# Задача A. В поисках доспехов...

| | |
|---|---|
| Имя входного файла: | `armor.in` |
| Имя выходного файла: | `armor.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 64 мегабайта |

Благодаря вашим усилиям главный герой получил великолепное оружие. Но для освобождения мира этого мало. Он быстро смекнул, что ему не помешает приобрести доспехи. Естественно, его интересуют самые мощные доспехи — даэдрические. Полный комплект доспехов состоит из $k$ элементов (например, кольчуга, шлем, щит...). Герой хочет собрать полный комплект даэдрических доспехов.

В мире Morrowind существует $n$ городов. Некоторые пары городов соединены дорогами. По каждой дороге разрешено движение в обоих направлениях. Между парой городов может быть более одной дороги. Не существует дороги, ведущей из города в себя. Всего $m$ дорог. За то, чтобы пройти по любой дороге, герой должен заплатить определённую сумму денег. В каждом городе есть оружейный магазин. Там можно покупать составляющие комплекта. Если купить что-нибудь в некотором магазине, то на все последующие покупки в этом магазине предоставляется скидка, зависящая от магазина и покупки. Несколько скидок в одном магазине складываются: например, после покупки в каком-то магазине вещей, дающих скидки 5% и 7%, следующая покупка в этом магазине будет стоить на 12% дешевле.

Главный герой живёт в городе $v$. Он хочет купить все $k$ элементов доспехов (каждого по одной штуке) и вернуться домой, потратив как можно меньше денег. Вы просто обязаны ему помочь!

## Формат входного файла

В первой строке входного файла записаны целые числа $n$ ($1 \leqslant n \leqslant 50$), $m$ ($1 \leqslant m \leqslant 500$), $k$ ($1 \leqslant k \leqslant 7$) и $v$ ($1 \leqslant v \leqslant n$). Далее следует $n$ строк с описанием магазинов. В $(i+1)$-й строке находится описание магазина, расположенного в $i$-м городе: оно состоит из $k$ пар целых чисел. Первое число — стоимость $j$-ой составляющей ($1 \leqslant a_{ij} \leqslant 10^6$, либо $a_{ij} = 0$, если купить нельзя; все $a_{ij}$ кратны 100), а второе — скидка в процентах, если купить $j$-ю составляющую в $i$-ом магазине ($0 \leqslant p_{ij} \leqslant 10$). Далее следует $m$ строк с описанием дорог. Каждая строка содержит три целых числа $a$, $b$ и $c$ — номера городов, которые соединяет дорога, и стоимость проезда по ней ($1 \leqslant a, b \leqslant n$, $a \neq b$, $1 \leqslant c \leqslant 10^6$).

## Формат выходного файла

В выходной файл выведите единственное число — минимальную сумму, необходимую герою, либо $-1$, если полный комплект доспехов собрать невозможно.

## Пример

| armor.in | armor.out |
|---|---|
| 2 1 2 1<br>500 0 500 0<br>300 10 500 5<br>1 2 50 | 850 |

# Задача B. Дерево

| | |
|---|---|
| Имя входного файла: | *нет* |
| Имя выходного файла: | *нет* |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 64 мегабайта |

Есть дерево из $n$ вершин. Степень каждой вершины не превосходит 5. Алиса и Боб играют в такую игру: Алиса загадывает какую-то вершину, Боб пытается её угадать. В процессе игры Боб может указать на ребро, и Алиса сообщает, по какую сторону от ребра лежит загаданная вершина. Вам предлагается написать программу, которая играет за Боба и угадывает загаданную вершину не более чем за 100 запросов.

## Работа с модулем

Ваша программа должна взаимодействовать с модулем. Для этого она должна подключить модуль `treeunit` на Pascal или заголовочный файл `treeunit.h` на C/C++. Необходимые материалы вы можете найти на сайте сборов.

В модуле доступны следующие функции:

```
procedure init;
```

```
void init();
```

Эта функция служебная. Вы должны вызвать её в начале вашей программы.

```
function getN: longint;
```

```
int getN(void);
```

Функция возвращает $n$ — количество вершин в дереве ($1 \leqslant n \leqslant 200\,000$).

```
function getA(edgeNum: longint): longint;
function getB(edgeNum: longint): longint;

int getA(int edgeNum);
int getB(int edgeNum);
```

Эти функции возвращают номера концов ребра с номером `edgeNum` ($1 \leqslant$ `edgeNum` $\leqslant n-1$). Возвращаемые значения — натуральные числа от 1 до $n$.

```
function query(edgeNum: longint): longint;

int query(int edgeNum);
```

С помощью этой функции вы можете узнать по какую сторону ребра с номером `edgeNum` лежит загаданная вершина. Функция возвращает 0, если вершина лежит по ту же сторону, что и `getA(edgeNum)`, и 1 в противном случае.

```
procedure report(vertexNum: longint);

void report(int vertexNum);
```

Вы должны использовать эту функцию для того, чтобы сообщить ваше предположение о загаданной вершине. Если вы вызовете эту функцию, то ваша программа будет завершена.

Если будет вызвана какая-либо функция с некорректными параметрами, либо не будет вызвана `init` в начале программы, то ваша программа также будет завершена.

Для отладки ваших решений на сайте сборов можно будет скачать демо-версии модулей. Эти версии читают данные из файла `input.txt` и записывают результат в стандартный поток вывода. Их интерфейс взаимодействия с вашей программой будет идентичен используемому при тестировании.

При компиляции программ на C++ используйте следующую команду:

```
g++ -o a a.cpp treeunit.o
```

При компиляции программ на C используйте следующую команду:

```
gcc -o a a.c treeunit.o
```

При компиляции программ на Pascal используйте следующую команду:

```
fpc a.pas
```

### Формат входного файла

В первой строке находится $n$ — количество вершин в дереве. Следующие $n-1$ строк содержат описания рёбер дерева — каждое ребро задано номерами своих концов. Следующая строка содержит номер загаданной вершины.

### Формат выходного файла

В стандартном потоке вывода будет содержаться некоторая полезная информация — лог взаимодействия вашей программы с модулем и описание произошедших ошибок.

### Пример

| input.txt | стандартный поток вывода |
|---|---|
| 5<br>1 2<br>2 3<br>3 4<br>4 5<br>3 | init()<br>getN()<br>getA(1)<br>getB(1)<br>query(1)<br>getA(2)<br>getB(2)<br>query(2)<br>getA(3)<br>getB(3)<br>query(3)<br>getA(4)<br>getB(4)<br>query(4)<br>report(3)<br>ok |

## Задача C. Префиксные коды-6

| | |
|---|---|
| Имя входного файла: | `code.in` |
| Имя выходного файла: | `code.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 64 мегабайта |

Кодирование символов последовательностями нулей и единиц — одна из первых проблем, с которой столкнулись пионеры компьютерной науки. *Кодом* называется соответствие каждому символу из некоторого алфавита $S$ непустой последовательности нулей и единиц (будем называть её *записью* соответствующего символа).

Вот несколько примеров кодов, на которых будут иллюстрироваться дальнейшие определения. Во всех примерах мы будем рассматривать алфавит из 4 символов: $\{A, B, C, D\}$.

1. $A \to 000$, $B \to 001$, $C \to 01$, $D \to 10$.
2. $A \to 11$, $B \to 00$, $C \to 10$, $D \to 01$.
3. $A \to 0$, $B \to 00$, $C \to 000$, $D \to 1$.

Код является *префиксным*, если ни для каких двух различных символов запись одного из них не является началом записи второго (а также записи никаких двух символов не должны совпадать). Например, коды 1 и 2, приведённые выше, являются префиксными, а код 3 — нет.

Код является *алфавитным*, если записи всех символов, взятых в алфавитном порядке по возрастанию, идут в лексикографическом порядке по возрастанию. Напомним, что строка $X$ идёт в лексикографическом порядке раньше строки $Y$, если строка $X$ является началом строки $Y$, или если на первом слева месте, где символы строк $X$ и $Y$ не совпадают, в строке $X$ идёт меньший символ. Например, следующие строки расположены в лексикографическом порядке: $0 < 00 < 000 < 1 < 100 < 101 < 11 < 110$.

Из приведенных выше кодов только код 1 является алфавитным.

Пусть для каждого символа известна его частота — целое число. Тогда *эффективностью* кода называется сумма длин записей всех символов, умноженных на частоты соответствующих символов. Например, если частота буквы A — 5, B — 1, C — 7, D — 3, то эффективность кода 1 равна $5 \cdot 3 + 1 \cdot 3 + 7 \cdot 2 + 3 \cdot 2 = 38$, эффективность кода 2 равна 32, а эффективность кода 3 равна 31.

По заданным частотам символов найдите алфавитный префиксный код с минимальной эффективностью. Если таких кодов несколько, найдите любой из них.

## Формат входного файла

В первой строке входного файла записано число $N$, $1 \leqslant N \leqslant 100\,000$ — количество символов в алфавите. Во второй строке записаны частоты этих символов, разделённые пробелами. Частоты — целые числа между 1 и 1 000 000 включительно. Частоты записаны в том же порядке, в котором идут соответствующие символы в алфавите.

## Формат выходного файла

В выходной файл выведите $N$ строк, $i$-я из которых должна содержать запись $i$-го символа.

## Пример

| code.in | code.out |
|---|---|
| 4<br>5 1 7 3 | 00<br>01<br>10<br>11 |
| 4<br>5 1 3 7 | 00<br>010<br>011<br>1 |