

Алгоритмы и структуры данных

Лекция 4: Декомпозиция графов

А. Куликов

Академия современного программирования

План лекции

- 1 Поиск в глубину в ненаправленных графах
 - Связность

План лекции

- 1 Поиск в глубину в ненаправленных графах
 - Связность
- 2 Поиск в глубину в направленных графах
 - Типы рёбер
 - Ациклические графы
 - Топологическая сортировка

План лекции

- 1 Поиск в глубину в ненаправленных графах
 - Связность
- 2 Поиск в глубину в направленных графах
 - Типы рёбер
 - Ациклические графы
 - Топологическая сортировка
- 3 Сильно связанные компоненты

План лекции

- 1 Поиск в глубину в ненаправленных графах
 - Связность
- 2 Поиск в глубину в направленных графах
 - Типы рёбер
 - Ациклические графы
 - Топологическая сортировка
- 3 Сильно связанные компоненты
- 4 Упражнения

План лекции

1 Поиск в глубину в ненаправленных графах

- Связность

2 Поиск в глубину в направленных графах

- Типы рёбер
- Ациклические графы
- Топологическая сортировка

3 Сильно связанные компоненты

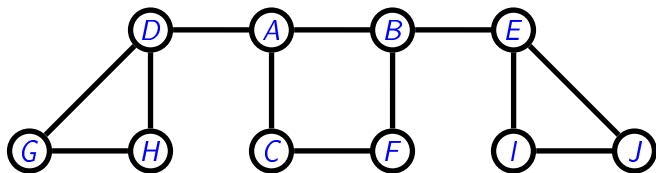
4 Упражнения

Нахождение всех вершин, достижимых из данной

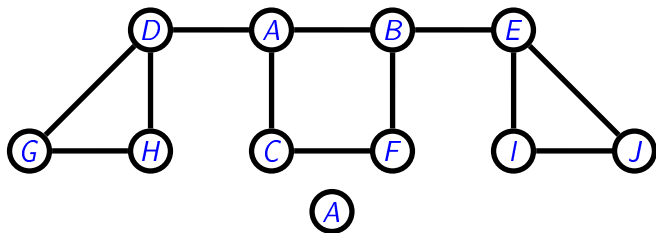
EXPLORE(G, v)

- 1 ▷ Вход: граф $G = (V, E)$ и его вершина $v \in V$.
- 2 ▷ Выход: $visited[u] = \text{TRUE}$ для всех вершин u , достижимых из v
- 3 $visited(v) = \text{TRUE}$
- 4 PREVISIT(v)
- 5 for каждого ребра $(v, u) \in E$
- 6 do if $visited(u) = \text{FALSE}$
- 7 then EXPLORE(u)
- 8 POSTVISIT(v)

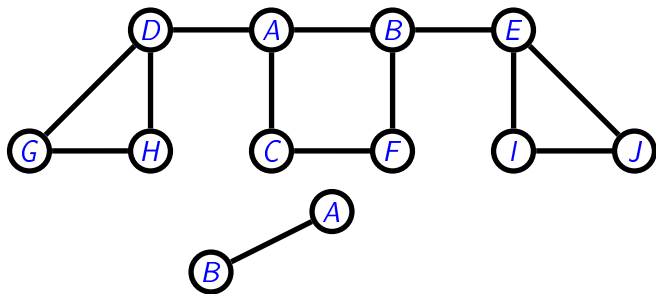
Пример обхода



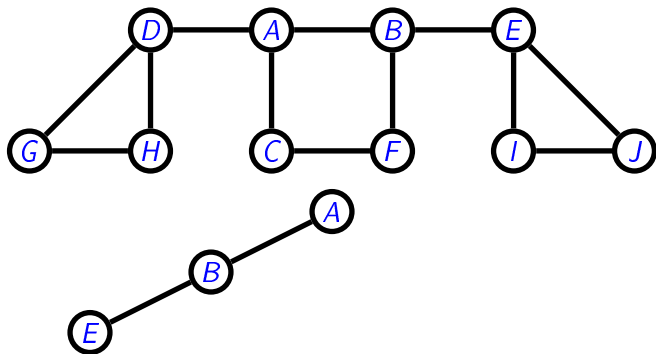
Пример обхода



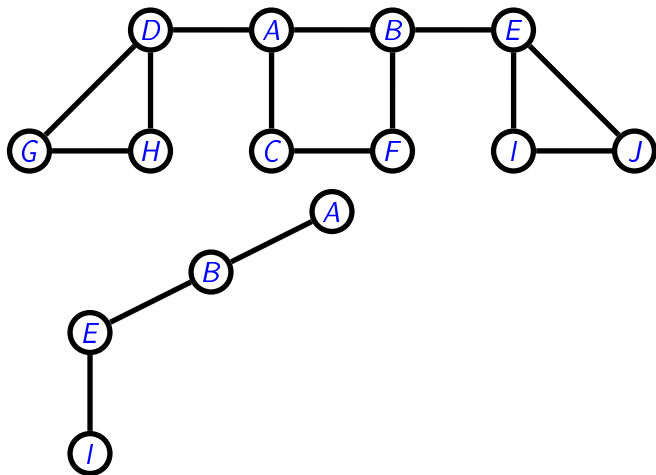
Пример обхода



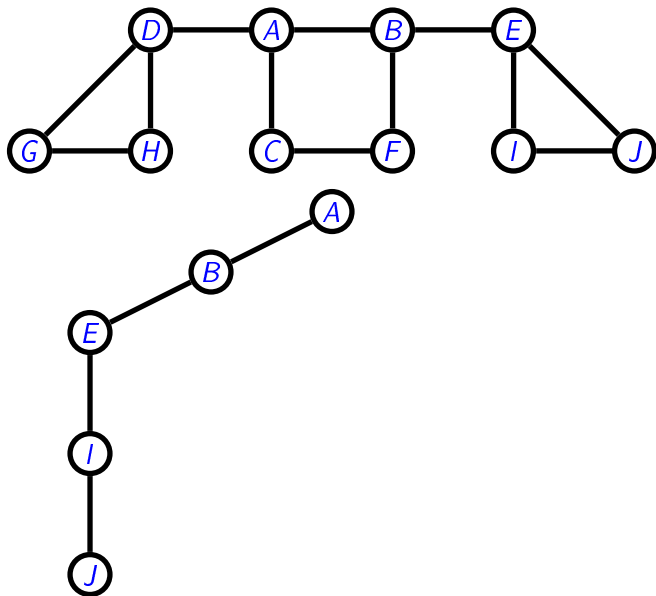
Пример обхода



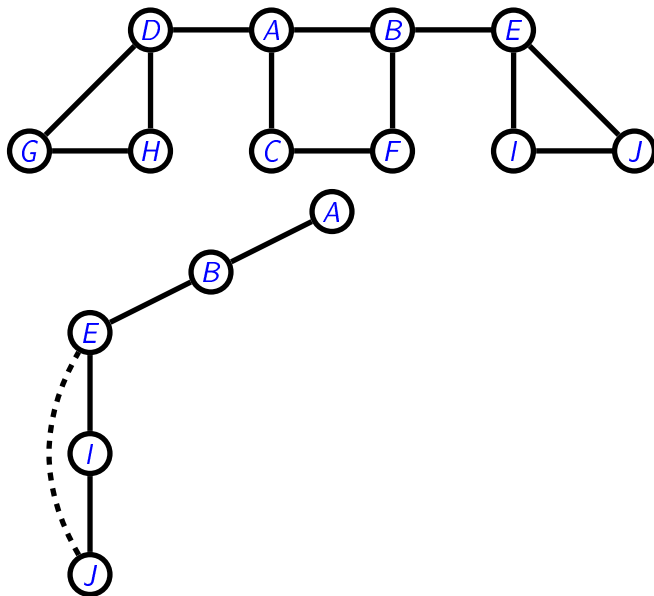
Пример обхода



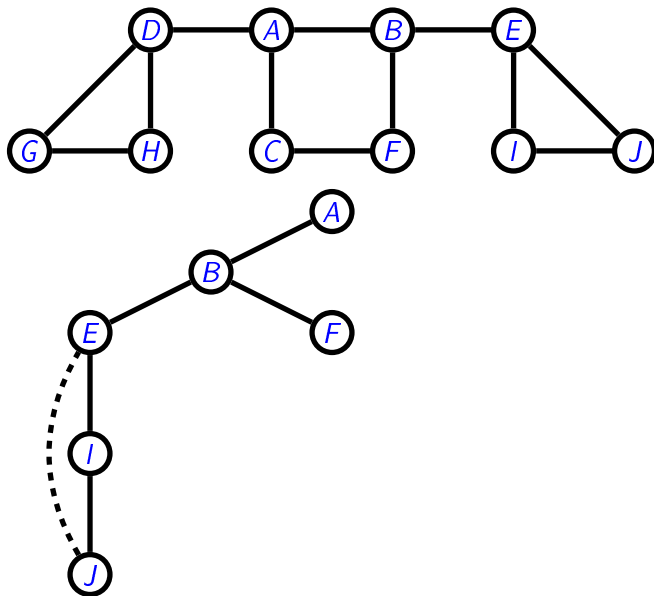
Пример обхода



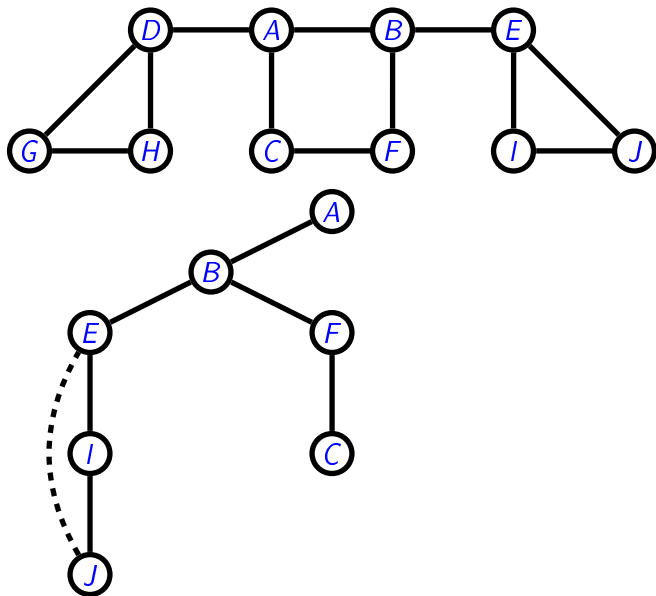
Пример обхода



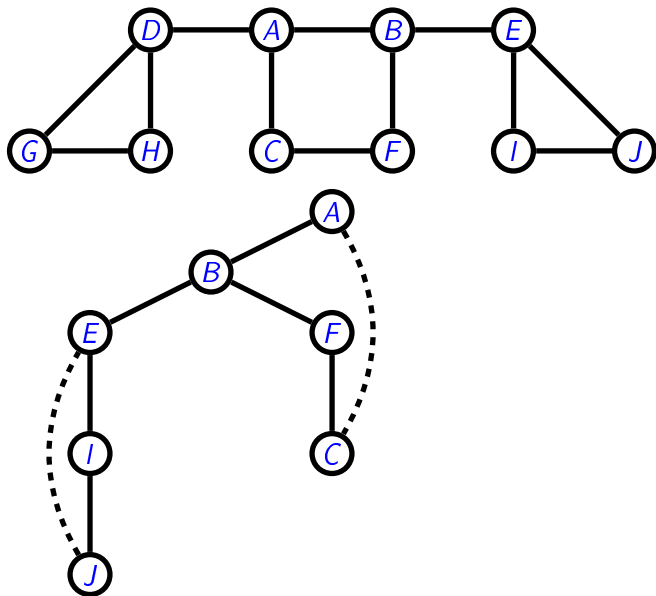
Пример обхода



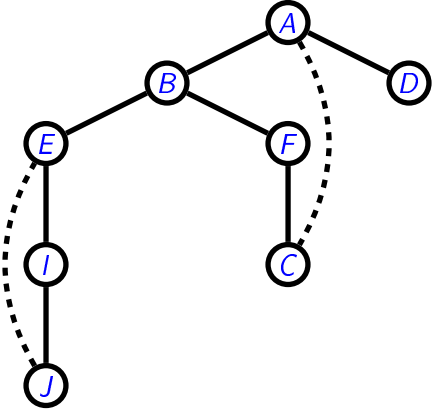
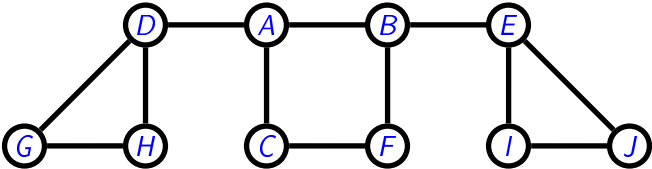
Пример обхода



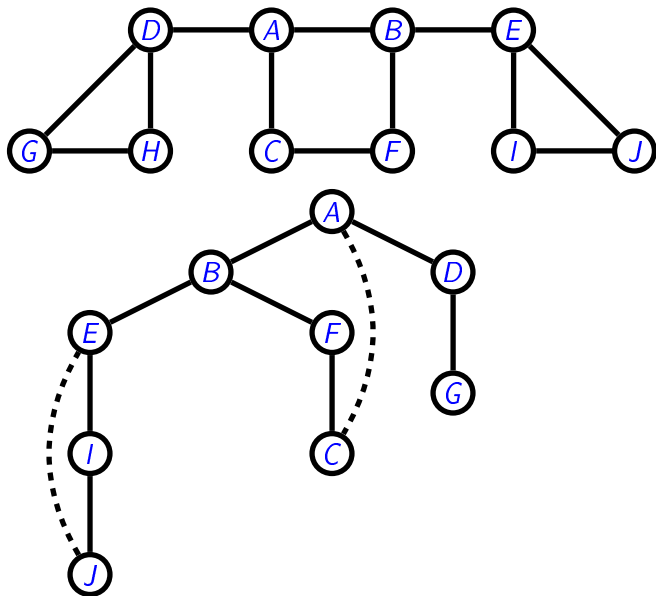
Пример обхода



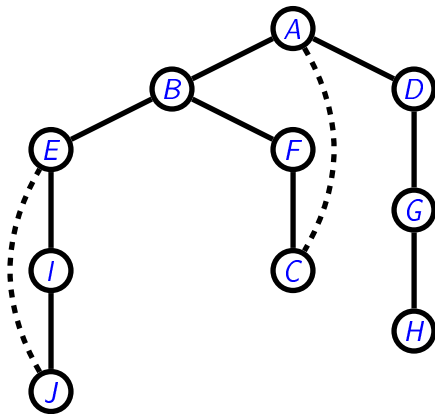
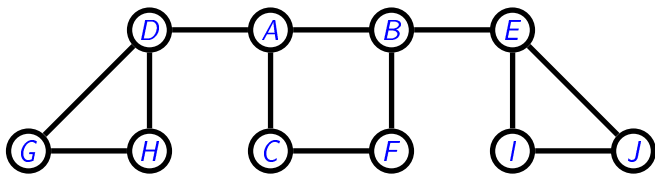
Пример обхода



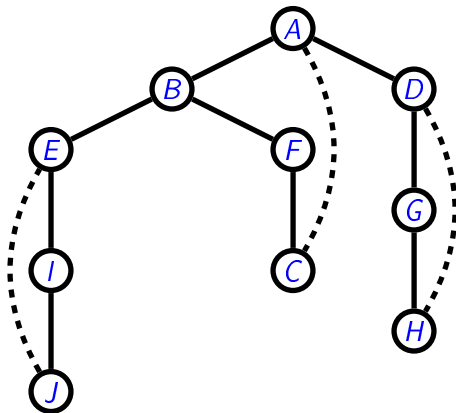
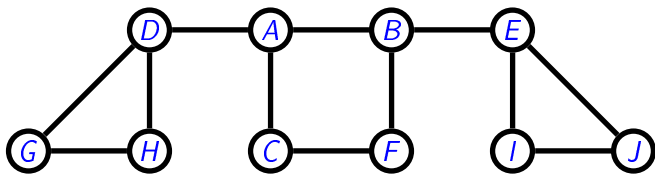
Пример обхода



Пример обхода



Пример обхода



Типы рёбер

Типы рёбер

Типы рёбер

Типы рёбер

- Алгоритм `EXPLORE` строит дерево с корнем в вершине v .

Типы рёбер

Типы рёбер

- Алгоритм `EXPLORE` строит дерево с корнем в вершине v .
- Рёбра этого дерева называются **рёбрами дерева**.

Типы рёбер

Типы рёбер

- Алгоритм EXPLORE строит дерево с корнем в вершине v .
- Рёбра этого дерева называются **рёбрами дерева**.
- Все оставшиеся рёбра графа называются **обратными** (такие рёбра ведут в вершину, в которой алгоритм уже бывал).

Поиск в глубину

Алгоритм

DFS(G)

```
1  for всех вершин  $v \in V$ 
2      do  $visited(v) = FALSE$ 
3  for всех вершин  $v \in V$ 
4      do if  $visited(v) = FALSE$ 
5          then EXPLORE( $v$ )
```

Поиск в глубину

Алгоритм

DFS(G)

```
1  for всех вершин  $v \in V$ 
2      do  $visited(v) = \text{FALSE}$ 
3  for всех вершин  $v \in V$ 
4      do if  $visited(v) = \text{FALSE}$ 
5          then EXPLORE( $v$ )
```

Время работы

Ясно, что время работы такого алгоритма есть $O(|V| + |E|)$.

Связность

Определение

Связность

Определение

- Неориентированный граф называется **связным**, если в нём существует путь между любыми двумя вершинами.

Связность

Определение

- Неориентированный граф называется **связным**, если в нём существует путь между любыми двумя вершинами.
- Если граф не связан, но он состоит из нескольких **компонент связности**.

Связность

Определение

- Неориентированный граф называется **связным**, если в нём существует путь между любыми двумя вершинами.
- Если граф не связан, но он состоит из нескольких **компонент связности**.

Проверка связности

Алгоритм поиска в глубину легко модифицировать так, чтобы он для каждой вершины находил также номер компоненты связности, к которой принадлежит данная вершина. Для этого заводится массив *ccnum* и переменная *cc*, которая изначально установлена в ноль и увеличивается на единицу каждый раз, когда DFS вызывает EXPLORE, и пишется такая строчка кода:

PREVISIT(*v*)

```
1  ccnum[v] = cc
```

Времена обработки

Времена обработки

Будем запоминать время начала и время конца обработки вершины следующим образом:

PREVISIT(v)

- 1 $pre[v] = clock$
- 2 $clock = clock + 1$

POSTVISIT(v)

- 1 $post[v] = clock$
- 2 $clock = clock + 1$

Времена обработки

Времена обработки

Будем запоминать время начала и время конца обработки вершины следующим образом:

PREVISIT(v)

- 1 $pre[v] = clock$
- 2 $clock = clock + 1$

POSTVISIT(v)

- 1 $post[v] = clock$
- 2 $clock = clock + 1$

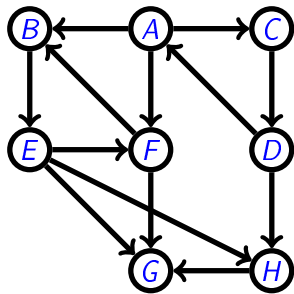
Замечание

Нетрудно видеть, что для любых двух вершин u и v интервалы $[pre(u), post(u)]$ и $[pre(v), post(v)]$ либо не пересекаются, либо один содержится в другом.

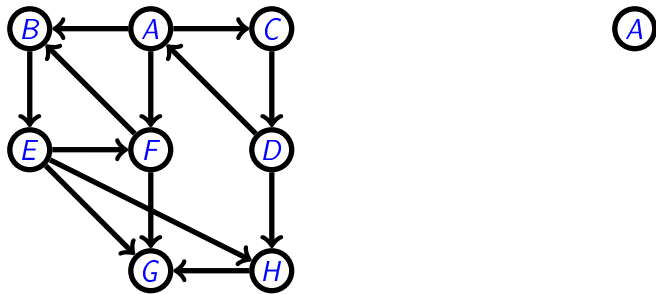
План лекции

- 1 Поиск в глубину в ненаправленных графах
 - Связность
- 2 Поиск в глубину в направленных графах
 - Типы рёбер
 - Ациклические графы
 - Топологическая сортировка
- 3 Сильно связные компоненты
- 4 Упражнения

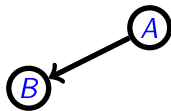
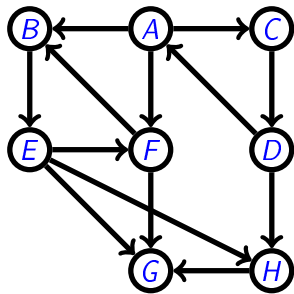
Пример обхода ориентированного графа



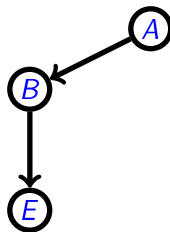
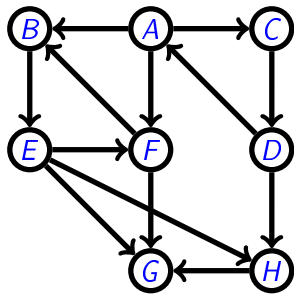
Пример обхода ориентированного графа



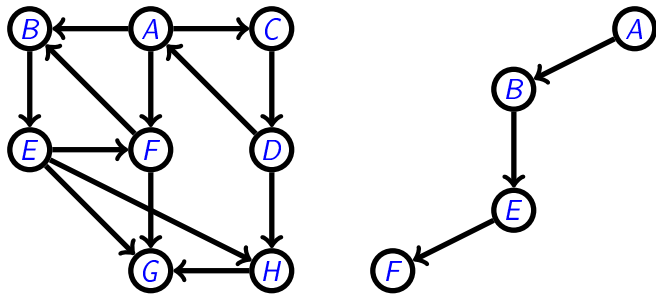
Пример обхода ориентированного графа



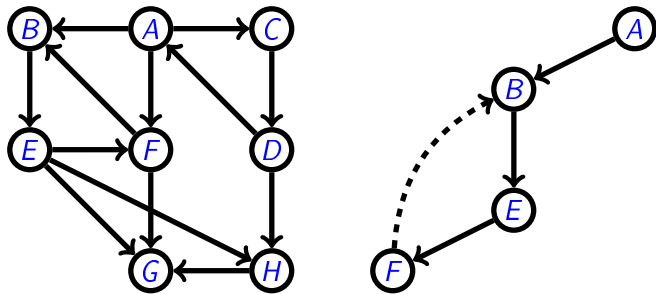
Пример обхода ориентированного графа



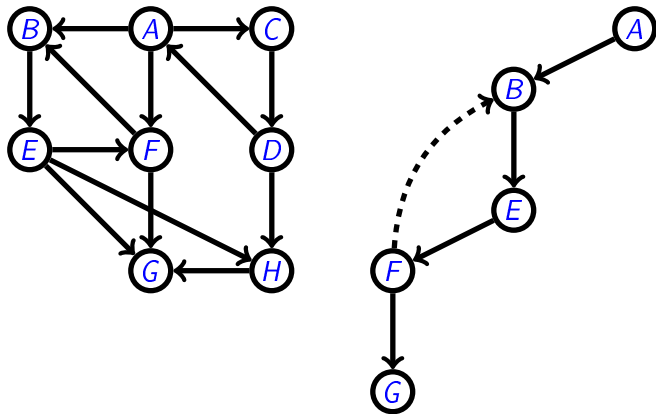
Пример обхода ориентированного графа



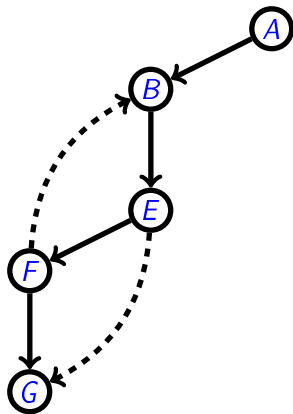
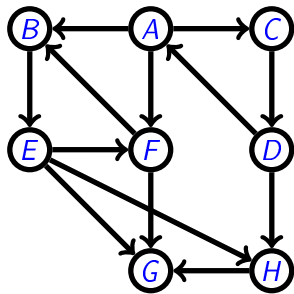
Пример обхода ориентированного графа



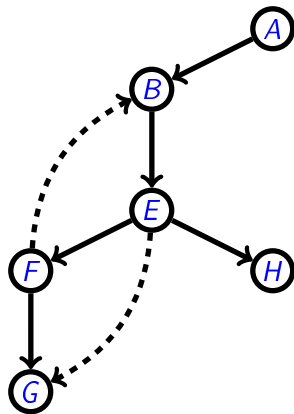
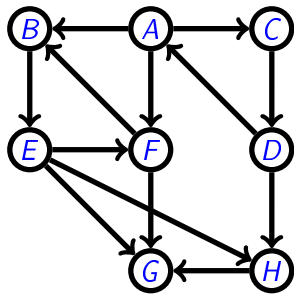
Пример обхода ориентированного графа



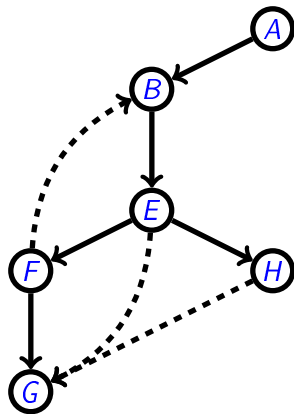
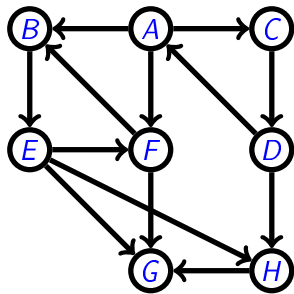
Пример обхода ориентированного графа



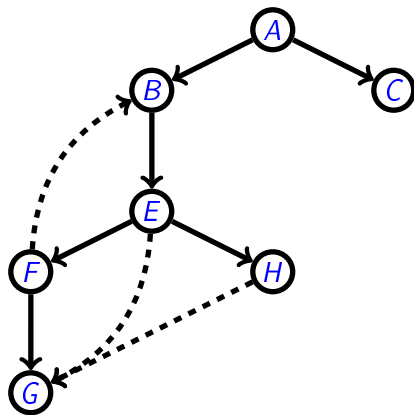
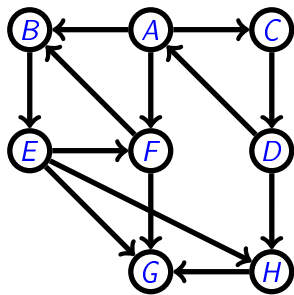
Пример обхода ориентированного графа



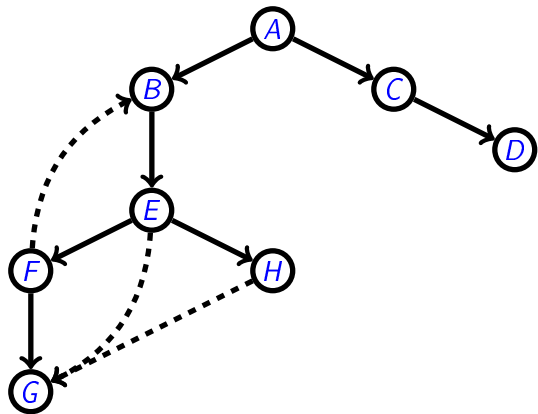
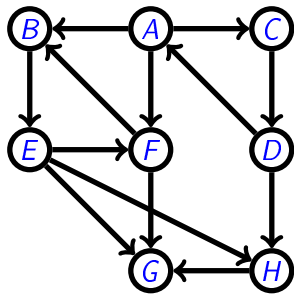
Пример обхода ориентированного графа



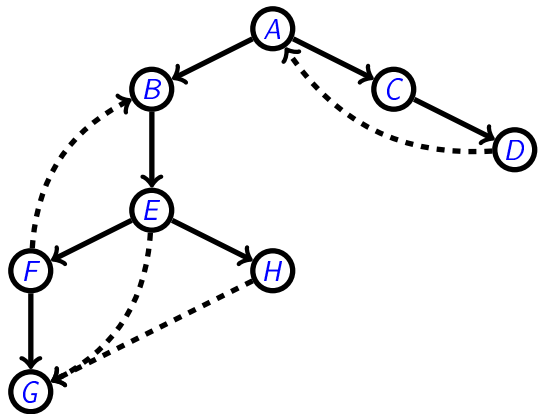
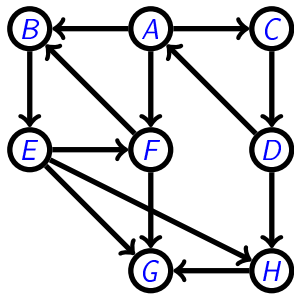
Пример обхода ориентированного графа



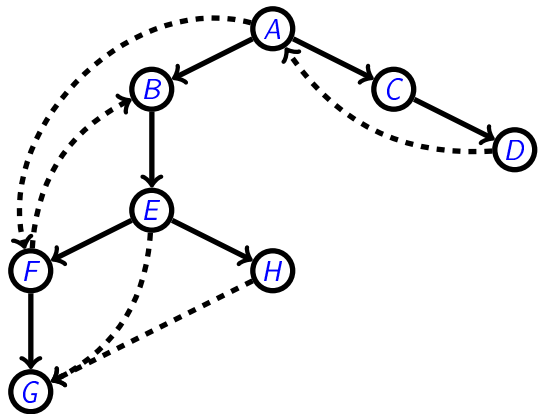
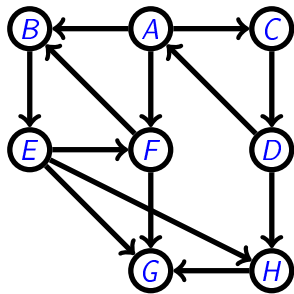
Пример обхода ориентированного графа



Пример обхода ориентированного графа



Пример обхода ориентированного графа



Типы рёбер

Типы рёбер

Типы рёбер

Типы рёбер

- **Ребро дерева** — ребро леса, построенного поиском в глубину.

Типы рёбер

Типы рёбер

- **Ребро дерева** — ребро леса, построенного поиском в глубину.
- **Прямое ребро** — ребро дерева, ведущее из вершины в её непосредственного потомка.

Типы рёбер

Типы рёбер

- **Ребро дерева** — ребро леса, построенного поиском в глубину.
- **Прямое ребро** — ребро дерева, ведущее из вершины в её непосредственного потомка.
- **Обратное ребро** — ребро из вершины дерева в её предка (не обязательно непосредственного).

Типы рёбер

Типы рёбер

- **Ребро дерева** — ребро леса, построенного поиском в глубину.
- **Прямое ребро** — ребро дерева, ведущее из вершины в её не непосредственного потомка.
- **Обратное ребро** — ребро из вершины дерева в её предка (не обязательно непосредственного).
- **Обратное ребро** — ребро из вершины дерева в её предка (не обязательно непосредственного).

Ациклические графы

Лемма

ориентированный граф содержит цикл тогда и только тогда, когда DFS находит обратное ребро.

Ациклические графы

Лемма

ориентированный граф содержит цикл тогда и только тогда, когда DFS находит обратное ребро.

Доказательство

Ациклические графы

Лемма

ориентированный граф содержит цикл тогда и только тогда, когда DFS находит обратное ребро.

Доказательство

- Если обратное ребро есть, то и цикл, очевидно, есть.

Ациклические графы

Лемма

ориентированный граф содержит цикл тогда и только тогда, когда DFS находит обратное ребро.

Доказательство

- Если обратное ребро есть, то и цикл, очевидно, есть.
- Допустим теперь, что есть цикл $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k \rightarrow v_0$.

Ациклические графы

Лемма

ориентированный граф содержит цикл тогда и только тогда, когда DFS находит обратное ребро.

Доказательство

- Если обратное ребро есть, то и цикл, очевидно, есть.
- Допустим теперь, что есть цикл $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k \rightarrow v_0$.
- Пусть v_i — первая вершина цикла, которую обнаружит DFS.

Ациклические графы

Лемма

ориентированный граф содержит цикл тогда и только тогда, когда DFS находит обратное ребро.

Доказательство

- Если обратное ребро есть, то и цикл, очевидно, есть.
- Допустим теперь, что есть цикл $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k \rightarrow v_0$.
- Пусть v_i — первая вершина цикла, которую обнаружит DFS.
- Тогда DFS найдет и все другие вершины и, в частности, обратное ребро $v_{i-1} \rightarrow v_i$. □

Топологическая сортировка

Определение

Топологическая сортировка — такая нумерация вершин ориентированного графа, при которой рёбра все рёбра идут из вершины с меньшим номером в вершину с большим номером.

Топологическая сортировка

Определение

Топологическая сортировка — такая нумерация вершин ориентированного графа, при которой рёбра все рёбра идут из вершины с меньшим номером в вершину с большим номером.

Замечания

Топологическая сортировка

Определение

Топологическая сортировка — такая нумерация вершин ориентированного графа, при которой рёбра все рёбра идут из вершины с меньшим номером в вершину с большим номером.

Замечания

- Легко видеть, что у ориентированного графа есть топологическая сортировка тогда и только тогда, когда он является ациклическим.

Топологическая сортировка

Определение

Топологическая сортировка — такая нумерация вершин ориентированного графа, при которой рёбра все рёбра идут из вершины с меньшим номером в вершину с большим номером.

Замечания

- Легко видеть, что у ориентированного графа есть топологическая сортировка тогда и только тогда, когда он является ациклическим.
- Более того, в ориентированном графе каждое ребро ведёт в вершину с меньшим значением *post*.

Топологическая сортировка

Определение

Топологическая сортировка — такая нумерация вершин ориентированного графа, при которой рёбра все рёбра идут из вершины с меньшим номером в вершину с большим номером.

Замечания

- Легко видеть, что у ориентированного графа есть топологическая сортировка тогда и только тогда, когда он является ациклическим.
- Более того, в ориентированном графе каждое ребро ведёт в вершину с меньшим значением *post*.
- Таким образом, топологическая сортировка графа может быть найдена за линейное время.

Топологическая сортировка

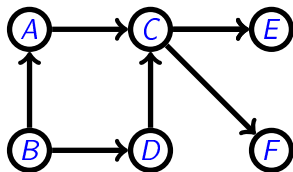
Определение

Топологическая сортировка — такая нумерация вершин ориентированного графа, при которой рёбра все рёбра идут из вершины с меньшим номером в вершину с большим номером.

Замечания

- Легко видеть, что у ориентированного графа есть топологическая сортировка тогда и только тогда, когда он является ациклическим.
- Более того, в ориентированном графе каждое ребро ведёт в вершину с меньшим значением *post*.
- Таким образом, топологическая сортировка графа может быть найдена за линейное время.
- Ясно также, что у любого ориентированного ациклического графа есть **исток** (вершина, в которую не входит ни одно ребро) и **сток** (вершина, из которой не выходит ни одного ребра).

Пример топологической сортировки



План лекции

- 1 Поиск в глубину в ненаправленных графах
 - Связность
- 2 Поиск в глубину в направленных графах
 - Типы рёбер
 - Ациклические графы
 - Топологическая сортировка
- 3 **Сильно связанные компоненты**
- 4 Упражнения

Связность ориентированных графов

Связность ориентированных графов

Связность ориентированных графов

Связность ориентированных графов

- Говорим, что вершины u и v связаны, если в графе есть путь из u в v и путь из v в u .

Связность ориентированных графов

Связность ориентированных графов

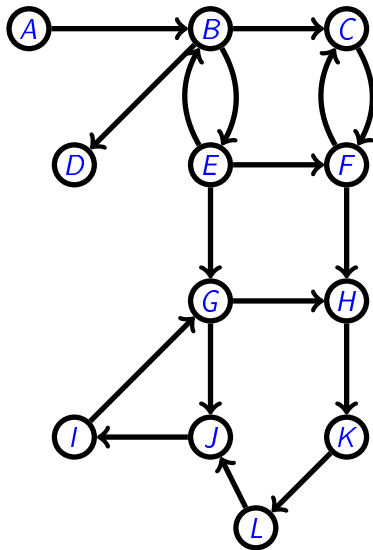
- Говорим, что вершины u и v связаны, если в графе есть путь из u в v и путь из v в u .
- Такое отношение разбивает множество вершин на непересекающиеся подмножества, называемые **СИЛЬНО СВЯЗНЫМИ КОМПОНЕНТАМИ**.

Связность ориентированных графов

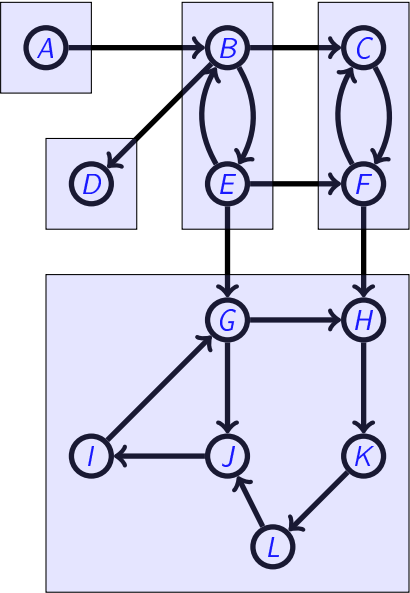
Связность ориентированных графов

- Говорим, что вершины u и v связаны, если в графе есть путь из u в v и путь из v в u .
- Такое отношение разбивает множество вершин на непересекающиеся подмножества, называемые **СИЛЬНО СВЯЗНЫМИ КОМПОНЕНТАМИ**.
- Легко видеть, что при стягивании каждой такой компоненты в одну вершину получается ациклический ориентированный граф.

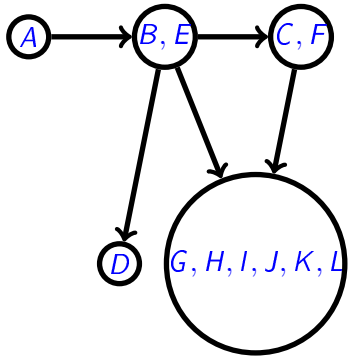
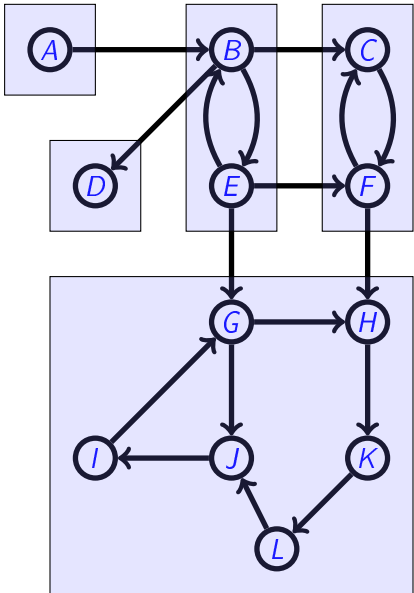
Пример стягивания



Пример стягивания



Пример стягивания



Как искать сильно связанные компоненты?

Как искать сильно связанные компоненты?

Как искать сильно связанные компоненты?

Как искать сильно связанные компоненты?

- Процедура EXPLORE, вызванная для вершины u , обходит те и только те вершины, которые достижимы из u .

Как искать сильно связанные компоненты?

Как искать сильно связанные компоненты?

- Процедура EXPLORE, вызванная для вершины u , обходит те и только те вершины, которые достижимы из u .
- Значит, если начать с вершины, которая лежит в стоке стянутого графа, то мы найдем компоненту, соответствующую стоку.

Как искать сильно связанные компоненты?

Как искать сильно связанные компоненты?

- Процедура EXPLORE, вызванная для вершины u , обходит те и только те вершины, которые достижимы из u .
- Значит, если начать с вершины, которая лежит в стоке стянутого графа, то мы найдем компоненту, соответствующую стоку.
- Если бы мы могли быстро найти компоненту-сток, то мы бы просто выкинули её из графа и продолжили бы поиск.

Как искать сильно связанные компоненты?

Как искать сильно связанные компоненты?

- Процедура EXPLORE, вызванная для вершины u , обходит те и только те вершины, которые достижимы из u .
- Значит, если начать с вершины, которая лежит в стоке стянутого графа, то мы найдем компоненту, соответствующую стоку.
- Если бы мы могли быстро найти компоненту-сток, то мы бы просто выкинули её из графа и продолжили бы поиск.
- Вершина графа с самым большим значением $post$ лежит в истоке стянутого графа.

Как искать сильно связанные компоненты?

Как искать сильно связанные компоненты?

- Процедура EXPLORE, вызванная для вершины u , обходит те и только те вершины, которые достижимы из u .
- Значит, если начать с вершины, которая лежит в стоке стянутого графа, то мы найдем компоненту, соответствующую стоку.
- Если бы мы могли быстро найти компоненту-сток, то мы бы просто выкинули её из графа и продолжили бы поиск.
- Вершина графа с самым большим значением $post$ лежит в истоке стянутого графа.
- Сильно связанные компоненты можно топологически упорядочить по максимальным значениям $post$ содержащихся в них вершин.

Обратный граф

Обратный граф

Обратный граф

Обратный граф

- Рассмотрим обратный граф G^R , получаемый из G переворачиванием всех рёбер.

Обратный граф

Обратный граф

- Рассмотрим обратный граф G^R , получаемый из G переворачиванием всех рёбер.
- Ясно, что в G^R точно такие же сильно связные компоненты, как и в G .

Обратный граф

Обратный граф

- Рассмотрим обратный граф G^R , получаемый из G переворачиванием всех рёбер.
- Ясно, что в G^R точно такие же сильно связанные компоненты, как и в G .
- Значит, вершина с максимальным значением *post* в G^R принадлежит компоненте-стоку графа G .

Обратный граф

Обратный граф

- Рассмотрим обратный граф G^R , получаемый из G переворачиванием всех рёбер.
- Ясно, что в G^R точно такие же сильно связные компоненты, как и в G .
- Значит, вершина с максимальным значением $post$ в G^R принадлежит компоненте-стоку графа G .

Алгоритм

- 1 $DFS(G^R)$
- 2 Запустить $DFS(G)$, перебирая вершины в порядке убывания значений $post$, найденных на предыдущем шаге.

План лекции

- 1 Поиск в глубину в ненаправленных графах
 - Связность
- 2 Поиск в глубину в направленных графах
 - Типы рёбер
 - Ациклические графы
 - Топологическая сортировка
- 3 Сильно связанные компоненты
- 4 Упражнения

Упражнения

Упражнения

Упражнения

Упражнения

- Как быстро построить G^R ?

Упражнения

Упражнения

- Как быстро построить G^R ?
- Как быстро упорядочить вершины в порядке убывания значений *post*?

Упражнения

Упражнения

- Как быстро построить G^R ?
- Как быстро упорядочить вершины в порядке убывания значений *post*?
- Постройте алгоритм, проверяющий, является ли входной граф двудольным.

Упражнения

Упражнения

- Как быстро построить G^R ?
- Как быстро упорядочить вершины в порядке убывания значений *post*?
- Постройте алгоритм, проверяющий, является ли входной граф двудольным.
- Докажите, что граф является двудольным тогда и только тогда, когда он не содержит циклов нечётной длины.

Упражнения

Упражнения

- Как быстро построить G^R ?
- Как быстро упорядочить вершины в порядке убывания значений *post*?
- Постройте алгоритм, проверяющий, является ли входной граф двудольным.
- Докажите, что граф является двудольным тогда и только тогда, когда он не содержит циклов нечётной длины.
- Во сколько цветов можно покрасить граф, содержащий ровно один цикл нечётной длины.

Упражнения

Упражнения

Упражнения

Упражнения

- Есть три сосуда ёмкостей 4, 7 и 10 литров. Два первых полностью наполнены, последний пуст. Из одного сосуда в другой можно переливать до тех пор, пока один из сосудов не наполнится или не опустошится. Необходимо проверить, есть ли способ оставить ровно 2 литра воды в одном из первых двух сосудов.

Упражнения

Упражнения

- Есть три сосуда ёмкостей 4, 7 и 10 литров. Два первых полностью наполнены, последний пуст. Из одного сосуда в другой можно переливать до тех пор, пока один из сосудов не наполнится или не опустошится. Необходимо проверить, есть ли способ оставить ровно 2 литра воды в одном из первых двух сосудов.
 - ▶ Сформулируйте данную задачу на языке графов.

Упражнения

Упражнения

- Есть три сосуда ёмкостей 4, 7 и 10 литров. Два первых полностью наполнены, последний пуст. Из одного сосуда в другой можно переливать до тех пор, пока один из сосудов не наполнится или не опустошится. Необходимо проверить, есть ли способ оставить ровно 2 литра воды в одном из первых двух сосудов.
 - ▶ Сформулируйте данную задачу на языке графов.
 - ▶ Какой алгоритм нужно применить, чтобы найти ответ?

Упражнения

Упражнения

- Есть три сосуда ёмкостей 4, 7 и 10 литров. Два первых полностью наполнены, последний пуст. Из одного сосуда в другой можно переливать до тех пор, пока один из сосудов не наполнится или не опустошится. Необходимо проверить, есть ли способ оставить ровно 2 литра воды в одном из первых двух сосудов.
 - ▶ Сформулируйте данную задачу на языке графов.
 - ▶ Какой алгоритм нужно применить, чтобы найти ответ?
- Двойной степенью вершины неориентированного графа называется сумма степеней его соседей. Постройте алгоритм, находящий двойные степени всех вершин графа.

Упражнения

Упражнения

- Есть три сосуда ёмкостей 4, 7 и 10 литров. Два первых полностью наполнены, последний пуст. Из одного сосуда в другой можно переливать до тех пор, пока один из сосудов не наполнится или не опустошится. Необходимо проверить, есть ли способ оставить ровно 2 литра воды в одном из первых двух сосудов.
 - ▶ Сформулируйте данную задачу на языке графов.
 - ▶ Какой алгоритм нужно применить, чтобы найти ответ?
- Двойной степенью вершины неориентированного графа называется сумма степеней его соседей. Постройте алгоритм, находящий двойные степени всех вершин графа.
- Постройте алгоритм, проверяющий, существует ли в данном неориентированном графе цикл, содержащий данное ребро.

Упражнения

Упражнения

- Есть три сосуда ёмкостей 4, 7 и 10 литров. Два первых полностью наполнены, последний пуст. Из одного сосуда в другой можно переливать до тех пор, пока один из сосудов не наполнится или не опустошится. Необходимо проверить, есть ли способ оставить ровно 2 литра воды в одном из первых двух сосудов.
 - ▶ Сформулируйте данную задачу на языке графов.
 - ▶ Какой алгоритм нужно применить, чтобы найти ответ?
- Двойной степенью вершины неориентированного графа называется сумма степеней его соседей. Постройте алгоритм, находящий двойные степени всех вершин графа.
- Постройте алгоритм, проверяющий, существует ли в данном неориентированном графе цикл, содержащий данное ребро.
- Докажите, что в любом связном ориентированном графе есть вершина, удаление которой оставляет граф связным.

Упражнения

Упражнения

Упражнения

Упражнения

- Приведите пример сильно связного ориентированного графа, удаление любой вершины которого оставляет не сильно связный граф.

Упражнения

Упражнения

- Приведите пример сильно связного ориентированного графа, удаление любой вершины которого оставляет не сильно связный граф.
- Рассмотрим несколько курсов, некоторые из которых предполагают знание других. Допустив, что студент в течение курса может слушать любое число курсов, найдите минимальное количество семестров, необходимых для прочтения всех курсов.

Упражнения

Упражнения

- Приведите пример сильно связного ориентированного графа, удаление любой вершины которого оставляет не сильно связный граф.
- Рассмотрим несколько курсов, некоторые из которых предполагают знание других. Допустив, что студент в течение курса может слушать любое число курсов, найдите минимальное количество семестров, необходимых для прочтения всех курсов.
- Дано бинарное дерево (в виде списков смежности) с выделенным корнем. Приведите алгоритм предобработки, который позволил бы отвечать на вопросы типа “является ли u потомком v ” за константное время.

Упражнения

Упражнения

Упражнения

Упражнения

- Дано бинарное дерево с выделенным корнем, а также массив $x[\cdot]$, проиндексированный вершинами. Зададим новый массив z так: $z[u]$ есть максимальное значение x на потомках u . Постройте алгоритм, заполняющий массив $z[\cdot]$.

Упражнения

Упражнения

- Дано бинарное дерево с выделенным корнем, а также массив $x[\cdot]$, проиндексированный вершинами. Зададим новый массив z так: $z[u]$ есть максимальное значение x на потомках u . Постройте алгоритм, заполняющий массив $z[\cdot]$.
- Постройте алгоритм, находящий цикл нечётной длины в ориентированном графе. (Подсказка: рассмотрите сначала случай, когда входной граф сильно связан.)

Упражнения

Упражнения

- Дано бинарное дерево с выделенным корнем, а также массив $x[\cdot]$, проиндексированный вершинами. Зададим новый массив z так: $z[u]$ есть максимальное значение x на потомках u . Постройте алгоритм, заполняющий массив $z[\cdot]$.
- Постройте алгоритм, находящий цикл нечётной длины в ориентированном графе. (Подсказка: рассмотрите сначала случай, когда входной граф сильно связан.)
- Постройте алгоритм, который проверяет, есть ли в данном ориентированном графе вершина, из которой достижимы все остальные вершины.

Упражнения

Упражнения

- Дано бинарное дерево с выделенным корнем, а также массив $x[\cdot]$, проиндексированный вершинами. Зададим новый массив z так: $z[u]$ есть максимальное значение x на потомках u . Постройте алгоритм, заполняющий массив $z[\cdot]$.
- Постройте алгоритм, находящий цикл нечётной длины в ориентированном графе. (Подсказка: рассмотрите сначала случай, когда входной граф сильно связан.)
- Постройте алгоритм, который проверяет, есть ли в данном ориентированном графе вершина, из которой достижимы все остальные вершины.
- Постройте алгоритм, находящий количество различных путей между данными двумя вершинами в данном ориентированном графе.

Упражнения

Упражнения

Упражнения

Упражнения

- Постройте алгоритм, проверяющий, есть ли в данном ориентированном графе путь, проходящий через все вершины ровно по одному разу.

Упражнения

Упражнения

- Постройте алгоритм, проверяющий, есть ли в данном ориентированном графе путь, проходящий через все вершины ровно по одному разу.
- Дан ориентированный граф, каждой вершине которого приписана натуральная стоимость. Постройте алгоритм, заполняющий массив *cost*, где *cost*[*u*] есть минимальная стоимость вершины, достижимой из *u*. (Подсказка: рассмотрите сначала случай ациклического графа.)

Спасибо за внимание!