
Yandex

Yandex

Решение задач на графах в MapReduce

Роман Едемский
Разработчик БК

- MapReduce
- PageRank
- Алгоритм Борувки

Что такое MapReduce?

MapReduce – это модель вычислений, предназначенная для обработки больших массивов данных посредством распределенных вычислений на кластере.

Зачем нужен MapReduce?

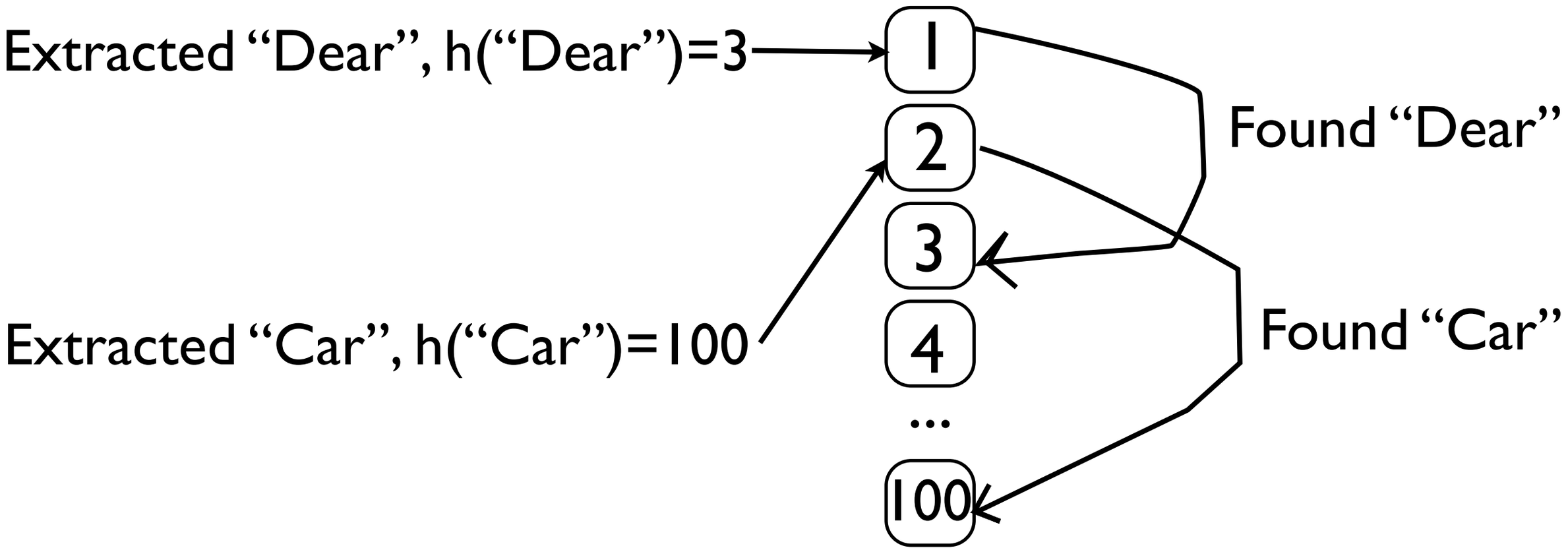
- **100** документов
- Суммарно занимающих **100 Mb**
- Локально на **1** машине

Зачем нужен MapReduce?

- **10,000,000** документов
- Каждый документ примерно по **1Mb**
- Суммарно ~ **10Tb**
- Распределены равномерно по **100** машинам

Зачем нужен MapReduce?

Хеш функция **h: word→{1,2,...,100}**



Зачем нужен MapReduce?

Однако, что если...

- Поломался жесткий диск или отключили электричество
  - Как востановить потеряные данные?
  - Как продолжить выполнение задачи?
  - Как перекинуть задачу на другой кластер?
- Разные характеристики машин и каналы в ДЦ
  - Как эффективно балансировать нагрузку?
- Заказчик решил, что хочет считать частоты тандемов слов
- Объем данных вырос в 10 раз

Зачем нужен MapReduce?

MapReduce - это:

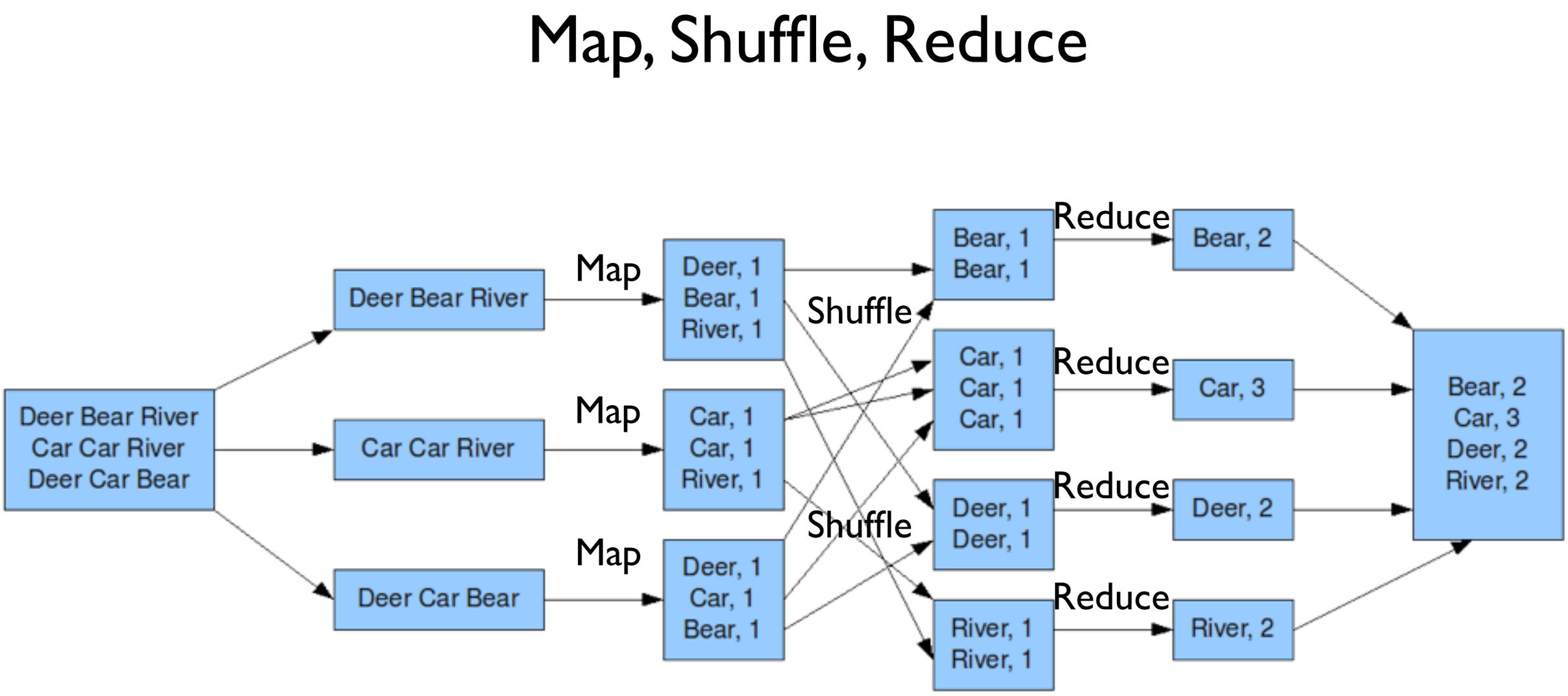
- Автоматическая масштабируемость
- Отказоустойчивость
- Интеллектуальное управление нагрузкой кластера

Как работать с MapReduce? Данные

Table, Record, Key, Value

	Key	Value
Record 1 →	Key 1	Value 1
Record 2 →	Key 2	Value 2
Record 3 →	Key 3	Value 3
Record 4 →	Key 4	Value 4
…	…	…
Record N →	Key N	Value N

Как работать с MapReduce? Обработка



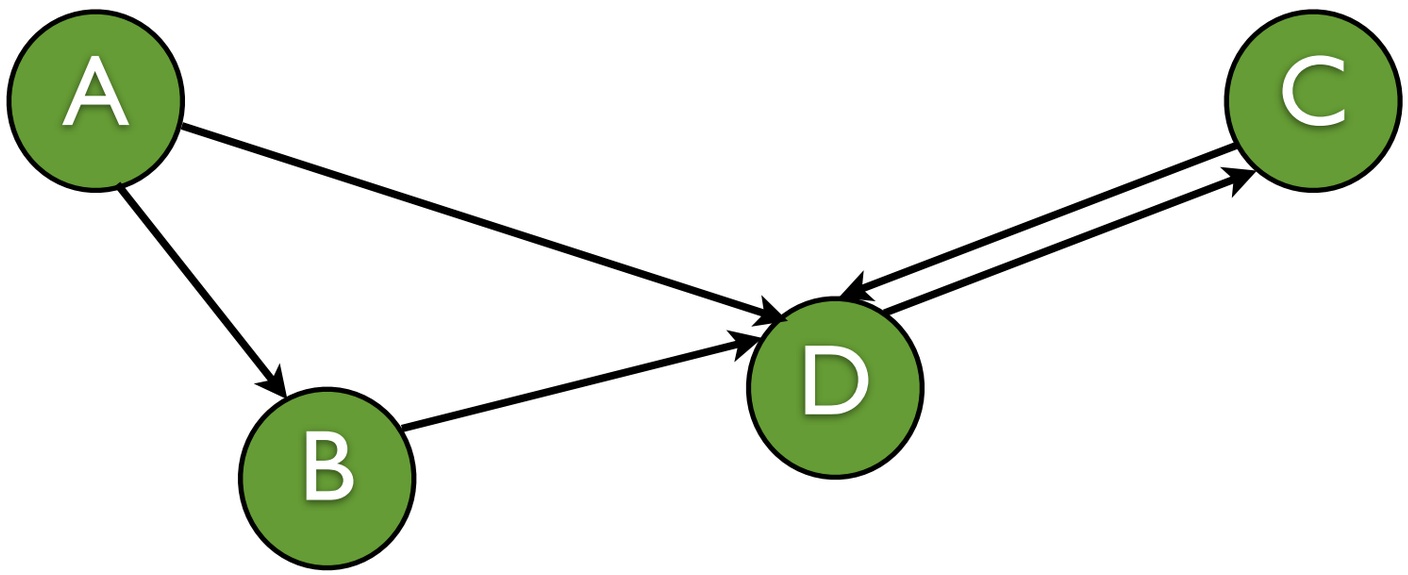
Как работать с MapReduce? Код

```
class WordCountMapper : public MRMapper {
  public:
    void Map(const string &key, const string &value) {
      for_each(word in value) {
         emit(word, “1”);
      }
    }
}

class WordCountReducer : public MRReducer {
  public:
    void Reduce(const string &key,
                const vector<string> &value) {
      emit(key, uintToString(value.size()) );
    }
}
```

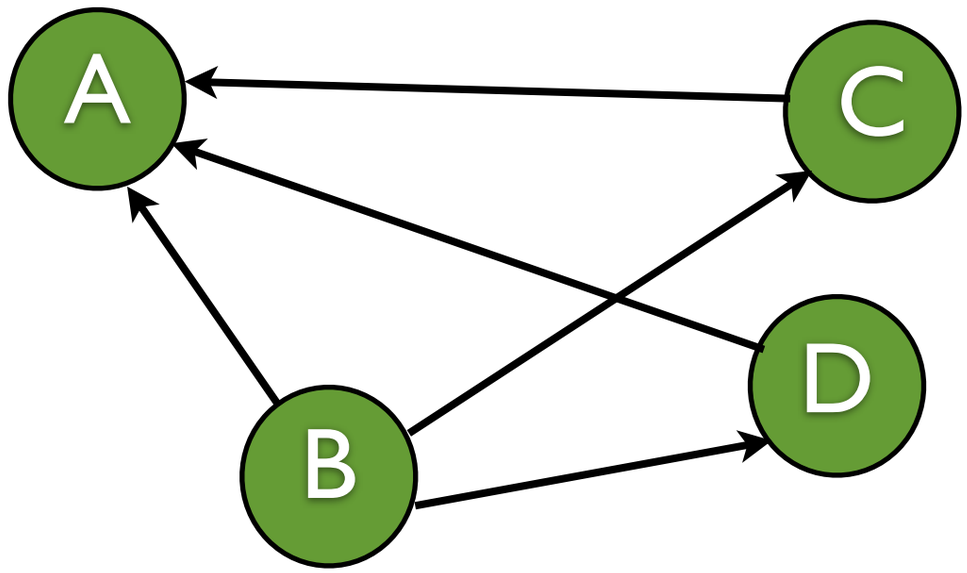
- MapReduce
- PageRank
- Алгоритм Борувки

Что такое PageRank?

PageRank — это числовая величина, характеризующая «важность» веб-страницы. Чем больше ссылок на страницу, тем она «важнее»

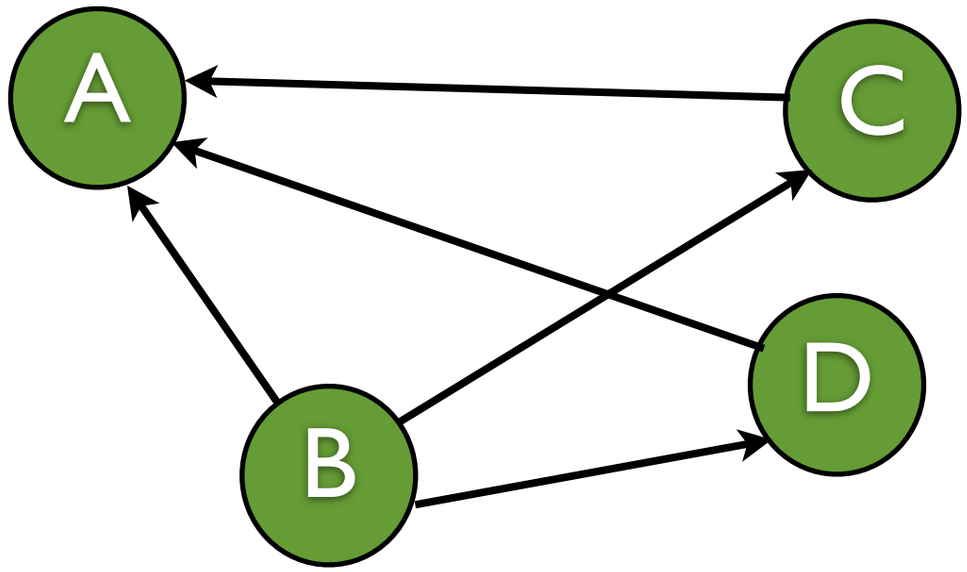


Расчёт PageRank на MapReduce



$$PR(u) = (1-d) \sum_{v \in V} \frac{1}{N} \cdot PR(v) + d \sum_{v:(v,u) \in E} \frac{PR(v)}{deg_{out}(v)} = \frac{1-d}{N} + d \sum_{v:(v,u) \in E} \frac{PR(v)}{deg_{out}(v)}$$

Расчёт PageRank на MapReduce



$$PR(A) = \frac{1-d}{4} + d \cdot \left(\frac{PR(B)}{3} + \frac{PR(C)}{1} + \frac{PR(D)}{1} \right)$$

$$PR(B) = \frac{1-d}{4}$$

$$PR(C) = \frac{1-d}{4} + d \cdot \left(\frac{PR(B)}{3} \right)$$

$$PR(D) = \frac{1-d}{4} + d \cdot \left(\frac{PR(B)}{3} \right)$$

PageRank

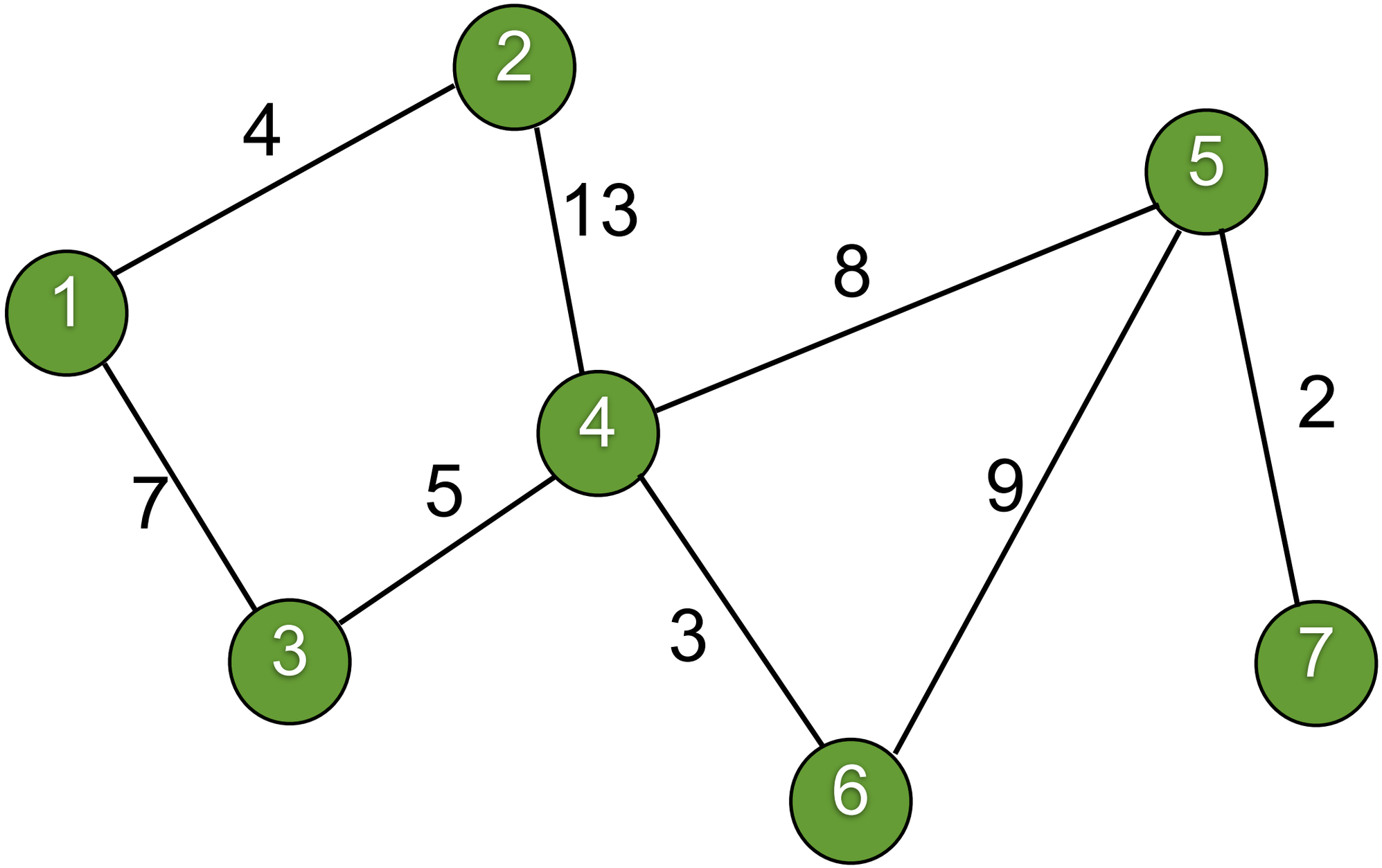
```
(Key; Value) # Будем обозначать записи так
# Хранить граф удобнее всего списком смежности:
(V; “PageRank”: PR, “edgesTo”: [V1, V2, V3, ... Vn] )
# Псевдокод PageRank
Map(Key; Value):
  # Re-emitting
  Emit (Key, Value)
  # Emitting update messages to the neighborhood
  for_each (U in Value.edgesTo):
    Emit (U; “UpdatePageRank”: PR / Value.edgesTo.size())

Reduce(V; vertex_info, incoming_messages):
  sum_incoming = 0.0
  for_each (message in incoming_messages):
    sum_incoming += message.UpdatePageRank
  vertex_info.PageRank = d / N + (1 - d) * sum_incoming
  Emit(V; vertex_info)
```

- MapReduce
- PageRank
- Алгоритм Борувки

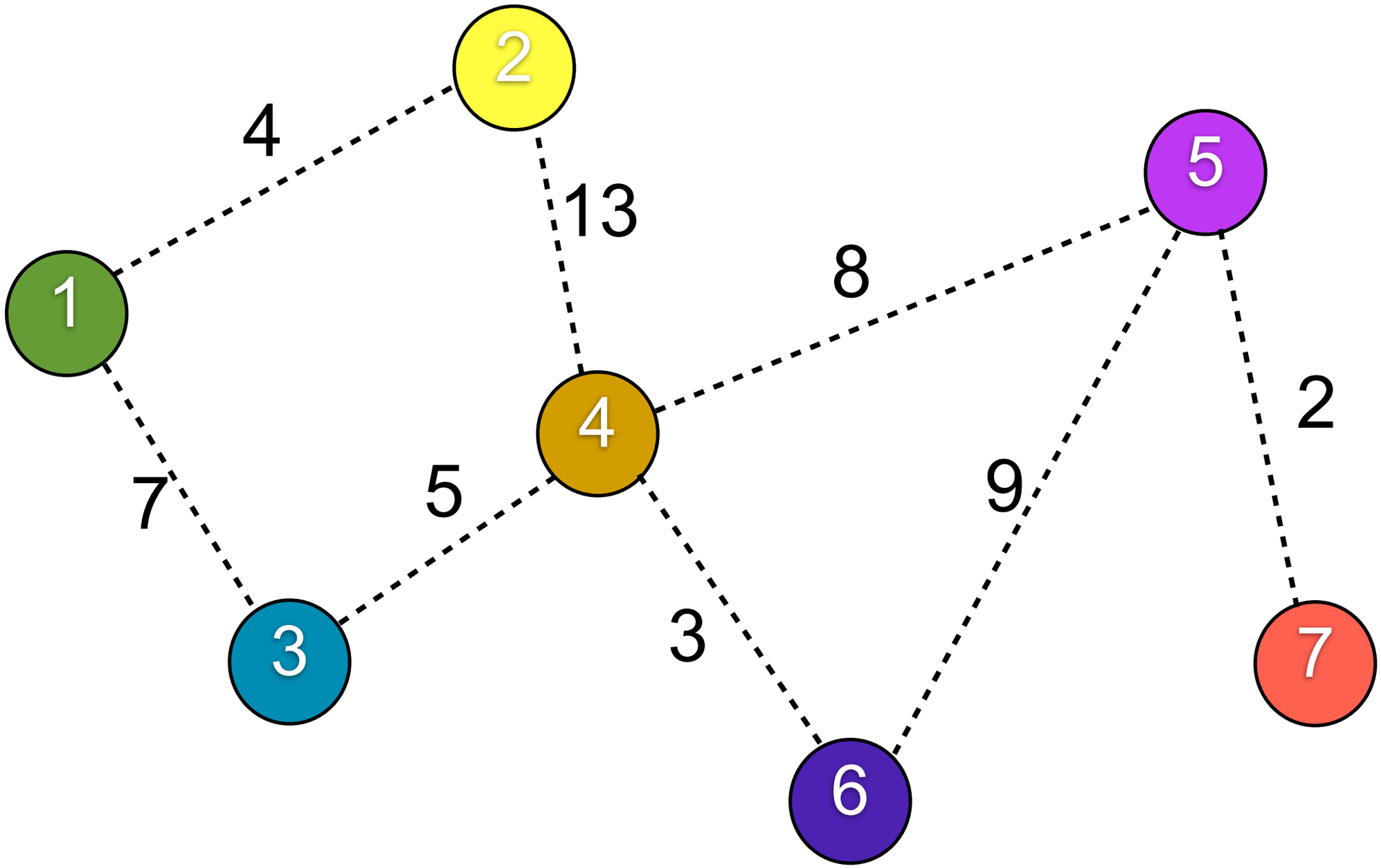
Подходит ли алгоритм для MapReduce?

- Подходит хорошо:
  - PageRank
  - Форда-Беллмана
- Не подходит:
  - Дейкстры
  - Крускала
  - Прима
- Подходит плохо:
  - Поиск в ширину, O(D*(|E|+|V|))

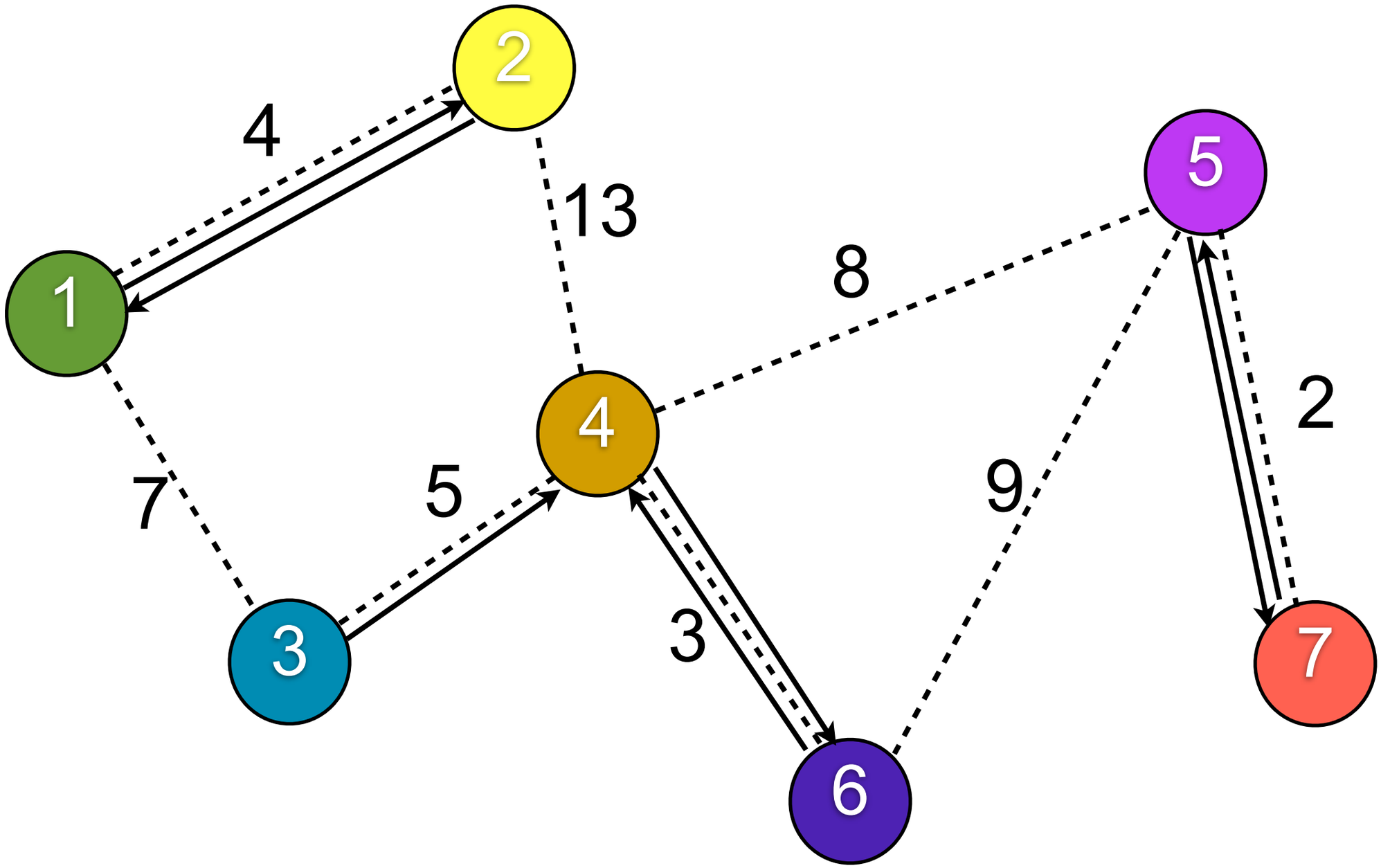
Алгоритм Борувки



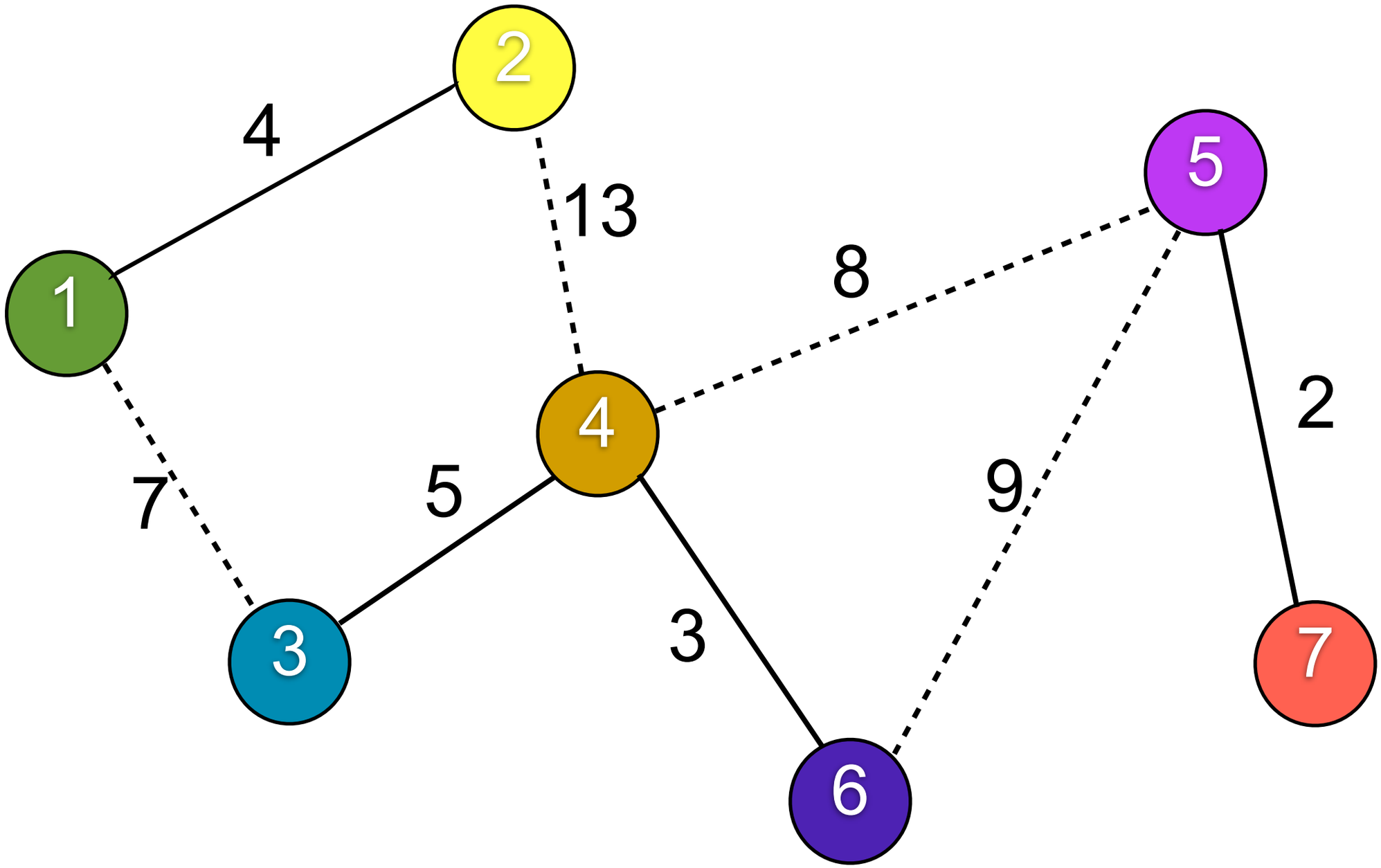
Алгоритм Борувки



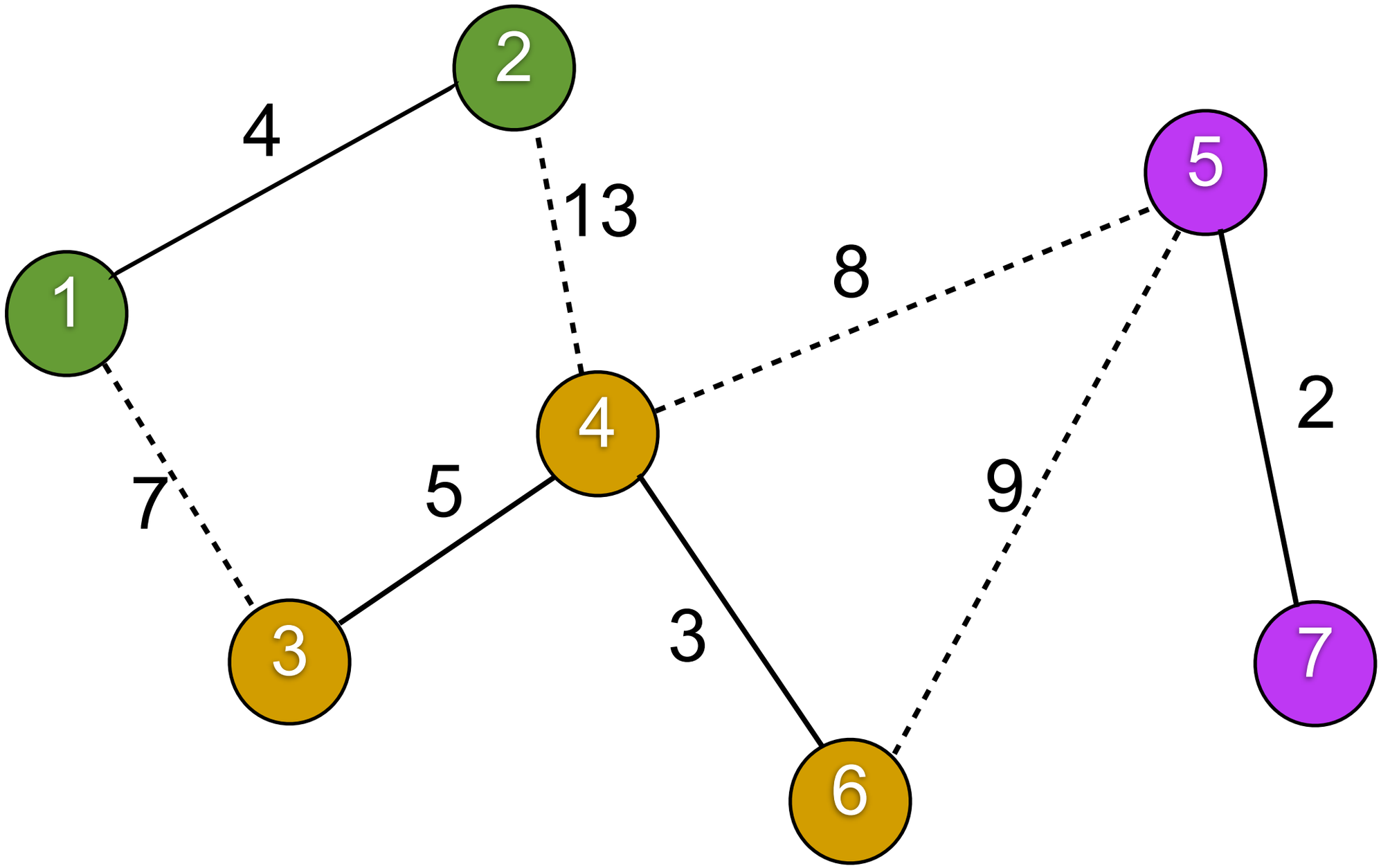
Алгоритм Борувки



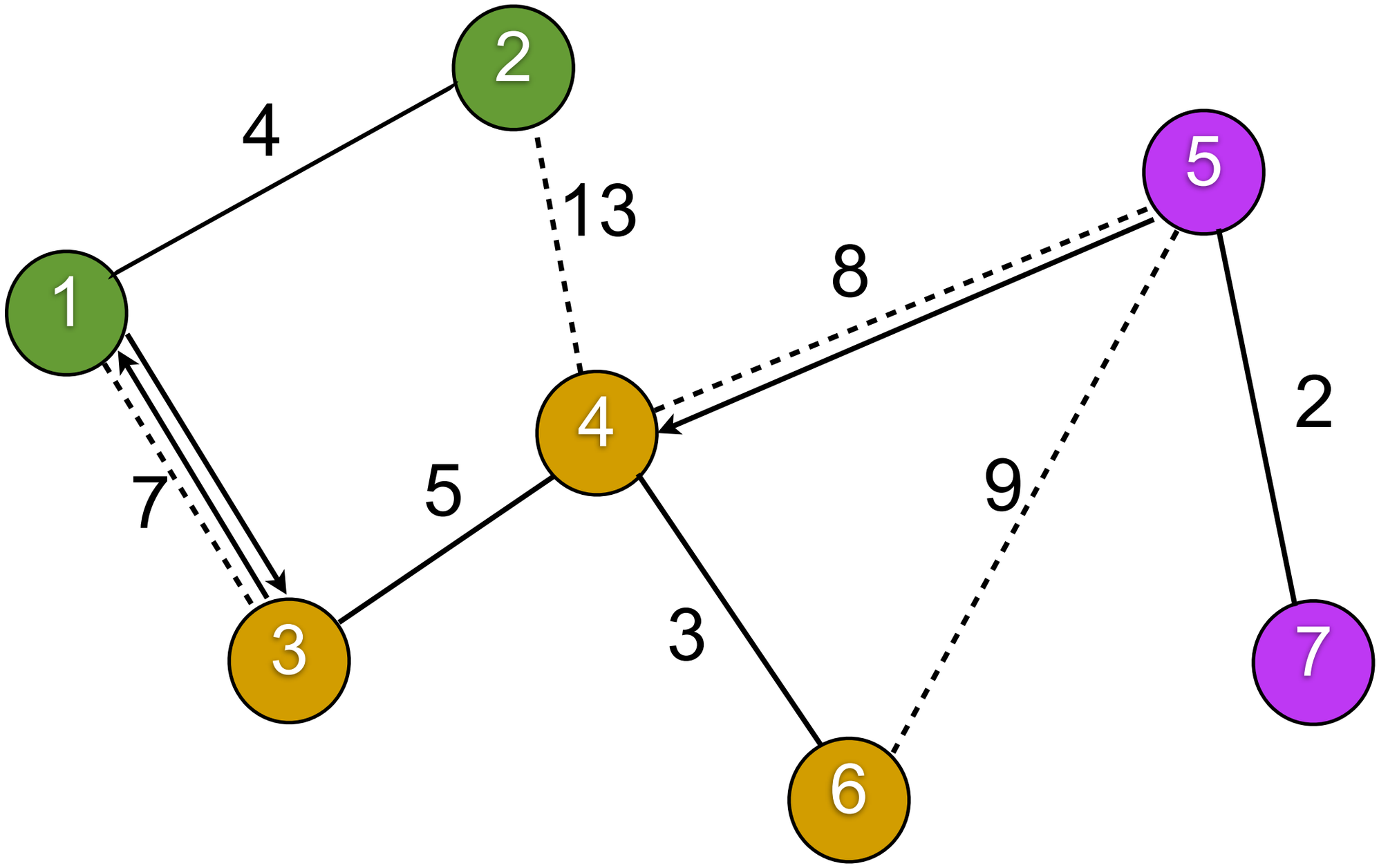
Алгоритм Борувки



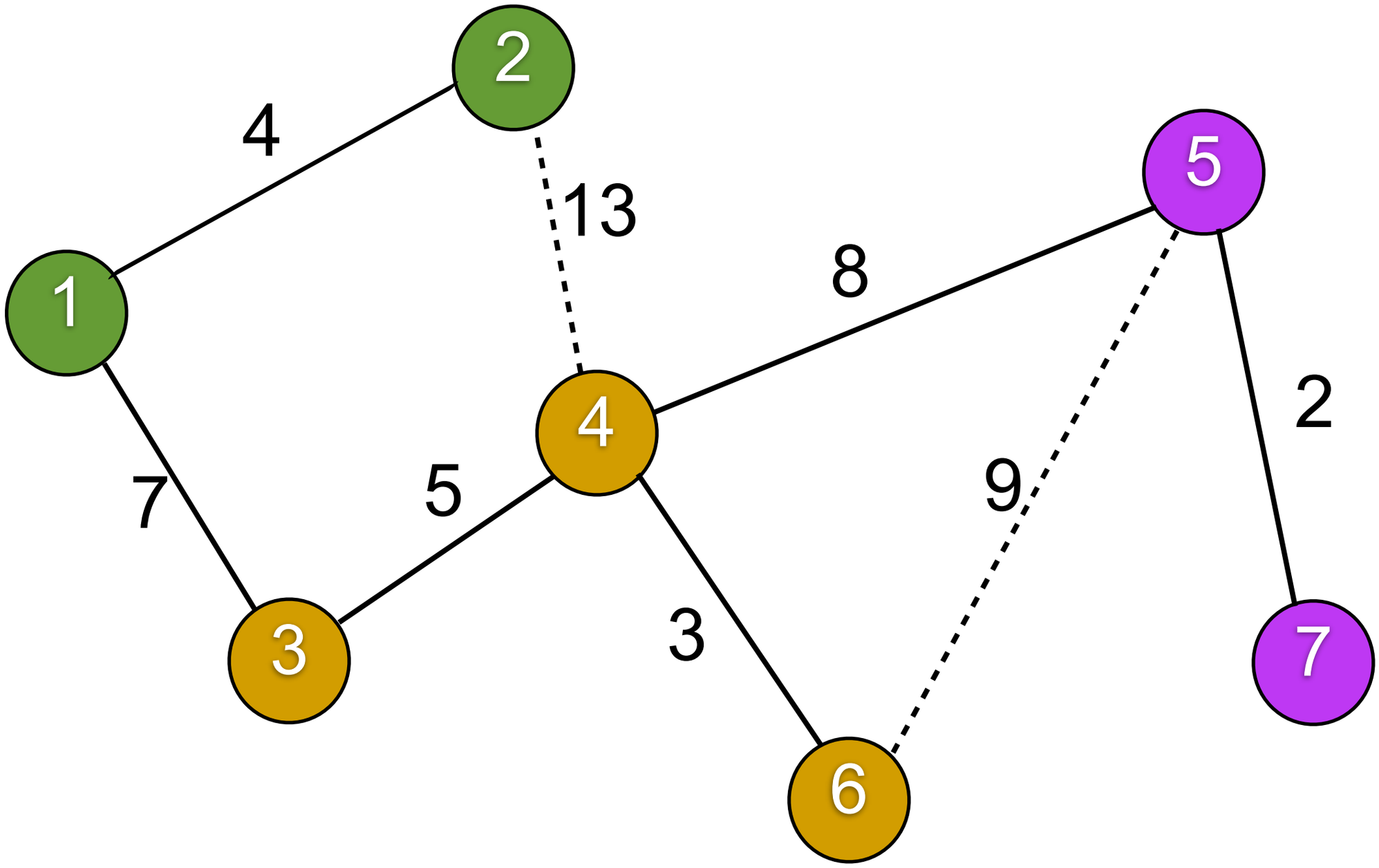
Алгоритм Борувки



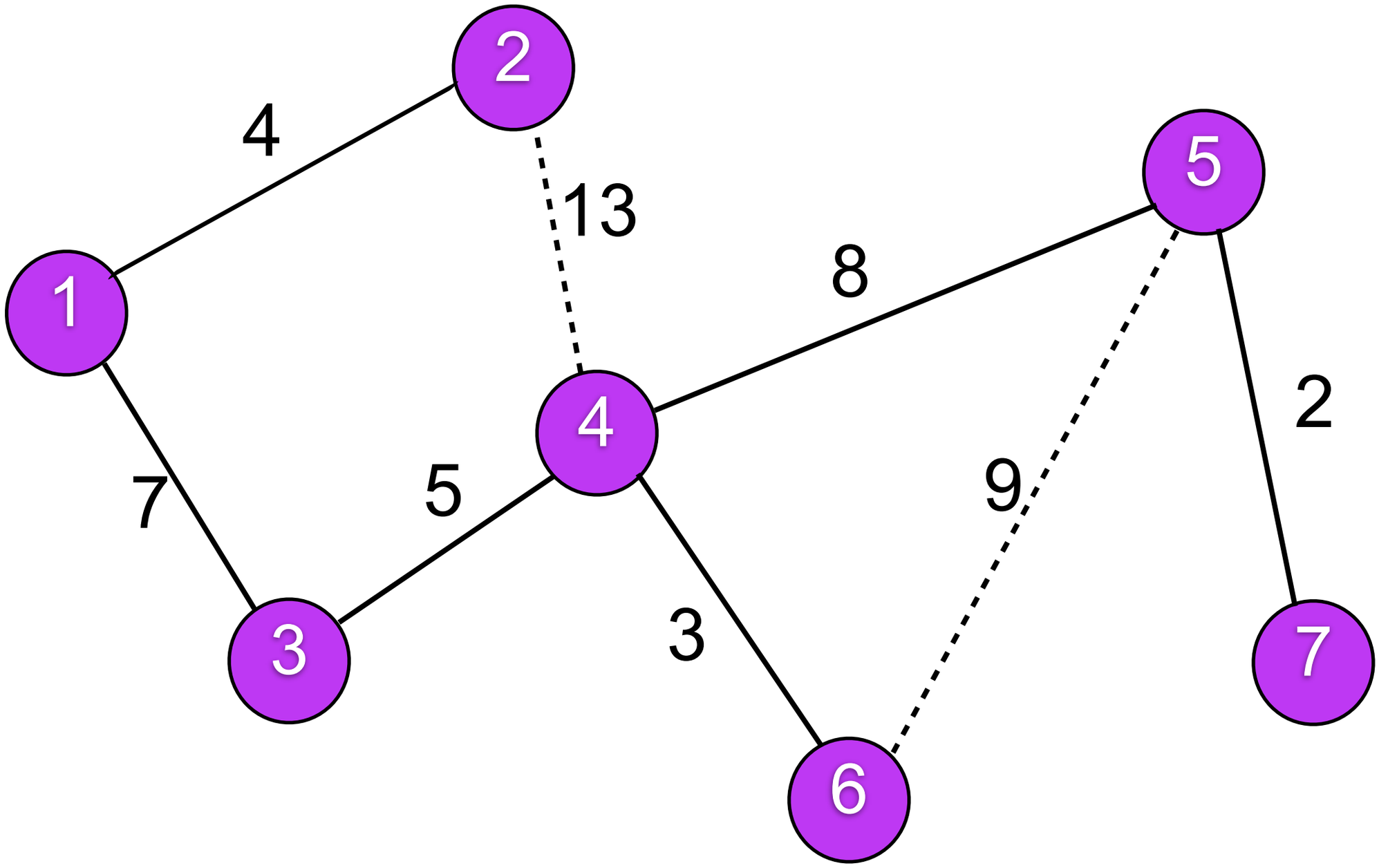
Алгоритм Борувки



Алгоритм Борувки



Алгоритм Борувки



Алгоритм Борувки

```
# Псевдокод алгоритма Борувки
boruvki():
  while |components| > 1:
    new_edges = []
    # Шаг-1: нахождения ближайших соседей
    for_each (component in components):
      (u, v) = component.get_shortest_outgoing_edge()
      new_edges.append(u, v)
    # Шаг-2: перекраска компонент
    for_each (edge in new_edges):
      add_to_result(edge)
      components.merge(edge.u, edge.v)
```

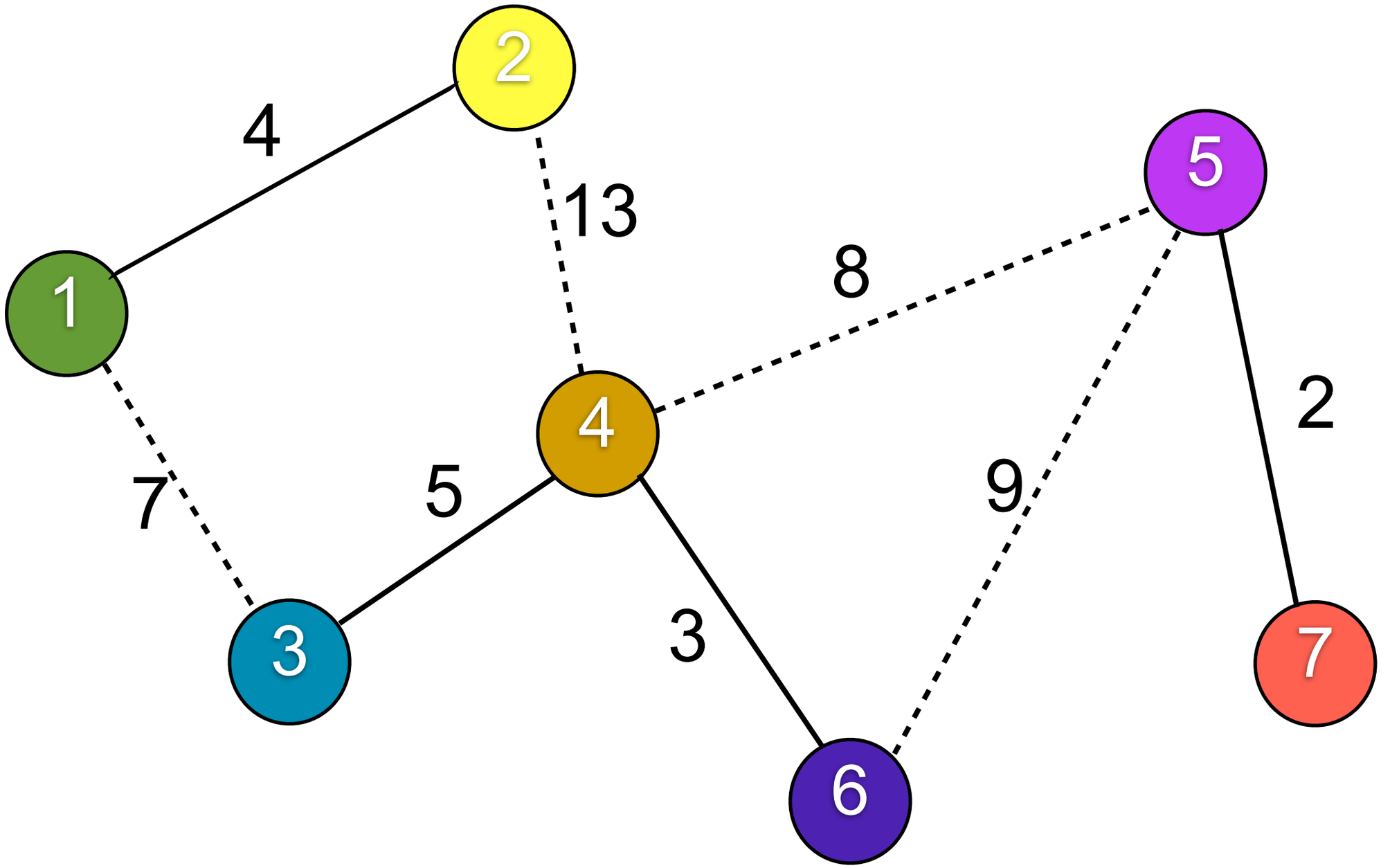
Алгоритм Борувки. Шаг-1

```
# Граф будем хранить так:
(V; “edgesTo”: [(V1, W1), (V2, W2), ... (Vn, Wn)] )
# Псевдокод нахождения ближайших соседей
Map(Key; Value):
  # Re-emitting
  Emit (Key, Value)
  closest = find_closest_component(Value.edgesTo)
  Emit (closest_component; “Message”: Key)

Reduce(V; vertex_info, incoming_messages):
  vertex_info.neighbours = incoming_messages
```

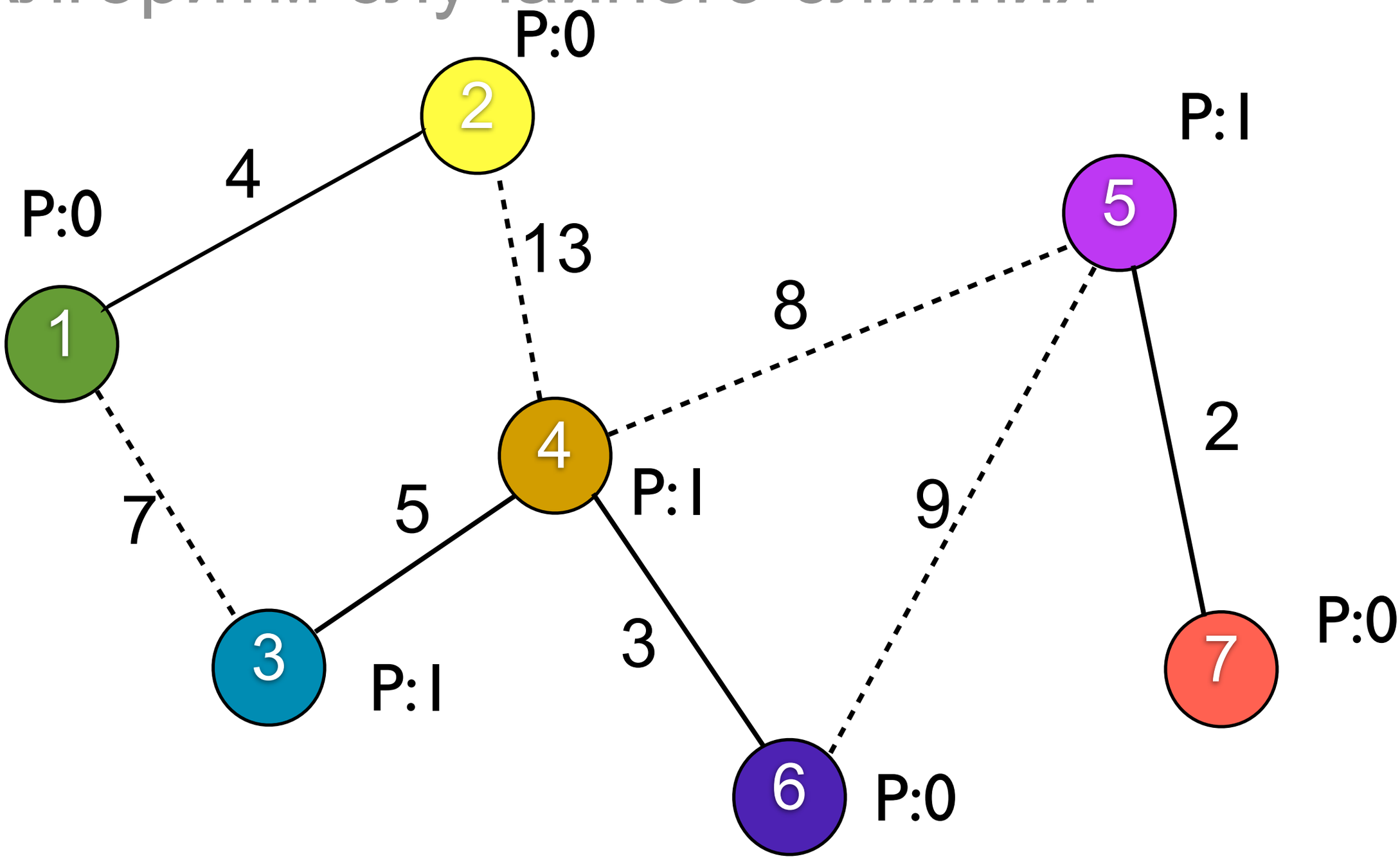
Алгоритм Борувки

```
#Псевдокод алгоритма Борувки
boruvki():
  while |V| > 1:
    new_edges = []
    # Шаг-1: нахождения ближайших соседей
    for_each (component in components):
      (u, v) = component.get_shortest_outgoing_edge()
      new_edges.append(u, v)
    # Шаг-2: перекраска компонент
    for_each (edge in new_edges):
      add_to_result(edge)
      components.merge(edge.u, edge.v)
```

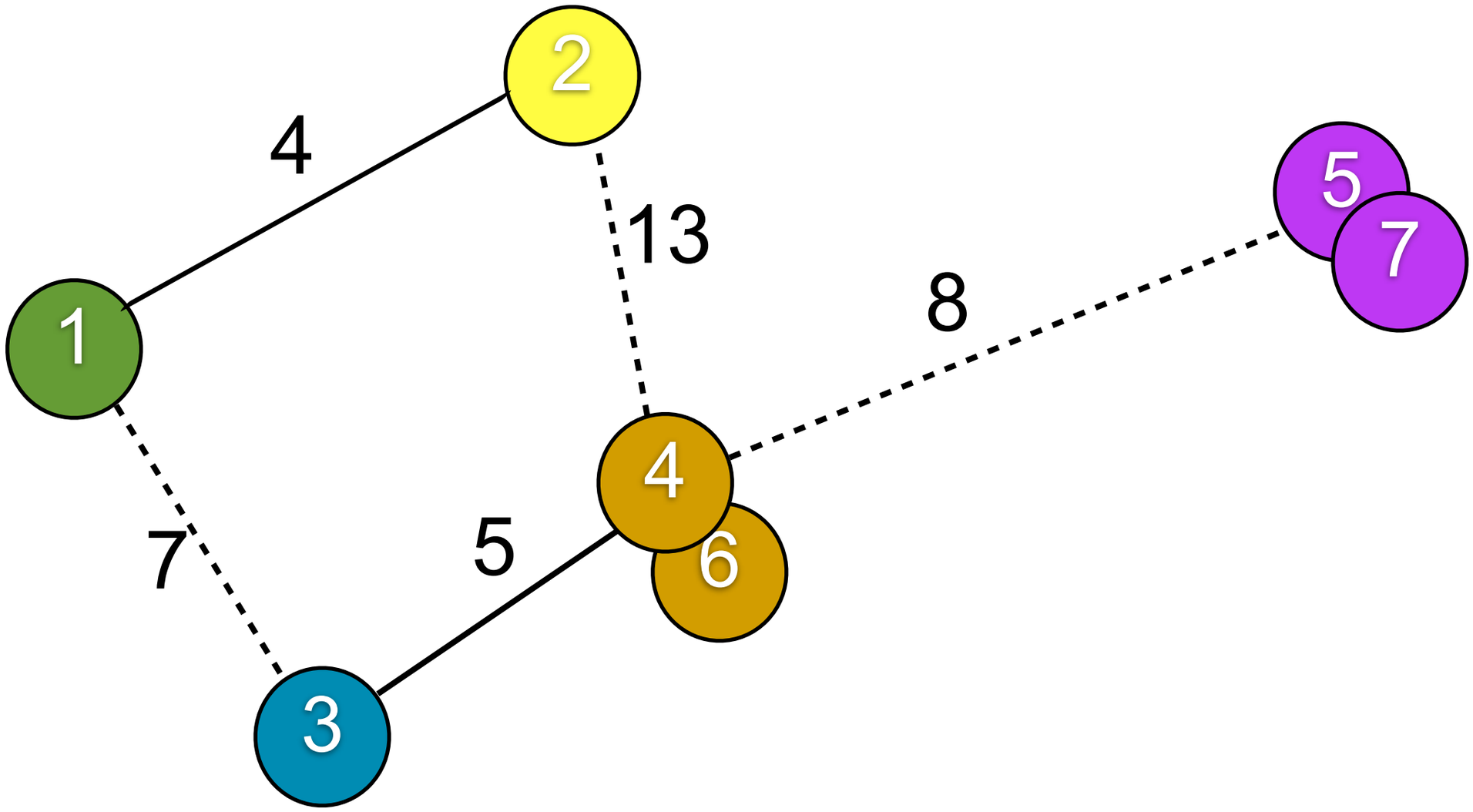
Алгоритм случайного слияния



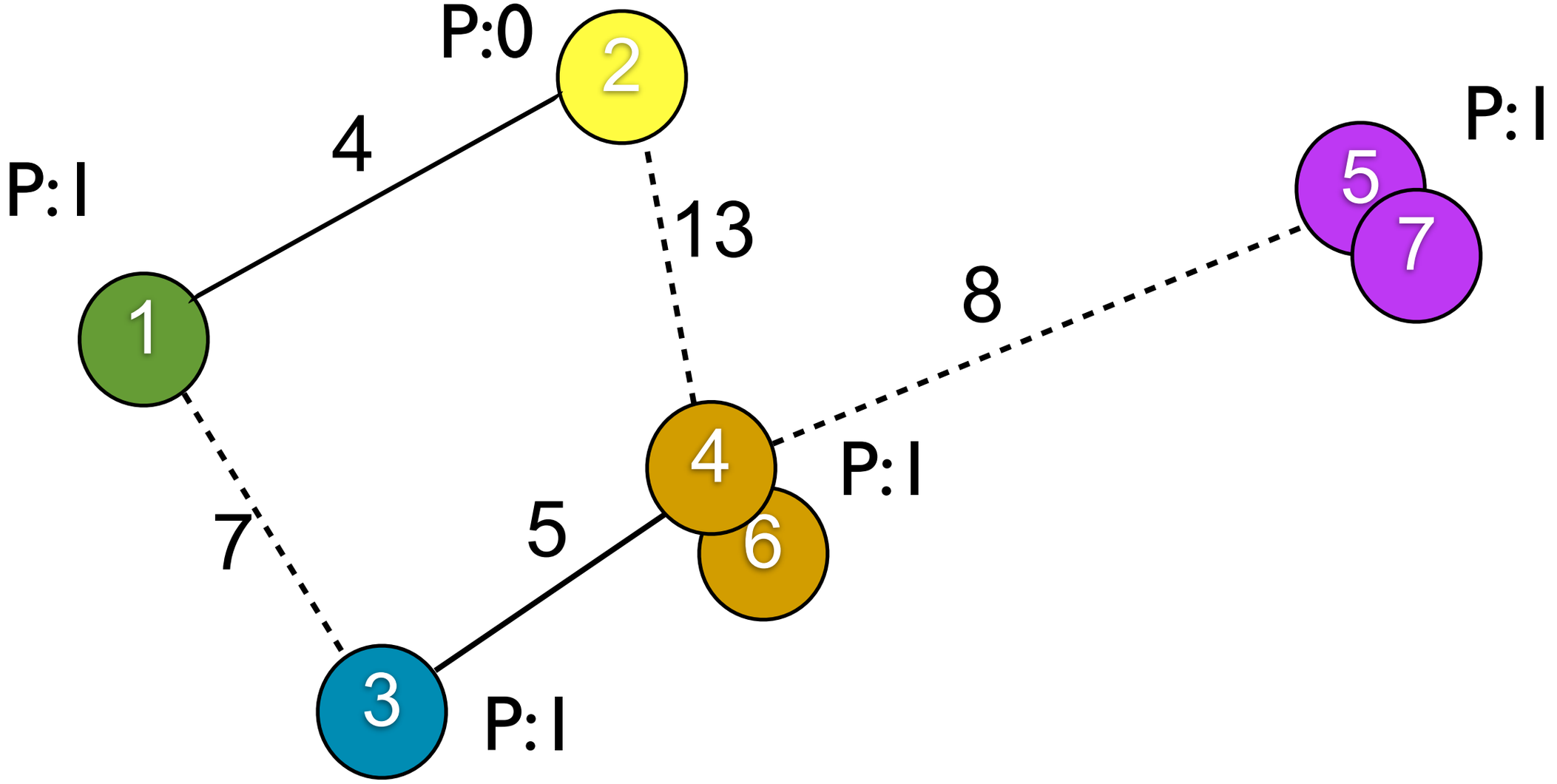
Алгоритм случайного слияния



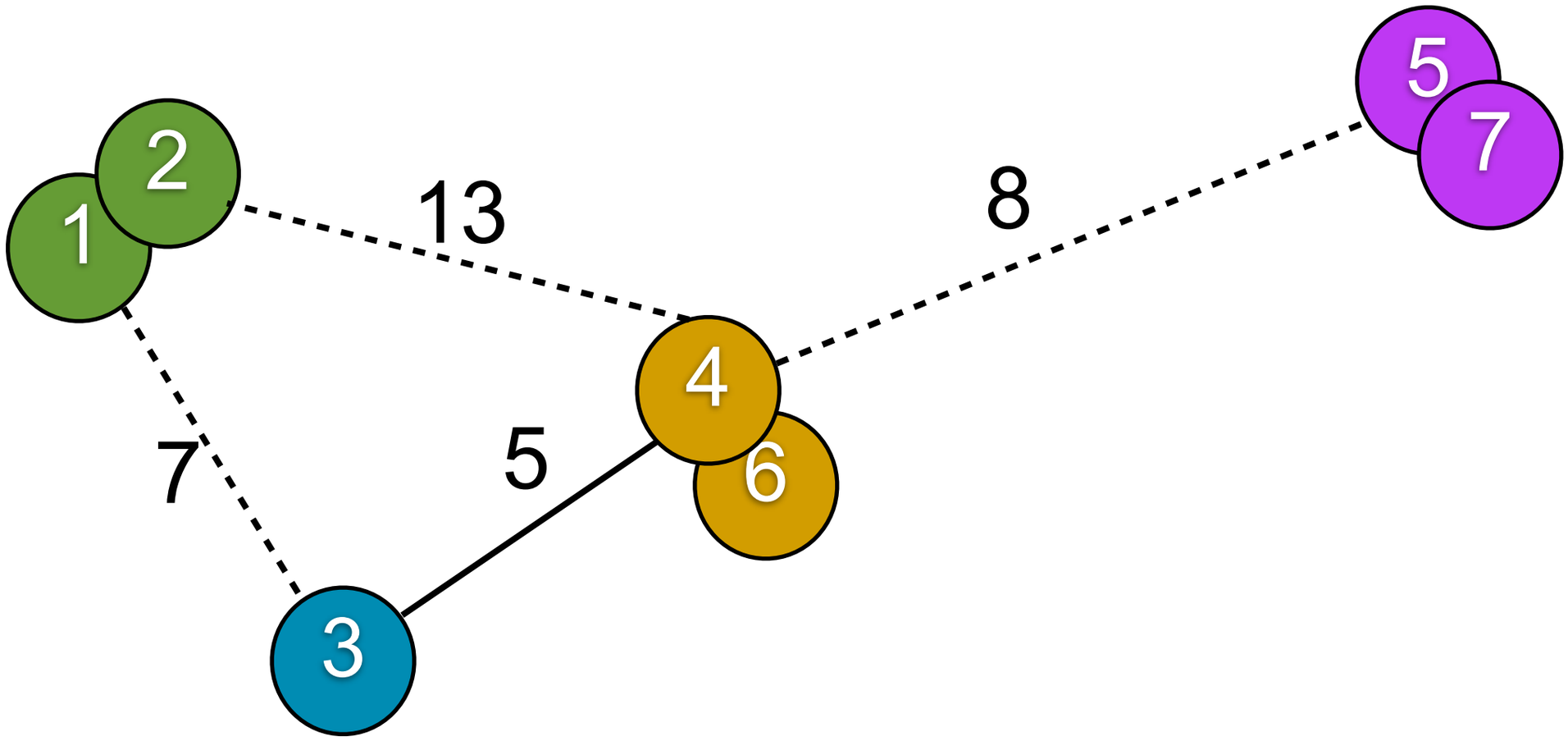
Алгоритм случайного слияния



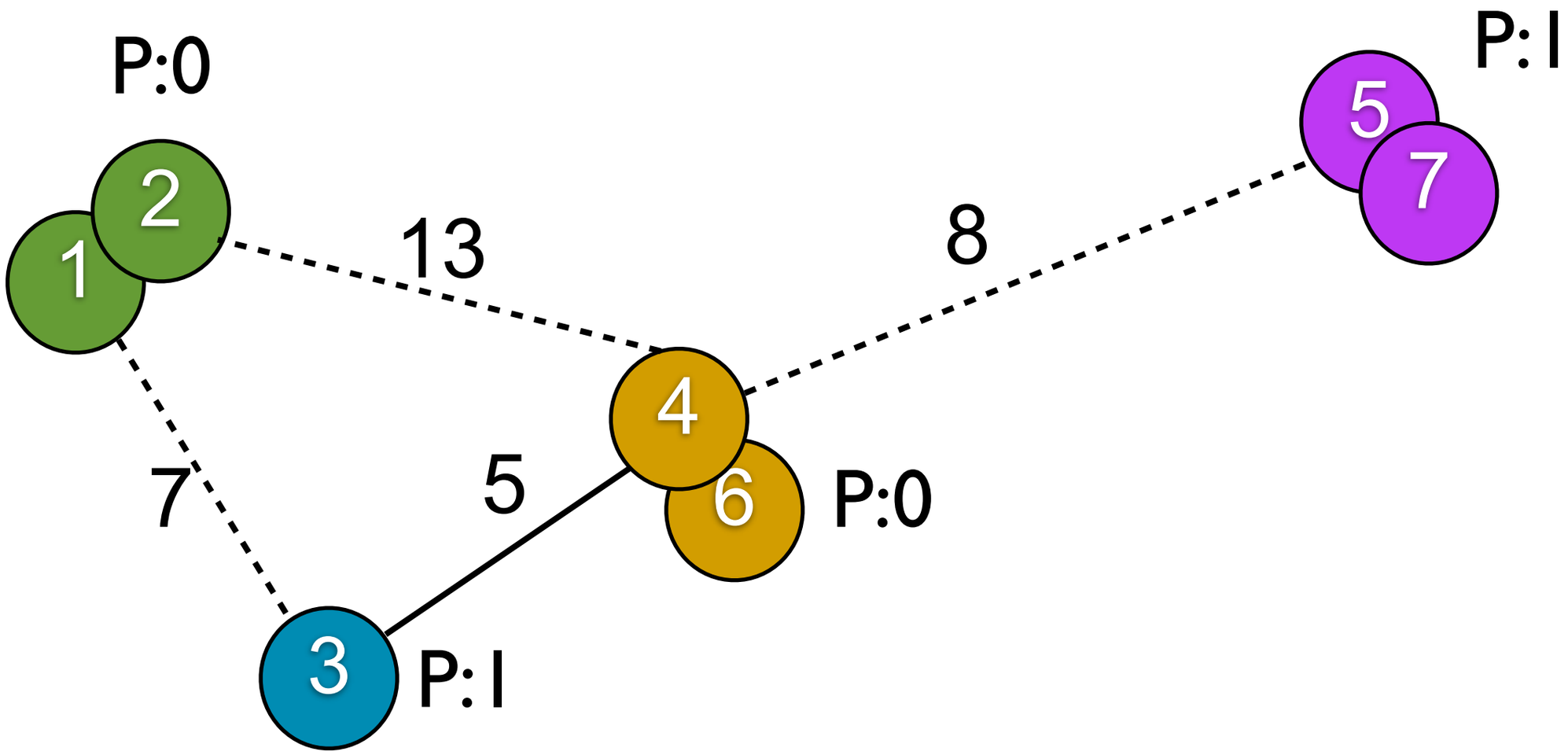
Алгоритм случайного слияния



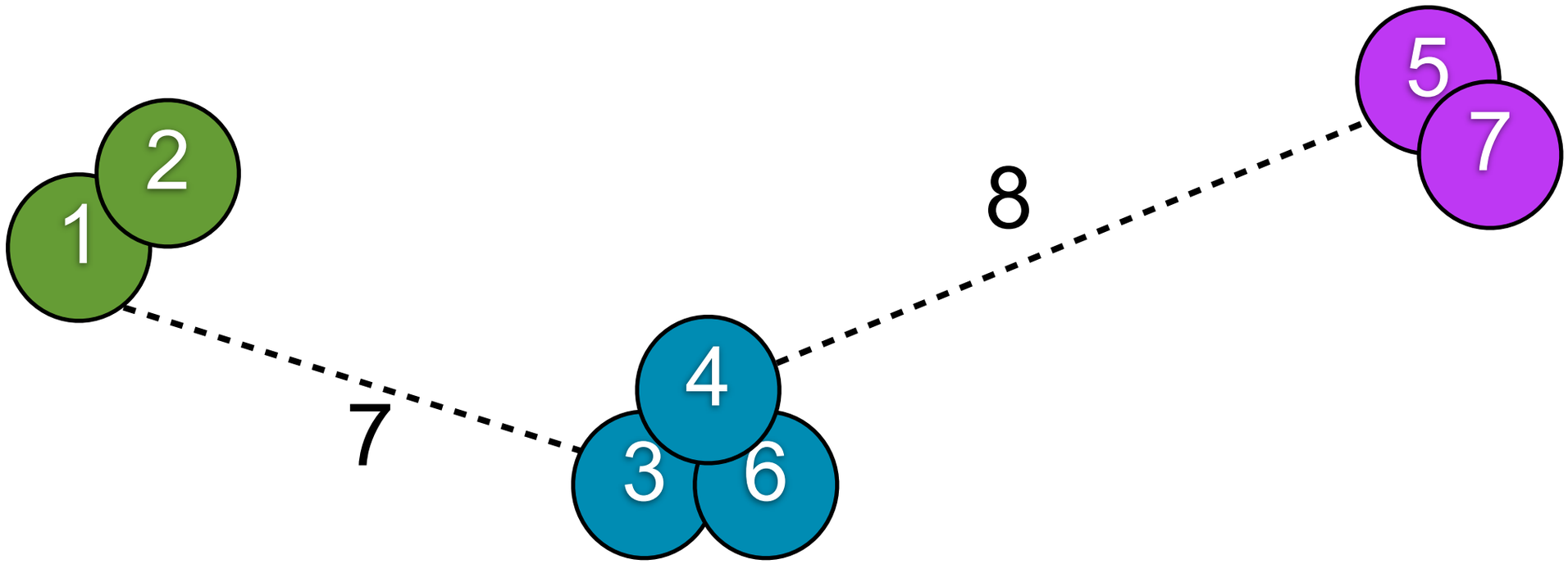
Алгоритм случайного слияния



Алгоритм случайного слияния



Алгоритм случайного слияния



Роман Едемский

Разработчик БК

+380504401948

romanie@yandex-team.ru

Спасибо!