Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №8 по курсу «Дискретный анализ»

Студент: А. А. Кабанов
Преподаватель: С. А. Михайлова
Группа: М8О-301Б-22

Дата:
Оценка:
Подпись:

Москва, 2024

Лабораторная работа №8

Задача: Откорм бычков.

Бычкам дают пищевые добавки, чтобы ускорить их рост. Каждая добавка содержит некоторые из N действующих веществ. Соотношения количеств веществ в добавках могут отличаться.

Воздействие добавки определяется как $c_1a_1 + c_2a_2 + \cdots + c_Na_N$, где a_i — количество i-го вещества в добавке, c_i – неизвестный коэффициент, связанный с веществом и не зависящий от добавки. Чтобы найти неизвестные коэффициенты c_i, Биолог может измерить воздействие любой добавки, использовав один её мешок. Известна цена мешка каждой из $M(M \leq N)$ различных добавок. Нужно помочь Биологу подобрать самый дешевый наобор добавок, позволяющий найти коэффициенты c_i. Возможно, соотношения веществ в добавках таковы, что определить коэффициенты нельзя.

Формат ввода: В первой строке текста – целые числа M и N ; в каждой из следующих M строк записаны N чисел, задающих соотношение количеств веществ в ней, а за ними – цена мешка добавки. Порядок веществ во всех описаниях добавок один и тот же, все числа – неотрицательные целые не больше 50.

Формат вывода: Вывести -1 если определить коэффциенты невозможно, иначе набор добавок (и их номеров по порядоку во входных данных). Если вариантов несколько, вывести какой-либо из них.

1 Описание

Алгоритмы, предназначенные для решения задач оптимизации, - обычно последовательность шагов, на каждом из которых предоставляется некоторое множество выборов. Жадный алгоитм - тот, который выбирает самый лучший выбор на данный момент.

Переформулируем задачу так: дана линейная система линейных уравнений (количество уравнений не превышает количество переменных). Требуется найти все уравнения, которые могли бы позволить определить значения переменных и при этом оптимизировать общубю стоимость.

Основная часть алгоритма - применение метода Гаусса (приведение матрицы к ступенчатому виду). На каждом шаге выбираем строку матрицы с минимальной стоимостью; если в какой-то клетке матрицы возникает 0, решения нет - выводим -1.

Если шаг пройден успешно, то выведем индексы строк матрицы в возрастающем порядке.

2 Исходный код

Реализуем алгоритм, описанный в предыдущем пункте.

Опишем основной шаг алгоритма - метод Гаусса:

1. ищем минимальную строку по стоимости такую, что элемент по диагонали (по текущему i) ненулевой. Если такой строки нет, решения нет;
2. меняем местами рассматриваемую и найденную строку;
3. преобразуем к ступенчатому виду строки по текущему pivot-элементу.

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  using vd = vector<double>;
6  using vi = vector<int>;
7
8  int main() {
9    ios::sync_with_stdio(false);
10   cin.tie(nullptr);
11
12   int m, n;
13   cin >> m >> n;
14   vector<vd> v(m, vd(n + 2, 0));
15   for (size_t i = 0; i < m; ++i) {
16     for (size_t j = 0; j < n + 1; ++j) {
17       cin >> v[i][j];
18     }
19     v[i][n + 1] = i + 1;
20   }
21
22   vi min_v;
23   for (size_t i = 0; i < n; ++i) {
24     int min_cost = INT32_MAX;
25     int row = INT32_MIN;
26     for (size_t j = i; j < m; ++j) {
27       if (v[j][i] != 0 && v[j][n] < min_cost) {
28         min_cost = static_cast<int>(v[j][n]);
29         row = j;
30       }
31     }
32     if (row == INT32_MIN) {
33       cout << "-1\n";
34       return 0;
35     }
36     swap(v[i], v[row]);
```

```
    min_v.push_back(v[i][n + 1]);
    for (size_t j = i + 1; j < m; ++j) {
      double pivot = v[j][i] / v[i][i];
      for (size_t p = 1; p < n; ++p) {
        v[j][p] -= v[i][p] * pivot;
      }
    }
  }
  sort(min_v.begin(), min_v.end());
  for (const auto& elem : min_v) {
    cout << elem << " ";
  }
  cout << "\n";
}
```

3 Консоль

```
[anton@home lab8-greedy]$ make
g++ -O2 -lm -fno-stack-limit -std=c++20 -x c++ solution.cpp -o executable
[anton@home lab8-greedy]$ ./executable
3 3
1 0 2 3
1 0 2 4
2 0 1 2
-1
```

4 Тест производительности

Оценим сложность алгоритма: каждый из n раз (n строк) проводится приведение строк матриц сложностью $O(nm)$. Следовательно, общая сложность равна $O(n^2m)$.

Были проведены несколько тестов (от 10 до 50 строк). Время работы практически не изменилось: от 2 до 10 мс.

5 Выводы

Выполнив восьмую лабораторную работу по курсу «Дискретный анализ», я познакомился с жадными алгоритмами как способом решения - алгоритмами, заключающихся в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным.

Применение жадных алгоритмов можно увидеть в задачах на графах, алгоритмах сжатия и других.

Они, как мне показалось, легче понимаются и имплементируются, чем алгоритмы, основанные на динамическое программирование.

Список литературы

[1] *Жадный алгоритм - Википедия*
URL: `https://ru.wikipedia.org/wiki/Жадный_алгоритм` (дата обращения: 23.10.2024).