

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Образовательная программа бакалавриата «Программная инженерия»

**ОТЧЁТ К ДОМАШНЕМУ ЗАДАНИЮ № 4
ПО ДИСЦИПЛИНЕ
«АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ»**

ВАРИАНТ 15

Исполнитель
студент группы БПИ191
Н. К. Игумнов

Москва 2020

ЗАДАНИЕ

Требуется вывести список всех целых чисел, содержащих от 4 до 9 значащих цифр, которые после умножения будут содержать все те же самые цифры в произвольной последовательности и в произвольном количестве.

Входные данные: целое положительное число n , больше единицы и меньше десяти, количество потоков.

Также, согласно правкам:

- Помимо n и количества потоков вводятся целочисленные числа l и r – левая и правая границы чисел. Причём $[l, r] \subseteq [1000, 10^{10})$ – лежит в исходных границах.
- Выводится вся необходимая информация для демонстрации корректности работы: исходное число n , отрезок $[l, r]$, количество потоков, количество выводимых чисел. Список выводится построчно в формате "исходное число, результат и номер потока".

ИНТЕРПРЕТАЦИЯ

Пройдёмся по всем числам x из промежутка $[l, r]$, если после умножения на входной параметр $n \in (1, 10)$ все цифры числа $new_x = x \cdot n$ содержатся в x (и наоборот), то добавим его в итоговый ответ. Выведем список всех полученных чисел в консоль / файл.

РЕШЕНИЕ (АЛГОРИТМ)

Для проверки, содержатся ли все цифры нового числа new_x в x , всего было предложено пять вариантов (x – целочисленное число):

1. Приведём x в строку методом `std::to_string()`, пройдемся по всем символам y в полученной строке и добавим их в `std::set<char> s` (сделаем `s.insert(y)`). Повторим эту операцию для $x \cdot n$ (только выделим для данного числа другой `set`). Если `set`-ы совпадут, то условие выполняется.
2. Пройдемся по всем цифрам y числа x и добавим их в `std::set<int> s` (сделаем `s.insert(y)`). Повторим эту операцию для $x \cdot n$ (только выделим для данного числа другой `set`). Если `set`-ы совпадут, то условие выполняется.
3. Выделим для каждого числа (x и $x \cdot n$) отдельный массив `bool`-ов размера 10 (изначально заполненный `false`), отвечающий за наличие / отсутствие конкретных цифр в числе. Пройдемся по всем цифрам y числа x и обновим значение массива a , соответствующее данной цифре (сделаем `a[y] = true`). Повторим эту операцию для $x \cdot n$. Если массивы совпадут, то условие выполняется.
4. Выделим для каждого числа отдельный `std::bitset<10>`, отвечающий за наличие / отсутствие конкретных цифр в числе. Пройдемся по всем цифрам y числа x и обновим значение `bitset`-а b , соответствующее данной цифре (сделаем `b[y] = true`). Повторим эту операцию для $x \cdot n$. Если `bitset`-ы совпадут, то условие выполняется.
5. Выделим для каждого числа отдельное дополнительное целое число f_x (`int`), отвечающее за наличие / отсутствие конкретных цифр в числе. Пройдемся по всем цифрам y числа x и обновим f_x . Повторим эту операцию для $x \cdot n$. Если данные числа совпадут, то условие выполняется. – Это возможно, так как в худшем случае число состоит из всех цифр $0, \dots, 9$. Тогда значение этой функции будет равно $2^0 + \dots + 2^9 = 2^{10} - 1 = 1023$. То есть это число помещается в `int`.

ПРОГРАММА

Функция ***Read*** считывает входные данные: n – входной множитель, l – левая граница чисел, r – правая граница чисел, $thread_number$ – количество потоков $file_name$ – путь до выходного файла (в случае надобности).

В данной задаче я применяю модель **итеративного параллелизма**.

Функция ***Compute*** находит ответ на задачу, вызывая $thread_number$ потоков. Каждый из потоков будет проверять своё подмножество чисел из исходного диапазона с помощью функции ***Fun*** – возвращающей структуру данных, отвечающую за наличие / отсутствие конкретных цифр в числе (см. выше).

Функция ***Print*** выводит в консоль / выходной файл (в зависимости от FILE_OUT) список всех полученных чисел.

Для того, чтобы программа сгенерировала входные числа, требуется раскомментировать 12-ую строчку: `#define GENERATE`.

Для того, чтобы программа выводила всю информацию в файл, требуется раскомментировать 13-ую строчку: `#define FILE_OUT`.

ТЕСТИРОВАНИЕ ПРОГРАММЫ

1. Некорректные вводы ($N \leq 1$; $N \geq 10$; $l < 1000$; $r < l$)

```
multiThread — -zsh — 91x28
nikitaigumnov@MacBook-Pro-Nikita-I multiThread % clang++ -std=c++17 -Xpreprocessor -fopenmp
[ main.cpp -o main -lomp && ./main ]
Input your number n (in (1, 10)): 1
Incorrect number!
nikitaigumnov@MacBook-Pro-Nikita-I multiThread % clang++ -std=c++17 -Xpreprocessor -fopenmp
[ main.cpp -o main -lomp && ./main ]
Input your number n (in (1, 10)): 10
Incorrect number!
nikitaigumnov@MacBook-Pro-Nikita-I multiThread % clang++ -std=c++17 -Xpreprocessor -fopenmp
[ main.cpp -o main -lomp && ./main ]
Input your number n (in (1, 10)): 2
Input your left border (in [1000, 999999999]): 999
Incorrect l!
nikitaigumnov@MacBook-Pro-Nikita-I multiThread % clang++ -std=c++17 -Xpreprocessor -fopenmp
[ main.cpp -o main -lomp && ./main ]
Input your number n (in (1, 10)): 2
Input your left border (in [1000, 999999999]): 459423
Input your right border (in [459423, 999999999]): 459422
Incorrect r!
nikitaigumnov@MacBook-Pro-Nikita-I multiThread % clang++ -std=c++17 -Xpreprocessor -fopenmp
[ main.cpp -o main -lomp && ./main ]
Input your number n (in (1, 10)): 2
Input your left border (in [1000, 999999999]): 1000
Input your right border (in [1000, 999999999]): 999999999
Optimal number of threads for your machine: 4
Input amount of threads (in [1, 10000]): 0
Incorrect amount of threads!
nikitaigumnov@MacBook-Pro-Nikita-I multiThread %
```

Программа отработала успешно.

2. $N = 2, l = 1000, r = 999999999, thread_number = 4$, ВЫВОД В КОНСОЛЬ

```
multiThread — zsh — 91x28
original: 749987412, result: 1499974824, thread: 2
original: 749987421, result: 1499974842, thread: 2
original: 749988912, result: 1499977824, thread: 2
original: 749988921, result: 1499977842, thread: 2
original: 749989012, result: 1499978024, thread: 2
original: 749989021, result: 1499978042, thread: 2
original: 749989102, result: 1499978204, thread: 2
original: 749989112, result: 1499978224, thread: 2
original: 749989120, result: 1499978240, thread: 2
original: 749989121, result: 1499978242, thread: 2
original: 749989122, result: 1499978244, thread: 2
original: 749989124, result: 1499978248, thread: 2
original: 749989142, result: 1499978284, thread: 2
original: 749989201, result: 1499978402, thread: 2
original: 749989210, result: 1499978420, thread: 2
original: 749989211, result: 1499978422, thread: 2
original: 749989212, result: 1499978424, thread: 2
original: 749989214, result: 1499978428, thread: 2
original: 749989221, result: 1499978442, thread: 2
original: 749989241, result: 1499978482, thread: 2
original: 749989412, result: 1499978824, thread: 2
original: 749989421, result: 1499978842, thread: 2
original: 749989912, result: 1499979824, thread: 2
original: 749989921, result: 1499979842, thread: 2
original: 749998912, result: 1499997824, thread: 2
original: 749998921, result: 1499997842, thread: 2
Time (computation): 233.813 sec
nikitaigumnov@MacBook-Pro-Nikita-I multiThread %
```

Программа отработала успешно.

3. $N = 2, l = 1000, r = 100000, thread_number = 4$, ВЫВОД В КОНСОЛЬ

```
multiThread — zsh — 89x28
original: 52501, result: 105002, thread: 2
original: 52510, result: 105020, thread: 2
original: 10255, result: 20510, thread: 0
original: 10525, result: 21050, thread: 0
original: 12505, result: 25010, thread: 0
original: 12550, result: 25100, thread: 0
original: 87124, result: 174248, thread: 3
original: 87142, result: 174284, thread: 3
original: 87214, result: 174428, thread: 3
original: 87241, result: 174482, thread: 3
original: 87412, result: 174824, thread: 3
original: 87421, result: 174842, thread: 3
original: 25051, result: 50102, thread: 0
original: 25105, result: 50210, thread: 0
original: 25501, result: 51002, thread: 0
original: 25510, result: 51020, thread: 0
original: 44897, result: 89794, thread: 1
original: 44987, result: 89974, thread: 1
original: 47489, result: 94978, thread: 1
original: 48749, result: 97498, thread: 1
original: 48947, result: 97894, thread: 1
original: 48974, result: 97948, thread: 1
original: 49487, result: 98974, thread: 1
original: 49874, result: 99748, thread: 1
original: 50125, result: 100250, thread: 1
original: 50251, result: 100502, thread: 1
Time (computation): 0.012 sec
nikitaigumnov@MacBook-Pro-Nikita-I multiThread %
```

Программа отработала успешно.

4. $N = 3, l = 90000, r = 110000, thread_number = 10$, ВЫВОД В КОНСОЛЬ

```

multiThread — zsh — 89x28
Input amount of threads (in [1, 10000]): 10
Size of result array: 24
original: 98294, result: 294882, thread: 4
original: 98562, result: 295686, thread: 4
original: 98962, result: 296886, thread: 4
original: 107235, result: 321705, thread: 8
original: 99428, result: 298284, thread: 4
original: 105035, result: 315105, thread: 7
original: 100035, result: 300105, thread: 5
original: 100335, result: 301005, thread: 5
original: 100350, result: 301050, thread: 5
original: 100351, result: 301053, thread: 5
original: 102375, result: 307125, thread: 6
original: 96482, result: 289446, thread: 3
original: 101035, result: 303105, thread: 5
original: 103335, result: 310005, thread: 6
original: 103350, result: 310050, thread: 6
original: 103351, result: 310053, thread: 6
original: 103428, result: 310284, thread: 6
original: 103500, result: 310500, thread: 6
original: 103501, result: 310503, thread: 6
original: 103505, result: 310515, thread: 6
original: 103510, result: 310530, thread: 6
original: 103511, result: 310533, thread: 6
original: 97525, result: 292575, thread: 3
original: 94298, result: 282894, thread: 2
Time (computation): 0.002 sec
nikitaigumnov@MacBook-Pro-Nikita-I multiThread %

```

Программа отработала успешно.

5. $N = 3, l = 130000, r = 150000, thread_number = 3$, ВЫВОД В ФАЙЛ (см. tests/answer1.txt)

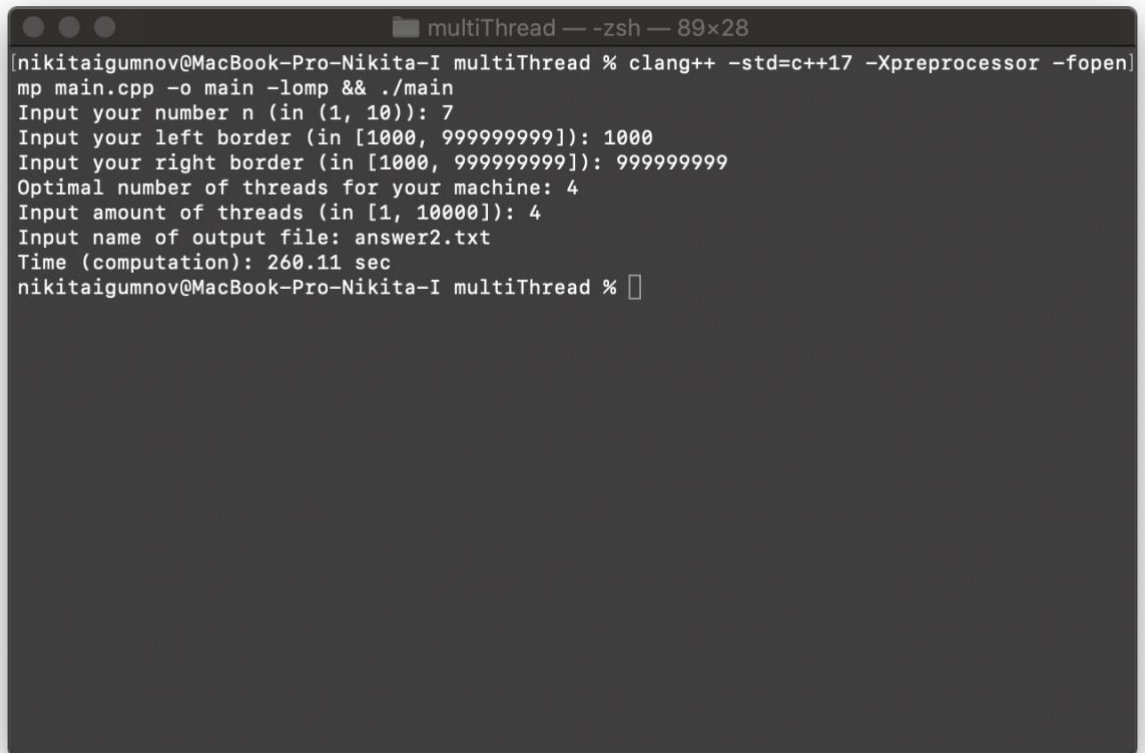
```

multiThread — zsh — 89x28
nikitaigumnov@MacBook-Pro-Nikita-I multiThread % clang++ -std=c++17 -Xpreprocessor -fopenmp main.cpp -o main -lomp && ./main
Input your number n (in (1, 10)): 3
Input your left border (in [1000, 999999999]): 130000
Input your right border (in [130000, 999999999]): 150000
Optimal number of threads for your machine: 4
Input amount of threads (in [1, 10000]): 3
Input name of output file: answer1.txt
Time (computation): 0.002 sec
nikitaigumnov@MacBook-Pro-Nikita-I multiThread %

```

Программа отработала успешно

6. $N = 7, l = 10000, r = 999999999, thread_number = 4$, вывод в файл (см. tests/answer2.txt)



```
multiThread — -zsh — 89x28
[nikitaigumnov@MacBook-Pro-Nikita-I multiThread % clang++ -std=c++17 -Xpreprocessor -fopenmp main.cpp -o main -lomp && ./main
Input your number n (in (1, 10)): 7
Input your left border (in [1000, 999999999]): 1000
Input your right border (in [1000, 999999999]): 999999999
Optimal number of threads for your machine: 4
Input amount of threads (in [1, 10000]): 4
Input name of output file: answer2.txt
Time (computation): 260.11 sec
nikitaigumnov@MacBook-Pro-Nikita-I multiThread %
```

Программа отработала успешно

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1) Ошибки многопоточности в C++ [Электронный ресурс]. Режим доступа: <https://otus.ru/nest/post/145/>, свободный. (дата обращения: 17.11.2020)
- 2) `std::thread::hardware_concurrency` [Электронный ресурс]. Режим доступа: https://en.cppreference.com/w/cpp/thread/thread/hardware_concurrency, свободный. (дата обращения: 17.11.2020)
- 3) Measuring execution time of a function in C++ [Электронный ресурс]. Режим доступа: <https://stackoverflow.com/questions/22387586/measuring-execution-time-of-a-function-in-c>, свободный. (дата обращения: 17.11.2020)
- 4) OpenMP [Электронный ресурс]. Режим доступа: <https://ru.wikipedia.org/wiki/OpenMP>, свободный. (дата обращения: 01.12.2020)
- 5) Параллельные заметки №3 – базовые конструкции OpenMP [Электронный ресурс]. Режим доступа: <https://habr.com/ru/company/intel/blog/85273/>, свободный. (дата обращения: 01.12.2020)
- 6) Choosing the number of threads at runtime [Электронный ресурс]. Режим доступа: <https://livebook.manning.com/book/c-plus-plus-concurrency-in-action/chapter-2/92/>, свободный. (дата обращения: 17.11.2020)

КОД ПРОГРАММЫ

Смотрите main.cpp

```
#include <iostream>
#include <optional>
#include <utility>
#include <thread>
#include <chrono>
#include <random>
#include <fstream>
#include <vector>
#include <mutex>
#include "omp.h"

// #define GENERATE
// #define FILE_OUT

constexpr int64_t MIN_VALUE = 1000;
constexpr int64_t MAX_VALUE = 999'999'999'11;

std::mutex mut;

bool Read(int *n, int64_t *l, int64_t *r, int *thread_number, std::string
*file_name) { // NOLINT
#ifdef GENERATE
    std::mt19937
gen(std::chrono::high_resolution_clock::now().time_since_epoch().count());
    std::uniform_int_distribution<int> dist_n(2, 9);
    *n = dist_n(gen);
    std::uniform_int_distribution<int64_t> dist_border(MIN_VALUE, MAX_VALUE);
    *l = dist_border(gen);
    *r = dist_border(gen);
    if (l > r) {
        std::swap(l, r);
    }
    std::uniform_int_distribution<int> dist_t(1, 10000);
    *thread_number = dist_t(gen);
    *file_name = "answer.txt";
    std::cout << "Number n: " << *n << '\n';
    std::cout << "Your borders: [l, r] = [" << *l << ", " << r << "]\n";
    std::cout << "Amount of threads: " << *thread_number << '\n';
#else
    std::cout << "Input your number n (in (1, 10)):";
    std::cin >> *n;
    if (*n <= 1 || *n >= 10) {
        std::cout << "Incorrect number!\n";
        return false;
    }
    std::cout << "Input your left border (in [" << MIN_VALUE << ", " <<
MAX_VALUE << "]): ";
    std::cin >> *l;
    if (*l < MIN_VALUE || *l > MAX_VALUE) {
        std::cout << "Incorrect l!\n";
        return false;
    }
    std::cout << "Input your right border (in [" << *l << ", " << MAX_VALUE
<< "]): ";
    std::cin >> *r;
    if (*r < *l || *r > MAX_VALUE) {
```

```

        std::cout << "Incorrect r!\n";
        return false;
    }
    // see https://livebook.manning.com/book/c-plus-plus-concurrency-in-
    action/chapter-2/92 for details
    unsigned int hardware_threads = std::thread::hardware_concurrency();
    std::cout << "Optimal number of threads for your machine: " <<
    (hardware_threads != 0 ? hardware_threads : 2)
        << '\n';
    std::cout << "Input amount of threads (in [1, 10000]): ";
    std::cin >> *thread_number;
    if (*thread_number < 1 || *thread_number > 10000) {
        std::cout << "Incorrect amount of threads!\n";
        return false;
    }
#endif
#ifdef FILE_OUT
    std::cout << "Input name of output file: ";
    std::cin >> *file_name;
    std::ofstream out(*file_name);
    if (!out.is_open()) {
        std::cout << "Incorrect file name!\n";
        out.close();
        return false;
    }
    out.close();
#endif
    return true;
}

unsigned int Fun(int64_t x) {
    unsigned int y = 0; // [0, 1024)
    while (x != 0) {
        y |= 1u << unsigned(x % 10);
        x /= 10;
    }
    return y;
}

struct Info {
    int64_t number;
    int thread_id;
};

void Compute(int n, int64_t from, int64_t to, std::vector<Info> &numbers) {
#pragma omp parallel for default(none) shared(numbers, mut, from, to, n)
    for (int64_t i = from; i <= to; ++i) {
        if (Fun(i) == Fun(i * n)) {
            mut.lock();
            numbers.push_back({i, omp_get_thread_num()});
            mut.unlock();
        }
    }
}

void Print(int n, int l, int r, int threads_num, const std::string
&file_name, const std::vector<Info> &answer) {
#ifdef FILE_OUT
    std::ofstream out(file_name);
    out << "Number n: " << n << '\n';
    out << "Your borders: [l, r] = [" << l << ", " << r << "]\n";
    out << "Size of result array: " << answer.size() << '\n';

```

```

        for (const Info &info : answer) {
            out << "original: " << info.number << ", modified: " << info.number *
n <<
                ", thread: " << info.thread_id << '\n';
        }
        out.close();
    #else
        std::cout << "Size of result array: " << answer.size() << '\n';
        for (const Info &info : answer) {
            std::cout << "original: " << info.number << ", result: " <<
info.number * n <<
                ", thread: " << info.thread_id << '\n';
        }
    #endif
}

int main() {
    std::ios_base::sync_with_stdio(false);
    int n, thread_number;
    int64_t l, r;
    std::string file_name;
    if (!Read(&n, &l, &r, &thread_number, &file_name)) {
        return 1;
    }
    omp_set_dynamic(0);          // запретить библиотеке openmp менять число
потокoв во время исполнения
    omp_set_num_threads(thread_number); // установить число потокoв в
thread_number

    std::vector<Info> numbers;
    auto begin = std::chrono::steady_clock::now();

    Compute(n, l, r, numbers);

    auto end = std::chrono::steady_clock::now();
    auto elapsed_ms =
std::chrono::duration_cast<std::chrono::milliseconds>(end - begin);

    Print(n, l, r, thread_number, file_name, numbers);
    std::cout << "Time (computation): " << elapsed_ms.count() / 1000. << "
sec\n";
    return 0;
}

```