Алгоритмы на графах

Лекция 5.

Бесконтурные графы.

Адигеев Михаил Георгиевич

2023

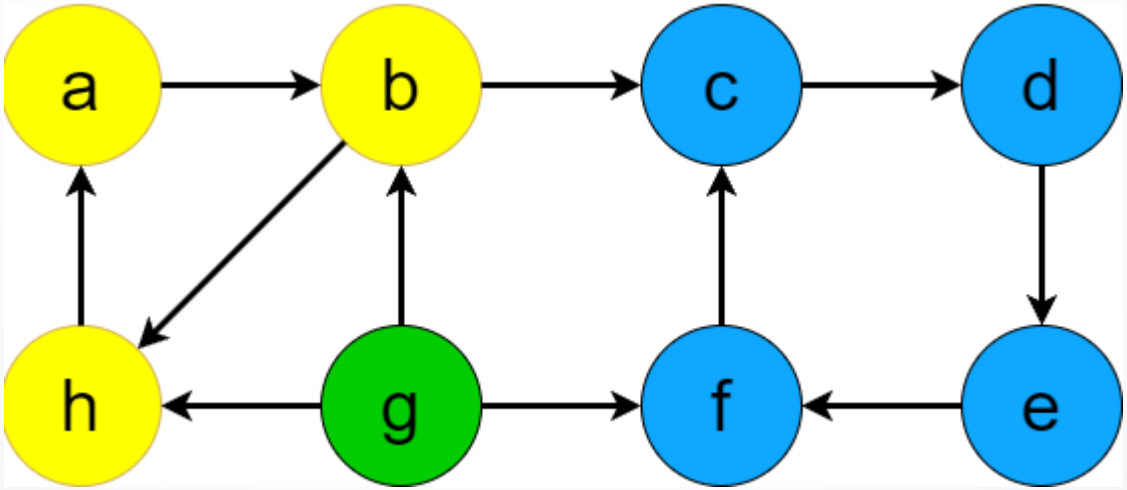
План лекции

1. Сильные компоненты и конденсация графа.
2. Ярусная форма графа.
   - ✓ Связь с бесконтурностью.
   - ✓ Минимальное количество ярусов.
   - ✓ Максимальная мощность яруса.
3. Пути на бесконтурных графах.
   - ✓ Кратчайшие пути.
   - ✓ Длиннейшие пути.
   - ✓ Самые надёжные пути.
   - ✓ Пути максимальной пропускной способности.

Сильные компоненты

Для ориентированных графов вводят понятие *сильной связности*: граф ***сильно связен***, если все пары вершин взаимно достижимы друг из друга.

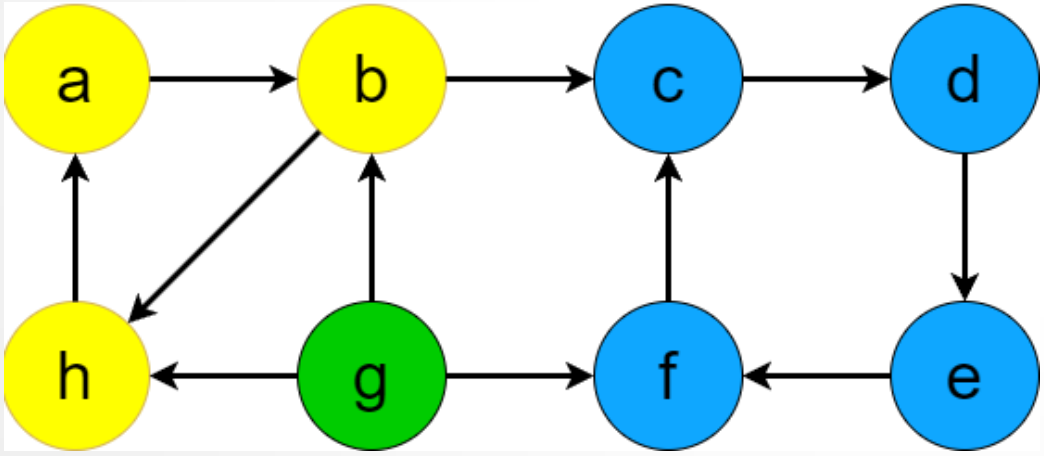
Максимальный сильно связный *подграф* на графе называется ***сильной (сильно связной) компонентой***.

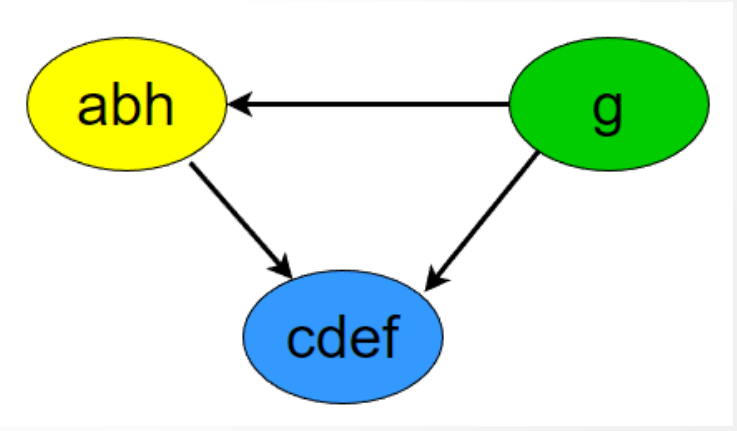


Сильные компоненты

Конденсацией ориентированного графа $G(V, E)$ называется граф $C(V_C, E_C)$:

- Вершины – сильные компоненты графа *G*.
- Из вершины *u* в вершину *v* ведёт дуга тогда и только тогда, когда на *G* существует дуга из какой-то вершины компоненты *u* в какую-то вершину компоненты *v*.



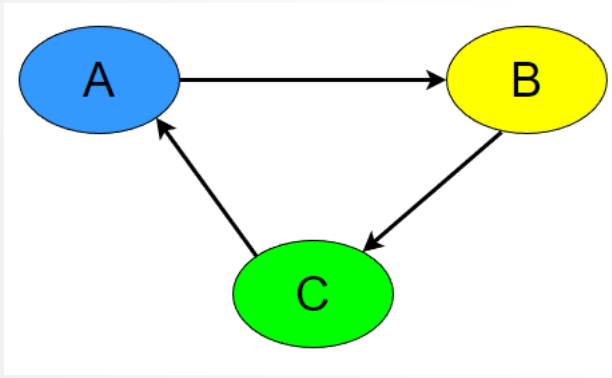


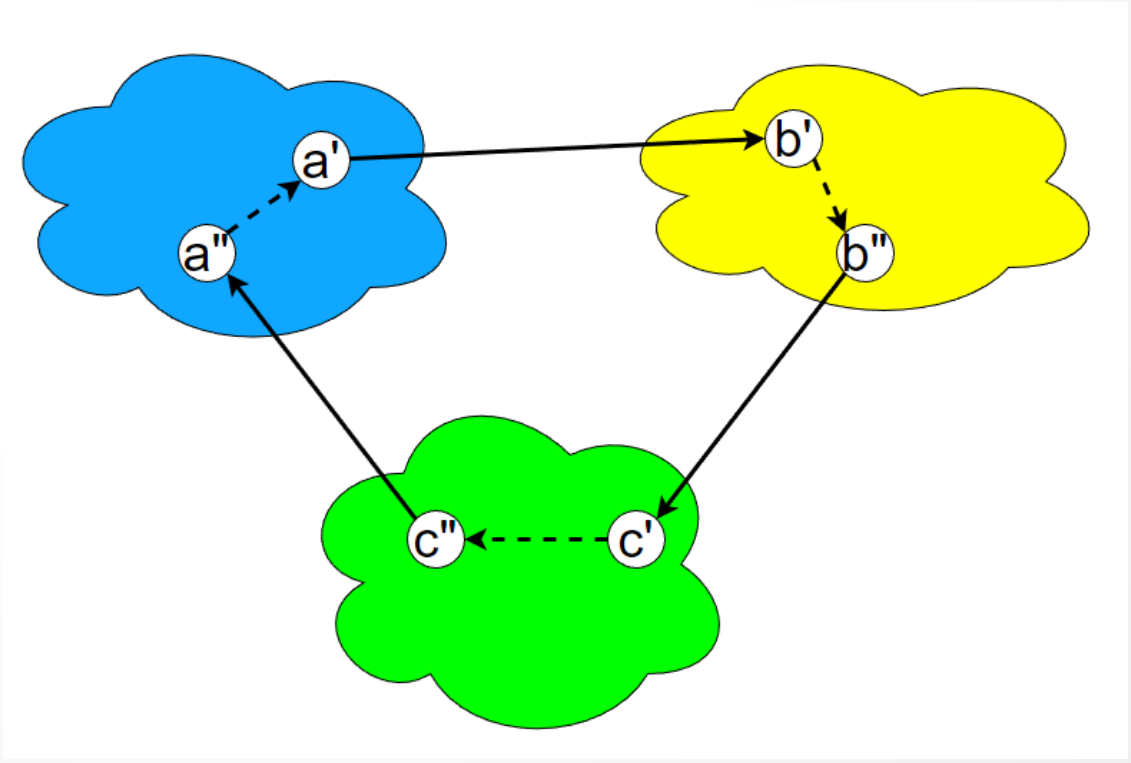
Сильные компоненты

Теорема. Конденсация любого ориентированного графа является бесконтурным графом.

Доказательство.

Допустим, что есть граф $G(V, E)$, у которого конденсация $C(V_C, E_C)$ содержит контур.





Сильные компоненты

Построение сильных компонент с помощью поиска в глубину.

1) Построить «топологическую сортировку» (формально применив алгоритм `DFS_TopSort`). Аналог: пронумеровать вершины в порядке уменьшения `Time_Out`.
2) Инвертировать дуги графа.
3) Выбирать необработанные вершины в порядке «топологических номеров» и запускать обход в глубину (`DFS_TopSort_Visit`). Множество вершин, которые обходятся за один такой обход, образуют сильную компоненту.

Ярусная форма

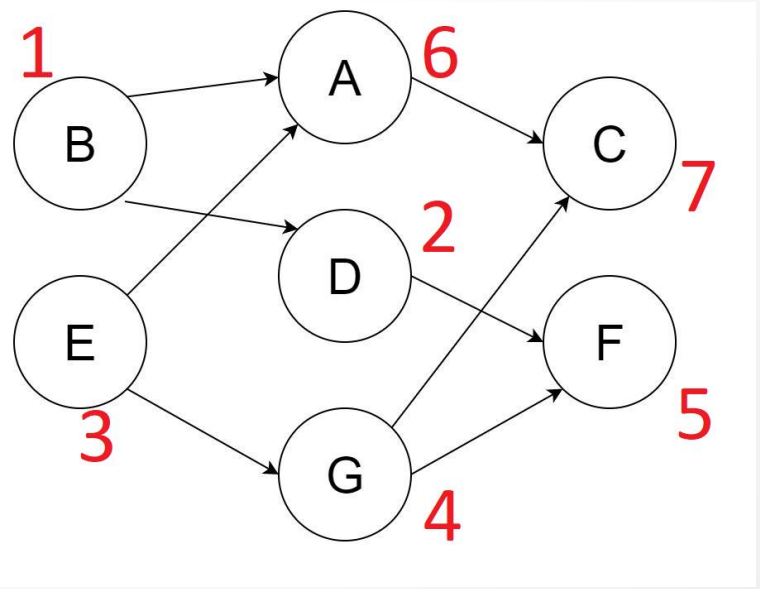
Пусть G(V,E) – ориентированный граф.

Ярусной формой графа разбиение множества вершин на подмножества (*ярусы*, *слои*) $V = \cup V_i$, при котором каждая дуга графа ведёт из яруса с меньшим номером в ярус с большим номером.

Утверждение. Все вершины одного яруса попарно недостижимы друг из друга.

Теорема. Граф имеет ярусную форму тогда и только тогда, когда этот граф является бесконтурным.

Доказательство аналогично теореме о топологической сортировке.



Ярусная форма

Задача: для заданного графа найти ярусную форму с минимальным количеством ярусов.

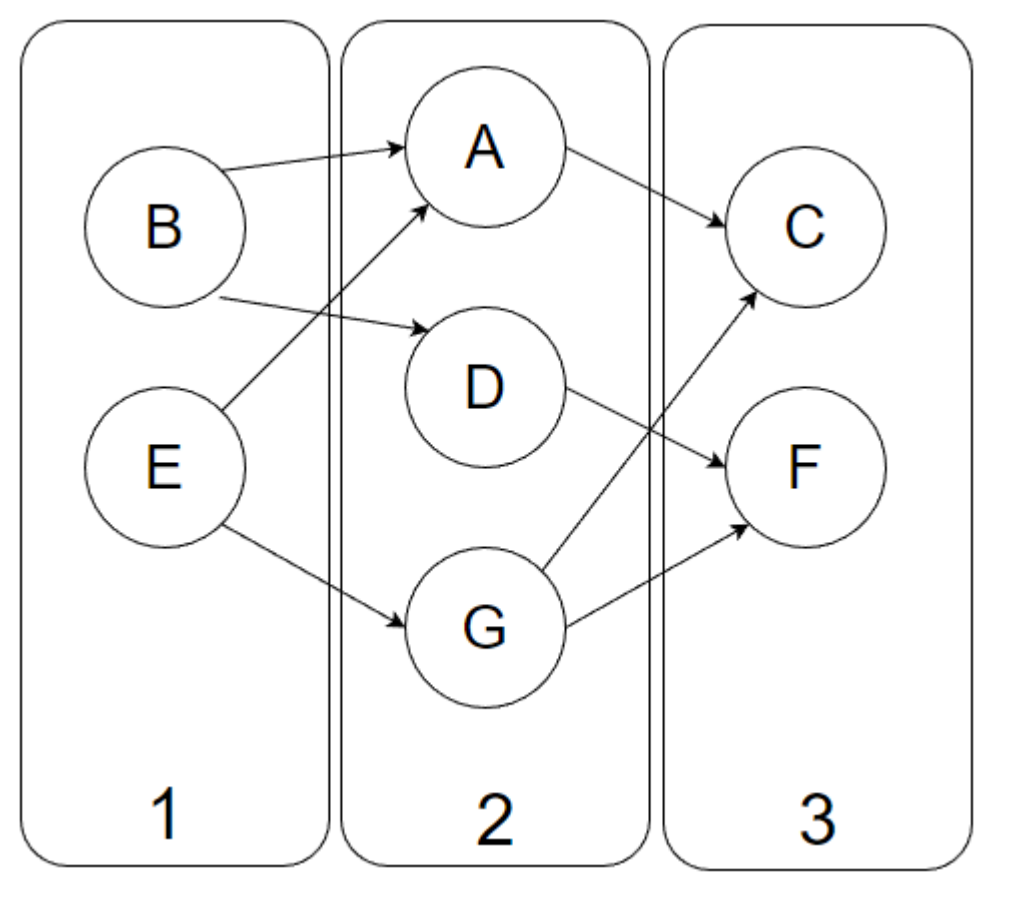
Задача возникает при планировании выполнения задач при возможности некоторые задачи выполнять параллельно.

Минимальное количество ярусов называется ***глубиной*** графа.

Алгоритм минимального ярусного разбиения:

1. Создаём счётчик, инициализируем 1.
2. Пока $|V| > 0$
   - Находим все источники и помещаем их в ярус с очередным номером. Увеличиваем счётчик.
   - Удаляем все источники из графа.

Ярусная форма



Ярусная форма

Как доказать, что этот алгоритм строит *минимальную* ярусную форму?

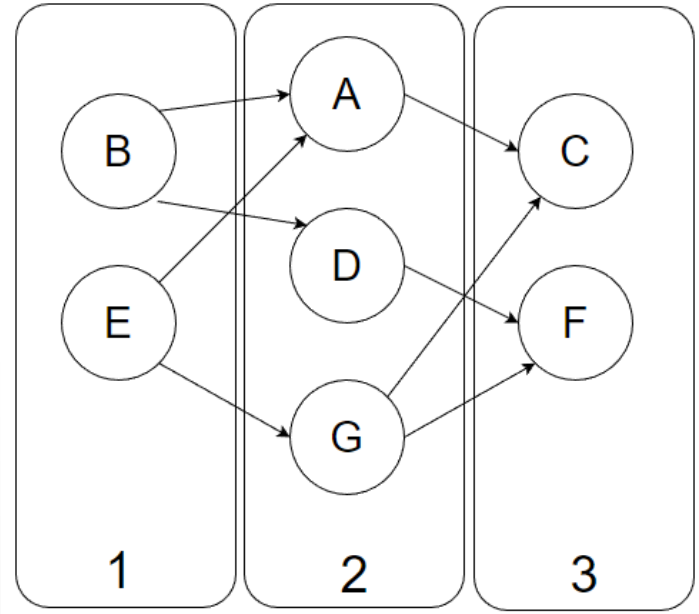
Определение. ***Критическим путём*** на (бесконтурном) графе называется самый длинный путь (=содержащий максимальное количество дуг).

Теорема. Минимальное количество ярусов равно длине критического пути + 1 (= количеству вершин в критическом пути).

Доказательство.

1) Количество ярусов не может быть меньше, чем количество вершин в критическом пути.

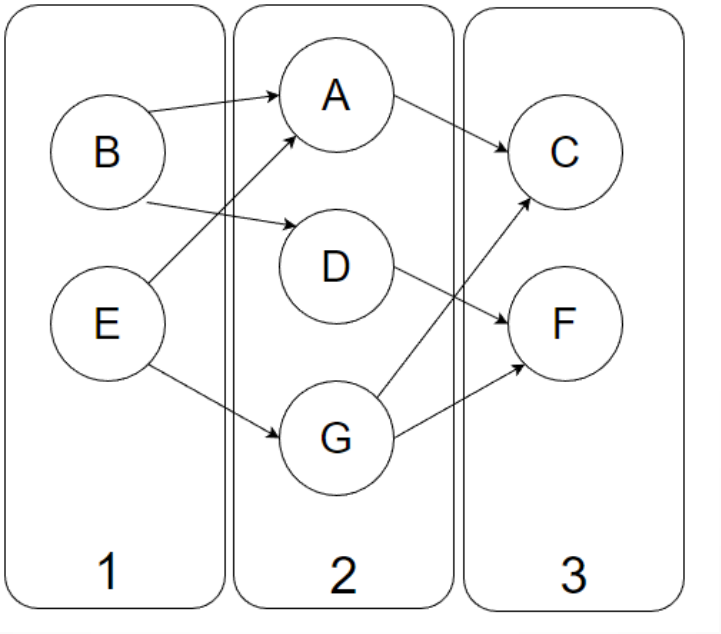
2) Приведённый алгоритм строит ярусное представление с количеством ярусов = количеству вершин критического пути.



Ярусная форма

Теорема. Минимальное количество путей, *покрывающих* все вершины графа, равно максимальной мощности яруса.

Доказательство аналогично предыдущей теореме.



Пути в бесконтурных графах

Рассмотрим несколько похожих задач, связанных с нахождением оптимальных путей на *взвешенных* графах. В данной лекции рассматриваем частный случай этих задач: предполагаем, что граф является бесконтурным.

Определение. Граф $G(V, E)$, $w: E \to R_+$ называется ***взвешенным***, если задана функция веса/стоимости дуг $w: E \to R$.

Случай отрицательных весов иногда рассматриваются отдельно.

Взвешенный граф удобно представлять в виде матрицы весов *W*.

Аналогична матрице смежности, но w_{ij} равен весу дуги (i, j). Если такой дуги нет, то присваиваем значение, зависящее от решаемой задачи.

Пути в бесконтурных графах

Задача о кратчайшем пути

Вес пути определим как *сумму* весов входящих в него дуг.

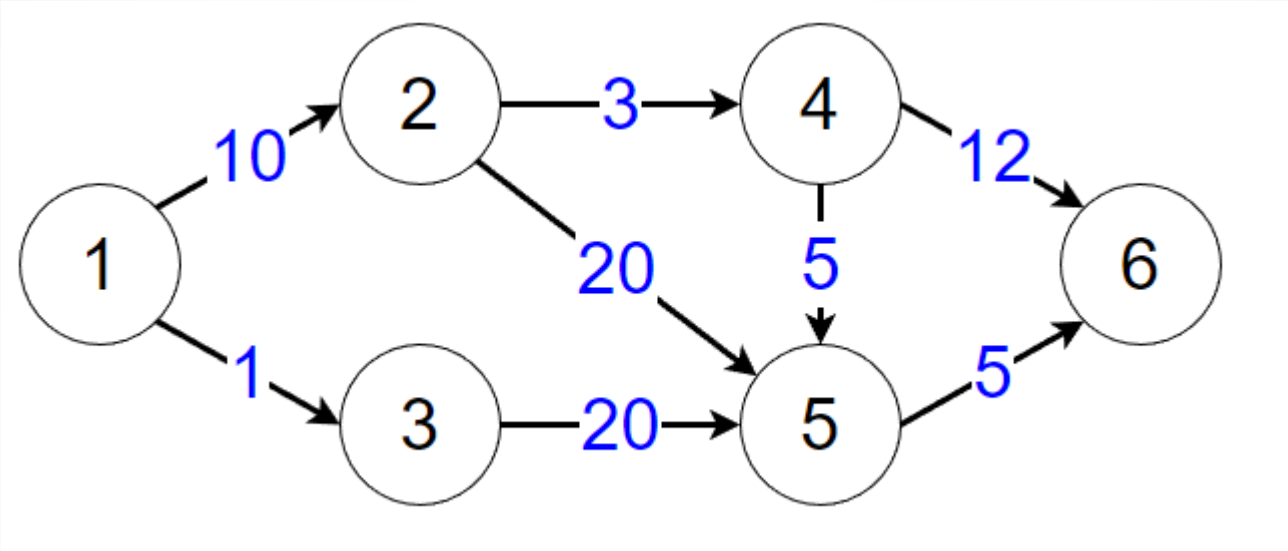
Кратчайший путь = путь минимального веса.

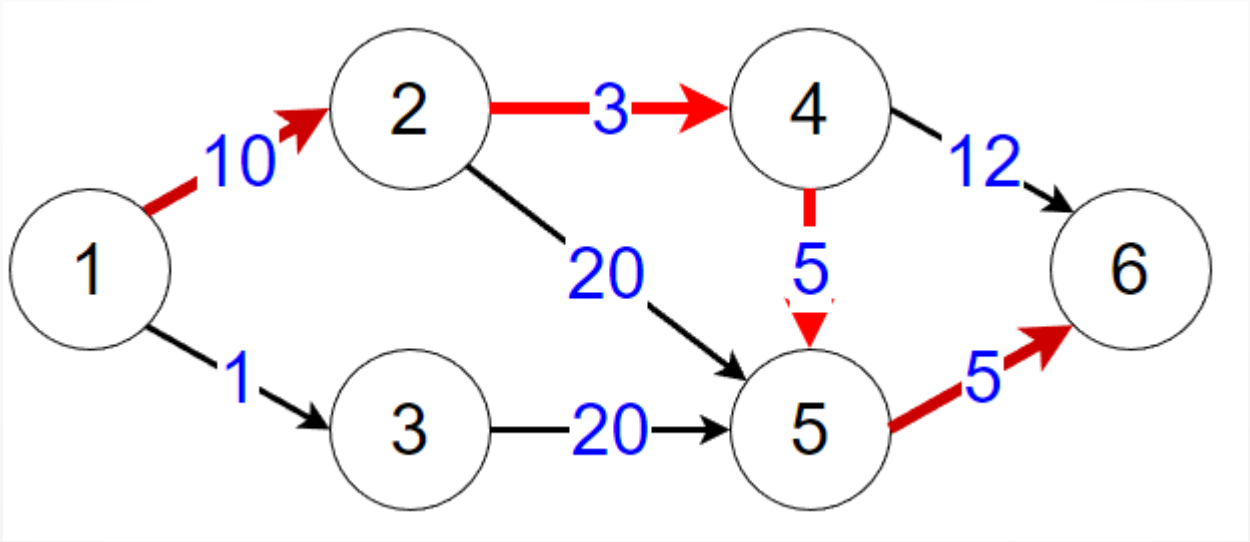
Возможны три формулировки задачи:

1) Заданы вершины $s, t \in V$. Найти кратчайший путь из s в t.
2) Задана вершина $s \in V$. Найти кратчайшие пути из s во все остальные вершины графа.
3) Найти кратчайшие пути для всех пар вершин на графе.

Рассмотрим алгоритм, решающий варианты (1) и (2).

Пути в бесконтурных графах





Пути в бесконтурных графах

Алгоритм нахождения кратчайших путей

Вход: матрица весов W. При отсутствии дуги (i, j) полагаем $w_{ij} = +\infty$.

Выход:

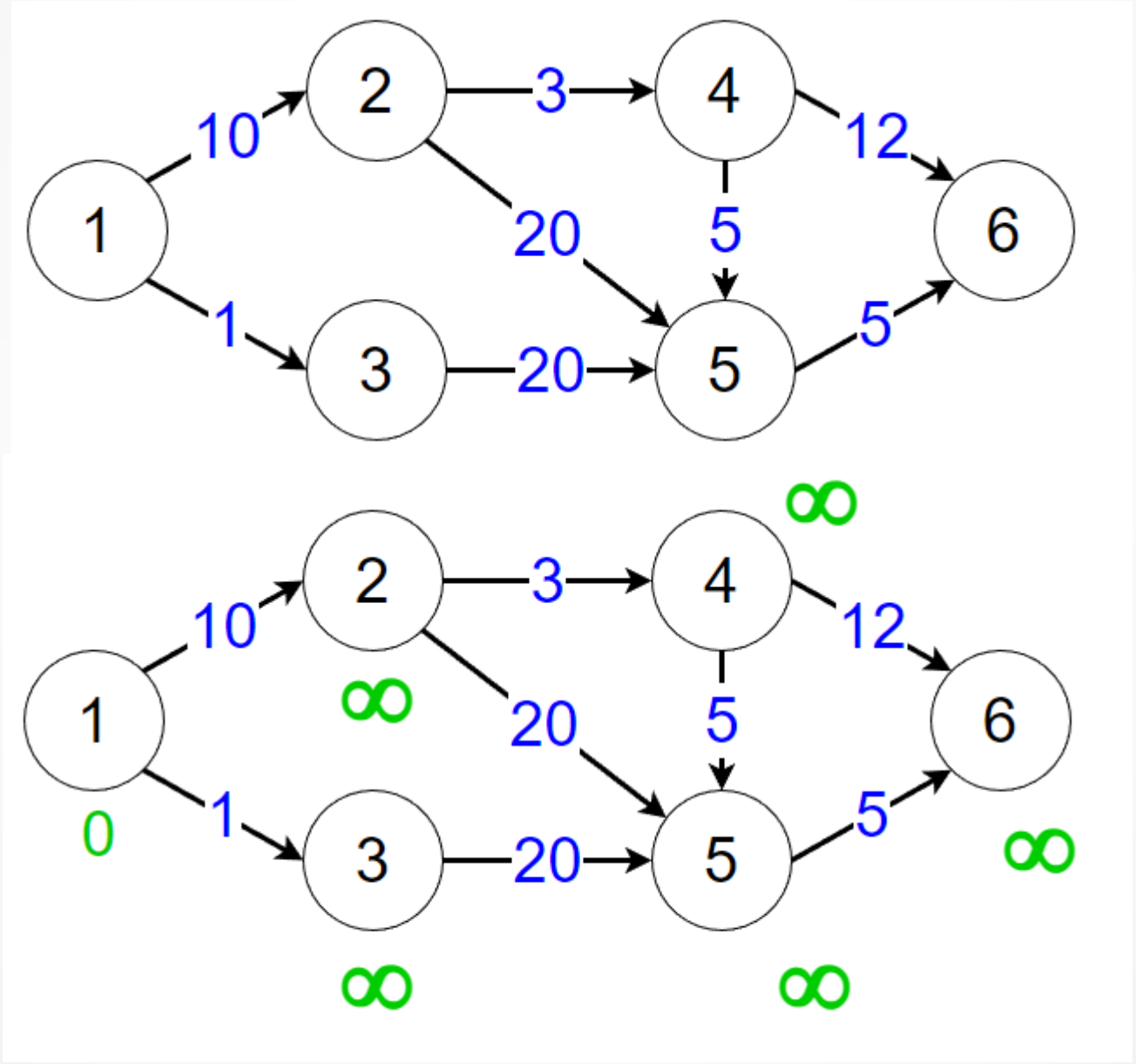
1) Массив D: $d[i]$=расстоянию (длине кратчайшего пути) от s до i.
2) Массив P: $p[i]$=номер вершины, которая является предпоследней на кратчайшем пути из s в i.

Пути в бесконтурных графах

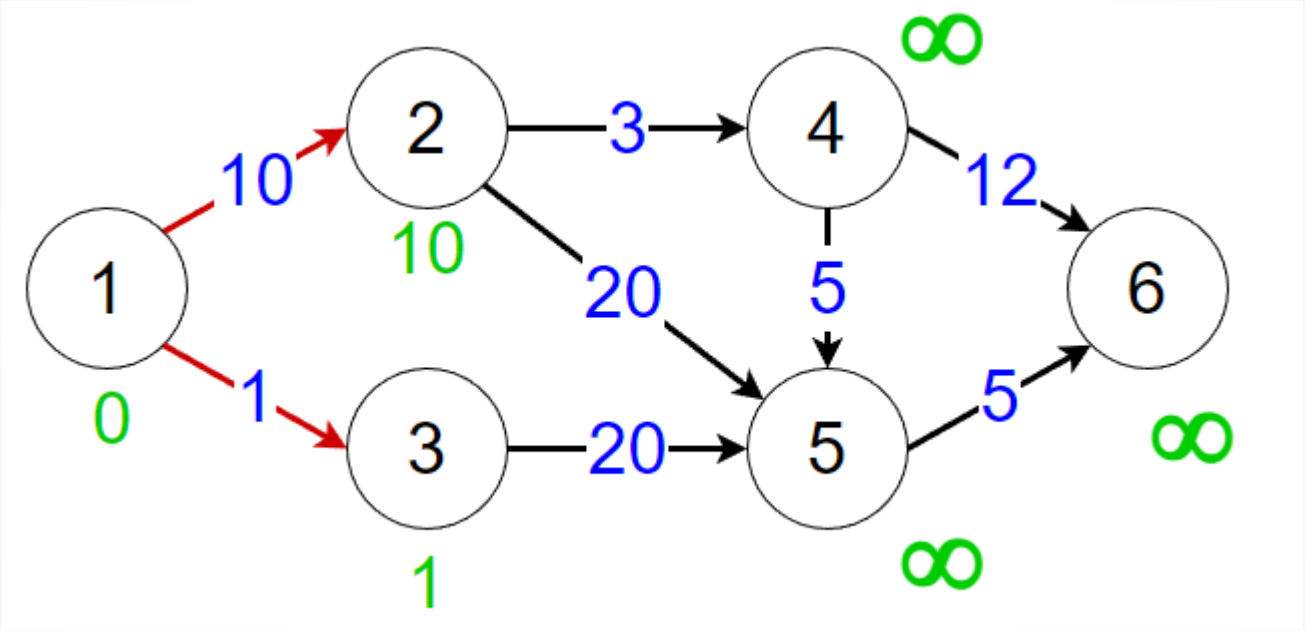
Алгоритм нахождения кратчайших путей

```
For each v ∈ V:
      {
            d[i] := +∞;
            p[i] := NULL;
      }
d[s] := 0;
Построить топологическую сортировку вершин графа;
For each v ∈ V в порядке топологических номеров:
      For each u in Adj(v)
            If d[u] > d[v] + w[v,u]:
            {
                  d[u] := d[v] + w[v,u];
                  p[u] := v;
            }
```

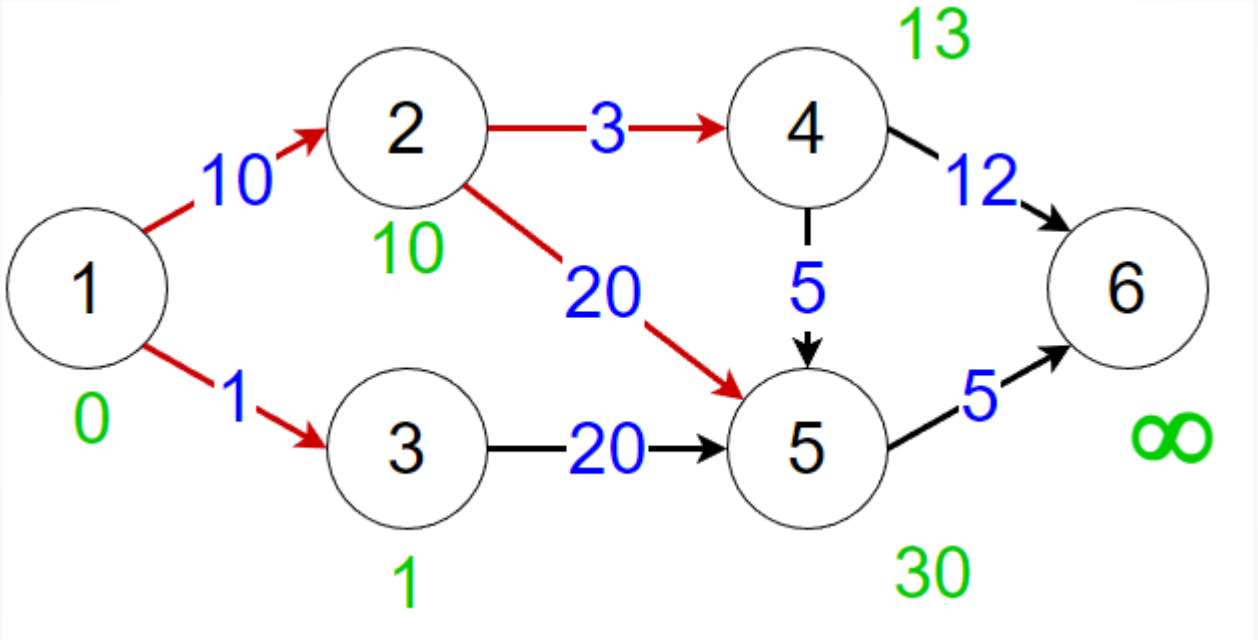
Пути в бесконтурных графах



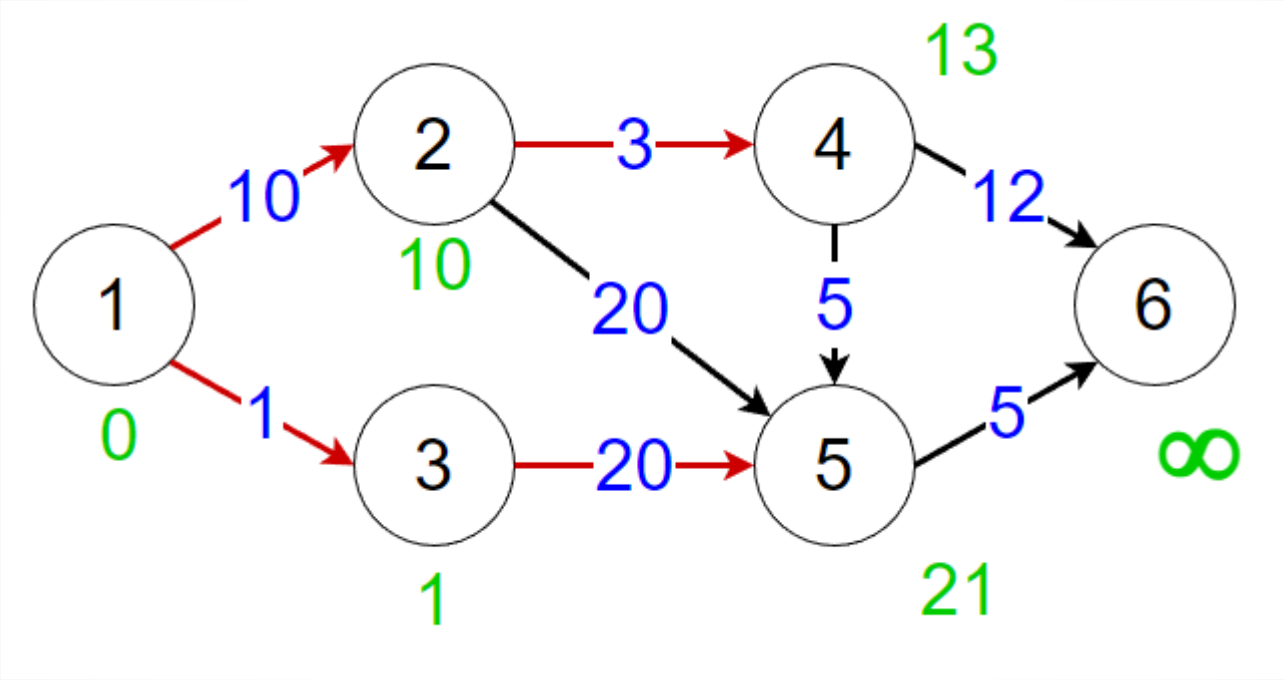
Пути в бесконтурных графах



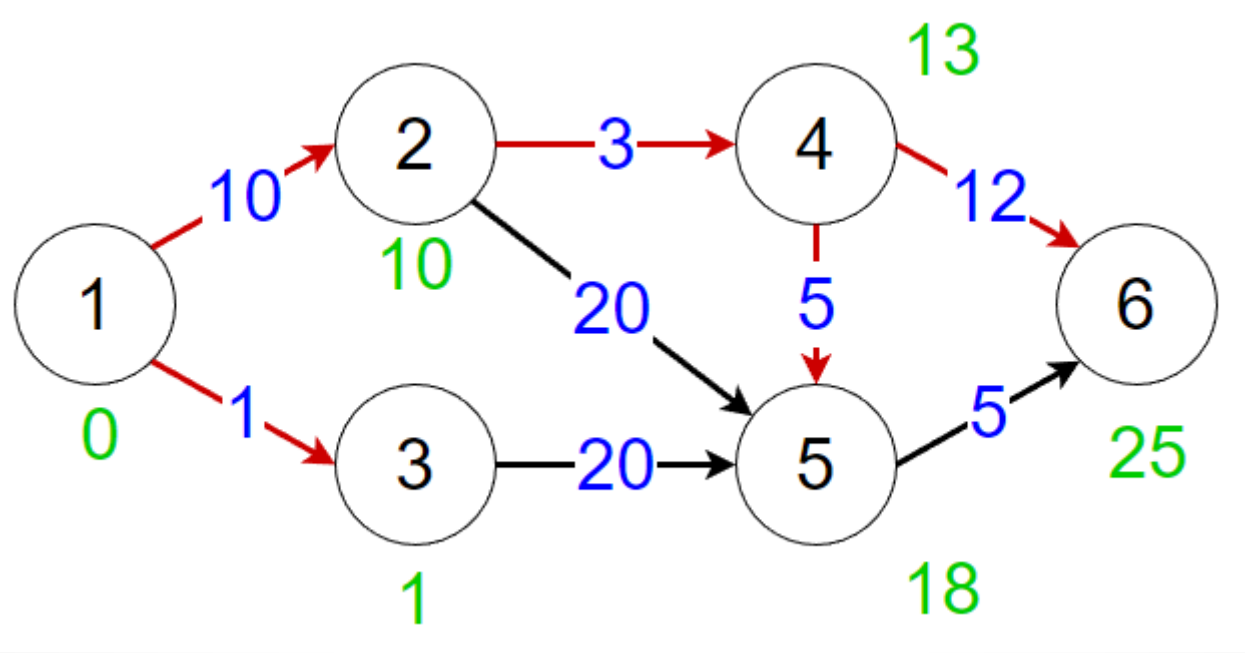
Пути в бесконтурных графах



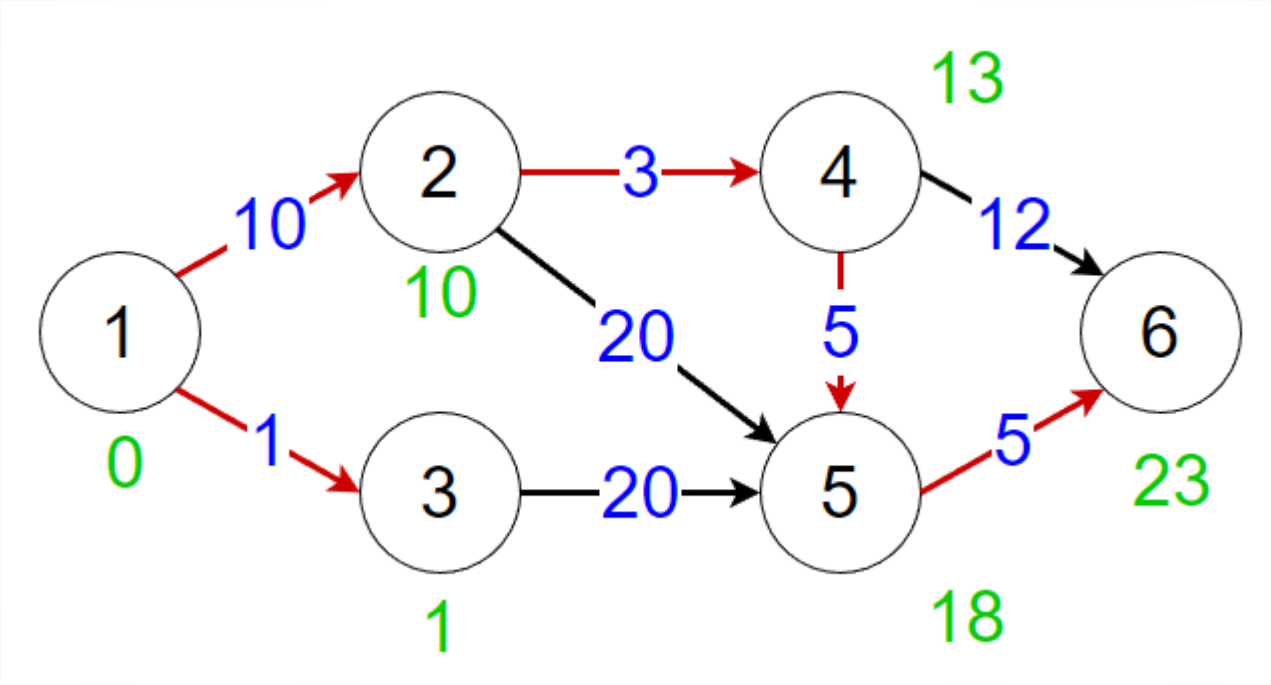
Пути в бесконтурных графах



Пути в бесконтурных графах



Пути в бесконтурных графах



Пути в бесконтурных графах

Сложность алгоритма: $O(n+m)$.

Как обосновать корректность этого алгоритма?

Теорема. После окончания данного алгоритма для всех $v \in V$ значение $d[v]$ равно расстоянию от s до v.

Доказательство

Применим метод полной математической индукции. Параметр индукции – топологический номер (ТН) вершины.

Базис: ТН=0 – это справедливо только для вершины *s*. Для неё $d[s] = 0$ на этапе инициализации, и больше это значение не меняется.

Предположение: для всех вершин с ТН < *k* утверждение справедливо.

Переход: выберем вершину *u*: ТН(u)=k.

Пути в бесконтурных графах

Переход: выберем вершину *u*: TH(u)=k.

Поскольку граф бесконтурный, для *u* существует определённый путь минимального веса. Пусть это путь $\pi = s\text{-}a\text{-}b\text{-}\ldots\text{-}v\text{-}u$.

Для всех вершин этого пути TH < *k* => для них выполняется индуктивное предположение: *d*[*x*] = минимальному весу пути от *s* до *x*.

После обработки вершины *v* выполняется условие: d[u] <= d[v]+w[v,u].

Но d[v]+w[v,u] – длина кратчайшего пути π. Поэтому d[u] = минимальному весу пути от *s* до *u*. И при дальнейших итерациях эта величина уже не изменится.

Пути в бесконтурных графах

Задача о самом длинном пути

Вес пути определим как *сумму* весов входящих в него дуг.

Самый длинный путь = путь *максимального* веса.

Варианты решения:

- Свести к задаче о кратчайшем пути, изменив веса (умножив все веса на -1).
- Модифицировать алгоритм.

Пути в бесконтурных графах

Алгоритм нахождения самых длинных путей

```
For each v ∈ V:
      {
            d[i] := −∞;
            p[i] := NULL;
      }
d[s] := 0;
Построить топологическую сортировку вершин графа;
For each v ∈ V в порядке топологических номеров:
      For each u in Adj(v)
            If d[u] < d[v] + w[v,u]:
            {
                  d[u] := d[v] + w[v,u];
                  p[u] := v;
            }
```

Пути в бесконтурных графах

Задача о самом надёжном пути

Вес дуги характеризует вероятность успешного прохождения по этой дуге.

Вес пути определим как *произведение* весов входящих в него дуг.

Самый надёжный путь = путь *максимального* веса.

Варианты решения:

- Модифицировать алгоритм.

Пути в бесконтурных графах

Алгоритм нахождения самых надёжных путей

```
For each v ∈ V:
      {
            d[i] := 0;
            p[i] := NULL;
      }
d[s] := 1;
Построить топологическую сортировку вершин графа;
For each v ∈ V в порядке топологических номеров:
      For each u in Adj(v)
            If d[u] < d[v] * w[v,u]:
            {
                  d[u] := d[v] * w[v,u];
                  p[u] := v;
            }
```

Пути в бесконтурных графах

Задача о пути максимальной пропускной способности

Вес дуги характеризует её пропускную способность.

Вес пути определим как *минимум* весов входящих в него дуг.

Задача: найти путь *максимального* веса.

Варианты решения:

- Модифицировать алгоритм.

Пути в бесконтурных графах

Алгоритм нахождения путей максимальной пропускной способности

```
For each v ∈ V:
      {
            d[i] := 0;
            p[i] := NULL;
      }
d[s] := +∞;
Построить топологическую сортировку вершин графа;
For each v ∈ V в порядке топологических номеров:
      For each u in Adj(v)
            If d[u] < min{d[v],w[v,u]}:
            {
                  d[u] := min{d[v],w[v,u]};
                  p[u] := v;
            }
```