Алгоритмы и структуры данных

Динамическая связность в графах

CS Center, Новосибирск

Постановка задачи

- Неориентированный граф
- Обрабатываем запросы:
  - добавить ребро (u, v)
  - удалить ребро (u, v)
  - проверить, что u достижима из v (u и v в одной компоненте связности)

Постановка задачи

- Неориентированный граф
- Обрабатываем запросы:
  - добавить ребро (u, v)
  - удалить ребро (u, v)
  - проверить, что u достижима из v (u и v в одной компоненте связности)
- Соображения:
  - нужно поддерживать компоненты связности

Постановка задачи

- Неориентированный граф
- Обрабатываем запросы:
  - добавить ребро (u, v)
  - удалить ребро (u, v)
  - проверить, что u достижима из v (u и v в одной компоненте связности)
- Соображения:
  - нужно поддерживать компоненты связности
  - для начала стоит научиться решать более простую задачу

Постановка задачи

- Неориентированный граф
- Обрабатываем запросы:
  - добавить ребро (u, v)
  - удалить ребро (u, v)
  - проверить, что u достижима из v (u и v в одной компоненте связности)
- Соображения:
  - нужно поддерживать компоненты связности
  - для начала стоит научиться решать более простую задачу
    - как упростить?

Динамическая связность в лесе

- Ограничение на граф: нет циклов

Динамическая связность в лесе

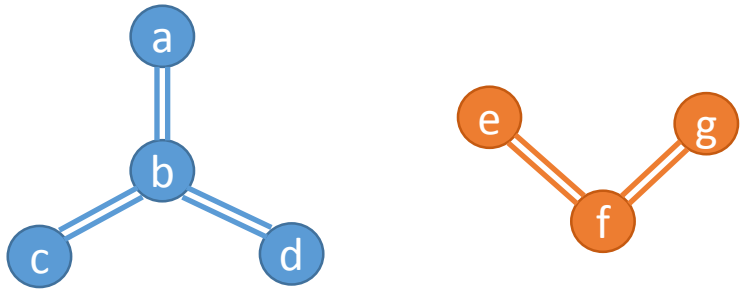
- Ограничение на граф: нет циклов
  - Любое удаление ребра приводит к образование новой компоненты связности

Динамическая связность в лесе

- Ограничение на граф: нет циклов
  - Любое удаление ребра приводит к образование новой компоненты связности
- Используем эйлеров обход дерева

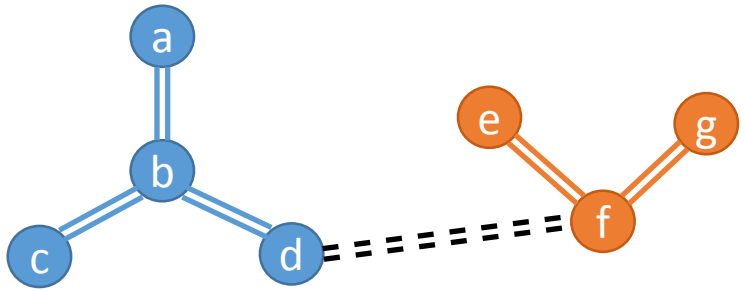
Динамическая связность в лесе

- Ограничение на граф: нет циклов
  - Любое удаление ребра приводит к образование новой компоненты связности
- Используем эйлеров обход дерева



Динамическая связность в лесе

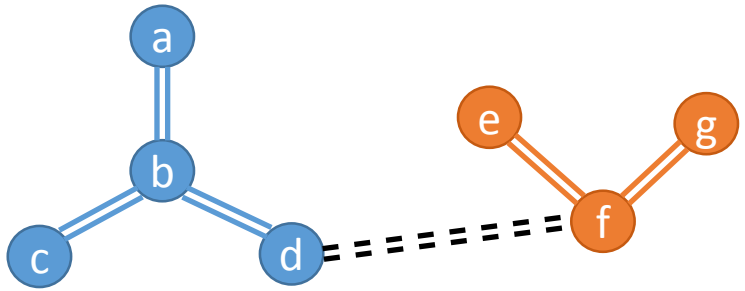
- Ограничение на граф: нет циклов
  - Любое удаление ребра приводит к образование новой компоненты связности
- Используем эйлеров обход дерева



abcbdb efgf

Динамическая связность в лесе

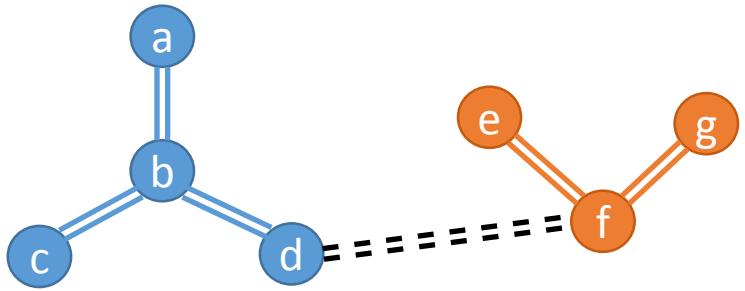
- Ограничение на граф: нет циклов
  - Любое удаление ребра приводит к образование новой компоненты связности
- Используем эйлеров обход дерева



abcbd|b ef|gf

Динамическая связность в лесе

- Ограничение на граф: нет циклов
  - Любое удаление ребра приводит к образование новой компоненты связности
- Используем эйлеров обход дерева

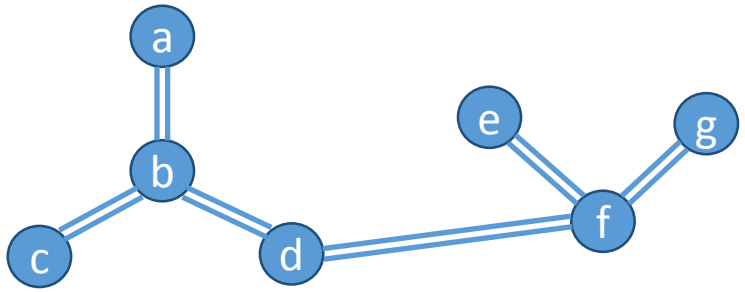


abcbd|b ef|gf

abcbdfgfefdb

Динамическая связность в лесе

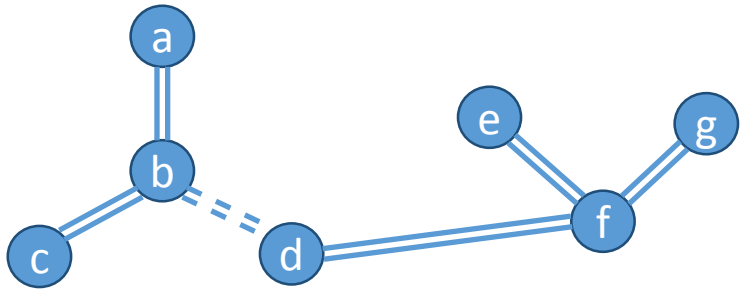
- Ограничение на граф: нет циклов
  - Любое удаление ребра приводит к образование новой компоненты связности
- Используем эйлеров обход дерева



abcbdfgfefdb

Динамическая связность в лесе

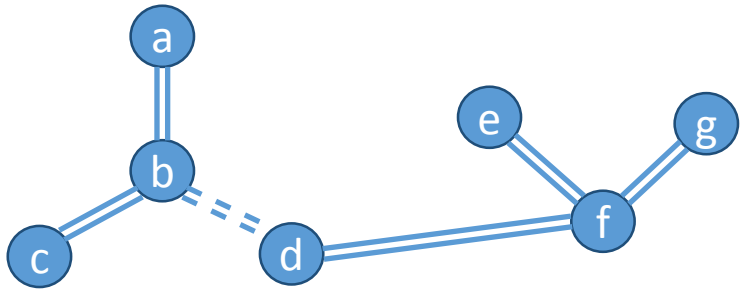
- Ограничение на граф: нет циклов
  - Любое удаление ребра приводит к образование новой компоненты связности
- Используем эйлеров обход дерева



abcbdfgfefdb

Динамическая связность в лесе

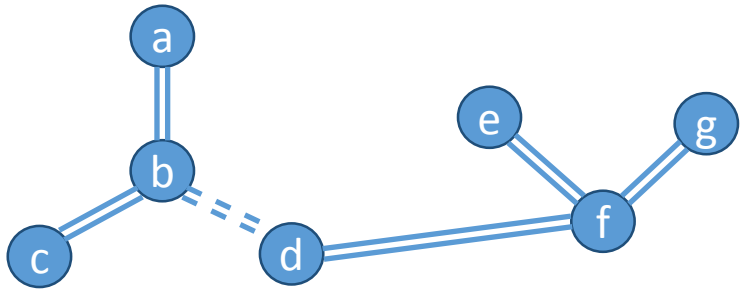
- Ограничение на граф: нет циклов
  - Любое удаление ребра приводит к образование новой компоненты связности
- Используем эйлеров обход дерева



abcbdfgfefdb

Динамическая связность в лесе

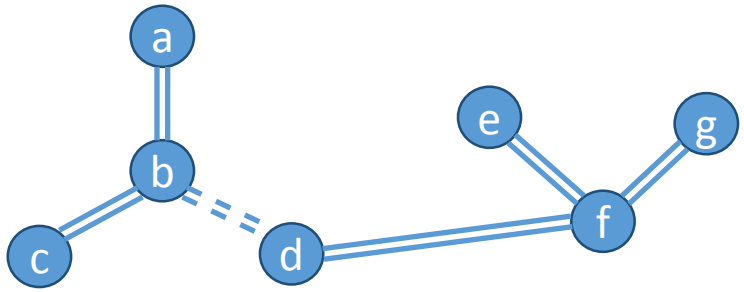
- Ограничение на граф: нет циклов
  - Любое удаление ребра приводит к образование новой компоненты связности
- Используем эйлеров обход дерева



abcb|d|fgfefd|b|

Динамическая связность в лесе

- Ограничение на граф: нет циклов
  - Любое удаление ребра приводит к образование новой компоненты связности
- Используем эйлеров обход дерева

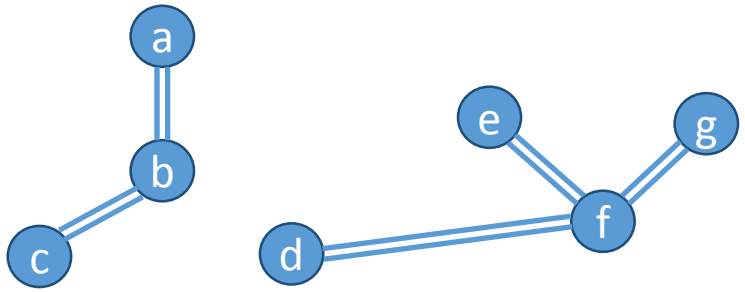


abcb|d|fgfefd|b|

abcb fgfefd

Динамическая связность в лесе

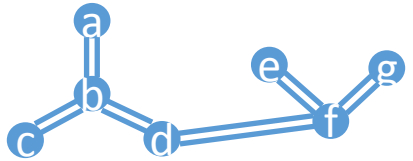
- Ограничение на граф: нет циклов
  - Любое удаление ребра приводит к образование новой компоненты связности
- Используем эйлеров обход дерева



abcb fgfefd

Динамическая связность в лесе

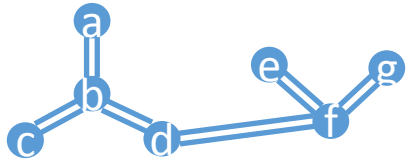
- Ограничение на граф: нет циклов
  - Любое удаление ребра приводит к образование новой компоненты связности
- Используем эйлеров обход дерева
- Добавление ребра (u, v)
  - находим вхождение u: S = AuB



abcbdfgfefdb

Динамическая связность в лесе

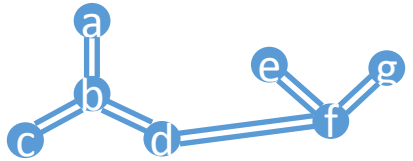
- Ограничение на граф: нет циклов
  - Любое удаление ребра приводит к образование новой компоненты связности
- Используем эйлеров обход дерева
- Добавление ребра (u, v)
  - находим вхождение u: S = AuB
  - находим вхождение v: T = CvD



abcbdfgfefdb

Динамическая связность в лесе

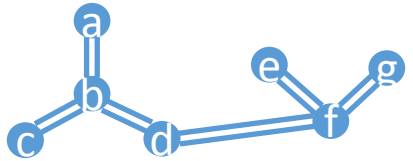
- Ограничение на граф: нет циклов
  - Любое удаление ребра приводит к образование новой компоненты связности
- Используем эйлеров обход дерева
- Добавление ребра (u, v)
  - находим вхождение u: S = AuB
  - находим вхождение v: T = CvD
  - новая компонента: AuvDCvuB



abcbdfgfefdb

Динамическая связность в лесе

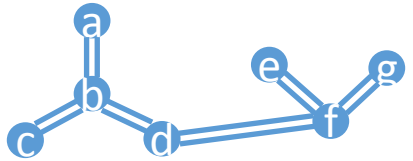
- Ограничение на граф: нет циклов
  - Любое удаление ребра приводит к образование новой компоненты связности
- Используем эйлеров обход дерева
- Добавление ребра (u, v)
  - находим вхождение u: S = AuB
  - находим вхождение v: T = CvD
  - новая компонента: AuvDCvuB
- Удаление ребра (u, v):
  - S = AuvBvuC



abcbdfgfefdb

Динамическая связность в лесе

- Ограничение на граф: нет циклов
  - Любое удаление ребра приводит к образование новой компоненты связности
- Используем эйлеров обход дерева
- Добавление ребра (u, v)
  - находим вхождение u: S = A u B
  - находим вхождение v: T = C v D
  - новая компонента: A u v D C v u B
- Удаление ребра (u, v):
  - S = AuvBvuC
  - новые компоненты: AuC, vB



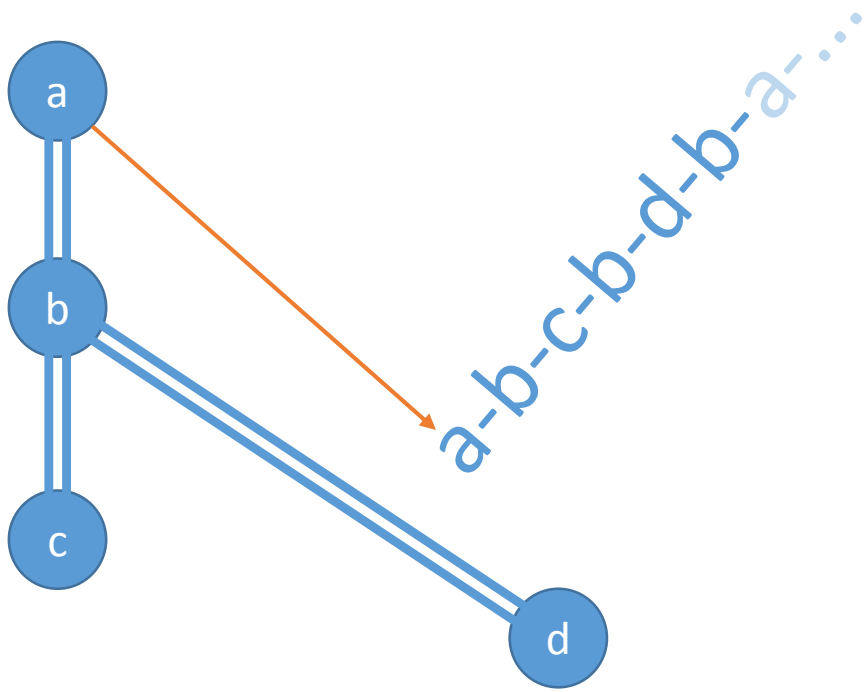
abcbdfgfefdb

Динамическая связность в лесе: Rope

- Операции со строками, представляющими эйлеровы обходы:
  - split – разрезать строку
  - concat – объединить строки
  - check – проверить, что два символа в одной строке
- Структрура данных «Rope»
  - Реализация – дерево поиска по неявному ключу
    - Каждому символу соответсвует вершина дерева
      - Строка читается при inorder-обходе дерева
    - split -> split дерева
    - concat -> merge деревьев
    - check -> проверить, что две вершины в одном дереве

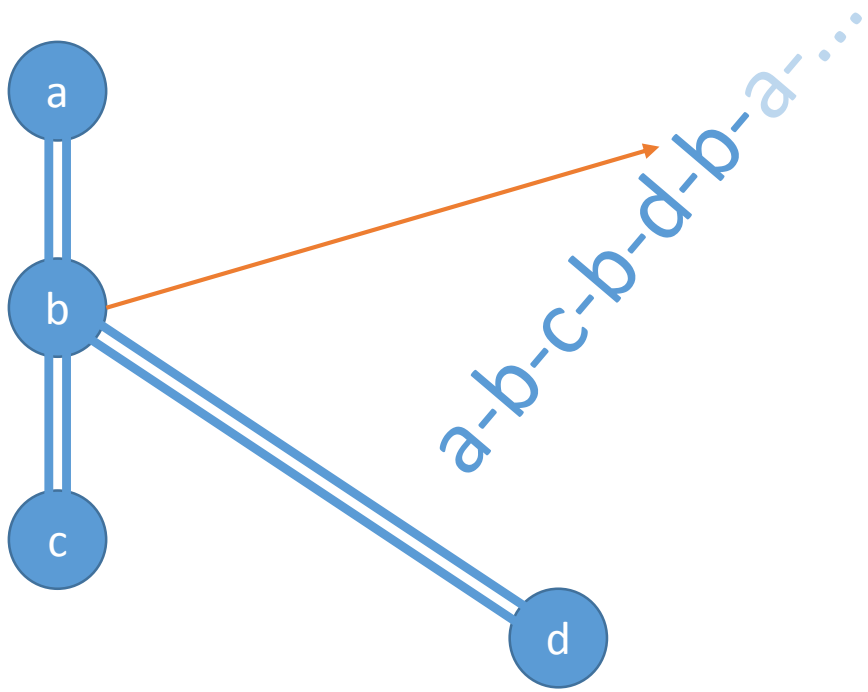
Rope и позиция в строке

- Для каждой вершины запомним ссылку на некторое её вхождение в строку



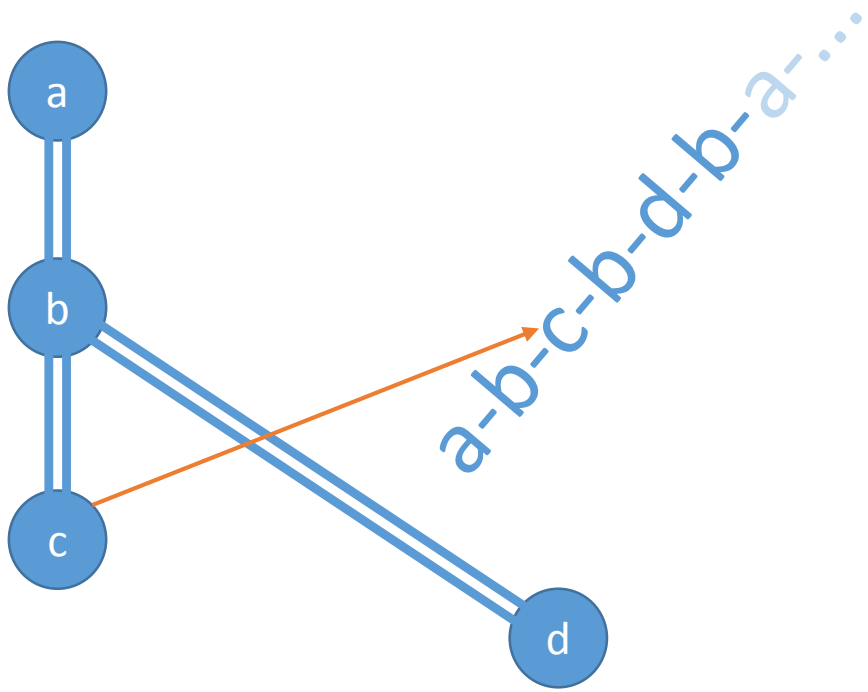
Rope и позиция в строке

- Для каждой вершины запомним ссылку на некторое её вхождение в строку



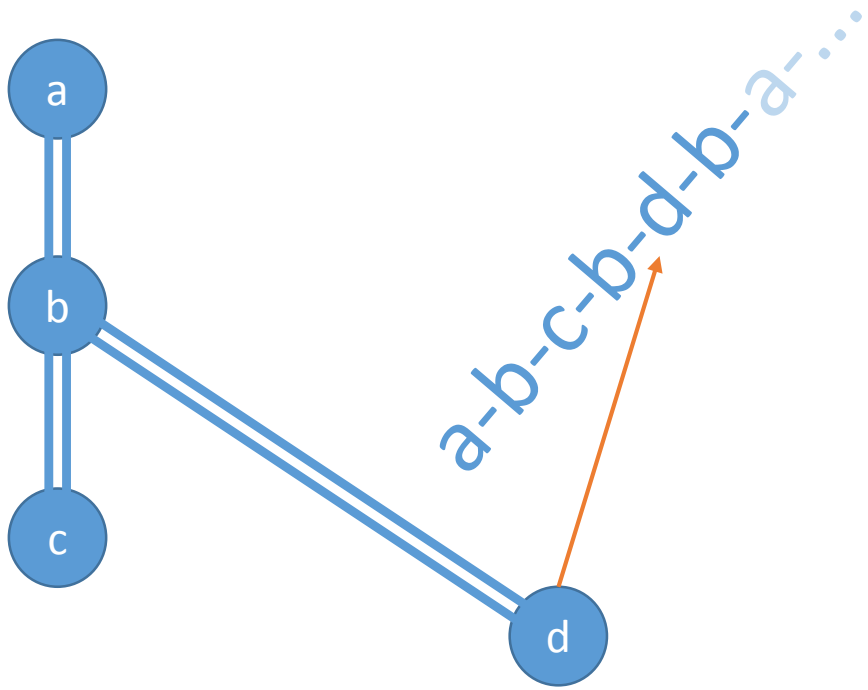
Rope и позиция в строке

- Для каждой вершины запомним ссылку на некторое её вхождение в строку



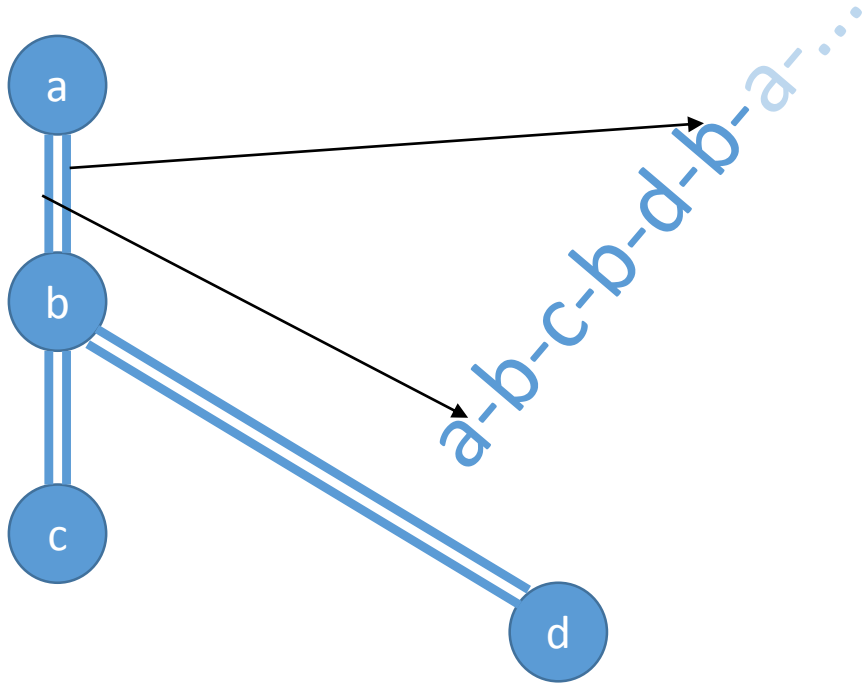
Rope и позиция в строке

- Для каждой вершины запомним ссылку на некторое её вхождение в строку



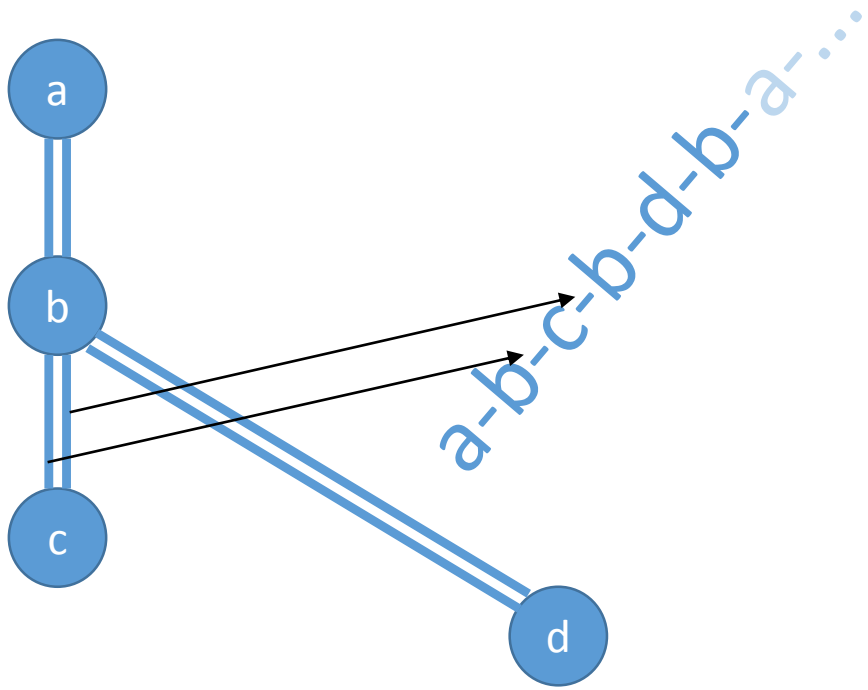
Rope и позиция в строке

- Каждому ребру соответствует две позиции в строке. Эти позиции нужно хранить, чтобы использовать при удалении ребра.



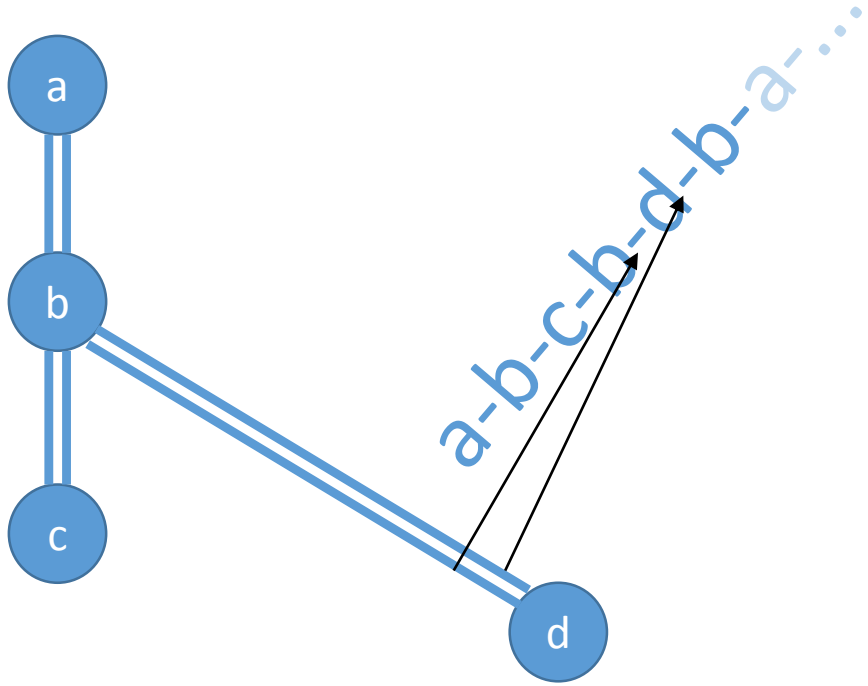
Rope и позиция в строке

- Каждому ребру соответствует две позиции в строке. Эти позиции нужно хранить, чтобы использовать при удалении ребра.



Rope и позиция в строке

- Каждому ребру соответствует две позиции в строке. Эти позиции нужно хранить, чтобы использовать при удалении ребра.



Динамическая связность в графе

- Вернёмся к исходной задаче

Динамическая связность в графе

- Вернёмся к исходной задаче
- Будем использовать динамическую связность в лесе

Динамическая связность в графе

- Вернёмся к исходной задаче
- Будем использовать динамическую связность в лесе
  - Поддерживаем остовный лес

Динамическая связность в графе

- Вернёмся к исходной задаче
- Будем использовать динамическую связность в лесе
  - Поддерживаем остовный лес
  - Как его перестроить при удалении ребра?

Динамическая связность в графе

- Вернёмся к исходной задаче
- Будем использовать динамическую связность в лесе
  - Поддерживаем остовный лес
  - Как его перестроить при удалении ребра?
- Определим функцию L: E -> {1, ... logN}

Динамическая связность в графе

- Вернёмся к исходной задаче
- Будем использовать динамическую связность в лесе
  - Поддерживаем остовный лес
  - Как его перестроить при удалении ребра?
- Определим функцию L: E -> {1, ... logN}
- Определим граф $G_i = (V, E_i)$, $E_i = \{e, L(e) \leq i\}$

Динамическая связность в графе

- Вернёмся к исходной задаче
- Будем использовать динамическую связность в лесе
  - Поддерживаем остовный лес
  - Как его перестроить при удалении ребра?
- Определим функцию L: E -> {1, ... logN}
- Определим граф $G_i = (V, E_i)$, $E_i = \{e, L(e) \leq i\}$
  - $E_i \subseteq E_{i+1}$
  - $E_{logN} = E$

Динамическая связность в графе

- Вернёмся к исходной задаче
- Будем использовать динамическую связность в лесе
  - Поддерживаем остовный лес
  - Как его перестроить при удалении ребра?
- Определим функцию L: E -> {1, ... logN}
- Определим граф $G_i = (V, E_i)$, $E_i = \{e, L(e) \leq i\}$
  - $E_i \subseteq E_{i+1}$
  - $E_{logN} = E$
- F_i – остовный лес G_i

Динамическая связность в графе

- Вернёмся к исходной задаче
- Будем использовать динамическую связность в лесе
  - Поддерживаем остовный лес
  - Как его перестроить при удалении ребра?
- Определим функцию L: E -> {1, ... logN}
- Определим граф $G_i = (V, E_i)$, $E_i = \{e, L(e) \le i\}$
  - $E_i \subseteq E_{i+1}$
  - $E_{logN} = E$
- F_i – остовный лес G_i
  - $F_i \subseteq F_{i+1}$

Динамическая связность в графе

- L: E -> {1, ... logN}
- $G_i = (V, E_i)$, $E_i = \{e, L(e) \leq i\}$
  - $E_i \subseteq E_{i+1}$
  - $E_{logN} = E$
- F_i – остовный лес G_i
  - $F_i \subseteq F_{i+1}$

Динамическая связность в графе

- L: E -> {1, ... logN}
- $G_i = (V, E_i)$, $E_i = \{e, L(e) \leq i\}$
  - $E_i \subseteq E_{i+1}$
  - $E_{logN} = E$
- F_i – остовный лес G_i
  - $F_i \subseteq F_{i+1}$

Условие:

- в каждой компоненте связности C леса F_i не более 2^i вершин

Динамическая связность в графе

- L: E -> {1, ... logN}
- $G_i = (V, E_i)$, $E_i = \{e, L(e) \le i\}$
  - $E_i \subseteq E_{i+1}$
  - $E_{logN} = E$
- F_i – остовный лес G_i
  - $F_i \subseteq F_{i+1}$

Условие:

  - в каждой компоненте связности C леса F_i не более 2^i вершин

- Потенциал структуры = ∑{ L(e) }

Динамическая связность в графе

- Проверка связности в G = проверка связности в лесе F_{logN}

Динамическая связность в графе

- Проверка связности в G = проверка связности в лесе F_{logN}
- Добавление ребра e = (u, v) в G:
  - L(e) - ?

Динамическая связность в графе

- Проверка связности в G = проверка связности в лесе F_{logN}
- Добавление ребра e = (u, v) в G:
  - L(e) = logN
    - Изменился только граф G_{logN}

Динамическая связность в графе

- Проверка связности в G = проверка связности в лесе F_{logN}
- Добавление ребра e = (u, v) в G:
  - L(e) = logN
    - Изменился только граф G_{logN}
    - Если компоненты G_{logN} объединились, то добавляем e в F_{logN}

Динамическая связность в графе

- Проверка связности в G = проверка связности в лесе F_{logN}
- Добавление ребра e = (u, v) в G:
  - L(e) = logN
    - Изменился только граф G_{logN}
    - Если компоненты G_{logN} объединились, то добавляем e в F_{logN}
      - Размер компонеты при этом остаётся допустимым

Динамическая связность в графе

- Проверка связности в G = проверка связности в лесе F_{logN}
- Добавление ребра e = (u, v) в G:
  - L(e) = logN
    - Изменился только граф G_{logN}
    - Если компоненты G_{logN} объединились, то добавляем e в F_{logN}
      - Размер компонеты при этом остаётся допустимым
    - Потенциал увеличился на logN

Динамическая связность в графе: удаление

- Удаление ребра e = (u, v)

Динамическая связность в графе: удаление

- Удаление ребра e = (u, v)
  - L(e) = i

Динамическая связность в графе: удаление

- Удаление ребра e = (u, v)
  - L(e) = i
  - Ребро нужно удалить на уровнях с i по logN

Динамическая связность в графе: удаление

- Удаление ребра e = (u, v)
  - L(e) = i
  - Ребро нужно удалить на уровнях с i по logN
- Наблюдения:
  - Если e есть в F_i, то оно есть во всех F_{i+1}, ... F_{logN}

Динамическая связность в графе: удаление

- Удаление ребра e = (u, v)
  - L(e) = i
  - Ребро нужно удалить на уровнях с i по logN
- Наблюдения:
  - Если e есть в F_i, то оно есть во всех F_{i+1}, ... F_{logN}
  - Если e нет в F_i, то его нет ни в каком из F_{i+1}, ... F_{logN}

Динамическая связность в графе: удаление

- Удаление ребра e = (u, v)
  - L(e) = i
  - Ребро нужно удалить на уровнях с i по logN
- Наблюдения:
  - Если e есть в F_i, то оно есть во всех F_{i+1}, ... F_{logN}
  - Если e нет в F_i, то его нет ни в каком из F_{i+1}, ... F_{logN}
- Во втором случае ребро просто удаляем

Динамическая связность в графе: удаление

- Удаление ребра e = (u, v)
  - L(e) = i
  - Ребро нужно удалить на уровнях с i по logN
- Наблюдения:
  - Если e есть в F_i, то оно есть во всех F_{i+1}, ... F_{logN}
  - Если e нет в F_i, то его нет ни в каком из F_{i+1}, ... F_{logN}
- Во втором случае ребро просто удаляем
  - При этом леса не приходится перестраивать

Динамическая связность в графе: удаление

- Удаление ребра e = (u, v)
  - L(e) = i
  - Ребро нужно удалить на уровнях с i по logN
- Наблюдения:
  - Если e есть в F_i, то оно есть во всех F_{i+1}, ... F_{logN}
  - Если e нет в F_i, то его нет ни в каком из F_{i+1}, ... F_{logN}
- Во втором случае ребро просто удаляем
  - При этом леса не приходится перестраивать
- Как удалить ребро, если оно присутствует в остовном лесе?

Динамическая связность в графе: удаление

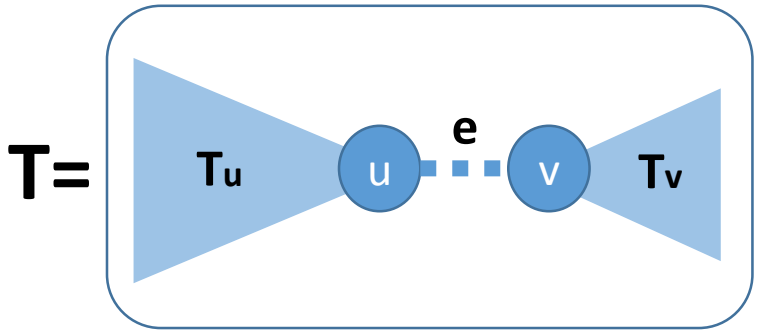
- Удаление ребра e = (u, v)
  - L(e) = i
  - Ребро e присутствует в F_i, F_{i+1}, ... F_{logN}

Динамическая связность в графе: удаление

- Удаление ребра e = (u, v)
  - L(e) = i
  - Ребро e присутствует в F_i, F_{i+1}, ... F_{logN}
- T = T_u + e + T_v – компонента леса F_i

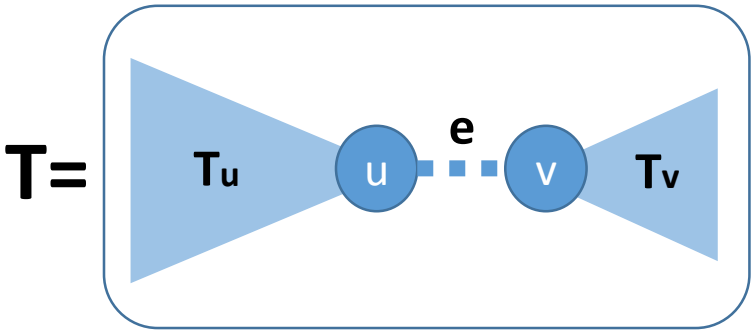
Динамическая связность в графе: удаление

- Удаление ребра e = (u, v)
  - L(e) = i
  - Ребро e присутствует в F_i, F_{i+1}, ... F_{logN}
- T = T_u + e + T_v – компонента леса F_i



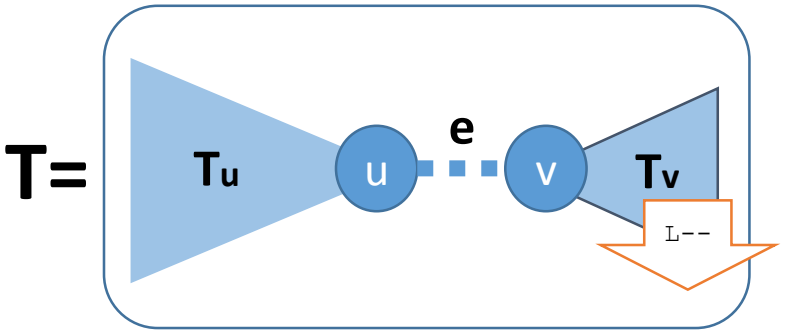
Динамическая связность в графе: удаление

- Удаление ребра e = (u, v)
  - L(e) = i
  - Ребро e присутствует в F_i, F_{i+1}, ... F_{logN}
- T = T_u + e + T_v – компонента леса F_i
  - Пусть T_v не превосходит T_u по числу вершин



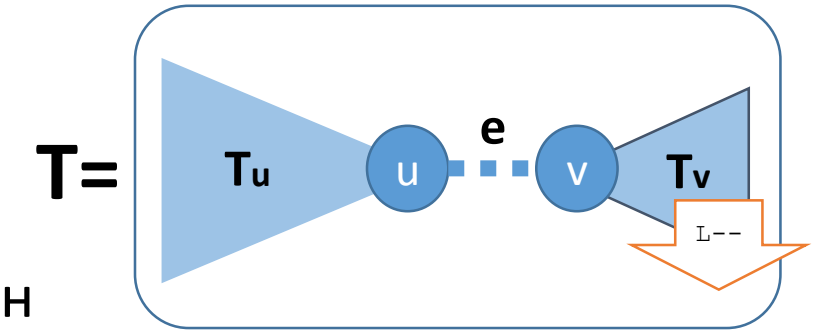
Динамическая связность в графе: удаление

- Удаление ребра e = (u, v)
  - L(e) = i
  - Ребро e присутствует в F_i, F_{i+1}, ... F_{logN}
- T = T_u + e + T_v – компонента леса F_i
  - Пусть T_v не превосходит T_u по числу вершин
  - Рёбра входящие в T_v перемещаем на уровень (i – 1)



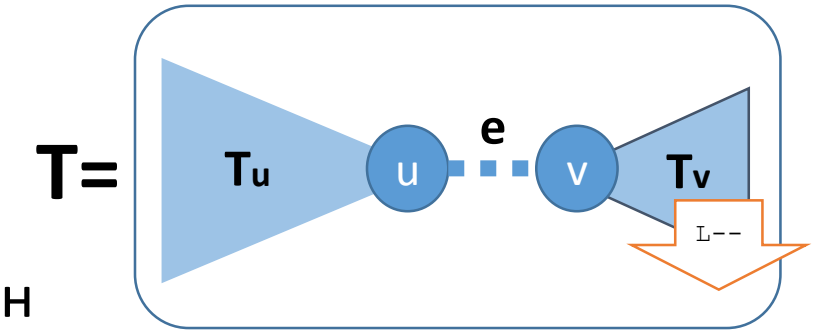
Динамическая связность в графе: удаление

- Удаление ребра e = (u, v)
  - L(e) = i
  - Ребро e присутствует в F_i, F_{i+1}, ... F_{logN}
- T = T_u + e + T_v – компонента леса F_i
  - Пусть T_v не превосходит T_u по числу вершин
  - Рёбра входящие в T_v перемещаем на уровень (i – 1)
    - Эти рёбра нужно добавить в остовный лес F_{i-1}



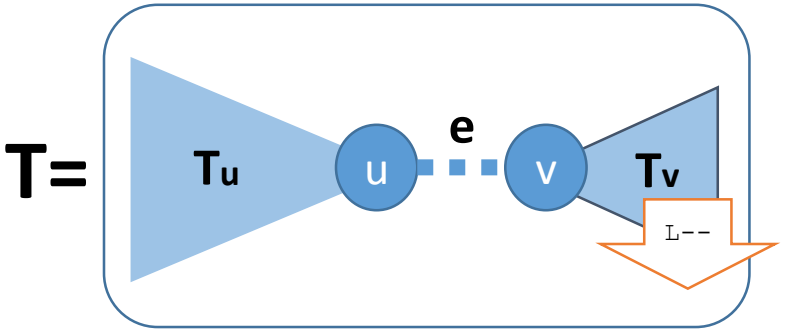
Динамическая связность в графе: удаление

- Удаление ребра e = (u, v)
  - L(e) = i
  - Ребро e присутствует в F_i, F_{i+1}, ... F_{logN}
- T = T_u + e + T_v – компонента леса F_i
  - Пусть T_v не превосходит T_u по числу вершин
  - Рёбра входящие в T_v перемещаем на уровень (i – 1)
    - Эти рёбра нужно добавить в остовный лес F_{i-1}
      - Добавление ребра оплачивается уменьшением потенциала



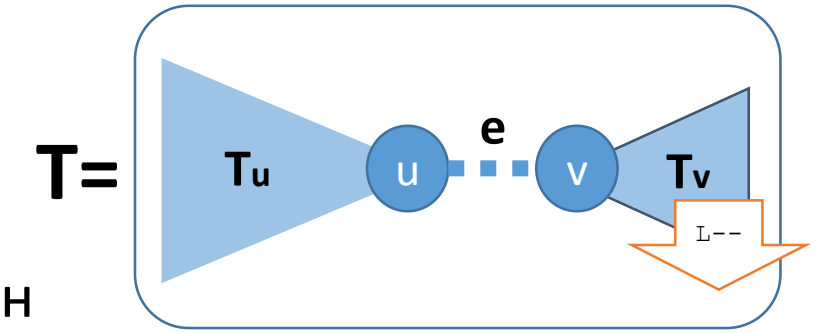
Динамическая связность в графе: удаление

- Удаление ребра e = (u, v)
  - L(e) = i
  - Ребро e присутствует в F_i, F_{i+1}, ... F_{logN}
- T = T_u + e + T_v – компонента леса F_i
  - Пусть T_v не превосходит T_u по числу вершин
  - Рёбра входящие в T_v перемещаем на уровень (i – 1)
    - Эти рёбра нужно добавить в остовный лес F_{i-1}
      - Добавление ребра оплачивается уменьшением потенциала
    - При этом ограничение на размер компоненты связности не нарушится



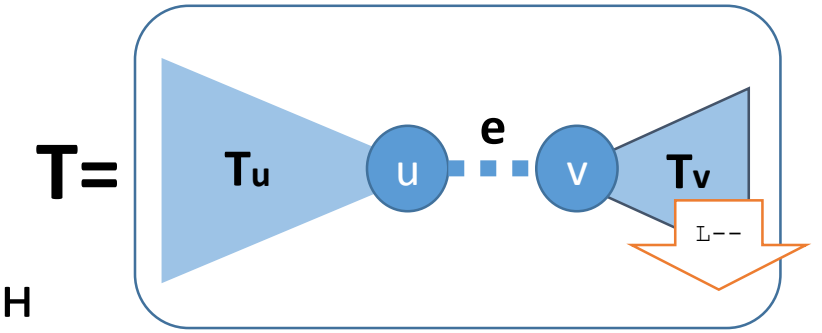
Динамическая связность в графе: удаление

- Удаление ребра e = (u, v)
  - L(e) = i
  - Ребро e присутствует в F_i, F_{i+1}, ... F_{logN}
- T = T_u + e + T_v – компонента леса F_i
  - Пусть T_v не превосходит T_u по числу вершин
  - Рёбра входящие в T_v перемещаем на уровень (i – 1)
    - Эти рёбра нужно добавить в остовный лес F_{i-1}
      - Добавление ребра оплачивается уменьшением потенциала
    - При этом ограничение на размер компоненты связности не нарушится
  - Нужно искать ребро для восстановления связности T на уровнях i, i+1, ... logN



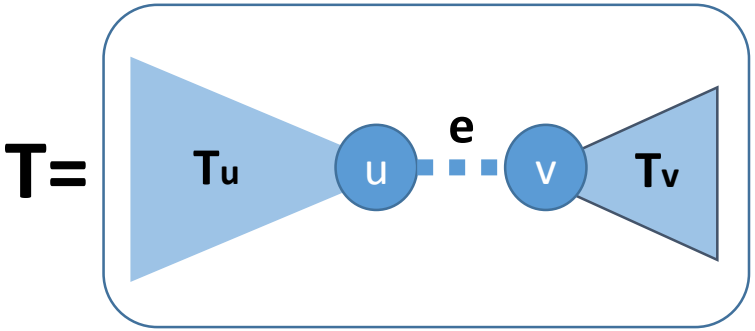
Динамическая связность в графе: удаление

- Удаление ребра e = (u, v)
  - L(e) = i
  - Ребро e присутствует в F_i, F_{i+1}, ... F_{logN}
- T = T_u + e + T_v – компонента леса F_i
  - Пусть T_v не превосходит T_u по числу вершин
  - Рёбра входящие в T_v перемещаем на уровень (i – 1)
    - Эти рёбра нужно добавить в остовный лес F_{i-1}
      - Добавление ребра оплачивается уменьшением потенциала
    - При этом ограничение на размер компоненты связности не нарушится
  - Нужно искать ребро для восстановления связности T на уровнях i, i+1, ... logN
    - У каждого просмотренного ребра понижаем уровень



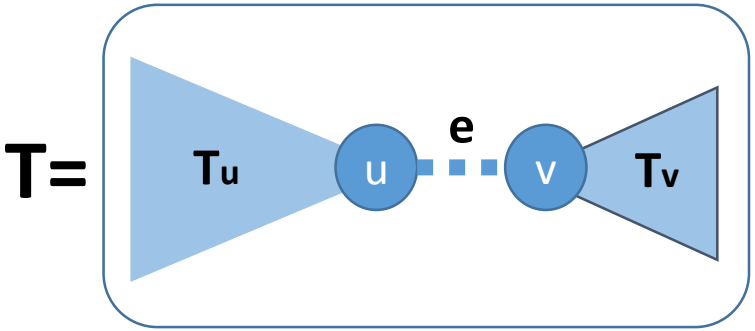
Динамическая связность в графе: удаление

- Ищем ребро для восстановления связности T на уровнях i, i+1, ... logN



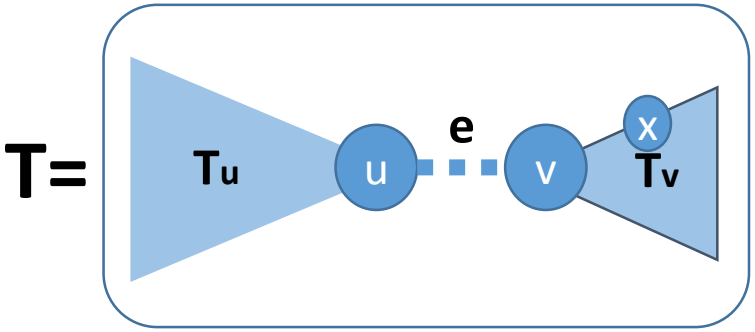
Динамическая связность в графе: удаление

- Ищем ребро для восстановления связности T на уровнях i, i+1, ... logN
  - Начинаем с уровня i



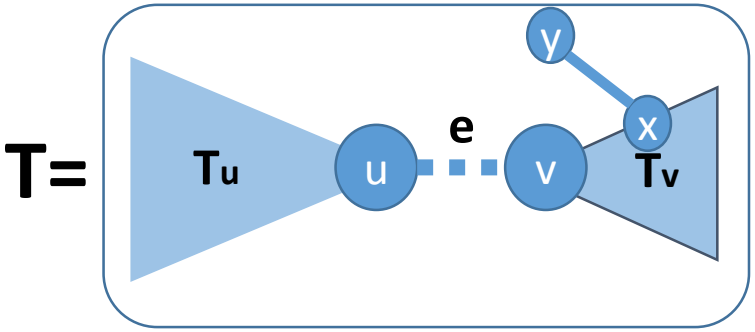
Динамическая связность в графе: удаление

- Ищем ребро для восстановления связности T на уровнях i, i+1, ... logN
  - Начинаем с уровня i
  - Просматриваем вершины x из T_v



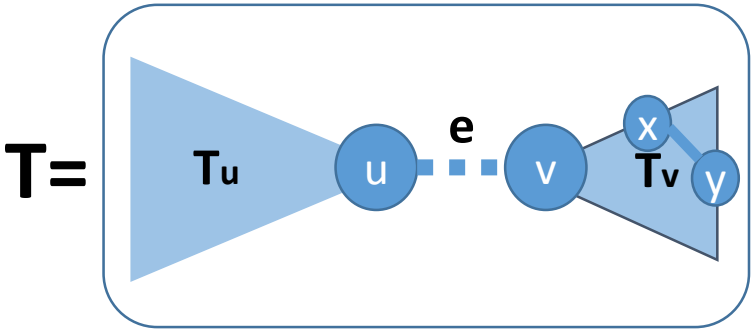
Динамическая связность в графе: удаление

- Ищем ребро для восстановления связности T на уровнях i, i+1, ... logN
  - Начинаем с уровня i
  - Просматриваем вершины x из T_v
  - Просматриваем все ребра (x, y)



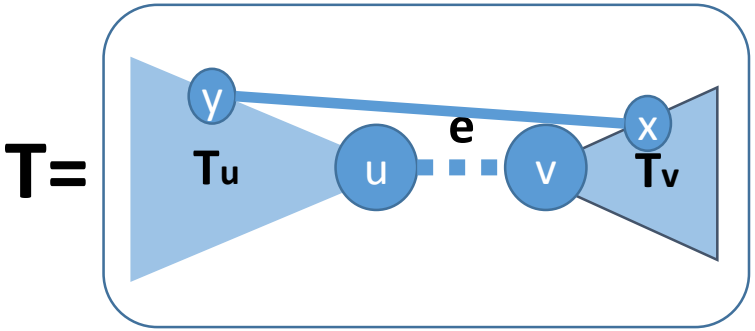
Динамическая связность в графе: удаление

- Ищем ребро для восстановления связности T на уровнях i, i+1, ... logN
  - Начинаем с уровня i
  - Просматриваем вершины x из T_v
  - Просматриваем все ребра (x, y)
    - y лежит в T_v -> понижаем уровень (x, y)



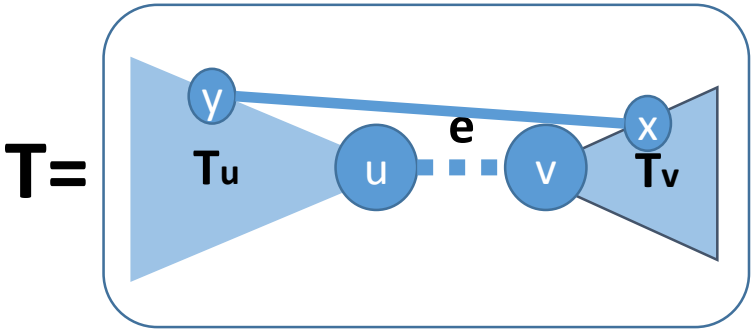
Динамическая связность в графе: удаление

- Ищем ребро для восстановления связности T на уровнях i, i+1, ... logN
  - Начинаем с уровня i
  - Просматриваем вершины x из T_v
  - Просматриваем все ребра (x, y)
    - y лежит в T_v -> понижаем уровень (x, y)
    - y лежит в T_u -> успех



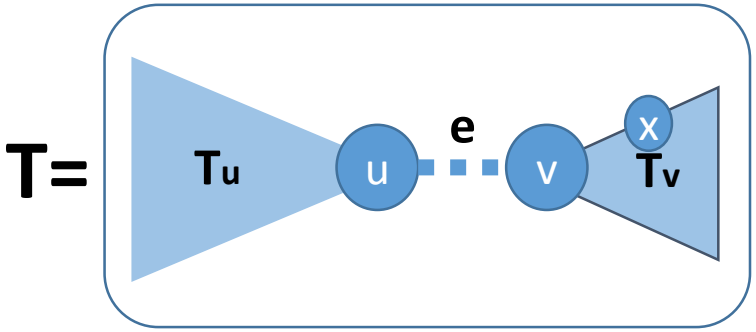
Динамическая связность в графе: удаление

- Ищем ребро для восстановления связности T на уровнях i, i+1, ... logN
  - Начинаем с уровня i
  - Просматриваем вершины x из T_v
  - Просматриваем все ребра (x, y)
    - y лежит в T_v -> понижаем уровень (x, y)
    - y лежит в T_u -> успех
      - добавляем (x, y) в F_i, ... F_{i+1}



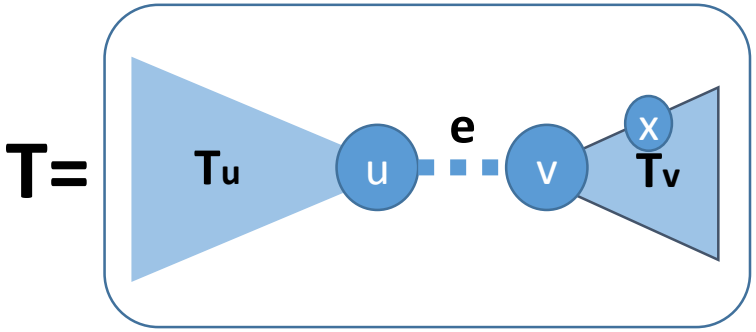
Динамическая связность в графе: удаление

- Ищем ребро для восстановления связности T на уровнях i, i+1, ... logN
  - Начинаем с уровня i
  - Просматриваем вершины x из T_v
  - Просматриваем все ребра (x, y)
    - y лежит в T_v -> понижаем уровень (x, y)
    - y лежит в T_u -> успех
      - добавляем (x, y) в F_i, ... F_{i+1}
    - Рёбра закончились -> переходим на следующий уровень



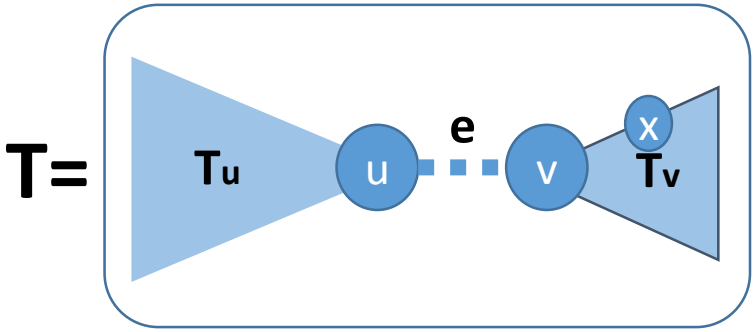
Динамическая связность в графе: удаление

- Ищем ребро для восстановления связности T на уровнях i, i+1, ... logN
  - Начинаем с уровня i
  - Просматриваем вершины x из T_v
  - Просматриваем все ребра (x, y)
    - y лежит в T_v -> понижаем уровень (x, y)
    - y лежит в T_u -> успех
      - добавляем (x, y) в F_i, ... F_{i+1}
    - Рёбра закончились -> переходим на следующий уровень
      - Уровни закончились -> компонента гарантированно распалась



Динамическая связность в графе: удаление

- Ищем ребро для восстановления связности T на уровнях i, i+1, ... logN
  - Начинаем с уровня i
  - Просматриваем вершины x из T_v
  - Просматриваем все ребра (x, y)
    - y лежит в T_v -> понижаем уровень (x, y)
    - y лежит в T_u -> успех
      - добавляем (x, y) в F_i, ... F_{i+1}
    - Рёбра закончились -> переходим на следующий уровень
      - Уровни закончились -> компонента гарантированно распалась



Амортизированное время работы $O(\log^2 N)$