

академия
больших
данных

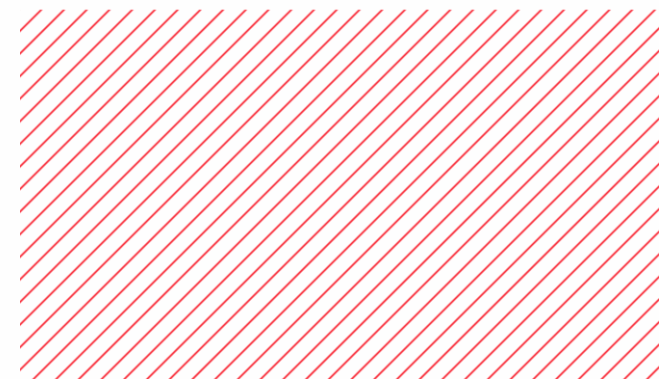
mail.ru
group



Продвинутое динамическое программирование

Шовкоплас Григорий

Алгоритмы и структуры данных Advanced



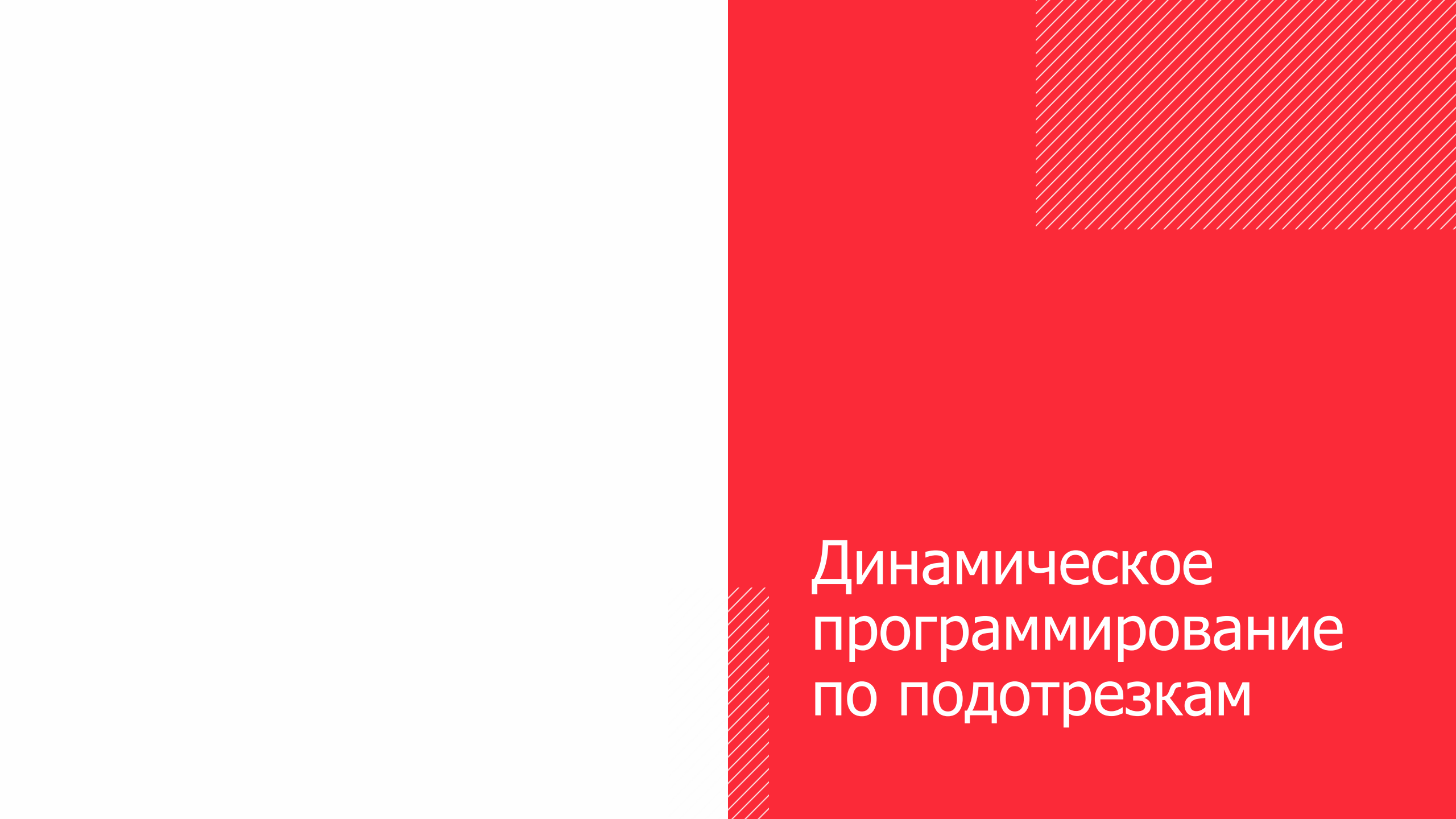


Каким еще бывает
динамическое
программирование?



«Виды» динамического программирования

- Рассмотрено:
 - ДП по префиксам
- Что еще бывает?
 - ДП по подотрезкам
 - ДП по поддеревьям
 - ДП по подмножествам
 - ДП по профилю



Динамическое программирование по подотрезкам



ДП по подотрезкам

- Решить задачу для подотрезка = решить задачу для массива меньшего размера
- Знаем как решать для отрезков меньшей длины
 - → знаем как решать для отрезка



Порядок перемножения матриц

- Дана последовательность матриц размеров $n_0 \times n_1, n_1 \times n_2 \dots$
- Требуется найти оптимальный порядок их перемножения
- $A - 10 \times 30, B - 30 \times 5, C - 5 \times 60$
 - $(A \times B) \times C: (10 \times 30 \times 5) + (10 \times 5 \times 60) = 4500$
 - $A \times (B \times C): (30 \times 5 \times 60) + (10 \times 30 \times 60) = 27000$

Порядок перемножения матриц

- Что храним в ДП?
 - База
 - Переход
 - Порядок обхода
 - Где ответ?
- $dp[i][j]$ — минимальное число операций на отрезке $[i; j)$
 - $dp[i][i+1] = 0$
 - $dp[i][j] = \min_{k=i}^j dp[i][k] + dp[k][j] + n[i-1]*n[k]*n[j]$
 - По убыванию i , По возрастанию j
 - $dp[0][n]$

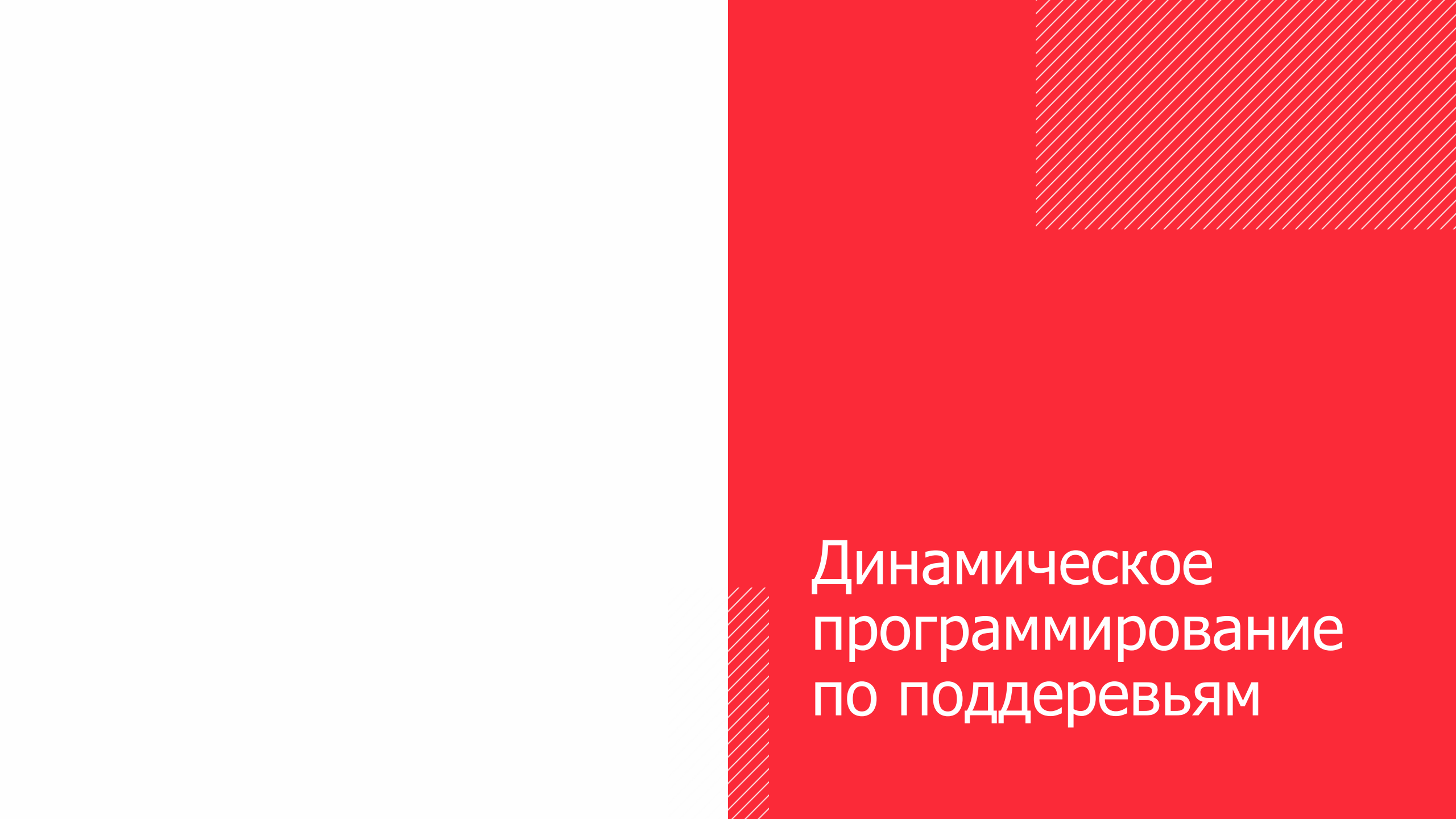


Расстановка знаков в выражении

- Дана последовательность чисел
- Можно ставить знаки: «+», «x» и скобки
- Требуется получить наибольшее значение выражения

Расстановка знаков в выражении

- Что храним в ДП?
 - База
 - Переход
 - Порядок обхода
 - Где ответ?
- $dp[i][j]$ – максимальный ответ на отрезке $[i; j)$
 - $dp[i][i+1] = a[i]$
 - $dp[i][j] = \max_{k=i}^j \max(dp[i][k] + dp[k][j], dp[i][k] * dp[k][j])$
 - По убыванию i , По возрастанию j
 - $dp[0][n]$



Динамическое программирование по поддеревьям



ДП по поддеревьям

- Решить задачу для поддерева = решить задачу для дерева меньшего размера
- Знаем как решать для поддеревьев
 - → знаем как решать для дерева



Поиск максимального паросочетания в дереве

- Дано дерево
- Найти максимальное паросочетание
 - Наибольшее число ребер, без общих вершин

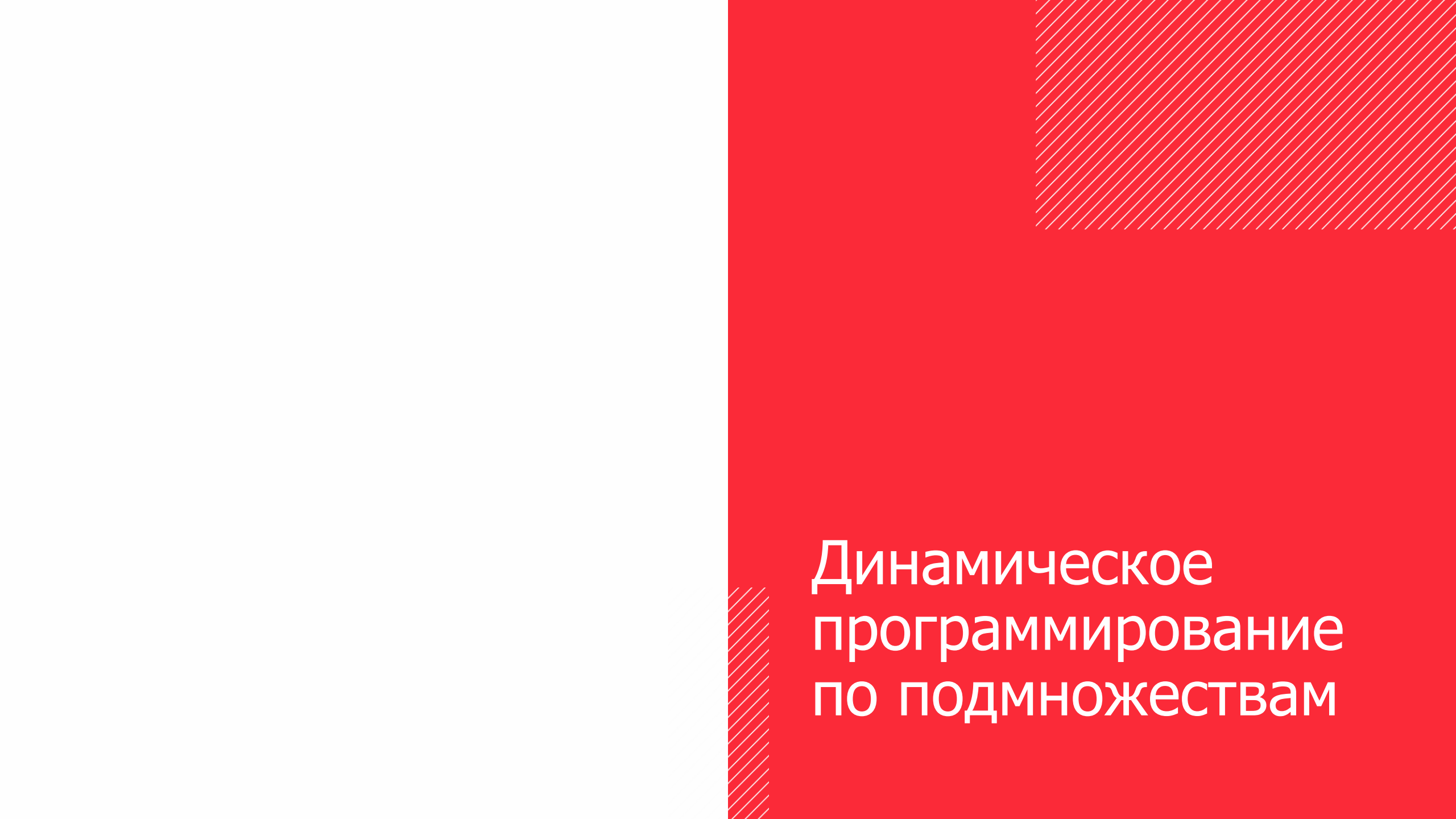
Поиск максимального паросочетания в дереве

- Что храним в ДП?
 - База
 - Переход
 - Порядок обхода
 - Где ответ?
- $dp[v][0]$ – максимальный ответ для поддерева v , если v **не входит** в паросочетание;
 - $dp[v][1]$ – максимальный ответ для поддерева v , если v **входит** в паросочетание;
 - $dp[leaf][0] = 0; dp[leaf][1] = -1$
 - $dp[v][0] = \sum_c \max(dp[c][0], dp[c][1])$
 - $dp[v][1] = \sum_{c \neq u} \max(dp[c][0], dp[c][1]) + dp[u][0] + 1$
 - Обратный DFS
 - $\max(dp[root][0], dp[root][1])$



ДП по поддеревьям

- Максимальное взвешенное паросочетание
- Проверка корректности красно-черного дерева
- Сумма длин всех путей в дереве



Динамическое программирование по подмножествам



ДП по подмножествам

- Решить задачу для подмножества = решить задачу для множества меньшего размера 😊
- Знаем как решать для подмножеств
 - → знаем как решать для множества




Битовые маски

- Как хранить подмножество?
 - Массив булеанов `used[n]` – содержится i -й элемент в данном подмножестве или нет?
 - 2^n вариантов
 - $n < 30 \rightarrow$ закодируем целым числом 32-битного типа данных
 - i -й бит соответствует i -му элементу массива



Битовые маски

i -й элемент битовой маски

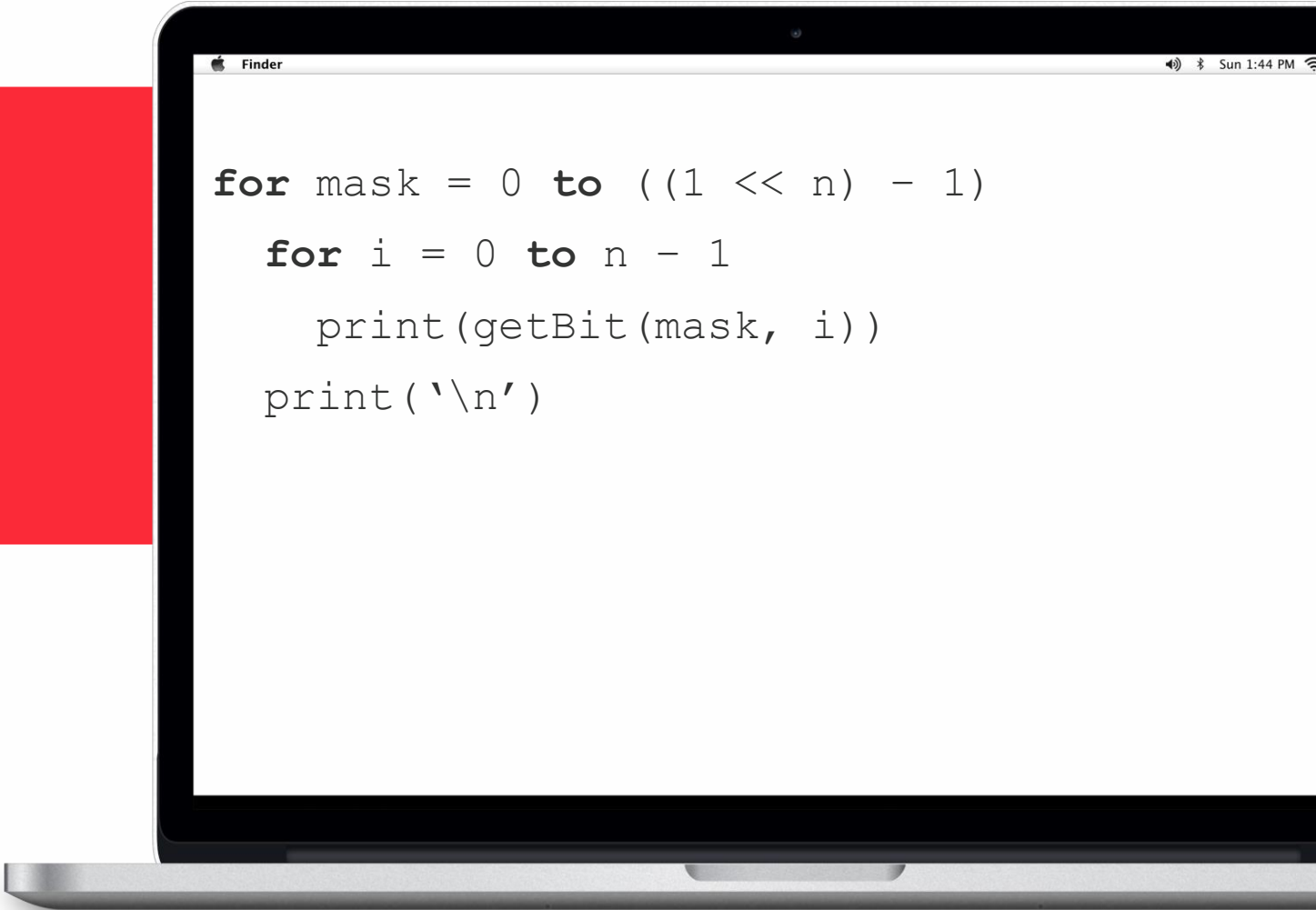


```
getBit(mask, i)  
    return (mask >> i) & 1
```



Битовые маски

Перебор масок всех
подмножеств



```
for mask = 0 to ((1 << n) - 1)
  for i = 0 to n - 1
    print(getBit(mask, i))
  print('\n')
```



Поиск максимального паросочетания в **графе**

- Дан **граф**
- Найти максимальное паросочетание
 - Наибольшее число ребер, без общих вершин

Поиск максимального паросочетания в графе

- Что храним в ДП?
 - База
 - Переход
 - Порядок обхода
 - Где ответ?
- $dp[mask]$ – максимальный ответ для подмножества $mask$
 - $dp[0] = 0$
 - $dp[mask] = \max(dp[mask'], dp[mask''] + 1)$
 - $mask' = mask - 2^i$
 - $mask'' = mask - 2^i - 2^j$, есть ребро (i, j)
 - По возрастанию $mask$
 - $dp[2^n - 1]$

Поиск максимального паросочетания в графе

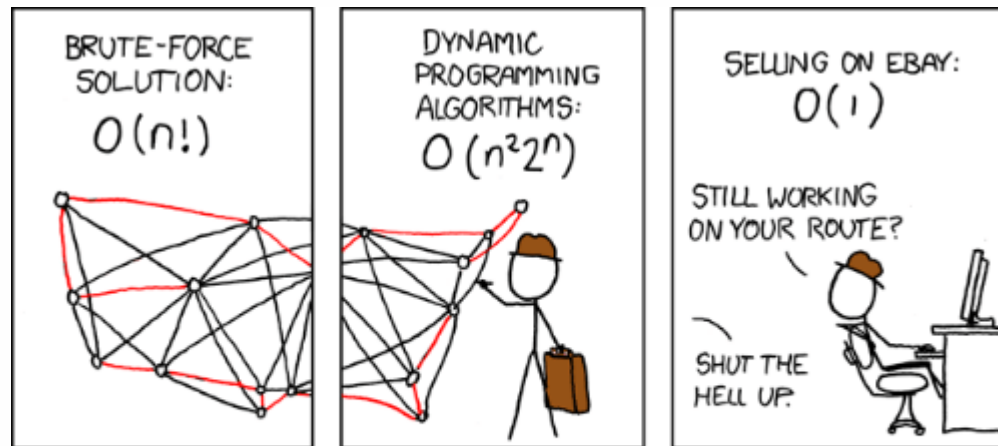
Как это писать?

Как слышится — так и пишется

```
dp[0] = 0
for mask = 1 to ((1 << n) - 1)
  for i = 0 to n - 1
    if getBit(mask, i) = 1
      dp[mask] = dp[mask - (1 << i)]
      for j = 0 to n - 1
        if getBit(mask, j) = 1 && isE[i][j]
          dp[mask] = max(dp[mask],
                        dp[mask - (1 << i) - (1 << j)] + 1)
print(dp[(1 << n) - 1])
```

Задача коммивояжёра

- Дан граф
- Нужно побывать в каждой вершине ровно один раз
- Найти такой маршрут
- Вернуться в исходный город
- *Гамильтонов цикл*



The Traveling Salesman...



Задача коммивояжёра

- Что храним в ДП?
- База
- Переход
- Порядок обхода
- Где ответ?
- $dp[v][mask]$ – можно ли прийти в вершину v посетив вершины $mask$
- $dp[start][0] = \text{True}; dp[v][i] = \text{False}$
- $dp[v][mask] = OR_{\text{есть ребро } (u,v)} dp[u][mask - 2^u]$
- По возрастанию $mask$
- $dp[start][2^n - 1]$

ДП по профилю

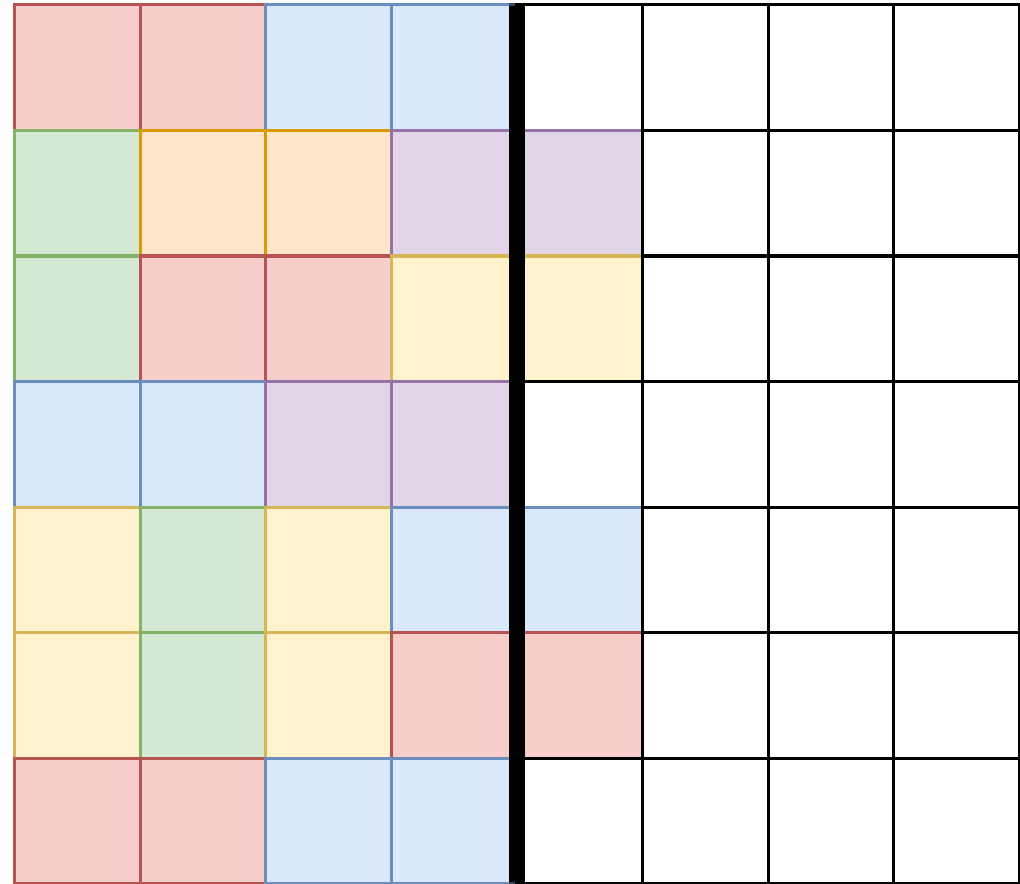


Замощение доминошками

- Есть поле $N \times M$
- Некоторые клетки вырезали
- Сколько есть способов замостить его доминошками 1×2 и 2×1 ?

Что такое профиль?

- Замостили первые i столбцов
- Что-то осталось торчать
- `mask` для торчащего – профиль
 - 0110110

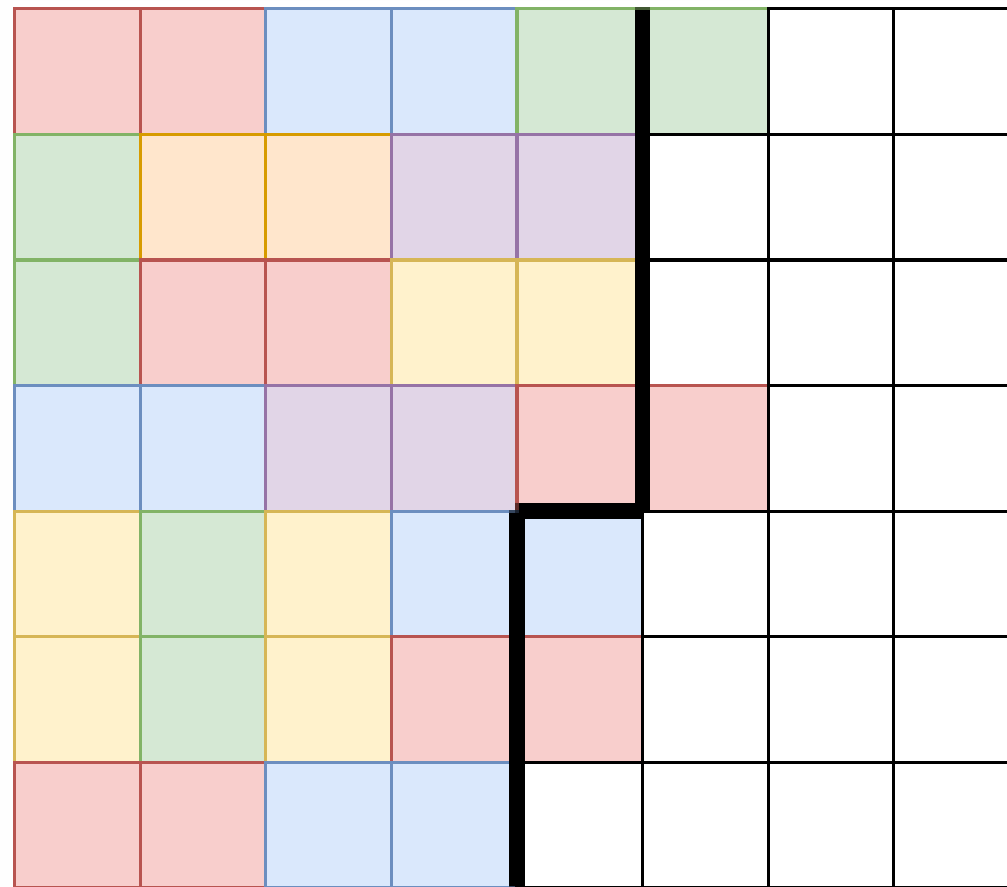


Замоещение доминошками

- Что храним в ДП?
- База
- Переход
- Порядок обхода
- Где ответ?
- $dp[i][mask]$ – число способов заполнить первые i столбцов, чтобы торчал профиль $mask$
- $dp[0][0] = 1; dp[0][i] = 0$
- $dp[i][mask] = \sum dp[i-1][mask']$
 - $mask'$ «дружит» с $mask$
- По возрастанию i
- $dp[n][0]$

Изломанный профиль?

- Замостили первые i **клеток**
- Что-то осталось торчать
- `mask` для торчащего – профиль
 - 1101001



Замоещение доминошками

- Что храним в ДП?
- База
- Переход
- Порядок обхода
- Где ответ?
- $dp[x][mask]$ – число способов заполнить первые x **клеток**, чтобы торчал профиль $mask$
- $dp[0][0] = 1; dp[0][i] = 0$
- $dp[x+1][mask \gg 1] += dp[x][mask]$
 - ничего не кладем
- $dp[x+1][mask'] += dp[x][mask]$
 - положили горизонтальную: $mask' = (mask \gg 1) + 2^{n-1}$
- $dp[x+1][mask''] += dp[x][mask]$
 - положили вертикальную: $mask'' = (mask \gg 1) + 1$
 - x не последняя клетка в столбце, $(mask \& 2) = 0$
- По возрастанию x
- $dp[nm][0]$



Bce!