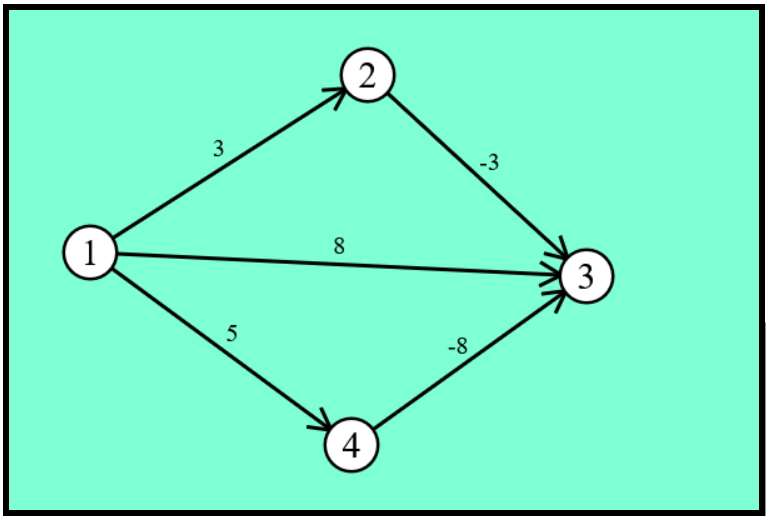
**Алгоритм Форда-Беллмана**

Алгоритм Дейкстры имел недостаток: он работал только с неотрицательными весами. Теперь рассмотрим задачу нахождения кратчайшего пути в графе, где веса могут быть отрицательными. Граф будет ориентированным, потому что в неориентированном графе можно ходить по любому отрицательному ребру бесконечное количество раз.



На графе из примера выше кратчайший путь из 1 в 3 будет иметь вид 1-4-3 (длина пути -3).

Сразу рассмотрим все шаги алгоритма.

**Алгоритм Форда-Беллмана:**

1. Завести массив d кратчайших расстояний от стартовой вершины до всех остальных. Изначально d от стартовой вершины равно 0, от остальных вершин +∞.

2. Повторить n-1 раз, где n – число вершин:

Перебираем все рёбра графа. Пусть текущее ребро ведёт из v в u и имеет вес w. Обновляем расстояние до вершины u по формуле d[u] = min(d[u], d[v] + w).

Таким образом, алгоритм много раз перебирает все рёбра и пытается обновить расстояние до конца ребра через расстояние до его начала.

**Почему n-1 раз?**

Пусть в графе нет отрицательных циклов (циклов с отрицательной суммой весов). Тогда обязательно есть кратчайший путь, не проходящий 2 раза по одной вершине (нет смысла ходить по циклу, если он неотрицательный). Это значит, что всегда есть кратчайший путь длины не более n-1 (потому что такую длину будет иметь путь, проходящий по всем вершинам).

При первом проходе по всем рёбрам правильно посчитаются все кратчайшие пути из одного ребра. При втором проходе – из двух рёбер. Таким образом, n-1 прохода достаточно, чтобы найти кратчайшие пути от стартовой вершины до всех, если в графе нет отрицательных циклов.

**Что делать, если есть отрицательные циклы?**

Если в графе есть отрицательные циклы, то кратчайшее расстояние до некоторых вершин может быть равно -∞.

Чтобы найти все такие вершины, можно запустить больше n-1 итерации алгоритма (например, 2n). Если расстояние до вершины изменилось после первых n-1 итераций, то она лежит на каком-то отрицательном цикле (и расстояние до неё равно -∞, если она вообще достижима).

**Время работы алгоритма**

Так как алгоритм n-1 раз проходит по всем m рёбрам, то он работает за время **O(nm),** что медленнее алгоритма Дейкстры.