Giải Thuật Lập Trình

Nơi tổng hợp và chia sẻ những kiến thức liên quan tới giải thuật nói chung và lý thuyết khoa học máy tính nói riêng.

overview

You are currently browsing articles tagged **overview**.

Tổng quan về cây khung nhỏ nhất.

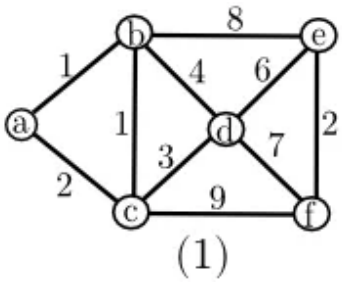
June 17, 2016 in Uncategorized | No comments

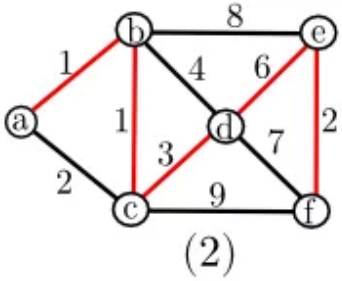
Bài này mình đưa ra một cách nhìn tổng quan về bài toán cây khung nhỏ nhất và một số comments của mình trên những tài liệu tham khảo liên quan. Một phần của bài viết này trước đây mình viết trên trang facebook, sau đó được viết lại một cách hình thức hơn trong bài thuật toán Kruskal và bây giờ thì mình viết thành một bài riêng biệt.

Bài toán

Bài toán cây khung nhỏ nhất (minimum spanning tree):
Cho đồ thị vô hướng, liên thông $G(V, E)$ trong đó mỗi cạnh $e \in E$ được gán một trọng số (weight) $w(e) \in \mathbb{R}$. Một **cây khung** (spanning tree) là một đồ thị con của G, không có chu trình, và chứa mọi đỉnh của G. Trọng số của một cây khung là tổng trọng số các cạnh của cây khung đó. Tìm cây khung có trọng số nhỏ nhất của $G(V, E)$.

Trong bài toán cây khung nhỏ nhất, ta chỉ xét đồ thị vô hướng. Do đó, từ giờ trở đi, chúng ta sẽ mặc định đồ thị đầu vào là vô hướng. Một ví dụ đồ thị (hình (1)) với trọng số và một cây khung nhỏ nhất (hình (2)) với trọng số 13.





Cây khung và matroid

Trong bài lý thuyết matroid (http://www.giaithuatlaptrinh.com/?p=201) mà mình đã viết trước đây, bạn đọc đã có cơ hội làm quen với **Graphic**

Matroid và cũng đã thấy cây khung có liên quan chặt chẽ với matroid này. Cụ thể, cơ sở của Graphic Matroid của một đồ thị chính là cây khung của đồ thị đó. Từ lý thuyết matroid, ta biết bài toán tìm cơ sở của matroid có trọng số lớn nhất (nhỏ nhất) có thể tìm được bằng giải thuật tham lam. Do đó, cũng không có gì lạ nếu ta có thể tìm được cây khung nhỏ nhất sử dụng thuật toán tham lam. Thực tế, thuật toán trong bài lý thuyết matroid có thể được sửa đổi (chi tiết coi như bài tập cho bạn đọc) để thu được thuật toán tìm cây khung nhỏ nhất trong thời gian $O(V^2)$.

Thuật toán Kruskal mà ta sẽ tìm hiểu sau đây chính là thuật toán tham lam trong matroid. Dưới góc độ đồ thị, ta có thể thực thi thuật toán Kruskal trong thời gian $O((V+E)\log V)$. Trước khi đi vào chi tiết tìm hiểu các thuật toán cho bài toán cây khung nhỏ nhất, chúng ta sẽ tìm hiểu một số tính chất của cây khung nhỏ nhất trước.

Một số tính chất của cây khung

Sau đây là một số tính chất chung của một cây khung (không nhất thiết là nhỏ nhất):

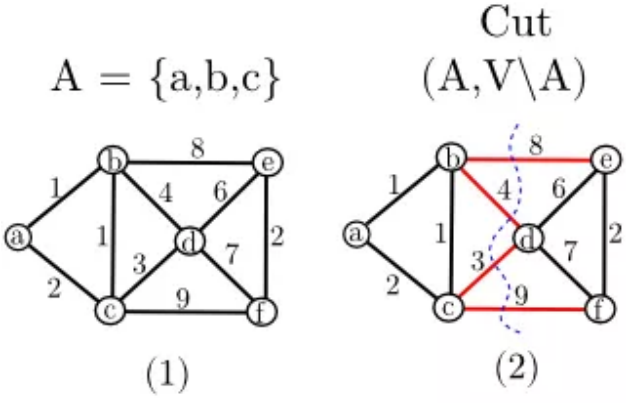
Gọi T là một cây khung của đồ thị (liên thông) $G(V,E)$ với n đỉnh và m cạnh.

1. T có đúng $n-1$ cạnh.
2. Với mỗi cặp đỉnh u,v, tồn tại một và chỉ một đường đi nối hai đỉnh u,v trong T.
3. Thêm bất kì một cạnh nào vào T sẽ tạo ra một (và chỉ một) chu trình.

Chứng minh các tính chất trên không khó và coi như bài tập cho bạn đọc. Chứng minh tính chất (1) ta có thể dùng quy nạp. Tính chất (2) dựa vào tính chất liên thông của đồ thị và (3) dựa vào tính chất T là một cây.

Lát cắt (Cut): Gọi A là một tập đỉnh của đồ thị. Lát cắt tương ứng với A, kí hiệu là $(A, V \setminus A)$, là tập **tất cả** các cạnh của G có đúng một đầu mút (incident vertex) nằm trong A (đầu mút còn lại hiển nhiên là nằm trong $V \setminus A$).

Ví dụ lát cắt tương ứng với $\{a,b,c\}$ của đồ thị trong hình (a) là các cạnh màu đỏ trong hình (b).



Một số tính chất sau của cây khung **nhỏ nhất** sẽ rất hữu ích trong thiết kế thuật toán.

4. **Tính Chu Trình (Cycle Property):** Cạnh có trọng số lớn hơn hoàn toàn (strictly larger) các cạnh khác trong cùng một chu trình (nào đó) của đồ thị **không** nằm trong bất kì cây khung nhỏ nhất nào.
5. **Tính Cắt (Cut Property):** Cạnh có trọng số nhỏ hơn hoàn toàn (strictly smaller) các cạnh khác trong cùng một lát cắt (nào đó) của đồ thị G nằm trong **mọi** cây khung nhỏ nhất.
6. Nếu có duy nhất một cạnh có trọng số nhỏ nhất thì bất kì cây khung nhỏ nhất nào cũng phải chứa cạnh này.

Phương pháp chứng minh các tính chất trên đều giống nhau, sử dụng phương pháp lập luận tráo đổi (exchange argument). Mình sẽ chứng minh tính chất (4) và (5) để minh họa cho lập luận này. Chứng minh (6) tương tự và coi như bài tập cho bạn đọc.

Chứng minh: (4) Giả sử cạnh e lớn hơn (hoàn toàn) các cạnh khác trong cùng một chu trình C và giả sử một cây khung nhỏ nhất T chứa e. Xóa e khỏi T, ta sẽ thu được hai cây F_1, F_2. Do C là một chu trình, tồn tại một cạnh khác e, gọi là f, của T nối F_1 với F_2. Do đó $T \cup \{f\} - \{e\}$ là một cây khung của G có trọng số nhỏ hơn T (do $w(f) < w(e)$), trái với giả thiết T là một cây khung nhỏ nhất.

(5) Giả sử cạnh e nhỏ hơn (hoàn toàn) các cạnh khác trong cùng một lát cắt L và giả sử tồn tại một cây khung nhỏ nhất T **không** chứa e. Thêm e vào T, ta sẽ thu được một chu trình C chứa e. Do L là lát cắt, C phải chứa ít nhất một cạnh f khác e của L. Do đó, $T \cup \{e\} - \{f\}$ là một cây khung của G có trọng số nhỏ hơn T (do $w(e) < w(f)$), trái với giả thiết T là một cây khung nhỏ nhất.

Các thuật toán

Có rất nhiều thuật toán đã được phát triển cho bài toán cây khung nhỏ nhất. Mình sẽ viết 4 bài về 4 thuật toán khác nhau. Mỗi thuật toán có điểm mạnh và yếu riêng. Phần đọc thêm dưới đây mình sẽ giới thiệu thêm một số thuật toán khác, và liên kết tới nguồn tham khảo để bạn đọc dễ tìm.

Thuật toán Kruskal

Như đã nói ở phần matroid, thuật toán Kruskal [1] là một thuật toán tham lam. Thuật toán này khá dễ hiêu và chứng minh. Thuật toán này tìm cây khung nhỏ nhất dựa trên tính chất chu trình: liên tục thêm cạnh theo thứ tự tăng dần của trọng số vào cây khung nếu cạnh đó không tạo ra chu tình với cây khung hiện tại.

```
KRUSKAL(G(V, E), w):
    T ← V                    ≪ there is no edge in T ≫
    sort edges in E in ↑ order of weight
    let e_1, ..., e_m be edges after sorting
    for i ← 1 to m
        if T ∪ {e_i} has no cycle
```

$T \leftarrow T \cup \{e_i\}$
return T.

Tuy nhiên, để thực hiện hiệu quả với thời gian $O((V+E)\log V)$, ta phải sử dụng đến cấu trúc Union-Find (http://(http://www.giaithuatlaptrinh.com/?p=218) . Chi tiết xem thêm tại đây (http://www.giaithuatlaptrinh.com/?p=764) .

Thuật toán Prim

Thuật toán Prim [2] tìm cây khung nhỏ nhất dựa trên tính cắt (tính chất 5) của cây khung. Thuật toán duy trì một tập đỉnh A và một cây khung T cho tập đỉnh này. Ban đầu, A chỉ có một đỉnh (bất kì) của đồ thị và T không chứa cạnh nào. Mỗi bước, thuật toán tìm ra đỉnh u không thuộc A sao cho cạnh kề với u là cạnh có trọng số nhỏ nhất trong số các cạnh trong lắt cắt $(A, V \setminus A)$ và thêm u vào A. Trong giả mã dưới đây, ta sử dụng T để biểu diễn cả tập A và cây khung của tập A này.

PRIM$(G(V,E), w)$:
pick any vertex u
add the smallest edge of the cut $(\{u\}, V \setminus \{u\})$ to T
while $T \neq V$ $\ll T$ does not contain all vertices$\gg$
$e \leftarrow$ the smallest edge of the cut $(T, V \setminus T)$
$T \leftarrow T \cup \{e\}$
return T

Thuật toán PRIM có thể dễ dàng thực thi với thời gian $O(V^2)$, và nếu kết hợp với Heap nhị phân (http://(http://www.giaithuatlaptrinh.com/?p=736) , ta có thể giảm xuống $O((E+V)\log V)$. Với những bạn đã quen với thuật toán Dijkstra sử dụng Heap (http://www.giaithuatlaptrinh.com/?p=764) thì có lẽ thực thi thuật toán này sẽ dễ hơn với bạn. Chi tiết xem thêm tại đây (http://www.giaithuatlaptrinh.com/?p=1175) .

Thuật toán Borůvka

Thuật toán Borůvka tìm cây khung nhỏ nhất dựa trên phép co cạnh (edge contraction). Co một cạnh được hiểu là phép gộp hai đỉnh kề với cạnh đó lại với nhau thành một đỉnh mới, và xóa đi các cạnh lặp (loop) và các cạnh song song. Phép co cạnh là một phép rất mạnh và được sử dụng nhiều trong lý thuyết đồ thị. Tuy nhiên, trong máy tính, việc thực thi phép co cạnh lại không hề đơn giản.

Đặc biệt của thuật toán này đó là nó có một số tính chất cự kì đẹp. Với đồ thị thưa, $E = O(V)$, thuật toán này có thời gian tuyến tính. Với đồ thị dày, thuật toán có thời gian $O((V+E)\log V)$. Các thuật toán tốt nhất bây giờ cho bài toán cây khung cũng xuất phát từ sửa đổi thuật toán Boruvka. Chi tiết xem thêm tại đây (http://www.giaithuatlaptrinh.com/?p=1204) .

Thuật toán Karger-Klein-Tarjan

Thuật toán Karger-Klein-Tarjan (KKT)[5] là thuật toán ngẫu nhiên, có thời gian **tuyến tính** $O(V+E)$, trả lại cây khung nhỏ nhất với xác suất ít nhất $1-\frac{1}{m}$ trong đó m là số cạnh của đồ thị (xác suất cao). Thuật toán này là sự kết hợp hoàn hảo giữa phương pháp lấy mẫu và thuật toán Boruvka. Cho đến này thì chưa có thuật toán tất định (deterministic) nào có thời gian tuyến tính. Thuật toán tất định nhanh nhất được phát triển bởi Bernard Chazelle (http://www.cs.princeton.edu/~chazelle/) [6] với thời gian $O(E\alpha(V,E))$ ($\alpha(.)$ là hàm Ackerman (https://en.wikipedia.org/wiki/Ackermann_function) ngược.).

Câu hỏi tồn tại hay không một thuật toán tuyến tính tìm cây khung nhỏ nhất vẫn là một câu hỏi mở cho đến nay. Thuật toán tuyến tính KKT cũng như một lời gợi ý cho chúng ta biết thuật toán như vậy có thể tồn tại. Xem tại **sẽ link sau khi viết**

Đọc thêm

Tutorial (https://www.cs.jhu.edu/~jason/papers/eisner.mst-tutorial.pdf) của Jason Eisner, 1997, về bài toán cây khung nhỏ nhất khá đầy đủ. Ngoài bài toán tìm cây khung nhỏ nhất, bạn đọc còn có thể tìm được những bài toán liên quan như kiểm tra xem một cây khung có phải là nhỏ nhất hay không, và phiên bản động (dynamic) của cây khung tại đó. Cách viết rất dễ hiểu.

Paper của Karger, Klein và Tarjan [5] viết về thuật toán KKT cũng cực kì dễ đọc, dễ hiểu. Mô tả thuật toán khá ngắn gọn. Chỉ có phân tích là hơi phức tạp hơn một chút.

Về lecture note, mình thấy lecture note (http://www.cs.tau.ac.il/~zwick/grad-algo-13/mst.pdf) của Uri Zwick khá dễ hiểu. Phần bài tập của note này có nhiều bài tập hay, trong đó có một số tính chất khá thú vị của thuật toán Borůvka mà mình đã nói ở trên.

Về bài toán cây khung nhỏ nhất cho đồ thị có hướng, mình khuyến khích bạn đọc tham khảo lecture note (http://www.cs.tau.ac.il/~zwick/grad-algo-13/directed-mst.pdf) của Uri Zwick.

Nếu bạn nào nghiên cứu sâu về bài toán cây khung, thuật toán Chazelle [6] cũng là bài báo phải đọc. Thuật toán sử dụng cấu trúc dữ liệu Soft Heap, và mô tả của cấu trúc này trong [6] khá phức tạp. Kaplan và Zwick đã đơn giản hóa cấu trúc dữ liệu này khá nhiều, và bạn đọc có thể tìm thấy tại đây (http://epubs.siam.org/doi/pdf/10.1137/1.9781611973068.53) .

Bạn nào biết các paper hoặc bài viết nào hay về cây khung, xin để lại comment bên dưới để mình cập nhật vào bài viết.

Tham khảo

[1] Kruskal, Joseph B. *On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem.* Proceedings of the American Mathematical society 7.1 (1956): 48-50.
[2] Prim, Robert Clay. *Shortest Connection Networks and some Generalizations.* Bell System Technical Journal 36.6 (1957): 1389-1401.
[3] Borůvka, Otakar (1926). *Příspěvek k řešení otázky ekonomické stavby elektrovodních sítí (Contribution to the Solution of a Problem of Economical Construction of Electrical Networks)*. Elektronický Obzor (in Czech) 15: 153–154.

[4] Nešetřil, Jaroslav, Eva Milková, and Helena Nešetřilová. *Otakar Borůvka on minimum spanning tree problem translation of both the 1926 papers, comments, history.* Discrete Mathematics 233.1 (2001): 3-36.
[5] Karger, David R.; Klein, Philip N.; Tarjan, Robert E. *A Randomized Linear-time Algorithm to Find Minimum Spanning Trees*. Journal of the ACM 42 (2): 321–328, 1995.
[6] Chazelle, Bernard. *A Minimum Spanning Tree Algorithm with Inverse-Ackermann Type Complexity.* Journal of the ACM (JACM) 47.6 (2000): 1028-1047.

Tags: minimum-spanning-tree, overview, soft-heap