Муниципальное автономное общеобразовательное учреждение «Лицей №7»

**Тема:**

**сравнительная характеристика алгоритмов Прима и Крускала при решении олимпиадных задач**

Выполнила:
*Иванова Юлия Анатольевна*,
учитель информатики

(подпись)

*22.05.2024г.*

г.Бердск 2024г.

Оглавление

[Цель работы: 2](#_Toc167261808)

[Актуальность проекта: 2](#_Toc167261809)

[Задачи: 2](#_Toc167261810)

[Определения из теории графов. 2](#_Toc167261811)

[Описание алгоритмов Прима и Крускала. 3](#_Toc167261812)

[Реализация алгоритмов Прима и Крускала. 5](#_Toc167261813)

[Текст программы в Приложении 1. 5](#_Toc167261814)

[Сравнительный анализ времени работы алгоритмов на различных графах. 5](#_Toc167261815)

[Список литературы: 8](#_Toc167261816)

Цель работы: определить более эффективный из двух алгоритмов для нахождения остовного дерева

Актуальность проекта: на олимпиадах по информатике популярными являются задачи, включающие нахождение остовного дерева графа. При этом, существует 2 основных алгоритма: Прима и Крускала. Существует общая рекомендация, что, если в графе много вершин и мало ребер (разреженный граф), то применяется алгоритм Крускала. Если же мало вершин и много ребер – алгоритм Прима. Однако, встречаются ситуации, когда определиться с выбором алгоритма на основании понятий «мало» и «много» затруднительно. На уроках информатики и проектной мастерской мы с учеником 10А класса Болуц Романом(победитель муниципальной, региональной олимпиад, а также призер Всероссийской и Всесибирской олимпиад) поставили задачу: получить более точные критерии для выбора алгоритма.

# Задачи:

1. Дать необходимые определения из теории графов.
2. Описать алгоритмы Прима и Крускала.
3. Написать реализацию данных алгоритмов.
4. Провести сравнительный анализ времени работы алгоритмов на различных графах и сформулировать критерии для выбора алгоритма в зависимости от количества вершин и ребер в графе.

# Определения из теории графов.

**Граф** - это структура, состоящая из множества **вершин**, в которых некоторые пары связаны **ребрами**.

Графы делятся на **ориентированные** и **неориентированные**. Мы будем рассматривать только **неориентированные** графы.

**Неориентированный граф** – граф, рёбра которого не имеют определённого направления.

**Путь** - последовательность вершин, в которой каждая вершина соединена со следующей ребром.

**Цикл** – замкнутый путь, т.е. первая и последняя вершины совпадают.

Все графы можно разделить на 2 группы: **связные** и **несвязные**. **Связный** граф – это граф, у которого между любой парой вершин этого графа существует как минимум один путь.

**Дерево** – связный ацикличный граф, то есть, связный граф без циклов. Можно доказать, что дерево на n вершинах будет иметь ровно n-1 ребро.

**Остовное дерево** — это  дерево, подграф данного графа, с тем же числом вершин, что и у исходного графа.

**Остовное дерево (остов графа)** — это связный подграф, не имеющий циклов, который содержит все вершины заданного графа.

Также графы делятся на **взвешенные** графы и **невзвешенные**. Мы будем рассматривать **взвешенные** графы. Взвешенные графы – это графы, на ребрах которого указано число (так называемый **вес**).

**Минимальное остовное дерево** (или **миностов**) в неориентированном связном **взвешенном** графе— это остовное дерево графа, имеющее минимальный возможный вес, где под весом дерева понимается сумма весов входящих в него рёбер.

#

# Описание алгоритмов Прима и Крускала.

Перед нами стоит задача нахождения минимального остовного дерева графа.

Для решения этой задачи подходят 2 алгоритма: алгоритм Прима и алгоритм Крускала.

## Алгоритм Прима.

Искомый минимальный остов строится постепенно, добавлением в него рёбер по одному. Изначально остов полагается состоящим из единственной вершины (её можно выбрать произвольно). Затем выбирается ребро минимального веса, исходящее из этой вершины, и добавляется в минимальный остов. После этого остов содержит уже две вершины, и теперь ищется и добавляется ребро минимального веса, имеющее один конец в одной из двух выбранных вершин, а другой — наоборот, во всех остальных, кроме этих двух. И так далее, т.е. всякий раз ищется минимальное по весу ребро, один конец которого — уже взятая в остов вершина, а другой конец — ещё не взятая, и это ребро добавляется в остов (если таких рёбер несколько, можно взять любое). Этот процесс повторяется до тех пор, пока остов не станет содержать все вершины (или, что то же самое,  ребро).

Если искать каждый раз ребро простым просмотром среди всех возможных вариантов, то асимптотически будет требоваться просмотр О(m) рёбер, чтобы найти среди всех допустимых ребро с наименьшим весом. Суммарная асимптотика алгоритма составит в таком случае О(nm) (n – количество вершин, а m – количество ребер)

Подойдём к вопросу поиска наименьшего ребра с другой стороны: для каждой ещё не выбранной будем хранить минимальное ребро, ведущее в уже выбранную вершину.

Тогда, чтобы на текущем шаге произвести выбор минимального ребра, надо просто просмотреть эти минимальные рёбра у каждой не выбранной ещё вершины — асимптотика составит О(n)

Но теперь при добавлении в остов очередного ребра и вершины эти указатели надо пересчитывать. Заметим, что эти указатели могут только уменьшаться, т.е. у каждой не просмотренной ещё вершины надо либо оставить её указатель без изменения, либо присвоить ему вес ребра в только что добавленную вершину. Следовательно, эту фазу можно сделать также за O(n). Таким образом, мы получили вариант алгоритма Прима с асимптотикой О(n^2+m).

## Алгоритм Крускала

Алгоритм Крускала изначально помещает каждую вершину в своё дерево, а затем постепенно объединяет эти деревья, объединяя на каждой итерации два некоторых дерева некоторым ребром. Перед началом выполнения алгоритма, все рёбра сортируются по весу (в порядке неубывания). Затем начинается процесс объединения: перебираются все рёбра от первого до последнего (в порядке сортировки), и если у текущего ребра его концы принадлежат разным поддеревьям, то эти поддеревья объединяются, а ребро добавляется к ответу. По окончании перебора всех рёбер все вершины окажутся принадлежащими одному поддереву, и ответ найден.

Сортировка ребер занимает О(m\*log n) времени. Чтобы научиться быстро объединять поддеревья и проверять, принадлежат ли его концы разным поддеревьям. С этой задачей справляется структура данных «Система непересекающихся множеств» (СНМ). Запросы в этой структуре данных выполняется примерно за О(1). Запросов у нас О(m).

 В итоге получается алгоритм за О(m\*log n).

# Реализация алгоритмов Прима и Крускала.

Алгоритмы реализованы на языке С++, как основном, используемом в олимпиадном программировании.

Будем хранить граф в списке смежности (для каждой вершины в списке указаны все ребра, выходящие из нее).

В качестве параметров в программе задаются количество вершин и количество ребер графа.

Сам граф, а также веса ребер (от 0 до 99999) задаются с помощью датчика случайных чисел. Вначале случайным образом зададим дерево (чтобы граф был связным), а затем случайно добавим ребра, и веса на ребра.

Для каждого полученного графа осуществляем поиск остовного дерева двумя алгоритмами. Программа выводит для каждого дерева ответ (вес миностова) и затраченное время.

Совпадение ответов на множестве тестов подтвердит правильность работы алгоритмов.

# Текст программы в Приложении 1.









#

# Сравнительный анализ времени работы алгоритмов на различных графах.

Так как нас интересует соотношение количества ребер к количеству вершин, то, можно зафиксировать количество вершин (скажем, 10000) и менять только параметр – количество ребер. Полученные результаты приведены в таблице. (время в мс)

Для наглядности, посмотрим на результаты в графическом виде:

Как мы видим, время работы алгоритма Прима определяется фактически только количеством вершин и очень мало зависит от количества ребер. Это согласуется с асимптотикой данного алгоритма: О(n^2+m), где n – количество вершин, m – количество ребер. Как результат, мы получаем фактически горизонтальную прямую. Этот алгоритм можно смело использовать при маленьком количестве вершин.

Время работы алгоритма Крускала определяется, прежде всего, количеством ребер, что также согласуется с его асимптотикой: О(m\*log n).

В нашем примере графики пересекаются примерно при m = 1000000.

Если пытаться определить точку пересечения на основании асимптотики, то мы получим: для n=10000

n^2+m=m\*log n => m = n^2/(log n – 1) = (10000^2)/(log 10000 – 1) = 10^8/12,3 = 8000000

То есть, если вычислять чисто по асимптотикам, то переходить от Прима к Крускалу нужно при 8 млн ребер. В действительности, оказалось, что переход надо осуществлять уже при 1 млн. (отличается в 8 раз) Это объясняется тем, что асимптотические оценки задают характер поведения функции без конкретного коэффициента.

Проведя подобный эксперимент при количестве вершин n = 5000, мы получили, что вместо 2,3 млн ребер переход выгодно осуществлять уже при 250 тысячах (отличается в 9 раз)

Получаем более точную границу перехода от Прима к Крускалу, а именно: вместо m = n^2/(log n – 1) переход будет выгодно осуществить примерно при значении m примерно в 10 раз меньшем.

Вывод: Более точной формулой, определяющей границу перехода от алгоритма Прима к алгоритму Крускала является формула m = n^2/((log n – 1) \* 10), то есть, алгоритм Прима следует применять, количество ребер меньше указанной величины, а алгоритм Крускала, если больше.

# Список литературы:

1. <http://e-maxx.ru/algo/mst_prim>
2. <http://e-maxx.ru/algo/mst_kruskal>
3. <http://e-maxx.ru/algo/dsu>