Динамическое программирование

Что это такое?

Динамическое программирование — это когда у нас есть задача, которую непонятно как решать, и мы разбиваем ее на меньшие задачи, которые тоже непонятно как решать.

Основные принципы

1) Что считаем?
2) Пересчёт
3) Начальные значения
4) Порядок пересчёта
5) Где ответ?

Пример

Числа Фибоначчи

Пример

Числа Фибоначчи – найти N-е число

1) Что считаем?
2) Пересчёт
3) Начальные значения
4) Порядок пересчёта
5) Где ответ?

Пример

Числа Фибоначчи – найти N-е число

1) Что считаем? F[i] – i-е число Фибоначчи

2) Пересчёт

3) Начальные значения

4) Порядок пересчёта

5) Где ответ?

Пример

Числа Фибоначчи – найти N-е число

1) Что считаем? F[i] – i-е число Фибоначчи

2) Пересчёт

3) Начальные значения

4) Порядок пересчёта

5) Где ответ? F[N]

Пример

Числа Фибоначчи – найти N-е число

1) Что считаем? F[i] – i-е число Фибоначчи
2) Пересчёт F[i] = F[i - 1] + F[i - 2]
3) Начальные значения
4) Порядок пересчёта
5) Где ответ? F[N]

Пример

Числа Фибоначчи – найти N-е число

1) Что считаем? F[i] – i-е число Фибоначчи

2) Пересчёт F[i] = F[i - 1] + F[i - 2]

3) Начальные значения F[0] = 0, F[1] = 1

4) Порядок пересчёта

5) Где ответ? F[N]

Пример

Числа Фибоначчи – найти N-е число

1) Что считаем? F[i] – i-е число Фибоначчи

2) Пересчёт F[i] = F[i - 1] + F[i - 2]

3) Начальные значения F[0] = 0, F[1] = 1

4) Порядок пересчёта for i = 2 .. N

5) Где ответ? F[N]

Задача 1

Исполнитель Калькулятор работает с целыми числами. Он хранит в памяти одно число и может выполнять с ним два действия:

1) прибавить 1

2) умножить на 3

Программа для Калькулятора – это последовательность команд, в которой могут использоваться только эти две команды (неограниченное число раз). Определите количество различных программ, которые преобразуют число 1 в число 20.

Решение

1) Что считаем?
2) Пересчёт
3) Начальные значения
4) Порядок пересчёта
5) Где ответ?

Решение

1) Что считаем? dp[i] – число способов получить i
2) Пересчёт
3) Начальные значения
4) Порядок пересчёта
5) Где ответ?

Решение

1) Что считаем? dp[i] – число способов получить i
2) Пересчёт
3) Начальные значения
4) Порядок пересчёта
5) Где ответ? dp[20]

Решение

1) Что считаем? dp[i] – число способов получить i
2) Пересчёт
3) Начальные значения dp[1] = 1
4) Порядок пересчёта
5) Где ответ? dp[20]

Решение

1) Что считаем? dp[i] – число способов получить i
2) Пересчёт
3) Начальные значения dp[1] = 1
4) Порядок пересчёта i = 2 .. 20
5) Где ответ? dp[20]

Решение

1) Что считаем? dp[i] – число способов получить i
2) Пересчёт dp[i] = dp[i - 1]
3) Начальные значения dp[1] = 1
4) Порядок пересчёта i = 2 .. 20
5) Где ответ? dp[20]

Решение

1) Что считаем? dp[i] – число способов получить i
2) Пересчёт dp[i] = dp[i - 1] + dp[i / 3], i % 3 == 0
3) Начальные значения dp[1] = 1
4) Порядок пересчёта i = 2 .. 20
5) Где ответ? dp[20]

```
dp[1] = 1
for i = 2 .. 20:
    dp[i] = dp[i - 1]
    if (i % 3 == 0):
        dp[i] += dp[i / 3]
```

Задача 2

Исполнитель Калькулятор работает с целыми числами. Он хранит в памяти одно число и может выполнять с ним два действия:

1) прибавить 1

2) умножить на 3

Программа для Калькулятора – это последовательность команд, в которой могут использоваться только эти две команды (неограниченное число раз). Определите количество различных программ, которые преобразуют число 1 в число 20 и при этом траектория вычислений не содержит числа 9? Например, программа 121 будет состоять из чисел 2, 6 и 7.

Решение

1) Что считаем? dp[i] – число способов получить i
2) Пересчёт dp[i] = dp[i - 1] + dp[i / 3], i % 3 == 0
3) Начальные значения dp[1] = 1
4) Порядок пересчёта i = 2 .. 20
5) Где ответ? dp[20]

```
dp[1] = 1
for i = 2 .. 20:
    dp[i] = dp[i - 1]
    if (i % 3 == 0):
        dp[i] += dp[i / 3]
```

Решение

1) Что считаем? dp[i] – число способов получить i
2) Пересчёт dp[i] = dp[i - 1] + dp[i / 3], i % 3 == 0
3) Начальные значения dp[1] = 1
4) Порядок пересчёта i = 2 .. 20
5) Где ответ? dp[20]

```
dp[1] = 1
for i = 2 .. 20:
    if i == 9:
        dp[i] = 0
    dp[i] = dp[i - 1]
    if (i % 3 == 0):
        dp[i] += dp[i / 3]
```

Задача 3

Исполнитель Калькулятор работает с целыми числами. Он хранит в памяти одно число и может выполнять с ним два действия:

1) прибавить 1

2) умножить на 3

Программа для Калькулятора – это последовательность команд, в которой могут использоваться только эти две команды (неограниченное число раз). Определите количество различных программ, которые преобразуют число 1 в число 20 и при этом траектория вычислений содержит число 9? Например, программа 121 будет состоять из чисел 2, 6 и 7.

Решение

1) Что считаем? dp[i] – число способов получить i
2) Пересчёт dp[i] = dp[i - 1] + dp[i / 3], i % 3 == 0
3) Начальные значения dp[1] = 1
4) Порядок пересчёта i = 2 .. 20
5) Где ответ? dp[20]

```
dp[1] = 1
for i = 2 .. 20:
    dp[i] = dp[i - 1]
    if (i % 3 == 0 && (i < 10 or i >= 27)):
        dp[i] += dp[i / 3]
```

Задача 4

Найти количество последовательностей длины N, состоящих из нулей и единиц, таких, что в них нет двух подряд идущих нулей.

Задача 4

Найти количество последовательностей длины N, состоящих из нулей и единиц, таких, что в них нет двух подряд идущих нулей.

N = 2

01

10

11

Задача 4

Найти количество последовательностей длины N, состоящих из нулей и единиц, таких, что в них нет двух подряд идущих нулей.

N = 3

000	100
001	101
010	110
011	111

Задача 4

Найти количество последовательностей длины N, состоящих из нулей и единиц, таких, что в них нет двух подряд идущих нулей.

N = 3

000 100
001 101
010 110
011 111

Идея

Все хорошие последовательности делятся на два типа:

Идея

Все хорошие последовательности делятся на два типа:

1) Оканчивающиеся на 0

Идея

Все хорошие последовательности делятся на два типа:

1) Оканчивающиеся на 0
2) Оканчивающиеся на 1

Идея

Все хорошие последовательности делятся на два типа:

1) Оканчивающиеся на 0
2) Оканчивающиеся на 1

Ответ – 1) + 2)

Идея

Все хорошие последовательности делятся на два типа:

1) Оканчивающиеся на 0 (dp0)
2) Оканчивающиеся на 1 (dp1)

Ответ – 1) + 2) (dp0 + dp1)

Идея

Все хорошие последовательности делятся на два типа:

1) Оканчивающиеся на 0 (dp0)
2) Оканчивающиеся на 1 (dp1)

Ответ – 1) + 2) (dp0 + dp1)

Как получить последовательность длины N?

Идея

Все хорошие последовательности делятся на два типа:

1) Оканчивающиеся на 0 (dp0)
2) Оканчивающиеся на 1 (dp1)

Ответ – 1) + 2) (dp0 + dp1)

Как получить последовательность длины N?
Дописать в конец последовательности длины N – 1 0 или 1

Идея

Все хорошие последовательности делятся на два типа:

Посчитаем dp0[i] и dp1[i]

Идея

Все хорошие последовательности делятся на два типа:

Посчитаем dp0[i] и dp1[i]

dp0[i] =

Идея

Все хорошие последовательности делятся на два типа:

Посчитаем dp0[i] и dp1[i]

dp0[i] = dp1[i - 1]

Идея

Все хорошие последовательности делятся на два типа:

Посчитаем dp0[i] и dp1[i]

dp0[i] = dp1[i - 1]

dp1[i] =

Идея

Все хорошие последовательности делятся на два типа:

Посчитаем dp0[i] и dp1[i]

dp0[i] = dp1[i - 1]

dp1[i] = dp1[i - 1] + dp0[i - 1]

Идея

Все хорошие последовательности делятся на два типа:

Посчитаем dp0[i] и dp1[i]

dp0[i] = dp1[i - 1]

dp1[i] = dp1[i - 1] + dp0[i - 1]

Ответ: dp0[N] + dp1[N]

Идея

Все хорошие последовательности делятся на два типа:

Посчитаем dp0[i] и dp1[i]

dp0[i] = dp1[i - 1]
dp1[i] = dp1[i - 1] + dp0[i - 1]
Ответ: dp0[N] + dp1[N]
База: dp0[1] = dp1[1] = 1

Задача 5

Прямоугольник разлинован на N * M клеток (1 < N, M <= 30). Исполнитель Робот может перемещаться по клеткам, выполняя за одно перемещение одну из двух команд: вправо или вниз. По команде вправо Робот перемещается в соседнюю правую клетку, по команде вниз - в соседнюю нижнюю. При попытке выхода за границу квадрата Робот разрушается. На поле могут быть стенки. При врезании в стенку робот разрушается. Перед каждым запуском Робота в каждой клетке квадрата лежит монета достоинством от 1 до 100. Посетив клетку, Робот забирает монету с собой; это также относится к начальной и конечной клетке маршрута Робота.

Определите максимальную и минимальную денежную сумму, которую может собрать Робот, пройдя из левой верхней клетки в правую нижнюю. В ответе укажите два числа без пробела - сначала максимальную сумму, затем минимальную.

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53					

Dp[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53					

Dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53					

Dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134				

Dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153			

Dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198		

Dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	

Dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)

В крайние клетки можно попасть только по одному направлению

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273

Dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)

В крайние клетки можно попасть только по одному направлению

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119					

Dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119					
168					

Dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119					
168					
189					

Dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119					
168					
189					
257					

Dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119					
168					
189					
257					

Dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению
Dp[0][i] =

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119					
168					
189					
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению
dp[0][i] = dp[0][i - 1] + a[0][i],

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119					
168					
189					
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению
dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M - 1

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119					
168					
189					
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)

В крайние клетки можно попасть только по одному направлению

dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1

dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119					
168					
189					
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)

В крайние клетки можно попасть только по одному направлению

dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1

dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1

Как считать dp[i][j]?

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119					
168					
189					
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению
dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1
dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1
Как считать dp[i][j]? Это минимум из значений сверху и слева

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119					
168					
189					
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)

В крайние клетки можно попасть только по одному направлению

dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1

dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1

Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144				
168					
189					
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)

В крайние клетки можно попасть только по одному направлению

dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1

dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1

Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174			
168					
189					
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению
dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1
dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1
Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210		
168					
189					
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению
dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1
dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1
Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	
168					
189					
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению
dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1
dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1
Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	262
168					
189					
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)

В крайние клетки можно попасть только по одному направлению

dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1

dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1

Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	262
168	225				
189					
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)

В крайние клетки можно попасть только по одному направлению

dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1

dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1

Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	262
168	225	189			
189					
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению
dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1
dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1
Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	262
168	225	189	228		
189					
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению
dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1
dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1
Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	262
168	225	189	228	299	
189					
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению
dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1
dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1
Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	262
168	225	189	228	299	264
189					
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению
dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1
dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1
Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	262
168	225	189	228	299	264
189	278				
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)

В крайние клетки можно попасть только по одному направлению

dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1

dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1

Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	262
168	225	189	228	299	264
189	278	214			
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению
dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1
dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1
Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	262
168	225	189	228	299	264
189	278	214	312		
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению
dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1
dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1
Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	262
168	225	189	228	299	264
189	278	214	312	367	
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)

В крайние клетки можно попасть только по одному направлению

dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1

dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1

Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	262
168	225	189	228	299	264
189	278	214	312	367	343
257					

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)

В крайние клетки можно попасть только по одному направлению

dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1

dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1

Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	262
168	225	189	228	299	264
189	278	214	312	367	343
257	304				

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению
dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1
dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1
Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	262
168	225	189	228	299	264
189	278	214	312	367	343
257	304	275			

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению
dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1
dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1
Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	262
168	225	189	228	299	264
189	278	214	312	367	343
257	304	275	312		

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)

В крайние клетки можно попасть только по одному направлению

dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1

dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1

Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	262
168	225	189	228	299	264
189	278	214	312	367	343
257	304	275	312	406	

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)

В крайние клетки можно попасть только по одному направлению

dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1

dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1

Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	262
168	225	189	228	299	264
189	278	214	312	367	343
257	304	275	312	406	415

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению
dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1
dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1
Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]

Решение для минимума

53	81	19	45	26	49
66	25	30	36	16	36
49	81	15	39	73	2
21	89	25	98	68	79
68	47	61	37	94	72

53	134	153	198	224	273
119	144	174	210	226	262
168	225	189	228	299	264
189	278	214	312	367	343
257	304	275	312	406	415

dp[i][j] – минимальная сумма, которую получаем, попадая в клетку (i, j)
В крайние клетки можно попасть только по одному направлению
dp[0][i] = dp[0][i - 1] + a[0][i], i = 1 .. M – 1
dp[i][0] = dp[i - 1][0] + a[i][0], i = 1 .. N – 1
Как считать dp[i][j]? Это минимум из значений сверху и слева + a[i][j]
dp[i][j] = min(dp[i - 1][j], dp[i][j - 1]) + a[i][j]

Задача 6

Дана числовая последовательность, требуется найти длину наибольшей возрастающей подпоследовательности.

Задача 6

Дана числовая последовательность, требуется найти длину наибольшей возрастающей подпоследовательности.

Пример:

6
3 29 5 5 28 6

Задача 6

Дана числовая последовательность, требуется найти длину наибольшей возрастающей подпоследовательности.

Пример:

6
3 29 5 5 28 6

Ответ – 3

Задача 6

Дана числовая последовательность, требуется найти длину наибольшей возрастающей подпоследовательности.

Пример:

6
3 29 5 5 28 6

Ответ – 3 (3, 5, 6 – 2 раза)

Задача 6

Дана числовая последовательность, требуется найти длину наибольшей возрастающей подпоследовательности.

Пример:

6
3 29 5 5 28 6

Ответ – 3 (3, 5, 6 – 2 раза, 3 5 28 – 2 раза)

Решение

Решение

dp[i] – длина наидлиннейшей ВП, последний элемент которой равен a[i]

Решение

dp[i] – длина наидлиннейшей ВП, последний элемент которой равен a[i]

База: dp[i] = 1

Решение

dp[i] – длина наидлиннейшей ВП, последний элемент которой равен a[i]

База: dp[i] = 1

Решение

dp[i] – длина наидлиннейшей ВП, последний элемент которой равен a[i]

База: dp[i] = 1

Как делать пересчёт?

Решение

dp[i] – длина наидлиннейшей ВП, последний элемент которой равен a[i]

База: dp[i] = 1

Как делать пересчёт?

a[i] может быть последним элементом, если он больше всех, в том числе предпоследнего

Решение

dp[i] – длина наидлиннейшей ВП, последний элемент которой равен a[i]

База: dp[i] = 1

Как делать пересчёт?

a[i] может быть последним элементом, если он больше всех, в том числе предпоследнего

Просмотрим все элементы левее и меньше a[i] и посмотрим длины последовательностей, которые они образуют

Решение

dp[i] – длина наидлиннейшей ВП, последний элемент которой равен a[i]

База: dp[i] = 1

Как делать пересчёт?

a[i] может быть последним элементом, если он больше всех, в том числе предпоследнего

Просмотрим все элементы левее и меньше a[i] и посмотрим длины последовательностей, которые они образуют. Выгоднее взять элемент, который последний в самой длинной последовательности

Решение

dp[i] – длина наидлиннейшей ВП, последний элемент которой равен a[i]

База: dp[i] = 1

Как делать пересчёт?

a[i] может быть последним элементом, если он больше всех, в том числе предпоследнего

Просмотрим все элементы левее и меньше a[i] и посмотрим длины последовательностей, которые они образуют.
Выгоднее взять элемент, который последний в самой длинной последовательности
dp[i] = max(dp[j]) + 1

Решение

dp[i] – длина наидлиннейшей ВП, последний
элемент которой равен a[i]

База: dp[i] = 1

Как делать пересчёт?

a[i] может быть последним элементом, если он
больше всех, в том числе предпоследнего

Просмотрим все элементы левее и меньше a[i] и
посмотрим длины последовательностей, которые они образуют.
Выгоднее взять элемент, который последний в самой длинной
последовательности
$dp[i] = \max(dp[j]) + 1, j < i, a[j] < a[i]$

Решение

dp[i] – длина наидлиннейшей ВП, последний элемент которой равен a[i]

База: dp[i] = 1

Как делать пересчёт?

a[i] может быть последним элементом, если он больше всех, в том числе предпоследнего

Просмотрим все элементы левее и меньше a[i] и посмотрим длины последовательностей, которые они образуют.
Выгоднее взять элемент, который последний в самой длинной последовательности
dp[i] = max(dp[j]) + 1, j < i, a[j] < a[i] (a[j] – предпоследний, a[i] – последний)

Решение

dp[i] – длина наидлиннейшей ВП, последний
элемент которой равен a[i]

База: dp[i] = 1

Как делать пересчёт?

a[i] может быть последним элементом, если он
больше всех, в том числе предпоследнего

Просмотрим все элементы левее и меньше a[i] и
посмотрим длины последовательностей, которые они образуют.
Выгоднее взять элемент, который последний в самой длинной
последовательности
dp[i] = max(dp[j]) + 1, j < i, a[j] < a[i] (a[j] – предпоследний, a[i] – последний)
Где ответ?

Решение

dp[i] – длина наидлиннейшей ВП, последний элемент которой равен a[i]

База: dp[i] = 1

Как делать пересчёт?

a[i] может быть последним элементом, если он больше всех, в том числе предпоследнего

Просмотрим все элементы левее и меньше a[i] и посмотрим длины последовательностей, которые они образуют.
Выгоднее взять элемент, который последний в самой длинной последовательности
dp[i] = max(dp[j]) + 1, j < i, a[j] < a[i] (a[j] – предпоследний, a[i] – последний)
Где ответ? Это максимальное значение в массиве dp.