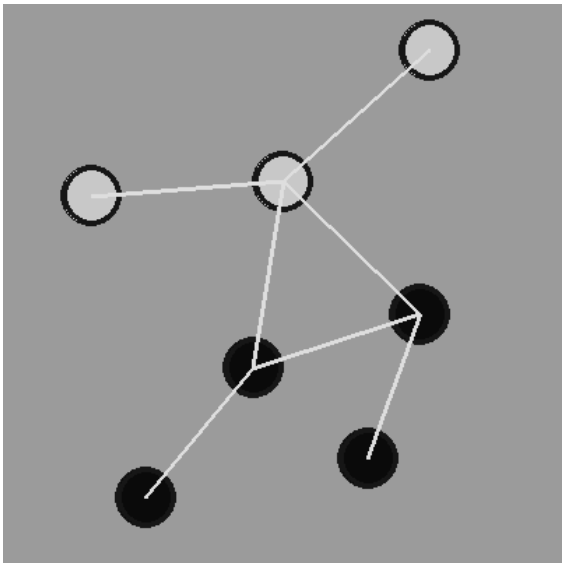
Для решения этой задачи я использовал язык программирования C++. А так же добавил для визуализации и тестирования графов свою старую программу на языке питон и библиотекой pyGames.

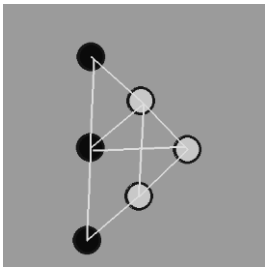
Условие для будущих рассуждений :

- Ребра в графе неориентированные.
- Вершины с черным цветом имеют номер 1, вершины с серым цветом имеют номер 0.
- Друзья для вершины V – это те вершины, которые являются смежными с ней и имеют тот же цвет , что и сама вершина V.
- Враги V для вершины V – это те вершины, которые являются смежными с ней и имеют Отличный от вершины V цвет.
- Количество вершин в графе обозначается буквой N.
- Компонентой называется множество вершин, которые могут быть достижимыми друг из друга по ребрам графа.



1

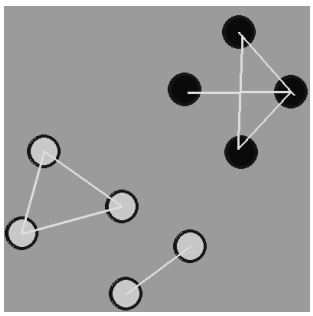
Граф называется стабильным, если в каждой вершине количество врагов не больше количества друзей. Тогда мы можем просто перебрать бит маску для всех вершин и проверить наличие стабильности для графа. Этот способ является сам медленным, ведь работает за O(2^N). Однако работает во всех случаях.



Пример графа, где этот алгоритм хорошо себя покажет ->
Из-за маленького числа вершин, алгоритм способен быстро перебрать все возможные способы расскраски и выдать правильный варинат.

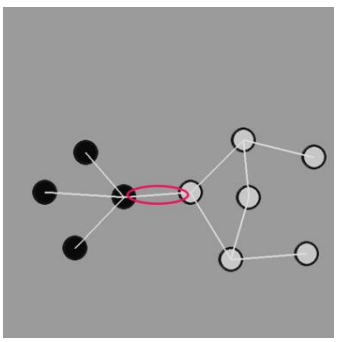
2

Есть несколько конкретных случаев графа, в которых задача решается быстрее.



<– Граф с двумя и более компонентами.
Достаточно простой пример, ведь расскраска одной компоненты никак не влияет на расскраску другой. Поэтому их можно полностью перекрасить. А найти компоненты связности можно за время O(N).

Граф с мостом–>
В таком графе достаточно найти ребро, которое удовлетворяет такому условию

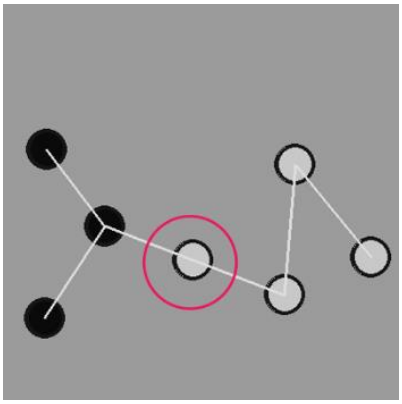


- Если мы уберём ребро, то у нас появится на одну компоненту больше, чем было.
- Ребро должно соединять вершины, которые имеют больше одного ребра.

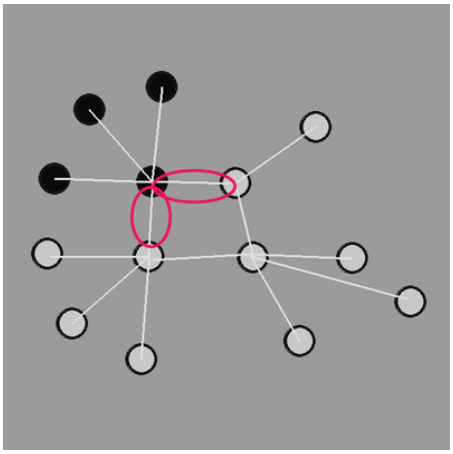
1

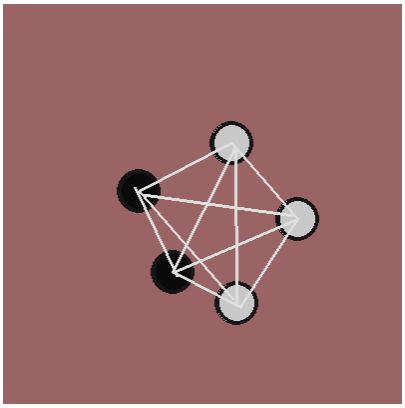
Сложность алгоритма O(N). Так как нам необходимо только найти мосты и проверить выполнение второго условия для них. А так же расскраить в разные цвета обе компоненты по разные стороны моста.

Граф, имеющий точку сочлинения. –>
Точка сочлинения, это та вершина, убрав которую, количество компонент увеличется. В нашем случае, для этой вершины должно выполняться так же условие, что её соседи должны иметь больше одного ребра.
Найдя такую вершину за O(N) мы перекрашиваем её соседние компоненты в разные цвета. Саму же вершину красим в тот цвет, который чаще встречается для смежных ей вершин.



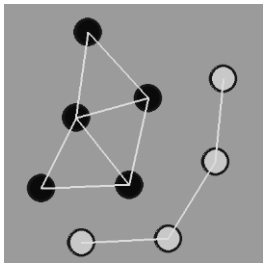
Граф, имеющий 2 и более моста.
Иногда, при большом количестве вершин и маленьком количестве ребёр, эфективно будет перебрать возможные мосты , удалить их и посмотреть не увеличилось ли количество компонент связностей.
Скорость работы O((K^z)*N)), где K – ребер у графа , z – возможных мостов. Для случая –>
сложность будет приблизительно
O(14^2*14 ==2744) VS O(2^14 ==32789)
..Ощутимо!





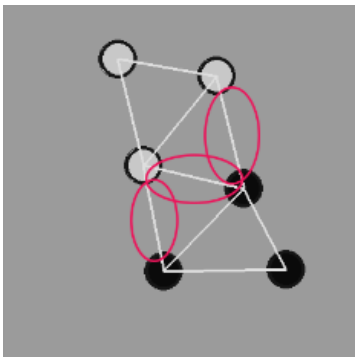
Полный граф.
Полный граф это такой граф, в котором каждая вершина имеет ребро во все остальные вершины. Такой граф никогда не будет стабильным. Так как каждая вершина имеет N-1 рёбер и ей необходимо хотя бы (N-1)/2 с округлением вверх соседей того же цвета, чтобы быть стабильной. Тогда у нас остается (N – (N-1*(N%2))/2 – 1) вершин другого цвета , что меньше половины от N. Тогда первого цвета всегда будет больше, чем второго, а сам второй цвет будет хотеть стать первым из-за отстутствия условия стабильности.

3



Максимальная прибыль в графе с несколькоми компонентами будет равна N*(N-1).

Если в графе одна компонента, то максимальная прибыль равна N*(N-1) –(количество ребер, соединяющие вершины разного цвета)*2



Максимальная суммарная стоимость для этого графа –> (6*(6-1)-3*2)

4

Достаточно много было идей другого решения этой задачи, однако большая часть находила свой ошибочный тест.
Перечисление прошлых,провальных идей:

- Минимальное остовное дерево – можно заметить, что часто встречается решение, где один цвет образует компоненту сильной связности. Проблема в реализации компоратора для сортировки вершин. Так как в некоторых случаях нам надо смотреть на один и более шагов вперёд.
- Если не получилось изнутри, можно попробовать снаружи! Обрубать куски.. но эта идея осталось нереализованной.
- Перебирать дфсом возможную компоненту. Однако такое решение работало даже дольше, чем битмаски.

Поэтому, не решив изначальную задачу, достаточно трудно решить её усложнённую версию.