Алгоритмы и структуры данных. Домашнее задание. Сортировки, простые структуры данных и кучи до кучи.

Артюхин С., Евстропов Г., Резников Г., Сендерович Н., Смирнов И., Фельдшеров С.
hse.algo@gmail.com

Символ «*» рядом с пунктом означает, что это сложный бонусный вопрос.

1. Пусть все элементы последовательности $a_0, a_1, \ldots, a_{n-1}$ были сгенерированы случайно на отрезке от 0 до 1. Выполните некоторый предподсчёт за гарантированное время $O(n)$, после чего за ожидаемое время $O(1)$ (в данном случае имеется в виду ожидаемое по случайным входным данным) отвечайте на вопрос поиска ближайшего к x элемента a_i ($|x - a_i| \to \min$).

2. Пусть дан массив длины n и k запросов поиска порядковой статистики. Придумайте рандомизированный и детерменированный алгоритмы, отвечающие на все запросы на время $O(n \log k)$.

3. Даны n точек $a_0, \ldots, a_{n-1}$ на прямой. Найти $\max_{i=1}^{n-1}(a_{(i)} - a_{(i-1)})$, где $a_{(i)}$ — i-я порядковая статистика. Сложнось $O(n)$.

4. Предложите полиномиальный алгоритм сортировки стека, использующий ещё один стек и $O(1)$ дополнительной памяти. Элементы в стеке являются чёрными ящиками, для которых доступны операции сравнения. Создавать копии элементов не разрешается, алгоритм должен быть полиномиальным в смысле количества перемещений элементов и сравнений. Использование $O(1)$ дополнительной памяти не запрещает хранить $O(1)$ элементов вне стеков.

5. Пусть для некоторого типа данных доступна операция сравнения и волшебная структура данных, которая позволяет за $O(1)$ выполнять с этим типом данных два типа операций: добавить элемент в структуру, сказать медиану множества. Удаление элементов невозможно, но разрешается создавать неограниченное количество структур. Покажите, как отсортировать массив из n элементов данного типа за время $O(n)$. Дополнительное ограничение: известно, что все элементы различны, и в структуру не разрешается добавлять один и тот же элемент несколько раз.

6. К структуре данных k-ичная куча применяют n^a запросов извлечения минимума и n^b запросов добавления элемента. Считая, что a и b некоторые константы, причём $0 < a < b$, покажите, что при оптимальном выборе параметра k общее время работы структуы будет составлять $O(n^b)$.

7. Предложите способ реализовать кучу во внешней памяти, чтобы время выполнения всех основных операций (добавление, извлечение минимума и т.п.) составляло $O(1 + \frac{\log \frac{n}{M}}{\log B})$.

8. **Напишите псевдокод** процедуры, принимающей указатель на элемент двусвязного закольцованного списка, и выполняющей разворот этого списка. Разворот закольцованного списка означает, что порядок обхода, который получался при следовании указателям *left* теперь получается при следовании указателями *right*, и наоборот. Время работы линейное, разрешается использовать $O(1)$ дополнительной памяти, не разрешается создавать новые элементы структуры (только перенаправлять имеющиеся указатели). Создавать вспомогательные указатели разрешается.

9. Пусть имеется массив $a[i]$ длины n и m запросов *foo(l, r)* и *boo(l, r, x)*, листинг которых приведён ниже. Используя метод амортизационного анализа с помощью потенциалов покажите, что **количество вызовов** функции *doMagic()* имеет порядок $O(n+m)$. Функция *doMagic()* не изменяет массив.

```
void foo(int l, int r) {
    for (int i = l + 1; i <= r; i++) {
        if (a[i] != a[i - 1]) {
            doMagic();
        }
    }
    for (int i = l + 1; i <= r; i++) {
        a[i] = a[i - 1];
    }
}

void boo(int l, int r, int x) {
    for (int i = l; i <= r; i++) {
        a[i] += x;
    }
}
```

10. В массиве длины $2n$ записаны две отсортированные последовательности по n элементов в каждой. Первая находится в элементах $a_1 \leq a_2 \leq \ldots \leq a_n$, вторая $a_{n+1} \leq a_{n+2} \leq \ldots \leq a_{2n}$. Предложите алгоритм inplace stable merge (разрешается использовать не более $O(\log n)$ дополнительной памяти).

    (a) $O(n^2)$;

    (b) *$O(n \log n)$;

    (c) *$O(n)$.