Задача раскраски графов

**Задача раскраски графов**

Задан неориентированный граф  с  вершинами.

Раскрасить вершины  в  цветов – это значит найти такое отображение , что  ребра  выполняется  (т.е. смежные вершины имеют разные цвета).

Граф называется -раскрашиваемым, если для него существует хотя бы одна раскраска в  цветов.

Характеристикой любого графа  является его хроматическое число  – минимальное число цветов, необходимое для раскраски . Обычно требуется найти именно минимальную раскраску вершин в  цветов.

**Применение раскраски**: кластеризация, график работ, совместное использование ресурсов.

Точные оценки  существуют только для полного () и пустого графа (все вершины изолированные, ).

**Нижние оценки**:

 – число вершин клики (максимального полного подграфа) ;

.

**Верхняя оценка**: , где  – степени вершин (для неполных графов, не содержащих циклов нечетной длины – на 1 меньше).

Особый случай – **планарные графы** (их можно так уложить на плоскости, что ребра не будут пересекаться).

Доказано, что любой планарный граф можно раскрасить не более, чем в 4 цвета (**задача раскраски карт**).

Для раскраски  вершин можно использовать  красок. Тогда **общее число потенциальных вариантов раскраски** составляет , но многие из них могут быть недопустимыми, неоптимальными или инвариантными с точностью до перекраски (пример для ).

Пусть вершины графа красятся последовательно (1,2,3,…).

Предположим, что для раскраски первых  вершин потребовалось  цветов и было проверено  вариантов раскраски (, , при ).

()-я вершина может быть покрашена в цвет  или , поэтому всего возможно  вариантов раскраски первых () вершин графа.

Значение  может достигать , поэтому в наихудшем случае **общее число проверяемых вариантов** раскраски может быть равно 

Поиск минимальной раскраски по методу ветвей и границ:

1. Каким-то образом определяется начальная раскраска с числом цветов  (например, , вершина  красится в цвет ). Далее эта раскраска считается текущей минимальной.

2. Вершина 1 красится в цвет 1 (начало формирования текущей раскраски).

3. Пусть в текущей раскраске первые  вершин уже покрашены в  цветов. Для вершины  проверяется возможность ее покраски в цвета , и для каждого допустимого варианта раскраска продолжается рекурсивно.

4. Если в текущейраскраске покрашено  вершин и использовано  цветов, то поиск продолжается, только если нужны все решения.

5. Текущаяраскраска  вершин, содержащая  цветов, становится текущей минимальной.

В приведенном ниже алгоритме используются переменные (для связи с предыдущим описанием используем нумерацию вершин и цветов от 1):

**Pmin[1…n]** – текущая **минимальная** раскраска,

**zmin** – число **различных** цветов в **Pmin**,

**Pcur[1…n]** – текущая раскраска,

**zmax** – максимальное число цветов, используемых на текущем шаге,

**zcur** – число **различных** цветов в **Pcur** на текущем шаге,

**С** – матрица смежности.

**Формирование начальной раскраски, первый вызов рекурсивной функции paint**:

**for (zmin = n, k = 1; k <= n; k++) Pmin[k] = k;**

**Pcur[1] = 1;**

**paint(1, 1);**

Рекурсивная функция раскраски ( вершин уже раскрашены в  цветов)

**void paint(k, z) {**

**zmax = min(z+1, zmin-1);** // макс.число цветов

**for (col=1; col<=zmax; col++) {** // по цветам

**for (j=1; j<=k; j++)** // проверка смежных

**if (C[j][k+1]==1 && Pcur[j]==col) break;**

**if (j > k) {** // красим k+1 в цвет col

**Pcur[k+1] = col; zcur = max(col,z);**

**if (k+1 < n) paint(k+1, zcur);**

**else if (zcur < zmin)** // мин.раскраска

**for (zmin=zcur, j=1; j<=n; j++)**

**Pmin[j] = Pcur[j];**

**}**

**}**

**}**

**Приближенные алгоритмы, основанные на степенях вершин**

**Обозначения**:

 – множество вершин, смежных с ;

 – одношаговая степень вершины ;

 – множество вершин, смежных с вершинами  и не входящих в ;

 – двушаговая степень вершины .

**Основная идея для одношаговых степеней**:

1. Пусть в исходном графе несколько попарно несмежных вершин покрашены в цвет 1. Тогда данные вершины вместе с инцидентными ребрами можно исключить и раскрашивать оставшийся граф, начиная с цвета 2.

2. При покраске очередной вершины желательно исключать как можно больше ребер, чтобы для оставшейся части графа требовалось меньше различных красок.

**Алгоритм** **раскраски**, основанный на одношаговых степенях:

1. Вершины графа сортируются в порядке убывания .

2. Вершина с максимальным  красится в цвет 1.

3. Все непокрашенные вершины просматриваются в порядке убывания  и те из них, которые не смежны с уже покрашенными в цвет 1, красятся в этот цвет.

4. Покрашенные вершины исключаются из рассмотрения; оставшиеся красятся аналогично в цвета 2, 3,…, пока все вершины не будут покрашены.

Возможны 2 варианта алгоритма: с пересортировкой  после покраски в очередной цвет или без нее.

Трудоемкость в наихудшем составляет , для расчета  требуется  ЭШ.

**Основная идея для двушаговых степеней**:

1. Вершину  и попарно несмежные вершины из  можно красить в один цвет.

2. Выбирая для покраски вершину с максимальным значением  можно исключить из дальнейшего рассмотрения большее число вершин и инцидентных им ребер.

**Алгоритм** **раскраски**, основанный на двушаговых степенях, полностью аналогичен предыдущему алгоритму (в описании нужно только заменить  на ). Алгоритм имеет те же два варианта и оценку трудоемкости.

Иллюстрация работы алгоритма для одношаговых степеней:

6

4

4

1

1

2

3

3

2

3

3

2

**Приближенные алгоритмы, основанные на склеивании вершин**

**Склеивание несмежных вершин**  и  – это объединение  и  в одну вершину, инцидентную объединению множеств ребер, инцидентных  и .

**Идея алгоритма**:

1. Выбрать некоторую вершину , не смежную со всеми остальными вершинами графа (не звезду).

2. Последовательно подклеивать к  подходящие для этого вершины, пока  не станет звездой.

3. Повторить предыдущие шаги для всех остальных вершин, не являющихся звездами.

4. Раскрасить полученный полный граф.

Примеры склеивания пар вершин:

6

8

10

12

1,11

5

4

3

2

7

9

6

8

10

11

12

1

5

4

3

2

7

9

6

8

10,2

11

12

1

5

4

3

7

9

6

8,3

10

11

12

1

5

4

2

7

9

Если склеиваются вершины  и , и существуют ребра  и , то общее число ребер в преобразованном графе будет меньше, чем в исходном (исчезновение ребер при наличии разделяющей точки).

При этом , 

=>

Вершины, подклеиваемые к , нужно выбирать из .

Чем меньше ребер будет иметь преобразованный граф, тем, возможно, позднее будет получен полный граф, и число цветов в раскраске окажется меньше.

**Определение**: 2 вершины называются соцветными, если  такая точная (минимальная) раскраска, в которой данные вершины имеют одинаковый цвет.

При раскраске графа нужно склеивать соцветные вершины, однако точного критерия соцветности не существует. Поэтому на практике используются дополнительные теоремы и основанные на них эвристики.

**Лемма**: в любом неполном графе  по крайней мере 2 соцветные вершины.

**Теорема**:  графа с хроматическим числом  существует последовательность попарных склеиваний, приводящих к ‑полному графу.

**Теорема Кожухина**:  вершины  связного графа , не являющейся звездой, в  имеется по крайней мере одна соцветная вершина (т.е.  минимальная раскраска, в которой  имеет тот же цвет, что и одна из ).

**Доказательство**: пусть имеется некоторая минимальная раскраска, в которой вершина  покрашена в цвет . Тогда возможен один из 3 случаев.

1. Некоторая вершина  также имеет цвет  –  и  являются соцветными по определению.

*β2*

*β1*

*β3*

*α*

*α*

***w***

***v***

2.  для ,  для    красим в .

*β2*

*β1*

*β3*

*α*

*γ*

***w***

***v***

*β2*

*β1*

*β3*

*γ*

*γ*

***w***

***v***



3.  для  и нек-рых   красим  в , а  – в .

*β*

*α*

*α*

*γ*

*γ*

***w***

***v***

*β*

*γ*

*γ*

*α*

*γ*

***w***

***v***



Перекраска не влияет на другие вершины и не изменяет число цветов (т.е. остается минимальной).

Для практического использования предлагаются **3 эвристики**:

1. При обработке вершины  выбирать соцветную в .

2. Брать в качестве соцветной  такую вершину из , которая смежна с максимальным числом вершин из .

3. Выбирать такую пару соцветных вершин  и , для которой условие 2 дает максимум по всему графу.

**Трудоемкость**:

* поиск соцветной вершины – ;
* склеивание пары вершин – ;
* общая по эвристикам –  для 1. и 2.,  для 3.

**Условие оптимального склеивания** (Гладких, Матушевский):

 смежна только с   нужно искать пару для оптимального склеивания, а в ее отсутствие применять 3‑ю эвристику.

Результаты практического использования приближенных алгоритмов раскраски сильно зависят от класса графов.

**Транзитивно-ориентируемые графы**

Неориентированный граф  является транзитивно-ориентируемым, если в  можно так задать ориентацию, что полученный граф станет транзитивным.

Для ТО-графа существует 2 противоположные ориентации.

Транзитивно ориентировать можно не каждый граф, и не каждая ориентация приводит к транзитивному графу:

**Два правила транзитивной ориентации графа** для тройки вершин  (Пнуели, Лемпель, Ивен):

1. Есть **дуга**  и **ребро** , **нет связи**  – ориентировать .

*i*

*j*

*k*

*i*

*j*

*k*



2. Есть **дуга**  и **ребро** , **нет связи**  – ориентировать .

*i*

*j*

*k*

*i*

*j*

*k*



На практике условие «нет связи » означает одно из двух:

* в графе не существует ребра , дуги  или дуги ;
* в графе существует дуга  или дуга , для которой проверены все связи вершин  и , и поэтому данная дуга может быть исключена из дальнейшего рассмотрения.

Без 2-го пункта невозможно сориентировать даже простейший полный граф с 3 вершинами.

Возможны 3 варианта обработки произвольного графа :

1. Завершить работу, если обнаружено, что  – не ТО-граф.

2. Ввести дополнительные ребра, чтобы превратить  в ТО-граф.

3. Ориентировать  по правилам ПЛИ до конца, даже если обнаружено, что  – не ТО-граф.

Приведенный ниже алгоритм использует 3-й вариант.

Если исходный граф является **транзитивно-ориентируемым**, то полученный граф будет **транзитивным**. В противном случае ориентированный граф, по крайней мере, не будет содержать циклов.

В алгоритме рассматриваются **3 типа связей** пар вершин:

1. **Ребро** исходного графа.

2. **Дуга без метки**  – ребро, сориентированное . Для данной дуги пока **не завершена проверка** всех ребер, инцидентных  и .

3. **Дуга с меткой**  – дуга, для которой **завершена проверка** всех ребер, инцидентных  и . При ориентировании оставшихся ребер такая дуга может быть исключена из рассмотрения.

**Алгоритм транзитивной ориентации графа**:

**while (найдено\_ребро(a, b)) {**

**создать\_дугу\_без\_метки(a, b);** // дуга a->b

**while (найдена\_дуга\_без\_метки(i, j)) {**

**for (k  L[j])** // правило ПЛИ-1

**if (ребро(j, k) && нет\_связи(i, k))**

**создать\_дугу\_без\_метки(k, j);**

**for (k  L[i])** // правило ПЛИ-2

**if (ребро(i, k) && нет\_связи(j, k))**

**создать\_дугу\_без\_метки(i, k);**

**отметить\_дугу(i, j);** // (i,j) проверена

**}**

**}**

Дуги без меток нужно обрабатывать в том порядке, в каком они создаются, т.е. для их хранения нужно использовать очередь.

**Трудоемкость** данного алгоритма составляет:

 – при использовании списков смежных вершин ( – максимальная степень вершины),

 – при использовании матрицы смежности.

**Пример ориентирования графа** (дуги без меток – красные, с метками – синие, ребра – черные):

**Раскраска транзитивно-ориентированного графа**

**Теорема**: хроматическое число  для ТО-графа равно максимальной длине пути + 1 (т.е. числу вершин в максимальном пути).

**В доказательстве** используются два свойства транзитивно-ориентированного графа.

**Свойство 1**:

ТО-граф разбивается на подграфы, содержащие ровно одну вершину без входящих дуг и одну вершину без исходящих дуг. Каждая пара подграфов имеет не более одной общей вершины одного или другого типа.

Вершины, не имеющие либо входящих, либо исходящих дуг, существуют всегда, в противном случае в графе должны существовать циклы и, следовательно, петли.

Примеры разбиения на подграфы (красные дуги принадлежат одновременно нескольким подграфам).

Каждый из подграфов соответствует некоторому полному подграфу исходного неориентированного графа .

Вершины максимального полного подграфа  являются также вершинами пути максимальной длины в ТО-графе. Пусть число этих вершин равно . Тогда  (оценка снизу).

**Свойство 2**: если в транзитивно-ориентированном графе удалить вершины, не имеющие исходящих дуг, вместе с входящими в них дугами, то граф останется транзитивным (т.е. опять будет включать некоторые вершины, не имеющие исходящих дуг).

Вершины, не имеющие исходящих дуг, можно красить в один цвет (1). После удаления таких вершин и входящих в них дуг в графе снова появляются аналогичные вершины, которые можно красить в цвет 2 и т.д.

Тогда общее количество цветов, используемых при раскраске, будет равно , т.е. и . **Конец доказательства**.

Трудоемкость алгоритма раскраски составляет .