

Лабораторная работа № 8 по курсу дискретного анализа: Жадные алгоритмы

Выполнил студент группы М80-308Б-22 МАИ *Кочкожаров Иван*.

Условие

Краткое описание задачи:

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

Реализовать программу на языке С или С++, соответствующую построенному алгоритму. Формат входных и выходных данных описан в варианте 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

$$c_1 \times a_1 + c_2 \times a_2 + \dots + c_n \times a_n,$$

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, используя один её мешок. Известна цена мешка каждой из M ($M \geq N$) различных добавок. Нужно помочь Биологу подобрать самый дешёвый набор добавок, позволяющий найти коэффициенты c_i . Возможно, соотношения веществ в добавках таковы, что определить коэффициенты нельзя.

Метод решения

Жадные алгоритмы применимы в том случае, если принятие наиболее оптимального решения на каждом шаге решения задачи означает наиболее оптимальное решение задачи в целом. К этой задаче можно применить жадный алгоритм, поскольку чтобы её решить, мы должны отобрать ровно N добавок, а значит эти добавки должны быть наиболее дешёвыми. Идея решения в том, чтобы привести матрицу, составленную из соотношений веществ (для этого вызывается функция `SubtractRows`), к ступенчатому виду, при этом наверх продвигать строки, характеризующие наиболее дешёвые добавки (строка ищется функцией `FindLowestPriceRow`), тогда N верхних строк и будут ответом к задаче. Для получения ответа, необходимо сохранять номера добавок. Таким образом, алгоритм заключается в прохождении по всем добавкам, вызывая функции `FindLowestPriceRow` и выдвигании найденной строки вперед и вычитанию этой строки из последующих, для приведения матрицы к ступенчатому виду

Итоговая сложность

Описание программы

Разделение по файлам, описание основных типов данных и функций.

```
#include <algorithm>
#include <iostream>
#include <vector>

constexpr const int MAX_NUM = 50;

struct Addition {
    Addition(size_t n) : ratios(n) {}
    std::vector<double> ratios;
    int price;
    int index;
};

int FindLowestPriceRow(std::vector<Addition> &v, int t) {
    size_t m = v.size();
    size_t n = v[0].ratios.size();
    int minPrice = MAX_NUM + 1;
    int index = -1;
    for (int i = t; i < m; ++i) {
        if ((v[i].ratios[t] != 0.0) && (v[i].price < minPrice)) {
            index = i;
            minPrice = v[i].price;
        }
    }
    return index;
}

void SubtractRows(std::vector<Addition> &v, int t) {
    size_t m = v.size();
    size_t n = v[0].ratios.size();
    for (int i = t + 1; i < m; ++i) {
        double coeff = v[i].ratios[t] / v[t].ratios[t];
        for (int j = t; j < n; ++j) {
            v[i].ratios[j] -= v[t].ratios[j] * coeff;
        }
    }
}

int main() {
```

```

    int n, m;
    std::cin >> m >> n;
    std::vector<int> res;
    std::vector<Addition> additions(m, Addition(n));
    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < n; ++j) {
            std::cin >> additions[i].ratios[j];
        }
        std::cin >> additions[i].price;
        additions[i].index = i;
    }
    for (int i = 0; i < n; ++i) {
        int index = FindLowestPriceRow(additions, i);
        if (index == -1) {
            std::cout << "-1" << std::endl;
            return 0;
        }

        std::swap(additions[i], additions[index]);
        res.push_back(additions[i].index);
        SubtractRows(additions, i);
    }

    std::sort(res.begin(), res.end());
    for (auto r : res) {
        std::cout << r + 1 << ' ';
    }
    std::cout << '\n';
    return 0;
}

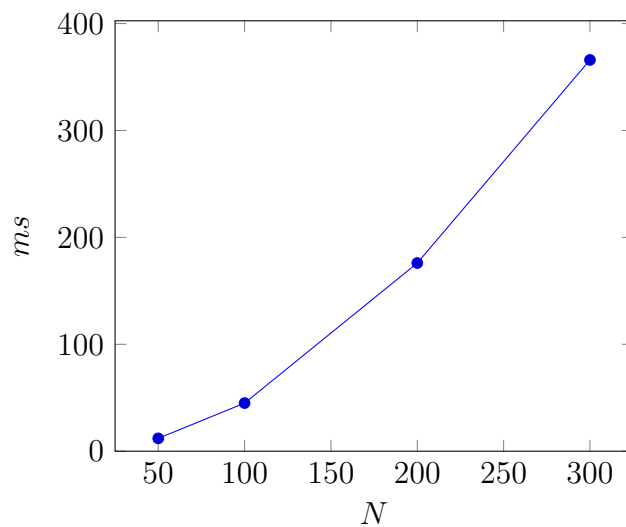
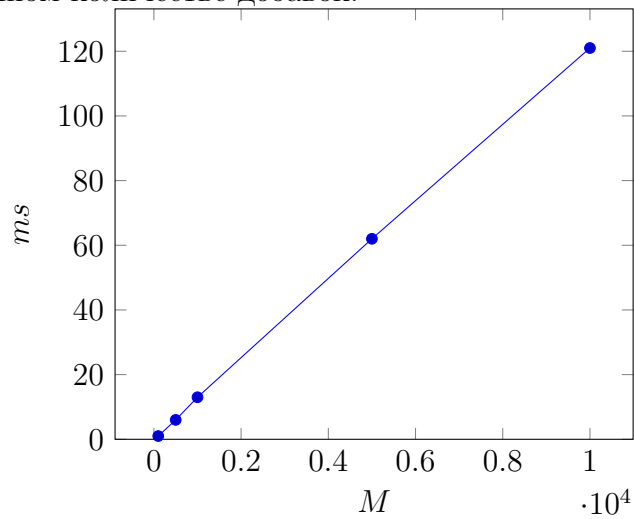
```

Дневник отладки

1. Было решено объединять все строки в матрицу и приводить ее к ступенчатому виду постепенно, вместо формирования матрицы для каждого набора строк и проверки его ранга.
2. Изменены типы данных некоторых переменных
3. Выполнена отладка некоторых функций, обнаружены ошибки в индексах матрицы

Тест производительности

Ниже приведен тест времени работы алгоритма. По оси X — количество добавок при фиксированном количестве веществ, по оси Y — время выполнения алгоритма в мс (меньше — лучше). На втором графике изменяется количество веществ при фиксированном количестве добавок.



Сложность алгоритма $O(n^2m)$ показана графически

Ниже приведены программы, использовавшиеся для генерации данных для теста:

```
import random
import sys

file = open('test.txt', 'w')

m = int(sys.argv[1])
n = int(sys.argv[2])
```

```
MAX_NUM = 50
```

```
file.write(str(m) + ' ' + str(n) + '\n')
```

```
for _ in range(m):
```

```
    for _ in range(n):
```

```
        file.write(str(random.randint(0, MAX_NUM)) + ' ')
```

```
        file.write(str(random.randint(0, MAX_NUM)) + '\n')
```

```
file.close()
```

Выводы