**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.*

‹ Tổng quan về cây khung nhỏ nhất. • Giải thuật cho hệ thống gợi ý với dữ liệu lớn -- A Recommendation Algorithm for Big Data ›

# Cây khung nhỏ nhất: thuật toán KKT -- KKT Algorithm

July 8, 2016 in Uncategorized | No comments

Thuật toán KKT, là thuật toán ngẫu nhiên tìm cây khung nhỏ nhất với xác suất cao trong thời gian tuyến tính được phát triển bởi Klein, Karger và Tarjan[1]. Ý tưởng của thuật toán này rất đẹp, và hi vọng bài này sẽ truyền tải được cái đẹp đó đến các bạn.

**Remarks:** Nội dung kiến thức của bài viết này có thể coi ở mức độ cao cấp, do đó nhiều chi tiết đơn giản mình sẽ bỏ qua. Bạn đọc nếu chưa từng đọc các bài viết trước về cây khung nhỏ nhất thì xin xem lại các bài viết đó tại đây (http://www.giaithuatlaptrinh.com/?page_id=4) trước khi đọc tiếp.

**Remark:** Trong bài viết này, mình sẽ giải sử trọng số của các cạnh là duy nhất, i.e, không có hai cạnh nào có trọng số giống nhau. Trong bài thuật toán Borůvka (http://www.giaithuatlaptrinh.com/?p=1204), mình đã đưa ra cách để thực hiện giải thuyết này với đồ thị đầu vào bất kì trong thời gian tuyến tính. Mình khuyến khích bạn đọc xem lại. Với giả sử này, **cây khung nhỏ nhất của đồ thị là duy nhất**.

## Phân phối hình học

Giả sử bạn tung một đồng xu với xác suất mặt ngửa là $p$ (do đó xác suất mặt sấp là $1-p$). Bạn muốn tung đồng xu đó đến khi nào bạn thu được một mặt ngửa. Gọi $X$ là số lần tung cần thiết. Khi đó, ta có phân phối:

$$\Pr[X=k] = (1-p)^{k-1}p \qquad (1)$$

Phân phối của $X$ được gọi là phân phối hình học (geometric distribution (https://en.wikipedia.org/wiki/Geometric_distribution)). Kì vọng của phân phối này là:

$$E[X] = \frac{1}{p} \qquad (2)$$

Tính chất này mình không chứng minh mà gợi ý bạn đọc xem tại wikipedia (https://en.wikipedia.org/wiki/Geometric_distribution). Từ đó ta có hệ quả:

**Corollary 1:** Kì vọng số lần tung đồng xu để thu được $n$ mặt ngửa là $\frac{n}{p}$.

## Xác nhận cây khung nhỏ nhất

**Problem 1:** Cho một đồ thị và một cây khung $T$, thiết kế giải thuật để xác nhận xem $T$ có phải là cây khung nhỏ nhất hay không?

Bài toán trên gọi là bài toán xác nhận cây khung nhỏ nhất (MST Verification). Khác với bài toán cây khung nhỏ nhất, bài toán này có thể được giải quyết trong thời gian tuyến tính $O(V+E)$. Giải thuật không đơn giản và ý tưởng cũng không quá đẹp. Xem [2,3] để biết thêm chi tiết. Sau đây mình sẽ đưa ra mấy tính chất cơ bản của cây khung mà các thuật toán xác nhận áp dụng.

Gọi $F$ là một rừng (forest) của đồ thị $G$. Với mỗi cặp đỉnh $(u, v)$ của $F$, gọi $w_F(u, v)$ là trọng số của cạnh (có trọng số) lớn nhất trên đường đi (duy nhất) từ $u$ đến $v$ trong $F$. Quy ước nếu $u, v$ thuộc hai cây khác nhau của $F$ thì $w_F(u, v) = +\infty$. Cạnh $uv$ được gọi là $F$**-heavy** nếu $w(uv) > w_F(u, v)$ và được gọi là $F$**-light** nếu ngược lại.

Ta có một số nhận xét:

1. Mọi cạnh của rừng $F$ là $F$-light.
2. Bất kì cạnh $F$-heavy nào đều không nằm trong cây khung nhỏ nhất của đồ thị. Nhận xét này là hệ quả trực tiếp từ tính chu trình của cây khung.

Giải thuật của King[2] cho chúng ta Theorem sau:

> **Theorem 1:** Cho đồ thị có trọng số $G(V, E)$ và một rừng $F$. Tồn tại một giải thuật có thời gian $O(V + E)$ tìm tất cả các cạnh $F$-heavy của đồ thị.

Để xác nhận cây khung $T$ có phải là nhỏ nhất không, ta áp dụng Theorem 1 để tìm tất cả các cạnh $T$-heavy của đồ thị. Nếu tập trả về là rỗng, cây $T$ sẽ là cây khung nhỏ nhất, và ngược lại nếu tập trả về là khác rỗng (tại sao?). Do đó ta có:

> **Corollary 2:** Tồn tại một giải thuật có thời gian $O(V + E)$ xác nhận xem một cây khung cho trước của đồ thị có trọng số $G(V, E)$ có phải là cây khung nhỏ nhất hay không.

# Giải thuật KKT

Thuật toán KKT trực tiếp áp dụng thuật toán Borůvka và giải thuật xác nhận cây khung nhỏ nhất của King (Theorem 1). Chúng ta đã biết thuật toán Borůvka tìm cây khung nhỏ nhất bằng cách lặp lại 2 thao tác sau cho đến khi đồ thị chỉ còn một đỉnh:

1. Với mỗi đỉnh $u$ của đồ thị, tìm cạnh có trọng số nhỏ nhất trong số các cạnh có $u$ là đầu mút. Gọi cạnh đó là **cạnh an toàn** (safe-edge).
2. Co tất cả các cạnh an toàn tìm được ở bước 1.

Ta gọi 2 thao tác trên là một Borůvka-step. Sau mỗi Borůvka-step, ít nhất $n/2$ cạnh sẽ bị co, và do đó, số đỉnh của đồ thị giảm ít nhất một nửa. Từ đó suy ra số vòng lặp của thuật toán Borůvka là $O(\log V)$. Do đó, tổng thời gian của thuật toán là O((V+E)\log V) vì mỗi bước co có thể được thực hiện trong thời gian $O(V + E)$.

Giả sử đồ thị sau mỗi bước đều thưa ($E \leq cV$ với hằng số $c$ nào đó) thì thời gian của thuật toán Borůvka là:

$$\sum_{i=1}^{K} O((c+1)V/2^i) \leq \sum_{i=1}^{\infty} O((c+1)V/2^i) = O(2(c+1)V) = O(V) \qquad (3)$$

trong đó $K$ là tổng số vòng lặp của thuật toán Borůvka. Do đó, thời gian của thuật toán là tuyến tính. (Trong trường hợp đồ thị phẳng thì đồ thị sau mỗi bước của thuật toán Borůvka đều thưa, do đó, thuật toán là tuyến tính).

Tuy nhiên, sau một số bước co cạnh, đồ thị sẽ trở nên dày hơn. Giải quyết vấn đề này chính là cái hay của thuật toán KKT. Thuật toán KKT sẽ xóa đi một số cạnh dư thừa, i.e, các cạnh mà ta biết chắc chúng không nằm trong cây khung nhỏ nhất. Sau khi xóa, đồ thị lại trở nên thưa và do đó, tiếp tục áp dụng thuật toán Borůvka. Phân tích ở trên chỉ ra rằng, thời gian cuối cùng sẽ là tuyến tính. (Nice!).

Làm thế nào xóa được các cạnh dư thừa. Xem lại Theorem 1. Các cạnh $F$-heavy (của bất kì rừng $F$ nào đó) sẽ là dư thừa. Xóa các cạnh dưa thừa đó đi thì chỉ còn các cạnh $F$-light trong đồ thị. Thuật toán KKT sử dụng ngẫu nhiên để tìm ra rừng $F$ sao cho số cạnh $F$-light là nhỏ. Do đó, đồ thị sau khi xóa cạnh $F$-heavy là thưa. Chi tiết như sau:

Gọi $H$ là đồ thị thu được bằng cách lấy mẫu (sampling) mỗi cạnh của đồ thị với xác suất $p$. Hay nói cách khác, với mỗi cạnh của $G$, ta tung một đồng xu có xác suất mặt ngửa là $p$. Nếu đồng xu là ngửa thì ta thêm cạnh đó vào $H$, và không làm gì cả nếu đồng xu có mặt sấp. Gọi $F$ là rừng khung nhỏ nhất của $H$ (gọi là rừng khung vì $H$ có thể không liên thông). Ta có:

**Theorem 2:** Số cạnh $F$-light của đồ thị $G$ có kì vọng là $\frac{n}{p}$, trong đó $n$ là số đỉnh của đồ thị.

**Chứng minh:** Ta sẽ sử dụng thuật toán Kruskal để chứng minh (các bạn nên xem lại thuật toán này). Giả sử ta sắp xếp các cạnh của $G$ theo trọng số giảm dần. Khởi tạo $H$ và $F$ rỗng.

Với mỗi cạnh $e_i$ (theo thứ tự đã sắp xếp), ta tung một đồng xu (có mặt ngửa xác suất $p$). Nếu đồng xu có mặt ngửa thì ta thêm $e_i$ vào $H$. Nếu $e_i$ là $F$-light thì ta thêm vào $F$. Sau khi quá trình kết thúc, $F$ chính là rừng khung nhỏ nhất của $H$ (quá trình thêm cạnh vào $F$ chính là thuật toán Kruskal). Nên nhớ các cạnh $F$-heavy trong $H$ thì cũng là $F$-heavy trong $G$. Do đó, các cạnh này là dư thừa.

Ta giờ phân loại đồng xu thành 2 loại: loại 5k và loại 1k. Nếu cạnh $e_i$ là $F$-heavy trong $H$ thì đồng xu mà ta tung là loại 1k, và ngược lại là loại 5k. Loại đồng xu 1k sẽ không liên quan gì đến số cạnh $F$-light cuối cùng trong $G$ vì cho dù nó là mặt sấp hay ngửa thì cạnh đó vẫn là cạnh dư thừa. Ta chỉ cần đếm xem có bao nhiêu loại 5k ta đã tung. Nên nhớ rừng $F$ có tối đa $n-1$ cạnh. Theo Corollary 1, với kì vọng tung tối đa $\frac{n-1}{p}$ lần, ta sẽ thu được rừng $F$ (những cạnh với đồng xu 5k mà không thuộc $F$ sẽ là các cạnh $F$-light của $G$). Do đó, số cạnh $F$-light tối đa trong $G$ là $\frac{n-1}{p}$.

Tổng hợp tất cả những gì ta đã thảo luận, ta thu được giải thuật sau:

```
KKTSpanningTree(G(V,E), w(.)):
  if V = 1
    return G
  X ← BorůvkaStep(G, w(.))
  Y ← BorůvkaStep(G, w(.))
  H ← Sampling edges of G with p = 1/2
  F ← KKTSpanningTree(H, w(.))
  Z ← the set of F-heavy edges in G.
  G0 ← G \ Z        ≪ remove Z from G ≫
  T ← KKTSpanningTree(G0, w(.))
  return T ∪ X ∪ Y
```

```
BorůvkaStep(G(V,E), w(.)):
  X ← the set of safe edges
  G ← G/X        ≪ contract the set of edges X ≫
  return X
```

**Phân tích** Trong giải thuật KKT, chúng ta 2 lần gọi đệ quy, một lần trên $H$ và một lần trên $G_0$. Gọi $T(n,m)$ là thời gian tính toán của thuật toán trên đồ thị gốc $G$, với $n, m$ lần lượt là số đỉnh, cạnh của đồ thị. Chú ý do thuật toán là ngẫu nhiên, $T(n,m)$ là một biến ngẫu nhiên. Ta sẽ tính kì vọng $E[T(n,m)]$.

Nhận xét thấy: kì vọng số đỉnh và cạnh của đồ thị $H$ lần lượt là $n/4$ và $m/2$. Kì vọng số đỉnh của $G_0$ là $n/4$ và kì vọng số cạnh (theo Theorem 2) là $2n/4 = n/2$. Chú ý ở đây ta áp dụng 2 bước Borůvka nên số đỉnh giảm đi còn tối đa $n/4$ sau hai bước này.

Ngoại trừ 2 thủ tục đệ quy, các thao tác khác trong thuật toán KKT đều có thể tính được trong thời gian tuyến tính. Gọi thời gian đó là $a(n+m)$ với $a$ là một hằng số. Ta có:

$$E[T(n,m)] = E[T(n/4,m/2)] + E[T(n/4,n/2)] + a(n+m) \qquad (4)$$

Ta sẽ chứng minh $E[T(n,m)] \leq 2a(2n+m)$ bằng quy nạp. Thật vậy, từ $(4)$, ta có:

$$\begin{aligned} E[T(n,m)] &\leq a(n+m) + 2a(2n/4+m/2) + 2a(2n/4+n/2) \\ &= 4an + 2am = 2a(2n+m) \end{aligned}$$

Do đó, ta có:

**Theorem 3:** Thời gian kì vọng của thuật toán KKT là $O(V+E)$.

**Remarks:** Trong [1], các tác giả còn chứng minh được rằng thời gian chạy của thuật toán là $O(V+E)$ với xác suất ít nhất $O(1-\frac{1}{E})$. Mình khuyến khích bạn đọc tham khảo thêm.

# Tham khảo

[1] Karger, David R., Philip N. Klein, and Robert E. Tarjan. *A randomized linear-time algorithm to find minimum spanning trees.* Journal of the ACM (JACM) 42.2 (1995): 321-328.
[2] King, Valerie. *A simpler minimum spanning tree verification algorithm.* Algorithmica 18.2 (1997): 263-270.
[3] Hagerup, Torben. *An even simpler linear-time algorithm for verifying minimum spanning trees.* International Workshop on Graph-Theoretic Concepts in Computer Science. Springer Berlin Heidelberg, 2009. PDF (http://download.springer.com/static/pdf/643/chp%253A10.1007%252F978-3-642-11409-0_16.pdf?originUrl=http%3A%2F%2Flink.springer.com%2Fchapter%2F10.1007%2F978-3-642-11409-0_16&token2=exp=1468014166~acl=%2Fstatic%2Fpdf%2F643%2Fchp%25253A10.1007%252F978-3-642-11409-0_16.pdf%3ForiginUrl%3Dhttp%253A%252F%252Flink.springer.com%252Fchapter%252F10.1007%252F978-3-642-11409-0_16*~hmac=b3a6d00a107d0e344c865c2e646c2c08cd1127ef882cf0b04695e31ce21eb756)
.
[4] Uri Zwick. Lecture Notes on Minimum Spanning Tree (http://www.cs.tau.ac.il/~zwick/grad-algo-13/mst.pdf) . Tel Aviv University, 2013.

(http://www.giaithua... p=1266)
June 17, 2016
In "minimum-spanning-tree"

Algorithm (http://www.giaithua... p=1204)
June 1, 2016
In "boruvka-algorithm"

(http://www.giaithua... p=2422)
November 27, 2017
In "News"

**Tags:** boruvka-algorithm, KKT algorithm, linear time algorithm, minimum-spanning-tree, randomized algorithm

## No comments

Comments feed for this article

**Trackback link:** http://www.giaithuatlaptrinh.com/wp-trackback.php?p=1297

## Reply

Your email address will not be published. Required fields are marked *

Your comment

Name *

Email *

Website

Post Comment

Notify me of follow-up comments by email.

Notify me of new posts by email.