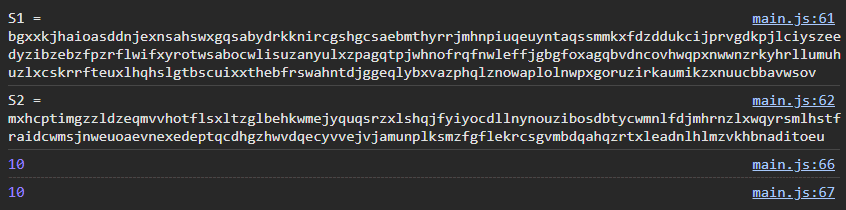
**№3**

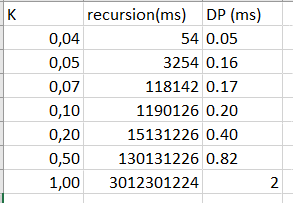
**Задание 1,2:**



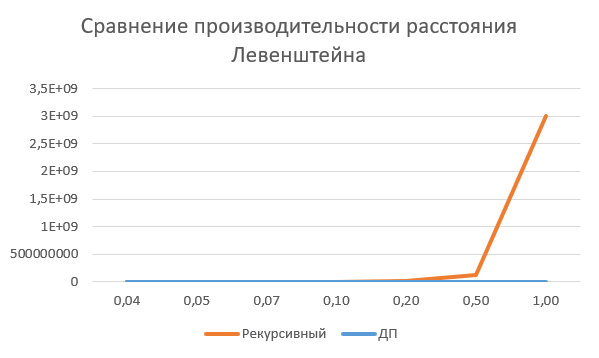
**Реализация**:

function recLevenshtein(s1, s2){  
 if (s1 === s2) {return 0;}  
 if (s1.length === 0) {return s2.length;}  
 if (s2.length === 0) {return s1.length;}  
  
 const cost = s1[s1.length - 1] === s2[s2.length - 1] ? 0 : 1;  
  
 return *Math*.min(  
 recLevenshtein(s1.slice(0, -1), s2) + 1, // Удаление  
 recLevenshtein(s1, s2.slice(0, -1)) + 1, // Вставка  
 recLevenshtein(s1.slice(0, -1), s2.slice(0, -1)) + cost // Замена  
 );  
}  
function levenshteinDP(s1, s2) {  
 const m = s1.length + 1;  
 const n = s2.length + 1;  
 const dp = new *Array*(m).fill(0).map(() => new *Array*(n).fill(0));  
  
 for (let i = 0; i < m; i++) {  
 dp[i][0] = i;  
 }  
  
 for (let j = 0; j < n; j++) {  
 dp[0][j] = j;  
 }  
  
 for (let i = 1; i < m; i++) {  
 for (let j = 1; j < n; j++) {  
 const cost = s1[i - 1] === s2[j - 1] ? 0 : 1;  
  
 dp[i][j] = *Math*.min(  
 dp[i - 1][j] + 1, // Удаление  
 dp[i][j - 1] + 1, // Вставка  
 dp[i - 1][j - 1] + cost // Замена  
 );  
 }  
 }  
  
 return dp[m - 1][n - 1];  
}

На основе измерений заполним таблицу данными, получим следующий результат:



Построим график для этих данных:



Я, как обладатель 2 варианта получил на разбор слова “том” и “исток”.

1. L(“том”, “исток”) = min (L(“то”, “исток”)+1, L(“том”, “исто”)+1, L(“то”, “исто”)+1)
2. L(“то”, “исток”) = min (L(“т”, “исток”)+1, L(“то”, “исто”)+1, L(“т”, “исто”)+1)
3. L(“том”, “исто”) = min (L(“то”, “исток”)+1, L(“том”, “ист”)+1, L(“то”, “исто”)+1)
4. L(“то”, “исток”) = min(L(“”, “исток”)+1, L(“т”, “исто”)+1, L(“т”, “ист”)+1)
5. L(“то”, “исто”) = min(L(“т”, “исто”)+1, L(“то”, “ист”)+1, L(“т”, “ист”)+1)
6. L(“то”, “сто”) = min(L(“”, “исто”)+1, L(“”, “исток”)+1, L(“”, “ист”)+1)
7. L(“т”, “то”) = min(L(“”, “исто”)+1, L(“”, “исток”)+1, L(“”, “ист”)+1)
8. L(“том”, “исто”) = min(L(“то”, “ист”)+1, L(“том”, “ис”)+1, L(“то”, “ист”)+1)
9. L(“то”, “сто”) = min(L(“т”, “ист”)+1, L(“то”, “ис”)+1, L(“т”, “ист”)+1)
10. L(“т”, “сто”) = min(L(“”, “ист”)+1, L(“т”, “ис”)+1, L(“”, “ист”)+1)
11. L(“”, “сто”) = min(L(“”, “ис”)+1, L(“”, “ист”)+1, L(“”, “ис”)+1)
12. L(“том”, “сто”) = min(L(“то”, “ис”)+1, L(“том”, “и”)+1, L(“то”, “ис”)+1)
13. L(“то”, “сто”) = min(L(“т”, “ис”)+1, L(“то”, “и”)+1, L(“т”, “ис”)+1)
14. L(“т”, “сто”) = min(L(“”, “ис”)+1, L(“т”, “и”)+1, L(“”, “ис”)+1)
15. L(“т”, “то”) = min(L(“”, “и”)+1, L(“”, “ис”)+1, L(“”, “и”)+1)
16. L(“т”, “то”) = 1
17. L(“то”, “сто”) = 1 - MIN
18. L(“том”, “сто”) = 2
19. L(“том”, “исто”) = 3
20. L(“то”, “исто”) = 2
21. L(“то”, “исток”) = 3
22. L(“то”, “исто”) = 2
23. L(“том”,”исто”) = 3
24. L(“том”,”исток”) = 4

**5) Матрицы**

#include "stdafx.h"

#include <cmath>

#include <memory.h>

#include <iostream>

#include "MultiMatrix.h" // умножение матриц

#define N 6

int main()

{

int Mc[N+1] = {5,10,15,20,25,30,35}, Ms[N][N], r = 0, rd = 0;

memset(Ms,0,sizeof(int)\*N\*N);

r = OptimalM(1, N, N, Mc, OPTIMALM\_PARM(Ms));

setlocale(LC\_ALL, "rus");

std::cout<<std::endl;

std::cout<<std::endl<< "-- расстановка скобок (рекурсивное решение) "

<< std::endl;

std::cout<<std::endl<< "размерности матриц: ";

for (int i = 1; i <= N; i++) std::cout<<"("<<Mc[i-1]<<","<<Mc[i]<<") ";

std::cout<<std::endl<< "минимальное количество операций умножения: " << r;

std::cout<<std::endl<<std::endl<<"матрица S"<<std::endl;

for (int i = 0; i < N; i++)

{

std::cout<<std::endl;

for (int j = 0; j < N; j++) std::cout<<Ms[i][j]<< " " ;

}

std::cout<<std::endl;

memset(Ms,0,sizeof(int)\*N\*N);

rd = OptimalMD(N, Mc, OPTIMALM\_PARM(Ms));

std::cout<<std::endl

<< "-- расстановка скобок (динамичеое программирование) "<< std::endl;

std::cout<<std::endl<< "размерности матриц: ";

for (int i = 1; i <= N; i++)

std::cout<<"("<<Mc[i-1]<<","<<Mc[i]<<") ";

std::cout<<std::endl<< "минимальное количество операций умножения: "

<< rd;

std::cout<<std::endl<<std::endl<<"матрица S"<<std::endl;

for (int i = 0; i < N; i++)

{

std::cout<<std::endl;

for (int j = 0; j < N; j++) std::cout<<Ms[i][j]<< " " ;

}

std::cout<<std::endl<<std::endl;

system("pause");

return 0;

}

**Результат:**

