ITIVITI

LET IT TRADE

Алгоритмы и структуры данных

Косяков Михаил Сергеевич
к.т.н., доцент кафедры ВТ

УНИВЕРСИТЕТ ИТМО

Содержание курса

- Введение в теорию алгоритмов
- Алгоритмы сортировок
- Структуры данных
  - Линейные структуры
  - Бинарные деревья поиска
  - Хеши и хеш-функции
- **Алгоритмы на графах**
  - Обходы графов в ширину и глубину
  - **Минимальные остовные деревья**
  - **Поиск кратчайших путей в графе**

ITIVITI

LET IT TRADE

Поиск кратчайших путей в графе

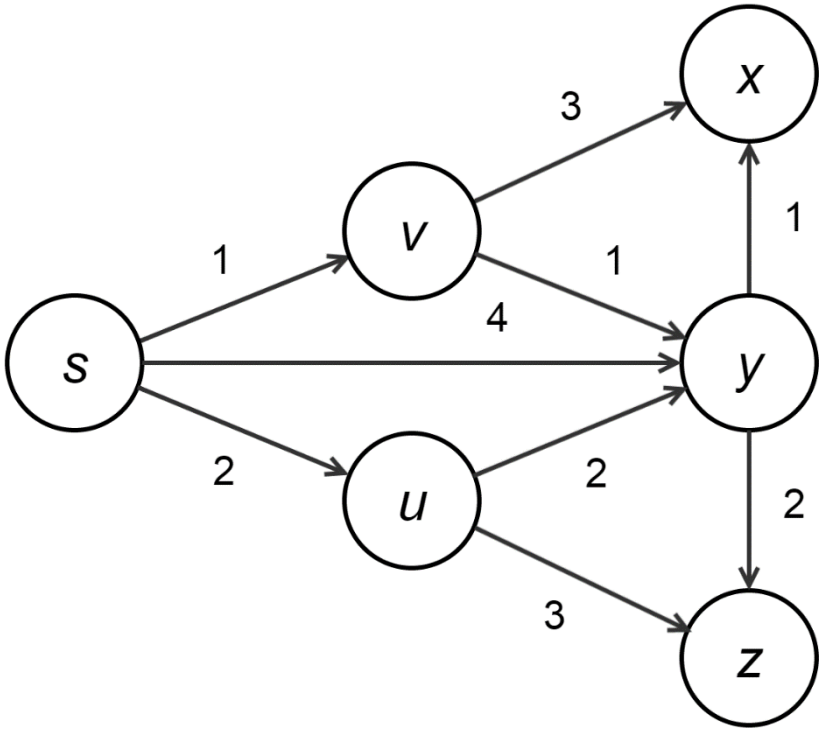
УНИВЕРСИТЕТ ИТМО

Поиск кратчайших путей

- Примеры:
  - Транспортные сети и автоматическая навигация (в т.ч. компьютерные игры)
  - Маршрутизация в компьютерных сетях
  - Финансовые транзакции
- Кратчайшие пути из одной вершины:
  - Задан взвешенный орграф $G = (V, E)$ и вершина-источник s
  - Каждое ребро (u, v) имеет вес $weight(u, v) \geq 0$
  - Для каждой вершины t требуется найти кратчайший путь из s

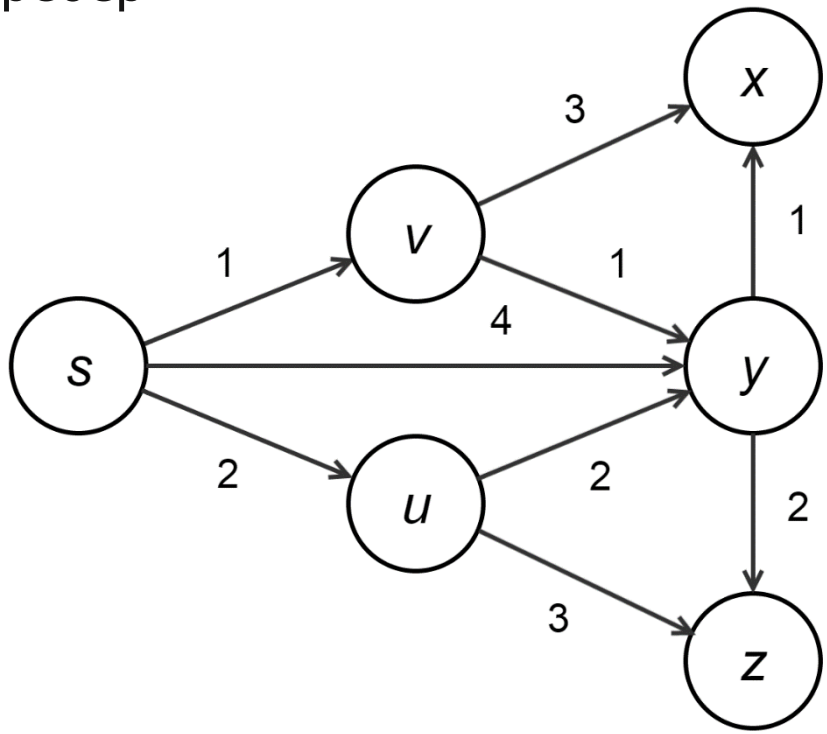
Кратчайшие пути из одной вершины

- Вес пути – суммарный вес входящих в него ребер
- Какие есть пути из s в x? Какие их веса?



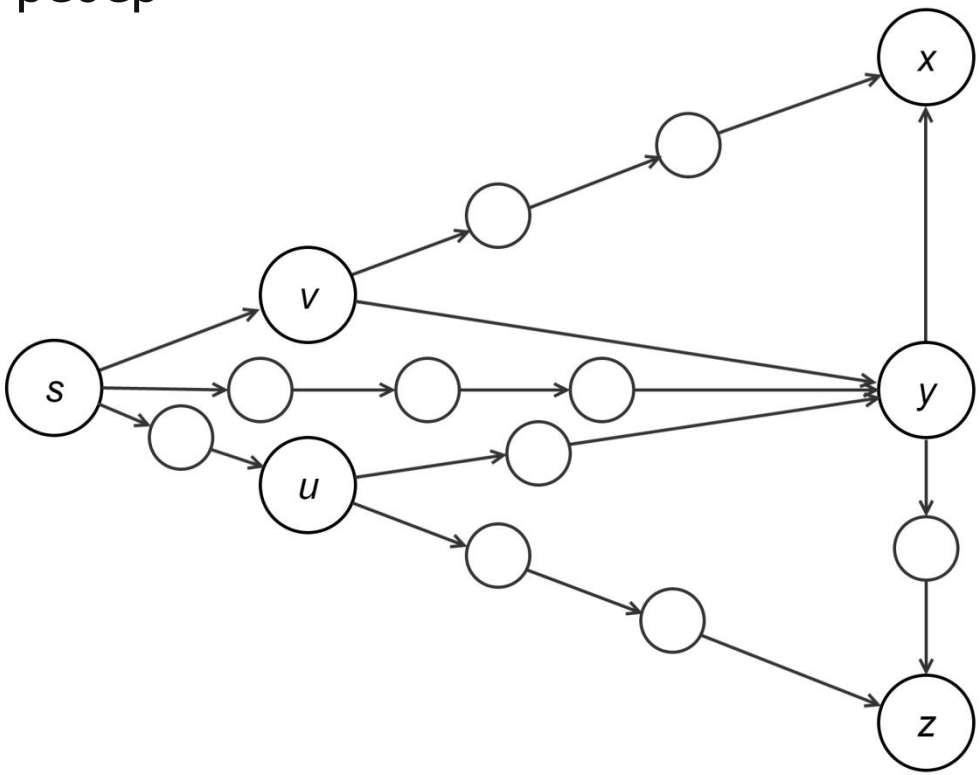
Кратчайшие пути из одной вершины

- Какой алгоритм позволяет вычислять кратчайшие пути?
- Давайте заменим ребро с весом k на k - 1 вершину и k ребер



Кратчайшие пути из одной вершины

- Какой алгоритм позволяет вычислять кратчайшие пути?
- Давайте заменим ребро с весом k на k - 1 вершину и k ребер



Кратчайшие пути из одной вершины

ITIVITI

- Давайте заменим ребро с весом k на k - 1 вершину и k ребер
- Пусть W – общий вес графа, тогда
  - число ребер в новом графе $m' = W$
  - число вершин в новом графе $n' = n + W - m$
- Время работы $O(n' + m') = O(n + W)$
- Как обрабатывать нецелые веса?
- Решение: алгоритм Дейкстры

LET IT TRADE

УНИВЕРСИТЕТ ИТМО

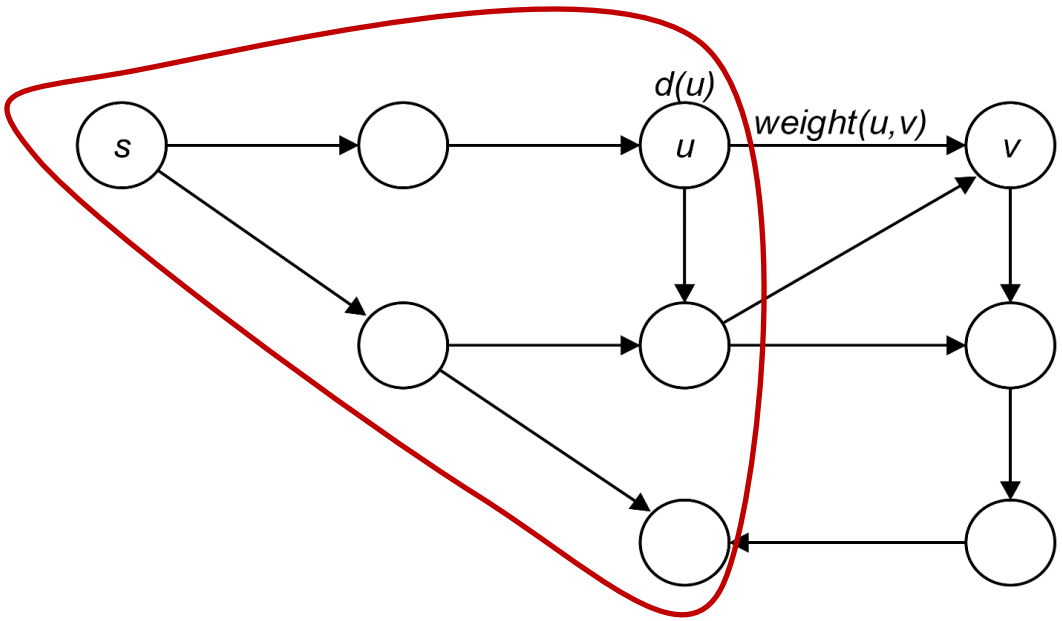
Алгоритм Дейкстры

01: X – множество вершин с вычисленными весами $d(x)$

02: Инициализация: $X = \{s\}$, $d(s) = 0$

03: while ($X \neq V$) {

04: среди всех ребер (u, v) таких, что $u \in X$ и $v \in V-X$



- Среди каких ребер на рисунке ищем?

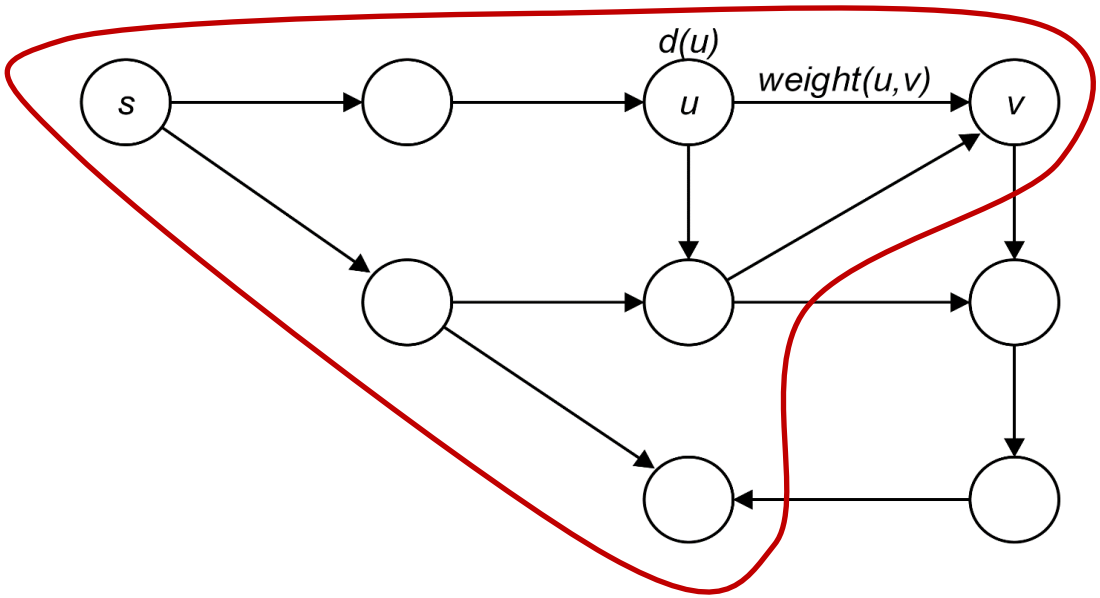
Алгоритм Дейкстры

05: выбрать одно, которое имеет минимальное

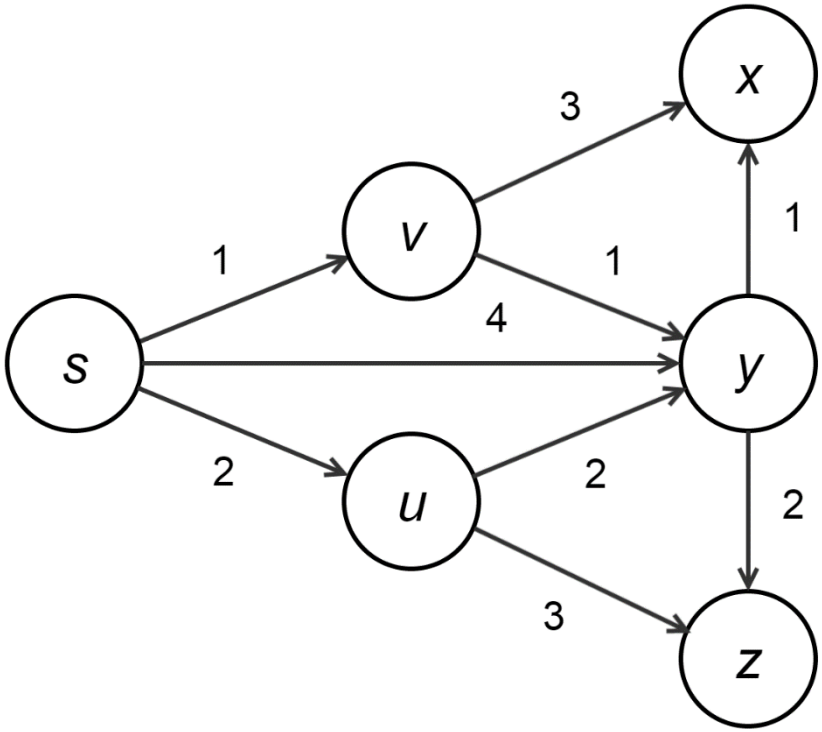
06: значение $DS(v) = d(u) + weight(u, v)$

07: добавить v в X, $d(v) = DS(v)$, $predecessor(v) = u$

08: }



Алгоритм Дейкстры: пример выполнения



- Как формируется путь $P(s \rightsquigarrow u)$?
- Как напечатать построенный путь $P(s \rightsquigarrow u)$?

Алгоритм Дейкстры: корректность

ITIVITI

LET IT TRADE

- **Теорема:** Рассмотрим множество X на любом шаге алгоритма. Для любой вершины u построенный путь $P(s \rightsquigarrow u)$ – кратчайший из s в u
- Следствие: Если $X = V$, то …
- База индукции:
  - $X = \{s\}$, $d(s) = 0$
- Шаг индукции:
  - пусть v – следующая добавляемая вершина через ребро (u, v)
  - $P(s \rightsquigarrow u)$ – кратчайший путь из s в u
  - покажем, что $P(s \rightsquigarrow v)$ – кратчайший путь

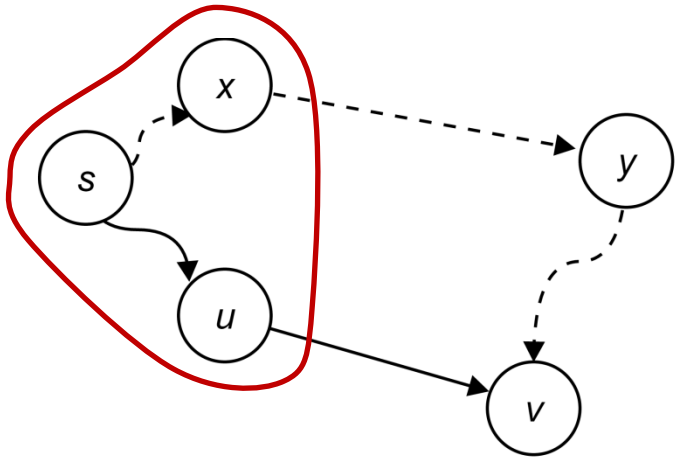
УНИВЕРСИТЕТ ИТМО

Алгоритм Дейкстры: корректность

ITIVITI

LET IT TRADE

- Пусть P – альтернативный путь из s в v и (x, y) – первое ребро, выходящее из X, на пути P
- Обозначим P' – часть пути P от s до x



- $weight(P) \geq weight(P') + weight(x, y) \geq$
  $\geq d(x) + weight(x, y) = DS(y) \geq DS(v)$

УНИВЕРСИТЕТ ИТМО

Алгоритм Дейкстры: реализация

ITIVITI

- Какое время выполнения прямой реализации алгоритма Дейкстры, представленной выше?
  - $\Theta(m \log n)$
  - $\Theta(n\, m)$
  - $\Theta(n + m)$
  - $\Theta(n^2)$
- Как нам улучшить время работы, не меняя самого алгоритма?
- Какие структуры данных нам помогут?

LET IT TRADE

Алгоритм Дейкстры: реализация

ITIVITI

LET IT TRADE

- Какое время выполнения прямой реализации алгоритма Дейкстры, представленной выше?
  - $\Theta(m \log n)$
  - $\Theta(n\, m)$
  - $\Theta(n + m)$
  - $\Theta(n^2)$
- Как нам улучшить время работы, не меняя самого алгоритма?
- Какие структуры данных нам помогут?
- Приоритетная очередь! Какие поддерживаемые операции?
- Что будем хранить в очереди?

Алгоритм Дейкстры: реализация

ITIVITI

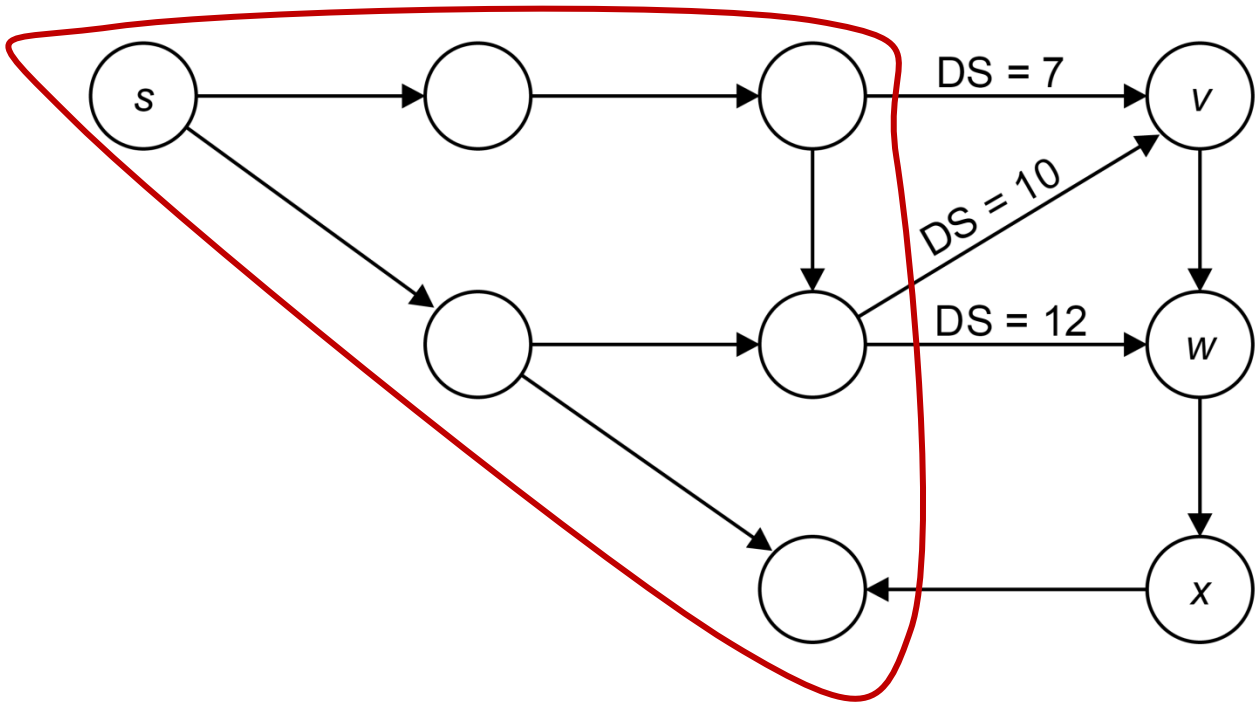
LET IT TRADE

- Элементы приоритетной очереди: вершины множества $V - X$
- Что будет ключом элемента?
- Ключ вершины $v \in V - X$: наименьшее значение $DS(v)$ среди всех вершин $u \in X$, имеющих ребро (u, v), входящее в v
- Тогда `Extract_Min()` будет возвращать правильную вершину v для включения в множество X
- При этом $d(v)$ будет определяться ключом v

УНИВЕРСИТЕТ ИТМО

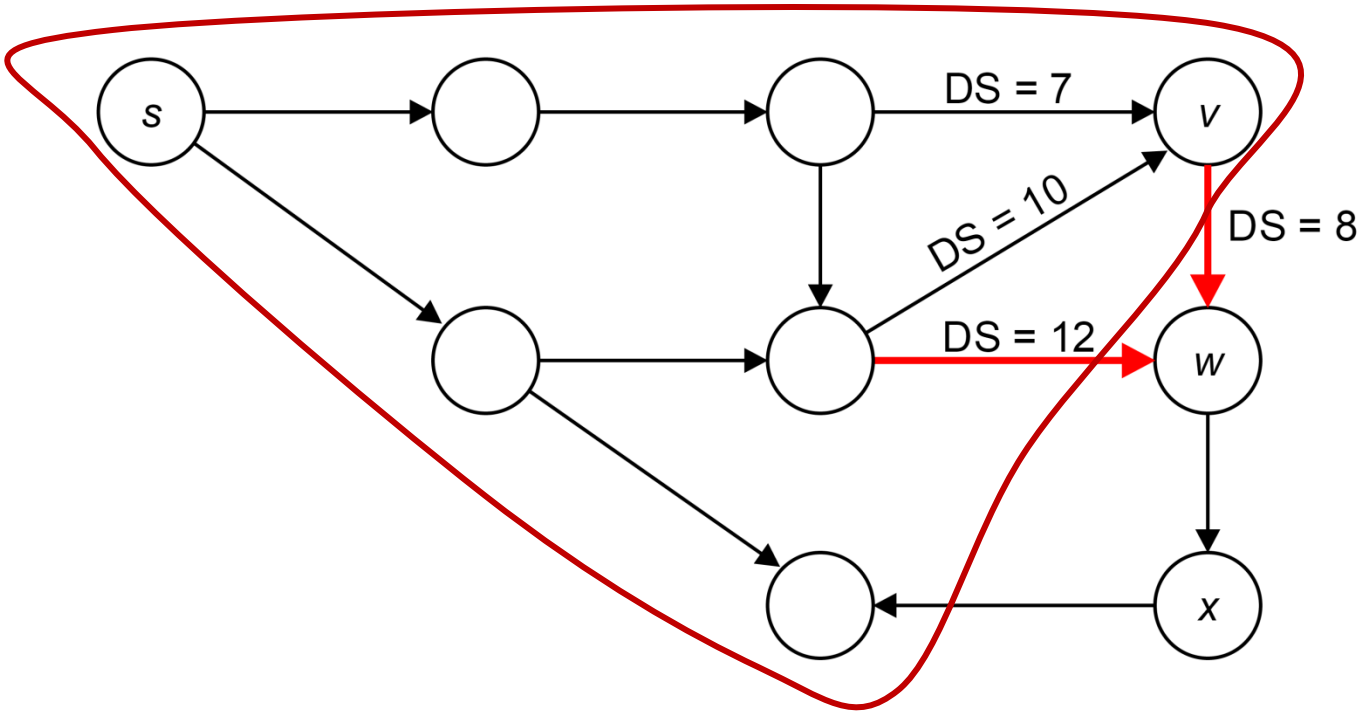
Алгоритм Дейкстры: реализация

- Какие ключи у вершин *v*, *w*, *x*?
- Как будут меняться ключи вершин при включении вершины *v* в *X*?



Алгоритм Дейкстры: реализация

- Как должен измениться ключ вершины *w*?
- Вывод: необходимо обновлять ключи вершин, смежных с добавляемой, с помощью `Decrease_Key()`!



Алгоритм Дейкстры: реализация


```
01: for each v in V:
02:  v.d = ∞ # кратчайшее расстояние до s
03:  v.p = 0 # вершина, предшествующая v
04: X = {}   # множество обработанных вершин
05: s.d = 0
06: s.p = s
07: Q = PriorityQueue(V)
08:
09: while !Q.empty():
10:  u = Q.Extract_Min()
11:  X = X ∪ {u}
12:  for each v in G[u]:
13:    if v.d > u.d + weight(u,v):
14:    # true only if v is in Q
15:      v.d = u.d + weight(u,v)
16:      v.p = u
17:      Q.Decrease_Key(v)
```


Алгоритм Дейкстры

- Какое будет время выполнения?
  - n операций `Insert()`
  - n операций `Extract_Min()`
  - не более m операций `Decrease_Key()`
- На бинарной пирамиде?
  - $O((n + m) \log n)$
- На массиве?
  - $O(n^2 + m)$

ITIVITI

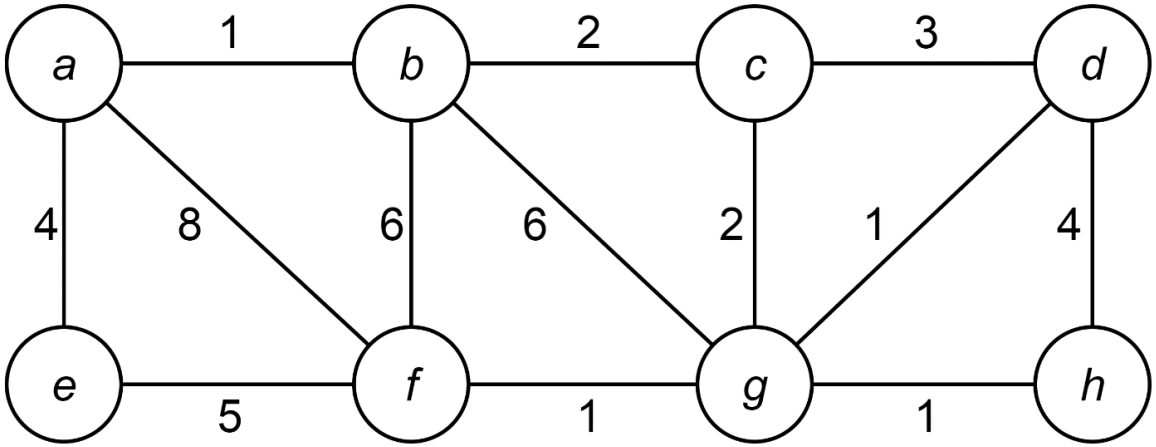
LET IT TRADE

Минимальные остовные деревья

УНИВЕРСИТЕТ ИТМО

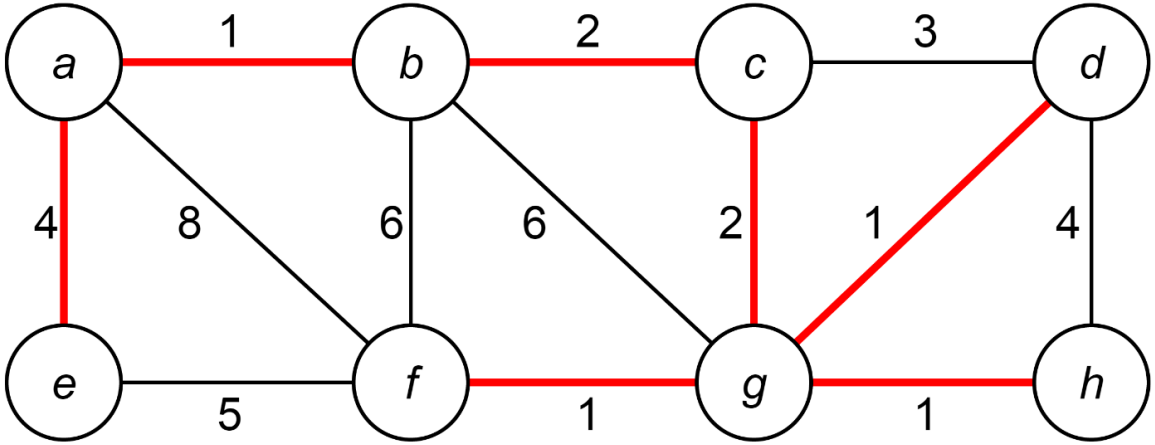
Минимальные остовные деревья

- Хотим соединить объекты наиболее дешевым способом
  - Задан связный взвешенный неориентированный граф $G = (V, E)$
  - Каждое ребро (u, v) имеет вес $cost(u, v)$
  - Требуется найти остовное дерево наименьшего общего веса



Минимальные остовные деревья

- Хотим соединить объекты наиболее дешевым способом
  - Задан связный взвешенный неориентированный граф $G = (V, E)$
  - Каждое ребро (u, v) имеет вес $cost(u, v)$
  - Требуется найти остовное дерево наименьшего общего веса



Минимальные остовные деревья

- Если для всех ребер $cost(u, v) = 1$ (невзвешенный граф), то любое дерево – MST (Minimum Spanning Tree)
  - можно построить с помощью DFS
  - время работы $O(n + m)$
- Давайте заменим ребро с весом k на k - 1 вершину и k ребер
  - время работы $O(n + W)$
  - как обрабатывать нецелые веса?
- Решение: нужен другой алгоритм
- Предположение: все веса ребер различны

Алгоритм Прима

01: X – множество вершин MST, T – множество ребер MST

02: Инициализация: $X = \{$произвольная вершина $s\}$, $T = \{\}$

03: while ($X \neq V$) {

04: среди всех ребер (u, v) таких, что $u \in X$ и $v \in V-X$

05: выбрать одно, которое имеет минимальное

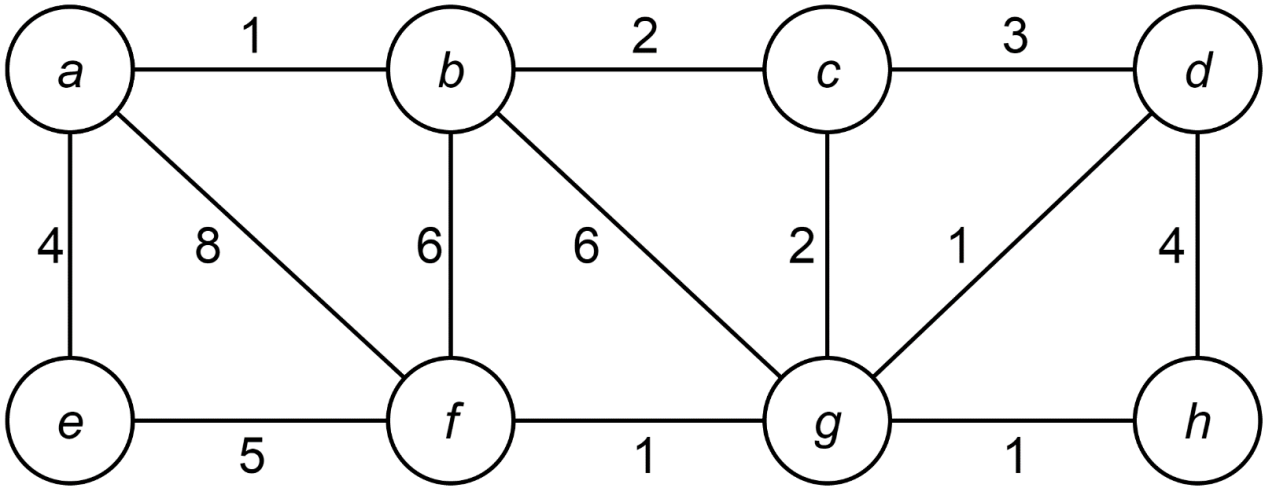
06: значение $cost(u, v)$

07: добавить v в X, добавить ребро в T, $predecessor(v) = u$

08: }

- Растим дерево – одно ребро за одну итерацию
- Сравните с алгоритмом Дейкстры:
  - добавляем вершину, ближайшую к источнику, добавляем вершину, ближайшую к дереву

Алгоритм Прима: пример выполнения



- Почему это работает?
  - Алгоритм Прима строит остовное дерево?
  - Это дерево имеет наименьший общий вес?

Алгоритм Прима: корректность

ITIVITI

- Остовное? Алгоритм покроет все вершины?
- Разрез (*cut*) неориентированного графа $G = (V, E)$ – разбиение V на два непустых множества X и $V - X$
- Ребро пересекает (*crosses*) разрез $(X, V - X)$, если один из концов принадлежит X, другой $V - X$
- Граф несвязный, тогда и только тогда, когда существует разрез без пересекающих ребер
  - Другой способ рассуждений о связности / несвязности графа
- Вывод: перебирая пересекающие ребра, алгоритм покроет все вершины V связного графа G

LET IT TRADE

УНИВЕРСИТЕТ ИТМО

Алгоритм Прима: корректность

ITIVITI

LET IT TRADE

- Дерево? Алгоритм не создает циклов?
- Для любого разреза и любого цикла верно, что число ребер цикла, пересекающих разрез, четно: 0, 2, …
- Следствие: если ребро (u, v) единственное, пересекающее какой-либо разрез, то оно не принадлежит никакому циклу
  - Надо найти такой разрез
- Вывод: на каждой итерации ребра из T не пересекают разрез (X, $V - X$), и выбирается одно пересекающее ребро, поэтому алгоритм не создает циклов в T

УНИВЕРСИТЕТ ИТМО

Алгоритм Прима: корректность

ITIVITI

LET IT TRADE

- Дерево минимального веса?
- **Теорема (свойство разреза):** рассмотрим ребро $e = (u, v)$ и предположим, что существует такой разрез, что e является пересекающим с наименьшим весом, тогда $e \in$ MST
- По построению алгоритма, он включает в T ребра из MST => $T \subseteq$ MST
  - Т.е. на каждом шаге алгоритм делает правильный выбор
- Но T – остовное дерево => $T =$ MST
- Если все веса ребер различны, сколько м.б. MST ?

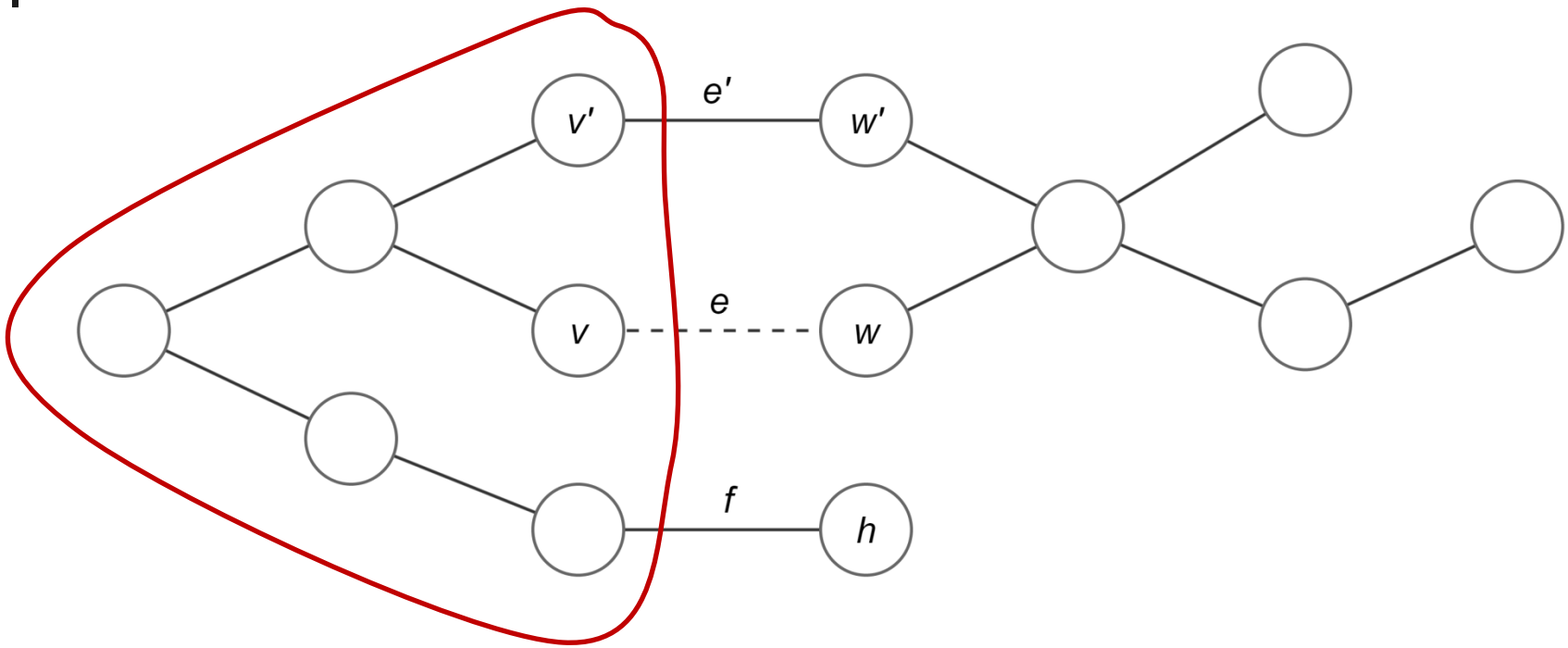
Свойство разреза: доказательство

ITIVITI

LET IT TRADE

- От противного: пусть $e \notin$ MST
- Рассмотрим MST без ребра e
  - … точнее, рассмотрим пересекающее ребро $f \in$ MST
  - почему пересекающее ребро $f \in$ MST существует?
- Если из MST удалить f но добавить e и получить остовное дерево веса меньшего, чем вес MST ($cost(f) > cost(e)$), то получим противоречие
- Но MST $\cup \{e\} - \{f\}$ будет остовным деревом?
- Как выбрать $f \in$ MST ?

Свойство разреза: доказательство



- Что будет, если e поменять с f ?
- Что будет, если e поменять с e'?
  - $cost(e) < cost(e')$

Свойство разреза: доказательство

ITIVITI

LET IT TRADE

Давайте поменяем пересекающее ребро e с пересекающим ребром e', которое входит в тот же цикл, что и e, образующийся при добавлении e в остовное дерево T

УНИВЕРСИТЕТ ИТМО

Алгоритм Прима: реализация

ITIVITI

LET IT TRADE

Аналогично алгоритму Дейкстры

- В приоритетной очереди храним ребра
  - Ключ ребра: его вес
  - Но надо проверять, является ли ребро пересекающим
- В приоритетной очереди храним вершины множества $V - X$
  - Ключ вершины $v \in V - X$: наименьший вес ребра среди всех ребер (u, v), $u \in X$
  - При включении вершины v в X необходимо обновить ключи смежных с v вершин $w \in V - X$ с помощью `Decrease_Key()`

УНИВЕРСИТЕТ ИТМО

ITIVITI

LET IT TRADE

Спасибо за внимание!

УНИВЕРСИТЕТ ИТМО