Поиск кратчайшего пути в графе с помощью алгоритма Дейкстры

Найти длины кратчайших путей из вершины A до всех остальных вершин.

	A	B	C	D
A	0	6	2	10
B	6	0	3	3
C	2	3	0	9
D	10	3	9	0

1. Составим информационную таблицу:

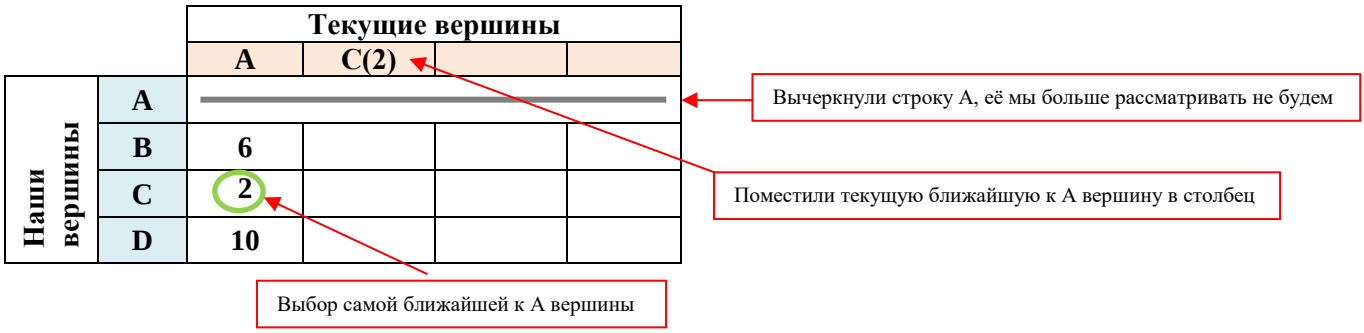
		Здесь будут текущие вершины			
Наши вершины	**A**				
	B				
	C				
	D				

Каждый столбец отвечает за шаг алгоритма. В ячейках мы будем обновлять расстояние **из вершины A до остальных вершин**. Т.е. каждый новый шаг (новый столбец) предоставляет актуальную информацию, доступную на текущий момент, о кратчайшем расстоянии из вершины A до соответствующих вершин.

Шаг 1. Выбираем стартовую вершину A и помещаем её в первый столбец. Далее смотрим, до каких вершин мы из вершины A можем дойти, и напротив каждой вершины в строке записываем длину пути из вершины A (эта будет актуальная информация на шаге 1).

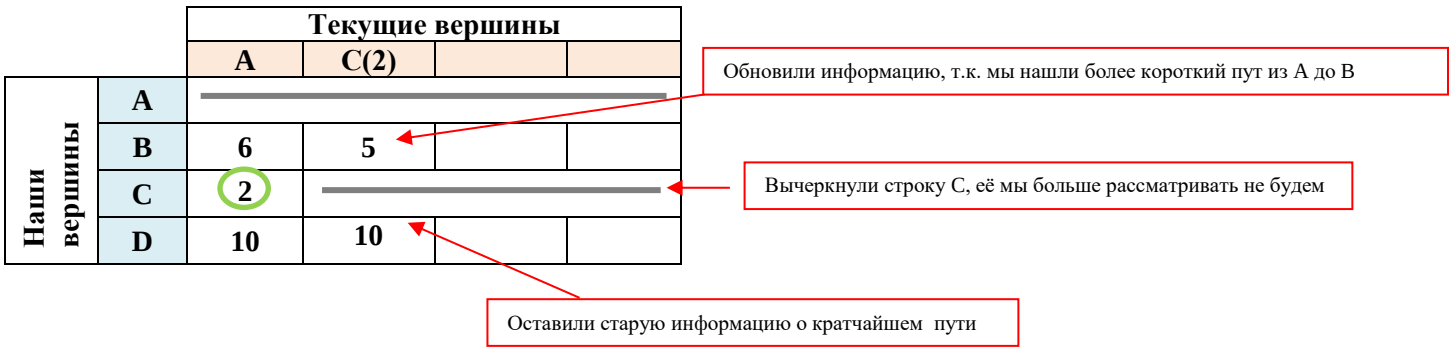
Вершина A считается рассмотренной, и в неё мы возвращаться больше не будем. Можно вычеркнуть соответствующую строку.

Шаг 2. Выбираем следующую вершину, которая ближе всего расположена к вершине A. В нашем случае, это вершина C. Помещаем её в следующий столбец, указывая в скобках расстояние из A до C.

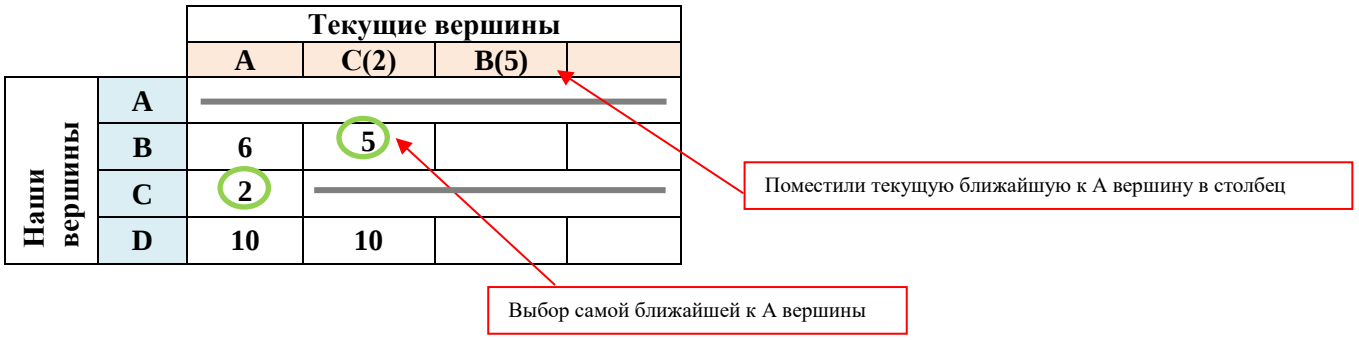


Далее нам необходимо перепроверить информацию о кратчайших путях из вершины A. Т.е. надо понять, нашли ли мы путь короче, чем то, что мы уже знаем, либо информация с предыдущих шагов остаётся актуальной. Для этого смотрим, до каких нерассмотренных ранее вершин мы из вершины C можем дойти. Мы

можем дойти из C до B за 3, тогда получается, что из A до B (через C) можно дойти за 5 (складываем значение метки C и значение на ребре между C и B (2+3=5)). И это оказывается лучше, чем то, что мы знали до этого (5<6). Тогда обновим эту информацию в строке!
Ещё мы из C можем пойти в D за 9, тогда получается, что из A до D (через C) можно дойти за 11 (складываем значение метки C и значение на ребре между C и D (2+9=11)). И это оказывается хуже, чем то, что мы уже нашли до этого (11>10). Поэтому новая информация нам не принесла никакой пользы, и мы оставляем старую информацию о кратчайшем пути между A и D.
Вершина C считается рассмотренной, и в неё мы возвращаться больше не будем. Можно вычеркнуть соответствующую строку.
Таблица выглядит так:



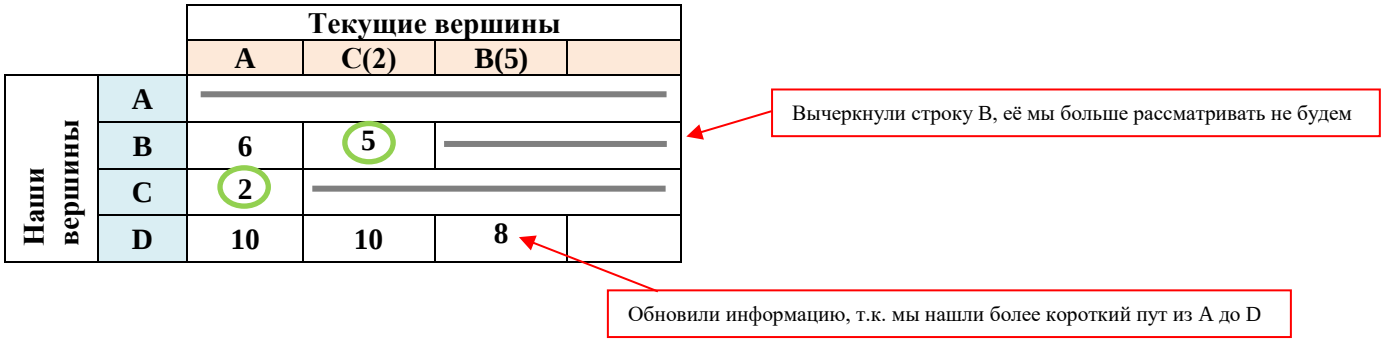
Шаг 3. Выбираем следующую вершину, которая ближе всего расположена к вершине A. В нашем случае, это вершина B. Помещаем её в следующий столбец, указывая в скобках расстояние из A до B.



Далее опять перепроверяем информацию о текущих кратчайших путях из вершины A. Из B можно попасть только в D (из нерассмотренных вершин), поэтому сравниваем нашу прошлую информацию о пути из A в D и новую информацию (о пути из A в D через B). Раньше мы добирались за 10. Теперь же из A до B мы можем добраться за 5 (указано в скобках рядом с B), а из B до D можем добраться за 3 (это указано в исходном графе). Т.е. новый путь из A до D через B будет равен 5+3=8, и это короче, чем то, что мы знали до этого. Обновляем информацию!

Вершина B считается рассмотренной, и в неё мы возвращаться больше не будем. Можно вычеркнуть соответствующую строку.

Таблица выглядит так:



Шаг 4. Выбираем следующую вершину, которая ближе всего расположена к вершине A. В нашем случае, это вершина D. Помещаем её в следующий столбец, указывая в скобках расстояние из A до D.

Наши вершины	Текущие вершины: A	C(2)	B(5)	D(8)
A				
B	6	5		
C	2			
D	10	10	8	

Поместили текущую ближайшую к A вершину в столбец

Выбор самой ближайшей к A вершины

Из вершины D мы никуда пойти не можем (все вершины уже рассмотрены). Поэтому алгоритм заканчивается. Финальная таблица выглядит так:

Наши вершины	Текущие вершины: A	C(2)	B(5)	D(8)
A				
B	6	5		
C	2			
D	10	10	8	

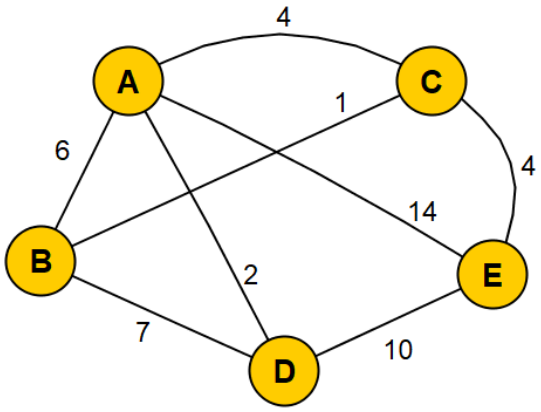
В скобках указаны длины самых коротких путей из вершины A
Это и есть наш ответ!

Внимание ещё раз: алгоритм заканчивается, когда будут рассмотрены все вершины!

Главное понять, что в ячейках мы записываем расстояние из вершины A. На каждом шаге мы либо обновляем информацию, если она лучше, либо переносим старую информацию, если новая информация хуже. Выбор вершины на каждом шаге происходит по принципу «самая близкая вершина к вершине A». Если самых ближайших вершин несколько, то выбираем любую.

Попробуем ещё раз на другом примере (теперь только таблицы).

Найти кратчайший путь из A до всех остальных вершин.



Шаг 1.

Наши вершины	Текущие вершины: A				
A					
B	6				
C	4				
D	2				
E	14				

Шаг 2.

Наши вершины	Текущие вершины: A	D(2)			
A					
B	6	6			
C	4	4			
D	2				
E	14	12			

Информация не обновилась, т.к. старый путь лучше нового (6 < 2 + 7)

Информация не обновилась, т.к. нового пути не обнаружено

Информация обновилась, т.к. новый путь лучше старого (2 + 10 < 14)

Шаг 3.

Наши вершины	Текущие вершины: A	D(2)	C(4)		
A					
B	6	6	5		
C	4	4			
D	2				
E	14	12	8		

Информация обновилась, т.к. новый путь лучше старого (4 + 1 < 6)

Информация обновилась, т.к. новый путь лучше старого (4 + 4 < 12)

Шаг 4.

Наши вершины	Текущие вершины: A	D(2)	C(4)	B(5)	
A					
B	6	6	5		
C	4	4			
D	2				
E	14	12	8	8	

Информация не обновилась, т.к. нового пути не обнаружено

Шаг 5.

Наши вершины	Текущие вершины: A	D(2)	C(4)	B(5)	D(8)
A					
B	6	6	5		
C	4	4			
D	2				
E	14	12	8	8	

Ответ

Все вершины рассмотрены. Конец алгоритма

Задача о соединении городов (алгоритмы Прима и Краскала) (иначе, поиск минимального остовного дерева)

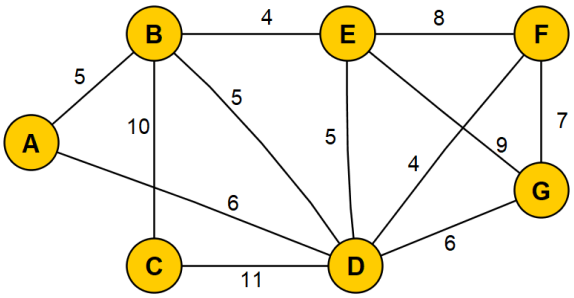
Постановка задачи:

Имеется информация о стоимости постройки дорог между различными городами. Необходимо построить сеть дорог, связывающей все города, минимальной стоимости (разумеется, города могут быть связаны не напрямую).

Рассмотрим два алгоритма решения данной задачи.

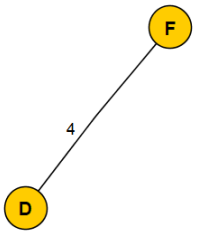
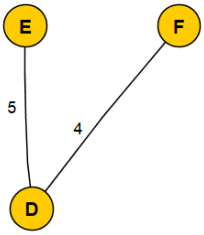
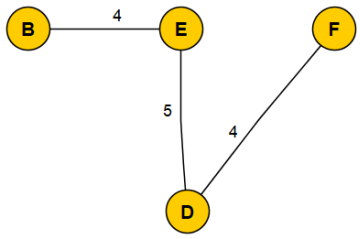
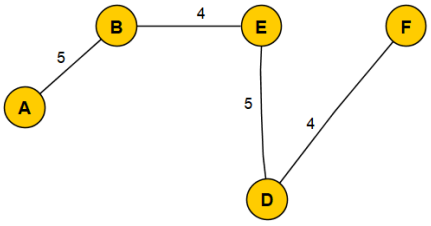
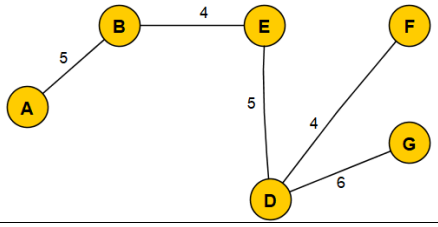
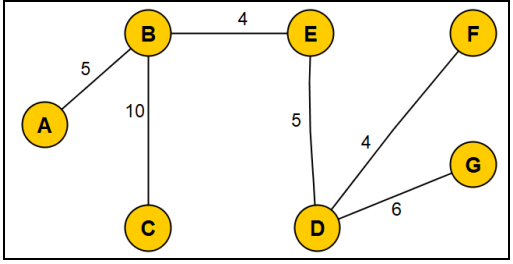
1. **Алгоритм Прима**: на каждом шаге алгоритма присоединяем самую дешёвую нерассмотренную вершину к текущей «стройке». Заканчиваем алгоритм, когда присоединим все вершины.
2. **Алгоритм Краскала**: на каждом шаге алгоритма строим дорогу минимальной стоимости (следим, чтобы не образовывались циклы). Заканчиваем алгоритм, когда из каждой вершины можно попасть в любую другую.

Пример:



Дан граф, в котором указана стоимость дорог между некоторыми городами. Необходимо связать все города дорогой минимальной стоимости.

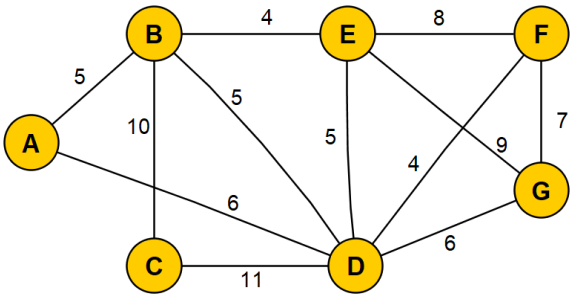
Алгоритм Прима:

Шаг 1. Выбираем любую вершину (например, D)	**Шаг 2.** Присоединяем к текущим вершинам (к D) самую дешёвую из нерассмотренных вершин (это будет F).	**Шаг 3.** Присоединяем к текущим вершинам (к D или к F) самую дешёвую из нерассмотренных вершин (это будет E).
		
Шаг 4. Присоединяем к текущим вершинам (к D, к F или к E) самую дешёвую из нерассмотренных вершин (это будет B).	**Шаг 5.** Присоединяем к текущим вершинам (к D, к F , к E или к B) самую дешёвую из нерассмотренных вершин (это будет A).	**Шаг 6.** Присоединяем к текущим вершинам (к D, к F , к E, к B или A) самую дешёвую из нерассмотренных вершин (это будет G).
		
Шаг 6. Присоединяем к текущим вершинам (к D, к F , к E, к B, к A или к G) самую дешёвую из нерассмотренных вершин (это будет C).		
	**Суммарная стоимость дороги:** **5+10+4+5+4+6 = 34**	

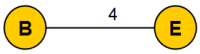
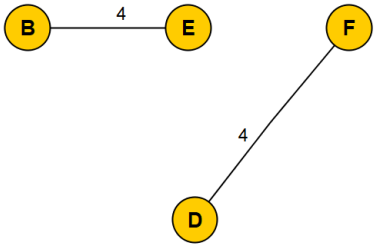
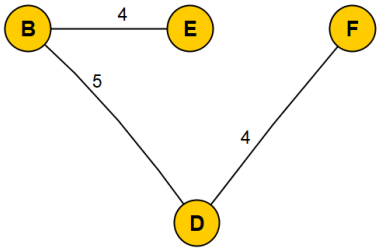
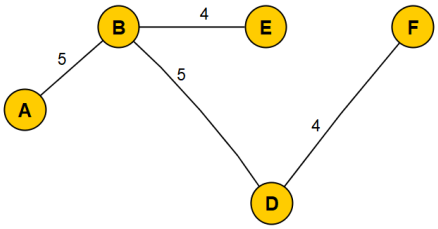
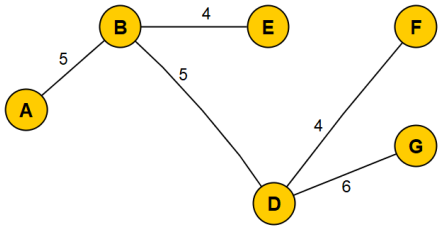
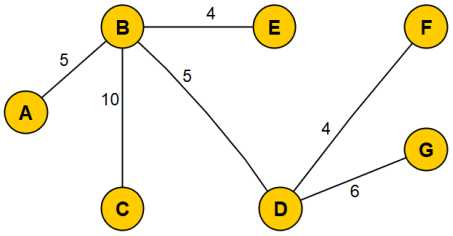
ВНИМАНИЕ: не надо пытаться строить цепочки!!! Из одной вершины может быть несколько ответвлений (как тут из вершины D)! Необходимо, чтобы из каждой вершины можно было добраться в любую другую (можно через посредников).

ВНИМНИЕ 2: вам необходимо показывать шаги алгоритма! Либо выписывайте, какие вершины были присоединены (и к каким) на каждом шаге, либо помечайте вершины в порядке их добавления (например, в скобках указывать, на каком шаге они были добавлены).

Сделаем тот же пример по алгоритму Краскала



Алгоритм Краскала:

Шаг 1. Выбираем самую дешёвую дорогу и строим её. Если таких дорог несколько, выбираем любую. Например, B – E.	**Шаг 2.** Выбираем самую дешёвую дорогу и строим её. Это будет D – F, добавляем к стройке.	**Шаг 3.** Выбираем самую дешёвую дорогу и строим её. Если таких дорог несколько, выбираем любую. Например, B – D.
		
Шаг 4. Выбираем самую дешёвую дорогу и строим её. Внимание, мы не можем выбрать E – D, т.к. будет цикл! А это плохо. Тогда выбираем A – B.	**Шаг 5.** Выбираем самую дешёвую дорогу и строим её. Внимание, мы не можем выбрать A – D, т.к. будет цикл! А это плохо. Тогда выбираем D - G.	**Шаг 6.** Выбираем самую дешёвую дорогу и строим её. Это будет B – C (C – последняя оставшаяся вершина).
		

Суммарная стоимость дороги:
5+10+4+5+4+6 = 34

Обратите внимание, что графы в алгоритме Прима и Краскала у нас не совпали (из-за разного порядка выбора элементов с одним и тем же весом), но суммарная стоимость совпала.

Аналогично, вам необходимо показывать шаги алгоритма! Либо выписывайте, какие рёбра были добавлены на каждом шаге, либо помечайте рёбра в порядке их добавления (например, в скобках указывать, на каком шаге они были добавлены).

Enjoy!