Курсовая работа N 8

- Выбранный алгоритм — *нахождение максимального паросочетания*
- Необходимая теория:
    - Паросочетанием называется произвольное множество рёбер такое, что никакие два ребра не имеют общей вершины
    - Максимальным парсочетанием называется такое паросочетание M в графе G, которое не содержится ни в каком другом паросочетании этого графа, то есть к нему невозможно добавить ни одно ребро, которое бы являлось несмежным ко всем рёбрам паросочетания.
- Идея алгоритма:

Идея алгоритма Эдмондса (Jack Edmonds, 1965 г.) - в **сжатии цветков** (blossom shrinking). Сжатие цветка — это сжатие всего нечётного цикла в одну псевдо-вершину (соответственно, все рёбра, инцидентные вершинам этого цикла, становятся инцидентными псевдо-вершине). Алгоритм Эдмондса ищет в графе все цветки, сжимает их, после чего в графе не остаётся "плохих" циклов нечётной длины, и на таком графе (называемом "поверхностным" (surface) графом) уже можно искать увеличивающую цепь простым обходом в глубину/ширину. После нахождения увеличивающей цепи в поверхностном графе необходимо "развернуть" цветки, восстановив тем самым увеличивающую цепь в исходном графе. Если в графе G существовала увеличивающая цепь, то она существует и в графе $\overline{G}$, полученном после сжатия цветка, и наоборот.

- Описание алгоритма:

a) На ввод подаётся неориентированный граф
b) Найдём нечётный цикл (если такого в данном графе не существует то переходим к пункту 5).
c) Находим наименьшего общего предка.
d) Сожмём этот цикл.
e) Начнём обход в ширину. В процессе обхода в ширину будем строить дерево пути содержащее v, которое будет содержать вершину, которая является паросочетанием для v

- Алгоритм:

Начнём обход в ширину. Находим увеличивающую цепь из свободной вершины root. В процессе обхода рассмотрим текущее ребро (v, to). У нас есть несколько вариантов:

a) Ребро несуществующее. Под этим мы подразумеваем, что v и to принадлежат одной сжатой псевдо-вершине, поэтому в текущем поверхностном графе этого ребра нет. Кроме этого случая, есть ещё один случай: когда ребро (v, to) уже принадлежит текущему паросочетанию; т.к. мы считаем, что вершина v является чётной вершиной, то проход по этому ребру означает в дереве путей подъём к предку вершины v, что недопустимо.

b) Ребро замыкает цикл нечётной длины, т.е. обнаруживается нечётный цикл. В этом случае нужно провести сжатие цветка:

    1) Для u и v находим наименьшего общего предка (НОП). НОП будет являться псевдо-вершиной
    2) Находим те ребра которые находятся в цикле, т.е Проходим от u, v до НОП. Нужно симитировать обход в глубину от псевдо-вершины. Тем самым мы избежим явного объединения списков смежности.
    3) Проставим предков для чётных вершин.

c) Если это «обычное» ребро, то мы делаем обход в ширину. Если мы в вершину to ещё не заходили, и она оказалась ненасыщенной, то мы нашли увеличивающую цепь, заканчивающуюся на вершине to, возвращаем её.

- Блок-схема:

Программа написана на языке C++ с использованием набора утилит Graphviz.

- Инструкция:

Запустить программу. Ввести граф в программе. Формат ввода: Сначала нужно ввести количество рёбер в графе. Далее ввести сами ребра u v, где u, v вершины графа. Выводом программы будет изображение под названием “graph.jpg”, где красные рёбра будут входить в максимальное паросочетание. Изображение будет находиться в той папке, где храниться программа.

- Асимптотическая сложность.

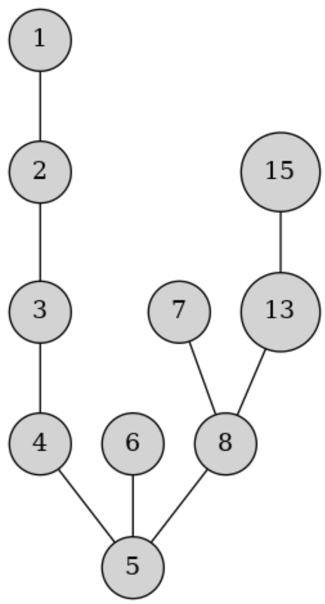
Всего имеется n итераций, на каждой из которых выполняется обход в ширину за $O(m)$, кроме того, могут происходить операции сжатия цветков — их может быть $O(n)$. Сжимаем цветок за O(n). Таким образом, общая асимптотика алгоритма составит $O(n \cdot (m + n^2)) = O(n^3)$.

- Тестовые примеры. Скриншоты программ:

Тест 1

Введём такой граф:
9
1 2
2 3
3 4
4 5
6 5
7 8
8 5
13 8
15 13

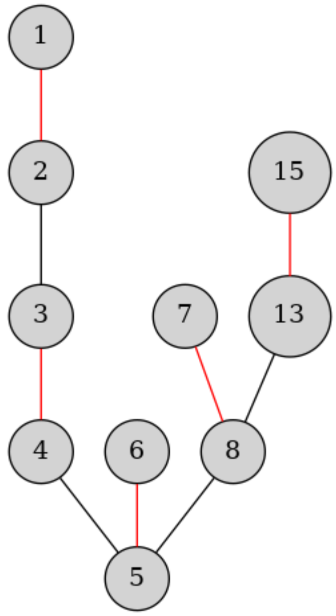
Изначально граф будет выглядеть так:



Найдём максимальное паросочетание вручную:
1. Начнём программу с вершины v_1. Она не в паросочетании, значит запускаем обход в ширину.
2. Перебираем все рёбра из вершины v_1.
Если обнаружили цикл нечётной длины, то сжимаем его
Если пришли в свободную вершину возвращаем значение и останавливаем обход
Если пришли в несвободную вершину, то добавить в очередь смежную ей в паросочетании
Если ничего из вышеперечисленного не произошло, то возвращаем значение -1

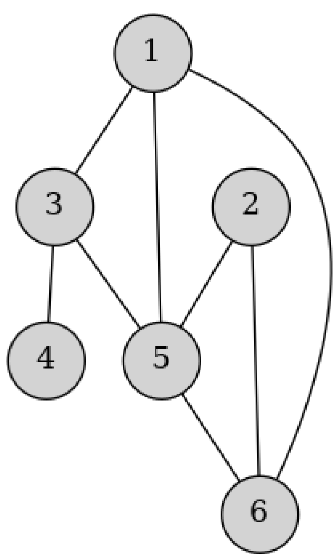
3. Проверяем не равно ли значение -1, если нет, то выполним чередование вдоль пути из v_1. Так как вершина v_2 свободная, то начнём чередование из вершины v_1 в вершину v_2. Итак, вершины v_1 и v_2 будут в паросочетнии.
4. Повторяем шаги 1-3 для каждой вершины.
5. Получаем массив match [2, 1, 4, 3, 6, 5, 8, 7, -1, -1, -1, -1, 15, 13], где i-элемент массива является v_i вершиной, которая имеет паросочетание с $match_i$ вершиной.

Выводом программы будет:

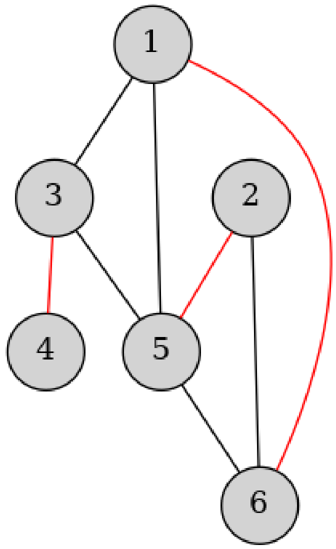


Тест 2
Ввод:

8
1 3
1 5
1 6
2 5
2 6
3 4
3 5
5 6



Вывод:

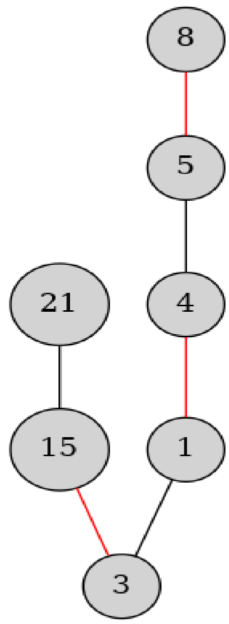


Скриншот:

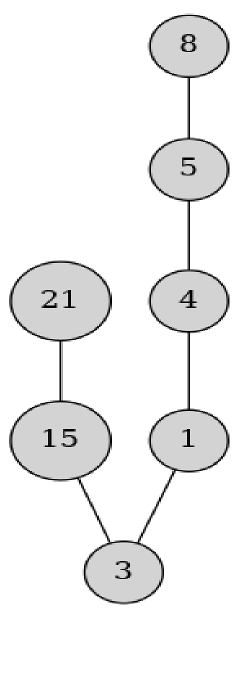
```
Введите количество ребер, а потом ребра графа
8
1 3
1 5
1 6
2 5
2 6
3 4
3 5
5 6

Максимальное паросочетание
1 6
2 5
3 4
```

Тест 3

Ввод:
6
1 3
15 3
4 1
8 5
5 4
21 15



Вывод:



Скриншот:

```
Введите количество ребер, а потом ребра графа
6
1 3
15 3
4 1
8 5
5 4
21 15

Максимальное паросочетание
1 4
3 15
5 8
```

- Прикладная задача

Распределение аудиторий в учебном заведении. Студенты имеют определенный график занятий и должны быть распределены в соответствующих аудиториях. Необходимо назначить аудитории для каждого урока, учитывая ограничения пропускной способности (вершинами в графе являются учебные группы, которые имеют занятия в одно время, а ребра соединяют пары групп, которые имеют общие занятия). Максимальное паросочетание позволит определить оптимальное распределение аудиторий для студентов.