ДЗ по алгоритмам для ИС. Графы.

Минимальное остовное дерево и кратчайшие пути.

X = 22

Y = 3

Z = 4

K = 1

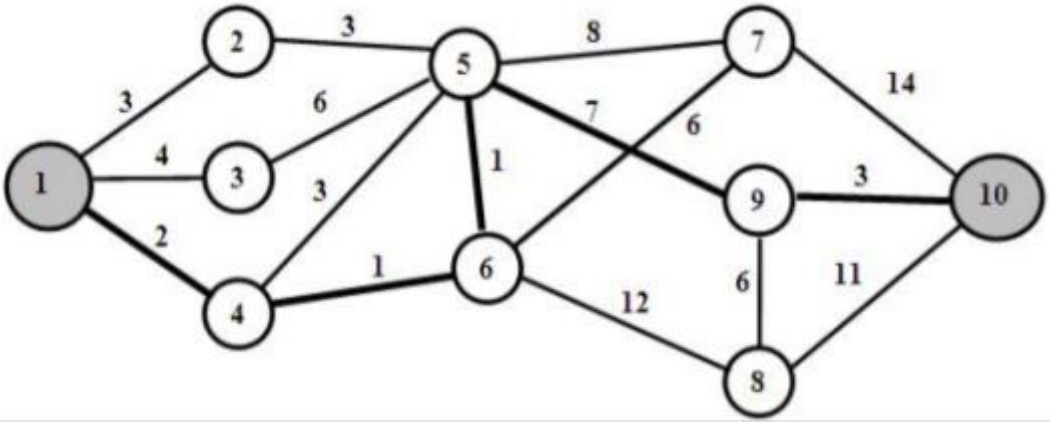
Задание №1.

Условие:

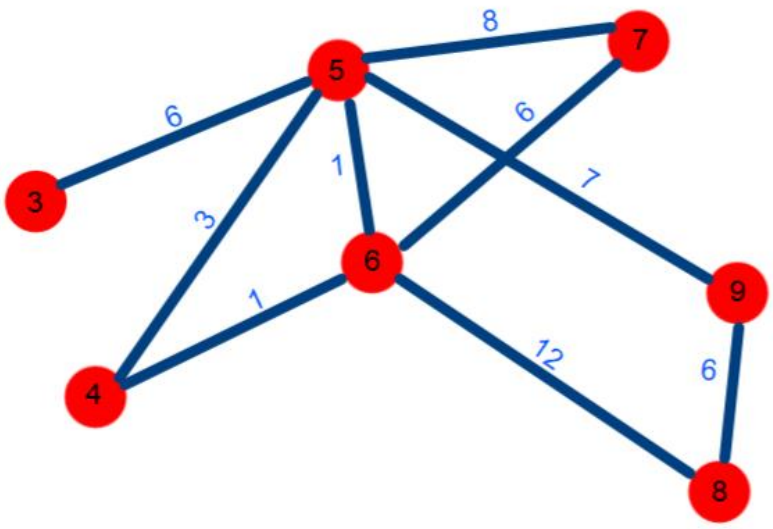
1. Покажите, как будет работать алгоритм Краскала на подграфе с вершины Y по вершину Y + 6 включительно

Решение:

Исходный граф:



Подграф:



Идея алгоритма:

Идея алгоритма заключается в том, чтобы отсортировать все ребра по весу, а потом добавлять их последовательно в граф пока количество ребер не будет равно количеству вершин минус один.

Алгоритм:

1. Отсортируем все ребра.

Последовательность ребер:

(4;6) вес = 1 ✓

(5;6) вес = 1 ✓

(4;5) вес = 3 ✗

(3;5) вес = 6 ✓

(6;7) вес = 6 ✓

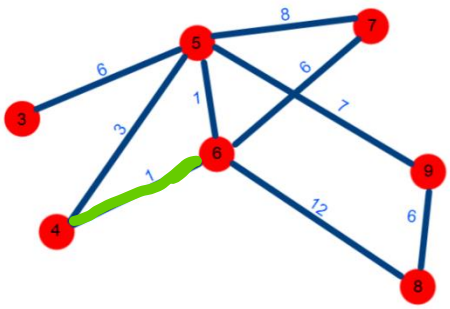
(8;9) вес = 6 ✓

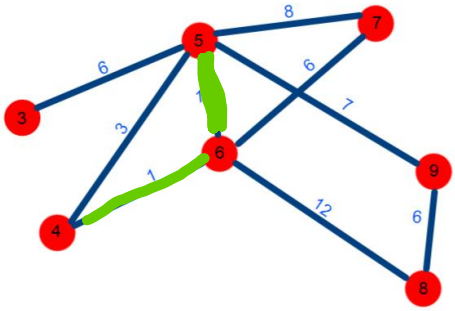
(5;9) вес = 7 ✓

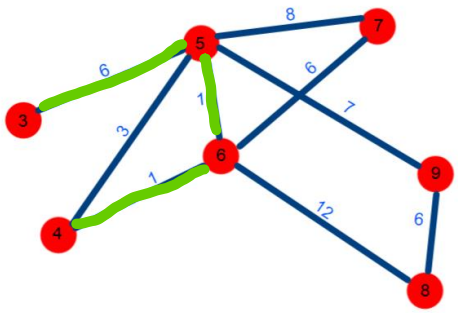
(5;7) вес = 8

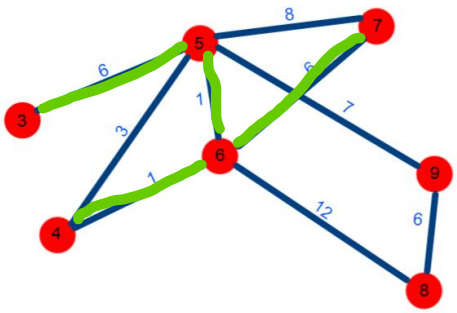
(6;8) вес = 12

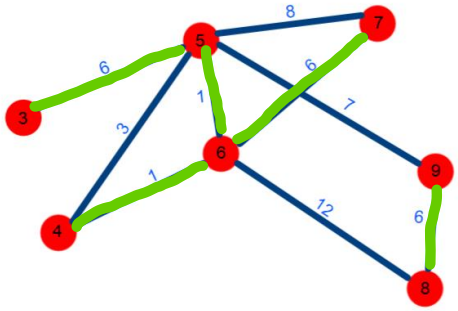
2.добавляем ребра в граф (показано на подграфе исходного графа)

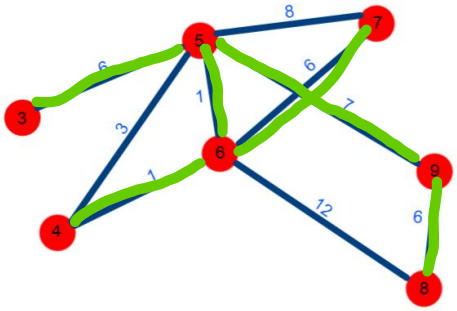












мст найден!

Оценка времени: ElogE

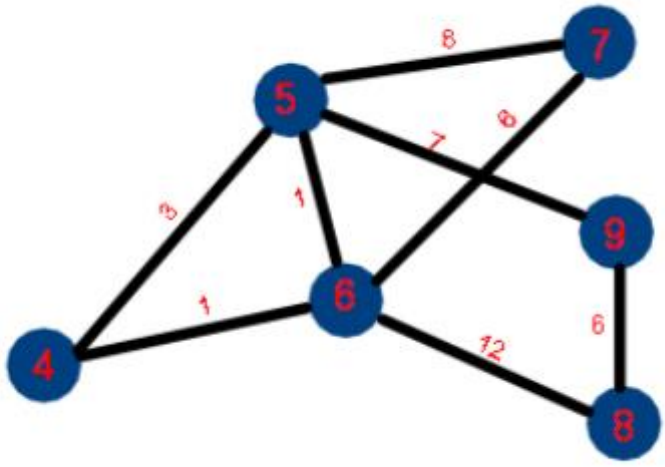
Задание №2.

Условие:

2. Покажите, как будет работать алгоритм Прима на подграфе с вершины Z по вершину Z + 5 включительно, если стартовать из вершины Z+K.

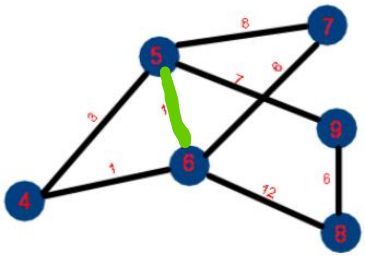
Решение:

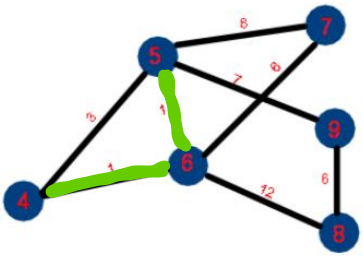
Подграф:

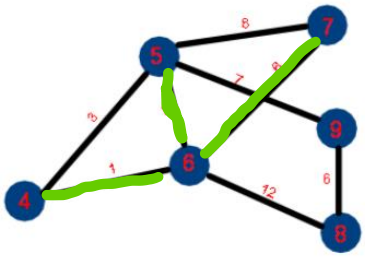


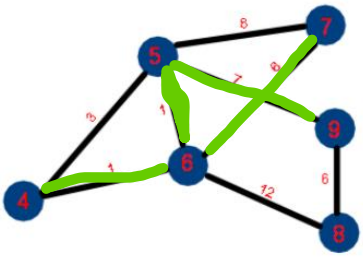
Решение:

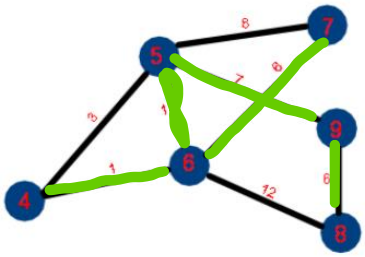
Берем и запихиваем начальную вершину в приоритетную очередь, после чего создаем цикл, по которому будем ходить пока очередь не пуста. Действия, выполняемые в цикле: берем достаем вершину из очереди с максимальным приоритетом, то есть расстояние до которой минимальное из всех остальных расстояний. После обновляем приоритеты у всех смежных вершин, и идем дальше по циклу.











мст найден!

Оценка времени: ElogV(приоритетная очередь); V^2 + E (массив); VlogV + E (фибоначиева куча)

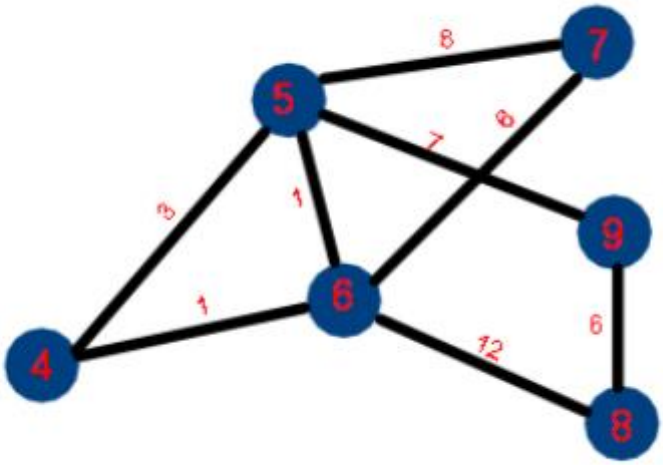
Задание №3.

Условие:

3. Покажите, как будет работать алгоритм Беллмана-Форда на подграфе с вершины Z по вершину Z + 5 включительно, если стартовать из вершины Z+K+1.

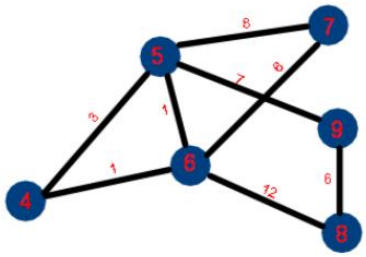
Решение:

Подграф:



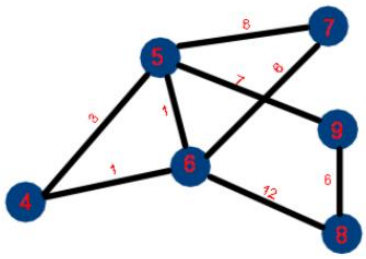
Решение:

На каждой N итерации алгоритма мы релаксируем расстояние до вершины по ребрам, количество которых N.



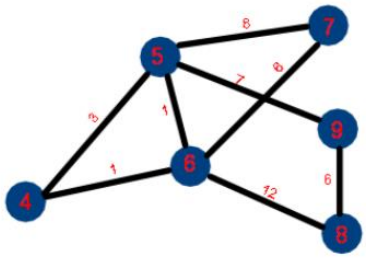
1. Рассматриваем вершины на расстоянии 0.

То есть 6. Из 6 до 6 расстояние 0. До всех остальных бесконечность.



2. Рассматриваем вершины на расстоянии 1.

Обновляем расстояния. Из 6 можно добраться в 4 за 1, в 5 за 1, в 7 за 6, в 8 за 12.



3. Рассматриваем вершины на расстоянии 2.

Учитывая рассматриваемую реализацию, можно будет обновить расстояние до 9, оно будет равно 8. Дальше можно алгоритм не рассматривать (кратчайшие расстояния меняться не будут).

Оценка времени: VE

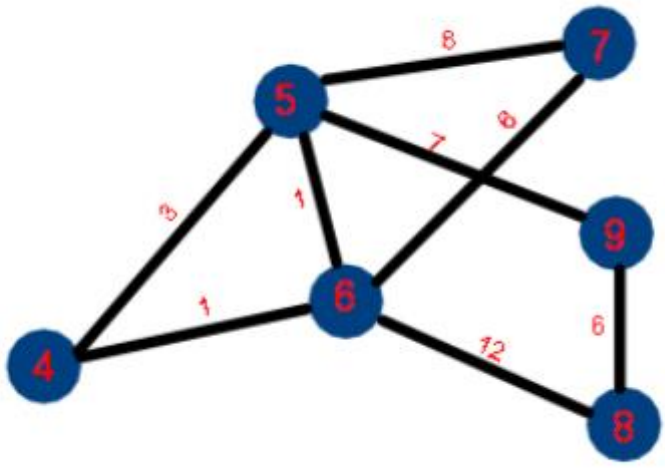
Задание №4.

Условие:

4. Покажите, как будет работать алгоритм Дейкстры на подграфе с вершины Z по вершину Z + 5 включительно, если стартовать из вершины Z+K+2.

Решение:

Подграф:



Решение:

Мы берем вершину 7, кладем в приоритетную очередь, после чего запускаем цикл в котором, достаем вершину 7 из очереди, обновляем расстояния до смежных вершин (у вершины 5 расстояние 8, у вершины 6 расстояние 6), кладем их в очередь. Достаем вершину с максимальным приоритетом (это 6) , обновляем расстояния (если вершина была в очереди, то выполняем операцию изменения приоритета), теперь в очереди лежат следующие вершины (5 с расстоянием 7, 4 с расстоянием 7, 8 с расстоянием 18), теперь достаем 5, обновив расстояния получим следующую очередь (4 с расстоянием 7, 9 с расстоянием 14, 8 с расстоянием 18), дальше достанем 4 из очереди, потом достанем 9 и достанем 8) Все расстояния найдены!

Оценка времени: ElogV(приоритетная очередь); V^2 + E (массив); VlogV + E (фибоначиева куча)

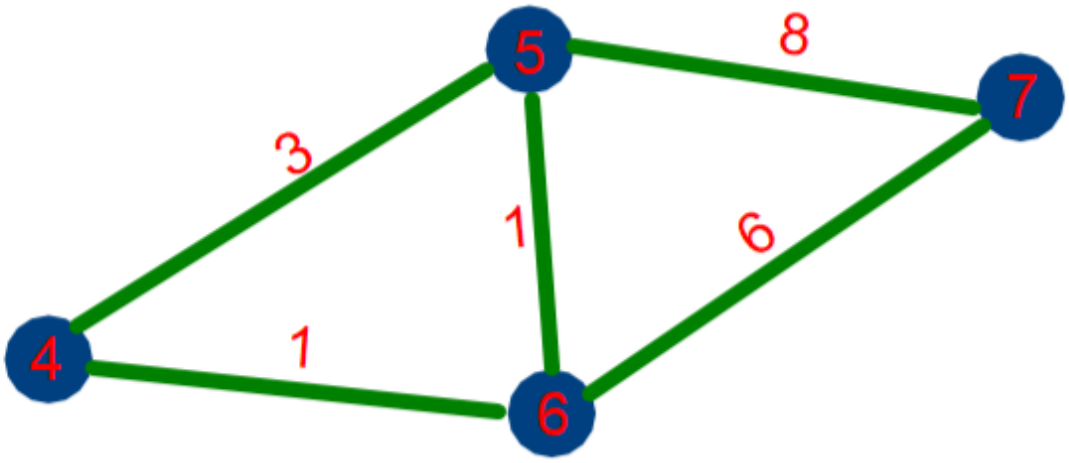
Задание №5.

Условие:

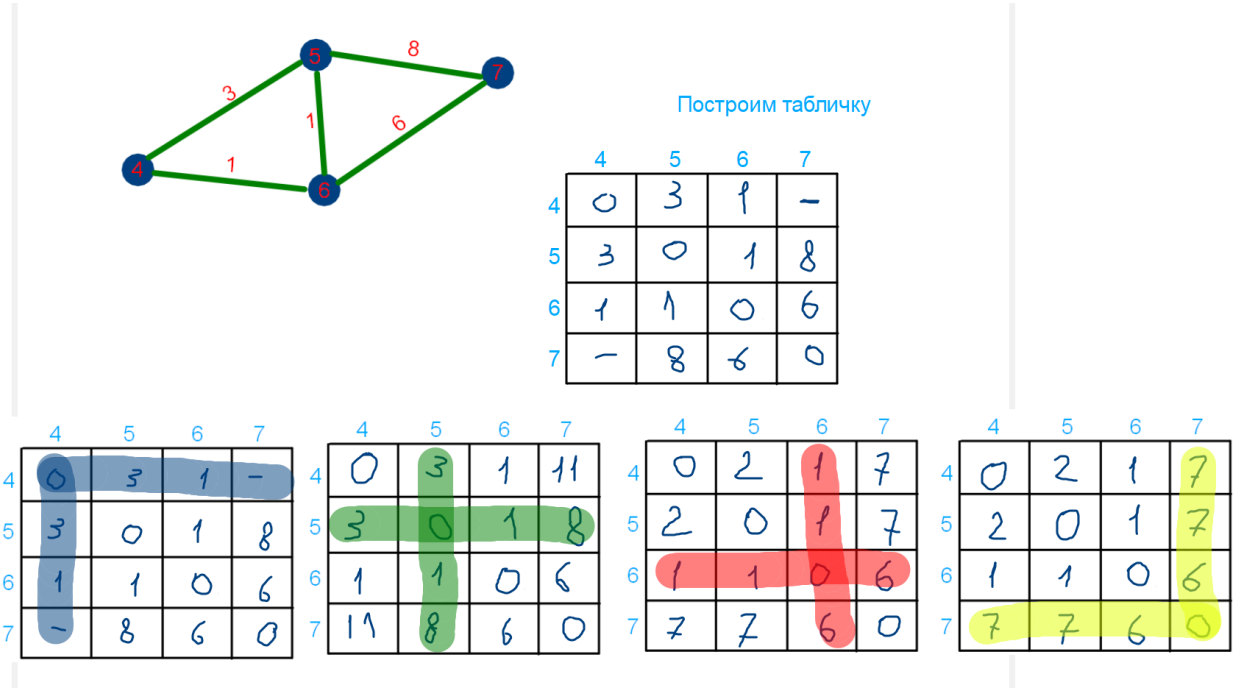
5. Покажите, как будет работать алгоритм Флойда на подграфе с вершины Z по вершину Z + 3 включительно.

Решение:

Подграф:



Решение:



Оценка времени:

V^3

Задание №6.

Условие:

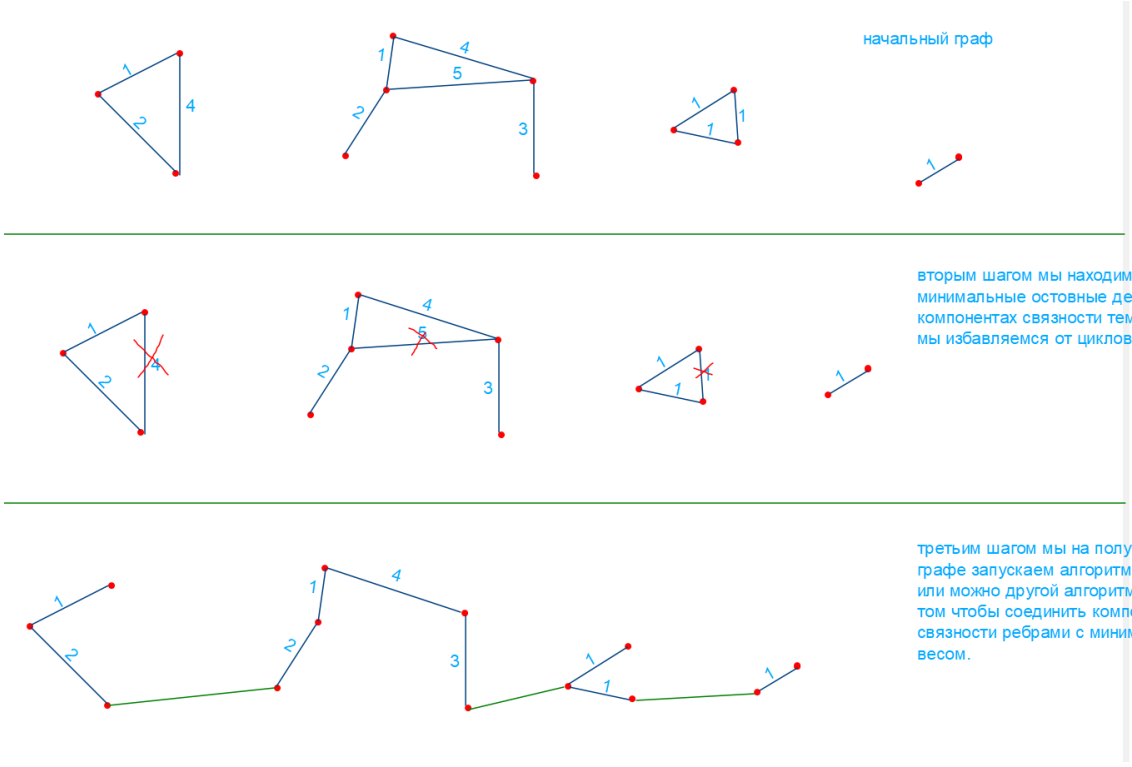
6. В стране n городов. Однажды в ней решили построить n-1 дорогу, чтобы связать все города. Однако что-то пошло не так, и получилось, что эти n - 1 дорог образуют не дерево, а k компонент связности. Нужно перестроить k - 1 дорогу (то есть убрать k - 1 и построить другие k - 1), так, чтобы получилось дерево. Из всех таких способов нужно выбрать такой, чтобы суммарная длина дорог была минимальна.

Решение:

Идея:

В каждой компоненте связности можно найти минимальное остовное дерево. После можно запустить алгоритм Борувки, но не с начальной стадии, мы должны написать алгоритм так, чтобы соединить именно имеющиеся компоненты связности минимальными ребрами.

Пример:



Оценка времени:

Минимальное остовное дерево можно найти через триангуляцию Делона, на которой можно запустить алгоритм Борувки за VloglogV. Это надо сделать для каждой компоненты связности, то есть k раз. По факту мы можем запустить здесь алгоритм Борувки который отработает за ElogV. То есть примерная оценка по времени равна kVloglogV + ElogV.

Задание №7.

Условие:

7. Есть n городов. Можно соединить два города дорогой, потратив A×len денег, где len — длина дороги, а можно построить в городе аэропорт, потратив B денег. Нужно за минимальное число денег соединить все города (чтобы от каждого до каждого можно было добраться с помощью дорог и самолетов).

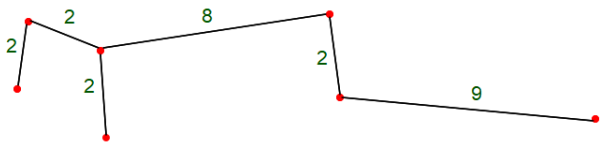
Решение:

Идея:

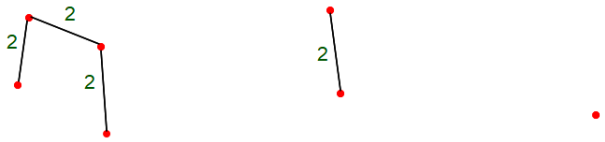
Сначала найдем минимальное остовное дерево по дорогам. Затем если построить в двух городах аэропорты дешевле чем построить дорогу, то заменим дорогу, связывающую два города, на аэропорты.

Пример.

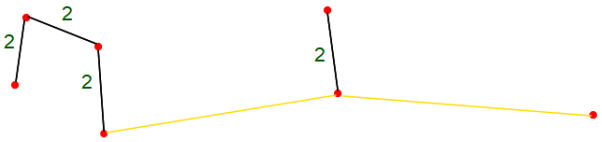
допустим A == 4; B == 10



первым шагом построим минимальное остовное дерево по дорогам.



Вторым шагом удалим из графа ребра вес которых соотвествует A*len > 2B.



Третьим шагом в каждой из компонент связности построим по одному аэропорту (не более)

+во втором шаге: если в компоненте связности содержится аэропорт, то мы делаем другую проверку A*len > B.

Оценка по времени:

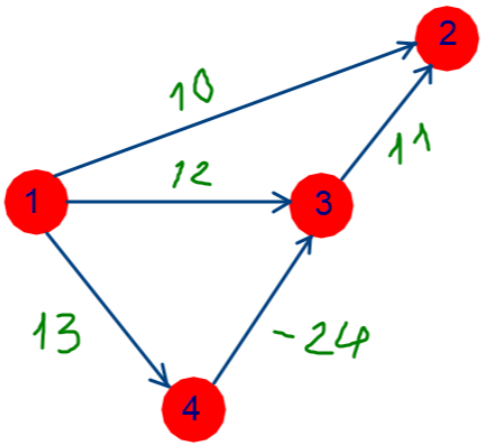
Также как и в предыдущей задаче построим мст за VloglogV, после чего за E пройдемся по всем ребрам, удалив ненужные и за k поставим аэропорты. То есть примерная оценка по времени равна VloglogV + E + k.

Задание №8.

Условие:

8. Почемму алгоритм Дейкстры не работает на отрицтельных весах? Приведите пример, когда такое получается

Решение:



Если стартовать из вершины 1 то в кучу сначала положим 1, когда достанем ее из кучи 1, то положим 2, 3, 4; Потом достанем 2 и уже будет ошибка так как туда расстояние будет меньше.

Задание №9.

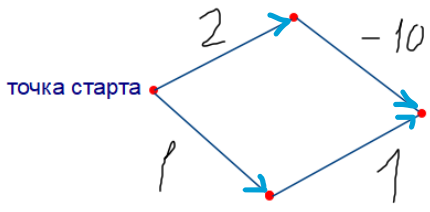
Условие:

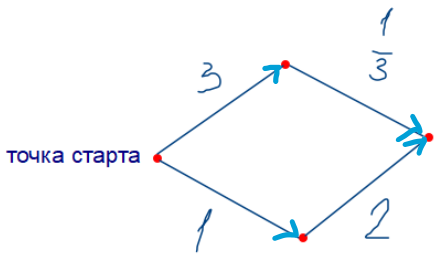
9. Пусть вес пути — это не сумма весов ребер, а произведение. Модифицируйте алгоритм Дейкстры. Для каких весов он будет работать?

Решение:

У начальной вершины будет вес 1, а не 0, так как иначе будет 0 у всех остальных вершин. Алгоритм не будет работать для графов, где есть веса меньше 1. Веса просто перемножаются.

Примеры, когда не работает:





Задание №10.

Условие:

10. Покажите, что алгоритмы Дейкстры и Прима делают по сути одно и то же.

Решение:

Алгоритмы правда очень похожи. Единственное различие данных алгоритмов состоит в том, что в алгоритме Прима мы записываем в вершину вес минимального ведущего в нее ребра, а в алгоритме Дейкстры мы записываем вес минимального пути ведущего в вершину.

Задание №11.

Условие:

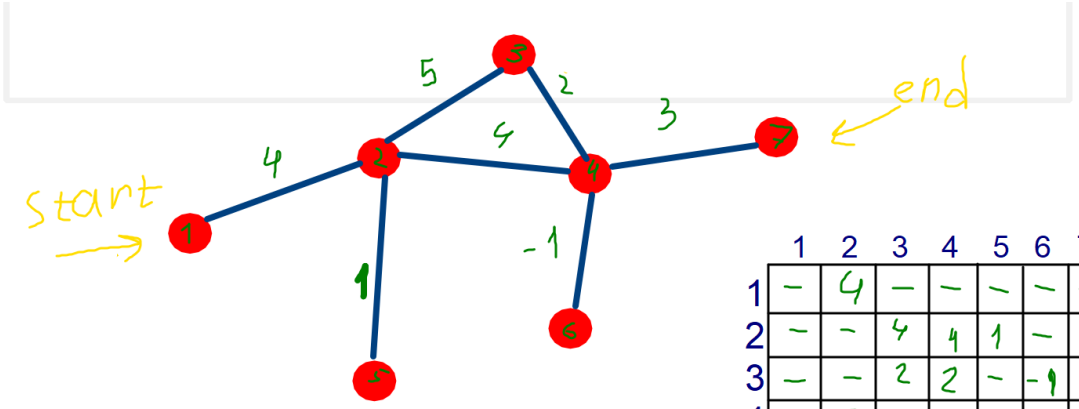
11. Пусть вес пути — это минимум из весов ребер. Как найти кратчайший путь?

Решение:

Из условия делается вывод что граф неориентированный так как написано ребер а не дуг или ориентированных ребер. Так как в условии сказано путь то мы будем искать только один путь из одной вершины в другую, а не во все сразу.

Запустим алгоритм Беллмана Форда, но немного его модифицируем, так чтобы у нас каждый раз в ячейку записывалось значение минимального ребра в пути до вершины. Потом просто в ячейке последнего элемента в пути возьмем минимальное число.

Пример:



	1	2	3	4	5	6	7
1	–	4	–	–	–	–	–
2	–	–	4	4	1	–	–
3	–	–	2	2	–	-1	3
4	–	2	–	–	–	-1	2
5							
6							
7							

ит.д.

Задание №12.

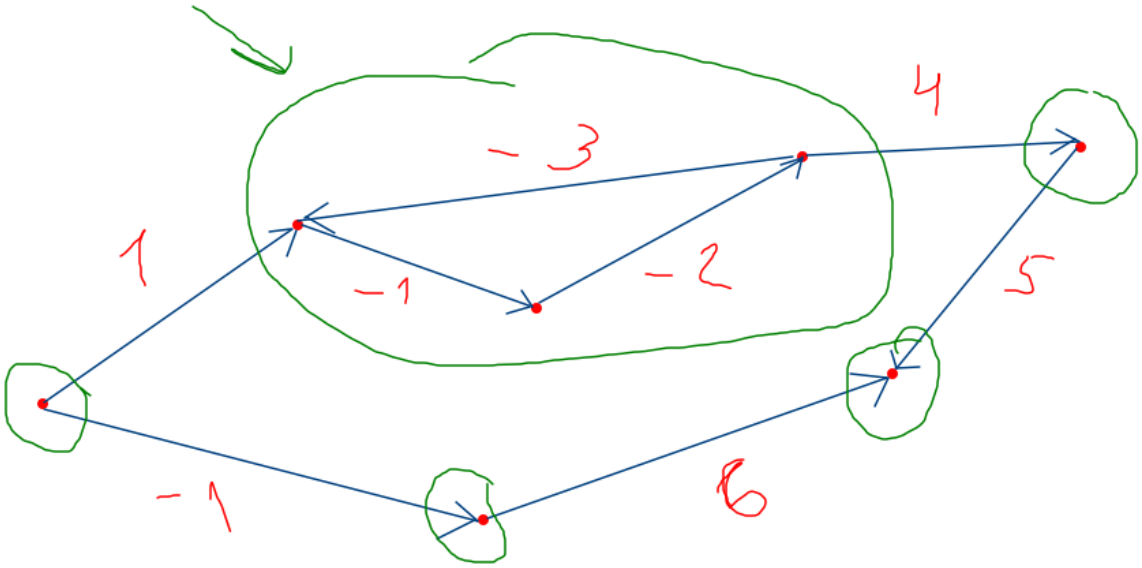
Условие:

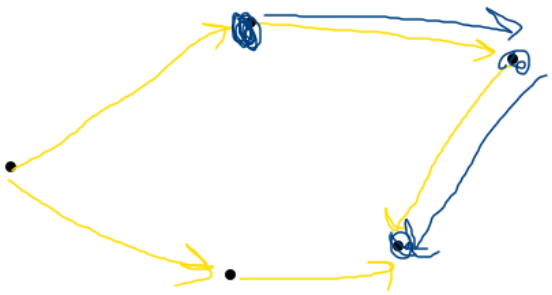
12. Как найти все вершины, вес пути до которых может быть сколь угодно малым?

Решение:

Для начала найдем все компоненты сильной связности, потом запускаем алгоритм Белмана Форда на каждой компоненте сильной связности. Алгоритмом Белмана Форда мы найдем цикл отрицательного веса. После этого составим граф конденсации. Все вершины в графе конденсации, которые были составлены из компонент сильной связности, такие что в компоненте связности был цикл отрицательного веса, мы пометим каким-нибудь цветом. А после от них запустим обход в ширину и пометим все вершины достижимые из них тем же цветом. По окончании алгоритма нам будут известны все вершины, вес пути до которых может быть сколь угодно малым, они будут помечены цветом.

Пример:





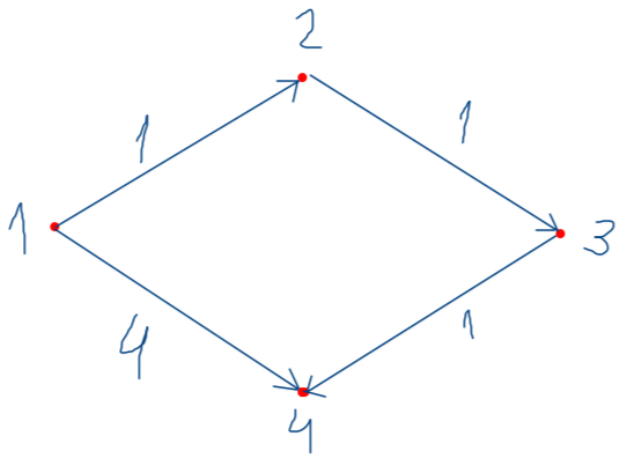
Задание №13.

Условие:

13. Петя перепутал и написал алгоритм Флойда так:
for i:
for j:
for k:
d[i][j] = min(d[i][j], d[i][k] + d[k][j]).
Постройте тест, на котором получившийся алгоритм работает неверно.

Решение:

Пример:



Из-за того что Петя перепутал счетчики, d[i][j] получает не те значения при сравнивании.