Министерство образования Республики Беларусь

Учреждение образования “Белорусский государственный технологический университет”

Кафедра полиграфического оборудования и систем обработки информации

**Лабораторная работа №4**

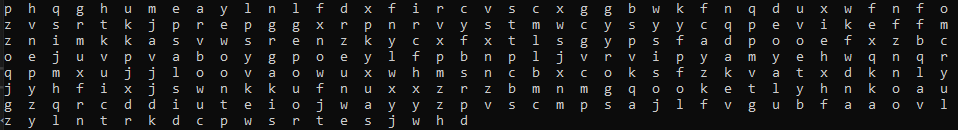
**Динамическое программирование**

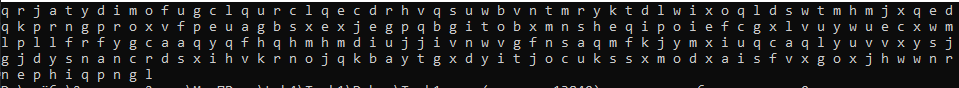
**Выполнил:** Грудинский Павел

**ЦЕЛЬ РАБОТЫ:** освоить общие принципы решения задач методом динамического программирования, сравнить полученные решения задач с рекурсивным методом.

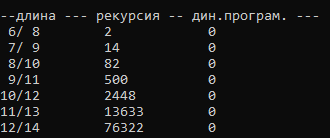
**Ход работы**

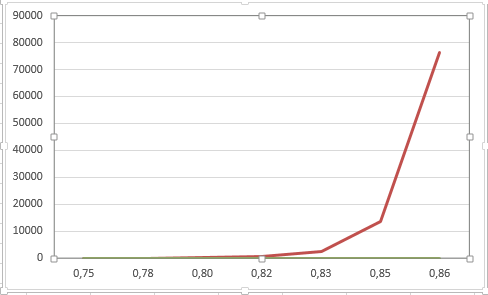
**Задание 1.** На языке С++ сгенерировать случайным образом строку букв латинского алфавита  длиной  символов и длиной символов.



 **Задание 2.** Вычислить двумя способами (рекурсивно и с помощью динамического программирования)  – дистанцию Левенштейна для , где - длина строки ,  - строка состоящая из первых  символов строки .

**Задание 3.** Выполнить сравнительный анализ времени, затраченного на вычисление дистанции Левенштейна для двух методов решения. Построить графики зависимости времени вычисления от . (копии экрана и график вставить в отчет).





**Задание 4.** Реализовать вручную пример вычисления дистанции Левенштейна при помощи рекурсивного алгоритма.



1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

*11.*

*12.*

*13.*

*14.*

*15.*

*16.*

*17.*

*18.*

*19.*

*20.*

*21.*

*22.*

*23.*

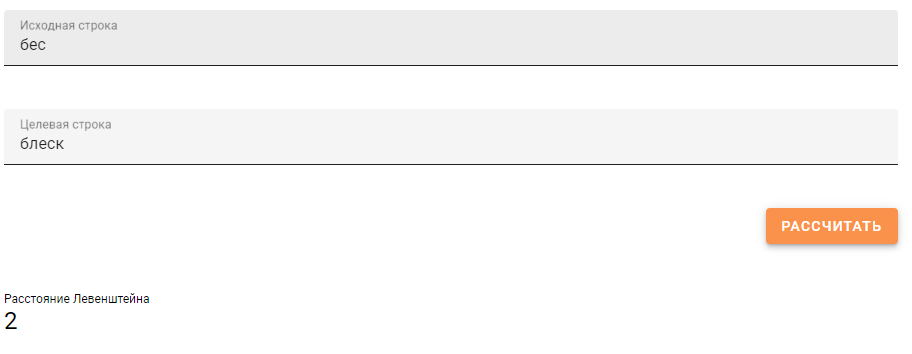
*24.*

*25.*

*26.*

*27.*

*28.*

**

**Задание 5. Нечетные варианты**. Выполнить сравнительный анализ времени затраченного на решение задачи о наибольшей общей подпоследовательности для двух методов решения (рекурсивное решение, динамическое программирование). Две последовательности взять в соответствии с вариантом. Построить графики зависимости времени вычисления от . **Отобразить ход решения в отчете** (по примеру из лекции) + код и копии экрана.

Вариант 1: ALBDACD и CDLDCA

Код программы.Source.cpp

|  |
| --- |
| #include <iostream>  #include "LCS.h"  #include "LCH.h"  #include <ctime>  int main2();  int main(int argc, char\* argv[])  {  setlocale(LC\_ALL, "rus");  char X[] = "ALBDACD", Y[] = "CDLDCA";  clock\_t t1 = 0, t2;  std::cout << std::endl << "-- вычисление длины LCS для X и Y(рекурсия)";  std::cout << std::endl << "-- последовательность X: " << X;  std::cout << std::endl << "-- последовательность Y: " << Y;  t1 = clock();  int s = lcs(  sizeof(X) - 1, // длина последовательности X  "ALBDACD", // последовательность X  sizeof(Y) - 1, // длина последовательности Y  "CDLDCA" // последовательность Y  );  t2 = clock();  std::cout << std::endl << "-- длина LCS: " << s << std::endl;  std::cout << "Time " << t2 - t1 << std::endl;  main2();  system("pause");  return 0;  }  int main2() {  char z[100] = "";  char x[] = "ALBDACD",  y[] = "CDLDCA";  clock\_t t3 = 0, t4;  t3 = clock();  int l = lcsd(x, y, z);  t4 = clock();  std::cout << std::endl  << "-- наибольшая общая подпоследовательость - LCS(динамическое"  << "программирование)" << std::endl;  std::cout << std::endl << "последовательость X: " << x;  std::cout << std::endl << "последовательость Y: " << y;  std::cout << std::endl << " LCS: " << z;  std::cout << std::endl << " длина LCS: " << l;  std::cout << std::endl << "Time :" << t4 - t3;  system("pause");  return 0;  } |

Файл LCS.cpp

|  |
| --- |
| #include "stdafx.h"  #include <algorithm>  #include "LCS.h"  int lcs(int lenx, const char x[],  int leny, const char y[])  {  int rc = 0;  if (lenx > 0 && leny > 0)  {  if (x[lenx - 1] == y[leny - 1]) rc = 1 + lcs(lenx - 1, x, leny - 1, y);  else rc = std::max(lcs(lenx, x, leny - 1, y), lcs(lenx - 1, x, leny, y));  }  return rc; //длина LCS  } |

Файл LCS.h

|  |
| --- |
| // -- рекурсивное вычисление длины LCS  int lcs(  int lenx, // длина последовательности X  const char x[], // последовательность X  int leny, // длина последовательности Y  const char y[] // последовательность Y  ); |

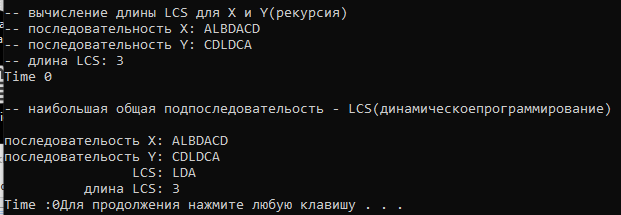
Файл LCH.cpp

|  |
| --- |
| //- LCS.cpp  #include "stdafx.h"  #include <cstring>  #include "LCS.h"  #define LCS\_C(x1,x2) (C[(x1)\*(leny+1)+(x2)])  #define LCS\_B(x1,x2) (B[(x1)\*(leny+1)+(x2)])  #define LCS\_X(i) (x[(i)-1])  #define LCS\_Y(i) (y[(i)-1])  #define LCS\_Z(i) (z[(i)-1])  enum Dart { TOP, LEFT, LEFTTOP };  void getLCScontent(int lenx, int leny, const char x[],  const Dart\* B,  int n, int i, int j, char z[])  {  if ((i > 0 && j > 0 && n > 0))  {  if (LCS\_B(i, j) == LEFTTOP)  {  getLCScontent(lenx, leny, x, B, n - 1, i - 1, j - 1, z);  LCS\_Z(n) = LCS\_X(i);  LCS\_Z(n + 1) = 0;  }  else if (LCS\_B(i, j) == TOP)  getLCScontent(lenx, leny, x, B, n, i - 1, j, z);  else getLCScontent(lenx, leny, x, B, n, i, j - 1, z);  }  };  int lcsd(const char x[], const char y[], char z[])  {  int n;  int lenx = strlen(x), leny = strlen(x),  \* C = new int[(lenx + 1) \* (leny + 1)];  Dart\* B = new Dart[(lenx + 1) \* (leny + 1)];  memset(C, 0, sizeof(int) \* (lenx + 1) \* (leny + 1));  for (int i = 1; i <= lenx; i++)  for (int j = 1; j <= leny; j++)  if (LCS\_X(i) == LCS\_Y(j))  {  LCS\_C(i, j) = LCS\_C(i - 1, j - 1) + 1;  LCS\_B(i, j) = LEFTTOP;  }  else if (LCS\_C(i - 1, j) >= LCS\_C(i, j - 1))  {  LCS\_C(i, j) = LCS\_C(i - 1, j);  LCS\_B(i, j) = TOP;  }  else  {  LCS\_C(i, j) = LCS\_C(i, j - 1);  LCS\_B(i, j) = LEFT;  }  getLCScontent(lenx, leny, x, B, LCS\_C(lenx, leny), lenx, leny, z);  return LCS\_C(lenx, leny);  }  #undef LCS\_Z  #undef LCS\_C  #undef LCS\_B  #undef LCS\_X  #undef LCS\_Y |

ыФайл LCH.h

|  |
| --- |
| //- LCH.h  int lcsd(  const char x[], // последовательность X  const char y[], // последовательность Y  char z[] // наибольшая общая подпоследовательность  ); |

Результат программы:



Время, затраченное на рекурсивный и динамический способ решения, одинаковое для данного варианта.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | C | D | L | D | C | A |
|  | *0* | *0* | *0* | *0* | *0* | *0* | *0* |
| A | *0* | *0* | *0* | *0* | *0* | *0* | *1* |
| L | *0* | *0* | *0* | *1* | *1* | *1* | *1* |
| B | *0* | *0* | *0* | *1* | *1* | *1* | *1* |
| D | *0* | *0* | *1* | *1* | *2* | *2* | *2* |
| A | *0* | *0* | *1* | *1* | *2* | *2* | *3* |
| C | *0* | *1* | *1* | *1* | *2* | *3* | *3* |
| D | *0* | *1* | *2* | *2* | *2* | *3* | *3* |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | C | D | L | D | C | A |
|  |  |  |  |  |  |  |  |
| A |  |  |  |  |  |  |  |
| L |  |  |  |  |  |  |  |
| B |  |  |  |  |  |  |  |
| D |  |  |  |  |  |  |  |
| A |  |  |  |  |  |  |  |
| C |  |  |  |  |  |  |  |
| D |  |  |  |  |  |  |  |