11.2 Основы работы со сп...
11.3 Методы списков. Час...
11.4 Вывод элементов сп...
11.5 Методы строк. split()...
11.6 Методы списков. Час...
11.7 Списочные выражен...
11.8 Сортировка списков
Экзамен 12 Итоговая раб...
12.1 Часть 1
12.2 Часть 2
13 Функции
13.1 Функции без параметр...

Сортировка простыми вставками

Алгоритм сортировки простыми вставками делит список на 2 части — отсортированную и неотсортированную. Из неотсортированной части извлекается очередной элемент и вставляется на нужную позицию в отсортированной части, в результате чего отсортированная часть списка увеличивается, а неотсортированная уменьшается. Так происходит, пока не исчерпан набор входных данных и не отсортированы все элементы.

Сортировка простыми вставками наиболее эффективна, когда список уже частично отсортирован и элементов массива немного. Если элементов в списке меньше 10, то этот алгоритм — один из самых быстрых.

Рассмотрим его работу на примере сортировки списка `a = [5, 1, 8, 2, 4]` по возрастанию.

Первый проход:

Делим список на две части: отсортированную `[5]` и неотсортированную `[1, 8, 2, 4]`.

Извлекаем первый элемент `1` из неотсортированной части списка и находим ему место в отсортированной части:

`[1, 5, 8, 2, 4]`.

Второй проход:

Делим список на две части: отсортированную `[1, 5]` и неотсортированную `[8, 2, 4]`.

Извлекаем первый элемент `8` из неотсортированной части списка и находим ему место в отсортированной части:

`[1, 5, 8, 2, 4]`.

Третий проход:

Делим список на две части: отсортированную `[1, 5, 8]` и неотсортированную `[2, 4]`.

Извлекаем первый элемент `2` из неотсортированной части списка и находим ему место в отсортированной части:

`[1, 2, 5, 8, 4]`.

Четвертый проход:

Делим список на две части: отсортированную `[1, 2, 5, 8]` и неотсортированную `[4]`.

Извлекаем первый элемент `4` из неотсортированной части списка и находим ему место в отсортированной части:

`[1, 2, 4, 5, 8]`.

Теперь список отсортирован, и алгоритм может быть завершен.

Визуализация алгоритма

Выберите в выпадающем списке алгоритм **Insert sort**.

Реализация алгоритма

Пусть требуется отсортировать по возрастанию список чисел: `a = [1, 7, -3, 9, 0, -67, 34, 12, 45, 1000, 6, 8, -2, 99]`.

Следующий программный код реализует алгоритм сортировки простыми вставками:

```
a = [1, 7, -3, 9, 0, -67, 34, 12, 45, 1000, 6, 8, -2, 99]
n = len(a)

for i in range(1, n):
    elem = a[i]  # берем первый элемент из неотсортированной части списка
    j = i

    # пока элемент слева существует и больше нашего текущего элемента
    while j >= 1 and a[j - 1] > elem:
        # меняем j-1-й элемент отсортированной части вправо
        a[j] = a[j - 1]
        # сами идём влево, дальше ищем место для нашего текущего элемента
        j -= 1

    # нашли место для нашего текущего элемента из неотсортированной части
    # и вставляем его на индекс j в отсортированной части
    a[j] = elem

print('Отсортированный список:', a)
```

Результатом выполнения такого кода будет:

```
Отсортированный список: [-67, -3, -2, 0, 1, 6, 7, 8, 9, 12, 34, 45, 99, 1000]
```

Оптимизация алгоритма

Алгоритм сортировки простыми вставками можно значительно ускорить, если осуществлять поиск нужной позиции для вставки очередного элемента из неотсортированной части списка с помощью бинарного поиска.

Happy Pythoning!

9708 465 Шаг 8

Следующий шаг

198 Комментариев

Самые популярные

Будьте вежливы и соблюдайте наши принципы сообщества. Пожалуйста, не оставляйте решения и подсказки в комментариях, для этого есть отдельный форум.

Оставить комментарий

9 Комментарий закреплён

Поколение Python · 2 года назад

Официальный telegram-канал 'Поколение Python'

Рады объявить, что у линейки курсов "Поколение Python" появился официальный telegram-канал!
Узнавайте первыми о новых курсах, изучайте с нами программирование и язык Python, решайте новые задачи и заряжайтесь отличным настроением!
Прямая ссылка на канал: t.me/pygen_ru.

Made with by BEEGEEK

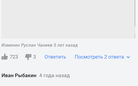
15 3 Ответить · Посмотреть 14 ответов (ответ преподавателя)

Дарья Топорова · 5 лет назад

пока что самая сложная тема для понимания, не очень понимаю, как работает реализация алгоритма, хотя теория ясна

543 1 Ответить · Посмотреть 10 ответов

Alex Bezsulrov · 5 лет назад



Изменил Руслан Чанев 3 лет назад

723 3 Ответить · Посмотреть 2 ответа

Иван Рыбакин · 4 года назад

Иногда, для понимания всех процессов в программе, в вставляю принты в код. Оно помогает разобраться.

```
a = [1, 7, -3, 9, 0, -67, 34, 12, 45, 1000, 6, 8, -2, 99]
n = len(a)
print('Начинаем цикл перебора всех элементов списка, начиная со второго (неотсортированный список)')
for i in range(1, n):
    print(a, i, 'итерация: a[i] >', a[i])
    elem = a[i]  # первый элемент из неотсортированной части списка
    print('Запоминаем проверяемый элемент списка в доп память : elem = a[i] =', a[i])
    j = i
    if a[j - 1] <= elem:
        print('Я while не заходил, тк проверяемое число ({a[i]}) больше предыдущего ({a[i - 1]})')
    while j >= 1 and a[j - 1] > elem:
        print('Я while: сравниваем индекс = {j}: на месте a[j] = {a[j]} записываем число {a[j - 1]}, и получаем', end= ' ')
        a[j] = a[j - 1]
        print(a)
        j -= 1
    print('Извлекаем из доп памяти elem = {elem} в индекс {j}')
    a[j] = elem
print(a)
```

532 11 Ответить · Посмотреть 21 ответ

Виталий Колосов · 5 лет назад

Да сантехи или list.sort(), да избавят оно от переборов дывольских и код сократит. И если я правильно понял - в примере кода реализация алгоритма предпоследняя строка кода лишняя: print(a)

Изменил Виталий Колосов 5 лет назад

286 3 Ответить · Посмотреть 6 ответов

Banunyka Denis · 4 года назад

Пожелание по подаче материала.

Сама логика алгоритма понятна, есть и ваш детальный пример, и визуальное оформление. Но очень не хватило детализации по какой логике стремилась каждая строчка кода. Зачем j=i, почему в условии есть j>=1 и тд. Если добавить чуть больше пояснений и Я будет в разы легче разобраться, какие именно действия закладывались с точки зрения кода, а не самого принципа работы сортировки.

255 2 Ответить

Виктор Селезнёв · 3 года назад

Я конечно понял как это работает, но сам бы такой код рождал долго и в муках, может даже с летальным исходом для одного из нас. Даже следить за работой алгоритма больно

131 Ответить · Посмотреть 1 ответ

Aleksandr Filippov · 4 года назад

По-моему, так больше похоже на описанный алгоритм

```
a = [1, 7, -3, 9, 0, -67, 34, 12, 45, 1000, 6, 8, -2, 99]
n = len(a)

for i in range(1, n):
    e = a.pop(i)
    while i >= 1 and a[i - 1] > e:
        i -= 1
    a.insert(i, e)

print(a)
```

131 9 Ответить · Посмотреть 10 ответов

Александр Шевцов · 3 года назад

https://www.youtube.com/watch?v=MWvNTp_wFA

очень понятно рассказано

80 Ответить · Посмотреть 9 ответов

Александра Толпа · 3 года назад

Эта же сортировка, но тут код в разы понятнее:

```
a = [1, 7, -3, 9, 0, -67, 34, 12, 45, 1000, 6, 8, -2, 99]
n = len(a)

for i in range(1, n):
    for j in range(i, 0, -1):
        if a[j] < a[j - 1]:
            a[j], a[j - 1] = a[j - 1], a[j]
        else:
            break
print(a)
```

76 3 Ответить · Посмотреть 2 ответа

Анастасия Попова · 4 года назад

Спокойно понимаю наглядный пример с текстом, но код это какой-то мрак для меня))

57 Ответить

Александр Никитин · 3 года назад

самый непонятный раздел

51 Ответить · Посмотреть 3 ответа

Артем Становов · 4 года назад

Хорошо, что хоть этот алгоритм оптимизировать не пришлось:))

41 Ответить · Посмотреть 1 ответ

Stanislav Kazantsev · 5 лет назад

Как то условности для понимания алгоритм тем, что постоянно переносили бОльшие числа из младших разрядов в старшие, а само сравнивание чисел в цикле держали. А вот в визуализаторе код на запчасть, не мог до конца понять, что происходит. Да и даже после этого мне это кажется усложнением.

Думаю, что если бы в цикле while сразу меняли местами два числа (a[j], a[j - 1] = a[j - 1], a[j]), а строку a[j] = elem после цикла вообще удалили бы - было бы понятнее.

```
a = [1, 7, -3, 9, 0, -67, 34, 12, 45, 1000, 6, 8, -2, 99]
n = len(a)

for i in range(1, n):
    elem = a[i]  # первый элемент из неотсортированной части списка
    j = i
    while j >= 1 and a[j - 1] > elem:
        a[j], a[j - 1] = a[j - 1], a[j]
        j -= 1

print(a)
print('Отсортированный список:', a)
```

40 2 Ответить · Посмотреть 4 ответа

Антон Бушин · 3 года назад

Ни гидро про оптимизацию:

```
Небинарная сортировка из урока:
        скорость для 10 элем.: 6.46164451699953314
        скорость для 100 элем.: 3.1105287959918107
        скорость для 1_000 элем.: 35.97372513880072
Небинарная сортировка из комментов:
        скорость для 10 элем.: 8.65661354099973876
        скорость для 100 элем.: 3.99552208399915445
        скорость для 1_000 элем.: 47.79769407380187
Бинарная сортировка:
        скорость для 10 элем.: 8.7600599535051376
        скорость для 100 элем.: 2.48348960099997465
        скорость для 1_000 элем.: 23.65789416080205
Бинарная сортировка + .pop() & .insert():
        скорость для 10 элем.: 9.55889957599886446
        скорость для 100 элем.: 6.43743364408424385
        скорость для 1_000 элем.: 1.15926492998875
```

```
import timeit
a = [1, 7, -3, 9, 0, -67, 34, 12, 45, 1000, 6, 8, -2, 99]
```

Показать полностью

Изменил Антон Бушин 3 года назад

31 Ответить · Посмотреть 7 ответов

Константин Константин · 4 года назад

не понятно совсем

23 Ответить

Дмитрий Гудков · 3 года назад

Немного непонятное объяснение данного метода сортировки. Кому интересно, более понятно и доступно:

(4) #9. Сортировка вставками | Алгоритмы на Python - YouTube

18 Ответить · Посмотреть 3 ответа

Показать обсуждения