



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 №6 по курсу "Анализ алгоритмов"

Тема Муравьиный алгоритм

Студент Ковалев Д.А.

Группа ИУ7-53Б

Оценка (баллы) _____

Преподаватели Волкова Л.Л., Строганов Ю.В.

Москва — 2020 г.

Оглавление

Введение	3
1 Аналитическая часть	4
1.1 Простой алгоритм	4
1.2 Улучшенный алгоритм	4
1.3 Улучшенный алгоритм с использование решета Эратосфена	4
2 Технологическая часть	5
2.1 Средства реализации	5
2.2 Листинг кода	5
3 Исследовательская часть	8
Заключение	9
Литература	10

Введение

В современной теории алгоритмов существует ряд задач, получение точного ответа в которых возможно только полным перебором всевозможных вариантов множества решений (т.н. NP-полные задачи). Однако зачастую в подобных задачах оптимизации допустимо получение ответа, приближенного к идеальному. В этих целях используются алгоритмы, работающие за приемлемое полиномиальное время, такие как генетические и муравьиные алгоритмы.

Муравьиный алгоритм — один из эффективных полиномиальных алгоритмов для нахождения приближенных решений задачи коммивояжера, а также решения аналогичных задач поиска маршрутов на графах.

В ходе лабораторной работы предстоит:

- рассмотреть муравьиный алгоритм и алгоритм полного перебора в задаче коммивояжера;
- реализовать алгоритмы;
- сравнить временную эффективность алгоритмов.

1 Аналитическая часть

В данном разделе будут описаны алгоритмы поиска первых N простых чисел

1.1 Простой алгоритм

Простой перебор всех чисел с проверкой на простоту числа, пока не найдем первые нужные N

1.2 Улучшенный алгоритм

Во-первых, проверять четные числа не имеет смысла, ведь 2 — единственное простое четное число. То же самое с делителями — на четные можно не проверять. В-третьих, в качестве делителей достаточно брать числа не превышающие половину проверяемого числа.

1.3 Улучшенный алгоритм с использование решета Эратосфена

Наиболее идейно простым алгоритмом решения задачи коммивояжера [1] является полный перебор решений с выбором кратчайшего из полученных путей. Очевидным недостатком данного алгоритма является необходимость перебора значительного числа комбинаций, которое с ростом числа городов быстро выходит за рамки вычислительных мощностей современных компьютеров. Трудоёмкость алгоритма полного перебора — $O(n!)$.

2 Технологическая часть

В данном разделе приведены требования к программному обеспечению, средства реализации и листинги кода.

2.1 Средства реализации

Для реализации был выбран язык с++, так как он хорошо подходит для измерения скорости и в нем есть все необходимые инструменты.

2.2 Листинг кода

В листингах 2.1 – 2.3 приведены реализации алгоритмов.

Листинг 2.1 – Простой алгоритм

```
1 int main() {  
2     const int AIM = 100000;  
3     vector<int> prime_nums;  
4     prime_nums.push_back(2);  
5     for (int num = 3; prime_nums.size() < AIM; num += 1) {  
6         bool isprime = true;  
7         for (int i = 3; i < num; i += 1) {  
8             if (num % i == 0) {  
9                 isprime = false;  
10                break;  
11            }  
12        }  
13        if (isprime)  
14            prime_nums.push_back(num);  
15    }  
16 }
```

Листинг 2.2 – Улучшенный алгоритм

```
1 int main() {  
2     const int AIM = 100000;  
3     vector<int> prime_nums;  
4     prime_nums.push_back(2);
```

```

5     for (int num = 3; prime_nums.size() < AIM; num += 2) {
6         bool isprime = true;
7         for (int i = 3; i <= num / 2; i += 2) {
8             if (num % i == 0) {
9                 isprime = false;
10                break;
11            }
12        }
13        if (isprime)
14            prime_nums.push_back(num);
15    }
16 }

```

Листинг 2.3 – Улучшенный алгоритм

```

1  int main() {
2      const int AIM = 1000000;
3
4      int startSize = AIM;
5      int addSize = AIM;
6      vector<bool> nums(startSize);
7      vector<int> primeNums(AIM);
8
9      int foundPrimes = 0;
10     for (int i = 2; i < startSize; i++)
11         nums[i] = true;
12
13     bool addition = false;
14     int adder = 0;
15     while (true) {
16         if (addition) {
17             nums.resize(nums.size() + addSize, true);
18
19             for (int i = 0; i < foundPrimes; i++) {
20                 int cur_num = primeNums[i];
21                 if ((addSize + ((nums.size() - addSize) %
22                     cur_num)) < cur_num)
23                     continue;
24                 for (int j = ((nums.size() - addSize) / cur_num
25                     ) * cur_num; j < nums.size(); j += cur_num)
26                     nums[j] = false;
27             }
28         }
29         addition = true;
30         adder++;
31         addSize = adder * addSize;
32     }
33 }

```

```

26     }
27     else
28         addition = true;
29
30     int iter;
31     if (foundPrimes == 0)
32         iter = 2;
33     else
34         iter = primeNums[foundPrimes - 1] + 2;
35
36     for ( ; iter < nums.size(); iter++) {
37         if (nums[iter]) {
38             primeNums[foundPrimes] = iter;
39             foundPrimes++;
40             if (foundPrimes == AIM)
41                 break;
42             for (int j = iter + iter; j < nums.size(); j +=
43                 iter)
44                 nums[j] = false;
45         }
46         else
47             continue;
48     }
49     if (foundPrimes == AIM)
50         break;
51 }

```

3 Исследовательская часть

Результат 1 алгоритма

10 000 — 0,589 с.

100 000 — 72,662 с.

1 000 000 — 426,208 с.

Результат 2 алгоритма

10 000 — 0,356 с.

100 000 — 46,436 с.

1 000 000 — 238,354 с.

Результат 3 алгоритма

10 000 — 0,001 с.

100 000 — 0,021 с.

1 000 000 — 0,262 с.

Заключение

В ходе лабораторной работы были изучены и реализованы алгоритмы решения задачи нахождения простых чисел.

Были получены результаты работы.

Литература

- [1] Perl problems - Commis Voyageur [Электронный ресурс]. Режим доступа: <http://mech.math.msu.su/~shvetz/54/inf/perl-problems/chCommisVoyageur.xhtml> (дата обращения: 12.12.2020).