SET-5

Группа БПИ243, Тупицин Тимофей Романович

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
```

1.

- 1. Использование операции вставки после операции удаления элемента.
  2. Использование операции поиска элемента после операции удаления элемента.
- Состояние хэш-таблицы после выполнения данных последовательностей операций называется засорением. При каждом DELETE мы по факту не удаляем элемент, а ставим вместо него флаг ERASED, а флаг в INSERT и SEARCH никак не проверяем. Таким образов, наши операции неэффектино работают.
  Пример: у нас есть хэщ-таблица размера $n = 10000$, пусть мы удалили 9999 элементов, и один остался NULL. Теперь, мы хотим вставить новый элемент, но при этом мы будем идти по всей таблице, так как все элементы будут помечены как ERASED, и только в конце мы дойдем до NULL и вставим туда новый элемент. Таким образом, время работы операции INSERT в худшем случае будет $O(n)$.
  Аналогично для операции SEARCH, если мы будем искать элемент, который отсутствует в таблице, то мы будем идти по всей таблице, так как все элементы будут помечены как ERASED, и только в конце мы дойдем до NULL и поймем, что элемента нет. Таким образом, время работы операции SEARCH в худшем случае будет тоже $O(n)$.

2. Чтобы решил обнаруженную проблему, нам нужно добавить механизм проверки на ERASED в методах вставки и поиска. Но если сделать это красиво, то метод SEARCH можно не трогать.
При вставке элемента будем запоминать индекс первого вхождения ERASED. Если мы дойдем до NULL, то вставим элемент на запомненный индекс, а не на NULL. Если же ERASED не было, то вставим на NULL, как и раньше.
Теперь метод INSERT будет закрывать места ERASED, что облегчит поиск элемента.

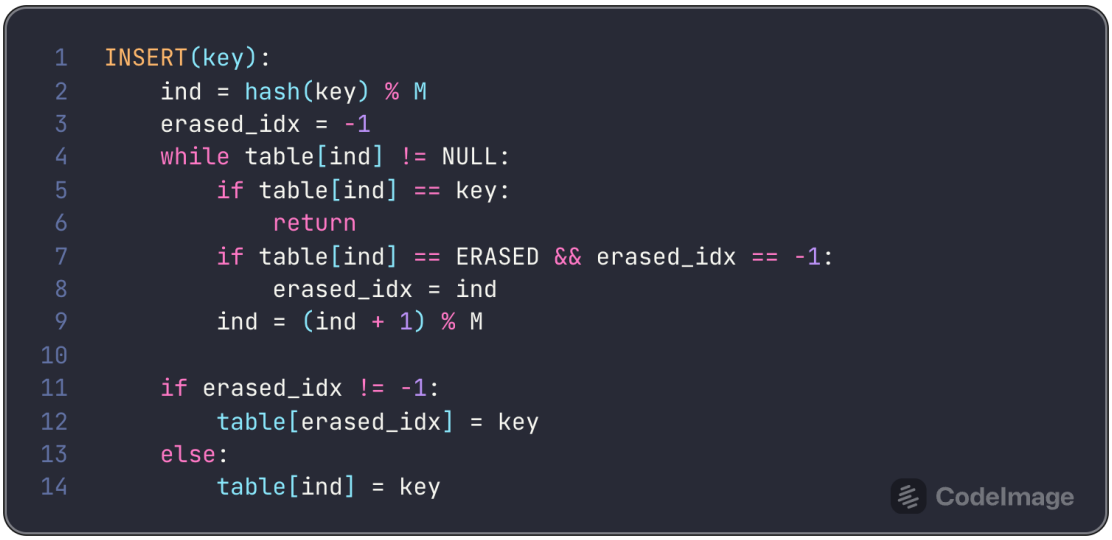

Рис. 1: Исправленный метод INSERT