Методы дискретной оптимизации

Алгоритмы поиска в глубину и поиска в ширину и их сложность

Введение

Поиск в графе сводится к обходу вершин графа в той или иной последовательности. В зависимости от цели поиска, структуры графа алгоритмы обхода могут быть различными. Например, поиск данных по пользователям социальной сети предполагает различные алгоритмы чтения при поиске конкретного пользователя или при наборе статистики по предпочтениям конкретных групп. Само по себе чтение данных может занимать недопустимо большое время, поэтому исследование эффективности алгоритмов чтения является актуальной проблемой.

Представление графов

Два стандартных способа хранения графов: в виде матрицы смежности и в виде списка смежности. Матрицы смежности имеют простую структуру, полезны при изучении свойств графов. При выполнении некоторых стандартных операций, например, при определении существования ребра, соединяющего две заданные вершины, хранение графа в виде матрицы смежности позволяет выполнить эти операции наиболее быстро. Недостаток матриц смежности: необходимость хранить массивы, состоящие в основном из нулей (при малом количестве ребер графа). С точки зрения экономии объема памяти представление в виде матрицы смежности целесообразно в случае плотного графа, т.е. при условии, что отношение близко $|E|/|V|^2$ к 1. Другой метод хранения графа – представления его в виде списка смежности. Список смежности графа G, т.е. $Adj(G)$ – набор последовательностей вида $\{v|(u,v) \in E\} \forall u \in V$. Таким образом, список смежности состоит из $|V|$ последовательностей, в каждой из которой $|E|$ элементов. Для орграфа длина списка смежности равна $|E|$, для неориентированного – $2|E|$. В обоих случаях необходимый для хранения объем памяти равен $O(|V| + |E|)$. В случае матрицы смежности необходимый объем равен $O(|V|^2)$.

Обозначения

1. Атрибуты вершин и ребер обозначаются после точки, следующей за именем данной вершины (ребра).
2. Множество вершин, смежных с u, обозначается как $Adj[u]$. Таким образом, списки смежности для данного графа – набор последовательностей $Adj[u]$ для всех u из множества V.

Задача обхода графа:

Обойти все вершины графа G исходя из $s \in V$.

Поиск в ширину

Смысл термина: в данном алгоритме обход происходит в соответствии с очередностью в списке $Adj[u]$, где в качестве начальной вершины u рассматривается s, а затем – первая в построенном списке очереди. Для наиболее простого описания алгоритма каждой вершине u из множества V задаются атрибуты: цвет, расстояние от исходной вершины s, имя предшественника, т.е. предыдущей вершины. Принцип работы алгоритма состоит в построении очереди Q. На каждом шаге первая стоящая в очереди вершина окрашивается в красный цвет и удаляется из очереди, в которую по определенному правилу становятся другие вершины. Таким образом, каждая вершина может быть окрашена в три цвета: зеленый, желтый и красный. В красный цвет окрашивается вершина, просмотренная и удаленная из очереди. В зеленый цвет окрашены вершины, которые не поставлены в очередь. Желтые вершины – это те, которые находятся в очереди.

Поиск в ширину. Описание алгоритма

1. $\forall u \in V \backslash \{s\} : u.color = green; u.d = \infty; u.\pi = \emptyset$
2. $s.color = yellow; s.d = 0; s.\pi = \emptyset$
3. $Q = \emptyset$
4. $s \to Q$
5. While $Q \neq \emptyset$
6. Положить u: первая в списке Q
7. Для каждой вершины $v \in Adj[u]$
8. $v.color = green \Rightarrow v.color = yellow; v.d = u.d + 1; v.\pi = u$
9. $v \to Q$
10. $u.color = red$

Пример

$$B = \begin{pmatrix} & a & b & c & d & e & f \\ a & 0 & 1 & 0 & 1 & 1 & 0 \\ b & 1 & 0 & 1 & 0 & 1 & 1 \\ c & 0 & 1 & 0 & 0 & 0 & 1 \\ d & 1 & 0 & 0 & 0 & 0 & 0 \\ e & 1 & 1 & 0 & 0 & 0 & 1 \\ f & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}; List : \begin{pmatrix} a \to b, d, e \\ b \to a, c, e, f \\ c \to b, f \\ d \to a \\ e \to a, b, f \\ f \to b, c, e \end{pmatrix}$$

Для начальной вершины b последовательность шагов алгоритма:

- 1. Желтая вершина
  $u.color = yellow, v.color = green, v \in V \backslash \{u\}, u.d = 0, Q = \{b\}$
  $v.d = \infty, v \in V$
  $Adj[u] = \{a, c, e, f\} \Rightarrow$
  $a.color = yellow; a.d = 0 + 1 = 1; a.\pi = u = b$
  $c.color = yellow; c.d = 0 + 1 = 1; c.\pi = u = b$
  $e.color = yellow; e.d = 0 + 1 = 1; e.\pi = u = b$
  $f.color = yellow; f.d = 0 + 1 = 1; f.\pi = u = b$
  $Q = \{a, c, e, f\}$
  $b.color = red$

Пример

- 2.

$$
\begin{aligned}
&u = a \\
&Adj[u] = \{b, d, e\} \Rightarrow \\
&d.color = yellow \\
&d.d = a.d + 1 = 2 \\
&d.\pi = a \\
&Q = \{c, e, f, d\} \\
&a.color = red
\end{aligned}
$$

- 3.

$$
\begin{aligned}
&u = c \\
&Adj[u] = \{b, f\} \Rightarrow \\
&f.color = yellow \\
&f.d = c.d + 1 = 2 \\
&f.\pi = c \\
&Q = \{e, f, d\} \\
&c.color = red
\end{aligned}
$$

Пример

- 4.

$$u = e$$
$$Adj[u] = \{a, b, f\} \Rightarrow$$
$$Q = \{f, d\}$$
$$e.color = red$$

- 5.

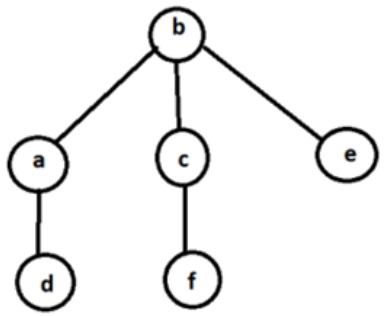
$$u = f$$
$$Adj[u] = \{b, c, e\} \Rightarrow$$
$$Q = \{d\}$$
$$f.color = red$$

Пример

На каждом шаге работы алгоритма к построенному дереву добавляется вершина, в результате работы алгоритма строится дерево поиска:

$$B = \begin{pmatrix} & a & b & c & d & e & f \\ a & 0 & 1 & 0 & 1 & 0 & 0 \\ b & 1 & 0 & 1 & 0 & 1 & 0 \\ c & 0 & 1 & 0 & 0 & 0 & 1 \\ d & 1 & 0 & 0 & 0 & 0 & 0 \\ e & 0 & 1 & 0 & 0 & 0 & 0 \\ f & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}; List : \begin{pmatrix} a \to b, d \\ b \to a, c, e \\ c \to b \\ d \to a \\ e \to b \\ f \to c \end{pmatrix};$$

$$\begin{pmatrix} & a & b & c & d & e & f \\ v.d & 1 & 0 & 1 & 2 & 1 & 2 \\ \Pr ev(v) & b & \emptyset & b & a & b & c \end{pmatrix}$$



Анализ алгоритма

Покажем, что построенное дерево обладает свойством: для любого $v \in V$ величина $v.d = \delta(s, v)$, где $v.d = \delta(s, v)$ – длина кратчайшего из s в v пути. Под длиной пути из s в v понимается количество ребер при переходе.

Лемма 1

Для очереди $Q = \{v_i, i = 1, 2, ..., r\}$, полученной на любом шаге алгоритма, справедливо

$$\begin{cases} v_r.d \leq v_1.d + 1, \\ v_i.d \leq v_{i+1}.d, i = 1, 2, ..., r - 1 \end{cases}$$

Доказательство. Пусть $V = \{v_1, v_2, ..., v_n\}$. Рассуждения по индукции. Если очередь получена из вершин первого списка смежности, то после исключения вершины $v_1 = s$ из очереди все вершины, смежные с ней, получают значение $v_1.d = 1$ и далее в роли вершины u выступает следующая по списку вершина. Предположим, после изменения атрибутов всех вершин, смежных с u, имеется очередь, первая вершина исключается и рассматриваются атрибуты смежных с ней по горизонтальному списку.

Анализ алгоритма

Продолжение доказательства Леммы. По индуктивному предположению те вершины v_1, которые не изменили атрибуты, поскольку были в очереди ранее, удовлетворяют неравенствам $u.d + 1 \leq v_i.d \leq v_{i+1}.d$, новые вершины, т.е. смежные с u и меняющие атрибуты, получают значение $v_i.d = u.d + 1$ и добавляются в конец очереди с атрибутом $v_i.d = u.d + 1$. Это значит, что новые добавленные вершины соответствуют неравенствам леммы, что завершает индуктивный переход. Лемма доказана.

Замечание. Если вершина v_i вносится в очередь до вершины v_j, то $v_i.d \leq v_j.d$.

Анализ алгоритма

Теорема 1

Для любой начальной вершины в результате работы алгоритма справедливо равенство $v.d = \delta(s, v), v \in V$.

Доказательство. Предположим противное: для некоторой вершины v значение атрибута d не равно кратчайшему пути от s к ней. Пусть выбранная вершина такова, что для нее величина $\delta(s, v)$ минимальна среди тех вершин, для которых значение d больше $\delta(s, v)$. Пусть u – вершина, непосредственно предшествующая v на кратчайшем от s до v пути, т.е. $\delta(s, v) = \delta(s, u) + 1$. Это в частности означает, что v находится в списке смежности u. По определению v величина $\delta(s, u) = u.d$ и кроме того $v.d > \delta(s, u) + 1 = u.d + 1$. В момент удаления вершины u из очереди вершина v может быть одного из трех цветов.

Анализ алгоритма

Продолжение доказательства Теоремы. Если по удалении u вершина v зеленая, то сразу после этого полагается $v.d = u.d + 1$, что противоречит предположению. Если вершина v на момент удаления u красная, то это противоречит замечанию к лемме 1. Если вершина v на момент удаления u желтая, то она приобрела такой цвет до момента удаления u, значит $v.d = w.d + 1$, где w – вершина, удаленная из очереди до удаления u, откуда $v.d = w.d + 1 \leq u.d + 1$, что тоже невозможно. Таким образом, противоречия, полученные во всех случаях, доказывают равенство $v.d = \delta(s, v), v \in V$. Аналогично показывается, что процедура алгоритма проходит все вершины графа, в противном случае для некоторой вершины v будет $\infty = v.d > \delta(s, v)$. Теорема доказана.

Замечание. Кратчайший путь из s в v, построенный алгоритмом, удовлетворяет равенству $v.d = (v.\pi).d + 1$.

Деревья поиска в ширину

В процессе работы алгоритма строится граф предшествования $G_\pi = (V_\pi, E_\pi)$, где $V_\pi = \{v \in V | v.\pi \neq \emptyset\} \cup \{s\}, E_\pi = \{(v.\pi, v) | v \in V \backslash \{s\}\}$.

Теорема 2

Процедура алгоритма поиска в ширину строит подграф предшествования, который является деревом поиска в ширину.

Доказательство. Необходимо показать, что для любой вершины $v \in V$ путь из s в v является единственным простым путем. Это действительно так, поскольку по построению графа предшествования алгоритмом поиска в ширину имеется равенство $|E_\pi| = |V_\pi| - 1$ и указанный граф связен. Теорема доказана.

Сложность алгоритма. При разумной организации метода время исполнения $O(n + m)$.

Поиск в глубину

Общие соображения: стратегия обхода в глубину состоит в том, чтобы исследовать все ребра, выходящие из вершины, открытой последней и произвести возврат к вершине, когда не остается неисследованных ребер. В момент открытия вершины v, т.е. когда ее цвет становится желтым, атрибут $v.\pi$ устанавливается равным u. Подграф предшествования может состоять из нескольких деревьев, т.е. быть лесом. В отличие от поиска в ширину у каждой вершины имеется атрибут $v.f$ в паре с $v.d$. При этом атрибут $v.f$ означает время завершения работы с v. В отличие от поиска в ширину поиск в глубину работает не с очередью, а со стеком S. Пусть s – корень, т.е. начальная вершина поиска, сразу помещается в стек, полагается $u = s$ – вершина стека. Далее возможны случаи:

1. Среди вершин, смежных с u, есть зеленая вершина v. Тогда v записывается в стек, окрашивается в желтый цвет.
2. Все вершины, смежные с u, желтые. Вершина u окрашивается красным и удаляется из стека.
3. Стек пуст. Вершины все пройдены.

Алгоритм поиска в глубину

Для каждой вершины $u \in V$: $u.color = white; u.\pi = \emptyset; time = 0$. $S = \{s\}$

Для каждой вершины $u \in V$:
$if\ u.color = white \Rightarrow time = time + 1; u.d = time; u.color = yellow$.
$S = S \cup \{u\}$. Очередность перебора вершин u в соответствии со стеком S.

Для каждой $v \in Adj[u]$: $if\ v.color = white \Rightarrow v.pi = u; time =$
$time + 1; v.d = time; v.color = yellow; S = S \cup \{u\}$. Если белых вершин из u не просматривается, то
$S = S \backslash \{u\}; u.color = red; time = time + 1; u, f = time$ (Завершение работы с u)

Просмотр последнего элемента в списке S в качестве u. Число просмотров зеленых вершин $O(E)$, следовательно время работы алгоритма $O(|E| + |V|)$.

Свойства поиска в глубину

Вершина окрашивается в желтый цвет один раз, в момент включения ребра в дерево поиска. Поскольку в случае простого связного графа число таких окрашиваний на 1 меньше вершин, то полученный граф – дерево.

Теорема 3

Для любых вершин u, v графа G справедливо одно и только одно из следующих свойств:

А. Отрезки $[u.d, u.f]$ и $[v.d, v.f]$ не пересекаются. u не является потомком v в построенном лесу поиска и наоборот.

Б. Отрезок $[u.d, u.f]$ содержится в отрезке $[v.d, v.f]$ и u – потомок v в лесу поиска.

В. Отрезок $[v.d, v.f]$ содержится в отрезке $[u.d, u.f]$ и v – потомок u в лесу поиска.

Доказательство. Если $u.d < v.d$, то вершина u была желтой в то время, когда v была зеленой. При этом условие $u.f > v.f$ означает, что вершина v была закрашена в красный и удалена из стека, т.е. v – потомок u. Случай $v.f < u.d$ означает, что вершина v была окрашена в красный, когда u была еще зеленой. Следовательно, в дереве поиска эти вершины на являются потомками друг друга. Теорема доказана

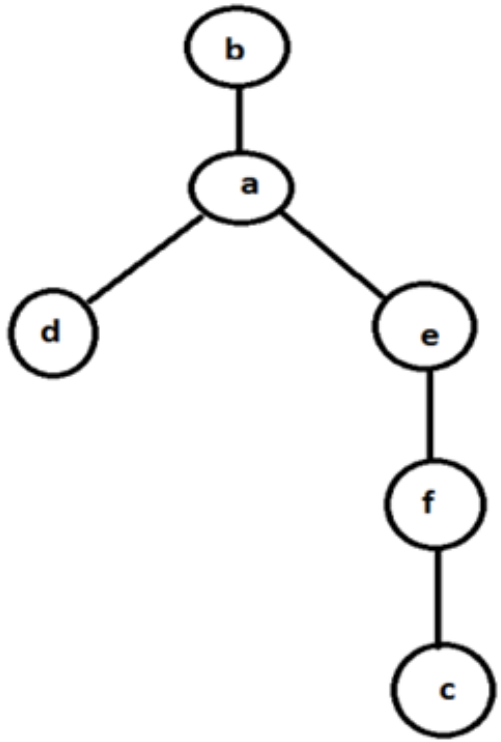
Пример

Рассмотрим предыдущий пример.

1. $S = \{b\}, b.d = 1$, b – желтая все другие вершины зеленые.
2. $S = \{b, a\}, a.d = 2$.
3. $S = \{b, a, d\}, d.d = 3$.
4. $S = \{b, a\}, d.color = red, d.f = 4$.
5. $S = \{b, a, e\}, e.d = 5$.
6. $S = \{b, a, e, f\}, f.d = 6$.
7. $S = \{b, a, e, f, c\}, c.d = 7$.
8. $S = \{b, a, e, f\}, c.color = red, c.f = 8$.
9. $S = \{b, a, e\}, f.color = red, f.f = 9$.
10. $S = \{b, a\}, e.color = red, e.f = 10$.
11. $S = \{b\}, a.color = red, a.f = 11$.
12. S – пуст

Пример

Дерево поиска:



Литература


Т.Кормен, Ч.Лейсерзон, Р. Риверст, К. Штайн. Алгоритмы. Построение и анализ. М., ООО «И.Д.Вильямс», 2013


Скиена С. Алгоритмы. Руководство по разработке. 2-е изд. Санкт-Петербург, «БХВ-Петербург», 2014