Билет 25. Стек и очередь с минимумом

Курс "Алгоритмы и структуры данных"

Билет 25. Стек и очередь с минимумом

Основная идея: Научиться получать минимум за O(1) в стеке и очереди!

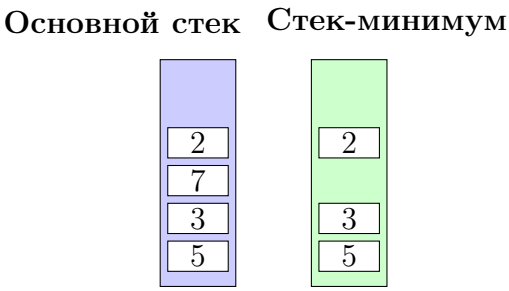
Проблема

Обычный стек/очередь умеет быстро добавлять/удалять элементы, но чтобы найти минимум, нужно O(n).

Хотим: Все операции за O(1), включая получение минимума!

1. Стек с минимумом

Стек-помощник



Алгоритм работы

- **Добавление (push(x)):**

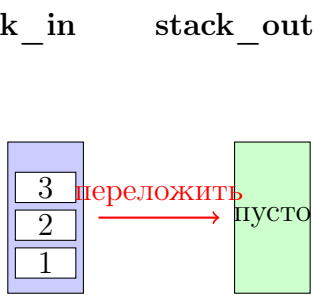
- В основной стек: всегда добавляем
- В стек-минимум: добавляем только если **x <= текущий минимум**

- **Удаление (pop()):**
  - Из основного стека: всегда удаляем
  - Из стека-минимума: удаляем только если **удаляемый элемент = текущий минимум**

- **Получение минимума (getMin()):** O(1) - просто смотрим верх стека-минимума

2. Очередь на двух стеках

Идея: Эмулируем очередь двумя стеками



Алгоритм работы

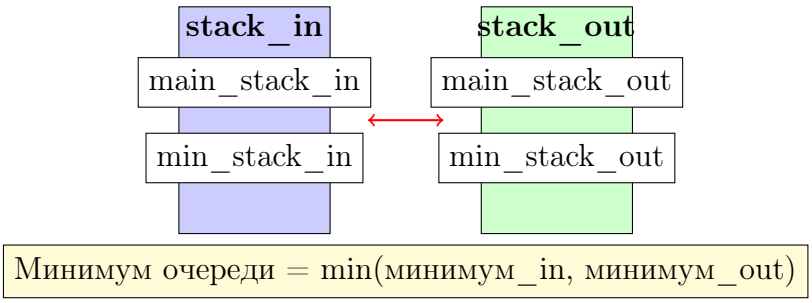
- **Добавление (enqueue(x)):** push в `stack_in`
- **Удаление (dequeue()):**
  - Если `stack_out` пуст: перекладываем ВСЕ элементы из `stack_in` в `stack_out`
  - pop из `stack_out`

Пример

Добавляем 1, 2, 3:
stack in: [1, 2, 3] stack out: []
stack out пустой, поэтому перекладываем в обр порядке:
stack in: [] stack out: [3, 2, 1]
Забираем 1:
stack in: [] stack out: [3, 2]

3. Очередь с минимумом

Система из 4 стеков



Структура данных

- **stack_in:** Для добавления элементов
  - `main_stack_in[]` - основные данные
  - `min_stack_in[]` - минимумы (как в стеке с минимумом)
- **stack_out:** Для удаления элементов
  - `main_stack_out[]` - основные данные
  - `min_stack_out[]` - минимумы (как в стеке с минимумом)

Операции

- **enqueue(x):**
  - Добавляем x в `main_stack_in`

- В `min_stack_in` добавляем **min(x, предыдущий минимум _in)**

- **dequeue():**
  - Если `stack_out` пуст: перекладываем ВСЕ элементы из `stack_in` в `stack_out`
  - Удаляем из `stack_out`

- **getMin():** O(1)

$$\text{min\_queue} = \min(\text{min\_in}, \text{min\_out})$$

Если какой-то стек пуст, берём минимум другого стека.

Пример работы очереди с минимумом

Операция	**stack_in**	**stack_out**	**Минимумы**	**min_queue**
enqueue(5)	[5]	[]	min_in=5, min_out=	5
enqueue(3)	[5,3]	[]	min_in=3, min_out=	3
enqueue(7)	[5,3,7]	[]	min_in=3, min_out=	3
dequeue()	[]	[7,3]	min_in=, min_out=3	3
dequeue()	[]	[7]	min_in=, min_out=7	7
enqueue(2)	[2]	[7]	min_in=2, min_out=7	2
enqueue(1)	[2,1]	[7]	min_in=1, min_out=7	1

Анализ сложности

- **Все операции:** O(1) амортизированно
- **Память:** O(n) - каждый элемент хранится в одном из стеков
- **Перекладывание:** Каждый элемент перекладывается максимум 2 раза (из in в out)

Преимущества

Преимущества:

- Все операции за O(1) амортизированно
- Простая реализация на основе стандартных стеков
- Минимум доступен мгновенно