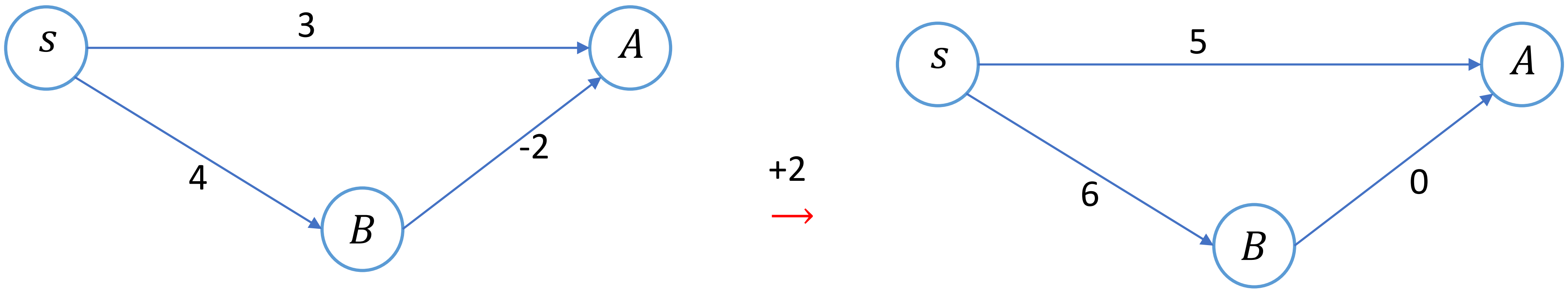
Семинар 8

Кратчайшие пути в графах с отрицательными весами

Пример, когда алгоритм Дейкстры не работает



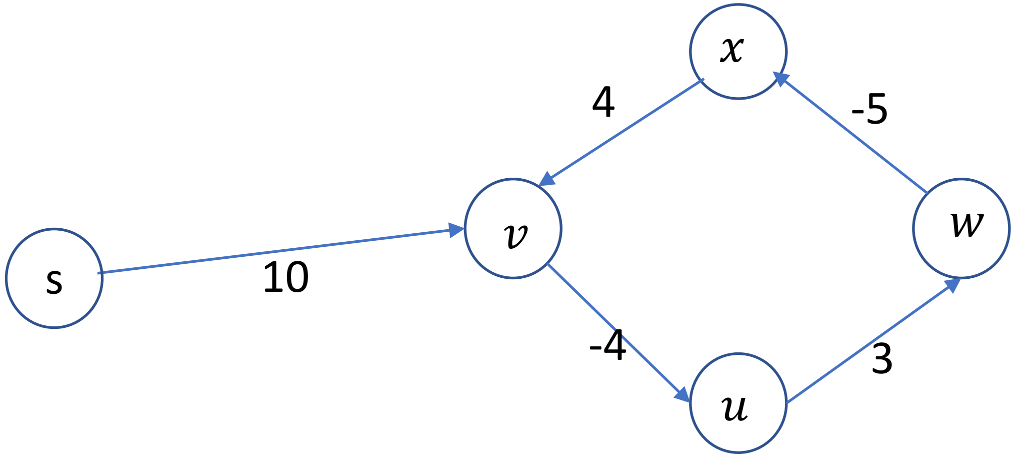
Присваиваем вершинам метки $\min\{v_m, v_{m-1} + l(v_{m-1}, v_m)\}$ (как в алгоритме Дейкстры), но можем их корректировать. Нет разбиения на X и V-X.
Максимальная длина пути (по количеству пройденных ребер) $|V| - 1$. Сделаем такое количество проходов, корректируя метки.
Если перебирать ребра в порядке SA, SB, BA, то за 1 проход найдем расстояния до всех вершин. $(A(3), B(4), A(2))$. Но обычно хороший порядок мы не знаем. Пусть порядок такой: BA, SB, SA.

Имеем 3-1=2 итерации:

итерации / Вершина	0	1	2
S	0	0	0
B	∞	4	4
A	∞	3	2

Отрицательный цикл

Если в графе есть отрицательный цикл, то кратчайший путь определить невозможно.



Если на итерации с номером $|V|$ какие-либо метки продолжают уменьшаться, значит в графе есть отрицательный цикл.

Вершина \ итерации	0	1	2	3	4	5
s	0	0	0	0	0	0
v	∞	10	10	10	10	8
x	∞					
w	∞					
u	∞					

Алгоритм Беллмана-Форда

Вход: орграф G=(*V*, *E*), $\forall x \;\; l(x) \in \mathbb{R}, \mathrm{s} \in V$

Выход: $\forall x \in V \quad len(v) = dist(s, v)$ либо объявление, что граф содержит отрицательный цикл

for каждой $v \neq s$ do

  $len(v) := \infty; \quad prev[v] := nil$

  $len[s] := 0$

for $i := 1$ to $|V| - 1$ do // перебрать $|V| - 1$ раз все ребра

  for каждого ребра $(u, v) \in E$

    if $len[v] > len[u] + l_{uv}$ then

      $len[v] \coloneqq len[u] + l_{uv}$

      $prev[v] \coloneqq u$

for каждого ребра $(u, v) \in E$

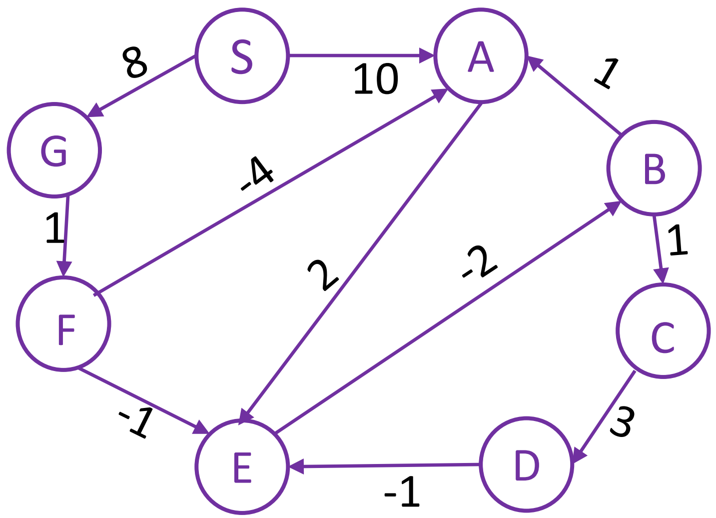
    if $len[v] > len[u] + l_{uv}$ then

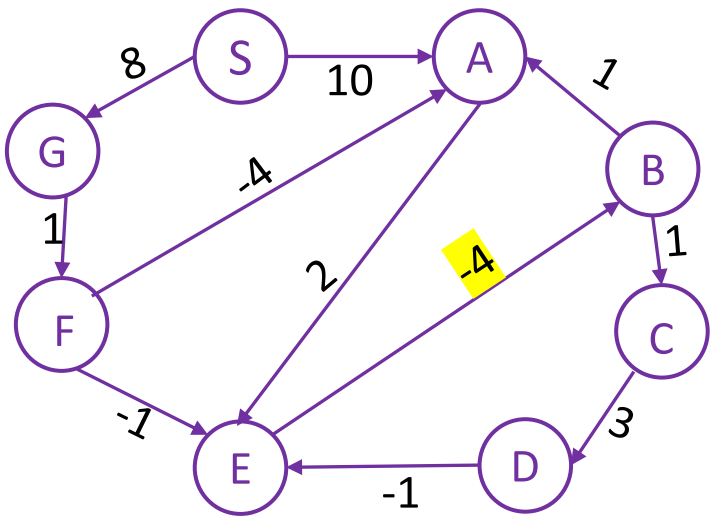
      return false // есть цикл отрицательной длины

return true

Время?

Пример.



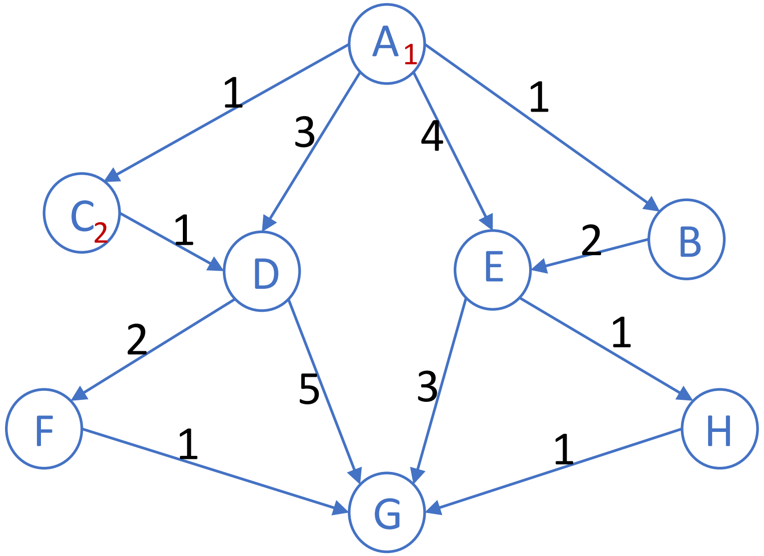


итерации / Вершина	0	1	2	3	4	5	6	7
S	0	0	0	0	0	0	0	0
A	∞	10S	10S	5F	5F	5F	5F	5F
B	∞	∞	∞	10E	6E	5E	5E	5E
C	∞							
D	∞							
E	∞							
F	∞							
G	∞							

В каких графах наверняка нет циклов отрицательного веса?

- Графы без отрицательных ребер -> алгоритм Дейкстры
- Ациклические графы

Применим к такому графу топологическую сортировку (DFS) и пронумеруем вершины. Пройдемся по вершинам в найденном порядке, просматривая (по одному разу) исходящие ребра и обновляя оценку $len[v_m] = \min\{len[v_m],\ len[v_{m-1}] + l(v_{m-1}, v_m)\}$. Т.к. граф топологически отсортирован, на шаге m всем вершинам с номером, меньшим m, уже присвоены значения меток.



Поиск в глубину

вершина	A	B	C	D	E	F	G	H
№ в топ.сорт				.			8	

Кратчайшие пути

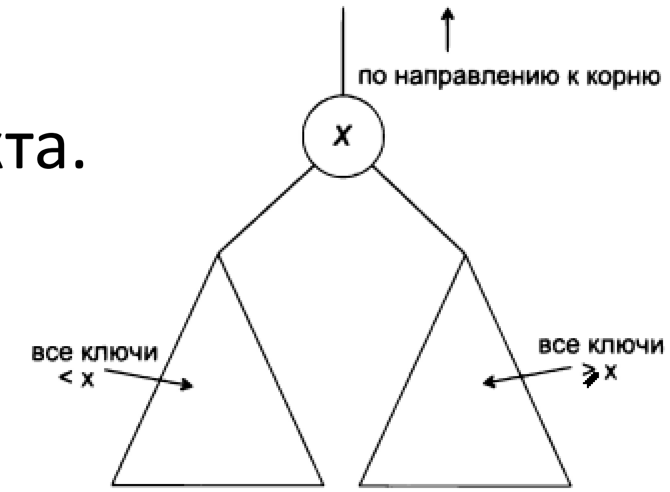
№вершины m	1	2	3	4	5	6	7	8
$len[v_m]$	0	1	2	.	.			

Время?

Бинарные деревья поиска

Ключ левого потомка меньше ключа объекта.

Ключ правого потомка не меньше ключа объекта.



Сбалансированные деревья поиска и отсортированные массивы: поддерживаемые операции и их время выполнения, где n обозначает текущее число объектов, хранящихся в структуре данных

Операция	Отсортированный массив	Сбалансированное дерево поиска
Отыскать	$O(\log n)$	$O(\log n)$
Минимум	$O(1)$	$O(\log n)$
Максимум	$O(1)$	$O(\log n)$
Предшественник	$O(\log n)$	$O(\log n)$
Преемник	$O(\log n)$	$O(\log n)$
Вывести в отсортированном порядке	$O(n)$	$O(n)$
Выбрать	$O(1)$	$O(\log n)$
Взять ранг	$O(\log n)$	$O(\log n)$
Вставить	$O(n)$	$O(\log n)$
Удалить	$O(n)$	$O(\log n)$

Задание

Начертить бинарные деревья поиска с высотой 2, 3, 4, 5, 6 для множества ключей {1, 4, 5, 10 16, 17, 21}

Обходы дерева: прямой, симметричный и обратный