Федеральное государственное автономное образовательное учреждение  
высшего образования

«Нижегородский государственный университет

им. Н.И. Лобачевского»

Институт информационных технологий математики и механики

**Отчёт по лабораторной работе**

**«Percolation: разделённые множества»**

**Выполнила:**

студентка группы 0823-2

Кабалова А.А.

**Проверил:**

ассистент каф. МОСТ института ИТММ

Сиднев А.А.

Нижний Новгород

2016 г.

Оглавление

[Введение 3](#_Toc453889635)

[Постановка задачи 4](#_Toc453889636)

[Структура проекта 5](#_Toc453889637)

[Описание Google Tests 7](#_Toc453889638)

[Заключение 8](#_Toc453889639)

# Введение

Теории перколяции уже более пятидесяти лет. Ежегодно на западе публикуются сотни статей, посвященных как теоретическим вопросам перколяции, так и ее приложениям.

С точки зрения математика, теорию перколяции следует отнести к теории вероятности в графах. С точки зрения физика – перколяция – это геометрический фазовый переход. С точки зрения программиста – широчайшее поле для разработки новых алгоритмов. С точки зрения практика – простой, но мощный инструмент, позволяющий в едином подходе решать самые разнообразные жизненные задачи.

Сама задача реализуется с помощью разделённых множеств – абстрактный тип данных, предназначенный для представления набора попарно непересекающихся подмножеств. Формирование новых множеств происходит с помощью:

* создания одноэлементного множества;
* объединение уже существующих множеств.

# Постановка задачи

Имеется прямоугольная решетка, состоящая из N×M открытых и закрытых ячеек. Будем называть кластером соседние открытые ячейки и стягивающим кластером (перколяционным кластером) такой кластер, который начинается на одной границе и заканчивается на противоположной границе решетки. Необходимо установить долю открытых ячеек, при которой возникает стягивающий кластер, то есть имеется путь от верхней до нижней границы решётки.

Таким образом, необходимо решить следующую задачу: реализовать структуру данных разделённые множества с использованием древесной структуры с рангами. С помощью разделённых множеств реализовать алгоритм поиска числа открытых ячеек, при котором в решётке имеется стягивающий кластер. Провести эксперименты, определяющий долю открытых ячеек. Для структуры данных разделённые множества написать тесты с помощью Google C++ Testing Framework. Разработать консольное приложение для демонстрации результатов эксперимента.

# Структура проекта

Описание классов.

Класс «Разделённое множество» *(set)*. Разделенные множества — это абстрактный тип данных, предназначенный для представления коллекции, состоящей из некоторого числа k попарно непересекающихся подмножеств. Этот тип данных применяется в таких задачах, как поиск минимального остовного дерева для заданного взвешенного неориентированного графа, построение компонент связности графа, минимизация конечного автомата, и многих других.

Операции над разделенными множествами:

1. Создание синглетона. Эта операция предназначена для введения в коллекцию нового подмножества, состоящего из одного элемента x, при этом предполагается, что x не входит ни в одно из подмножеств коллекции, созданной к моменту выполнения этой операции. Элемент x указывается в качестве параметра. Именем созданного подмножества будет считаться сам элемент x.
2. Объединение двух множеств. С помощью этой операции можно объединить два подмножества коллекции, имеющие, соответственно, имена x и y, в одно новое подмножество, при этом оба объединяемые подмножества удаляются из коллекции, а вновь построенное подмножество получает некоторое имя. Во всех рассматриваемых нами случаях именем нового полученного в результате этой операции подмножества будет одно из имен x или y. Имена объединяемых подмножеств указываются в качестве параметров.
3. Операция поиска. Эта операция позволяет определить имя y того подмножества коллекции, которому принадлежит элемент x. Если элемент x до выполнения операции не входил ни в одно из подмножеств коллекции, то в качестве y берется 0.

Реализовать разделённые множества можно с помощью:

* с помощью массива;
* с помощью древовидной структуры;
* с помощью древовидной структуры с использованием рангов вершин.

Реализация с помощью массива.

Пусть А — множество, из элементов которого будет строиться коллекция разделенных подмножеств. Одним из очевидных способов представления коллекции является представление ее с помощью массива. При таком способе для каждого элемента i в соответствующей (i -й) ячейке массива помещаем имя (канонический элемент) того подмножества, которому принадлежит элемент i. Если элемент i не принадлежит ни одному из подмножеств коллекции, то в i -ю ячейку записываем 0.

Реализация с помощью древовидной структуры.

Каждое подмножество коллекции представляется корневым деревом, узлы которого являются элементами этого подмножества, то есть отождествляются с номерами из множества {1, 2…n}. Корень дерева используется в качестве имени соответствующего подмножества (канонический элемент). Для каждого узла дерева определяется узел p(x), являющийся его родителем в дереве; если x — корень, то полагаем p(x) = x.

Фактически в памяти компьютера это дерево представляется массивом p[1…n] так, что p(x) будет предком узла x, если x не является корнем, и p(x) = x, если x — корень. Если же x не входит ни в одно из подмножеств коллекции, то p(x) = 0.

Реализация операций с использованием рангов вершин

Для такой реализации разделенных множеств необходимо хранить с каждым узлом x дополнительно еще одну величину — высоту поддерева, корнем которого является узел x. Будем называть ее высотой, или рангом, узла x. Остальные операции нужно настроить на корректную работу с этим полем. Будем хранить высоту каждого узла x в ячейке h[x] массива h.

В данном случае использовалась последняя структура.

## Описание Google Tests

1. TEST(Sets, can\_create\_set). Проверка создания множества на корректность.
2. TEST(Sets, can\_not\_creat\_set\_with\_negative\_size). Невозможность создания множества с отрицательным размером.
3. TEST(Sets, can\_not\_add\_element\_with\_too\_large\_index\_in\_set). Невозможность добавления элемента с большим индексом, чем размер множества.
4. TEST(Sets, can\_merge\_sets). Проверка функции слияния двух множеств.
5. TEST(Sets, Sets, can\_not\_creat\_node\_which\_creat\_early). Невозможность добавления уже добавленного элемента.
6. TEST(Sets, can\_search\_element). Проверка функции поиска множества с заданным элементом.
7. TEST(Sets, can\_not\_search\_element\_with\_negativ\_index). Невозможность поиска элемента с отрицательным размером.
8. TEST(Sets, can\_not\_merge\_sets\_which\_one\_of\_two\_sets\_is\_empty). Проверка создания исключения при вызове функции с некорректным параметром.

# Заключение

В ходе выполнения данного проекта было достигнуто несколько результатов. Приведём самые важные из них. Была написана программа, выполняющая подсчёт доли открытых ячеек. Разработан класс разделённые множества, реализованный с помощью древесной структуры с рангами. Были проведены эксперименты с решётками различных размеров. Результаты экспериментов представлены в таблице.

|  |  |  |
| --- | --- | --- |
| Размер матрицы | Количество экспериментов | Результат |
| 50x50 | 15.000 | 0.6028 |
| 100x100 | 10.000 | 0.6003 |
| 200x200 | 50.000 | 0.5963 |