Динамический массив

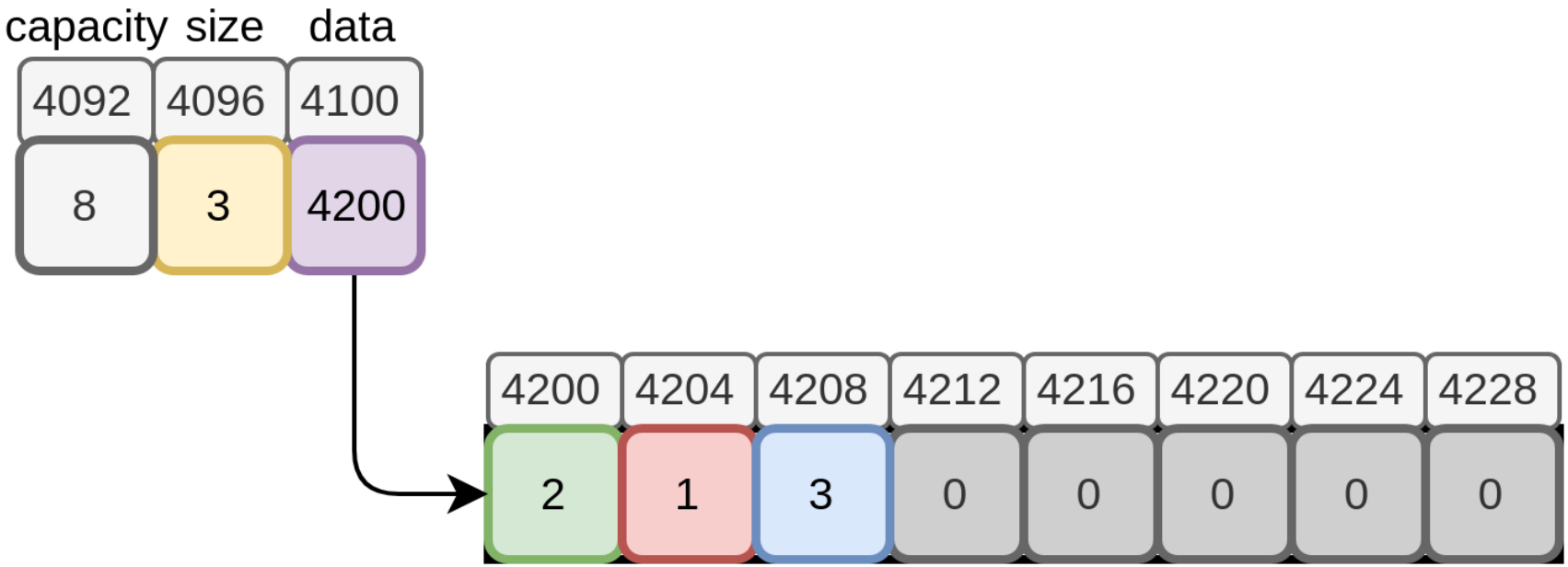
Гусев Илья, Булгаков Илья

Московский физико-технический институт

Москва, 2021

Содержание

Динамический массив




```
struct DynamicArray {
    unsigned int capacity;
    unsigned int size;
    int* data;
};
typedef struct DynamicArray DynamicArray;
```

Динамический массив

Наивное расширение

```
void Reserve(DynamicArray* array, unsigned int newCapacity) {
    assert(newCapacity >= array->size);
    int* newData = (int*)realloc(array->data,
        newCapacity * sizeof(int));
    assert(newData != NULL);
    array->data = newData;
    array->capacity = newCapacity;
}
void NaiveExtend(DynamicArray* array) {
    if (array->size < array->capacity) {
        return;
    }
    assert(array->size == array->capacity);
    Reserve(array, array->capacity + 1);
}
```

- realloc: перемещает кусок размера *capacity*
- Для выделения места под N элементов:
  $1 + 2 + 3 + ... + N = \frac{N \cdot (N+1)}{2} = O(N^2)$

Динамический массив

Классическое расширение

```
void Extend(DynamicArray* array) {
    if (array->size < array->capacity) {
        return;
    }
    assert(array->size == array->capacity);
    Reserve(array, array->capacity * 2);
}
```

Для выделения места под N элементов: $2 + 4 + 8 + ... + N \approx 2 \cdot N = O(N)$

Динамический массив

Наивная обрезка

```
void NaiveShrink(DynamicArray* array) {
    if (array->size > array->capacity / 2) {
        return;
    }
    Reserve(array, array->capacity / 2);
}
```

Что может пойти не так?

Динамический массив

Наивная обрезка

```
void NaiveShrink(DynamicArray* array) {
    if (array->size > array->capacity / 2) {
        return;
    }
    Reserve(array, array->capacity / 2);
}
```

push-pop-push-pop-push...

$O(n) + O(n) + O(n) + O(n) + \ldots$

Динамический массив

Классическая обрезка

```
void Shrink(DynamicArray* array) {
    if (array->size > array->capacity / 4) {
        return;
    }
    Reserve(array, array->capacity / 2);
}
```

Динамический массив

Реализация стека

```
void PushBack(DynamicArray* array, int element) {
    array->data[array->size] = element;
    array->size += 1;
    Extend(array);
}

int PopBack(DynamicArray* array) {
    int element = array->data[array->size - 1];
    array->data[array->size - 1] = 0;
    array->size -= 1;
    Shrink(array);
    return element;
}

bool IsEmpty(DynamicArray* array) {
    return (array->size == 0);
}
```

Динамический массив

Сложность

Интерфейс стека:

- PushBack: амортизированная $O(1)$
- PopBack: амортизированная $O(1)$
- IsEmpty: $O(1)$

Другие операции:

- PushFront: амортизированная $O(1)$ через циклический буфер
- PopFront: амортизированная $O(1)$ через циклический буфер
- GetByIndex: $O(1)$
- Find: $O(n)$; $O(log(n))$ через поддержку отсортированности
- ExtractMax: $O(n)$; $O(1)$ через поддержку отсортированности

Динамический массив vs односвязный список

Плюсы и минусы для реализации стека

Односвязный список:

- Константное время всех операций
- Разбросанные куски памяти с гарантированными затратами на указатели

Динамический массив:

- Константное *амортизированное* время всех операций
- Сплошной кусок памяти: кешам процессора приятно, нет дополнительных затрат на указатели

Стандартная библиотека

vector

std::vector

- push_back
- pop_back
- empty
- size
- assign
- reserve

Полезные ссылки I

Т.Кормен, Ч.Лейзерсон, Р.Ривест, К.Штайн - Алгоритмы. Построение и анализ. Глава 6
https://bit.ly/2wFzphU

Lecture Slides for Algorithm Design
https://algs4.cs.princeton.edu/lectures/