Фильтр Блума

Реализуйте фильтр Блума, позволяющий дать быстрый, но вероятностный ответ, присутствует ли объект в коллекции.

Реализация самой структуры данных должна быть инкапуслирована, т.е. не зависеть от форматов входных/выходных данных и непосредственно ввода/вывода.

Реализация битового массива также должна быть инкапсулирована. Массив битов должен быть эффективно расположен в памяти.

Параметрами структуры данных являются n - приблизительное количество элементов, P - вероятность ложноположительного ответа.

Размер структуры, m, вычисляется как

$$m = \frac{-n \log_2 P}{\ln 2},$$

а количество хэш-функций - как $-\log_2 P$. Оба значения округляются до ближайшего целого.

В качестве семейства функций используйте семейство хэш-функций вида

$h_i(x) = (((i + 1)*x + p_{i+1}) \bmod M) \bmod m,$

где - x - ключ, i - номер хэш-функции, p_i - i + 1 по счету простое число, а M - 31^{oe} число Мерсенна.

Формат ввода

На стандартном потоке ввода задаётся последовательность команд. Пустые строки игнорируются.

Первая строка содержит команду вида set n P.

Каждая последующая строка содержит ровно одну команду: add K, search K или print, где K - неотрицательное число (64 бита вам хватит), ключ.

Формат вывода

Команда set инициализирует структуру и выводит вычисленные параметры в формате "m k".

Команда add добавляет в структуру ключ K.

Команда search выводит либо "1", если элемент возможно присутствует в структуре, либо "0", если он там отсутствует.

Команда print выводит внутреннее состояние струтуры - последовательность из 0 и 1, не разделенную пробелами.

В любой непонятной ситуации результатом работы любой команды будет "error".

Результат работы программы выводится в стандартный поток вывода.

Пример

Ввод:

```
set 2 0.250
add 7
add 5
add 14
print
search 7
search 10
search 15
search 14
search 5
search 13
```

Вывод:

```
6 2
010111
1
0
1
1
1
1
```