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.

‹ Cây nhị phân cân bằng --- Balanced binary search tree • PhD Life sucks! Chém gió sau deadline ›

Quy hoạch động trên cây --- Dynamic programming on trees

March 24, 2017 in Uncategorized | No comments

Nhiều bài toán **NP-hard (http://www.giaithuatlaptrinh.com/?p=1763)** trên đồ thị trở nên dễ dàng khi đồ thị đầu vào là một cây, i.e, không có chu trình. Dễ dàng ở đây hiểu là giải được trong thời gian đa thức (thông thường tuyến tính). Công cụ chúng ta thường sử dụng khi giải các bài toán **NP-hard** trên cây là quy hoạch động. Trong bài này, chúng ta sẽ tìm hiểu thuật toán quy hoạch động giải bài toán tập độc lập (http://www.giaithuatlaptrinh.com/?p=1791) (independent set)trên cây; một bài toán **NP-hard** trên đồ thị. Ta nhắc lại bài toán tập độc lập.

Một tập đỉnh X của đồ thị (đơn, vô hướng) $G(V, E)$ gọi là một tập độc lập nếu không tồn tại cạnh của đồ thị nối hai đỉnh trong X.

Tập độc lập trên cây: Cho một cây vô hướng $T(V_T, E_T)$ và một hàm trọng số $w : V_T \to \mathbb{R}^+$ gán cho mỗi đỉnh v của cây T một trọng số $w(v)$. Tìm tập độc lập của cây T có tổng trọng số đỉnh lớn nhất.

Ví dụ:

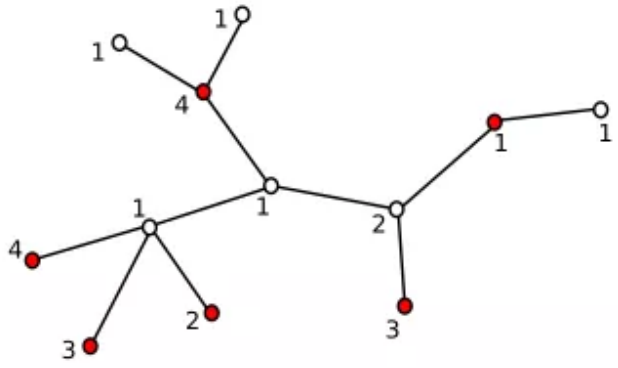

Figure 1: Tập độc lập lớn nhất là tập các đỉnh được tô màu đỏ. Tập này có trọng số 17. Số bên cạnh mỗi đỉnh là trọng số của nó.

Ta gọi tập độc lập có tổng trọng số lớn nhất là tập độc lập lớn nhất.

Thuật toán: Gọi $n = |V_T|$ là số đỉnh của T. Ta chỉ định một đỉnh r bất kì nào đó của $T(V_T, E_T)$ làm gốc của cây T. Các đỉnh có bậc 1 còn lại ta gọi là đỉnh lá (leaf vertex). Mỗi nút trong $v \in T$ sẽ có một cây con tương ứng, kí hiệu T_v, nhận v làm gốc. Gọi $D[1, \ldots, n][0, 1]$ là bảng quy hoạch động trong đó:

1. D[v,0] là trọng số của tập độc lập lớn nhất của T_v **không chứa** v.
2. D[v,1] là trọng số của tập độc lập lớn nhất của T_v **có chứa** v.

Với mỗi nút lá u, theo định nghĩa, ta có $D[u,0] = 0$ và $D[u,1] = w(u)$. Ta sẽ cập nhật bảng theo thứ tự từ lá dần lên gốc. Khi cập nhật bảng D tại nút v, ta sẽ đảm bảo giá trị của bảng tại các nút con đã được cập nhật trong vòng lặp trước đó.

Gọi $v_1, v_2, \ldots, v_k$ là các nút con của một nút trong v. Theo định nghĩa của tập độc lập, nếu v thuộc tập độc lập X thì các nút con của v sẽ không nằm trong X. Do đó:

$$D[v,1] = w(v) + \sum_{i=1}^{k} D[v_i, 0] \qquad (1)$$

Nếu v không thuộc tập độc lập X, thì ta có thể cho hoặc không cho các nút con của v vào tập độc lập, miễn sao tổng trọng số của tập là lớn nhất. Do đó:

$$D[v,0] = \sum_{i=1}^{k} \max\{D[v_i,0], D[v_i,1]\} \qquad (2)$$

Khi ta cập nhật xong $D[r,0]$ và $D[r,1]$ của nút gốc r, thì trọng số của tập độc lập lớn nhất là:

$$\max\{D[r,0], D[r,1]\} \qquad (3)$$

Tìm tập độc lập ta chỉ cần truy ngược lại bảng. Kĩ thuật này ta đã thảo luận tại bài trước (http://www.giaithuatlaptrinh.com/?p=78).

Khi thực thi thuật toán, để đảm bảo các giá trị của nút con đã được cập nhật trước khi ta cập nhật nút cha, ta sẽ duyệt và cập nhật theo thứ tự sau (post-order tranversal (https://en.wikipedia.org/wiki/Tree_traversal)). Sau khi ta cập nhật xong bảng tại một nút v, ta sẽ "gửi" các giá trị đó lên nút cha để cập nhật nút cha theo phương trình (1) và (2).

```
DPOnTrees(T(V_T, E_T), w(.)):
    root T at a vertex r
    for each vertex v of T
        D[v,0] ← 0
        D[v,1] ← w(v)
    PostOrderUpdate(T, r, D)
    return max{D[r,0], D[r,1]}
```

```
PostOrderUpdate(T(V_T, E_T), v, D):
    for each child u of v
        PostOrderUpdate(T, u, D)
    w ← parent(v)          ≪ parent of v in T ≫
    if w ≠ Null
        D[w,1] ← D[w,1] + D[v,0]
        D[w,0] ← D[w,0] + max{D[v,0], D[v,1]}
```

Ví du:

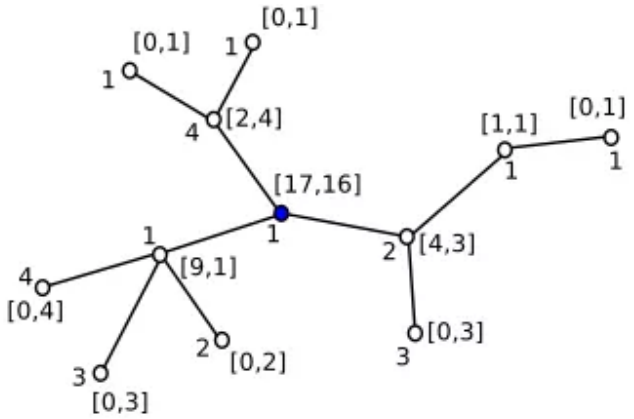

Figure 2: Kết quả của bảng quy hoạch động áp dụng cho đồ thị trong Figure 1. Đỉnh màu xanh được lấy làm gốc của cây. Cạnh mỗi đỉnh v là nhãn có dạng $[a, b]$ trong đó $a = D[v, 0]$ và $b = D[v, 1]$.

Phân tích thời gian: Thủ tục cập nhật bảng quy hoạch động duyệt qua mỗi đỉnh đúng một lần. Mỗi lần duyệt một đỉnh, ta mất O(1) thao tác để "gửi" giá trị của nút v để cập nhật giá trị của nút cha. Do đó, tổng thời gian của thuật toán là $O(n)$

Remark: Ta có thể giải một số bài toán NP-hard trên cây như: phủ đỉnh (vertex cover), đường đi dài nhất (longest path), dominating set, v.v. Tất cả đều có thể giải bằng quy hoạch động.

Tham khảo

[1] J. Erickson, Dynamic Programming Lecture Notes (http://jeffe.cs.illinois.edu/teaching/algorithms/notes/05-dynprog.pdf), UIUC, 2014.

Một số bài toán NP-complete trên đồ thị -- NP-complete problems on graphs (http://www.giaithua...p=1791)
January 8, 2017
In "clique"

Giới thiệu về P và NP --- P vs NP (http://www.giaithua...p=1719)
December 12, 2016
In "complexity"

Tổng quan về cây khung nhỏ nhất. (http://www.giaithua...p=1266)
June 17, 2016
In "minimum-spanning-tree"

Tags: dp-tree, dynamic programming, graph algorithm, independent-set, queue

No comments

Comments feed for this article

Trackback link: http://www.giaithuatlaptrinh.com/wp-trackback.php?p=1968

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.