3. ЗАКЛЮЧИТЕЛЬНЫЙ ЭТАП

Предметный тур

4.1. Математика

Задача 4.1.1. (20 баллов)

Рядом с офисом IT-компании расположено несколько видеокамер, посредством которых сотрудники компании отслеживают количество своих посетителей. Записи с камеры №1 показали, что с понедельника до субботы включительно офис компании посетили 510 человек. Камера №2 зафиксировала 392 посетителя с понедельника по среду включительно, а по камере №3, работавшей во вторник и пятницу, насчитали 220 человек. Четвертая камера была включена в среду, четверг и субботу, ее показания составили 208 человек. Наконец, обработав показания камеры №5, удалось определить, что с четверга по субботу включительно за услугами компании обратилось 118 человек. Обработка данных со всех камер ведется автоматически и выдается в конце недели суммарным показателем за те дни, когда камеры работали. Определите, сколько посетителей побывало в офисе компании в понедельник?

Решение

Способ 1 Визуализируем условия задачи в виде таблицы посещаемости офиса компании:

–		Пн	Вт	Ср	Чт	Пт	Сб	Всего
1 камера		+	+	+	+	+	+	510
2 камера		+	+	+	–	–	–	392
3 камера		–	+	–	–	+	–	220
4 камера		–	–	+	+	–	+	208
5 камера		–	–	–	+	+	+	118

За исключение понедельника каждый день упоминается 3 раза. Это приводит к двойному учету посетителей четырьмя последними камерами во все дни кроме понедельника. Таким образом, искомое количество посетителей в понедельник можно найти из следующего выражения:

$$2 \cdot 510 - (392 + 220 + 208 + 118) = 1020 - 938 = 82$$

Способ 2 Составим систему уравнений:

$$\text{ПН} + \text{ВТ} + \text{СР} + \text{ЧТ} + \text{ПТ} + \text{СБ} = 510 \quad (4.1)$$

$$\text{ПН} + \text{ВТ} + \text{СР} = 392 \quad (4.2)$$

$$\text{ВТ} + \text{ПТ} = 220 \quad (4.3)$$

$$\text{СР} + \text{ЧТ} + \text{СБ} = 208 \quad (4.4)$$

$$\text{ЧТ} + \text{ПТ} + \text{СБ} = 118 \quad (4.5)$$

Решив систему уравнений, можно попытаться найти ответ. Однако, следует заметить, решение системы значительно упрощается, если обратить внимание на тот факт, что ответ можно получить, вычитая уравнения (3) и (4) из уравнения (1), т.е. ПН = 82.

Ответ: В офисе компании в понедельник побывало 82 человека.

Система оценки

Правильно составлена система уравнений или есть таблица учета наблюдений посещаемости офиса компании (5 баллов).

Получено правильное решение (15 баллов).

Задача 4.1.2. (5 баллов)

В цехе работают четыре электрика Пётр, Фёдор, Виталий и Леонид. Они устанавливают светильники. Утром на склад привезли четыре светильника, один светильник бракованный, остальные исправные. Виталий умеет пользоваться тестером светильников, а остальные электрики – нет. Первым пришёл Пётр и случайным образом выбрал светильник для установки, за ним случайно выбрал светильник Фёдор. Виталий забрал исправный светильник. Какова вероятность что Леонид установит бракованный светильник?

Решение

Способ 1 По условной вероятности. Во всех событиях, которые приводят к описываемому в условии результату – В забирает исправный светильник (с вероятность 1), поэтому его удаляем из рассотрения сразу. П – Петр берет исправный светильник (выбирая из 3 из которых 2 исправных, 1 нет), т.е. берет любой за исключением неисправного, вероятность события: $\left(1 - \frac{1}{3}\right)$, 2) Ф – то же самое при условии что на один исправный светильник в наборе меньше, вероятность $\left(1 - \frac{1}{2}\right)$, 3) $\overline{\text{Л}}$ – берет оставшийся неисправный светильник, вероятность 1. Вероятность события ПВФ$\overline{\text{Л}}$, при выполнении всех описанных условий равна $\frac{2}{3} \cdot \frac{1}{2} \cdot 1 = \frac{1}{3}$
Ответ: $= \frac{1}{3}$

Примечание. Вариант ответа $\frac{3}{4} \cdot \frac{2}{3} \cdot 1 \cdot 1 = \frac{1}{2}$ (дает ошибочный ответ, если не исключить сразу того, кто не выбирает лампочки)

Способ 2 2.1. Считаем варианты всех возможных событий методом перебора арифметически

П	1	2	3	1	2	3
Ф	2	3	1	3	1	2
В	3	1	2	2	3	1
Л	–	–	–	–	–	–

1,2,3 – исправные лампочки, – неисправная

6 благоприятствующих событий. Всего событий $6 \cdot (4-1) = 18$, так как события где В неисправная лампочка исключается.
Ответ: $\frac{6}{18} = \frac{1}{3}$

Примечание. Вариант простого перебора, при котором лампочки 1,2,3 неразличимы тоже может учитываться как правильный, поскольку при делении числа вариантов $n!$ (событий, в данном случае 3!) при подсчете вероятности сокращаются.

Способ 3 Рисуем дерево графов, считаем методом перебора геометрически.
Ответ: $\frac{6}{18} = \frac{1}{3}$

Система оценки

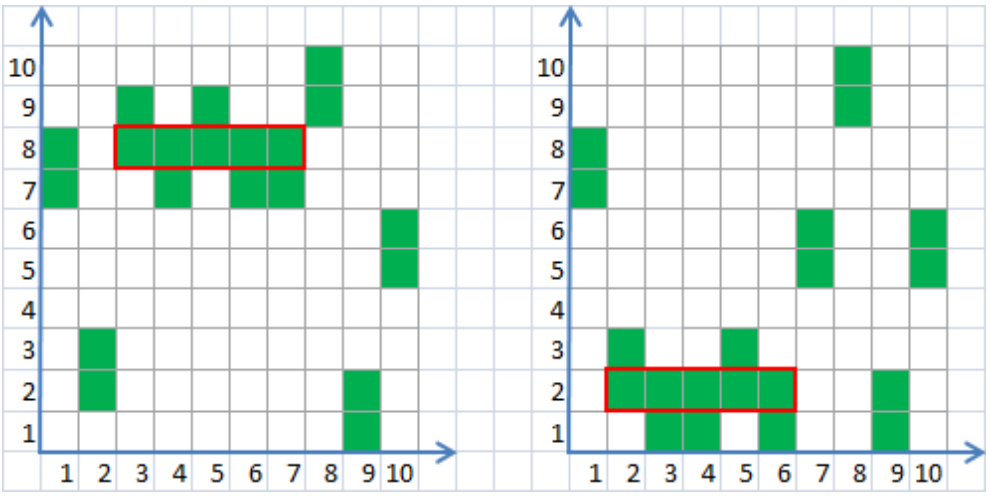
Определено и описано правильное событие (2 балла).

Правильно рассчитаны условные вероятности событий (2 балла).

Правильно получен ответ (1 балл).

Задача 4.1.3. (35 баллов)

Ребята играют в следующую игру. Последовательно делают 10 бросков доминошек на доску, причем доминошки могут падать только вертикально, как показано на рисунке. Ребята делают 10 бросков, по одному на каждую линию. Найдите вероятность того, что доминошки упадут таким образом, что через них можно будет провести горизонтальную прямую длиной 5 клеток. Примеры приведены на рисунке.



Решение

Вероятность линии в первой сроке (все доминошки должны опираться на линию 1), а так как события независимы, тогда: $p_1 = \left(\frac{1}{9}\right)^5$.

<table>
<tr><td>10</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>9</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>8</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td></tr>
<tr><td></td><td colspan="7">5-ти элементный блок № 1</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td colspan="6">5-ти элементный блок № 1</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="5">5-ти элементный блок №6</td></tr>
</table>

1. Вероятность линии во второй строке (доминошки могут стоять на линии 1 или на линии 2), так как события независимы:
$$p_2 = \left(\frac{2}{9}\right)^5.$$
2. Вероятности на линиях $3-9$: $p_3 = p_4 = p_5 = p_6 = p_7 = p_8 = p_9 = p_2$.
3. Вероятность появления линии в строке 10 означает, что доминошка стоит на линии 9, т.е. $p_{10} = p_1$.
4. Заполнение перовой линии, означает и заполнение второй и, следовательно, такой вариант уже учтен при расчете вероятности появления второй линии. Тоже утверждение справедливо для 10-й и 9-й линии.
5. Таким образом, вероятность появления 5-ти элементной линии в пятиэлементном блоке, равна сумме вероятностей: $p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8 + p_9 = 8 \cdot 2 = 8 \cdot \left(\frac{2}{9}\right)^5$.
6. Таких блоков 5-ти элементных блоков у нас 6. Таким образом, вероятность появления линии $6 \cdot 8 \cdot \left(\frac{2}{9}\right)^5$.

Ответ: $48 \cdot \left(\frac{2}{9}\right)^5$.

Система оценки

Правильно определены вероятности отдельных независимых событий (5 баллов).

Описана логика решения (5 баллов).

Правильно определены вероятности появления линий в 1 и 10 строках (10 баллов).

Правильно определены вероятности появления линий в 3 – 9 строках (10 баллов).

Правильно произведены расчеты и получен ответ (5 баллов).

Задача 4.1.4. (10 баллов)

Есть бесконечная решетка с квадратными ячейками, ширина прутьев 10 см, размер ячеек таков что куб со стороной 20 см проходит ячейку без зазоров. Решетка расположена горизонтально. Сверху вертикально вниз падают мячики, диаметр которых 15 см. количество мячиков 100000. Координаты мячиков выбираются случайным образом. Мячики, которые задевают решетку исчезают. Мячики выпускаются таким образом, что соударений между ними нет. Найти количество целых мячиков. Ответ округлить до целых.

Решение

Шарик пройдет решетку только в случае если его координаты его центра находятся в диапазоне $(\pm 2.5, \pm 2.5)$ см от центра ячейки. Т.е. окно безопасности равно $5 \cdot 5 = 25$ кв. см. Так как у нас есть N ячеек, есть N окон безопасности. Вероятность шарика попасть в любое окно пропорционально отношению площадей окна безопасности и площади ячейки с прутьями. Так как ширина прутьев 10 см, можно считать, что к каждой ячейке добавляются по 5 см с каждой стороны. Поэтому площадь ячейки с прутьями $30 \cdot 30 = 900$ кв. см.

$$p = N \cdot \frac{25}{N} \cdot 900 = \frac{25}{900} = 0.027778$$
$$N = p \cdot 100000 = 2777.8 = 2778$$

Ответ: 2778.

Система оценки

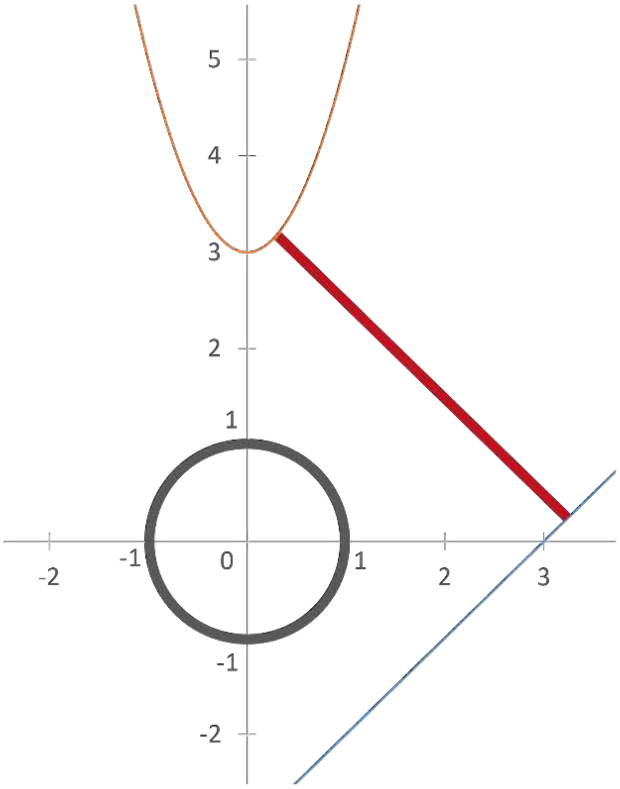
Дано соотношение вероятности мячика попасть (или миновать) в решетку (5 баллов)

Дан верный ответ (5 баллов).

Задача 4.1.5. (30 баллов)

Оптимизируйте расстояние для перекладки груза между автофургоном, перевозящем груз по трассе, описываемой прямой $y = x - 3$ и железнодорожными вагонами, проходящими вблизи данной трассы по параболе $y_1 = 2x^2 + 3$. В начале координат лежит город, радиусом 1 через который перевозки запрещены. Расстояния измеряются в км.

Решение



Способ 1 Согласно условию требуется найти наименьшее расстояние меду прямой $= x - 3$ и параболой $y_1 = 2x^2 + 3$ и проверить выполнение дополнительного условия. Выберем на параболе произвольную точку $M(x, 2x^2 + 3)$. Запишем функцию $f(x)$ расстояния между точкой, принадлежащей параболе и прямой $x - y - 3 = 0 (A = 1, B = -1, C = -3)$:

$$f(x) = \frac{|Ax + By + C|}{\sqrt{A^2 + B^2}} = \frac{|x - y - 3|}{\sqrt{2}} = \frac{|x - 2x^2 - 6|}{\sqrt{2}}$$

Для нахождения минимума функции, находим производную функции (модуль остается) и приравниваем к нулю.

$$f^{'}(x) = \frac{1}{\sqrt{2}}|1 - 4x| = 0$$

$$x = \frac{1}{4}$$

Проверим выполнение достаточного условия экстремума.

$$f^{''}(x) = \frac{1}{\sqrt{2}}|0 - 4| > 0$$

Выполняется для всех x, т .е. функция f(x) достигает минимума при $x = \frac{1}{4}$

$$f\left(\frac{1}{4}\right)_{min} = \frac{1}{\sqrt{2}}\left|\frac{1}{4} - \frac{1}{8} - 6\right| = \frac{47}{8\sqrt{2}}$$

Искомая дорога – перпендикуляр из этой точки на прямую, и он не пересекает окружности города.
Ответ: Оптимальное расстояние для перевоза груза $\frac{47}{8\sqrt{2}} \approx 4.15$ км

Способ 2 Минимальное расстояние – это нормаль к прямой, пересекающая параболу в некоторой точке M, координаты которой можно определить, записав уравнение касательной $f^{'}(x) = 4$ и потребовав, чтобы угол наклона касательной к этой точке соответствовал углу наклона исходной прямой $= x - 3$. Получим точку пересечения с абсциссой $x = 14$ Далее находим расстояние от точки M до прямой $478\sqrt{2} \approx 4.15$ км

Система оценки

Способ 1

Выписано уравнении от параболы до прямой (10 баллов).

Рассчитана первая производная и найдена критическая точка (5 баллов).

Произведена проверка на экстремум (5 баллов).

Правильно получен ответ и проверено условие не пересечения окружности города (10 баллов).

Способ 2

Сделан рисунок (5 баллов).

Доказано, что минимальное расстояние проходит по нормали (5 баллов).

Найдена точка пересечения и ее координаты (10 баллов).

Определено расстояние до прямой (10 баллов).

4.2. Информатика

Задача 4.2.1. Стевина на вас нет! (20 баллов)

Десятичные дроби, которыми мы пользуемся, не появились сами, а были изобретены — их придумал в конце 16 века математик Симон Стевин.

Чтобы найти десятичную дробь для числа, меньшего единицы, мы делим единичный отрезок на 10 одинаковых частей, и записываем номер отрезка (начиная с нуля), в который попало число. Затем этот отрезок снова делим на 10 частей, и так и далее, пока не получим достаточную точность. Если число больше единицы, то похожий процесс можно проводить и в обратную сторону, тогда мы найдём десятичные цифры слева от десятичной запятой.

В десятичных дробях, к которым мы привыкли, все цифры, стоящие на фиксированной позиции, отвечают за отрезки одинаковой длины, а позиция цифры отвечает за масштаб — при смещении вправо по записи числа масштаб увеличивается в 10 раз на один шаг.

Но числам не обязательно быть такими! Каждая цифра может быть особенной и отвечать за отрезок своей, особой длины!

Выберем следующие 10 цифр особенными:

0	1	2	3	4	5	6	7	8	9
$e/12$	$\pi/22$	e/π^3	$0,04$	$1/31$	$\sqrt{1/999}$	$\sqrt{\pi}/48$	$1/7$	$0,2$	остаток от единичного отрезка

(все цифры в таблице — нормальные, безликие десятичные). Вам нужно будет сконвертировать набор из 20 десятичных чисел в особенные.

Формат входных данных

20 вещественных чисел, по одному в каждой строке.

Формат выходных данных

20 чисел в такой же последовательности, но записанные в особенной форме.

Решение

Для чисел, меньших единицы, цифры находятся таким же процессом, что и для десятичных: делением отрезка на десять и нахождением, в какой попадает число. Отличием является только то, что длина отрезка уменьшается на как степени 10, а менее предсказуемо, что, однако, на работу такого алгоритма не влияет.

Для чисел, больших единицы, нужно найти, сколько степеней «десяти» в них помещается. Нетривиальный момент в том, что «единица» составляет не 1/10, а 0.2265234857 от «десяти». Соответственно, основание искомой степени составляет не 10, а 4.41455329406. Нужно найти, при делении на какую степень 4.41455329406-ти число становится меньше единицы, разделить его на 4.41455329406 в этой степени, преобразовать число, а затем сдвинуть десятичную запятую вправо на число символов, равное найденной степени.

Пример программы-решения

Ниже представлено решение на языке Python3

```
from math import e, pi, sqrt, trunc, log
import random

snowflakes = [ e / 12
             , pi / 22
             , e / ( pi ** 3 )
             , 0.04
             , 1/31
             , sqrt ( 1/999 )
             , sqrt (pi) / 48
```

```
              , 1/7
              , 0.2
              ]

snowflakes.append(1-sum(snowflakes))

def partSum(i):
    return sum(snowflakes[:i+1])

def findSmallDigitAndReduce(n):
    for i in range(10):
        if n < partSum(i):
            return (n-partSum(i-1),i)
    return None

def findLowerDigits(x,stahp):
    current = x
    result = []
    for _ in range(stahp):
        (newX,digit) = findSmallDigitAndReduce(current)
        current = newX / snowflakes[digit]
        result.append(digit)
    return result

onepow = 1 / snowflakes[0]

def findOrder(x):
    if x < 0:
        return 0
    else:
        return 1+trunc(log(x,onepow))

def gen():
    x = random.random()
    y = random.random()
    if y > 0.5:
        return x
    else:
        return 1/x

def generate():
    tests = []
    for _ in range(20):
        case = str(gen())
        tests.append(case)
    return "\n".join(tests) + "\n"

def solve(dataset):
    ones=dataset.splitlines(False)
    result = []
    for d in ones:
        result.append(d)
    return "\n".join(result)

def solve1(data):
    digits = data.replace(",",".")
    number = eval(digits)

    pw = findOrder(number)
    scaled = number / ( onepow ** pw )
```

```
    digits = findLowerDigits(scaled,10+pw)

    before = "".join(map(str,digits[:pw]))
    if len(before) == 0:
        before = "0"
    after  = "".join(map(str,digits[pw:]))

    return before + "." + after

def check(reply,clue):
    delta = 1e-10
    ours = [float(x) for x in clue.splitlines(False)]
    their = [float(x.replace(",",".")) for x in reply.splitlines(False)]
    if len(ours) != len(their):
        return False
    for i in range(len(ours)):
        if abs(their[i]-ours[i]) > 1e-10:
            return False
    return True

#Пример решения
#Можно получить из проверяющего добавлением к нему
#строки (если входные данные лежат в файле "in.txt".

with open("in.txt") as fin:
    data = fin.read()
    print(solve(data))
```

Ответ: Для решения нужно верно преобразовать все 20 случайно заданных чисел.

Задача 4.2.2. Трудный выбор (10 + 14 баллов)

Вы отвечаете за техническое обслуживание и модернизацию местной электростанции. Пришло время очередной модернизации, и, открыв корпоративную почту, вы увидели набор дорогостоящих предложений по модернизации отдельных модулей электростанции.

С помощью специалистов вы определили вероятность отказа модуля за расчётный период без модернизации и с ней для каждого предложения. Вероятности независимы между предложениями, но в случае отказа одного из модулей регистрируется авария на всей электростанции.

Вам необходимо минимизировать вероятность аварии, при этом уложившись в бюджет на модернизацию.

Подсчитайте, во сколько раз уменьшится вероятность аварии при наиболее рациональном наборе принятых предложений.

Формат входных данных

Первая строка — количество предложений $N (7 \leqslant N \leqslant 15)$ и бюджет M в млн.руб. $(70 \leqslant M \leqslant 120)$ через пробел. Далее N строк с тремя числами через пробел: стоимость предложения в млн. руб., вероятность отказа модуля без реализации и после реализации предложения.

Формат выходных данных

Единственное число — во сколько раз уменьшится вероятность аварии. Допустимо расхождение не более, чем на 0.01

Система оценки

Задача имеет два варианта, один с небольшим размером входных данных и лимитами времени выполнения, достаточными для решения задачи перебором. Второй вариант имеет размеры входных данных и лимиты времени выполнения, исключающие перебор. За решение первого варианта: 10 баллов, за решение второго — 14 баллов.

Пример №1

Стандартный ввод
7 99 28 0.2521 0.1965 11 0.3811 0.2454 9 0.4621 0.03367 12 0.4761 0.3627 5 0.3171 0.0548 30 0.4007 0.3879 34 0.2658 0.2053
Стандартный вывод
4.401

Решение

Это задача о рюкзаке с мультипликативной целевой функцией. Каждый вариант преобразуется к 1/<уменьшение вероятности отказа>, находится вектор, максимизирующий данную величину, затем преобразуется в ответ. Кроме владения техникой решения задач типа «рюкзак» и умения их распознавать, нужно также владение теорией вероятности на уровне понимания независимых и дополнительных событий.

Пример программы-решения

Ниже представлено решение на языке Python3

```
1  import random
2  import sys
3
4  random.seed(6162)
5  SUITE_SIZE = 10
6
7  def generateTask():
8      result = []
9      resultSum = 0
10     variantsCount = random.randint(7,10)
11     for _ in range(variantsCount):
```

```
12          (m,p,a) = genVariant()
13          result.append((m,p,a))
14          resultSum += m
15      return resultSum, len(result), result
16
17  def printTask():
18      s, l, r = generateTask()
19      pass
20
21  def readTask(data):
22      ps, *items_raw = data.splitlines(False)
23      n, w = map(int, ps.split())
24      items = [(int(p), a, b) for (p, a, b) in (i.split() for i in items_raw)]
25      return items, w
26
27  def genVariant():
28      money = random.randint(5,35)
29      probprev = 0.2 + 0.2 * random.random()
30      probafter = 0.9 + 0.1 * random.random()
31      return (money,probprev,probafter)
32
33  def vector(size,num):
34      result = []
35      for i in range(size):
36          result.append(((1 << i ) & num)>0)
37      return result
38
39  def countVariant(variants,n):
40      prod = 1
41      summ = 0
42      l = len(variants)
43      for i in range(l):
44          (m,p,a) = variants[i]
45          if vector(l,n)[i]:
46              prod *= (1-float(a))
47              summ += int(m)
48          else:
49              prod *= (1-float(p))
50      newFail = 1 - prod
51      return summ, newFail
52
53  def all(variants,limit):
54      bestProb = 1
55      for i in range(2 ** len(variants)):
56          score, fail = countVariant(variants,i)
57          if score <= limit and fail < bestProb:
58              bestProb = fail
59      return bestProb
60
61  def generate_item():
62      price = random.randint(5,35)
63      x = random.random() * 0.2 + 0.2
64      y = x * (random.random() * 0.9 )
65      return "{} {:.4} {:.4}".format(price, x, y)
66
67  def generate_one():
68      item_n = random.randint(7, 15)
69      money = random.randint(100, 200)
70      items = "\n".join(generate_item() for _ in range(item_n))
71      return "{} {}\n{}\n".format(item_n, money, items)
```

```
72
73  def generate():
74      return [generate_one() for x in range(SUITE_SIZE)]
75
76  def solve(dataset):
77      #print(dataset)
78      s,lim = readTask(dataset)
79      _,p0 = countVariant(s,0)
80      pMax = all(s,lim)
81      #print(p0,pMax)
82      return str(p0/pMax)
83
84  def check(reply, clue):
85      try:
86          your = float(reply)
87          mine = float(clue)
88          return abs(your-mine) <= 0.01
89      except Exception as e:
90          return False, "ошибка формата"
91
92  x1 = '7 91\n34 0.3694 0.1881\n6 0.3967 0.05904\n29 0.3629 0.3279\n18 0.4685 \
93        0.3988\n 7 0.251 0.2173\n15 0.4967 0.4916\n19 0.3923 0.09796\n'
94  x2 = '30 320\n33 0.3824 0.1641\n27 0.28 0.2045\n8 0.2727 0.03715\n16 0.3769 0.2285\n \
95        31 0.2833 0.07882\n10 0.4299 0.3793\n19 0.3323 0.3034\n29 0.4066 0.3941\n10 \
96        0.4703 0.07739\n27 0.3413 0.1661\n14 0.258 0.0139\n26 0.2661 0.1518\n29 0.3395 \
97        0.291\n16 0.2516 0.1018\n13 0.2779 0.1294\n31 0.4401 0.3566\n14 0.2754 \
98        0.07968\n 33 0.3246 0.3212\n23 0.4771 0.3942\n13 0.4212 0.2745\n19 0.3201 \
99        0.2555\n5 0.4157 0.006087\n34 0.4888 0.4167\n30 0.4451 0.06692\n16 0.2776 \
100       0.02759\n12 0.2625 0.0521\n21 0.4434 0.1725\n35 0.4749 0.2688\n25 0.4311 \
101       0.202\n20 0.3104 0.2109\n'
102
103 print(solve(sys.stdin.read()))
```

Задача 4.2.3. Некубические кубики (13 баллов)

Привычный нам шестигранный кубик при броске выдает любое число очков на верхней грани с равной вероятностью (1/6).

В ваше же распоряжение попали необычные кубики, у которых вероятности выпадения для каждой грани отличаются от привычной 1/6.

Подсчитайте вероятность того, что в результате броска на верхних гранях в сумме будет 21 точка.

Формат входных данных

6 строк, в каждой через пробел по 6 вещественных чисел $p_{ij} (1 \leqslant i, j \leqslant 6)$, представляющих вероятность выпадения j точек на i-м кубике. Гарантируется, что распределения для кубиков корректны, т.е. $\sum_{j-1}^{6} a_{ij} = 1 (1 \leqslant i \leqslant 6)$.

Формат выходных данных

Единственное вещественное число — вероятность выпадения 21 точки на всех кубиках. Допускается расхождение с ответом не более, чем на 0.00001

Система оценки

Пример №1

Стандартный ввод
0.11195 0.25259 0.0337 0.39412 0.16683 0.04081 0.09433 0.1281 0.17695 0.11216 0.31335 0.17511 0.04646 0.07009 0.13142 0.43325 0.14784 0.17094 0.18595 0.25611 0.2401 0.11454 0.08692 0.11638 0.18818 0.12266 0.09079 0.02577 0.51837 0.05423 0.00422 0.24524 0.09319 0.11028 0.35783 0.18924
Стандартный вывод
0.123123

Решение

Нужно написать программу, которая сначала находит всех комбинаций 6 кубиков, имеющие сумму 21, а затем вычислить и сложить вероятности выпадения всех этих комбинаций.

Пример программы-решения

Ниже представлено решение на языке Python3

```
import random
import sys

SEED = 6161
NUM_CUBES = 6
SUITE_SIZE = 20
DESIRED_NUMBER = 21

random.seed(SEED)

def generate_cube():
    c = []
    for x in range(5):
        c.append(round(random.random() * 0.3 * (1 - sum(c)), 5))
    c.append(1-sum(c))
    random.shuffle(c)
    return c

def generate_one():
    cubes = "\n".join(" ".join("{:.5}".format(x) for x in generate_cube())
        for _ in range(NUM_CUBES))
    return cubes

def generate():
    return [generate_one() for x in range(SUITE_SIZE)]

def solve(data):
    cubes = [[float(x) for x in l.split()] for l in data.splitlines(False)]
    ans = 0

    def looper(acc, i, p):
        nonlocal ans
        if i == NUM_CUBES:
```

```
34              return
35          for x in range(1, 6 + 1):
36              if acc + x == DESIRED_NUMBER and i == NUM_CUBES - 1:
37                  ans += p * cubes[i][x-1]
38                  return
39              if acc + x > DESIRED_NUMBER:
40                  return
41              looper(acc+x, i+1, p * cubes[i][x-1])
42
43      looper(0, 0, 1)
44      return str(ans)
45
46  def check(reply, clue):
47      try:
48          your = float(reply)
49          mine = float(clue)
50          return abs(your-mine) <= 0.00001
51      except Exception as e:
52          return False, "ошибка формата"
53
54  def sanity(fn):
55      tests = generate()
56      ms = [solve(test) for test in tests]
57      ys = [fn(test) for test in tests]
58      flag = False
59      for y, m, t in zip(ys, ms, tests):
60          if not check(y, m):
61              print("BAD TRY>>>\n{}\n>>>\nYOUR: {}\nMINE: {}".format(t, y, m))
62              flag = True
63      if flag:
64          print("TEST FAILED :(")
65      else:
66          print("ALL RIGHT :D")
67
68  print(solve(sys.stdin.read()))
```

Задача 4.2.4. Тьюринг 2D (6 баллов)

Дан прототип двумерной машины Тьюринга, описывающей программу движения курсора по полю. Программа представлена матрицей из N строк и M столбцов. В каждой клетке матрицы, соответствующей определённой точке поля, содержится команда перемещения курсора: l — влево на 1 клетку r — вправо на 1 клетку u — вверх на 1 клетку d — вниз на 1 клетку

Поле зациклено, при переходе через границу поля курсор выходит с другой его стороны.

В начале выполнения курсор установлен в точке $(0, 0)$.

Очевидно, что рано или поздно выполнение программы зациклится. Вам нужно определить, в какой точке матрицы начинается цикл.

Формат входных данных

Первая строка — количество строк M и столбцов $N (5 \leqslant M, N \leqslant 20)$ через пробел. Далее M строк длиной N, содержащих символы из множества {l,r,u,d}.

Формат выходных данных

Два числа через пробел — строка и столбец ячейки, где начинается цикл. Нумерация начинается с нуля!

Пример №1

Стандартный ввод
4 4 rrrd ldll urru rlrl
Стандартный вывод
1 3

Решение

Решение рассчитано на простую реализацию. Храним посещённые клетки в булевой матрице. Начинаем с (0, 0), читаем символ по этим координатам. Вправо — увеличение столбца, вниз — увеличение строки. Вышли за пределы матрицы — присвоили координаты с другого конца. Попали на посещённую ранее клетку — выводим её координаты и завершаем выполнение.

Пример программы-решения

Ниже представлено решение на языке Python3

```
inp = """4 4
rrrd
ldll
urru
rlrl
"""

import random
import sys

SEED = 6161
SUITE_SIZE = 20

random.seed(SEED)

def generate_one():
    m = random.randint(5, 20)
    n = random.randint(5, 20)
    field = "\n".join("".join(random.choice("uldr") for _ in range(n))
        for _ in range(m))
    return "{} {}\n{}\n".format(m, n, field)

def generate():
    return [generate_one() for x in range(SUITE_SIZE)]

```

```
def solve(data):
    sizes, *lines = data.splitlines(False)
    m, n = map(int, sizes.split())
    vs = [[False for _ in range(n)] for _ in range(m)]
    x, y = 0, 0
    while True:
        if vs[y][x]:
            return("{} {}".format(y, x))
        vs[y][x] = True
        d = lines[y][x]
        if d == 'l':
            x, y = (x-1)%n, y
        elif d == 'r':
            x, y = (x+1)%n, y
        elif d == 'u':
            x, y = x, (y-1)%m
        elif d == 'd':
            x, y = x, (y+1)%m
        else:
            raise Exception("what?!")

def check(reply, clue):
    try:
        your = [int(x) for x in reply.split()]
        mine = [int(x) for x in clue.split()]
        return your == mine
    except Exception as e:
        return False, "ошибка формата"

def sanity(fn):
    tests = generate()
    ms = [solve(test) for test in tests]
    ys = [fn(test) for test in tests]
    flag = False
    for y, m, t in zip(ys, ms, tests):
        if not check(y, m):
            print("BAD TRY>>>\n{}\n>>>\nYOUR: {}\nMINE: {}".format(t, y, m))
            flag = True
    if flag:
        print("TEST FAILED :(")
    else:
        print("ALL RIGHT :D")

print(solve(sys.stdin.read()))
```

Задача 4.2.5. Посади электродерево (37 баллов)

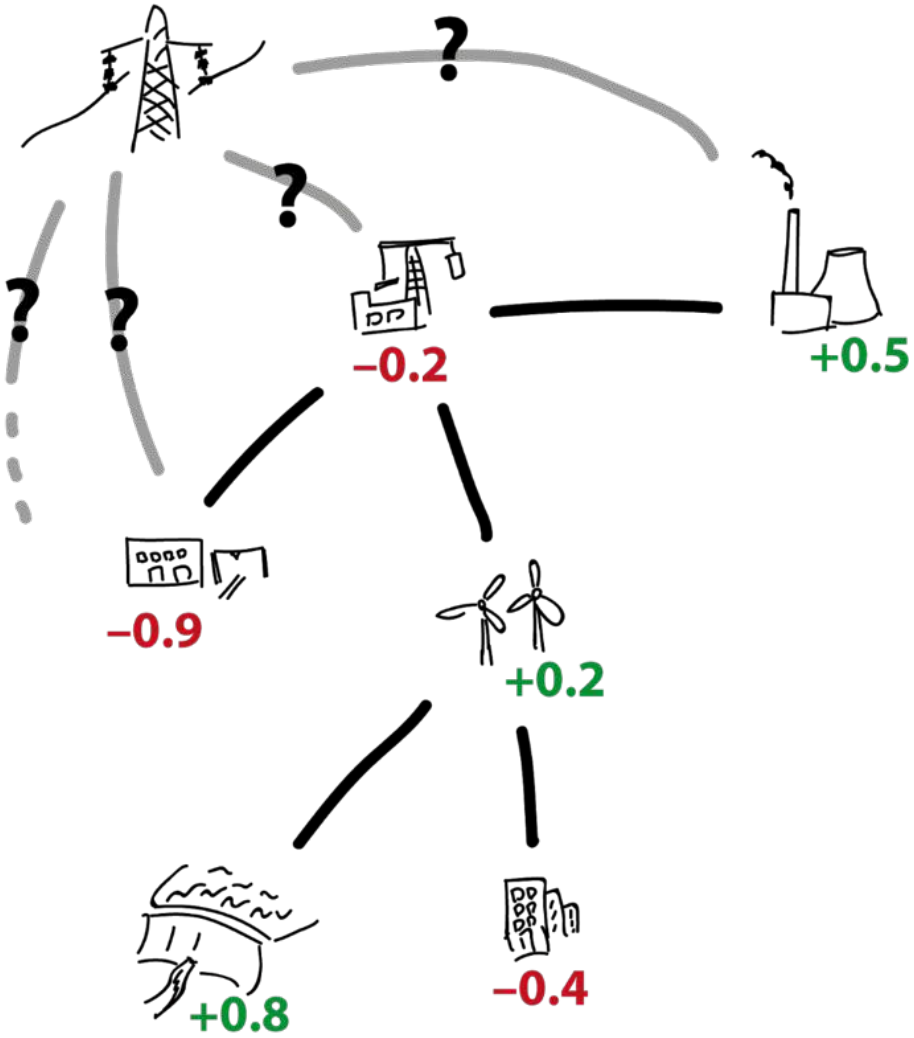
Внимание! Эта задача весьма сложна, поэтому беритесь за неё, только если очень сильно уверены в том, что её решите, или после того, как решите остальные задачи.

В вашем распоряжении имеется энергосистема из генераторов и потребителей, объединённых в сеть топологии «дерево».

Мощность потребления/генерации каждого потребителя известена (она находится в пределах от -1 до 1).

Известен коэффициент потерь на каждой линии, соединяющей объекты энергосистемы, R (от 0.01 до 0.02).

Потери на линии можно вычислить по формуле: $Q = P^2 \times R$, где P — мощность, передаваемая по линии, а R — коэффициент её потерь.



Энергосистема не сбалансирована (потребление не равно генерации), и вы выбираете место для подключения сети к внешней энергосистеме для её балансировки. Его нужно выбрать так, чтобы суммарные потери электроэнергии в энергосистеме стали минимальными.

Формат входных данных

Дерево, представленное списками вершин (узлов) и рёбер (линий).

Для каждого списка сначала приводится его длина, затем элементы на отдельных строках.

Каждый узел представлен номером и потреблением, записанными через пробел.

Каждая линия — двумя номерами узлов и сопротивлением, также записанными через пробел.

Все элементы нумеруются с единицы, а не с нуля.

Формат выходных данных

Число — номер узла, в котором нужно установить подключение к внешней энергосистеме.

Пример №1

Стандартный ввод

```
20
1 0.024163778035099635
2 0.19565699001037376
3 -0.7637821782044889
4 -0.028226916356001587
5 0.8697284181820586
6 0.3733013894822612
7 0.6840817064112741
8 -0.6131926407131862
9 0.48967147884270656
10 -0.15099795336447938
11 0.2811479872131111
12 -0.26269659300009707
13 0.5398361561230487
14 -0.882166220101082
15 0.42636847196784555
16 0.14358218753248633
17 -0.6419599456437441
18 0.27050898163756587
19 -0.17770902338537795
20 -0.7773160746693744
19
1 2 0.01535068537697453
2 3 0.019389261888600864
2 4 0.014736532285625591
4 5 0.01790151673588955
1 6 0.01161291996565344
4 7 0.013004263512249842
5 8 0.011458276350586605
4 9 0.015554695564785236
3 10 0.01329141854582266
9 11 0.01567961098227486
3 12 0.018451984551052913
3 13 0.01511693727529641
3 14 0.01123448474818787
2 15 0.014035505859012452
10 16 0.016065517014648877
10 17 0.011598946777423888
12 18 0.013050336805736836
15 19 0.01904939072374451
10 20 0.01097086560277972
```

Стандартный вывод

```
11
```

Решение

Перебираются все варианты установки подключения, для каждого вычисляются потери в энергосистеме. Потери вычисляются свёрткой дерева к корню — рассматриваемой вершине. Свёртка листа происходит следующим образом: вычисляются потери от передачи энергии от листа к вышестоящему узлу (передаваться может и отрицательное значение). Модуль передаваемой мощности уменьшается на это число, после чего оставшаяся мощность добавляется к значению вышестоящей вершины. Потери добавляются к общим потерям, лист удаляется.

Пример программы-решения

Ниже представлено решение на языке Python3

```
from copy import copy, deepcopy
import random
import sys

random.seed(99)

# Генерирует дерево как набор рёбер
def genTree(size):
    tree = []
    for i in range(1,size):
        to = random.randint(1,i)
        tree.append((to,i+1))
    return tree

# Каждому ребру задаёт коэффициент потерь
def addLosses(graph):
    new = []
    for (a,b) in graph:
        weight = random.uniform(0.01,0.02)
        new.append((a,b,weight))
    return new

# Делает граф с потерями
def makeGraph(base):
    return addLosses(genTree(base))

# Генерирует набор нод с потреблением/генерацией
def makeNodes(size):
    nodes = []
    sum = 0
    for i in range (size):
        x = random.uniform(-1,1)
        sum += x
        nodes.append((i+1,x))
    sum /= size
    zeroed = []
    for (n,x) in nodes:
        zeroed.append((n,x-sum))
    return zeroed

# Генерирует пару (ноды, рёбра с потерями)
def makeTask(base):
    nodes = makeNodes(base)
    graph = makeGraph(base)
```

```
45      return (nodes,graph)
46
47  # Делает из нод и рёбер с потерями полную таблицу связности
48  def makeTable(nodes,graph):
49      links = [[ None for i in range(len(nodes))] for j in range(len(nodes))]
50      for (a,b,c) in graph:
51          links[a-1][b-1] = c
52          links[b-1][a-1] = c
53      return links
54
55  # Возвращает None, если не лист, иначе возвращает родителя
56  def isLeaf(node,table,root):
57      edges = 0
58      uplink = None
59      for i in range(len(table)):
60          if table[i][node-1] == None or node == root:
61              pass
62          else:
63              edges += 1
64              uplink = i+1
65      if edges == 1:
66          return uplink
67      else:
68          return None
69
70  # Съедает дерево
71  def runTree(nodes_,table_,root):
72      table = deepcopy(table_)
73      nodes = deepcopy(nodes_)
74      allLosses = 0
75      for _ in range(len(table)):
76          for i in range(len(table)):
77              uplink = isLeaf(i+1,table,root)
78              if uplink == None:
79                  pass
80              else:
81                  (a,b) = nodes[uplink-1]
82                  (c,d) = nodes[i]
83                  losses = min(abs(d),d**2 * table[i][uplink-1])
84                  #if losses == abs(d):
85                      #print("Alarm:",d)
86                  arrived = d / abs(d) * ( abs(d) - losses )
87                  allLosses += losses
88                  nodes[uplink-1] = (a,b+arrived)
89                  table[i][uplink-1] = None
90                  table[uplink-1][i] = None
91      return allLosses
92
93  def solution(nodes,edges):
94      #(loss,nodes,edges,mapp) = shrink(nodes_,edges_)
95      #loss = 0
96      #nodes = deepcopy(nodes_)
97      #edges = deepcopy(edges_)
98      table = makeTable(nodes,edges)
99      roots = [ i+1 for i in range(len(table)) ]
100     best = float("inf")
101     guess = None
102     for r in roots:
103         que = runTree(nodes,table,r)
104         if que < best:
```

```
105             best = que
106             guess = r
107     #return(unMapEdges(mapp,guess),best+loss)
108     return(guess,best)
109 
110 def test():
111     (a,b) = makeTask(20)
112     print(a)
113     print("~~~~~~~~~~~~")
114     print(b)
115     print("~~~~~~~~~~~~")
116     print(solution(a,b))
117 #test()
118 
119 def format_task(task):
120     (nodes, edges) = task
121     return "{}\n{}\n{}\n{}\n".format(
122         len(nodes),
123         "\n".join("{} {}".format(*n) for n in nodes),
124         len(edges),
125         "\n".join("{} {} {}".format(*n) for n in edges),
126     )
127 
128 def generate():
129     num_tests = 10
130     tests = []
131     for test in range(num_tests):
132         task = makeTask(20)
133         test_case = format_task(task)
134         tests.append((test_case,task))
135     return tests
136 
137 def solve(dataset):
138     ls = dataset.splitlines()
139     nnodes = int(ls[0])
140     nodes = [(int(x), float(y)) for (x, y) in (l.split() for l in ls[1:nnodes+1])]
141     nedges = int(ls[nnodes+1])
142     edges = [(int(x), int(y), float(z)) for (x, y, z) in
143         (l.split() for l in ls[nnodes+2:nnodes+nedges+2])]
144 
145     (root,score) = solution(nodes,edges)
146     return str(root)
147 
148 def check(reply, clue):
149     (nodes,edges) = clue
150     table = makeTable(nodes,edges)
151     (my_root,my_score) = solution(nodes,edges)
152     root = eval(reply)
153     return runTree(nodes,table,root) + 0.01 > my_score
154 
155 def sanity(fn):
156     tests = generate()
157     flag = False
158     for (t, c) in tests:
159         y = fn(t)
160         if not check(y, c):
161             m = solve(t)
162             print("BAD TRY>>>\n{}\n>>>\nYOUR: {}\nMINE: {}".format(t, y, m))
163             flag = True
164     if flag:
```

```
165         print("TEST FAILED :(")
166     else:
167         print("ALL RIGHT :D")
168 
169 
170 print(solve(sys.stdin.read()))
```