**Лекция 5. Улучшенные алгоритмы внутренней сортировки**

**Улучшение алгоритма пузырьковой сортировки**

**Кролики и черепахи**. Позиция элементов, подлежащих сортировке играет большую роль в вопросе производительности данного алгоритма. Большие элементы в начале список не представляют проблемы, поскольку они достаточно быстро перемещаются на свои места. Однако, малые элементы в конце списка перемещаются на его начало очень медленно. Это привело к тому, что оба типа элементов были названы кроликами и черепахами, соответственно.

С целью повышения быстродействия алгоритма, в свое время было осуществлено немало усилий для уменьшения количества "черепах". Сортировка перемешиванием является сравнительно плохим, однако, все еще в своем худшем случае имеет сложность O (n2). Сортировка расческой сначала сравнивает крупные элементы друг с другом, а уже потом постепенно переходит к все меньшим и меньшим. Его среднестатистическая скорость примерно равна таковой в алгоритме Быстрая сортировка.

**Сортировка смешиванием** - одна из разновидностей алгоритма сортировки пузырьком. Отличается от сортировки пузырьком тем, что сортировка происходит в обоих направлениях, меняя направление при каждом проходе. Данный алгоритм лишь немного сложнее сортировка пузырьком, однако, решает так называемую проблему "черепах".

**быстродействие**

Эффективность алгоритма равна O (n2) одновременно для среднего статистического и наихудшего случая, однако, она стремится к O (n) если список уже не плохо отсортирован, например, если каждый элемент находится в позиции, которая отличается от конечной больше, чем на k ( k ≥ 1), то его быстродействие равно O (k \* n).

**Отличия от сортировки пузырьком**

Сортировка смешиванием мало чем отличается от сортировки пузырьком. Единственное его отличие в том, что вместо многократного прохождения через список снизу вверх, он проходит по очереди снизу вверх и сверху вниз. Он может достигать несколько более высокой эффективности, чем алгоритм сортировки пузырьком. Причиной этому является то, что алгоритм сортировки пузырьком проходит по списку только в одном направлении, а потому за одну итерацию элементы списка можно переместить только на один шаг.

Например, для того, чтобы отсортировать список (2,3,4,5,1), алгоритма сортировки смешиванием достаточно одного прохода, в то время, как алгоритма сортировки пузырьком понадобится четыре прохода. Однако, один проход сортировки смешиванием следует считать за два прохода сортировки пузырьком. Обычно, сортировка перемешиванием вдвое быстрее сортировки пузырьком.

Другой возможной оптимизацией является запоминание предыдущих перестановок. В следующей итерации, перестановки не будут повторяться, поэтому алгоритм будет короче проходы по списку.

Пример на Паскале:

procedure bubble;

var i, j, t: byte;

begin

for i: = 2 to N do

for j = N down to i do

if x [i-1]> x [j] then

begin t = x [j-1]; x [j-1] = x [j]; x [j]: = t; end;

end;

end;

**Сортировка расческой** - упрощенный алгоритм сортировки, разработанный Влодеком Добошевичем (Wlodek Dobosiewicz) в 1980 году, и позже заново исследованы и популяризированных Стефаном Лакеем (Stephen Lacey) и Ричардом Боксом (Richard Box), которые написали о нем в журнале Byte Magazine в апреле 1991 г. . Сортировка расческой является улучшением алгоритма сортировки пузырьком, и конкурирует в быстродействии с алгоритмом Быстрая сортировка. Основная его идея заключается в том, чтобы устранить так называемых "черепах", или малых значений ближе к концу списка, поскольку в сортировки пузырьком они сильно замедляют процесс сортировки. (Кролики и большие сортировки в начале списка в сортировке пузырьком не являются проблемы).

В сортировке пузырьком, когда два элемента сравниваются, они всегда имеют разрыв (отставной друг от друга) равную 1. Основная идея сортировки расческой заключается в том, что этот разрыв может быть больше единицы. (Алгоритм Сортировка Шелла также основанное на идее, однако, он является модификацией алгоритма сортировки вставками, а не сортировки пузырьком).

Разрыв начинается со значения, что равно длине списка, разделенного на фактор уменьшения (обычно, 1.3, см. Ниже), и список сортируется с учетом этого значения (при необходимости оно заокруглуеться к целому). Затем разрыв снова делится на фактор разрыва, и список продолжает сортироваться с новым значением, процесс вродовжуеться до тех пор, пока разрыв равен 1. Далее список сортируется с разрывом равным 1 до тех пор, пока не будет полностью отсортирован. Таким образом, финальный этап сортировки аналогичен таковому же в сортировке пузырьком, однако, к этому "черепах" устраняется.

**фактор уменьшения**

Фактор уменьшения производит большой эффект на скорость алгоритма слортування расческой. В оригинальной статье, автор предлагает значение 1.3 после многих экспериментов с другими значениями.

Текст описывает процесс совершенствования алгоритма используя значение в качестве фактора уменьшения. Она также вместо пример использования алгоритма на псевдокоде.

Примеры реализации на разных языках программирования

псевдокод

function combsort11 (array input)

gap = input.size

loop until gap <= 1 and swaps = 0

if gap> 1

gap = gap / 1.3

if gap = 10 or gap = 9

gap = 11

end if

end if

i = 0

swaps = 0

loop until i + gap <= input.size

if input [i]> input [i + gap]

swap (input [i], input [i + gap])

swaps = 1

end if

i = i + 1

end loop

end loop

end function

Код на C ++

void sort (data \* array, dword size)

{

if (! array ||! size)

return;

dword jump = size;

bool swapped = true;

while (jump> 1 || swapped)

{

if (jump> 1)

jump = (dword) (jump / 1.25)

swapped = false;

for (dword i = 0; i + jump <size; i + = jump)

if (array [i]> array [i + jump])

swap (array, i, i + jump), swapped = true;

}

}

**Улучшенные методы сортировки**

**Сортировка Шелла**

Сортировка Шелла - это алгоритм сортировки, является обобщением сортировки вставкой.

Алгоритм базируется на двух тезисах:

- Сортировка включением эффективное для почти упорядоченных массивов.

- Сортировка включением неэффективно, потому что перемещает элемент только на одну позицию за раз.

Поэтому сортировка Шелла выполняет несколько впорядкувань включением, каждый раз сравнивая и переставляя элементы, находящиеся на разной

расстоянии друг от друга.

Сортировка Шелла не является стабильным.

**История**

Сортировка Шелла названа в честь автора-Дональда Шелла, который опубликовал этот алгоритм в 1959 [1] году. В некоторых поздних изданиях алгоритм называют сортировкой Шелла-Мацнера, по имени Нортона Мацнера. Но сам Мацнер утверждал: «Мне не пришлось ничего делать с этим алгоритмом, и ч мое имя не имеет связываться с ним». [2]

**идея алгоритма**

В начале избираются m-элементов:, причем,.

Затем выполняется m упорядования методом включения, сначала для элементов, стоящих за, затем для элементов из и т. Д. До.

Эффективность достигается тем, что каждое последующее благоустройство требует меньшего количества перестановок, поскольку некоторые элементы уже стали на свои места.

псевдокод

Сам алгоритм не зависит от выбора m и d, поэтому будем считать, что они заданы.

1. for to

2. do for to

3. do

4.

5. while и

6. do

7.

8.

**корректность алгоритма**

Поскольку то на последнем шаге выполняется обычное упорядочение включением всего массива, а значит конечный массив будет упорядочен.

**время работы**

Время работы зависит от выбора значений элементов массива d. Существует несколько подходов выбора этих значений:

• При выборе время работы алгоритма, в худшем случае, есть.

• Если d - впорядкованний по убыванию набор чисел вида, то время работы есть.

• Если d - впорядкованний по убыванию набор чисел вида, то время работы есть.

пример работы

Проиллюстрируем работу алгоритма на входном массиве A = (5,16,1,32,44,3,16,7), d = (5,3,1).

1. Массив после благоустройства с шагом в 5: (3,16,1,32,44,5,16,7) - сделано 1 обмен.

2. Массив после благоустройства с шагом 3: (3,7,1,16,16,5,32,44) - сделано 3 обмене.

3. Массив после благоустройства с шагом 1: (1,3,5,7,16,16,32,44) - сделано 5 обменов.

Итак, весь массив отсортированы по 8 операций обмена.