Задача A1. Анализ линейного пробирования

В хеш-таблице с открытой адресацией разрешение коллизий производится с помощью *линейного пробирования*. При удалении объекта из хеш-таблицы свободная ячейка получает значение `ERASED`, отличное от `NULL`, которое обозначает *пустое* значение.

Ниже приведены алгоритмы вставки, удаления и поиска, где `M` обозначает размер хеш-таблицы:

```
INSERT(key):
    ind = hash(key) mod M

    while (table[ind] != NULL)
        if (table[ind] == key) return
        ind = (ind + 1) mod M

    table[ind] = key
```

```
DELETE(key):
    ind = hash(key) mod M

    while (table[ind] != NULL)
        if (table[ind] == key)
            table[ind] = ERASED
            return
        ind = (ind + 1) mod M
```

```
SEARCH(key):
    ind = hash(key) mod M

    while (table[ind] != NULL)
        if (table[ind] == key)
            return true
        ind = (ind + 1) mod M
    return false
```

Система оценки

1. 5 баллов Приведенные выше алгоритмы вставки, удаления и поиска ключа имеют проблему, которая приводит к *долгому* выполнению некоторой(-ых) последовательности(-ей) этих операций.
   - Найдите такую(-ие) последовательность(-и) операций вставки, удаления и поиска.
   - Охарактеризуйте соответствующее состояние хеш-таблицы. Приведите примеры.
2. 3 балла Предложите доработки (*кроме* перехеширования) исходных алгоритмов вставки, удаления и поиска, которые помогут исправить обнаруженную вами проблему.