**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №3**

**по дисциплине «Построение и анализ алгоритмов»**

**Тема: Метод ветвей и границ. Задача о назначениях**

|  |  |  |
| --- | --- | --- |
| Студент гр. 5382 |  | Никитин В.А. |
| Преподаватель |  | Шолохова О.М. |

Санкт-Петербург

2017 г.

**Цель работы.**

Ознакомиться с методом ветвей и границ и получить навыки программирования данного метода на языке программирования С++.

**Постановка задачи.**

Имеется N человек, которых нужно назначить на N работ. Стоимость назначения i-го человека на j-ю работу равна C[i][j]. Задача состоит в отыскании назначения, при котором каждая работа выполняется некоторым человеком и которое минимизирует общую стоимость назначения.

**Основные теоретические положения.**

**Метод ветвей и границ** - общий алгоритмический метод для нахождения оптимальных решений различных задач оптимизации. По существу, метод является вариацией полного перебора с отсевом подмножеств допустимых решений, заведомо не содержащих оптимальных решений.

Общая идея метода может быть описана на примере поиска минимума функции **F(x)** на множестве допустимых значений переменной **x**. Функция **F** и переменная **x** могут быть произвольной природы. Для метода ветвей и границ необходимы две процедуры: ветвление и нахождение оценок (границ).

Процедура *ветвления* состоит в разбиении множества допустимых значений переменной **x** на подобласти (подмножества) меньших размеров. Процедуру можно рекурсивно применять к подобластям. Полученные подобласти образуют дерево, называемое *деревом поиска* или *деревом ветвей и границ*. *Узлами* этого дерева являются построенные подобласти (подмножества множества значений переменной **x**).

Процедура *нахождения оценок* заключается в поиске верхних и нижних границ для решения задачи на подобласти допустимых значений переменной **x**

В основе метода ветвей и границ лежит следующая идея: если нижняя граница значений функции на подобласти **A** дерева поиска больше, чем верхняя граница на какой-либо ранее просмотренной подобласти **B**, то **A** может быть исключена из дальнейшего рассмотрения (*правило отсева*). Обычно минимальную из полученных верхних оценок записывают в глобальную переменную **m**; любой узел дерева поиска, нижняя граница которого больше значения **m**, может быть исключён из дальнейшего рассмотрения.

Если нижняя граница для узла дерева совпадает с верхней границей, то это значение является минимумом функции и достигается на соответствующей подобласти.

**Спецификация программы.**

*Назначение программы*.

Программа предназначена для решения задачи о назначениях методом ветвей и границ.

*Описание программы*.

*Ключевая идея*: мы будем добавлять в рассмотрение строки матрицы одну за одной.

* Добавляем в рассмотрение очередную строку матрицы a.
* Пока нет увеличивающей цепи, начинающейся в этой строке, пересчитываем потенциал.
* Как только появляется увеличивающая цепь, чередуем паросочетание вдоль неё (включая тем самым последнюю строку в паросочетание), и переходим к началу (к рассмотрению следующей строки).
* После каждого пересчёта потенциала мы просматриваем добавившиеся жёсткие рёбра и, если их левые концы были достижимыми, помечаем их правые концы также как достижимые и продолжаем обход из них.
* Развивая эту идею дальше, можно прийти к такому представлению алгоритма: это цикл, на каждом шаге которого сначала пересчитывается потенциал, затем находится столбец, ставший достижимым (а таковой всегда найдётся, поскольку после пересчёта потенциала всегда появляются новые достижимые вершины), и если этот столбец был ненасыщен, то найдена увеличивающая цепь, а если столбец был насыщен — то соответствующая ему в паросочетании строка также становится достижимой.

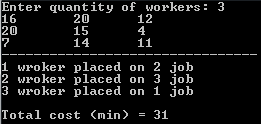
Теперь алгоритм принимает вид: цикл добавления столбцов, на каждом из которых сначала пересчитывается потенциал, а затем какой-то новый столбец помечается как достижимый.

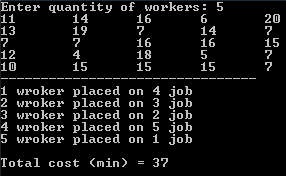
*Реализация*.

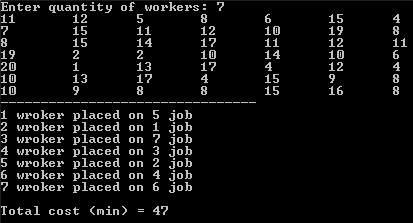
Функции:

* void check(int \*\*arr, int \*work, int n) // основная функция реализующая алгоритм. На функцию подаются матрица работник/работа, массив, в который заносятся распределение работ по работникам и количество работ (размер матрицы).

**Тестирование.**







**Выводы**

В ходе выполнения лабораторной работы были получены практические навыки реализации метода ветвей и границ на языке C++ на примере задачи о назначениях.

**Приложение А. Исходный код.**

#include "stdafx.h"

#include <iostream>

#include <ctime>

using namespace std;

const int MAXINT = 1000;

void check(int \*\*arr, int \*work, int n)

{

int\* RowSumm = new int[n];//индекс - работник, значение-работа

int\* ColSumm = new int[n];

for (int count = 0; count < n; count++){

RowSumm[count] = 0;

ColSumm[count] = 0;

}

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

cout << arr[i][j] << "\t ";

}

cout << endl;

}

cout << "--------------------------------" << endl;

int minzp = 0;

for (int i1 = 0; i1 < n; i1++)

{

for (int j1 = 0; j1 < n; j1++)

{

int minRow = MAXINT, minCol = MAXINT, sumRow = 0, sumCol = 0;

for (int i = i1 + 1; i < n - i1; i++)

{

for (int j = 0; j < n; j++)

{

if (i != i1 && j != j1)

{

if ((j == n - 1 && j != j1) || (j == n - 2 && j + 1 == j1))

{

if (minRow>arr[i][j]){ sumRow = sumRow + arr[i][j]; }

else if (minRow != MAXINT) { sumRow = minRow + sumRow; }

minRow = MAXINT;

}

else if (minRow>arr[i][j])minRow = arr[i][j];

}

}

}

for (int j = 0; j < n; j++)

{

for (int i = i1 + 1; i < n - i1; i++)

{

if (i != i1 && j != j1)

{

if ((i == n - 1 && i != i1) || (i == n - 2 && i + 1 == j1))

{

if (minCol>arr[i][j]){ sumCol = sumCol + arr[i][j]; }

else if (minCol != MAXINT){ sumCol = minCol + sumCol; }

minCol = MAXINT;

}

else if (minCol>arr[i][j])minCol = arr[i][j];

}

}

}

RowSumm[j1] = sumRow + arr[i1][j1];

ColSumm[j1] = sumCol + arr[i1][j1];

for (int m = 0; m<n; m++){

if (work[m]<n + 1 && work[m]>0){ RowSumm[work[m] - 1] = MAXINT; ColSumm[work[m] - 1] = MAXINT; }

}

}

int min1 = RowSumm[0];

for (int j = 0; j < n; j++)

{

if (RowSumm[j]<ColSumm[j])

{

if (min1 >= RowSumm[j]) { min1 = RowSumm[j]; work[i1] = j + 1; }

}

else

{

if (min1 >= ColSumm[j]){ min1 = ColSumm[j]; work[i1] = j + 1; }

}

}

cout << i1 + 1 << " wroker placed on " << work[i1] << " job" << endl;

minzp = minzp + arr[i1][work[i1] - 1];

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

if (i == i1 || j == work[i1] - 1)arr[i][j] = MAXINT;

}

}

}

cout << endl << "Total cost (min) = " << minzp << endl;

cout << endl;

}

int main()

{

int n=0;

cout << "Enter quantity of workers: ";

cin >> n;

while (!cin){

cin.clear();

while (cin.get() != '\n') continue;

cout << "Enter a number, try again: ";

cin >> n;

}

int \*\*arr; //массив стоимостей

arr = new int\*[n];

for (int i = 0; i < n; i++) arr[i] = new int[n];

srand(time(0));

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++) arr[i][j] = rand() % 20+1;

int\* work = new int[n]; //массив распределеиня работников по работам

for (int count = 0; count < n; count++) work[count] = -1;

check(arr, work, n);

system("pause");

return 0;

}