Задача о независимом множестве

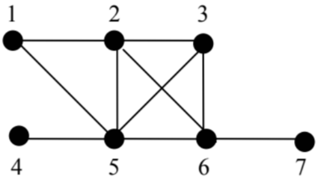
Графы и алгоритмы

13 ноября 2020 г.

Что нужно сделать?

Пусть G— произвольный граф. Необходимо найти наибольшее независимое множество вершин графа G, т.е такое наибольшее подмножество $A(G)$ вершин графа, что любые две вершины $v, u \in A(G)$ не смежны в G

В качестве графа-подопытного возьмём такой граф:



Стратегия перебора

- Выберем в графе произвольную вершину a.
  Возьмем $G_1 = G - a$, а через G_2 обозначим граф, получающийся из G удалением всех вершин, смежных с a.
  В нашем случае будет такая картина

Стратегия перебора

- Пусть X - произвольное независимое множество вершин графа G. Возможны два случая
  1. Если $a \in X$, то в X нет вершин, смежных с a. Тогда X будет независимым множеством и для графа G_2. Если a - не изолированная, то в G_2 вершин меньше, чем в исходном G.
  2. Если $a \notin X$, то X - независимое множество для графа G_1. И в G_1 тоже меньше (на одну) вершин, чем в G
- Зная это, мы можем свести задачу к меньшей размерности, ведь имеется реккурентное соотношение для числа независимости:

$$\alpha(G) = max(\alpha(G_1), \alpha(G_2))$$

  И, соответственно, имеем реккурентный алгоритм для нахождения наибольшего независимого множества графа G.

Деревянная интерпретация

Теперь, имея реккурентный алгоритм, можно представить его работу при помощи некоторого дерева D. Вершины дерева будем называть узлами, дабы не запутаться.

- Каждому узлу D будут соответсвовать некоторый граф H и его неизолированная вершина x
- Внутренний узел - узел, не являющийся листом.
- Каждый внутренний узел имеет двух сыновей. Левому соотвутствуют $H - x$ и его произвольная вершина, а правому - граф, получающийся из H удалением вершин, смежных с x, и еще какая-то вершина.

Сложность

Теперь нужно обойти дерево в том или ином порядке, запоминая на каждом шаге небольшую часть информации об устройстве этого дерева - так мы завершим работу алгоритма. Например - поиск в глубину.

- Сложность зависит от выбора активной вершины x в каждом узле дерева.
- Листьев в дереве будет не меньше, чем максимальных независимых множеств у графа.

Сложность

Проблема такого алгоритма состоит в том, что на каждом шаге получается разветвление на два случая, и, к тому же, неизвестно, какую активную вершину x выбрать, чтобы закончить работу быстрее. Появляется эвристическая идея рассмотреть только один путь от корня до листа в дереве вариантов. Остается надеяться, что в результате получится достаточно большое независимое множество.

Жадные алгоритмы

Допустим, мы решили каждый раз удалять активную вершину, пока не получим граф без рёбер, то есть нужное нам независимое множество. Хотелось бы, чтобы в таком графе было как можно больше вершин.
Удалим меньше вершин - больше останется (логично). Кажется, что стоит на каждом шаге выбирать такую активную вершину, чтобы при её удалении исчезло наибольшее число рёбер. То есть на каждом шаге можно выбирать вершину с наибольшей степенью в качестве активной.

- Оптимальный выбор на каждом шаге не гарантирует оптимального решения.

Жадные алгоритмы

Другая крайность - удалять окрестность активной вершины на каждом шаге до тех пор, пока не получится независимое множество. Из тех же соображений,

- На каждом шаге нужно выбирать вершину наименьшей степени.

Жадные алгоритмы

Имеется немало графов, для которых жадные алгоритмы дают решения, близкие к оптимальным Однако есть примеры графов, для которых они приводят к довольно плохим решениям.

Пример - граф $G_k = 2K_k \circ O_k$

G_k разбивается на три части мощности k каждая: $V_G = A \cup B_1 \cup B_2$, где A - независимое множество, а B_1, B_2 - клики.

$deg(v \in A) = 2k, deg(u \in B_i) = 2k - 1$

Первый жадный алгоритм (удаляющий активные вершины) выдаст независимое множество, состоящее из двух вершин.

Второй (удаляющий окрестности активных вершин) возьмет в качестве активной вершины одну вершину из $B_1 \cup B_2$ и удалит всю ее окрестность. Получится независимое множество из двух вершин.