Введение в Python

Сортировки. Продолжение.

Алексей Сорокин

спецкурс, ОТИПЛ МГУ,
осенний семестр 2017–2018 учебного года
1 ноября 2017 г.

Сортировка вставками

- Таким образом, сортировка выбором требует $O(n^2)$ операций. Нельзя ли сделать быстрее?

Сортировка вставками

- Таким образом, сортировка выбором требует $O(n^2)$ операций. Нельзя ли сделать быстрее?
- Попробуем применить двоичный поиск...

Сортировка вставками

- Таким образом, сортировка выбором требует $O(n^2)$ операций. Нельзя ли сделать быстрее?
- Попробуем применить двоичный поиск...
- Алгоритм сортировки вставками:
  - i — длина уже отсортированного участка, вначале $i = 0$.

Сортировка вставками

- Таким образом, сортировка выбором требует $O(n^2)$ операций. Нельзя ли сделать быстрее?
- Попробуем применить двоичный поиск...
- Алгоритм сортировки вставками:
  - i — длина уже отсортированного участка, вначале $\mathrm{i} = 0$.
  - найти j — место вставки $\mathrm{a[i]}$ в $\mathrm{a[:i]}$ с помощью двоичного поиска.

Сортировка вставками

- Таким образом, сортировка выбором требует $O(n^2)$ операций. Нельзя ли сделать быстрее?
- Попробуем применить двоичный поиск...
- Алгоритм сортировки вставками:
  - i — длина уже отсортированного участка, вначале $\mathrm{i} = 0$.
  - найти j — место вставки $\mathrm{a[i]}$ в $\mathrm{a[:i]}$ с помощью двоичного поиска.
  - “Сдвинуть” срез $\mathrm{a[j:i]}$ вправо на 1 элемент.
  - Присвоить $\mathrm{a[j]} = \mathrm{a[i]}$.

Сортировка вставками

- Таким образом, сортировка выбором требует $O(n^2)$ операций. Нельзя ли сделать быстрее?
- Попробуем применить двоичный поиск...
- Алгоритм сортировки вставками:
  - i — длина уже отсортированного участка, вначале $\mathrm{i} = 0$.
  - найти j — место вставки $\mathrm{a[i]}$ в $\mathrm{a[:i]}$ с помощью двоичного поиска.
  - “Сдвинуть” срез $\mathrm{a[j:i]}$ вправо на 1 элемент.
  - Присвоить $\mathrm{a[j]} = \mathrm{a[i]}$.
  - Увеличить i на 1, если $i = \textbf{len}(\mathrm{a})$, завершить работу.

Сложность сортировки вставками

Утверждение

На определение места вставки в упорядоченный массив длины m требуется не более, чем $\log_2 m + 1$ сравнений.

Сложность сортировки вставками

Утверждение

На определение места вставки в упорядоченный массив длины m требуется не более, чем $\log_2 m + 1$ сравнений.

Утверждение

На сортировку массива длины n двоичными вставками требуется не более $n(\log_2 n + 1)$ сравнений и $\frac{(n)(n-1)}{2}$ перестановок.

Сложность сортировки вставками

Утверждение

На сортировку массива длины n двоичными вставками требуется не более $n(\log_2 n + 1)$ сравнений и $\frac{(n)(n-1)}{2}$ перестановок.

- Индукция по n, для массива длины 1 очевидно.

Сложность сортировки вставками

Утверждение

На сортировку массива длины n двоичными вставками требуется не более $n(\log_2 n + 1)$ сравнений и $\frac{(n)(n-1)}{2}$ перестановок.

- Индукция по n, для массива длины 1 очевидно.
- Поиск позиции для вставки в массив длины m требует не более, чем $\log_2 m + 1$ сравнений, в процессе работы требуется вставка в массивы длины $1, \ldots, n-1$.
- Это занимает не более $(\log_2 1 + 1) + (\log_2 2 + 1) + \ldots + (\log_2 (n-1) + 1) \leqslant (n-1)(\log_2 (n-1) + 1)$ сравнений.

Сложность сортировки вставками

Утверждение

На сортировку массива длины n двоичными вставками требуется не более $n(\log_2 n + 1)$ сравнений и $\frac{(n)(n-1)}{2}$ перестановок.

- Индукция по n, для массива длины 1 очевидно.
- Поиск позиции для вставки в массив длины m требует не более, чем $\log_2 m + 1$ сравнений, в процессе работы требуется вставка в массивы длины $1, \ldots, n-1$.
- Это занимает не более $(\log_2 1 + 1) + (\log_2 2 + 1) + \ldots + (\log_2 (n-1) + 1) \leqslant (n-1)(\log_2 (n-1) + 1)$ сравнений.
- Для "сдвига" массива длины m требуется m присваиваний, на i-ой итерации мы двигаем массив длины $i - j$, то есть не длиннее i. Всего нужно не более $1 + \ldots + (n-1) = \frac{n(n-1)}{2}$ присваиваний.

Упорядоченные списки

- В прошлый раз мы написали алгоритм сортировки двоичными вставками.
- Он требует $O(n \log n)$ сравнений и $O(n^2)$ присваиваний, где n — длина массива.
- Цель: ограничиться $O(n \log n)$ присваиваниями.

Упорядоченные списки

- В прошлый раз мы написали алгоритм сортировки двоичными вставками.
- Он требует $O(n \log n)$ сравнений и $O(n^2)$ присваиваний, где n — длина массива.
- Цель: ограничиться $O(n \log n)$ присваиваниями.
- При сортировке выбором или вставками объекты добавляются в список по одному.
- Нельзя ли добавлять по несколько?

Упорядоченные списки

- В прошлый раз мы написали алгоритм сортировки двоичными вставками.
- Он требует $O(n \log n)$ сравнений и $O(n^2)$ присваиваний, где n — длина массива.
- Цель: ограничиться $O(n \log n)$ присваиваниями.
- При сортировке выбором или вставками объекты добавляются в список по одному.
- Нельзя ли добавлять по несколько?
- Попробуем применить слияние упорядоченных списков.

Сортировка слиянием: основная идея

- Можно разделить массив на две половины и их объединить...
- Но половины должны быть уже отсортированы.

Сортировка слиянием: основная идея

- Можно разделить массив на две половины и их объединить...
- Но половины должны быть уже отсортированы.
- Значит, нужно начинать с массивов наименьшей длины.

Сортировка слиянием: основная идея

- Можно разделить массив на две половины и их объединить...
- Но половины должны быть уже отсортированы.
- Значит, нужно начинать с массивов наименьшей длины.
- Вначале объединяем подмассивы длины 1, потом длины 2 (уже отсортированные), потом длины 4...
- Постепенно дойдём до всего массива.

Сортировка слиянием: алгоритм

Вход: Список a чисел длины n.
Выход: Отсортированный в порядке убывания список a.

```
segment_length = 1
while segment_length < n do
    start = 0                                          ▷ позиция начала сегмента
    while start < segment_length do
        middle = start + segment_length                ▷ первый объединяемый сегмент
        end = middle + segment_length                  ▷ второй объединяемый сегмент
        Merge(a[start : middle], a[middle : end])      ▷ слияние сегментов
        start = end
    end while
    segment_length *= 2                                ▷ увеличиваем длину сегмента
end while
    return a
```

Сортировка слиянием: анализ

Теорема

После k шагов алгоритма слияния все подмассивы $a[0 : 2^k]$, $a[2^k : 2^k * 2], a[2^k * 2 : 2^k * 3]$ отсортированы по возрастанию.

Доказательство.

Индукция по k. □

Следствие

После $\lceil \log_2 n \rceil$ итераций весь массив будет отсортирован правильно.

Сортировка слиянием: анализ

Теорема

Сортировка слиянием требует $O(n \log_2 n)$ *сравнений и* $O(n \log_2 n)$ *присваиваний.*

Доказательство.

- Достаточно доказать, что одна итерация требует $O(n)$ операций.

Сортировка слиянием: анализ

Теорема

Сортировка слиянием требует $O(n \log_2 n)$ *сравнений и* $O(n \log_2 n)$ *присваиваний.*

Доказательство.

- Достаточно доказать, что одна итерация требует $O(n)$ операций.
- Нужно оценить суммарные затраты на все слияния.
- Слияние массивов длины n_1 и n_2 требует $O(n_1 + n_2)$ арифметических операций (как сравнений, так и присваиваний).

Сортировка слиянием: анализ

Теорема

Сортировка слиянием требует $O(n \log_2 n)$ *сравнений и* $O(n \log_2 n)$ *присваиваний.*

Доказательство.

- Достаточно доказать, что одна итерация требует $O(n)$ операций.
- Нужно оценить суммарные затраты на все слияния.
- Слияние массивов длины n_1 и n_2 требует $O(n_1 + n_2)$ арифметических операций (как сравнений, так и присваиваний).
- На k-ой итерации объединяются массивы $a[0 : 2^{k-1}]$ и $a[2^{k-1} : 2^k]$, $a[2^k : 2^k + 2^{k-1}]$ и $a[2^k + 2^{k-1} : 2 * 2^k]$ и т. д.

Сортировка слиянием: анализ

Теорема

Сортировка слиянием требует $O(n \log_2 n)$ *сравнений и* $O(n \log_2 n)$ *присваиваний.*

Доказательство.

- Достаточно доказать, что одна итерация требует $O(n)$ операций.
- Нужно оценить суммарные затраты на все слияния.
- Слияние массивов длины n_1 и n_2 требует $O(n_1 + n_2)$ арифметических операций (как сравнений, так и присваиваний).
- На k-ой итерации объединяются массивы $a[0 : 2^{k-1}]$ и $a[2^{k-1} : 2^k]$, $a[2^k : 2^k + 2^{k-1}]$ и $a[2^k + 2^{k-1} : 2 * 2^k]$ и т. д.
- Суммарная длина всех объединяемых массивов n, значит требуется $O(n)$ операций.

Сортировка слиянием: анализ

Теорема

Сортировка слиянием требует $O(n \log_2 n)$ *сравнений и* $O(n \log_2 n)$ *присваиваний.*

Доказательство.

- Достаточно доказать, что одна итерация требует $O(n)$ операций.
- Нужно оценить суммарные затраты на все слияния.
- Слияние массивов длины n_1 и n_2 требует $O(n_1 + n_2)$ арифметических операций (как сравнений, так и присваиваний).
- На k-ой итерации объединяются массивы $a[0 : 2^{k-1}]$ и $a[2^{k-1} : 2^k]$, $a[2^k : 2^k + 2^{k-1}]$ и $a[2^k + 2^{k-1} : 2 * 2^k]$ и т. д.
- Суммарная длина всех объединяемых массивов n, значит требуется $O(n)$ операций.Что нам и требовалось!

Сортировка слиянием: демонстрация

$$[6, 3, 2, 1, 5, 8, 4]$$
$$[6, 3, 2, 1, 5, 8, 4]$$

Сортировка слиянием: демонстрация

$$[6, 3, 2, 1, 5, 8, 4]$$
$$[6, 3, 2, 1, 5, 8, 4]$$
$$[3, 6, 2, 1, 5, 8, 4]$$

Сортировка слиянием: демонстрация

$$[6, 3, 2, 1, 5, 8, 4]$$
$$[6, 3, 2, 1, 5, 8, 4]$$
$$[3, 6, 2, 1, 5, 8, 4]$$
$$[3, 6, 1, 2, 5, 8, 4]$$

Сортировка слиянием: демонстрация

$$[6, 3, 2, 1, 5, 8, 4]$$
$$[6, 3, 2, 1, 5, 8, 4]$$
$$[3, 6, 2, 1, 5, 8, 4]$$
$$[3, 6, 1, 2, 5, 8, 4]$$
$$[3, 6, 1, 2, 5, 8, 4]$$
$$[3, 6, 1, 2, 5, 8, 4]$$

Сортировка слиянием: демонстрация

$$[6, 3, 2, 1, 5, 8, 4]$$
$$[6, 3, 2, 1, 5, 8, 4]$$
$$[3, 6, 2, 1, 5, 8, 4]$$
$$[3, 6, 1, 2, 5, 8, 4]$$
$$[3, 6, 1, 2, 5, 8, 4]$$
$$[3, 6, 1, 2, 5, 8, 4]$$
$$[1, 2, 3, 6, 4, 5, 8]$$

Сортировка слиянием: демонстрация

$$[6, 3, 2, 1, 5, 8, 4]$$
$$[6, 3, 2, 1, 5, 8, 4]$$
$$[3, 6, 2, 1, 5, 8, 4]$$
$$[3, 6, 1, 2, 5, 8, 4]$$
$$[3, 6, 1, 2, 5, 8, 4]$$
$$[3, 6, 1, 2, 5, 8, 4]$$
$$[1, 2, 3, 6, 4, 5, 8]$$
$$[1, 2, 3, 4, 5, 6, 8]$$

Сортировка слиянием: анализ

- Мы построили алгоритм с требуемой сложностью.
- Однако у него есть один недостаток.

Сортировка слиянием: анализ

- Мы построили алгоритм с требуемой сложностью.
- Однако у него есть один недостаток.
- При слиянии массивов длины n_1, n_2 требуется дополнительная память порядка $O(n_1 + n_2)$ для хранения объединённого массива.

Сортировка слиянием: анализ

- Мы построили алгоритм с требуемой сложностью.
- Однако у него есть один недостаток.
- При слиянии массивов длины n_1, n_2 требуется дополнительная память порядка $O(n_1 + n_2)$ для хранения объединённого массива.
- То есть на последней итерации алгоритма потребуется массив такого же размера.
- Это существенно для больших данных.
- Можно ли без этого обойтись?

Сортировка слиянием: анализ

- Мы построили алгоритм с требуемой сложностью.
- Однако у него есть один недостаток.
- При слиянии массивов длины n_1, n_2 требуется дополнительная память порядка $O(n_1 + n_2)$ для хранения объединённого массива.
- То есть на последней итерации алгоритма потребуется массив такого же размера.
- Это существенно для больших данных.
- Можно ли без этого обойтись?
- Можно, но потребуется дополнительная структура данных.