Учреждение образования

«Белорусский государственный технологический университет»

**Отчет по лабораторной работе №4**

**Тема «Динамическое программирование»**

Выполнил:

Студент 2 курса 7 группы ФИТ

Банкузов Михаил Олегович

Минск 2024

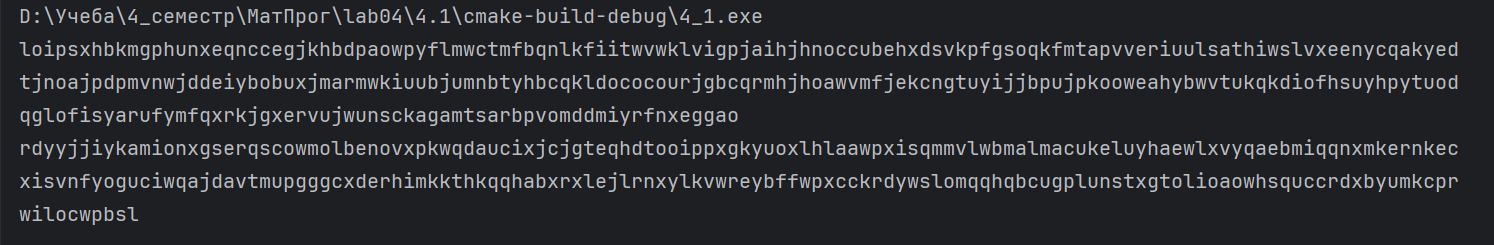
Цель работы: освоить общие принципы решения задач методом динамического программирования, сравнить полученные решения задач с рекурсивным методом.

**Задание 1.**

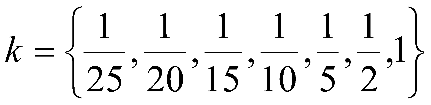
На языке С++ сгенерировать случайным образом строку букв латинского алфавита  длиной  символов и длиной .

#include <algorithm>  
#include <iostream>  
#include <ctime>  
#include <iomanip>  
#include "Levenshtein.h"  
  
int main()  
{  
 setlocale(LC\_ALL, "rus");  
 clock\_t t1 = 0, t2 = 0, t3, t4;  
 char x[301], y[251];  
  
 srand(time(0));  
  
 for (int i = 0; i < 300; i++)  
 {  
 x[i] = ('a' + rand() % ('z' - 'a'));  
 std::cout << x[i];  
 }  
  
 std::cout << std::endl;  
  
 for (int i = 0; i < 250; i++)  
 {  
 y[i] = ('a' + rand() % ('z' - 'a'));  
 std::cout << y[i];  
 }   
 system("pause");  
 return 0;  
}

Результат.



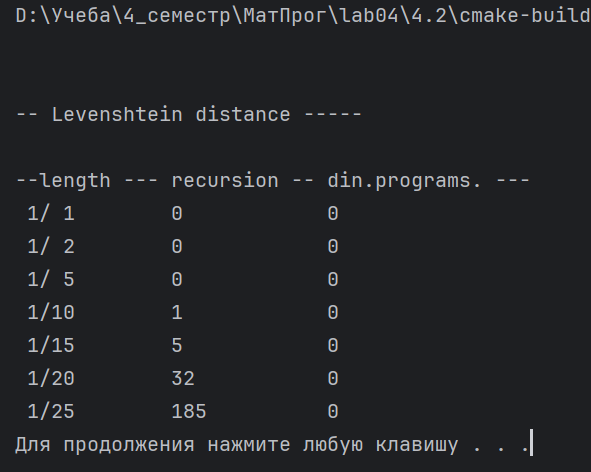
**Задание 2.**

Вычислить двумя способами (рекурсивно и с помощью динамического программирования)  – дистанцию Левенштейна для , где - длина строки ,  - строка, состоящая из первых  символов строки . (копии экрана и код вставить в отчет).

Код:

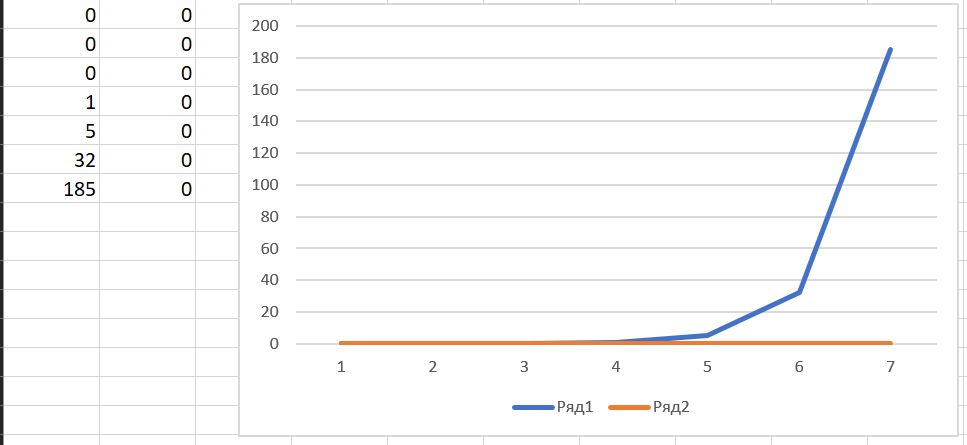
#include <algorithm>  
#include <iostream>  
#include <ctime>  
#include <iomanip>  
#include "Levenshtein.h"  
using namespace std;  
int main()  
{  
 setlocale(LC\_ALL, "rus");  
 clock\_t t1 = 0, t2 = 0, t3, t4;  
 char x[25] = "abcdefghdd", y[] = "xyabcdeshrtrh";  
 int lx = sizeof(x) - 1, ly = sizeof(y) - 1;  
 std::cout << std::endl;  
 std::cout << std::endl << "-- Levenshtein distance -----" << std::endl;  
 std::cout << std::endl << "--length --- recursion -- din.programs. ---"  
 << std::endl;  
 int u = 1;  
 for (int i = 6; i < std::min(lx, ly); i++)  
 {  
  
  
 t1 = clock(); levenshtein\_r(i, x, i - 2, y); t2 = clock();  
 t3 = clock(); levenshtein(i, x, i - 2, y); t4 = clock();  
  
 cout << right << setw(2) << 1 << "/" << setw(2) << u << " " << left << setw(10) << (t2 - t1) << " " << setw(10) << (t4 - t3) << endl;  
 if (u > 0) {  
  
 if (u == 20) { u = u + 5; }  
 if (u == 15) { u = u + 5; }  
 if (u == 10) { u = u + 5; }  
 if (u == 5) { u = u + 5; }  
 if (u == 2) { u = u + 3; }  
 if (u == 1) { u = u + 1; }  
 }  
  
 }  
 system("pause");  
 return 0;  
}

Результат:



**Задание 3.**

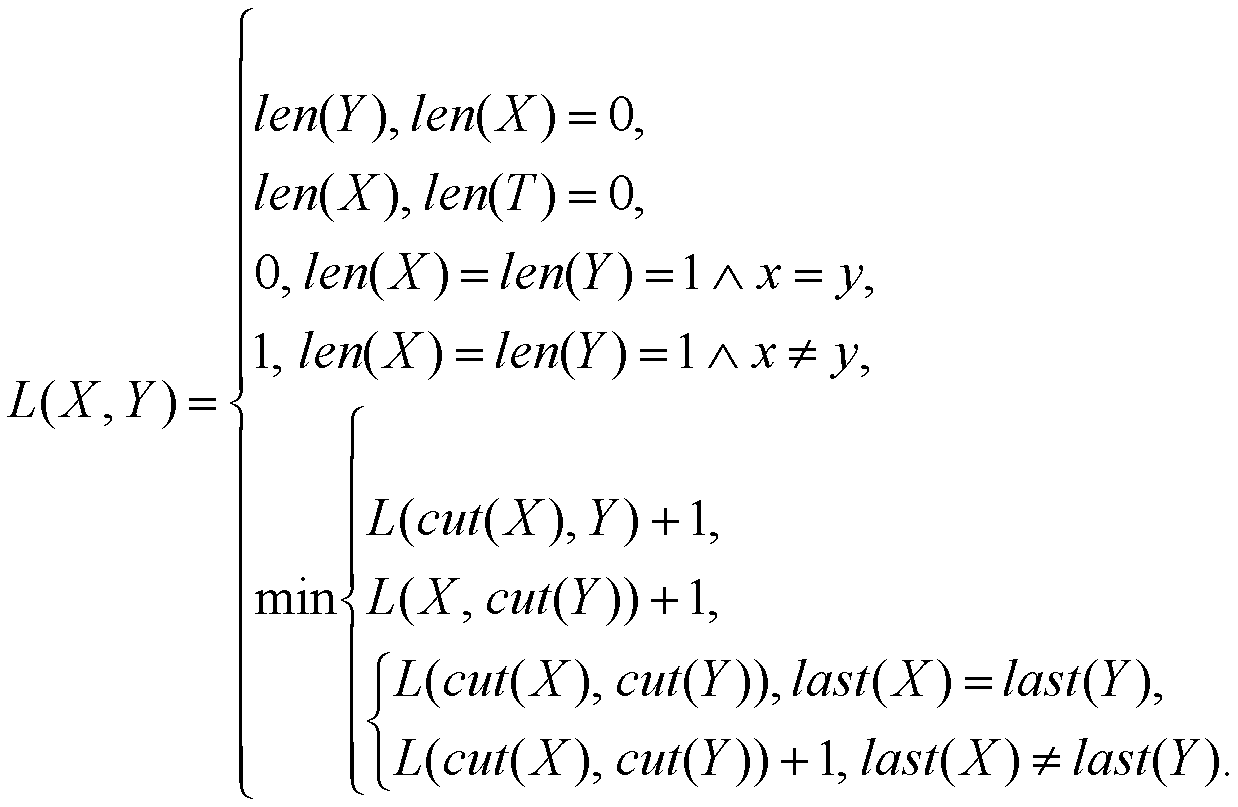
Выполнить сравнительный анализ времени, затраченного на вычисление дистанции Левенштейна для двух методов решения. Построить графики зависимости времени вычисления от . (копии экрана и график вставить в отчет).



**Задание 4.**

Реализовать вручную пример вычисления дистанции Левенштейна при помощи рекурсивного алгоритма (в соответствии с вариантом) (каждый шаг алгоритма по примеру из лекции вставить в отчет).

Пусть и  – две символьные строки, тогда для вычисления дистанции Левенштейна  между ними может быть использовано следующее рекуррентное соотношение:



В предыдущем выражении используются символы   и  Разъясним их смысл:

 – количество символов в заданной строке. Например, 

 – заданная строка без последнего символа. Например, 

 – последний символ заданной строки. Например, 

|  |  |  |
| --- | --- | --- |
| 1 | бес | блеск |



L (“”, “блеск”) = 5; L (“”, “блес”) = 4



L (“”, “блес”) = 4; L (“”, “бле”) = 3



L (“”, “бле”) = 3; L (“”, “бл”) = 2



L (“”, “бл”) = 2; L (“”, “б”) = 1



L (“бес”, “”) = 3; L (“бе”, “”) = 2



L (“бе”, “”) = 2; L (“б”, “”) = 1

1. L (“бе”, “б”) = min (1,3,2) = 1
2. L (“бес”, “б”) = min (2,4,3) = 2
3. L (“б”, “бл”) = min (3,1,2) = 1
4. L (“бе”, “бл”) = min (2,2,1) = 1
5. L (“бес”, “бл”) = min (2,3,2) = 2
6. L (“б”, “бле”) = min (4,2,3) = 2
7. L (“бе”, “бле”) = min (2,1,1) = 1
8. L (“бес”, “бле”) = min (2,3,2) = 2
9. L (“б”, “блес”) = min (5,3,4) = 3
10. L (“б”, “блеск”) = min (6,4,5) = 4
11. L (“бе”, “блес”) = min (5,2,3) = 2
12. L (“бес”, “блес”) = min (2,2,1) = 1
13. L (“бе”, “блеск”) = min (5,3,4) = 3
14. L (“бес”, “блеск”) = min (4,2,3) = 2

Следовательно, дистанция Левинштейна: 2

**Задание 5.**

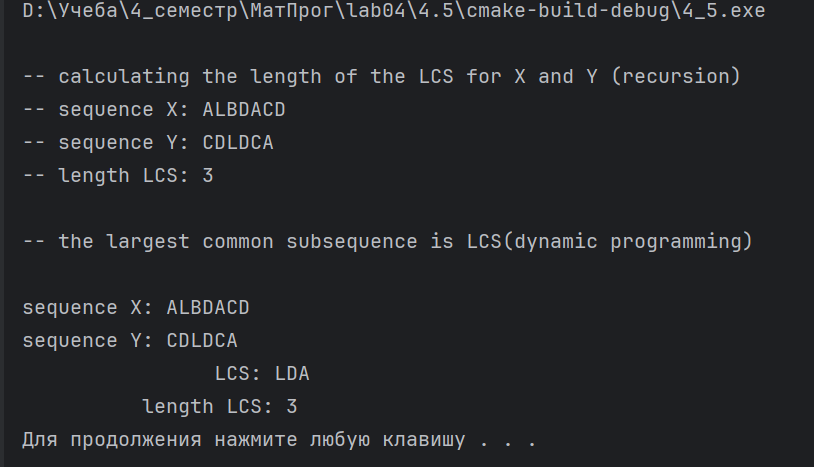
Нечетные варианты. Выполнить сравнительный анализ времени, затраченного на решение задачи о наибольшей общей подпоследовательности для двух методов решения (рекурсивное решение, динамическое программирование). Две последовательности взять в соответствии с вариантом. Построить графики зависимости времени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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | бес | блеск | ALBDACD | CDLDCA |

Код:

#include <algorithm>  
#include <iostream>  
#include <ctime>  
#include <iomanip>  
#include "LCD.h"  
  
#include <iostream>  
  
int main() {  
 setlocale(LC\_ALL, "rus");  
 char X[] = "ALBDACD", Y[] = "CDLDCA";  
 std::cout << std::endl << "-- calculating the length of the LCS for X and Y (recursion)";  
 std::cout << std::endl << "-- sequence X: " << X;  
 std::cout << std::endl << "-- sequence Y: " << Y;  
 int s = lcs(  
 sizeof(X) - 1, // длина последовательности X  
 "ALBDACD", // последовательность X  
 sizeof(Y) - 1, // длина последовательности Y  
 "CDLDCA" // последовательность Y  
 );  
 std::cout << std::endl << "-- length LCS: " << s << std::endl;  
  
  
 //DYNAMIC  
 char z[100] = "";  
 char x[] = "ALBDACD",  
 y[] = "CDLDCA";  
  
 int l = lcsd(x, y, z);  
 std::cout << std::endl  
 << "-- the largest common subsequence is LCS(dynamic"  
 << " programming)" << std::endl;  
 std::cout << std::endl << "sequence X: " << x;  
 std::cout << std::endl << "sequence Y: " << y;  
 std::cout << std::endl << " LCS: " << z;  
 std::cout << std::endl << " length LCS: " << l;  
 std::cout << std::endl;  
  
  
 system("pause");  
 return 0;  
}

Вывод.



X=A, L, B, D, A, C, D

Y=C, D, L, D, C, A

1. C (7, 6) = max (C 7,5; C 6,6) = max (3, 3) = 3
2. C (7, 5) = max (C 7,4; C 6,5) = max (3, 3) = 3
3. C (7, 4) = C 6,3 + 1
4. C (6, 6) = max (C 6,5; C 5,6) = max (3, 3) = 3
5. C (6, 5) = C 5,4 +1
6. C (5, 6) = C 4,5 +1
7. C (7, 3) = max (C 7,2; C6,3) = max (2, 2) = 2
8. C (7, 2) = C 6,1 + 1
9. C (6, 3) = max (C 6,2; C5,3) = max (2, 1) = 2
10. C (6, 2) = max (C 6,1; C5,2) = max (1, 1) = 1
11. C (6, 1) = C 5,0 + 1 = 1
12. C (5, 3) = max (C 5,2; C 4,1) = max (1, 0) = 1
13. C (5, 2) = max (C 5,1; C 4,2) = max (0, 1) = 1
14. C (5, 1) = max (C 5,0; C 4,1) = C 4,1
15. C (4, 1) = max (C 4,0; C 3,1) = C 3,1
16. C (4, 2) = C 3,1 + 1
17. C (3, 1) = max (C 3,0; C 2,1) = C 2,1
18. C (2, 1) = max (C 2,0; C 1,1) = C 1,1
19. C (1, 1) = max (C 1,0; C 0,1) = 0
20. C (2, 1) = max (C 2,0; C 1,1) = C 1,1
21. C (3, 1) = max (C 3,0; C 2,1) = C 2,1
22. C (4, 1) = max (C 4,0; C 3,1) = C 3,1
23. C (5, 1) = max (C 5,0; C 4,1) = C 4,1
24. C (6, 1) = max (C 6,0; C 5,1) = C 5,1
25. C (7, 1) = max (C 7,0; C 6,1) = C 6,1
26. C 4,5 = max (C 4,4; C 3,5) = max (2, 1) = 2
27. C 4,4 = C 3,3 + 1;
28. C 3,5 = max (C 3,4; C 2,5) = max (1, 1) = 1
29. C 3,4 = max (C 3,3; C 2,4) = max (1, 1) = 1
30. C 3,3 = max (C 3,2; C 2,3) = max (0, 1) = 1
31. C 3,2 = max (C 3,1; C 2,2) = max (0, 0) = 0
32. C 3,1 = max (C 3,0; C 2,1) = C 2,1
33. C 2,5 = max (C 2,4; C 1,5) = max (1, 0) = 1
34. C 2,4 = max (C 2,3; C 1,4) = max (1, 0) = 1
35. C 2,3 = C 1,2 + 1;
36. C 2,4 = max (C 2,3; C 1,4) = max (1, 0) = 1
37. C 2,2 = max (C 2,1; C 1,2) = max (0, 0) = 0
38. C 2,1 = max (C 2,0; C 1,1) = C 1,1
39. C 7,1 = max (C 7,0; C 6,1) = C 6,1
40. C 1,2 = max (C 1,1; C 0,2) = C 1,1
41. C 1,3 = max (C 1,2; C 0,3) = C 1,2
42. C 4,3 = max (C 4,2; C 3,3) = max (1, 1) = 2
43. C 1,4 = max (C 1,3; C 0,4) = C 1,3
44. C 5,4 = max (C 5,3; C 4,4) = max (1, 2) = 2
45. C 6,6 = max (C 6,5; C 5,6) = max (3, 3) = 3
46. C 4,5 = max (C 4,4; C 3,5) = max (2, 1) = 2
47. C 1,5 = max (C 1,4; C 0,5) = C 1,4
48. C 6,4 = max (C 6,3; C 5,4) = max (2, 2) = 2
49. C 5,5 = max (C 5,4; C 4,5) = max (2, 2) = 2
50. C 2,6 = max (C 2,5; C 1,6) = max (1, 1) = 1
51. C 3,6 = max (C 3,5; C 2,6) = max (1, 1) = 1
52. C 4,6 = max (C 4,5; C 3,6) = max (2, 1) = 2
53. C 5,6 = C 4,5 + 1
54. C 1,1 = 0
55. C 2,1 = 0
56. C 3,1 = 0
57. C 4,1 = 0
58. C 5,1 = 0
59. C 6,1 = 1
60. C 7,1 = 1
61. C 1,2 = 0
62. C 2,2 = 0
63. C 3,2 = 0
64. C 4,2 = 1
65. C 5,2 = 1
66. C 6,2 = 2
67. C 7,2 = 2
68. C 1,3 = 0
69. C 2,3 = 1
70. C 3,3 = 1
71. C 4,3 = 1
72. C 5,3 = 1
73. C 6,3 = 2
74. C 7,3 = 2
75. C 1,4 = 0
76. C 2,4 = 1
77. C 3,4 = 1
78. C 4,4 = 2
79. C 5,4 = 2
80. C 6,4 = 2
81. C 7,4 = 3
82. C 1,5 = 0
83. C 2,5 = 1
84. C 3,5 = 1
85. C 4,5 = 2
86. C 5,5 = 2
87. C 6,5 = 3
88. C 7,5 = 3
89. C 1,6 = 1
90. C 2,6 = 1
91. C 3,6 = 1
92. C 4,6 = 2
93. C 5,6 = 3
94. C 6,6 = 3
95. C 7,6 = 3

**Вывод.**

В ходе лабораторной работы были освоены методы динамического программирования. Полученные решения были сравнены с рекурсивным методом.