МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧЕРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»

(БГТУ им.В.Г.Шухова)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №2.2

дисциплина: Дискретная математика

тема: «Задачи выбора»

Выполнил: ст. группы ПВ-201

Машуров Дмитрий Русланович

Проверил: Бондаренко Т.В.

Белгород 2021

Лабораторная работа № 2.2

Задачи выбора

Цель работы: приобрести практические навыки в использовании алгоритмов порождения комбинаторных объектов при проектировании алгоритмов решения задач выбора.

Задания

1. Ознакомиться с задачей (см. варианты заданий).
2. Определить класс комбинаторных объектов, содержащих решение задачи (траекторию задачи).
3. Определить, что в задаче является функционалом и способ его вычисления.
4. Определить способ распознавания решения по значению функционала.
5. Реализовать алгоритм решения задачи.
6. Подготовить тестовые данные и решить задачу.

Задание варианта №17

Задана целочисленная матрица размером n × m. Выбрать минимальное количество строк матрицы, таких, что сумма элементов каждого столбца была бы больше заданного числа.

Выполнение:

1. Определю класс комбинаторных объектов, содержащих решение задачи (траекторию задачи):

   Исходя из условия задачи, мне необходимо получить различные комбинации строк матрицы, которые будут участвовать в вычислении минимального значения

   Таким образом классом комбинаторных объектов будут являться подмножества множества строк матрицы

2. Определю, что в задаче является функционалом и способ его вычисления

   Функционалом будет являться двоичный вектор D, определяющий подмножества множества строк матрицы

   Вычисляться он будет с помощью переборного алгоритма порождения подмножеств множества

3. Определю способ распознавания решения по значению функционала:

   Двоичный вектор D (функционал), определяющий некоторое подмножество множества строк матрицы, будет являться решением, если сумма каждого столбца данного подмножества строк будет больше заданного числа num

4. Реализую алгоритм решения задачи:

```c
#include <stdio.h>

#define N 100

int MIN_ROWS = N, num, ROWS_MASK[N], a[N][N];

/*вывод матрицы с маской row_mask*/
void print_matr_with_row_mask(const int row_mask[], int m, int n) {
    for (int i = 0; i < m; ++i) {
        if (row_mask[i]) {
            for (int j = 0; j < n; ++j) {
                printf("%d ", a[i][j]);
            }

            printf("\n");
        }
    }
}

/*копирование массива this размера n в массив other*/
void copy_arr(const int this[], int n, int other[]) {
    for (int i = 0; i < n; ++i) {
        other[i] = this[i];
    }
}

/*возвращает сумму столбца col по выбранной маске строк mask длины m*/
int get_col_sum_by_mask(const int *mask, int m, int col) {
    int sum = 0;

    for (int i = 0; i < m; ++i) {
        if (mask[i] == 1) {
            int t = a[i][col];
            sum += t;
        }
    }

    return sum;
}

/*возвращает "истину", если каждое значение массива cols_sum размера n
 *превышает num, иначе - "ложь"*/
static int check_cols_sum(int n, const int *cols_sum) {
    for (int j = 0; j < n; ++j) {
        if (cols_sum[j] <= num) {
            return 0;
        }
    }

    return 1;
}

/*помещает сумму столбцов в массив cols_sum*/
static void set_cols_sum(int m, int n, const int *d, int *cols_sum) {
    for (int j = 0; j < n; ++j) {
        int sum = get_col_sum_by_mask(d, m, j);
        cols_sum[j] = sum;
    }
}
```

```c
/*рекурсивная функция: генерирует подмножества множества строк матрицы a
 *размера m строк и n столбцов*/
static void get_subsets_inner(int m, int n, int i) {
    static int d[100];
    static int cols_sum[N];

    for (int x = 0; x < 2; ++x) {
        d[i] = x;

        if (i == n - 1) {
            set_cols_sum(m, n, d, cols_sum);

            int c = 0;
            for (int j = 0; j < n; ++j) {
                if (d[j] == 1) c++;
            }

            if (check_cols_sum(n, cols_sum) && c < MIN_ROWS) {
                MIN_ROWS = c;
                copy_arr(d, n, ROWS_MASK);
            }
        } else {
            get_subsets_inner(m, n, i + 1);
        }
    }
}

/*функция-оболочка для get_subsets_inner*/
void get_rows_subsets(int m, int n) {
    get_subsets_inner(m, n, 0);
}

int main() {
    scanf("%d", &num);

    int n, m;
    scanf("%d %d", &m, &n);

    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < n; ++j) {
            scanf("%d", &a[i][j]);
        }
    }

    get_rows_subsets(m, n);
    printf("%d\n", MIN_ROWS);
    print_matr_with_row_mask(ROWS_MASK, m, n);

}
```

Тестовые данные:

№	Вход	Выход
1	10 4 3 1 2 3 5 5 5 6 7 8 4 3 2	2 5 5 5 6 7 8
2	5 3 3 2 3 1 4 3 8 6 1 1	2 2 3 1 4 3 8
3	7 4 4 7 6 3 5 0 1 5 4 3 3 3 3 5 6 7 8	2 3 3 3 3 5 6 7 8