МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Дальневосточный федеральный унирерситет»
(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ
Департамент математического и компьютерного моделирования

ДОКЛАД
о практическом задании по дисциплине АИСД
«Алгоритм сжатия цветков»

Направление подготовки 09.03.03 «Прикладная информатика»
профиль «Прикладная информатика в компьютерном дизайне»

Выполнил студент
гр. Б9121-09.03.03пикд
Яцевич К.А. _____
(ФИО) *(подпись)*

« 2 » Февраля 2023 г.

г.Владивосток
2023

Содержание

Глоссарий

Вершина(узел) — структурная единица графа

Ребро — соединяет вершины графа

Граф — математическая система, объекты которой обладают парными связями

Паросочетание — набор несмежных ребер

Свободная вершина — вершина графа, не покрытая паросочетанием

Дополняющий(увеличивающий) путь — чередующаяся цепь, которая начинается и кончается голыми вершинами.

Цветок(соцветие/бутон) — нечетный цикл графа

Стебель — чередующаяся цепь чётной длины

База — вершина графа, которая пренадлежит стеблю и является частю цикла

Введение

Историческая справка

Алгоритм разработал Джек Эдмондс в 1961 году и опубликовал в 1965 году.

Основной причиной, почему алгоритм сжатия цветков важен, является то, что он дал первое доказательство возможности нахождения наибольшего паросочетания за полиномиальное время. Другой причиной является то, что метод приводит к описанию многогранника линейного программирования для многогранника паросочетаний, что приводит к алгоритму паросочетания минимального веса.

Как уточнил Александр Схрейвер, дальнейшая важность результата следует из факта, что этот многогранник был первым, доказательство целочисленности которого «не просто следовало из тотальной унимодулярности, а его описание было прорывом в комбинаторике многогранников».

Описание алгоритма

Идея алгоритма

Алгоритм сжатия цветков (англ. Blossom algorithm) — это алгоритм в теории графов для построения наибольших паросочетаний на графах.

Если дан граф $G = (V, E)$ общего вида, алгоритм находит паросочетание M такое, что каждая вершина из V инцидентна не более чем одному ребру из M и $|M|$ максимально. Паросочетание строится путем итеративного улучшения начального пустого паросочетания вдоль увеличивающих путей графа.

В отличие от двудольного паросочетания ключевой новой идеей было сжатие нечетного цикла в графе (цветка) в одну вершину с продолжением поиска итеративно по сжатому графу.

Оценка сложности

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 $O(n_1)$. Сжатие соцветий работает за $O(n_2)$, стоит отметить $n_1 \equiv n_2$, то есть общая асимптотика алгоритма составит $O(n(m + n^2)) = O(n^3)$.

Нахождение дополняющего(увеличивающего) пути

Пусть зафиксировано некоторое паросочетание M. Тогда простая цепь $P = (v_1, v_2, \ldots, v_k)$ называется чередующейся цепью, если в ней рёбра по очереди принадлежат — не принадлежат паросочетанию M. Чередующаяся цепь называется увеличивающей, если её первая и последняя вершины не принадлежат паросочетанию. Иными словами, простая цепь P является увеличивающей тогда и только тогда, когда вершина $v_1 \notin M$, ребро $(v_2, v_3) \in M$, ребро $(v_4, v_5) \in M$, ..., ребро $(v_{k-2}, v_{k-1}) \in M$, и вершина $v_k \notin M$.

Мы сможем найти максимальное паросочитание путем инверсии дополняющего пути.

Основная проблема заключается в том, как находить увеличивающий путь. Если в графе имеются циклы нечётной длины, то просто обход в глубину/ширину будет работать не корректно — при попадании в цикл нечётной длины обход может пойти по циклу в неправильном направлении.

Сжатие цветка

Сжатие цветка — это сжатие всего нечётного цикла в одну псевдо-вершину (соответственно, все рёбра, инцидентные вершинам этого цикла, становятся инцидентными псевдо-вершине).

Теорема Эдмондса

Пусть граф $\overline{G}$ был получен из графа G сжатием одного цветка. Тогда в графе $\overline{G}$ существует увеличивающая цепь тогда и только тогда, когда существует увеличивающая цепь в G.

Доказательство.

Обозначим через B цикл цветка, и через $\overline{B}$ соответствующую сжатую вершину. Вначале заметим, что достаточно рассматривать случай, когда база цветка является свободной вершиной (не принадлежащей паросочетанию). Действительно, в противном случае в базе цветка оканчивается чередующийся путь чётной длины, начинающийся в свободной вершине. Прочередовав паросочетание вдоль этого пути, мощность паросочетания не изменится, а база цветка станет свободной вершиной. Итак, при доказательстве можно считать, что база цветка является свободной вершиной.

Пусть путь P является увеличивающим в графе G. Если он не проходит через B, то тогда, очевидно, он будет увеличивающим и в графе $\overline{G}$. Пусть P проходит через B. Тогда можно считать, что путь P представляет собой некоторый путь P_1, не проходящий по вершинам B, плюс некоторый путь P_2, проходящий по вершинам B и, возможно, другим вершинам. Но тогда путь $P_1 + \overline{B}$ будет являться увеличивающим путём в графе $\overline{G}$, что и требовалось доказать.

Общая схема алгоритма

```
void edmonds() {
    for (int i=0; i<n; ++i)
        if (вершина i не в паросочетании) {
            int last_v = find_augment_path (i);
            if (last_v != -1)
                выполнить чередование вдоль пути из i в last_v;
        }
}

int find_augment_path (int root) {
    обход в ширину:
        int v = текущая вершина;
        перебрать все рёбра из v
            если обнаружили цикл нечётной длины, сжать цикл
            если пришли в свободную вершину, возвращаем дополняющий путь
            если пришли в несвободную вершину, то добавить в очередь смежную ей в паросочетании
    return -1;
}
```

Пример работы алгоритма

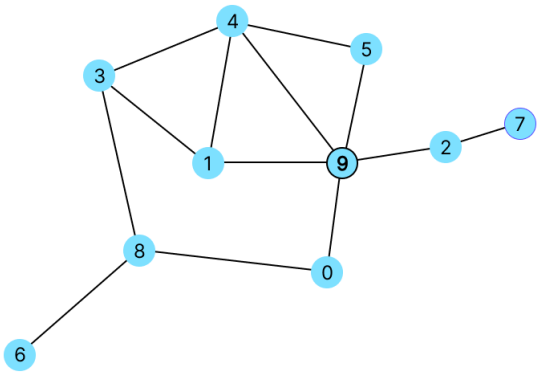

Рис. 1: Исходный граф

Начинаем рассматривать граф со свободной вершины 7. Двигаясь по ребрам обнаруживаем стебель: ребра 7-2 и 2-9. Следовательно, предполагаемый дополняющий путь будет начинаться со свободной вершины 7, ребро 2-9 — паросочетание. Тогда вершина 9 является базой. С помощию BFS идем дальшне по графу: ребро 9-5, ребро 5-4, ребро 4-9 — составляют нечетный цикл — цветок.

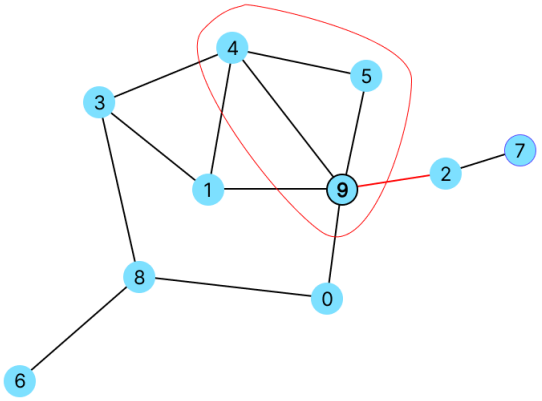

Рис. 2: Нахождение первого цветка

Вершины цикла сжимаем в базу. Получаем следующий граф, который продолжаем обрабатывать по тому же алгоритму. С помощию BFS идем дальше по графу: ребро 9-3, ребро 3-1, ребро 1-9 — составляют нечетный цикл — цветок.

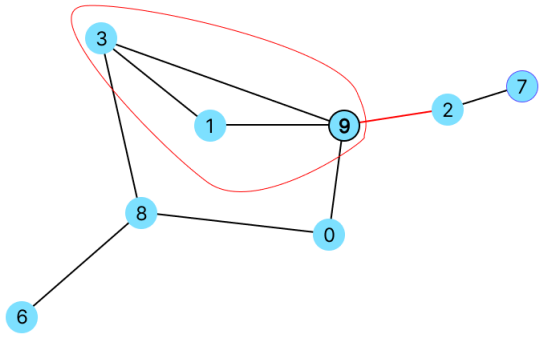

Рис. 3: Нахождение второго цветка

Вершины цикла сжимаем в базу. Новый граф снова обрабатываем. Ребра 9-8 8-0 и 0-9 — составляют нечетный цикл — цветок.

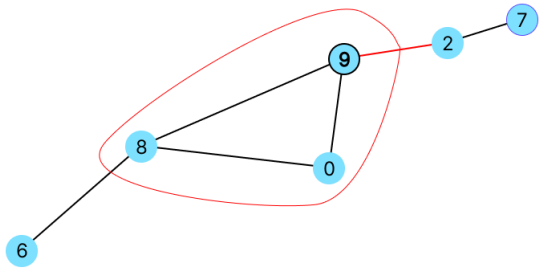

Рис. 4: Нахождение третьего цветка

Сжимаем цветок и продолжаем анализировать граф. После базы 9 идет только одно ребро 9-6, окончание которого — свободная вершина 6. Следовательно можем утверждать, что мы нашли дополняющий путь: 7-2, 2-9, 9-6.

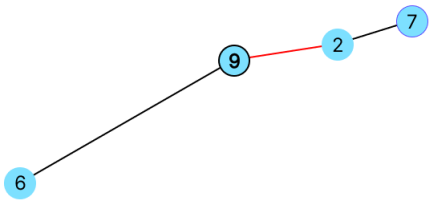

Рис. 5: Дополняющий путь

Обращаемся к теореме Эдмондса:

В графе $\overline{G}$ существует увеличивающая цепь тогда и только тогда, когда существует увеличивающая цепь в G.

Значит мы можем приступить к восстановлению графа путем последовательного возвращения цветков. Для нахождения максимального паросочетания начнём с инверсии дополняющего пути. Начинаем восстановление с последнего сжатия цветка, инвертируя путь. При этом инвертрование пути подразумевает переопределение паросочетания. В цветке мы определяем паросочетание "в обратной последовательности": начиная с 9-0, переходя к 0-8. Так как в дополняющем пути база 9 была соединена с вершиной 6, то дойдя до узла, соединенного с вершиной 6, мы продолжаем переопределять паросочетание в направлении вершины 6.

На данном этапе паросочетание составляет следующие не инцидентные ребра: 7-2, 9-0, 8-6

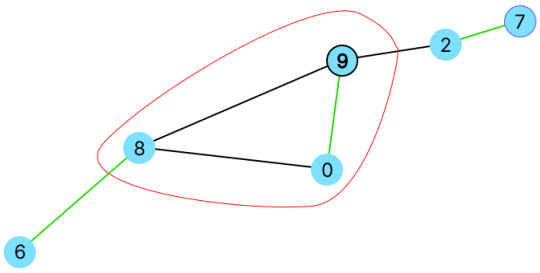

Рис. 6: Восстановление третьего цветка

Запоминаем проставленное паросочетание и продолжаем восстановление графа. Ребро 7-2 уже инвертированно, поэтому переходим сразу к базе: в цветке определяем паросочетание "в обратной последовательности". База уже относится к ребру паросочетания, поэтому мы не можем пометить ребро 9-1. Значит помечаем следующее ребро 1-3. Аналогично с ребром 3-9.

На данном этапе паросочетание составляет следующие не инцидентные ребра: 7-2, 9-0, 8-6, 3-1

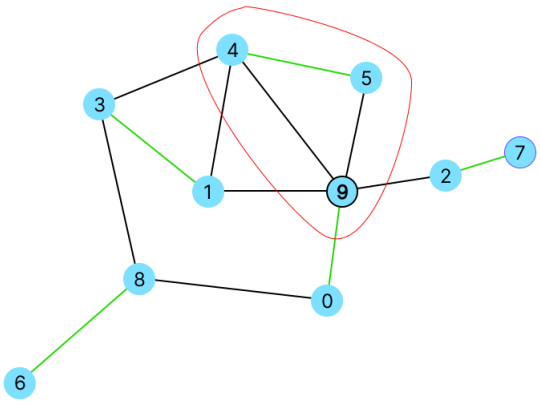

Рис. 7: Восстановление второго цветка

Восстанавливаем последний цветок: аналогично помечаем ребра, начиная с базы. Ребро 9-4 — не помечаем, ребро 4-5 — помечаем, ребро 5-9 — не помечаем.

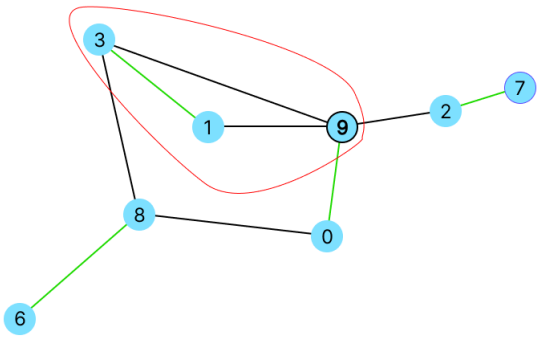

Рис. 8: Восстановление первого цветка

Мы закончили восстановление графа и нашли максимальное паросочетание.

Реализация

Нахождение ближайшего предка

```
int lca(int a, int b) {
        bool used[MAXN] = { 0 };

        for (;;) {
                a = base[a];
                used[a] = true;
                if (match[a] == -1)  break; //
                a = p[match[a]];
        }

        for (;;) {
                b = base[b];
                if (used[b])  return b;
                b = p[match[b]];
        }
}
```

Обозначение дополняющего пути

```
void mark_path(int v, int b, int children) {
    while (base[v] != b) {
        blossom[base[v]] = blossom[base[match[v]]] = true;
        p[v] = children;
        children = match[v];
        v = p[match[v]];
    }
}
```

Поиск дополняющего пути

```
int find_path(int root) {
    memset(used, 0, sizeof used);
    memset(p, -1, sizeof p);
    for (int i = 0; i < MAXN; i++)
        base[i] = i;

    used[root] = true;
    int qh = 0, qt = 0;
    q[qt++] = root;
    while (qh < qt) {
        int v = q[qh++];
        for (size_t i = 0; i < g[v].size(); i++) {
            int to = g[v][i];
            if (base[v] == base[to] || match[v] == to)
                continue;
            if (to == root || match[to] != -1 && p[match[to]] !=
                int curbase = lca(v, to);
                memset(blossom, 0, sizeof blossom);
                mark_path(v, curbase, to);
                mark_path(to, curbase, v);
                for (int i = 0; i < MAXN; i++)
                    if (blossom[base[i]]) {
                        base[i] = curbase;
                        if (!used[i]) {
                            used[i] = true;
                            q[qt++] = i;
                        }
                    }
            }
            else if (p[to] == -1) {
                p[to] = v;
                if (match[to] == -1)
                    return to;
                to = match[to];
                used[to] = true;
                q[qt++] = to;
            }
        }
    }
    return -1;
```

Список литературы

[1] ИТМО, Викиконспекты — Алгоритм вырезания соцветий. 4 сентября 2022.

[2] MAXimal — Алгоритм Эдмондса нахождения наибольшего паросочетания в произвольных графах. 6 декабря 2012.

[3] Энциклопедии Руниверсалис — Алгоритм сжатия цветков. 28 ноября 2022.

[4] Единый центр по исследованию искусственного интеллекта "ЕЦИИИ— Алгоритм вырезания соцветий и сжатия цветков.