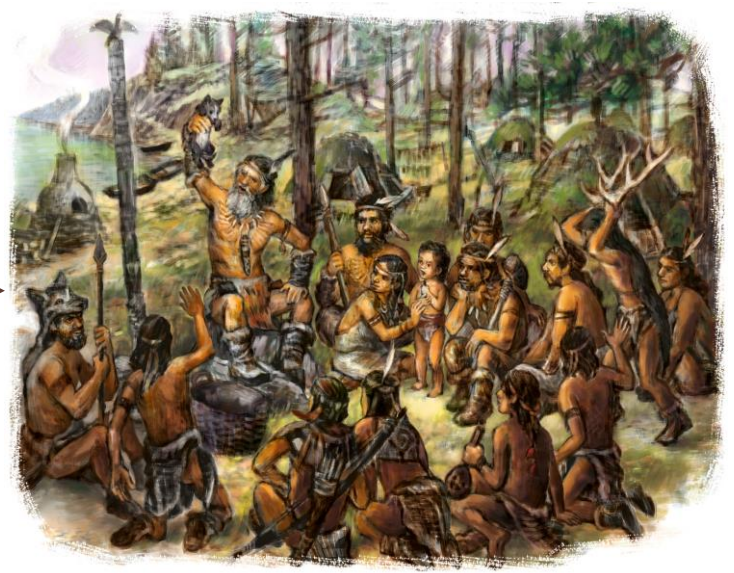
Динамическое программирование
13.12.2022
ИТМО ИС

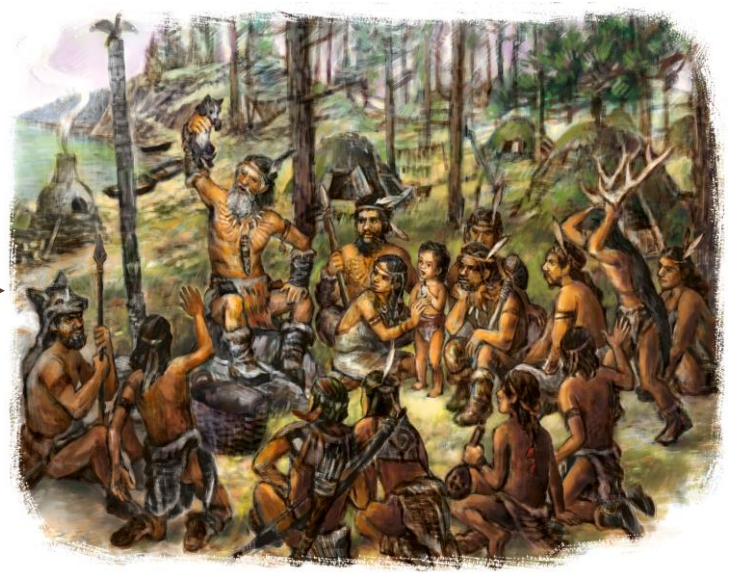
Задача о редакционном расстоянии

ПОЛЕНО



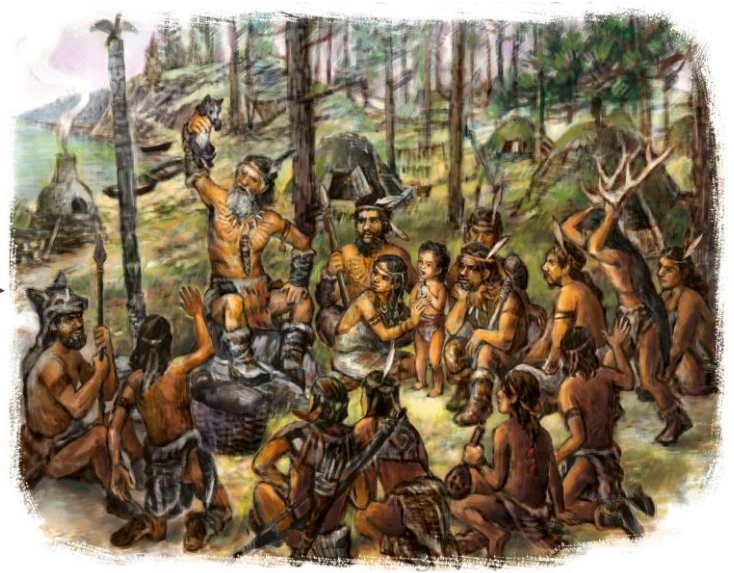
ПЛЕМЯ

Задача о редакционном расстоянии



ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

Задача о редакционном расстоянии



ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

Задача: Имеем два слова. Хотим из первого получить второе.
Можем делать три операции: удалять букву, вставлять букву, изменять букву.

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

Задача: Имеем два слова. Хотим из первого получить второе.
Можем делать три операции: удалять букву, вставлять букву, изменять букву.

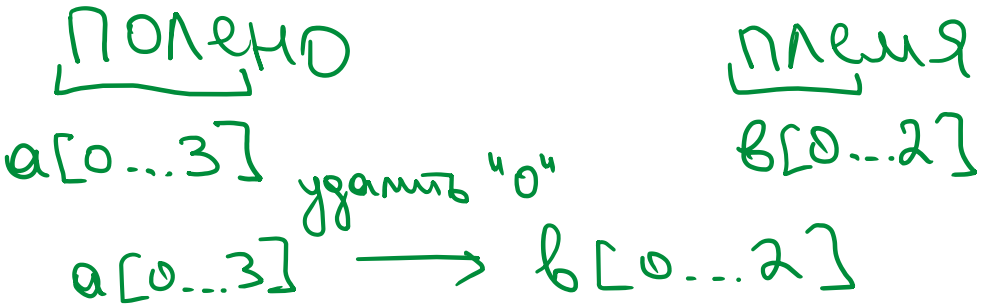
Динамика?

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

Задача: Имеем два слова. Хотим из первого получить второе.
Можем делать три операции: удалять букву, вставлять букву, изменять букву.

Будем пользоваться **принципом оптимальности на префиксе**:
сколько надо сделать изменений, чтобы получить из префикса первого слова префикс второго?



Задача о редакционном расстоянии

Задача: Имеем два слова. Хотим из первого получить второе.
Можем делать три операции: удалять букву, вставлять букву, изменять букву.

Будем пользоваться **принципом оптимальности на префиксе**:
сколько надо сделать изменений, чтобы получить из префикса первого слова префикс второго?

Что тогда будем хранить в d[i][j]?

Задача о редакционном расстоянии

Задача: Имеем два слова. Хотим из первого получить второе.
Можем делать три операции: удалять букву, вставлять букву, изменять букву.

Будем пользоваться **принципом оптимальности на префиксе**:
сколько надо сделать изменений, чтобы получить из префикса первого слова префикс второго?

d[i][j] – стоимость получения из префикса длины i первого слова префикс длины j второго слова

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

Будем пользоваться **принципом оптимальности на префиксе**:
сколько надо сделать изменений, чтобы получить из префикса первого слова префикс второго?

d[i][j] – стоимость получения из префикса длины i первого слова префикс длины j второго слова

вставка:

ав → авс ← хотим получить префикс второго слова b[0...2]

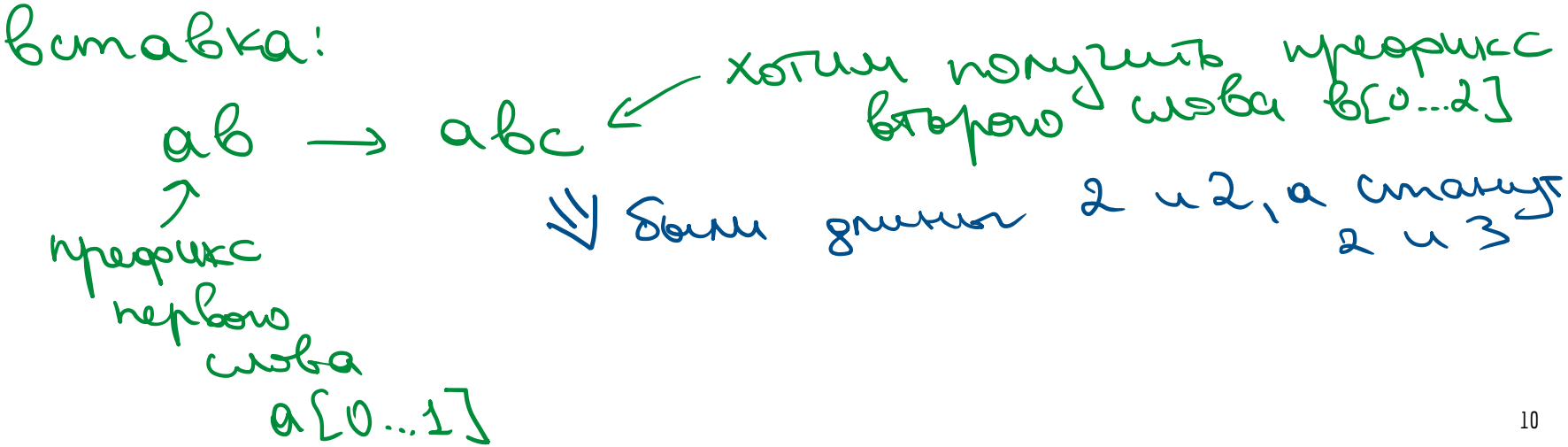
↑ префикс первого слова a[0...1]

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

Будем пользоваться **принципом оптимальности на префиксе**:
сколько надо сделать изменений, чтобы получить из префикса первого слова префикс второго?

d[i][j] – стоимость получения из префикса длины i первого слова префикс длины j второго слова



Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

Будем пользоваться **принципом оптимальности на префиксе**:
сколько надо сделать изменений, чтобы получить из префикса первого слова префикс второго?

d[i][j] – стоимость получения из префикса длины i первого слова префикс длины j второго слова

вставка:

ав → авс ← хотим получить префикс второго слова b[0...2]

⇓ были длины 2 и 2, а стали 2 и 3

префикс первого слова a[0...1]

тогда в динамике продвинемся из d[2][2] в d[2][3]

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

Будем пользоваться **принципом оптимальности на префиксе**:
сколько надо сделать изменений, чтобы получить из префикса первого слова префикс второго?

d[i][j] – стоимость получения из префикса длины i первого слова префикс длины j второго слова

вставка:

ав → авс ← хотим получить префикс второго слова b[0...2]

префикс первого слова a[0...1]

Первое правило (insert): d[i][j] = d[i][j-1] + 1

тогда в динамике продвинемся из d[2][2] в d[2][3]

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

Будем пользоваться **принципом оптимальности на префиксе**:
сколько надо сделать изменений, чтобы получить из префикса первого слова префикс второго?

d[i][j] – стоимость получения из префикса длины i первого слова префикс длины j второго слова

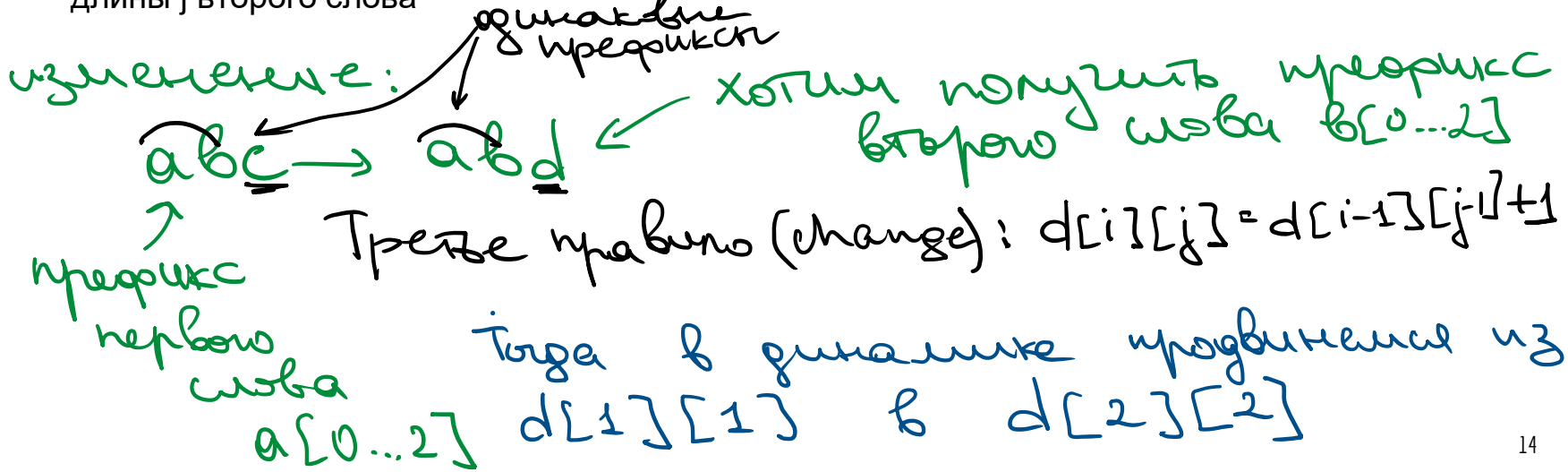
удаление:
abc → ab
хотим получить префикс второго слова b[0...1]
префикс первого слова a[0...2]
Второе правило (delete): d[i][j] = d[i-1][j] +1
тогда в динамике продвинемся из d[3][2] в d[2][2]

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

Будем пользоваться **принципом оптимальности на префиксе**:
сколько надо сделать изменений, чтобы получить из префикса первого слова префикс второго?

d[i][j] – стоимость получения из префикса длины i первого слова префикс длины j второго слова



Задача о редакционном расстоянии

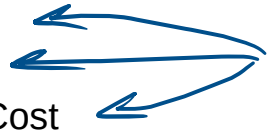
ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

Будем пользоваться **принципом оптимальности на префиксе**:
сколько надо сделать изменений, чтобы получить из префикса первого слова префикс второго?

d[i][j] – стоимость получения из префикса длины i первого слова префикс длины j второго слова

Итого три правила:

1) Delete: d[i][j] = d[i-1][j] + deleteCost
2) Insert: d[i][j] = d[i][j-1] + insertCost
3) Change: d[i][j] = d[i-1][j-1] + changeCost



эти числа могут быть ∀, в общем случае

Задача о редакционном расстоянии

Будем пользоваться **принципом оптимальности на префиксе**:
сколько надо сделать изменений, чтобы получить из префикса первого слова префикс второго?

d[i][j] – стоимость получения из префикса длины i первого слова префикс длины j второго слова

Итого три правила:

1) Delete: d[i][j] = d[i-1][j] + deleteCost
2) Insert: d[i][j] = d[i][j-1] + insertCost
3) Change: d[i][j] = d[i-1][j-1] + changeCost

Из этих трех возможностей хотим ту, что имеет минимальную стоимость

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

Будем пользоваться **принципом оптимальности на префиксе**:
сколько надо сделать изменений, чтобы получить из префикса первого слова префикс второго?

d[i][j] – стоимость получения из префикса длины i первого слова префикс длины j второго слова

Из этих трех возможностей хотим ту, что имеет минимальную стоимость

$$D(i,j) = \begin{cases} 0 & ;\ i = 0,\ j = 0 \\ i * deleteCost & ;\ j = 0,\ i > 0 \\ j * insertCost & ;\ i = 0,\ j > 0 \\ D(i-1, j-1) & ;\ S_1[i] = S_2[j] \\ \min (& \\ \quad D(i, j-1) + insertCost & \\ \quad D(i-1, j) + deleteCost & ;\ j > 0,\ i > 0,\ S_1[i] \neq S_2[j] \\ \quad D(i-1, j-1) + replaceCost & \\ ) & \end{cases}$$

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

первую строку и столбец заполним просто вставками и удалениями т.к. какое-то из слов вообще не рассматривается

$$D(i,j) = \begin{cases} 0 & ; i = 0,\ j = 0 \\ i * deleteCost & ; j = 0,\ i > 0 \\ j * insertCost & ; i = 0,\ j > 0 \\ D(i-1, j-1) & ; S_1[i] = S_2[j] \\ \min(& \\ \quad D(i, j-1) + insertCost & \\ \quad D(i-1, j) + deleteCost & ; j > 0,\ i > 0,\ S_1[i] \neq S_2[j] \\ \quad D(i-1, j-1) + replaceCost & \\ ) & \end{cases}$$

	0	**1 (п)**	**2 (л)**	**3 (е)**	**4 (м)**	**5 (я)**
0	0	1	2	3	4	5
1 (п)	1					
2 (о)	2					
3 (л)	3					
4 (е)	4					
5 (н)	5					
6 (о)	6					

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

буквы совпали

$$D(i,j) = \begin{cases} 0 & ; i = 0,\ j = 0 \\ i * deleteCost & ; j = 0,\ i > 0 \\ j * insertCost & ; i = 0,\ j > 0 \\ D(i-1, j-1) & ; S_1[i] = S_2[j] \\ \min(& \\ \quad D(i, j-1) + insertCost & \\ \quad D(i-1, j) + deleteCost & ; j > 0,\ i > 0,\ S_1[i] \neq S_2[j] \\ \quad D(i-1, j-1) + replaceCost & \\ ) & \end{cases}$$

	0	**1 (п)**	**2 (л)**	**3 (е)**	**4 (м)**	**5 (я)**
0	0	1	2	3	4	5
1 (п)	1	0				
2 (о)	2					
3 (л)	3					
4 (е)	4					
5 (н)	5					
6 (о)	6					

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

делаем вставку буквы "п" во второе слово

$$D(i,j) = \begin{cases} 0 & ; i = 0,\ j = 0 \\ i * deleteCost & ; j = 0,\ i > 0 \\ j * insertCost & ; i = 0,\ j > 0 \\ D(i-1, j-1) & ; S_1[i] = S_2[j] \\ \min(& \\ \quad D(i, j-1) + insertCost & \\ \quad D(i-1, j) + deleteCost & ; j > 0,\ i > 0,\ S_1[i] \neq S_2[j] \\ \quad D(i-1, j-1) + replaceCost & \\ ) & \end{cases}$$

	0	1 (п)	2 (л)	3 (е)	4 (м)	5 (я)
0	0	1	2	3	4	5
1 (п)	1	0	1			
2 (о)	2					
3 (л)	3					
4 (е)	4					
5 (н)	5					
6 (о)	6					

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

продолжаем делать вставки

$$D(i,j) = \begin{cases} 0 & ;\ i=0,\ j=0 \\ i * deleteCost & ;\ j=0,\ i>0 \\ j * insertCost & ;\ i=0,\ j>0 \\ D(i-1,j-1) & ;\ S_1[i]=S_2[j] \\ \min(& \\ \quad D(i,j-1)+insertCost & \\ \quad D(i-1,j)+deleteCost & ;\ j>0,\ i>0,\ S_1[i]\neq S_2[j] \\ \quad D(i-1,j-1)+replaceCost & \\ ) & \end{cases}$$

	0	1 (п)	2 (л)	3 (е)	4 (м)	5 (я)
0	0	1	2	3	4	5
1 (п)	1	0	1	2	3	4
2 (о)	2					
3 (л)	3					
4 (е)	4					
5 (н)	5					
6 (о)	6					

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

из "по" удаляем "о"

$$D(i,j) = \begin{cases} 0 & ;\ i=0,\ j=0 \\ i * deleteCost & ;\ j=0,\ i>0 \\ j * insertCost & ;\ i=0,\ j>0 \\ D(i-1,j-1) & ;\ S_1[i]=S_2[j] \\ \min(& \\ \quad D(i,j-1)+insertCost & \\ \quad D(i-1,j)+deleteCost & ;\ j>0,\ i>0,\ S_1[i]\neq S_2[j] \\ \quad D(i-1,j-1)+replaceCost & \\ ) & \end{cases}$$

	0	1 (п)	2 (л)	3 (е)	4 (м)	5 (я)
0	0	1	2	3	4	5
1 (п)	1	0	1	2	3	4
2 (о)	2	1				
3 (л)	3					
4 (е)	4					
5 (н)	5					
6 (о)	6					

+1

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

Замена "о" на "л"

$$D(i,j) = \begin{cases} 0 & ; i = 0,\ j = 0 \\ i * deleteCost & ; j = 0,\ i > 0 \\ j * insertCost & ; i = 0,\ j > 0 \\ D(i-1, j-1) & ; S_1[i] = S_2[j] \\ \min(& \\ \quad D(i, j-1) + insertCost & \\ \quad D(i-1, j) + deleteCost & ; j > 0,\ i > 0,\ S_1[i] \neq S_2[j] \\ \quad D(i-1, j-1) + replaceCost & \\ ) & \end{cases}$$

	0	1 (п)	2 (л)	3 (е)	4 (м)	5 (я)
0	0	1	2	3	4	5
1 (п)	1	0	1	2	3	4
2 (о)	2	1	1			
3 (л)	3					
4 (е)	4					
5 (н)	5					
6 (о)	6					

+1

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

тут совпадение

$$D(i,j) = \begin{cases} 0 & ; i = 0,\ j = 0 \\ i * deleteCost & ; j = 0,\ i > 0 \\ j * insertCost & ; i = 0,\ j > 0 \\ D(i-1, j-1) & ; S_1[i] = S_2[j] \\ \min(& \\ \quad D(i, j-1) + insertCost & \\ \quad D(i-1, j) + deleteCost & ; j > 0,\ i > 0,\ S_1[i] \neq S_2[j] \\ \quad D(i-1, j-1) + replaceCost & \\ ) & \end{cases}$$

	0	**1 (п)**	**2 (л)**	**3 (е)**	**4 (м)**	**5 (я)**
0	0	1	2	3	4	5
1 (п)	1	0	1	2	3	4
2 (о)	2	1	1	2	3	4
3 (л)	3	2	1			
4 (е)	4					
5 (н)	5					
6 (о)	6					

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

$$D(i,j) = \begin{cases} 0 & ; i = 0,\ j = 0 \\ i * deleteCost & ; j = 0,\ i > 0 \\ j * insertCost & ; i = 0,\ j > 0 \\ D(i-1, j-1) & ; S_1[i] = S_2[j] \\ \min(& \\ \quad D(i, j-1) + insertCost & \\ \quad D(i-1, j) + deleteCost & ; j > 0,\ i > 0,\ S_1[i] \neq S_2[j] \\ \quad D(i-1, j-1) + replaceCost & \\ ) & \end{cases}$$

	0	**1 (п)**	**2 (л)**	**3 (е)**	**4 (м)**	**5 (я)**
0	0	1	2	3	4	5
1 (п)	1	0	1	2	3	4
2 (о)	2	1	1	2	3	4
3 (л)	3	2	1	2	3	4
4 (е)	4	3	2	1	2	3
5 (н)	5	4	3	2	2	3
6 (о)	6	5	4	3	3	3

редакционное расстояние

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

как восстановить ответ?

$$D(i,j) = \begin{cases} 0 & ;\ i = 0,\ j = 0 \\ i * deleteCost & ;\ j = 0,\ i > 0 \\ j * insertCost & ;\ i = 0,\ j > 0 \\ D(i-1, j-1) & ;\ S_1[i] = S_2[j] \\ \min(& \\ \quad D(i, j-1) + insertCost & \\ \quad D(i-1, j) + deleteCost & ;\ j > 0,\ i > 0,\ S_1[i] \neq S_2[j] \\ \quad D(i-1, j-1) + replaceCost & \\ ) & \end{cases}$$

	0	1 (п)	2 (л)	3 (е)	4 (м)	5 (я)
0	0	1	2	3	4	5
1 (п)	1	0	1	2	3	4
2 (о)	2	1	1	2	3	4
3 (л)	3	2	1	2	3	4
4 (е)	4	3	2	1	2	3
5 (н)	5	4	3	2	2	3
6 (о)	6	5	4	3	3	3

идем по min значениям:

вверх – delete

влево – insert

диагональ – change/equal

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

как восстановить ответ?

$$D(i,j) = \begin{cases} 0 & ;\ i = 0,\ j = 0 \\ i * deleteCost & ;\ j = 0,\ i > 0 \\ j * insertCost & ;\ i = 0,\ j > 0 \\ D(i-1, j-1) & ;\ S_1[i] = S_2[j] \\ \min (& \\ \quad D(i, j-1) + insertCost & \\ \quad D(i-1, j) + deleteCost & ;\ j > 0,\ i > 0,\ S_1[i] \neq S_2[j] \\ \quad D(i-1, j-1) + replaceCost & \\ ) & \end{cases}$$

	0	1 (п)	2 (л)	3 (е)	4 (м)	5 (я)
0	0	1	2	3	4	5
1 (п)	1	0	1	2	3	4
2 (о)	2	1	1	2	3	4
3 (л)	3	2	1	2	3	4
4 (е)	4	3	2	1	2	3
5 (н)	5	4	3	2	2	3
6 (о)	6	5	4	3	3	3

Иначе

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

как восстановить ответ?

$$D(i,j) = \begin{cases} 0 & ;\ i = 0,\ j = 0 \\ i * deleteCost & ;\ j = 0,\ i > 0 \\ j * insertCost & ;\ i = 0,\ j > 0 \\ D(i-1, j-1) & ;\ S_1[i] = S_2[j] \\ \min (& \\ \quad D(i, j-1) + insertCost & \\ \quad D(i-1, j) + deleteCost & ;\ j > 0,\ i > 0,\ S_1[i] \neq S_2[j] \\ \quad D(i-1, j-1) + replaceCost & \\ ) & \end{cases}$$

	0	1 (п)	2 (л)	3 (е)	4 (м)	5 (я)
0	0	1	2	3	4	5
1 (п)	1	0	1	2	3	4
2 (о)	2	1	1	2	3	4
3 (л)	3	2	1	2	3	4
4 (е)	4	3	2	1	2	3
5 (н)	5	4	3	2	2	3
6 (о)	6	5	4	3	3	3

Иначе

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

как восстановить ответ?

$$D(i,j) = \begin{cases} 0 & ;\ i = 0,\ j = 0 \\ i * deleteCost & ;\ j = 0,\ i > 0 \\ j * insertCost & ;\ i = 0,\ j > 0 \\ D(i-1, j-1) & ;\ S_1[i] = S_2[j] \\ \min(& \\ \quad D(i, j-1) + insertCost & \\ \quad D(i-1, j) + deleteCost & ;\ j > 0,\ i > 0,\ S_1[i] \neq S_2[j] \\ \quad D(i-1, j-1) + replaceCost & \\ ) & \end{cases}$$

	0	1 (п)	2 (л)	3 (е)	4 (м)	5 (я)
0	0	1	2	3	4	5
1 (п)	1	0	1	2	3	4
2 (о)	2	1	1	2	3	4
3 (л)	3	2	1	2	3	4
4 (е)	4	3	2	1	2	3
5 (н)	5	4	3	2	2	3
6 (о)	6	5	4	3	3	3

equal

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

как восстановить ответ?

$$D(i,j) = \begin{cases} 0 & ;\ i=0,\ j=0 \\ i * deleteCost & ;\ j=0,\ i>0 \\ j * insertCost & ;\ i=0,\ j>0 \\ D(i-1,j-1) & ;\ S_1[i]=S_2[j] \\ \min(& \\ \quad D(i,j-1)+insertCost & \\ \quad D(i-1,j)+deleteCost & ;\ j>0,\ i>0,\ S_1[i]\neq S_2[j] \\ \quad D(i-1,j-1)+replaceCost & \\ ) & \end{cases}$$

	0	1 (п)	2 (л)	3 (е)	4 (м)	5 (я)
0	0	1	2	3	4	5
1 (п)	1	0	1	2	3	4
2 (о)	2	1	1	2	3	4
3 (л)	3	2	1	2	3	4
4 (е)	4	3	2	1	2	3
5 (н)	5	4	3	2	2	3
6 (о)	6	5	4	3	3	3

equal

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

как восстановить ответ?

$$D(i,j) = \begin{cases} 0 & ;\ i = 0,\ j = 0 \\ i * deleteCost & ;\ j = 0,\ i > 0 \\ j * insertCost & ;\ i = 0,\ j > 0 \\ D(i-1, j-1) & ;\ S_1[i] = S_2[j] \\ \min(& \\ \quad D(i, j-1) + insertCost & \\ \quad D(i-1, j) + deleteCost & ;\ j > 0,\ i > 0,\ S_1[i] \neq S_2[j] \\ \quad D(i-1, j-1) + replaceCost & \\ ) & \end{cases}$$

	0	1 (п)	2 (л)	3 (е)	4 (м)	5 (я)
0	0	1	2	3	4	5
1 (п)	1	0	1	2	3	4
2 (о)	2	1	1	2	3	4
3 (л)	3	2	1	2	3	4
4 (е)	4	3	2	1	2	3
5 (н)	5	4	3	2	2	3
6 (о)	6	5	4	3	3	3

delete

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

как восстановить ответ?

$$D(i,j) = \begin{cases} 0 & ;\ i = 0,\ j = 0 \\ i * deleteCost & ;\ j = 0,\ i > 0 \\ j * insertCost & ;\ i = 0,\ j > 0 \\ D(i-1, j-1) & ;\ S_1[i] = S_2[j] \\ \min(& \\ \quad D(i, j-1) + insertCost & \\ \quad D(i-1, j) + deleteCost & ;\ j > 0,\ i > 0,\ S_1[i] \neq S_2[j] \\ \quad D(i-1, j-1) + replaceCost & \\ ) & \end{cases}$$

	0	1 (п)	2 (л)	3 (е)	4 (м)	5 (я)
0	0	1	2	3	4	5
1 (п)	1	0	1	2	3	4
2 (о)	2	1	1	2	3	4
3 (л)	3	2	1	2	3	4
4 (е)	4	3	2	1	2	3
5 (н)	5	4	3	2	2	3
6 (о)	6	5	4	3	3	3

equal

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

Время работы?

$$D(i,j) = \begin{cases} 0 & ;\ i=0,\ j=0 \\ i * deleteCost & ;\ j=0,\ i>0 \\ j * insertCost & ;\ i=0,\ j>0 \\ D(i-1,j-1) & ;\ S_1[i]=S_2[j] \\ \min(& \\ \quad D(i,j-1)+insertCost & \\ \quad D(i-1,j)+deleteCost & ;\ j>0,\ i>0,\ S_1[i]\neq S_2[j] \\ \quad D(i-1,j-1)+replaceCost & \\ ) & \end{cases}$$

	0	**1 (п)**	**2 (л)**	**3 (е)**	**4 (м)**	**5 (я)**
0	0	1	2	3	4	5
1 (п)	1	0	1	2	3	4
2 (о)	2	1	1	2	3	4
3 (л)	3	2	1	2	3	4
4 (е)	4	3	2	1	2	3
5 (н)	5	4	3	2	2	3
6 (о)	6	5	4	3	3	3

Задача о редакционном расстоянии

ПОЛЕНО → ПЛЕНО → ПЛЕМО → ПЛЕМЯ

Время работы?

$$D(i,j) = \begin{cases} 0 & ;\ i=0,\ j=0 \\ i * deleteCost & ;\ j=0,\ i>0 \\ j * insertCost & ;\ i=0,\ j>0 \\ D(i-1,j-1) & ;\ S_1[i]=S_2[j] \\ \min(& \\ \quad D(i,j-1)+insertCost & \\ \quad D(i-1,j)+deleteCost & ;\ j>0,\ i>0,\ S_1[i]\neq S_2[j] \\ \quad D(i-1,j-1)+replaceCost & \\ ) & \end{cases}$$

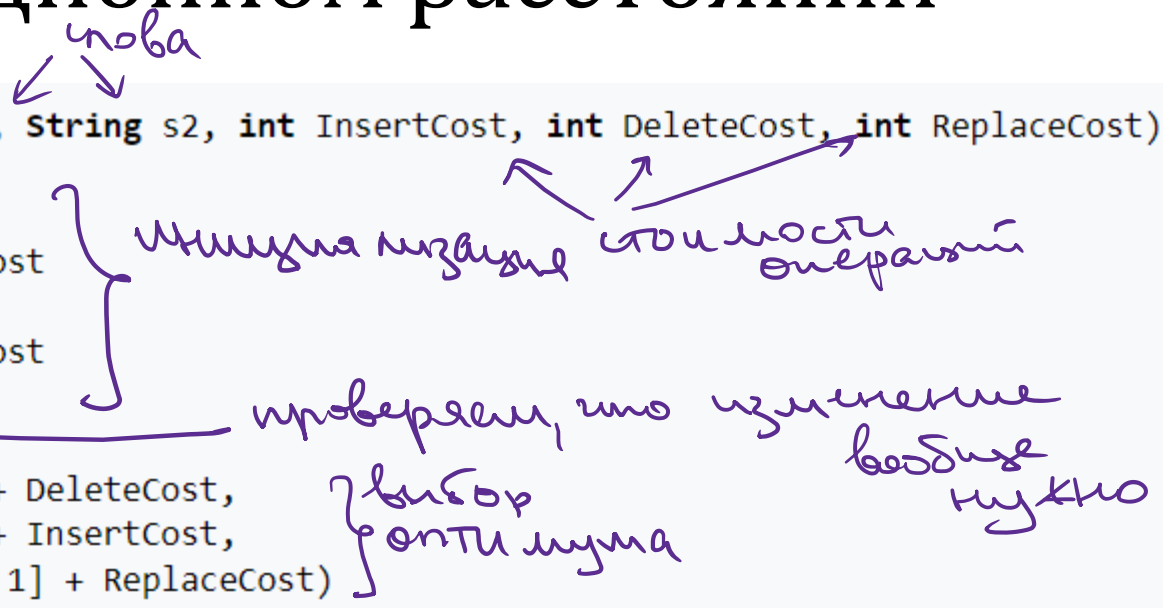
	0	1 (п)	2 (л)	3 (е)	4 (м)	5 (я)
0	0	1	2	3	4	5
1 (п)	1	0	1	2	3	4
2 (о)	2	1	1	2	3	4
3 (л)	3	2	1	2	3	4
4 (е)	4	3	2	1	2	3
5 (н)	5	4	3	2	2	3
6 (о)	6	5	4	3	3	3

размер слова

$O(|s_1| \cdot |s_2|)$

Задача о редакционном расстоянии

```
int levensteinInstruction(String s1, String s2, int InsertCost, int DeleteCost, int ReplaceCost):
  D[0][0] = 0
  for j = 1 to N
    D[0][j] = D[0][j - 1] + InsertCost
  for i = 1 to M
    D[i][0] = D[i - 1][0] + DeleteCost
    for j = 1 to N
      if S1[i] != S2[j]
         D[i][j] = min(D[i - 1][j] + DeleteCost,
                       D[i][j - 1] + InsertCost,
                       D[i - 1][j - 1] + ReplaceCost)
      else
         D[i][j] = D[i - 1][j - 1]
  return D[M][N]
```



Задача о расстановке знаков в выражении

1 + 3 + 2 + 4 = 10

Задача о расстановке знаков в выражении

1 + 3 + 2 + 4 = 10

(1 + 3) * (2 + 4) = 24

Задача о расстановке знаков в выражении

1 + 3 + 2 + 4 = 10

(1 + 3) * (2 + 4) = 24

(1 + 3 + 2) * 4 = 24

Задача о расстановке знаков в выражении

1 + 3 + 2 + 4 = 10

(1 + 3) * (2 + 4) = 24

(1 + 3 + 2) * 4 = 24

1 * 3 * 2 * 4 = 24

Задача о расстановке знаков в выражении

1 + 3 + 2 + 4 = 10

(1 + 3) * (2 + 4) = 24

(1 + 3 + 2) * 4 = 24

1 * 3 * 2 * 4 = 24

(1 + 3) * 2 * 4 = 32

Задача о расстановке знаков в выражении

1 + 3 + 2 + 4 = 10

(1 + 3) * (2 + 4) = 24

(1 + 3 + 2) * 4 = 24

1 * 3 * 2 * 4 = 24

(1 + 3) * 2 * 4 = 32

Задача: Даны числа. Хотим расставить +/*/скобки так, чтобы выражение имело максимальное значение

Задача о расстановке знаков в выражении

1 3 2 4

Задача: Даны числа. Хотим расставить +/*/скобки так, чтобы выражение имело максимальное значение

Идеи?

Задача о расстановке знаков в выражении

1 3 2 4

Задача: Даны числа. Хотим расставить +/*/скобки так, чтобы выражение имело максимальное значение

Динамика: будем пользоваться принципом оптимальности на подотрезке

Задача о расстановке знаков в выражении

1 3 2 4

Задача: Даны неотрицательные числа. Хотим расставить +/*/скобки так, чтобы выражение имело максимальное значение

Динамика: будем пользоваться принципом оптимальности на подотрезке
Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

Задача о расстановке знаков в выражении

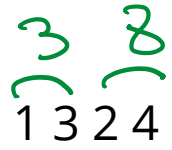
1 3 2 4

Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

Если известны значения на отрезках $[i \dots k]$, $[k \dots j]$, то на $[i \dots j]$ будет

$$\max(d[i \dots k] + d[k+1 \dots j],\ d[i \dots k] \cdot d[k+1 \dots j])$$

Задача о расстановке знаков в выражении



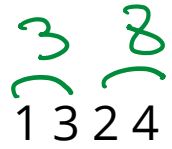
Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

3+8 ? 3*8

Если известны значения на отрезках [i…k], [k…j], то на [i…j] будет

max(d[i…k] + d[k+1…j], d[i…k] • d[k+1…j])

Задача о расстановке знаков в выражении



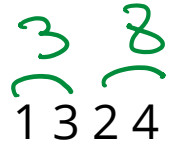
Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

3+8 < 3*8

Если известны значения на отрезках [i…k], [k…j], то на [i…j] будет

max(d[i…k] + d[k+1…j], d[i…k] • d[k+1…j])

Задача о расстановке знаков в выражении



Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

3+8 < 3*8 ← лучший вариант!

Если известны значения на отрезках $[i \dots k]$, $[k \dots j]$, то на $[i \dots j]$ будет

$$\max(d[i \dots k] + d[k+1 \dots j],\ d[i \dots k] \cdot d[k+1 \dots j])$$

Задача о расстановке знаков в выражении

1 3 2 4

Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

	0 (1)	1 (3)	2 (2)	3 (4)
0 (1)				
1 (3)	0			
2 (2)	0	0		
3 (4)	0	0	0	

Нули, т.к не бывает отрезков, где левая граница правее правой

Задача о расстановке знаков в выражении

1 3 2 4

Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

	0 (1)	**1 (3)**	**2 (2)**	**3 (4)**
0 (1)	1			
1 (3)	0	3		
2 (2)	0	0	2	
3 (4)	0	0	0	4

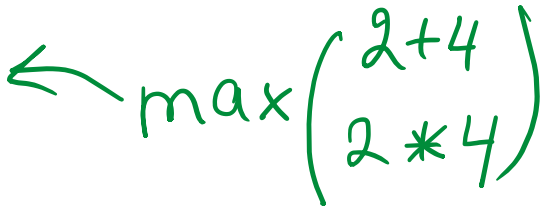
на отрезках длиной 1 имеем только один вариант ⇓ сразу записываем

Задача о расстановке знаков в выражении

1 3 2 4

Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

	0 (1)	**1 (3)**	**2 (2)**	**3 (4)**
0 (1)	1			
1 (3)	0	3		
2 (2)	0	0	2	8
3 (4)	0	0	0	4



Задача о расстановке знаков в выражении

1 3 2 4

Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

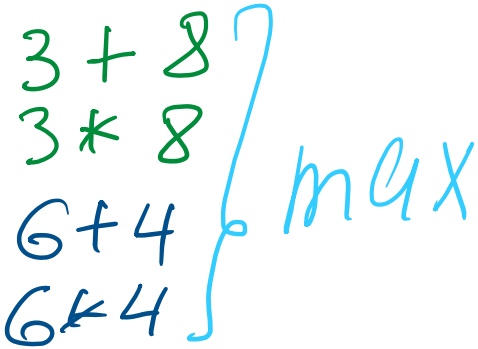
	0 (1)	**1 (3)**	**2 (2)**	**3 (4)**
0 (1)	1			
1 (3)	0	3	6	
2 (2)	0	0	2	8
3 (4)	0	0	0	4

Задача о расстановке знаков в выражении

1 3 2 4

Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

	0 (1)	**1 (3)**	**2 (2)**	**3 (4)**
0 (1)	1			
1 (3)	0	3	6	24
2 (2)	0	0	2	8
3 (4)	0	0	0	4



Задача о расстановке знаков в выражении

1 3 2 4

Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

	0 (1)	**1 (3)**	**2 (2)**	**3 (4)**
0 (1)	1	4		
1 (3)	0	3	6	24
2 (2)	0	0	2	8
3 (4)	0	0	0	4

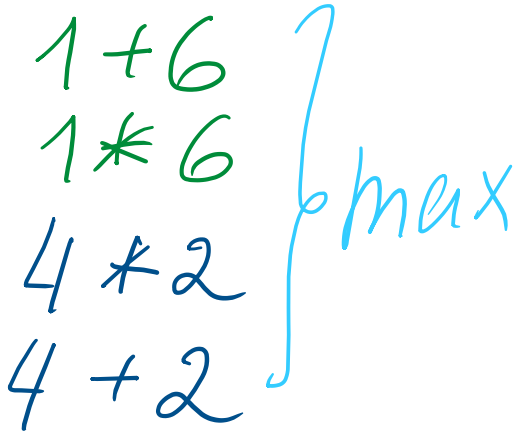
1*3
1+3
max

Задача о расстановке знаков в выражении

1 3 2 4

Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

	0 (1)	**1 (3)**	**2 (2)**	**3 (4)**
0 (1)	1	4	8	
1 (3)	0	3	6	24
2 (2)	0	0	2	8
3 (4)	0	0	0	4

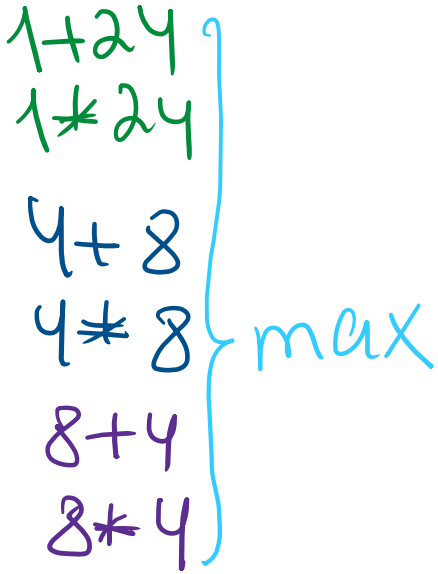


Задача о расстановке знаков в выражении

1 3 2 4

Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

	0 (1)	**1 (3)**	**2 (2)**	**3 (4)**
0 (1)	1	4	8	32
1 (3)	0	3	6	24
2 (2)	0	0	2	8
3 (4)	0	0	0	4



Задача о расстановке знаков в выражении

1 3 2 4

Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

	0 (1)	**1 (3)**	**2 (2)**	**3 (4)**
0 (1)	1	4	8	32
1 (3)	0	3	6	24
2 (2)	0	0	2	8
3 (4)	0	0	0	4

Как восстановить ответ?

Задача о расстановке знаков в выражении

1 3 2 4

Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

	0 (1)	**1 (3)**	**2 (2)**	**3 (4)**
0 (1)	1	4	8	32
1 (3)	0	3	6	24
2 (2)	0	0	2	8
3 (4)	0	0	0	4

Как восстановить ответ?

Хранить, какую пару выбрали

Задача о расстановке знаков в выражении

1 3 2 4

Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

	0 (1)	**1 (3)**	**2 (2)**	**3 (4)**
0 (1)	1	4	8	32 (0,2), (3,3)
1 (3)	0	3	6	24
2 (2)	0	0	2	8
3 (4)	0	0	0	4

Как восстановить ответ?

Хранить, какую пару выбрали

Задача о расстановке знаков в выражении

1 3 2 4

Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

	0 (1)	**1 (3)**	**2 (2)**	**3 (4)**
0 (1)	1	4	8	32 (0,2), (3,3)
1 (3)	0	3	6	24
2 (2)	0	0	2	8
3 (4)	0	0	0	4

Как восстановить ответ?

Хранить, какую пару выбрали

32 = ( )*( )

Задача о расстановке знаков в выражении

1 3 2 4

Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

	0 (1)	**1 (3)**	**2 (2)**	**3 (4)**
0 (1)	1	4	8 (0,1)(2,2)	32 (0,2), (3,3)
1 (3)	0	3	6	24
2 (2)	0	0	2	8
3 (4)	0	0	0	4

Как восстановить ответ?

32 = (( )*2)*(4)

Задача о расстановке знаков в выражении

1 3 2 4

Пусть **d[i][j]** - **максимальное значение, которое мы можем получить на отрезке от i до j.**

	0 (1)	**1 (3)**	**2 (2)**	**3 (4)**
0 (1)	1	4	8	32
1 (3)	0	3	6	24
2 (2)	0	0	2	8
3 (4)	0	0	0	4

(0,0)+(1,1) (0,1)(2,2) (0,2), (3,3)

Как восстановить ответ?

32 = ((1+3) * 2) * (1)

Задача о расстановке знаков в выражении

1 3 2 4

Пусть **d[i][j]** - **максимальное значение, которое мы можем получить на отрезке от i до j.**

	0 (1)	**1 (3)**	**2 (2)**	**3 (4)**
0 (1)	1	4	8	32
1 (3)	0	3	6	24
2 (2)	0	0	2	8
3 (4)	0	0	0	4

(0,0)+(1,1) (0,1)(2,2) (0,2), (3,3)

Как восстановить ответ?

32 = ((1+3) * 2) * (4)

Задача о расстановке знаков в выражении

1 3 2 4

Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

	0 (1)	**1 (3)**	**2 (2)**	**3 (4)**
0 (1)	1	4	8	32
1 (3)	0	3	6	24
2 (2)	0	0	2	8
3 (4)	0	0	0	4

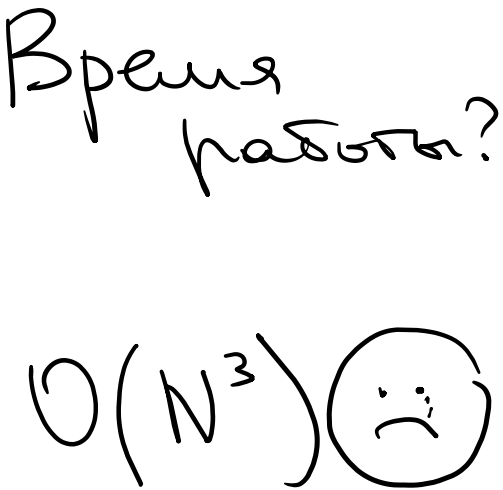
Время работы?

Задача о расстановке знаков в выражении

1 3 2 4

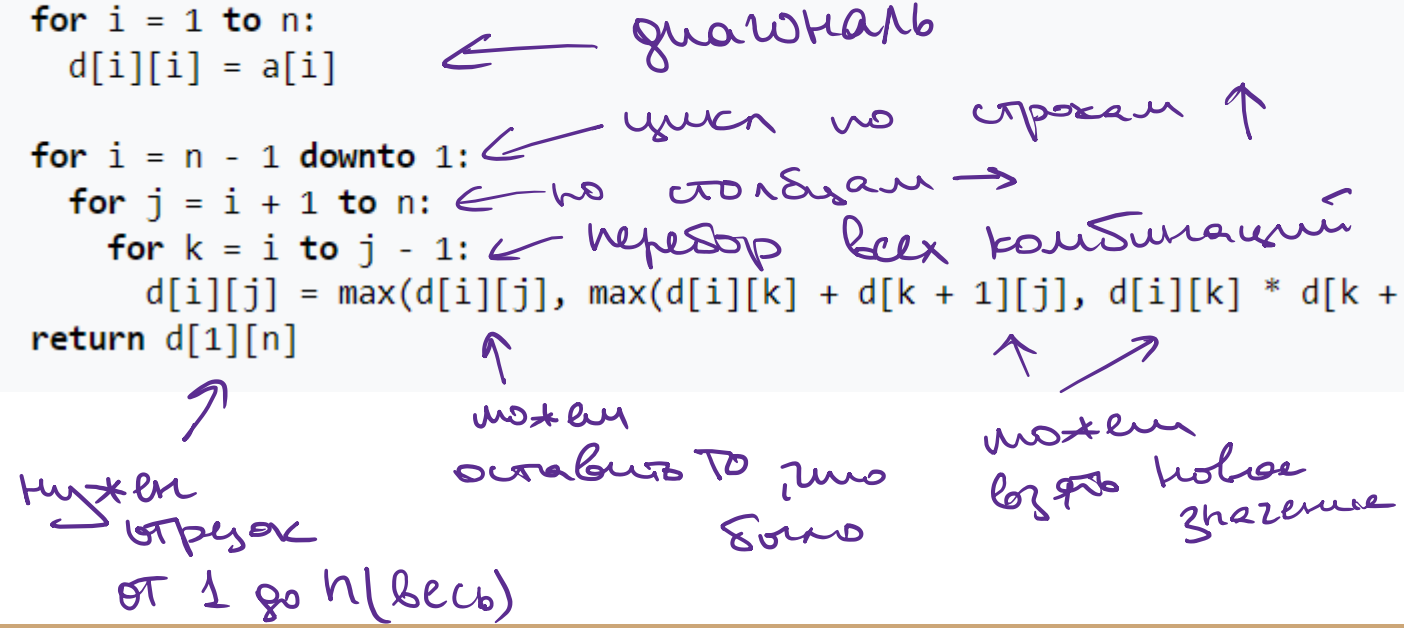
Пусть **d[i][j] - максимальное значение, которое мы можем получить на отрезке от i до j.**

	0 (1)	**1 (3)**	**2 (2)**	**3 (4)**
0 (1)	1	4	8	32
1 (3)	0	3	6	24
2 (2)	0	0	2	8
3 (4)	0	0	0	4

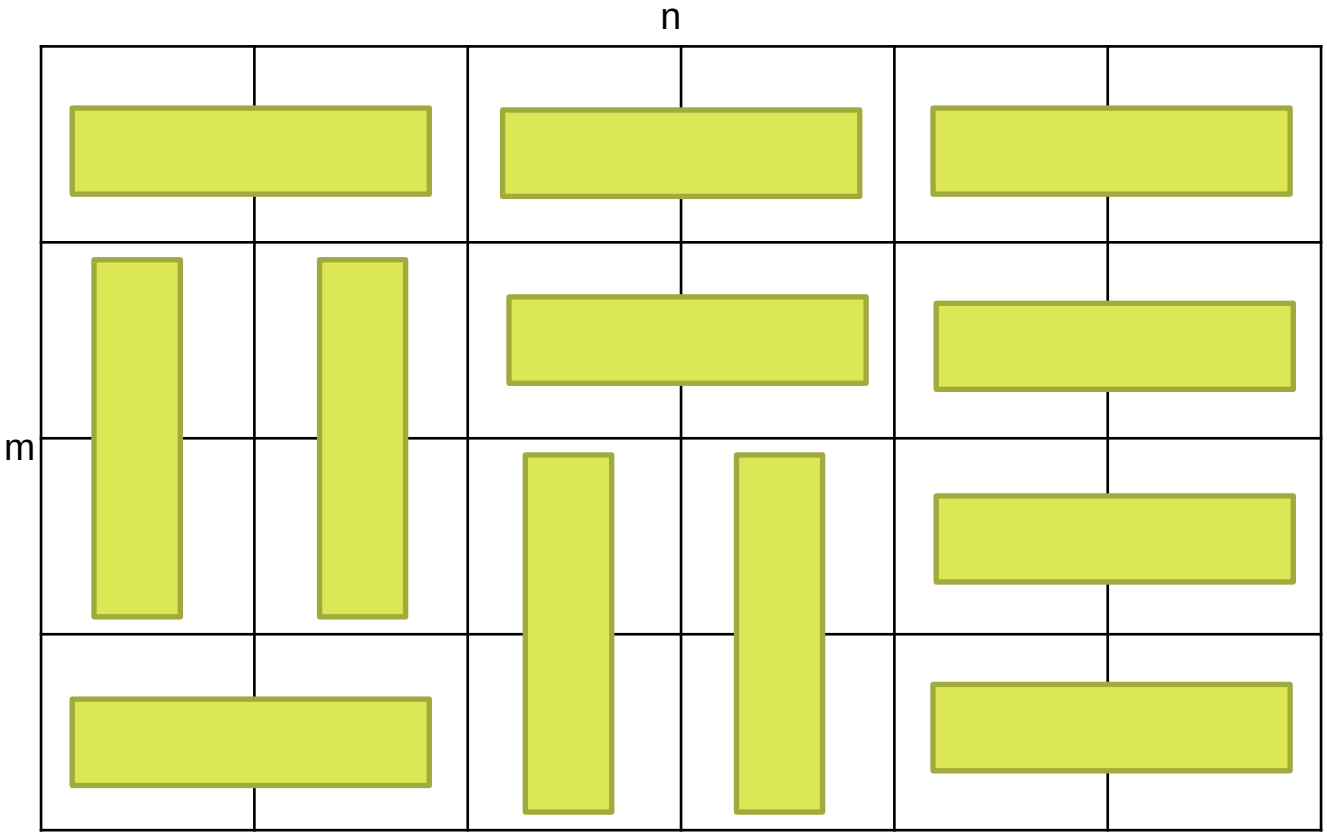


Задача о расстановке знаков в выражении

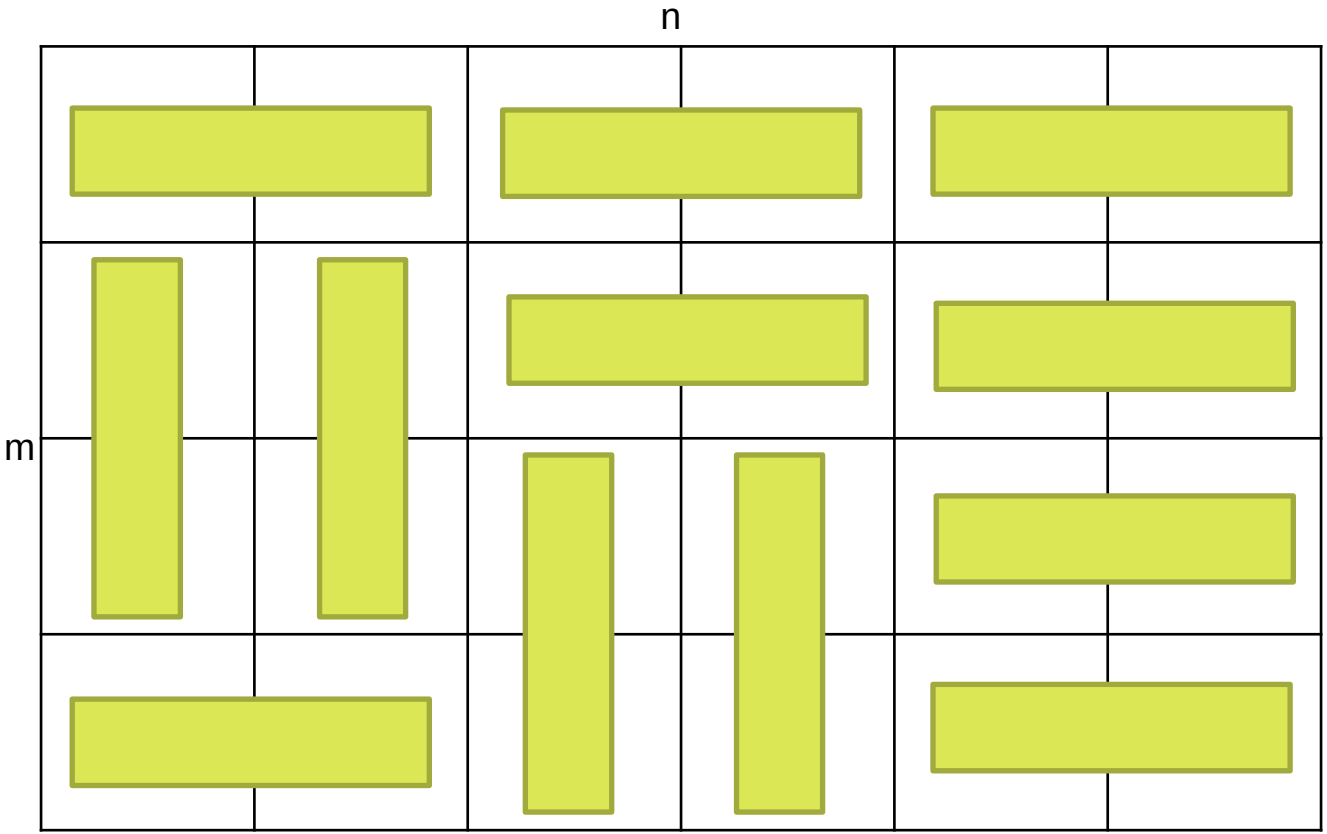
```
// a - заданная последовательность из n элементов

int maxValueOfExpression(a, n):
  for i = 1 to n:
    d[i][i] = a[i]

  for i = n - 1 downto 1:
    for j = i + 1 to n:
      for k = i to j - 1:
        d[i][j] = max(d[i][j], max(d[i][k] + d[k + 1][j], d[i][k] * d[k + 1][j]))
  return d[1][n]
```



Задача о замощении домино

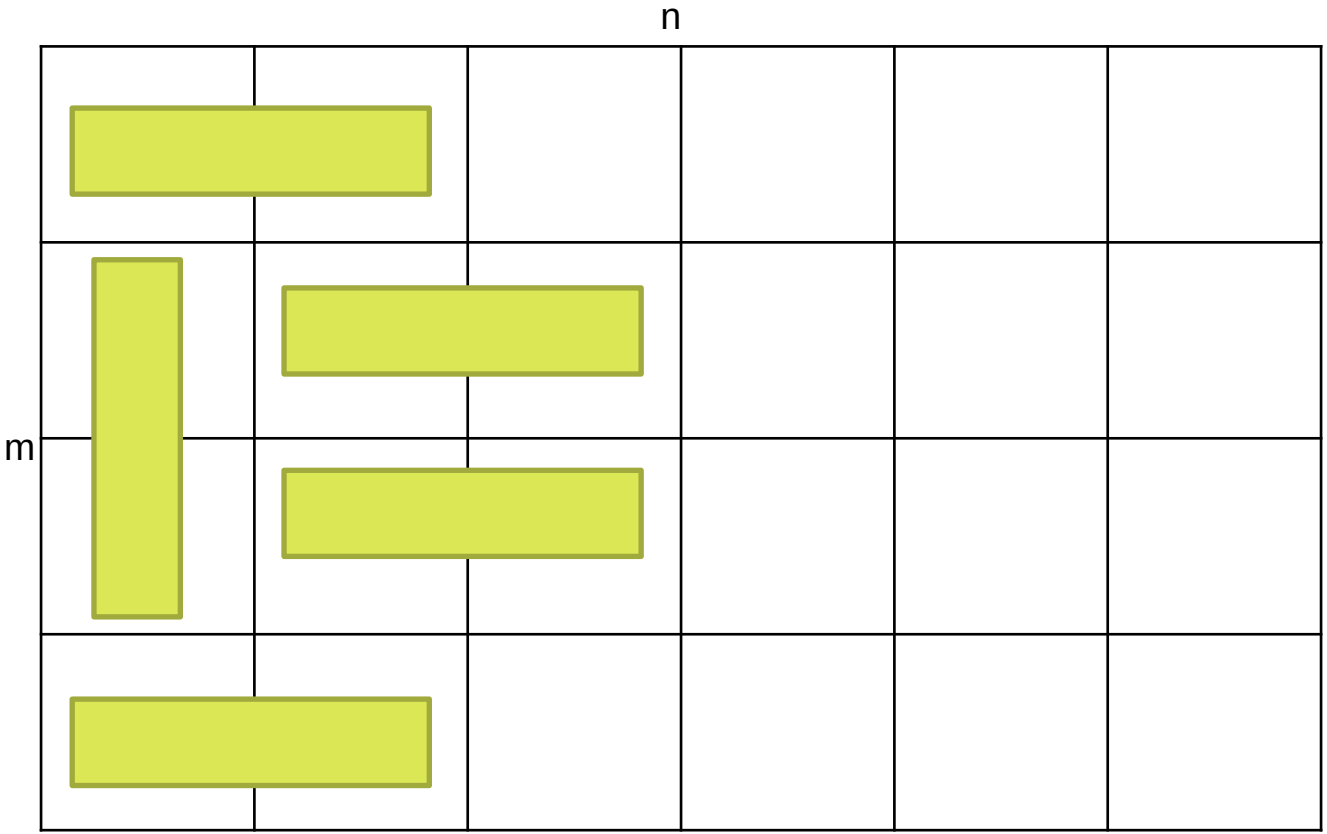


Задача о замощении домино



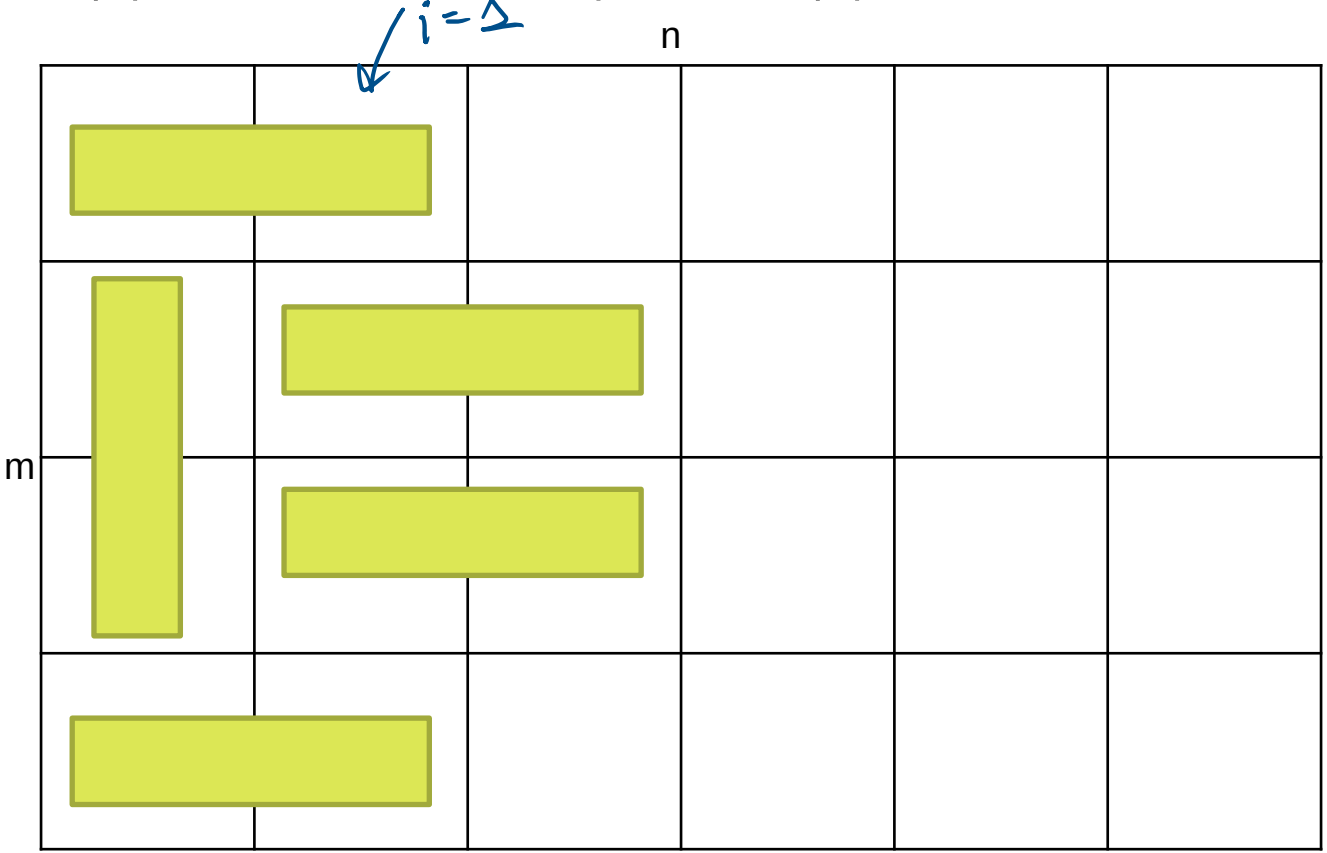
Дана таблица m*n.
Необходимо вычислить количество способов заполнить ее доминошками

Задача о замощении домино



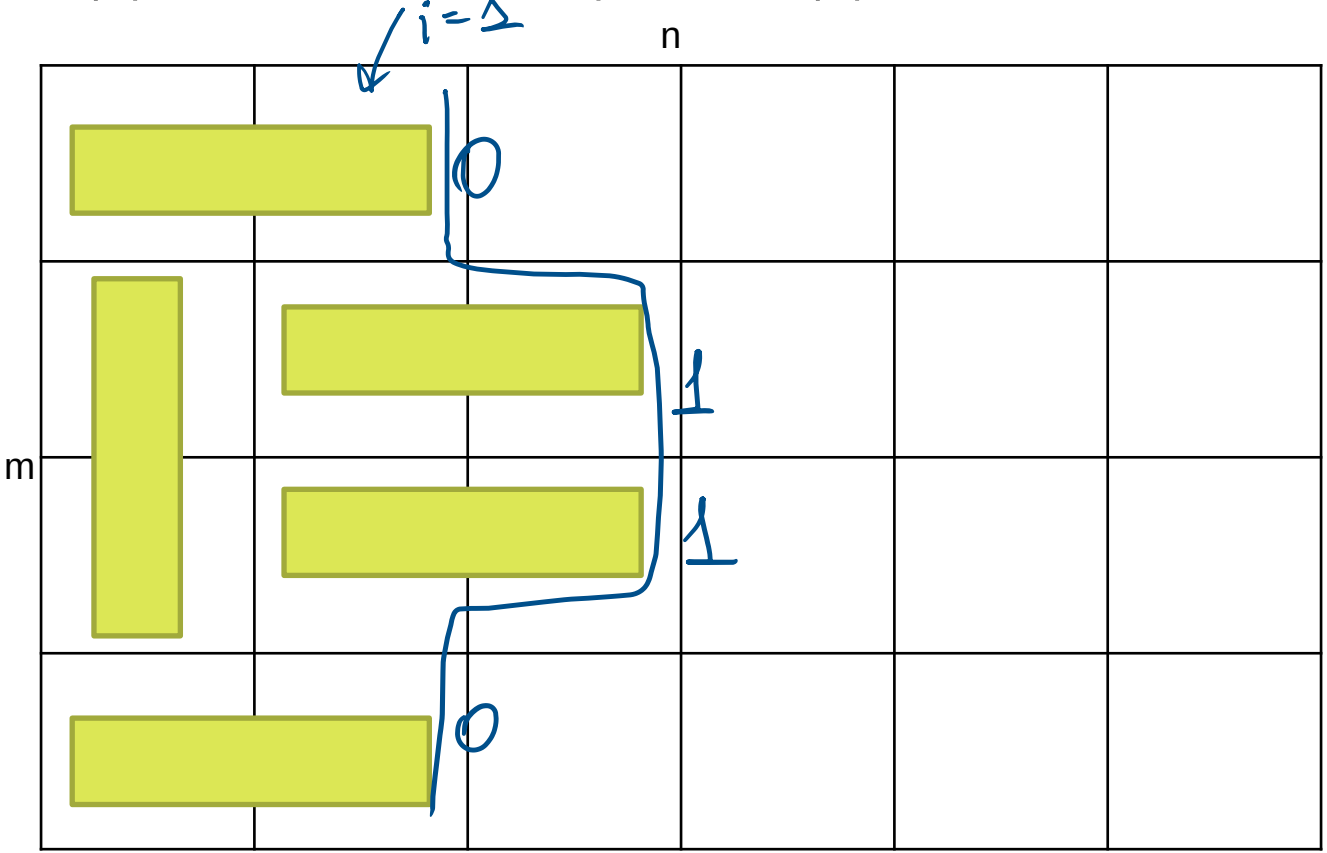
Профиль – для i-ого столбца набор из ноликов и единичек.
0 – домино «не вылезает»
1 – домино вылезает за границы

Задача о замощении домино



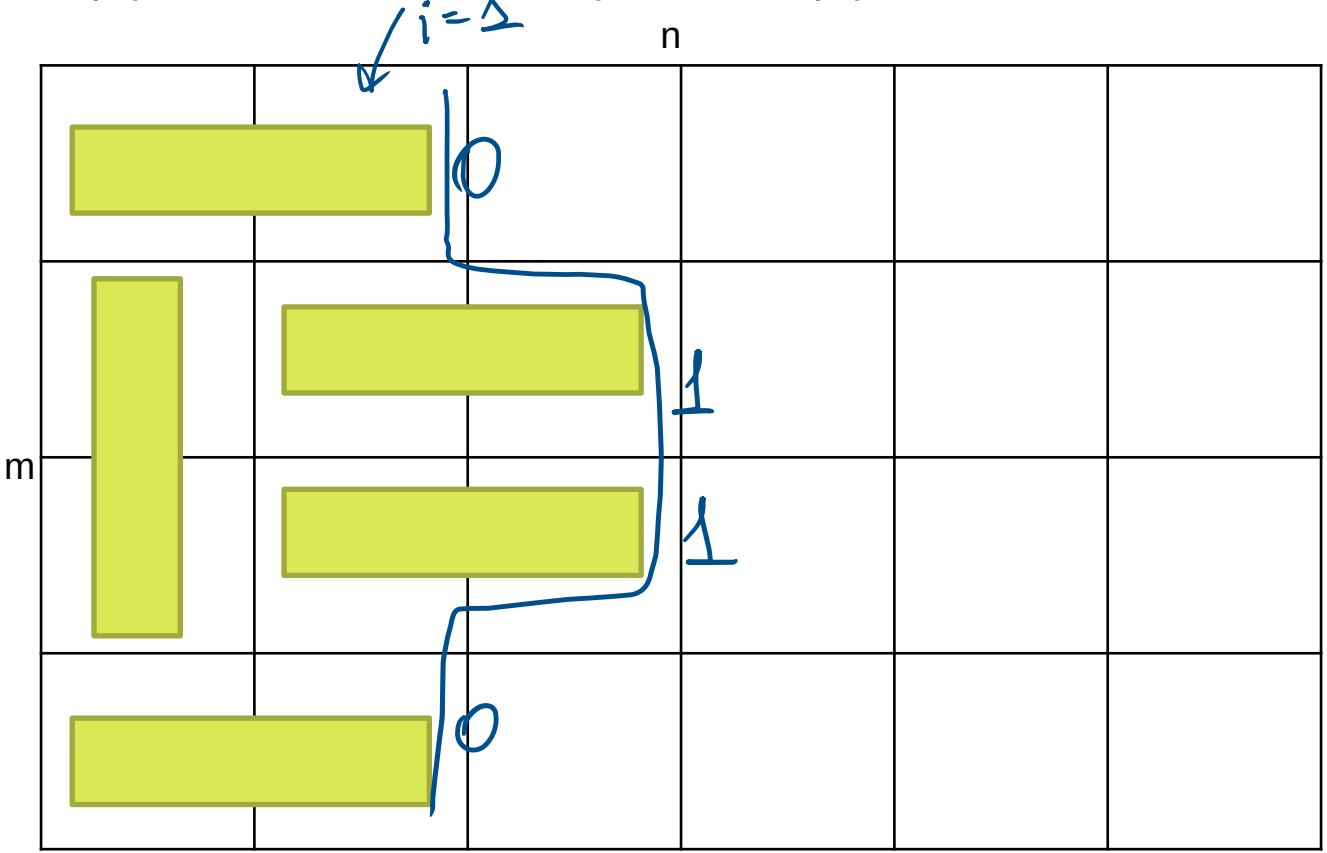
Профиль – для i-ого столбца набор из ноликов и единичек.
0 – домино «не вылезает»
1 – домино вылезает за границы

Задача о замощении домино



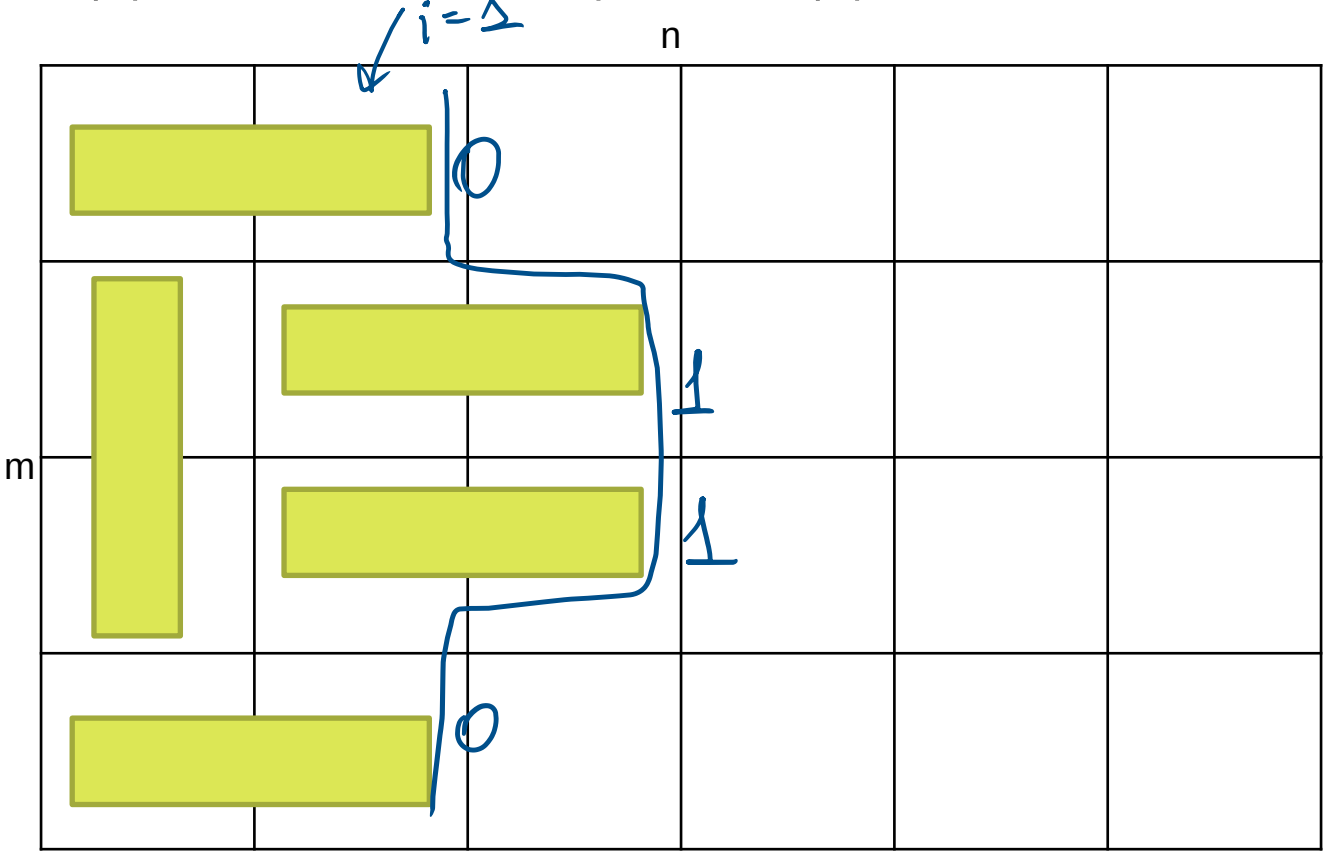
Профиль – для i-ого столбца набор из ноликов и единичек.
0 – домино «не вылезает»
1 – домино вылезает за границы

Задача о замощении домино



Динамика по (i, p) – по номеру столбца и профилю.

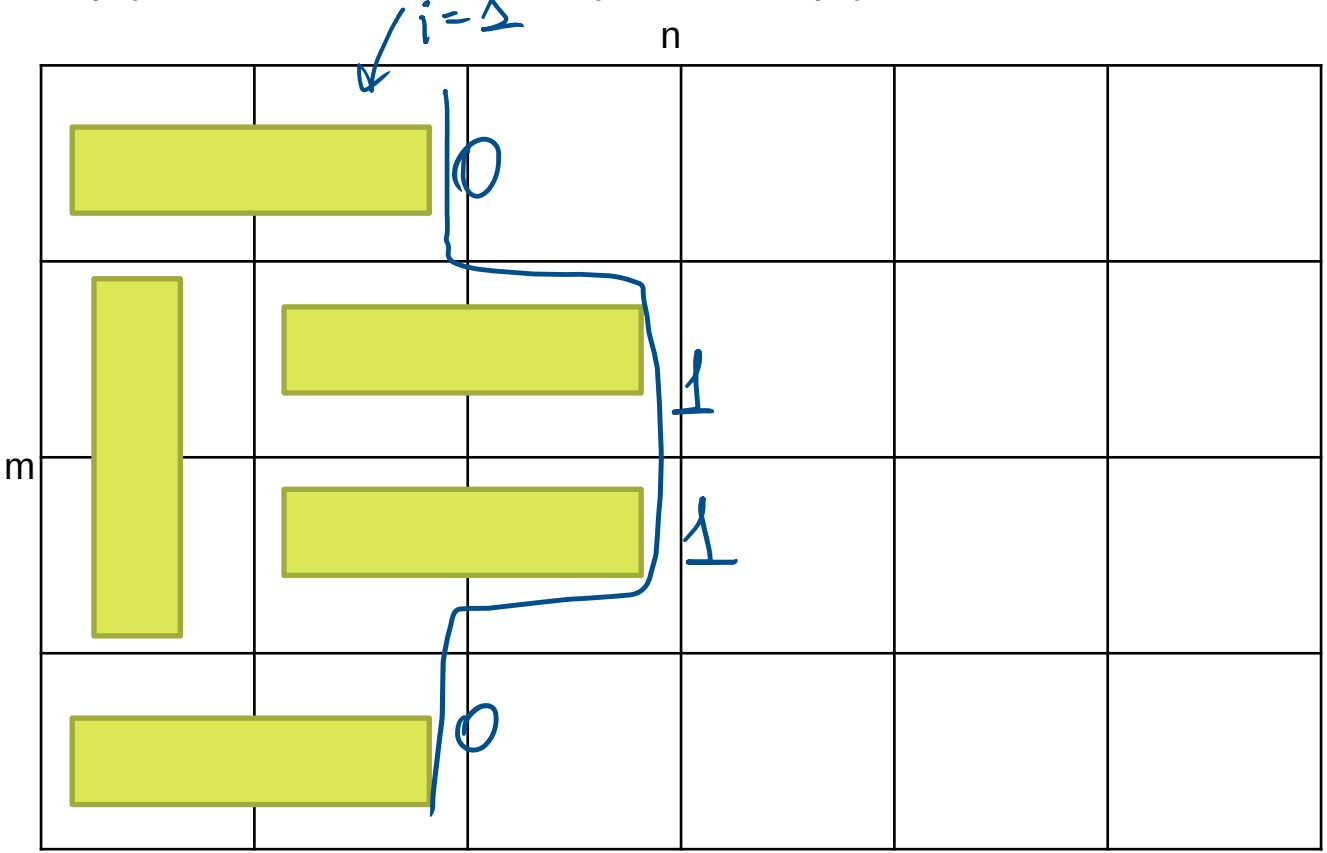
Задача о замощении домино



Динамика по (i, p) – по номеру столбца и профилю.

Сколько может быть профилей?

Задача о замощении домино

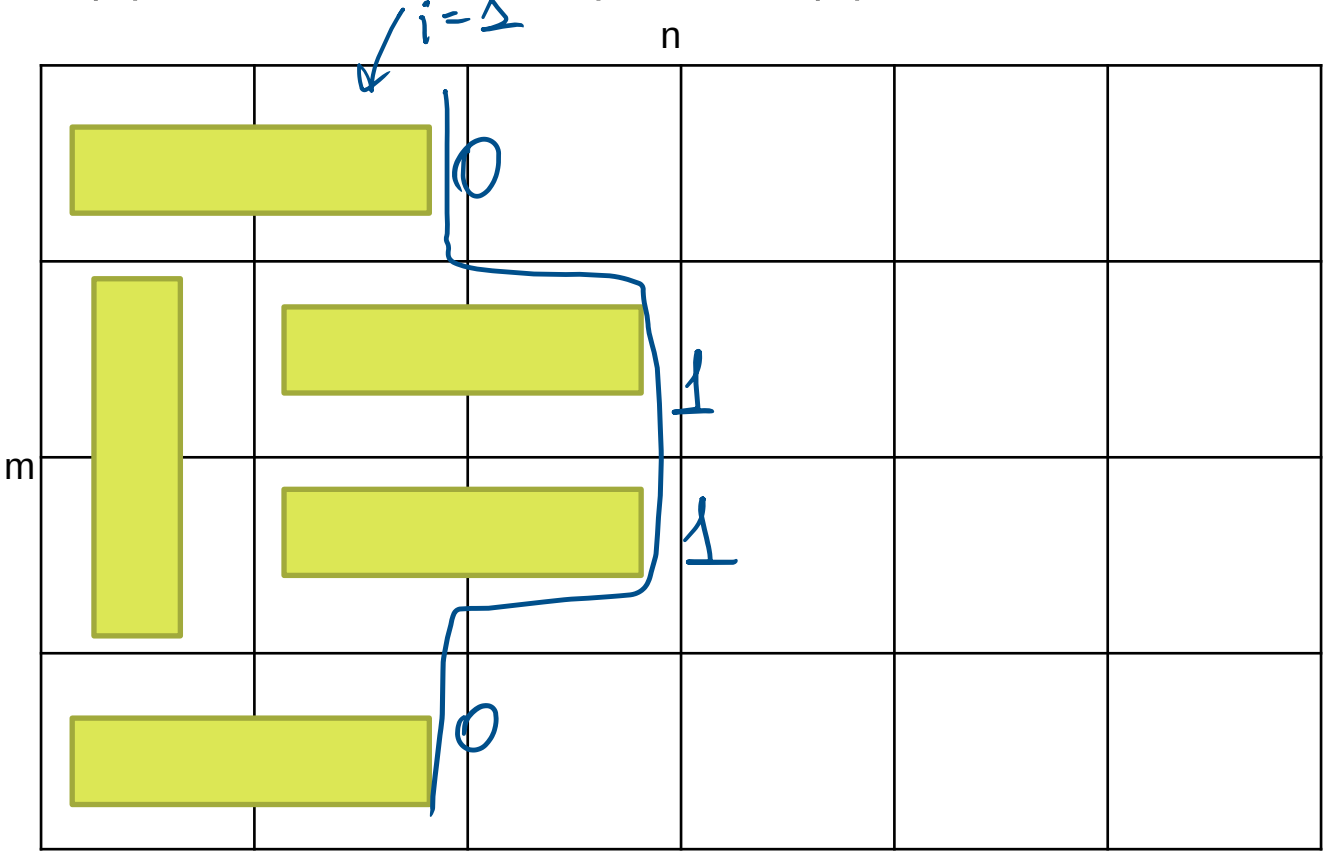


Динамика по (i, p) – по номеру столбца и профилю.

Сколько может быть профилей?

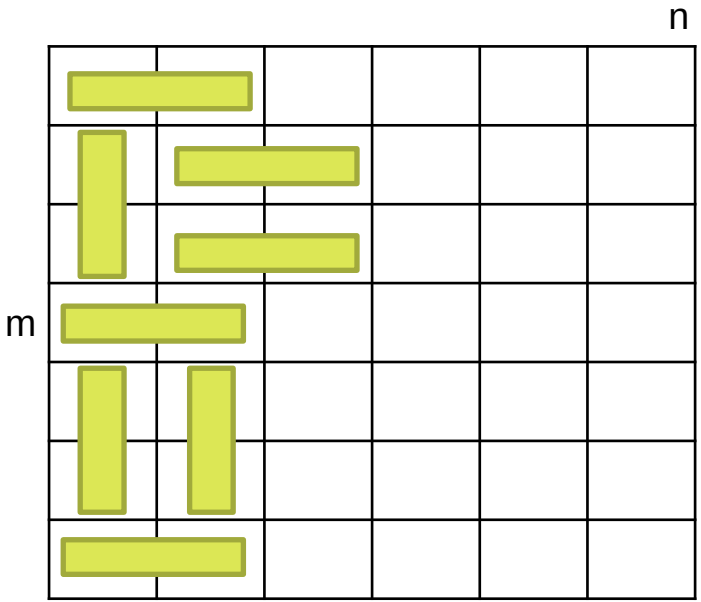
2^m

Задача о замощении домино



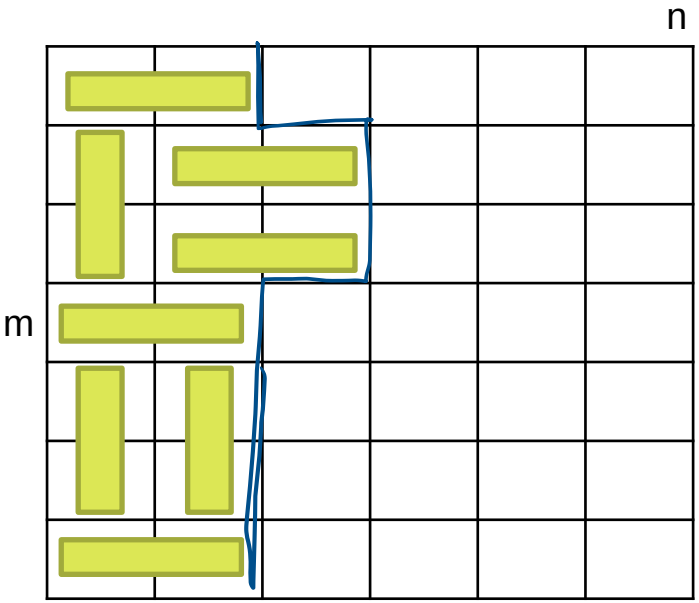
Какие могут быть переходы между профилями?

Задача о замощении домино



Какие могут быть переходы между профилями?

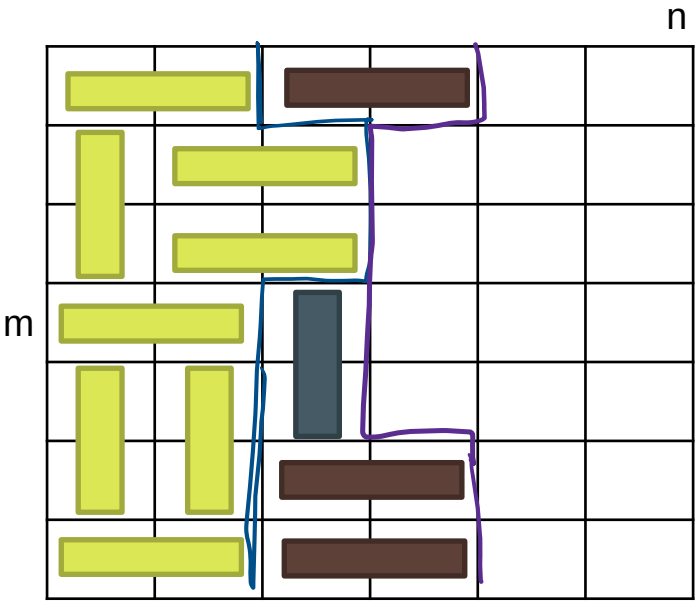
Задача о замощении домино

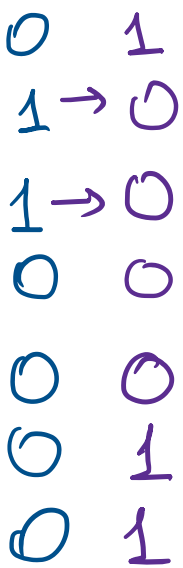


Из профиля i в профиль j можно перейти если выполняются условия:

- Можно положить горизонтальные домино. То есть там где в j профиле стоит 1, в i профиле должен стоять 0.
- Можно доложить в оставшиеся клетки вертикальные домино. То есть оставшиеся 0 в i профиле должны образовывать четные подстроки.

Задача о замощении домино

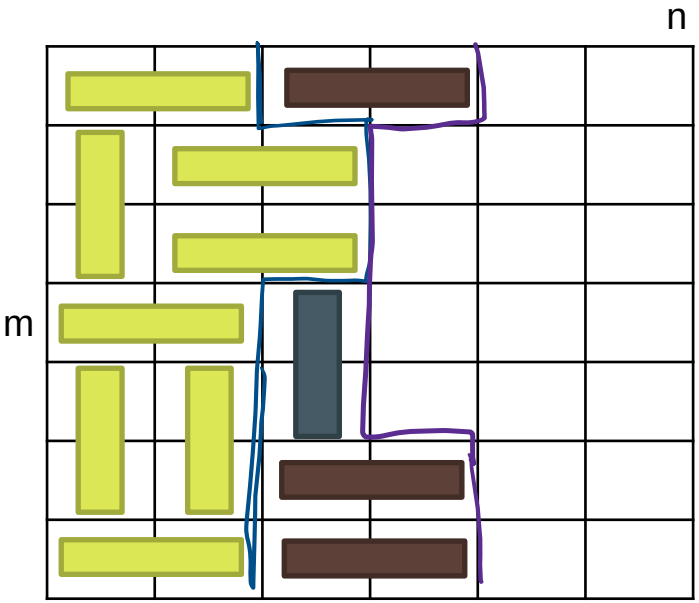


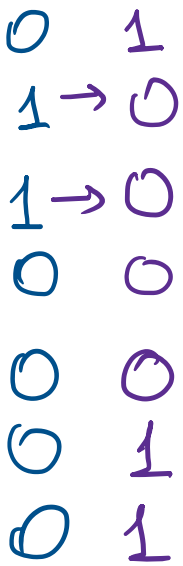


Из профиля i в профиль j можно перейти если выполняются условия:

- Можно положить горизонтальные домино. То есть там где в j профиле стоит 1, в i профиле должен стоять 0.
- Можно доложить в оставшиеся клетки вертикальные домино. То есть оставшиеся 0 в i профиле должны образовывать четные подстроки.

Задача о замощении домино



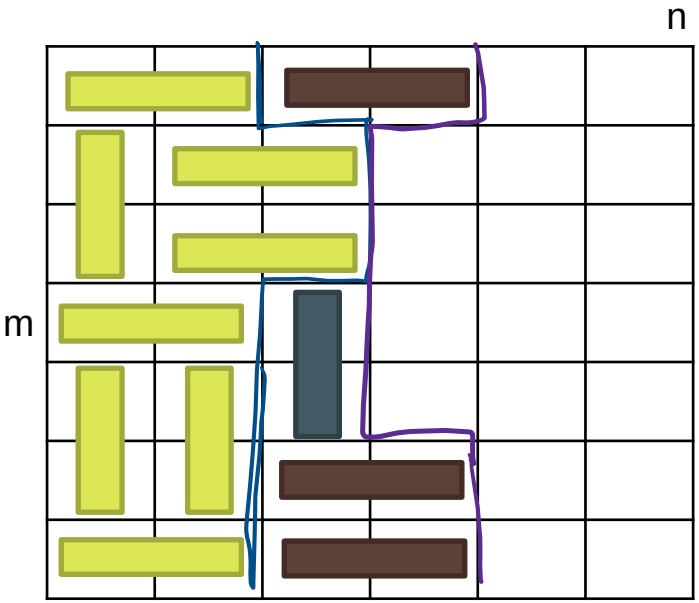


Будем для всех пар профилей хранить, можно ли между ними перейти.

Из профиля i в профиль j можно перейти если выполняются условия:

- Можно положить горизонтальные домино. То есть там где в j профиле стоит 1, в i профиле должен стоять 0.
- Можно доложить в оставшиеся клетки вертикальные домино. То есть оставшиеся 0 в i профиле должны образовывать четные подстроки.

Задача о замощении домино



0 1
1 → 0
1 → 0
0 0
0 0
0 1
0 1

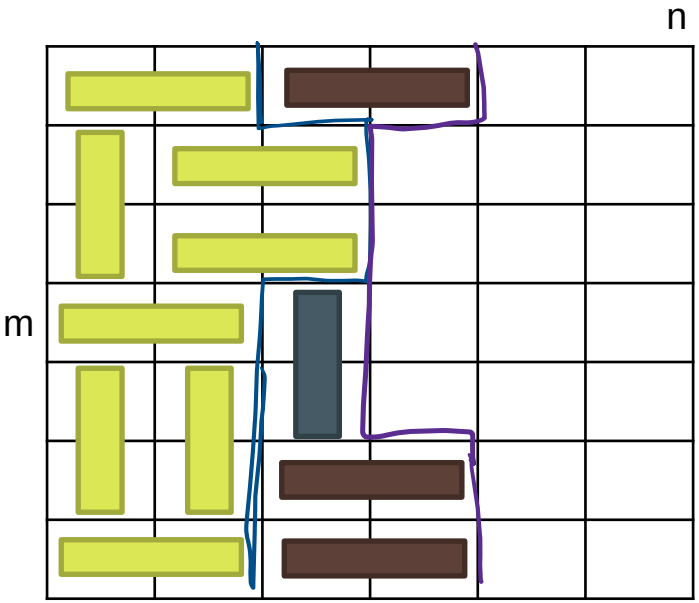
Будем для всех пар профилей хранить, можно ли между ними перейти.

d[i][j] - возможен ли переход из i профиля в j-ый

Из профиля i в профиль j можно перейти если выполняются условия:

- Можно положить горизонтальные домино. То есть там где в j профиле стоит 1, в i профиле должен стоять 0.
- Можно доложить в оставшиеся клетки вертикальные домино. То есть оставшиеся 0 в i профиле должны образовывать четные подстроки.

Задача о замощении домино



0 1
1 → 0
1 → 0
0 0
0 0
0 1
0 1

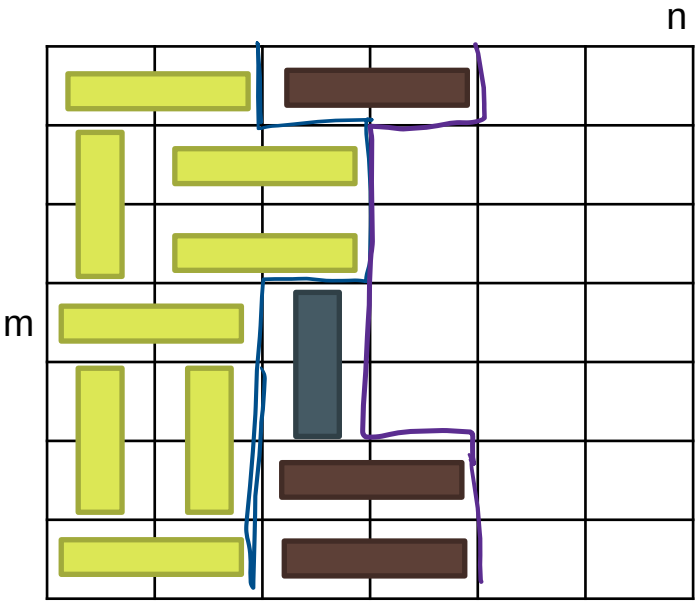
Будем для всех пар профилей хранить, можно ли между ними перейти.

d[i][j] - возможен ли переход из i профиля в j-ый

Из профиля i в профиль j можно перейти если выполняются условия:

- Можно положить горизонтальные домино. То есть там где в j профиле стоит 1, в i профиле должен стоять 0.
- Можно доложить в оставшиеся клетки вертикальные домино. То есть оставшиеся 0 в i профиле должны образовывать четные подстроки.

Задача о замощении домино



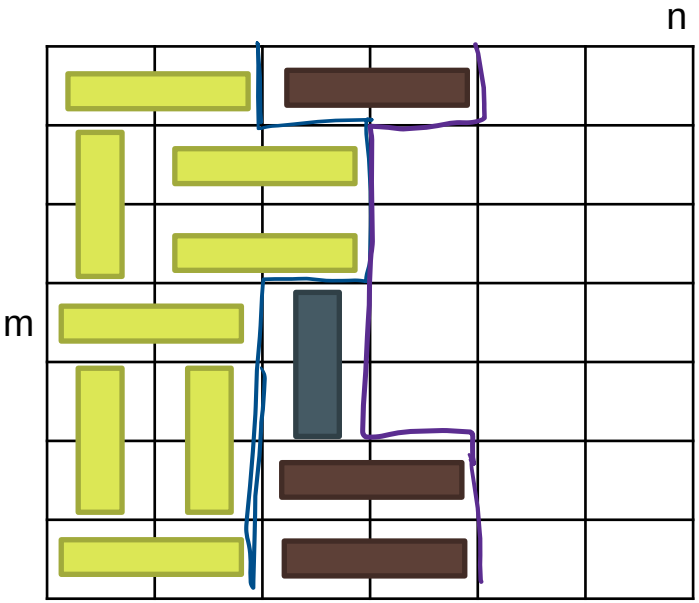
A[i][p] – количество способов замостить первые i столбцов так, чтобы они заканчивались на p-ый профиль

D[i][j] – можно ли из профиля i попасть в профиль j

Из профиля i в профиль j можно перейти если выполняются условия:

- Можно положить горизонтальные домино. То есть там где в j профиле стоит 1, в i профиле должен стоять 0.
- Можно доложить в оставшиеся клетки вертикальные домино. То есть оставшиеся 0 в i профиле должны образовывать четные подстроки.

Задача о замощении домино



A[i][p] – количество способов замостить первые i столбцов так, чтобы они заканчивались на p-ый профиль

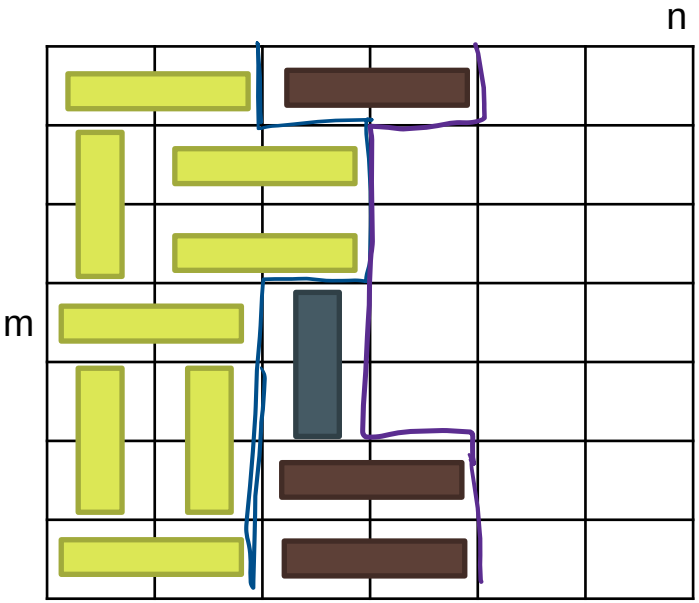
D[i][j] – можно ли из профиля i попасть в профиль j

A[i][p] -?

Из профиля i в профиль j можно перейти если выполняются условия:

- Можно положить горизонтальные домино. То есть там где в j профиле стоит 1, в i профиле должен стоять 0.
- Можно доложить в оставшиеся клетки вертикальные домино. То есть оставшиеся 0 в i профиле должны образовывать четные подстроки.

Задача о замощении домино



A[i][p] – количество способов замостить первые i столбцов так, чтобы они заканчивались на p-ый профиль

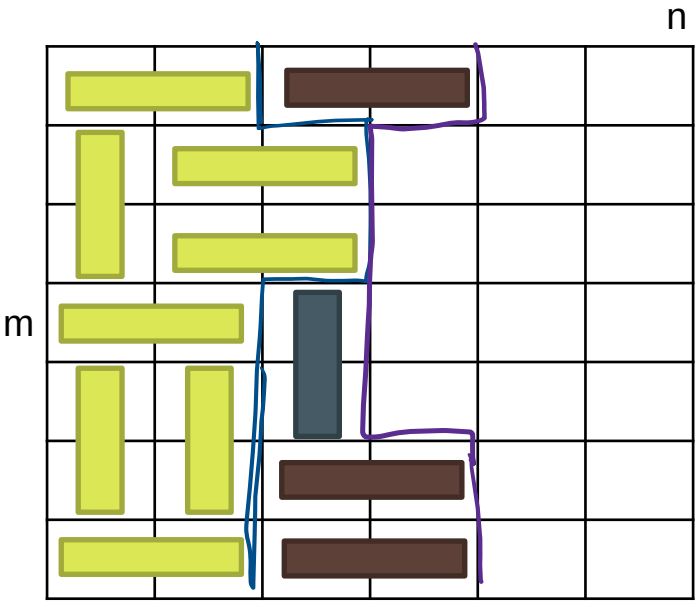
D[i][j] – можно ли из профиля i попасть в профиль j

$$A[i][p] = \sum_{\text{по всем профилям } j} A[i-1][j] \cdot d[j][p]$$

Из профиля i в профиль j можно перейти если выполняются условия:

- Можно положить горизонтальные домино. То есть там где в j профиле стоит 1, в i профиле должен стоять 0.
- Можно доложить в оставшиеся клетки вертикальные домино. То есть оставшиеся 0 в i профиле должны образовывать четные подстроки.

Задача о замощении домино



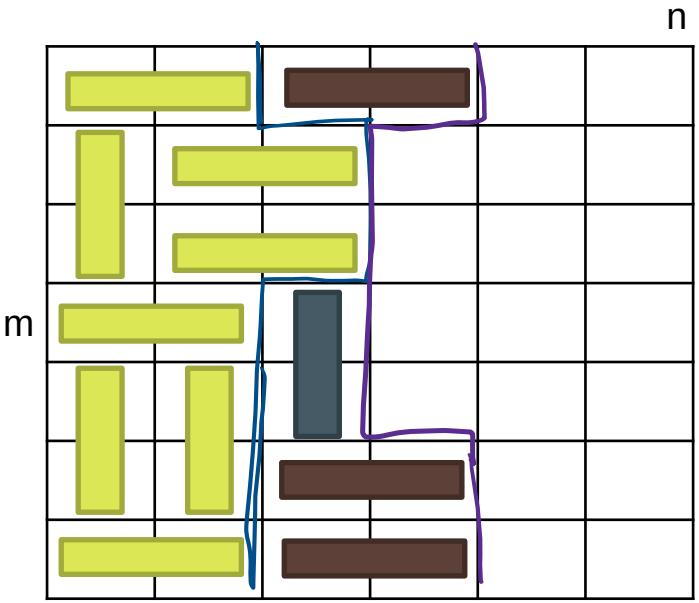
A[i][p] – количество способов замостить первые i столбцов так, чтобы они заканчивались на p-ый профиль

D[i][j] – можно ли из профиля i попасть в профиль j

$$A[i][p] = \sum_{\text{по всем профилям } j} A[i-1][j] \cdot d[j][p]$$

Где живет ответ?

Задача о замощении домино



A[i][p] – количество способов замостить первые i столбцов так, чтобы они заканчивались на p-ый профиль

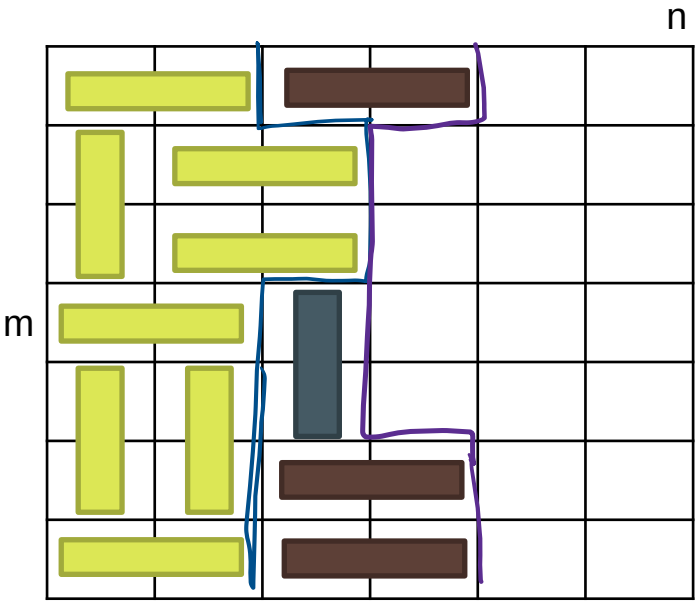
D[i][j] – можно ли из профиля i попасть в профиль j

$$A[i][p] = \sum_{\text{по всем профилям } j} A[i-1][j] \cdot d[j][p]$$

Где живет ответ?

Ответом будет $\sum a[m][i]$, где i — профиль, который может быть последним (т.е. все группы из 0 имеют четные размеры).

Задача о замощении домино



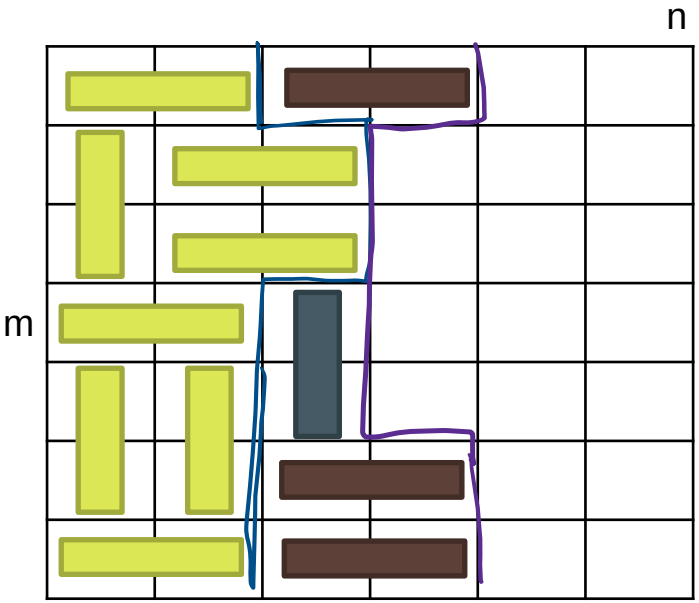
A[i][p] – количество способов замостить первые i столбцов так, чтобы они заканчивались на p-ый профиль

D[i][j] – можно ли из профиля i попасть в профиль j

$$A[i][p] = \sum_{\text{по всем профилям } j} A[i-1][j] \cdot d[j][p]$$

Сложность?

Задача о замощении домино



A[i][p] – количество способов замостить первые i столбцов так, чтобы они заканчивались на p-ый профиль

D[i][j] – можно ли из профиля i попасть в профиль j

$$A[i][p] = \sum_{\text{по всем профилям } j} A[i-1][j] \cdot d[j][p]$$

Сложность?

Оценка сложности: подсчет d — 2^{2m}, и подсчет a — $2^{2m} n$ в итоге $O(2^{2m} n)$.

Задача о замощении домино

```
// n, m – размер таблицы
  for i = 0..(1 << n) - 1
    for j = 0..(1 << n) - 1
        if можно перейти из i в j профиль
           d[i][j] = 1
        else
            d[i][j] = 0
a[0][0] = 1 // Так как мы можем начать только с профиля где все клетки 0
for k = 1..m - 1
    for i = 0..(1 << n) - 1
        for j = 0..(1 << n) - 1
            a[k][i] = a[k][i] + a[k - 1][j] · d[j][i]
ans = 0
for i = 0..(1 << n) - 1
    if можно закончить i профилем
        ans = ans + a[m - 1][i]
return ans
```

для каждой пары профилей проверяем совместимость

составляем динамику для всех профилей

проверяем профили они ОК для конца

Итоги

Рассмотрели три способа задавать динамику:

1. Динамика на префиксе (рюкзак, подпоследовательности, редакционное расстояние)
2. Динамика по подотрезкам (расстановка знаков)
3. Динамика по профилю (замощение доминошками)

Также стоит помнить, что динамику можно делать «вперед» (итеративно) и «назад» (рекурсивно).