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: | Д. С. Ляшун |
| Преподаватель: | А. А. Кухтичев |
| Группа: | М8О-207Б |
| 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.

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

**Вариант: 4.**

**Описание:** 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 дешёвый набор добавок, позволяющий найти коэффициенты $c_i$. Возможно, соотношения веществ в добавках таковы, что определить коэффициенты нельзя.

# 1 Описание

Требуется написать реализацию алгоритма решения поставленной задачи, которую можно переформулировать так: среди $M$ линейных уравнений с $N$ неизвестными необходимо выбрать такие, которые образуют систему с единственным решением, при этом она является минимальной по стоимости (стоимость системы – сумма стоимости входящих в неё уравнений). Система линейных арифметических уравнений имеет единственное решение, когда строки с её коэффициентами являются линейно-независимыми, что можно проверить, найдя ранг матрицы, образуемой из строк с коэффициентами каждого уравнения.

Очевидно, что выгоднее брать мешки, начиная с самых дешёвых. Для этого сперва необходимо отсортировать их по неубыванию цены. Далее будет происходить последовательное добавление мешков в ответ, при этом каждый новый мешок перед добавлением нужно проверить на то, что с уже добавленными мешками матрица с количествами их вещества состоит из линейно-независимых строк. Для проверки этого будет использоваться модифицированный метод Гаусса [2] решения системы линейных уравнений – выполняются абсолютно те же самые операции, что и при решении системы, но если на каком-либо шаге в $i$-ом столбце среди невыбранных до этого строк нет ненулевых, то этот шаг пропускается, и ранг матрицы уменьшается на единицу (изначально ранг полагается равным $max(N, m)$, где $m$ - текущее число взятых мешков). Иначе, если найдена на $i$-ом шаге строка с ненулевым элементом в $i$-ом столбце, то она помечается как выбранная, и выполняются обычные операции отнимания этой строки от остальных. Если среди $M$ мешков нельзя набрать $N$ таких, которые образуют линейно-независимые строки, то тогда можно сделать вывод о том, что коэффициенты с таким набором мешков найти не получится.

Оценим время работы полученного алгоритма. Сортировка $M$ мешков по неубыванию стоимости будет выполняться за $O(Mlog_2M)$, выбор среди $M$ мешков $N$ штук в худшем случае произойдет за $O(M)$, при этом проверка на допустимость включения (строки с количествами являются линейно-независимыми) методом Гаусса будет работать за $O(N^2M)$ – всего выполняется $N$ итераций, на каждой из которых выбирается неиспользованная строка с ненулевым $i$ коэффициентом, которая затем отнимается от других строк с ненулевым $i$ коэффициентом за $O(MN)$. Итого, суммарное время работы алгоритма составляет $O(Mlog_2M + N^2M) = O(N^2M)$, при $N \approx M$ – $O(N^3)$.

## 2 Исходный код

Ниже приведён исходный код программы, в котором производится: чтение входных данных – количества мешков $M$ и число добавок $N$, а также количеств вещества в каждом мешке и их стоимость; затем производится сортировка мешков по неубыванию цены; после чего выполняется последовательный выбор среди них $N$ таких, что матрица, составленная из их количеств вещества, является линейно независимой, что указывает на найденный ответ, в противном случае найти коэффициенты нельзя:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
const double EPS = 1E-9;
int CalculateRank(std::vector<std::vector<double> > a) {
    int n = a.size();
    int m = a[0].size();
    int rank = std::max(n, m);
    std::vector<bool> line_used(n);
    for (int i = 0; i < m; ++i) {
        int j;
        for (j = 0; j < n; ++j) {
            if (!line_used[j] && abs(a[j][i]) > EPS) {
                break;
            }
        }
        if (j == n) {
            --rank;
        }
        else {
            line_used[j] = true;
            for (int p = i + 1; p < m; ++p) {
                a[j][p] /= a[j][i];
            }
            for (int k = 0; k < n; ++k) {
                if (k != j && std::abs(a[k][i]) > EPS) {
                    for (int p = i + 1; p < m; ++p) {
                        a[k][p] -= a[j][p] * a[k][i];
                    }
                }
            }
        }
    }
    return rank;
}

int main() {
    int n, m;
```

```
    std::cin >> m >> n;
    std::vector<std::vector<double> > counts(m, std::vector<double>(n));
    std::vector<std::pair<int, int> > bags(m);
    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < n; ++j) {
            std::cin >> counts[i][j];
        }
        std::cin >> bags[i].first;
        bags[i].second = i;
    }
    std::sort(bags.begin(), bags.end());
    std::vector<std::vector<double> > matrix;
    std::vector<int> answer;
    for (int i = 0; i < m && answer.size() != n; ++i) {
        matrix.push_back(counts[bags[i].second]);
        if (CalculateRank(matrix) == matrix.size()) {
            answer.push_back(bags[i].second);
        }
        else {
            matrix.pop_back();
        }
    }
    if (answer.size() != n) {
        std::cout << -1;
    }
    else {
        std::sort(answer.begin(), answer.end());
        for (int i = 0; i < answer.size(); ++i) {
            std::cout << answer[i] + 1 << " ";
        }
    }
    std::cout << std::endl;
    return 0;
}
```

# 3 Консоль

```
dmitry@dmitry-VirtualBox:~/Work_place/DA_labs/lab8$ g++ main.cpp -o main
dmitry@dmitry-VirtualBox:~/Work_place/DA_labs/lab8$ cat test.txt
3 3
1 0 2 3
1 0 2 4
2 0 1 2
dmitry@dmitry-VirtualBox:~/Work_place/DA_labs/lab8$ ./main <test.txt
-1
dmitry@dmitry-VirtualBox:~/Work_place/DA_labs/lab8$ cat test.txt
4 3
1 2 3 4
0 0 4 1
2 2 2 2
3 4 5 2
dmitry@dmitry-VirtualBox:~/Work_place/DA_labs/lab8$ ./main <test.txt
2 3 4
```

# 4 Тест производительности

Тест производительности представляет из себя сравнение времени работы жадного алгоритма и наивного решения, основанного на переборе всех возможных вариантов наборов мешков.

| Число мешков $M$ | Число веществ $N$ | Время работы жадного алгоритма | Время работы наивного решения |
|---|---|---|---|
| 5 | 3 | 83 мкс. | 180 мкс. |
| 10 | 3 | 42 мкс. | 352 мкс. |
| 15 | 5 | 127 мкс. | 18725 мкс. |
| 20 | 5 | 127 мкс. | 98223 мкс. |
| 25 | 7 | 184 мкс. | 3552376 мкс. |

Как видно, несмотря на то, что размер входных данных довольно небольшой, с ростом числа мешков и веществ в них программа, реализованная на основе полного перебора, начинает довольно сильно проседать по времени работы. Это связано с тем, что ей всегда необходимо рассматривать все возможные пути решения, которых в общем случае будет $C_M^N = \frac{M!}{N!(M-N)!}$. Также стоит отметить, что время работы жадного алгоритма на данных тестах здесь почти постоянное, это связано с особенностью генератора тестов, полученные количества веществ почти всегда образовывали линейно-независимые строки матрицы, отчего ответом являлись первые N мешков, взятых в порядке неубывания их цены.

# 5 Выводы

Выполнив восьмую лабораторную работу по курсу «Дискретный анализ», я познакомился с идеей жадных алгоритмов для решения различных оптимизационных задач и написал программу, реализующую такую идею для нахождения системы уравнений, образующих линейно-независимые строки в матрице, с минимальной стоимостью.

Основную трудность в ходе выполнения работы для меня составил правильный выбор жадного алгоритма, в частности, в первом варианте идеи решения я производил выбор минимального по стоимости уравнения для каждого ненулевого соответствующего ему коэффициента, и это не гарантировало того, что выбранные уравнения образуют линейно-независимые строки.

# 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] *MAXimal: Нахождение ранга матрицы*
URL: `https://e-maxx.ru/algo/matrix_rank` (дата обращения: 11.05.2021).