Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ
КАФЕДРА ИНФОРМАТИКИ

Лабораторная работа №5
«Поиск максимального паросочетания в двудольном графе»

Выполнил: ст. гр.953503
Басенко К.А.
Проверил: Дугинов О. И.

Минск 2022

Постановка задачи

Пусть дан двудольный граф $G=(V, E)$. Требуется найти максимальное паросочетание.

Описание алгоритма метода

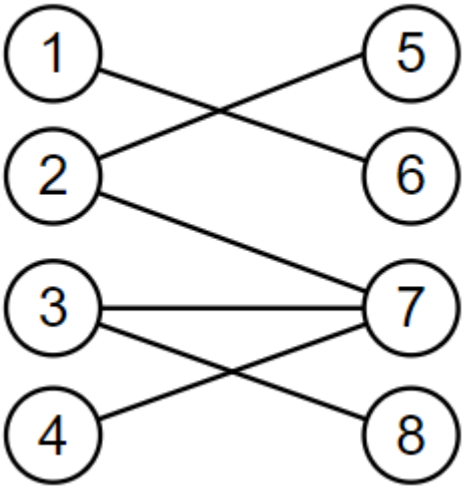
Вход: $G(V, E)$

Выход: максимальное паросочетание M

1. $M=\{\}$
2. Ищем увеличивающую цепь
   a) Строим граф $G'(V,A)$, в котором из второй доли в первую можно идти только по рёбрам паросочетания, а из первой во вторую — по остальным.
   b) Объявляем вершины, не покрытые паросочетанием из первой доли графа $G'(V,A)$, стартовыми.
   c) Объявляем верщины, не покрытые паросочетанием из второй доли графа $G'(V,A)$, финишными.
   d) Ищем путь в графе $G'(V,A)$ из какой-нибудь стартовой вершины в какую-нибудь финишную вершину.
   e) Если такого пути нет, то STOP текущее паросочетание - максимально.
   f) Полученный путь является увеличивающей цепью.
3. Перестраиваем паросочетание: убираем из паросочетания все рёбра, принадлежащие цепи, и, наоборот, добавляем все остальные.
4. Переходим на шаг 2.

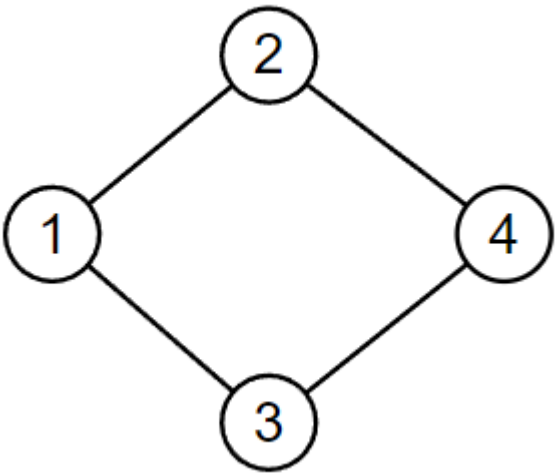
Работа программы

Тест 1




```
Результат:
{
  {1; 6},
  {2; 5},
  {4; 7},
  {3; 8},
}
```

Тест 2




```
Результат:
{
  {1; 2},
  {4; 3},
}
```

Код программы

```
def paint_vertex(V1 ,V2, E, v):
    edges = [i for i in E if i[0] == v or i[1] == v]
    for j in edges:
        new_v = j[1]
        if j[0] != v:
            new_v = j[0]
        if (new_v in V1 and v in V1) or (new_v in V2 and v in V2):
            return False, "Не двудольный граф"
        elif new_v in V1 or new_v in V2:
            pass
        else:
            if v in V1:
                V2.append(new_v)
            else:
                V1.append(new_v)
            r = paint_vertex(V1 ,V2, E, new_v)
            if r is not True:
                return False, "Не двудольный граф"
    return True
def get_separated_vertex(V, E):
    V1 = []
    V2 = []
    for i in range(len(V)):
        if V[i] not in V1 and V[i] not in V2:
            V1.append(V[i])
            r = paint_vertex(V1, V2, E, V[i])
            if r is not True:
                return False, "Не двудольный граф"
    for i in E:
        if i[0] in V2:
            i[0], i[1] = i[1], i[0]
    return True, [V1, V2]
def max_matching(V1, V2, A):
    M = []
    while True:
        mv1 = [i[0] for i in M]
        mv2 = [i[1] for i in M]
        start = [i for i in V1 if i not in mv1]
        finish = [i for i in V2 if i not in mv2]
        if start == [] or finish == []:
            return M
        t = None
        labels = {i: None for i in V1 + V2}
        for i in start:
            labels[i] = 0
        Q = start.copy()
        while len(Q) != 0:
            point = Q.pop(0)
            out_arcs = [i for i in A if i[0] == point]
            for i in out_arcs:
                if labels[i[1]] == None:
```

```
                    labels[i[1]] = i[0]
                    Q.append(i[1])
            for i in Q:
                if i in finish:
                    t = i
                    break
            if t is not None:
                break
            if len(Q) == 0:
                # STOP
                return M
        path = []
        path.append(t)
        while labels[path[len(path) - 1]] != 0:
            point = path[len(path) - 1]
            path.append(labels[point])
        path.reverse()
        for i in range(1, len(path)):
            A.remove([path[i - 1], path[i]])
            A.append([path[i], path[i - 1]])
            if [path[i - 1], path[i]] in M:
                M.remove([path[i - 1], path[i]])
            elif [path[i], path[i - 1]] in M:
                M.remove([path[i], path[i - 1]])
            else:
                M.append([path[i - 1], path[i]])


if __name__=='__main__':
    V = [1, 2, 3, 4, 5]
    A = [
        [1, 2],
        [4, 2],
        [3, 2],
        [3, 5],
    ]
    is_valid, data = get_separated_vertex(V, A)
    V1, V2 = data[0], data[1]
    M = max_matching(V1, V2, A)
    print("Результат:\n{\n", end = '')
    M = [[1, 4], [2, 3]]
    for i in M:
        print(f"  {{{i[0]}; {i[1]}}}, ",)
    print("}")
```