# Пример работы алгоритма Прима

Этот документ содержит пошаговое объяснение работы алгоритма Прима на примере графа.  
  
Граф с 4 вершинами и следующими ребрами (вес указан в скобках):  
- Вершина 0 соединена с вершинами 1 (вес 10), 2 (вес 5)  
- Вершина 1 соединена с вершинами 0 (вес 10), 2 (вес 2), 3 (вес 1)  
- Вершина 2 соединена с вершинами 0 (вес 5), 1 (вес 2), 3 (вес 9)  
- Вершина 3 соединена с вершинами 1 (вес 1), 2 (вес 9)  
  
Алгоритм Прима строит минимальное остовное дерево (MST), добавляя вершины с минимальным весом ребра, ведущим к ним из уже построенного остова. Далее представлен пошаговый процесс работы алгоритма.

1. Инициализация:  
- `min\_edge`: [0, ∞, ∞, ∞] (вес для начальной вершины 0 установлен в 0)  
- `best\_edge`: [-1, -1, -1, -1] (изначально все значения неопределенные)  
- `used`: [false, false, false, false] (ни одна вершина не посещена)

2. Первая итерация (начало с вершины 0):  
- Мы выбираем вершину 0, так как `min\_edge[0] = 0` (это начальная вершина).  
- Отмечаем вершину 0 как использованную: `used[0] = true`  
- Рассматриваем ребра, исходящие из вершины 0:  
 - Ребро (0-1) с весом 10: `min\_edge[1]` обновляется с ∞ до 10, `best\_edge[1]` обновляется с -1 до 0.  
 - Ребро (0-2) с весом 5: `min\_edge[2]` обновляется с ∞ до 5, `best\_edge[2]` обновляется с -1 до 0.  
- `min\_edge` теперь: [0, 10, 5, ∞]  
- `best\_edge` теперь: [-1, 0, 0, -1]

3. Вторая итерация (выбор вершины 2):  
- Мы выбираем вершину 2, так как `min\_edge[2] = 5` (минимальное значение среди неиспользованных вершин).  
- Отмечаем вершину 2 как использованную: `used[2] = true`  
- Рассматриваем ребра, исходящие из вершины 2:  
 - Ребро (2-0) с весом 5: уже используется, пропускаем.  
 - Ребро (2-1) с весом 2: `min\_edge[1]` обновляется с 10 до 2, `best\_edge[1]` обновляется с 0 до 2.  
 - Ребро (2-3) с весом 9: `min\_edge[3]` обновляется с ∞ до 9, `best\_edge[3]` обновляется с -1 до 2.  
- `min\_edge` теперь: [0, 2, 5, 9]  
- `best\_edge` теперь: [-1, 2, 0, 2]

4. Третья итерация (выбор вершины 1):  
- Мы выбираем вершину 1, так как `min\_edge[1] = 2` (минимальное значение среди неиспользованных вершин).  
- Отмечаем вершину 1 как использованную: `used[1] = true`  
- Рассматриваем ребра, исходящие из вершины 1:  
 - Ребро (1-0) с весом 10: уже используется, пропускаем.  
 - Ребро (1-2) с весом 2: уже используется, пропускаем.  
 - Ребро (1-3) с весом 1: `min\_edge[3]` обновляется с 9 до 1, `best\_edge[3]` обновляется с 2 до 1.  
- `min\_edge` теперь: [0, 2, 5, 1]  
- `best\_edge` теперь: [-1, 2, 0, 1]

5. Четвертая итерация (выбор вершины 3):  
- Мы выбираем вершину 3, так как `min\_edge[3] = 1` (минимальное значение среди неиспользованных вершин).  
- Отмечаем вершину 3 как использованную: `used[3] = true`  
- Рассматриваем ребра, исходящие из вершины 3:  
 - Ребро (3-1) с весом 1: уже используется, пропускаем.  
 - Ребро (3-2) с весом 9: уже используется, пропускаем.  
- `min\_edge` и `best\_edge` остаются без изменений: [0, 2, 5, 1] и [-1, 2, 0, 1]

На этом этапе все вершины использованы, и алгоритм завершен. Результатом работы алгоритма является минимальное остовное дерево, которое включает следующие ребра:  
- (2-0) с весом 5  
- (1-2) с весом 2  
- (3-1) с весом 1  
  
Итоговое минимальное остовное дерево (MST):  
- Вершина 2 соединяется с вершиной 0 ребром весом 5  
- Вершина 1 соединяется с вершиной 2 ребром весом 2  
- Вершина 3 соединяется с вершиной 1 ребром весом 1