

Лабораторная работа № 7 по курсу дискретного анализа: жадные алгоритмы

Выполнил студент группы 08-303 МАИ Арусланов Кирилл.

Условие

Дано M видов пищевых добавок, каждая добавка содержит N действующих веществ в известных соотношениях. Для i -го вещества введён неизвестный коэффициент c_i (один и тот же для всех добавок), а воздействие одной мешка добавки определяется как

$$S = c_1 a_1 + c_2 a_2 + \cdots + c_N a_N,$$

где a_i — количество i -го вещества в данной добавке. Биолог может измерить воздействие любой добавки, потратив один мешок этой добавки (то есть получая одно линейное уравнение для коэффициентов c_i). Каждая добавка имеет цену (стоимость мешка). Необходимо подобрать самый дешёвый набор добавок (и их номера в исходном списке), по измерениям которых можно однозначно определить все коэффициенты c_1, \dots, c_N . Если это сделать невозможно (все доступные наборы дают ранг системы меньше N), вывести -1 . Порядок веществ в описаниях добавок одинаков, все числа неотрицательные целые и не больше 50. Известно, что $M \leq N$ не исключается (в этом случае ответ, как правило, -1).

Метод решения

Идея решения опирается на линейную алгебру и свойства векторных матроидов. Каждая добавка соответствует вектору из \mathbb{R}^N (или \mathbb{Q}^N); измерение добавки даёт линейное уравнение для векторов коэффициентов c . Набор измерений даёт систему линейных уравнений — и коэффициенты c однозначно определяются тогда и только тогда, когда матрица, составленная из выбранных строк (векторов добавок), имеет ранг N .

Задача минимизации суммы цен при требовании получить базис пространства (полный ранг) — классическая задача выбора минимума веса базиса в векторном матроиде. Для векторного матроида жадный алгоритм (сортировка элементов по неубыванию веса и поочерёдное включение элемента, если он увеличивает ранг множества) даёт оптимальное решение.

Реализованный алгоритм:

1. Считать M, N . Прочитать M строк по N чисел и цену каждой добавки.
2. Если $M < N$ — сразу возвращаем -1 .
3. Отсортировать индексы добавок в порядке неубывания цены (при равной цене — по возрастанию исходного индекса).

4. Идём в отсортированном порядке: пытаемся добавить в текущее множество следующую добавку; если ранг матрицы, составленной из выбранных строк, увеличился (по сравнению с предыдущим), фиксируем добавку в ответе, иначе отбрасываем её. Останавливаемся, как только достигнут ранг N или перебраны все добавки.
5. Если в конце размера выбранного множества меньше N — выводим -1 , иначе выводим номера выбранных добавок в порядке возрастания.

Для вычисления ранга используется метод Гаусса (приведение к верхнетреугольному/строчно-редуцированному виду) с вещественной арифметикой и малым порогом $\text{EPS} = 10^{-12}$ для сравнения с нулем.

Описание программы

Программа реализована в одном файле на C++. Основные компоненты:

- `rank_matrix(std::vector< std::vector< long double > > a)` — вычисляет ранг матрицы (строки = векторы добавок). Замечание: параметр передаётся по значению, поскольку в реализации выполняются нормализация и вычитания строк; это облегчает код, но влечёт дополнительные копирования.
- Главный код: чтение входных данных, сортировка индексов по цене, поочерёдное добавление строк и вызов `rank_matrix` для проверки увеличения ранга, формирование результата и вывод.

Дневник отладки

Проблем при разработке не возникало, программа прошла чеккер с первой попытки.

Тест производительности

Были проведены замеры на случайно сгенерированных данных (коэффициенты веществ в добавках в диапазоне $[-10, 10]$, цены в $[1, 1000]$, фиксированный генератор случайных чисел). Для каждой пары (M, N) измерялось время полного выполнения алгоритма (включая сортировку и многократные вызовы вычисления ранга). Полученные результаты:

M	N	Время (ms)
100	20	0
200	30	1
400	40	3
800	50	10
800	80	57
800	120	205

Анализ результатов показывает, что время работы растёт быстро при увеличении N . Теоретический анализ предсказывает асимптотику $T(M, N) = O(M \cdot N^3)$, и экспериментальная скорость увеличения времени при различных сериях измерений согласуется с этой оценкой (при фиксированном M рост по M практически линейный, при увеличении N при прочих равных — кубический рост).

Асимптотический анализ (коротко)

- Время сортировки индексов: $O(M \log M)$.
- Основной цикл: в худшем случае для каждой из M добавок вызывается `rank_matrix` на матрице размером до $N \times N$. Прямой метод Гаусса на квадратной матрице $N \times N$ работает за $O(N^3)$.
- Следовательно, худшая оценка времени: $O(M \cdot N^3)$. На практике итерации прекращаются, как только найден базис ранга N , поэтому среднее время часто значительно меньше.

Недочёты

Программа корректно решает поставленную задачу, однако есть места для улучшения производительности и качества реализации:

1. В `rank_matrix` матрица передаётся по значению, т.е. при каждом вызове выполняется копирование данных.
2. Используется `long double` для устойчивости, но в большинстве практических случаев `double` даёт достаточную точность и выполняется быстрее.
3. Можно было бы использовать `reserve()`.

Выводы

Реализованный алгоритм корректно находит минимальный по сумме цен набор добавок, достаточный для однозначного определения коэффициентов c_1, \dots, c_N (если такой набор существует). Алгоритм основан на жадной стратегии, которая допустима в силу того, что семейство независимых множеств образует матроид (векторный матроид) и жадный алгоритм минимального базиса в матроиде оптимален.

Сложность реализации невысока, но вычислительная сложность метода Гаусса даёт кубический рост по N , что делает решение затратным при больших значениях числа веществ N . Эксперимент подтвердил теоретическую оценку: при увеличении N время растёт кубически, а при увеличении M — примерно линейно, до момента достижения ранга N .