技術室奥プロコン#5 Day2-L 灯籠流し 解説

kaage（@ageprocpp） 2020/8/30

統計情報

First AC: noshi91

Fastest Code: autumn_eel(Tester)

Shortest Code: autumn_eel(Tester)

一応ラスボス問でしたが、そんなに難しくできませんでした…

見た目が JOI すぎますが、JOI ほど良問ではないです。

問題概要

・根付き木があって、根に向かってたくさん灯籠を流す

・それぞれ流す時刻と頂点が決まっていて、一定時間で消える

・灯籠を流したり、ある時刻までにある頂点で見られる灯籠の数を答えたりせよ（クエリたくさん）

問題

・札[illegible]向かってたくさん灯籠を流す

・[illegible]ていて、一定時間で消える

[illegible]で見られる灯籠の数を答え[illegible]

意味不明

考察

・データ構造で頑張って答えられそうな問題でもない

・途中で灯籠が追加されるので、前計算系のアルゴリズムが使えなそう

・ヤバそう

考察（小課題 1）

愚直　やるだけ

シミュレーションをします。

計算量はどれだけ雑に実装しても $O(NQ^2)$ になります。

これで1/100点なのでコスパは悪いですね（要出典）

考察（小課題 2）

辺の長さと流し始める時刻が小さくて、観測時刻が遅い。
→実質時間のことは考えなくてよい（流れてくる灯籠は必ず観測できるため）
結局、灯籠追加は全部根まで流してやって、通った過程を観測すれば良い。
これは超典型で、HL分解などを使って根まで灯籠を流してやれば解ける。
ここまでで 16/100 点

考察（小課題 3）

ダブリングなどを使えば、小課題 2 の HLD と同様に根へのパスの一部に灯籠を加算することができるが、これだと流す時間や観測時間の条件を満たせない。

クエリ先読みをすることを考えても、途中で追加とクエリ応答を繰り返すので、時間でソートしたりするわけにもいかない。

ここで、ある灯籠があるクエリで数えられる条件を考える。

考察（小課題 3）

このとき、関わる条件は2種類に分けられる。

「灯籠がそこにたどり着く時刻が観測時刻より早い」

「そこまで灯籠が流れてくる時間が消滅時間より短い」

逆に、これさえ満たしていれば、灯籠を流した場所から根へのパス上のすべての頂点で観測されうる。

このように、2つの不等式を満たす点集合を探すときは、片方の条件でクエリと点をソートしておいて、Fenwick Tree などに載せれば高速に求めることができる。

考察（小課題 3）

各回答クエリに反応すべき灯籠の流れ始める頂点は高々 $O(N)$ 個なので、すべての回答クエリから BFS を使って頂点にクエリの情報を配る。
次に、灯籠の情報も頂点に配ってやって、それぞれの頂点で先ほどの Fenwick Tree を使った処理をしてやる。
これで、$O(Q(N+\log Q))$ くらいの計算量になるので、通すことができる。
ここまでで 43/100 点

考察（満点）

先ほどの解法を考えてみると、BFS で情報を伝えるのに時間がかかっていそうだということがわかる。
情報を伝えるための BFS の時間をなんとか削減できないか？
→十ク　エ　リ　平　方　分　割十
クエリ平方分割をして、クエリに関係ある頂点以外を無視して木を縮約すればすれば、BFS の時間がバケット数と同じになって、計算時間が $O((N+Q\log Q)\sqrt{Q})$ になる。

これは頑張れば間に合うので、これで 100 点が獲得できる！

考察（満点）

実は Tester 解はこれより定数倍が 4 倍くらい早いです。
考えてみましょう。
木に対するクエリを平方分割して頂点数を減らすテクは、Link-Cut-Tree をオフラインで使う問題の別解法として有名です。
Writer 解の実装はめちゃくちゃに重くて、本質部分だけで 170 行くらいあります。
Tester 解はもうちょっと軽い上、余裕を持って通ります。

参加ありがとうございました