OTUS

Онлайн образование

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Алгоритмы поиска и сортировки

Плисенко Ольга

Преподаватель курса «Программист С»

Эл. почта plolga.otus@yandex.ru
Telegram: @PlisenkoOlga

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в Telegram @OTUS C-2023-07

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое
на активность

Пишем в чат

Говорим голосом

Документ

Ответьте себе или
задайте вопрос

Маршрут вебинара

- Нотация «О - большое».
- Виды сортировок
- Двоичный поиск
- Рефлексия

Цели вебинара

После занятия вы сможете

1. Определять класс эффективности алгоритмов
2. Использовать оптимальный алгоритм сортировки или поиск
3. Описать алгоритмы базовых методов сортировки

Нотация «O - большое»

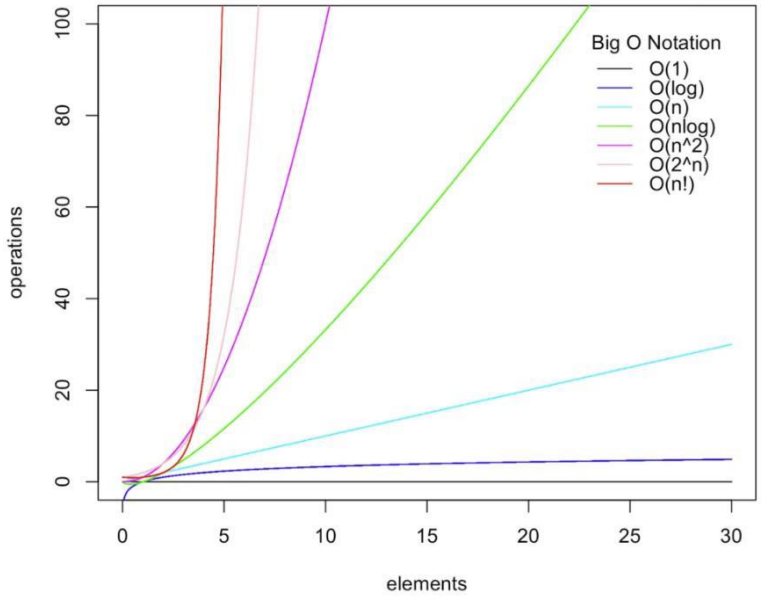
- **«O – большое»** - определяет рост времени выполнения алгоритма с увеличением количества обрабатываемых элементов в худшем случае. Обозначается *O(f)*. Этот класс состоит из функций, растущих не быстрее *f*. Функция *f* образует верхнюю границу для класса *O(f)*

Литература:

1. Седжвик Р. «Фундаментальные алгоритмы на С. Части 1- 5».
2. Дж. Макконнелл. Основы современных алгоритмов. 2004
3. Дейтел П., Дейтел Х. «С для программистов с введением в С11».
4. Адитъя Бхаргава «Грокаем алгоритмы».

Основные функции «O – большое»:

O(1)	Постоянное время – выполнение НЕ зависит от N (индексы, хеш-функции)
O(log N)	Логарифмическое время - разбиение крупной задачи на более мелкие, с уменьшением на каждом шаге размера задачи (бинарный поиск)
O(N)	Линейное время –проход всех элементов за 1 цикл (простой поиск)
O(N * log N)	Разбиение задачи на мелкие подзадачи, их решение и объединение этих решений (быстрая сортировка)
$O(N^2)$	Квадратичное время –обработка алгоритмов в циклах двойного уровня вложения (медленные алгоритмы сортировки)
$O(2^N)$	Экспоненциальное время – редко применяются, возникают при решении задачи «в лоб»
O(N!)	Факториальное время – очень медленные алгоритмы (задача коммивояжера)



Время решения задач

Операций в секунду	Размер задачи 1 миллион			Размер задачи 1 миллиард		
	N	$N \lg N$	N^2	N	$N \lg N$	N^2
10^6	секунд	секунд	недель	часов	часов	никогда
10^9	мгновенно	мгновенно	часов	секунд	секунд	десятилетий
10^{12}	мгновенно	мгновенно	секунд	мгновенно	мгновенно	недель

* Седжвик Р.: Фундаментальные алгоритмы на С. Часть 1 «Анализ», Глава 2 «Принципы анализа алгоритмов».

Виды сортировок:

Сортировка – расстановка элементов в заданном порядке

Простые, но НЕ эффективные – сложность O(N^2)	Сложные, но более эффективные – сложность O(N * log N)
Сортировка вставками (Insertion Sort)	Быстрая сортировка (Quick Sort)
Сортировка выбором (Selection Sort)	Сортировка слиянием
Пузырьковая сортировка (Bubble Sort)	Сортировка кучей (Heap Sort)
Шейкерная сортировка	Сортировка бинарным деревом (Tree Sort)

Основные виды сортировок (https://academy.yandex.ru/posts/osnovnye-vidy-sortirovok-i-primery-ikh-realizatsii)

Описание алгоритмов сортировки (https://habr.com/ru/post/335920/)

Сортировка на примере танцев (https://habr.com/ru/post/117200/)

Сортировка вставками

Массив постепенно перебирается слева направо.

При этом каждый последующий элемент размещается так, чтобы он оказался между ближайшими элементами с меньшим и большим значениями.

- Худшее время: O(N2)
- Среднее время: O(N2)
- Лучшее время: O(N)

Сортировка выбором

Нужно найти в массиве максимум (или минимум) и поменять его со значением первого неотсортированного элемента. Этот шаг нужно повторять до тех пор, пока в массиве не закончатся неотсортированные под массивы.

- Худшее время: O(N2)
- Среднее время: O(N2)
- Лучшее время: O(N2)

Сортировка пузырьком

Последовательное перемещение по массиву, сравнение значений соседних элементов и обмен их местами, если предыдущее оказывается больше последующего.

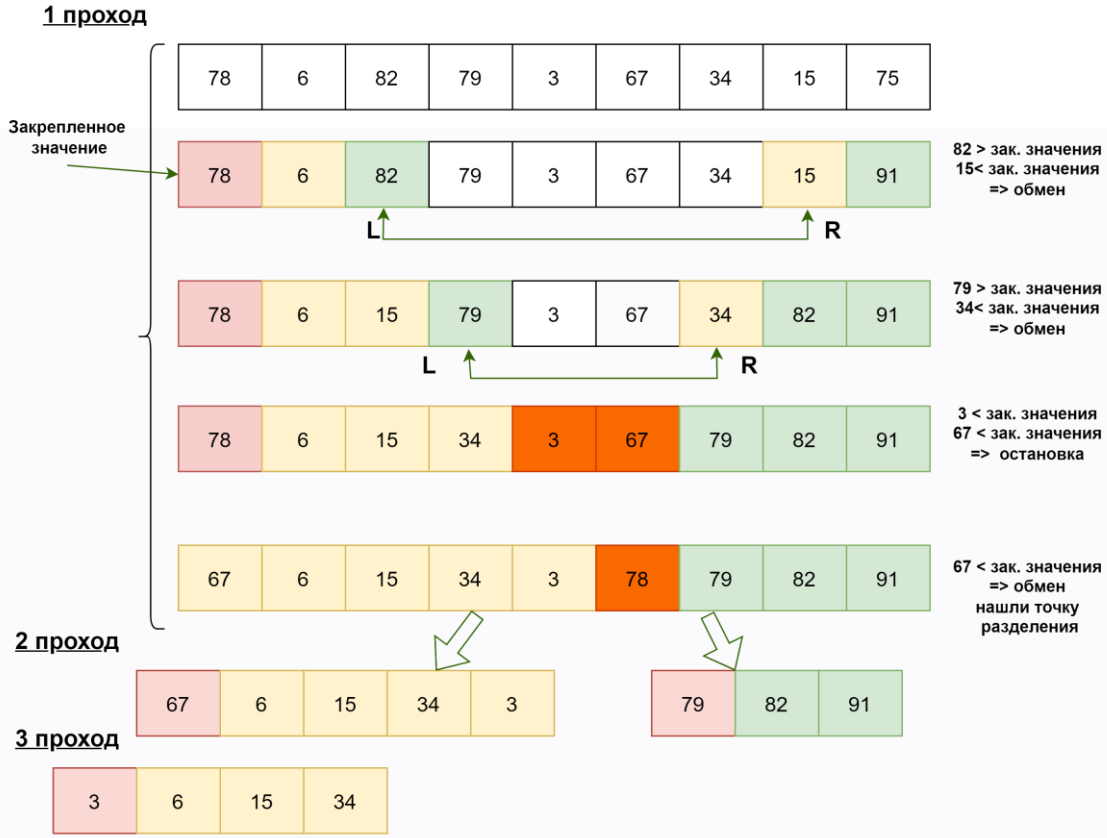
- Худшее время: O(N2)
- Среднее время: O(N2)
- Лучшее время: O(N)

Шейкерная сортировка

Аналогична пузырьковой, только движение происходит в двух направлениях.

- Худшее время: O(N2)
- Среднее время: O(N2)
- Лучшее время: O(N)

Быстрая сортировка



Быстрая сортировка

<stdlib.h>

qsort(a, n, size, compare);

a –массив, n –количество элементов массива, size –размер элементов массива

compare –функция сравнения элементов массива:

int compare(const void * val1,const void * val2);

вх. параметры:

-указатели на элементы массива, типа void* (должны быть приведены к определенным типам данных)

вых. параметр:

- отрицательное значение, если val1 < val2;
- нуль, если val1 ==val2;
- положительное, если val1 > val2 .

```c
#include <stdio.h>
#include <stdlib.h>
int compare_int(const void *a, const void *b) {
    return *(int*)a - *(int*)b;
}
int compare_int_desc(const void *a, const void *b) {
    return compare_int(b, a);
}
int compare_str(const void *a, const void *b) {
    char** pa = (char**)a;
    char** pb = (char**)b;
    return strcmp(*pa, *pb);
}
```

```c
int main()
{
    int n;
    scanf("%d", &n);
    int *a = (int*)malloc(sizeof(int)*n);
    //....... заполнение массива
    qsort(a, n, sizeof(int), compare_int_desc);
```

Сортировка слиянием

Массив разбивается на две (почти) равные части, каждая часть сортируется, а затем объединяется в один общий.

Массив рекурсивно разбивается на части, пока не получится массив из одного элемента.

Худшее время: O(N * log N)

Пирамидальная сортировка (кучей)

1. Построение пирамиды.
2. Сортировка на построенной пирамиде.

- Худшее время: O(N * logN)
- Среднее время: O(N * logN)
- Лучшее время: O(N)

Пирамидальная сортировка (https://habr.com/ru/company/otus/blog/460087/)

Сортировка с помощью бинарного дерева

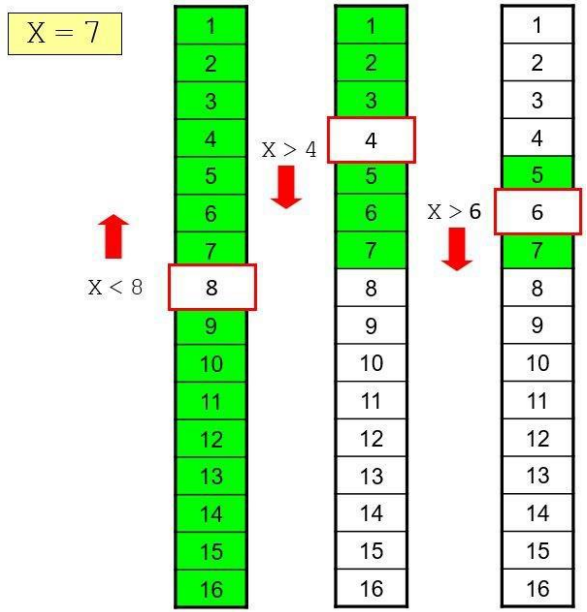
1. Построение двоичного дерева.
2. Сборка массива путем обхода узлов в нужном порядке следования ключей.

- Худшее время: O(N * logN)
- Среднее время: O(N * logN)
- Лучшее время: O(N * logN)

Сортировка двоичным деревом (https://program.rin.ru/razdel/html/782.html)

Двоичный поиск

1. Выбрать средний элемент A[c]и сравнить с X.
2. Если X=A[c], нашли (выход).
3. Если X<A[c], искать дальше в первой половине.
4. Если X>A[c], искать дальше во второй половине.



Сравнение методов поиска

	Линейный	Двоичный
подготовка	нет	**отсортировать**
	число шагов	
N = 2	**2**	**2**
N = 16	**16**	**5**
N = 1024	**1024**	**11**
N= 1048576	**1048576**	**21**
N	**≤ N**	**≤ $\log_2 N+1$**

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Рефлексия

Цели вебинара

Проверка достижения целей

1. Написать код, использующий динамическое распределение памяти для разных структур даннных
2. Разрабатывать программы, эффективно использующие оперативную память
3. Сформулировать основные этапы динамического распределения памяти в языке С

Рефлексия

Насколько тема была для вас сложной?

Как будете применять на практике то,
что узнали на вебинаре?

Заполните, пожалуйста, опрос о занятии по ссылке в чате

Спасибо за внимание!

Приходите на следующие вебинары

Плисенко Ольга

Эл. почта plolga.otus@yandex.ru
Telegram: @PlisenkoOlga