Структуры даных II.

1. В структуре данных «очередь с приоритетами» хранятся пары ключ-приоритет. В этой структуре данных есть следующие операции:

- `insert`(k, p) — добавляет элемент с ключом k и приоритетом p;
- `extract_max`$()$ — возвращает пару (k, p) с максимальным приориоритетом p;
- `set_priority`(k, p) усанавливает приоритет p ключу k.

Опишите реализацию очереди с приоритетами с помощью красно-чёрных деревьев. Каждая операция должна выполняться за $O(\log n)$, где n — число элементов в структуре данных.

2. Определим хэш-функции

$$h_n(x) = x \mod n \quad \text{и} \quad f_k(x) = 1 + (x \mod k).$$

Считаем, что ключи x — некоторые положительные целые числа. Рассмотрим следующий алгоритм хэширвания (добавления и поиска ключа x в описываемую далее структуру данных «хэш-таблица»). В начале алгоритма заводится массив a размера M (хэш-таблица) и выбираются функции h_n и f_k. Получив на вход ключ x, алгоритм вычисляет значение $i = h_n(x) \mod M$ и если i-ая ячейка массива a пуста, то $a[i] = x$. В случае, если в i-ой ячейке лежит ключ (отличный от x), то к i применяется функция f_k: $j = i + f_k(x) \mod M$ и если j-ая ячейка пуста, то $a[j] = x$, а если нет, то к j ещё раз прибавляется значение $f_k(x)$ ($j_2 = i + 2f_k(x)$) и так до тех пор, пока для x не будет найдена свободная ячейка.

1. Продемонстрируйте работу алгоритма хэширования при $M = 6$, $n = 5$, $k = 4$ для последовательности ключей $7, 12, 2$.

2. При каких соотношениях между числами M, n и k гарантируется, что алгоритм полностью заполнит хэш-таблицу, какие бы M элементов $x_1, x_2, \ldots, x_M$ ни были бы поданы на вход?

3. Пусть M, n и k выбраны правильно (гарантируется выполнение условия из предыдущего пункта). Сколько операций в худшем случае потребуется для добавления нового элемента (считается, что таблица не полностью заполнена)?

Комментарий. Пусть в хэш-таблице находится m элементом *коэффициент заполнения* определяется как $\alpha = \frac{m}{M}$. Решив задачу, мы изучили алгоритм двойного хэширования с открытой адресацией. Математическое ожидание (среднее число операций) шагов по массиву при поиске места для элемента равно $\frac{1}{1-\alpha}$. Этот алгоритм удобен тем, что при его реализации не требуется использование указателей.

3. Переменная-счётчик содержит k битов и последовательно меняется от 0 до $n = 2^k - 1$. При каждом увеличении алгоритм меняет каждый бит. Так, чтобы перейти от $00\ldots00$ к $00\ldots01$ потребуется всего одна операция (нужно изменить только первый бит), а чтобы перейти от $00\ldots01111$ к $00\ldots10000$ потребуется пять операций изменения битов. Используя амортизационный анализ покажите, что при изменении счётчика от 0 до n произойдёт $O(n)$ операций (грубая оценка показывает, что произойдёт $O(kn)$ операций).

4. При динамическом выделении памяти используют следующую эвристику. Пусть сначала в динамическом массиве хранится один элемент. Как только место в массиве заканчивается, размер массива увеличивают в два раза. Считая, что выделить память на k элементов стоит

$O(k)$ операций, докажите, что при добавлении в динамический массив n элементов потребуется $O(n)$ операций. Считайте, что если в массиве не закончилось место, то вставка элемента стоит $O(1)$.

Комментарий. Эта техника справедлива и для хэш-таблиц. В случае, если коэфициент заполнения превышает допустимый (и в таблице n элементов), перестроение хэш-таблицы (с увеличением её размера примерно в два раза) будет стоить $O(n)$ операций, однако амортизационная стоимость операции перестроения будет $O(1)$.