# Ассимптотическая сложность операций

## [BOOK time hotel_name client_id room_count]

– при записи о бронировании номера в отеле, который ещё не упоминался:

$O(\log_2(h))$ , где h – количество отелей;

– если отель уже существует в базе, то

лучший случай: $O(\log_2(h))+O(\log_2(c))$ , где h – тоже, что и выше, c – количество клиентов, сделавших бронирование в отеле с названием hotel_name;

**худший случай**: $O(\log_2(h))+O(\log_2(c))+O(r)$ , где h, c – тоже, что и выше, r – количество записей о бронировании в отеле с названием hotel_name.

Когда количество записей о бронировании некоторого отеля превысит capacity вектора, в котором они хранятся, произойдёт копирование всех элементов вектора с записями в новую память, при этом размер вектора удваивается (size*2) (если это gcc), или size*1.5 (если MCVS).

## [CLIENTS hotel_name]

$O(c\cdot\log_2(c))$ , где c – количество клиентов, сделавших бронирование в отеле с названием hotel_name.

Необходимо в красно-черном дереве найти информацию об отеле с именем hotel_name, эта операция имеет сложность $O(\log_2(h))$ . Обход красно-черного дерева, где ключом является client_id, имеет сложность O(c), a erase (на каждой итерации мы убираем тех клиентов, которые бронировали отель >= 1 суток назад) в красно-черном дереве имеет сложность $O(\log_2(c))$ , в худшем случае мы будем делать это на каждой итерации, поэтому общая сложность будет $O\left(\sum_{i=c}^{1}\log_2(i)\right)=O(\log_2(c\cdot(c-1)\cdot(c-2)...\cdot1))\sim O(\log_2(c^c))=O(c\cdot\log_2(c))$ .

## [ROOMS hotel_name]

$O(\log_2(h))+O(\log_2(\hat{r}))+O(\hat{r})$ , где r – количество записей о бронировании в отеле с названием hotel_name, где $\hat{r}$ – количество записей о бронировании в отеле с названием hotel_name за последние сутки (исключая запись, сделанную ровно за сутки), $\hat{r}\leqslant r$ .

Необходимо в красно-черном дереве найти информацию об отеле с именем hotel_name, эта операция имеет сложность $O(\log_2(h))$ . Поиск граничной записи “по времени <= 1 суток назад” в отсортированном по времени массиве занимает $O(\log_2(\hat{r}))$ , где $\hat{r}\leqslant r$ . Проход по нужным записям с помощью accumulate имеет сложность $O(\hat{r})$ . В сумме мы имеем сложность $O(\hat{r})$ .