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.

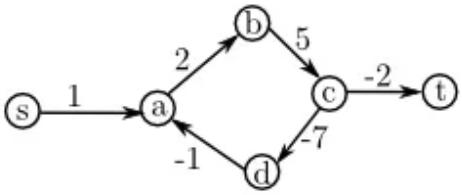
‹ Thuật toán Dijkstra tìm đường đi ngắn nhất từ một đỉnh-- Dijkstra Algorithm • Quy hoạch động tìm đường đi ngắn nhất giữa mọi cặp đỉnh--- Floyd-Warshall Algorithm ›

Thuật toán Bellman-Ford tìm đường đi ngắn nhất từ một đỉnh trong đồ thị có trọng số âm -- Bellman Ford Algorithm

January 20, 2016 in Uncategorized | No comments

Bài trước chúng ta đã tìm hiểu thuật toán Dijkstra (http://www.giaithuatlaptrinh.com/?p=764) tìm đường đi ngắn nhất từ một đỉnh tới mọi đỉnh khác trong đồ thị có hướng với trọng số không âm trong thời gian $O(E + V \log V)$. Nếu đồ thị có trọng số âm, chúng ta sẽ không thể áp dụng thuật toán Dijkstra được (tại sao?). Trong trường hợp này, chúng ta sẽ áp dụng thuật toán Bellman Ford (https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm) và đó cũng là nội dung chính của bài viết này.

Điểm đặc biệt của đố thị có trọng số âm là có thể có sự xuất hiện của chu trình âm, là chu trình mà tổng trọng số các cạnh nhỏ hơn 0. Sự xuất hiện của chu trình âm làm cho bài toán trở nên không xác định. Ví dụ trong hình dưới đây, ta có một chu trình âm là $a \to b \to c \to d \to a$. Ta muốn tìm đường đi ngắn nhất từ đỉnh s đến đỉnh t. Ta có thể đi từ s đến a sau đó đi xung quanh chu trình âm vô hạn lần, trước khi ta đi đến t. Như vậy, đường đi ngắn nhất từ s đến t là không xác định (hoặc có thể coi là âm vô hạn).



Tuy nhiên, nếu trong đồ thị không có chu trình âm, ta luôn có thể tìm được đường đi ngắn nhất từ một đỉnh tới mọi đỉnh khác trong đồ thị.

Problem 1: Cho một đồ thị có hướng, có trọng số $G(V, \vec{E})$ và một đỉnh $s \in V$. Xác định xem G có chu trình âm hay không và nếu G không có chu trình âm, tìm đường đi ngắn nhất từ s tới mọi đỉnh khác trong G.

Thuật toán tìm đường đi ngắn nhất trong đồ thị không có chu trình âm được phát triển bởi Bellman [1] năm 1958 và Ford [2] năm 1956, do đó, nó được gọi là thuật toán Bellman-Ford. Tuy nhiên, Shimbel phát hiện ra thuật toán này vào năm 1954, trước cả Bellman và Ford , do đó một số nguồn gọi đây là thuật toán Shimbel [3].

Theorem 1: Trong thời gian $O(VE)$, thuật toán Bellman-Ford sẽ xác định được G có chu trình âm hay không và nếu không, tìm đường đi ngắn nhất từ s tới mọi đỉnh trong G.

Thuật toán Bellman-Ford

Ta sẽ quy ước nếu trong đồ thị mà cung $u \to v$ không tồn tại, ta sẽ coi như nó tồn tại với trọng số $+\infty$. Mục đích của quy ước này là làm cho việc viết gia mã và chứng minh trở nên đơn giản hơn.

Gọi $\delta(s, v)$ là khoảng cách ngắn nhất từ s đến v. Cũng giống như thuật toán Dijkstra, với mỗi đỉnh v, ta sẽ lưu một nhãn tạm thời $d[v]$ và sau đó cập nhật $d[v]$ trong các vòng lặp. Ta có thể coi nhãn tạm thời này là một *ước lượng tạm thời* của khoảng cách ngắn nhất từ s tới v. Mỗi giá trị của $d[v]$ sẽ có ý nghĩa như sau:

> **Ý nghĩa nhãn:** Giá trị $d[v]$ sẽ tương ứng với độ dài của *một đường đi nào đó* (có thể chưa phải là ngắn nhất) từ s tới v.

Ta khởi tạo $d[v] = w(s \to v)$. Ý nghĩa của việc khởi tạo này là tạm thời coi cạnh $s \to v$ là ước lượng của đường đi ngắn nhất từ s tới v. Trong thuật toán Dijkstra, ở bước đầu tiên, ta sẽ lấy ra đỉnh v có $d[v]$ nhỏ nhất và đó cũng là khoảng cách ngắn nhất từ s tới v do đồ thị không có trọng số âm (các bạn xem lại phần experimental thought). Tuy nhiên, nếu đồ thị trọng số âm thì điều này không đúng.

Ta sẽ sử dụng ý nghĩa của nhãn trên để tìm đường đi ngắn nhất. Dễ thấy, nếu tồn tại cung $u \to v$ mà $d[u] + w(u \to v) < d[v]$ thì đường đi từ v qua u và tới s sẽ ngắn hơn đường đi hiện tại tương ứng với ước lượng $d[v]$. Do đó, ta sẽ cập nhật lại:

$$d[v] = d[u] + w(u \to v) \qquad (1)$$

Ý nghĩa của việc cập nhật lại là đường đi từ v qua u tới s là một ước lượng tốt hơn với ước lượng hiện tại của v. Ta gọi cung $(u \to v)$ thỏa mãn $d[u] + w(u \to v) < d[v]$ là cung bị *căng* (tense), và thao tác gán lại nhãn của v theo phương trình (1) là thao tác *nới lỏng* (relax) cung $(u \to v)$.

Như vậy, thuật toán tìm đường đi ngắn nhất có thể được phát biểu đơn giản trong *một câu* như sau:

> **Luật nới lỏng:** Tìm các cung căng và nới lỏng chúng.

Giả mã:

```
GenericBellmanFord(G(V, E), w, s):
    d[s] ← 0
    for each v ∈ V
        d[v] ← w(s → v)
    repeat
        for every tense arc (u → v)
            Relax(u → v)
    until there is no tense edge
```

```
RELAX(u → v):
    d[v] ← d[u] + w(u → v)
```

Hiển nhiên, ta nới lỏng đến khi nào đồ thị không còn cung nào căng thì lúc đó ước lượng $d[v]$ cũng chính là khoảng cách ngắn nhất, i.e, $d[v] = \delta(s, v)$. Nếu G có chu trình âm (như trong ví dụ trên) thì luôn tồn tại một cung căng trong mỗi vòng lặp của thuật toán trên (tại sao?). Do đó, thuật toán sẽ không bao giờ kết thúc.

Bây giờ ta sẽ giả sử G không có chu trình âm, thuật toán GENERICBELLMANFORD sẽ luôn kết thúc do mỗi bước, ít nhất một đỉnh v sẽ có nhãn $d[v]$ giảm xuống ít nhất là một đơn vị (giả sử trọng số các cạnh là các số nguyên) và nhãn đó không thể giảm tới âm vô hạn được. Vấn đề còn lại là thuật toán trên sẽ kết thúc sao bao nhiêu vòng lặp?

Để trả lời câu hỏi trên thì ta hãy xét một trường hợp đơn giản sau: Giả sử u là đỉnh mà cung $s \to u$ cũng chính là đường đi ngắn nhất từ s tới u (luôn tồn tại đỉnh như vậy trong đồ thị, nhưng không nhất thiết là đỉnh có $d[u]$ nhỏ nhất; xem lại Lemma 1 của bài trước (http://www.giaithuatlaptrinh.com/?p=764)). Do đó $d[u] = w(s \to u) = \delta(s, u)$ và từ đó suy ra ta sẽ không bao giờ phải cập nhật lại nhãn $d[u]$ sau khi nó được khởi tạo. Như vậy, các đỉnh có đường đi ngắn nhất từ s mà *không* đi qua đỉnh nào khác sẽ không cần phải cập nhật lại nhãn sau khi khởi tạo.

Mở rộng ví dụ trên ra một chút, xét đỉnh v mà đường đi ngắn nhất từ s tới v chỉ đi qua một đỉnh u nào đó. Hay nói cách khác $s \to u \to v$ là đường đi ngắn nhất. Do đó, $s \to u$ là đường đi ngắn nhất từ s tới u, và theo như trên $d[u]$ sẽ không cần phải cập nhật lại sau khi nó được khởi tạo. Trong vòng lặp đầu tiên, cung $u \to v$ sẽ căng (tại sao?) và ta sẽ cập nhật lại $d[v] = d[u] + w(u \to v)$. Do đó, $d[v]$ chính là độ dài đường đi $s \to u \to v$ và từ đó suy ra $d[v] = \delta(s, v)$. Như vậy, các đỉnh v có đường đi ngắn nhất từ s đi qua một đỉnh khác sẽ có nhãn được cập nhật trong vòng lặp đầu tiên và nhãn này sẽ không bao giờ được cập nhật ở các vòng lặp sau đó nữa.

Tổng quát hóa lên ta sẽ thấy, nếu một đỉnh v có đường đi ngắn nhất từ s đi qua k đỉnh khác thì nhãn của v sẽ không bao giờ được cập nhật sau k vòng lặp của thuật toán. Do đường đi ngắn nhất từ s tới một đỉnh bất kì không đi qua nhiều hơn $|V| - 2$ đỉnh khác, thuật toán trên sẽ lặp sau $|V| - 2$ bước.

Điều gì xảy ra nếu sau $|V| - 2$ bước mà vẫn có một cung bị căng? Dễ thấy trường hợp này chỉ xảy ra khi đồ thị có chu trình âm. Qua đó, ta có thể phát hiện được đồ thị có chu trình âm hay không. Giả mã của thuật toán như sau:

```
BELLMANFORD(G(V, E), w, s):
    d[s] ← 0
    P[s] ← s
    for each v ∈ V
        d[v] ← w(s → v)
        P[v] ← s
    for i ← 1 to V − 2
        for every tense arc (u → v)
```

```
        RELAX(u → v)
    if there is a tense arc
        print G has a negative cycle
```

```
RELAX((u → v)):
    d[v] ← d[u] + w(u → v)
    P[v] ← u
```

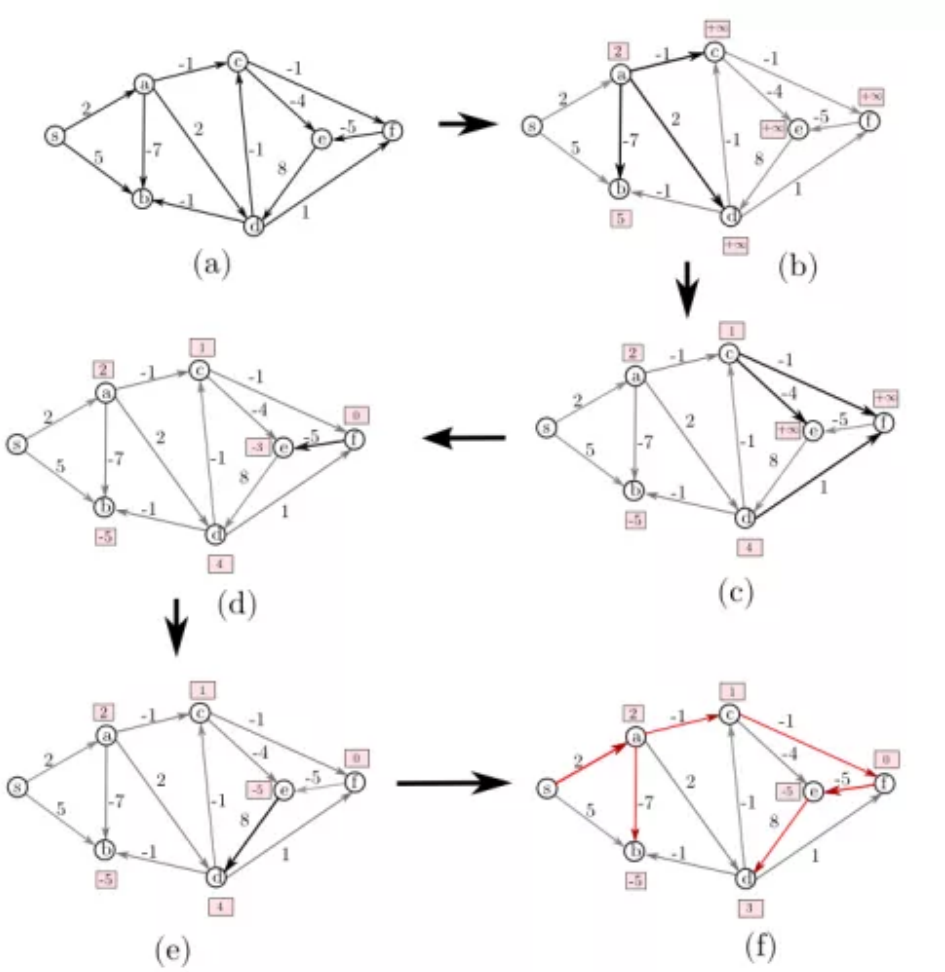
```
FINDREVERSESHORTESTPATH(t, P[1, 2, ..., V]):
    print t
    while P[t] ≠ t
        t ← P[t]
        print t
```

Mảng $P[1, 2, \ldots, V]$ ở trên được dùng để lưu vết đường đi. Ta sử dụng mảng này để in ra đường đi ngắn nhất từ s tới t.
Code C:

```
+ expand source (#)
```

Phân tích thuật toán: Để phát hiện ra các cung căng, ta chỉ cần duyệt qua tất cả các cung. Sử dụng danh sách kề, các cung có thể được duyệt qua trong thời gian $O(E)$, do đó, tổng thời gian của thuật toán là $O(VE)$. So với Dijkstra ($O(V \log V + E)$) thì thuật toán Bellman Ford chậm hơn rất nhiều.

Một ví dụ thực thi thuật toán Bellman-Ford được minh họa trong hình dưới đây. Các cung tô đậm là các cung bị căng. Trong hình (f), các cung màu đỏ là các cung trong cây đường đi ngắn nhất xuất phát tại s. Các số trong hộp được tô màu hồng là nhãn $d[v]$ của đỉnh.



Remark Nếu chúng ta biết trước được G không có chu trình âm thì chúng ta không nhất thiết phải chạy hết $|V|-2$ vòng lặp mà ta có thể dừng ngay khi đồ thị G không còn cung căng nữa.

Thuật toán Bellman-Ford và quy hoạch động

Ta có thể phát biểu lại thuật toán Bellman-Ford dưới góc nhìn của quy hoạch động (http://www.giaithuatlaptrinh.com/?p=66) .

> Gọi $d[v][i]$ là độ dài của đường đi ngắn nhất từ s đến v đi qua tối đa i đỉnh.

Ta sẽ khởi tạo, $d[v][0] = w(s \to v)$ và $d[v][i] = +\infty$ với mọi $i > 0$. Do đường đi ngắn nhất từ s đến v trong đồ thị không có chu trình âm đi qua tối đa $V-2$ đỉnh, ta có:

$$\delta(s,v) = d[v][V-2] \qquad (2)$$

Từ định nghĩa của $d[v][i]$, ta có công thức đệ quy sau:

$$d[v][i] = \min_{u:(u \to v) \in E}(d[u][i-1] + w(u \to v)) \qquad (3)$$

Ý nghĩa của công thức đệ quy trên là như sau: Nếu gọi $s \to \ldots \to u \to v$ là một đường đi ngắn nhất từ s tới v đi qua tối đa i đỉnh thì đường đi con $s \to \ldots \to u$ là đường đi ngắn nhất từ s tới u đi qua tối đa $i-1$ đỉnh. Do

đó, $d[v][i] = d[u][i-1] + w(u \to v)$. Do ta không biết u, ta sẽ lặp qua tất cả các đỉnh u mà $u \to v \in \vec{E}$ rồi lấy giá trị nhỏ nhất.

Ta có thuật toán sau:

```
DynamicBellmanFordA(G(V, E), w, s):
    for each v ∈ V
        d[v][0] ← w(s → v)
    for i ← 1 to V − 2
        for each v ∈ V
            for each u → v ∈ E
                d[v][i] = min(d[u][i − 1] + w(u → v), d[v][i])
    output d[v][V − 2] as the shortest distance of v
```

Phân tích thuật toán: Ta sẽ phân tích hai vòng lặp for trong cùng của gỉa mã trên. Với mỗi đỉnh $v \in V$, ta sẽ duyệt qua các đỉnh u mà có một cung $u \to v \in \vec{E}$. Do đó, số vòng lặp của hai vòng for trong cùng là:

$$\sum_{v \in V} \deg^+(v) = O(E) \qquad (4)$$

trong đó $\deg^+(v)$ là bậc tới của đỉnh v. Như vậy thời gian của thuật toán là $O(VE)$. Bộ nhớ của thuật toán sử dụng là $O(V^2)$ vì ta sử dụng mảng hai chiều d.

Tuy nhiên, ta có thể quan sát thấy dòng $d[1, 2, \ldots, V][i]$ của bảng hai chiều d chỉ phụ thuộc vào dòng trước đó $d[1, 2, \ldots, V][i-1]$ của bảng này, do đó, ta có thể giảm bộ nhớ bằng cách chỉ dùng mảng 1 chiều. Thuật toán có thể viết lại như sau:

```
DynamicBellmanFordB(G(V, E), w, s):
    for each v ∈ V
        d[v][0] ← w(s → v)
    for i ← 1 to V − 2
        for each v ∈ V
            for each u → v ∈ E
                d[v] = min(d[u] + w(u → v), d[v])        (∗)
    output d[v] as the shortest distance of v
```

Để ý trong dòng $(*)$ của thuật toán DynamicBellmanFordB, gía trị $d[v]$ chỉ được cập nhật khi và chỉ khi cung $d[v] < d[u] + w(u \to v)$. Hay nói cách khác, cung $(u \to v)$ bị căng. Do đó, ta có thể viết lại thuật toán như sau:

```
DynamicBellmanFordC(G(V, E), w, s):
    for each v ∈ V
        d[v][0] ← w(s → v)
    for i ← 1 to V − 2
        for each v ∈ V
            for each u → v ∈ E
                if (u → v) is tense
                    Relax(u → v)
    output d[v] as the shortest distance of v
```

Thực chất hai vòng for trong cùng chính là duyệt qua tất cả các cạnh của G và nới lỏng các cạnh bị căng. Do đó, thu gọn lại thuật toán DynamicBellmanFordC ta sẽ thu được thuật toán Bellman-Ford ban đầu.

Remarks Như đã phân tích ở trên, ta cần tối đa $V-2$ vòng lặp (ngoài cùng) để nới lỏng các cung căng và sau $V-2$ vòng lặp đó, nhãn của các đỉnh chính là đường đi ngắn nhất. Mặc dù cho đến nay chưa có thuật toán nào có thời gian tốt hơn $O(VE)$ về mặt lý thuyết, ta vẫn có thể tăng tốc thuật toán Bellman-Ford trong thực tế. Nhận xét thấy trong giả mã BellmanFord ở trên, ta không quan tâm đến thứ tự các đỉnh hay cung mà ta sẽ nới lỏng.

Năm 1970, Yen [2] đưa ra một thứ tự để nới lỏng các đỉnh, giảm số vòng lặp ngoài cùng xuống còn cỡ $V/2$. Năm 2012, Bannister và Eppstein [6] chứng minh rằng nếu thứ tự các đỉnh nới lỏng được chọn ngẫu nhiên thì sau $V/3$ vòng lặp, với xác suất cao, chúng ta sẽ tìm được đường đi ngắn nhất.

Chú ý cuối cùng đó là thuật toán Bellman-Ford **không thể** áp dụng được cho đồ thị vô hướng có trọng số âm. Nguyên nhân là khi ta chuyển mỗi cạnh vô hướng thành hai cạnh có hướng ngược chiều nhau, cạnh vô hướng với trọng số âm sẽ trở thành chu trình âm trong đồ thị. Bài toán đường đi ngắn nhất trong đồ thị vô hướng không có chu trình âm khó hơn rất nhiều.

Code đầy đủ: bellman-ford-matrix-representation (http://www.giaithuatlaptrinh.com/wp-content/uploads/2016/01/bf_matrix.c), bellman-ford-list-represetation (http://www.giaithuatlaptrinh.com/wp-content/uploads/2016/01/bf_adj_list.c).

Tham khảo

[1] Bellman, Richard. *On a routing problem*. Quarterly of Applied Mathematics 16: 87–90, 1958.
[2] Ford Jr., Lester R. *Network Flow Theory*. Paper P-923. Santa Monica, California: RAND Corporation, 1956.
[3] Jeff Erickson. *Lecture Notes on Single Source Shortest Paths (http://jeffe.cs.illinois.edu/teaching/algorithms/notes/21-sssp.pdf)*, UIUC, 2014.
[4] Uri Zwick. *Lecture Notes on Shortest Paths (http://www.cs.tau.ac.il/~zwick/grad-algo-09/short-path.pdf)*, Tel Aviv University, 2009.
[5] Yen, Jin Y. *An algorithm for finding shortest routes from all source nodes to a given destination in general networks*. Quarterly of Applied Mathematics 27: 526–530, 1970.
[6] Bannister, M. J.; Eppstein, D. *Randomized speedup of the Bellman–Ford algorithm (http://arxiv.org/abs/1111.5414)*. Analytic Algorithmics and Combinatorics (ANALCO12), Kyoto, Japan. pp. 41–47, 2012.

0 Comments Sort by **Oldest**

Add a comment...

Facebook Comments Plugin

SHARE THIS:

(http://www.giaithuatlaptrinh.com/?p=789&share=twitter&nb=1)

(http://www.giaithuatlaptrinh.com/?p=789&share=facebook&nb=1)

(http://www.giaithuatlaptrinh.com/?p=789&share=google-plus-1&nb=1)

RELATED

Đường đi ngắn nhất giữa mọi cặp đỉnh trong đồ thị thưa -- Johnson's Algorithm (http://www.giaithua...p=874)
February 29, 2016
In "all-pairs-shortest-paths"

Quy hoạch động tìm đường đi ngắn nhất giữa mọi cặp đỉnh--- Floyd-Warshall Algorithm (http://www.giaithua...p=814)
January 30, 2016
In "all-pairs-shortest-paths"

Luồng trong mạng II: Thuật toán Edmonds-Karp -- Network Flow II: Edmonds-Karp Algorithm (http://www.giaithua...p=1539)
October 3, 2016
In "Edmond-Karp"

Tags: Bellman-Ford, dynamic programming, graph algorithm, negative-cycles, shortest-paths

No comments

Comments feed for this article

Trackback link: http://www.giaithuatlaptrinh.com/wp-trackback.php?p=789

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.