Форма 3: разбор

1:

> Мы обсудили skip-list как структуру, в которой мы заранее сделали log n уровней, после чего узлы при добавлении сами решают, на сколько уровнях они окажутся добавлены. Но вообще-то, пока в списке 10 элементов, поддерживать 20 уровней смысла нет. Предложите в общих чертах, как можно изменить скиплист, чтобы количество уровней динамически менялось и соответствовало логарифму от текущего количества элементов.

В целом, тут можно придумать много подходов. Самым простым, наверное, будет договориться, что самый первый список всегда содержит только head и tail, и тогда если элемент волей рандома попадает в этот список, то мы добавляем еще один уровень. Для добавления уровня достаточно склонировать head, tail и сделать новый head предком старого head.

2:

> Ситуация - вам нужно сделать очередь. Вы знаете, что в очереди никогда не будет больше, чем K элементов. На лекции мы обсудили, как сделать очередь на массиве, но этого может не хватить - пусть программа с очередью будет работать три месяца подряд, обрабатывая запросы на добавления и удаления. Если делать сдвиги указателей (или очередь на списках), нам придется делать новые аллокации - это плохо. Если делать очередь на двух стеках, то у нас будет амортизированный алгоритм - это плохо. Хочется выделить O(K) памяти один раз, и построить над ними очередь, которая за O(1) отвечает на каждый запрос. Что делать?

Это задача про такую структуру данных, как циклический буфер. Можно единожды выделить K элементов (или K+1 элемент в зависимости от задачи), завести указатели head и tail, которые двигаются так же, как и в обычной очереди на массиве, но они всегда соответствуют индексам $head \pmod{K}$ и $tail \pmod{K}$. То есть, в какой-то момент указателя переходят за последний элемент массива и автоматически телепортируются в начало массива. Таким образом, иногда head указывает на элемент с большим индексом, чем tail, потому что tail переиспользует старую память. Можно заметить, что до тех пор, пока хвост не обогнал начало на K ходов, элементы остаются в сохранности. Именно это условие и дано в задаче.