Лекция 2

Динамическое программирование. Декомпозиция. Многомерные варианты.

Сергей Леонидович Бабичев

Декомпозиция задачи

- Для задачи о количестве путей до точки i подзадачей было определение количества путей до точек, находящихся в одном шаге от i.
  1. При условии отсутствия циклов размер подзадачи всегда был несколько меньше размера задачи.
  2. Подзадача решалась тем же методом, что и задача.
- Наличие таких условий — подозрение на то, что задачу можно решить методом динамического программирования.

Декомпозиция задачи

- Общая схема динамического программирования:
  1. Можно выделить множество подзадач.
  2. Имеется порядок на подзадачах.
  3. Имеется рекуррентное соотношение решения задачи через решения подзадач.
  4. Рекуррентное соотношение есть рекурсивная функция с целочисленными аргументами.

Декомпозиция задачи

При динамическом программировании:

1. принципиально исключаются повторные вычисления в любой рекурсивной функции если есть возможность запоминать значения функции для аргументов, меньших текущего;
2. снижает время выполнения рекурсивной функции до времени, порядок которого равен сумме времён выполнения всех функций с аргументом, меньших текущего, если затраты на рекурсивный вызов постоянны.

Декомпозиция задачи

- Правильная декомпозиция задачи — ключ к её решению.
- Вернёмся к задаче о рюкзаке.
- Имеется рюкзак объёма V и N предметов весом H_i и стоимостью C_i. Найти комбинацию предметов, имеющую наибольшую суммарную стоимость, которые можно унести.
- Что есть подзадача меньшего размера?

Декомпозиция задачи

- Правильная декомпозиция задачи — ключ к её решению.
- Вернёмся к задаче о рюкзаке.
- Имеется рюкзак объёма V и N предметов весом H_i и стоимостью C_i. Найти комбинацию предметов, имеющую наибольшую суммарную стоимость, которые можно унести.
- Что есть подзадача меньшего размера?
  - Наполнение рюкзака меньшего размера?

Декомпозиция задачи

- Правильная декомпозиция задачи — ключ к её решению.
- Вернёмся к задаче о рюкзаке.
- Имеется рюкзак объёма V и N предметов весом H_i и стоимостью C_i. Найти комбинацию предметов, имеющую наибольшую суммарную стоимость, которые можно унести.
- Что есть подзадача меньшего размера?
  - Наполнение рюкзака меньшего размера?
  - Наполнение рюкзака меньшим количеством предметов?

Декомпозиция задачи

- Требуется ответ на вопросы:
  1. какие ресурсы потребуются для запоминания результатов подзадач?
  2. если имеется решение подзадач, можно ли на основе этого получить решение задачи?
- Для подзадачи «рюкзак меньшего размера»
  1. размер памяти для результатов есть размер рюкзака
  2. если мы знаем результаты для всех меньших рюкзаков V_k, то поможет ли это решить задачу?
- Для подзадачи «рюкзак с меньшим количеством предметов»
  1. размер памяти для результатов есть количество предметов
  2. если мы знаем результаты для всех подзадач с меньшим количеством предметов, то поможет ли это решить задачу?

Декомпозиция задачи

- Задача про банкомат с неограниченным запасом купюр каждого номинала идентична задаче про рюкзак и магазин с неограниченным количеством товара каждой номенклатуры.
- Если количество предметов ограничено, то входные данные задачи — множество предметов, которые можно взять.
- Решение задачи про банкомат здесь не подходит — после выбора одного из предметов множество для выбора изменяется.

Декомпозиция задачи

Декомпозиция по размеру рюкзака.

1. Пусть аргументами подзадачи будут оставшееся множество предметов S и оставшееся место в рюкзаке L.
2. w — веса предметов, c — их стоимости.

$$f(S, L) = \begin{cases} \max\limits_{e \in S} \left(f(S - e, L - w_e) + c_e \right), & \text{если } L > 0 \\ 0, & \text{если } L = 0 \\ -\infty, & \text{если } L < 0 \end{cases}$$

Каков размер пространства аргументов этой задачи?

$$D = O(2^N \cdot L_0) = O(2^N)$$

Декомпозиция задачи

Декомпозиция по количеству предметов.
Пусть мы берёмся за решение задачи с $K+1$ предметами, зная решения всех задач с K предметами.

1. Аргументы подзадачи есть номер K и оставшееся место в рюкзаке L.
   Если у нас есть решение задачи для $K-1$ предмета, то либо мы берём (K)-й предмет, либо нет.

$$F(K,L) = \begin{cases} \max\limits_{i=1}^{K-1}(F(K-1, L-H_i)+C_i, F(K-1,L)) & \text{если } L>0 \\ 0, & \text{если } L=0 \\ -\infty, & \text{если } L<0 \end{cases}$$

Каков размер пространства аргументов этой задачи?

$$D = O(N \cdot L_0)$$

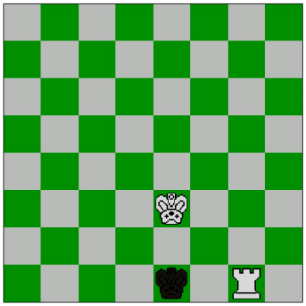
Динамическое программирование и игры.

Динамическое программирование и игры

- Шахматы — антагонистическая конечная игра с наличием цели.
- Деревья игры заканчиваются либо в позициях с оценкой $+\infty$, если выигрывают белые, либо с оценкой 0, если заключительная позиция — ничья, либо $-\infty$, если выигрывают чёрные.
- В конкретно заданной позиции результат обоюдно лучшей игры предопределён.
- Сложность лишь заключается в большом размере дерева игры.

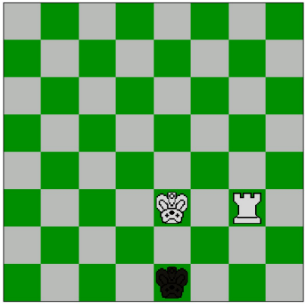
Динамическое программирование и игры

- Назовём позициями ранга 0 те позиции, в которых игра завершилась с каким-либо результатом.



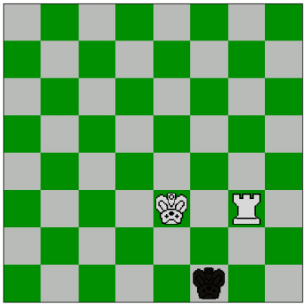
Динамическое программирование и игры

- Позиции ранга один — те позиции, которые при очередном ходе могут привести к позициям ранга 0.



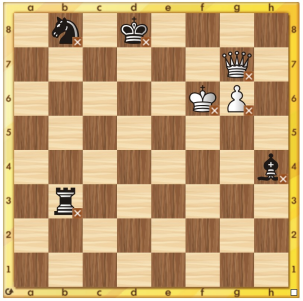
Динамическое программирование и игры

- Позиции ранга два — те позиции, которые при очередном ходе могут привести к позициям ранга 1.



Динамическое программирование и игры

- Шахматы — одна из игр, решаемых динамическим программированием.
- Существует большое множество позиций, особенно с небольшим количеством фигур, для которых известен ранг.
- Сейчас сюда входят *все* позиции с количеством фигур до 7 включительно.



- Например, известны позиции с рангом 1097.
- На 549-м ходу белые ставят мат.

Динамическое программирование и игры

- Табличный подход изобрёл Кен Томпсон в 1970-х годах.
- Эффективную реализацию с помощью динамического программирования — новосибирский математик Евгений Налимов. Таблицы с рангом позиций называются таблицами Налимова.
- Размер таблиц с 7-ю фигурами составляет 140 тибибайт. Расчёт таблиц производился в 2012 году на компьютере «Ломоносов» ВМК МГУ.
- Наилучшая игра обоих сторон заключается в том, чтобы каждым очередным ходом переходить в позицию с тем же результатом и рангом, меньшим ровно на единицу.

Этапы решения задачи методом динамического программирования.

Этапы решения задачи

1. Определяется необходимость именно в динамическом программировании. При быстром уменьшении подзадач задача решается методом «разделяй и властвуй».
2. Определяется максимальный уровень рекурсии в главной задаче. Например, в задаче на покрытие прямоугольника максимальный уровень равен 15.
3. Для нетривиальных задач всегда разрабатывается рекурсивный метод решения задачи.
4. Разрабатывается метод отображения аргументов задачи в результат.
5. Реализуется процесс мемоизации в нисходящем методе.
6. Если максимальный уровень слишком большой, то нисходящий метод неприменим, но по результатам исследования реализуется восходящий метод.

Многомерные варианты.

Расстояние редактирования

- При исправлении слова в текстовом редакторе: три операции:
  - замена одной буквы на другую;
  - удаление лишней буквы;
  - вставка буквы.
- Больше операций редактирования — меньше похожи слова.
- Количество операций — *расстояние редактирования* или *расстояние Левенштейна*.
- Это — мера различия между двумя строками.

Расстояние редактирования

Сколько нужно операций, чтобы превратить слово СЛОН в слово ОГОНЬ?

Один из вариантов:
СЛОН $\rightarrow$ СГОН $\rightarrow$ СГОНЬ $\rightarrow$ ОГОНЬ

Расстояние редактирования: ищем рекурсию

Вопросы:

- Задача ли это динамического программирования?
- Что является рекурсивной функцией, решающей данную задачу?
- Что есть аргументы функции?

Расстояние редактирования: ищем рекурсию

Зафиксируем входные строки — исходную строку `s` и строку назначения `d` (образец). `s` должна превратиться в `d`.

- Как найти более простые подзадачи? Подзадач для более коротких строк-префиксов.
- Что является мерой простоты? Длина строки.

Расстояние редактирования

- Пусть `i` — длина префикса `s`, а `j` — длина префикса `d`.
- Три варианта:
  - Заменить последний символ строки `s` на последний символ строки `d`.
  - Добавить символ к концу строки `s`.
  - Удалить последний символ строки `s`.

Расстояние редактирования

Последовательность переходов СЛОН $\to$ СГОН $\to$ СГОНЬ $\to$ ОГОНЬ не обладает свойством порядка на подзадачах.

Имеется способ с сохранением порядка на подзадачах:

1. В префиксах строк длины 1 С и О меняем букву С на О. СЛОН $\to$ ОЛОН.
2. В префиксах длины 2 меняем Л на Г. ОЛОН $\to$ ОГОН.
3. В префиксах длины 4 добавляем букву Ь. ОГОН $\to$ ОГОНЬ.

Расстояние редактирования: уравнение Беллмана

Введём функцию $F(i, j)$ как решение задачи для префиксов строк `s` и `d` длинами `i` и `j`.

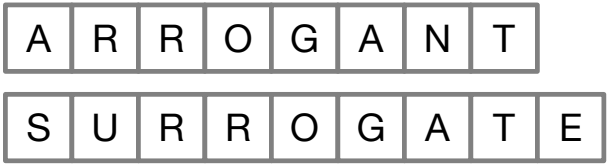
$$F(i,j) = \min \left(\begin{array}{l} F(i-1, j-1), \text{если } s_i = t_i \text{ или } F(i-1, j-1) + 1, \text{если } s_i \neq t_i \\ F(i-1, j) + 1 \\ F(i, j-1) + 1 \end{array} \right).$$

1. Различает случай совпадения последних символов.
2. Мы должны дописать символ в конец `s`.
3. Мы должны удалить последний символ из `s`.

Расстояние редактирования: мемоизация

- Аргументы — дискретные.
- Значений аргументов — плотные.
- Множество пар аргументов конечно и равно $|s| \times |d|$.
- Допускается восходящее решение без рекурсии заполнением таблицы по строкам.

Расстояние редактирования: пример решения



Расстояние редактирования: пример решения

- Добавим лишний левый столбец и лишнюю верхнюю строку, заполним их последовательно возрастающими числами.
- Пустая строка может превратиться в образец за число операций, равное длине образца.
- Непустая строка превращается в пустой образец за число операций, равное длине строки.

		S	U	R	R	O	G	A	T	E
	0	1	2	3	4	5	6	7	8	9
A	1									
R	2									
R	3									
O	4									
G	5									
A	6									
N	7									
T	8									

Расстояние редактирования: пример решения

Заполнение таблицы ведём по строкам. Значение в первой свободной ячейке зависит от трёх элементов таблицы:

		S	U	R	R	O	G	A	T	E
	0	1	2	3	4	5	6	7	8	9
A	1	1								
R	2									
R	3									
O	4									
G	5									
A	6									
N	7									
T	8									

Буквы S столбца и A строки, не совпадают $\rightarrow$ значение, полученное по диагональное стрелке, увеличивается на 1. По горизонтальной и вертикальной стрелке в ячейку приходят увеличенные на 1 числа из тех ячеек.

Расстояние редактирования: пример решения

А вот при совпадении букв штраф за замену отсутствует и в клеточку записывается число 6, значение, полученное по диагональной стрелке.

		S	U	R	R	O	G	A	T	E
	0	1	2	3	4	5	6	7	8	9
A	1	1	2	3	4	5	6	6		
R	2									
R	3									
O	4									
G	5									
A	6									
N	7									
T	8									

Расстояние редактирования: пример решения

Решение задачи — правый нижний элемент таблицы.

		S	U	R	R	O	G	A	T	E
	0	1	2	3	4	5	6	7	8	9
A	1	1	2	3	4	5	6	6	7	8
R	2	2	2	2	3	4	5	6	7	8
R	3	3	3	2	2	3	4	5	6	7
O	4	4	4	3	3	2	3	4	5	6
G	5	5	5	4	4	3	2	2	3	5
A	6	6	6	5	5	4	3	2	3	4
N	7	7	7	6	6	5	4	3	3	4
T	8	8	8	7	7	6	5	4	3	4

Расстояние редактирования

- Сложность по времени — $O(|s| \times |d|)$.
- Сложность по памяти — $O(|d|)$.

Многомерные варианты

Задача о счастливых билетах

Билет состоит из N цифр от 0 до 9 и является счастливым, если сумма первой половины его цифр равна сумме второй половины. Найти количество счастливых билетов.

Задача о счастливых билетах

- Причём здесь динамическое программирование? Это ведь комбинаторная задача.
- Сведём задачу к другой. Пусть $N = 6$.

3	3	5	6	4	1
3	3	5	3	5	8

- Заменим во второй половине числа все цифры на их дополнение до девяти.
- Количество таких чисел в точности равно количеству счастливых, так как отображение биективное.
- Исходный инвариант: $x_1 + x_2 + x_3 = x_4 + x_5 + x_6$.
- Инвариант после отображения:
  $x_1 + x_2 + x_3 + (9 - x_1) + (9 - x_2) + (9 - x_3) = 3 \cdot 9$.
- Требуется найти количество $N-$значных чисел, сумма которых равна $9 \cdot \frac{N}{2}$.

Задача о счастливых билетах

- Привычное решение задачи наталкивается на проблему: нам нужно найти не просто количество любых чисел от 0 до 9, сумма которых 27, нужно, чтобы таких чисел было именно 6.
- Количество таких чисел есть сумма количеств чисел:
  - первая цифра которых 0 и сумма пяти остальных равна 27;
  - первая цифра которых равна 1 и сумма пяти остальных равна 26;
  - ...
  - первая цифра которых равна 9 и сумма остальных пяти равна 18

Задача о счастливых билетах

- Нам удалось разбить задачу подзадачи меньшего ранга.
- Обозначим за $f(n, left)$ количество чисел, имеющих n знаков, сумма которых $left$.
- Тогда $f(6, 27) = f(5, 27) + f(5, 26) + f(5, 25) + f(5, 24) + f(5, 23) + f(5, 22) + f(5, 21) + f(5, 20) + f(5, 19) + f(5, 18)$.
- Доопределим функцию $f(n, left)$ таким образом, что при $n > 0$ и $left < 0$ она возвращала 0 и $f(1, left) = 1$, если $0 \leqslant left \leqslant 9$ и $f(1, left) = 0$ в противном случае.
- Тогда $f(n, left) = \sum_{i=0}^{9} f(n-1, left-i)$.
- Мы свели задачу к задаче динамического программирования, но двухмерной.

Задача о счастливых билетах

- Если задача двухмерная, то можно использовать двухмерную таблицу решений.
- Первый размер определяется размерностью задачи n.
- Второй размер определяется максимальным значением $left = 9 \cdot \frac{n}{2}$.
- Нерешённые подзадачи в таблице помечаются числом -1, так как ни одна из подзадач не может возвратить отрицательной число.

- Значения в таблице решений занимают последовательные ячейки, она не разрежена.
- Следовательно, задача допускает восходящее решение.
- Таблица заполняется, начиная от значения $n = 1$ и всех возможных $left$ от 0 до 9.
- Максимальный уровень рекурсии равен n, это немного и восходящее решение, хотя и возможно, но не требуется.

Задача о вторичной структуре РНК

Задача из вычислительной биологии:
Имеется последовательность *оснований*

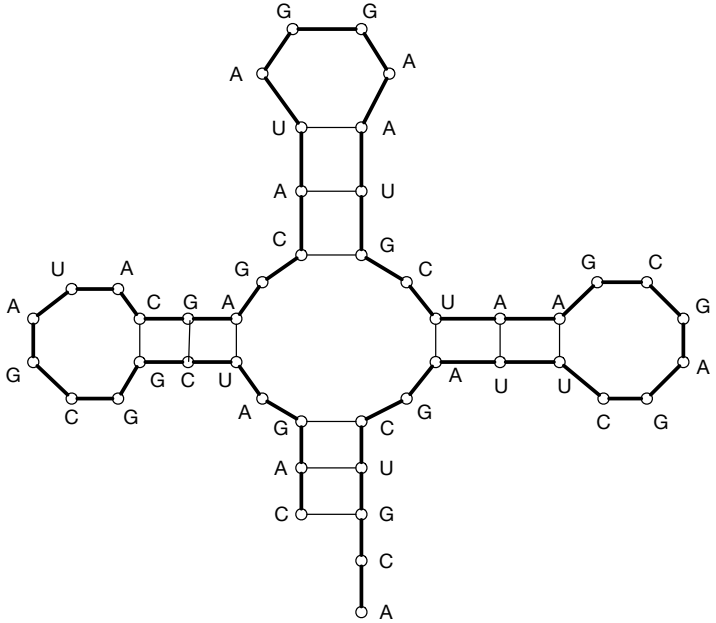
$$B = b_1 b_2 \dots b_n, b_i \in \{A, C, G, U\}.$$

Каждое основание может образовывать пару не более, чем одним другим основанием.

$$\begin{aligned} A &\leftrightarrow U \\ C &\leftrightarrow G. \end{aligned}$$

Образуется *вторичная структура*.
Сколько различных вторичных структур существует для данной последовательности?

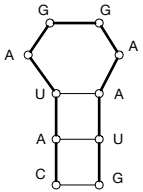
Задача о вторичной структуре РНК

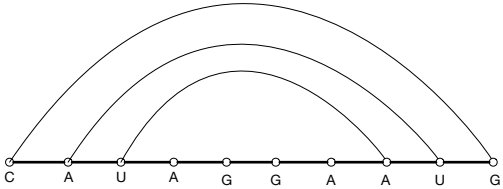


Задача о вторичной структуре РНК

Условия, накладываемые на вторичную структуру.

1. **Отсутствие резких поворотов** Не существует пар (b_i, b_j) для которых $|i - j| \leqslant 4$.
2. **Состав пар** Возможны только пары (A, U) и (C, G).
3. **Вхождение основание** Каждое основание входит не более, чем в одну пару.
4. **Отсутствие пересечений** Для двух произвольных пар (b_i, b_j) и (b_k, b_l) невозможно условие $i < k < j < l$





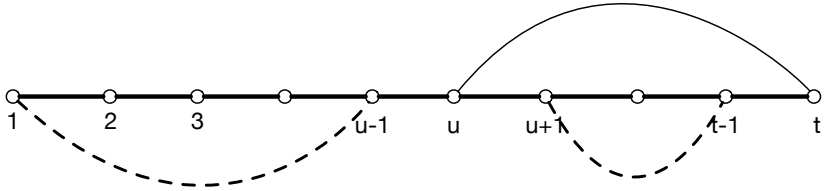
Задача о вторичной структуре РНК

- Задача ли это динамического программирования?
- Если да, то что есть «подзадача»?
- Если да, то что есть «консолидация»?

Задача о вторичной структуре РНК: попытка 1

- Пусть $F(k)$ — максимальное количество пар для вторичной структуры $b_1, b_2, \ldots, b_k$
- Тогда $F(1) = F(2) = F(3) = F(4) = F(5) = 0$
- Пусть решены задачи для $F(1), F(2), \ldots, F(t-1),$. Как решить задачу для $F(t)$?
- Возможны два варианта:
  - во вторичной структуре $b_1, b_2, \ldots, b_t$ t не участвует в паре.
  - t образует пару с каким-то элементов u, $u < t - 4$.
- Первый случай порождает подзадачу $F(t-1)$.
- А что во втором случае?

Задача о вторичной структуре РНК:попытка 1



Имеется запрет на пересечения $\rightarrow$ нет пар, левый конец которых меньше u, а правый — больше u.

Возникает две подзадачи, первую мы решить можем, а вторую — нет.

Задача о вторичной структуре РНК: вторая попытка

Необходимость решения задач второго рода подсказывает: за целую задачу взять $F(k, l)$, то есть два параметра.

$$F(k,l) = \begin{cases} 0, \text{ если } k+4 \leqslant l \\ \max(F(k,l-1), 1 + \max_u(F(k,u-1) + F(u+1,l-1)) & \text{иначе} \end{cases}$$

Восходящее решение: замечание: сначала решаются более *короткие* задачи.