1　81 ページ目翻訳

定理 3.10 T を接続グラフ G の深さ優先探索木とし、u を T の根ではない頂点とする。u は、$l(v) \leq dfi(u)$ のような子 v が T にある場合に限り、G のカット頂点になります。

証拠。まず、u が T のルートではなく、u が G のカット頂点であると仮定します。r (u) を T のルートとします。 この場合、r は u で G のブランチ B_1 に属します。u はカット頂点であるため、未訪問の頂点がまだある間に、深さ優先検索は最終的に u に到達する必要があります。v を、B_1 に属さない u に隣接する最初の頂点とする。 v がブランチ B_2 に属しているとします。u と異なる B_2 の頂点は $G - V(B_2)$ の頂点に結合されていないため、 $l(v) \leq dfi(u)$ となり、uv は $l(v) \leq dfi(u)$ を持つ T の端。 r を T のルートとし、S_2 を v をルートとする T の最大部分木の頂点の集合を表すものとします。定理 3.8 より、S_2 の頂点と $G - (S_1 \cup S_2 \cup \{u\}$ の頂点を結ぶ辺は存在しません。

G にエッジ x_1x_2 が含まれているとします $(x_i \in S_i(i = 1, 2))$。 その場合、x_2x_1 はバックエッジ、$dfi(x_1)dfi(u)$ でなければなりません。 T の $v - x_2$ パスに続いてエッジ x_2x_1 を取ると、$l(v) \leq dfi(x_1)$ であることがわかります。 したがって、$l(v) < dfi(u)$ となり、矛盾します。したがって、S_1 の頂点と S_2 の頂点を接続する G 内のすべてのパスは u を通過します。これは、u が G のカット頂点であることを意味します。

グラフにブリッジがあるかどうかを知ることが重要な場合があります。第 11 章では、そのような状況に遭遇します。 深さ優先探索を利用して、橋の特徴を表示することもできます。エッジ e がグラフ G のブリッジである場合、e は、e が G のすべての深さ優先探索フォレストに属することに注目してください。カット頂点の場合と同様に、接続されたグラフのブリッジを特徴付けるだけで済みます。

2　82 ページ目翻訳

定理 3.11。T を接続グラフ G の深さ優先探索木とする。$dfi(u) < dfi(v)$ である G のエッジ uv は、uv が T と $l(v)$ のエッジである場合に限り、G のブリッジになります。) $> dfi(u)$。

証拠。まず、uv が $dfi(u) < dfi(v)$ である G のブリッジであると仮定します。前に述べたように、uv は T のエッジであり、v は u の子です。 $l(v) \leq dfi(u)$ の場合、定理 3.8 より、v または v の子孫から u または u の祖先までのバックエッジが存在します。したがって、uv は G のブリッジではないため、v または v の子孫から u または u の親へのバックエッジが存在する必要があります。しかし、そうすると $l(v) \leq dfi(u)$ となり、仮説に反します。

グラフのブロックを知ることが重要な状況があります。たとえば、第 9 章では、グラフのブロックにのみ適用できるアルゴリズムが示されています。次に、深さ優先検索を使用してグラフのブロックを見つけるアルゴリズムを開発する方法について説明します。グラフ G の深さ優先検索が実行されると、当然のことながら、検索は G 内のコンポーネントからコンポーネントへと進みます。G が接続されているとします。G に少なくとも 2 つのブロックがある場合、G には少なくとも 2 つのエンドブロック (G のカット頂点を 1 つだけ含むブロック) があります。

G のカット頂点ではない頂点 r から検索が始まるとします。r がエンドブロックに属するかどうかに関係なく、検索は最終的にエンドブロックに進むかどうかにかかわらず、検索は最終的にエッジ uv で初めてエンドブロック B に進みます。 ここで、u は、G 内の B の一意のカット頂点です。定理 3.10 より、u がカット頂点であることが確認できます。

B に入ってから完全に探索されるまで、B に留まります。したがって、定理 3.9 より、v は B における u の唯一の子です。つまり、B のすべての頂点は v の子孫です。 このため、スタックを使用します。B の頂点は、最初にアクセスされた順序でスタックに配置されます。uv をバックトラックしたときに、u がカット頂点であることがわかった場合は、スタックの最上位から v までの頂点を出力します。 これらの頂点は u とともに、G から B を誘導します。プロセスは u で続行され、検出するブロックが 1 つ減りました。実際、アルゴリズムは、G のすべてのブロックが決定されるまで、エンドブロックを連続的に検出します。

検索をカット頂点 r から開始する必要がある場合、アルゴリズムは r を含まないすべてのブロックを検出し、以前と同様に出力します。r を含むブロックは、定理 3.9 を使用して見つけることができます。r まで後戻りすると、たとえばエッジに沿って rx が決定されます。このブロックは、r とともに x までのスタック上の頂点によって誘発されるブロックです。

3　85 ページ目翻訳

ステップ 6 では、最大でもその頂点に付随するすべての辺について頂点の最低点の値が更新されるため、ステップ 6 が実行される合計回数は $\sum_{v \in V(G)} \deg v = 2q$ に比例します。したがって、ステップ 6 の計算量は $O(q)$ になります。 ステップ 7 の計算量は $O(p)$ です。そして、ステップ 8 〜 11 は各頂点から最大 1 回バックトラックするため、これらのステップの計算量は $O(p)$ になります。ステップ 12 では各頂点を最大 1 回スキャンします。これらのステップの計算量は $O(p)$ です。ステップ 12 では、各隣接辺を根で最大 1 回スキャンします。根に隣接するスキャンされていない辺がない場合、頂点の割り当てが行われます。したがって、ステップ 12 の計算量は $O(max\{p, q\})$ になります。結論として、アルゴリズム 3.2 の計算量は $O(max\{p, q\})$ になります。

3.5 幅優先探索

次のセクションと第 4 章で説明するように、幅優先探索 (通常は bfs と略されます) は、多くのグラフアルゴリズムで役立つもう 1 つのツールです。幅優先探索は、ある頂点から開始して、グラフまたは有向グラフの頂点を体系的に訪問します。G の r (根とも呼ばれます)。 根は最初のアクティブな頂点です。検索中のどの段階でも、現在アクティブな頂点に隣接するすべての頂点がまだ訪問されていない頂点を探すためにスキャンされます。つまり、未訪問の頂点に対して「広範な」検索が実行されます。頂点に初めてアクセスするたびに、(達成する目標に応じた何らかのルールに従って) ラベルが付けられ、キューの最後尾に追加されます。 この検索では、スタックではなくキューが使用されることに注意してください。現在アクティブな頂点は、キューの先頭にある頂点です。

近隣が訪問されるとすぐに削除され、キューから削除されます。キューが空で、グラフまたは有向グラフのいくつかの頂点がまだ訪問されていない場合は、未訪問の頂点を選択し、ラベルを割り当ててキューに追加します。 グラフのすべての頂点を訪問すると、検索は完了します。

ここで、BFS 中のグラフの頂点のラベル付けの 1 種類について説明します。G がその隣接リストによって表されるグラフであると仮定します。最初は、G のすべての頂点は、その隣接リストによって表されるグラフです。 最初に、G のすべての頂点には 0 のラベルが付けられます。まず、r にラベル 1 を割り当て、r をキュー Q に配置します。