У нас есть какое-то количество предметов, каждый из этих предметов имеет стоимость и вес, а также рюкзак с определенной вместительностью, наша задача из всего множества предметов выбрать такое подмножество которое во-первых будет помещаться в рюкзак, а во вторых суммарная стоимость этих предметов будет максимальной.

Рассматриваемая нами задача является NP – полной, то есть для нее не существует полиномиального алгоритма , решающего её за разумное время, в этом и есть проблема. Либо мы выбираем быстрый алгоритм, но он как известно не всегда решает задачу наилучшим образом, либо выбираем точный, который опять же не является работоспособным для больших значений.

**NP-полная задача** — в [теории алгоритмов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%BE%D0%B2) [задача с ответом «да» или «нет»](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B5%D1%88%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8) из [класса NP](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_NP), к которой можно свести любую другую задачу из этого класса за [полиноминальное время](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BD%D0%BE%D0%BC%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%B2%D1%80%D0%B5%D0%BC%D1%8F) (то есть при помощи операций, число которых не превышает некоторого полинома в зависимости от размера исходных данных).

Простые вычислительные задачи могут быть решены за полиномиальное время (класс P). Это значит, что количество итераций или время поиска решения, полиномиально зависит от числа наблюдений исходных данных.

Трудоёмкость более сложных задач экспоненциально растёт с увеличением объема данных. Алгоритмы с экспоненциальным временем считаются неэффективными.

## Примеры NP-полных задач[[править](https://ru.wikipedia.org/w/index.php?title=NP-%D0%BF%D0%BE%D0%BB%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B0&veaction=edit&section=4" \o "Редактировать раздел \«Примеры NP-полных задач\») | [править код](https://ru.wikipedia.org/w/index.php?title=NP-%D0%BF%D0%BE%D0%BB%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B0&action=edit&section=4)]

* [Задача о выполнимости булевых формул](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8_%D0%B1%D1%83%D0%BB%D0%B5%D0%B2%D1%8B%D1%85_%D1%84%D0%BE%D1%80%D0%BC%D1%83%D0%BB)
* [Кратчайшее решение «пятнашек» размера {\displaystyle n\times n}](https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%B0_%D0%B2_15)
* [Задача коммивояжёра](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BC%D0%B8%D0%B2%D0%BE%D1%8F%D0%B6%D1%91%D1%80%D0%B0)
* [Проблема Штейнера](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D0%B0_%D0%A8%D1%82%D0%B5%D0%B9%D0%BD%D0%B5%D1%80%D0%B0)
* [Проблема раскраски графа](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BA%D1%80%D0%B0%D1%81%D0%BA%D0%B0_%D0%B3%D1%80%D0%B0%D1%84%D0%B0)
* [Задача о вершинном покрытии](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D0%B2%D0%B5%D1%80%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%BC_%D0%BF%D0%BE%D0%BA%D1%80%D1%8B%D1%82%D0%B8%D0%B8)
* [Задача о покрытии множества](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D0%BF%D0%BE%D0%BA%D1%80%D1%8B%D1%82%D0%B8%D0%B8_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B0)
* [Задача о клике](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D0%BA%D0%BB%D0%B8%D0%BA%D0%B5)
* [Задача о независимом множестве](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D0%BD%D0%B5%D0%B7%D0%B0%D0%B2%D0%B8%D1%81%D0%B8%D0%BC%D0%BE%D0%BC_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5)
* [Сапер (игра)](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%BF%D0%B5%D1%80_(%D0%B8%D0%B3%D1%80%D0%B0))
* [Тетрис](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%82%D1%80%D0%B8%D1%81)[[2]](https://ru.wikipedia.org/wiki/NP-%D0%BF%D0%BE%D0%BB%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B0#cite_note-2)

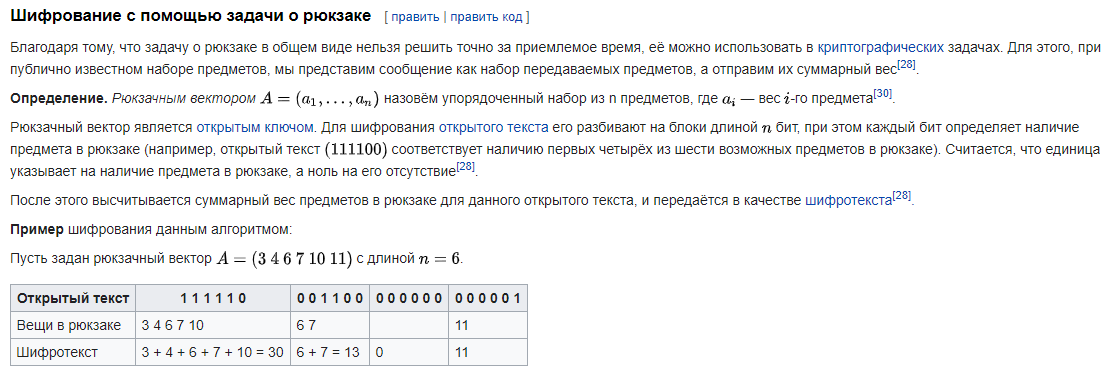
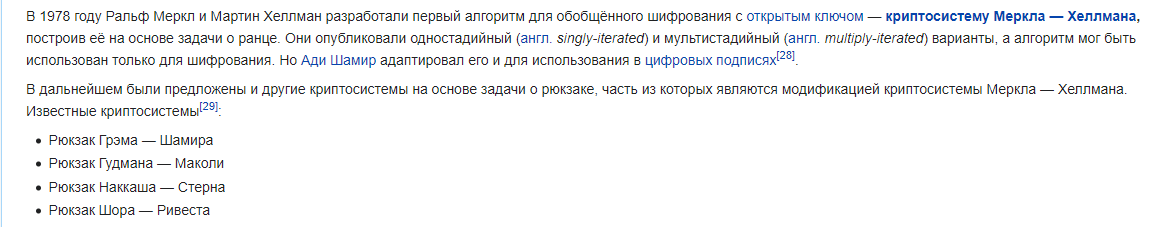
Проще говоря оптимальное решение на i шаге находится исходя из найденных ранее оптимальных решений на предшествующих шагах. Из этого следует, что для того чтобы найти оптимальное решение на последнем шаге надо сначала найти оптимальное решения для первого, затем для второго и так далее пока не пройдем все шаги до последнего.

Словосочетание «динамическое программирование» впервые было использовано в [1940](https://ru.wikipedia.org/wiki/1940-%D0%B5_%D0%B3%D0%BE%D0%B4%D1%8B)-х годах [Ричардом Беллманом](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD,_%D0%A0%D0%B8%D1%87%D0%B0%D1%80%D0%B4) для описания процесса нахождения решения задачи, где ответ на одну задачу может быть получен только после решения задачи, «предшествующей» ей. В [1953 году](https://ru.wikipedia.org/wiki/1953_%D0%B3%D0%BE%D0%B4) он уточнил это определение до современного.

Суть метода динамического программирования – на каждом шаге по весу 1<Wi <W находим максимальную загрузку Value[Wi , i], для веса Wi . Допустим мы уже нашли Value[1..W, 1..i-1], то есть для веса меньше либо равного W и с предметами, взятыми из 1..N-1. Рассмотрим предмет N, если его вес WN меньше W проверим стоит ли его брать.

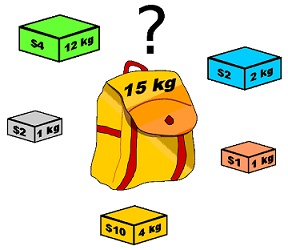
В 1978 году Ральф Меркл и Мартин Хеллман разработали первый алгоритм для обобщённого шифрования с [открытым ключом](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%BC_%D0%BA%D0%BB%D1%8E%D1%87%D0%BE%D0%BC) — [**криптосистему Меркла — Хеллмана**](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%BD%D1%86%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%9C%D0%B5%D1%80%D0%BA%D0%BB%D0%B0_%E2%80%94_%D0%A5%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD%D0%B0)**,** построив её на основе задачи о ранце. Они опубликовали одностадийный ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *singly-iterated*) и мультистадийный ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *multiply-iterated*) варианты, а алгоритм мог быть использован только для шифрования. Но [Ади Шамир](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%BC%D0%B8%D1%80,_%D0%90%D0%B4%D0%B8" \o "Шамир, Ади) адаптировал его и для использования в [цифровых подписях](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%86%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%8C)[[28]](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D1%80%D1%8E%D0%BA%D0%B7%D0%B0%D0%BA%D0%B5#cite_note-_742311dd2e8e0fe1-28).

В дальнейшем были предложены и другие криптосистемы на основе задачи о рюкзаке, часть из которых являются модификацией криптосистемы Меркла — Хеллмана. Известные криптосистемы[[29]](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D1%80%D1%8E%D0%BA%D0%B7%D0%B0%D0%BA%D0%B5#cite_note-Kin-29):

* Рюкзак Грэма — Шамира
* Рюкзак Гудмана — Маколи
* Рюкзак Наккаша — Стерна
* Рюкзак Шора — Риве
* **Разбиение**. Эта задача с очень простым условием. Дан набор чисел, нужно ответить на вопрос: можно ли разбить его на два подмножества с одинаковой суммой. Несмотря на простоту условия, для ее решения существуют только лишь псевдополиномиальный алгоритм. То есть задача решается за число шагов, полиномиально зависящее от числового параметра, но не от длины его записи.

Напоследок мы рассмотрим один из способов доказательства, что задача является NP-полной. Про многие задачи известно, что они достаточно трудны и что до сих пор нет алгоритма, который позволяет их решать за полиномиальное время. Однако вам может встретиться задача, условие которой отличается от известных NP-полных задач, но тем не менее она столь же трудна.

**Метод сужения** позволяет использовать известные NP-полные задачи для доказательства NP-полноты неизвестной. Для этого нужно доказать, что ваша задача принадлежит классу NP и сузить ее условие до известной NP-полной.

Разберем метод сужения на примере **задачи о рюкзаке**. Суть задачи в следующем: дано множество предметов с заданным весом и стоимостью. Нужно выбрать подмножество вещей с максимальной стоимостью так, чтобы они поместились в ранце с ограничением на вес. Давно известно, что эта задача NP-полная, но мы докажем это испольльзуя задачу о разбиении.

Сначала нужно доказать, что наша задача лежит в классе NP. Для этого опишем недетерминированную машину Тьюринга, которую ее решает. Первым делом МТ недетерминированно порождает всевозможные подмножества предметов. Так как это происходит независимо на различных лентах, потребуется линейное от количества предметов число шагов. Теперь осталось проверить для каждого множества, что сумма стоимостей предметов больше K, а сумма весов меньше или равна той что умещается в рюкзак. Это также требует полиномаильного числа шагов. Если найдется хоть одно такое множество, то ответ в задаче положительный.

Теперь воспользуемся задачей о разбиении для доказательства NP-полноты. Рассмотрим такую подзадачу задачи о рюкзаке, в которой вес каждого предмета совпадает с его стоимостью. Пусть ограничение на вес равно минимальной стоимости. То есть, теперь нужно найти подмножество, сумма элементов строго равна заданному числу. Если мы установим это число равным половине суммы стоимостей всех предметов, то получим в точности задачу о разбиении. Таким образом, задача о рюкзаке также NP-полная.