Задача «объединить-найти». Система непересекающихся множеств. Алгоритм со сжатием путей сложности O(nG(n))

**Комбинаторика и теория графов**

Егор Миронов БИВТ-23-8 НИТУ МИСИС (https://github.com/Valet-V0ult-de-Fur1e/combinatorics\_and\_graphs\_repo)

2024

**Определение**

Система непересекающихся множеств (СНМ) — иерархическая структура данных, позволяющая эффективно работать с множествами. Структура хранит набор объектов в виде непересекающихся множеств, у каждого множества есть конкретный представитель.

**Примеры применения**

Поиск компонент связности. Например, если два города лежат в разных множествах, то физически не существует пути между ними.

Остов минимального веса (алгоритмы Краскала и Прима). Нужно оставить граф связным, но из всех рёбер взять такие, сумма которых минимальна.

Задачи, связанные с генерированием связанных пространств, например, генерирование лабиринта. Например, если есть поле 3х3 клетки, между каждой клеткой есть стена, то каждая клетка — это множество, и удаление стены между двух клеток можно заменить на объединение двух множеств.

**Теоретическое описание**

Изначально имеется n элементов, каждый из которых находится в отдельном (своём собственном) множестве.

Структура поддерживает две базовые операции:

Объединить два каких-либо множества. При этом все элементы обоих множеств становятся элементами результирующего множества.

Запросить, в каком множестве сейчас находится указанный элемент.

СНМ часто используется в графовых алгоритмах для хранения информации о связности компонент.

**Устройство структуры**

Множества элементов хранятся в виде деревьев: одно дерево соответствует одному множеству. Корень дерева — это представитель (лидер) множества. Для описания множества используется номер вершины, являющейся корнем соответствующего дерева.

Для определения, принадлежат ли два элемента к одному и тому же множеству, для каждого элемента нужно найти корень соответствующего дерева (поднимаясь вверх, пока это возможно) и сравнить эти корни.

Для объединения множеств нужно подвесить корень одного за корень другого.

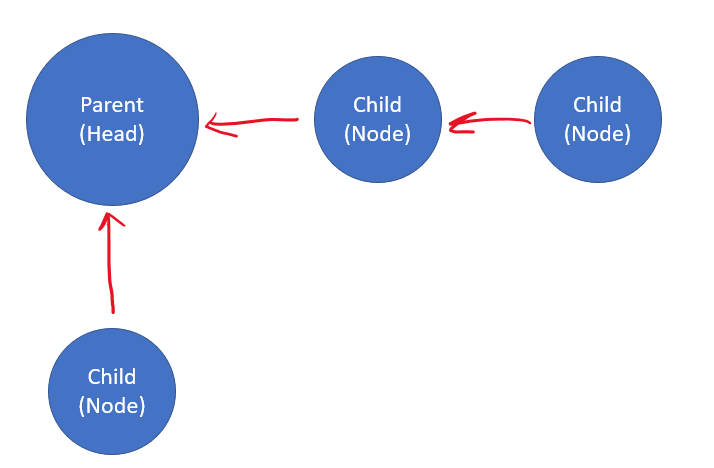


Рис.1 Схема базовой реализации

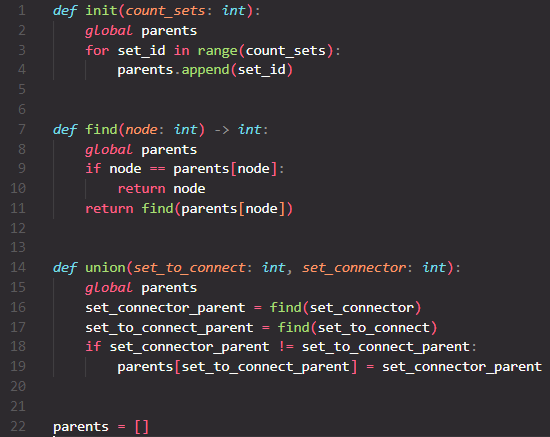


Рис.2 код базовой реализации

**Асимптотика базового поиска «главной» вершины**

В худшем случае такая реализация работает за 𝑂(𝑛)— можно построить «бамбук», подвешивая его n раз за новую вершину.

**Оптимизация**

Для оптимизации можно при поиске «главной вершины» для конкретной подвешивать попутно встречающиеся вершины к «главной», что ускорит поиск

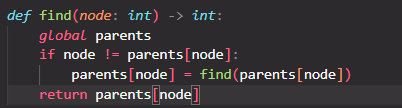


Рис.5 реализация оптимизации сокращения пути

**Асимптотика оптимизированного поиска «главной» вершины**

Применение оптимизации путей позволяет достичь логарифмической асимптотики:

O(log n) в среднем на один запрос. То есть при n запросах асимптотика будет равна O(nlogn)

**Доказательство**

Зафиксируем произвольный элемент и проследим, как на него воздействовали операции объединения. Когда на элемент воздействовали первый раз, можно утверждать, что размер его нового множества будет как минимум 2. Когда на элемент воздействовали второй раз — можно утверждать, что он попадёт в множество размера не менее 4 (так как мы добавляем меньшее множество в большее). И так далее — на элемент могло воздействовать максимум операций объединения. Таким образом, в сумме по всем вершинам это составляет O(nlogn)

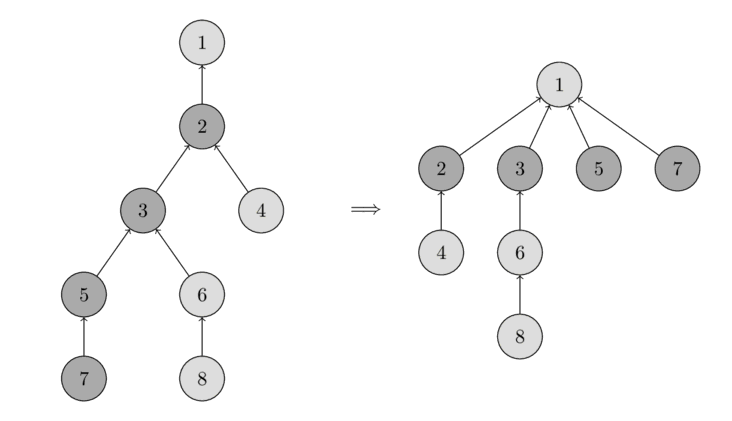


Рис.6 пример работы сокращения пути после поиска «главной» вершины для вершины 7

**Весовая эвристика**

Недостаток наивной реализации проявляется при слиянии относительно большого множества с множеством из одного элемента. В наивной реализации список указанный первым всегда подвешивается ко второму. Хотя в данном случае гораздо выгоднее подвесить меньший список к большему, обновив один указатель на представителя, вместо обновления большого числа указателей в первом списке. Отсюда следуют очевидная оптимизация — будем для каждого множества хранить его размер и изменять указатели на представителя всегда элементам из "меньшего" списка.

Для сравнения вершин при объединении множеств можно использовать количество вершин в каждом множестве.

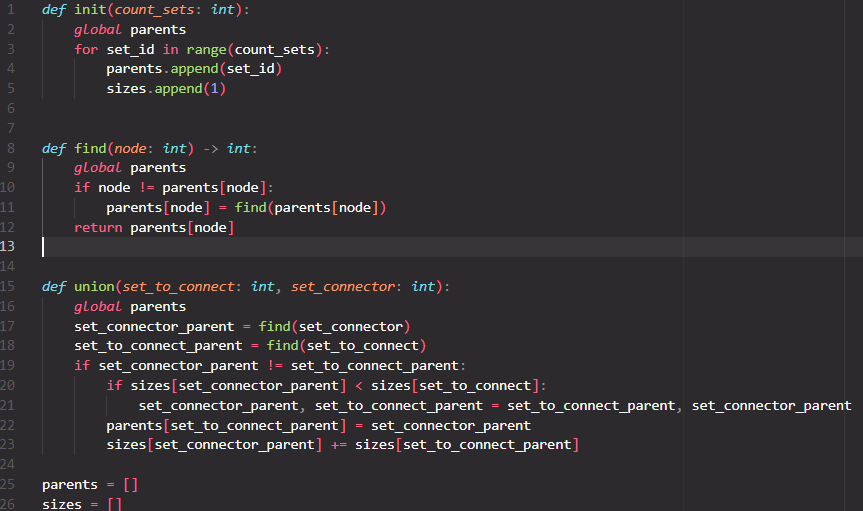


Рис.7 реализация через размерность множеств

Так же для сравнения вершин при объединении множеств можно использовать ранги, то есть количество уровней в дереве множества.



Рис.8 реализация через глубину множеств

**Анализ аналогов**

В качестве альтернативы для поиска компоненты связанности можно использовать dfs с асимптотикой О(n^2)