2.1.1 Реализовать стек с операция Push, Pop и Max, работающими за константное время.

Решена на практике, но все же:
Основная идея – завести 2 стека, первый будет использоваться для хранения данных, второй стек будет поддерживать минимальный элемент на отрезке [дно стека, текущий элемент]. Максимальный элемент всегда находится на вершине второго стека. Pop снимает по одному элементу с вершины обоих стеков, а Push работает так: если вновь вставляемый элемент больше либо равен текущего максимума, то во второй стек положить этот элемент, иначе во второй стек положить текущий максимум.

2.1.2 Реализовать очередь с операциями Enqueue, Dequeue и Max, работающими за константное время.

Тоже разобрана на практике:
Так как очередь можно реализовать с помощью двух стеков (стек для push, стек для pop и в случае пустоты второго стека – делать перемещение из первого), то можно для каждого стека аналогично п.1 завести еще по одному стеку, в которых хранить максимальный элемент и аналогично п.1 поддерживать максимальный. Итоговый максимальный элемент – максимум из двух максимумов – первого стека и второго.

2.1.3 Найти в последовательности целых чисел подотрезок с максимальной суммой.

На практике не разобрана:
Заведем массив частичных сумм S: S[i+1] = S[i] + a[i+1]; (очевидно за линейное время)
Нам необходимо найти такие r и l, чтобы S[r]-S[l-1] была максимальной. Значит, нужно для каждого фиксированного r находить такое l (l<=r) чтобы S[l-1] была минимальной.
Будем пробегать индексом r=1..n по этому массиву и поддерживать минимум частичных сумм на отрезке [1..r]. На каждом шаге будем пересчитывать найденную сумму, запоминать ее максимум (Smax = max(Smax, S)) и обновлять минимальную сумму (соответствующую индексу l).
Искомые индексы тоже поддерживаются при обновлении максимальной суммы + минимальной суммы S[l].

2.1.4 Найти в последовательности натуральных чисел подотрезок с суммой, равной заданной.

Решена на практике. Основная идея – 2 указателя.

Домашнее задание

2.2.1 Дана скобочная последовательность из скобок (,[,{,},],). Определить правильность последовательности.

Пусть, для определенности, скобке (соответствует число 0, [- 1, { - 2,) – 3,] – 4, } – 5.
Будем обрабатывать последовательность слева направо и для каждого поступающего числа от 0 до 2 – делать Push в стек, а для 3-5 – Pop.
Условие правильности скобочной последовательности простое: во-первых стек в конце должен быть пуст, во вторых при Pop необходимо проверять, что разность между вершиной стека и числом, которое вызвало Pop равно трем (таким образом исключается неправильная последовательность скобок), в третьих не должно быть Pop пустого стека.

2.2.2. Придумать структуру данных, которая бы отвечала онлайн на запросы вида «Найти сумму на отрезке [l, r]» для массива целых чисел.

Классическая идея:

В задаче 2.1.3 такая структура уже была представлена – массив частичных сумм.

Ответ на запрос будет следующим: S[r] – S[l-1]. Очевидно на ответ уходит O(1) времени.

Предподсчет занимает O(n) времени и столько же памяти.

2.2.3. Дано n-значное число без ведущих нулей. Из числа нужно вычеркнуть ровно k цифр, чтобы число получилось максимальным.

Граничные случаи не рассматриваем, они не интересны.

Чтобы число было максимальной, старшей должна быть наибольшая из возможных цифр. Значит, алгоритм следующий: выбираем старшие k+1 цифру и ищем среди них максимальную. Пусть она находится на месте m. Все что слева от этого максимального – удаляем (цифры с 1 по m-1). Далее запускаем ту же задачу только на числе от m+1 до n-го места, причем k = k – m + 1. (уменьшается на столько, сколько цифр мы вычеркнули).

Некоторые моменты технической реализации: можно не вычеркивать, а например заменять на -1.

Максимум можно не искать, а «поддерживать», для этого заведем очередь, которая возвращает максимум за O(1) (такую мы уже сделали в п. 2.1.2). При движении «окна» шириной в k+1 будем вставлять в очередь очередную цифру и удалять последние. Аналогично можно хранить не только максимальное значение, а также индекс этого максимального.

Окно двигать можно так:

- Пусть l указывает на левую границу окна, r – на правую.
- Двинем r пока ширина окна не станет равной k+1. Одновременно с движением r направо будет вводить в очередь появляющиеся в окне цифры. Очередь нам за O(1) скажет максимум и его индекс.
- Двинем l направо до найденного индекса и удалим из очереди эти цифры (одновременно запишем туда, например, -1).
- Двинем l еще на единичку направо: максимум не вычеркиваем, но из очереди удаляем.
- Возвращаемся к пункту 2 пока есть куда возвращаться, либо пока еще не вычеркнуто k цифр.

Время работы – линейное относительно n. Это очевидно следует из того факта, что r пройдет максимум n и l пройдет максимум n цифр.

2.2.4 Найти максимальное произведение длины подотрезка на минимум на нем.

Идея: решение на отрезке будет максимумом из трех чисел: первое – длина отрезка умножить на минимум на нем; второе – решение на отрезке от первого до минимума не включая его; третье – решение на отрезке от следующего после минимума до конца.

Это очевидно, перебраны все варианты, кроме отрезка, который содержит в себе этот минимум, но это неоптимально, потому что отрезок можно раздвинуть до конца – но тогда он превратится в «первое число» = длина отрезка * минимум в нем.

У меня получается пока лишь за N*lnN, идея хотя бы верная? Стоит ее дальше развивать в линию или совсем не в ту степь?