




NATIONAL RESEARCH
UNIVERSITY

АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

Базовые структуры данных (часть 1 из 2)

Рамон Антонио Родригес Залепинос
arodriges@hse.ru

Базовые структуры данных

- стек – *проход Грэхема, системное прогр., ...*
- список
 - односвязный
 - двусвязный
 - с ограничителями
 - XOR список
 - Y-связный список
 - циклический список
 - *часто используются*
 - *must know (собеседования; рассмотрим)*
- очередь – *вторая СД по частоте исполз. после списка*
- дек – *редко, некоторые алгоритмы*
- двоичное дерево поиска – *основа для понимания всех видов деревьев*

Особенности

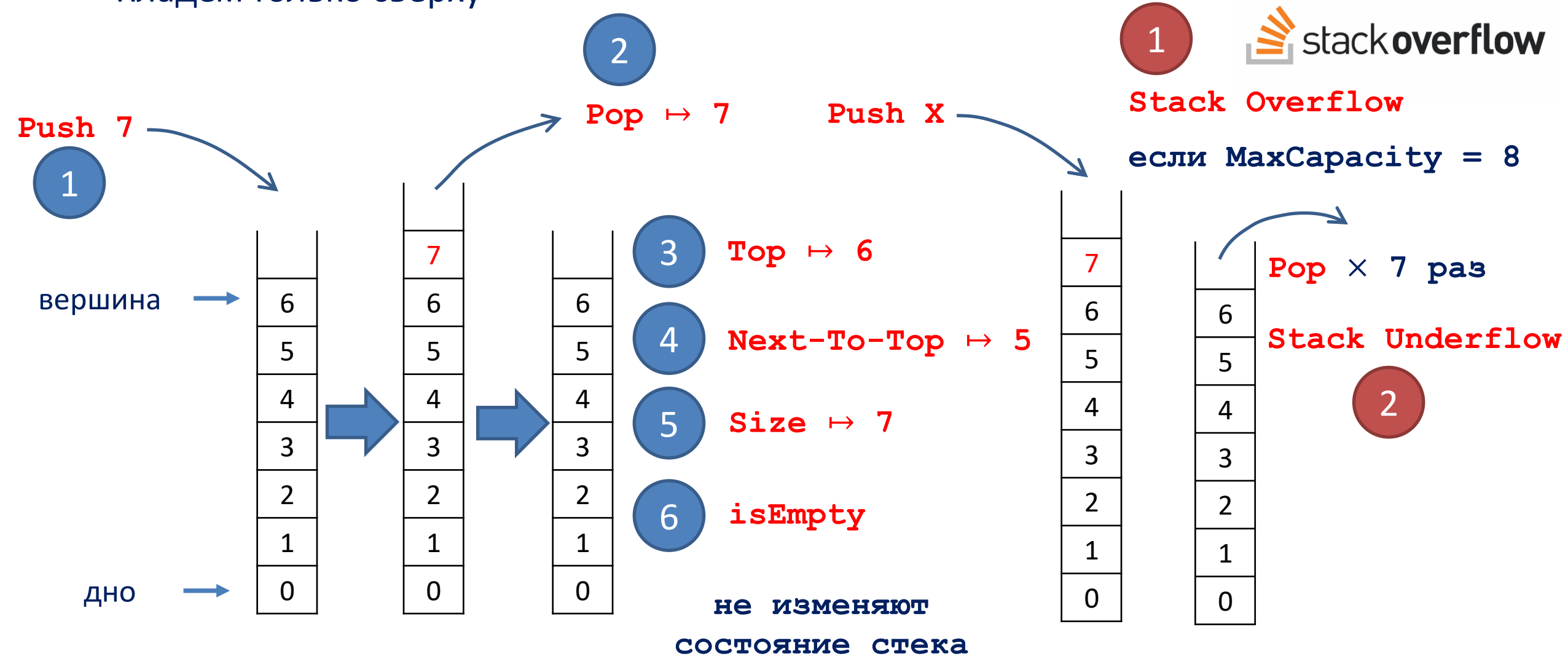
- Хранят все данные в оперативной памяти
- Можно реализовать с помощью массива либо друг через друга

Стек / Stack

- Ассоциация: стопка бумаг
- Берем только сверху
- Кладем только сверху

Все операции за $O(1)$

LIFO = Last Input,
First Output

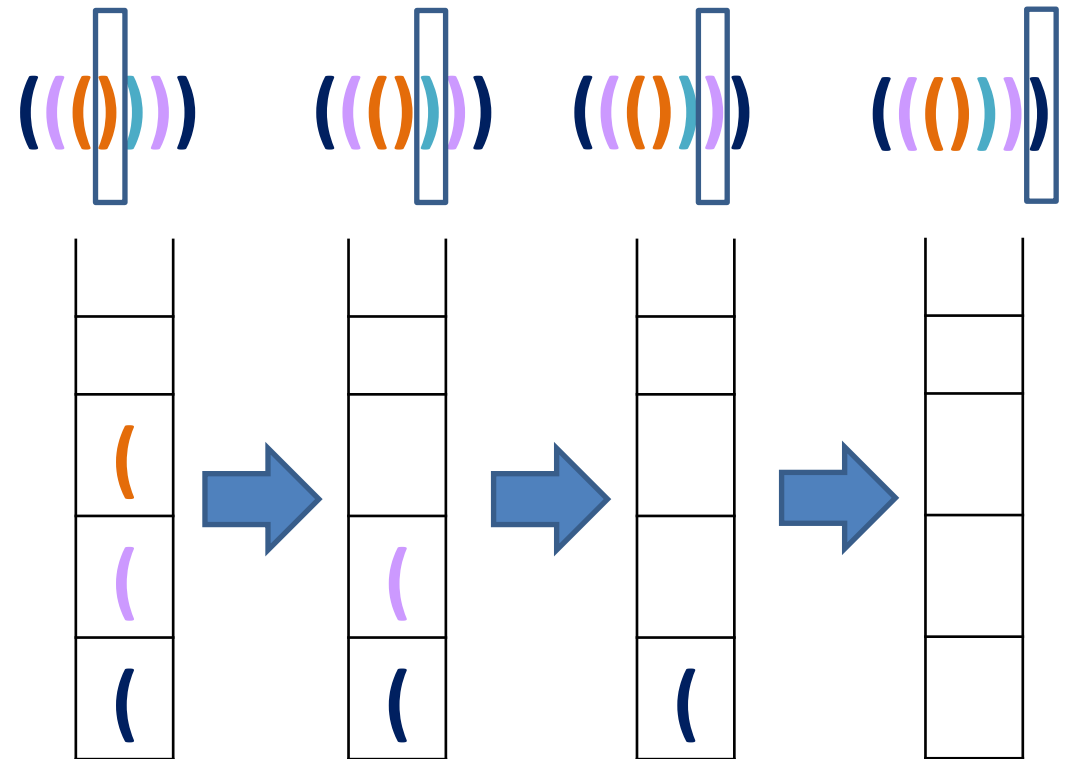


Проверка расстановки скобок

- Вход: последовательность скобок, напр. **(())()((()()))**
- Условие: у каждой открывающейся скобки должна быть соответствующая закрывающаяся скобка
- Выход: true/false (верно/неверно)

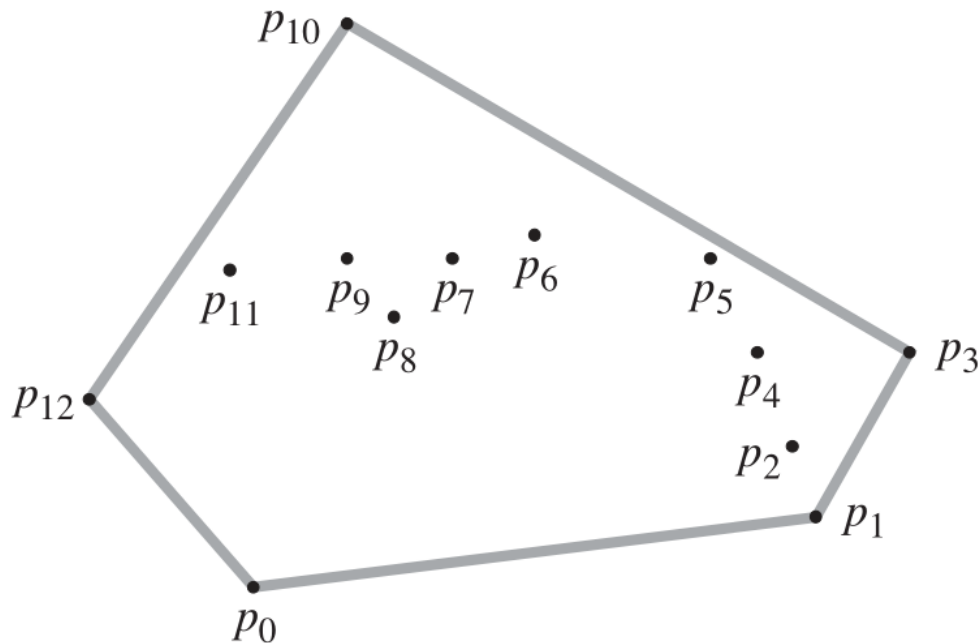
Решение:

- заносить открывающуюся скобку в стек
- извлекать из стека скобку, если встретили закрывающуюся скобку
- ошибка, если StackUnderflow



Выпуклая оболочка / Convex Hull (CH)

- Пусть Q – множество точек (двумерные координаты)
- Пусть $CH(Q)$ – выпуклая оболочка множества точек Q
- Выпуклая оболочка – наименьший выпуклый многоугольник P такой, что $\forall p \in Q$ находится либо на границе P либо внутри него.
- Многоугольник P – выпуклый, т.е. все вершины P лежат по одну сторону от любой прямой, проходящей через две его соседние вершины.



Выпуклая оболочка для
множества точек

$$Q = \{p_1, p_2, \dots, p_{12}\}$$

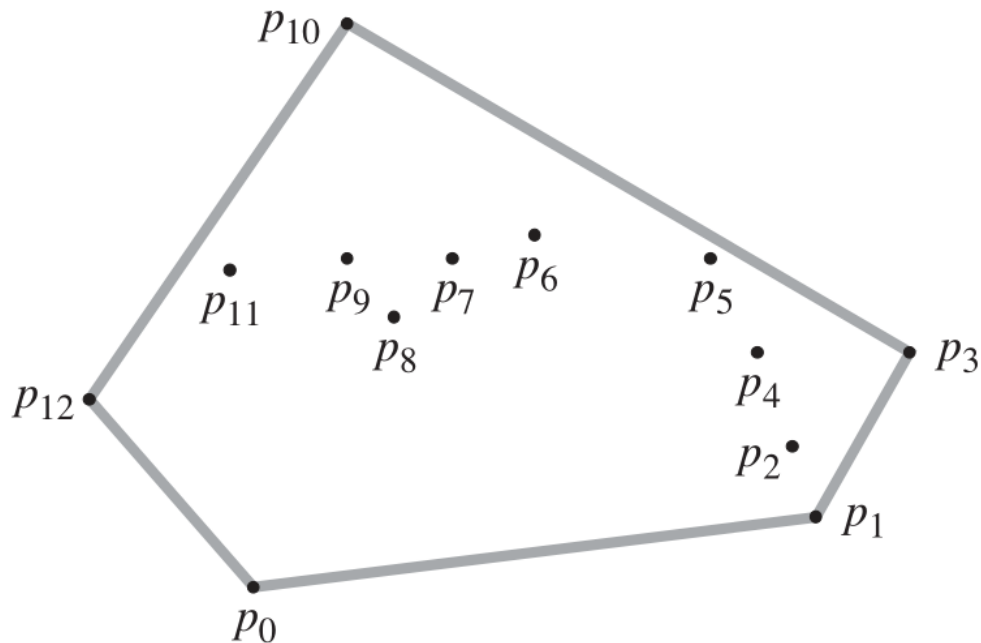
Выпуклый многоугольник
образован вершинами
 $\{p_0, p_1, p_3, p_{10}, p_{12}\}$

Пример использования

- Найти две самые удаленные друг от друга вершины

Сканирование по Грэему

- Использует **стек**
- Результат – стек с вершинами выпуклой оболочки (в стеке расположены в порядке обхода против часовой стрелки, если смотреть снизу вверх ↑)



p_{12}
p_{10}
p_3
p_1
p_0

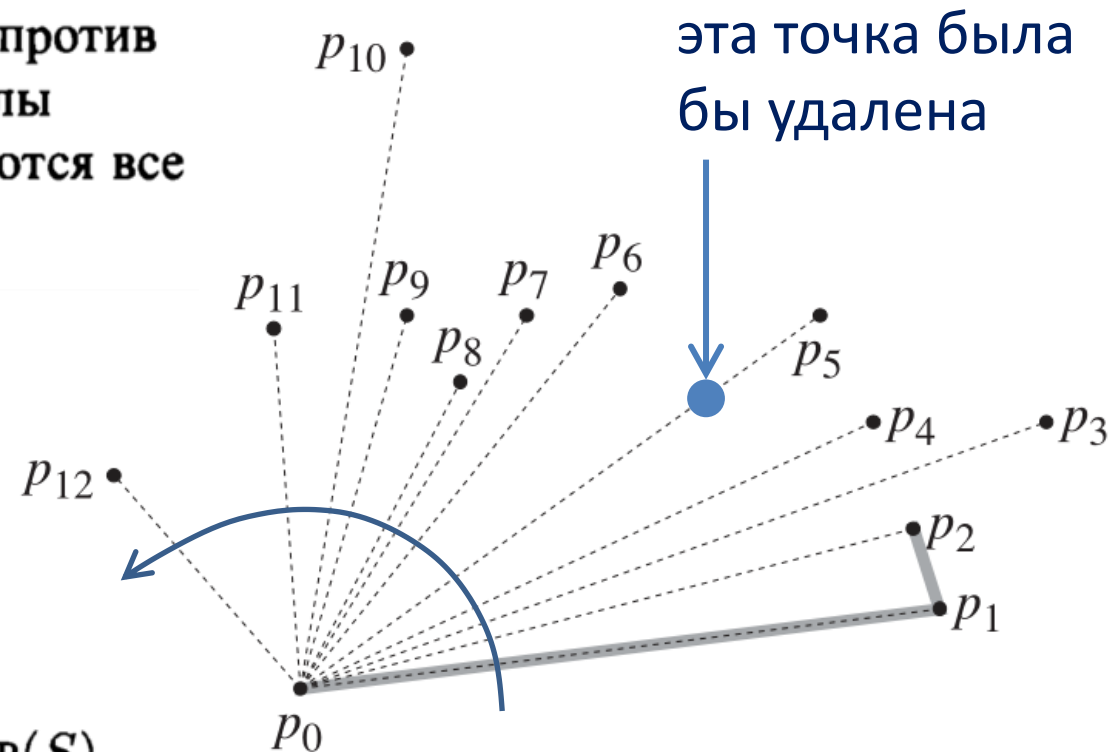
Выпуклая оболочка из точек
 $\{p_0, p_1, p_3, p_{10}, p_{12}\}$

GRAHAM-SCAN(Q)

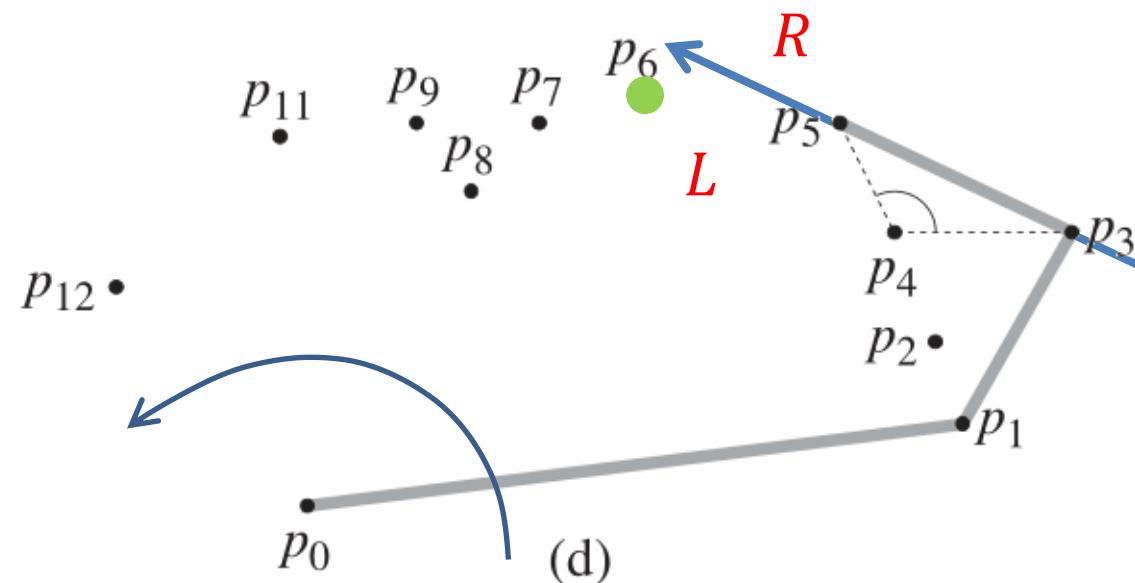
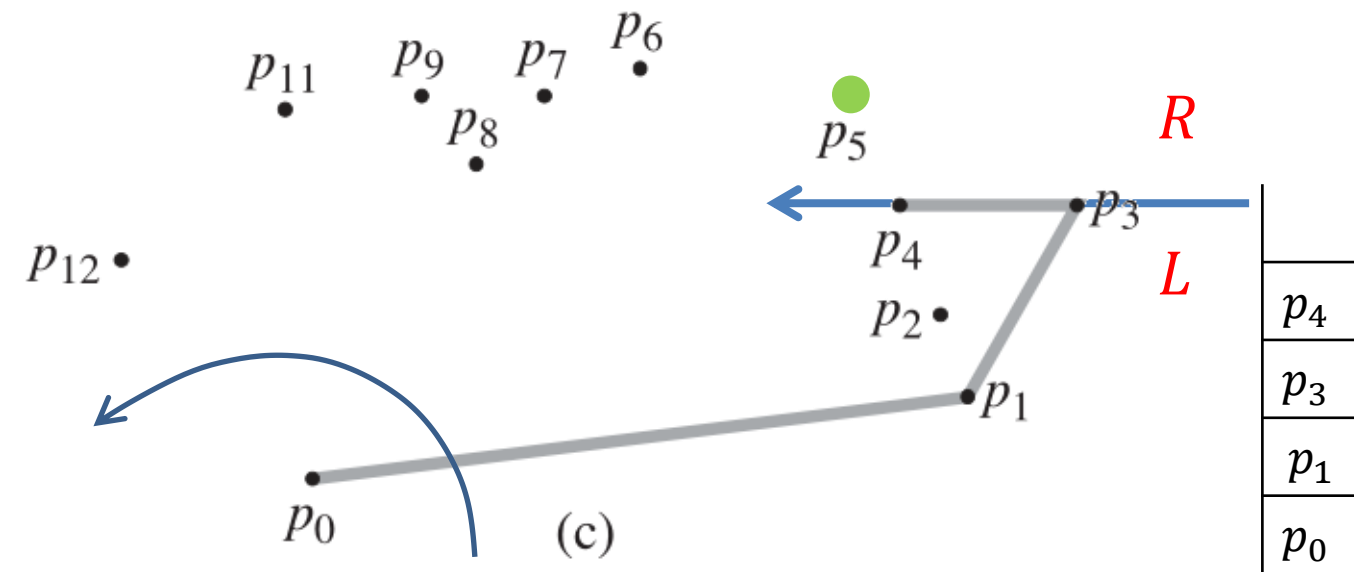
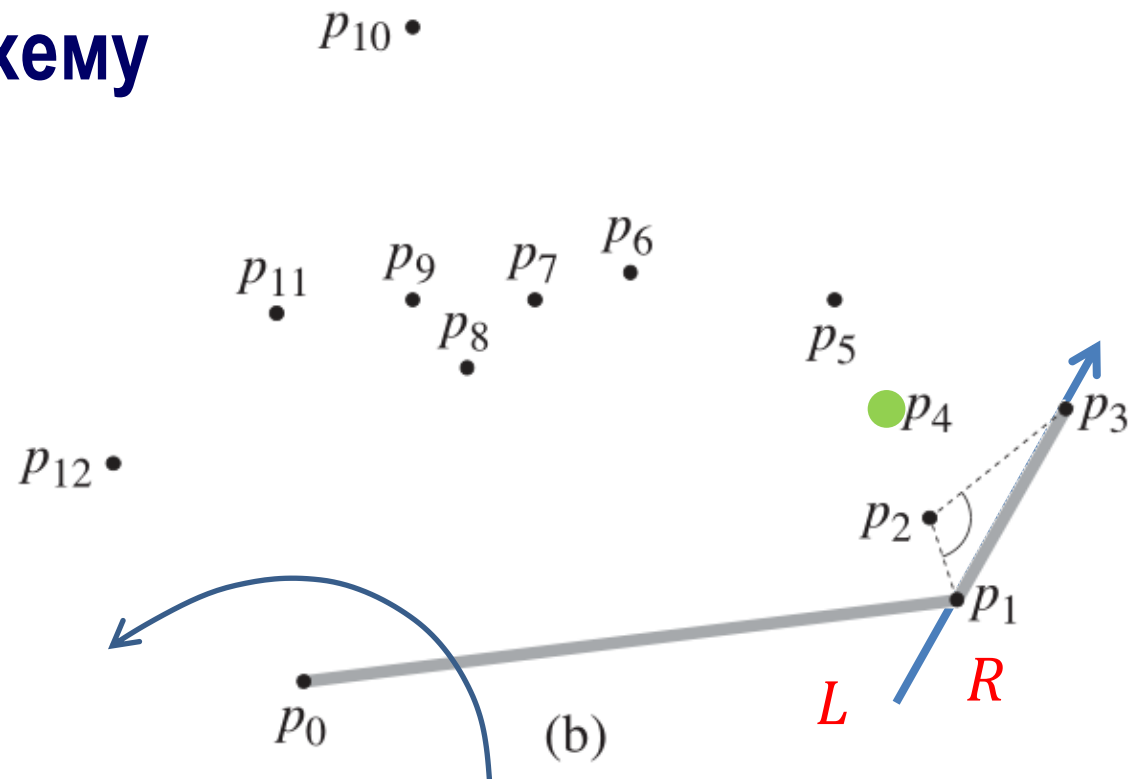
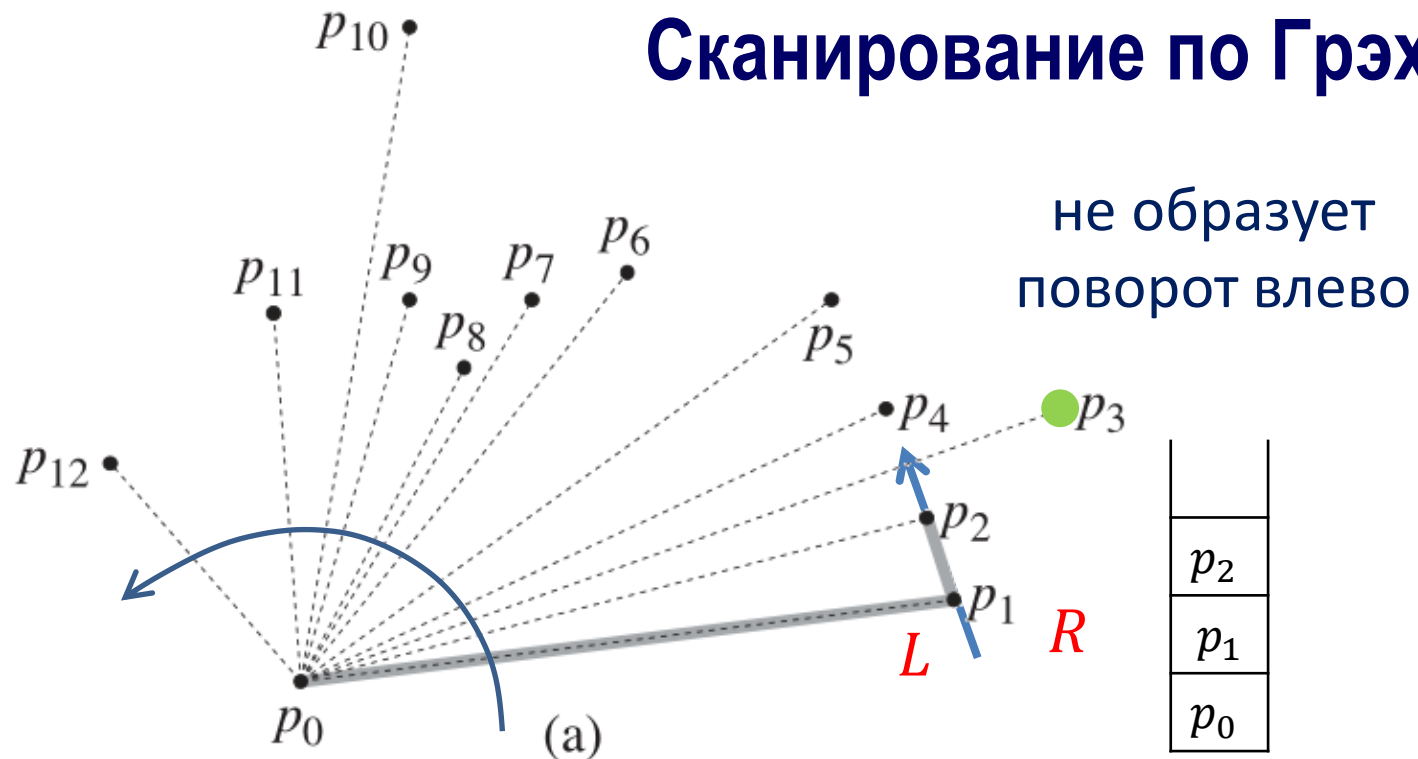
- 1 Пусть p_0 — точка Q с минимальной координатой y , или крайняя слева из таких точек при наличии совпадений
- 2 Пусть $\langle p_1, p_2, \dots, p_m \rangle$ — остальные точки Q , отсортированные в порядке возрастания полярного угла, измеряемого против часовой стрелки относительно p_0 (если полярные углы нескольких точек совпадают, то из множества удаляются все эти точки, кроме одной, самой дальней от точки p_0)
- 3 **if** $m < 2$
- 4 **return** “выпуклая оболочка пуста”
- 5 **else** пусть S — пустой стек
- 6 PUSH(p_0, S)
- 7 PUSH(p_1, S)
- 8 PUSH(p_2, S)
- 9 **for** $i = 3$ **to** m
- 10 **while** угол, образованный точками NEXT-TO-TOP(S), TOP(S) и p_i не образует поворот влево
- 11 POP(S)
- 12 PUSH(p_i, S)
- 13 **return** S

Сканирование по Грэему

– сначала сортируем по Y , только потом по X (нижняя слева точка)



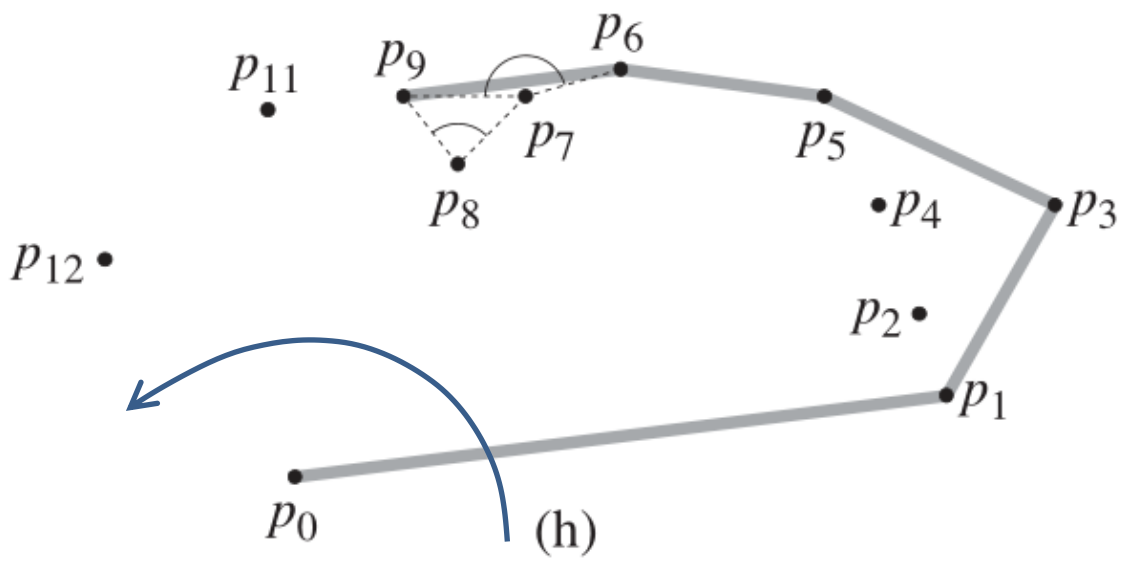
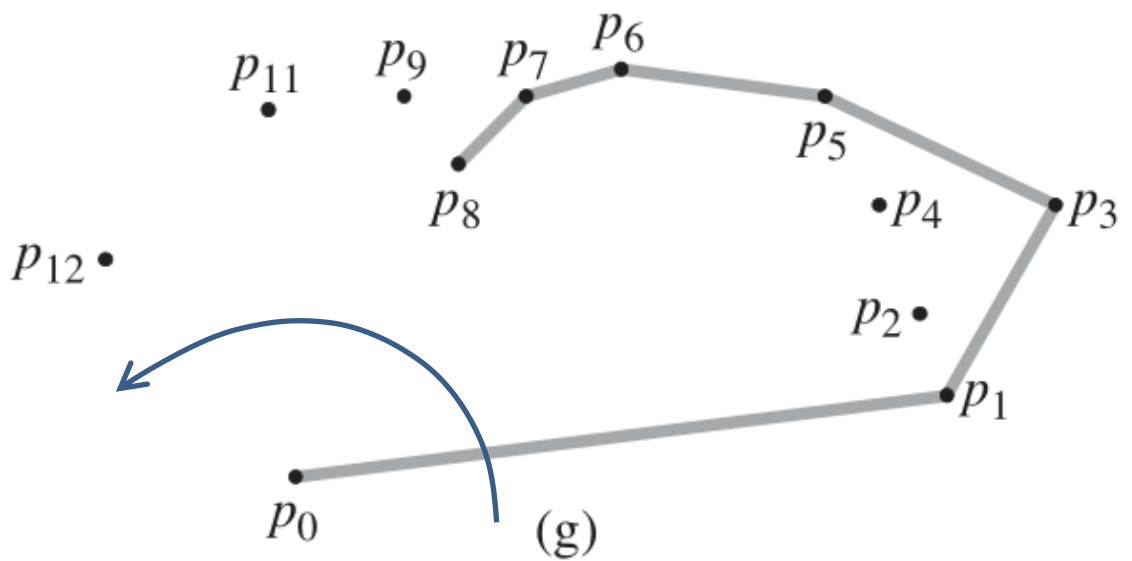
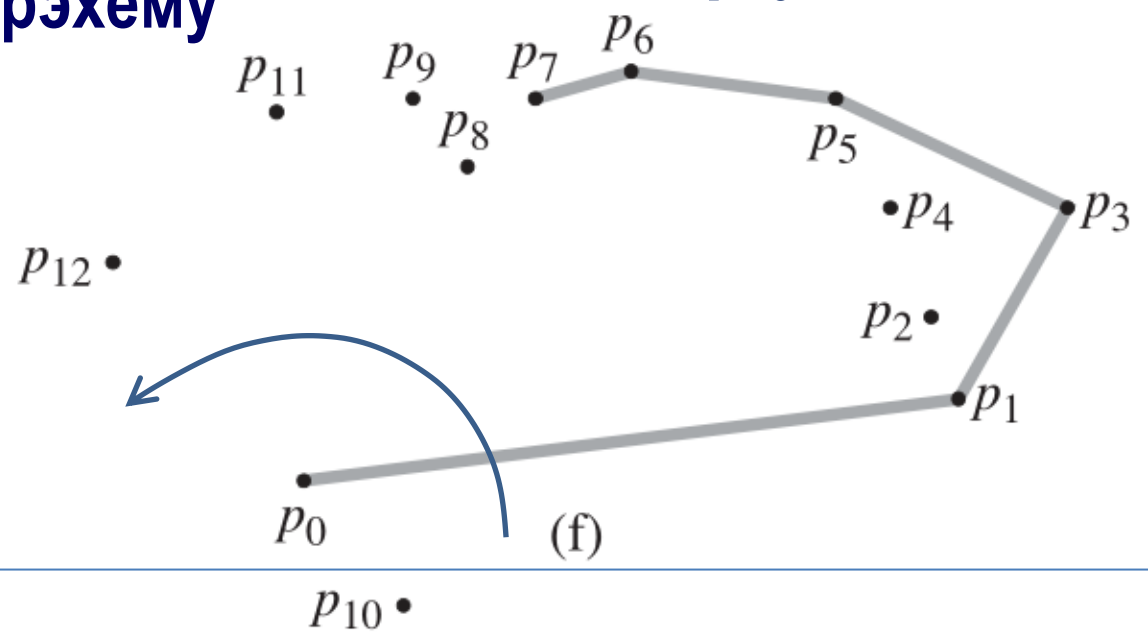
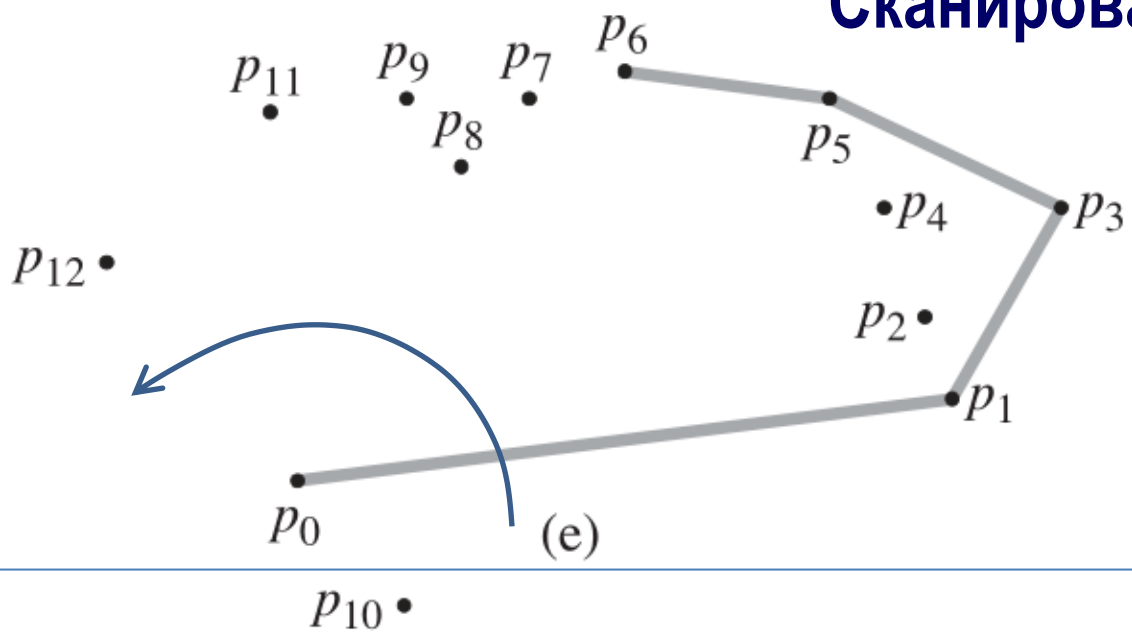
Сканирование по Грэему



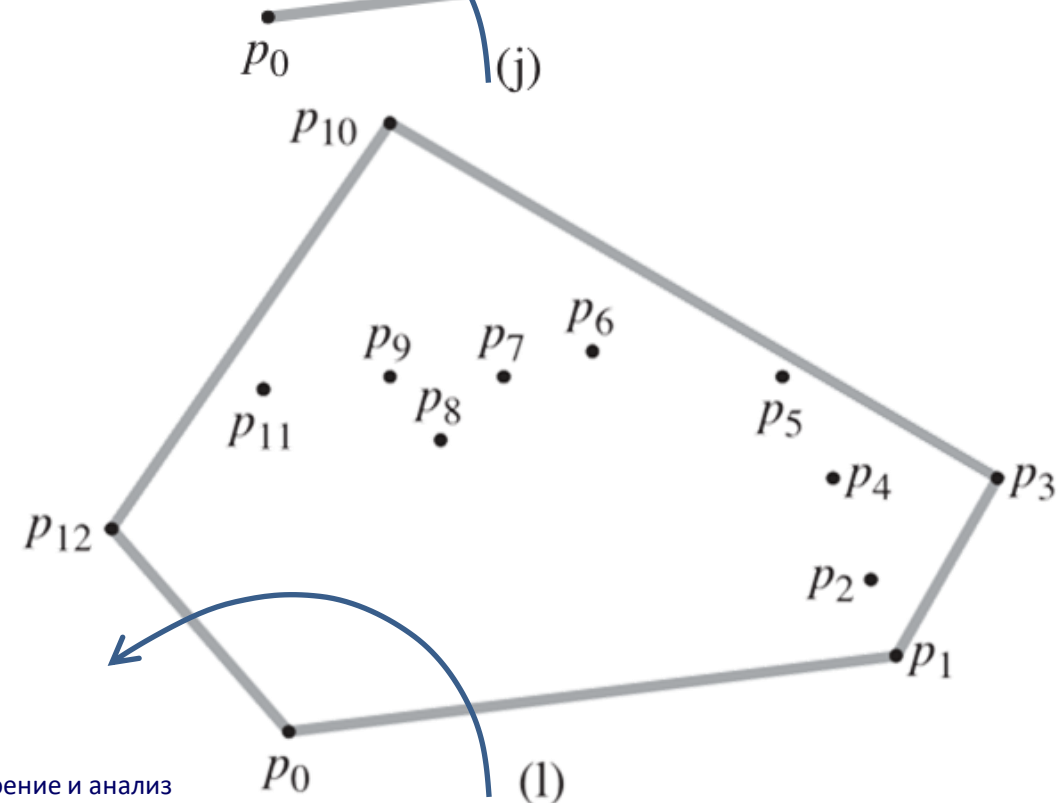
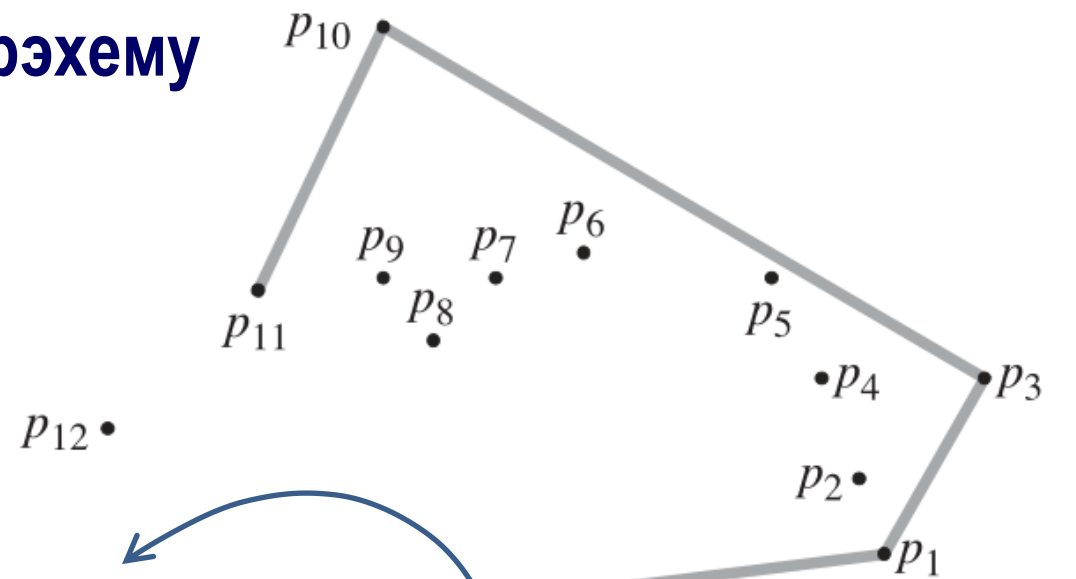
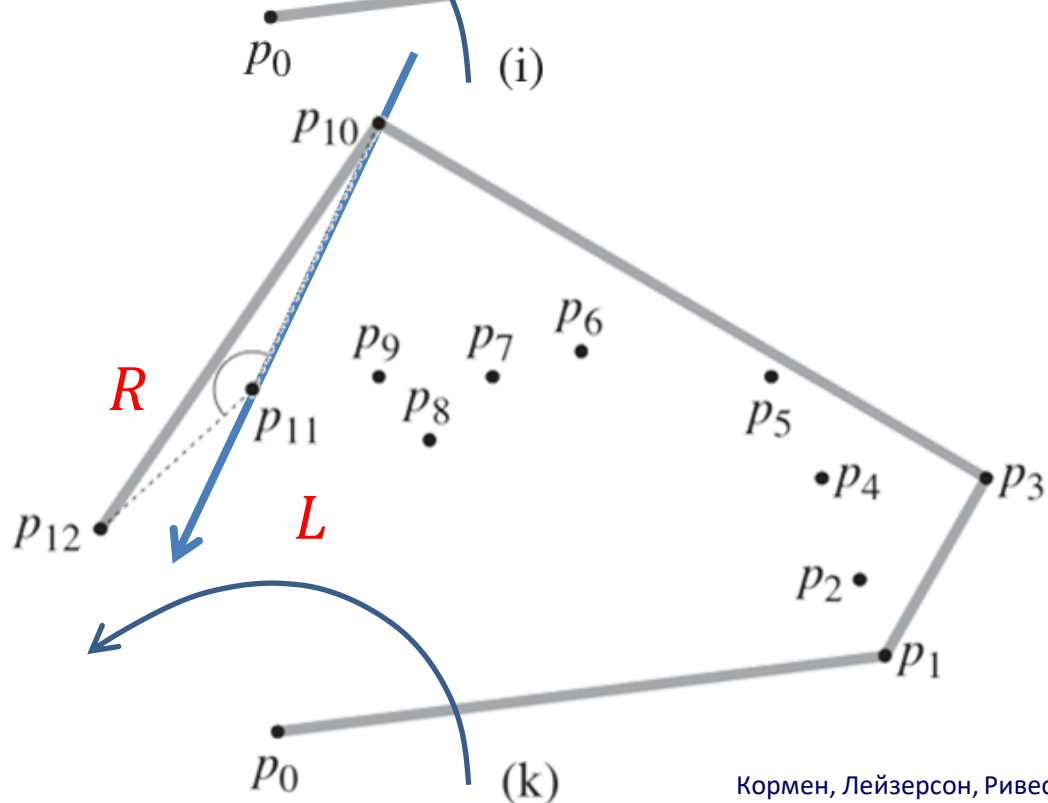
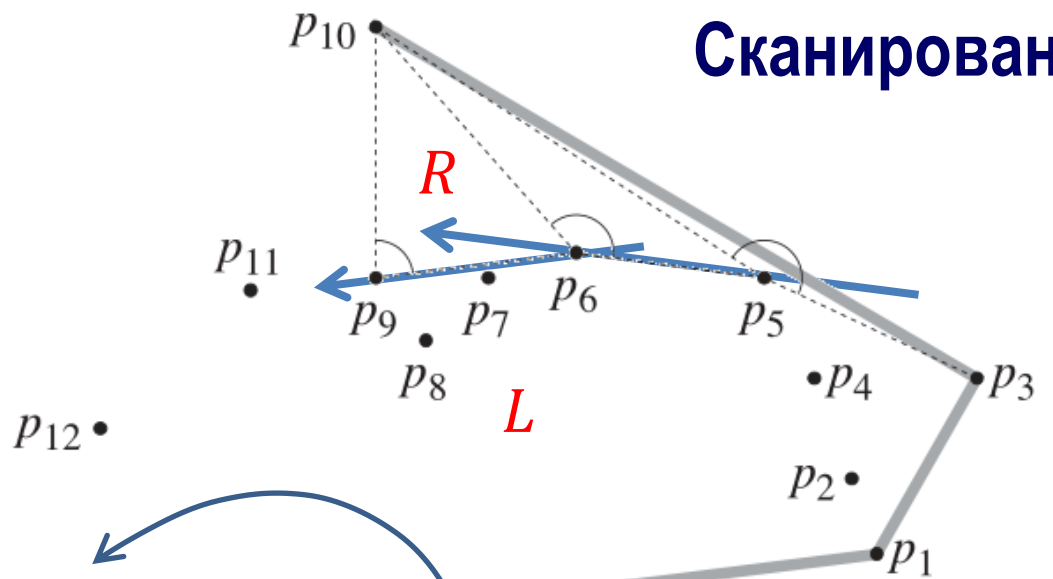
p_{10} не показана

Сканирование по Грэему

p_{10} не показана



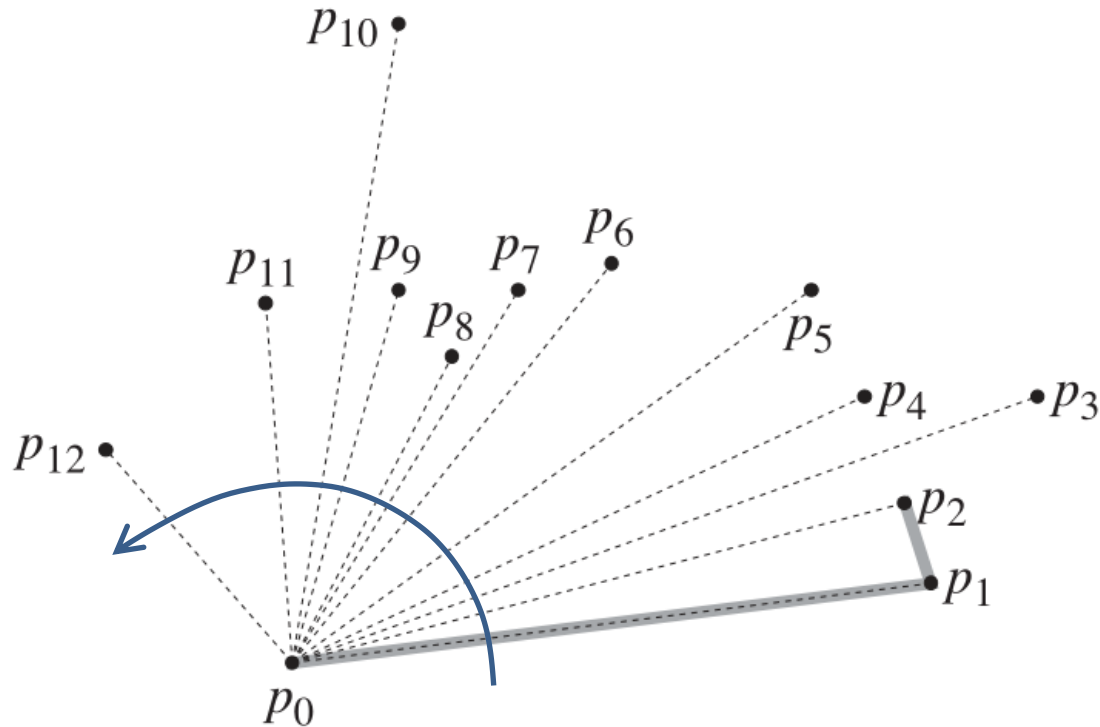
Сканирование по Грэему



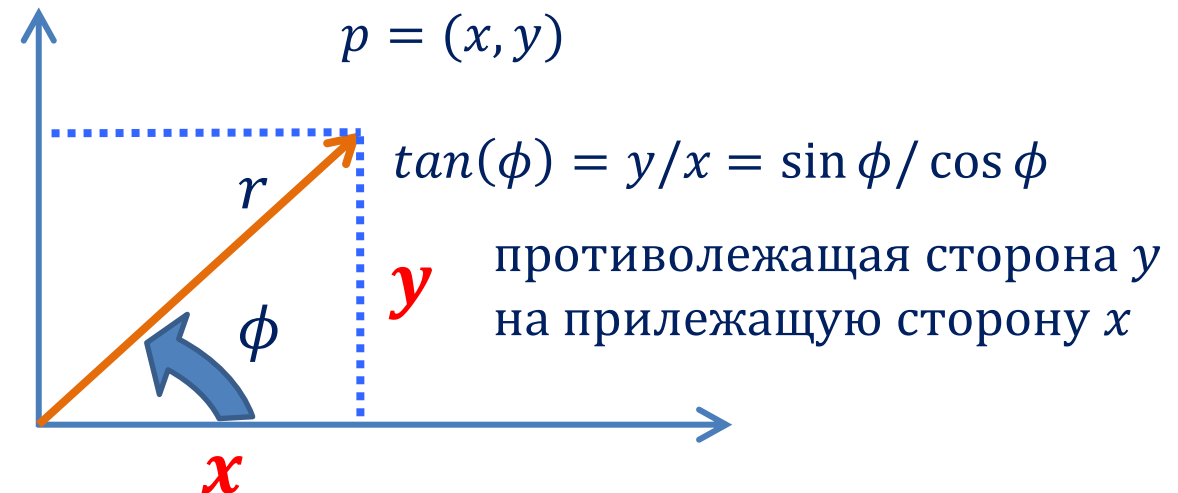
Полярный угол

Функция `atan2(double y, double x)` вычисляет значение арктангенса y/x в пределах $(-\pi; \pi]$. Если $x = 0$, или оба параметра равны нулю, то функция возвращает 0. Если полярный угол нужен в интервале $[0; 2\pi)$, то в случае отрицательного значения функции `atan2` к результату следует прибавить 2π .

```
double res = atan2(y, x);  
if (res < 0) res += 2 * PI;  
return res;
```



$$\begin{cases} x = r \cos \phi \\ y = r \sin \phi \end{cases}$$

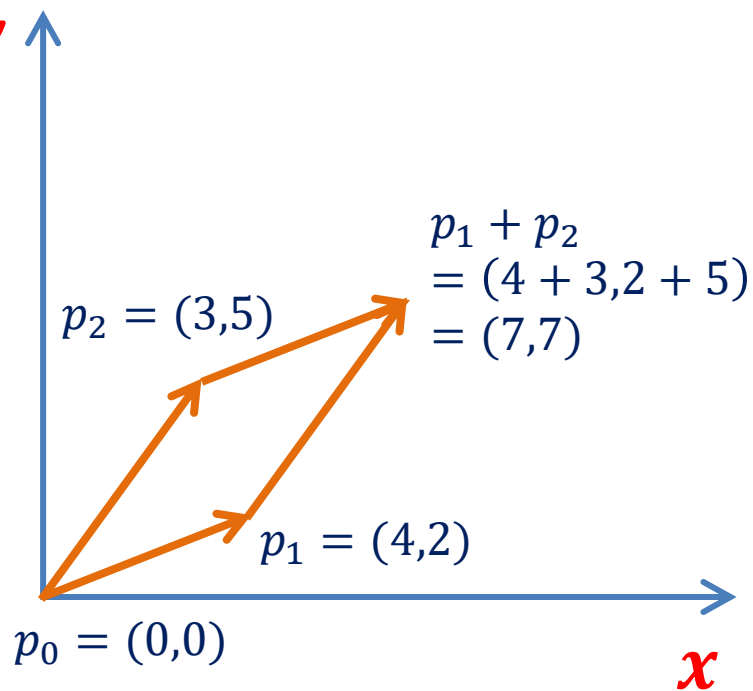


Векторное произведение, повороты влево и вправо

- **Вход:** векторы $p_1 = (x_1, y_1)$ и $p_2 = (x_, y_2)$
- **Выход:** $p_1 \times p_2 = x_1 y_2 - x_2 y_1$
- **Интерпретация:** площадь параллелограмма, образованного точками $(0, 0)$, p_1 , p_2 , $p_1 + p_2$
- **Однако** $p_1 \times p_2$ **имеет знак**

$$\begin{aligned} p_1 \times p_2 &= \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} \\ &= x_1 y_2 - x_2 y_1 \\ &= -p_2 \times p_1 \end{aligned}$$

$$\begin{aligned} p_1 \times p_2 &= \det \begin{pmatrix} 4 & 3 \\ 2 & 5 \end{pmatrix} \\ &= 4 \times 5 - 3 \times 2 \\ &= 20 - 6 = 14 > 0 \end{aligned}$$



$$p_1 \times p_2 = \begin{cases} > 0, \overline{p_0 p_2} - \text{против часовой стрелки от } \overline{p_0 p_1} \\ < 0, \overline{p_0 p_2} - \text{по часовой стрелке от } \overline{p_0 p_1} \end{cases}$$

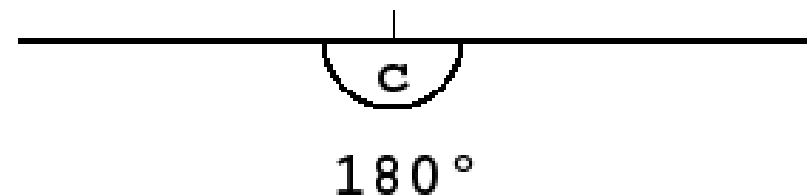
$$\begin{array}{c} p_1 \times p_2 = 0 \\ \overleftarrow{\hspace{1cm}} \quad \overrightarrow{\hspace{1cm}} \\ p_1 \qquad \text{коллинеарны} \qquad p_2 \end{array}$$

$$\begin{array}{c} p_1 \overrightarrow{\hspace{1cm}} \\ p_2 \overrightarrow{\hspace{1cm}} \\ \text{коллинеарны} \end{array}$$

Сканирование по Грэему

GRAHAM-SCAN(Q)

- 1 Пусть p_0 — точка Q с минимальной координатой y , или крайняя слева из таких точек при наличии совпадений
 - 2 Пусть $\langle p_1, p_2, \dots, p_m \rangle$ — остальные точки Q , отсортированные в порядке возрастания полярного угла, измеряемого против часовой стрелки относительно p_0 (если полярные углы нескольких точек совпадают, то из множества удаляются все эти точки, кроме одной, самой дальней от точки p_0)
 - 3 **if** $m < 2$
 - 4 **return** “выпуклая оболочка пуста”
 - 5 **else** пусть S — пустой стек
 - 6 PUSH(p_0, S)
 - 7 PUSH(p_1, S)
 - 8 PUSH(p_2, S)
 - 9 **for** $i = 3$ **to** m
 - 10 **while** угол, образованный точками NEXT-TO-TOP(S),
TOP(S) и p_i **не образует поворот влево**
 - 11 POP(S)
 - 12 PUSH(p_i, S)
 - 13 **return** S
- исключаем развернутые углы,
мы всегда должны поворачивать



Сканирование по Грэему

Сложность алгоритма

GRAHAM-SCAN(Q)

```
1 Пусть  $p_0$  — точка  $Q$  с минимальной координатой  $y$ , или
   крайняя слева из таких точек при наличии совпадений
2 Пусть  $\langle p_1, p_2, \dots, p_m \rangle$  — остальные точки  $Q$ , отсортированные
   в порядке возрастания полярного угла, измеряемого против
   часовой стрелки относительно  $p_0$  (если полярные углы
   нескольких точек совпадают, то из множества удаляются все
   эти точки, кроме одной, самой дальней от точки  $p_0$ )
3 if  $m < 2$ 
4   return “выпуклая оболочка пуста”
5 else пусть  $S$  — пустой стек
6   PUSH( $p_0, S$ )
7   PUSH( $p_1, S$ )
8   PUSH( $p_2, S$ )
9   for  $i = 3$  to  $m$ 
10     while угол, образованный точками NEXT-TO-TOP( $S$ ),
        TOP( $S$ ) и  $p_i$  не образует поворот влево
11       POP( $S$ )
12     PUSH( $p_i, S$ )
13 return  $S$ 
```

$O(n)$

$O(n \log n)$

$|Q| = n$ — число
входных точек

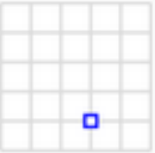
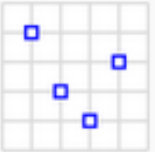

$O(n)$

Общая сложность

$O(n \log n)$

Point, MultiPoint, Polygon

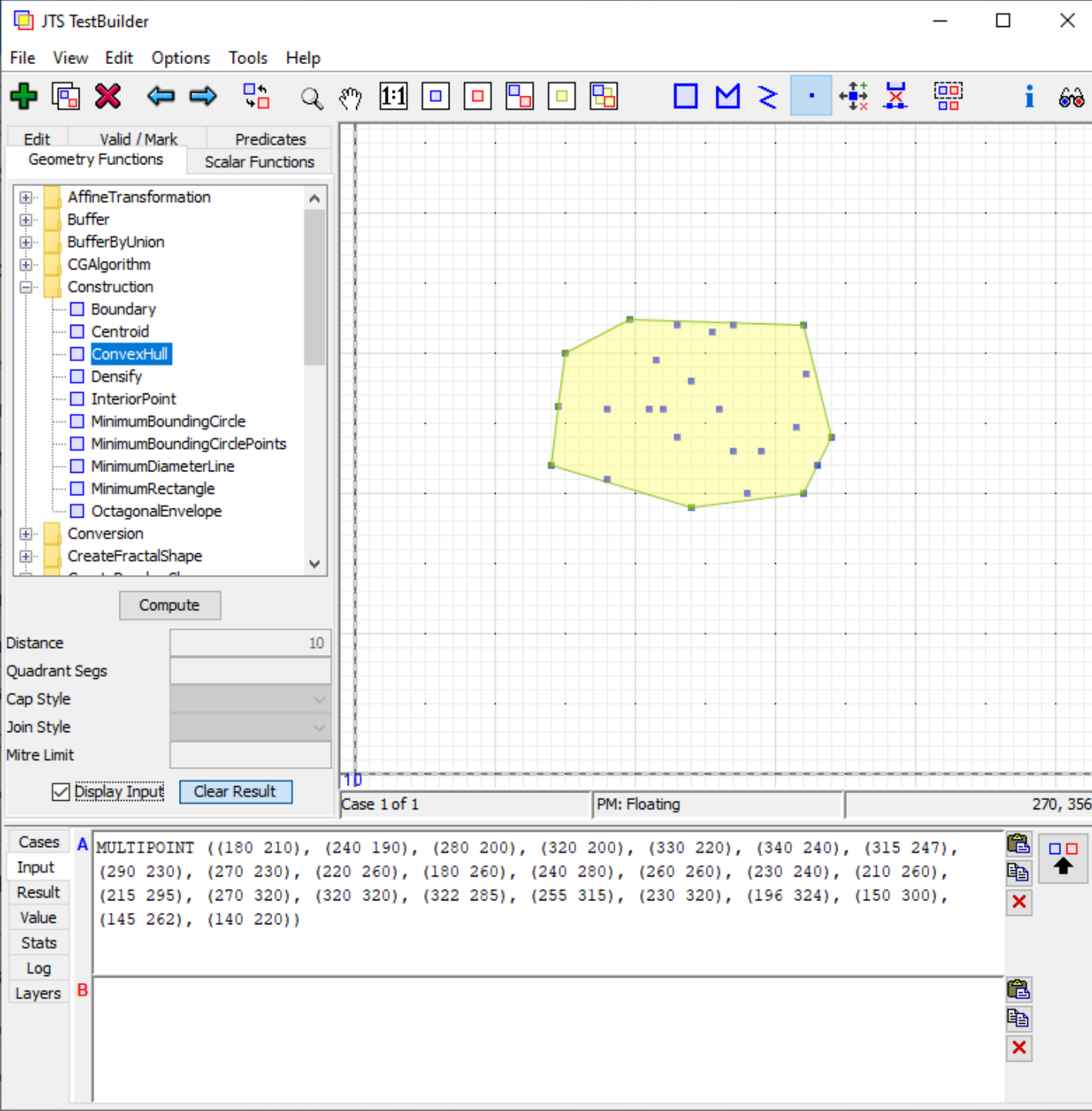
Geometry primitives (2D)

Examples	
	<code>POINT (30 10)</code>
	<code>MULTIPOINT ((10 40), (40 30), (20 20), (30 10))</code> <code>MULTIPOINT (10 40, 40 30, 20 20, 30 10)</code>
	<code>POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))</code>

WKT: Well-Known Text

- **Читаемый человеком текст. Используется повсеместно для представления геометрий в базах данных и других системах**
- vector (geometry) objects –
ISO 19107 Geometry instances, transformations between CRS, GCS/CRS.

https://en.wikipedia.org/wiki/Well-known_text



Выпуклая оболочка: графический интерфейс

JTS Builder

Результат:

POLYGON ((240 190, 140 220, 145 262, 150 300,
196 324, 320 320, 340 240, 320 200, 240 190))

Примечание:

последовательность вершин в
полигоне – по часовой стрелке

<https://sourceforge.net/projects/jts-topo-suite/>

Очередь / Queue

Все операции за $O(1)$

FIFO = First Input,
First Output

- Ассоциация: обычная очередь
- Поступление элементов с хвоста
- Извлечение элементов с головы

1 Queue Overflow

1

направление движения элементов

2

Enqueue

Dequeue – классическое название

доб. + exception – Add

Peak – извлечь не удаляя

доб. без exception – Offer
(вернуть T/F)

Poll – извлечь удаляя

хвост (tail)

голова (head)

...

названия
зависят от ЯП

Примеры использования

- Процессы в ОС
- Передача данных по сети
- ...

3

isEmpty

4

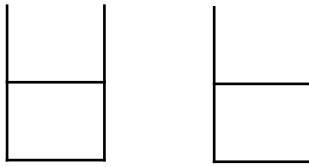
Size \mapsto 7

не изменяют состояние очереди

Типичная задача собеседования: очередь на двух стеках

На сообразительность

- **Задача:** реализовать очередь с помощью двух стеков
- **Требующиеся операции:** `isEmpty()`, `size()`, `Enqueue(x)`, `Dequeue(x)`
- **Подсказка:** достаточно, если `Dequeue(x)` будет работать за $O(n)$

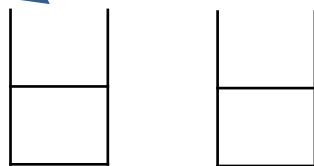
1 **isEmpty**  Оба стека пустые

A **B**

2 **size** `A.size() + B.size()`

3 **Enqueue(x)**

Push x

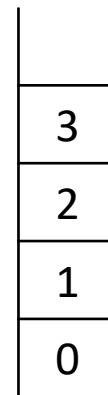
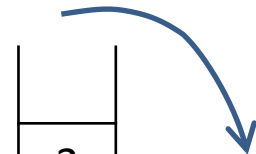


A

B

4 **Dequeue**

Если **B** пуст
перенести все элементы
из **A** в **B**
`return B.pop()`



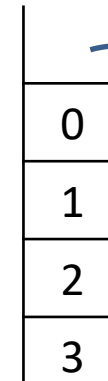
A



B



A



B

Pop \mapsto 0

очередь

	3	2	1	0
--	---	---	---	---



NATIONAL RESEARCH
UNIVERSITY

Благодарю за внимание!

Рамон Антонио Родригес Залепинос
arodriges@hse.ru