

## Лекция по АлСД

Разбор задачи о минимуме в очереди. Бинарный поиск.

# Минимум в очереди

Реализовать очередь с операциями

1. Push (добавляем в конец)
2. Pop (удаляем из начала)
3. GetMin (узнать минимум среди всех элементов в очереди)

# Минимум в очереди

Тривиальное решение

1. Заведем deque
2. Push – `deque.push()`
2. Pop – `deque.popleft()`
3. GetMin – Бежим по всему deque, честно ищем минимум

Сложность GetMin –  $O(n)$ , хотим быстрее ( $O(1)$ )

## Подзадача: Минимум в стеке

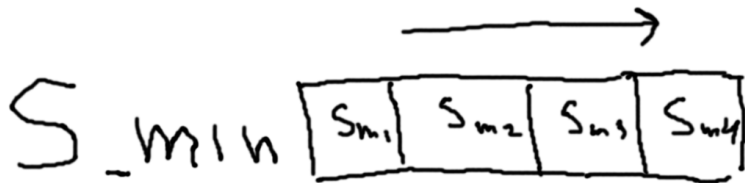
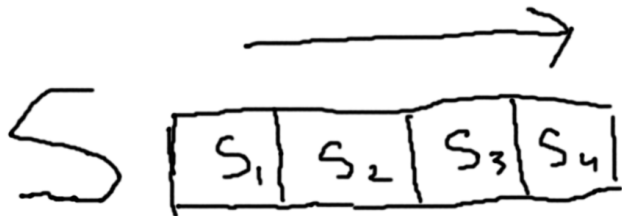
Реализовать стек с операциями

1. Push (добавляем в конец)
2. Pop (удаляем из конца)
3. GetMin (узнать минимум среди всех элементов в стеке)

Все операции хотим за  $O(1)$

## Минимум в стеке

Заведем вспомогательный стек для минимумов



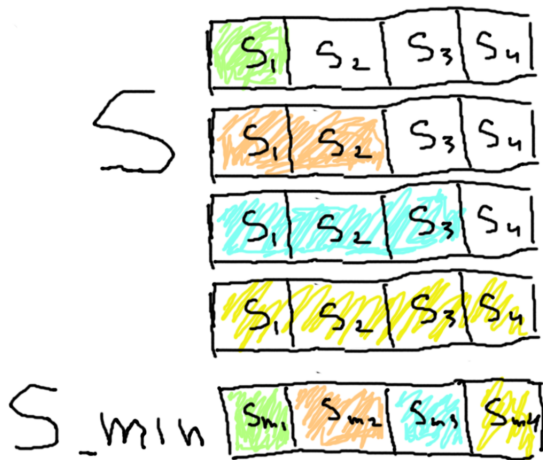
$$S_{min} = \text{Min}(S_0, S_1, \dots, S_i)$$

## Минимум в стеке

$$S_{min} = \text{Min}(S_0, S_1, \dots, S_i)$$

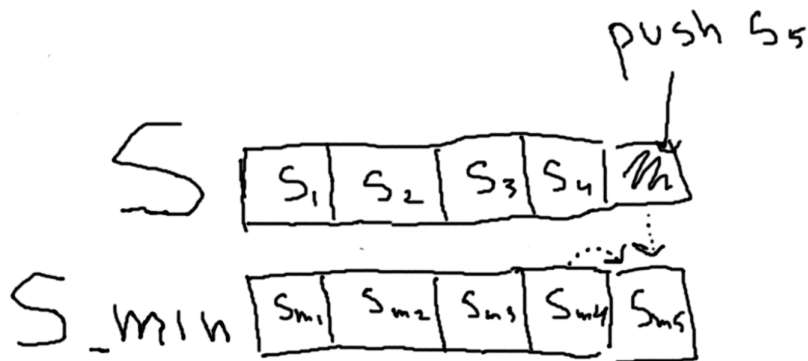


## Минимум в стеке



Заметим, что  $S_{min}$  – минимум для всего стека  $S$   
Тогда ответ на запрос GetMin – вершина стека  $S_{min}$

Как поддерживать стек минимумов?



$S_{min5} = \text{Min}(S_{min4}, S_5)$  – минимум для всего стека  $S$   
pop – просто удаляем элементы из обоих стеков



## Пример

↪ [4]

↪<sub>m</sub> [4]

push 4

push 2

push 3

pop

pop

## Пример



push 4

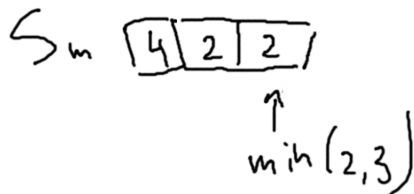
push 2

push 3

pop

pop

## Пример



push 4

push 2

push 3

pop

pop

## Пример



push 4

push 2

push 3

pop

pop

## Пример



push 4

push 2

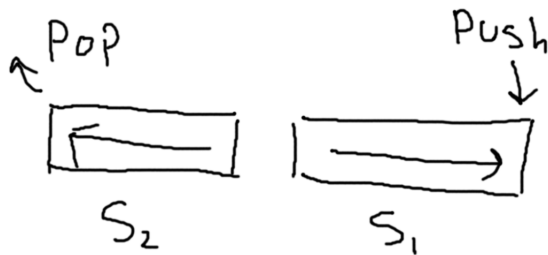
push 3

pop

pop

## Очередь из 2х стеков

Заведём 2 стека –  $S_1$  и  $S_2$



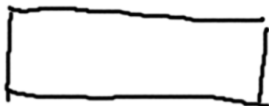
push будем делать в  $S_1$

pop будем делать из  $S_2$

Если  $S_2$  пуст, переложим в него все элементы из  $S_1$

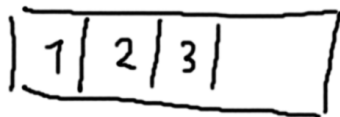
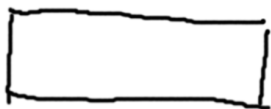
## Пример

1 2 3 P P 4 P 5-



## Пример

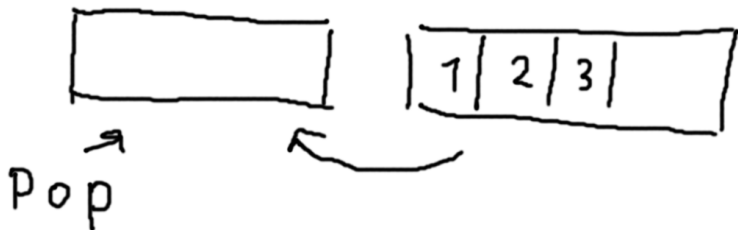
1 2 3 P P 4 P 5





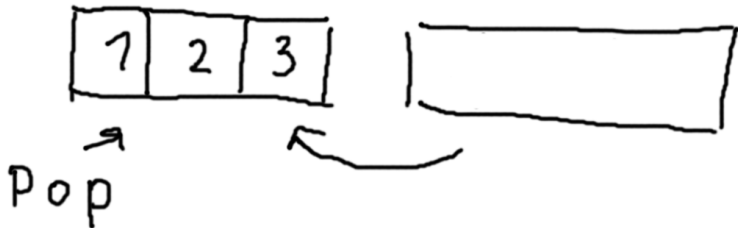
## Пример

1 2 3 P P 4 P 5-

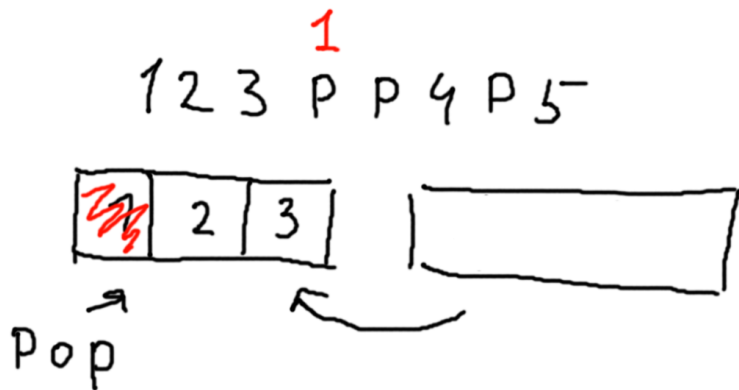


## Пример

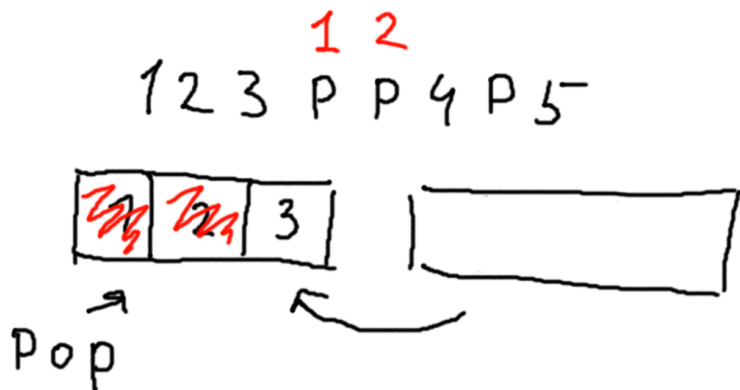
1 2 3 P P 4 P 5



## Пример

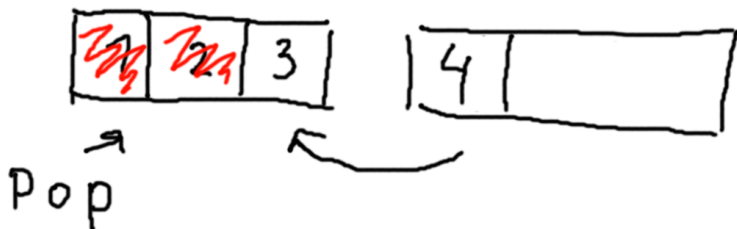


## Пример

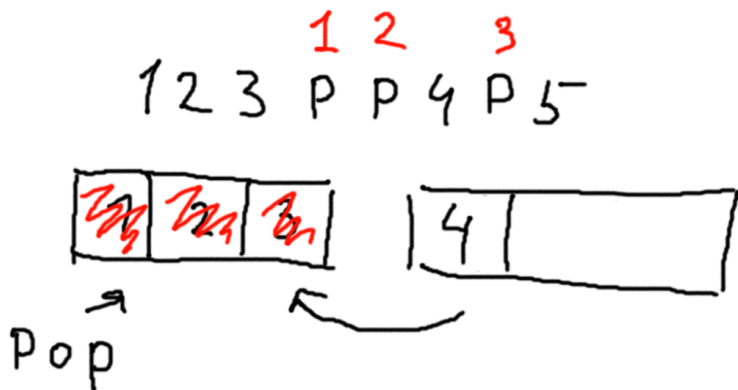


## Пример

1 2 3 P P 4 P 5



## Пример



## Очередь из 2х стеков

Можно заметить, что такая конструкция сохраняет порядок очереди.

Учетная стоимость push и pop при этом равна  $O(1)$

## Очередь из 2х стеков: Оценка времени работы

Воспользуемся банковским методом

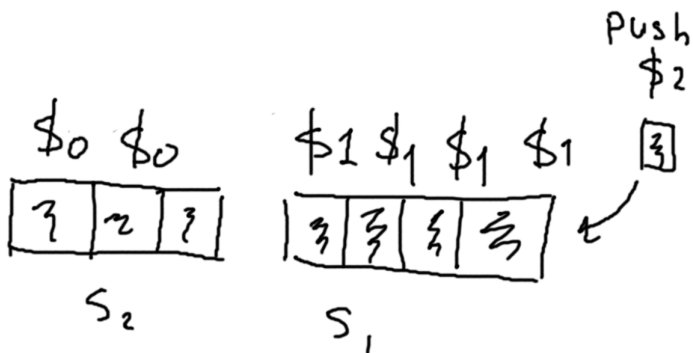
Будем платить \$2 за push, \$1 тратить сразу, \$1 сохраним для перекладывания элементов (тогда над каждым элементом в  $S1$  будет \$1)

Если нам нужно переложить элементы из  $S1$  в  $S2$ , возьмем деньги, которые лежат над элементами в  $S1$

Можно заметить, что каждый элемент мы переложим только 1 раз



## Очередь из 2х стеков: Оценка времени работы



## Минимум в очереди

Для реализации очереди, поддерживающей минимум, сделаем очередь из 2х стеков, поддерживающих минимум ( $S1$  и  $S2$ )

Теперь  $GetMin$  в очереди –  $Min(GetMin(S1), GetMin(S2))$

# Бинарный поиск

Пусть у нас есть отсортированный массив  $a_1, a_2, \dots, a_n$ ,  
 $a_i < a_{i+1}$ .

Мы хотим найти в нем число  $x$

## Тривиальное решение

Пишем цикл

```
for elem in a:
    if elem == x:
        return True
return False
```

Сложность –  $O(n)$ , хотим быстрее

# Бинарный поиск

Рассмотрим некоторый элемент массива  $a_i$  и сравним его с  $x$   
Вспомним, что массив отсортирован

1. Если  $a_i < x$ , то и все  $a_j < x, j < i$  (Левее  $i$  нашего элемента точно нет)
2. Если  $a_i > x$ , то и все  $a_j > x, j > i$  (Правее  $i$  нашего элемента точно нет)
3. Если  $a_i = x$ , то мы нашли наш элемент

# Бинарный поиск

$$a_i < x$$



$$a_i > x$$



# Бинарный поиск

Пусть  $l, r$  – границы отрезка, в котором мы сейчас ищем элемент

Изначально  $l = 0, r = \text{len}(a)$ , то есть ищем во всем массиве

Рассмотрим середину текущего отрезка  $a[m], m = (l + r) // 2$  1.

Если  $a_m < x$ , то продолжим поиск только в правой половине массива ( $l = m$ )

2. Если  $a_m > x$ , то продолжим поиск только в левой половине массива ( $r = m$ )

3. Если  $a_m = x$ , то мы нашли наш элемент

Остановимся, когда  $l + 1 = r$  (когда мы сошлись к отрезку из одного элемента)

Сложность –  $O(\log \text{len}(a))$

## Пример

$$X = 7$$

0	2	3	5	7	11	18
---	---	---	---	---	----	----

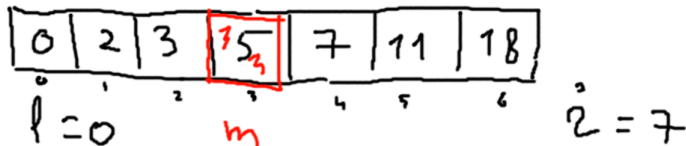
$$l = 0$$

$$r = 7$$



## Пример

$$X = 7$$



$$m = \frac{0 + 7}{2} = 3$$

$$a[m] < x \quad \Rightarrow \quad l = m$$

5                  7

## Пример

$$X = 7$$

<del>0</del>	<del>2</del>	<del>3</del>	5	7	11	18
0	1	2	3	4	5	6

$$l = 3$$

$$r = 7$$

# Пример

$$X = 7$$

<del>0</del>	<del>2</del>	<del>3</del>	5	7	11	18
0	1	2	3	4	5	6

$$l = 3$$

$$r = 7$$

$$m = \frac{l + r}{2} = 5$$

$$a[m] > X \Rightarrow r = m$$

11                      7

## Пример

$$X = 7$$

<del>0</del>	<del>2</del>	<del>3</del>	5	7	11	<del>18</del>
0	1	2	3	4	5	6

$l = 3$        $r = 5$

## Пример

$$X = 7$$

<del>0</del>	<del>2</del>	<del>3</del>	5	7	11	<del>18</del>
<del>0</del>	<del>1</del>	<del>2</del>	3	4	5	<del>6</del>

$$l = 3 \quad \uparrow \quad r = 5$$

$$m = \frac{l + r}{2} = \frac{3 + 5}{2} = 4$$

$$\underline{a[m] = X!}$$

## Бинарный поиск: код

```
def binary_search(a, k):  
    l = 0  
    r = len(a)  
  
    while (r - l > 1):  
        m = l + (r - l) // 2  
        if a[m] > k:  
            r = m  
        else:  
            l = m  
    return a[l] == k
```