Постановка задачи:

Пусть имеется n предметов. Каждый тип предмета i характеризуется весом wi, стоимостью ci и количеством предметов ki данного типа. Также имеется рюкзак вместимостью W.

Требуется собрать набор с максимальной полезностью таким образом, чтобы он имел вместимость не больше W. При этом количество предметов типа i не должно превышать k\_i.

В математической форме задача представлена на слайде. Т.е. требуется максимизировать первую функцию с ограничением в виде второго неравенства и того, что каждая переменная принимает строго определенные целые неотрицательные значения.

Метод ветвей и границ:

Эту задачу можно решить методом ветвей и границ.

Метод является развитием метода полного перебора. В отличие от последнего — с отсевом подмножеств допустимых решений, заведомо не содержащих оптимальных решений.

Дерево полного перебора, соответствующее поиску решения для трех предметов. В каждом узле определяется, будет ли данный предмет уложен в рюкзак. Цифра в узле соответствует номеру предмета. Цифры на рёбрах: 0 означает, что предмет не был взят, 1 — что был.

Алгоритм предлагает отсортировать предметы по их удельной стоимости (отношению ценности к весу) и строить дерево полного перебора. Его улучшение заключается в том, что в процессе построения дерева для каждого узла оценивается верхняя граница ценности решения, и продолжается построение дерева только для узла с максимальной оценкой. Когда максимальная верхняя граница оказывается в листе дерева, алгоритм заканчивает свою работу.

Суть метода заключается в разбиении исходной задачи на подзадачи в виде дерева и оценкой сверху решения каждой подзадачи. На основе этой оценке можно заранее отсечь некоторые варианты. Т.е. если у нас уже есть решение стоимостью, например, 200, а оценка сверху некоторой подзадачи 100, то нет смысла искать точное решение этой подзадачи.

Для оценки сверху мы ослабляем исходную задачу: разрешаем брать дробное число предмета. Такая задача решается алгоритмом Данцига за O(n).

Работу самого метода можно продемонстрировать на примере, представленном на экране. Далее описываем картинку: пусть мы честно посчитали для n=3, для предмета n=2 мы не будем так же честно считать, потому что оценка ниже текущего рекорда, n=1 не подходит по максимальной стоимости, n=0 дает новый рекорд.

Метод динамического программирования:

Также задачу о ранце можно решать с помощью метода динамического программирования

Подход динамического программирования состоит в том, что если при решении исходной задачи часто решаются одинаковые подзадачи, то имеет смысл сохранять решения таких подзадач, сократив тем самым количество вычислений.

В качестве таких подзадач будем решать задачу о ранце для первых n’<n предметов и рюкзака вместимости W’<W. Для хранения результатов введем массив dp[n’,W’], в каждой такой ячейке будем хранить максимальную стоимость набора, удовлетворяющему этому условию. Заполним этот массив следующим образом: для каждого n’ от 1 до n будем заполнять строку dp[n’,W’], где W’ изменяется от 1 до W.

Про заполнение одной ячейки. Тоже можно показать на примере картинки…

В итоге, в ячейке dp[n, W] будет лежать максимально возможная стоимость исходной задачи. Сложность алгоритма составляет O(n W^2), потому что таблица dp состоит из n\*W ячеек, а заполнение одной ячейки в худшем случае составляет O(W), т.к. вес предмета может равняться единице, а его кол-во быть очень большим.

Для восстановления ответа движемся в обратном направлении из ячейки dp[n, W]

Восстановление только одного решения имеет сложность O(n), т.к. мы для каждого предмета ищем только одно максимальное значение стоимости набора.

Пример работы:

Тут показываем третий пример с двумя решениями из отчета, но можем дать имена из второго примера для наглядности (т.е. вместо 1, 2, 3 можно написать Apple, Banana и т.п.)

На слайде продемонстрирован пример задачи, имеющей два решения. Оба решения, естественно, имеют одинаковую (максимальную) стоимость, но вес каждого решения может различаться.

Сравнение времени:

На слайде представлен график зависимости времени исполнения метода ветвей и границ и динамического программирования в зависимости от количества типов предметов на языке C++ при вместимости рюкзака W=10 000, количество каждого типа предмета не более 10.

На графике видно, что в случае небольшого количества типов предмета оба метода работают примерно за одно и то же время (сохраняется <<паритет>>), но при увеличении количества типов предметов метод <<Динамического программирования>> начинает работать заметно быстрее другого.

А при малом проигрывает, потому что на время ДП влияет еще размер рюкзака. (Ну и можно добавить, что в ДП больше операций с памятью, но это немного опасно.)

Также мы построили график зависимости времени исполнения обоих методов, реализованных в Wolfram Mathematica. Из графика видно, что метод динамического программирования снова выигрывает при большом количестве типов предмета.

Спасибо за внимание!