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: П. А. Харьков
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б-19

Дата:
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №8

Задача: Разработать жадный алгоритм решения задачи, определяемой своим вариантом. Доказать его корректность, оценить скорость и объём затрачиваемой оперативной памяти.

Вариант задания: 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 $\geq$ 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 входных данных: 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 результата: -1, если определить коэффициенты невозможно, иначе набор добавок (и их номеров по порядоку во входных данных). Если вариантов несколько, вывести какой-либо из них.

1 Описание

Как сказано в [1]: «В жадном алгоритме всегда делается выбор, который кажется лучшим на данный момент, в надежде на то, что он приведет к оптимальному решению глобальной задачи». Жадные алгоритмы используются в таких задачах как "размен монет составление максимального треугольника и был положен в основу алгоритма Дейкстры. Для того, чтобы доказать, что он этот алгоритм приведет к оптимальному решению можно использовать математические доказательство и доказательство через матроиды.

Самым очевидным доказательставом, что решить данную задачу и получить оптимальный ответ можно с помощью жадного алгоритма является то, что на каждом шагу нам нужно взять подходящий мешок с минимальной ценой, а значит в результате мы возьмем N мешков с минимальной ценой, то есть результат будет минимальный.

2 Исходный код

Я буду рассматривать набор мешков, как систему уравнений - то есть, для того, чтобы можно было найти коэффициенты у количеств веществ в мешке, нам необходимо найти совместную систему уравнений, состоящую из N уравнений, и имеющую единственное решение.

Для начала я считываю все количества веществ в мешках в матрицу *subst* и цены за каждый мешок в вектор *prices*. Затем для каждого вещества я ищу самый дешевый мешок, количество вещества которого не равно 0 - чтобы гарантировать, что существует хотя бы одно уравнение, определяющие коэффициент для этого вещества. Если такого мешка нет, то значит, что определить коэффициенты невозможно. Затем я убираю пропорциональные строки из матрицы тем, что вычитаю из каждой строки числа, пропорциональные выбранной строке, и ставлю цену выбранного мешка больше максимальной, чтобы его нельзя было снова выбрать.

В конце я сортирую результат, так как взятые мешки могут не идти последовательно.

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <string>
using namespace std;

int main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);

    int m, n;
    bool ok = true;
    cin >> m >> n;
    vector<int> res(n);
    vector<int> prices(m);
    vector<vector<double>> subst(m, vector<double>(n));

    for (int i = 0; i < m; i++){
        for (int j = 0; j < n; j++){
            cin >> subst[i][j];
        }
        cin >> prices[i];
    }

    for (int j = 0; j < n; j++){
        int minPrice = 51; int minRow = -1;
        for (int i = 0; i < m; i++){
            if (prices[i] < minPrice && subst[i][j] != 0){
                minRow = i;
                minPrice = prices[i];
```

```
            }
        }

        if (minRow == -1){
            ok = false;
            break;
        }

        prices[minRow] = 51;
        res[j] = minRow + 1;

        for (int i = 0; i < m; i++){
            if(prices[i] == 51){
                continue;
            }

            double koeff = subst[i][j] / subst[minRow][j];
            for (int col = j; col < n; col++){
                subst[i][col] = subst[i][col] - (subst[minRow][col] * koeff);
            }
        }

    }

    if(!ok){
        cout << "-1\n";
        return 0;
    }

    sort(res.begin(), res.end());
    for (int i = 0; i < n; i++){
        cout << res[i] << ' ';
    }
    cout << '\n';

    return 0;

}
```

3 Консоль

```
p.kharkov$ make
g++ -std=c++14 -pedantic -Wall src/laba8.cpp -o laba8
p.kharkov$ cat tests/main
3 3
1 0 2 3
1 0 2 4
2 0 1 2
p.kharkov$ ./laba8 <tests/main
-1
```

4 Тест производительности

Тест производительности представляет из себя следующее: сравнивается жадный алгоритм решения и наивный алгоритм, то есть полный перебор всех возможных комбинаций мешков.

Первый тест состоит из $M = 10$ и $N = 10$ элементов, максимальное значение каждого равно 50. Как можно увидеть, жадный алгоритм отработал за 0 миллисекунда, а наивный за 608:

```
p.kharkov$ ./benchmark <tests/01.t
Greed solution time: 0ms
Naive solution time: 608ms
```

Второй тест состоит из $M = 40000$ и $N = 1000$ элементов, максимальное значение каждого равно 50. Я решил не тестировать тест на наивном алгоритме, а время работы жадного алгоритма равно времени работы наивного на первом тесте:

```
p.kharkov$ ./benchmark <tests/02.t
Greed solution time: 651ms
```

Как видно, жадный алгоритм решения работает гораздо быстрее наивного - это связано с тем, что сложность жадного алгоритма $O(n^2 * m)$, а сложность наивной реализации - $O(n^2 * 2^m * m)$.

5 Выводы

Выполнив восьмую лабораторную работу по курсу «Дискретный анализ», я научился разрабатывать жадный алгоритм для решения задач. Иногда использовать динамическое программирование для решение задачи является недостаточно эффективным и необходимо находить решение более простым и быстрым способом. К сожалению, при решение с помощью жадного алгоритм не всегда получается самое точное решение, в отличие от того же динамического программирования, но чаще всего оно является почти оптимальным для этой задачи. Также жадные алгоритмы используются в кодах Хаффмана, поисках путей в графах и во многом другом.

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.))

[2] *Жадный алгоритм — Википедия.*
URL: `https://ru.wikipedia.org/wiki/Жадный_алгоритм` (дата обращения: 7.05.2021).