**Приложение №3**

|  |  |
| --- | --- |
| **Дата** | 11.12.2023 |
| **Тема** | Анализ нижней границы алгоритма. Изучение эффективности алгоритма при работе с переменным количеством данных. Симуляция среднего случая алгоритма. Симуляция наихудшего случая алгоритма. Основные понятия трехленточной сортировки. |
| **Задания** | 1. Проанализировать нижнюю границу алгоритма 2. Проанализировать эффективность алгоритма. 3. Просимулировать средний случай алгоритма. 4. Просимулировать наихудший случай алгоритма. 5. Проанализировать основные понятия трехлеенточной сортировки. 6. Выполнение индивидуального задания варианта 10 (1 – 2). |

**1. Проанализировать нижнюю границу алгоритма:**

Алгоритм является оптимальным, если любой другой алгоритм, решающий данную задачу, работает не быстрее данного. Как узнать оптимальность алгоритма? Для этого мы должны знать минимальное количество операций, необходимое для решения поставленной задачи, которое называется нижней границей. Но для этого нам нужно изучать именно задачу, а не конкретный алгоритм.

Как пример, рассмотрим анализ процесса сортировки списка из 3 чисел, воспользовавшись бинарным деревом. Такое дерево называется деревом решений.

Самый длинный путь в данном дереве соответствует наихудшему случаю и наоборот, кратчайший — наилучшему. Средний случай описывается частным от деления числа ребер в дереве на число листов.

Для перестановки 2-х элементов мы имеем 1лист, для N элементов, по правилам комбинаторики, N! листов. Число листов на уровне K равно 2k-1, поэтому в нашем дереве решений буде L -уровней, где L — наименьшее целое число, для которого N! ≤ 2L-1.

**2. Проанализировать эффективность алгоритма.**

Эффективность алгоритма — это свойство [алгоритма](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC), которое связано с [вычислительными ресурсами](https://ru.wikipedia.org/w/index.php?title=%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81&action=edit&redlink=1), используемыми алгоритмом. Алгоритм должен быть [проанализирован](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%BE%D0%B2) с целью определения необходимых алгоритму ресурсов. Эффективность алгоритма можно рассматривать как аналог производственной производительности повторяющихся или непрерывных процессов.

Для достижения максимальной эффективности мы желаем уменьшить использование ресурсов. Однако различные ресурсы (такие как время и память) нельзя сравнить напрямую, так что какой из двух алгоритмов считать более эффективным часто зависит от того, какой фактор более важен, например, требование высокой скорости, минимального использования памяти или другой меры эффективности.

Сравнение алгоритмов между собой - основная тема компьютерных наук. Измерение эффективности алгоритмов чрезвычайно важно, поскольку выбор алгоритма сильно влияет на работу приложения. Эффективность алгоритмов, положенных в основу программы, определяет ее успех, будь то текстовый процессор, кассовый аппарат, банкомат, видеоигра или что-нибудь еще.

Допустим, два алгоритма решают одну и ту же задачу, например осуществляют поиск данных. Как их сравнить между собой и решить, какой из них лучше? Выбирая алгоритм, оцените его эффективность.

Сравнение эффективности алгоритмов должно быть сосредоточено на их существенных различиях. Анализ алгоритмов (analysis с algorithms) – это, область компьютерных наук изучающая способы сравнения эффективности разных методов решения задач. В этом определении использован термин "метод решения задачи", а не "программа". Следует подчеркнуть, что анализ алгоритмов, как правило, исследует существенные различия эффективности, которые обусловлены собственно методами решения задач, а не остроумными программистскими трюками. Изощренные приемы кодирования, позволяющие снизить стоимость вычислений, чаще всего снижают читабельность программы, тем самым повышая затраты на ее сопровождение и модификацию.

**3. Просимулировать средний случай алгоритма.**

Например, дан алгоритм последовательного поиска элемента K в массиве размера N. Тогда средним случаем алгоритма будет являться: K обнаружен после (N+1)/2 сравнений.

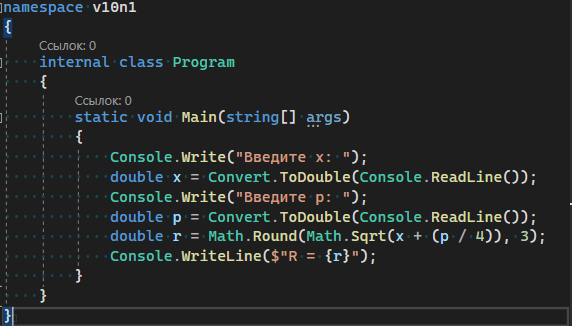
**4. Просимулировать наихудший случай алгоритма.**

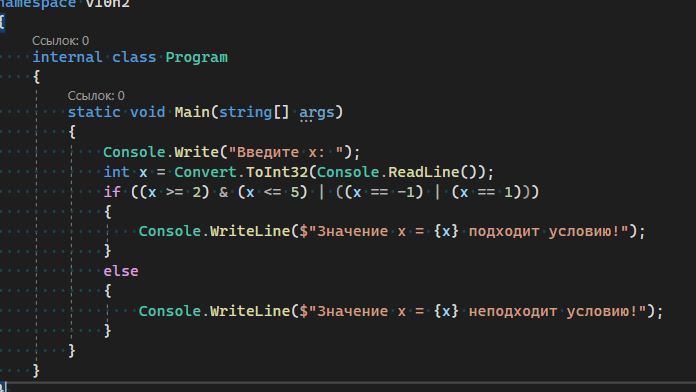
Например, дан алгоритм последовательного поиска элемента K в массиве размера N. Тогда наихудшим случаем алгоритма будет являться: K обнаружен на n-й позиции.

**5. Проанализировать основные понятия трехленточной сортировки.**

Трехленточная сортировка последовательностей, представляет из себя простое прямое слияние последовательностей, для осуществления которого требуется три ленты. Суть метода состоит в следующем - исходная последовательность разбивается на две части, затем эти части сливаются, причем одиночные элементы образуют упорядоченные пары. Полученная последовательность вновь разбивается на две части и сливаются, при этом упорядоченные пары образуют упорядоченные четверки. И так далее - четверки сливаются в восьмерки и т.п., каждый раз удваивая длину слитых последовательностей до тех пор, пока не будет упорядочена вся последовательность целиком.

**6. Выполнение индивидуального задания (1-2)**





**Выводы:** Проанализировал нижнюю границу алгоритма. Проанализировал эффективность алгоритма. Просимулировал средний случай алгоритма. Просимулировал наихудший случай алгоритма. Проанализировать основные понятия трехленточной сортировки. Выполнил индивидуально задание варианта 10 (1-2).

Студент: / Ульянов Никита Анатольевич