Алгоритм Краскала

С применением системы непересекающихся множеств и сжатия пути

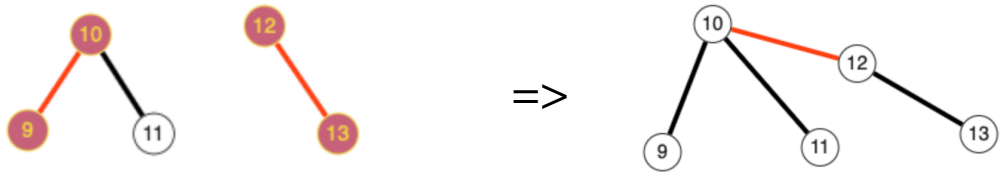
Позиев Алексей Владимирович
22Б-10, Технологии программирования

В чем заключается идея непересекающихся множеств?

Система пересекающихся множеств представляется каким-то множеством деревьев, которые хранятся в массивах, списках или же других коллекциях. Система содержит три основных метода: **MakeSet**, **UnionSet** и **FindSet**

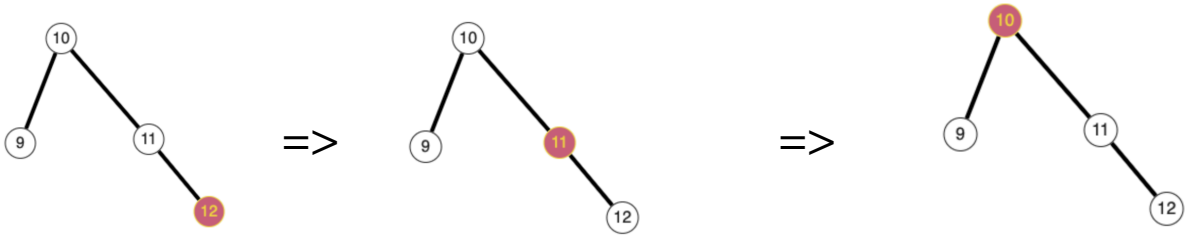
1. **MakeSet** - позволяет **создать** дерево, которое первоначально будет состоять только из одного элемента
2. **UnionSet** - **объединяет** деревья(множества), в которых находятся вершины зачастую меньшее дерево “вешается” на корень большего дерева

Пример: UnionSet(9, 13). 9 лежит в множестве 10, а 13 лежит в множестве 12.



3. **FindSet** - самая важная функция, позволяющая узнать, **в каком множестве мы находимся**, в случае алгоритма Краскала мы можем понять, какому компоненту связности принадлежит наша вершина.

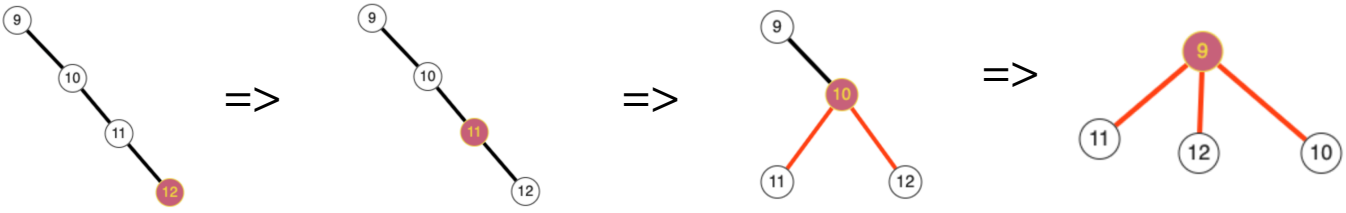
Пример: Хотим узнать, какому множеству принадлежит вершина **12** для этого поднимаемся по родителям, пока не найдем вершины, у которого нет родителя, значение в этой вершине и **будет** номером множества, в котором лежит наша вершина



У **вершины 10** нет вершины, тогда наша вершина 12, содержится в множестве 10.

3.5. Дерево множества может выродится в линию, для исправления таких ситуаций, есть **Сжатие Пути.** Просто все вершины, через которые мы прошли при подъеме, в конечном

Пример: Также хотим найти множество которому принадлежит вершина 12.



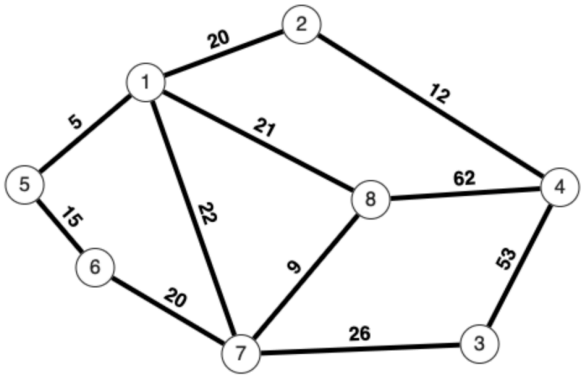
Зачем они нужны в Алгоритме Краскала?

При наивной реализации алгоритма мы сталкиваемся с проблемой: Как понять, что при добавлении дуги **не возникнет цикла?** Хорошей идеей является разделить все вершины на **компоненты связности**, завести массив, который будет для каждой вершины содержать номер компонента связности, в котором она лежит. Но возникает **проблема:** чтобы объединить достаточно большие компоненты связности, по одному из них придется **полностью** пройтись и поменять значение всех его элементов в массиве.

Данная проблема решается с помощью **Системы Непересекающихся Множеств.** С помощью них для смены компонента связности нам не нужно менять значения во всех их элементах.

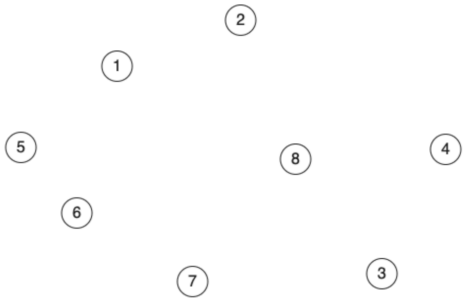
Рассмотрим пример:

Первоначальный граф.



1. Изначально представляем его графом без дуг. Система Непресекающихся множеств представляется из себе 8 деревьев с одной вершиной

Граф



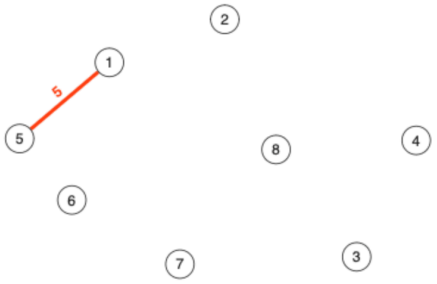
Система непересекающихся множеств

2. Сортируем все дуги по их длинам:

Дуга	1 — 5	7 — 8	2 — 4	5 — 6	1 — 2	6 — 7	1 — 8	1 — 7	3 — 7	3 — 4	4 — 8
Длина	5	9	12	15	20	20	21	22	26	53	62

3. Берем первую длину из списка (1 — 5) и подставляем в граф, также применяем UnionSet(1, 5):

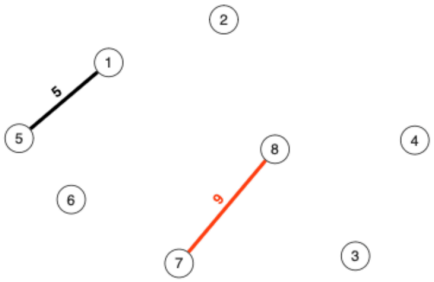
Граф



Система непересекающихся множеств

4. Далее смотрим на следующую по длине дугу, 7 — 8, проверяем не замыкает ли цикл, используем FindSet(7) и FindSet(8), их значения 7 и 8 соответственно. Значения не равны. Значит добавляем дугу в граф и используем UnionSet(7, 8):

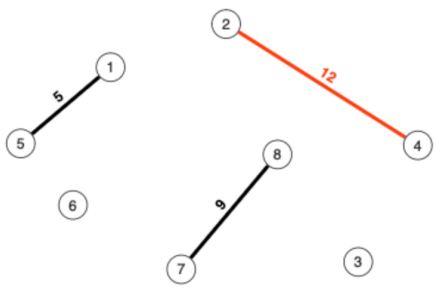
Граф



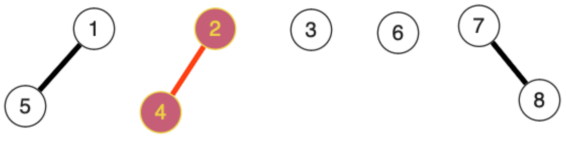
Система непересекающихся множеств

5. Смотрим следующую дугу, 2 — 4. Используем FindSet(2) и FindSet(4). Они принимают значение 2 и 4 соответственно. Значения не равны.
6. Значит добавляем дугу в граф и используем UnionSet(2, 4):

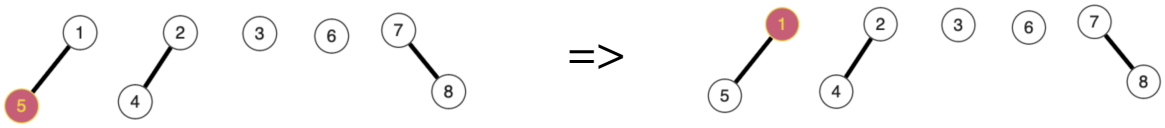
Граф



Система непересекающихся множеств

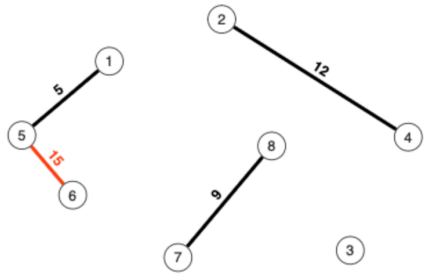


7. Следующая дуга, 5 — 6. Рассмотрим FindSet(5):

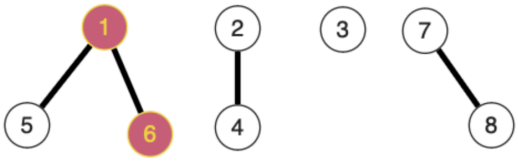


Получается, значение FindSet(5) равно 1, FindSet(6) равно 6. Добавляем дугу и применяем UnionSet(5, 6), т.е. **Добавляем множество 6 в множество 1**, **так как 5 находится в множестве 1.**

Граф

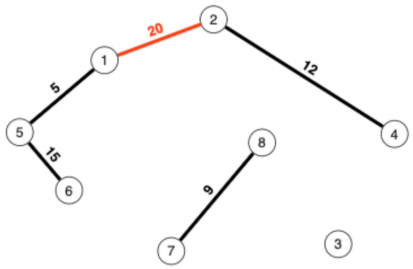


Система непересекающихся множеств

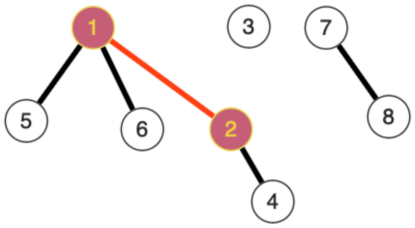


8. При следующем взятии дуги, можем заметить, что длины дуг 1 — 2 и 6 — 7 равны, в таком случае берем любую из них. Возьмем дугу 1 — 2, проверим в каких компонентах связности они находятся: FindSet(2) == 2, FindSet(1) == 1. Значения не равны => добавляем дугу и применяем UnionSet(1, 2).

Граф

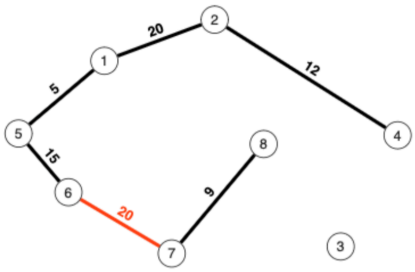


Система непересекающихся множеств

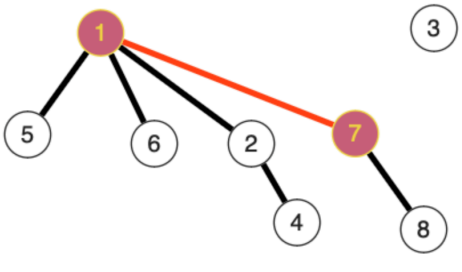


9. С дугой 6 — 7 такой уже проблемы не возникает, проверяем, не замыкает ли цикл и добавляем. FindSet(6) == 1, FindSet(7) == 7. Не равны => добавляем и используем UnionSet(6, 7)

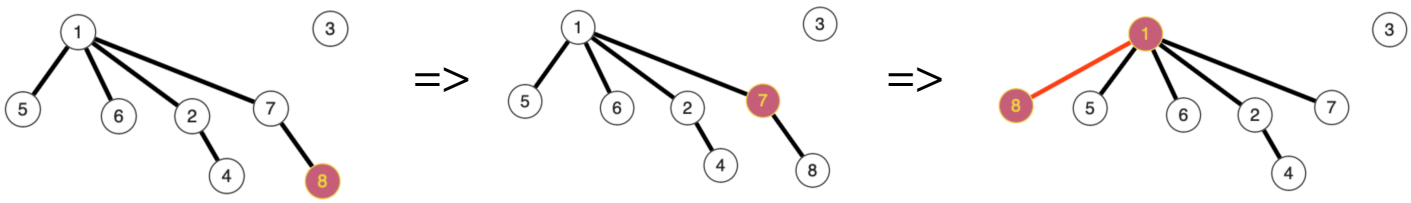
Граф



Система непересекающихся множеств

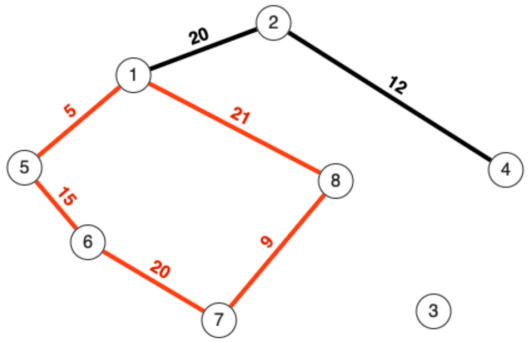


10. Следующая дуга 1 — 8, при ее проверке оказывается, что она замыкает цикл: FindSet(8):



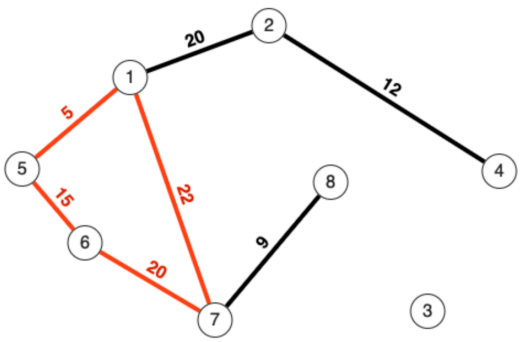
FindSet(8) == 1 и FindSet(1) == 1. Поэтому мы пропускаем данную дугу (Наш граф не меняется) и идем дальше по списку дуг

Такого допускать нельзя



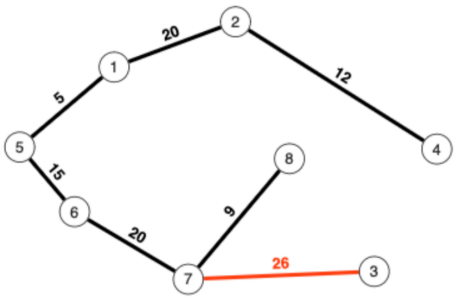
11. Следующая дуга также замыкает цикл: Также как и прошлом случае пропускаем данную дугу и идем к следующей (Граф остается нетронутым)

Такого допускать нельзя

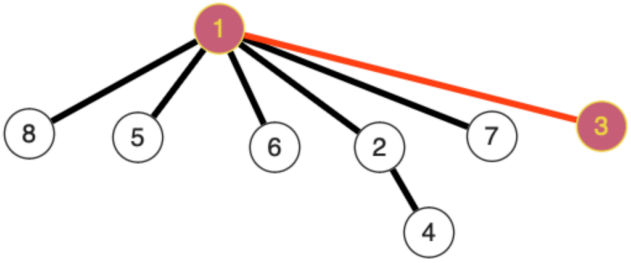


12. Следующая дуга 3 — 7 уже не замыкает цикл => подставляем в граф и используем UnionSet(3, 7)

Граф



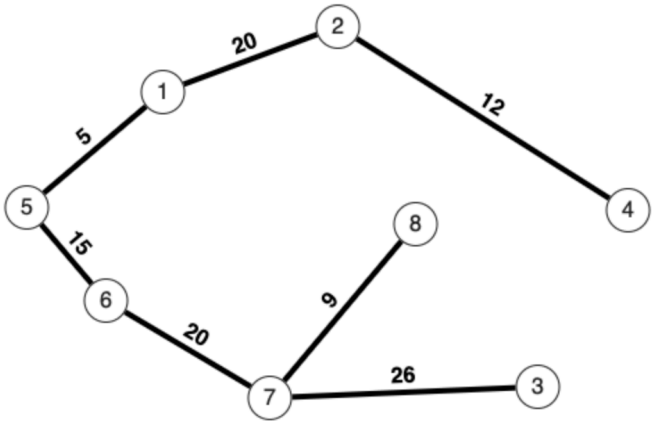
Система непересекающихся множеств



13. Заметим, что все вершины находятся в одном компоненте связности и, следовательно, граф теперь связный, с количеством дуг N - 1, где N - количество вершин, то есть остовное дерево, причем благодаря алгоритму минимальное. На этом алгоритм завершает свою работу.

Результат работы алгоритма Краскала

Граф



Система непересекающихся множеств