Домашняя работа № 2

Автор: Минеева Екатерина

Задача 2

Лемма 1. Отношение $\sim$ «должны находиться в одном кластере» является отношением экваивалентности, максимальное число кластеров – число классов эквивалентности.

▲ Проверим, что $\sim$ является отношением экваивалентности:

1. Рефлексивность: действительно, равные строки находятся в одном кластере, так как расстояние между ними меньше 3.
2. Симметричность: действительно, если если s_1, s_2 должны находиться в одном кластере, то s_2, s_1 также должны быть в одном кластере.
3. Транзитивность: действительно, если если s_1, s_2 должны быть в одном кластере и s_2, s_3 должны находиться одном кластере, то s_1 и s_3 не могут находиться в разных кластерах.

Максимальное количество кластеров равно числу классов эквивалентности: с одной стороны кластеров не может быть больше, иначе по принципу Дирихле в найдутся строки из разных кластеров, лежащие в одном классе эквивалентности относительно $\sim$, что невозможно. С другой стороны, разбиение на классы эквивалентности является корректным разбиением на кластеры: между любой парой сторок из разных кластеров (классов эквивалентности) не менее 3.

■

Лемма 2. Если $dist(s_1, s_2) < 3$, то $s_1 \sim s_2$.

▲ Очевидно из определния расстояния между кластерами.

■

Вариант 1. Решение при помощи DSU.

Реализуем структуру DSU, элементы – строки. Если расстояние между строками меньше 3, они находятся в одном множестве. Тогда максимальное количество кластеров равно числу непересекающихся множеств в DSU.

▲ Заметим, что множество элементов DSU с отношением «лежать в одном множестве DSU» изоморфно множеству битовых строк с отношением $\sim$. Поэтому количество классов эквивалентности (максимальное количество кластеров) равняется количеству непересекающихся множеств в DSU.

■

Детали реализации и оценка сложности:

Будем хранить в хэш-таблице пары: ключ – битовая строка, значение – номер соответствующего ей элемента в DSU, построение таблицы $\underline{O}(n)$.

Далее для каждой строки будем перебирать все строки отличающиеся от нее не более, чем 2 битами, и объединять множества, в которых они лежат. Поскольку поиск в хэш-таблице работает за $\underline{O}(1)$, а объединение множеств в среднем за $\underline{O}(\alpha(n))$ (α – обратная функция Аккермана), для каждой строки такая процедура работает $\underline{O}(m^2\alpha(n))$, так делаем для каждой строки, поэтому построение DSU работает за $\underline{O}(m^2 n\alpha(n))$. После чего за $\underline{O}(1)$ выдаем ответ – число непересекающихся множеств.

Итого сложность работы: $\underline{O}(m^2 n\alpha(n))$. Затраченная память $\underline{O}(n)$.

Вариант 2. Решение при помощи поиска компонет связности.

Построим граф, в котором вершины – строки, ребрами соединены строки, расстояние между которыми меньше 3. Заметим, что отношение «находятся в одной компоненте связности» на множестве вершин графа – это в точности отношение «должны находиться в одном кластере» на множестве строк. Таким образом, количество классов эквивалентноси (= максимальное количество кластеров) это количество компонент связности в построенном графе.

Детали реализации и оценка сложности:

Будем хранить в хэш-таблице пары: ключ – битовая строка, значение – номер вершины ей соответствующей. При построении матрицы смежности для каждой вершины-строки, будем перебирать все строки отличающиеся не более, чем 2 битами от вершины и если такая строка встречается во входных данных, то будем соединять ребром соответствующие вершины.

Поскольку поиск в хэш-таблице работает за $\underline{\mathrm{O}}(1)$, поиск всех таких строк работает $\underline{\mathrm{O}}(m^2)$, добавление ребра в матрицу смежности тоже $\underline{\mathrm{O}}(1)$, то для одной строки такая процедура в среднем занимает $\underline{\mathrm{O}}(m^2)$. Делаем это для каждой строки, поэтому построение графа $\underline{\mathrm{O}}(m^2 n)$. Ребер в графе не более, чем $\underline{\mathrm{O}}(m^2 n)$, поэтому поиск компонент связности (серия запусков обхода в ширину или в глубину) работает за $\underline{\mathrm{O}}(m^2 n + n) = \underline{\mathrm{O}}(m^2 n)$.

Итого время работы $\underline{\mathrm{O}}(m^2 n)$, памяти требуется также $\underline{\mathrm{O}}(m^2 n)$ так как храним матрицу смежности.