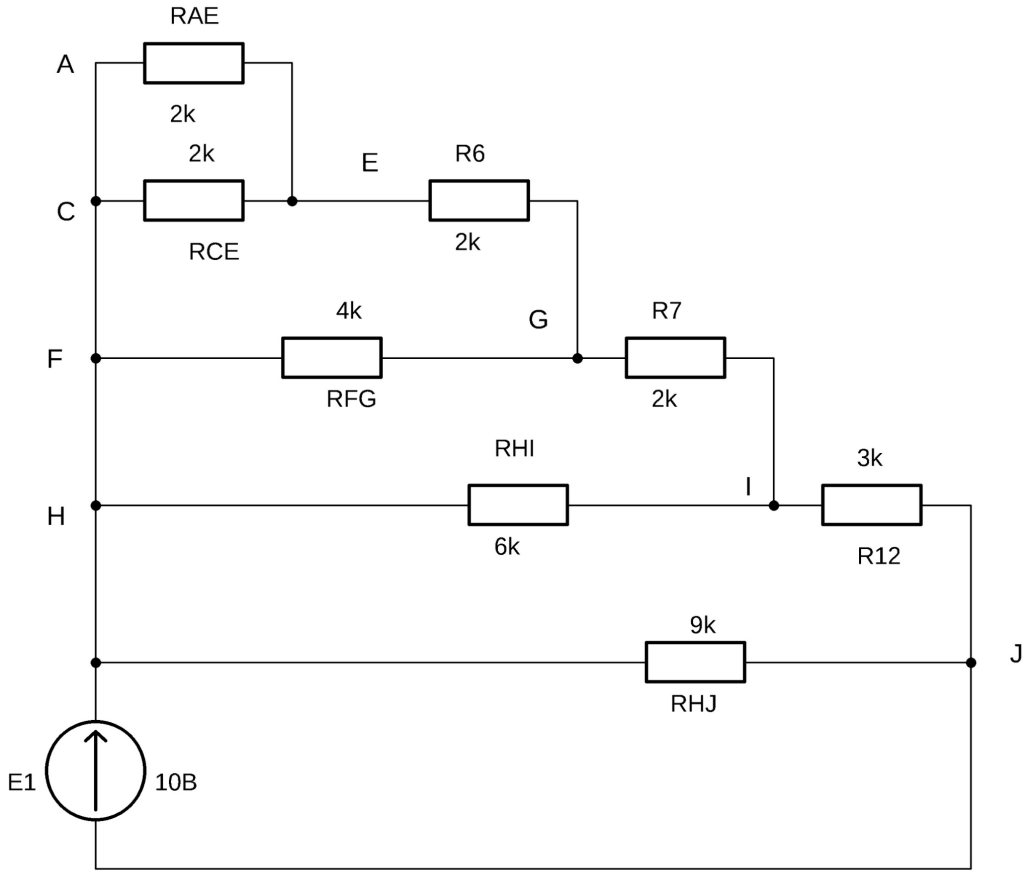
$$R_{JH} = R_{13} + R_{14} + R_{15} = 3 + 3 + 3 = 9 \text{ кОм},$$

$$R_{IH} = R_{10} + R_{11} = 3 + 3 = 6 \text{ кОм},$$

$$R_{GF} = R_8 + R_9 = 2 + 2 = 4 \text{ кОм},$$

$$R_{EC} = R_{EA} = R_1 + R_2 = 1 + 1 = 2 \text{ кОм}.$$

В итоге получится следующая схема замещения.



Сопротивление параллельного участка находится по формуле:

$$R = \frac{1}{R_1} + \frac{1}{R_2} + ... + \frac{1}{R_n}.$$

Резисторы R_{EC} и R_{EA} соединены параллельно, следовательно сопротивление этого участка равно:

$$R_{\text{пар1}} = \frac{1}{R_{EC}} + \frac{1}{R_{EA}} = 1 \text{ кОм}.$$

Первый параллельный участок и R_6 соединены последовательно:

$$R_{\text{смеш1}} = R_{\text{пар1}} + R_6 = 1 + 2 = 3 \text{ кОм}.$$

Далее наш смешанный участок соединен параллельно со следующим участком, поэтому вычисляем его так же, как и выше до тех пор, пока не найдем эквивалентное сопротивление всей цепи.

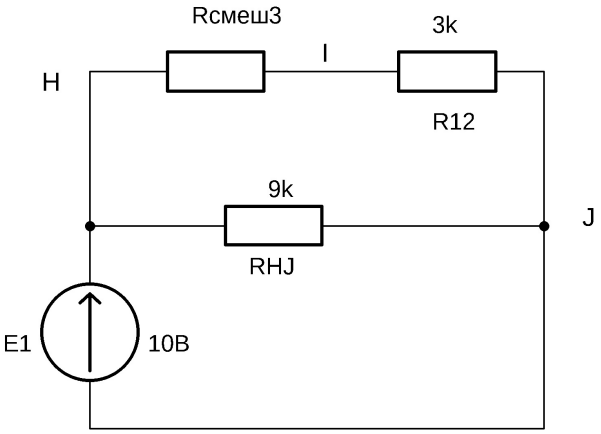
$$R_{\text{экв}} = 3,33 \text{ кОм}.$$

Мощность можно найти по формуле:

$$P = U \cdot I = I^2 \cdot R = \frac{U^2}{R}.$$

Очевидно, что минимальная мощность, выделяющаяся на одном резисторе, выделяется на резисторе $R5$ и равна нулю.

Для нахождения максимальной мощности представим изначальную схему в виде.



Где $R_{\text{смеш3}} = 2,29$ кОм, таким образом, мощность выделяемая на всём:

$$P_{\text{смеш3}} = I^2 \cdot R = \frac{E^2 \cdot R}{(R_{\text{смеш3}} + R_{12})} = \frac{10^2 \cdot 2,29}{(2,29+3)^2} \text{ мВт} = 8,18 \text{ мВт},$$

$$P_{R12} = I^2 \cdot R = \frac{E^2 \cdot R}{(R_{\text{смеш3}} + R_1 2)} = 10^2 \cdot 3(2,29+3)^2 \text{ мВт} = 10,72 \text{ мВт},$$

$$P_{RHJ} = \frac{U^2}{R} = \frac{E^2}{R_{HJ}} = \frac{10^2}{9} \text{ мВт} = 11,11 \text{ мВт}.$$

Как видим, мощность, выделяющаяся на резисторе R_{HJ}, максимальна, но так как он является эквивалентным сопротивлением трех последовательных резисторов, делаем вывод, что максимальная мощность выделяется на резисторе 12.

Ответ: $R_{\text{экв}} = 3,33$ кОм, $P_{min} = 0$ Вт, $P_{min} = 10,72$ Вт.

Задача IV.2. Разряд конденсаторов (8 баллов)

Темы: физика, электротехника.

Проверяет навыки расчета и применения конденсаторов.

Условие

Два конденсатора подключены параллельно к нагрузке R_1 (рис. 2). Емкость первого воздушного конденсатора $C_1 = 100$ мкФ, сопротивление нагрузки $R_1 = 10$ кОм. Второй конденсатор обладает такими же геометрическими размерами, но погружен в непроводящую жидкость, диэлектрическая проницаемость которой прямо пропорциональна напряжению на обкладках конденсатора. Коэффициент пропорциональности равен $\alpha = 0,1$ В$^{-1}$. В какой-то момент времени ключи `SA1` и `SA2` одновременно замыкаются. До замыкания ключей первый конденсатор заряжен до напряжения $U_{C1} = 73$ В, а второй до напряжения $U_{C2} = 15$ В.

Каково будет значение напряжения на конденсаторах в первый момент времени после замыкания ключей? Каково мгновенное значение мощности, выделяющейся на резисторе в первый момент времени? Какое количество теплоты выделится на резисторе за всё время разрядки конденсаторов? Ответ округлить до тысячных.

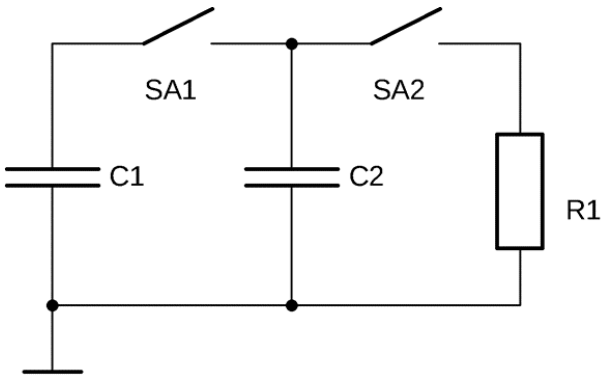


Рис. 2. Принципиальная схема

Решение

Перед замыканием ключей на конденсаторах находился начальный заряд, равный $q_{01} + q_{02} = U_{01} \cdot C_1 + U^2{}_{02} \cdot \alpha \cdot C_1$. В первый момент времени после замыкания ключа заряд перераспределяется таким образом, чтобы напряжение на обоих конденсаторах стало равным. То есть:

$$q = (C_1 + \alpha \cdot U \cdot C_1) \cdot U.$$

Следовательно, получаем следующее уравнение:

$$U_{01} \cdot C_1 + U^2{}_{02} \cdot \alpha \cdot C_1 = (C_1 + \alpha \cdot U \cdot C_1) \cdot U.$$

Решая данное квадратное относительно U уравнение получаем, что:

$$U_1 = 26,305 \text{ В}, \; U_2 = -36,305 \text{ В}.$$

Но так как в данном случае напряжение на обкладках мгновенно не может изменить свой знак, то остаётся только U_1.

Мгновенная мощность резистора в первый момент времени равна:

$$P = \frac{U^2{}_1}{R_1} = 0,069 \text{ Вт}.$$

Количество теплоты, выделившееся на резисторе за всё время разрядки конденсаторов равно начальному значению энергии конденсаторов:

$$Q = \frac{C_1 \cdot (1 + \alpha \cdot U_1) \cdot U^2{}_1}{2} = 0,127 \text{ Дж}$$

Ответ: $U_1 = 26,305$ В; $P = 0,069$ Вт; $Q = 0,127$ Дж.

Задача IV.3. Умный фонарь (8 баллов)

Темы: освещенность, фоторезистор.

В финальной задаче есть задание на управление освещенностью.

Условие

В принципиальной схеме умного фонаря лампочка подключена параллельно к фоторезистору, как показано на рисунке 3. Фоторезистор — это полупроводниковый прибор, у которого сопротивление зависит от освещённости. График данной зависимости приведен на рисунке 4. Освещенность на фоторезисторе складывается по принципу суперпозиции. Сопротивление $R_1 = 2$ кОм, внутреннее сопротивление лампы $R_л = 2$ кОм и напряжение источника питания $E = 120$ В. Каково будет сопротивление фоторезистора и ток лампочки при внешней освещенности равной 3000 люкс? Ответ округлить до сотых.

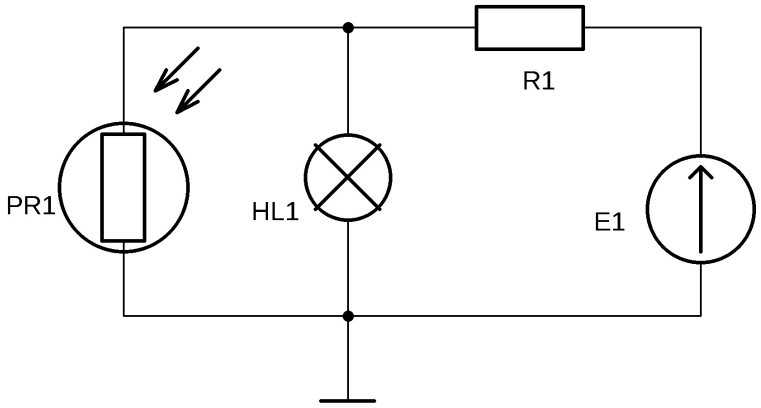


Рис. 3. Принципиальная схема

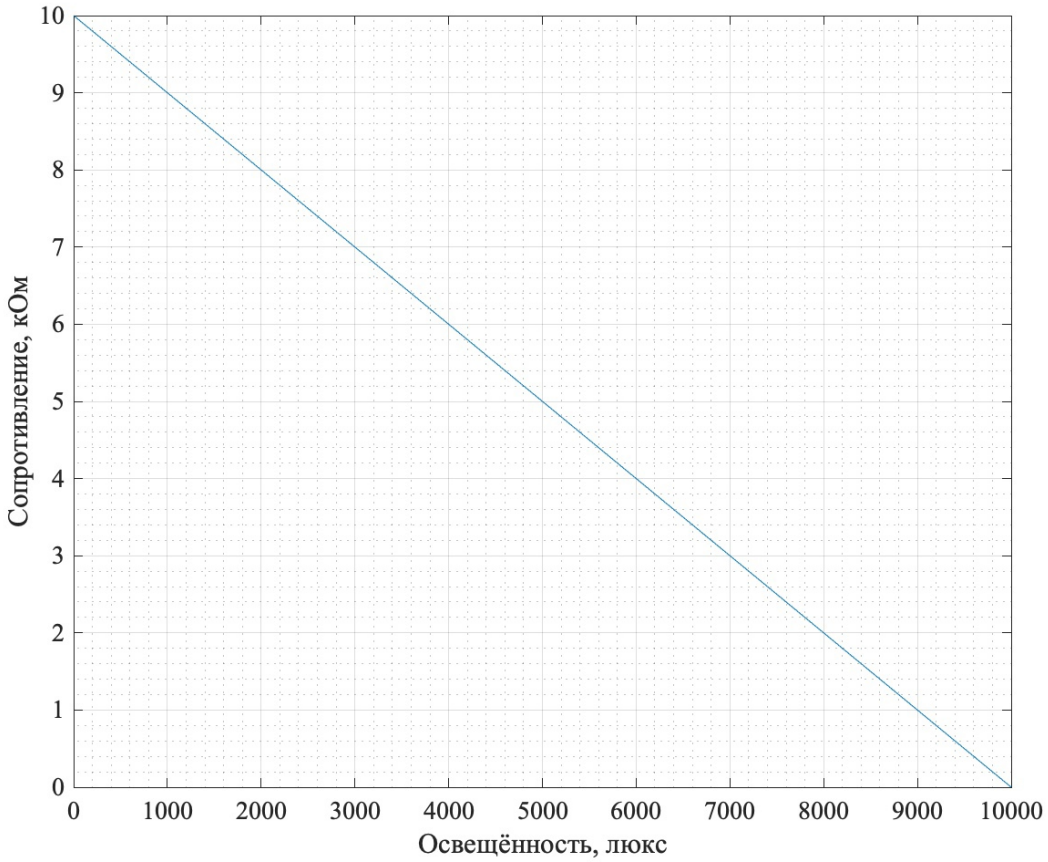


Рис. 4. График зависимости сопротивления фоторезистора от освещенности

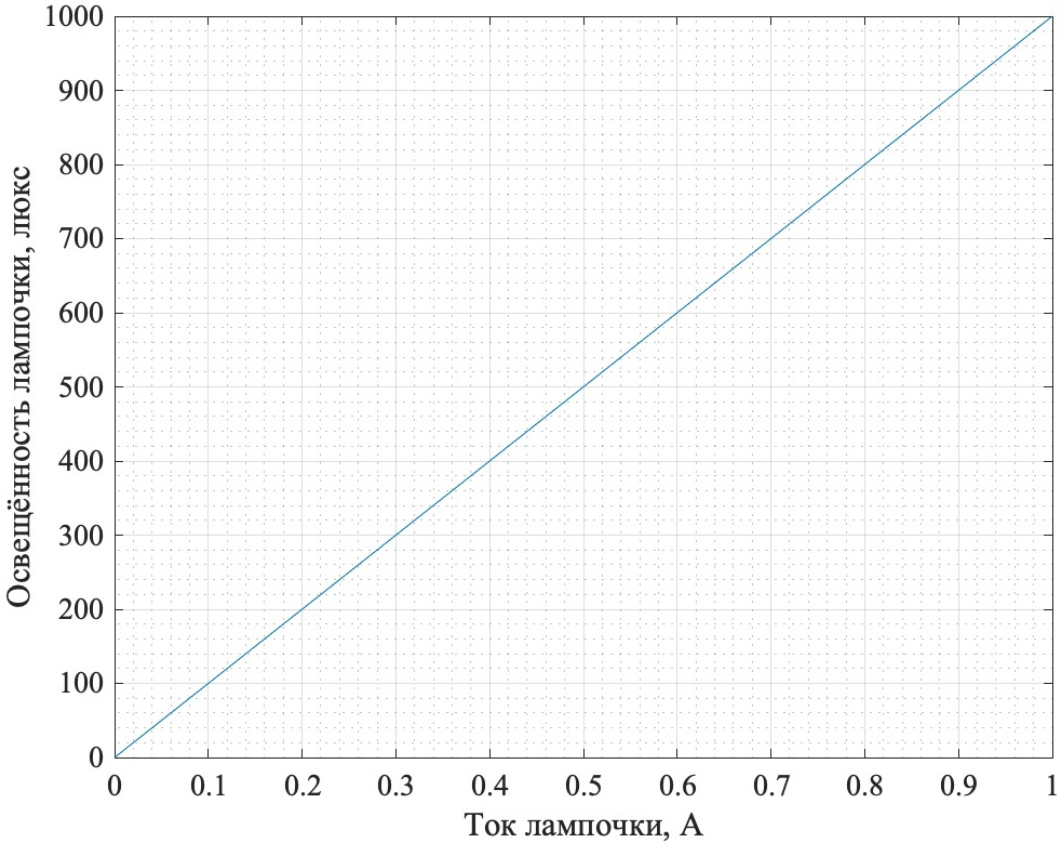


Рис. 5. График зависимости освещенности лампочки от тока

Решение

По первому графику видим, что $R(E_v) = 10000 - E_v$ Ом.

Из второго графика видим, что $E_{vл}(I_л) = I_л \cdot 1000$.

Так как освещённость складывается по принципу суперпозиций получаем, что:

$$E_v = E_{v0} + E_{vл}.$$

Так как: $U_R = I_R \cdot R = U_л = I_л \cdot R_л$, а по второму закону Кирхгофа:

$$E = I \cdot R_1 + I_л \cdot R_л = (I_л + I_R) \cdot R_1 + I_л \cdot R_л = I_л \cdot (R_1 + R_л) + R_1 \cdot I_R.$$

Так как $R_л = R_1$:

$$E = 2R_л \cdot I_л + R_1 \cdot I_R = 2U_л + \frac{U_R \cdot R_1}{R}.$$

Напряжение на лампочке: $U_л = \frac{E}{2+R_1/R} = \frac{E \cdot R}{2R+R_1}$, тогда ток лампочки равен:

$$I_л = \frac{E \cdot R}{(2R + R_1) \cdot R_л}.$$

Тогда:

$$E_v = E_{v0} + E_{vл} = E_{v0} + \frac{E \cdot R}{(2R + R_1) \cdot R_л} \cdot 1000.$$

После чего получается уравнение:

$$R = 10000 - E_{v0} - \frac{E \cdot R}{(2R + R_1) \cdot R_{\text{л}}} \cdot 1000.$$

После решения данного уравнения относительно R, получаем при $E_{v0} = 3000$ люкс: $R = 6,97$ кОм, $E_{v\text{л}} = 26,24$ люкс.

Ответ: $R = 6,97$ кОм, $E_{v\text{л}} = 26,24$ люкс.

Задача IV.4. Зарядка аккумулятора (10 баллов)

Темы: аккумулятор, тригонометрия.

В финальной задаче необходимы знания и навыки работы с аккумуляторными батареями.

Условие

Для работы автономного устройства нужен аккумулятор. Чтобы зарядить аккумулятор с э.д.с равной 3,7 В его подключили через диод к источнику периодического напряжения (рисунок 6), которое задано уравнением:

$$U_{\text{вх}}(t) = \frac{U_m}{2}(\sin(\omega t) - \cos(2\omega t)) = 6(\sin(100\pi \cdot t) - \cos(200\pi \cdot t))$$

.

Внутреннее сопротивление аккумулятора $R_1 = 100$ Ом. Необходимо найти количество электричества, переданное аккумулятору за один час, если считать напряжение аккумулятора практически неизменным. Ответ округлить до целых.

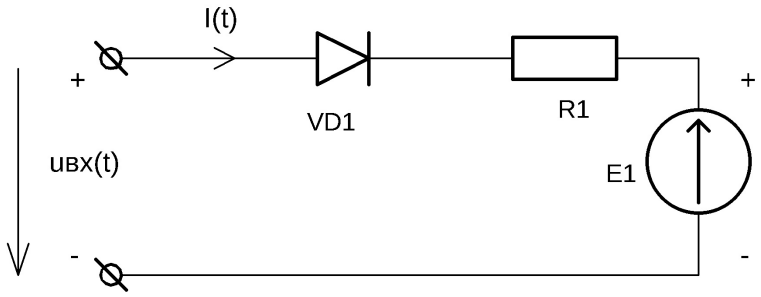


Рис. 6. Принципиальная схема подключения аккумуляторной батареи

Решение

Ток в данной цепи равен: $I(t) = \frac{U_{\text{вх}}(t) - E}{R_1}$, но только в моменты времени, когда ток в цепи положителен. Когда наступает отрицательная полуволна ток становится равным нулю. Для нахождения моментов времени, в которых ток становится равным нулю необходимо следует решить следующее уравнение относительно t:

$$6(\sin(100\pi \cdot t) - \cos(200\pi \cdot t)) - 3,7 = 0.$$

Распишем косинус по формуле двойного угла:

$$6(\sin(100\pi \cdot t) + 2\sin^2(100\pi \cdot t) - 1) - 3,7 = 0.$$

Получаем квадратное уравнение относительно $\sin(100\pi \cdot t)$:

$$12\sin^2(100\pi \cdot t) + 6\sin(100\pi \cdot t) - 9,7 = 0.$$

Тогда:

$\sin(100\pi \cdot t_1) = -1,183$ — синус не может быть больше 1 по модулю,

$$\sin(100\pi \cdot t_2) = 0,683,$$

$$t_{21} = \frac{\arcsin(0,683) + 2\pi k}{100\pi},$$

$$t_{22} = \frac{\pi - \arcsin(0,683) + 2\pi k}{100\pi}, \text{ где } k = 0, 1, 2...$$

При подстановке k=0 получаем:

$$t_{21} = \frac{\arcsin(0,683)}{100\pi},$$

$$t_{22} = \frac{\pi - \arcsin(0,683)}{100\pi}.$$

На интервале времени $(t_{21}; t_{22})$ ток в цепи положителен, и происходит зарядка аккумулятора.

Для нахождения количества электричества за 1 период зарядки необходимо воспользоваться следующей формулой:

$$Q_0 = \int_{t_{21}}^{t_{22}} I(t)dt = \frac{1}{R}\int_{t_{21}}^{t_{22}} (U_{\text{вх}}(t) - E)dt = 118 \text{ мкКл}$$

Всего за час прошло $n = \frac{1\text{ч}}{T} = \frac{3600\text{с}}{0,02\text{с}} = 180000$ периодов, где T — время одного периода. Тогда итоговый заряд равен: $Q = Q_0 \cdot n = 178 \text{ мкКл} \cdot 180000 = 21$ Кл.

Ответ: $Q = 21$ Кл.

Задача IV.5. Метеостанция (9 баллов)

Темы: температура, термистор, измерительный мост.

В финальной задаче есть задание на использование датчика температуры.

Условие

Четыре термистора в метеостанции измеряют температуру в разных точках, они соединены по схеме, изображенной на рисунке 7. Источник питания выдает напряжение равное 11 В и имеет внутреннее сопротивление $r = 200$ Ом. Термисторы TH_1 и TH_4 имеют график зависимости сопротивления от температуры, изображенный на рисунке 8 красным цветом, а термисторы TH_2 и TH_3 — график,

изображенный на рисунке 8 синим цветом. При какой температуре через резистор не будет течь ток? Найдите мощность, потребляемую схемой при этой температуре. Ответ округлить до тысячных.

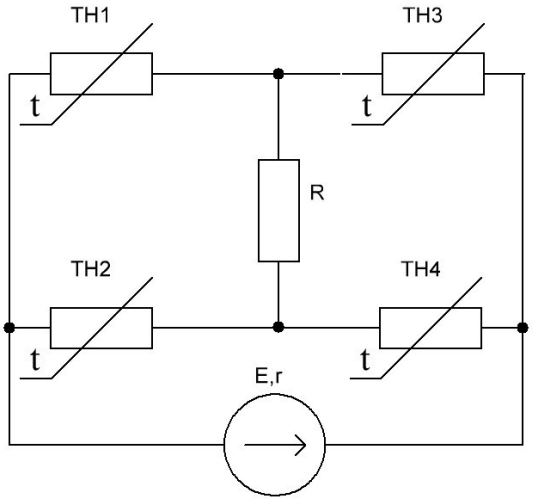


Рис. 7. Принципиальная схема подключения термисторов

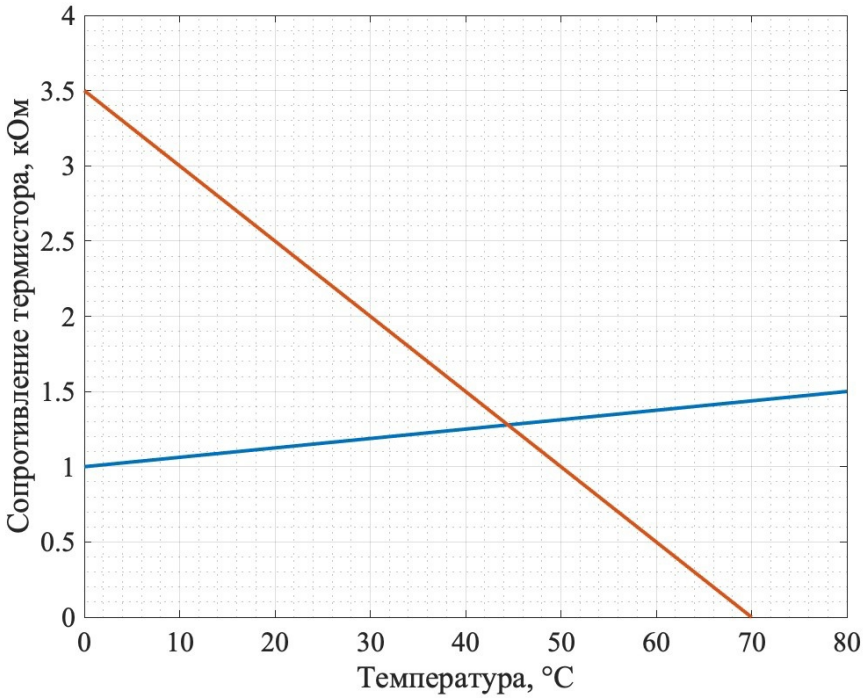


Рис. 8. Зависимость сопротивления термисторов от температуры

Решение

Ток через резистор не будет течь при условии равновесия моста.

Условием равновесия измерительного моста является следующее соотношение:

$$\frac{R_{TH1}}{R_{TH3}} = \frac{R_{TH2}}{R_{TH4}}.$$

Из графиков видим:

$$R_{TH1}(T) = R_{TH4}(T) = -0,05 \cdot T + 3,5 \text{ кОм},$$

$$R_{TH2}(T) = R_{TH3}(T) = 0,00625 \cdot T + 1 \text{ кОм}.$$

Получаем уравнение:

$$\frac{-0,05 \cdot T + 3,5}{0,00625 \cdot T + 1} = \frac{0,00625 \cdot T + 1}{-0,05 \cdot T + 3,5},$$

тогда

$$(0,00625 \cdot T + 1)^2 = (-0,05 \cdot T + 3,5)^2.$$

Преобразуя данное уравнение, получаем:

$$(3,9 \cdot 10^{-5} - 0,0025) \cdot T^2 + (0,0125 + 0,35) \cdot T + (1 - 12,25) = 0,$$

$$T_1 = 44,444^\circ\text{C},$$

$T_2 = 102,9^\circ\text{C}$ — при данной температуре сопротивление первого и четвёртого термистора по формуле становится отрицательным, следовательно данная температура является ложным решением.

$$R_{TH1}(T_1) = R_{TH4}(T_1) = R_{TH2}(T_1) = R_{TH3}(T_1) = R_1 = 1,278 \text{ кОм}.$$

Общее сопротивление схемы:

$$R_{\text{общ}} = \frac{4{R_1}^2}{4R_1} + r = R_1 + r.$$

Схема потребляет мощность:

$$N = \frac{E^2}{R_1 + r} = \frac{121}{1478} = 0,082 \text{ Вт}.$$

Ответ: $T = 44,444^\circ\text{C}$, $N = 0,082$ Вт.

Задача IV.6. Солнечная энергия (6 баллов)

Темы: заряд, потери.

В финальной задаче участникам будет необходимо осуществлять менеджмент электопитания.

Условие

В станции зарядки телефонов стоит фотоэлемент, который представляет из себя источник тока, берущий энергию из света. График зависимости тока фотоэлемента от времени суток представлен на рисунке 9. Фотоэлемент заряжает аккумулятор, теряющий на заряд и разряд 20% заряда. В период времени с 8 до 15 часов аккумулятор заряжался, а после был отключен для заряда телефонов. Сколько телефонов, обладающих емкостью 5 А·ч, сможет полностью зарядить аккумулятор?

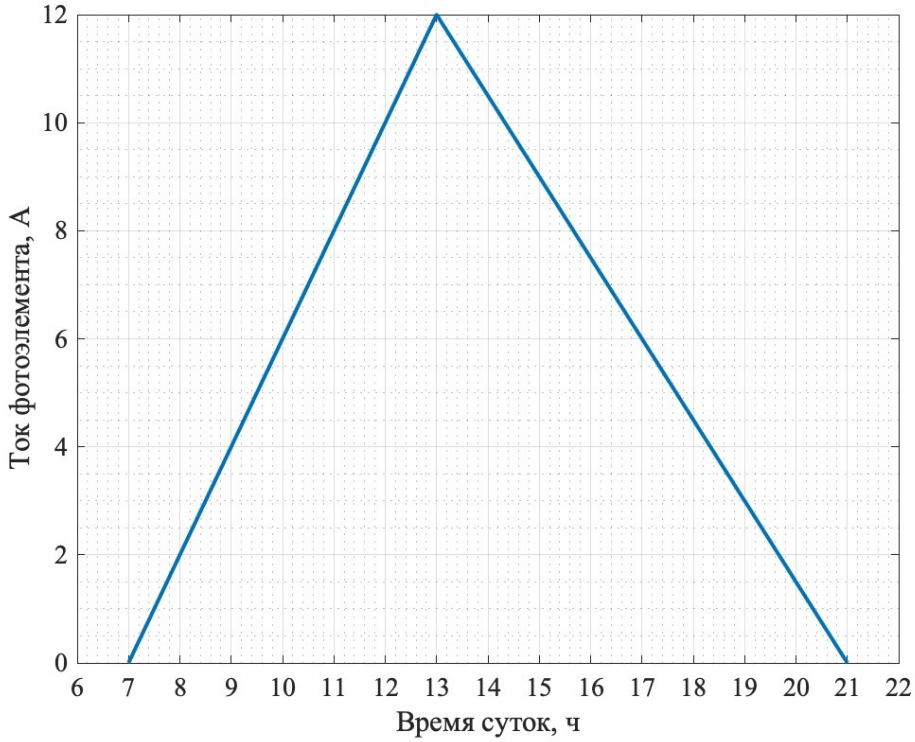


Рис. 9. График зависимости тока фотоэлемента от времени суток

Решение

Заряд аккумулятора равен площади под графиком на нужном интервале времени умноженный на 0,8, с учётом потерь:

$$Q_{\text{ак}} = S \cdot 0,8.$$

Площадь под графиком равна сумме площадей трапеций на интервалах времени (8 ч; 13 ч) и (13 ч; 15 ч), то есть:

$$S = (13-8) \cdot 3600 \cdot \frac{(2+12)}{2} + (15-13) \cdot 3600 \cdot \frac{(12+9)}{2} = 201600 \text{ Кл},$$

$$Q_{\text{ак}} = S \cdot 0,8 = 161280 \text{ Кл} = 44,8 \text{ А} \cdot \text{ч}.$$

Количество телефонов, которое можно зарядить таким количеством заряда равно:

$$n = \frac{44,8 \cdot 0,8}{5} = 7 \text{ штук}$$

Ответ: $n = 7$.

Задача IV.7. Сортировка багажа (8 баллов)

Темы: программирование, строки, структуры данных.

Данная задача проверяет навыки работы со структурными данными, стеком.

Условие

После прилета самолета в аэропорт Умного города все чемоданы пассажиров увозят из багажного отсека в здание аэропорта, и выгружают на конвейер. Каждый чемодан перед вылетом предварительно маркируется одной из 26 заглавных букв латинского алфавита, каждая буква обозначает первую букву имени владельца чемодана.

Далее манипулятор берет ближайший к нему чемодан и перемещает чемодан в отсек, соответствующей букве на чемодане — такая система позволяет пассажирам быстро найти свой чемодан после прилета.

Так как все чемоданы лежат на конвейере в ряд и отмечены буквами, то их можно представить в виде строки s. При этом ближайший к манипулятору чемодан — это чемодан s_1. Назовем несколько подряд идущих чемоданов $s_i, s_{i+1}, ..., s_k$ одинаковой группой, если:

$$s_i = s_{i+1} = ... = s_k,$$
$$s_{i-1} \neq s_i (i \neq 1),$$
$$s_{k+1} \neq s_k (k \neq n),$$
$$k - i \geqslant 1.$$

На время ремонта старого манипулятора, ремонтная бригада установила новый манипулятор, который действует по следующему алгоритму: берется первая справа от манипулятора одинаковая группа и перемещается с конвейера в соответствующий отдел, после чего чемоданы, которые остались правее сдвигаются влево так, чтобы между соседними чемоданами не было пустого места.

Иногда новый манипулятор, действуя по своему алгоритму, может убрать все чемоданы с ленты:

$$\text{BCBBBAACBBAA} \to \text{BCAACBBAA} \to \text{BCCBBAA} \to \text{BBBAA} \to \text{AA} \to$$

Однако в некоторых случаях на конвейерной ленте остаются чемоданы:

$$\text{ABCCDDBAD} \to \text{ABDDBAD} \to \text{ABBAD} \to \text{AAD} \to \text{D}$$

В таких случаях на помощь приходят сотрудники аэропорта и сортируют оставшиеся чемоданы вручную. Ваша задача — по имеющейся строке s определить может ли новый манипулятор отсортировать все чемоданы самостоятельно или придется вызывать штатных грузчиков аэропорта.

Формат входных данных

В первой строке входных данных содержится целое число $n(1 \leqslant n \leqslant 5 \cdot 105)$ — длина строки. Во второй строке задана строка s, состоящая из заглавных букв латинского алфавита.

Формат выходных данных

Если новый манипулятор может отсортировать все чемоданы самостоятельно и вызывать рабочих не придется, выведите «No», иначе — выведите «Yes».

Примеры

Пример №1

Стандартный ввод
7 ABCBBBADDACBBAA
Стандартный вывод
No

Пример №2

Стандартный ввод
6 BCDDCB
Стандартный вывод
No

Пример №3

Стандартный ввод
9 ABCCDDBAD
Стандартный вывод
Yes

Пример №4

Стандартный ввод
12 ABCABBCBCABC
Стандартный вывод
Yes

Пояснения к примерам

1. ABCBBBADDACBBAA → ABCADDACBBAA → ABCAACBBAA →
   → ABCCBBAA→ ABBBAA → AAA →
2. BCDDCB → BCCB → BB →
3. ABCCDDBAD → ABDDBAD → ABBAD → AAD → D
4. ABCABBCBCABC → ABCACBCABC

Решение

Будем идти по строке слева-направо и записывать в стек встреченные чемоданы:

```
for symb in s:
```

В случае, если последний чемодан в стеке подписан той же буквой, что и текущий рассматриваемый, то тогда нужно удалить этот чемодан в стеке и не заносить в стек текущий чемодан:

```
while stack[-1] == symb:
    deletions = True
    last_deleted = symb
    stack.pop()
```

Также нужно запомнить чемодан с какой буквой мы только что удалили:

```
last_deleted = symb
```

Потому что далее могут идти чемоданы с такой же буквой, их мы рассматривать не будем:

```
if symb == last_deleted:
    continue
```

Если же текущий чемодан не совпадает с последним в стеке, то заносим в стек текущий чемодан:

```
if not deletions:
    stack.append(symb)
```

Отметим, что изначально мы создали не пустой стек — там находится символ «#»:

```
stack = ['#']
```

Это сделано для того чтобы не обрабатывать отдельно случаи когда стек пустой. Также есть момент, где мы приравниваем:

```
last_deleted = '#'
```

Цель этого — отметить, что на предыдущем действии мы не удаляли чемоданы и новых таких же не ждем. Символ «#» в строке не встречается по условию.

В последних строках выводится ответ — если в стеке один элемент (изначальное «#»), то все чемоданы удалились и ответ «No», в противном случае — «Yes»

Пример программы-решения

Ниже представлено решение на языке Python 3.

```python
def solve():
    n = int(input())
    s = str(input())

    stack = ['#']
    last_deleted = ''

    for symb in s:
        if symb == last_deleted:
            continue

        deletions = False
        last_deleted = '#'

        while stack[-1] == symb:
            deletions = True
            last_deleted = symb
            stack.pop()

        if not deletions:
            stack.append(symb)

    if len(stack) == 1:
        print("No")
    else:
        print("Yes")

if __name__ == '__main__':
    solve()
```

Задача IV.8. Распределение площади (8 баллов)

Темы: программирование, геометрия.

Данная задача важна в контексте решения задачи управления динамикой.

Условие

На фестивале Умного города будет представлена продукция малого бизнеса. Для каждого предприятия выделяется киоск, основание которого имеет форму квадрата со стороной d, а стены вертикальны. Киоски будут располагаться на главной площади города, она имеет форму прямоугольника со сторонами a и b.

Киоски планируется расставить вдоль границ площади, соблюдая следующие требования:

- расстояние между любой парой киосков не меньше s (измеряется как минимальное расстояние между точками двух киосков);
- одну из сторон площади надо оставить свободной, кроме, возможно, киосков в углах;
- расположение киосков симметрично относительно линии, проведенной перпендикулярно свободной стороне.

Какое максимальное число киосков можно разместить, соблюдая эти требования?

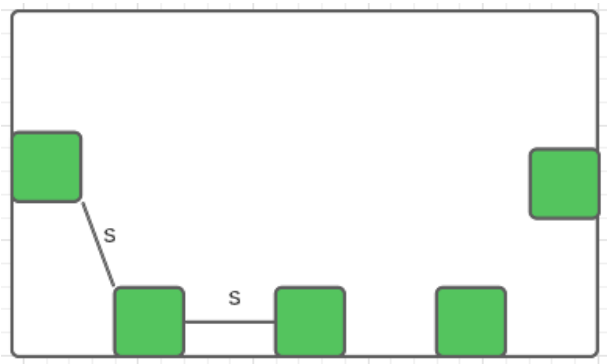
Формат входных данных

В первой строке содержатся два целых числа a, b ($1 \leqslant a, b \leqslant 109$) — размеры площади. Во второй строке содержатся два целых числа, $d, s (1 \leqslant d, s \leqslant 109)$ — размер стороны квадратного основания киоска и минимальное расстояние между любой парой киосков.

Гарантируется, что соблюдаются следующие соотношения: $d \leqslant s, d < a, d < b$.

Формат выходных данных

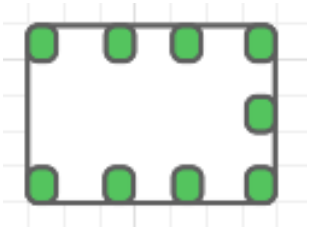
Одно целое число — максимальное число киосков.



Примеры

Пример №1

Стандартный ввод
70 50 10 10
Стандартный вывод
9



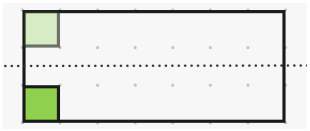
Пример №2

Стандартный ввод
100 200 10 15
Стандартный вывод
18

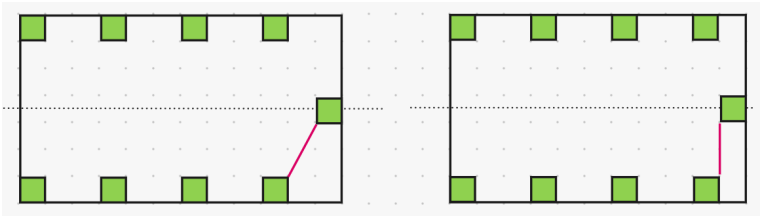
Решение

Рассмотрим несколько случаев:

1. Пусть свободной останется одна (или две) сторона b
   Тогда должно выполнятся правило симметрии относительно стороны b. То есть при расположении одного киоска на нижней стороне, автоматически размещается киоск на верхней.



   Расстояние между такой парой киосков может быть меньше s, и это надо проверить.
   Если места достаточно, то расставляем киоски вдоль стороны a как можно более близко друг другу, начиная с самого угла (можно показать, что множество расстановок с максимальным ответом включает в себя то, как будем расставлять мы, ну или просто из жадности).
   Руководствуясь таким правилом, мы расположим некоторое максимальное число киосков, которое можно расставить вдоль стороны a (тут так же легко доказать, что для максимизации ответа мы можем сначала максимизировать ответ по стороне a, потом по b. Что это значит: значит, что если у нас не влезают несколько киосков на сторону b, то мы не будем в угоду того, чтобы их поставить убирать крайние киоски со стороны a, т. к. это не улучшит ответ).
   Далее могут возникнуть две разные ситуации:



   Принципиальное их различие в том, как мы считаем расстояние от крайнего киоска по стороне a и крайнего по стороне b:
   1.1. Считаем расстояние между углами. Этот случай попадает под условие $h > d$, где h — место оставшееся после размещения максимального числа киосков по стороне a.
   Всего мы можем разместить $side_a = 1 + \frac{a-d}{d+s}$ киосков по стороне a. Тогда $h = a - side_a \cdot d$.

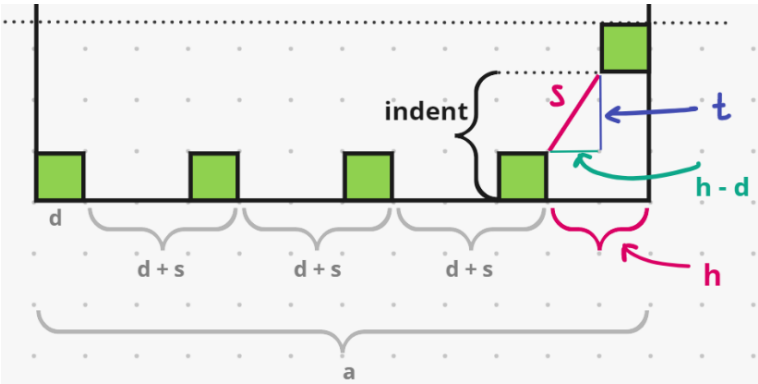


Рис. 10

Мы хотим найти минимальный отступ по стороне b, такой, что, начиная от него, мы можем ставить киоски по стороне b. Это такое расстояние, что если мы отступим его по стороне b и поставим там киоск, то расстояние между ним и крайним по стороне a будет равно в точности s.

Что мы знаем кроме s? Мы знаем h, и знаем d, тогда можно найти зеленый катет $h - d$ (см. рисунок 10).

Зная s и $h - d$ найдем синий катет t. Заметим, что $h - d \Leftarrow s$, т. к. если это не так, то можно расположить еще один киоск на стороне a.

Если мы знаем t, то можем найти отступ $indent = t + d$ — то, что мы и искали изначально.

Тогда на стороне b есть $b - 2 \cdot indent$ свободного места для киосков (берем два отступа с каждой стороны, т. к. они симметричны).

Если $free\_space\_b = (b - 2 * indent) < 0$, будем считать $free\_space\_b = = 0$.

Тогда максимальное количество киосков на стороне b равно:

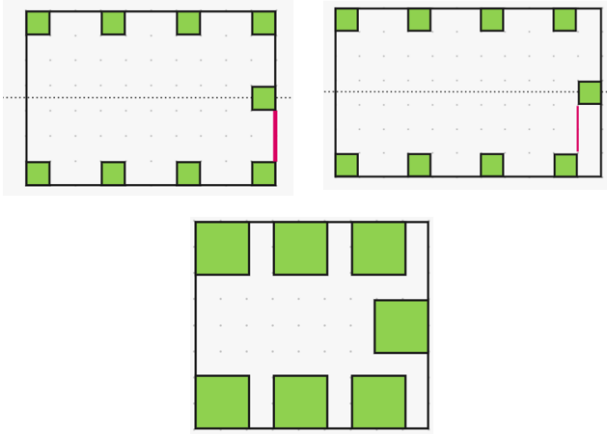
$$int(free\_space\_b \geqslant d) + \max(0, \frac{free\_space\_b - d}{d + s}).$$

Сначала мы посмотрели можно ли поставить хотя бы один киоск, потом руководствовались тем же правилом что и при размещении киосков по стороне a.

Нетрудно заметить, что после мы можем распределить киоски по стороне b чтобы все было симметрично.

1.2. Если $h \leqslant d$.

Рассуждаем как и прежде, единственное, $t = s$ в этом случае и $indent = = d + s$.



2. Пусть свободной останется одна (или две) сторона a.
   Тоже самое.
   Что бы не менять буквы в коде a на b и b на a, можно просто поменять переменные местами и вставить тот же код что и для случая 1.

Среди двух случаев выбираем тот, где ответ больше.

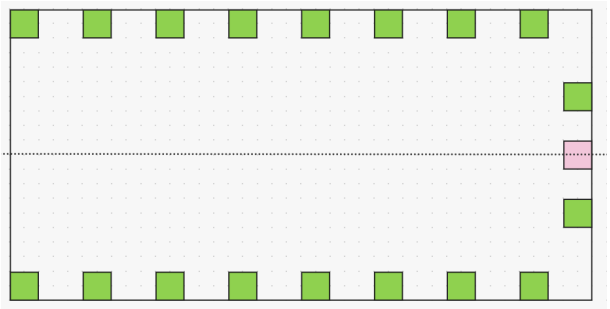
Примеры

Пример №1

Стандартный ввод
70 50 10 10
Стандартный вывод
9

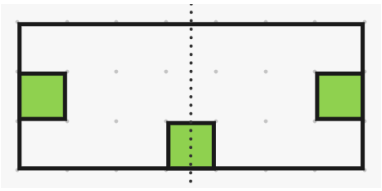
Пример №2

Стандартный ввод
100 200 10 15
Стандартный вывод
18



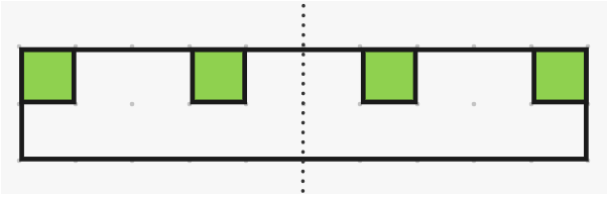
Пример №3

Стандартный ввод
7 3 1 2
Стандартный вывод
3



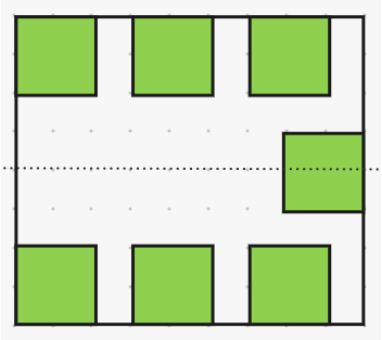
Пример №4

Стандартный ввод
10 2 1 2
Стандартный вывод
4



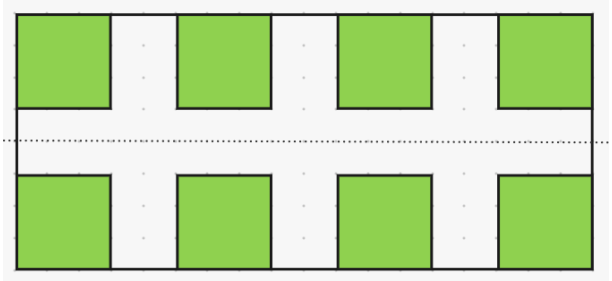
Пример №5

Стандартный ввод
9 8 2 1
Стандартный вывод
7



Пример №6

Стандартный ввод
8 18 3 2
Стандартный вывод
8



Пример программы-решения

Ниже представлено решение на языке Python 3.

```python
import math

def solve():
    a, b = map(int, input().split())
    d, s = map(int, input().split())

    free_side_a_best_result = 0
    free_side_b_best_result = 0

    free_space_btw_sides_a = b - 2 * d
    # OK
    if free_space_btw_sides_a >= s:

        side_a = 1 + ((a - d) // (d + s))
        h = a - side_a * (d + s) + d

        if h > d:
            hd = h - d
            t = math.sqrt(s * s - hd * hd)
            indent = t + d
        else:
            indent = d + s

        free_space_b = max(0, b - 2 * indent)
        side_b = int(free_space_b >= d) + (max(0, (free_space_b - d)) // (d + s))

        free_side_a_best_result = 2 * side_a + side_b

    free_space_btw_sides_b = a - 2 * d
    # OK
    if free_space_btw_sides_b >= s:
        a, b = b, a

        side_a = 1 + ((a - d) // (d + s))
        h = a - side_a * (d + s) + d

        if h > d:
            hd = h - d
            t = math.sqrt(s * s - hd * hd)
            indent = t + d
        else:
            indent = d + s

        free_space_b = max(0, b - 2 * indent)
        side_b = int(free_space_b >= d) + (max(0, (free_space_b - d)) // (d + s))

        free_side_b_best_result = 2 * side_a + side_b

    print(max(free_side_a_best_result, free_side_b_best_result))

if __name__ == '__main__':
    solve()
```

Задача IV.9. Игра (8 баллов)

Темы: динамическое программирование, деревья, обход графов.

Данная задача проверяет навыки работы с графами.

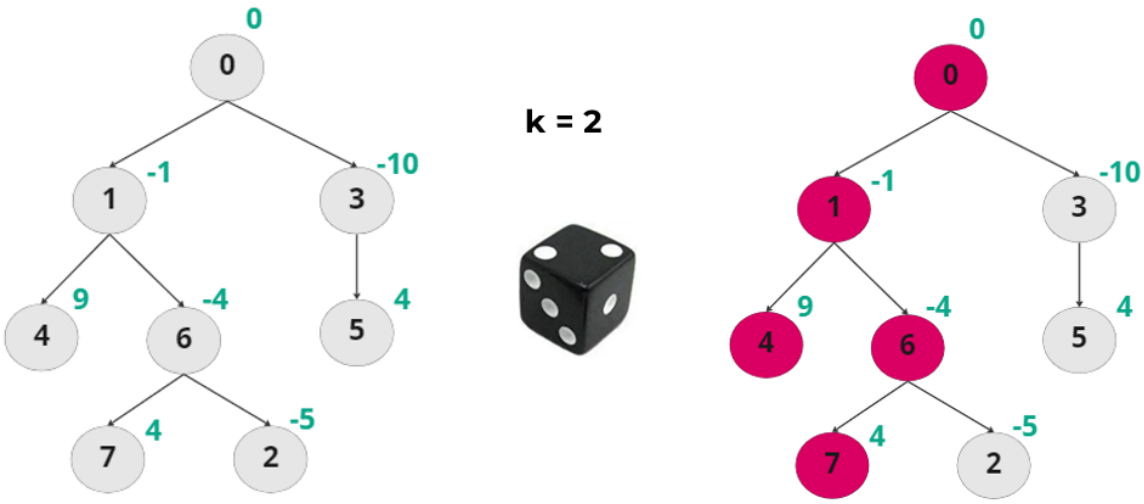
Условие

Петя играл на фестивале Умного города в игры, которым удалось обучить компьютеры, и компьютеры не просто научились, а во много раз превзошли самих людей. Это вызвало сомнительные чувства у Пети, однако придя домой он захотел написать свою программу для своей любимой настольной игры.

Его любимая настольная игра — «Средние века». В этой игре каждый участник управляет своим королевством: развивается, расширяет территории, атакует королевства других игроков. На протяжении игры участники получают сундуки, из которых в зависимости от числа на игровом кубике, можно получить различные вещи и ресурсы, а самое главное — героев.

При получении карточки героя игрок должен выбрать одного из уже имеющихся у него героев, и назначить ему в подчиненные нового героя. Герои имеют параметр силы — целое число (возможно отрицательное), показывающее насколько успешно сражается герой.

При атаке или защите игрок ставит во главе армии своего игрового персонажа (героя с индексом 0), потом кидает игровой кубик, если на нем выпадает число k, то для каждого героя игрок может выбрать не более k героев, которые ему подчиняются и взять их в армию, при этом от каждого героя, двигаясь по иерархии вверх, можно достигнуть героя с индексом 0.



Итоговая сила собранной армии — это сумма всех параметров сил по всем выбранным героям. Например, на рисунке выше, у Пети есть 8 героев, стрелка, идущая от героя a к герою b, означает, что герой b подчиняется герою a. Числа рядом с вершинами — силы героев.

Тогда максимальная сила при $k = 2$ равняется 8 единицам.

Петя хочет выигрывать как можно чаще, поэтому он не должен ошибаться при сборе самой сильной армии. Помогите Пете и напишите программу, которая будет находить наибольшую силу армии, которую он может получить, если известны все имеющиеся у него герои, информация о том какой герой кому подчиняется, и число на игровом кубике.

Не волнуйтесь, это единственное, что от вас требуется, всем самым сложным займется сам Петя, например, тем как нужно оптимально подкидывать кубик и прочим другим.

Формат входных данных

В первой строке даны два целых числа n, k ($2 \leqslant n, k + 1 \leqslant 2 \cdot 105$) — количество героев в распоряжении у Пети и число, выпавшее на кубике. Во второй строке заданы $n - 1$ целых чисел p, где p_i — индекс героя, которому подчиняется герой с индексом $i + 1$. В третьей строке заданы n целых чисел v, где v_i ($|v_i| \leqslant 109$) — параметр силы героя с индексом i. Гарантируется, что $v_0 = 0$ — сила героя с индексом 0 равна нулю.

Формат выходных данных

Выведите единственное целое число — максимальную силу армии, которую может собрать Петя при имеющейся у него конфигурации героев и их связей между друг другом, при заданном значении на игровом кубике k.

Примеры

Пример №1

Стандартный ввод
15 2 0 0 0 1 3 1 6 6 6 4 5 5 12 12 0 2 6 -1 3 3 -4 3 5 2 1 5 0 2 -3
Стандартный вывод
19

Пример №2

Стандартный ввод
8 2 0 6 0 1 3 1 6 0 -1 -5 -10 9 4 -4 4
Стандартный вывод
8

Пример №3

Стандартный ввод
10 1 0 0 1 3 3 5 5 6 8 0 -20 14 -30 70 -10 -20 100 500 -1
Стандартный вывод
420

Решение

Эта задача на динамическое программирование по деревьям. Для каждой вершины `node` будем считать число `max_res_on_subtrees[node]` обозначающее ответ на поставленный в задаче вопрос, но не для всего дерева, а для поддерева, на котором вершина node самая верхняя, то есть герой node самый главный.

Тогда для того, чтобы вычислить `max_res_on_subtrees[node]` нужно, во-первых, принять изначально `max_res_on_subtrees[node] = node_values[node]`, а во-вторых, так как мы хотим получить самую сильную армию, надо взять k самых максимальных значений `max_res_on_subtrees[u]` среди всех детей u вершины `node`, тут также стоит учитывать, что нам выгодно брать только неотрицательные значения, поэтому берем или первые k максимальных или первые l ненулевых значений или все значения (т. к. детей может быть меньше k).

При таком алгоритме нам нужно, чтобы для каждой родительской вершины уже были посчитаны значения во всех дочерних, чтобы организовать такой порядок, сначала обойдем дерево обходом в ширину и каждую вершину занесем в ячейку массива с индексом равным ее глубине, считая от корня.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
import queue

def solve():
    def depth_found(root):
        depth[root] = 0
        depth_levels[0].append(root)

        q = queue.Queue()
        q.put(root)

        while not q.empty():
            node = q.get()
            for child in children[node]:
                if depth[child] == -1:
                    depth[child] = depth[node] + 1
                    depth_levels[depth[child]].append(child)
                    q.put(child)

    n, k = map(int, input().split())
    parents = list(map(int, input().split()))
    node_values = list(map(int, input().split()))

    children = [[] for i in range(n)]
    for node in range(n - 1):
        children[parents[node]].append(node + 1)

    depth = [-1 for i in range(n)]
    depth_levels = [[] for i in range(n)]

    depth_found(0)

    max_res_on_subtrees = [0 for i in range(n)]

```

```
    for nodes_on_depth_level in depth_levels[::-1]:
        for node in nodes_on_depth_level:
            max_res_on_subtrees[node] = node_values[node]

            max_res_on_child_nodes = [max_res_on_subtrees[child] for child in
            ↪  children[node]]
            max_res_on_child_nodes.sort(reverse=True)

            for first_k in range(min(len(max_res_on_child_nodes), k)):
                max_res_on_subtrees[node] += max(0,
                ↪  max_res_on_child_nodes[first_k])

    print(max_res_on_subtrees[0])

if __name__ == '__main__':
    solve()
```

Задача IV.10. Ярмарка изобретений (9 баллов)

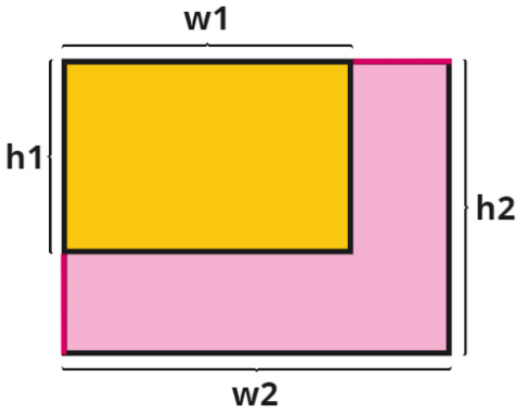
Темы: программирование, математика, перебор.

Данная задача проверяет навыки поиска оптимального решения.

Условие

В город приезжает ярмарка изобретений. Администрация решила, что она будет проходить на городском рынке, территория которого является прямоугольником со сторонами h_1, w_1 метров. Известно, что на ярмарку приедут n купцов, и каждый из них получит свою торговую площадь s.

Все n купцов поместятся на территории рынка только в том случае, когда суммарная площадь всех торговых площадей не больше площади рынка, то есть $ns \leqslant w_1 h_1$. Возможно, администрации придется арендовать на время ярмарки некоторую прилежащую к рынку территорию, увеличив некоторые из сторон на целое число метров так, чтобы форма территории рынка все еще оставалась прямоугольником с целыми длинами сторон.



Так как цена аренды напрямую зависит от арендуемой площади, администрация хочет минимизировать итоговую площадь рынка, при которой на ней поместятся все купцы. Определите эту площадь и конечные длины сторон рынка, при которых эта площадь получается.

Формат входных данных

В первой строке входных данных содержатся два целых числа n, s ($1 \leqslant n \leqslant 10^9$; $1 \leqslant s \leqslant 10$) — количество купцов и торговая площадь каждого купца. Во второй строке входных данных содержатся два целых числа h_1, w_1 ($1 \leqslant h_1, w_1 \leqslant 10^9$) — изначальные размеры сторон прямоугольной территории рынка.

Формат выходных данных

В первой строке выведите целое число — минимальную площадь рынка $s_{min} \leqslant s$, при которой поместятся все купцы. Во второй строке выведите два целых числа h_2, w_2 ($h_1 \leqslant h_2, w_1 \leqslant w_2$) — стороны рынка после расширения, для них должно выполнятся $h_2 w_2 = s_{min}$.

Примеры

Пример №1

Стандартный ввод
14 3 5 4
Стандартный вывод
42 7 6

Пример №2

Стандартный ввод
12 3 8 10
Стандартный вывод
80 8 10

Решение

В начале проверим, что при текущей площади рынка можно разместить всех купцов:

```
if h * w >= r:...
```

Если всех разместить нельзя, то давайте примем $h \leqslant w$:

```
swap = False
   if h > w:
       swap = True
       h, w = w, h
```

Если это не так, то в конце поменяем найденные значения, чтобы выполнялось $h_1 \leqslant h_2, w_1 \leqslant w_2$:

```
if swap:
    ans = ans[::-1]
```

Теперь будем перебирать сторону h, и для каждого значения h напрямую вычислять какую сторону w нужно взять, чтобы выполнялись все условия:

```
for take_h in range(h, math.ceil(math.sqrt(r)) + 3):
    need_w = max(w, (r + take_h - 1) // take_h)
```

Ключевым замечанием является то, что перебирать меньшую сторону h имеет смысл только до корня из ns, округленного в большую сторону.

Это позволяет добиться сложности $O(\sqrt{ns})$.

Далее идет проверка на то что мы нашли ответ получше, и вывод самого ответа:

```
if take_h * need_w < min_res:
    min_res = take_h * need_w
    ans = [take_h, need_w]

    print(min_res)
    if swap:
       ans = ans[::-1]

    print(*ans)
```

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1  import math
2
3  def solve():
4      n, s = map(int, input().split())
5      h, w = map(int, input().split())
6
7      r = s * n
8
9      if h * w >= r:
10         print(h * w)
11         print(h, w)
12         return
13
14     swap = False
15     if h > w:
16         swap = True
17         h, w = w, h
18
19     min_res = 1 << 60
20     ans = [0, 0]
21
22     for take_h in range(h, math.ceil(math.sqrt(r)) + 3):
23         need_w = max(w, (r + take_h - 1) // take_h)
24
25         if take_h * need_w < min_res:
26             min_res = take_h * need_w
27             ans = [take_h, need_w]
28
```

```
    print(min_res)
    if swap:
        ans = ans[::-1]

    print(*ans)

if __name__ == '__main__':
    solve()
```

Задача IV.11. Красивая гирлянда (9 баллов)

Темы: динамическое программирование, перебор.

Данная задача проверяет навыки динамического программирования.

Условие

Жители Умного города готовятся к фестивалю и украшают центральный проспект города. Вдоль проспекта они хотят разместить красивую гирлянду. Гирлянду жители считают красивой, если любые две соседние лампочки имеют разные цвета.

В распоряжении имеются лампочки n различных цветов, при этом k_i — число лампочек цвета i. Помогите жителям посчитать длину самой длинной красивой гирлянды и количество различных таких гирлянд. Две гирлянды различны, если они отличаются порядком лампочек.

Подзадача: найти наидлиннейшую гирлянду.

Формат входных данных

В первой строке содержится натуральное число n ($2 \leqslant n \leqslant 5$) — количество цветов. В следующих n строках содержатся натуральные числа k_i ($1 \leqslant k_i \leqslant 10$) — количество лампочек цвета i.

Формат выходных данных

Два целых числа через пробел — максимальная длина красивой гирлянды и количество красивых гирлянд максимальной длины. Так как количество таких гирлянд может быть очень большим — выведите его по модулю $10^9 + 7$.

Примеры

Пример №1

Стандартный ввод
2 10 10

Стандартный вывод
20 2

Пример №2

Стандартный ввод
3 10 3 2
Стандартный вывод
11 10

Гирлянды имеют вид

01010102020
01010201020
01010202010
01020101020
01020102010
01020201010
02010101020
02010102010
02010201010
02020101010

Решение

Идея – динамическое программирование.

Составляем словарь для цепей длины m, ключ – набор значений `d[0]...d[N-1]` (число лампочек всех цветов, уже использованных в гирлянде) и `color` – цвет последней лампочки. Начальный словарь – цепи длины 1. Цепи длины m получаются из цепей длины $m-1$ добавлением лампочки в конец, ее цвет i должен отличаться от `color`, а также `d[i]` уменьшается на 1, если это возможно.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
Num_color = int(input())
Color = []

for i in range(Num_color):
    Color.append(int(input()))

Chains = dict()
for i in range(Num_color):
    l = Color.copy()
    l[i] -= 1
    l.append(i)
    t = tuple(l)
    Chains[t] = 1

Num_chains = Num_color
max_len = 0
```

```
17
18  mod = 10 ** 9 + 7
19
20  while Num_chains > 0:
21      New_chains = dict()
22      count = 0
23      count_max_len = 0
24      max_len += 1
25
26      for elem in Chains:
27          count_max_len += Chains[elem]
28
29          for i in range(Num_color):
30              l = list(elem)
31              if l[i] > 0 and l[Num_color] != i:
32                  l[i] -= 1
33                  l[Num_color] = i
34                  t = tuple(l)
35
36                  if t in New_chains:
37                      New_chains[t] += Chains[elem]
38                  else:
39                      New_chains[t] = Chains[elem]
40                      count += 1
41
42      Chains = New_chains.copy()
43      Num_chains = count
44
45  print(max_len, count_max_len % mod)
```

Поиск наидлиннейшей гирлянды

Рассмотрим цвет ламп k_{max}, которых у нас больше всего, за количество лампочек других цветов кроме k_{max} примем s.

Утверждение. Наидлиннейшая гирлянда содержит или $2s + 1$ ламп, когда мы имеем очень много ламп с цветом k_{max} и не можем использовать их все, потому что они будут соседними — тогда мы чередуем лампы остальных цветов начиная с цвета k_{max} и заканчивая им. Или же используем все лампы k_{max} и все лампы с другими цветами, можно доказать, что мы можем использовать все лампы, в этом случает ответ — сумма всех ламп.

```
1  n = int(input())
2  color = [int(input()) for i in range(n)]
3
4  mx = max(color)
5  s = sum(color) - mx
6
7  res = min(s * 2 + 1, s + mx)
8
9  print(res)
```

Задача IV.12. План посещения мероприятий (9 баллов)

Темы: динамическое программирование, структуры данных.

Данная задача проверяет навыки работы со структурами данных и динамическим программированием.

Условие

На фестивале Умного города будет проводиться множество мероприятий на нескольких площадках. Петя и его друзья хотят увидеть, как можно больше мероприятий. При этом они не хотят опаздывать на какие-либо мероприятия или уходить раньше, чем закончится мероприятие, на которое они пришли.

Друзьям известно расписание мероприятий, а также время, которое они потратят при переходе от одной площадки до другой. Начинать посещение мероприятий можно с любой площадки, время, которое требуется, чтобы дойти до нее, в задаче не учитывается.

Помогите ребятам составить план посещения мероприятий, при котором они посетят их как можно больше.

Формат входных данных

В первой строке задано целое число K ($1 \leqslant K \leqslant 400$), обозначающее количество площадок. Далее идет K строк, в которых содержится расписание мероприятий.

Первое число в каждой строке, m_i — количество мероприятий на i-той площадке, затем идет m_i пар целых чисел, первое из которых — время начала мероприятия, второе — время окончания (таким образом, числа упорядочены по неубыванию), все числа не превосходят 10^9. Сумма m_i по всем $0 \leqslant i \leqslant K-1$ не превышает 1000.

Далее идут строки, описывающие временные затраты на перемещение между площадками, в формате abc, где a и b — номера площадок, $1 \leqslant c \leqslant 10^9$ — время перемещения между этими площадками в любом направлении. Таких строк $K(K-1)/2$.

Формат выходных данных

Два числа — максимальное количество мероприятий, которое можно посетить при этих условиях, и минимальное время окончания всех мероприятий.

Примеры

Пример №1

Стандартный ввод
3 3 1 10 14 25 45 78 2 0 27 29 35 2 5 7 48 63 0 1 4 0 2 5 1 2 10

Стандартный вывод
4 63

Примечание: друзья могут посетить два мероприятия на площадке 0, они длятся с 1 до 10 и с 14 до 25. Затем они переходят на площадку 1, потратив на это 4 единицы времени. Там они посещают мероприятие, которое продолжается с 29 до 35.

Затем они переходят на площадку 2, потратив на это 10 единиц времени, и посещают мероприятие с 48 до 63.

Решение

Идея — динамика.

В таблицу `T[i][j]` записываем минимальное время окончания мероприятия, если до него друзья посетили j мероприятий, и закончили на площадке i.

`A[i][0]` содержит количество мероприятий на площадке i.

`A[i][2m+1]` и `A[i][2m+2]` — начало и конец мероприятия с номером m.

`D[i][j]` содержит время пути от i до l площадки.

`T[i][0]=A[i][2]` — окончание первого мероприятия на площадке [i].

$$\texttt{T[i][j]=A[i][2m+2]:T[l][j-1]+D[l][i]} \leqslant \texttt{A[i][2m+1]}$$

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
N_places = int(input())
A = [[] for i in range(N_places)]
N_events = 0

for i in range(N_places):
    A[i] = list(map(int, input().split()))
    N_events += A[i][0]

D = [[0 for j in range(N_places)] for i in range(N_places)]
K = N_places * (N_places - 1) // 2

for i in range(K):
    a, b, c = map(int, input().split())
    D[a][b] = D[b][a] = c

T = [[1<<60 for j in range(N_events)] for i in range(N_places)]
max_events = 1
last = 1<<60

for i in range(N_places):
    T[i][0] = A[i][2]
    if last > T[i][0]:
        last = T[i][0]

for j in range(1, N_events):
    time = 1<<60
    for i in range(N_places):
        t_min = 1<<60

        for l in range(N_places):
            for k in range(1, A[i][0] * 2, 2):
                if A[i][k] >= T[l][j - 1] + D[l][i] and A[i][k + 1] <= t_min:
                    t_min = A[i][k + 1]
                    break

```

```
36          if t_min < 1<<60:
37              T[i][j] = t_min
38              max_events = j + 1
39              if time > t_min:
40                  time = t_min
41
42      if max_events == j + 1:
43          last = time
44      else:
45          break
46
47  print(max_events, last)
```