Динамическое программирование. Примеры задач.

Руслан Назирович Мокаев

Математико-механический факультет,
Санкт-Петербургский государственный университет

Санкт-Петербург, 21.05.2024

Содержание лекции

- Задача о рюкзаке
- Задача о бродячем торговце

Задача о рюкзаке

Даны n предметов, каждый из них имеет положительный вес w_i и положительную ценность v_i; Дан рюкзак вместимостью (емкостью) W; Необходимо заполнить рюкзак предметами так, чтобы максимизировать их суммарную ценность и чтобы их суммарный вес не превысил W.

Задача о рюкзаке

Даны n предметов, каждый из них имеет положительный вес w_i и положительную ценность v_i; Дан рюкзак вместимостью (емкостью) W; Необходимо заполнить рюкзак предметами так, чтобы максимизировать их суммарную ценность и чтобы их суммарный вес не превысил W.

У этой задачи есть множество вариаций:

- Классической является **дискретная задача о рюкзаке** ($0-1$ рюкзак) – каждый предмет можно либо не класть, либо положить в единственном экземпляре.
- Более простым вариантом является **непрерывная** или **дробная задача о рюкзак** – предметы разрешено дробить на меньшие части и помещать в рюкзак.
- **Ограниченный рюкзак** – каждый предмет разрешено брать целое число раз, но не более, чем C раз ($C \in \mathbb{N}$).
- **Неограниченный рюкзак** – разрешено класть в рюкзак произвольное целое число экземпляров любого предмета.

$$\sum_{i=1}^{n} v_i x_i \to \max_{\Omega}$$

$$\Omega \begin{cases} \sum_{i=1}^{n} w_i x_i \le W \\ x_i \in \{0, 1\}, \ i \in \overline{1:n}, \end{cases}$$

(a) Дискретный рюкзак

$$\sum_{i=1}^{n} v_i x_i \to \max_{\Omega}$$

$$\Omega \begin{cases} \sum_{i=1}^{n} w_i x_i \le W \\ 0 \le x_i \le 1, \ i \in \overline{1:n}, \end{cases}$$

(b) Дробный рюкзак

$$\sum_{i=1}^{n} v_i x_i \to \max_{\Omega}$$

$$\Omega \begin{cases} \sum_{i=1}^{n} w_i x_i \le W \\ x_i \in \{0, 1, \ldots, C\}, \ i \in \overline{1:n}, \end{cases}$$

(c) Ограниченный рюкзак

$$\sum_{i=1}^{n} v_i x_i \to \max_{\Omega}$$

$$\Omega \begin{cases} \sum_{i=1}^{n} w_i x_i \le W \\ x_i \in \mathbb{N}_0, \ i \in \overline{1:n}, \end{cases}$$

(d) Неограниченный рюкзак

Замечание: задача дробного рюкзака формулировкой похожа на классический дискретный рюкзак, однако решается заметно проще (дробный рюкзак решается за время $O(n \log n)$ из-за необходимости сортировать, в то время как дробный рюкзак – NP-трудная задача, хотя динамический алгоритм и работает за $O(nW)$).

Замечание: задача дробного рюкзака формулировкой похожа на классический дискретный рюкзак, однако решается заметно проще (дробный рюкзак решается за время $O(n \log n)$ из-за необходимости сортировать, в то время как дробный рюкзак – NP-трудная задача, хотя динамический алгоритм и работает за $O(nW)$).

В случае дробного рюкзака нетрудно придумать жадный алгоритм, позволяющий получить оптимальное решение. Жадный подход в случае дискретного рюкзака не сработает: рассмотрим рюкзак вместимостью $W = 50$ и три предмета с весами и ценностями $(10, 60), (20, 100), (30, 120)$.

- в случае дробного жадная стратегия (отсортировать веса в порядке убывания значения ценности одного кг предмета и заполнять самыми ценными килограммами, пока не будет достигнут предельный вес) даст верный результат $60 + 100 + \frac{30}{20} \cdot 120 = 240$;
- в случае дискретного рюкзака жадная стратегия приведет к ответу $60 + 100 = 160$ – берем самые ценный по соотношению ценность/вес первый предмет, потом второй. На третий предмет места нет. Оптимальней же будет взять второй и третий предмет с суммарной ценностью 220.

Описание структуры оптимального решения

Утверждение: В дискретной задаче о рюкзаке имеется оптимальное решение $I \subseteq \overline{1:n}, n \geq 1$ с суммарной ценностью $V = \sum_{i \in I} w_i$. Тогда I равно

1. либо о.р. задачи с вместимостью W, но в которой доступны только первые $n-1$ предметы,
2. либо о.р. задачи с вместимостью $W - w_n$, в которой доступны только первые $n-1$ предметы, объединенному с n-м предметом.

Описание структуры оптимального решения

Утверждение: В дискретной задаче о рюкзаке имеется оптимальное решение $I \subseteq \overline{1:n}, n \geq 1$ с суммарной ценностью $V = \sum_{i \in I} w_i$. Тогда I равно

1. либо о.р. задачи с вместимостью W, но в которой доступны только первые $n-1$ предметы,
2. либо о.р. задачи с вместимостью $W - w_n$, в которой доступны только первые $n-1$ предметы, объединенному с n-м предметом.

Док-во: Возможны два случая:

- если $n \notin I$, то I будет о.р. задачи, в которой доступны только первые $n-1$ предметы. Это решение допустимо (т.к. $n \notin I$) и, если оно не оптимально ($\exists\ I'$ с суммарной стоимостью $V' > V$), то I' будет допустимым решением исходной задачи, причем $V' > V$ (?!).
- если $n \in S$, то $w_n \leq W \Rightarrow$ в рюкзаке емкостью W уже зарезервирован вес w_n под n-й предмет. Осталась емкость $W - w_n$. Мн-во предметов $I \setminus \{n\}$ – допустимое решением подзадачи с емкостью $W - w_n$ с суммарной ценностью $V - v_n$. Пусть существует о.р. $I' \subseteq \overline{1:(n-1)}$ подзадачи с суммарной стоимостью $V' > V - v_n$ и суммарным весом $\leq W - w_n \Rightarrow$ решение $I' \cup \{n\}$ будет иметь суммарный вес $\leq W$ и ценность $V' + v_n > (V - v_n) + v_n = V$ (?!).

Рекурсивное определение оптимального решения

Введем следующие обозначения: пусть $S_{i,B}$ – о.р. дискретной задачи о рюкзаке, в которой доступны только первые $i \in \overline{0:n}$ предметов, а емкость равна B, где $B \in \overline{0:W}$, а суммарную ценность о.р. будем хранить в клетке $c[i, B]$ специальной матрицы c размера $(n+1) \times (W+1)$.

Рекурсивное определение оптимального решения

Введем следующие обозначения: пусть $S_{i,B}$ – о.р. дискретной задачи о рюкзаке, в которой доступны только первые $i \in \overline{0:n}$ предметов, а емкость равна B, где $B \in \overline{0:W}$, а суммарную ценность о.р. будем хранить в клетке $c[i, B]$ специальной матрицы c размера $(n+1) \times (W+1)$.

Заметим, что

- если $B < w_i$, то $i \notin S_{i,B}$ и мы находимся в первом случае: $S_{i,B} = S_{i-1,B}$ и $c[i, B] = c[i-1, B]$.
- если $B \geq w_i$, то ситуация немного сложнее; Мы не можем наперед знать $i \in S_{i,B}$ или $i \notin S_{i,B}$, поэтому для определения оптимального решения необходимо выбрать максимум из $c[i-1, B]$ (первый случай) и $c[i-1, B-w_i] + v_i$ (второй случай).

Считаем, что если необходимо уместить в рюкзаке предметы из пустого множества, то о.р. равно нулю (нулевая ценность). Следовательно, рекурсивное соотношение выглядит следующим образом:

$$c[i, B] = \begin{cases} c[i-1, B], \text{ если } B < w_k, \\ \max\left(c[i-1, B], c[i-1, B-w_i] + v_i\right), \text{ если } B \geq w_k, \end{cases}$$

$$c[0, B] = 0, B \in [0, W]$$

Вычисления значения методом восходящего анализа

Алгоритм на основе рекурсивного определения о.р.:

1. инициализируем двумерный массив (или список списков) $c[(n+1) \times (W+1)]$ нулями (индексация идет с нуля);
2. далее пробегаемся по строкам (от 0 до n) и для каждого элемента B строки i применяем рекурсивное соотношение;
3. после заполнения всей таблицы выводим значение $c[n, W]$ в качестве ответа (это как раз будет оптимальное заполнения рюкзаке емкостью W с выбором всех доступных изначально предметов).

Вычисления значения методом восходящего анализа

Алгоритм на основе рекурсивного определения о.р.:

1. инициализируем двумерный массив (или список списков) $c[(n+1) \times (W+1)]$ нулями (индексация идет с нуля);
2. далее пробегаемся по строкам (от 0 до n) и для каждого элемента B строки i применяем рекурсивное соотношение;
3. после заполнения всей таблицы выводим значение $c[n, W]$ в качестве ответа (это как раз будет оптимальное заполнения рюкзаке емкостью W с выбором всех доступных изначально предметов).

Данный алгоритм имеет временную сложность $O(nW)$, т.к. итераций в цикле ровно $(n+1) \times (W+1)$, на каждой из них значение не вычисляется, а за константное время $O(1)$ берется из таблицы.

Составление оптимального решения

Алгоритм восстановления о.р. по таблице c похож на вытаскивание предметов под одному (в порядке убывания весов) из рюкзака:

1. необходимо инициализировать пустой список I, в который будем складывать номера предметов из рюкзака и на каждом шаге хранить остаточную емкость R, изначально равную W;
2. итерироваться по всем предметам в порядке убывания веса и если для i-го предмета справедливо
   - его вес меньше остаточной емкости: $w_i \leq R$ и
   - оптимальное решение задачи $S_{i-1,R}$ не превышает $S_{i-1,R-w_i} + v_i$ (то есть при построении таблице в момент заполнения $c[i, R]$ выбрали второй случай с i-м предметом),

   то значит i-й предмет входит в о.р. и добавляем его в список I, а остаточную емкость уменьшаем на w_i. Если одно из двух утверждений для i-го предмета не выполняется, то переходим к $i - 1$.
3. продолжаем пока не просмотрим все предметы. Оптимальное решение будет храниться в списке I.

Составление оптимального решения

Алгоритм восстановления о.р. по таблице c похож на вытаскивание предметов под одному (в порядке убывания весов) из рюкзака:

1. необходимо инициализировать пустой список I, в который будем складывать номера предметов из рюкзака и на каждом шаге хранить остаточную емкость R, изначально равную W;
2. итерироваться по всем предметам в порядке убывания веса и если для i-го предмета справедливо
   - его вес меньше остаточной емкости: $w_i \leq R$ и
   - оптимальное решение задачи $S_{i-1,R}$ не превышает $S_{i-1,R-w_i} + v_i$ (то есть при построении таблице в момент заполнения $c[i, R]$ выбрали второй случай с i-м предметом),

   то значит i-й предмет входит в о.р. и добавляем его в список I, а остаточную емкость уменьшаем на w_i. Если одно из двух утверждений для i-го предмета не выполняется, то переходим к $i - 1$.
3. продолжаем пока не просмотрим все предметы. Оптимальное решение будет храниться в списке I.

Оптимальное решение составляется за время $O(n)$ (по $O(1)$ на каждую итерацию), что не влияет на асимптотику алгоритма вычисления оптимального значения $O(nW)$.

Емкость $W = 8$ и предметы $\{(1, 60), (4, 120), (3, 150), (6, 320)\}$.

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	60	60	60	60	60	60	60	60
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	60	60	60	60	60	60	60	60
3	0	60	60	150	210	210	210	210	210
4	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	60	60	60	60	60	60	60	60
3	0	60	60	150	210	210	210	210	210
4	0	60	60	150	210	210	210	270	330
6	0	0	0	0	0	0	0	0	0

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	60	60	60	60	60	60	60	60
3	0	60	60	150	210	210	210	210	210
4	0	60	60	150	210	210	210	270	330
6	0	60	60	150	210	210	320	380	380

Задача о бродячем торговце

В стране есть N городов, любые два соединены дорогой. Поездка по каждой дороге стоит определенную сумму. Необходимо объехать все N городов ровно по одному разу и потратить минимальное количество средств. С точки зрения теории графов задача о коммивояжере состоит в поиске во взвешенном полном неорграфе гамильтонова цикла наименьшего веса.

Задача о бродячем торговце

В стране есть N городов, любые два соединены дорогой. Поездка по каждой дороге стоит определенную сумму. Необходимо объехать все N городов ровно по одному разу и потратить минимальное количество средств. С точки зрения теории графов задача о коммивояжере состоит в поиске во взвешенном полном неорграфе гамильтонова цикла наименьшего веса.

Оптимальное решение может быть рассмотрено как путь, начинающийся в произвольной вершине v_0 (должны обойти все вершины, поэтому не важно какую считать первой) и в ней же заканчивающийся.

Задача о бродячем торговце

В стране есть N городов, любые два соединены дорогой. Поездка по каждой дороге стоит определенную сумму. Необходимо объехать все N городов ровно по одному разу и потратить минимальное количество средств. С точки зрения теории графов задача о коммивояжере состоит в поиске во взвешенном полном неорграфе гамильтонова цикла наименьшего веса.

Оптимальное решение может быть рассмотрено как путь, начинающийся в произвольной вершине v_0 (должны обойти все вершины, поэтому не важно какую считать первой) и в ней же заканчивающийся.

Обозначим за $D(S, v)$ решение подзадачи поиска кратчайшего пути, начинающегося в v_0, проходящего через все вершины из множества S и заканчивающийся в v.

Задача о бродячем торговце

В стране есть N городов, любые два соединены дорогой. Поездка по каждой дороге стоит определенную сумму. Необходимо объехать все N городов ровно по одному разу и потратить минимальное количество средств. С точки зрения теории графов задача о коммивояжере состоит в поиске во взвешенном полном неорграфе гамильтонова цикла наименьшего веса.

Оптимальное решение может быть рассмотрено как путь, начинающийся в произвольной вершине v_0 (должны обойти все вершины, поэтому не важно какую считать первой) и в ней же заканчивающийся.

Обозначим за $D(S, v)$ решение подзадачи поиска кратчайшего пути, начинающегося в v_0, проходящего через все вершины из множества S и заканчивающийся в v.

Тогда кратчайший путь будет состоять из двух частей:

1. последнее ребро из некоторой вершины $w \in S$ в конечную вершину v;
2. кратчайший (если он не кратчайший, то легко дойти до противоречия с определением $D(S, v)$) путь из v_0 в w, проходящего через все вершины в $S \setminus \{v\}$.

Структура оптимального решения: наперед знать какая вершина будет предпоследней в оптимальном пути из v_0 в v не получится, поэтому надо будет посмотреть все ребра w, v при $w \in S \setminus \{v\}$ и для каждой из них решить соответствующую подзадачу поиска кратчайшего пути. Таким образом, мы показали наличие оптимальной подструктуры!

Структура оптимального решения: наперед знать какая вершина будет предпоследней в оптимальном пути из v_0 в v не получится, поэтому надо будет посмотреть все ребра w, v при $w \in S \setminus \{v\}$ и для каждой из них решить соответствующую подзадачу поиска кратчайшего пути. Таким образом, мы показали наличие оптимальной подструктуры!

Рекурсивное определение оптимального решения: пусть стоимости проезда по дорогам (веса в графе) заданы весовой функцией $c : E \to R$. На основе предыдущего анализа рекурсивное соотношение будет выглядеть следующим образом:

$$D(S, v) = \begin{cases} c(v_0, v), & S = \{v\} \\ +\infty, & v \notin S \\ \min\limits_{w \in S \setminus \{v\}} (D(S \setminus \{v\}, w) + c(w, v)), & \text{иначе} \end{cases}$$

Вычисление решения методом восходящего анализа:

1. Задаем произвольную стартовую вершину v_0; инициализируем матрицу D размера $2^n \times n$ значениями $+\infty$; для восстановления оптимального обхода введем матрицу P размера $2^n \times n$, где в $P[S][v]$ хранить последние ребра в оптимальных путях до v с обходом всем вершин в S;
2. Для всех вершин $v \in V$ записываем в $D[\{v\}][v]$ значение $c(v_0, v)$;
3. Далее итерируемся по всем возможным (от 2 до n) размерам i подмножеств вершин S, смотрим на каждое подмножество S: $|S| = i$ и на каждую возможно вершину $v \in S$. Для это вершины v находим такую $w \in S \setminus \{v\}$, что достигается минимум из соотношения **??**. В $P[S][v]$ записывается ребро (u, v).
4. После всех вычислений (заполнены матрицы D и P) составим оптимальный обход: необходимо начать с ребра (u, v), находящегося в $P[V][v_0]$ и далее в цикле обнаруживаем предшествующую вершину j в цикле и находим ребро из оптимального обхода, ведущее в нее, и переходим к рассмотрению вершины на другом конце (на забывая выкидывать уже просмотренные вершины). Для получения стоимость поездки необходимо просто складывать веса всех ребер при обходе.

Пример задачи о бродячем торговце

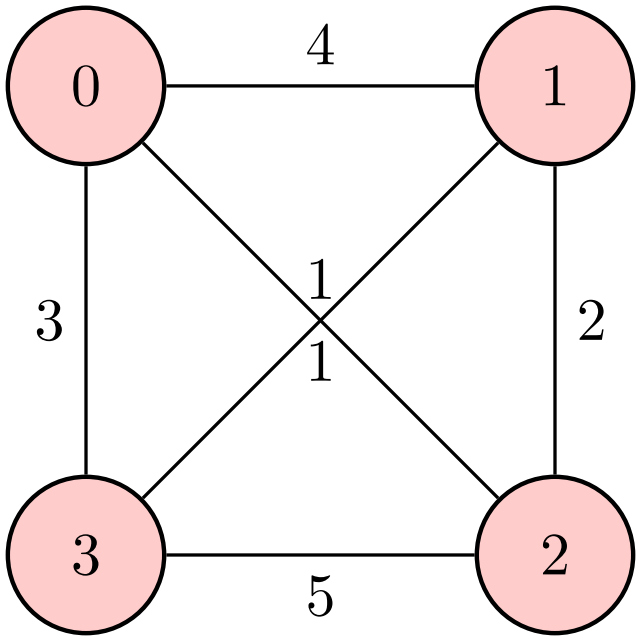

Рис.: Пример задачи о коммивояжере. Стартовая вершина 0.

Пример задачи о бродячем торговце

	0	1	2	3
()	$+\infty$	$+\infty$	$+\infty$	$+\infty$
0	$+\infty$	$+\infty$	$+\infty$	$+\infty$
1	$+\infty$	4	$+\infty$	$+\infty$
2	$+\infty$	$+\infty$	1	$+\infty$
3	$+\infty$	$+\infty$	$+\infty$	3
01	8	$+\infty$	$+\infty$	$+\infty$
02	2	$+\infty$	$+\infty$	$+\infty$
03	6	$+\infty$	$+\infty$	$+\infty$
12	$+\infty$	3	6	$+\infty$
13	$+\infty$	4	$+\infty$	5
23	$+\infty$	$+\infty$	8	6
012	7	6	9	$+\infty$
013	8	10	$+\infty$	11
023	9	$+\infty$	7	5
123	$+\infty$	7	6	4
0123	7	6	9	7

Рис.: Матрица D.

	0	1	2	3
()	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
0	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
1	$\varnothing$	$(0,1)$	$\varnothing$	$\varnothing$
2	$\varnothing$	$\varnothing$	$(0,2)$	$\varnothing$
3	$\varnothing$	$\varnothing$	$\varnothing$	$(0,3)$
01	$(1,0)$	$\varnothing$	$\varnothing$	$\varnothing$
02	$(2,0)$	$\varnothing$	$\varnothing$	$\varnothing$
03	$(3,0)$	$\varnothing$	$\varnothing$	$\varnothing$
12	$\varnothing$	$(2,1)$	$(1,2)$	$\varnothing$
13	$\varnothing$	$(3,1)$	$\varnothing$	$(1,3)$
23	$\varnothing$	$\varnothing$	$(3,2)$	$(2,3)$
012	$(1,0)$	$(0,1)$	$(0,2)$	$\varnothing$
013	$(3,0)$	$(0,1)$	$\varnothing$	$(0,3)$
023	$(3,0)$	$\varnothing$	$(0,2)$	$(0,3)$
123	$\varnothing$	$(3,1)$	$(1,2)$	$(1,3)$
0123	$(3,0)$	$(3,1)$	$(0,2)$	$(1,3)$

Рис.: Матрица P.