**Лабораторная работа 4**

**ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ**

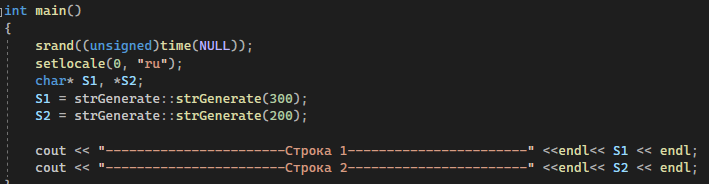
**ЦЕЛЬ РАБОТЫ:** освоить общие принципы решения задач методом динамического программирования, сравнить полученные решения задач с рекурсивным методом.

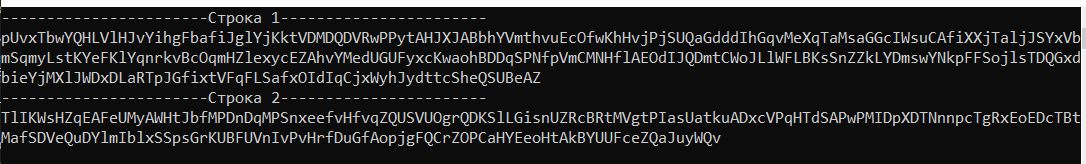
***Задание 1.***

Генератор случайных строк:



Генерирование строк S1(300 символов) и S2(200 символов):

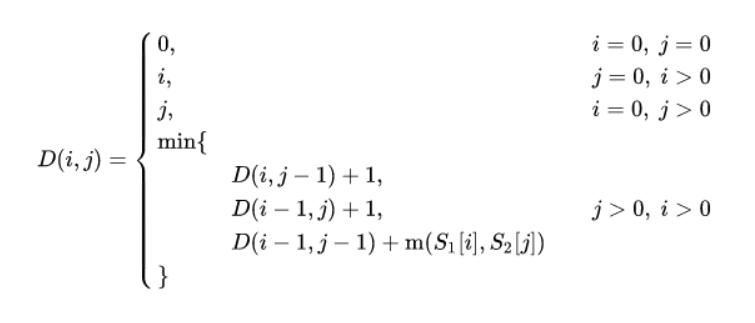




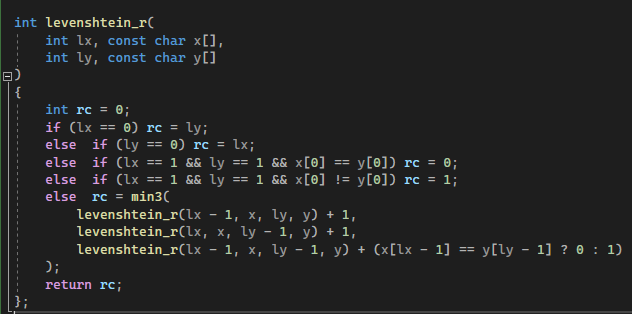
***Задание 2.***

Расстояние Левенштейна (редакционное расстояние, дистанция редактирования) — метрика, измеряющая по модулю разность между двумя последовательностями символов. Это минимальное число односимвольных преобразований (удаления, вставки или замены), необходимых, чтобы превратить одну последовательность в другую.

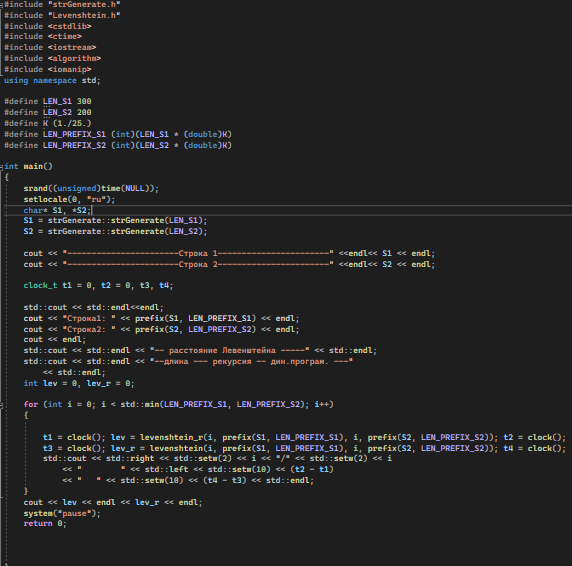
Вычисление каждого преобразования строки вычисляется по формуле Вагнера — Фишера:



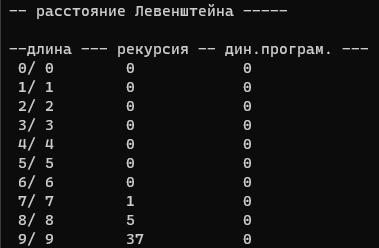
**Вычисление расстояния Левенштейна для строк S1 S2**



Реализация рекурсивного метода вычисления расстояния Левенштейна

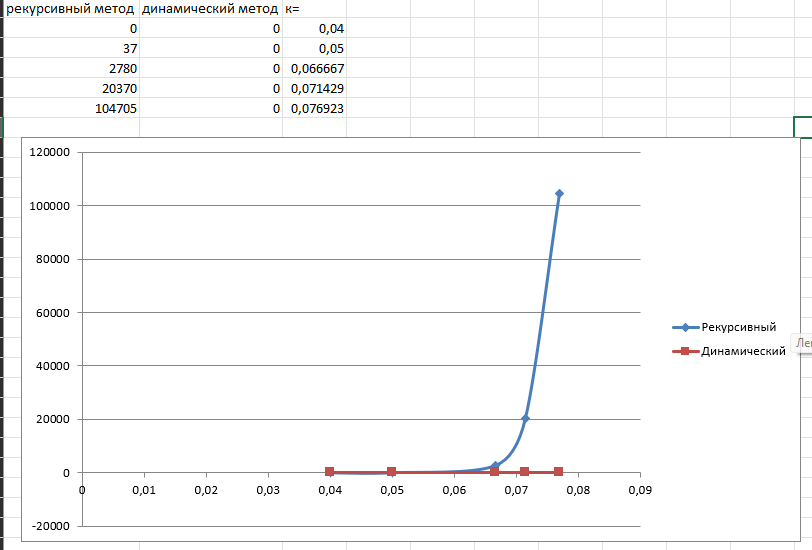


Вычисление



Результат для k = 1/20

***Задание 3.***



Из графиков видно, что рекурсивный метод крайне неэффективен, по сравнению с методом динамического программирования.

**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).

|  |  |
| --- | --- |
| Вариант 5 | |
| Сон | Донор |

1. Сон, Донор = min(Со, Донор+1)

( Сон, Доно+1)

(Со, Доно+1)

1. Со, Донор = min(С, Донор+1)

(Со,Доно+1)

(С, Доно+1)

1. Сон, Доно = min(Со, Доно+1)

(Сон, Дон+1)

(Со, Дон+1)

1. Со, Доно = min(С, Доно+1)

(Со, Дон+1)

(С, Дон)

1. С, Донор = min(« », Донор) = 5

(С, Доно+1)

(«», Доно) = 4

1. С, Доно = min («», Доно) = 4

(С, Дон+1)

(«», Дон) = 3

1. Сон, Дон = min(Со, Дон+1)

(Сон, До+1)

(Со, До)

1. Со, Дон = min(С, Дон+1)

(Со, До+1)

(С, До+1)

1. С, Дон= min(«», Дон) = 3

(С, До+1)

(«», До) = 2

1. Сон, До = min(Со, До+1)

(Сон, Д+1)

(Со, Д+1)

1. Со, До = min(С, До+1)

(Со, Д+1)

(С, Д)

1. С, До = min(«», До) = 2

(С, Д+1)

(«», Д) = 1

1. Сон, Д = min(Со, Д+1)

(Сон, «») = 3

(Со, «») = 2

1. Со, Д = min(С, Д+1)

(Со, «») = 2

(С, «») = 1

1. С,Д = min(«», Д) = 1

(С, «») = 1

(«», «») =0

1. С, Д = min (2, 2, 1) = 1
2. Со, Д = min (2, 3, 2) = 2
3. Сон, Д = min (3, 4, 3) = 3
4. С, До = min (3, 2, 2) = 2
5. Со, До = min (3, 3, 1) = 1
6. Сон, До = min (2, 4, 3) = 2
7. С, Дон = min (4, 3, 3) = 3
8. Со, Дон = min (4, 2, 3) = 2
9. Сон, Дон = min (3, 3, 1) = 1
10. С, Доно = min (5, 4, 4) = 4
11. С, Донор = min (6, 5, 5) = 5
12. Со, Доно = min (5, 3, 4) = 3
13. Сон, Доно = min (5, 4, 2) = 2
14. Со, Донор = min (6, 3, 4) = 3
15. Сон, Донор = min (4, 3, 4) = 3

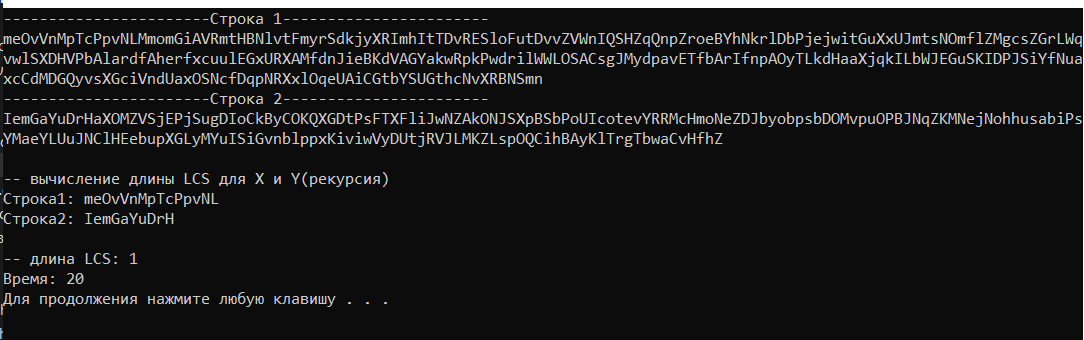
**Задание 5.**

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

|  |  |
| --- | --- |
| Вариант 5 | |
| HRWCYUJ | WLPCAU |

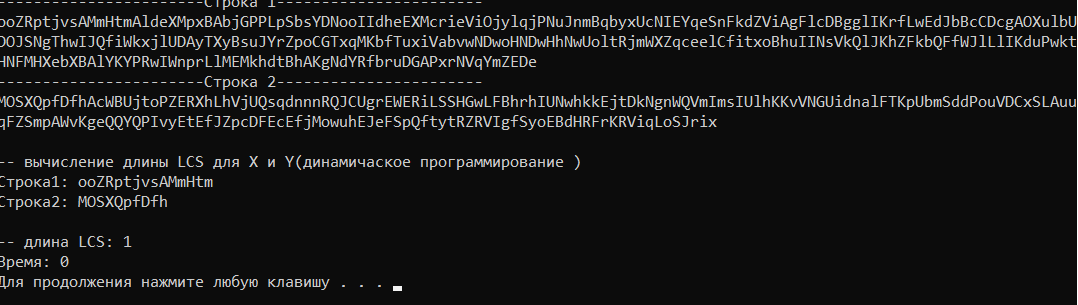
Рекурсия

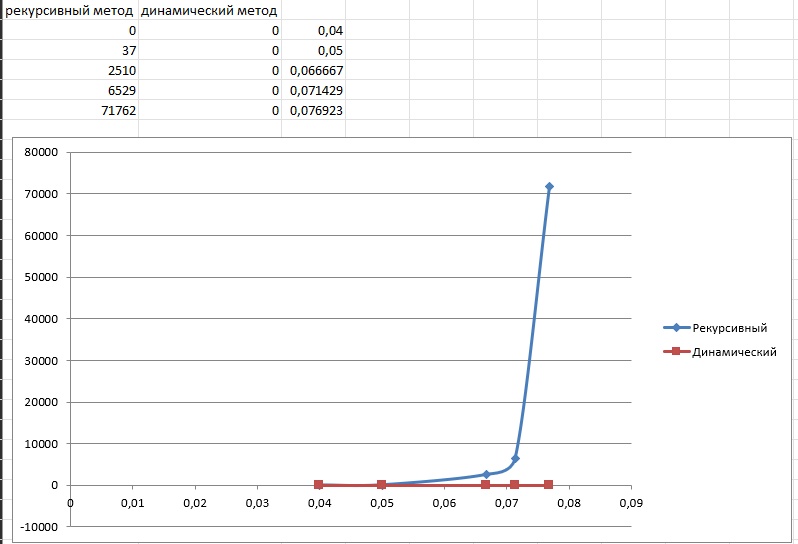
|  |
| --- |
| #include "strGenerate.h"  #include "Levenshtein.h"  #include <cstdlib>  #include <ctime>  #include <iostream>  #include <algorithm>  #include <iomanip>  using namespace std;  //  #define LEN\_S1 300  #define LEN\_S2 200  #define K (1./20.)  #define LEN\_PREFIX\_S1 (int)(LEN\_S1 \* (double)K)  #define LEN\_PREFIX\_S2 (int)(LEN\_S2 \* (double)K)  #include <iostream>  #include "LCS.h"  int main()  {  srand((unsigned)time(NULL));  setlocale(0, "ru");  char\* S1, \*S2;  S1 = strGenerate::strGenerate(LEN\_S1);  S2 = strGenerate::strGenerate(LEN\_S2);    cout << "-----------------------Строка 1-----------------------" <<endl<< S1 << endl;  cout << "-----------------------Строка 2-----------------------" <<endl<< S2 << endl;    std::cout << std::endl << "-- вычисление длины LCS для X и Y(рекурсия)\n";  cout << "Строка1: " << prefix(S1, LEN\_PREFIX\_S1) << endl;  cout << "Строка2: " << prefix(S2, LEN\_PREFIX\_S2) << endl;  clock\_t t1 = 0, t2 = 0;  t1 = clock();  int s = lcs(  LEN\_PREFIX\_S1, // длина последовательности X  prefix(S1, LEN\_PREFIX\_S1), // последовательность X  LEN\_PREFIX\_S2 - 1, // длина последовательности Y  prefix(S2, LEN\_PREFIX\_S2) // последовательность Y  );  t2 = clock();  std::cout << std::endl << "-- длина LCS: " << s << std::endl;  cout << "Время: " << t2 - t1 << endl;  system("pause");  return 0;  } |



Динамическое программирование

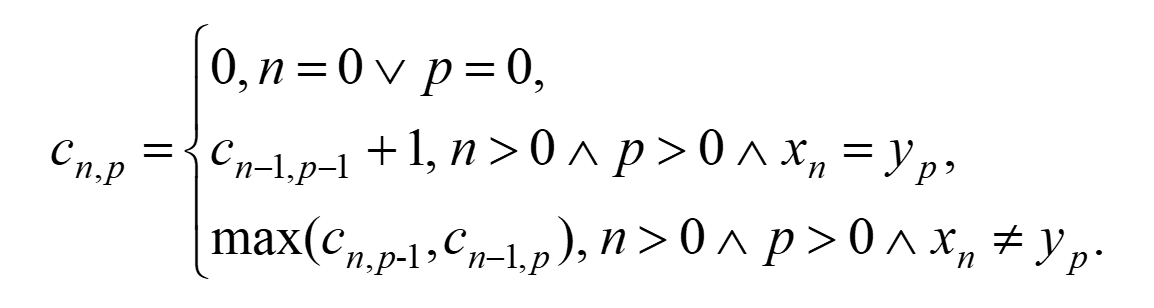
|  |
| --- |
| #include "strGenerate.h"  #include "Levenshtein.h"  #include <cstdlib>  #include <ctime>  #include <iostream>  #include <algorithm>  #include <iomanip>  using namespace std;  //  #define LEN\_S1 300  #define LEN\_S2 200  #define K (1./20.)  #define LEN\_PREFIX\_S1 (int)(LEN\_S1 \* (double)K)  #define LEN\_PREFIX\_S2 (int)(LEN\_S2 \* (double)K)  #include <iostream>  #include "LCH.h"  int main()  {  srand((unsigned)time(NULL));  setlocale(0, "ru");  char\* S1, \*S2;  S1 = strGenerate::strGenerate(LEN\_S1);  S2 = strGenerate::strGenerate(LEN\_S2);    cout << "-----------------------Строка 1-----------------------" <<endl<< S1 << endl;  cout << "-----------------------Строка 2-----------------------" <<endl<< S2 << endl;    std::cout << std::endl << "-- вычисление длины LCS для X и Y(динамичаское программирование )\n";  cout << "Строка1: " << prefix(S1, LEN\_PREFIX\_S1) << endl;  cout << "Строка2: " << prefix(S2, LEN\_PREFIX\_S2) << endl;  clock\_t t1 = 0, t2 = 0;  t1 = clock();  char z[100];  int s = lcsd(  prefix(S1, LEN\_PREFIX\_S1),  prefix(S2, LEN\_PREFIX\_S2),  z  );  t2 = clock();  std::cout << std::endl << "-- длина LCS: " << s << std::endl;  cout << "Время: " << t2 - t1 << endl;  system("pause");  return 0;  } |



График

Из графиков видно, что рекурсивный метод крайне неэффективен, по сравнению с методом динамического программирования.

Работа по вариантам



Вариант 5

|  |  |
| --- | --- |
| HRWCYUJ | WLPCAU |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | H -1 | R-2 | W - 3 | C - 4 | Y - 5 | U - 6 | J - 7 |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W -1 | 0 | ↑0 | ↑0 | ↖1 | ←1 | ←1 | ←1 | ←1 |
| L - 2 | 0 | ↖0 | ←0 | ↑1 | ←1 | ←1 | ←1 | ←1 |
| P - 3 | 0 | ↑0 | ←0 | ↑1 | ←1 | ←1 | ←1 | ↑1 |
| C - 4 | 0 | ↑0 | ←0 | ↑1 | ↖2 | ←2 | ←2 | ←2 |
| A - 5 | 0 | ↑0 | ←0 | ↑1 | ↑2 | ←2 | ←2 | ←2 |
| U- 6 | 0 | ↑0 | ←0 | ↑1 | ↑2 | ↑2 | ↖3 | ←3 |

A F I

1. C 7,6 = max (c 6,6, c 7,5)
2. C 6,6 = c 5,5 +1
3. C 7,5 = max (c 7,4, c 6,5)
4. C 5,5 = max (c 4,5, c 5,4)
5. C 7,4 = max (c 7,3, c 6,4)
6. C 6,5 = max (c 6,4, c 5,5)
7. C 4,5 = max (c 4,4, c 3,5)
8. C 5,4 = max (c 5,3, c 4,4)
9. C 7,3 = max (c 7,2, c 6,3)
10. C 6,4 = max (c 6,3, c 5,4)
11. C 4,4 = c 3,3 +1
12. C 3,5 = max (c 3,4, c 2,5)
13. C 5,3 = max (c 5,2, c 4,3)
14. C 7,2 = max (c 7,1, c 6,2)
15. C 6,3 = max (c 6,2, c 5,3)
16. C 3,3 = max (c 3,2, c 2,3)
17. C 3,4 = max (c 3,3, c 2,4)
18. C 2,5 = max (c 2,4, c 1,5)
19. C 5,2 = max (c 5,1, c 4,2)
20. C 4,3 = max (c 4,2, c 3,3)
21. C 7,1 = max (c 7,0, c 6,1)=c 6,1
22. C 6,2 = max (c 6,1, c 5,2)
23. C 5,3 = max (c 5,2, c 4,3)
24. C 3,2 = max (c 3,1, c 2,1)
25. C 2,3 = max (c 2,2, c 1,3)
26. C 3,3 = max (c 3,2, c 2,3)
27. C 2,4 = max (c 2,3, c 1,4)
28. C 6,4 = max (c 6,3, c 5,4)
29. C 1,5 = max (c 1,4, c 0,5)=c 1,4
30. C 1,4 = max (c 1,3, c 0,4)=c 1,3
31. C 1,3 = max (c 1,2, c 0,3)=c 1,2
32. C 1,2 = max (c 1,1, c 0,2)=c 1,1
33. C 1,1 = max (c 1,0, c 0,1)=c 1,0
34. C 5,1 = max (c 5,0, c 4,1)= c 4,1
35. C 4,2 = max (c 4,1, c 3,2)
36. C 6,1 = max (c 6,0, c 5,1)=c 5,1
37. C 3,1 =c 2,0+1
38. C 0,5 = 0
39. C 5,0 = 0
40. C 6,0 = 0
41. C 7,0 = 0
42. C 4,0 = 0
43. C 3,0 = 0
44. C 2,0 = 0
45. C 1,0 = 0
46. C 7,1 = max (c 7,0, c 6,1)=c 6,1
47. C 4,1 = max (c 4,0, c 3,1)=c 3,1
48. C 6,0 = 0
49. C 4,0 = 0
50. C3.1=0+1=1
51. C5.5=1+1=2
52. C7.6=2+1=3

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | H -1 | R-2 | W - 3 | C - 4 | Y - 5 | U - 6 | J - 7 |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W -1 | 0 | ↑0 | ↑0 | ↖1 | ←1 | ←1 | ←1 | ←1 |
| L - 2 | 0 | ↖0 | ←0 | ↑1 | ←1 | ←1 | ←1 | ←1 |
| P - 3 | 0 | ↑0 | ←0 | ↑1 | ←1 | ←1 | ←1 | ↑1 |
| C - 4 | 0 | ↑0 | ←0 | ↑1 | ↖2 | ←2 | ←2 | ←2 |
| A - 5 | 0 | ↑0 | ←0 | ↑1 | ↑2 | ←2 | ←2 | ←2 |
| U- 6 | 0 | ↑0 | ←0 | ↑1 | ↑2 | ↑2 | ↖3 | ←3 |

Вывод: в процессе выполнения были освоены общие приемы решения задач методом динамического программирования, были решены конкретные задачи на основе изученных приемов