Глава 3

Data Structures

1 Arrays

Массив – структура данных, хранящая фиксированное число элементов одинакового размера, смежно расположенных в непрерывном куске памяти.

Требование одинакового размера элементов (обозначим его как S) здесь существенно, поскольку позволяет находить адрес i-го элемента по смещению от начала массива:

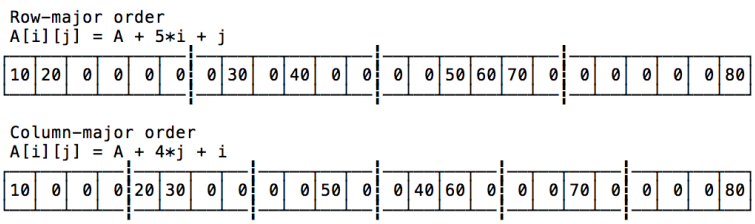
$$A_n = A_0 + S \cdot i$$

Таким образом, получение элемента массива по индексу занимает $O(1)$.

Multidimensional Arrays

Количество индексов, определяющих элемент в массиве, называется *размерностью* массива. Например, матрицу можно представить массивом размерности 2. При этом её элементы выстраиваются в линейный порядок по строкам (*row-major*, наиболее популярный, используется в частности в C и Numpy) или по столбцам (*column-major*, используется в Fortran), а в формуле смещения элемента к индексам добавляются множители, соответствующие размерностям.

$$\begin{pmatrix} 10 & 20 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 40 & 0 & 0 \\ 0 & 0 & 50 & 60 & 70 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \end{pmatrix}$$



Такое представление многомерных массивов называют *линейным*. Оно удобно для вычислений: кроме доступа к любому элементу за $O(1)$, при итерации по строке (для row-major) или столбцу (column-major) эффективно используется кэш процессора.

Sparse Matrices

Вычислительная математика часто имеет дело с *разреженными матрицами*, в которых подавляющая часть элементов – нули. Представлять такие матрицы с помощью непрерывных многомерных массивов затратно: для матрицы $M \times N$ всегда будет израсходовано $\Theta(MN)$ памяти.

Dynamic Arrays

Search by Value

2 Stacks

3 Queues

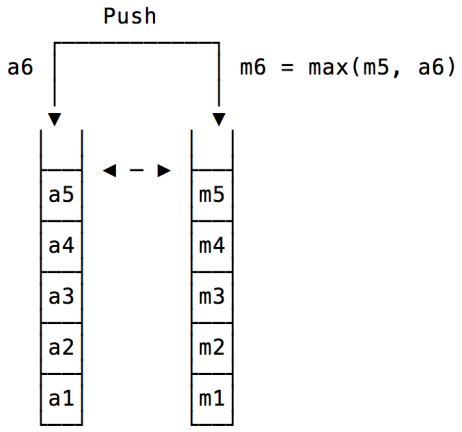
Implementation with Two Stacks

...

Dynamic Maximum

Задача. *Реализовать структуру данных, в которой добавление, удаление, и вычисление максимума (или минимума) выполняется за* $O(1)$

Для стека задача решается добавлением ещё одного стека, на вершине которого в любой момент времени будет лежать максимум текущего стека. При удалении элемента стека последний максимум тоже удаляется.

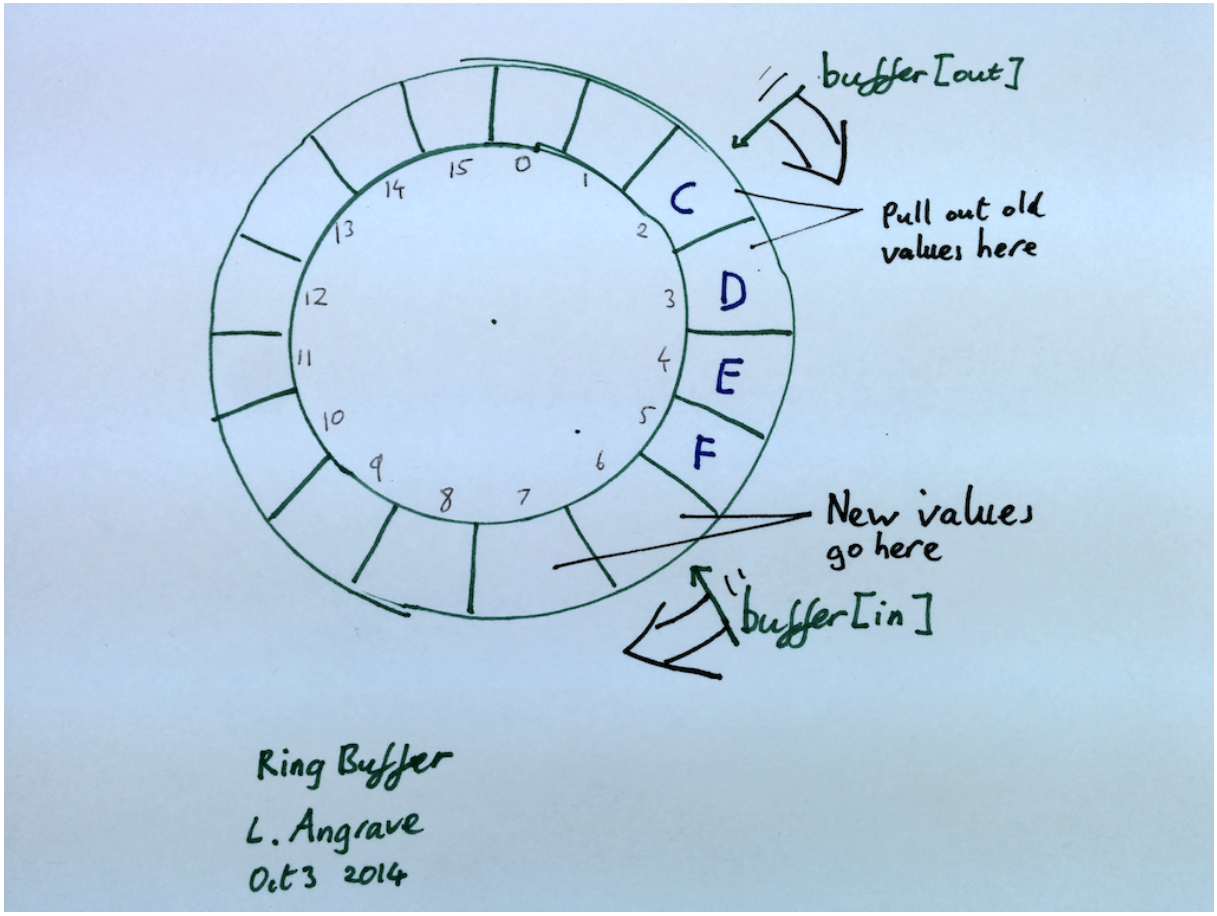


Построив из двух таких стеков очередь, её максимум можно вычислить как максимум из двух вершин добавочных стеков. Итого, добавление и удаление из очереди займёт амортизированное время $O(1)$, а получение максимума – за гарантированное $O(1)$.

Задача. *Построить очередь с максимумом, в которой* каждое *добавление и удаление выполняется за* $O(1)$

Circular/Ring Buffers

Рассмотренные нами массивы не позволяли эффективно реализовать очередь, для которой требуется удаление и добавление сразу в оба конца структуры данных. Это ограничение можно обойти с помощью



4 Linked Data Structures

При реализации структур данных через массивы проблемы с добавлением и удалением были вызваны тем, что элементы в памяти были смежными, а размер выделенной памяти – фиксированный.

Однако, пользуясь тем, что выделение области памяти происходит за $O(1)$, каждый элемент лежать в собственной, специально выделенной области памяти (объекте). При этом связь между объектами можно обеспечить с помощью ссылок из одних объектов на другие. Такие структуры данных называются *ссылочными*.

Linked Lists

Простейшая из ссылочных структур данных – *односвязный список* (singly-linked list). Он представляет упорядоченную последовательность элементов, каждый из которых содержит два поля: некоторое значение (payload) и ссылку, указывающую на следующий элемент, либо равную `None`/`null`/`0` если элемент последний в списке.

В односвязном списке доступны операции:

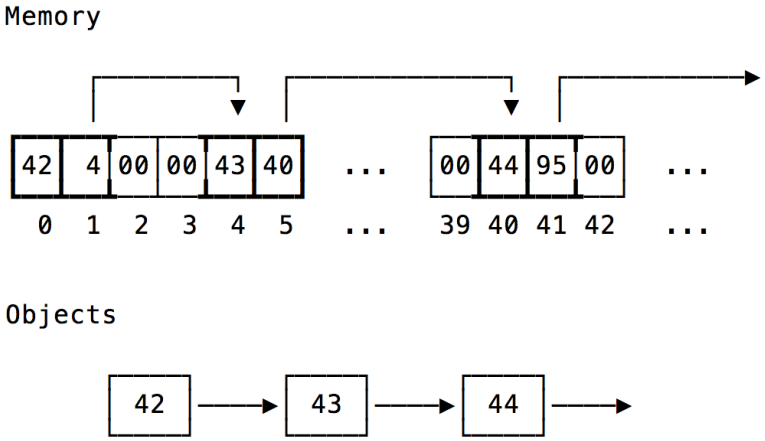


Рис. 3.1: Singly-linked list

- Добавление элемента в оба конца ($O(1)$), либо середину ($O(n)$) списка;
- Удаление элемента с начала ($O(1)$), либо конца ($O(n)$) списка.

При этом структура данных требует $O(n)$ дополнительной памяти для ссылок на следующие элементы.

Один из недостатков односвязного списка – невозможность удалить элемент *при наличии ссылки на него*. Это решается добавлением ссылок на предыдущие элементы:

Double-Ended Queues (Deque)

5 Hash Tables