ПРОГРАММИРОВАНИЕ

Семинар 4.
Сортировки

Новосибирский государственный университет, 2019

Сортировка

Процесс перестановки объектов заданной совокупности в определённом порядке (возрастающем или убывающем).

Цель сортировки

Облегчение последующего поиска элементов в отсортированном множестве (например, возможность применения бинарного поиска).

Виды сортировки

- внутренняя (массивы);
- внешняя (файлы).

Задача сортировки

Упорядочить N объектов $a_1, a_2, \ldots, a_N$, т. е. переставить их в такой последовательности $a_{p1}, a_{p2}, \ldots, a_{pN}$, чтобы их ключи расположились в неубывающем порядке $k_{p1} \le k_{p2} \le \ldots \le k_{pN}$.

Свойство устойчивости

Сортировка называется **устойчивой**, если записи с одинаковыми ключами остаются в прежнем порядке:

$$k_{pi} = k_{pj} \,\&\, i < j \Rightarrow p_i < p_j.$$

При устойчивой сортировке относительный порядок элементов не меняется.

Сортировка массивов

Требование: экономное использование памяти, т. е. не используются дополнительные массивы, а перестановка элементов производится внутри сортируемого массива.

Меры эффективности:

- **количество сравнения ключей *C(N)*;**
- **количество перестановок *M(N)*.**

Методы сортировки массивов

- включение;
- выбор;
- обмен;
- подсчёт;
- разделение;
- слияние.

Сортировка простыми вставками

Пусть $a_i, a_{i+1}, \ldots, a_N$ — неотсортированная последовательность (входная),

$a_1, a_2, \ldots, a_{i-1}$ — отсортированная (готовая).

На каждом i-м шаге i-ый элемент входной последовательности вставляется в подходящее место готовой.

Алгоритм

Условно разделить массив *A* на входную и готовую части. К входной части сначала относится только *A[0]*.

Цикл по *i* от *1* до *N - 1* с шагом *1*

x = A[i];

j = i - 1;

Пока *j ≥ 0* и *A[j] > x* выполнять

A[j + 1] = A[j];

j = j - 1;

Конец Пока

A[j + 1] = x;

Конец цикла

Пример

38 90 5 10 17 3 9 1

38 90 5 10 17 3 9 1

5 38 90 10 17 3 9 1

5 10 38 90 17 3 9 1

5 10 17 38 90 3 9 1

3 5 10 17 38 90 9 1

3 5 9 10 17 38 90 1

1 3 5 9 10 17 38 90

Анализ алгоритма

$max(C(i)) = i - 1$

Если перестановки равновероятны,
то в среднем $C(i) = i / 2$.

$max(M(i)) = C(i) + 2$

$C_{max} = 1 + 2 + \ldots + N - 1 = N (N - 1) / 2$

$C_{min} = N - 1$

$M_{min} = 2 (N - 1)$

$M_{max} = N (N - 1) / 2 + 2 (N - 1) \approx N (N + 3) / 2$

Анализ алгоритма

Наилучшие оценки — уже упорядоченный массив, наихудшие — обратный порядок.

Сортировка устойчива.

Сортировка бинарными включениями

При поиске подходящего места вставки в методе простой вставки использовать метод бинарного поиска

$$C(i) \approx \log_2 N$$

$$C(N) \approx N \log_2 N$$

$M(N)$ не изменится

Сортировка простым выбором

Идея многократного выбора.

На каждом *i*-м шаге выбирается наименьший элемент входной последовательности и меняется местами с a_i. Далее перемещаем его в готовую последовательность.

Процесс продолжаем до тех пор, пока во входной последовательности не останется один элемент.

Алгоритм

Условно разделить массив *A* на входную и готовую части.

Сначала весь массив — входная часть.

Цикл по *i* от *0* до *N - 2* с шагом *1*

r = i;

Цикл по *j* от *i + 1* до *N - 1* с шагом *1*

если *A[j] < A[r]*, то *r = j* и выйти из цикла;

Конец цикла

если *i ≠ r*, то обменять *A[i]* с *A[r]*.

Конец цикла

Пример

38 90 5 10 17 3 9 1

1 38 90 5 10 17 3 9

1 3 38 90 5 10 17 9

1 3 5 38 90 10 17 9

1 3 5 9 38 90 10 17

1 3 5 9 10 38 90 17

1 3 5 9 10 17 38 90

1 3 5 9 10 17 38 90

Анализ алгоритма

$C(i)$ не зависит от начального порядка элементов.

$$C(N) = N - 1 + N - 2 + \ldots + 1 = N(N - 1) / 2$$

$$M_{max} = N - 1$$

Сортировка простым обменом (метод пузырька)

Идея — сравнение и обмен соседних элементов, начиная с конца массива.

На каждом *i*-м шаге сравниваем *i*-ый и *(i - 1)*-ый элементы, меняя их местами, если они не упорядочены.

Повторяем процесс, начиная с конца до 2-го элемента и т. д.

Алгоритм

Цикл по *i* от *1* до *N - 1* с шагом *1*

Цикл по *j* от *N - 1* до *i* с шагом *1*

если *A[j] < A[j - 1]*, то обменять *A[j]* с *A[j - 1]*.

Конец цикла

Конец цикла

Пример

38 90 5 10 17 3 9 1
38 90 5 10 17 3 1 9
38 90 5 10 17 1 3 9
38 90 5 10 1 17 3 9
38 90 5 1 10 17 3 9
38 90 1 5 10 17 3 9
38 1 90 5 10 17 3 9
1 38 90 5 10 17 3 9
1 38 90 5 10 17 3 9
1 38 90 5 10 3 17 9
1 38 90 5 3 10 17 9
1 38 90 3 5 10 17 9
1 38 3 90 5 10 17 9
1 3 38 90 5 10 17 9
1 3 38 90 5 10 9 17
1 3 38 90 5 9 10 17

Анализ алгоритма

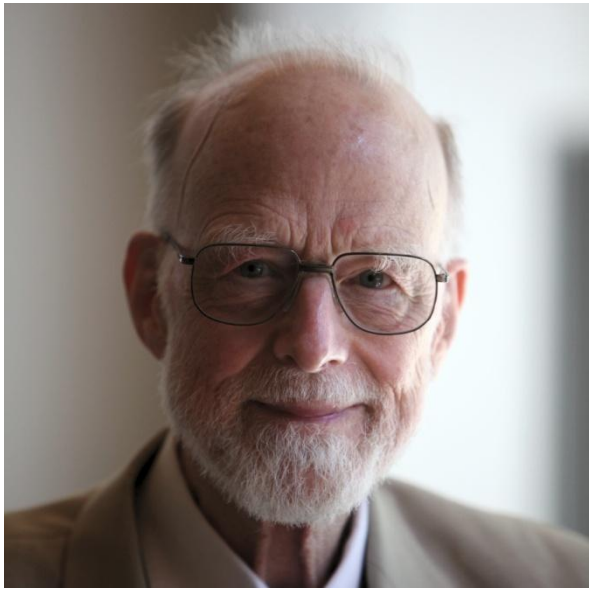
$C(i) = N - i$

$C(N) = N - 1 + N - 2 + \ldots + 1 = N(N - 1) / 2$

$M_{min} = 0$

$M_{max} = C(N)$ — обратно упорядоченный массив.

Сортировка с разделением (быстрая сортировка)



Чарльз Энтони Ричард Хоар (1934)
Charles Antony Richard Hoare

Сортировка с разделением (быстрая сортировка)

Идея — обмен несоседних элементов и сведение к сортировки меньших частей.

На каждом *i*-м шаге существует индекс *m*, такой что:

$$\forall i, j \; 0 \le i \le m \;\&\; m < j < N - 1 \; a_i \le a_j.$$

Назовём *m* медианой. Далее сортируем отдельно левую и правую части любым методом обмена.

Алгоритм

Процедура СортировкаРазделением(l, r)

Привести последовательность $a_l, \ldots, a_r$ к условию выше и определить медиану m;

Части длины 0 или 1 не сортируем;

Если $l < m$, то СортировкаРазделением(l, m);

Если $m + 1 < r$, то СортировкаРазделением($m + 1$, r);

Конец процедуры

Алгоритм

Процесс разделения:

Пока $i < j$

$i = l; j = r;$

Пока $a_i < x$ $i = i + 1;$

Пока $x < a_j$ $j = j - 1;$

Если $i < j$, **то обменять** a_i **с** a_j;

$i = i + 1;$

$j = j - 1;$

Конец пока

Пример

38 90 5 10 17 3 9 1

1 90 5 10 17 3 9 38

1 9 5 10 17 3 90 38

1 9 5 3 10 17 90 38

1 9 5 3 10 17 90 38

1 9 5 3

1 3 9 5

1 3 5 9

17 90 38

1 3 5 9 17 38 90

Сортировка подсчётом

Заводится дополнительный массив *B*:

B[j] содержит количество вхождений числа *j* в *A*.