Problem A

Balanced Paths

JAG 春コンテスト2015

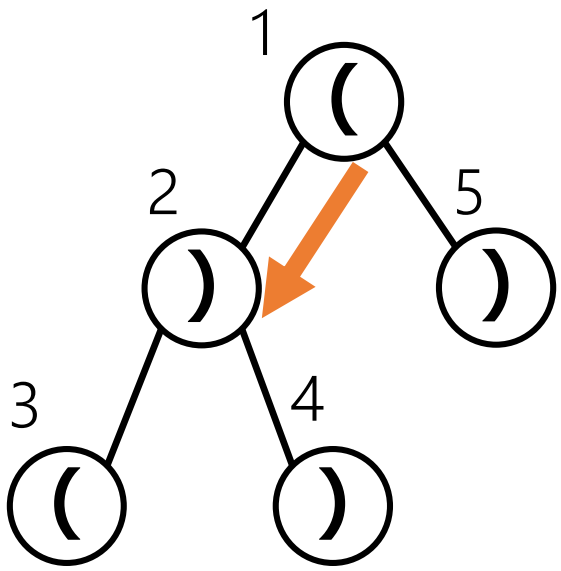
原案：八森
解答：井上・矢野・保坂
解説：矢野

問題概要

- 各頂点に "(" または ")" のラベルが付いた木が与えられる
- $l[u \to v]$ を頂点 u から頂点 v までのパス上の頂点についてラベルを順番に繋げた文字列であると定義する
- 頂点の順序対 (u, v) であって $l[u \to v]$ がバランスしているようなものの数を求めよ

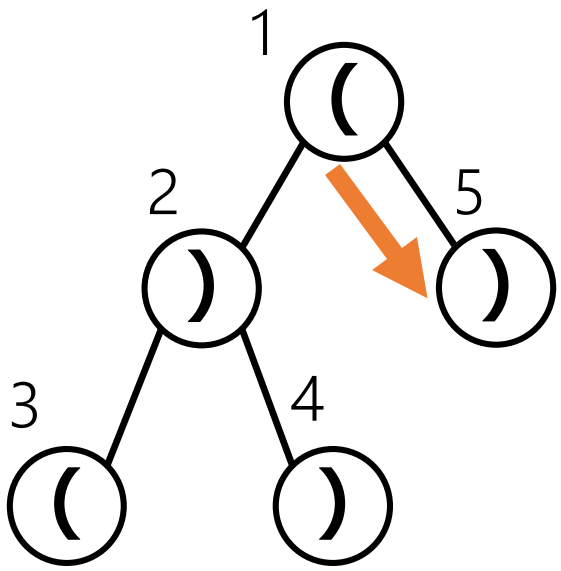
  バランスしている文字列は次のように定義される
  - 空文字列はバランスしている
  - s がバランスしているとき "(" s ")" はバランスしている
  - $s,\ t$ がバランスしているとき st はバランスしている
  - それ以外の文字列はバランスしていない
- 制約：頂点数 10^5 個以下

例（サンプル3）



$v[1 \to 2] = \text{"()"}$

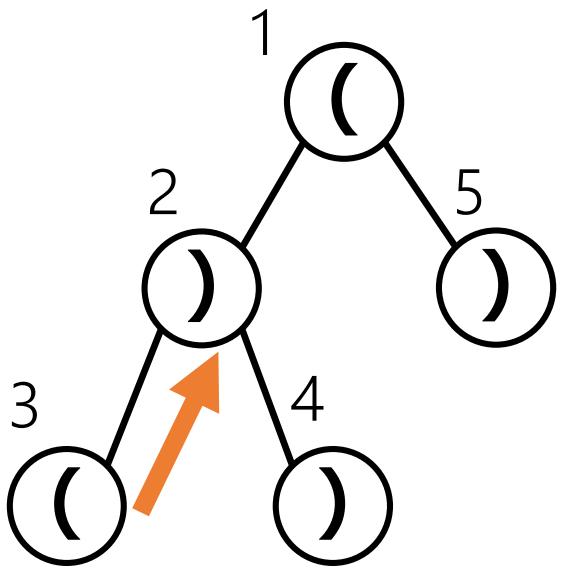
例（サンプル3）



$v[1 \to 2] =$ "()“

$v[1 \to 5] =$ "()"

例（サンプル3）

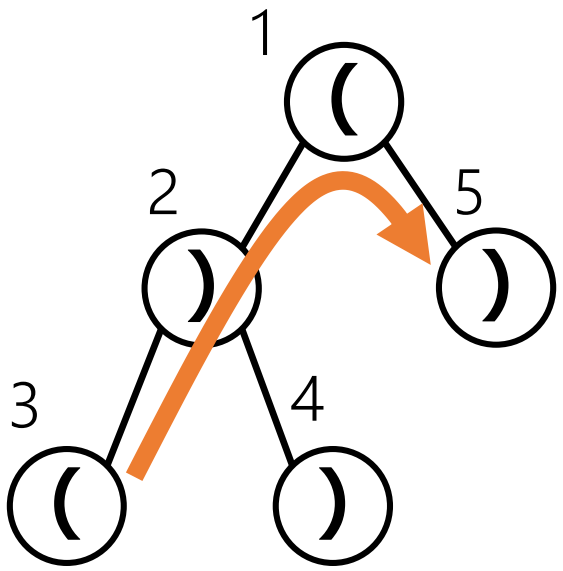


$v[1 \to 2] =$ "()“

$v[1 \to 5] =$ "()“

$v[3 \to 2] =$ "()"

例（サンプル3）

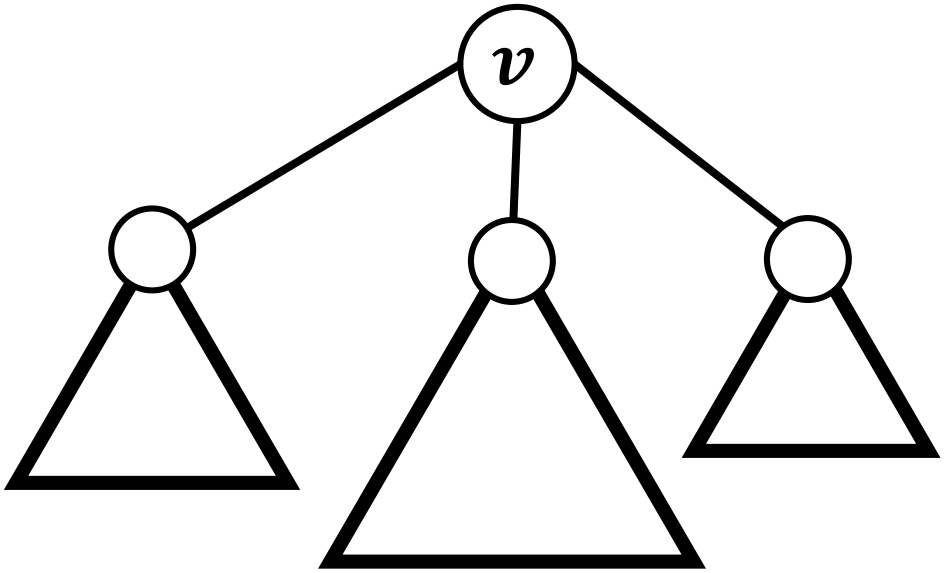


$v[1 \to 2] = \text{"()"}$
$v[1 \to 5] = \text{"()"}$
$v[3 \to 2] = \text{"()"}$
$v[3 \to 5] = \text{"()()"}$

答え：4

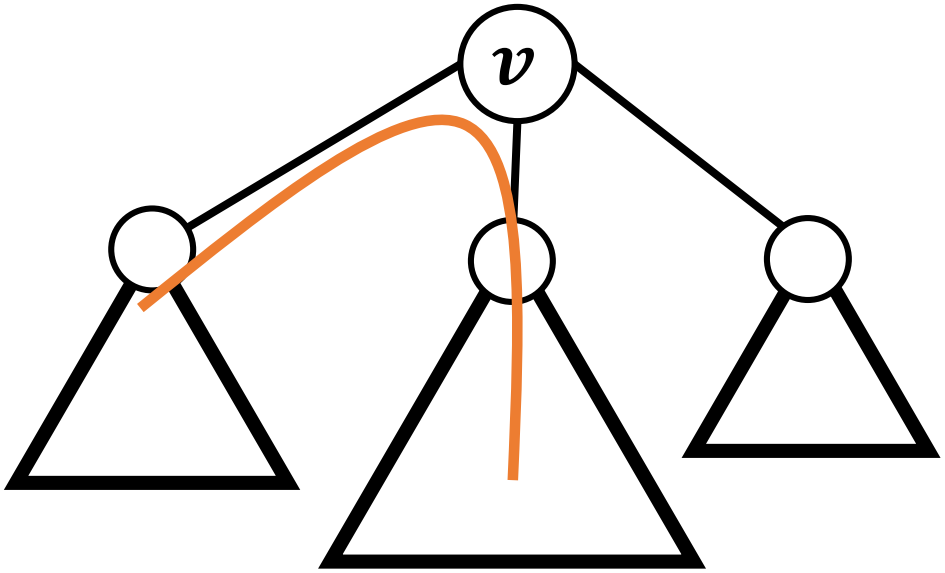
考察

- 木を見たらなんとなく部分木に分けたくなるのが人情なので，ある頂点 v とその部分木について考える



考察

- 木上のパスは v を通るものと各部分木で完結しているものに分けられる
- 後者は再帰的にそれぞれの部分木に任せることにして，前者でバランスしているパスの数を求めたい



考察

- それぞれの部分木について
  - open[u][i] = 部分木の根 u までのパスであって，後ろにi 個閉じ括弧を加えるとバランスした文字列になるものの数
  - close[u][i] = 部分木の根 u からのパスであって，前に i 個開き括弧を加えるとバランスした文字列になるものの数

  が分かれば v を通るバランスしたパスの数が分かる
- v が開き括弧の時は u, w を根とする異なる部分木について open[u][i] と close[w][i – 1] の積を足し合わせればよい
- v が閉じ括弧の時は open[u][i – 1] と close[w][i] の積

問題点

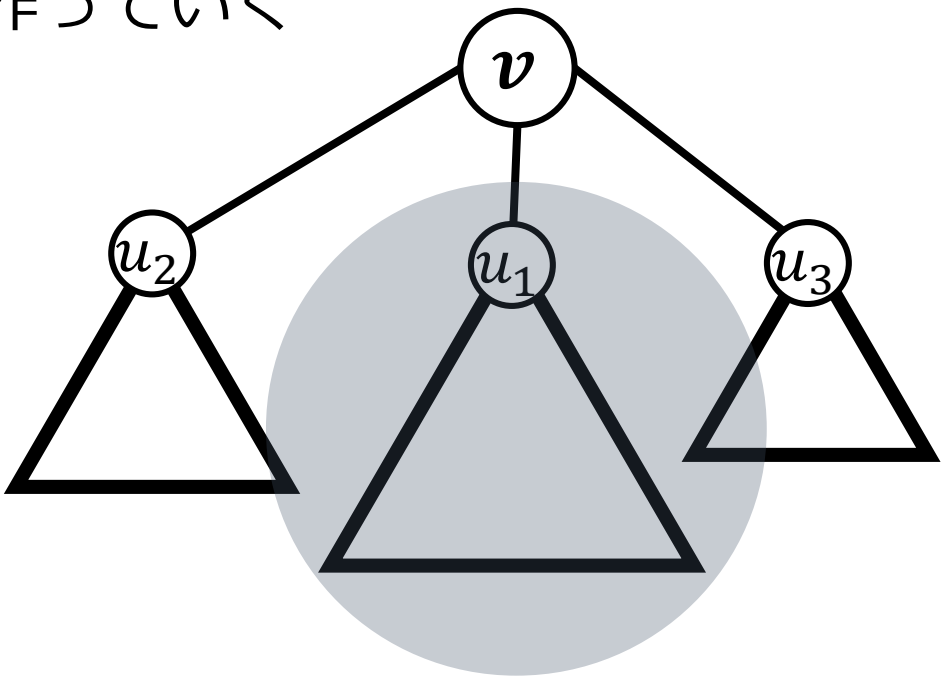
- 愚直に各頂点について open[v][i], close[v][i] をもつと頂点数の2乗の時間と空間が必要となる
- これを何とかする

解法1: Weighted-Union Heuristic

Weighted-Union Heuristics
(a.k.a. データ構造をマージする一般的なテク)

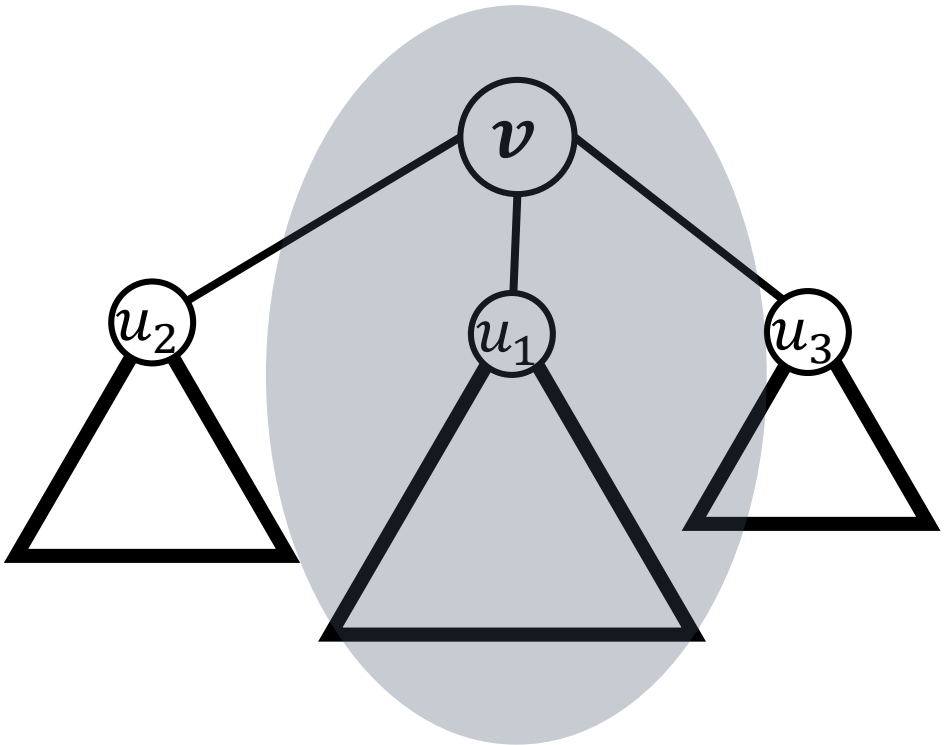
(参考: http://topcoder.g.hatena.ne.jp/iwiwi/20131226/1388062106)

- 部分木の open[u][i], close[u][i] を用いて全体の open[v][i] を構築するとき，**もっとも大きい部分木**のものを再利用しながら作っていく



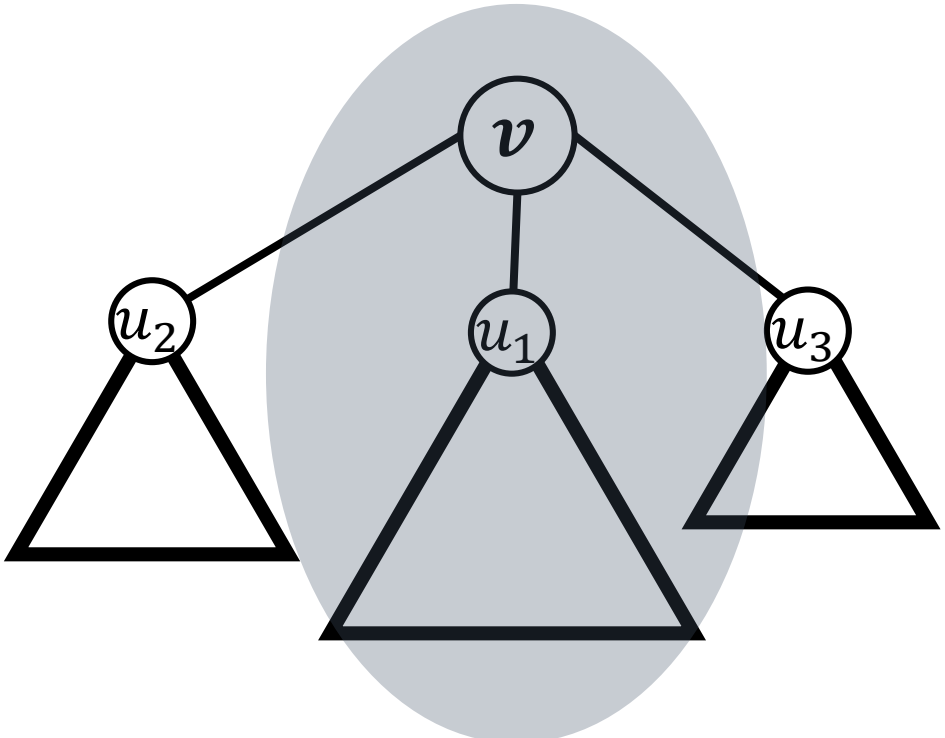
解法1: Weighted-Union Heuristic

- この部分の open[v][i], close[v][i] を open[u1][i], close[u1][i] を用いて作るにはどうすればよいか？
- u_1を根とする部分木のパスに頂点 v を加えたパスと，v 単独からなるパスを考えればよい



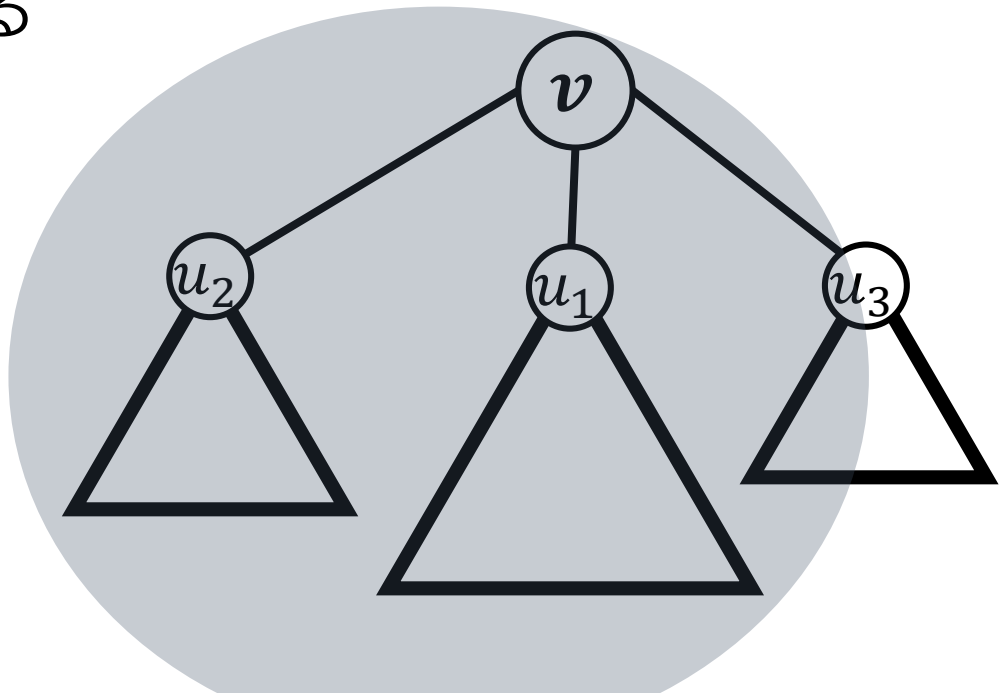
解法1: Weighted-Union Heuristic

- v が開き括弧の時，基本的にopen[v][i] = open[u1][i - 1]
- v 単独からなるパスも増えるので open[v][1] に1を足す
- 閉じ括弧の時は open[v][i] = open[u1][i + 1]
- 大体 index が 1 ずれるだけ → **deque** で定数時間で更新



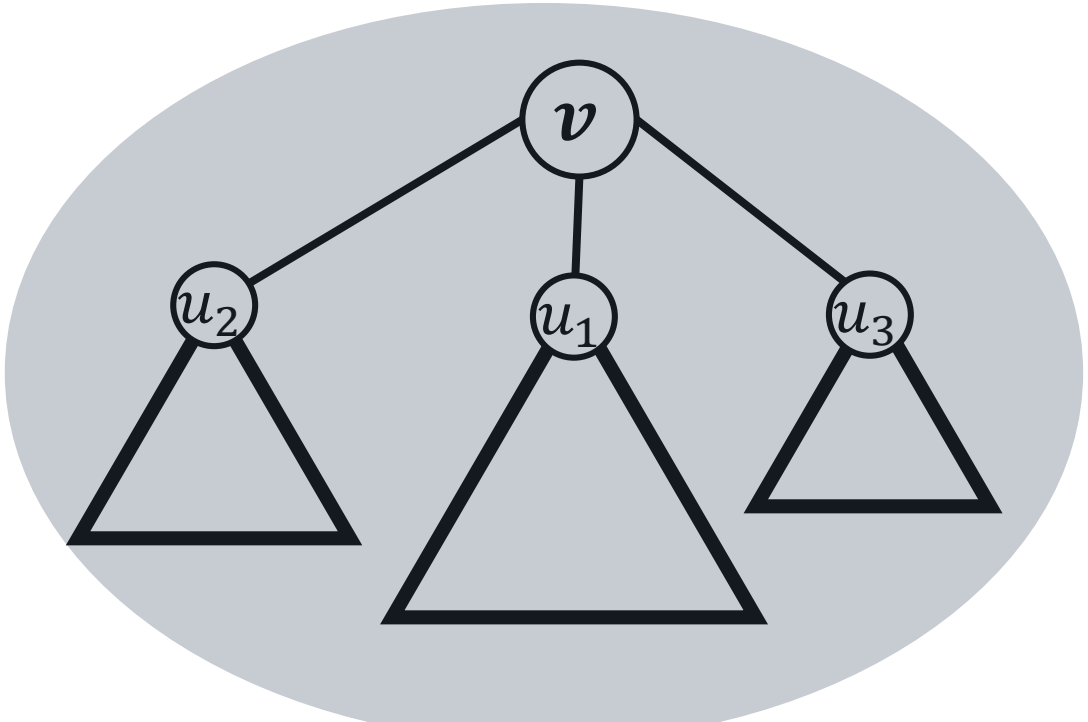
解法1: Weighted-Union Heuristic

- これに他の部分木をマージする際は
  open[v][i] += open[u2][i + 1]　（v が開き括弧の時）
  open[v][i] += open[u2][i – 1]　（v が閉じ括弧の時）
- これは高々マージする部分木の大きさの時間でできる
- 同時に二つの部分木をまたぐバランスした文字列の数を計算する



解法1: Weighted-Union Heuristic

- 部分木のマージを繰り返して全体の open[v][i] が構築できた（close についても同様）
- 全体の計算量は$O(n \log n)$



解法2: 重心分解

- 部分木の大きさが全て $n/2$ 以下になるような頂点が存在するので，そこで分けるというのを再帰的にやる（重心分解: centroid decomposition）
- 分割の深さが高々 $O(\log n)$ になるので毎回DFSなどで線形時間かけてopen[v][i], close[v][i] を求めてよい

解法2: 重心分解

- 注意としては v の次数が大きい時に，次数の2乗の時間をかけてしまうとTLEする（スターグラフ等）
- 部分木から出てきて v を通って同じ部分木に戻るようなパスもとりあえず数えておいて，後から引くという方針が簡単
- どちらの方針も結構丁寧にやらないとパスを過不足なく数えるのが少し大変かもしれません

ジャッジ解

- 解法1：井上 (C++) 104行 2361 bytes
- 解法1：矢野 (C++) 145行 3425 bytes
- 解法2：保坂 (Java) 153行 3734 bytes

統計

- Accepted / Submissions

  11 / 38 (29 %)

- First Acceptance

  yutaka1999 (34 : 18)