|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №2**

**«Рекуррентные соотношения»**

**ДИСЦИПЛИНА: «Логика и теория алгоритмов»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-42Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Амеличев Г.Э. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2022

**Цель:** формирование практических навыков применения метода динамического программирования для построения эффективных алгоритмов решения задач оптимизации.

**Задачи:** реализовать рекурсивное решение задачи и исследовать его вычислительную сложность, реализовать алгоритм с использованием метода динамического программирования и исследовать его вычислительную сложность.

**Вариант №11**

**Наибольшая общая возрастающая подпоследовательность**

Вам даны две последовательности целых чисел. Напишите программу для определения их общей возрастающей подпоследовательности максимально возможной длины. Последовательность S1, S2, …, SN длины N называется возрастающей подпоследовательностью последовательности A1, A2, …, AM длины M, если существуют 1 ≤ i1 < i2 < … < iN ≤ M такие, что Sj = для всех 1 ≤ j ≤ N и Sj < Sj + 1 для всех 1≤ j < N. Каждая последовательность задаётся длиной M (1 ≤ M ≤ 500) и целыми числами Ai (-231 ≤ Ai ≤ 231) – членами последовательности. Например, дана последовательность длиной 5: {1, 4, 2, 5, -12} и длиной 4: {-12, 1, 2, 4}, тогда длина наибольшей общей подпоследовательности 2: {1, 4}.

**Оптимальное решение:**

Пусть даны последовательности X = {x1, …, xn} и Y = {y1, …, ym}, а Z = {z1, …, zk} – их наибольшая возрастающая подпоследовательность. Тогда если:

1. xn = ym ≥ zk-1, то zk = xn = ym и zk-1 = LCS(Xn-1, Ym-1), где Xn-1 = {x1, …, xn-1} и Ym-1 = {y1, …, ym-1}
2. xn ≠ ym или xn = ym < zk-1, то из zk ≠ xn следует, что z = LCS(Xn-1, Y)
3. xn ≠ ym или xn = ym < zk-1, то из zk ≠ ym следует, что z = LCS(X, Ym-1)

**Псевдокод рекурсивного алгоритма:**

если i < 0 или j < 0 c1

ну

прошлое\_число = минимальное\_число - 1 p1 \* c2

вернуть 0 p1 \* c3

ку

достигнуто = ложь (1 - p1) \* c4

если X[i] = Y[i] (1 - p1) \* c5

ну

длина = LCS(X, Y, i - 1, j - 1, c, s, число) + 1 (1 - p1) \* p2 \* c6 \* T(i - 1, j - 1)

если число <= X[i] (1 - p1) \* p2 \* c7

ну

c[i][j] = длина (1 - p1) \* p2 \* p3 \* c8

s[i][j] = \ (1 - p1) \* p2 \* p3 \* c9

прошлое\_число = число (1 - p1) \* p2 \* p3 \* c10

достигнуто = истина (1 - p1) \* p2 \* p3 \* c11

ку

ку

если не достигнуто (1 - p1) \* c12

ну

длина\_1 = LCS(X, Y, i - 1, j, c, s, число\_1) (1 - p1)\*(1 - p2 \* p3)\*c13 \* T(i - 1, j)

длина\_2 = LCS(X, Y, i, j - 1, c, s, число\_2) (1 - p1)\*(1 - p2 \* p3)\*c14 \* T(i, j - 1)

если длина\_1 >= длина\_2 (1 - p1) \* (1 - p2 \* p3) \* c15

ну

c[i][j] = длина\_1 (1 - p1) \* (1 - p2 \* p3) \* p4 \* c16

предыдущее\_число = число\_1 (1 - p1) \* (1 - p2 \* p3) \* p4 \* c17

s[i][j] = | (1 - p1) \* (1 - p2 \* p3) \* p4 \* c18

ку

иначе

ну

c[i][j] = длина\_2 (1 - p1) \* (1 - p2 \* p3) \* (1 - p4) \* c19

предыдущее\_число = число\_2 (1 - p1) \* (1 - p2 \* p3) \* (1 - p4) \* c20

s[i][j] = - (1 - p1) \* (1 - p2 \* p3) \* (1 - p4) \* c21

ку

ку

вернуть c[i][j] (1 - p1) \* c22

**Сложность рекурсивного алгоритма:**

A = (1 - p1)\*p2\*c6

B = (1 - p1)\*(1 - p2\*p3)\*c13

C = (1 - p1)\*(1 - p2\*p3)\*c14

D = с1 + p1\*c2 + p1\*c3

E = c1 + (1 - p1)\*c4 + (1 - p1)\*c5 + (1 - p1)\*p2\*c7 + (1 - p1)\*p2\*p3\*c8 + (1 - p1)\*p2\*p3\*c9 + (1 - p1)\*p2\*p3\*c10 + (1 - p1)\*p2\*p3\*c11 + (1 - p1)\*c12 + (1 - p1)\*c22

F = c1 + (1 - p1)\*c4 + (1 - p1)\*c5 + (1 - p1)\*c12 + (1 - p1)\*(1 - p2\*p3)\* c15 + (1 - p1)\*(1 - p2\*p3)\*p4\*c16 + (1 - p1)\*(1 - p2\*p3)\*p4\*c17 + (1 - p1)\*(1 - p2\*p3)\*p4\*c18 + (1 - p1)\*(1 - p2\*p3)\*(1 - p4)\*c19 + (1 - p1)\*(1 - p2\*p3)\*(1 - p4)\*c20 + (1 - p1)\*(1 - p2\*p3)\*(1 - p4)\*c21 + (1 - p1)\*c22

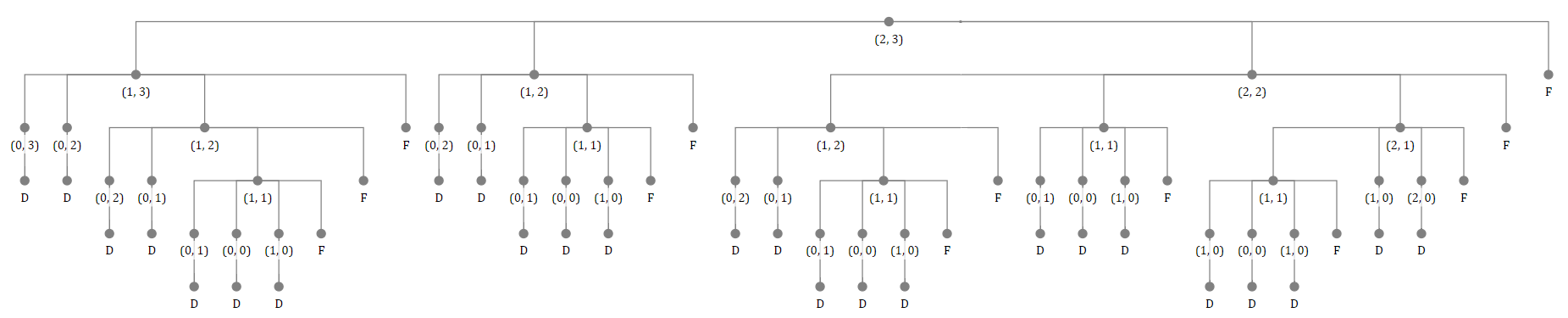


Рис. 1. Дерево рекурсии

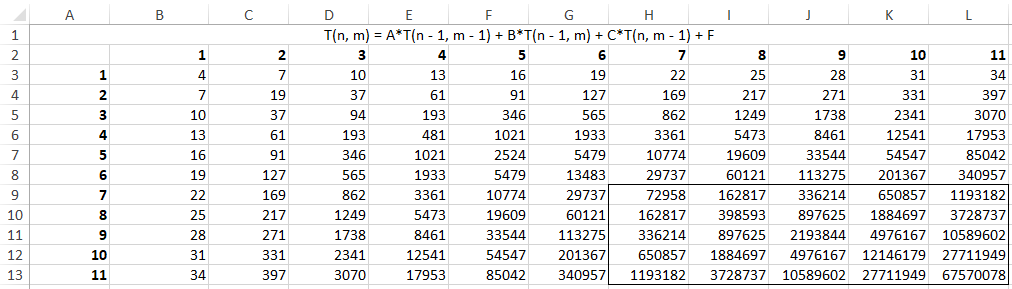


Рис. 2. Матричное представление

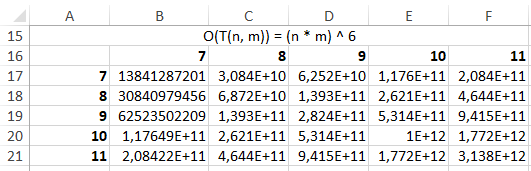


Рис. 3. Матрица сложности O(T(n, m)) = (n\*m)6

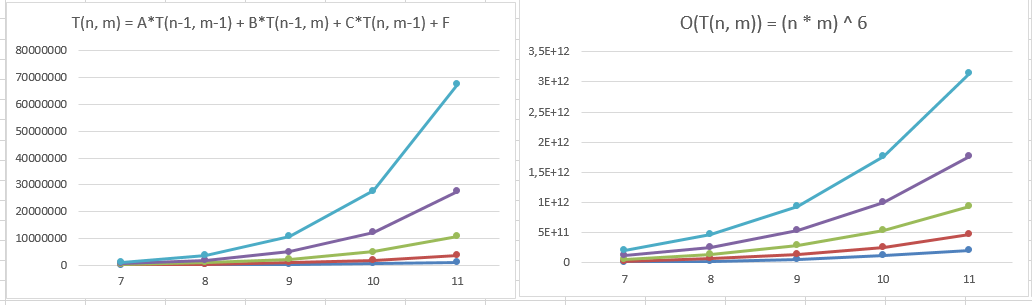


Рис. 4. Оценка сложности

Таким образом, O(T(n, m)) = (n\*m)6

**Листинг рекуррентного алгоритма:**

int LCS(vector<int> X, vector<int> Y, int i, int j,

int\*\*& c, char\*\*& s, int& prev)

{

if (i < 0 || j < 0)

{

prev = MinimumPossibleNumber - 1;

return 0;

}

bool isReached = false;

if (X[i] == Y[j])

{

int prevToReturn{};

int temp = LCS(X, Y, i - 1, j - 1, c, s, prevToReturn) + 1;

if (prevToReturn <= X[i])

{

c[i][j] = temp;

s[i][j] = '\\';

prev = X[i];

isReached = true;

}

}

if (!isReached)

{

int prev\_1 = prev;

int m\_1 = LCS(X, Y, i - 1, j, c, s, prev\_1);

int prev\_2 = prev;

int m\_2 = LCS(X, Y, i, j - 1, c, s, prev\_2);

if (m\_1 >= m\_2)

{

c[i][j] = m\_1;

prev = prev\_1;

s[i][j] = '|';

}

else

{

c[i][j] = m\_2;

prev = prev\_2;

s[i][j] = '-';

}

}

return c[i][j];

}

**Пример работы рекуррентного алгоритма:**

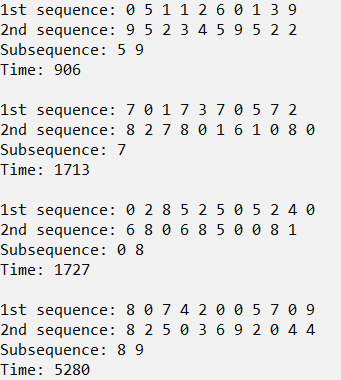


Рис. 5. Пример работы рекуррентного алгоритма

**Время выполнения рекурсивного алгоритма:**

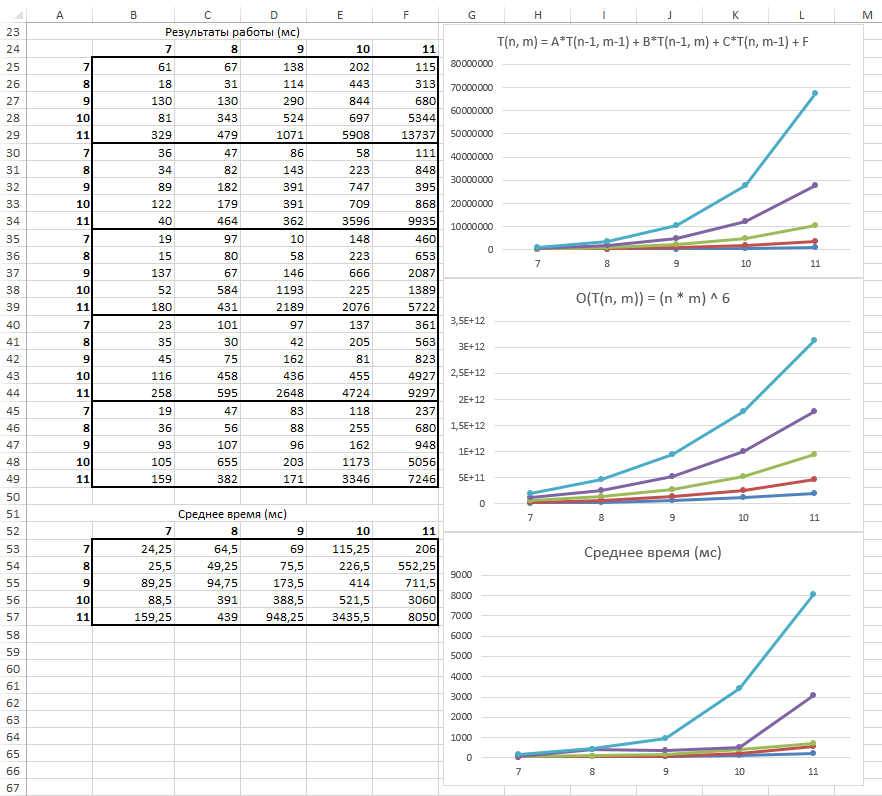


Рис. 6. Время выполнения рекуррентного алгоритма

**Псевдокод итеративного алгоритма:**

Если посмотреть на рис. 1, можно заметить, что T(1, 1), например, решается многократно. Таким образом, такое решение имеет свойство перекрывающихся подзадач, и имеет смысл составление итеративного алгоритма.

для i от 1 до n c1

нц

для j от 1 до m n \* c2

нц

если X[i] = Y[j] и v[i - 1][j - 1] <= X[i] m \* n \* c3

ну

c[i][j] = c[i - 1][j - 1] + 1 p1 \* m \* n \* c4

v[i][j] = X[i - 1] p1 \* m \* n \* c5

s[i][j] = \ p1 \* m \* n \* c6

ку

иначе если c[i - 1][j] >= c[i][j - 1] (1 - p1) \* m \* n \* c7

ну

c[i][j] = c[i - 1][j] (1 - p1) \* p2 \* m \* n \* c8

v[i][j] = v[i - 1][j] (1 - p1) \* p2 \* m \* n \* c9

s[i][j] = | (1 - p1) \* p2 \* m \* n \* c10

ку

иначе

ну

c[i][j] = c[i][j - 1] (1 - p1) \* (1 - p2)\*m\*n\*c11

v[i][j] = v[i][j - 1] (1 - p1) \* (1 - p2)\*m\*n\*c12

s[i][j] = - (1 - p1) \* (1 - p2)\*m\*n\*c13

ку

кц

кц

**Сложность итеративного алгоритма:**

O(T(n, m)) = O(c1 + n \* c2 + m \* n \* c3 + p1 \* m \* n \* c4 + p1 \* m \* n \* c5 + p1 \* m \* n \* c6 + (1 - p1) \* m \* n \* c7 + (1 - p1) \* p2 \* m \* n \* c8 + (1 - p1) \* p2 \* m \* n \* c9 + (1 - p1) \* p2 \* m \* n \* c10 + (1 - p1) \* (1 - p2) \* m \* n \* c11 + (1 - p1) \* (1 - p2) \* m \* n \* c12 + (1 - p1) \* (1 - p2) \* m \* n \* c13)

A = c3 + p1 \* c4 + p1 \* c5 + p1 \* c6 + (1 - p1) \* c7 + (1 - p1) \* p2 \* c8 + (1 - p1) \* p2 \* c9 + (1 - p1) \* p2 \* c10 + (1 - p1) \* (1 - p2) \* c11 + (1 - p1) \* (1 - p2) \* c12 + (1 - p1) \* (1 - p2) \* c13

B = c2

C = c1

O(T(n, m)) = O(A \* m \* n + B \* n + C) = m\*n

**Листинг итеративного алгоритма:**

void LCS(vector<int> X, vector<int> Y, char\*\*& s)

{

int n = X.size();

int m = Y.size();

int\*\* c = new int\* [n + 1]{};

int\*\* v = new int\* [n + 1]{};

for (int i{}; i < n + 1; ++i)

{

c[i] = new int[m + 1]{};

v[i] = new int[m + 1]{ MinimumPossibleNumber - 1 };

}

for (int i{}; i < m; ++i)

v[0][i + 1] = MinimumPossibleNumber - 1;

s = new char\* [n] {};

for (int i{}; i < n; ++i)

s[i] = new char[m] {};

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

for (int j = 1; j < m + 1; ++j)

{

if (X[i - 1] == Y[j - 1] && v[i - 1][j - 1] <= X[i - 1])

{

c[i][j] = c[i - 1][j - 1] + 1;

v[i][j] = X[i - 1];

s[i - 1][j - 1] = '\\';

}

else if (c[i - 1][j] >= c[i][j - 1])

{

c[i][j] = c[i - 1][j];

v[i][j] = v[i - 1][j];

s[i - 1][j - 1] = '|';

}

else

{

c[i][j] = c[i][j - 1];

v[i][j] = v[i][j - 1];

s[i - 1][j - 1] = '-';

}

}

for (int i{}; i < n + 1; ++i)

{

delete[] c[i];

delete[] v[i];

}

delete[] c;

delete[] v;

}

**Пример работы итеративного алгоритма:**

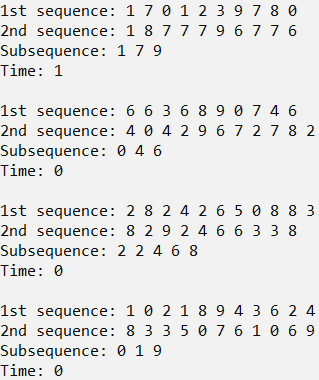


Рис. 7. Пример работы итеративного алгоритма

**Время выполнения итеративного алгоритма:**

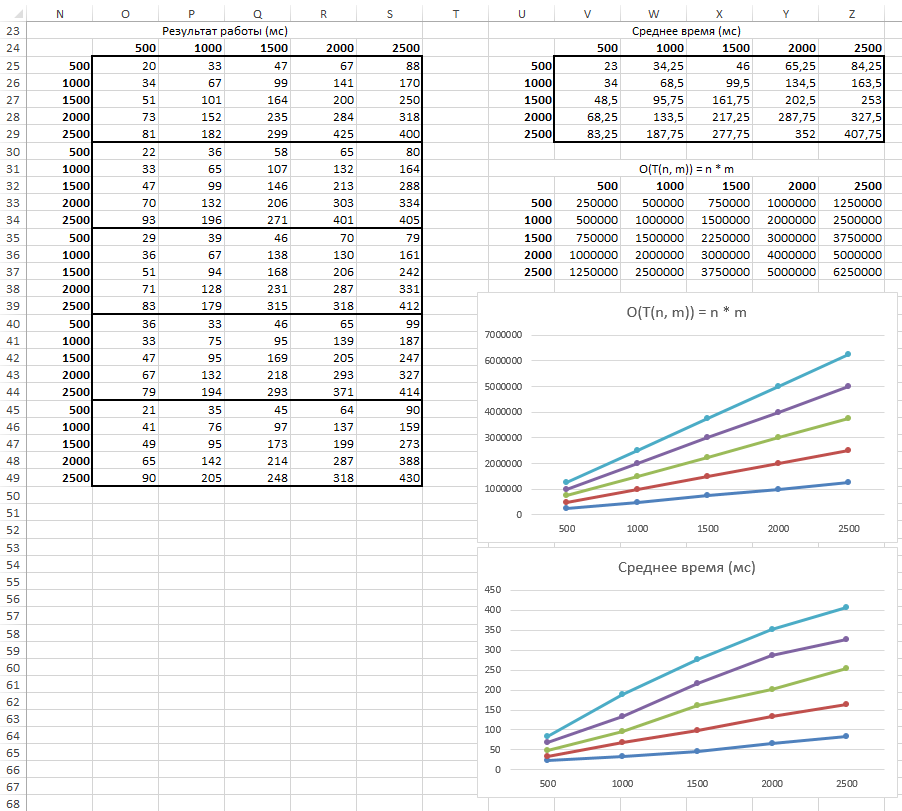


Рис. 8. Время выполнения итеративного алгоритма

**Сравнение алгоритмов:**

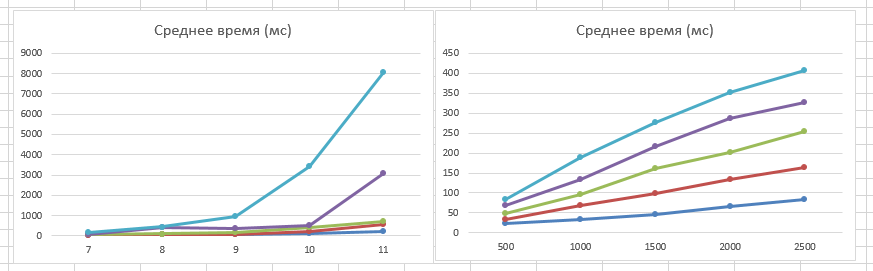


Рис. 9. Сравнение рекуррентного и итеративного алгоритмов

Исходя из расчетов и полученных результатов, можно сделать вывод, что итеративный алгоритм работает значительно быстрее, чем рекуррентный.

**Вывод:** в ходе выполнения лабораторной работы были получены практические навыки применения метода динамического программирования, составления рекуррентных и итеративных алгоритмов и расчета их сложностей.