ТУСУР
TUSUR UNIVERSITY

Динамическое программирование

Лекция 12

Доцент каф. АСУ: Суханов А.Я.

Введение

- **Динамическое программирование** (ДП) — это метод оптимизации, используемый для решения задач, которые можно разбить на перекрывающиеся подзадачи. Оно применяется в алгоритмах, где оптимальное решение большой задачи строится из оптимальных решений её подзадач.

$$z = \sum_{i=0}^{n} f_j(x_j)$$

Основные понятия динамического программирования

Динамическое программирование (ДП) решает задачи, которые обладают двумя ключевыми свойствами:

1.Принцип оптимальности:
Решение задачи можно построить на основе оптимальных решений её подзадач. Это позволяет разбивать задачу на более простые части.
2.Перекрывающиеся подзадачи:
Многие подзадачи решаются несколько раз. Вместо повторных вычислений их результаты сохраняются (мемоизация) и переи-спользуются.

Пример:
Вычисление чисел Фибоначчи с использованием рекурсии:
•Рекурсивный подход повторяет одинаковые вычисления.
•ДП сохраняет результаты для каждого числа и использует их, сокращая временную сложность O(2^n) до O(n).
ДП эффективно применяется, когда задачи можно описать с помощью состояний и переходов между ними.

Преимущества и недостатки ДП

Преимущества:

•Уменьшает временную сложность за счёт мемоизации.

•Обеспечивает точное решение задач.

Недостатки:

•Требует дополнительной памяти.

•Иногда сложна в реализации.

Этапы решения задач С использованием ДП

ТУСУР
TUSUR UNIVERSITY

- **Разделение на подзадачи:**
  Задачу разбивают на независимые части, которые можно решить по отдельности.
- **Определение состояния:**
  Состояние описывается таблицей или массивом **dp**, где каждая ячейка хранит решение подзадачи.
- **Рекуррентное соотношение:**
  Устанавливается связь текущего состояния с предыдущими. Например, **dp[i]=dp[i−1]+dp[i−2]d** для чисел Фибоначчи.
- **Заполнение таблицы:**
  Решение строится от базовых случаев к более сложным путём заполнения dp.
- **Оптимизация:**
  Для экономии памяти иногда сохраняют только текущие и предыдущие состояния.
- **Пример:**
  В задаче о рюкзаке **dp[i][w]** определяет максимальную стоимость для первых i предметов при вместимости рюкзака w.

Задача о рюкзаке

Задача: Максимизировать стоимость предметов в рюкзаке, ограниченном по весу.

ДП для задачи о рюкзаке:

1.Состояние: **dp[i][w]** — максимальная стоимость при учёте первых i предметов и рюкзака с вместимостью w.

2.Переход:

$$dp[i][w]=max(dp[i-1][w],dp[i-1][w-wi]+vi),$$

где wi — вес, vi — стоимость i-го предмета.

Диаграмма для задачи о рюкзаке

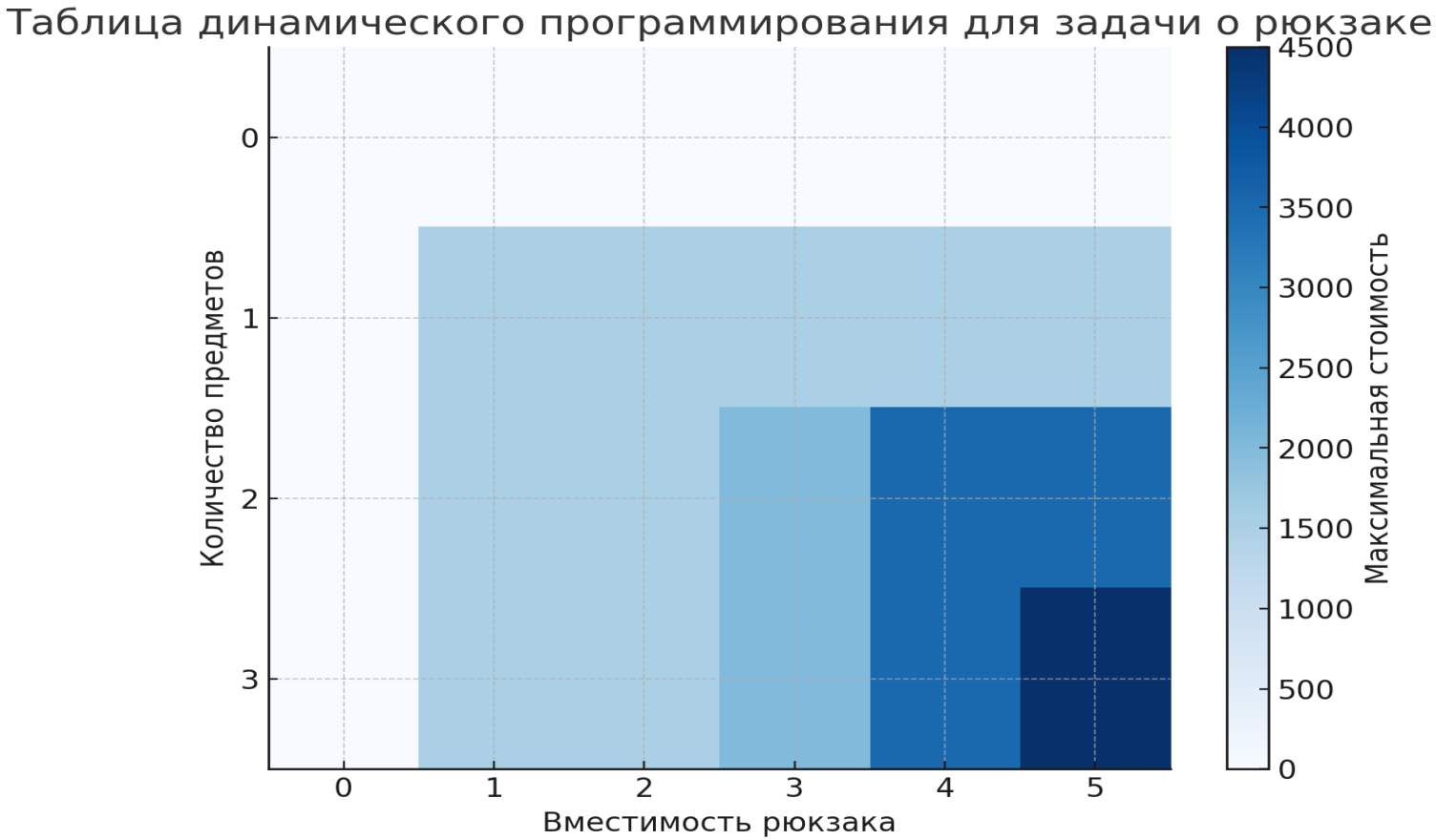

Диаграмма показывает заполнение таблицы динамического программирования для задачи о рюкзаке. Каждая ячейка dp[i][w] представляет максимальную стоимость, которую можно достичь при использовании первых ii предметов и рюкзака вместимости w.

Задача о длиннейшей общей подпоследовательности (LCS)

Задача: Найти длину наибольшей последовательности, которая является подпоследовательностью двух строк.

ДП для LCS:

1.Состояние: dp[i][j] — длина LCS для первых i символов первой строки и j символов второй строки.

2.Переход: $\mathbf{d_P[i][j]} = \begin{cases} \mathbf{dp[i-j][j-1]+1} \\ \mathbf{max(dp[i-1]d_P[i][j-1])} \end{cases}$

если s1[i]=s2[j], иначе.

Задача о длинейшей общей подпоследовательности (LCS)

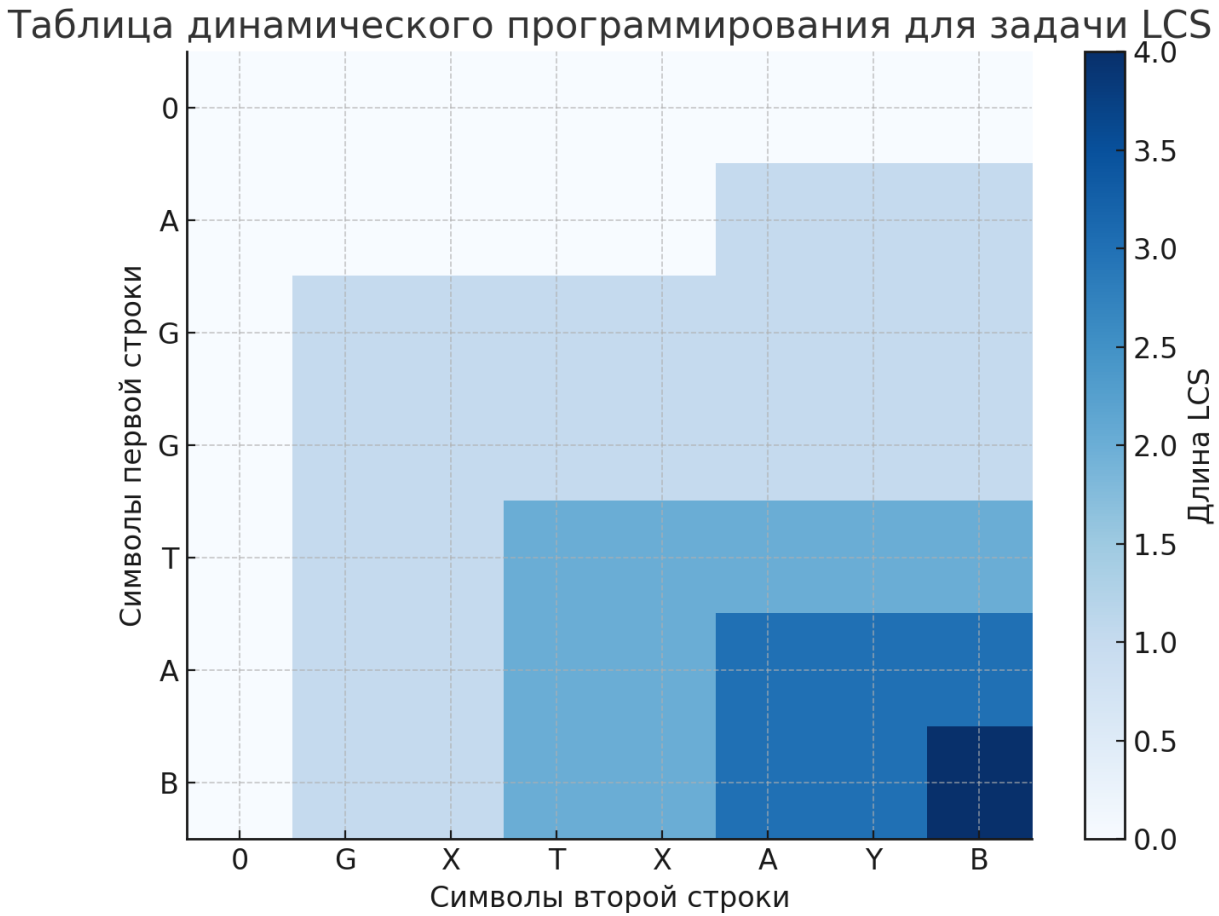

Диаграмма показывает заполнение таблицы динамического программирования для задачи нахождения наибольшей общей подпоследовательности (LCS). Каждая ячейка dp[i][j] содержит длину LCS для первых i символов строки s1 и первых j символов строки s2

Размен монет (Coin Change Problem)

Задача: Найти минимальное количество монет, необходимое для составления заданной суммы.

ДП для размена монет:

1.Состояние: dp[i] — минимальное количество монет для составления суммы i.

2.Переход:

dp[i]=min(dp[i−c1],dp[i−c2],…,dp[i−ck])+1,

где c1, c2,…,ck — номиналы монет.

Пример:

Для суммы 11 и монет [1,2,5]:

- dp[0]=0 (нет монет для суммы 0).
- dp[1]=1 (1 монета номиналом 1).
- И так далее, пока не заполним таблицу.

Размен монет (Coin Change Problem)

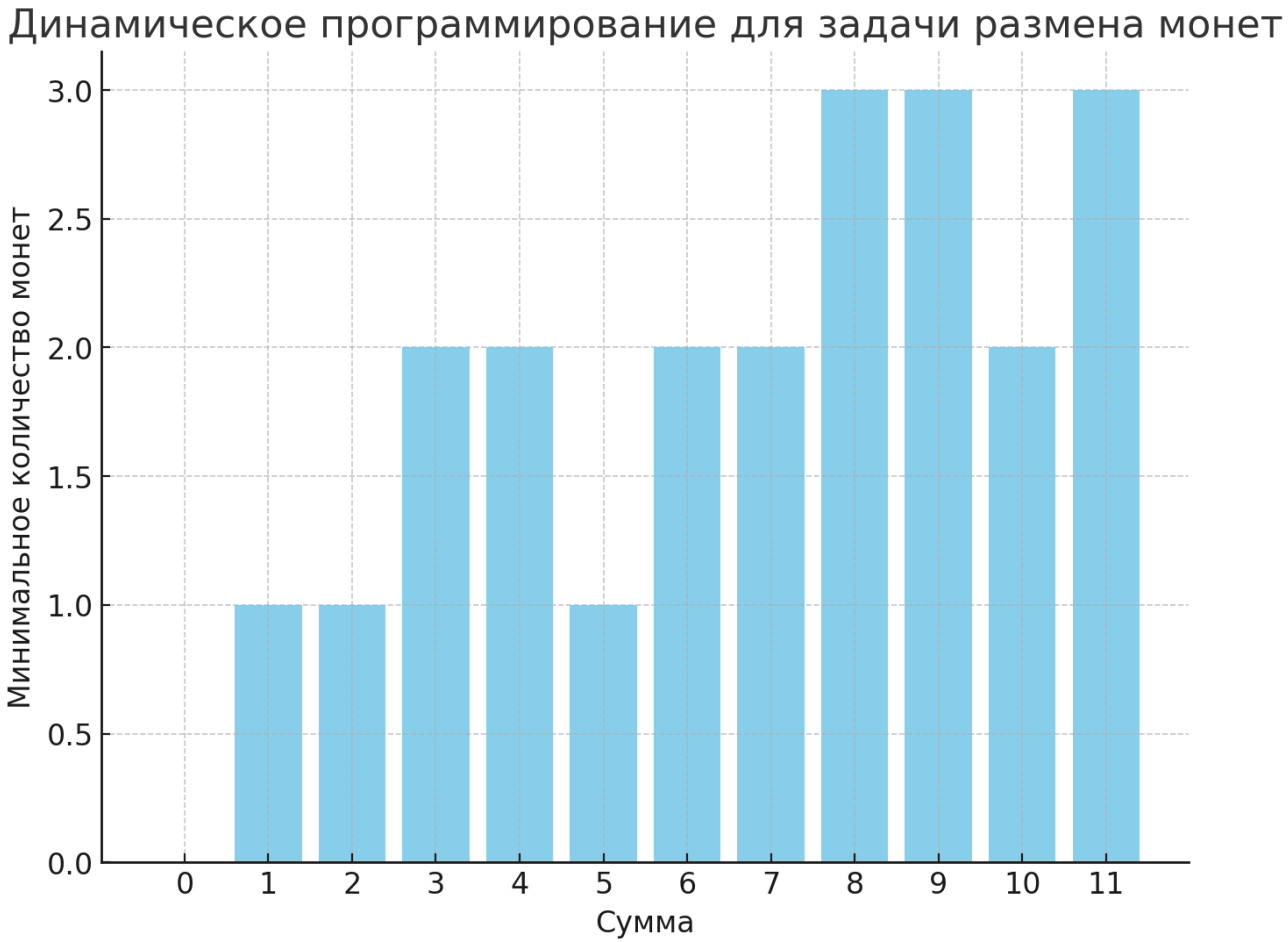

Диаграмма показывает минимальное количество монет, необходимое для составления каждой суммы от 0 до 11. Например, для суммы 11 требуется 3 монеты (5 + 5 + 1). Это демонстрирует, как динамическое программирование позволяет находить оптимальное решение.

Задача о подсчёте путей в матрице

Задача: Найти количество способов добраться из верхнего левого угла матрицы в правый нижний, двигаясь только вправо и вниз.

ДП для подсчёта путей:

1.Состояние: dp[i][j] — количество путей к клетке (i , j).
2.Переход:
dp[i][j]=dp[i−1][j]+dp[i][j−1],
где dp[i−1][j] — пути сверху,
dp[i][j−1] — пути слева.

Задача о подсчёте путей в матрице

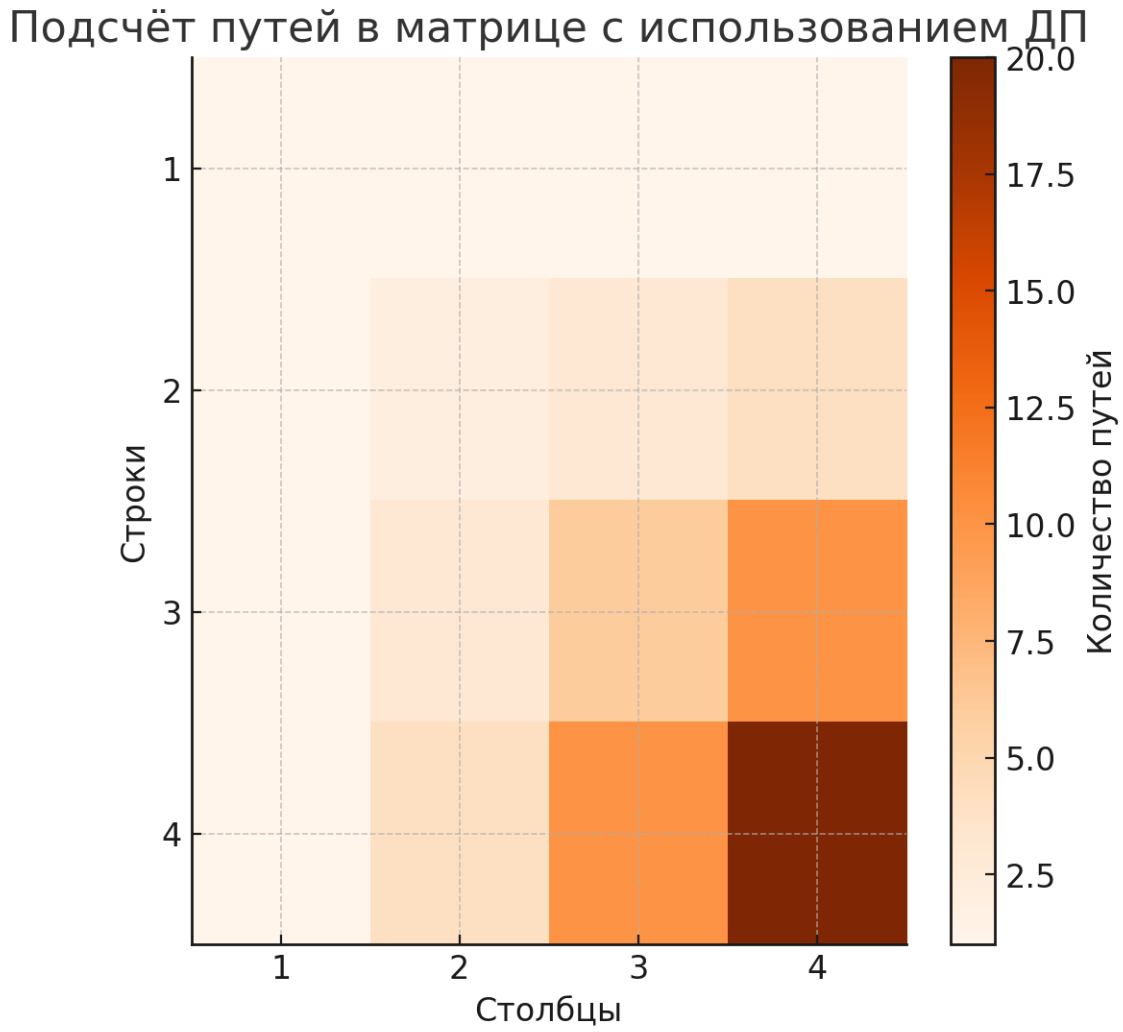

Диаграмма показывает заполнение таблицы для задачи подсчёта путей в матрице 4×4. Каждая ячейка dp[i][j] содержит количество путей, ведущих к этой клетке. Значение в правом нижнем углу (dp[4][4]) равно общему числу путей, равному 20.

Задача об оптимальной последовательности действий

Задача: Найти оптимальный порядок выполнения действий для минимизации затрат, например, умножение матриц или планирование задач.

ДП для оптимальной последовательности:
1.Состояние: **dp[i][j]** — минимальная стоимость выполнения подзадачи от i до j.
2.Переход:
dp[i][j]=min(dp[i][k]+dp[k+1][j]+затраты на объед инение),
где k — точка разбиения.

Применение динамического программирования

Динамическое программирование находит применение во многих областях:

1.Компьютерные науки:

1. Поиск кратчайшего пути (например, алгоритм Флойда).
2. Парсинг в контекстно-свободных грамматиках.

2.Оптимизация:

1. Планирование задач.
2. Задачи маршрутизации.

3.Биоинформатика:

1. Сравнение последовательностей ДНК (задачи LCS).

4.Финансы:

1. Оптимизация инвестиционного портфеля.

Преимущества динамического программирования

- **Эффективность:** снижает временную сложность за счёт пере-использования вычислений.
- **Гибкость:** применяется к широкому классу задач.
- **Надёжность:** обеспечивает точное решение.

Однако важно учитывать дополнительные требования к памяти и сложности реализации.

Заключение

Динамическое программирование — это мощный инструмент, позволяющий решать сложные задачи путём разбиения их на подзадачи. Его применение охватывает множество сфер, от оптимизации и маршрутизации до анализа данных и биоинформатики. Разумный подход к использованию ДП позволяет повысить эффективность вычислений и находить оптимальные решения.

ТУСУР
TUSUR UNIVERSITY

СПАСИБО ЗА ВНИМАНИЕ!

г. Томск, ул. Вершинина, 47, офис 434

e-mail: aleksandr.i.sukhanov@tusur.ru

тел.: (3822) 70-15-36

tusur.ru