Практическое занятие №1: Основные структуры данных, простые алгоритмы

**Часть 2: алгоритмы**

1. Наивная реализация алгоритмов:

def has\_duplicates(arr):

for elem in arr:

i = 0

for elem1 in arr:

if elem == elem1:

i += 1

if i > 1:

return True

return False

def get\_duplicates(arr):

arr1 = []

for elem in arr:

if elem in arr1:

continue

i = 0

for elem1 in arr:

if elem == elem1:

i += 1

if i > 1:

arr1.append(elem)

break

return arr1

Теоретическая временная сложность: O(N2)

Время работы алгоритмов на входных массивах разного размера:

has\_duplicates:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 10 | 100 | 1000 | 10000 |
| Одинаковые значения | 0.0001 | 0.0001 | 0.0001 | 0.0001 |
| Случайные значения | 0.0001 | 0.0001 | 0.0001 | 0.0001 |
| Разные значения | 0.0001 | 0.0004 | 0.0289 | 3.0823 |

get\_duplicates:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 10 | 100 | 1000 | 10000 |
| Одинаковые значения | 0.0001 | 0.0001 | 0.0002 | 0.0003 |
| Случайные значения | 0.0001 | 0.0001 | 0.0002 | 0.0006 |
| Разные значения | 0.0001 | 0.0004 | 0.0324 | 2.8948 |

1. Эффективная реализация алгоритмов:

def has\_duplicates(arr):

sortedArr = sorted(arr)

for i in range(len(sortedArr)-1):

if sortedArr[i] == sortedArr[i+1]:

return True

return False

def get\_duplicates(arr):

return [k for k, v in Counter(arr).items() if v > 1]

Теоретическая временная сложность:O(N)

Время работы алгоритмов на входных массивах разного размера:

has\_duplicates:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 10 | 100 | 1000 | 10000 |
| Одинаковые значения | 0.0001 | 0.0001 | 0.0001 | 0.0002 |
| Случайные значения | 0.0001 | 0.0001 | 0.0002 | 0.0008 |
| Разные значения | 0.0001 | 0.0001 | 0.0002 | 0.0009 |

get\_duplicates:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 10 | 100 | 1000 | 10000 |
| Одинаковые значения | 0.0002 | 0.0001 | 0.0002 | 0.0006 |
| Случайные значения | 0.0001 | 0.0001 | 0.0001 | 0.0005 |
| Разные значения | 0.0001 | 0.0004 | 0.0002 | 0.0008 |

Вывод: эффективная реализация алгоритма работает намного быстрее, чем наивная реализация.

**Часть 1:Структуры данных.**

Таблица замеров времени работы:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Добавление в конец списка | Добавление в начало списка | Поиск элемента в списке | Добавление в множество | Поиск элемента в множестве |
| 1000 | 0.000234 | 0.000634 | 0.00049769 | 0.00042109 | 0.0007621 |
| 10000 | 0.004 | 0.019477 | 0.0028859 | 0.00185 | 0.0003542 |
| 100000 | 0.04382 | 1.904026 | 0.0242693 | 0.0230791 | 0.0004657 |

Теоретическая oценка для всех заданных операций в структурах: O(n) - Добавление в начало списка

O(n) - Добавление в конец списка

O(n) - Линейный поиск элемента в списке

O(n) - Добавление в множество

O(n) - Поиск элемента в множестве