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»

Лабораторная работа №5
по дисциплине "Анализ Алгоритмов"

Тема Конвейерная обработка

Студент Сабуров С. М.

Группа ИУ7-53Б

Преподаватель Волкова Л. Л.

Москва

2021 г.

СОДЕРЖАНИЕ

Введение

На сегодняшний день в мире существует огромное количество задач, решение которых требует использования больших вычислительных мощностей, которые чаще всего недоступны в современных вычислительных системах. В связи с этим, возрастают требования к точности и к скорости решения таких задач. На помощь приходит создание параллельных вычислительных систем, в которых предусмотрена одновременная реализация ряда вычислительных процессов, связанных с решением одной задачи, что на сегодняшний день является одним из основных способов ускорения вычислений. При таком подходе увеличение производительности достигается путем параллельного выполнения частей более сложной задачи.

Целью данной работы является получение навыка реализации метода конвейерных вычислений.

Для достижения поставленной цели необходимо выполнить следующие задачи:

— описать методы обработки данных и сопоставить их с методами конвейера.
— привести схемы конвейера.
— реализовать конвейерную систему, описать данную реализацию.
— сравнить временные характеристики экспериментально.
— на основании проделанной работы сделать выводы.

1 Аналитическая часть

В данном разделе будет описана идея конвейерной обработки.

1.1 Конвейерная обработка

Конвейеризация – это техника, в результате которой задача или команда разбивается на некоторое число подзадач, которые выполняются последовательно. Каждая подкоманда выполняется на своем логическом устройстве. Все логические устройства (ступени) соединяются последовательно и работают одновременно. Множество ступеней называется конвейером. Выигрыш во времени достигается при выполнении нескольких задач за счет параллельной работы ступеней, вовлекая на каждом такте новую задачу или команду.

1.2 Область применения

Для данной лабораторной работы был разработан алгоритм шифрования строк, состоящий из трех этапов:

— шифрование методом цезаря.

— алгоритм, в котором к символу строки прибавляется сумма порядковых номеров символов всей строки из таблицы ASCII и порядковый номер символа в строке(индекс).

— алгоритм перестановки двух соседних символов.

.

1.3 Вывод

В данном разделе была поставлена задача реализации конвейерных вычислений . Была описана идея конвейеризации.

— Входные данные :количество слов в тексте.

— Выходные данные : на выходе имеем зашифрованный текст.

— Ограничения, в рамках которых будет работать программа : количество слов должно быть задано натуральным числом.

— Функциональные требования : функции, представленные на листингах 1 - 3 должны корректно шифровать исходный текст.
— Требования к программному обеспечению : к программе предъявляется ряд требований:
    — на вход подается количество слов, размер слова задан константой, программа сама генерирует слова, необязательно осмысленные;
    — на выходе - результат конвейеронго шифрования.

2 Конструкторская часть

В данном разделе будут приведены схемы алгоритмов, описание структур данных, способы тестирования и классы эквивалентности.

2.1 Разработка алгоритмов

На рисунках 1-2 приведены схемы работы конвейера и его составных частей.

2.2 Описание структур данных

Конвейер был преставлен в виде очереди. Текст хранился в виде массива строк.

2.3 Способы тестирования и классы эквивалентности

Была выбрана методика тестирования черным ящиком. Классы эквивалентности:

— Текст состоит из 1 строки.

— Текст состоит из 0 строк.

— Текст состоит из n строк, n - натуральное число.

— Задано количество строк - не натуральное число.

2.4 Вывод

На основе теоретических данных, полученных из аналитического раздела, были построены схемы требуемых алгоритмов, описаны стртуктуры данных, выделены способы тестирования и классы эквивалентности.

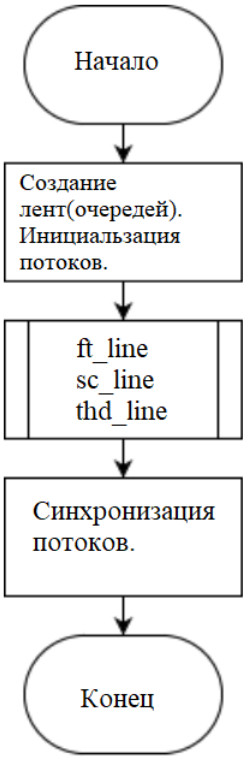

Рис. 1 — Схема запуска конвейера.

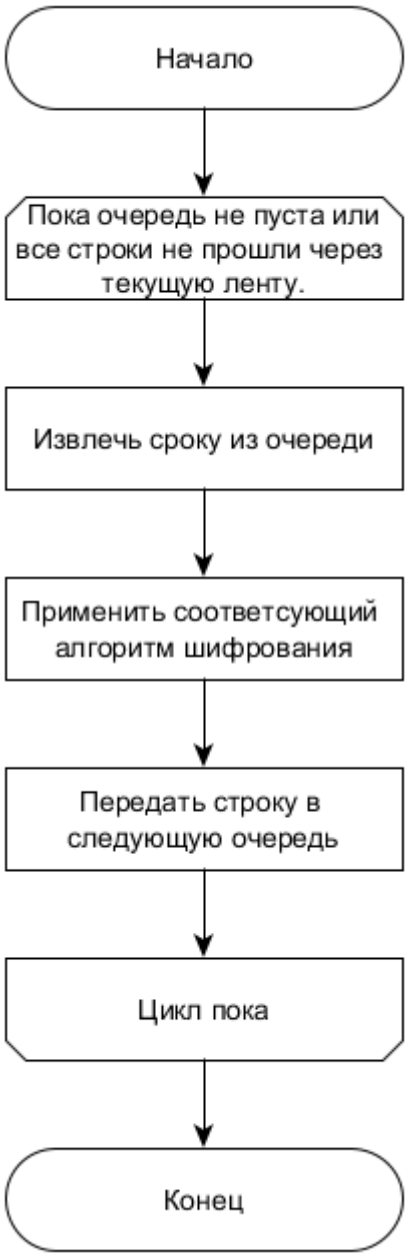

Рис. 2 — Схема работы ленты конвейера

3 Технологическая часть

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

3.1 Средства реализации

Для реализации программ был выбран язык программирования C++ [1]. Данный язык был выбран потому, что в нем присутствует инструментарий для замера процессорного времени и тестирования.

Листинг 1: Первый алгоритм шифрование: Метод цезаря

```
1  string encrypt_1(string s)
2  {
3
4          for (int i = 0; i < s.length(); i++)
5          {
6                  try
7                  {
8
9                          if (s[i] == 'z')
10                         {
11                                 s[i] = 'a';
12                         }
13                         else if (s[i] == 'Z')
14                                 s[i] = 'A';
15                         else
16                                 s[i] = s[i] + 1;
17                 }
18                 catch(...)
19                 {
20                         cout << "ENCRYPT 1" << s << endl;
21                 }
22
23         }
24         return s;
25 }
```

Листинг 2: Второй алгоритм шифрование: Пользовательский метод

```
string encrypt_2(string s)
{


        int sum = 0;
        for (int i = 0; i < s.length(); i++)
        {
                sum += s[i];
        }
        for (int i = 0; i < s.length(); i++)
        {
                try
                {

                        if (s[i] >= 'a' && s[i] <= 'z')
                        {
                                s[i] = (s[i] + sum + i) % ('z
    ' - 'a') + 'a';

                        }
                        else if (s[i] >= 'A' && s[i] <= 'Z')
                        {
                                s[i] = (s[i] + sum + i) % ('Z
    ' - 'A') + 'A';
                        }
                }
                catch (...)
                {
                        cout << "ENCRYPT 2 " << s << endl;
                }

        }
        return s;
}
```

Листинг 3: Третий алгоритм шифрование: Перестановка соседних символов

```
string encrypt_3(string s)
{
        if (s.length() > 0)
```

```
4           {
5
6                   for (int i = 0; i < s.length() - 1; i+= 2)
7                   {
8                           try
9                           {
10                                  char tmp = s[i];
11                                  s[i] = s[i + 1];
12                                  s[i + 1] = tmp;
13                          }
14                          catch (...)
15                          {
16                                  cout << "ENCRYPT 3" << s <<
    endl;
17                          }
18                  }
19          }
20          return s;
21  }
```

Листинг 4: Реализация первой ленты конвейера

```
1  void ft_line()
2  {
3
4           int num = 0;
5
6           while (true) {
7                   if (num == n)
8                           break;
9                   m1.lock();
10                  if (q1.empty()) {
11                          m1.unlock();
12                          continue;
13                  }
14                  string cur_str  = q1.front();
15                  q1.pop();
16
17                  m1.unlock();
18                  string new_str = encrypt_1(cur_str);
19                  m2.lock();
```

```
20
21                  q2.push(new_str);
22                  m2.unlock();
23                  num++;
24          }
25
26
27  }
```

Листинг 5: Реализация второй ленты конвейера

```
1  void sc_line()
2  {
3          int num = 0;
4          while (true) {
5                  if (num == n)
6                          break;
7                  m2.lock(); // wait in queue
8                  if (q2.empty()) {
9                          m2.unlock();
10                         continue;
11                 }
12                 string cur_str = q2.front();
13                 q2.pop();
14
15                 m2.unlock();
16                 string new_str = encrypt_2(cur_str);
17                 m3.lock();
18
19                 q3.push(new_str);
20                 m3.unlock();
21                 num++;
22         }
23 }
```

Листинг 6: Реализация третий ленты конвейера

```
1  void thd_line()
2  {
3          int num = 0;
4          while (true) {
5                  if (num == n)
```

```
6                           break;
7                   m3.lock(); // wait in queue
8                   if (q3.empty()) {
9                           m3.unlock();
10                          continue;
11                  }
12                  string cur_str = q3.front();
13                  q3.pop();
14
15                  m3.unlock();
16                  string new_str = encrypt_3(cur_str);
17                  resm.lock();
18
19                  q_final.push(new_str);
20                  resm.unlock();
21                  num++;
22          }
23  }
```

3.2 Тестирование функций

В таблице 1 приведены модульные тесты для функций шифрования текста. Все тесты были пройдены успешно.

Таблица 1 — Тестирование функций шифрования строки

Исходная строка	Ожидаемый результат
Hello	iDrqv
123	324
H	V
hello world	idrqv nnmrf
<Пустая строка>	Шифрование невозможно

3.3 Вывод

Были разработаны реализации алгоритмов шифрования и конвейеризации соотвественно. Также были протестированы реализации алгоритмов шифрования.

4 Исследовательская часть

В данном разделе будут приведены результаты исследовательской деятельности - замеры процессорного времени работы алгоритмов и тестирование алгоритмов.

4.1 Технические характеристики

Технические характеристики электронно-вычислительнй машины, на которой выполнялось тестирование:

— операционная система: Windows 10 64-bit;

— оперативная память: 8 гигабайт ;

— процессор: Intel i5 7th gen.

Тестирование проводилось на ноутбуке, включенном в сеть электропитания. Во время тестирования ноутбук был нагружен только встроенными приложениями окружения рабочего стола, окружением рабочего стола, а также непосредственно системой тестирования.

4.2 Время выполнения алгоритмов

Был проведен замер времени работы каждого из алгоритмов с помощью функции std::chrono::system clock::now. Эта функция замеряет процессорное время выполнения функции и усредняет его (проводится 10 замеров). В таблице 2 содержатся результаты исследований. В таблице 3 содержится лог программы.

На рисунке 3 демонстрируется зависимость времени выполнения последовательного и конвейерного алгоритмов от количества строк.

4.3 Вывод

В данном разделе было произведено сравнение количества затраченного времени вышеизложенных алгоритмов. Конвейерная реализация значительно выигрывает по времени в сравнении с линейной реализации. Как видно из рисунка

Таблица 2 — Время выполнения реализаций алгоритмов (в секундах) при количестве символов в строке 5000.

Кол-во строк	К	П
100	0.246199	0.555688
500	1.15154	2.75488
1000	2.36791	5.77721
2500	5.80938	14.0969
5000	11.7832	28.8174
10000	24.1511	55.4429

Таблица 3 — Лог программы при количестве строк 5.

Линия №	Строка №	Время начала (мс)	Время конца (мс)
1	0	7.227e+06	7.2795e+06
2	0	1.3869e+07	1.39594e+07
1	1	1.39611e+07	1.39689e+07
3	0	1.8941e+07	1.89564e+07
1	2	2.03209e+07	2.0327e+07
2	1	2.03243e+07	2.03323e+07
1	3	2.32028e+07	2.32085e+07
2	2	2.6129e+07	2.61349e+07
3	1	2.66033e+07	2.66153e+07
1	4	2.76257e+07	2.7632e+07
2	3	3.13649e+07	3.13702e+07
3	2	3.75349e+07	3.21479e+07
2	4	3.75349e+07	3.75402e+07
3	3	3.75711e+07	3.75798e+07
3	4	4.19849e+07	4.19913e+07

4, линейная реализация примерно в 2 раза медленнее параллельной при 10000 строках.

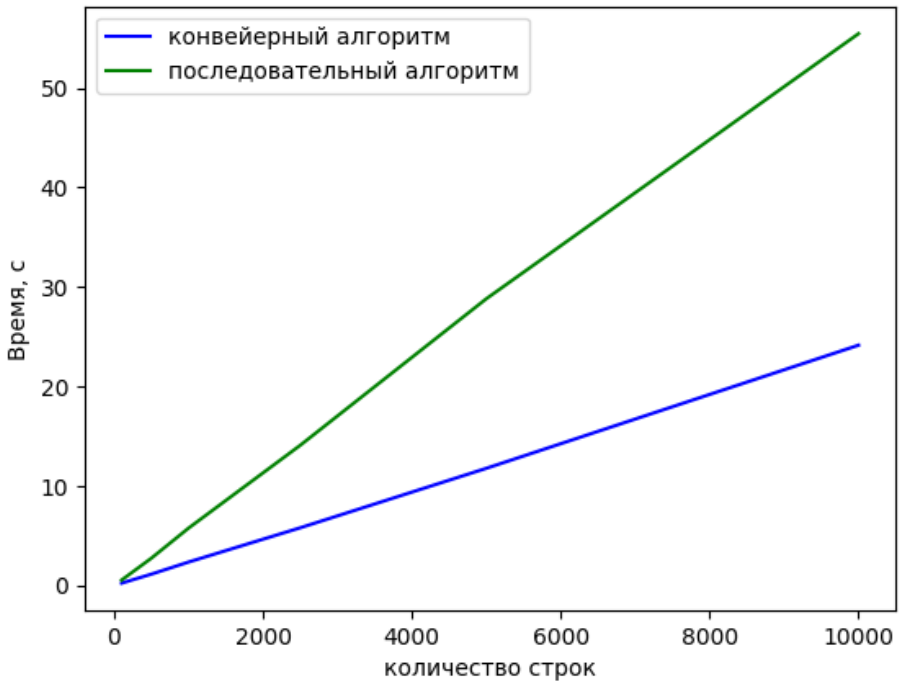

Рис. 3 — Зависимость времени выполнения алгоритмов от количества строк

Заключение

В ходе выполнения работы была достигнута цель выполнены все поставленные задачи:

— описать методы обработки данных и сопоставить их с методами конвейера.

— привести схемы конвейера.

— Реализовать конвейерную систему, описать данную реализацию

— сравнить временные характеристики экспериментально;

— на основании проделанной работы сделать выводы.

Экспериментально были установлены различия в производительности алгоритмов. Конвейерный алгоритм имеет большую эффективность, нежели линейный алгоритм.

Список литература

1. Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. Алгоритмы. Построение и анализ. Издательский дом “Вильямс”, 2011. 282 - 315.
2. Б. Страуструп. Язык программирования C++ . Addison-Wesley, 2000. 257 - 279.
3. Конвейерные вычисления [Электронный ресурс].Режим доступа: http://www.myshared.ru/slide/674082
4. Корнеев В.В. Параллельные вычислительные системы. М., 1999. 320 с.
5. Р. Седжвик. Фундаментальные алгоритмы C++. Diasoft, 2001. 567 - 597.