Задача A

Условие

Найти значения переменных в формуле 3SAT, при которых наибольшее число скобок принимают истинное значение.

Доказательство NP

Допустим, что всего n переменных и m скобок.

1. Переформулируем задачу

   Можно ли подобрать такие значения переменных, чтобы истинно было хотя бы K скобок?

2. Сертификат: последовательность $(k, x_1, x_2, ..., x_n)$

3. Верификатор: Посчитать количество истинных скобок. Сравнить с k.

Доказательство NPH

Надо доказать, что $3SAT \leq_p MAX3SAT$. Допустим мы умеем решать $MAX3SAT$, то есть мы нашли такой набор $X = (x_1, x_2, ..., x_n)$ при котором максимальное число скобок (обозначим k) истинно. Этот набор X и будет решением $3SAT$, если $k =$ числу скобок. $k \neq$ числу скобок, то $3SAT$ не разрешима.

Задача G

Условие

Дана матрица A размера $M \times N$ из целых чисел, вектор B размера M из целых чисел. Найти бинарный вектор x, такой что $Ax \leq B$.

Доказательство NP

1. Сертификат: Бинарная последовательность чисел размера M. Очевидно, что размер сертификата равен M, то есть он полиномиален относительно размера самой задачи.

2. Верификатор: Непосредственно производит умножение и поэлементное сравнение. Самый очевидный алгоритм умножения матриц имеет сложность $O(n^3), n = max(M, N)$, сравнение выполняется за $O(M)$. А значит, верификатор работает за полиномиальное время от входа.

Доказательство NPH

Задача K

Условие

Дан взвешенный ориентированный граф $G = (V, E)$, найти в нем гамильтонов путь.

Доказательство NP

Допустим, всего N вершин.

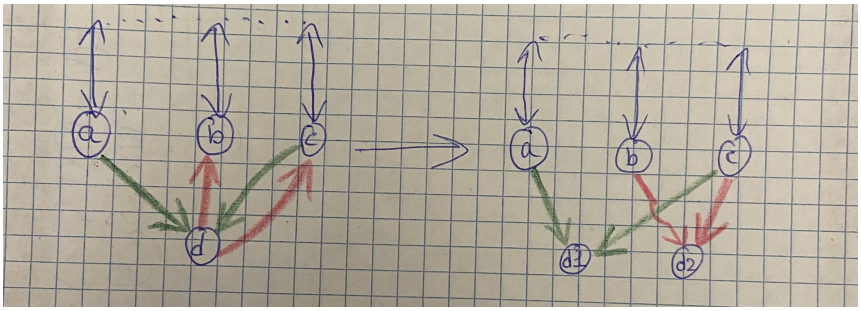
1. Сертификат: Последовательность вершин размера N. Очевидно, что размер сертификата равен N, то есть он полиномиален относительно размера самой задачи.

2. Верификатор: Проверяет, посещены ли все вершины(достаточно проверить, что в сертификате все элементы уникальны) и существуют ли все ребра между соседними элементами в последовательности.

Доказательство NPH

Для доказательства этого факта сведем задачу о поиске гамильтонова цикла (далее - ГЦ) в ориентированном графе к задаче о поиске гамильнового пути (далее - ГП) в ориентированном графе.

Пусть дан граф. Выберем произвольную вершину графа d и раздвоим ее на $d1$ и $d2$, так, что все входящие дуги в d теперь будут входить в $d2$, а все исходящие дуги из d теперь будут выходить из $d1$.



1. В одну сторону.

   Утверждается, что если в исходном графе был ориентированный ГЦ, то в полученном будет ориентированный ГП. Докажем это утверждение. Допустим, что в исходном графе был ГЦ. Без ограничения общности будем считать, что он начинался и заканчивался в вершине d. Тогда после такого преобразования будет существовать ГП и он будет начинаться в $d2$ и заканчиваться в $d1$.

2. В обратную сторону.

   Если в полученном графе будет ГП, то есть будет путь соединяющий все вершины, тогда он будет посещать вершины $d1$ и $d2$. Это значит, что он будет начинаться в $d2$ и заканчиваться в $d1$. Таким образом, если объединить эти вершины, то в полученном графе будет существовать ГЦ.

Использована информация из: Ссылка на источник .

Задача O

Условие

Дан набор задач A, для каждой есть время, за которое ее решает один процессор t_i, M процессоров и общий дедлайн D. Можно ли разбить все задачи на непересекающиеся множества задач A_j, такие что $\bigcup_{j=1}^{n} A_j = A$ и сумма t_i по каждому $A_j \leq D$. Формально: $\forall j \ \sum_{i \in A_j} t_i \leq D$

Доказательство NP

Допустим, что всего задач N.

1. Сертификат: последовательность $k_1, k_2, ..., k_N : \forall i \ k_i \in \{1, 2, ..., M\}$
2. Верификатор: Перебирает все процессоры, считает время работы на нем (суммирует те t_j, у которых k_j равно номеру этого процессора), если оно больше D хоть в одному случае, то возвращает ЛОЖЬ, иначе ИСТИНУ.

Доказательство NPH

Сведем задачу о рюкзаке к этой задаче. В задаче о рюкзаке у каждого объекта есть вес w_j и стоимость c_j . В нашей задаче $w_j = c_j = t_j$. Тогда

Задача R

Условие

Рассмотрим такой алгоритм построения максимальной клики: будем каждый раз удалять из графа вершину минимальной степени, пока не получим полный граф. Докажите или опровергните, что такой алгоритм дает решение, не более чем в X раз отличающееся от оптимального.

Решение

Утверждение. Если клика размера N, то все входящие в нее вершины имеет степень $= N - 1$. Доказательство очевидно.

Утверждение. Для любого n можно построить граф, в котором степень хотя бы одной вершины будет n, а всех остальных вершин будет $\geq n$, а размер максимальной клики будет 2.
Таким графом будет n-мерный куб. Степень каждой вершины в него входящий будет равна n, так как хроматическое число n-мерного куба равно 2. Если хроматическое число графа равно 2, то в нем по определению не существует клик размера больше 2.

Утверждение. Для любого N, существует граф, в котором есть клика размера N и используя этот алгоритм будет получаться клика размера 2.
Возьмем клику C размера $N+1$. Возьмем граф G, в котором существует вершина v, степень которой N, а степень всех остальных вершин $\geq N$. Соединим C и G любым ребром, главное, чтобы оно не касалось вершины v. В получившемся графе минимальная степень вершины N и существует вершина степени N, входившая в граф C. Допустим алгоритм начнет удаление именно с этой вершины. Затем он будет постоянно удалять все вершины графа C, так как при удалении вершины графа C, степень всех вершин графа C уменьшиться на 1. Таким образом граф C будет полностью удален и задача сведется к нахождению максимальной клики в графе G. А у этого графа максимальная клика имеет размер 2 по определению.

Вывод

Последнее утверждение опровергает существование константы X.

Задача U

Условие

Предложить $\frac{1}{2}$-приближенный алгоритм для решения следующей задачи. Требуется разбить множество вершин неориентированного графа $G = (V, E)$ на два непересекающихся множества S и T таким образом чтобы число ребер $(u, v) : u \in S$ и $v \in T$ было максимально.

Решение

Начинаем со случайного разбиения. На каждой итерации алгоритма из одного множества переносим вершину в другое множество, таким образом, что бы решение улучшалось. Как только решение перестает улучшаться, алгоритм останавливается. Во время остановки алгоритма верно, что для каждой вершины половина (или больше) ребер ведут в другое множество. Если бы это было не так, то мы могли бы перенести эту вершину и улучшить решение. Это значит, что как минимум половина ребер ведут из одного множества в другое, а значит это $\frac{1}{2}$-приближенный алгоритм.

Использована информация из: Ссылка на источник .