

Задача о рюкзаке

Шевкунов К.С. 594

14 ноября 2017 г.

1 Постановка задачи

1.1 Формулировка условия

Имеется набор из n предметов. У каждого предмета есть положительный вес w и стоимость c . Также дано неотрицательное число W — вместимость рюкзака. Требуется найти такое подмножество предметов M , чтобы оно помещалось в рюкзак, и суммарная стоимость предметов была максимальна. То есть:

$$\sum_{x \in M} w(x) \leq W, \sum_{x \in M} c(x) \rightarrow \max$$

1.2 Цель

Постройте полиномиальную схему приближения для данной задачи. То есть необходимо придумать и реализовать алгоритм, который получает на вход экземпляр задачи о рюкзаке, а также произвольное (рациональное) $\varepsilon > 0$, и находит $(1 + \varepsilon)$ - приближенное решение. Алгоритм должен работать за полиномиальное время относительно размера исходной задачи и $\frac{1}{\varepsilon}$.

1.3 Доказательство NP-трудности

Определим задачу:

$$SUBSET - SUM = \{(n_1, \dots, n_k, N) \mid \exists \alpha \in \{0, 1\}^k : \sum_{i=1}^k \alpha_i n_i = N\}$$

В книге (Д.В. Мусатов, "Сложность вычислений. Конспект лекций") дока Конспект лекций) доказано, что задача $SUBSET - SUM$ является NP-полной. Сведём эту задачу полиномиальной к нашей и этим докажем, что она является NP-трудной.

Обозначим предметы натуральными числами $1..N$. Определим $\forall i \in \{1..N\} : c(i) := w(i) := n_i, W := N$ Тогда исходная задача свелась к поиску M тако-го, что:

$$\sum_{x \in M} n_i \leq N, \sum_{x \in M} n_i \rightarrow \max$$

Или, иначе говоря к задаче поиска $\alpha \in \{0, 1\}^k$, такого, что:

$$\sum_{i=1}^k \alpha_i n_i \leq N, \sum_{i=1}^k \alpha_i n_i \rightarrow \max \leq N$$

Ясно, что искомый максимум не больше N , и, если равняется N , то $(n_1, \dots, n_k, N) \in SUBSET - SUM$. Иначе же оптимального подмножества не существует и $(n_1, \dots, n_k, N) \notin SUBSET - SUM$.

Таким образом, с помощью полиномиального сведения мы научились решать задачу $SUBSET - SUM$ с помощью нашей задачи, т.е. с некоторым полиномиальным сведением мы можем решать любую задачу из класса NP с помощью нашей.

Таким образом, мы обосновали уместность рассмотрения приближённых решений для данной задачи. Далее рассмотрим так называемое псевдополиномиальное решение.

2 Псевдополиномиальное решение $O(nW)$

Определение 1 Пусть в постановку задачи входит числовой параметр n (не количественный) и алгоритм работает полиномиальное время от самого n . Тогда такой алгоритм называется псевдополиномиальным.

В частности, для текущей задачи не известно полиномиального решения (т.е. такого, которое работает за полиномиальное от размера входа время, в частности, для числового параметра N размером входа будет $\log N$), но известно псевдополиномиальное решение.

2.1 Алгоритм

Пусть исходные объекты задачи пронумерованы числами $1..n$, W - ограничение на размер рюкзака. Применим метод динамического программирования:

knapsack_pseudopolynomial

1. Инициализируем матрицу A размера (n, W) нулями.
2. for i in $1 .. n$:
 for s in $1 .. W$:
 $A[i, s] := A[i - 1, s]$
 if $(w(i) \leq s)$ and $(A[i, s] > A[i - 1, s - w(i)] + c(i))$:
 $A[i, s] := A[i - 1, s - w(i)] + c(i)$
3. вернуть $A[n, W]$, если нужна стоимость и/или восстановить подмножество за $O(nW)$ по таблице A из ячейки $A[n, W]$ (TODO: написать, как)

2.2 Доказательство

Докажем по индукции, что $A[i, s]$ - максимальная стоимость предметов, которые поместятся в рюкзак размера s , если использовать только предметы с номерами $1..i$, назовём это подзадачей $[i, s]$ нашей задачи.

База индукции: $\forall x \in N : A[0, x] = A[x, 0] = 0$. Очевидно, она выполнена, в рюкзак нулевого размера ничего нельзя положить (в условии все веса и стоимости строго положительны). С другой стороны, если предметов нет, то положить в рюкзак нечего.

Переход: пусть $\forall 0 < j \leq i : \forall 0 < p \leq s : ((j, p) \neq (i, s)) \rightarrow A[j, p]$ вычислено корректно. Докажем, что $A[i, s]$ вычисленное из подсчитанных значений как алгоритме корректно.

Заметим, что оптимальный набор не обязательно единственен, например, для рюкзака размера 2 и предметами с массой и стоимостью 1, 1 и 2 можно положить как два предмета веса 1, так и один веса 2.

Рассмотрим множество оптимальных наборов для подзадачи $[i, s]$. Выполнен хотя бы один из двух вариантов:

- Существует оптимальный набор, в котором лежит предмет с номером i . Тогда $A[i, s] = A[i - 1, s - w(i)] + c(i)$ и $A[i, s] \geq A[i - 1, s]$. Первое утверждение прямо следует из того, что оптимальность набора гарантирует оптимальность поднабора. Иначе поднабор можно улучшить и получить противоречие с оптимальностью набора. Отрицание второго условия означает, что существует набор без предмета с номером i лучше нашего и приводит к противоречию.
- Существует оптимальный набор, в котором не лежит предмет с номером i . Аналогично получаем, что $A[i, s] \geq A[i - 1, s - w(i)] + c(i)$ (если $s \geq w(i)$) и $A[i, s] = A[i - 1, s]$.

Таким образом, наш алгоритм, который по-сути выбирает максимум из двух величин $A[i, s] = \max(A[i - 1, s], A[i - 1, s - w(i)] + c(i))$ (если обе существуют - иначе возможен только один вариант $A[i, s] = A[i - 1, s]$), вычисляет оптимальное значение в обоих случаях.

Итого, корректность $A[i, s]$ доказана по индукции для всех $0 \leq i \leq n$, $0 \leq s \leq W$, в том числе и для $A[n, W]$. Решение, очевидно, работает за $O(nW)$.

3 Псевдополиномиальное решение $O(nC)$

3.1 Алгоритм

knapsack_pseudopolynomial_two

$$1. C = \sum_{i=1}^n c(i)$$

2. Инициализируем матрицу A размера (n, C) символом $+\infty$, который будем считать большим любого числа.
3. $A[0, 0] := 0$
4. for i in $1..n$:
 for s in $1..C$:
 $A[i, s] = A[i - 1, s]$
 if $(s \geq c(i))$ and $(A[i - 1, s - c(i)] + w(i) \leq \min(W, A[i, s]))$:
 $A[i, s] := A[i - 1, s - c(i)] + w(i)$
5. $k := \operatorname{argmax}_k (A[n, k] < +\infty)$
6. вернуть k , если нужна стоимость и/или восстановить подмножество за $O(nW)$ по таблице A из ячейки $A[n, k]$ (TODO: написать, как)

3.2 Доказательство

Докажем по индукции, что $A[i, s]$ - минимальный вес подмножества предметов, которые можно поместить в рюкзак или $+\infty$, если такого нет, что использованы только предметы с номерами $1..i$ и их суммарная стоимость равна s .

База индукции: $A[0, 0] = 0$ - пустое множество предметов оптимально.
 $\forall x > 0 : A[0, x] = +\infty$ - из пустого множества предметов нельзя выбрать подмножество с положительной стоимостью.

Переход: пусть $\forall 0 < j \leq i : \forall 0 < p \leq s : ((j, p) \neq (i, s)) \rightarrow A[j, p]$ вычислено корректно. Докажем, что $A[i, s]$ вычисленное из подсчитанных значений как алгоритме корректно.

Возможны два случая (хотя бы один из них выполнен, возможно, оба):

- Пусть существует оптимальный (в смысле $A[i, s]$ как минимума) набор предметов, содержащий предмет с номером i . Тогда $A[i, s] = A[i, s - c(i)] + w(i)$, т.к. из оптимальности набора следует оптимальность поднабора и $A[i, s] \leq A[i - 1, s]$, иначе получим противоречие с оптимальностью.
- Пусть существует оптимальный набор предметов, не содержащий предмет с номером i . Тогда аналогично: $A[i, s] = A[i - 1, s]$ и $A[i, s] \leq A[i - 1, s - c(i)] + w(i)$.

Таким образом, наш алгоритм, который по-сути выбирает минимум из двух величин $A[i, s] = \min(A[i - 1, s], A[i - 1, s - c(i)] + w(i))$ (если обе допустимы, т.е. $A[i - 1, s - c(i)] + w(i) \leq C$, иначе $A[i, s] = A[i - 1, s]$), вычисляет оптимальное значение в обоих случаях.

Итого, корректность $A[i, s]$ доказана по индукции для всех $0 \leq i \leq n$, $0 \leq s \leq W$. По значениям $A[i, s]$, перебирая оптимальную суммарную стоимость $s \in 0..C$ можно восстановить ответ в задаче, достаточно просто найти

$A[n, s] \neq +\infty$ с наибольшим значением s . Значения в таблице A вычисляются за $O(nC)$, поиск наибольшего значения работает за $O(C)$, восстановление оптимального набора за $O(nC)$. Итого, решение работает за $O(nC)$.

4 Полиномиальное приближение

4.1 Алгоритм

knapsack_polynomial_estimation:

1. $P := \max(c(x) | w(x) \leq W)$ (вообще говоря, предметы, которые не помещаются в пустой рюкзак можно выкинуть на этапе формулировки задачи)
2. $K := \frac{\varepsilon P}{k}$
3. $c_{zipped}(x) := \lfloor \frac{c(x)}{K} \rfloor$
4. Найти оптимальное подмножество M^* для задачи со стоимостями $c_{zipped}(x)$ псевдополиномиальным алгоритмом за $O(nC)$ и вернуть его.

4.2 Доказательство

Пусть M - произвольное множество предметов.

$$\text{Обозначим } cost(M) := \sum_{x \in M} c(x), \text{ } cost_{zipped}(M) := \sum_{x \in M} c_{zipped}(x)$$

Пусть M^* - множество, которое было найдено псевдополиномиальным алгоритмом, а оптимальное множество предметов в задаче O .

Тогда $K cost_{zipped}(M^*) \geq cost(O) - nK$. Действительно, если бы мы не округляли веса до целых и нашли оптимальное решение, ты получили бы $cost_{zipped} \equiv \frac{cost}{K}$. Однако, из-за округления вниз имеем $\forall i : c_{zipped}(i) \leq \frac{c(i)}{K} - 1$, в частности $cost_{zipped}(M^*) \geq cost_{zipped}(O) \geq \frac{cost(O)}{K} - n$ (второе неравенство получается суммированием неравенств для каждого предмета набора O , а первое следует из оптимальности M^*). Умножая неравенство на K , получим требуемое неравенство.

Заметим, что, если в оптимальном наборе O есть хотя бы один предмет, то $cost(O) \geq P = \max(c(x) | w(x) \leq W)$. Заметим, что это верно и если оптимальный набор пустой.

$$\text{Тогда } cost(M^*) \geq K cost_{zipped}(M^*) \geq cost(O) - nK = cost(O) - \varepsilon P \geq cost(O) - \varepsilon cost(O) = (1 - \varepsilon) cost(O)$$

Полученное утверждение в точности есть доказываемое.

4.3 Работа на реальных данных