

Объяснение:

Получение элемента, s[i]: Сложность O(1): Доступ к элементу строки по индексу осуществляется за константное время. Строки в Python реализованы как массивы символов, и доступ к элементу по индексу не требует перебора других элементов.

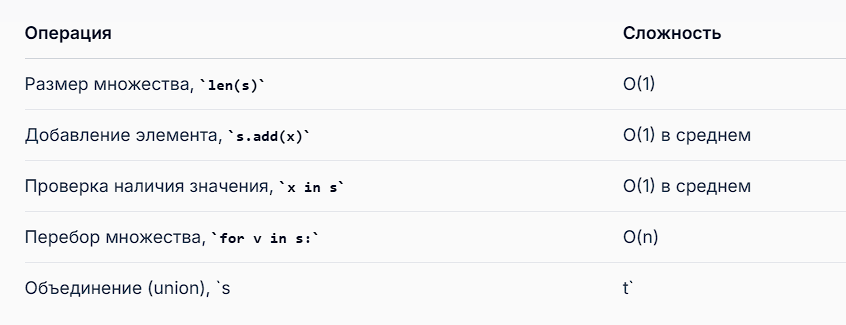
Пример: Если у вас есть строка s = "Hello", то s[1] вернет 'e' за O(1).

Размер строки, len(s): Сложность O(1): Python хранит длину строки как отдельное свойство. Таким образом, вызов len(s) выполняется за константное время, так как не требуется проходить по всем символам.

Пример: Для строки s = "Hello" вызов len(s) вернет 5 за O(1).

Получение среза, s[a:b]: Сложность O(k): При получении среза создается новый объект, который включает в себя копирование символов из исходной строки. Если длина среза равна k, то сложность будет O(k).

Пример: Для строки s = "Hello" вызов s[1:4] вернет 'ell', что требует копирования 3 символов, и, следовательно, будет O(3) = O(k).



Размер множества, len(s): Сложность O(1): Python также хранит длину множества как отдельное свойство. Таким образом, вызов len(s) выполняется за O(1).

Пример: Для множества s = {1, 2, 3} вызов len(s) вернет 3 за O(1).

Добавление элемента, s.add(x): Сложность O(1) в среднем: Добавление элемента в множество выполняется за O(1) в среднем, так как множества реализованы с использованием хеш-таблиц. Однако в худшем случае (например, при коллизиях) сложность может увеличиться.

Пример: Для множества s = {1, 2, 3} вызов s.add(4) добавит элемент за O(1) в среднем.

Проверка наличия значения, x in s: Сложность O(1) в среднем: Проверка наличия элемента в множестве также выполняется за O(1) в среднем благодаря хешированию. В худшем случае, как и при добавлении, сложность может быть выше.

Пример: Для множества s = {1, 2, 3} вызов 2 in s вернет True за O(1) в среднем.

Перебор множества, for v in s:: Сложность O(n): Перебор всех элементов множества выполняется за O(n), где n — количество элементов в множестве.

Пример: Для множества s = {1, 2, 3} перебор элементов с помощью for v in s: займет O(3).

Сложность O(len(s) + len(t)): Объединение двух множеств требует создания нового множества, в которое будут добавлены все уникальные элементы из обоих множеств. Для этого Python проходит по всем элементам первого множества s и добавляет их в новое множество, затем делает то же самое для второго множества t. Таким образом, если s содержит m элементов, а t содержит n элементов, то сложность будет O(m + n), что эквивалентно O(len(s) + len(t)).

Пример: Если у нас есть два множества s = {1, 2, 3} и t = {3, 4, 5}, то вызов s | t создаст новое множество {1, 2, 3, 4, 5}. Для этого потребуется пройти по всем элементам обоих множеств, что займет O(3 + 3) = O(6).