Махаев Анатолий BDINRT

B. Найти есть ли у формулы 3SAT решение, в котором хоть одна переменная истина и хоть одна ложная.

Покажем, что задача принадлежит классу NP.
Возьмём в качестве сертификата набор $x_{1,}\ldots, x_n, где\, x_i \in \{0,1\}$. Верификатор проверяет, что не все переменные равны друг другу и проверяет, что значение формулы при данных значениях переменных истинно. Время работы верификатора и длина сертификата, очевидно, полиномиальны.

Докажем, что задача NP-сложная.

Допустим, мы умеем её решать. Тогда сведём решение 3SAT к решению данной задачи. Пусть есть произвольное 3SAT выражение с n переменными:

$$(x_{i1} \vee \neg x_{i2} \vee x_{i3}) \wedge \ldots \wedge (x_{i(n-2)} \vee \neg x_{i(n-1)} \vee x_{in})$$

Добавим в него дизъюнкцию с таким выражением, содержащим две новые переменные:

$$(x_{i(n+1)} \vee \neg x_{i(n+1)} \vee x_{i(n+2)})$$

Получим:

$$(x_{i1} \vee \neg x_{i2} \vee x_{i3}) \wedge \ldots \wedge (x_{i(n-2)} \vee \neg x_{i(n-1)} \vee x_{in}) \wedge (x_{i(n+1)} \vee \neg x_{i(n+1)} \vee x_{i(n+2)})$$

Очевидно, что выражение в последних скобках всегда истинно, независимо от значения переменных в нём, поэтому если у задачи 3SAT для такого выражения есть какое-то решение, то найдётся и решение, в котором не все значения переменных равны друг другу. А такую задачу мы умеем решать. Результат решения этой задачи легко сводится к результату задачи 3SAT над исходным выражением: выражение в последних скобках всегда истинно, следовательно всё выражение истинно тогда и только тогда, когда истинно наше исходное выражение из n переменных. Поэтому ответ на данную задачу будет равен ответу на задачу 3SAT.

Очевидно, что сведение задачи производится за константное время (один фиксированный множитель мы добавим к выражению любой длины за одинаковое время).

Таким образом, мы свели решение задачи 3SAT к решению данной, а про 3SAT известно, что она NP-сложная, следовательно данная задача тоже NP-сложная.

D. Задача о рюкзаке. Есть n предметов, вес i-го предмета равен wi . Нужно положить некоторые предметы из них в рюкзак так, чтобы суммарный вес взятых предметов был максимален, но не более S.

Поставленную задачу с помощью бинарного (или линейного) поиска по новой переменной K можно свести к следующей задаче разрешимости:

> Задача о рюкзаке. Есть n предметов, вес i-го предмета равен wi . Возможно ли положить некоторые предметы из них в рюкзак так, чтобы суммарный вес взятых предметов был не меньше K, но не более S.

Покажем, что *новая* задача принадлежит классу NP.
Возьмём в качестве сертификата подмножество предметов, которые в ходе решения были помещены в рюкзак. Размер этого подмножества не более, чем размер множества всех предметов. Верификатор проверяет, что:
1) Суммарный вес взятых предметов не превышает S. За линейное время.
2) Проверить что взятый вес не меньше, чем K. За линейное время.

Докажем, что задача NP-сложная.

Допустим, мы умеем её решать. Сведём решение задачи о сумме подмножества к данной задаче. Пусть необходимо ответить, можно ли выбрать из множества W числа, сумма которых равна S. Вместо этого ответим, можно ли выбрать элементы wi из множества W так, чтобы их сумма не превысила S, но была не менее K = S. Такую задачу мы умеем решать. Ответ на неё будет равен ответу на *новую* задачу.

Очевидно, что сведение одной задачи к другой производится за константное время (сведение не требует преобразования входных данных, требуется сравнение только одного числа в ответе).

Таким образом, мы свели задачу о сумме подмножеств к *новой*, а про задачу о сумме подмножеств известно, что она NP-сложная, следовательно *новая* задача тоже NP-сложная.

I. Дан неориентированный граф G = (V, E), и число K. Проверить, что в графе есть остовное дерево, у которого степень каждой вершины хотя бы K.

Покажем, что задача принадлежит классу NP.
Возьмём в качестве сертификата подграф исходного графа $G' = (V', E'), где\ V' \subset V, E' \subset E$.
Длина сертификата не превышает размер входных данных. Верификатор должен убедиться, что в графе G’ нет циклов и убедиться, что степень вершины либо равна 1, либо не меньше K, что все вершины исходного графа входят в ответ (G == G’). Это можно сделать с помощью обхода в глубину за O(|V|+|E|).

Докажем, что задача NP-сложная.
...

N. Дано множество подмножеств Ci множества S и положительное число K. Можно ли выбрать такое множество S’ ⊂ S, что |S’| < K и при этом, для любого Ci : S’ ∩ Ci ≠ ∅.

Покажем, что задача принадлежит классу NP.
Возьмём в качестве сертификата подмножество исходного множества S’ ⊂ S. Длина сертификата не превышает |S|. Верификатор проверяет, что:
1) |S’| < K за время не более, чем линейное от мощности S’ : O(|S’|) (для стандартных структур данных за это время можно обойти контейнер поэлементно, если не сохранена информация о количестве элементов в нём). При этом мощность |S’| не может превышать мощности |S|, т. е. эта проверка выполняется за время O(|S|).
2) для любого Ci : S’ ∩ Ci ≠ ∅ за время
$O(|S'|*|C|*max_{i=1..|C|}(|C_i|))=O(|S|*|C|*|S|)=O(|S|^2*|C|)$ (поэлементное сравнение элементов из S’ с элементами из Ci для всех i).
Таким образом, верификатор работает за полиномиальное время от |S| и от |C|, а длина сертификата полиномиальна от |S|.

Докажем, что задача NP-сложная.

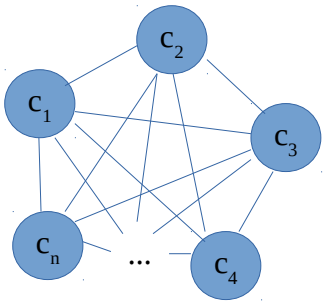
Допустим, мы умеем её решать. Сведём решение задачи о вершинном покрытии к данной задаче. Пусть есть неорграф (V, E) и число k. Существует ли вершинное покрытие мощностью не более k - 1? Пусть узлы графа — это элементы множества S, а множества Ci состоят из двух элементов и задают рёбра графа (если две вершины соединены ребром, то они составляют можество, количество рёбер равно количеству множеств Ci). Существует ли подмножество вершин S’ ⊂ S такое, что |S’| < K и для любого Ci : S’ ∩ C i ≠ ∅ (любое ребро Ci хотя бы одним концом входит в S’)? А это в точности данная задача. Ответ на неё равен ответу на исходную задачу о вершинном покрытии.

Сведение задачи выполняется за линейное время O(|V| + |E|) - это время, необходимое, чтобы представить граф в терминах задачи, которая нам дана, на самом деле преобразований данных не производится (только перекладывание в памяти), поэтому при желании можно обойтись и константным временем, если работать с исходным представлением графа в памяти. Сведение ответа производится за константное время.

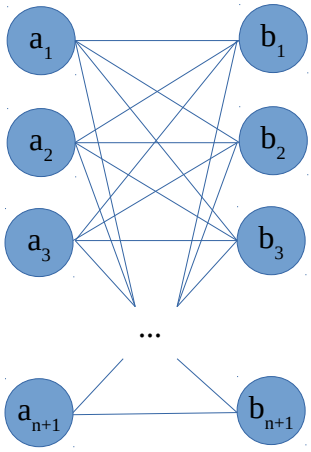
Таким образом, мы свели задачу о вершинном покрытии к данной, а про задачу о вершинном покрытии известно, что она NP-сложная, следовательно данная задача тоже NP-сложная.

R. 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.

Приведём пример, показывающий, что данный алгоритм не является X-приближённым в общем случае. Рассмотрим граф:



Полный подграф из n вершин



Полный двудольный подграф с 2 * (n + 1) вершинами

Очевидно, что максимальную клику размера n в этом графе образует левый полный подграф. Но степень любой его вершины меньше, чем степень любой вершины правого полного двудольного подграфа. Поэтому наш алгоритм сначала удалит из графа все узлы левого подграфа, затем будет удалять узлы правого двудольного подграфа, пока не получит клику размера 2. Граф такого типа можно составить для любого сколь угодно большого размера максимальной клики (левого подграфа), при этом предложенный алгоритм всегда будет находить только клику размера 2. Другими словами, для любого X мы можем построить подобный граф, в котором n = (X+1) * 2, применить к нему предложенный алгоритм и получить решение, отличающееся от истинного в (X + 1) раз. Следовательно, алгоритм не является X-приближенным ни для какого фиксированного X.

Т. Предложить 1⁄2-приближенный алгоритм для решения следующей задачи. В ориентированном графе G = (V, E) выбрать максимальное по мощности множество ребер, такое, что полученный подграф не содержит циклов.

Псевдокод:

```
Для каждой вершины в графе:
   Если количество рёбер, входящих в вершину, меньше, чем
   количество рёбер, исходящих из вершины, то:
      Удаляем все входящие рёбра
      Все исходящие рёбра добавляем в ответ
   Иначе:
      Удаляем все исходящие рёбра
      Все входящие рёбра добавляем в ответ
   Удаляем вершину и инцидентные ей рёбра из дальнейшего
   рассмотрения
Возвращаем ответ
```

Объяснение:

На каждой итерации (по вершинам), удаляя часть рёбер (все входящие или все исходящие) мы удаляем из графа все циклы, частью которых могла быть эта вершина, потому что после удаления рёбер в вершину либо нельзя придти, либо из неё нельзя выйти (она может быть только началом или концом пути, но не частью цикла). Оставшиеся рёбра попадают в ответ.

На каждой итерации мы рассматриваем какое-то подмножество множества рёбер. Меньшую часть из рассмотренных на каждой итерации рёбер мы удаляем, другую часть добавляем в ответ и убираем из дальнейшего рассмотрения. Очевидно, что мы рассмотрим каждое ребро ровно один раз. Таким образом, предложенный способ рассмотрения поделил всё множество рёбер на непересекающиеся подмножества, из каждого подмножества будет удалено не более половины его элементов, поэтому в сумме будет удалено не более половины всех рёбер. Т.е. количество рёбер в ответе не более, чем вдвое, меньше общего количества рёбер в цикле, а оно не меньше, чем количество рёбер в оптимальном решении. Следовательно, предложенный алгоритм 2-приближённый.