Структури от данни и програмиране

Лекция 3

Опашка

Опашки в реалния свят

начало

край

изтриване на елементи

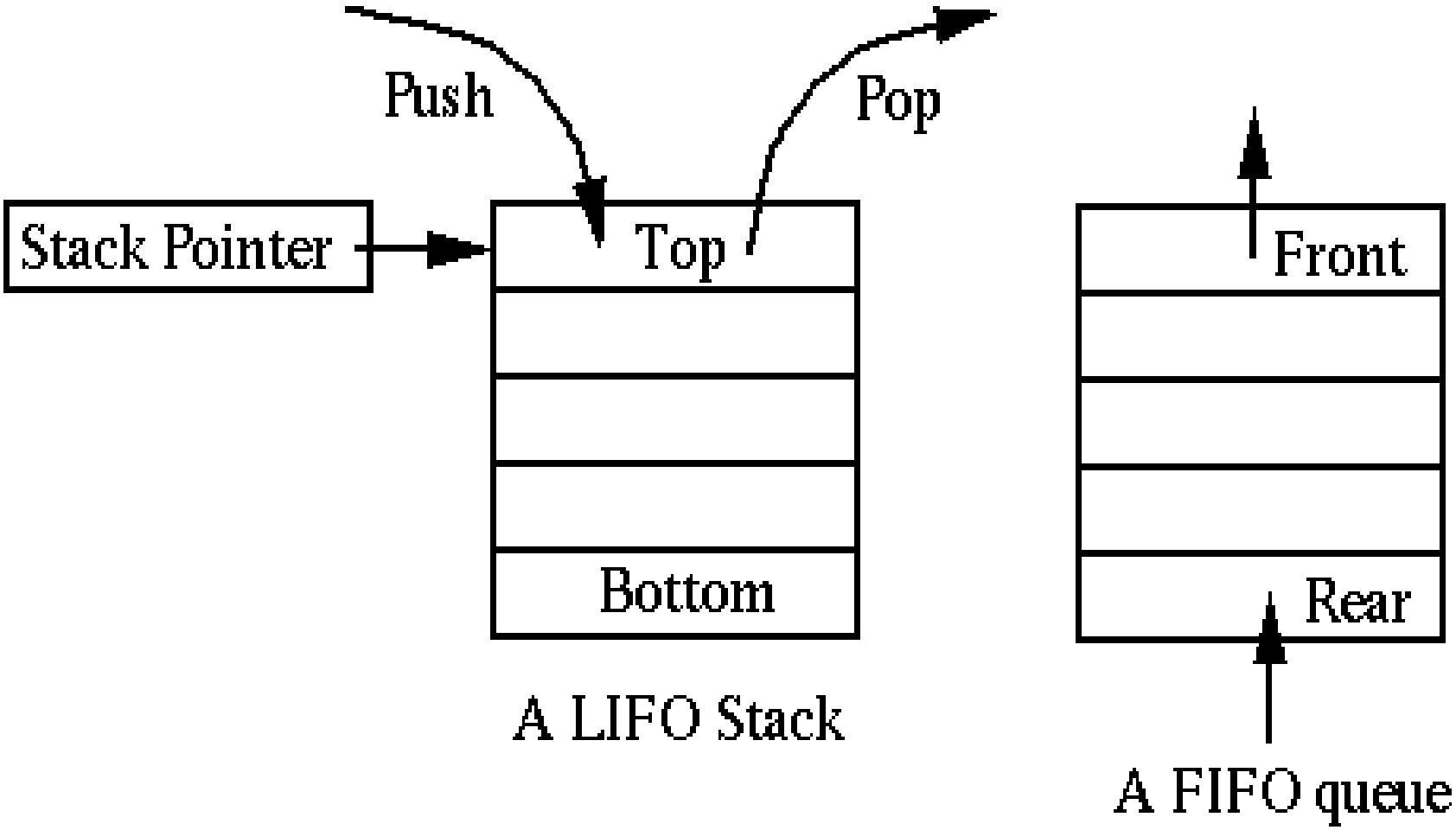
добавяне на елементи

първият влязъл ще излезе първи

Логическо описание

- Хомогенна линейна структура
- Пръв влязъл, пръв излязъл (First In, First Out - FIFO)
- Операции – подобни на операциите върху стек
  - създаване на празна опашка (create)
  - проверка за празнота (empty)
  - включване на елемент (push)
  - изключване на елемент (pop)
  - достъп до главата на опашката (head)

Сравнение със стек

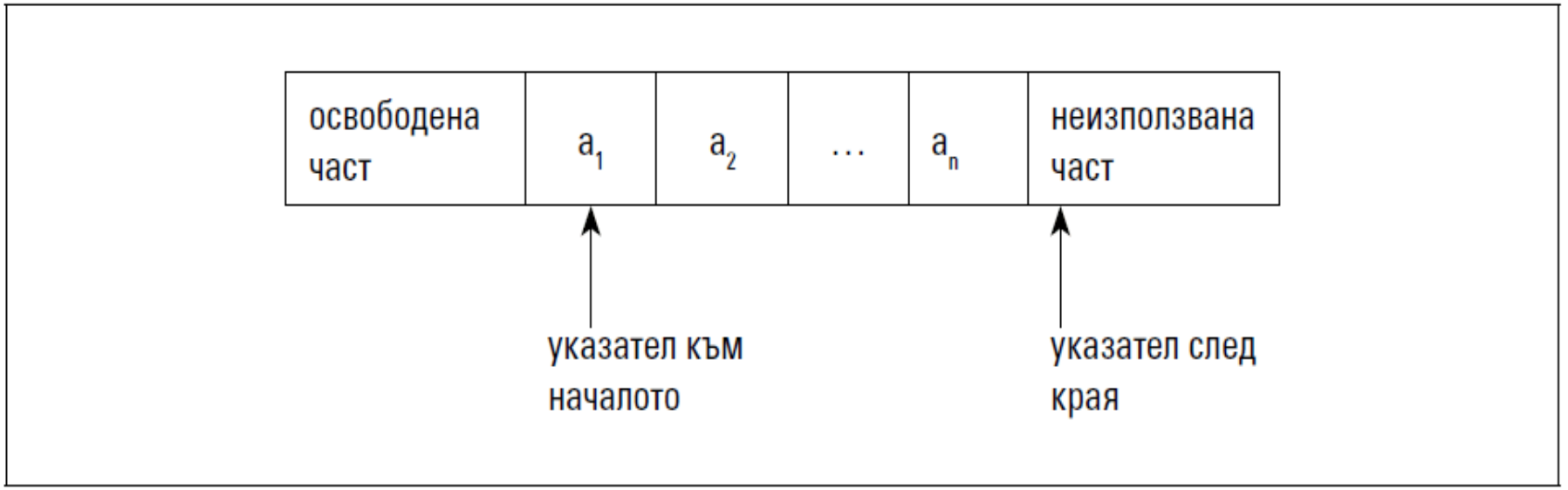


Основни начини за представяне на опашка

- Последователно представяне – с масив
- Свързано представяне

Последователно представяне

- Използва се масив

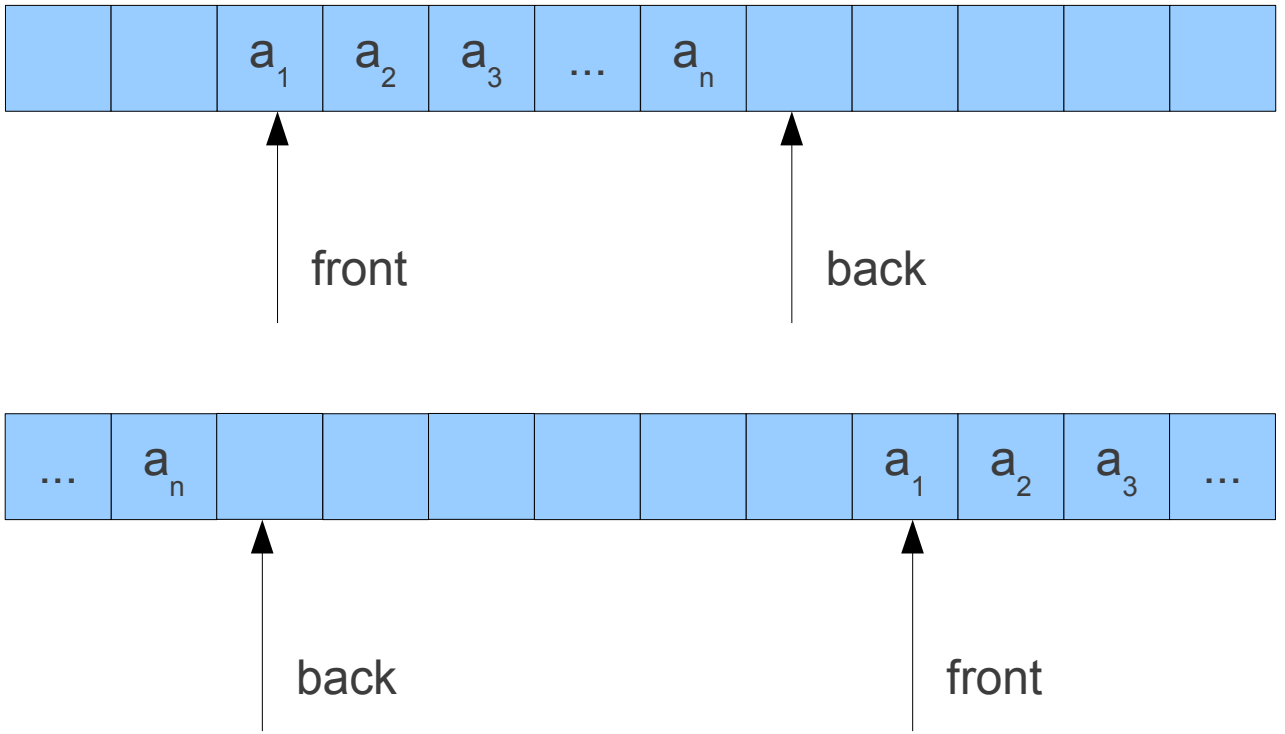


Фигура 2.4. Последователно представяне на опашка

- Проблем: след всяко добавяне/изтриване индексите растат, т.е. опашката се “движи” надясно и паметта бързо ще свърши, дори и да има свободно пространство

Последователно представяне (2)

- Решение: използване на “цикличен” масив



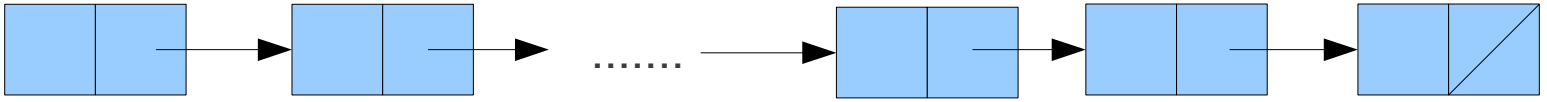
- Проблем: как ще разграничим празна от пълна опашка (front == back)?

Последователно представяне (3)

- Проблем: как ще разграничим празна от пълна опашка?
- Съществуват различни решения:
  - Оставяне на незапълнена позиция
  - Поле, указващо броя използвани елементи
  - Булев флаг, разграничаващ двата случая
  - Използване на абсолютни индекси
  - Поле, указващо коя е била последната операция
  - Други
- Всеки от подходите има различни предимства и недостатъци

Свързано представяне

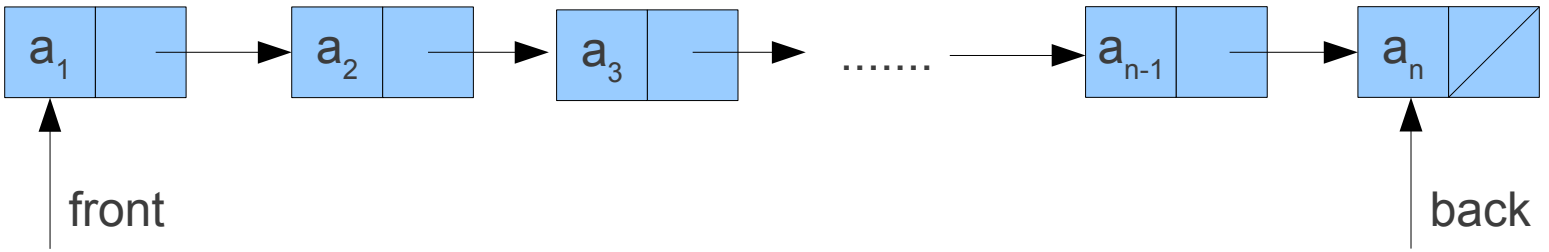
- Всеки елемент (без последният) реферира следващия



- В кой край е по-удобно да се извършва добавянето (съответно изтриването) на елементи?

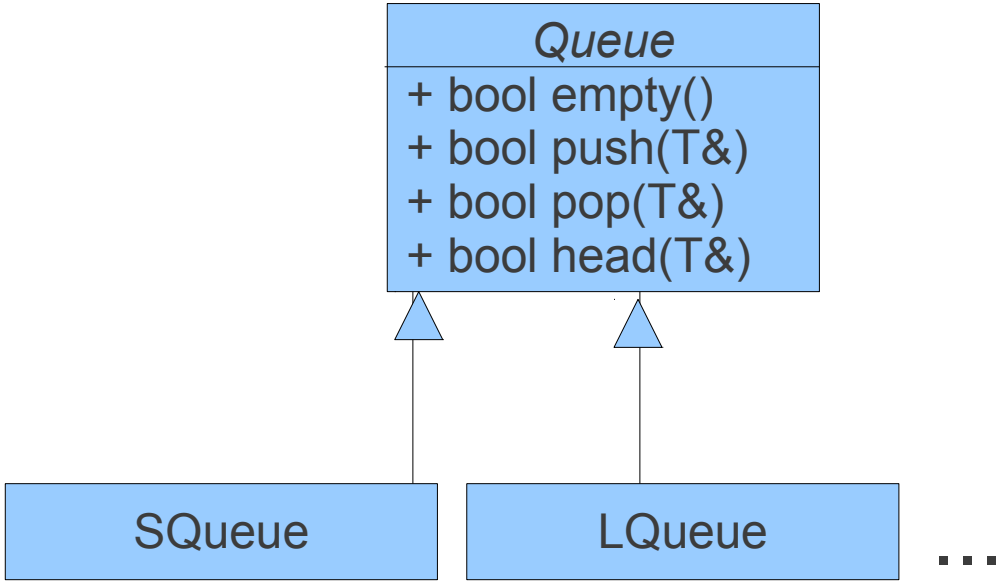
Свързано представяне (2)

- Ще извършваме добавяне в края и изтриване в началото



- Предимства и недостатъци на последователното и свързаното представяне – като при стек

Реализации
на различните представяния



Примерна реализация на базовия клас

- Базов абстрактен клас – дефинира интерфейса

```
template <typename T>
class Queue {
public:
    virtual bool empty() const = 0;
    virtual bool push(T const&) = 0;
    virtual bool pop(T&) = 0;
    virtual bool head(T&) const = 0;
};
```

Примерна реализация на свързаното представяне

- (`squeue.cpp`)

Примерна реализация на свързаното представяне

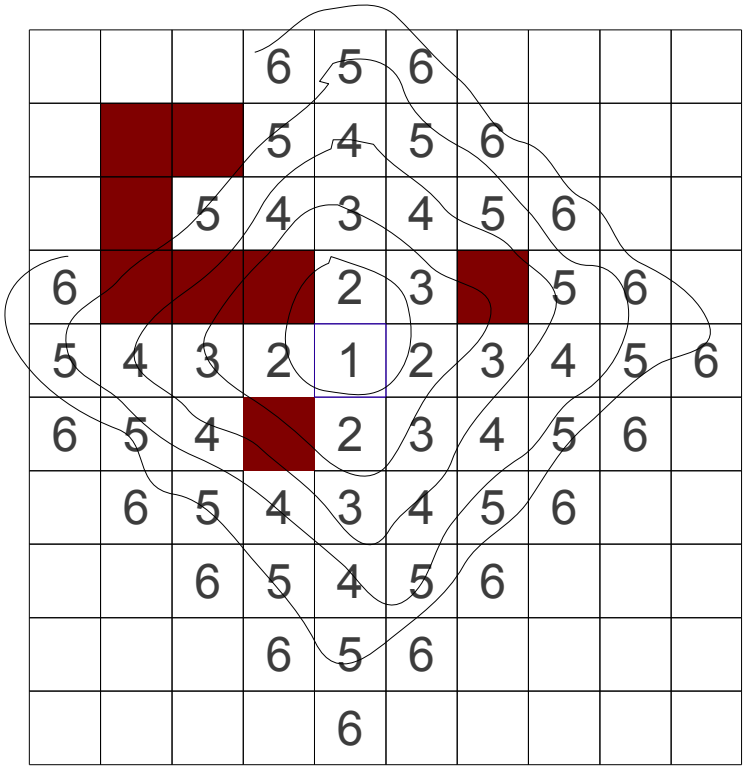
- Възел от свързаното представяне (като при стек):

```
template <typename U = int>
struct elem {
    U inf;
    elem<U>* link; // следващ елемент
};
```

- Останалата част – в `lqueue.cpp`

Приложения

- Метод на вълнàта



- Print spooling (**s**imultaneous **p**eripheral **o**perations **o**n-**l**ine)

Задачи

- Намиране на първите *n* числа, чиито прости делители са измежду 2, 3 и 5
- Извличане на минимален елемент на опашка
  - Ще използваме *сентинел*
  - За типа T трябва да е дефиниран оператор >
- Сортиране на опашка
  - Алгоритъм: пряка селекция
- Сливане на сортирани опашки

Допълнителен материал: подобни структури от данни

- Дек (double-ended queue, deque)
  - Позволява добавяне и изтриване на елементи и в двата края
  - Съществуват и ограничени варианти – напр. добавяне и в двата края, но изтриване само от единия
- Приоритетна опашка
  - Всеки елемент има приоритет (указва се при добавяне в опашката)
  - Първият извлечен елемент е този с най-висок приоритет
  - Въпрос от интервю за работа: опашката и стекът могат ли да се разгледат като частни случаи на приоритетната опашка? Ако да, как могат да се реализират?

Обобщение