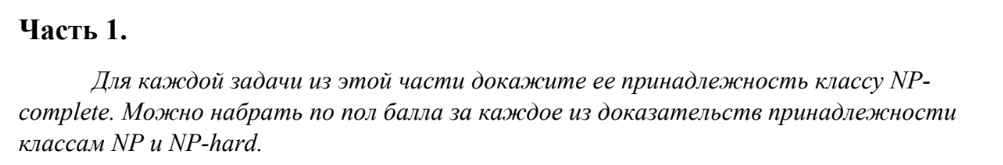
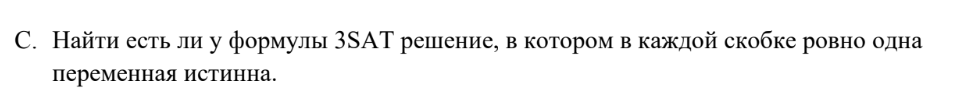
Вариант CDLPQV





Сертификат: переменные, использующиеся в задаче x1, x2, … ,xn

Верификатор: проверяет, что в каждой скобке только одна единица

Назовем эту задачу 3SAT\*, для доказательства ее принадлежности к NPH будем сводить 3SAT к ней.

Чтобы на выходе у нас в каждой скобке было не более одной единицы, будем каждую входную скобку раскладывать на 5 новых:

Вход -

B1 –

B2 –

B3 –

B4 –

B5 –

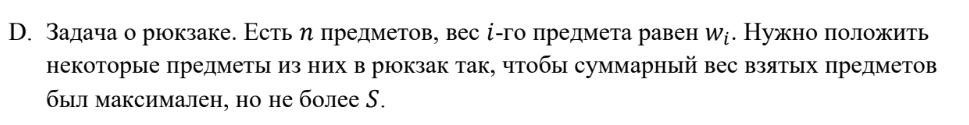
Выход – B1∩B2∩B3∩B4∩B5

Для наглядности построим таблицу для разных значений переменных

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| х1 | х2 | х3 | B1 | B2 | B3 | B4 | B5 | Выход |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

В строчках, где выход равен 1, удалось свести 3SAT к 3SAT\*, и во входной скобке было по одной единице.

Сведение полиномиально, так как при переходе добавляется 5 скобок для каждой скобки.



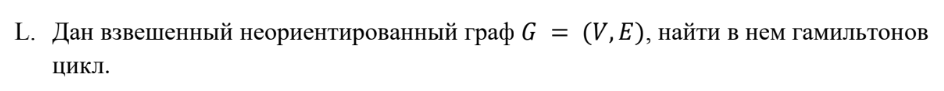
Сертификат: множество предметов

Верификатор: посчитать вес предметов

Сведем задачу SSP к поставленной.

Вес выбранных предметов должен быть >= W, но <= S. Заменим в задаче SSP S на W. А элементы множества весами предметов.

Сведение полиномиально, так как в обновленной задаче число предметов будет меньше или равно чем в SSP.



Сертификат: последовательность вершин

Верификатор: проход по циклу с проверкой

Введем обозначения

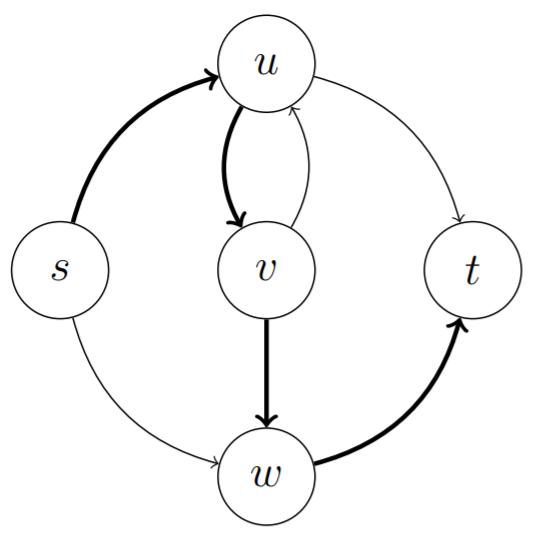
HAM – ориентированный граф с гамильтоновым циклом (он в NPH)

UHAM – неориентированный граф с гамильтоновым циклом

WUHAM – взвешенный неориентированный граф с гамильтоновым циклом

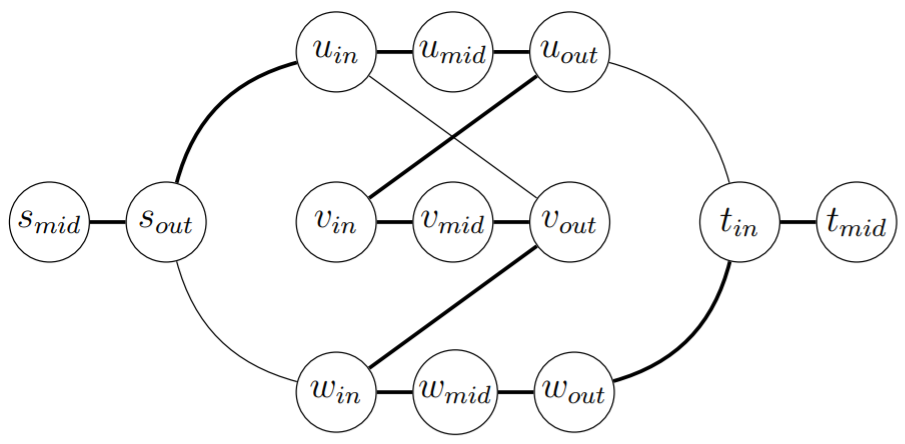
Жирными линиями на рисунках будут выделены участки гамильтонова пути.

Сведём HAM к UHAM. Воспользуемся следующим графом:



Каждую вершину Vi, кроме s и t, превратим в 3 вершины Vin, Vmid и Vout, соединённые в цепочку: Vin — Vmid — Vout. Аналогично s и t превратим в пары соединённых вершин smid — sout и tin — tmid. Для каждого ребра (u, v) исходного графа проведём ребро (uout и vin) в новом. Стартовой и конечной вершинами будем считать smid и tmid.

Получается новый граф:



По гамильтонову пути в исходном графе легко получить гамильтонов путь

в преобразованном. Достаточно вместо каждого ребра (u, v) из исходного пути взять цепочку вершин (umid , uout, vin , vmid). Или, иначе говоря, первую вершину пути заменить на пару вершин (smid , sout), каждую промежуточную вершину w заменить на три последовательные вершины (win , wmid , wout), а последнюю опять на пару (tin , tmid).

Теперь покажем, что и по гамильтонову пути в преобразованном графе можно восстановить гамильтонов путь в исходном. Для этого достаточно заметить, что вершины обязательно идут тройками вида (win , wmid , wout). Это доказывается по индукции.

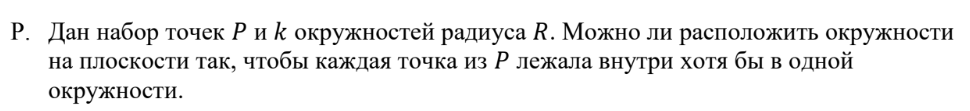
База: из smid пусть идёт в sout, поскольку других вариантов нет.

Переход: пусть до сих пор все вершины шли тройками. Из очередной out-вершины можно попасть только в in-вершины, причём из пока не задействованных троек. Если дальше путь пойдёт не в соответствующую mid-вершину, то эта вершина никогда не сможет быть посещена без повторного посещения текущей in-вершины, а значит путь не будет гамильтоновым. Значит, в гамильтоновом пути после in-вершины будет mid-вершина, а затем и out-вершина, поскольку других вариантов нет. Значит, снова получилась целая тройка, что и требовалось доказать.

Теперь сведем UHAM к WUHAM.

Для этого достаточно сопоставить всем ребрам единичные веса.

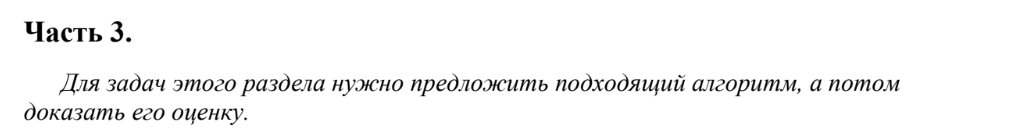
Сведение полиномиально. В первом случае оно прошло за (n-2)\*2+2, так как добавлялось по 2 пары вершин и ребер для всех вершин, кроме s и t, и по одной паре из вершины и ребра для s и t. Во втором случае добавилось столько весов, сколько было ребер.

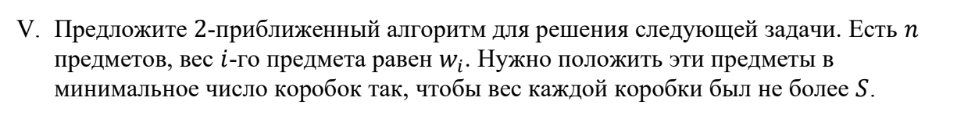


Сертификат: точки и окружности

Верификатор: проверка лежит точка внутри круга или нет

Пока без сведения(





Давайте раскладывать n предметов по k коробок, порядок взятия предметов при этом не важен. Каждый следующий предмет кладем в ту коробку, в которую он влезет, не превосходя S. Если текущие коробки не подошли, берем новую пустую коробку и кладем предмет в нее.

Такой алгоритм будет 2-приближенным для поставленной задачи.

Пусть открыто k ящиков, значит по крайней мере k-1 ящик заполнен хотя бы наполовину. Отсюда,

Так как сумма весов предметов является нижней оценкой OPT, то

2 \* OPT > k – 1. Получаем 2 \* OPT >= k.

Под OPT имеется ввиду стоимость оптимального решения задачи.