Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по Лабораторной работе №3

по курсу «Анализ Алгоритмов»

на тему: «Алгоритмы сортировки»

Студент группы ИУ7-51Б _____ (Подпись, дата) Шубенина Д. В. (Фамилия И.О.)

Преподаватель _____ (Подпись, дата) Волкова Л. Л. (Фамилия И.О.)

Преподаватель _____ (Подпись, дата) Строганов Ю. В. (Фамилия И.О.)

Москва — 2023 г.

Содержание

Введение

Сортировка является одной из самых важных операций над данными, которая имеет применения в различных задачах.

Целью данной лабораторной работы является изучение, реализация и исследование алгоритмов сортировки.

Необходимо выполнить следующие задачи:

1) изучить следующие алгоритмы сортировки:
   - блинная;
   - быстрая;
   - гномья;
2) реализовать данные алгоритмы;
3) выполнить сравнительный анализ алгоритмов по затрачиваемым ресурсам (времени, памяти);
4) описать и обосновать полученные результаты в отчете.

1 Аналитическая часть

В данном разделе будут рассмотрены алгоритмы блинной, быстрой и гномьей сортировки.

1.1 Блинная сортировка

Единственная операция, допустимая в алгоритме — переворот элементов последовательности до какого-либо индекса. В отличие от традиционных алгоритмов, в которых минимизируют количество сравнений, в блинной сортировке требуется сделать как можно меньше переворотов.

1.2 Быстрая сортировка

Процесс сортировки массива $A[p..r]$ алгоритмом быстрой сортировки состоит из трех этапов:

1) массив $A[p..r]$ разбивается на два (возможно, пустых) подмассива $A[p..q-1]$ и $A[q+1..r]$, таких, что каждый элемент $A[p..q-1]$ меньше или равен $A[q]$, который, в свою очередь, не превышает любой элемент подмассива $A[q+1..r]$; индекс q вычисляется в ходе процедуры разбиения;

2) подмассивы $A[p..q-1]$ и $A[q+1..r]$ сортируются с помощью рекурсивного вызова процедуры быстрой сортировки;

3) поскольку подмассивы сортируются на месте, весь массив $A[p..r]$ уже оказывается отсортированным — никаких дополнительных операций объединений подмассивов не требуется [1].

Элемент $A[q]$ также называется опорным элементом. В разных реализациях он может выбираться по-разному, например опорным элементом может быть выбран первый элемент массива, средний, случайный или медиана первого, среднего и последних элементов.

1.3 Гномья сортировка

Алгоритм ищет первую пару соседних элементов, которые находятся в неправильном порядке, и затем меняет их местами. Он пользуется тем фактом, что после обмена элементов может появиться новая пара, нарушающая порядок, только до или после переставленных элементов. Алгоритм не проверяет, отсортированы ли элементы после текущей позиции, поэтому необходимо лишь проверить порядок до переставленных элементов.

Вывод

В данном разделе были рассмотрены алгоритмы блинной, быстрой и гномьей сортировок.

2 Конструкторская часть

В данном разделе будут приведены псевдокоды алгоритмов блинной, быстрой и гномьей сортировок, оценки их трудоемкостей.

2.1 Требования к ПО

К программе предъявлен ряд требований:

— на вход программе подается два массива;

— результатом сортировки является массив, выводимый на экран;

— программа должна позволять производить измерения процессорного времени, затрачиваемого на выполнение реализуемых алгоритмов.

2.2 Разработка алгоритмов

Algorithm 1 Вспомогательная функция Flip

```
1:  procedure FLIP(arr, maxNumPos, n)
2:      i ← maxNumPos
3:      while i < n do
4:          swap ← arr[i]
5:          arr[i] ← arr[n]
6:          arr[n] ← swap
7:          n ← n − 1
8:          i ← i + 1
9:      end while
10: end procedure
```

Algorithm 2 Алгоритм блинной сортировки

Вход: ссылка на массив arr размером n

Выход: отсортированный массив arr

```
1:  procedure PancakeSort(arr, n)
2:      if n ≥ 2 then
3:          i ← n
4:          while i > 1 do
5:              maxNumPos ← 0
6:              for a ← 0, i do
7:                  if arr[i] > arr[maxNumPos] then
8:                      maxNumPos ← a
9:                  end if
10:             end for
11:             if maxNumPos ≠ (i − 1) and maxNumPos ≥ 0 then
12:                 Flip(arr, maxNumPos, i)
13:             end if
14:         end while
15:     end if
16: end procedure
```

Algorithm 3 Алгоритм быстрой сортировки

Вход: ссылка на массив arr индексы st и end

Выход: отсортированный массив arr

```
1:  procedure QSort(arr, st, end)
2:      l ← st
3:      r ← end
4:      piv ← arr[(l+r)/2]
5:      while l ≤ r do
6:          while arr[l] < piv do
7:              l ← l + 1
8:          end while
9:          while arr[l] < piv do
10:             r ← r − 1
11:         end while
12:         if l ≤ r then
13:             swap ← arr[l]
14:             arr[l] ← arr[r]
15:             arr[r] ← swap
16:             l ← l + 1
17:             r ← r − 1
18:         end if
19:     end while
20:     if st < r then
21:         QSort(arr, st, r)
22:     end if
23:     if end > l then
24:         QSort(arr, l, end)
25:     end if
26: end procedure
```

Algorithm 4 Алгоритм гномьей сортировки

Вход: ссылка на массив arr размером n

Выход: отсортированный массив arr

```
1:  procedure GnomeSort(arr, n)
2:      i ← 1
3:      j ← 2
4:      while i < n do
5:          if arr[i − 1] < arr[i] then
6:              i ← j
7:              j ← j + 1
8:          else
9:              swap ← arr[i − 1]
10:             arr[i − 1] ← arr[i]
11:             arr[i] ← swap
12:             i ← i − 1
13:             if i == 0 then
14:                 i ← j
15:                 j ← j + 1
16:             end if
17:         end if
18:     end while
19: end procedure
```

2.3 Описание используемых типов данных

При реализации алгоритмов будут использованы следующие типы данных:

— *массив* — динамический массив значений типа `int`.

2.4 Модель вычисления для проведения оценки трудоемкости

Введем модель вычислений, которая потребуется для определения трудоемкости каждого отдельного взятого алгоритма умножения матриц.

1) Трудоемкость базовых операций имеет:

   — значение 1 для операций:

$$+, -, =, + =, - =, ==, ! =, <, >, <=, >=, \\ [], ++, --, \&\&, ||, >>, <<, \&, | \tag{2.1}$$

   — значение 2 для операций:

$$*, /, \%, * =, / =, \% = . \tag{2.2}$$

2) Трудоемкость условного оператора:

$$f_{\text{if}} = \begin{cases} \min(f_1, f_2), & \text{лучший случай} \\ \max(f_1, f_2), & \text{худший случай.} \end{cases} \tag{2.3}$$

3) Трудоемкость цикла

$$f_{\text{for}} = f_{\text{инициализация}} + f_{\text{сравнение}} + \\ + M_{\text{итераций}} \cdot (f_{\text{тело}} + f_{\text{инкремент}} + f_{\text{сравнение}}). \tag{2.4}$$

4) Трудоемкость передачи параметра в функцию и возврат из нее равен 0.

2.5 Трудоемкость алгоритмов

Ниже приведены оценки трудоемкостей алгоритмов блинной, быстрой и гномьей сортировок.

Пусть требуется отсортировать массив A размером N.

2.6 Блинная сортировка

Трудоемкость блинной сортировки складывается из:

— трудоемкости проверки вырожденного случая, равной 1;

— трудоемкости цикла сортировки:

$$f_{PanSort} = 2 + N \cdot (2 + f_{FindMax} + f_{Flip}), \tag{2.5}$$

— трудоемкости поиска позиции максимального элемента в подмассиве $A[0..i]$, $i = \overline{1, N}$

$$f_{FindMax} = 3 + \\ +M \cdot \left(2 + \begin{cases} 1, & j\text{-й элемент больше максимального} \\ 0, & \text{иначе} \end{cases}\right), \tag{2.6}$$

где M — размер подмассива $A[1..i]$;

— трудоемкости отражения элементов массива

$$f_{Flip} = \begin{cases} 2, & \text{макс. элемент последний в } A[0..i] \\ f_{FlipLoop}, & \text{иначе,} \end{cases} \tag{2.7}$$

где $f_{FlipLoop}$ — трудоемкость цикла обмена элементами массива при его отражении;

$$f_{FlipLoop} = 2 + \frac{L}{2} \cdot (2 + 3 + 2), \tag{2.8}$$

где L — количество элементов в массиве между максимальным в подмассиве $A[0..i]$ и i-м.

Таким образом, трудоемкость блинной сортировки равна

$$F_{PanSort} = 2 + 7N + 3MN + \frac{7NL}{2} \approx O(N^2) \tag{2.9}$$

2.7 Быстрая сортировка

Трудоемкость быстрой сортировки складывается из:

$$f_{QSort} = 7 + 1 \cdot N \cdot (2 + \frac{N}{2} + 2 + \frac{N}{2} + 1 + 9) \tag{2.10}$$

2.8 Гномья сортировка

Трудоемкость гномьей сортировки складывается из:

— трудоемкости начальной инициализации переменных: 2;

— трудоемкости цикла сортировки:

$$f_{GnomeSort} = N \cdot (1 + f_{Branch}), \tag{2.11}$$

— трудоемкости обработки случая, когда обнаружен элемент, нарушающий порядок

$$f_{Branch} = 3 + \begin{cases} 2, & i\text{-й элемент} < i\text{-го} \\ 5 + 3 + 2 + 2, & \text{иначе} \end{cases} \tag{2.12}$$

Таким образом, трудоемкость гномьей сортировки равна

$$f_{GnomeSort} = 13N. \tag{2.13}$$

На неупорядоченных данных алгоритм имеет асимптотическую сложность $O(n^2)$, а на данных, близких к упорядоченным — $O(n)$ [2].

Вывод

3 Технологическая часть

В данном разделе приведены средства реализации программного обеспечения, сведения о модулях программы, листинг кода и функциональные тесты.

3.1 Средства реализации

В качестве языка программирования, используемого при написании данной лабораторной работы, был выбран C++ [3], так как в нем имеется контейнер `std::vector`, представляющий собой динамический массив данных произвольного типа, и библиотека `<ctime>` [4], позволяющая производить замеры процессорного времени.

В качестве средства написания кода была выбрана кроссплатформенная среда разработки *Visual Studio Code* за счет того, что она предоставляет функционал для проектирования, разработки и отладки ПО.

3.2 Сведения о модулях программы

Данная программа разбита на следующие модули:

— `main.cpp` — файл, содержащий точку входа в программу;

— `algorithms.cpp` — файл, содержащий реализации алгоритмов сортировки;

— `measure.cpp` — файл, содержащий функции, замеряющие процессорное время выполнения реализуемых алгоритмов.

3.3 Реализация алгоритмов

Вывод

Были реализованы алгоритмы сортировки (блинная, быстрая, гномья). Проведено тестирование реализованных алгортимов.

4 Исследовательская часть

В данном разделе будут приведены исследование временных характеристик реализуемых алгоритмов и оценка их затрат по памяти.

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялись замеры по времени:

— Процессор: Intel i5-1035G1 (8) @ 3.600 ГГц.

— Оперативная память: 16 ГБайт.

— Операционная система: Manjaro Linux x86_64 (версия ядра Linux 5.15.131-1-MANJARO).

Во время проведения измерений времени ноутбук был подключен к сети электропитания и был нагружен только системными приложениями.

4.2 Демонстрация работы программы

На рисунке 4.1 демонстрация работы программы для случая, когда пользователь выбирает опцию 1 «Сортировка массива» и передает в программу массив $[5, 3, 2, 9, 7]$.

```
                Меню
1. Сортировка массива с помощью:
   а) блинной сортировки;
   б) быстрой сортировки;
   в) гномьей сортировки.
2. Произвести замеры по времени реализуемых алгоритмов.
0. Выход.

Выберите опцию (0-2): 1

Введите размер сортируемого массива: 5
Введите сам массив: 5 3 2 9 7

Блинная сортировка: [ 2 3 5 7 9 ]
Гномья сортировка:  [ 2 3 5 7 9 ]
Быстрая сортировка: [ 2 3 5 7 9 ]

                Меню
1. Сортировка массива с помощью:
   а) блинной сортировки;
   б) быстрой сортировки;
   в) гномьей сортировки.
2. Произвести замеры по времени реализуемых алгоритмов.
0. Выход.

Выберите опцию (0-2): 0
```

Рисунок 4.1 – Демонстрация работы программы

4.3 Временные характеристики

4.4 Характеристики по памяти

4.5 Вывод

Заключение

В результате выполнения лаботарторной работы по исследованию алгоритмов сортировок решены следующие задачи:

1) описаны алгоритмы умножения матриц;

2) разработаны и реализованы соответствующие алгоритмы;

3) создан программный продукт, позволяющий протестировать реализованные алгоритмы;

4) проведен сравнительный анализ процессорного времени выполнения реализованных алгоритмов:

5) выполнена теоретическая оценка объема затрачиваемой памяти каждым из реализованных алгоритмов:

Список использованных источников

1 Кормен Т. Х., Лейзерсон Ч. И., Ривест Р. Л. [и др.] // Алгоритмы: построение и анализ. Издательский дом «Вильямс», 2013. — С. 1328.

2 Black Paul E. gnome sort. Dictionary of Algorithms and Data Structures. 2022.

3 Документация по Microsoft C++ [Электронный ресурс]. — Режим доступа: `https://learn.microsoft.com/ru-ru/cpp/?view=msvc-170&viewFallbackFrom=vs-2017` (дата обращения: 25.09.2023).

4 Standard library header <ctime> [Электронный ресурс]. — Режим доступа: `https://en.cppreference.com/w/cpp/header/ctime`.