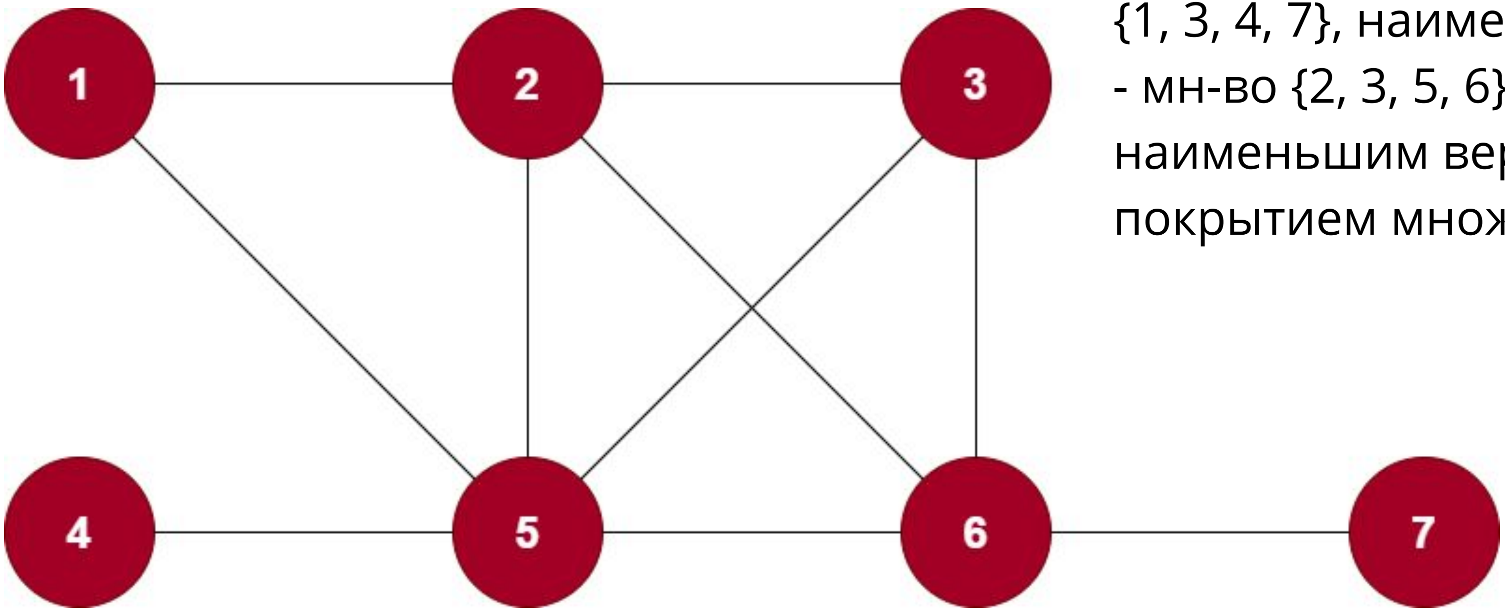
Вершинное покрытие графа. Теорема 1. Приближённый алгоритм для задачи о вершинном покрытие.

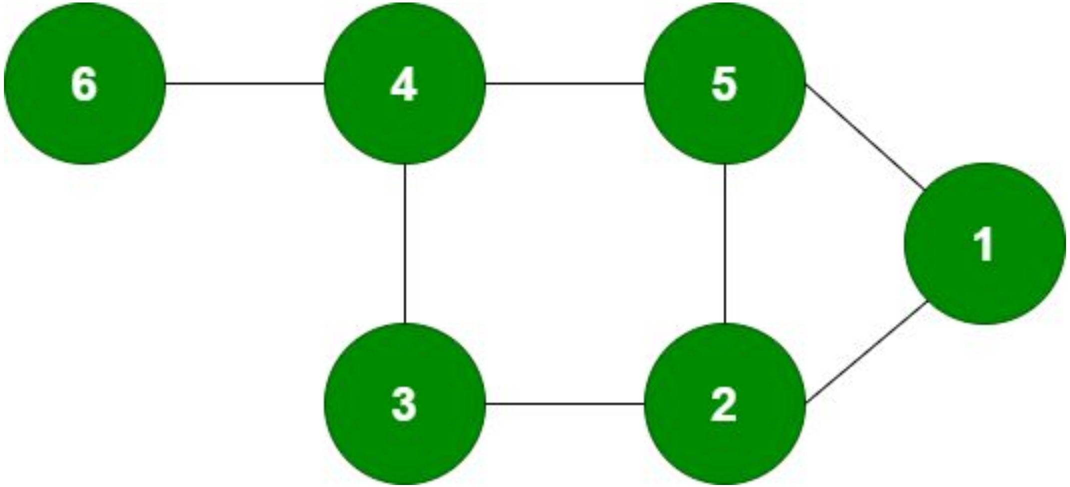
Что такое вершинное покрытие графа?

- это такое множество вершин, что каждое ребро графа инцидентно хотя бы одной из этих вершин. Наименьшее число вершин в вершинном покрытие графа G обозначается через β(G) и называется числом вершинного покрытия графа



Например, в данном графе наибольшим независимым множеством является мн-во {1, 3, 4, 7}, наименьшей кликой - мн-во {2, 3, 5, 6}, наименьшим вершинным покрытием множество {2, 5, 6}

Граф, изображенный справа, имеет вершинное покрытие {1,3,5,6} размера 4. Однако оно не является наименьшим вершинным покрытием, поскольку существуют вершинные покрытия меньшего размера, такие как {2,4,5} и {1,2,4}.



Теорема 1

Подмножество U множества вершин графа G является вершинным покрытием тогда и только тогда, когда $\overline{U} = VG - U$

- независимое множество.

Доказательство:

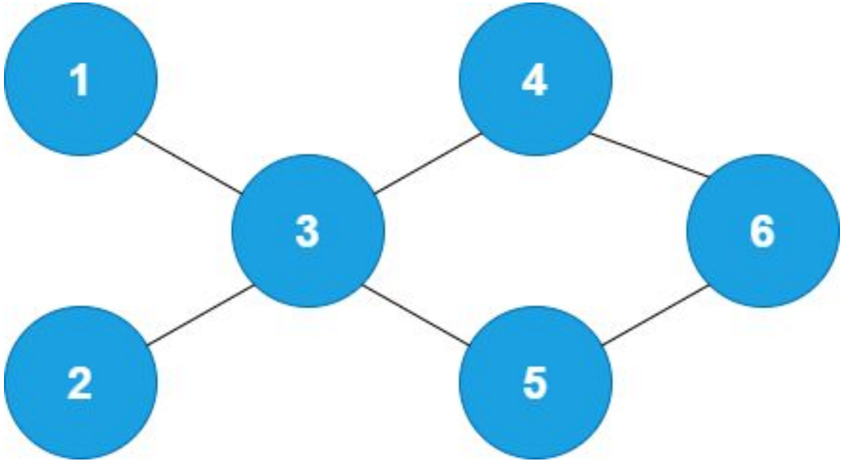
Если U - вершинное покрытие, то всякое ребро содержит хотя бы одну вершину из множества U и, значит, нет ни одного ребра, соединяющего две вершины из множества $\overline{U}$. Следовательно, $\overline{U}$ - независимое множество. Обратно, если $\overline{U}$ - независимое множество, то нет ребер, соединяющих вершины из $\overline{U}$ и, значит, у каждого ребра одна или обе вершины принадлежат множеству U. Следовательно, U - вершинное покрытие.

Из этой теоремы следует, что α(G) + β(G) = n для любого графа с вершинами.

Приближенный алгоритм для задачи о вершинном покрытие

Работа алгоритма начинается с создания пустого множества X и состоит в выполнении однотипных шагов, в результате каждого из которых к множеству X добавляются некоторые вершины. Допустим, перед очередным шагом имеется некоторое множество вершин X . Если оно покрывает все ребра, то процесс заканчивается и множество принимается в качестве искомого вершинного покрытия. В противном случае выбирается какое-нибудь непокрытое ребро (a,b), и вершины a и b добавляются к множеству X.

Пример работы алгоритма:



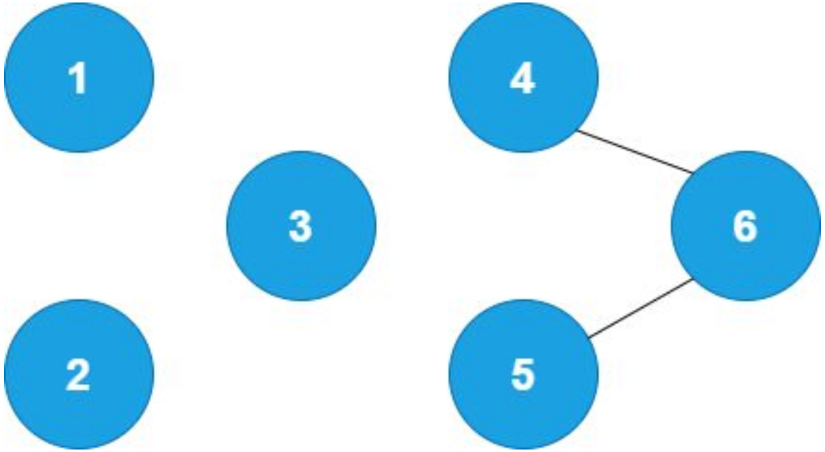
Первая итерация:

- Выбираем случайное ребро. Например, ребро (1, 3).
- Добавляем в решение S обе вершины выбранного ребра: S={1, 3}.
- Удаляем из графа все ребра, инцидентные вершинам 1 или 3.

Пример работы алгоритма:

Вторая итерация:

- Выбираем случайное ребро. Пусть это будет ребро (4, 6).
- Добавляем в решение S обе вершины выбранного ребра: S={1, 3, 4, 6}.



- Удаляем из графа все ребра, инцидентные вершинам 4 или 6.

В графе не осталось ребер. Следовательно, результатом работы нашего алгоритма будет вершинное покрытие S={1, 3, 4, 6}.

Обозначим через $\beta'(G)$ мощность вершинного покрытия, которое получится при применении этого алгоритма к графу G, и докажем, что $\beta'(G) \leq 2\beta(G)$. Иначе говоря, полученное с помощью этого алгоритма решение не более чем в два раза отличается от оптимального.

Действительно, допустим, что до окончания работы алгоритм выполняет k шагов, добавляя к множеству X вершины ребер (a1, b1)...(ak, bk). Тогда $\beta'(G)= 2k$. Никакие два из этих k ребер не имеют общей вершины. Значит, чтобы покрыть все эти ребра, нужно не меньше k вершин. Следовательно, $\beta(G)\geq k$ и $\beta'(G) \leq 2\beta(G)$.

Таким образом, мы получили простой полиномиальный по времени алгоритм с хорошей точностью для решения NP-трудной задачи.