Докажите, что вершины любого графа *G* можно так упорядочить, что жадный алгоритм раскрасит граф *ровно* в *χ*(*G*) цветов.

Указание. Заметьте, что Вам не нужно пытать указать, как строить такое упорядочение «с нуля»; нужно лишь доказать, что оно существует. Представьте, что у Вас в руках уже есть оптимальная раскраска графа и попробуйте по ней построить нужное упорядочение. Когда Вы придумаете нужный способ упорядочения вершин, не забудьте строго обосновать, что жадный алгоритм сработает на нём оптимально. Заметьте, что жадный алгоритм может построить раскраску, отличающуюся от взятой Вами изначально оптимальной раскраски. Возможно, потребуется провести индукцию *по номеру шага* жадного алгоритма.

**Схема возможного решения.** Предположим, что Добрый Волшебник сообщил нам, как раскрасить вершины графа минимально возможными количеством цветов (дадим им условные номера от первого до *χ*(*G*)-го). Упорядочим вершины графа так: сначала в любом порядке идут вершины первого цвета, затем в любом порядке идут вершины второго цвета, и т.д. Запустим ж.а. на такой раскраске и докажем индукцией по количеству проделанных шагов, что на каждом шаге алгоритм присваивает очередной вершине цвет, номер которого точно не больше, чем номер цвета, присвоенного этой же вершине в раскраске Доброго Волшебника.

Пусть нам уже дана оптимальная раскраска графа G в χ(G)=n цветов.

Рассмотрим следующий алгоритм упорядочивания графа G:

1. На первом шаге перебираем в любом порядке вершины, покрашенные в первый цвет.

2. Затем в любом порядке перебираем вершины, покрашенные во второй цвет.

...

n. Затем в любом порядке перебираем вершины, покрашенные во n-й цвет.

Докажем теперь, что на таком упорядочивании жадный алгоритм раскрасит граф ровно в n цветов.

1. Жадный алгоритм раскрасит все вершины, выбранные на первом шаге, в 1й цвет, т.к. никакие из них не инциденты одному ребру (они изначально были окрашены в один цвет).

2. Вершины, выбранные на втором шаге, жадный алгоритм может покрасить максимум в 2 цвета: если какие-то из них не инциденты одним ребрам с вершинами первого шага, они будут покрашены в первый цвет, иначе во второй цвет.

3. Аналогично вершины, выбранные на k-м шаге, жадный алгоритм может покрасить максимум в k цветов: если какие-то из них не инциденты одним ребрам с вершинами 1го шага, они будут покрашены в первый цвет, инциденты первого, но не инцидентны вершинам 2го шага - во 2й цвет ... инциденты вершинам с первого по (k-1)й шаг в k-й цвет.

Следовательно, вершины, выбранные на n-м шаге, жадный алгоритм может покрасить максимум в n цветов. Таким образом всего будет использовано максимум n цветов, но мы знаем, что n - это минимальное кол-во цветов, значит меньше их быть использовано не может, следовательно жадный алгоритм использует ровно n цветов для раскраски по нашему упорядочиванию. ЧТД.

Допустим известна оптимальная раскраска графа, тогда множество вершин графа разобьётся на χ(G) независимых подмножеств(цветов) Ai, i = 1, ... , χ(G), обозначим следующим образом вершины принадлежащие конкретной раскраске: aij ∈ Ai, j=1,..,si. Рассмотрим следующее упорядочивание: a11,...,a1s1,a21,...,aχ1,...,aχχs. Запишем цикл прохода в графа через равносильную ему пару циклов:   
for (i = 1, χ, i++)   
 for (j = 1, sχ, j++)   
 badColors = { color(aqp) |   
 (q < j and p == i) or (p < i and ∀ j) and apq aij ∈ E (G) }   
 color() = min(ℕ \ badColors)

Поскольку любые две вершины с одинаковым верхним индексом принадлежат одному цвету из исходной раскраски, то они, очевидно, независимы и   
 {color(aqp) | (p < i and ∀ j) and apq aij ∈ E (G)} = ∅

, тогда badColors = { color(aqp) | (p < i and ∀ j) and apq aij ∈ E (G) };   
Т.е. множество "плохих" цветов растет только со сменой верхнего индекса, причем ровно на единицу:

badColors ⊆ {1,...,i−1}. Таким образом для любой вершины aχ j, badColors ⊆ {1,...,χ−1}, что значит она точно может быть покрашена цветом χ. Т.е. жадный алгоритм даёт оптимальную раскраску.

Пусть у нас есть оптимальная раскраска графа в χ(G)цветов. Эта раскраска сопоставляет каждой вершине некоторый цвет, обозначим его натуральным числом от 1 до χ(G). Упорядочим вершины так: сначала добавляем в список все вершины цвета 1 в любом порядке, затем все вершины цвета 2, и т.д.

Покажем по индукции, что когда жадный алгоритм обходит наш список - он построит правильную раскраску. Т.к. изначальная раскраска была оптимальной - покрасить граф в меньшее количество цветов он не сможет. Т.е. надо показать, что цветов не станет больше Будем считать n-м шагом обход элементов, которые были цвета n в изначальной раскраске.

1) База индукции тривиальна - на первом шаге алгоритм обходит вершины, которые в изначальной раскраске были цвета 1. Это означает, что среди них нет ни одной пары соседей (иначе они не могли бы быть одного цвета), и жадный алгоритм покрасит их все в цвет 1.

2) Пусть до n-го шага жадный алгоритм задействует не более n цветов. Покажем, что тогда на (n+1) шаге он задействует не более (n+1) цвета.

На (n+1) шаге алгоритм обходит вершины и помечает их минимальным цветом, которого нет у соседних уже просмотренных вершин. Но у них могут быть только цвета от 1 до максимум n, т.к. до этого шага алгоритм использовал  не более n цветов. Значит, вершины с (n+1) шага могут быть покрашены в цвет не более (n+1). Что и требовалось доказать.

Есть простой граф G с мн-вом вершин V и мн-вом рёбер E. Допустим известна оптимальная раскраска графа, тогда множество вершин графа разобьётся на χ(G) независимых подмножеств Vk, k = 1, ... , χ(G), упорядоченные вершины по оптимальной раскраске обозначим так: vk,i ∈ Vk, i = 1, ... , |Vk|.

Поместив наши подмножества так же упорядоченно в стек Q запустим ж.а.:   
for k := 1 to χ(G):   
 Vk := получить\_подмножество(Q)   
 for i := 1 to |Vk|:   
 badColors := { color(vk,j) | j < i и vk,ivj ∈ E }   
 color(vk,i) := min(ℕ \ badColors)

Покажем по индукции, что когда жадный алгоритм обходя наш список построит правильную раскраску. Изначальная раскраска была оптимальной посему покрасить граф в меньшее количество цветов невозможно. Необходимо доказать, что цветов не станет больше Будем считать n-м шагом обход элементов, которые были цвета n в изначальной раскраске.

*База индукции*: при k = 1 алгоритм обходит вершины из V1, в этом пожмн-ве все вершины независимы т.к. изначальной раскраске были одного цвета и ж.а. покрасит их все в цвет 1.

*Индукционный переход*:   
Пусть до n-го шага ж.а. задействует не более n цветов. Покажем, что тогда на (n+1) шаге он задействует не более (n+1) цвета.

На (n+1) шаге алгоритм обходит вершины и помечает их минимальным цветом, которого нет у соседних уже просмотренных вершин. Но у них могут быть только цвета от 1 до максимум n, т.к. до этого шага алгоритм использовал  не более n цветов. Значит, вершины с (n+1) шага могут быть покрашены в цвет не более (n+1). Ч.т.д.   
   
.