sort_algos

Создано системой Doxygen 1.9.6

Глава 1

Алфавитный указатель пространств имен

1.1 Package List

Полный список документированных пакетов.

Глава 2

Алфавитный указатель классов

2.1 Классы

Классы с их кратким описанием.

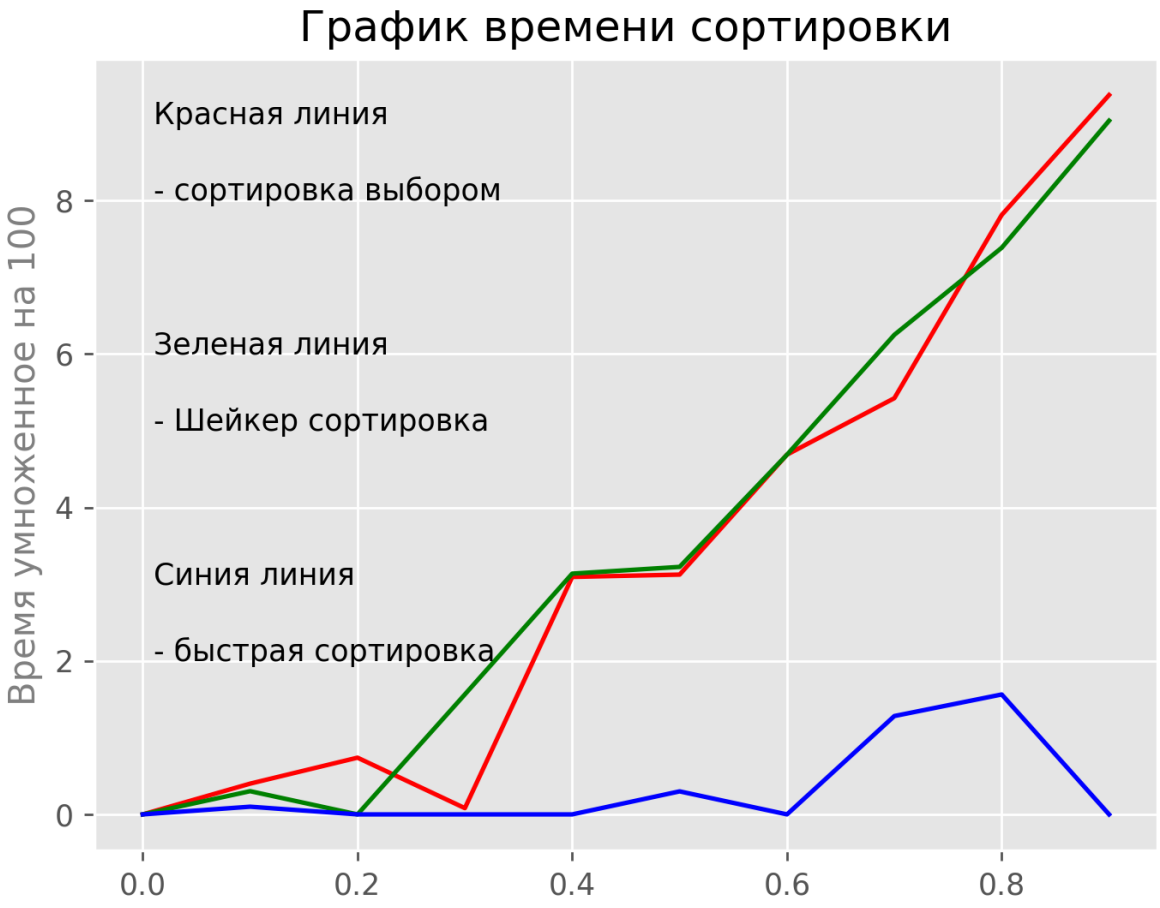
График времени сортировки
Красная линия
- сортировка выбором
Зеленая линия
- Шейкер сортировка
Синия линия
- быстрая сортировка
Время умноженное на 100
0
2
4
6
8
0.0
0.2
0.4
0.6
0.8

Глава 3

Пространства имен

3.1 Пространство имен Generation

Функции

- def str_time_prop (start, end, time_format, prop)
- def random_date (start, end, prop)
- def generation (n)

3.1.1 Подробное описание

Модуль для генерации объектов типа Brak: (Массив данных ЗАГСа)
ФИО жениха, дата рождения жениха,
ФИО невесты, дата рождения невесты,
дата бракосочетания, номер ЗАГСа
(сравнение по полям – номер ЗАГСа, дата бракосочетания, ФИО жениха)

3.1.2 Функции

3.1.2.1 generation()

def Generation.generation (
n)

Генерирует словарь длины n с полями:
ФИО жениха, дата рождения жениха,
ФИО невесты, дата рождения невесты,
дата бракосочетания, номер ЗАГСа.

3.1.2.2 str_time_prop()

```
def Generation.str_time_prop (
                start,
                end,
                time_format,
                prop )
```

Генерирует случайную дату между двумя датами

3.2 Пространство имен main

Переменные

- list size = [100, 200, 300, 400, 500, 1000, 100000]
- writer
- sheet_name
- index
- dict employees = {}
- pd c = pd.read_excel('./Marrigies.xlsx', sheet_name=f'{i}').to_dict('records')
- list c_employees = []
- list time_sel = []
- list time_fast = []
- list time_shaker = []
- list sorted_arrays = []
- copy sorted_arr_sel = copy.deepcopy(employees[j])
- time start = time.time()
- time end = time.time()
- copy sorted_arr_shaker = copy.deepcopy(employees[j])
- copy sorted_arr_fast = copy.deepcopy(employees[j])
- dict dictionary = {}
- list full_name_hus = []
- list b_data_hus = []
- list full_name_wife = []
- list b_data_wife = []
- list mar_date = []
- list num_zags = []
- str file_name = "./Marrigies_sorted_insert.xlsx"
- else :
- str mode = 'w'
- engine

3.2.1 Подробное описание

В данном модуле генерируем наборы данных различной размерности.
Далее записываем их в .xlsx. Затем считываем их из файла, сортируем и
сохраняем в новых файлах. С помощью таймера засекаем время работы функций сортировок.

3.3 Пространство имен Sortes

Функции

- def selection_sort (arr)
- def shaker_sort (arr)
- def quick_sort (arr)

3.3.1 Подробное описание

Модуль с реализованными сортировками:
Сортировка простыми вставками, шейкер, слиянием
Сортировка выбором, шейкер, быстрая

3.3.2 Функции

3.3.2.1 quick_sort()

def Sortes.quick_sort (
arr)

Быстрая сортировка
Выбрать опорный элемент из массива - обычно первый или средний элемент.
Разделить массив на два подмассива: элементы меньше опорного и элементы больше опорного.
Рекурсивно применить сортировку к двум подмассивам.

Средняя сложность O(n * log2n)

3.3.2.2 selection_sort()

def Sortes.selection_sort (
arr)

Сортировка выбором
Алгоритм сортировки:
1. Найти наименьшее значение в списке.
2. Записать его в начало списка, а первый элемент - на место, где раньше стоял наименьший.
3. Снова найти наименьший элемент в списке. При этом в поиске не участвует первый элемент.
4. Второй минимум поместить на второе место списка. Второй элемент при этом перемещается на освободившееся место.
5. Продолжать выполнять поиск и обмен, пока не будет достигнут конец списка.

Средняя сложность O(n ^ 2)

3.3.2.3 shaker_sort()

def Sortes.shaker_sort (
arr)

Шейкер сортировка
Коктейльная сортировка — разновидность пузырьковой сортировки. Алгоритм пузырьковой сортировки всегда обходит элементы слева и перемещает самый большой элемент в правильное положение на первой итерации,
второй по величине — на второй и так далее. Коктейльная сортировка попеременно проходит через заданный массив в обоих направлениях. Коктейльная сортировка не требует ненужных итераций,
что делает ее эффективной для больших массивов.
Этапы сортировки:
1. Массив перебирается слева направо, как и при пузырьковой сортировке.
Во время цикла сравниваются соседние элементы, и если значение слева больше значения справа,
значения меняются местами. В конце первой итерации наибольшее число будет находиться в конце массива.
2. Проход по массиву в обратном направлении — начиная с элемента,
непосредственно предшествующего последнему отсортированному элементу, и возвращаясь к началу массива.
Здесь также сравниваются соседние элементы и при необходимости меняются местами.

Средняя сложность O(n ^ 2)

Глава 4

Классы

4.1 Класс Brak.Brak

Открытые члены

- def __init__ (self, fio_hus, bd_hus, fio_wife, bd_wife, mar_date, num_zags)
- def __lt__ (self, other)
- def __le__ (self, other)
- def __gt__ (self, other)
- def __ge__ (self, other)

Открытые атрибуты

- fio_hus
- bd_hus
- fio_wife
- bd_wife
- mar_date
- num_zags

4.1.1 Подробное описание

Класс, описывающий брак

4.1.2 Конструктор(ы)

4.1.2.1 __init__()

```
def Brak.Brak.__init__ (
                self,
                fio_hus,
                bd_hus,
                fio_wife,
                bd_wife,
                mar_date,
                num_zags )
```

Инициализация объекта

4.1.3 Методы

4.1.3.1 __ge__()

```
def Brak.Brak.__ge__ (
                self,
                other )
```

Перегрузка оператора >=

4.1.3.2 __gt__()

```
def Brak.Brak.__gt__ (
                self,
                other )
```

Перегрузка оператора >

4.1.3.3 __le__()

```
def Brak.Brak.__le__ (
                self,
                other )
```

Перегрузка оператора <=

4.1.3.4 __lt__()

```
def Brak.Brak.__lt__ (
                self,
                other )
```

Сравнение по полям – номер ЗАГСа, дата бракосочетания, ФИО жениха
Перегрузка оператора <

Объявления и описания членов класса находятся в файле:

- Brak.py

Предметный указатель