Сортировка пузырьком

Алгоритм сортировки пузырьком состоит из повторяющихся проходов по сортируемому списку. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по списку повторяются $n - 1$ раз, где n – длина списка. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент списка ставится на свое место в конце списка рядом с предыдущим «наибольшим элементом».

Наибольший элемент каждый раз «всплывает» до нужной позиции, как пузырёк в воде — отсюда и название алгоритма.

Алгоритм пузырьковой сортировки считается учебным и практически не применяется вне учебной литературы, а на практике применяются более эффективные.

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

Первый проход:

1. `[5, 1, 4, 2, 8]` → `[1, 5, 4, 2, 8]` : меняем местами первый и второй элементы, так как $5 > 1$;
2. `[1, 5, 4, 2, 8]` → `[1, 4, 5, 2, 8]` : меняем местами второй и третий элементы, так как $5 > 4$;
3. `[1, 4, 5, 2, 8]` → `[1, 4, 2, 5, 8]` : меняем местами третий и четвертый элементы, так как $5 > 2$;
4. `[1, 4, 2, 5, 8]` → `[1, 4, 2, 5, 8]` : не меняем четвертый и пятый элементы местами, так как $5 < 8$;
5. Самый большой элемент встал («всплыл») на свое место.

Второй проход:

1. `[1, 4, 2, 5, 8]` → `[1, 4, 2, 5, 8]` : не меняем первый и второй элементы местами, так как $1 < 4$;
2. `[1, 4, 2, 5, 8]` → `[1, 2, 4, 5, 8]` : меняем местами второй и третий элементы, так как $4 > 2$;
3. `[1, 2, 4, 5, 8]` → `[1, 2, 4, 5, 8]` : не меняем местами третий и четвертый элементы, так как $4 < 5$;
4. Второй по величине элемент встал («всплыл») на свое место.

Теперь список полностью отсортирован, но алгоритму это неизвестно и он работает дальше.

Третий проход:

1. `[1, 2, 4, 5, 8]` → `[1, 2, 4, 5, 8]` : не меняем первый и второй элементы местами, так как $1 < 2$;
2. `[1, 2, 4, 5, 8]` → `[1, 2, 4, 5, 8]` : не меняем второй и третий элементы местами, так как $2 < 4$;
3. Третий по величине элемент встал («всплыл») на свое место. (на котором и был)

Четвертый проход:

1. `[1, 2, 4, 5, 8]` → `[1, 2, 4, 5, 8]` :
2. Четвертый по величине элемент встал («всплыл») на свое место.

Теперь список отсортирован и алгоритм может быть завершен.

Визуализация алгоритма

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 пузырьковой сортировки:

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

for i in range(n - 1):
    for j in range(n - 1 - i):
        if a[j] > a[j + 1]:                 # если порядок элементов пары неправильный
            a[j], a[j + 1] = a[j + 1], a[j]  # меняем элементы пары местами

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 пузырьковой сортировки можно немного ускорить. Если на одном из очередных проходов окажется, что обмены больше не нужны, то это означает, что все элементы списка находятся на своих местах, то есть список отсортирован. Для реализации такого ускорения нужно воспользоваться сигнальной меткой, то есть флажком и оператором прерывания `break`.

Happy Pythoning!

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

25 Ответить

July Romashin 4 года назад
Пузырьковая сортировка - сортировка, в которой без пузырька не разобраться)

1311 11 Ответить Посмотреть 6 ответов

Alex Beznutrov 5 лет назад

8 5 3 1 4 7 9

Изменен Руслан Чаниев 5 лет назад

876 13 Ответить Посмотреть 5 ответов

Андрей Налетов 3 года назад
короче, я долго пытался въехать как работает:
for i in range(n - 1):
    for j in range(n - i - 1)
Не мог понять даже по комментам подробным, но вот дошло, объясняю для таких как я))

в каждом цикле 'for' программа просто перебирает индексы (цифры по очереди)- представьте, как курсор просто идет вправо по каждой цифре. Так вот первое for - это сколько полных итераций сделает программа- 13 штук (меньше на одну т.к. последнее число в последовательности не с чем сравнивать.
А вот второе for значит, что каждый раз программа будет перебирать на 1 число меньше (справа). так как справа у нас после первой итерации оказывается самое большое число и его уже не нужно сравнивать, оно там остается. После второй итерации - справа у нас 2 числа самых больших, которые там остаются и так далее.

А вот после for уже идет условие, которое программа проверяет -если число с нулевым идексом (самое первое) больше числа с первым индексом (j +1), то они меняются местами. если нет, то программа просто дальше перебирает индексы.

))

702 Ответить Посмотреть 11 ответов

Юлия ренре 4 года назад
Уууу и тут я поняла что это лес...
Изменен Юлия ренре 4 года назад

232 6 Ответить Посмотреть 2 ответа

Абдурахман Танатар 4 года назад
прочитайте книгу Грокаем алгоритмы https://t.me/it_boooks/187 вот ссылка
Изменен Абдурахман Танатар 4 года назад

89 3 Ответить Посмотреть 6 ответов

Артём Лисейкин 5 лет назад
Объясните, почему во втором цикле границы n - i - 1?

72 1 Ответить Посмотреть 11 ответов

Olesja Fink 4 года назад
Кому всё равно непонятно. На ютубе есть отличные видео, где танец"элементов" нагляднейшим образом показывает работу алгоритмов сортировки. Посмотрите, там ну ОЧЕНЬ наглядно(есть отдельные ролики для разных алгоритмов, гуглить "sort with Hungarian folk dance")
Bubble-sort with Hungarian folk dance https://youtu.be/lyZQPjUT5B4

66 Ответить Посмотреть 9 ответов

Евгений Петрович Шамаев 4 года назад

```
a = [1, 7, -3, 9, 0, -67, 34, 12, 45, 1000, 6,  8, -2, 99]
n = len(a)
flag = False
for i in range(n - 1):
   for j in range(n - i - 1):
      if a[j] > a[j + 1]:               # если порядок элементов пары неправильный
         a[j], a[j + 1] = a[j + 1], a[j]  # меняем элементы пары местами
         flag = False
      else:
         flag = True
   if flag:
      break
```

83 24 Ответить Посмотреть 20 ответов

Гайворонский Никита 4 года назад
Можете объяснить зачем это знать, если есть метод sort()?

45 8 Ответить Посмотреть 29 ответов (ответ преподавателя)

Anonymous 395632846 4 года назад
метотд sort(): ну да ну да

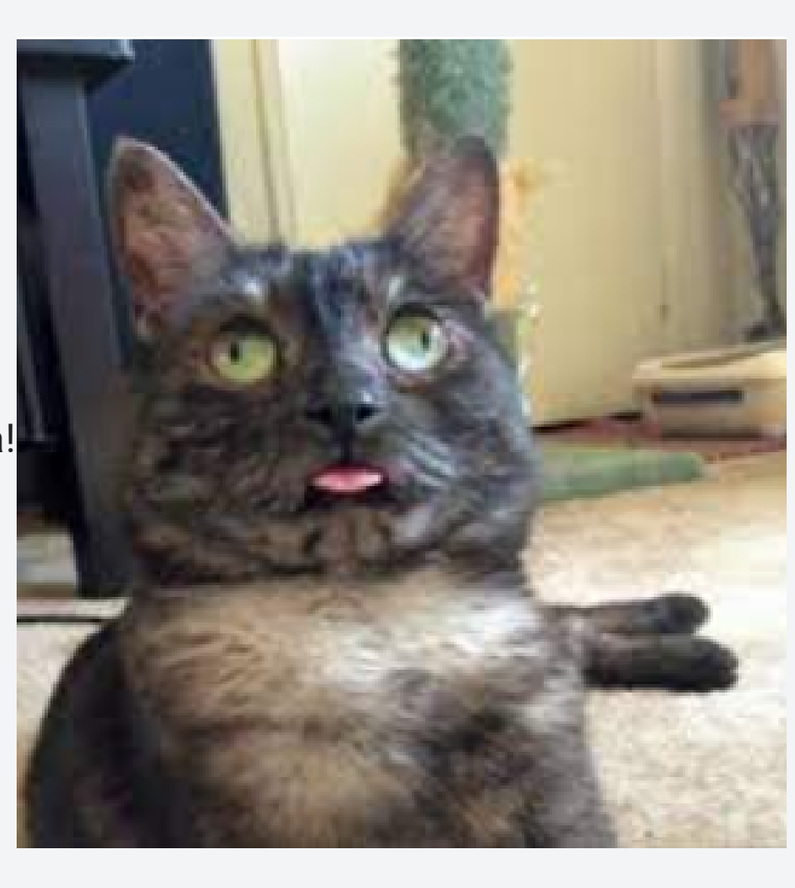
33 1 Ответить

Vladislav Rakitin 2 года назад
ГОСПОДА. Самаконцепция мне была понятна, но как именно это работает в коде я долго не мог понять, особенно range во втором цикле (n-i-1)
вот этот чел всё объясняет МАКСИМАЛЬНО ДОХОДЧИВО.
мне очень помогло
https://www.youtube.com/watch?v=WBaL7ANQbzQ

23 Ответить Посмотреть 7 ответов

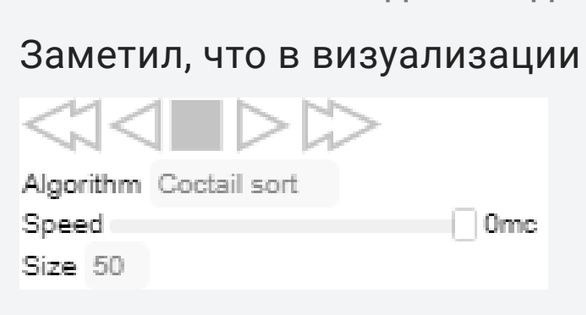
Тэмутджин Косжанов 2 года назад
метод sort() в пайтоне был придуман в 1980 году
Люди до 1980 года:

22 Ответить Посмотреть 1 ответ

Максим Сивишкин 2 года назад
Я залип на Визуализация алгоритма!  Медленно, но красиво, собака!)))
Изменен Максим Сивишкин 2 года назад

20 Ответить

Олег Маловик 4 года назад
Заметил, что в визуализации можно выбрать алгоритм, скорость и объём списка.



Изменен Олег Маловик 4 года назад

18 Ответить Посмотреть 1 ответ

Лидия Гавловская 2 года назад
Спойлеров решения задач в комменте нет, я просто попыталась разобраться в коде пузырьковой сортировки)) Долго смотрела на этот код, как баран на новые ворота: было непонятно, что именно олицетворяют собой циклы и почему такие аргументы у функций range. Кажется, разобралась. Прошу не кидаться тапками, если это уже было, так как пишу скорее для себя. Но поправьте меня, ежели чё.

Итак, проход - цикл попарных сравнений.

Внешний цикл - это все проходы по списку. Поскольку каждый проход ставит один элемент на своё место, проходов нужно на один меньше, чем число элементов: последний элемент (по умолчанию - наименьший) окажется на своём месте методом исключения, когда остальные (бОльшие) "всплывут" выше него. Таким образом, аргумент у range внешнего цикла - количество необходимых проходов. Наверное, можно сказать, что это количество степеней свободы внутри списка (тех, кто меня понял, прошу поправить формулировку, а кто не понял - не бойтесь, это не имеет отношения к делу).

Внутренний цикл - конкретный проход по списку. Здесь аргумент у range - количество попарных сравнений. На первом проходе (первой итерации внешнего цикла) сравнений должно быть n - 1, так как последний элемент сравнивать будет не с чем. А каждому последующему проходу будет нужно на одно сравнение меньше, так как каждый предшествующий проход поставил по одному элементу на своё место в конец списка. С этими поставленными на место элементами сравнивать всплывающий элемент не надо: они уже сравнивались не в пользу всплывающего сейчас. То есть, чем больше итераций успел сделать внешний цикл, тем меньше их нужно внутреннему. Чем больше было проходов, тем меньше нужно попарных сравнений.

14 Ответить

Bodnar Yaroslav в прошлом году
Этот код представляет собой алгоритм сортировки пузырьком, который проходит список множество раз, сравнивая каждую пару соседних элементов и меняя их местами, если они находятся в неправильном порядке. Давайте разберем его пошагово:

1. `a = [1, 7, -3, 9, 0, -67, 34, 12, 45, 1000, 6, 8, -2, 99]` : Задается список чисел, который нужно отсортировать.
2. `n = len(a)` : Определяется длина списка `a`.
3. `for i in range(n - 1):` : Это внешний цикл, который пройдет по списку `n - 1` раз. Он используется для контроля количества проходов по списку. После каждого прохода наибольший элемент становится на своем месте, поэтому нет необходимости проходить по всему списку на последующих итерациях.
4. `for j in range(n - 1 - i):` : Это внутренний цикл. Он проходит по каждому элементу в списке на текущем проходе.
5. `if a[j] > a[j + 1]:` : Это условие проверяет, является ли текущий элемент больше следующего элемента.
6. `a[j], a[j + 1] = a[j + 1], a[j]` : Если условие из пункта 5 истинно (текущий элемент больше следующего), то значения текущего элемента и следующего элемента меняются местами. Это и есть основной шаг сортировки пузырьком.
7. После завершения внутреннего цикла для одного прохода, наибольший элемент перемещается в конец списка.
8. Процесс повторяется для каждого элемента, пока весь список не будет отсортирован.
9. `print('Отсортированный список:', a)` : Выводится отсортированный список.

Может кому станет понятней

13 Ответить Посмотреть 3 ответа