ABC163: F - path pass i 解説

AtCoder: opt,　Twitter: opt_cp

Thursday 17th September, 2020

概要

“ABC163: F - path pass i” が全然解けなかった & 超良問だったので，自分の理解を深めるためにも解説を書きました．ツッコミ等あったら Twitter までよろしくお願いいたします．

1　問題

AtCoder Beginner Contest 163: F - path pass i

2　考察・解法

以下では，頂点集合を $V \coloneqq \{1, \ldots, N\}$ とおく．

2.1　重要な観察

まず，次の 2 点に気がつく:

- 「色 k の頂点を『一度以上』通るような単純パスの数」
  $\rightarrow$ 「色 k の頂点を『一度も通らない』単純パスの数」を全体から引けば良さそう
- 木上では，単純パスは両端点から一意に定まる
  $\rightarrow$ 頂点数 n の木の単純パスは $n(n+1)/2$ 個

これらから，次を得る:

観察 1. 木から色 k の頂点を削除したとき，残った各連結成分の頂点数を $n_1, \ldots, n_l$ とすると，問題 k の答えは

$$\frac{N(N+1)}{2} - \sum_{i=1}^{l} \frac{n_i(n_i+1)}{2}.$$

2.2 素朴な $O(N^2)$ 解法

上の観察から，次のような DP による解法が思いつく．与えられた木を，頂点 1 を根とする根付き木と考える．各頂点 $v \in V$ と色 $k = 1, \ldots, N$ に対し

$$D(v,k) \coloneqq (v \text{ を根とする部分木において，色 } k \text{ の頂点を通らずに } v \text{ から到達できる頂点の数}) \quad (1)$$

と定める．すると，漸化式

$$D(u,k) = \begin{cases} 1 + \displaystyle\sum_{v:\,(u \text{ の子})} D(v,k) & \text{if } k \neq c_u, \\ 0 & \text{if } k = c_u \end{cases}$$

が成り立つ．また，u の子 v に対し，

$$D(v,c_u) = (\text{色 } c_u \text{ の頂点を通らずに } v \text{ から到達できる頂点の数}) \quad (2)$$

も成り立つ．これらを用いて，問題 k に対する答え $A(k)$ を求めるのが，Algorithm 1 である．

Algorithm 1 素朴な解法

$k = 1, \ldots, N$ に対し，$A(k) \leftarrow \frac{N(N+1)}{2}$ と初期化
$v \in V$, $k = 1, \ldots, N$ に対し，$D(v,k) \leftarrow 0$ と初期化
DFS(1) を呼ぶ
for $k = 1, \ldots, N$:
    $A(k) \leftarrow A(k) - \frac{1}{2} D(1,k)(D(1,k)+1)$
return $A(1), \ldots, A(N)$

function DFS(u)
    for u の子 v :
        DFS(v) を呼ぶ
        $A(c_u) \leftarrow A(c_u) - \frac{1}{2} D(v,c_u)(D(v,c_u)+1)$
        for $k = 1, \ldots, N$:
            $D(u,k) \leftarrow D(u,k) + D(v,k)$
    for $k = 1, \ldots, N$:
        $D(u,k) \leftarrow D(u,k) + 1$
    $D(u,c_u) \leftarrow 0$

この解法の計算量は $O(N^2)$ であり，これでは間に合わない．

2.3 DP の in-place 化

Algorithm 1 の計算量が $O(N^2)$ になってしまった本質的な原因は，DP テーブル $D(v,k)$ のサイズが $O(N^2)$ であることである．この DP テーブルを「圧縮」するために，DP を **in-place 化**する．

Algorithm 1 において，頂点 $v \in V$ に対する $D(v,k)$ の値は，答え $A(k)$ の更新に一度使われたら，その後二度と使われることはない．つまり，全ての頂点 $v \in V$ に対して $D(v,k)$ の値を保持しておくのは無駄だと言える．さらに，答えの更新に用いられるのは，u の子 v に対し，$D(v,c_u)$ の値のみである．

そこで，DP テーブル $D(v,k)$ の添字 v を潰した新たな DP テーブル $D_2(k)$ を考える．式 (1)，(2) を参考にして，各頂点 $v \in V$ と色 $k = 1,\dots,N$ に対し，v の訪問終了時に

$$D_2(k) = (\text{色 } k \text{ の頂点を通らずに } v \text{ から到達できる「訪問終了済み頂点」の数}) \tag{3}$$

が成り立つように $D_2(k)$ を更新することにする．ここで，DFS(v) を実行することを「v を訪問する」と表現している．

このように DP テーブル $D_2(k)$ を更新すると，u の子 v の訪問終了時に

$$D_2(c_u) = D(v,c_u) = (\text{色 } c_u \text{ の頂点を通らずに } v \text{ から到達できる頂点の数}) \tag{4}$$

を満たすことが分かる．これにより，答え $A(c_u)$ の更新ができる．

この $D_2(k)$ の更新は「破壊的」であり，式 (3)，(4) は「v の訪問終了時」にしか成り立たないことに注意．このように，DP テーブルを破壊的に更新し，計算量を削減するテクニックは「DP の in-place 化」と呼ばれる[*1]．

上の等式を満たす更新は Algorithm 2 により実現できる．ややこしいポイントが 2 つあるので注意:

- $D_2(c_u) \leftarrow 0$ という更新は，各子 v の訪問開始時に行わないといけないので，for 文の中に入っている (4 行目)，
- u の訪問が終了して親に戻る時に，$D_2(c_u)$ の値を訪問開始時のものに戻さないといけない (2, 9 行目)．

Algorithm 2 in-place 化した解法 (DFS 部分のみ)

```
1: function DFS(u)
2:     t ← D2(c_u)                          ▷ u の訪問開始時点での値を保持しておく
3:     for u の子 v :
4:         D2(c_u) ← 0                      ▷ 「色 c_u を通らずに v から到達できる頂点」は v の祖先には無い
5:         DFS(v) を呼ぶ
6:         A(c_u) ← A(c_u) − (1/2) D2(c_u)(D2(c_u) + 1)   ▷ この時点では D2(c_u) = D(v, c_u) が成立
7:     for k = 1, ..., N :
8:         D2(k) ← D2(k) + 1                ▷ 新たに訪問終了済みになる u の分を足す
9:     D2(c_u) ← t                          ▷ u の訪問開始時点での値に戻す
```

[*1] in-place DP の典型問題: Educational DP Contest: Q - Flowers

2.4 for 文を取り除く

Algorithm 2 の DFS も，7 行目の for 文のせいで，素朴に実装すると計算量は $O(N^2)$ になってしまう．しかし，この問題は簡単に解決できる．

例えば，新たな変数 s と新たな DP テーブル $D_3(k)$ を用いて，常に

$$D_2(k) = s + D_3(k)$$

を満たすように s, $D_3(k)$ を更新する．7, 8 行目で，全ての k に対し $D_2(k)$ に 1 を加算する代わりに，s に 1 加算すればよい．

このように，7 行目の for 文は取り除くことができ，Algorithm 3 を得る．

Algorithm 3 $O(N)$ 解法

$k = 1, \ldots, N$ に対し，$A(k) \leftarrow \frac{N(N+1)}{2}$，$D_3(k) \leftarrow 0$ と初期化
$s \leftarrow 0$ と初期化
DFS(1) を呼ぶ
for $k = 1, \ldots, N$:
    $A(k) \leftarrow A(k) - \frac{1}{2}(s + D_3(k))(s + D_3(k) + 1)$
return $A(1), \ldots, A(N)$
function DFS(u)
    $t \leftarrow s + D_3(c_u)$ ▷ $t \leftarrow D_2(c_u)$ と同じ
    for u の子 v :
        $D_3(c_u) \leftarrow -s$ ▷ $D_2(c_u) \leftarrow 0$ と同じ
        DFS(v) を呼ぶ
        $A(c_u) \leftarrow A(c_u) - \frac{1}{2}(s + D_3(c_u))(s + D_3(c_u) + 1)$
    $s \leftarrow s + 1$
    $D_3(c_u) \leftarrow t - s$ ▷ $D_2(c_u) \leftarrow t$ と同じ

3 より深い理解のための余談

式 (3) で「訪問終了済み頂点」となっている部分を，「訪問開始済み頂点」に置き換えても，式 (4) は変わらず成り立つ．この変更は，Algorithm 3 の DFS 内 7 行目の $s \leftarrow s + 1$ を，2 行目の直後に移動することに対応する．このように $s \leftarrow s + 1$ を移動しても，アルゴリズムはもちろん全く同じ答えを返す．考察・実装する際には，好きな方を選べば良い．

4 提出コード

提出コード (C++, 882 Byte)