Курс по олимпиадной подготовке по информатике и программированию

для обучающихся 9—10 классов

Система непересекающихся множества

Определение

Система непересекающихся множеств - структура данных для эффективной работы с непересекающимися множествами (каждый элемент принадлежит только к одному множеству), позволяющий проверять, принадлежат ли два элемента к одинаковому множеству, и объединять множества.

Доступные операции

На СНМ, как на любую структуру данных, накладываются ограничения на то, что должна уметь делать эта структура.

Для СНМ таким операциями являются:

1. make_set(x) – **добавляет** новый элемент x, помещая его в новое множество, состоящее из одного него.
2. union_sets(x, y) – **объединяет** два указанных множества (множество, в котором находится элемент x, и множество, в котором находится элемент y).
3. find_set(x) – **возвращает**, **в каком множестве** находится указанный элемент x. В общем случае эта операция возвращает представителя множества.

Представление структуры данных

Чаще всего, СНМ реализуется на статическом массиве, ввиду преимущества в виде константного доступа к памяти. Для хранения нам понадобиться лишь один массив par, который помогает определить, кто является представителем множества.

Утверждается, $par_i = i$, тогда и только тогда, когда i элемент является представителем множества, в которое он входит.

Make_set

```
void make_set(int v)
{
    par[v] = v;
}
```

Код описанный выше, на самом деле реализует достаточный для нас функционал.
Он создает множество из одного элемента, в котором представителем будет являться сам элемент.

Итоговая сложность одной операции – O(1)

find_set

```
int find_set(int v)
{
    while (v != par[v])
        v = par[v];

    return v;
}
```

Будем итеративно проходиться по вершинам, пока не найдем ту, которая является представителем этого множества.

В худшем случае дерево, образованное подобными ребрами будет иметь вид бамбука, и мы получим сложность порядка O(n), на одну операцию.

Union_sets

```
void union_sets(int v, int u)
{
    v = find_set(v);
    u = find_set(u);
    if (v < u)
        par[v] = u;
    else
        par[u] = v;
}
```

Код описанный выше для каждой пары вершин находит их представителя, и “привязывает” одного представителя к другому.

Итоговая сложность одной операции зависит от сложности операции find_set и равна O(n).

Ассоциативные и коммутативные функции

Помимо просто поддержания множеств как таковых, можно поддерживать ассоциативные и коммутативные функции на них.

Операция ∅ называется ассоциативной, если её результат не зависит от того, в каком порядке её вычислять.

Операция ∅ называется коммутативной, если её результат не зависит от перестановки оперируемых.

Задача

На вход подается граф из n вершин и m ребер.
Следом поступает q запросов одного из двух типов:

1. Поступает пара вершин a, b, между которыми необходимо провести ребро.
2. Поступает пара вершин a, b, для которых необходимо определить, лежат ли они в одной компоненте.

Курс по олимпиадной подготовке по информатике и программированию

для обучающихся 9—10 классов

Спасибо!

Контакты