1. Медали (medalid)

2 сек / 10 сек 10 очков

Олег участвовал в нескольких сериях спортивных состязаний. В каждой серии соревнования имеют порядковый номер, например, “36-ой марафон Kükametsa в беге на финских санях”.

На каждом соревновании каждому участнику выдаётся медаль. На каждой медали написан порядковый номер соревнования в своей серии, а в остальном медали идентичны.

Олег участвовал во многих соревнованиях, и у него накопился большой мешок медалей. Определить наименьшее возможное количество различных серий, на которых он мог заработать эти медали.

Известно, что если Олег начинает участвовать в какой-либо серии, то он посещает все подряд соревнования этой серии, не пропуская ни одного. Таким образом, если он, например, был на 36-ом и 38-ом марафонах Kükametsa в беге на финских санях, то он обязательно участвовал и в 37-ом марафоне Kükametsa в беге на финских санях.

Входные данные. На первой строчке текстового файла `medalidsis.txt` указано количество медалей N ($1 \leqslant N < 10^6$), а на второй строчке N разделённых пробелами целых числа M_i ($1 \leqslant M_i \leqslant 1000$): порядковые номера соревнований, записанные на медалях.

Выходные данные. На первую строчку текстового файла `medalidval.txt` вывести наименьшее возможное количество серий состязаний V, а на следующие V строчек одно возможное распределение медалей по сериям. На каждую строчку вывести количество посещённых соревнований в одной из серий и номера медалей в этой серии в возрастающем порядке.

Пример.

`medalidsis.txt`
```
10
46 47 48 45 2 46 48 3 1 49
```

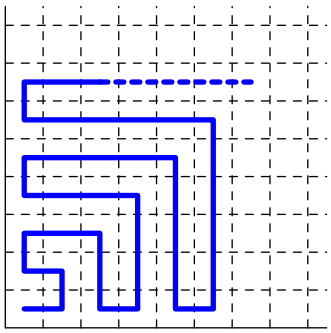
`medalidval.txt`
```
4
3 1 2 3
5 45 46 47 48 49
1 46
1 48
```

Оценивание. В тестах стоимостью 5 очков выполняется условие $1 \leqslant N \leqslant 100$.

2. Линия на бумаге (joon)

1 секунда 20 очков

На рисунке ниже показана ломаная линия, которую можно нарисовать, на каждом шаге передвигаясь на одну клетку.



Если на шаге 0 начинаем на клетке с координатами $(1; 1)$, то на какой клетке будем на шаге N?

Входные данные. На единственной строчке текстового файла `joonsis.txt` записано целое число N $(0 \leqslant N < 10^{18})$.

Выходные данные. На единственную строчку текстового файла `joonval.txt` вывести 2 разделённых пробелом целых числа, обозначающих X- и Y-координаты конечной клетки.

Пример.

joonsis.txt	joonval.txt
7	3 2

Оценивание. В тестах стоимостью 10 очков выполняется условие $N \leqslant 10^6$.

3. Кто хочет стать миллионером? (mil)

1 сек / 5 сек 30 очков

У компьютеров есть своя версия игры "Кто хочет стать миллионером?" В начале игры у игрока есть 1 евро и ему задаются 20 вопросов. Если текст вопроса содержит палиндром длиной 4 или более букв, то сумма игрока удваивается и к ней прибавляется длина самого длинного палиндрома, присутствующего в вопросе. В противном случае сумма не меняется.

Палиндромом называется строка, которая одинаково читается как слева направо, так и справа налево, например, "anna". В этой игре палиндром может быть образован и из нескольких слов, например, "... and name ..." содержит палиндром "andna". Палиндром также может быть образован из конца и начала вопроса, например, "Jane went walking with Jelena" содержит палиндром "enajane", а "Matti is I ttam" палиндром "ittammatti".

При поиске палиндромов не различают строчные и заглавные буквы. Пробелы и концы строк не учитываются, а вот знаки препинания учитываются.

Написать программу, которая зачитает вопросы и выведет финальную сумму, набранную игроком.

Входные данные. В текстовом файле `milsis.txt` ровно 20 отрывков текста в кодировке ASCII[1]. Отрывки отделены друг от друга пустыми строчками. Каждый отрывок может состоять из не более чем 600 знаков.

Выходные данные. На единственную строчку текстового файла `milval.txt` вывести сумму игрока в конце игры.

Пример.

`milsis.txt`

```
anna

title and name of the author

Jane went walking with Jelena

Matti is I ttam

lorem ipsum

Matti is I'ttam.

Jane went
walking
with Jelena
```

`milval.txt`

```
393
```

Четыре первых вопроса соответствуют приведённым примерам. То есть, за первый вопрос игрок получит из первоначального 1 евро $2 \cdot 1 + 4 = 6$, затем $2 \cdot 6 + 5 = 17$, затем $2 \cdot 17 + 7 = 41$ и затем $2 \cdot 41 + 10 = 92$. В тексте пятого вопроса палиндрома нет, поэтому сумма не меняется. Шестой вопрос содержит палиндром "ttam.matt", поэтому новой суммой станет $2 \cdot 92 + 9 = 193$. Седьмой вопрос такой же как третий, только текст разбит на несколько строчек, поэтому новой суммой станет $2 \cdot 193 + 7 = 393$. Следующие вопросы (которые слишком длинные для того, чтобы привести их в тексте задания, но видны в файле примера, доступного с тестового сервера) не содержат палиндромов, поэтому конечной суммой игрока в этой игре остаётся 393.

Оценивание. В тестах, дающих 10 очков, каждый вопрос находится на одной строчке

[1] `https://ru.wikipedia.org/wiki/ASCII`

и ни один палиндром не разделён между началом и концом вопроса. В тестах, дающих следующие 10 очков, ни один палиндром не разделён между началом и концом вопроса. В тестах, дающих последние 10 очков, дополнительных ограничений нет.

4. Делители факториала (fact)

1 сек / 5 сек 40 очков

Определим для любого положительного целого n функцию $s(n)$ как наименьшее положительно целое m, факториал[1] которого делится на n.

Например:

$s(1) = 1$,
$s(2) = 2$, (так как 1! (=1) не делится на 2, но 2! (=2) делится)
$s(4) = 4$, (3! (=6) не делится на 4, а 4! (=24) делится)
$s(6) = 3$, (3! (=6) делится на 6)
$s(9) = 6$, (6! (=720) делится на 9)
$s(10) = 5$, итд.

Задача состоит в том, чтобы для двух заданных целых чисел A и B вычислить сумму:

$$s(A) + s(A+1) + \ldots + s(B).$$

Входные данные. На единственной строке текстового файла `factsis.txt` даны два разделенных пробелом числа A и B ($1 \leqslant A \leqslant B \leqslant 1\,000\,000$).

Выходные данные. На единственной строке текстового файла `factval.txt` необходимо вывести искомую сумму.

Пример.

`factsis.txt`	`factval.txt`
`5 10`	`30`

Ответ равен $s(5) + s(6) + s(7) + s(8) + s(9) + s(10) = 5 + 3 + 7 + 4 + 6 + 5 = 30$.

[1] Факториал положительного целого числа m обозначается как $m!$ и означает произведение всех целых чисел от 1 до m:

$$m! := 1 \cdot 2 \cdot 3 \cdot \ldots \cdot m.$$

5. Логистика (`logi`)

6 сек / 30 сек 50 очков

Всемирно известная промышленная фирма Universal Manufacturing недавно решила расширить свою деятельность. Фирма производит множество разных товаров, от ламп до тракторов, и устраивает весь производственный процесс, от сырья до конечного продукта, на своих заводах.

Все заводы фирмы пронумерованы $1 \dots N$, причём главный завод имеет номер 1. Заводы соединены между собой с помощью $N - 1$ дорог, и известно, что из главного завода можно по этим дорогам добраться до любого другого завода.

У каждого завода i есть свой уровень T_i, который означает, что завод может производить как компоненты этого уровня для других заводов, так и готовые продукты этого уровня. Завод может использовать в качестве заготовок компоненты любого более низкого уровня, но уровень результата его деятельности всегда будет T_i.

Фирма решила открыть у заводов M магазинов, причём магазин i находится у завода C_i и может продавать товары уровней $L_i \dots R_i$. В процессе создания каждый товар проходит какую-то последовательность заводов $(j_1, j_2, \dots, j_c)$, где первоначальные компоненты изготовляются на заводе j_1, затем они отправляются на завод j_2, где из них изготовляются компоненты более высокого уровня, которые в свою очередь обрабатываются далее вплоть до завода j_c, где изготовляется конечный продукт, который транспортируется в магазин. Базовые компоненты всегда изготовляются на главном заводе, поэтому $j_1 = 1$. Так как каждый завод может обрабатывать только компоненты более низкого уровня, то для каждого $1 \leqslant i < c$ должно выполняться $T_{j_i} < T_{j_{i+1}}$.

Так как самой дорогой частью производственного процесса является транспорт, было решено, что ни одному товару в процессе своего изготовления не разрешено проходить (как в виде компонента, так и в виде готового продукта) ни через один завод более одного раза (даже если товар на этом заводе не обрабатывается). Если для какого-то товара не существует магазина, куда он мог бы таким образом попасть на продажу, то такой товар не продаётся.

Изучая предложения о работе в Universal Manufacturing, Ты заметил, что у них есть вакансия программиста с очень хорошей зарплатой. К тому же Ты обнаружил, что при планировании своей логистики они используют алгоритм, который для каждого магазина и для каждого товара рассматривает все возможные последовательности заводов, с помощью которых можно произвести этот товар, и затем выбирает из них такую, которая нагружает транспортную сеть меньше всего. Ты сразу сообразил, что при большой сети заводов этот алгоритм может не успеть завершиться до того, как Вселенная остынет.

Ты желаешь убедить фирму в неэффективности их алгоритма (и заодно получить возможность его исправить на новом рабочем месте). Для этого Тебе надо написать программу, которая для каждого магазина вычисляет, сколько последовательностей заводов рассматривает их алгоритм.

Входные данные. На первой строчке стандартного входа указано количество заводов N ($1 \leqslant N \leqslant 10^5$). На второй строчке стоят N целых чисел $P_1 \dots P_N$ ($1 \leqslant P_i < i$), где P_i означает, что между заводами i и P_i есть дорога (за исключением случая $P_1 = 0$, что не представляет из себя дороги). На третьей строке стоят N целых чисел $T_1 \dots T_N$ ($0 \leqslant T_i < N$), которые указывают уровни заводов. Известно, что $T_1 = 0$ и все значения T_i различны. На четвёртой строчке стоит количество магазинов M ($1 \leqslant M \leqslant 10^5$). На каждой из последних M строчек стоят три целых числа C_i, L_i и R_i ($1 \leqslant C_i \leqslant N$ и $0 \leqslant L_i \leqslant R_i < N$),

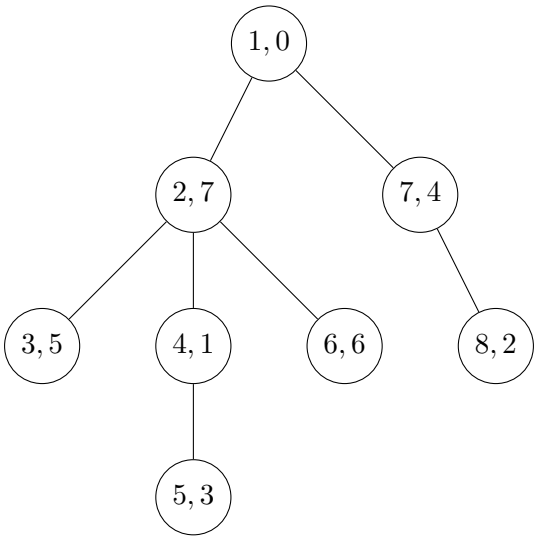
которые означают, что магазин i находится у завода C_i и может продавать товары уровней $L_i \dots R_i$.

Выходные данные. На строчку i стандартного выхода вывести количество всех последовательностей заводов, которые соответствуют всем товарам, подходящим магазину i. Так как вариантов может быть очень много, то вместо настоящего их количества вывести его остаток при делении на $10^9 + 7$.

Пример.

Входные данные	Выходные данные
8 0 1 2 2 4 2 1 7 0 7 5 1 3 6 4 2 4 5 1 4 8 2 4 6 1 4 2 1 7	3 2 0 1

Сеть заводов в примере выглядит следующим образом:



Первому магазину подходят последовательности заводов $(1, 4)$, $(1, 4, 5)$ и $(1, 5)$. Обратим внимание, что во всех последовательностях до прибытия в магазин компоненты проходят путь $1 \to 2 \to 4 \to 5$.

Второму магазину подходят последовательности $(1, 7)$ и $(1, 8)$.

Для третьего магазина не существует ни одной последовательности, которая производила бы подходящий товар и в которой путь компонентов не проходил бы через какой-нибудь завод несколько раз.

Оценивание. В этом задании тесты разбиты на группы и очки раздаются в соответствии тому, какую часть тестов группы программа решила. В группах выполняются следующие ограничения:

1. $N \leqslant 15$, $M \leqslant 15$ (10 очков)
2. $N \leqslant 2000$, $M \leqslant 2000$ (10 очков)
3. Дополнительные ограничения отсутствуют (10 очков)
4. Для каждого i имеет место $P_i = i - 1$, интерактивная подзадача (10 очков)

5. Дополнительные ограничения отсутствуют, интерактивная подзадача (10 очков)

Технические примечания. Во всех подзадачах входные данные зачитываются из стандартного входа, а выходные данные записываются в стандартный выход. В подзадачах 1–3 все входные данные даются сразу и ответы ожидаются после этого. В подзадачах 4 и 5 после задания данных об очередном магазине от программы сразу ожидается ответ. Это означает, что, например, в подзадаче 3 можно зачитать магазины и обработать их все одновременно, а в подзадаче 5 их нужно обрабатывать по одному. Помощь по использованию стандартных входа и выхода доступна на домашней страничке олимпиады по информатике[1].

[1] `http://eio.ut.ee/KKK/StdIO` или `http://eio.ut.ee/` → “Juhendid” → “Interaktiivsed ülesanded”

6. Миграция данных WordNet (`migr`)

50 очков

WordNet[1] — это база данных слов и отношений между ними, которую используют в исследованиях лексики языков. В данный момент начался переход на новое программное обеспечение.

Для того, чтобы начать использовать уже собранные данные с новым программным обеспечением, нужно экспортировать данные из старой системы, экспортированные файлы преобразовать в файлы импорта для новой системы, и в конце концов импортировать их в новую систему.

В тестовой среде дан один комплект файлов-примеров: `migrsis0.txt` — пример файла (отформатированного отступами), экспортированного из старой системы, и `migrval0.txt` — соответствующий ему файл импорта (в формате JSON); `migrval0.fmt.txt` — версия того же файла в более удобном для чтения виде.

JSON[2] как формат данных не учитывает переводов строки и лишних пробелов, поэтому файлы `migrval0.txt` и `migrval0.fmt.txt` как файлы импорта в WordNet равнозначны. Заметим также, что порядок перечисления атрибутов в JSON-файле не важен.

Оценивание. В этой задаче даются (через тестовую среду) 6 экспортированных из старой системы файлов `migrsis1.txt` по `migrsis6.txt` и в качестве решения нужно предоставить соответствующие им файлы импорта в новую систему `migrval1.txt` по `migrval6.txt`. Предоставлять саму программу не нужно, и она не оценивается.

[1] `https://en.wikipedia.org/wiki/WordNet`
[2] `https://en.wikipedia.org/wiki/JSON`

7. Xorshift+ предсказания (xors)

1 секунда 50 очков

Рассмотрим генератор псевдослучайных чисел:

```
function xorshift()
  last_value = internal_value
  for i = 1 to M do
    if S[i] < 0 then
      internal_value = internal_value ^ (internal_value >> -S[i])
    else
      internal_value = internal_value ^ (internal_value << S[i])
  return last_value + internal_value
```

где

- `last_value` — это N-битная локальная переменная;
- `internal_value` — это N-битная глобальная переменная, первоначальное значение которой неизвестно;
- `S` — это глобальный массив констант размером M элементов, значения которых известны;
- `^` обозначает побитовое исключающее "или" (XOR); в результате этого действия со значением 0 будут те биты, для которых соответствующие биты операндов равны, а со значением 1 те, для которых соответствующие биты операндов разные; например, 10`^`12 = 6, так как 10 в двоичном коде будет `1010`, а 12 в двоичном коде `1100`; поэтому результатом действия окажется `0110`, то есть двоичный код числа 6;
- `<<` и `>>` обозначают побитовые сдвиги влево и вправо; при сдвиге N-битного числа на s битов влево из двоичного представления числа стираются s самых левых битов, а справа добавляются s битов со значением 0; например, для 4-битной переменной 6 `<<` 2 = 8, так как 6 в двоичном коде будет `0110`, при удалении 2 левых битов которого получим `10`, а добавив к этому справа два 0-бита, получим `1000`, что является двоичным кодом числа 8; при сдвиге вправо биты удаляются справа и добавляются слева;
- `+` обозначает сложение, причём от результата используются только N самых правых битов;
- первоначальное значение `internal_value` не равно 0, а значения елементов массива `S` такие, что если вызывать функцию `xorshift` неограниченное количество раз, то `internal_value` принимает все значения $1 \dots 2^N - 1$.

Написать программу, которая принимает K значений, последовательно возвращённых функцией `xorshift`, и находит на их основании следующее значение, которое вернет эта функция.

Входные данные. На первой строчке текстового файла `xorssis.txt` находятся три целых числа N ($2 \leqslant N \leqslant 64$), M ($2 \leqslant M \leqslant 10$) и K ($100 \leqslant K \leqslant 200\,000$). На второй строчке находятся M целых чисел S_i ($1 - N \leqslant S_i \leqslant N - 1$). На третьей строчке находятся K целых чисел V_j ($0 \leqslant V_j < 2^N$), K последовательных значений функции `xorshift`.

Выходные данные. На единственную строчку текстового файла `xorsval.txt` вывести одно целое число — следующее значение, которое вернёт функция `xorshift`.

Пример.

`xorssis.txt`

```
2 2 100
-1 1
0 1 3 ... 0 1 3 0
```

`xorsval.txt`

```
1
```

Третья строчка входных данных слишком длинна для текста задачи, но видна в файле-примере, доступном с тестового сервера.

Оценивание. В тестах стоимостью 15 очков выполняются условия $2 \leqslant N \leqslant 26$ и $100 \leqslant K \leqslant 1000$. В последующих тестах стоимостью 10 очков выполняются условия $2 \leqslant N \leqslant 37$ и $100\,000 \leqslant K \leqslant 200\,000$. Затем в тестах стоимостью 25 очков дополнительные ограничения отсутствуют.