САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Эмпирический анализ алгоритма сортировки слиянием.

По дисциплине: «Алгоритмы и анализ сложности»

Направление: «Фундаментальная информатика и информационные технологии»

Выполнил студент 3 курса группы 20.Б11-ПУ

Храмцов Андрей Игоревич

Санкт-Петербург

2022 г.

Содержание

1. Описание алгоритма сортировки слиянием 3
2. Математический анализ алгоритма сортировки слиянием 3
3. Входные данные 3
    3.1. Описание входных данных 3
    3.2. Генерация входных данных 4
4. Измерение трудоемкости 4
5. Программная реализация алгоритма 5
6. Вычислительный эксперимент 6
7. Источники 7
8. Характеристики вычислительной среды и оборудования 7

1. Описание алгоритма сортировки слиянием

Автором алгоритма сортировки слиянием (merge sort) является Джон фон Нейман. Он подготовил в 1945 году программы сортировки методом слияния, чтобы убедиться в необходимости некоторых типов команд, которые он предложил для машины EDVAC. Детальное описание этого алгоритма было опубликовано 15 апреля 1948г.

В основе этой сортировки лежит принцип «разделяй и влавствуй» (divide and conquer), то есть задача сначала разделяется на меньшие подзадачи, и потом их решения комбинируются для решения исходной задачи.

Работа алгоритма заключается в следующем:

1. Разделить неотсортированный список на n подсписков, каждый из которых содержит один элемент (список из одного элемента считается отсортированным)
2. Соединять подсписки друг с другом, получая отсортированные подсписки, пока не получим лишь один подсписок, являющийся искомым отсортированным списком

В работе Неймана и Голдстайна соединение подсписков называется meshing. Сейчас это принято называть слиянием (merge).

Одно из преимуществ данного алгоритма – устойчивость (порядок равных элементов не изменяется при сортировке). Также можно написать многопоточную сортировку слиянием.

Из недостатков – для сортировки слиянием требуется дополнительная память $O(n)$.

Сортировка слиянием полезна, например, когда важно гарантированное время $O(nlog(n))$, или когда имеют дело со списками, где невозможен произвольный доступ к данным.

2. Математический анализ алгоритма сортировки слиянием

Пусть $T(n)$ — время сортировки массива длины n.

Слияние двух массивов длин n и m происходит за $O(n+m)$, то есть слияние двух массивов длин $n/2$ происходит за $O(n)$.

Получим рекуррентное соотношение:

$$T(n) = 2T(n/2) + O(n)$$

По основной теореме получается:

$$T(n) = O(nlog(n))$$

3. Входные данные

3.1. Описание входных данных

Входными данными алгоритма является массив из n элементов, для которых определены операции сравнения.

При анализе алгоритма в данной работе рассматриваются массивы вещественных чисел в диапазоне от -100 до 100.

Проводится 10 тестов по 25 запусков для длины массива $n = 2^i$, i целое число от 1 до 25 и 10 тестов по 25 запусков для длины массива $n = 3 \cdot 2^i$, i целое число от 0 до 24.

3.2. Генерация входных данных

```
48  void GenerateArray(double *A, int n, double lo = -100, double hi = 100) {
49      for (int i = 0; i < n; ++i) {
50          A[i] = lo + (double(rand()) / double(RAND_MAX)) * (hi - lo);
51      }
52  }
```

(Рисунок 1. Функция, генерирующая массив длины n из вещественных чисел)

4. Измерение трудоемкости

Единицы измерения трудоемкости – время выполнения программы. Имеется в виду время выполнения самого этапа сортировки. Запоминается время до выполнения алгоритма и после выполнения, разница этих значений является временем работы.

5. Программная реализация алгоритма

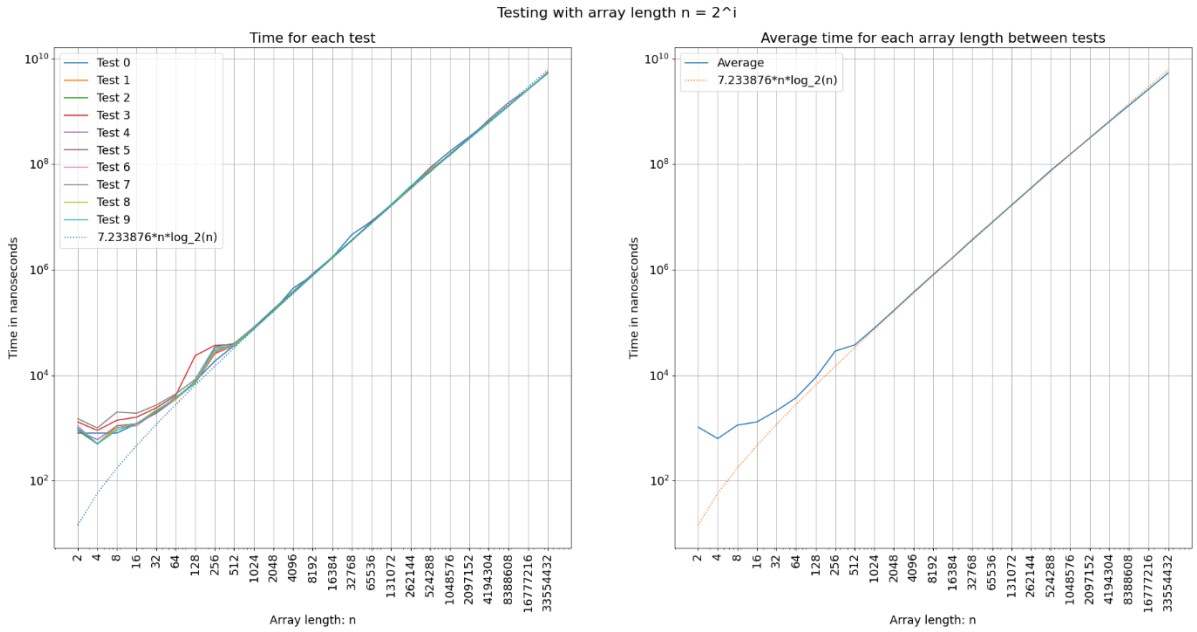
```
 9   template<typename T>
10   T* MergeSort(T* A, int n) {
11       T* a[2] = { A , new T[n] };
12       int c1 = 0; int c2 = 1;
13       int L = 1;
14       int i, j, topL, topR, k;
15       while (L < n) {
16           c2 = c1; c1 = 1 - c1;
17           k = 0;
18           while (k < n) {
19               i = k; j = k + L;
20               topL = j; topR = j + L;
21               if (topL > n) { topL = n; topR = 0; }
22               else if (topR > n) { topR = n; }
23               while (i < topL && j < topR) {
24                   if (a[c2][i] > a[c2][j]) {
25                       a[c1][k] = a[c2][j]; ++j; ++k;
26                   }
27                   else {
28                       a[c1][k] = a[c2][i]; ++i; ++k;
29                   }
30               }
31               if (i < topL) {
32                   while (i < topL) {
33                       a[c1][k] = a[c2][i]; ++i; ++k;
34                   }
35               }
36               else if (j < topR){
37                   while (j < topR) {
38                       a[c1][k] = a[c2][j]; ++j; ++k;
39                   }
40               }
41           }
42           L *= 2;
43       }
44       delete[] a[c2];
45       return a[c1];
46   }
```

(Рисунок 2. Функция сортировки слиянием)

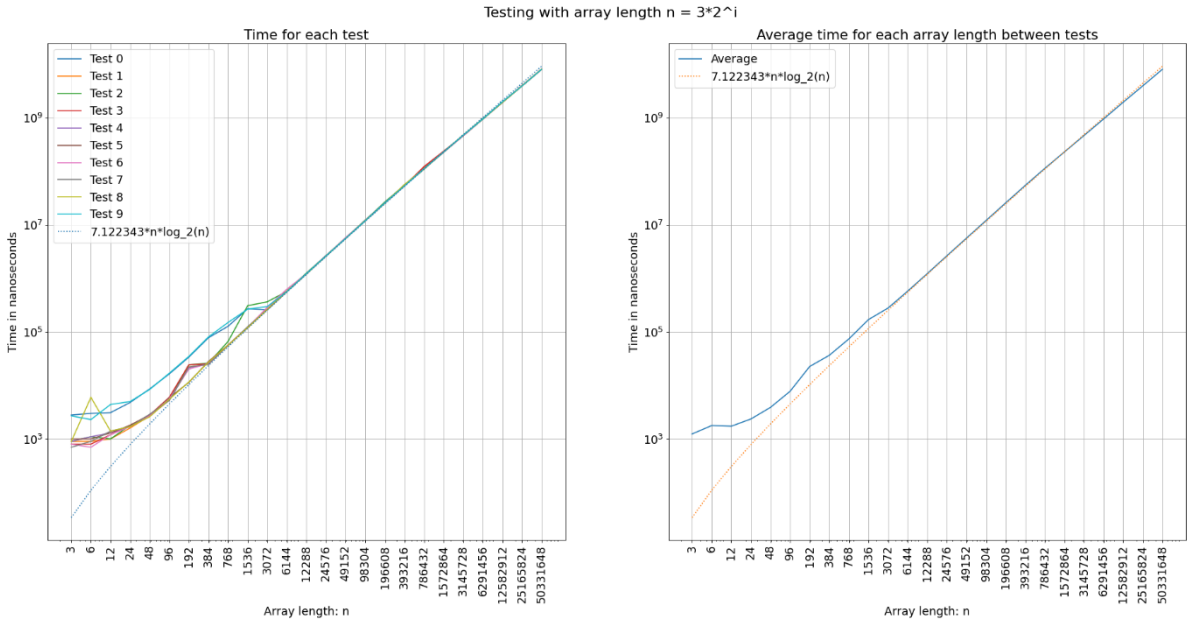
Полная реализация программы доступна по ссылке: https://github.com/ahrami/merge-sort

6. Вычислительный эксперимент

В соответствии с заданными ранее входными данными я провел вычислительный эксперимент. На рисунках 3 и 4 изображены его результаты для длин массива $n = 2^i$ и $n = 3 \cdot 2^i$ соответственно. Слева на этих рисунках каждый из 10 тестов изображен по отдельности. Справа – среднее значение занятого времени между 10 тестами, взятое для каждого n. На каждом графике также нарисована функция $g(n) = nlog(n)$, умноженная на константу, которая вычисляется как среднее значение отношения времени работы алгоритма к $nlog(n)$. Это сделано, чтобы проверить соответствие результатов теоретическим оценкам. Как мы видим, алгоритм действительно принадлежит классу временной сложности $O(nlog(n))$.



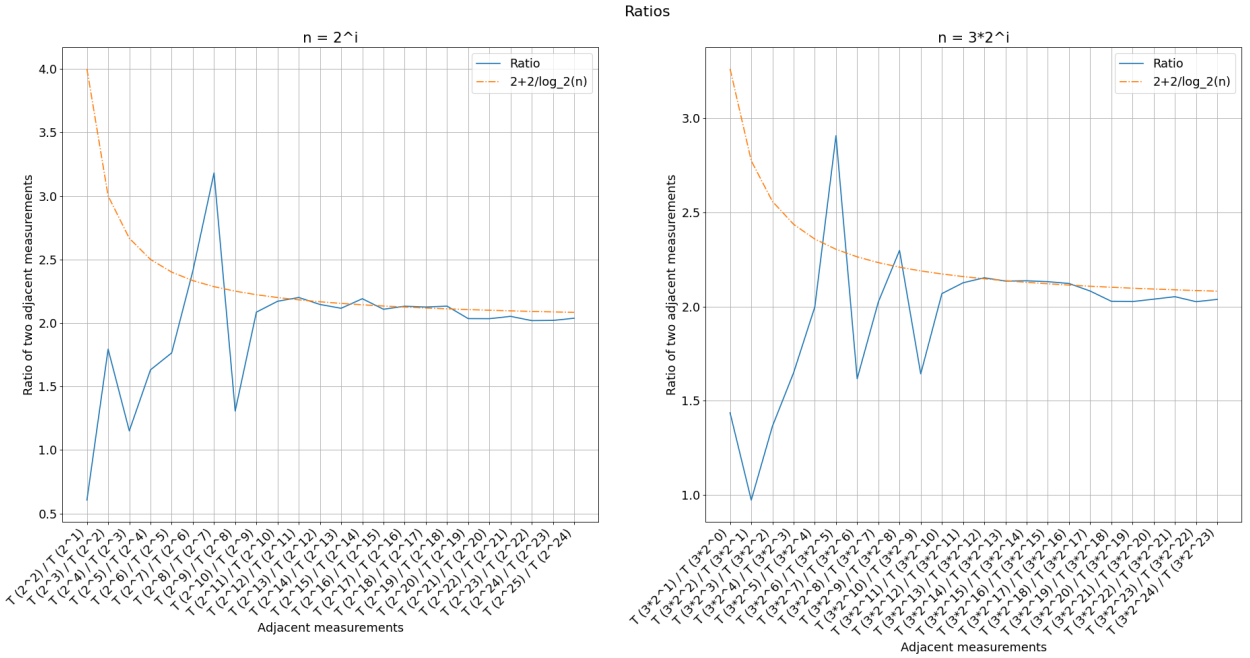
(Рисунок 3. Графики тестов, проведенных для длинны массива $n = 2^i$)



(Рисунок 4. Графики тестов, проведенных для длинны массива $n = 3 \cdot 2^i$)

Для рассмотрения отношения значений измеренной трудоемкости при удвоении размера входных данных я взял ранее найденные усредненные значения времени. На рисунке 5 изображены эти отношения вместе с графиком функции $f(n) = \frac{g(2n)}{g(n)} = \frac{2nlog(2n)}{nlog(n)} = 2 + \frac{2}{log(n)}$.

При малых n отношение очень сильно скачет, но с увеличением n мы видим, что графики идут близко друг к другу, что говорит о соответствии практических и теоретических оценок.



(Рисунок 5. Графики отношения значений измеренной трудоемкости при удвоении размера входных данных)

7. Источники

1. Planning and Coding of Problems for an Electronic Computing Instrument - Goldstine and von Neumann. Part II, Volume II. 15 April 1948.
2. Кнут Д. Э. Искусство программирования. Том 3. Сортировка и поиск под ред. В. Т. Тертышного (гл. 5) и И. В. Красикова (гл. 6). — 2-е изд. — Москва: Вильямс, 2007. — Т. 3. — 832 с.
3. Кормен, Лейзерсон, Ривест: Алгоритмы. Построение и анализ.

8. Характеристики вычислительной среды и оборудования

CPU – AMD Ryzen 7 Mobile 4800H

RAM – DDR4 16GBytes, dual channel

IDE – Visual Studio Community 2019 16.11.11