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. В. Семин
Преподаватель: А. А. Кухтичев
Группа: М8О-306Б-20

Дата:
Оценка:
Подпись:

Москва, 2022

Лабораторная работа №8

Задача: 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 $c_1a_1 + c_2a_2 + ... + c_Na_N$, 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 ($M \leqslant N$) 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

Приведем описание жадных алгоритмов из книги [1]:

> Алгоритмы, предназначенные для решения задач оптимизации, обычно представляют собой последовательность шагов, на каждом из которых предоставляется некоторое множество выборов. Определение наилучшего выбора, руководствуясь принципами динамического программирования, во многих задачах оптимизации напоминает стрельбу из пушки по воробьям; другими словами, для этих задач лучше подходят более простые и эффективные алгоритмы. В жадном алгоритме (greedy algorithm) всегда делается выбор, который кажется самым лучшим в данный момент — т.е. производится локально оптимальный выбор в надежде, что он приведет к оптимальному решению глобальной задачи.<...>
> Жадные алгоритмы не всегда приводят к оптимальному решению, но во многих задачах они дают нужный результат. Чтобы прийти к жадному алгоритму, сначала будет рассмотрено решение, основанное на парадигме динамического программирования, после чего будет показано, что оптимальное решение можно получить, исходя из принципов жадных алгоритмов. <...>
> Жадный метод обладает достаточной мощью и хорошо подходит для довольно широкого класса задач.

Следует выделить этапы построения жадного алгоритма:

- Привести задачу оптимизации к виду, когда после сделанного выбора остаётся решить только одну подзадачу.
- Доказать, что всегда существует такое оптимальное решение исходной задачи, которое можно получить путём жадного выбора, так что такой выбор всегда допустим.
- Продемонстрировать оптимальную структуру, показав, что после жадного выбора остаётся подзадача, обладающая тем свойством, что объединение оптимального решение подзадачи со сделанным жадным выбором приводит к оптимальному решению исходной задачиЭтапы построения жадного алгоритма:

Решение же непосредственно задачи из варианта состоит в следующем:
Приводим исходную матрицу к ступенчатому виду методом Гаусса. Главное, что нам не нужно искать решение системы, что упрощает задачу. Если на каком-то шаге не удается найти строку, содержащую ненулевой элемент, и привести матрицу к ступенчатому виду, то система несовместна, и её решения не существует.

Суть алгоритма: среди элементов первого столбца матрицы найти ненулевой, строка которого имеет наименьшую стоимость. Затем переместить его на крайнее верхнее положение перестановкой строк и вычесть получившуюся после перестановки первую строку из остальных строк, домножив её на величину, равную отношению первого элемента каждой из этих строк к первому элементу первой строки, обнуляя тем самым столбец под ним и выполняя преобразование матрицы к верхнетреугольному виду. После того, как указанные преобразования были совершены, проделываем тоже самое с остальными столбцами, только теперь перестановка строк и арифметические операции осуществляются со следующих по счёту строк сверху. В результате преобразований мы должны привести матрицу к верхнетреугольному виду, что позволяет в дальнейшем, согласно методу Гаусса, найти коэффициенты, хотя нам это и не надо в данном случае. Таким образом, если матрицу удалось привести к необходимому виду и она совместна, то решение существует. Также есть одна особенность: в задаче возможен случай, когда $N > M$, то есть количество коэффициентов будет больше, чем количество данных уравнений. В таком случае, найти их невозможно никаким образом.

2 Исходный код

Файл main.cpp:

```cpp
#include <bits/stdc++.h>

struct element {
  double value;
  int line;
  int price;
};


int main() {
  int n, m;
  std::cin >> m >> n;
  if (n > m) {//  >  ,
    std::cout << "-1\n";
    return 0;
  }
  std::vector<std::vector<element>> matr(m, std::vector<element> (++n));
  std::vector<int> res;
  // init
  for (int i = 0; i < m; i++) {
    int priceForI;
    for (int j = 0; j < n; j++) {
      double value;
      std::cin >> value;
      matr[i][j] = {value, i+1, -1};
      if (j == n - 1) {
        priceForI = value;
      }
    }
    for (int j = 0; j < n; j++) {
      matr[i][j].price = priceForI;
    }
  }
  // solve
  int curLine=0;
  for (int curRow = 0; curRow < n-1; curRow++) { //
    int minLine = -1;
    int minPrice = 111;

    for (int i = curLine; i < m; i++) { //
      if (matr[i][curRow].value != 0.0 && matr[i][curRow].price < minPrice) {
        minPrice = matr[i][curRow].price;
        minLine = i;
      }
    }

```

```
47      if (minLine == -1) {
48        std::cout << "-1\n";
49        return 0;
50        // continue;
51      }
52
53      res.push_back(matr[minLine][0].line);
54      matr[curLine].swap(matr[minLine]); //
55      for (int i = curLine+1; i < m; i++) { //  ,   ,
56        double div = matr[i][curRow].value / matr[curLine][curRow].value;
57        for (int k = curRow; k < n; k++) {
58
59          matr[i][k].value -= matr[curLine][k].value * div;
60        }
61      }
62      curLine++;
63    }
64
65    sort(res.begin(), res.end());
66    for (int i: res) {
67      std::cout << i << " ";
68    }
69  }
```

3 Консоль

console input:

```
3 3
1 0 2 3
1 0 2 4
2 0 1 2
```

console output:

```
-1
```

4 Тест производительности

Тест производительности представляет собой сравнение с наивным решением этой задачи, в котором перебираются все возможные подсистемы уравнений, а потом выбирается решение с наименьшей стоимостью.

Сравнение производится путем запуска запуска теста на большой матрице.
Тестовый файл содержит матрицу 90 на 90:

В результате работы видны следующие результаты:

```
a@WIN-THNQL51M105:~/Desktop/DA/lab8$ ./lab8 <test90x90
Greed time: 0.003213
a@WIN-THNQL51M105:~/Desktop/DA/lab8$ ./bench <test90x90
Naive time: 0.004 sec.
```

5 Выводы

Выполнив восьмую лабораторную работу по курсу «Дискретный анализ», мною были изучены жадные алгоритмы, а также различные задачи, которые с помощью них возможно решить наиболее оптимальным образом.
Стоит отметить, следующее: жадным является любой алгоритм, который на каждом шаге делает локально наилучший выбор в надежде, что итоговое решение будет оптимальным. В некотором смысле можно считать, что жадные алгоритмы – частный случай алгоритмов динамического программирования. Разница лишь в том, что в динамическом программировании подзадачи решаются до выполнения первого выбора, а жадный алгоритм делает первый выбор до решения подзадач. Область применения жадных алгоритмов широка: кодирование Хаффмана для сжатия данных, алгоритмы аллокации в операционных системах и другие.

Список литературы

[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. *Алгоритмы: построение и анализ, 2-е издание.* — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))