Дедлайн - 9 ноября.

Задача 1. Предложите вариант реализации класса vector - динамического массива, поддерживающего следующие операции:

а) push_back;

б) pop_back;

в) operator [] (доступ к i-му элементу в массиве);

-

Учетное время работы каждой из этих операций должно быть O(1). Количество занимаемой памяти в каждый момент времени должно быть O(текущее количество элементов в векторе). Проведите анализ с помощью метода бухгалтерского учета и с помощью метода потенциалов.

Задача 2. Предложите вариант реализации класса deque - динамического массива, поддерживающего следующие операции:

а) push_back;

б) pop_back;

в) push_front;

г) pop_front;

д) operator [] (доступ к i-му элементу в массиве);

Учетное время работы каждой из этих операций должно быть O(1). Количество занимаемой памяти в каждый момент времени должно быть O(текущее количество элементов в векторе). Проведите анализ с помощью метода бухгалтерского учета и с помощью метода потенциалов.

Задача 3. У вас есть бинарный счетчик, хранящийся в виде списка A[0..k - 1] своих бит. На каждом шаге последовательности нужно прибавить к счетчику единицу с помощью следующего кода:

```
i = 0;
while (i < A.length() && A[i] == 1) {
    A[i] = 0;
    ++i;
}
if (i < A.length())
    A[i] = 1;
```

За какое реальное и амортизированное время работает данный алгоритм? А если добавить аналогичную операцию Decrement?

Задача 4. Разработайте структуру данных для поддержки следующих двух операций над динамическим мультимножеством целых чисел S, в котором могут содержаться одинаковые значения:

Insert(x) - вставить элемент x в S;

Delete-LargerHalf(S), удаляющая наибольшие ceil(|S|/2) элементов из S.

Предложите реализацию такой структуры, т.ч. учетная стоимость каждой из операций была O(1).

Задача 5. Хотим реализовать структуру, которая умеет делать следующие запросы:

а) Insert(x): Добавить элемент в структуру;

б) Search(x): Узнать, есть ли данный элемент в структуре.

В качестве возможной реализации предлагается хранить A в виде набора отсортированных массивов, длины которых являются попарно различными степенями двойки.

а) Опишите функцию Search. За какое время она работает?

б) Опишите, как вставлять новый элемент в такую структуру данных. За какое реальное и учетное время работает Insert?

в) Каким образом можно реализовать операцию Delete(x) без потери асимптотической (как по времени, так и по памяти) эффективности? За какое время Delete(x) будет работать?

Задача 6(2 балла). Рассмотрим следующую модификацию алгоритма TimSort. Пусть во второй части алгоритма (там, где мы добавляем run-ы в стек) в каждый момент времени мы проверяем все тройки идущих подряд run-ов, которые могут не удовлетворять требуемым условиям (условия стандартные: если длины run-ов “сверху вниз” равны X, Y, Z, то выполняются неравенста X <= Y, X+Y <= Z), и если условия не выполнены, то в самой “низкой” такой ситуации выполнить слияние Y с минимальным из X и Z run-ом. Заметим, что непосредственно после добавления проверять нужно самую верхнюю “тройку”, после слияния - две тройки и т.д.. Докажите или опровергните утверждение о том, что предложенная модификация TimSort работает за O(n log n) (требуется уметь проделывать это не только для второй фазы, но и для остальных двух!)