Сириус
IT-Колледж

Лекция 3. Сортировка

Квадратичные сортировки (вставками, выбором минимума). Метод «разделяй и властвуй». Бинарный поиск. Сортировка слиянием: наивная и эффективная реализация. Быстрая сортировка: понятие вероятностного алгоритма, время работы в среднем, простейший алгоритм, inplace-алгоритм. IntroSort. Сравнение сортировок.

Сириус IT-Колледж

Постановка задачи

Алгоритм сортировки — это алгоритм для упорядочивания элементов в списке. В случае, когда элемент в списке имеет несколько полей, поле, служащее критерием порядка, называется ключом сортировки.

Пусть требуется упорядочить N элементов: $R_1, R_2, \ldots, R_n$. Каждый элемент представляет из себя запись R_j, содержащую некоторую информацию и ключ K_j, управляющий процессом сортировки. На множестве ключей определено отношение порядка $«<»$ так, чтобы для любых трёх значений ключей a, b, c выполнялись следующие условия:

- закон трихотомии: либо $a<b$, либо $a>b$, либо $a=b$;
- закон транзитивности: если $a<b$ и $b<c$, то $a<c$.

Данные условия определяют математическое понятие линейного или совершенного упорядочения, а удовлетворяющие им множества поддаются сортировке большинством методов.

Сириус
IT-Колледж

Квадратичные сортировки

Сортировка простыми обменами, сортировка пузырьком

Сириус IT-Колледж

Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется перестановка элементов. Проходы по массиву повторяются ***N-1*** раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован.

Сложность: $O(n^2)$

Сириус
IT-Колледж

Сортировка простыми обменами, сортировка пузырьком

Сортировка вставками

Сириус IT-Колледж

В начальный момент отсортированная последовательность пуста. На каждом шаге алгоритма выбирается один из элементов входных данных и помещается на нужную позицию в уже отсортированной последовательности до тех пор, пока набор входных данных не будет исчерпан.

Данный алгоритм можно ускорить при помощи использования бинарного поиска для нахождения места текущему элементу в отсортированной части. Проблема с долгим сдвигом массива вправо решается при помощи смены указателей.

Сложность: $O(n^2)$

Сириус
IT-Колледж

Сортировка вставками

Сириус
IT-Колледж

Метод “разделяй и властвуй”

Сириус IT-Колледж

Метод “разделяй и властвуй”

«Разделяй и властвуй» в информатике — схема разработки алгоритмов, заключающаяся в рекурсивном разбиении решаемой задачи на две или более подзадачи того же типа, но меньшего размера, и комбинировании их решений для получения ответа к исходной задаче; разбиения выполняются до тех пор, пока все подзадачи не окажутся элементарными.

Применяется в таких алгоритмах как бинарный (двоичный) поиск и сортировка слиянием.

Бинарный (двоичный) поиск

Сириус IT-Колледж

Дана таблица записей $R_1, R_2, \ldots, R_N$, ключи которых расположены в порядке возрастания: $K_1<K_2<\ldots<K_N$; алгоритм используется для поиска в таблице заданного аргумента K.

1. Установить $l \leftarrow 1, u \leftarrow N$.
2. Если $u<l$, алгоритм завершается неудачно; иначе установить $i \leftarrow floor((l+u)/2)$, чтобы i соответствовало примерно середине рассматриваемой части таблицы.
3. Если $K<K_i$, перейти к шагу 4; если $K>K_i$, перейти к шагу 5, если $K=K_i$, алгоритм успешно завершается.
4. Установить $u \leftarrow i-1$ и перейти к шагу 2.
5. Установить $l \leftarrow i+1$ и перейти к шагу 2.

Сложность: $O(\log n)$

Сортировка слиянием

Сириус IT-Колледж

1. Сортируемый массив разбивается на две части примерно одинакового размера. Рекурсивное разбиение задачи на меньшие происходит до тех пор, пока размер массива не достигнет единицы (любой массив длины 1 можно считать упорядоченным).
2. Каждая из получившихся частей сортируется отдельно, например — тем же самым алгоритмом;
3. Два упорядоченных массива половинного размера соединяются в один.
    a. На каждом шаге мы берем меньший из двух первых элементов подмассивов и записываем его в результирующий массив. Счётчики номеров элементов результирующего массива и подмассива, из которого был взят элемент, увеличиваем на 1.
    b. Когда один из подмассивов закончился, мы добавляем все оставшиеся элементы второго подмассива в результирующий массив.

Сириус
IT-Колледж

Сортировка слиянием

Сортировка слиянием

```
def merge(A, B):
    i, j, C = 0, 0, []
    while True:
        if A[i] < B[j]:
            C.append(A[i])
            i += 1
            if i == len(A):
                C.extend(B[j:])
                break

        else:
            C.append(B[j])
            j += 1
            if j == len(B):
                C.extend(A[i:])
                break

    return C
```

Сириус
IT-Колледж

Сириус
IT-Колледж

Сортировка слиянием

Сортировка слиянием

```
def top_down_merge_sort(A):
    if len(A) == 1:
        return A

    d = len(A) // 2
    left = top_down_merge_sort(A[:d])
    right = top_down_merge_sort(A[d:])

    return merge(left, right)
```

Сириус
IT-Колледж

Сортировка слиянием

```
def bottom_up_merge_sort(A):
    k = 1
    while k < len(A):
        for i in range(0, len(A)-k, 2*k):
            A[i:i+2*k] = merge(A[i:i+k], A[i+k:i+2*k])
        k *= 2

    return A
```

Сириус
IT-Колледж

Сортировка слиянием. Галопирование (galloping)

```
def galloping(AB, n, C):
    C[:] = AB[:n]

    # r – указатель на конец результата
    # j – место последней вставки
    # m – длина остатка B
    r, j, m = 0, n, len(AB) - n
    for i in range(n):
        # k – степень двойки
        # l – указатель на 2^k-1 элемент
        k, l = 0, 0
        while l < m and AB[j+l] < C[i]:
            k += 1
            l = 2**k - 1

        if l >= m:
            l = m - 1

        while l >= 0 and AB[j+l] > C[i]:
            l -= 1

        l += 1
        AB[r:r+l], AB[r+l] = AB[j:j+l], C[i]
        r, j, m = r + l + 1, j + l, m - l
```

Сириус
IT-Колледж

Сортировка слиянием. Chunking


```
def chunking(A):
    chunks = []
    a, d = 0, 0
    for b in range(1, len(A)):
        if d == 0:
            d = A[b] - A[a]
            continue

        if (A[b] - A[b-1])*d < 0:
            chunks.append((a, b-1) if d > 0 else (b-1, a))
            a, d = b, 0

    chunks.append((a, b) if d > 0 else (b, a))

    return chunks
```

Сириус
IT-Колледж

Быстрая сортировка

Сириус IT-Колледж

Быстрая сортировка

Быстрая сортировка, сортировка Хоара (quicksort, qsort (по имени в стандартной библиотеке языка Си) — алгоритм сортировки, разработанный английским информатиком Тони Хоаром во время своей работы в МГУ в 1960 году.

Один из самых быстрых известных универсальных алгоритмов сортировки массивов: в среднем ***O(n log n)*** обменов при упорядочении ***n*** элементов.

Быстрая сортировка

Сириус IT-Колледж

1. Массив ***a[l…r]*** разбивается на два (возможно пустых) подмассива ***a[l…q]*** и ***a[q+1…r]***, таких, что каждый элемент ***a[l…q]*** меньше или равен ***a[q]***, который в свою очередь, не превышает любой элемент подмассива ***a[q+1…r]***. Индекс вычисляется в ходе процедуры разбиения.
2. Подмассивы ***a[l…q]*** и ***a[q+1…r]*** сортируются с помощью рекурсивного вызова процедуры быстрой сортировки.
3. Поскольку подмассивы сортируются на месте, для их объединения не требуются никакие действия: весь массив ***a[l…r]*** оказывается отсортированным.

Быстрая сортировка

Сириус
IT-Колледж

```
def partition(array, low, high):
    pivot = array[high]

    i = low - 1

    for j in range(low, high):
        if array[j] <= pivot:

            i = i + 1

            (array[i], array[j]) = (array[j], array[i])

    (array[i + 1], array[high]) = (array[high], array[i + 1])

    return i + 1
```

Быстрая сортировка

```
def quicksort(array, low, high):
    if low < high:

        pi = partition(array, low, high)

        quicksort(array, low, pi - 1)

        quicksort(array, pi + 1, high)
```

Сириус
IT-Колледж

IntroSort. Интроспективная сортировка

Сириус IT-Колледж

Алгоритм сортировки, предложенный Дэвидом Мюссером в 1997 году. Он использует быструю сортировку и переключается на пирамидальную сортировку, когда глубина рекурсии превысит некоторый заранее установленный уровень (например, логарифм от числа сортируемых элементов).

Сириус IT-Колледж

Сравнение сортировок

Сортировка	Время в худшем случае	Время в среднем случае	Затраты памяти в лучшем случае
Пузырьком	$O(n^2)$	$O(n^2)$	$O(1)$
Вставками	$O(n^2)$	$O(n^2)$	$O(1)$
Слиянием	$O(n \log n)$	$O(n \log n)$	$O(n)$
Быстрая	$O(n^2)$	$O(n \log n)$	$O(1)$
IntroSort	$O(n \log n)$	$O(n \log n)$	$O(n)$