Теория графов

Изображение выглядит как текст, Шрифт, линия, диаграмма

Автоматически созданное описаниеГраф состоит из вершин, соединенных ребрами. Мы будем обозначать буквой п количество вершин графа, а буквой т - количество ребер. Ребра нумеруются числами 1, 2, …, п. На рис. 7.1 изображен граф с 5 вершинами и 7 ребрами.

Изображение выглядит как текст, Шрифт, диаграмма, линия

Автоматически созданное описание

Граф называется связным, если между любыми двумя вершинами существует путь. Левый граф на рис. 7.4 связный, а правый - нет, потому что из вершины 4 нельзя попасть ни в какую другую.

Деревом называется связный граф без циклов. На рис. 7.6 приведен пример дерева.

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описаниеИзображение выглядит как текст, Шрифт, диаграмма, линия

Автоматически созданное описание

Во взвешенном графе каждому ребру сопоставлен вес. Часто веса интерпретируются как длины ребер, а длиной пути считается сумма весов составляющих его ребер. На рис. 7.8 изображен взвешенный граф, длина пути 1 -> 3 -> 4 -> 5 равна 1 + 7 + 3 = 11. Это кратчайший путь из вершины 1 в вершину 5.

Связные части графа называются его компонентами связности. Граф на рис. 7.5 состоит из трех компонент связности: {1, 2, 3}, {4, 5, 6, 7} и {8}.

Изображение выглядит как текст, линия, Шрифт, диаграмма

Автоматически созданное описаниеИзображение выглядит как текст, линия, диаграмма, Шрифт

Автоматически созданное описание

В ориентированном графе по каждому ребру можно проходить только в одном направлении. На рис. 7.7 показан пример ориентированного графа. В нем имеется путь 3 -> 1 -> 2 -> 5 из вершины 3 в вершину 5, но не существует пути из вершины 5 в вершину 3.

Путь ведет из одной вершины в другую и проходит по ребрам графа. Длиной пути называется количество ребер в нем. На рис. 7.2 показан путь 1 -> 3 -> 4 -> 5 длины 3 из вершины 1 в вершину 5. Циклом называется путь, в котором последняя вершина совпадает с первой. На рис. 7.3 изображен цикл 1 -> 3 -> 4 -> 1.

Изображение выглядит как текст, линия, диаграмма, Шрифт

Автоматически созданное описание

Две вершины называются соседними, или смежными, если они соединены ребром. Степенью вершины называется число соседних с ней вершин. На рис. 7.9 показаны степени всех вершин графа. Так, степень вершины 2 равна 3, потому что ее соседями являются вершины 1, 4 и 5.

Изображение выглядит как текст, Шрифт, снимок экрана, письмо

Автоматически созданное описаниеИзображение выглядит как текст, диаграмма, линия, Шрифт

Автоматически созданное описание

Сумма степеней всех вершин графа равна 2m, где m - число ребер, поскольку каждое ребро увеличивает степени ровно двух вершин на единицу. Таким образом, сумма степеней вершин всегда четна. Граф называется регулярным, если степени всех его вершин одинаковы (равны некоторой константе d). Граф называется полным, если степень каждой его вершины равна n - 1, т. е. в графе присутствуют все возможные ребра.

В ориентированном графе полустепенью захода вершины называется число ребер, оканчивающихся в этой вершине, а полустепенью исхода - число ребер, начинающихся в вершине. На рис. 7.10 показаны полустепе- ни захода и исхода для каждой вершины графа. Например, для вершины 2 полустепень захода равна 2, а полустепень исхода - 1.

Граф называется двудольным, если его вершины можно раскрасить в два цвета, так что цвета любых двух соседних вершин различны. Можно доказать, что граф является двудольным тогда и только тогда, когда в нем нет цикла с нечетным числом вершин. На рис. 7.11 изображен двудольный граф и его раскраска.

Изображение выглядит как текст, снимок экрана, диаграмма, круг

Автоматически созданное описание

Обход графа

**Поиск в глубину** (depth-first search - DFS) - прямолинейный способ обхода графа. Алгоритм начинает работу в начальной вершине и перебирает все вершины, достижимые из нее по ребрам графа. Поиск в глубину всегда следует по одному пути, пока на нем еще имеются вершины. После этого он возвращается назад и начинает исследовать другие части графа. Алгоритм запоминает посещенные вершины, так что каждая обрабатывается только один раз. На рис. 7.13 показан порядок обработки вершин при поиске в глубину. Поиск может начинаться с любой вершины: в данном случае мы начали с вершины 1. Сначала исследуется путь 1 -> 2 -> 3 -> 5, затем алгоритм возвращается к вершине 1 и посещает оставшуюся вершину 4.

Изображение выглядит как круг, диаграмма, линия

Автоматически созданное описаниеИзображение выглядит как текст, диаграмма, круг, шаблон

Автоматически созданное описание

При **поиске в ширину** (breadth-first search - BFS) вершины графа посещаются в порядке возрастания расстояния от начальной вершины. Следовательно, используя поиск в ширину, мы сможем вычислить расстояния от начальной вершины до всех остальных. Однако реализовать поиск в ширину труднее, чем поиск в глубину. В процессе поиска в ширину мы обходим вершины уровень за уровнем. Сначала посещаются вершины, отстоящие от начальной на расстоянии 1, затем - на расстояние 2 и т. д. Процесс продолжается, пока не останется не посещённых вершин. На рис. 7.14 показано, как происходит обход графа при поиске в ширину. Предположим, что поиск начинается с вершины 1. Сначала мы посещаем вершины 2 и 4, удаленные на расстояние 1, затем - вершины 3 и 5, удаленные на расстояние 2, и, наконец, вершину 6, удаленную на расстояние 3.

Применение обходов

С помощью алгоритмов обхода мы можем проверить многие свойства графа. Обычно применимы как поиск в глубину, так и поиск в ширину, но на практике поиск в глубину предпочтительнее, потому что он проще реализуется. В описываемых ниже применениях предполагается, что граф неориентированный.

**Проверка связности**. Граф называется связным, если между любыми двумя вершинами существует путь. Следовательно, для проверки связности мы можем начать с произвольной вершины и выяснить, все ли вершины достижимы из нее. На рис. 7.15 видно, что поиск в глубину, начатый из вершины 1, не посещает все вершины, поэтому можно заключить, что граф не связный. Можно также найти все компоненты графа: для этого нужно перебрать все вершины и начинать новый поиск в глубину, если текущая вершина не принадлежит ни одной из уже найденных компонент связности.

**Обнаружение циклов**. Граф содержит цикл, если в процессе обхода мы встречаем такую вершину, что одна из соседних с ней (кроме той, что предшествует ей на текущем пути) уже посещалась. На рис. 7.16 поиск в глубину из вершины 1 обнаруживает, что в графе имеется цикл. При переходе из вершины 2 в вершину 5 мы видим, что соседняя с 5 вершина 3 уже посещалась. Следовательно, граф содержит цикл, проходящий через вершину 3, например 3 -> 2 -> 5 -> 3. Есть и другой способ узнать, содержит ли граф цикл: просто посчитать количество вершин и ребер в каждой компоненте. Если компонента содержит с вершин и ни одного цикла, то в ней должно быть ровно с - 1 ребер (т. е. она должна быть деревом). Если число ребер равно с или больше, то компонента обязательно содержит цикл.

Изображение выглядит как круг, диаграмма, линия, часы

Автоматически созданное описаниеИзображение выглядит как круг, часы, снимок экрана, Шрифт

Автоматически созданное описаниеИзображение выглядит как круг, диаграмма, снимок экрана, линия

Автоматически созданное описание

**Проверка на двудольность**. Граф называется двудольным, если его вершины можно раскрасить двумя цветами, так что никакие две соседние вершины не будут окрашены одним цветом. Удивительно, как просто можно проверить граф на двудольность с помощью алгоритмов обхода. Идея в том, чтобы выбрать два цвета Хи Y, окрасить начальную вершину цветом Х, всех ее соседей - цветом Y, всех их соседей - цветом Х и т. д. Если в какой-то момент мы обнаруживаем, что две соседние вершины окрашены одним цветом, значит, граф не является двудольным. В противном случае граф двудольный, и мы нашли доказывающую это раскраску. Поиск в глубину из вершины 1 показывает, что граф на рис. 7.17 не является двудольным, т. к. вершины 2 и 5 окрашены одним цветом, хотя и являются соседними. Этот алгоритм корректен, потому что при наличии всего двух цветов цвет начальной вершины однозначно определяет цвета всех остальных вершин, принадлежащих той же компоненте связности. Отметим, что в общем случае трудно определить, можно ли раскрасить вершины графа в к цветов, так чтобы никакие две соседние вершины не были окрашены одним цветом. Для k = 3 эта задача является NP-трудной.