ОНЛАЙН-ОБРАЗОВАНИЕ

OTUS

Базовые структуры данных

Евгений Волосатов

Содержание

- Массив
- Динамические массив
- Список
- Стек
- Очередь
- Очередь с приоритетами

Ответить на вопросы

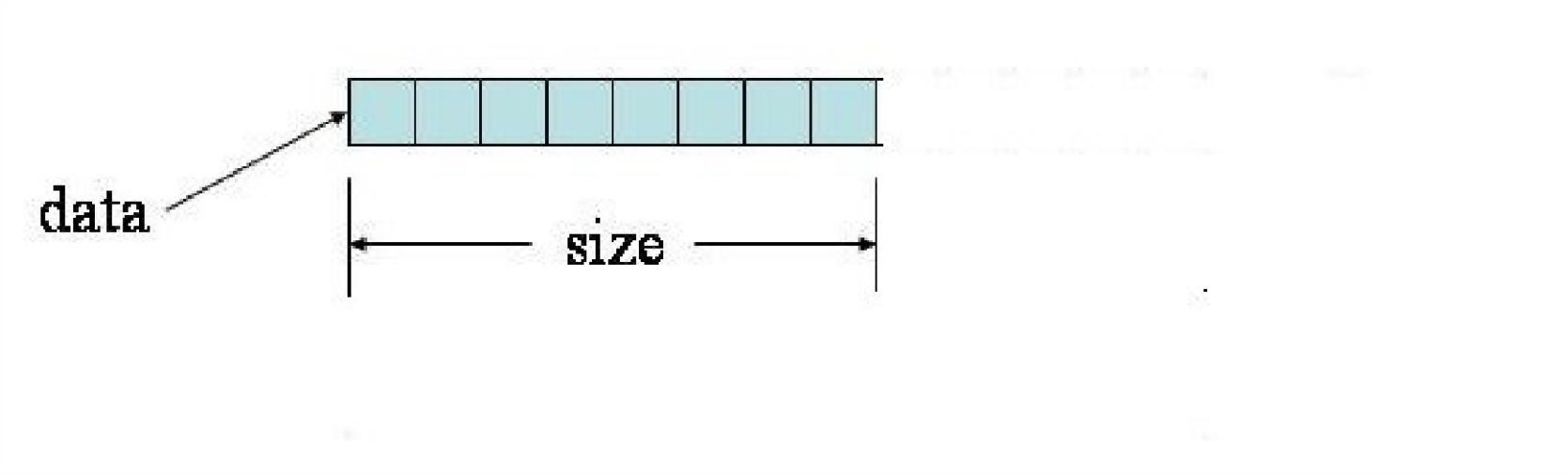
OTUS

- Какие структуры данных как устроены
- В каком случае какую структуру применять

- Что такое массив?
- Какие бывают массивы?

- Фиксированный массив
- Динамический массив
- Разряженный массив
- Ассоциативный массив
- Параллельные массивы

- Элементы одного типа
- Доступ по индексу



Описание в языках

Java/C#

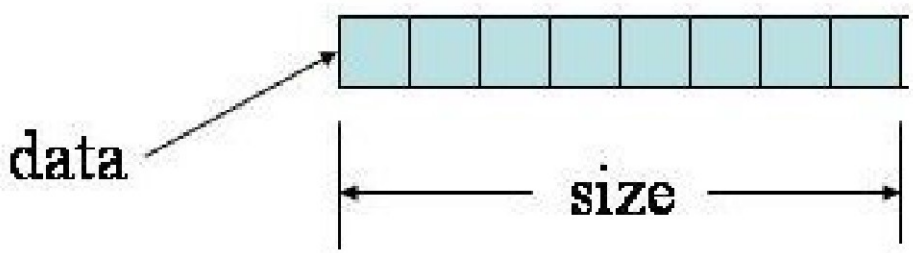
```
int[] a = new int[10];
```

C++

```
int a[10];
int* a = new int[10];
```

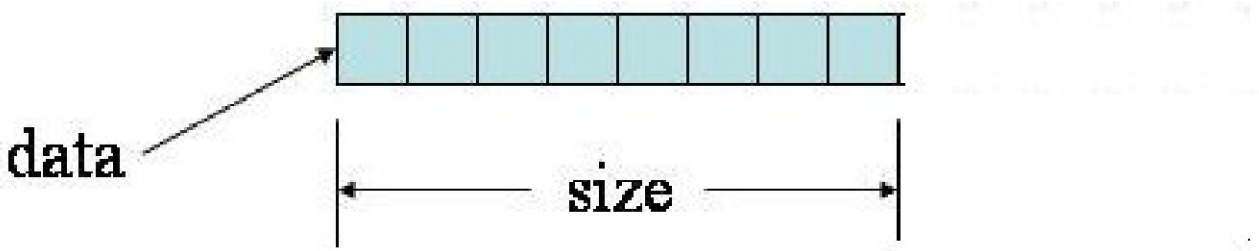
Динамические массивы

- Элементы одного типа
- Доступ по индексу
- **Растет по мере необходимости**



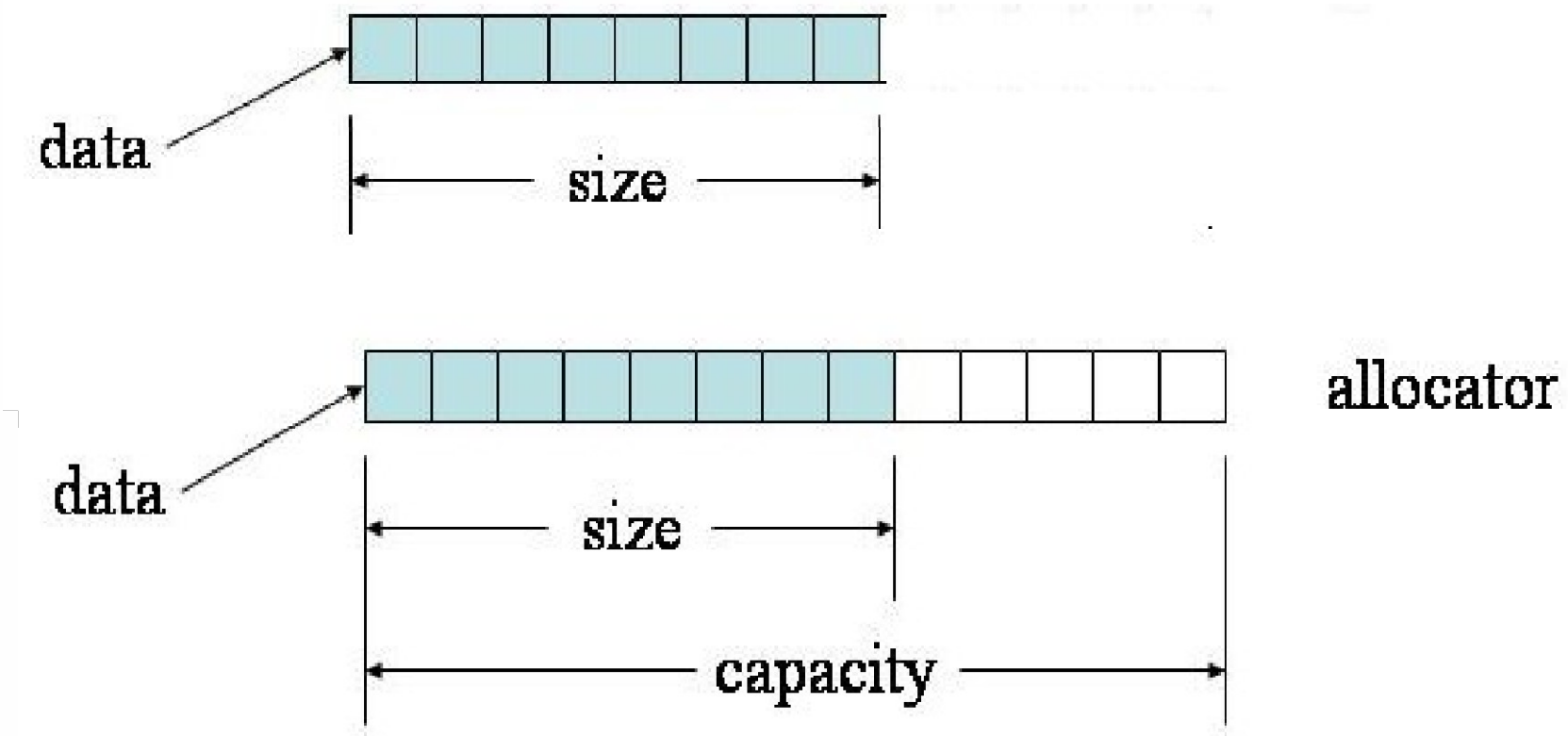
Динамические массивы

- в материалах к занятию класс DArray
- запустить, посмотреть код
- в чем неэффективность реализации?
- как улучшить?

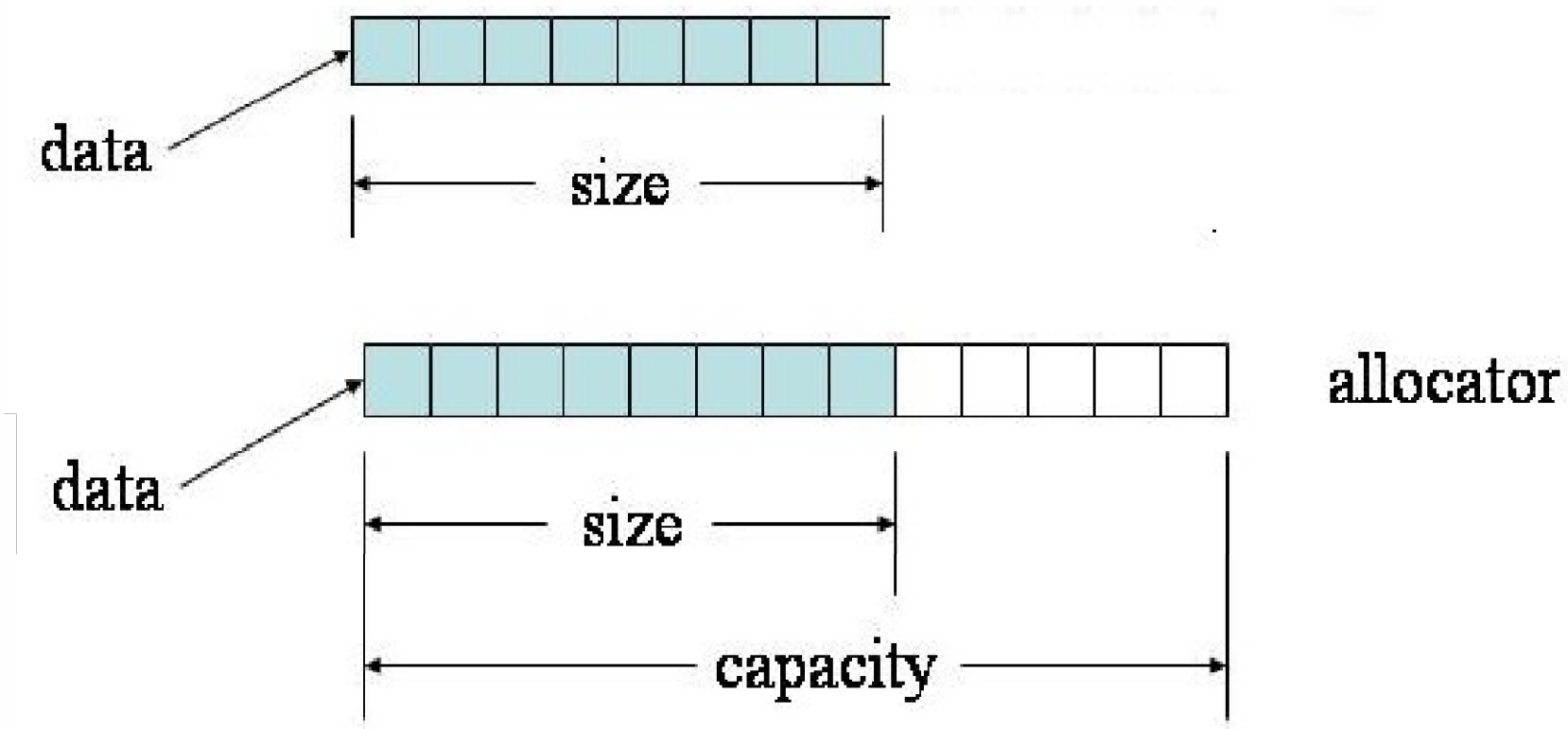


Динамические массивы

- Элементы одного типа
- Доступ по индексу
- Растет по мере необходимости
- **Растет блоками по N элементов**

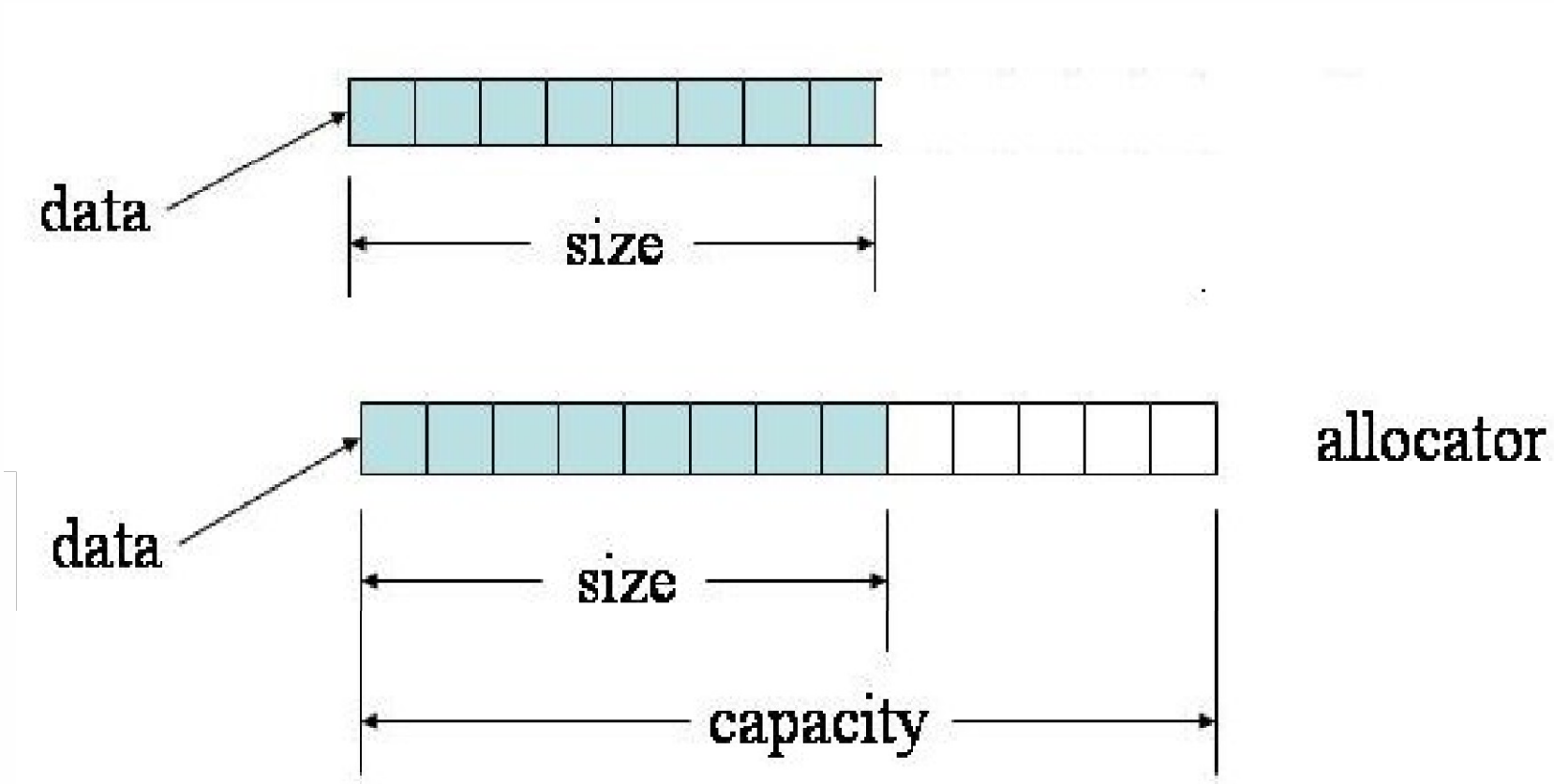


- Реализовать класс BArray
- Сравнить производительность с DArray
- Реализовать remove



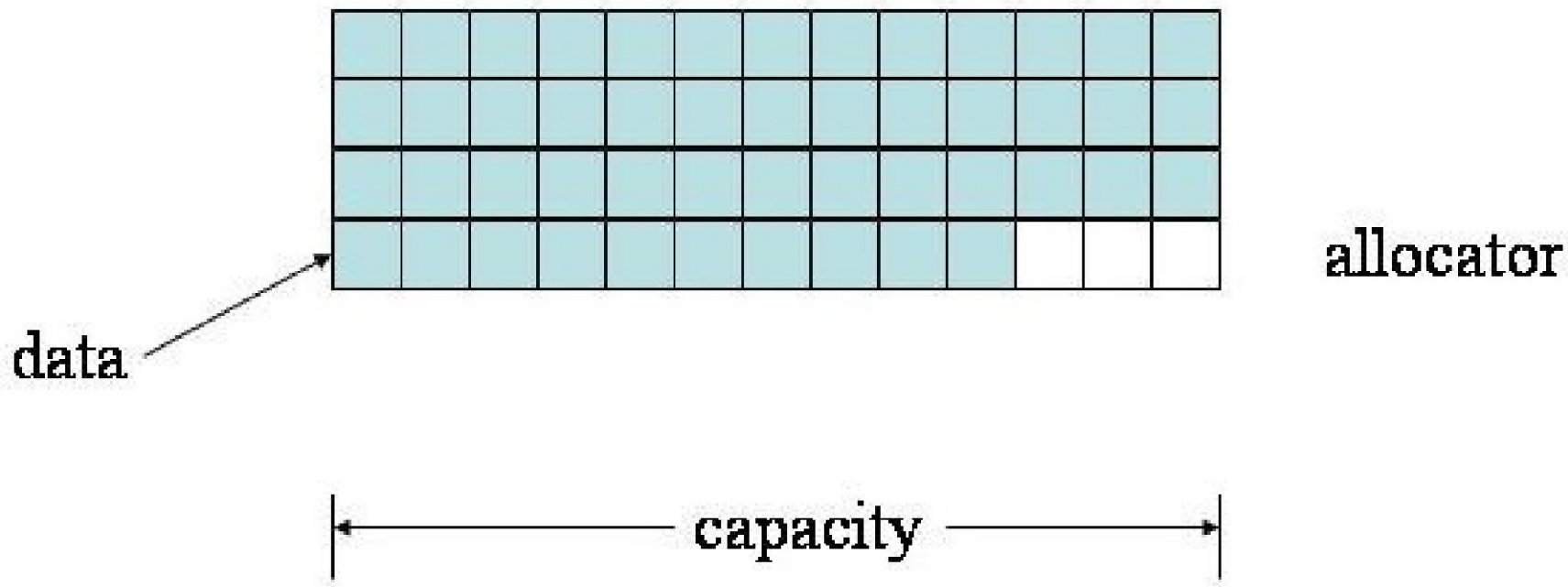
Динамические массивы

- В чем неэффективность?
- Как можно улучшить?



Динамические массивы

- Элементы одного типа
- Доступ по индексу
- Растет по мере необходимости
- Растет блоками по N элементов
- **При расширении копируется только часть данных**

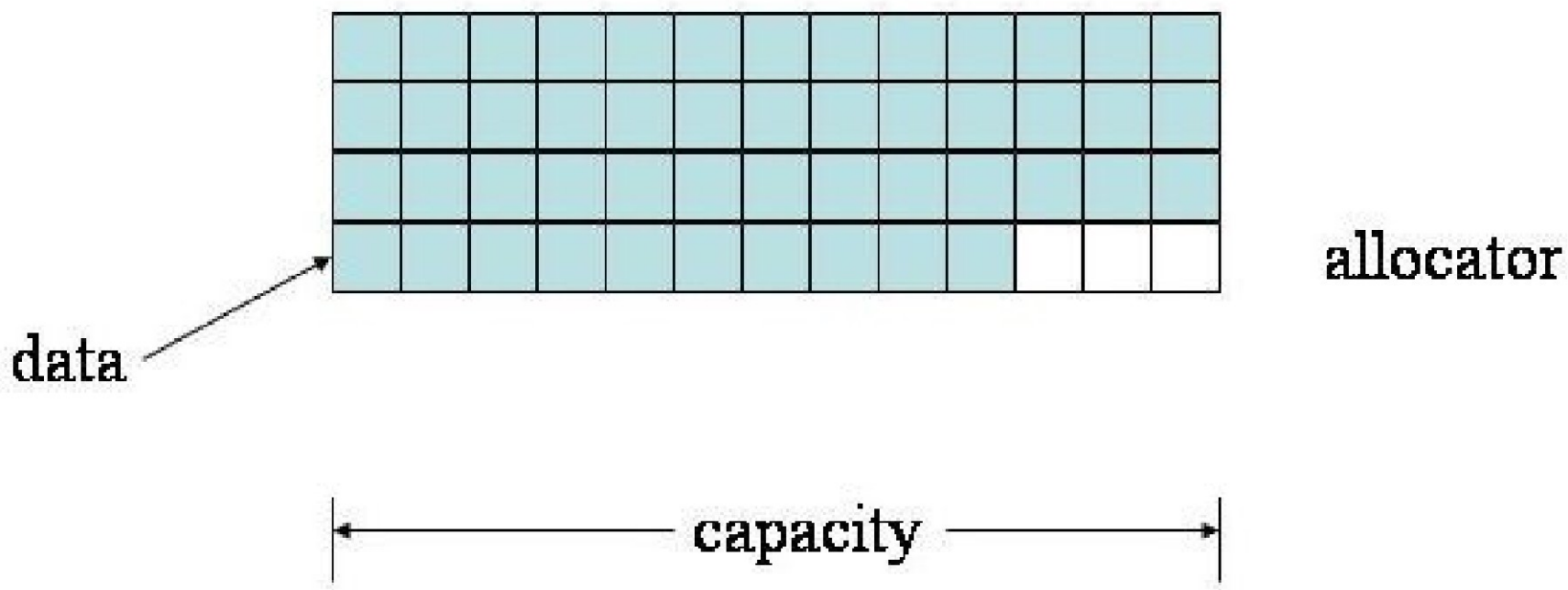


Массив массивов

```
public class IArray<T> {
                        BArray<BArray<T>> _arr;


            ...
}
```

Динамические массивы

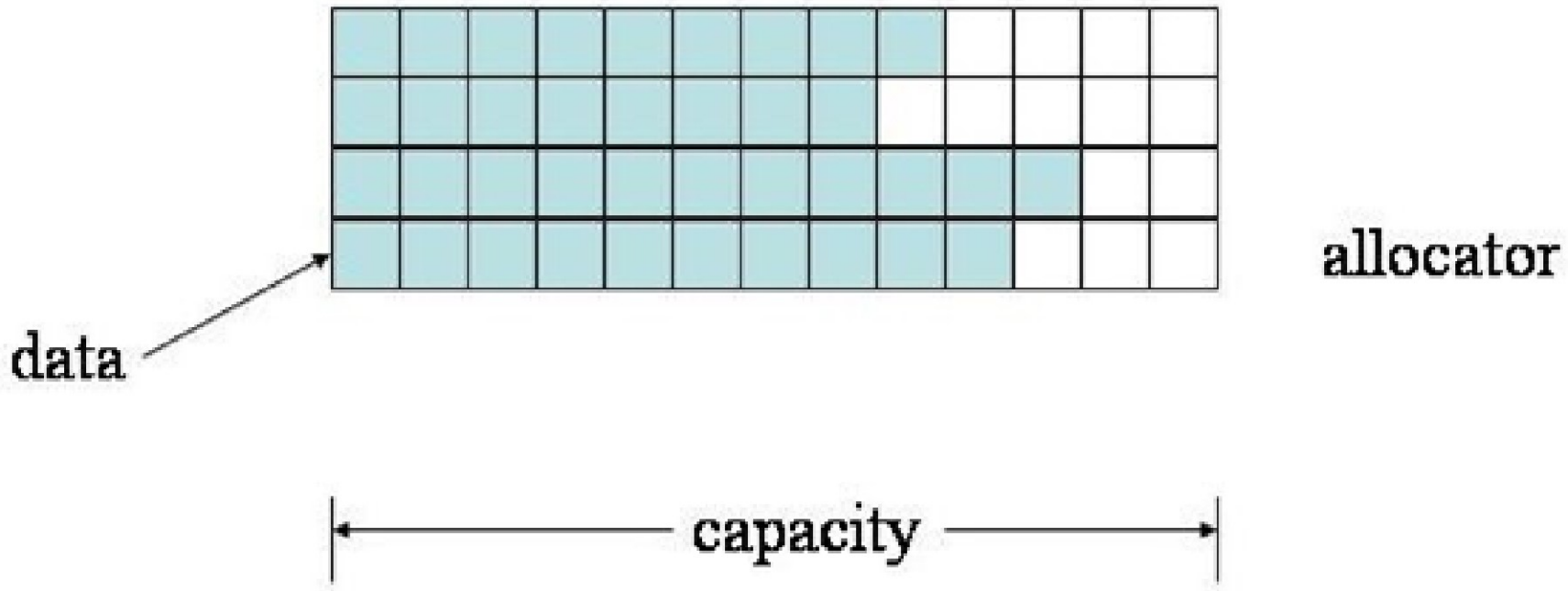
```
T at(int index) {
        int index1 = index / _delta;
        int index2 = index % _delta;
        return (T)_arr.get(index1).get(index2);
    }
```



Динамические массивы

OTUS

- Элементы одного типа
- Доступ по индексу
- Растет по мере необходимости
- Растет блоками по N элементов
- При расширении копируется только часть данных
- **Вставка в любое место массива одинаково эффективна**

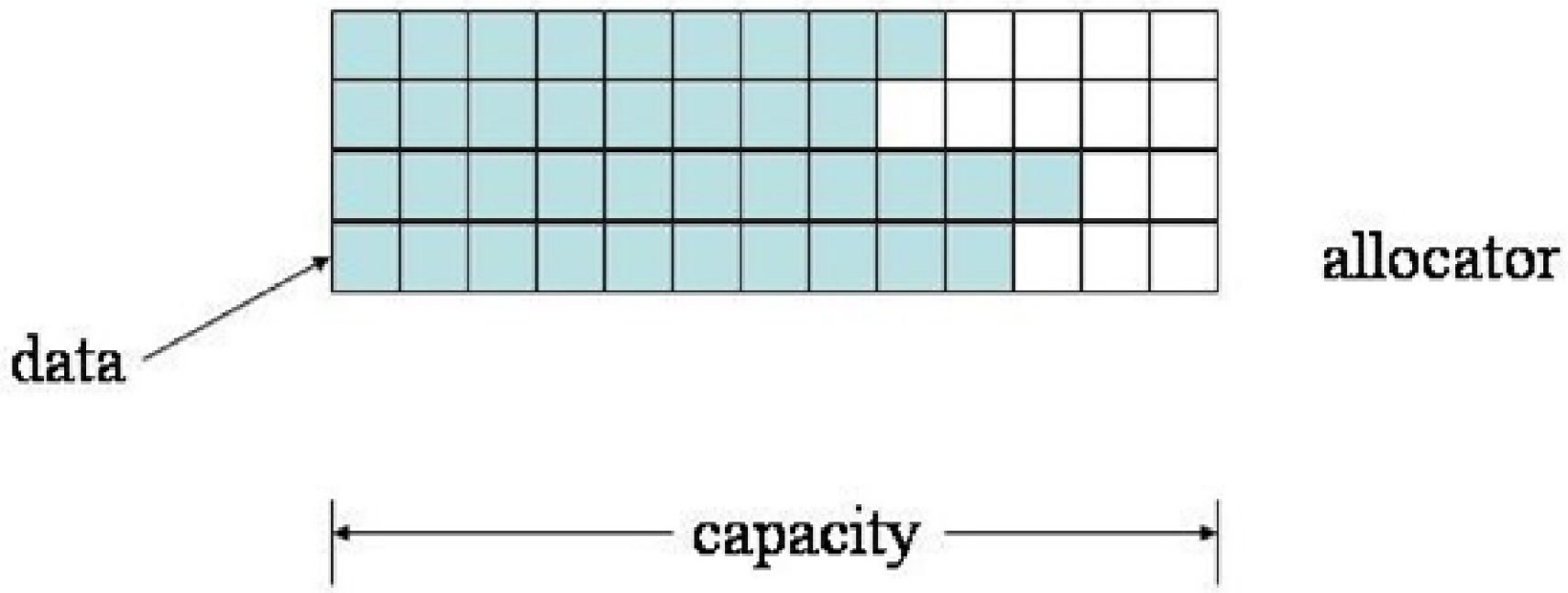


Динамический массив

- Структура данных
- item get()

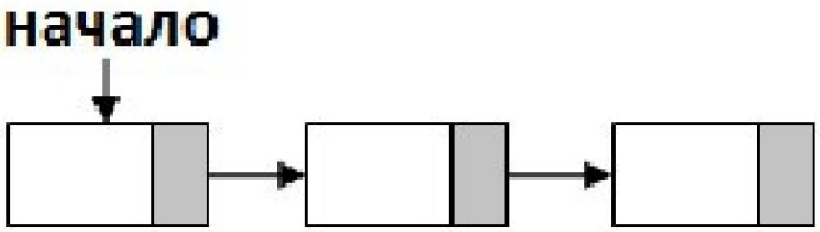
```
public class IArray<T> {
                        BArray<BArray<T>> _arr;


            ...
}
```

- Какая реализация эффективнее и почему?

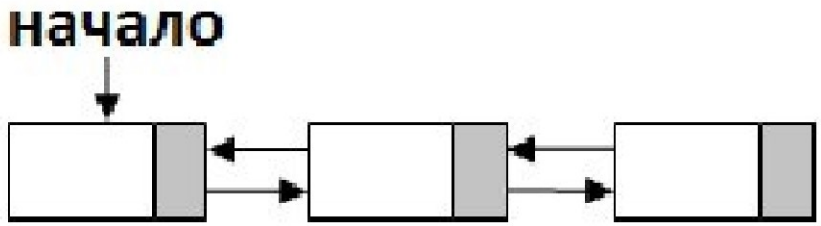


Список

- Элементы одного типа
- Последовательный доступ
- Растет по мере необходимости
- При расширении данные не копируются
- Вставка в любое место списка одинаково эффективна

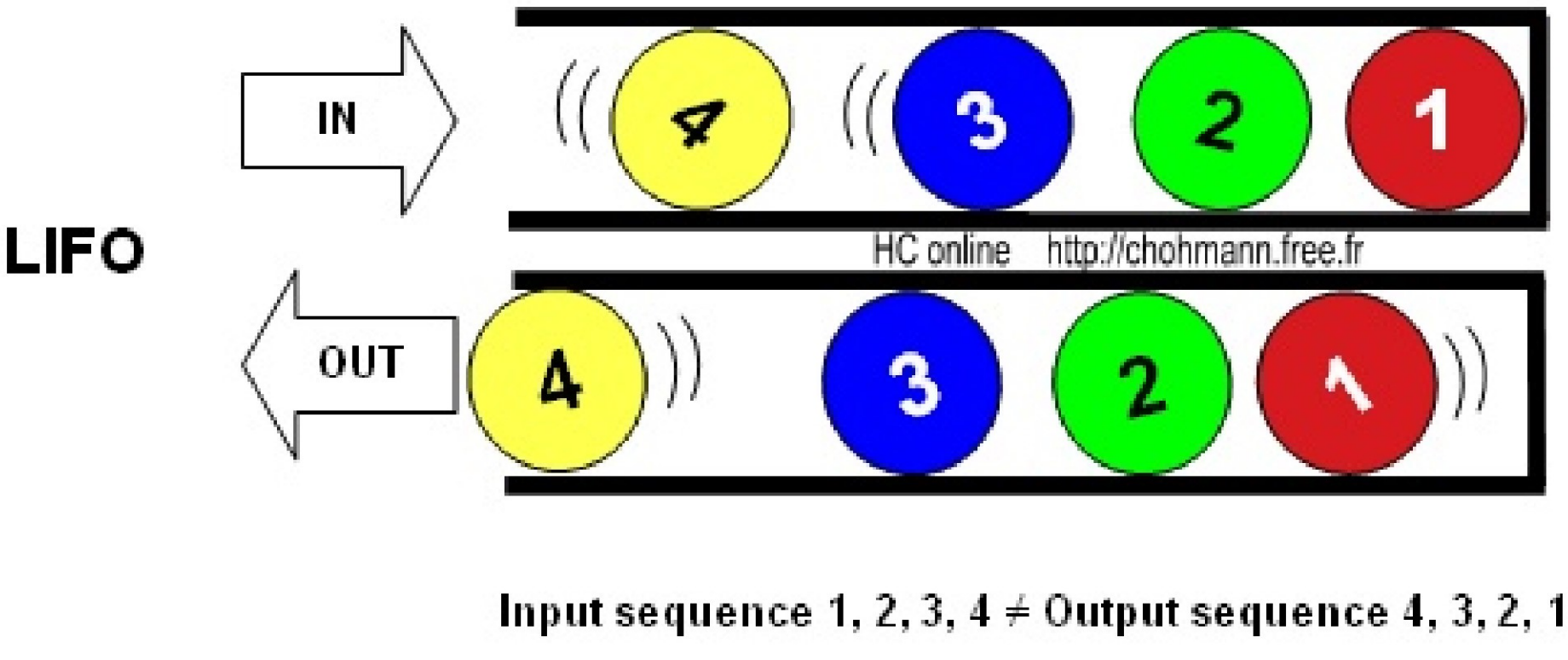


Список.

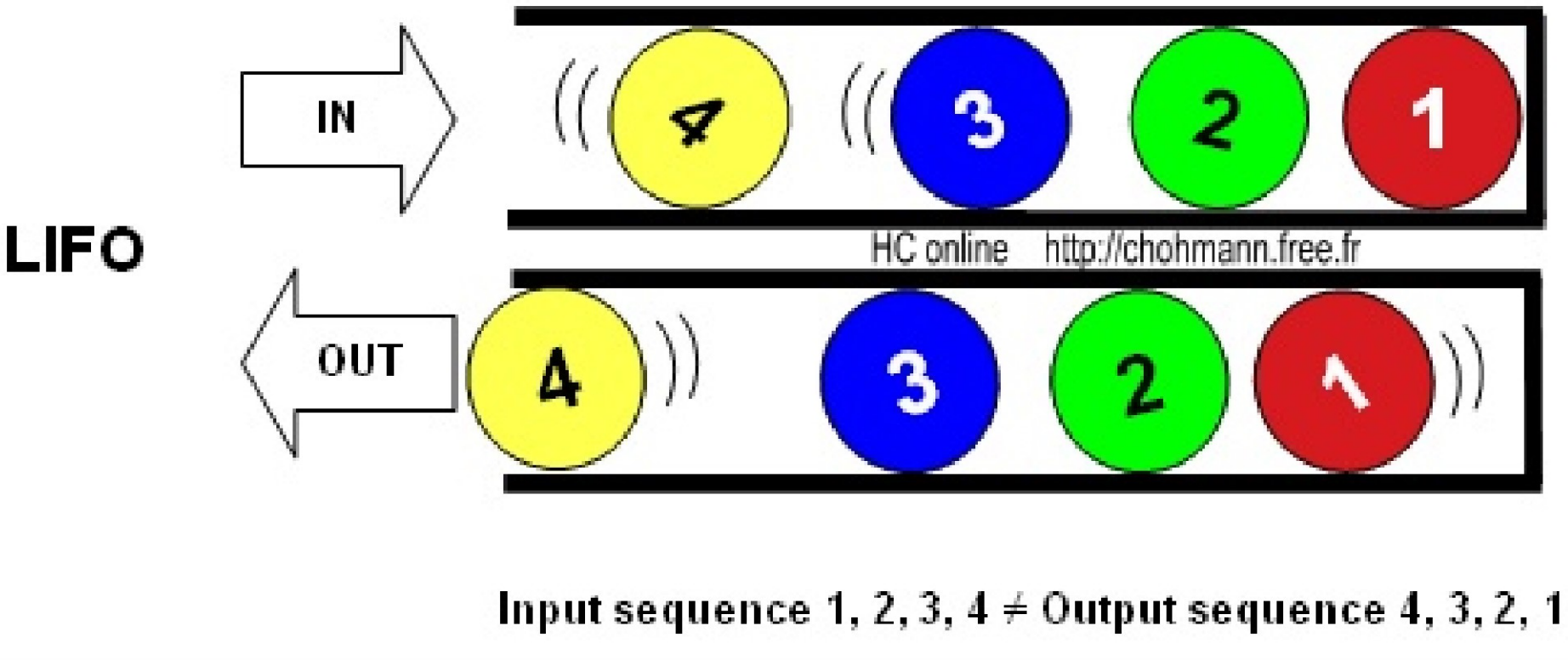


Двунаправленный список.

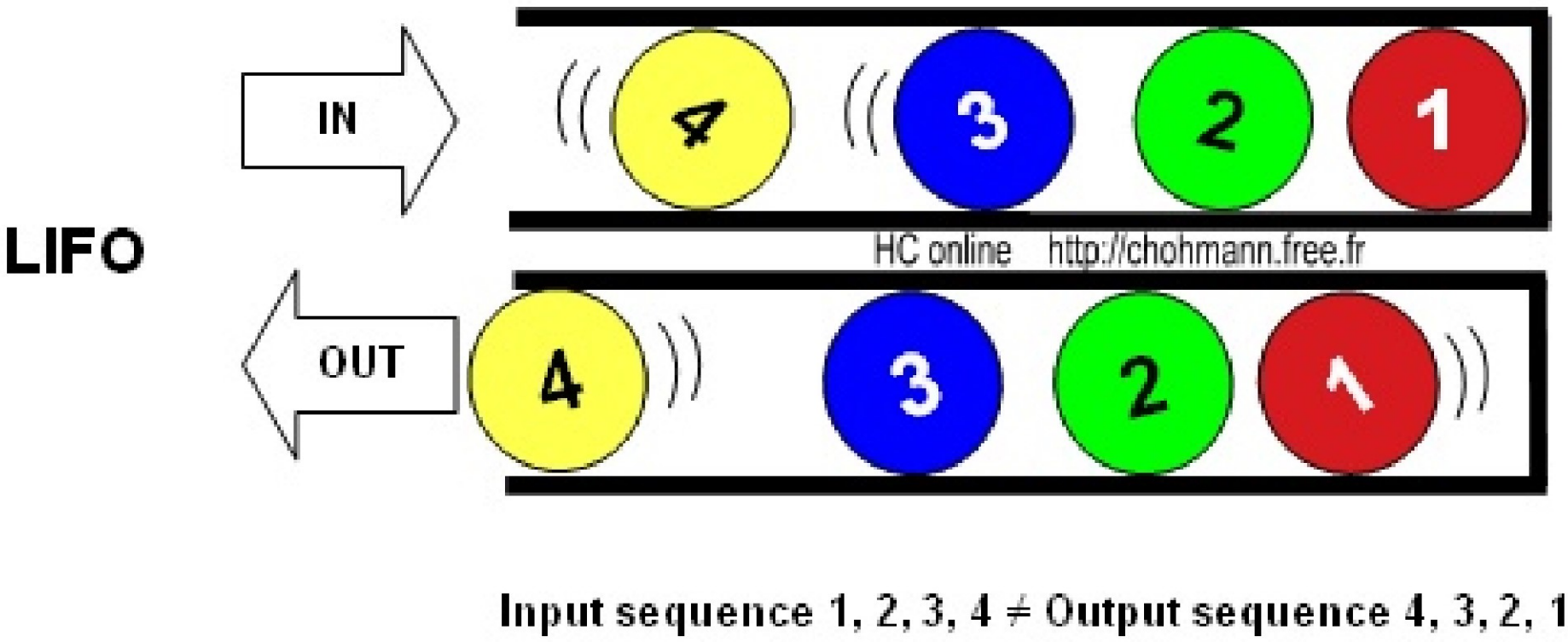
- Элементы одного типа
- Стратегия LIFO (Last In First Out)



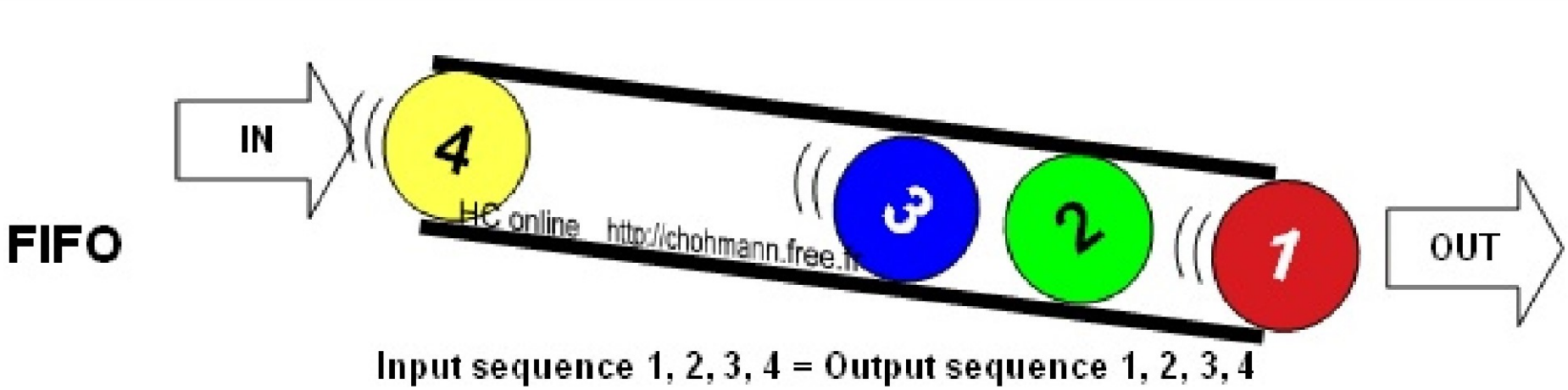
- На какой структуре данных эффективнее сделать стек?



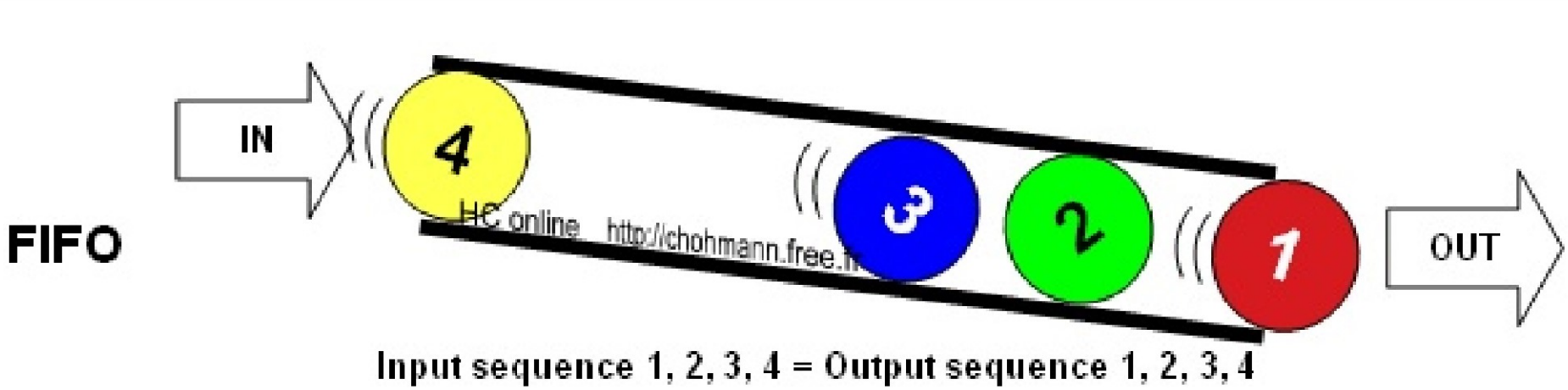
- push(T item)
- T pop()



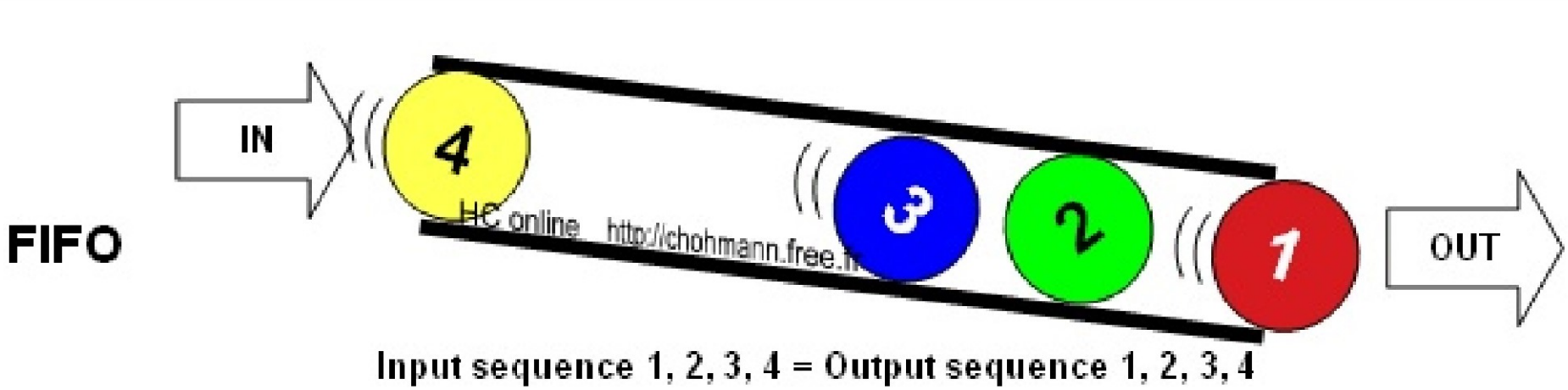
- Элементы одного типа
- Стратегия FIFO (First In First Out)



- На какой структуре данных эффективнее сделать очередь?

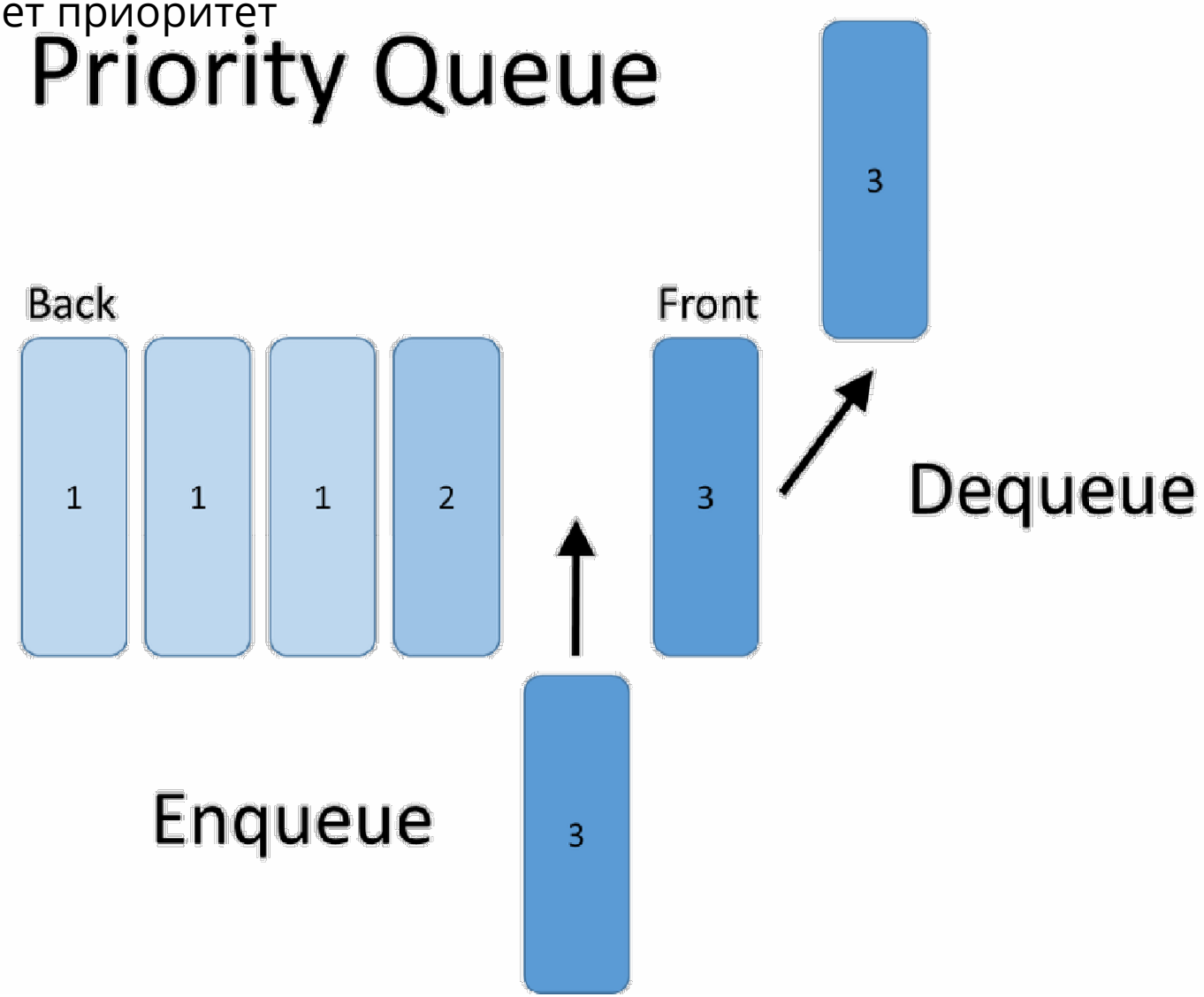


- enqueue(T item)
- T dequeue()



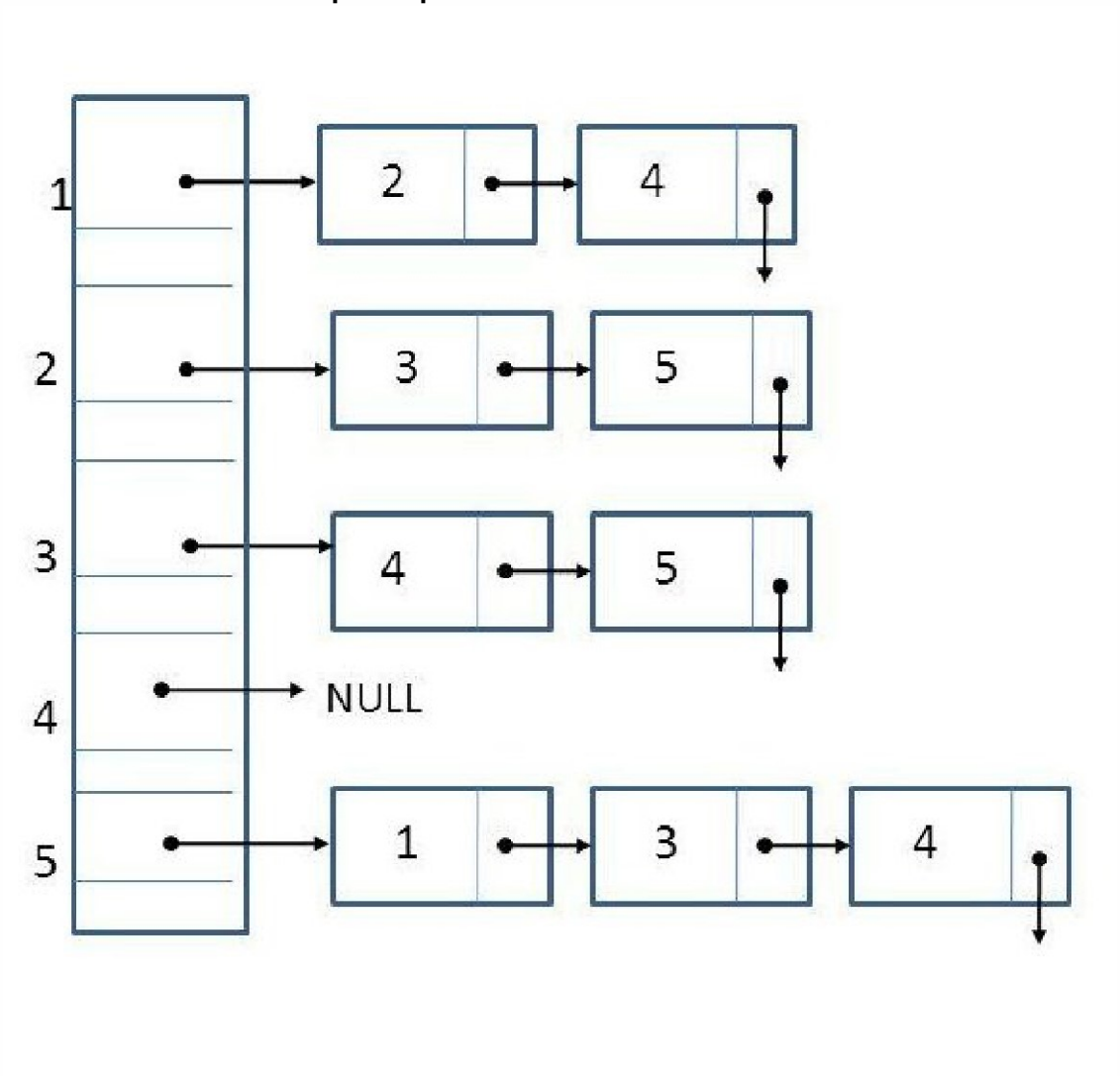
Очередь с приоритетами

- Элементы одного типа
- Элемент имеет приоритет



Очередь с приоритетами

- Элементы одного типа
- Элемент имеет приоритет



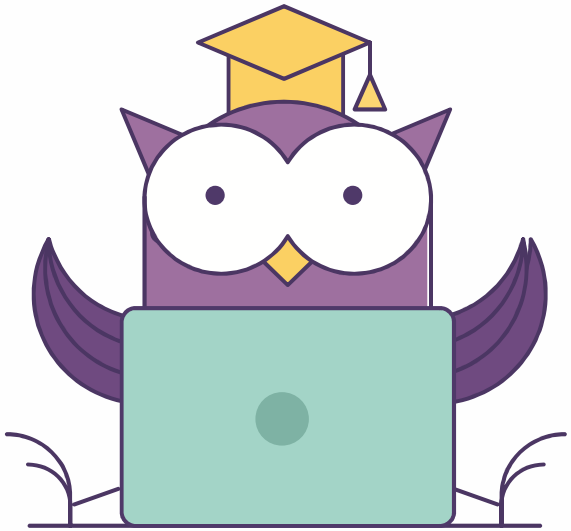
Список списков

```
public class PQueue<T> {
        SortedList<List<T>> _priority;

        ...
}
```

Что полезного я вынес из сегодняшнего занятия?

OTUS

- Массив
- Динамические массив
- Список
- Стек
- Очередь
- Очередь с приоритетами



Домашнее задание

OTUS

- IArray - массив с быстрой вставкой в любое место
- PQueue - очередь с приоритетами

Спасибо
за внимание!