Алгоритмы и Алгоритмические Языки

Семинар #13:

1. Ассоциативное хранилище данных.
2. Работа с бинарными файлами.

19.11.2025

Ассоциативное хранилище данных

Ассоциативное хранилище данных

```
❯ python3
>>> d = dict()
>>> d[12] = "aaa"
>>> d[155] = "bbb"
>>> d
{12: 'aaa', 155: 'bbb'}
```

```c
// Пара ключ-значение в базе данных.
typedef struct {
    // По ключу производится поиск.
    uint32_t key;
    // Значение - данные для хранения.
    char value[VALUE_SIZE];
} Entry_t;
```

Возможный API (программный интерфейс) хранилища:

```c
void db_alloc(struct Database* db);
void db_free(struct Database* db);
bool db_search(const struct Database* db, uint32_t key, char value[VALUE_SIZE], uint32_t* index);
bool db_insert(struct Database* db, uint32_t key, const char value[VALUE_SIZE]);
bool db_remove(struct Database* db, uint32_t key, char value[VALUE_SIZE]);
```

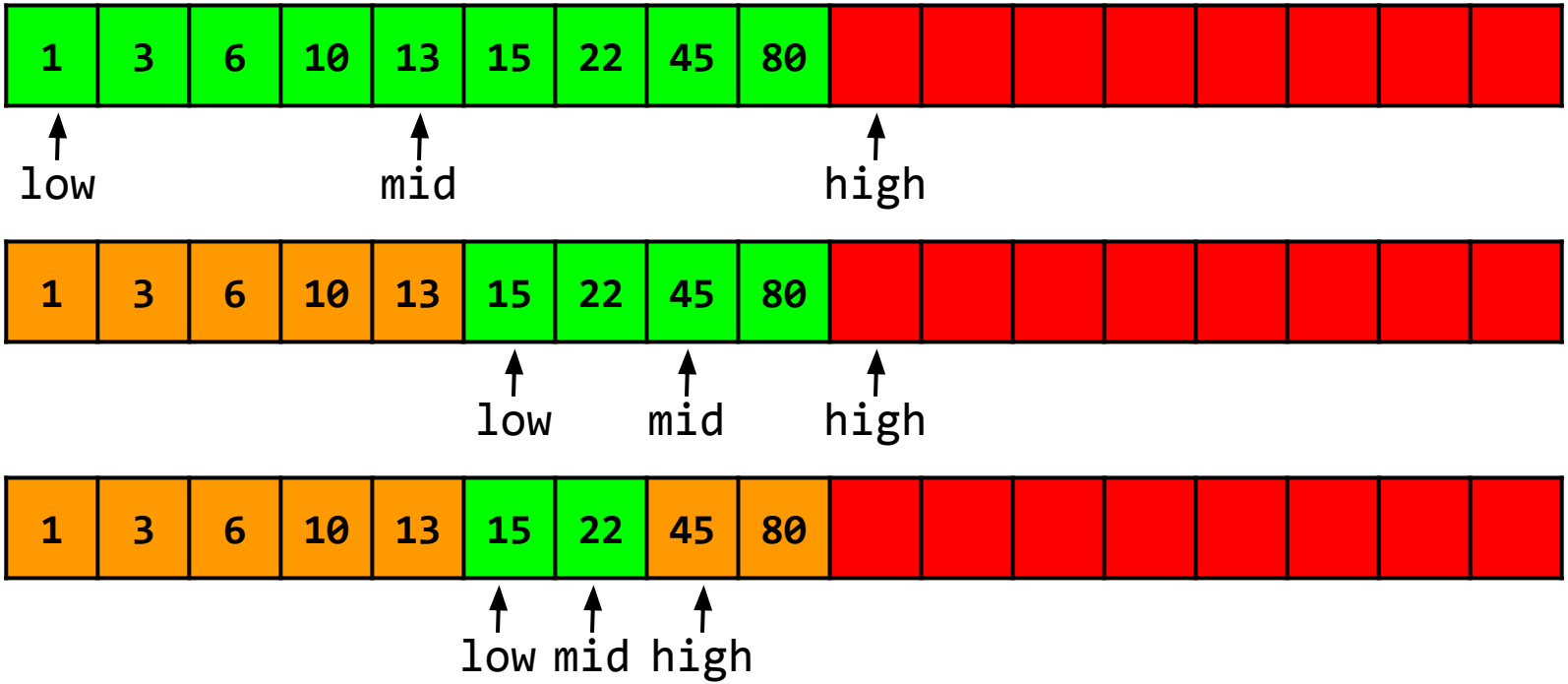
Подходы к размещению в памяти

Ассоциативная структура данных	Вставка по ключу	Удаление по ключу	Поиск по ключу
Неотсортированный массив	$O(1)$	$O(N)$	$O(N)$
Отсортированный массив	$O(N)$	$O(N)$	$O(\log_2(N))$
Список	$O(1)$	$O(N)$	$O(N)$
Дерево поиска	$O(\log_2(N))^{(1)}$	$O(\log_2(N))^{(1)}$	$O(\log_2(N))^{(1)}$
AVL-дерево	$O(\log_2(N))$	$O(\log_2(N))$	$O(\log_2(N))$

(1) В среднем случае $O(\log_2(N))$, в худшем – $O(N)$.

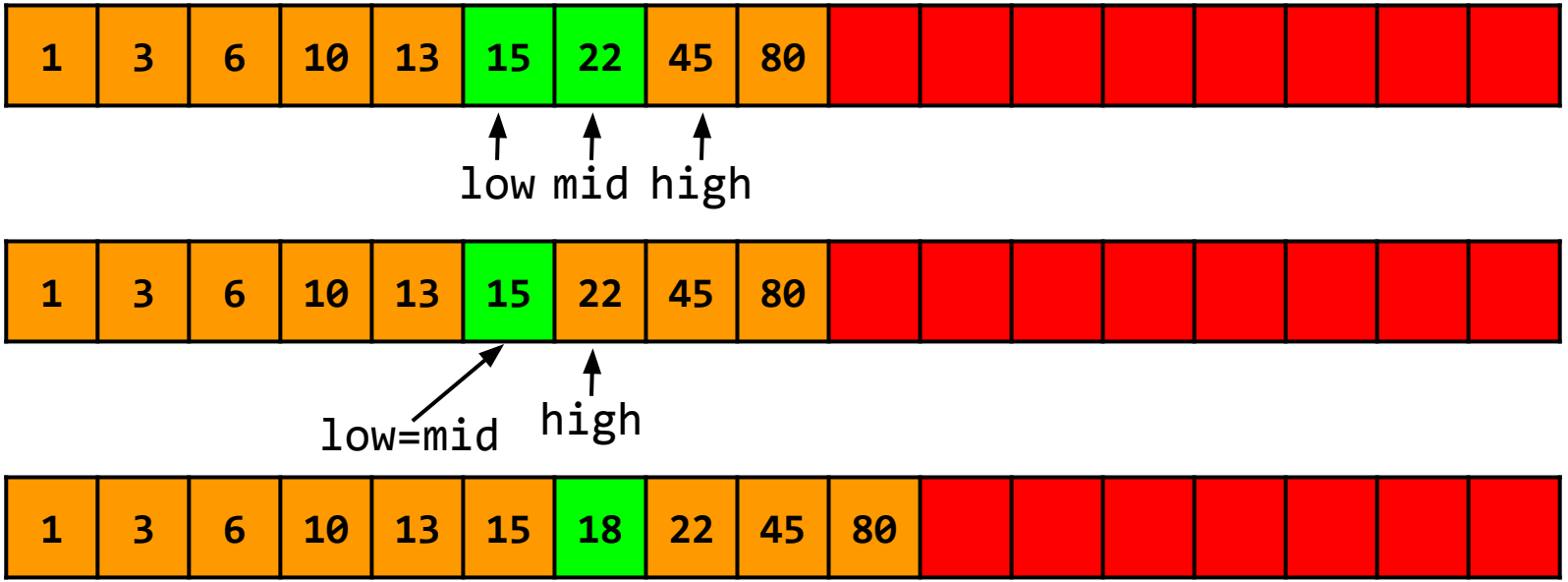
Вставка в отсортированный массив

Добавление элемента в хранилище: **db_insert(18)**



Вставка в отсортированный массив

Добавление элемента в хранилище: **db_insert(18)**

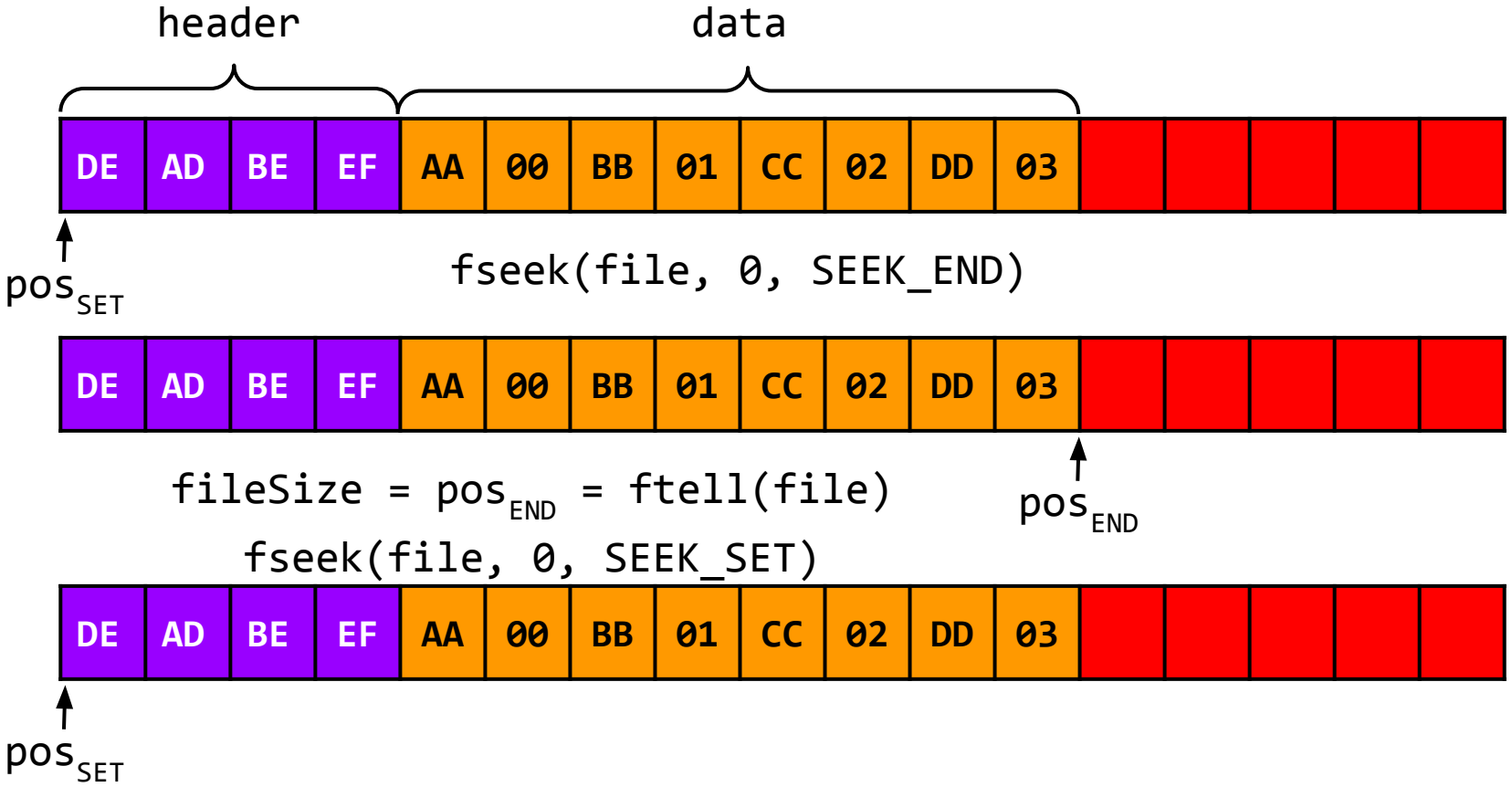


Вставка = поиск ($O(\log_2(N))$) + сдвиг ($O(N)$)

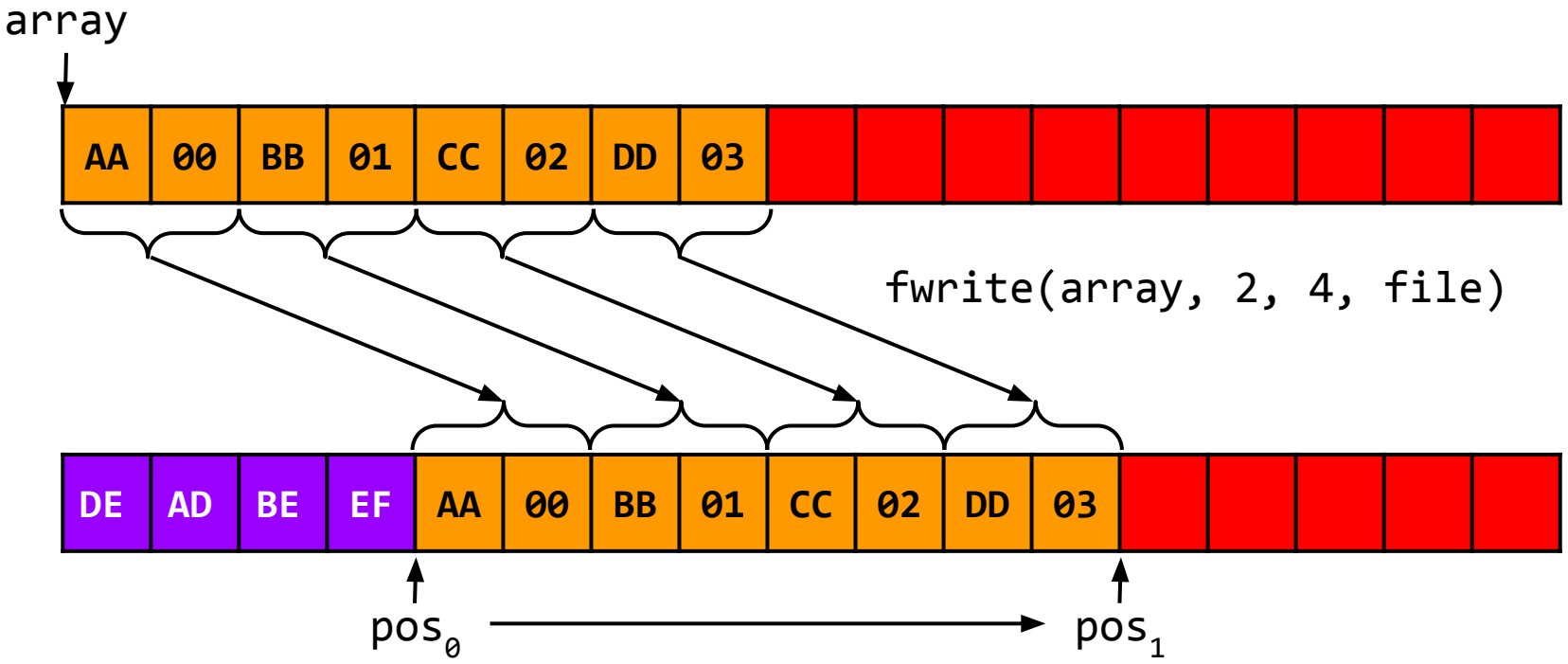
Удаление = поиск ($O(\log_2(N))$) + сдвиг ($O(N)$)

Работа с бинарными файлами

Измерение размера файла



Запись массива в бинарный файл



Запись в файл и чтение из него сдвигают текущую позицию в файле.
В режиме записи это также увеличивает файл.

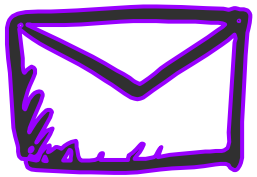
Проблема порядка байт

```
// Магические числа.
// Используются для идентификации формата.
const uint32_t MAGIC0 = 0xDEADBEEF;
const uint32_t MAGIC1 = 0xB01DFACE;
```

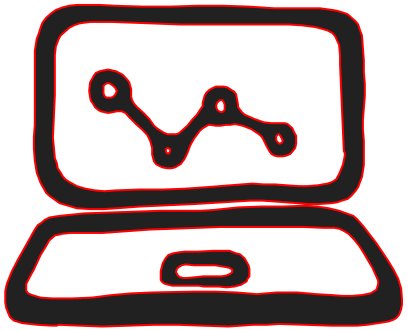
...	04	05	06	07	08	09	0A	0B	...
...	DE	AD	BE	EF	B0	1D	FA	CE	...

...	04	05	06	07	08	09	0A	0B	...
...	EF	BE	AD	DE	CE	FA	1D	B0	...

Big Endian



endianness = ???
(sizeof(int) = ???)



Little Endian

Смена порядка байт

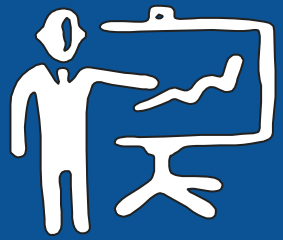
```c
uint32_t my_htobe32(uint32_t val)
{
#if __BYTE_ORDER__ == __ORDER_LITTLE_ENDIAN__
    // Используем встроенную функцию компилятора.
    return __builtin_bswap32(val);
#else
    return val;
#endif
}
```

```
my_htobe32(int32_t arg1)

        endbr64
        mov     eax, edi
        bswap   eax
        retn     {__retur
```

```c
uint32_t byteswap32_slow(uint32_t val)
{
    return ((val & 0xFF000000U) >> 24U) |
           ((val & 0x00FF0000U) >>  8U) |
           ((val & 0x0000FF00U) <<  8U) |
           ((val & 0x000000FFU) << 24U);
}
```

```
byteswap32_slow(int32_t arg

        endbr64
        mov     eax, edi
        bswap   eax
        retn     {__retur
```

Вопросы?



Красивые иконки взяты с сайта handdrawngoods.com