Алгоритми перетворення інформації.
Завдання 2, звіт

Михайло Голуб

27 лютого 2025 р.

Реалізація класу `BitArray`**:**

Цей клас зберігає бітові послідовності. На вхід приймається `in_bytes`, представник вбудованого класу `bytes`, та вказівник `bit_pointer`. Вбудований клас `bytes` це немутабельний масив чисел від 0 до 255. Вказівник `bit_pointer` вказує на перший біт, що не використовується в бітовій послідовності, яка знаходиться в `in_bytes`. Якщо послідовність займає усі біти байтів – вказівник на вході має бути рівним 8.

У вхідній послідовності байтів біти послідовності зберігаються в наступному порядку: [7, 6, 5, 4, 3, 2, 1, 0], [15, . . . , 8], . . . Тобто наймолодший біт першого байту відповідає найпершому біту послідовності.

Методи класу `BitArray`:

- `__str__` – повертає текстову репрезентацію бітової послідовності
- `__len__` – повертає довжину бітової послідовності
- `__rshift__` – дозволяє робити `bit_array >> n_bits`, що зменшує довжину послідовності шляхом видалення перших бітів
- `__lshift__` – дозволяє робити `bit_array << n_bits`, що дописує нулі на початок послідовності

Реалізація класу `BitSequenceFile`**:**

Цей клас є реалізацією бітового потоку. На вхід приймається розташування файлу та режим відкриття файлу. Режими відкриття файлу: -1 (за замовчуванням) – читання файлу, 0 – перезапис файлу, 1 – продовження файлу. Цей клас має методи `read, write, close` та атрибути `bit_pointer, byte_pointer, write_mode, file, opened_byte`.

Метод `read` бере на вхід аргумент `bits`. Метод переходить до байта на який вказує `byte_pointer` та записує у `BitArray` байти які мають довжину не менше `bit_pointer + bits` бітів, зміщує вправо `BitArray` на `bit_pointer`, змінює значення вказівників та повертає `BitArray`.

Метод `write` бере на вхід `BitArray`. Метод переходить до байта на який вказує `byte_pointer`, зміщує `BitArray` вліво на `bit_pointer`, додає `opened_byte` до першого байта з `BitArray`, записує останній байт з `BitArray` в `opened_byte`, та записує байти з `BitArray` в файл.

Метод `close` закриває відкритий файл. Оскільки в файл записуються та перезаписуються одразу байти, дописувати нулі на кінець файлу в цьому методі не потрібно.

Приклади роботи коду:

- `len(BitArray(bytes([0b1010]), 4))` → 4; Довжина послідовності 0101 – 4
- `str(BitArray(bytes([0b1010]), 4) >> 1)` → 101; Змістили послідовність 0101 направо – 101 (Зміщення направо скорочує (зменшує)

послідовність, для відповідності зменшенню значення байта при зміщенні направо)

- `str(BitArray(bytes([0b1010]), 4) << 1)` → 101; Змістили послідовність 0101 наліво – 00101 (Зміщення наліво додає нулі (збільшує) послідовність, для відповідності збільшенню значення байта при зміщенні наліво)
- `bit_reader = BitSequenceFile("files/hamlet.txt")` – створює представника бітового потоку для читання файлу
- `bit_writer = BitSequenceFile("files/hamlet2.txt 1)` – створює представника бітового потоку для дописування у файл
- `bit_writer = BitSequenceFile("files/hamlet2.txt 0)` – створює представника бітового потоку для переписування (перестворення) файлу
- `bit_reader.read(8)` – поверне бітову послідовність перших 8 бітів файлу
- `bit_writer.write(bit_array_instance)` – запише `bit_array_instance` в кінець файлу (якщо файл відкритий у режимі переписування – весь його зміст видалено і кінцеь файлу на початку).

У файлі `test.py` наведено код який демонструє результати виконання деяких команд і переписує перші `read_len` байтів з файлу (files/hamlet.txt) в файл (files/hamlet2.txt) шматками довжиною від `min_len` до `max_len` біт

Посилання на код на GitHub:
`https://github.com/MINIAProgramStudio/algorythms_of_data_transformation/tree/main/task2`