Государственное образовательное учреждение высшего профессионального образования
“Московский государственный технический университет имени Н.Э.Баумана”

Дисциплина: Анализ алгоритмов

Рубежный контроль №1

Нахождение клик наиболее эффективным способом

Студент группы ИУ7-55Б,
Руднев К. К.,

Преподаватель,
Волкова Л. Л.,
Строганов Ю. В.

2019 г.

1 Аналитическая часть

В рамках раздела будет дано аналитическое описание алгоритма Брона-Кербоша для нахождения всех клик в графе.

1.1 Описание алгоритмов

Алгоритм Брона-Кербоша - метод ветвей и границ для поиска всех клик (а также максимальных по включению независимых множеств вершин) неориентированного графа. Разработан голландскими математиками Броном и Кербошем в 1973 году и до сих пор является одним из самых эффективных алгоритмов поиска клик.

Алгоритм использует тот факт, что всякая клика в графе является его максимальным по включению полным подграфом. Начиная с одиночной вершины (образующей полный подграф), алгоритм на каждом шаге пытается увеличить уже построенный полный подграф, добавляя в него вершины из множества кандидатов. Высокая скорость обеспечивается отсечением при переборе вариантов, которые заведомо не приведут к построению клики, для чего используется дополнительное множество, в которое помещаются вершины, которые уже были использованы для увеличения полного подграфа.

Алгоритм можно представить в виде рекурсивной процедуры:
ПРОЦЕДУРА extend (candidates, not):
ПОКА candidates НЕ пусто И not НЕ содержит вершины, СОЕДИНЕННОЙ СО ВСЕМИ вершинами из candidates, ВЫПОЛНЯТЬ:
1 Выбрать вершину v из candidates и добавить её в compsub
2 Формировать new_candidates и new_not, удаляя из candidates и not вершины, не СОЕДИНЕННЫЕ с v
3 ЕСЛИ new_candidates и new_not пусты
4 ТО compsub – клика
5 ИНАЧЕ рекурсивно вызвать extend (new_candidates, new_not)
6 Удалить v из compsub и candidates и поместить в not

2 Конструкторская часть

В дальнейшем на рисунке 1 будет представлена схема расматриваемого алгоритма.

2.1 Разработка алгоритмов

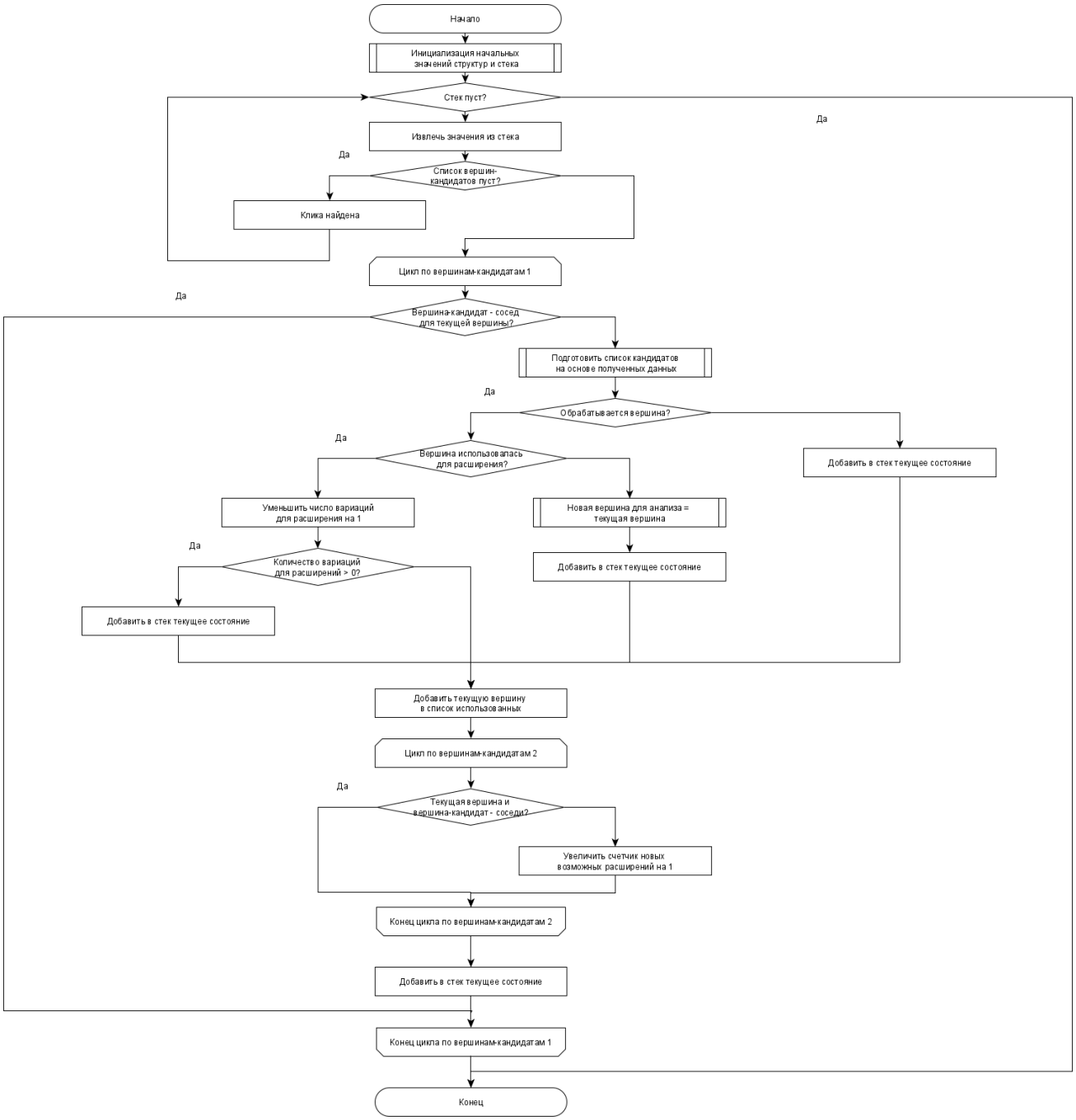

Рис. 1: Алгоритм Брона-Кербоша

3 Технологическая часть

В рамках раздела будут описаны инструментарии разработки, выбор среды, требования к ПО. Также будут предоставлены листинги конкретных реализаций алгоритмов.

3.1. Средства реализации

Для реализации алгоритмов использовался язык программирования Python 3.8.0 и среда разработки PyCharm Community Edition 2019.3.1 by JetBrains. У меня есть определенный опыт работы с данным языком, которого будет достаточно для реализации текущей лабораторной работы, а среда разработки имеет бесплатную комьюнити версию и удобный интерфейс, упрощающий разработку приложения/скрипта.

3.2. Требования к программному обеспечению

На вход программа должна получать граф, на выход отображать все существующие в этом графе клике.

3.3. Листинг кода

Листинг 1: Листинг метода поиска всех существующих клик в графе

```
    def find_all_cliques(self):
        Cliques = []
        Stack = []
        nd = None
        disc_num = len(self.Nodes)
        search_node = (set(), set(self.Nodes), set(), nd, disc_num)
        Stack.append(search_node)
        while len(Stack) != 0:
            (c_compsub, c_candidates, c_not, c_nd, c_disc_num) = Stack.pop()
            if not len(c_candidates) and c_compsub not in Cliques:
                    Cliques.append(c_compsub)
                    continue
            for u in list(c_candidates):
                    if (c_nd is None) or (not self.are_adjacent(u, c_nd)):
                            c_candidates.remove(u)
                            Nu = self.get_node_neighbors(u)
                            new_compsub = set(c_compsub)
                            new_compsub.add(u)
                            new_candidates = set(c_candidates.intersection(Nu))
                            new_not = set(c_not.intersection(Nu))
                            if c_nd is not None:
                                    if c_nd in new_not:
                                            new_disc_num = c_disc_num - 1
                                            if new_disc_num > 0:
                                                    new_search_node = (new_compsub,
                                                        new_candidates, new_not, c_nd,
                                                        new_disc_num)
                                                    Stack.append(new_search_node)
                                            else:
                                                    new_disc_num = len(self.Nodes)
                                                    new_nd = c_nd
```

```
                        for cand_nd in new_not:
                            cand_disc_num = len(new_candidates) - len(
                                new_candidates.intersection(self.
                                get_node_neighbors(cand_nd)))
                            if cand_disc_num < new_disc_num:
                                new_disc_num = cand_disc_num
                                new_nd = cand_nd
                        new_search_node = (new_compsub, new_candidates,
                            new_not, new_nd, new_disc_num)
                        Stack.append(new_search_node)
            else:
                new_search_node = (new_compsub, new_candidates, new_not, c_nd
                    , c_disc_num)
                Stack.append(new_search_node)
            c_not.add(u)
            new_disc_num = 0
            for x in c_candidates:
                if not self.are_adjacent(x, u):
                    new_disc_num += 1
            if (new_disc_num < c_disc_num) and (new_disc_num > 0):
                new1_search_node = (c_compsub, c_candidates, c_not, u,
                    new_disc_num)
                Stack.append(new1_search_node)
            else:
                new1_search_node = (c_compsub, c_candidates, c_not, c_nd,
                    c_disc_num)
                Stack.append(new1_search_node)
    while set() in Cliques:
        Cliques.pop(Cliques.index(set()))
    return Cliques
```

Заключение

Выполнен рубежный контроль по поиску клик эффективным способом. Изучен и реализован алгоритм Брона-Кербоша.