Московский государственный университет имени М.В.Ломоносова
Факультет вычислительной математики и кибернетики

Параллельные (высокопроизводительные) вычисления

Отчет по заданию №1 «Расписание сети сортировки»

Богатенкова Анастасия Олеговна
528 группа

12 октября 2021 г.

Содержание

1 Описание условия

Разработать последовательную программу вычисления расписания сети сортировки, числа использованных компараторов и числа тактов, необходимых для её срабатывания при выполнении на n процессорах. Число тактов сортировки при параллельной обработке не должно превышать числа тактов, затрачиваемых четно-нечетной сортировкой Бетчера.

Параметр командной строки запуска: n.

$n >= 1$ – количество элементов в упорядочиваемом массиве, элементы которого расположены на строках с номерами $[0...n-1]$.

Формат команды запуска:

bsort n

Требуется:

1. вывести в файл стандартного вывода расписание и его характеристики в представленном далее формате;
2. обеспечить возможность вычисления сети сортировки для числа элементов $1 <= n <= 10000$;
3. предусмотреть полную проверку правильности сети сортировки для значений числа сортируемых элементов $1 <= n <= 24$;
4. представить краткий отчет удовлетворяющий указанным далее требованиям.

Формат файла результата:

n 0 0
$cu_0 cd_0$
$cu_1 cd_1$
...
$cu_{n\_comp-1} cd_{n\_comp-1}$
n_comp
n_tact

Здесь:

n 0 0 – число сортируемых элементов, ноль, ноль.
cu_i cd_i – номера строк, соединяемых i-м компаратором сравнения перестановки.
n_comp – число компараторов.
n_tact – число тактов сети сортировки.

2 Описание метода решения

Реализован алгоритм четно-нечетной сортировки Бетчера для вычисления расписания сети сортировки. Общая схема четно-нечетной сортировки Бетчера для массива из 13 элементов представлена на рисунке 1.

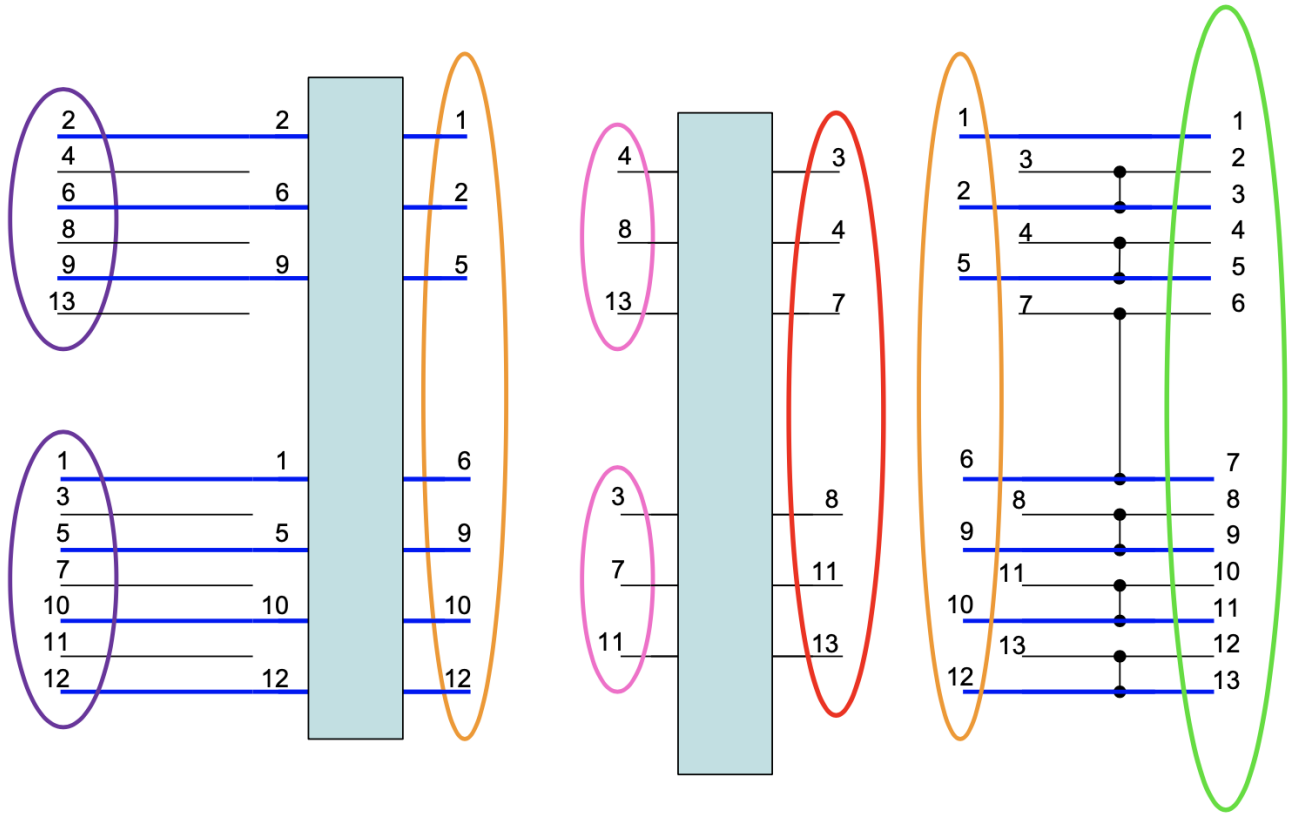

Рис. 1: Четно-нечетная сортировка Бетчера

В алгоритме можно выделить следующие основные части:

1. Разделение массива на две равные части, если массив четной длины, либо на части, отличающиеся длиной на 1 (пусть первая часть имеет меньший размер) в случае массива нечетной длины. Каждая из частей должна быть отсортирована тем же способом.

2. Выделение в каждой части элементов с четными и нечетными номерами, отдельная сортировка пар массивов из элементов с четными и нечетными номерами.

3. Слияние результатов сортировки массивов из элементов с четными и нечетными номерами.

Для реализации алгоритма написана программа на C++, в которой первый пункт алгоритма реализуется рекурсивной функцией *bsort*, а второй и третий пункты – рекурсивной функцией *sortTwoArrays*. Для проверки правильности реализации написан скрипт, проверяющий правильность числа сравнений и тактов для значений числа сортируемых элементов $1 <= n <= 24$.

3 Описание метода проверки

Количество сравнений и тактов для значений числа сортируемых элементов $1 <= n <= 24$ вычислялись вручную с использованием рекурсивной природы алгоритма. На рисунке 2 представлены сети сортировки для массивов с $1 <= n <= 5$ элементами.

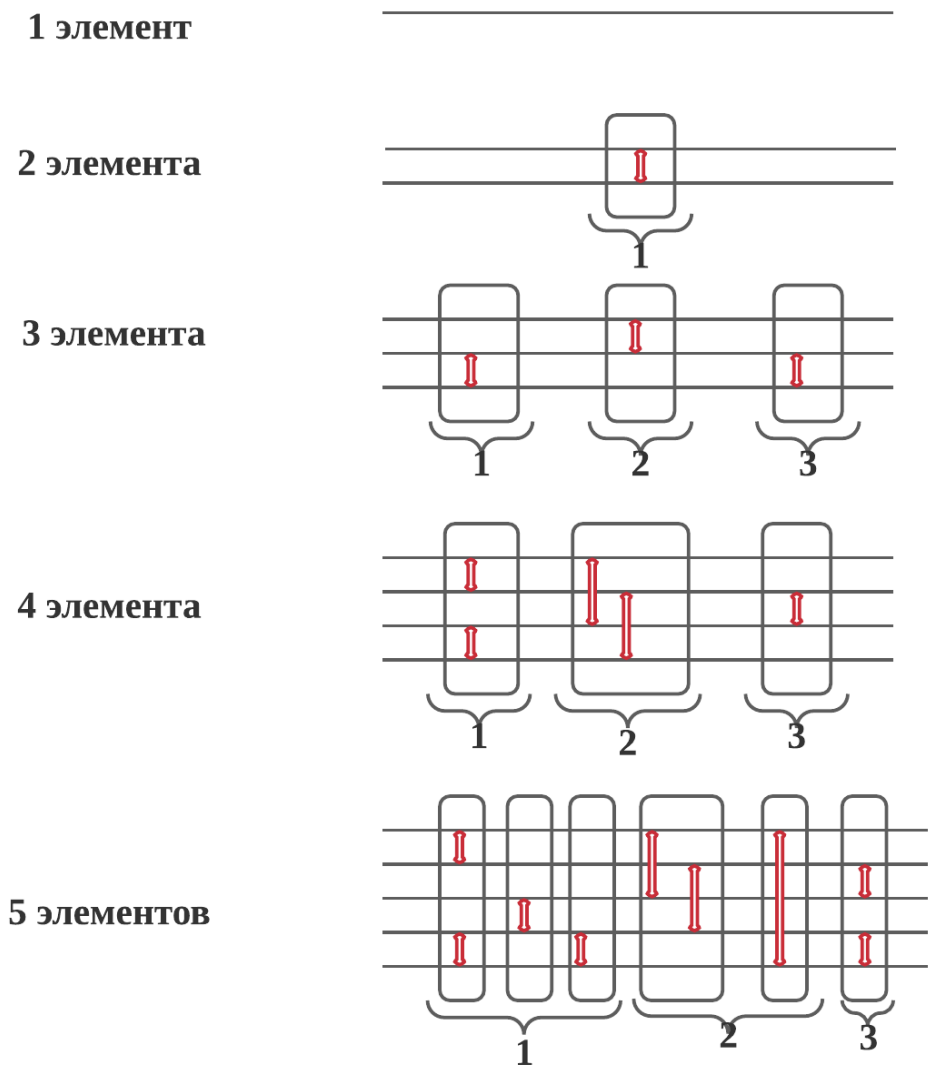

Рис. 2: Сети сортировки для массивов длины <= 5

Для массивов из 1 и 2 элементов число сравнений и тактов считается тривиально. Для больших размеров применялась следующая стратегия:

1. Разделить массив на две части и взять значение числа сравнений и тактов для размеров этих частей. Число сравнений суммируется, число тактов вычисляется как максимум из числа тактов, необходимых для сортировки каждой из частей, так как сортировку частей можно выполнять параллельно.

2. Выделить в каждой из частей массивы элементов с четными и нечетными номерами, основываясь на предыдущих значениях посчитать отдельно для четного и нечетного случая число тактов для слияния массивов. Число сравнений суммируемся, число тактов вычисляется как максимум из числа тактов, необходимых для слияния массивов из четных элементов и массивов из нечетных элементов, так как эти массивы не пересекаются и слияния можно выполнять параллельно.

3. Слияние результатов сортировки массивов из элементов с четными и нечетными номерами выполняется за один такт и требует $round(\frac{n}{2}) - 1$ сравнений.

Для примера рассмотрим сеть сортировки массива из 5 элементов.

1. Сначала массив делится на части из 2 и 3 элементов. Для сортировки массива из 2 элементов необходимо 1 сравнение и 1 такт, для сортировки массива из 3 элементов — 3 сравнения и 3 такта. Значит, для сортировки частей (пункт 1 алгоритма) нужно $1 + 3 = 4$ сравнений и $max(1, 3) = 3$ тактов.

2. Далее выделяются массивы четных и нечетных элементов. Необходимо соединить массивы нечетных элементов длины 1 и 2, а также массивы четных элементов длины 1 и 1. Слияние массивов длины 1 и 2 осуществляется аналогично пунктам 2 и 3 алгоритма для массива из 3 элементов и выполняется за 2 такта с 2 сравнениями. Для слияния массивов единичной длины необходимо 1 сравнение и 1 такт. Таким образом, для этого шага нужно $2+1 = 3$ сравнений и $max(2, 1) = 2$ такта.

3. Для слияния массивов из элементов с четными и нечетными номерами небходим 1 такт и $round(\frac{5}{2})-1 = 2$ сравнения. Суммируя результаты всех шагов, получаем $4 + 3 + 2 = 9$ сравнений и $3 + 2 + 1 = 6$ тактов.

Аналогичным образом были посчитаны числа сравнений и тактов для массивов с числом элементов до 24 включительно. Результаты подсчётов представлены в таблице 1.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
comp	0	1	3	5	9	12	16	19	26	31	37	41	48	53	59	63	74	82	91	97	107	114	122	127
tact	0	1	3	3	6	6	6	6	10	10	10	10	10	10	10	10	15	15	15	15	15	15	15	15

Таблица 1: Количество сравнений и тактов для значений числа сортируемых элементов $1 <= n <= 24$

4 Приложение 1

Для реализации алгоритма написана программа на C++, в текст которой был встроен подсчет числа сравнений и тактов в соответствии с описанием их подсчета в предыдущей секции. Программа проверялась с помощью скрипта на языке Python на правильность подсчета числа сравнений и тактов.