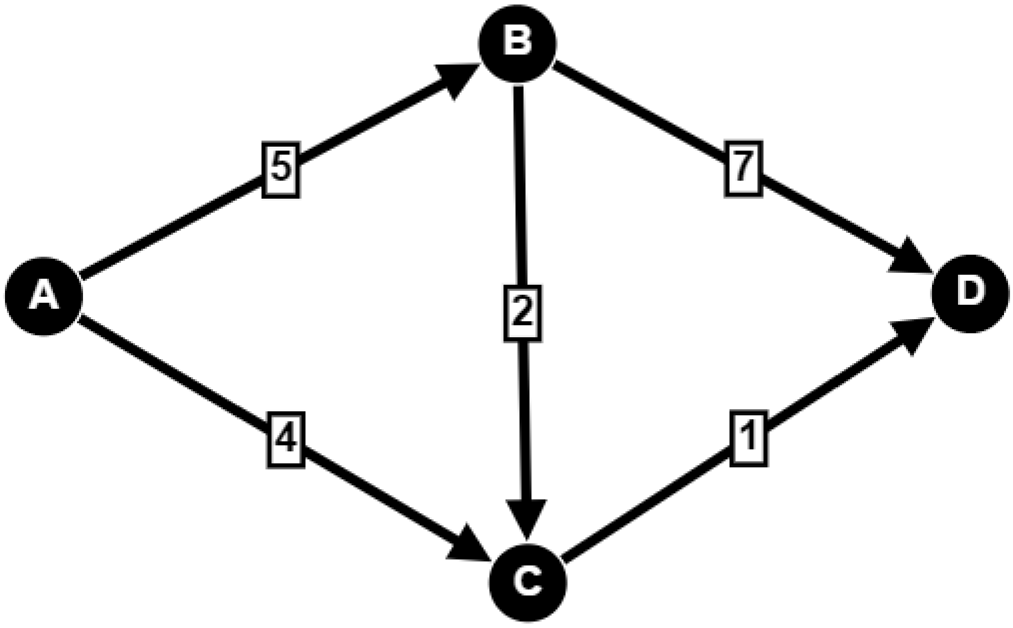
Задачи на нахождение различных маршрутов.

Выполнил: Селуянов Данила, гр. 932102

Задача 2.1. Алгоритм Дейкстры.
Найти кратчайшие пути от одной из вершин графа до всех остальных



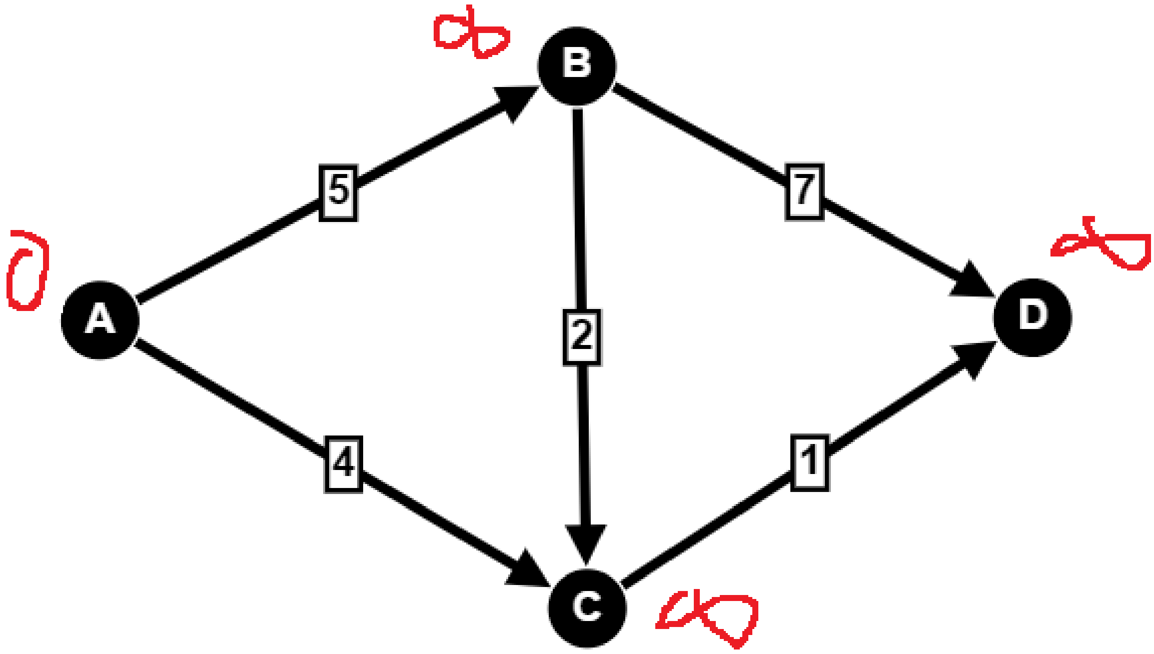
Для графа выше рассмотрим реализацию алгоритма Дейкстры по поиску кратчайшего пути из вершины **A** до всех остальных.

Работа алгоритма в виде таблицы итераций

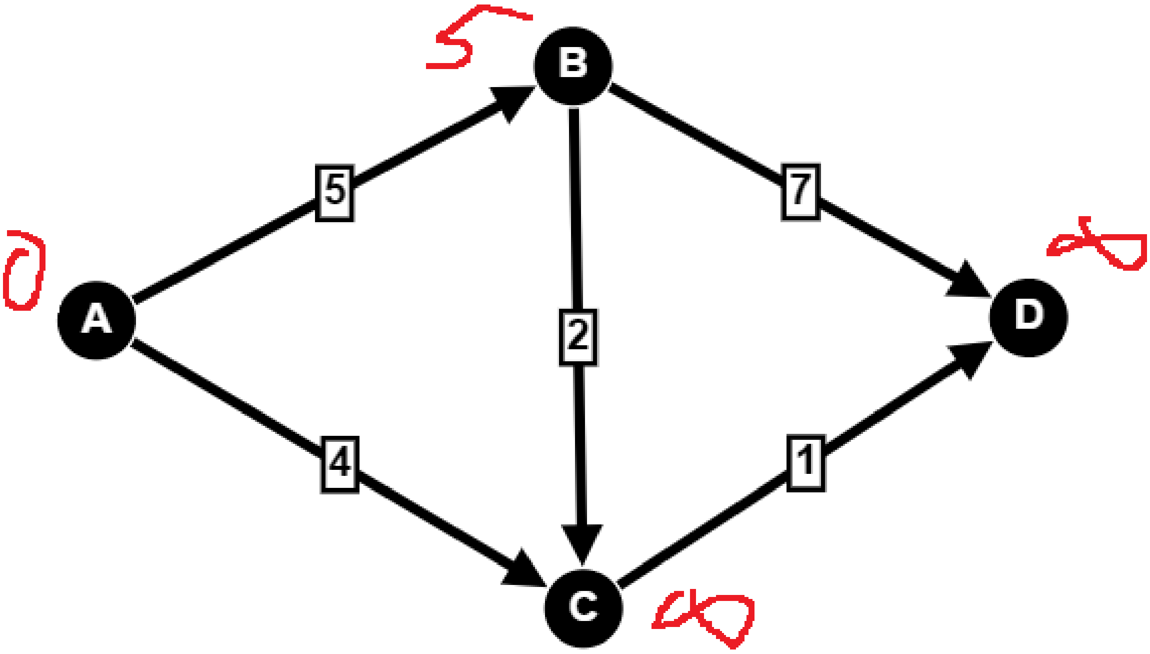
Итерация	Текущая вершина	Кратчайшее расстояние до вершины из **A**			
		A	**B**	**C**	**D**
0	**A**	0	∞	∞	∞
1	**A**	0	5	∞	∞
2	**A**	0	5	4	∞
3	**B**	0	5	4	10
4	**C**	0	5	4	5

Подробная реализация алгоритма по шагам

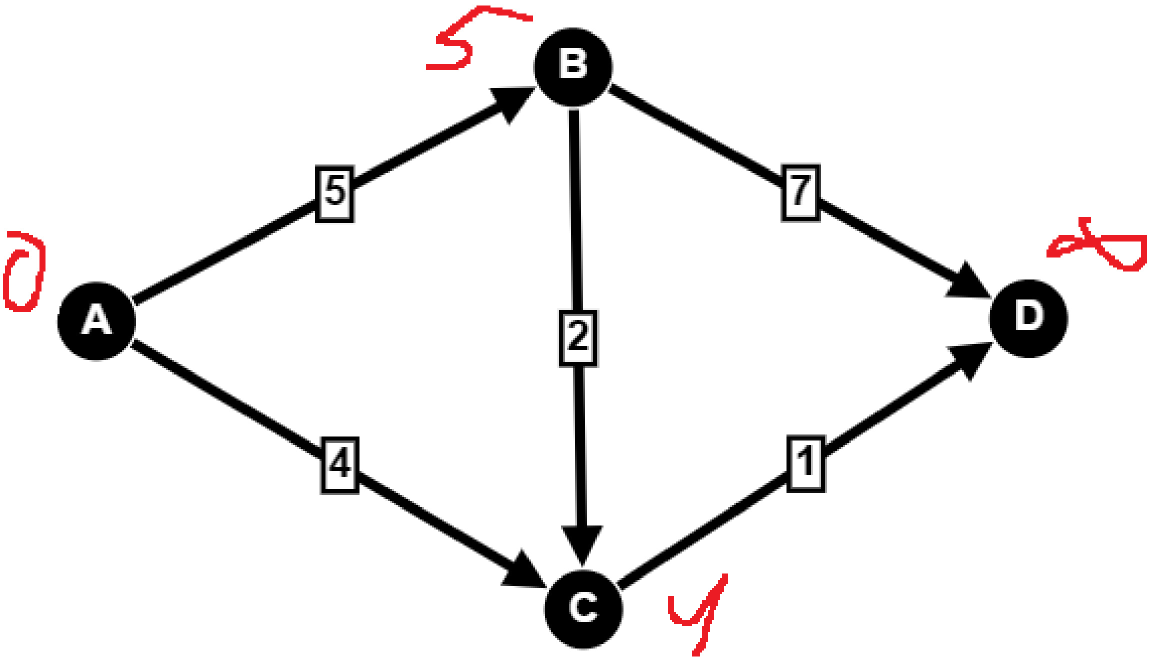
Инициализация
Разметим вершины графа так, что выбранная нам и вершина будет равна 0, а остальные бесконечности. Это отражает то, что расстояния от вершины **A** до других вершин нам пока неизвестны.



Первый шаг
Минимальную метку имеет вершина **A**. Её соседями являются вершины **B** и **C**. Обходим соседей вершины по очереди.
Первым соседом для вершины **A** является вершина **B**. До нее есть только один путь, который и будет являться минимальным (кратчайшим), поэтому сразу можем поменять метку вершины **B** на значение длины ребра **A-B**.

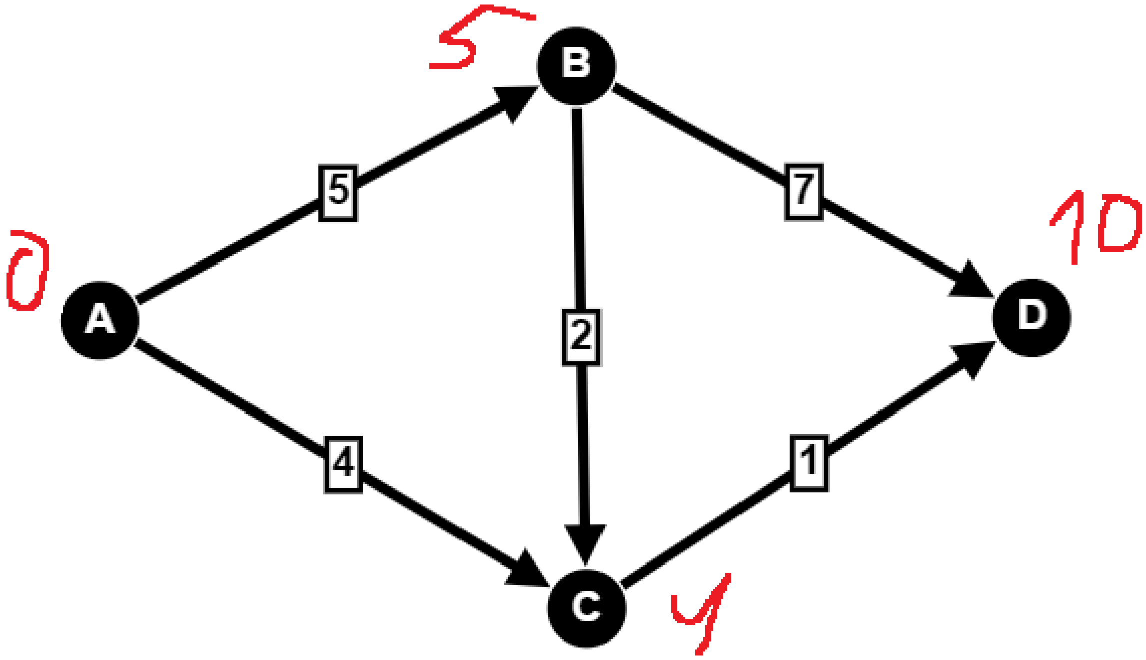


Второй шаг
По аналогии с первым шагом рассмотрим вершину **C**. До нее можно дойти уже двумя способами, поэтому рассмотрим каждый из них и сравним. Длина пути **A-B-C** =0 + 5 + 2 = 7, **A-C** =0 + 4 = 4 => **A-B-C** > **A-C** => новой меткой для вершины **C** будет число 4.

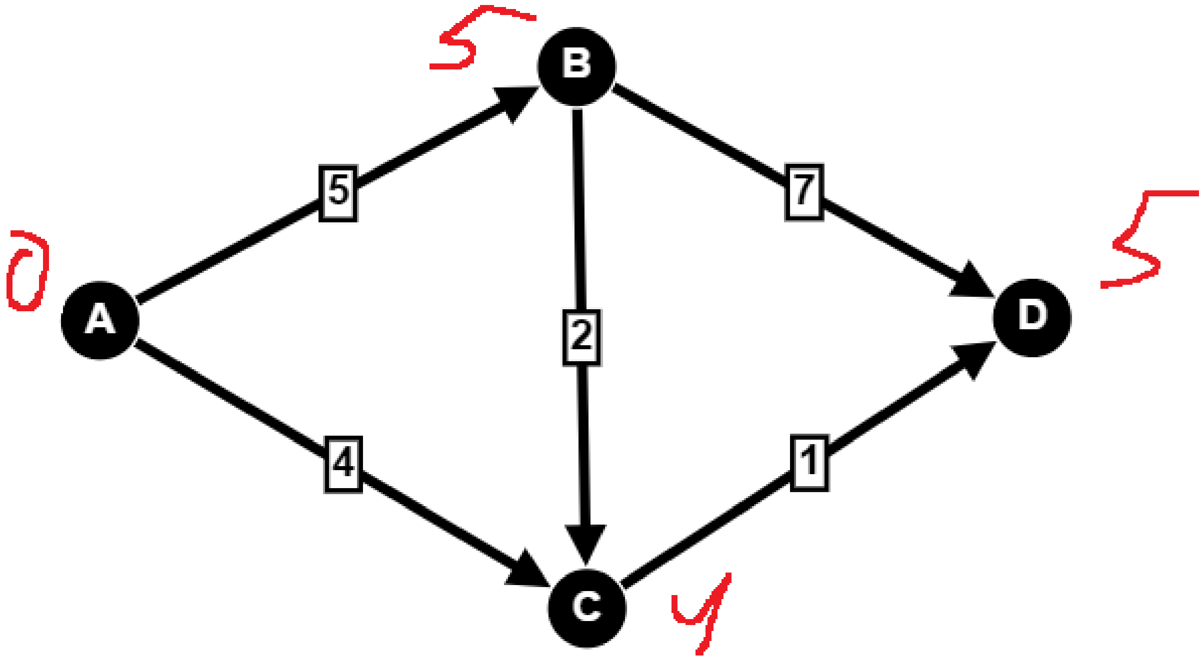


Так как мы проверили всех соседей вершины **A**, то текущее минимальное расстояние до вершины **A** считается окончательным и пересмотру не подлежит

Третий шаг
Повторяем предыдущие шаги алгоритма, выбрав вершину **B**.
Путь **B-D** = 5 + 7 = 12, **B-C-D** = 5 + 4 + 1 = 10 => **B-D** > **B-C-D** => новой меткой для **D** будет число 10



Четвертый шаг
Аналогично и для вершины **C**, которая имеет всего один путь в вершину **D**, который равен 4 + 1 = 5. Т.к. 5 <10=> новой меткой вершины **D** будет число 5

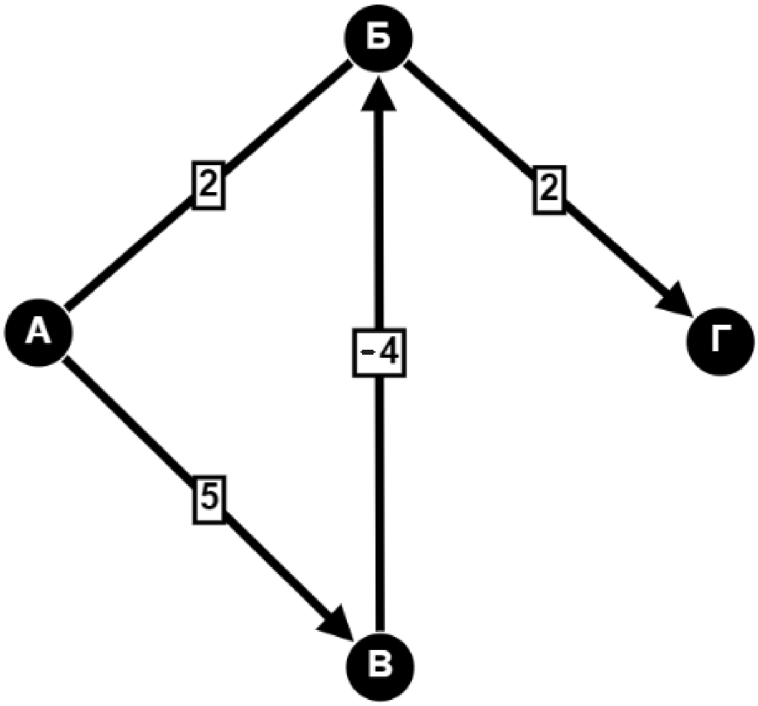


Итог
После выполнения алгоритма мы получили следующий результат для вершины **A**:
Из **A** в **B** = 5
Из **A** в **C** = 4
Из **A** в **D** = 5

Аналогично и для других вершин:

Вершина **B**	Вершина **C**	Вершина **D**
Из **B** в **C** = 2	Из **C** в **D** = 1	Не имеет
Из **B** в **D** = 3		

Задача 2.2. Алгоритм Беллмана-Форда (все пары вершин)
Найти кратчайшие пути для всех пар вершин, но в отличие от алгоритма Дейкстры допускается использование отрицательных весов



Для графа выше рассмотрим алгоритм Беллмана-Форда для всех пар вершин.

В чем заключается алгоритм
Алгоритм Беллмана-Форда проходит итеративный путь через все рёбра. Во время каждой итерации какое-то одно ребро ослабляется. В начале алгоритма расстояния для всех вершин, кроме исходной, инициализируются значением бесконечности.

Составим таблицу весов ребер:

Ребро	**А-Б**	**А-В**	**Б-А**	**Б-Г**	**В-Б**
Вес	**2**	**5**	**2**	**2**	**-4**

Найдём кратчайшие пути из вершины **А** до всех остальных

Инициализация	**А**	**Б**	**В**	**Г**
Вес	0	∞	∞	∞
Пред.	-	-	-	-

На первой итерации алгоритма перебираем все ребра графа и производим ослабления

Итерация 1	**А**	**Б**	**В**	**Г**
Вес	0	2	5	∞
Пред.	-	А	А	-

Итерация 2	**А**	**Б**	**В**	**Г**
Вес	0	1	5	4
Пред.	-	В	А	Б

Итерация 3	**А**	**Б**	**В**	**Г**
Вес	0	1	5	3
Пред.	-	В	А	В

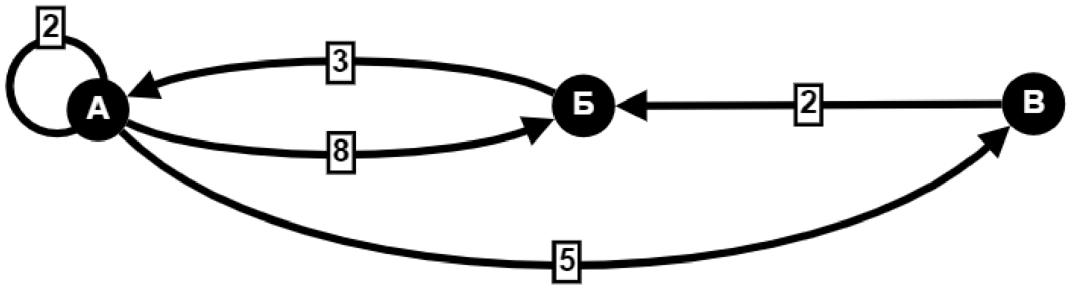
Аналогично найдем кратчайшие пути для всех остальных вершин

Вершина Б	**А**	**Б**	**В**	**Г**
Вес	2	0	7	2
Пред.	Б	-	А	Б

Вершина В	**А**	**Б**	**В**	**Г**
Вес	-2	-4	0	-2
Пред.	Б	В	-	Б

Задача 2.3. Алгоритм Флойда (все пары вершин)

Найти кратчайшие пути для всех пар вершин во взвешенном ориентированном графе. Работает корректно, если в графе нет циклов отрицательной величины, а в случае, когда такой цикл есть, позволяет найти хотя бы один такой цикл.



На примере графа выше рассмотрим работу алгоритма Флойда. В процессе алгоритм будет работать с таблицей, которая содержит информацию о кратчайших путях между парой вершин. Размер таблицы зависит от количества вершин графа. В нашем случае у нас 3 вершины, поэтому будем создавать таблицу 3x3. Также требуется игнорировать петли для корректности расчётов (но не забывать о них при выводе результата).

Инициализация

M0	**А**	**Б**	**В**
А	0	8	5
Б	3	0	∞
В	∞	2	0

Заполняем таблицу M0 таким образом, что пересечение строки и столбца есть вес дуги, важно сначала смотреть на строку (подразумевается что это начальная точка (вершина) дуги), а уже потом на столбец (это будет конечной точкой(вершиной) дуги). Если дуга не имеет прямой дуги из начальной вершины в конечную, то расстояние между ними мы принимаем за бесконечность. Также требуется опустить значения для

Шаг первый (первая итерация)

M1	**А**	**Б**	**В**
А	0	8	5
Б	3	0	8
В	∞	2	0

В первом шаге мы будем рассматривать заполнение новой таблицы M1 по формуле: M1[i, j] = min(M0[i, j], M0[i, **A**] + M0[**A**, j]), где **A** – вершина и на каждом шаге мы будем рассматривать последующие вершины пока не проверим их все

Рассмотрим каждый шаг для i, j – ой ячейки таблицы:

i j
А,А: min(0, 0 + 0) = 0
А,Б: min(8, 0 + 8) = 8
А,В: min(5, 0 + 5) = 5
Б,А: min(3, 3 + 0) = 3
Б,Б: min(0, 3 + 8) = 0
Б,В: min(∞, 3 + 5) = 8
В,А: min(∞, ∞ + 0) = ∞
В,Б: min(2, ∞ + 8) = 2
В,В: min(0, 0 + 5) = 0

Шаг второй
По аналогии с первым шагом выполняем все те же действия и получаем следующую матрицу

M2	**А**	**Б**	**В**
А	0	8	5
Б	3	0	8
В	5	2	0

Шаг третий

M3	**А**	**Б**	**В**
А	0	7	5
Б	3	0	8
В	5	2	0

Пройдя все шаги получили минимальные веса дуг для всех пар вершин:

А-А = 2	**Б-А** = 3	**В-А** = 5
А-Б = 7	**Б-Б** = 0	**В-Б** = 2
А-В = 5	**Б-В** = 8	**В-В** = 0