Post90210

Стажировка осень-зима 2022: бэкенд

11 ноя 2022, 22:54:24
старт: 11 ноя 2022, 17:54:24
финиш: 11 ноя 2022, 22:54:24
длительность: 05:00:00
начало: 24 авг 2022, 00:00:00

С. Приснится же такое...

Язык	Ограничение времени	Ограничение памяти	Ввод	Вывод
Все языки	2 секунды	512Mb	стандартный ввод или input.txt	стандартный вывод или output.txt
Python 3.7.3	3 секунды	512Mb		
Python 3.7 (PyPy 7.3.3)	3 секунды	512Mb		
Scala 2.13.4	3 секунды	512Mb		
OpenJDK Java 15	3 секунды	512Mb		
PHP 7.3.5	3 секунды	512Mb		
Kotlin 1.5.32 (JRE 11)	3 секунды	512Mb		

Наконец-то с царством Морфея удалось наладить дипломатические отношения! Первым делом в магазины поступили самые корректные и полные сонники, составленные в сотрудничестве с главными сномагами царства.

Ваш близкий друг Тирания Вампадур купила такой сонник одной из первых. Но тут же её ждало разочарование. Оказалось, что некоторые сны образуют целую последовательность сюжетов, которую надо интерпретировать только целиком.

И у Тирании оказалась именно такая ситуация. Когда-то давно ей приснилось двоичное дерево из N вершин, занумерованных целыми числами от 1 до N.

Вершина 1 являлась корнем. У каждой вершины v было до двух сыновей: левый имел номер $2 \cdot v$, правый — $2 \cdot v + 1$ (при условии, что их номера не превосходили N). Таким образом, зная число N, дерево можно было однозначно восстановить.

Но, к сожалению, следующие Q ночей Тирании снились похожие сны: одна из вершин дерева v менялась местами с её предком (если v была корнем дерева, то ничего не происходило). Причем эти изменения переносились между снами, всё больше и больше изменяя оригинальное дерево.

Чтобы верно интерпретировать значение снов, Тирании нужно узнать итоговую структуру дерева после всех произошедших с ним изменений. Она просит вас помочь ей и по последовательности менявшихся вершин найти итоговую структуру дерева из её снов.

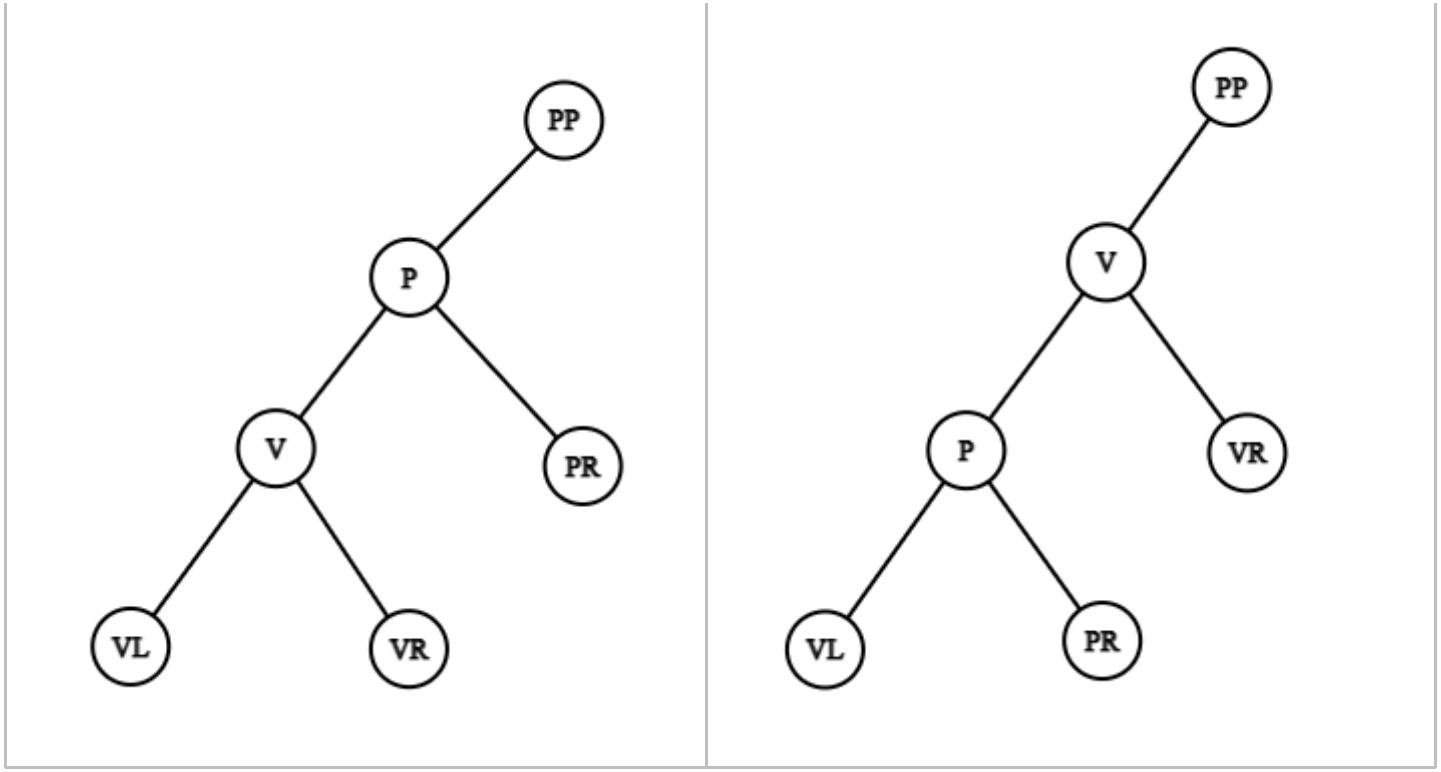
Понимая, что в этом деле важна точность, вы расспросили Тиранию насчет процесса обмена местами вершины v с её предком.

Введем обозначения:

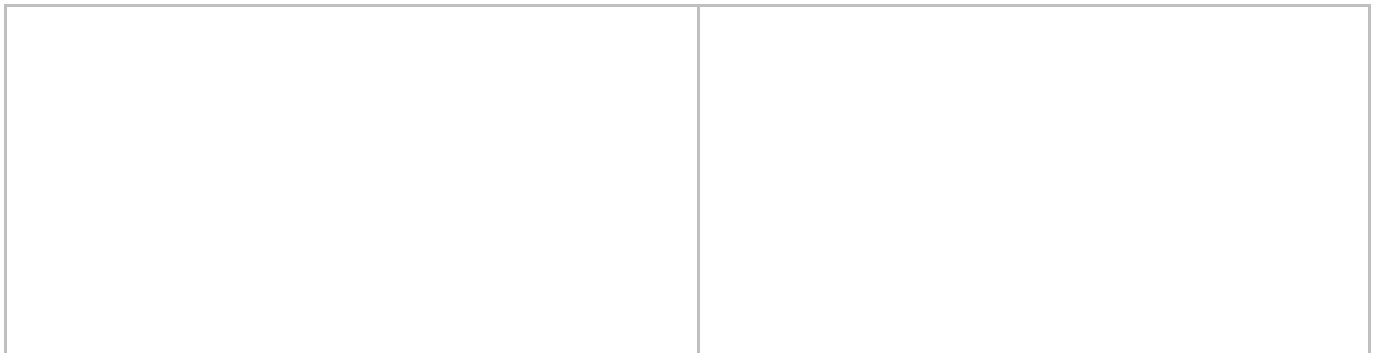
- p — предок вершины v, pp — предок вершины p (если таковые существуют);
- vl — левый ребенок v, vr — правый ребенок v;
- pl — левый ребенок p, pr — правый ребенок p.

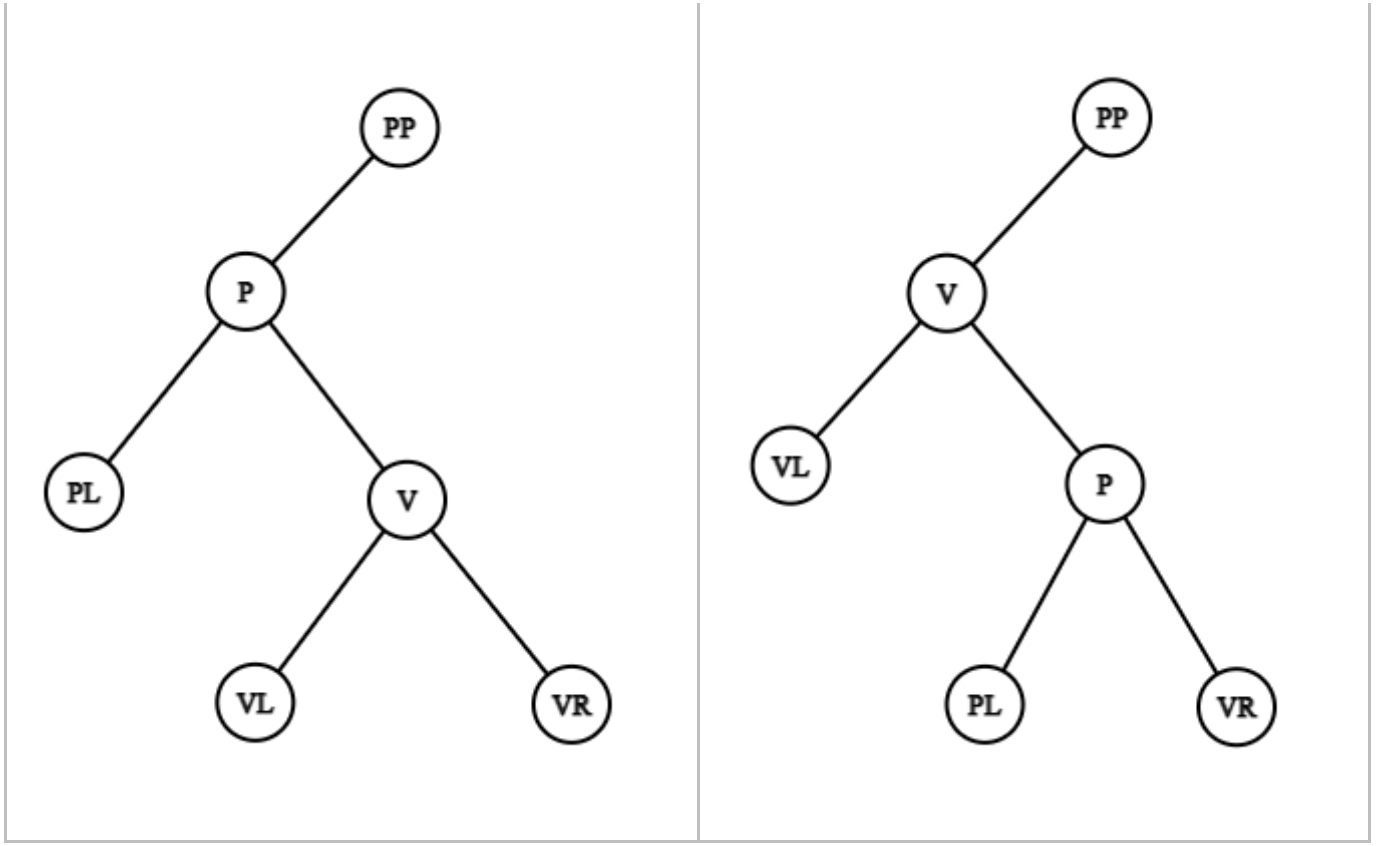
В таком случае обмен задаётся следующими условиями:

- **Изменение предка**: если p — левый ребенок вершины pp, то v становится левым ребенком pp, иначе — правым.
- если v — **левый ребенок** вершины p, то:
  1. p становится левым ребенком v;
  2. vr остаётся правым ребенком v;
  3. vl становится левым ребенком p;
  4. pr остаётся правым ребенком p.



- аналогично, если v — **правый ребенок** вершины p, то:
  1. p становится правым ребенком v;
  2. vl остаётся левым ребенком v;
  3. vr становится правым ребенком p;
  4. pl остаётся левым ребенком p.





Формат ввода

Первая строка содержит два целых числа N и Q ($1 \le N \le 750; 1 \le Q \le 10^6$) — количество вершин в дереве и количество изменений, произошедших с деревом.
В следующей строке дано Q целых чисел $v_1, v_2, \ldots, v_q$ ($1 \le v_i \le N$), где v_i — номер вершины, обменявшейся местами со своим предком в i-ю ночь.

Формат вывода

В единственной строке через пробел требуется вывести номера вершин дерева после всех изменений в формате LVR, начиная с корня дерева.
Формат $LVR\,(v)$ определяется рекурсивно для вершины v.

1. если у вершины v есть левый ребенок lv, то сначала выводится всё поддерево lv в формате $LVR\,(lv)$;
2. выводится номер вершины v;
3. если у вершины v есть правый ребенок rv, то выводится всё поддерево rv в формате $LVR\,(rv)$;

Пример

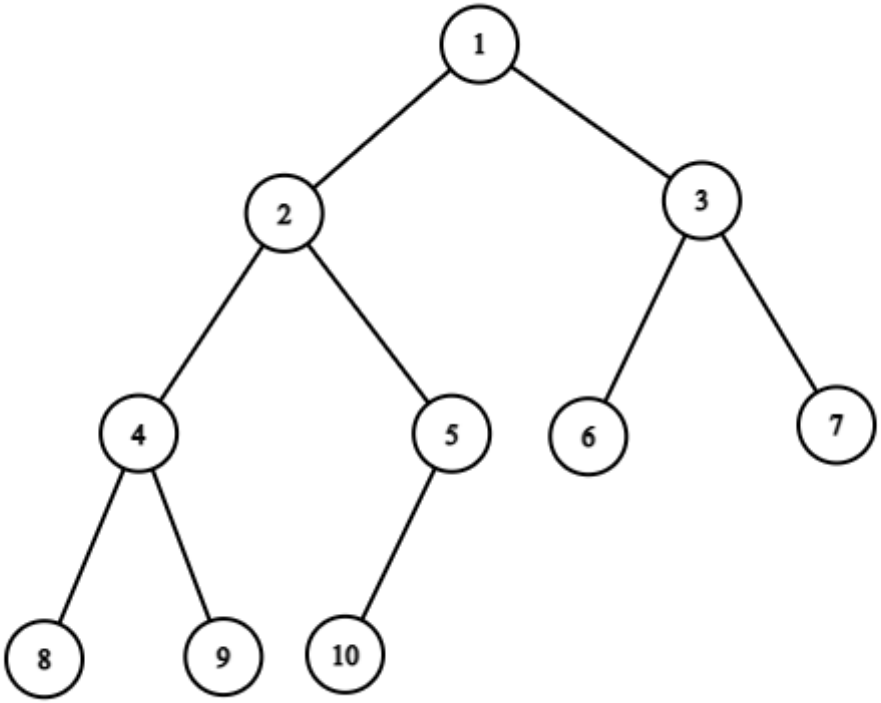
Ввод	**Вывод**
10 6 5 7 4 7 8 7	7 10 5 2 8 4 9 1 6 3

Примечания

Объяснение примера строится из двух частей:

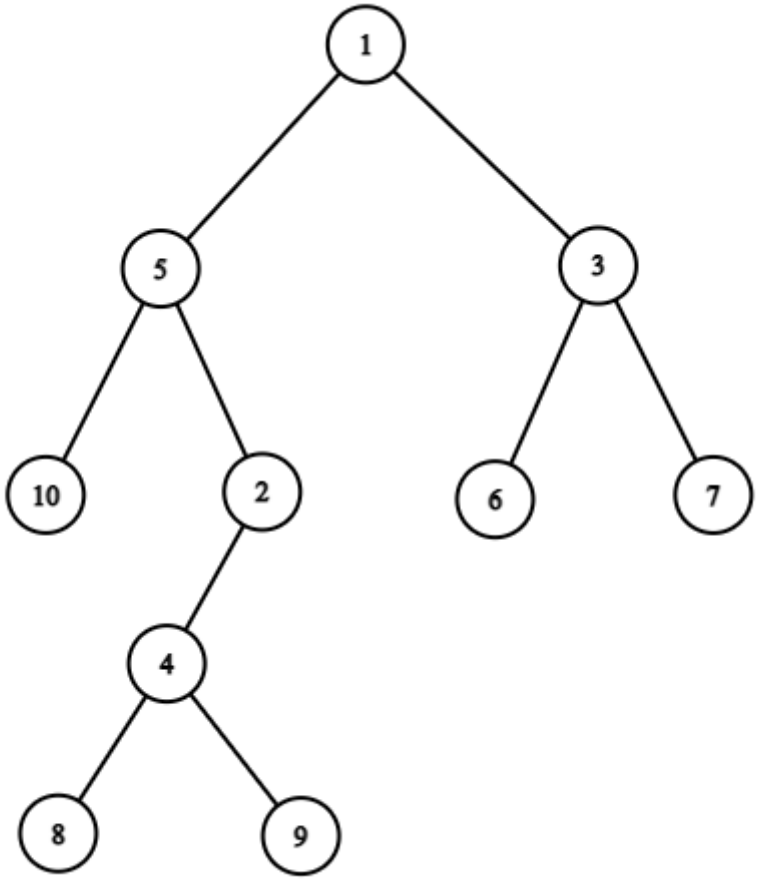
- Рисунки, показывающие структуру дерева в каждом сне;
- Текстовое пояснение к выводу на тест;

В тестовом примере дано дерево из 10 вершин:

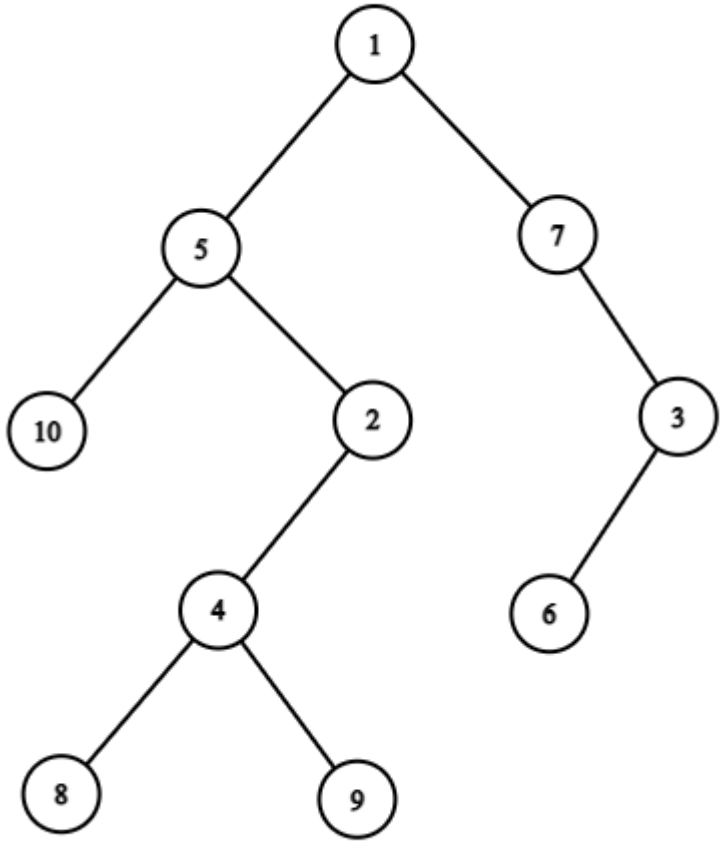


Происходит 6 изменений с деревом:

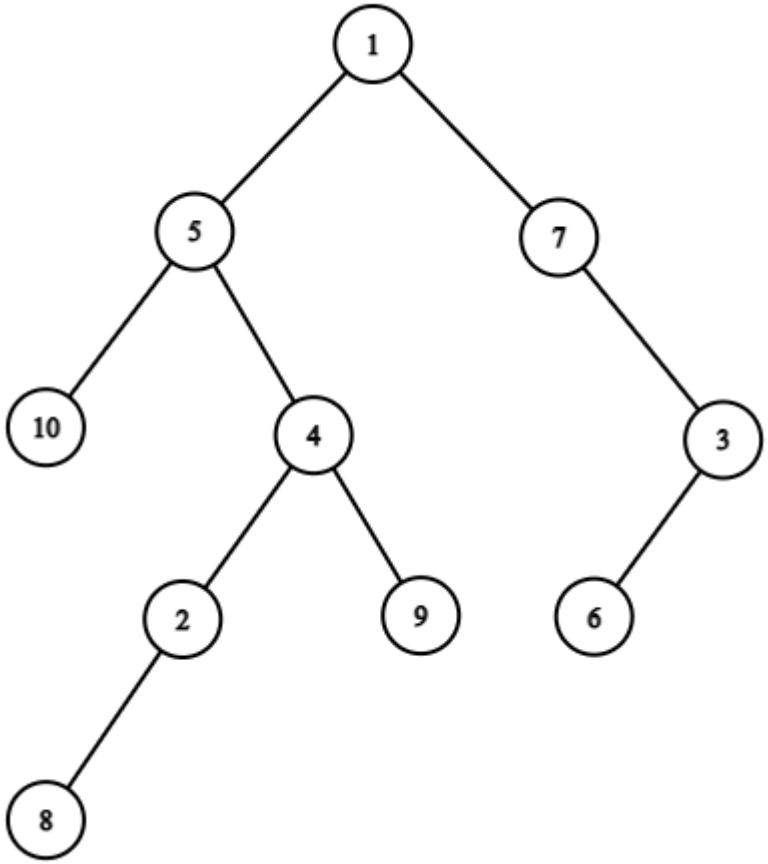
1. вершина 5 меняется с вершиной 2:



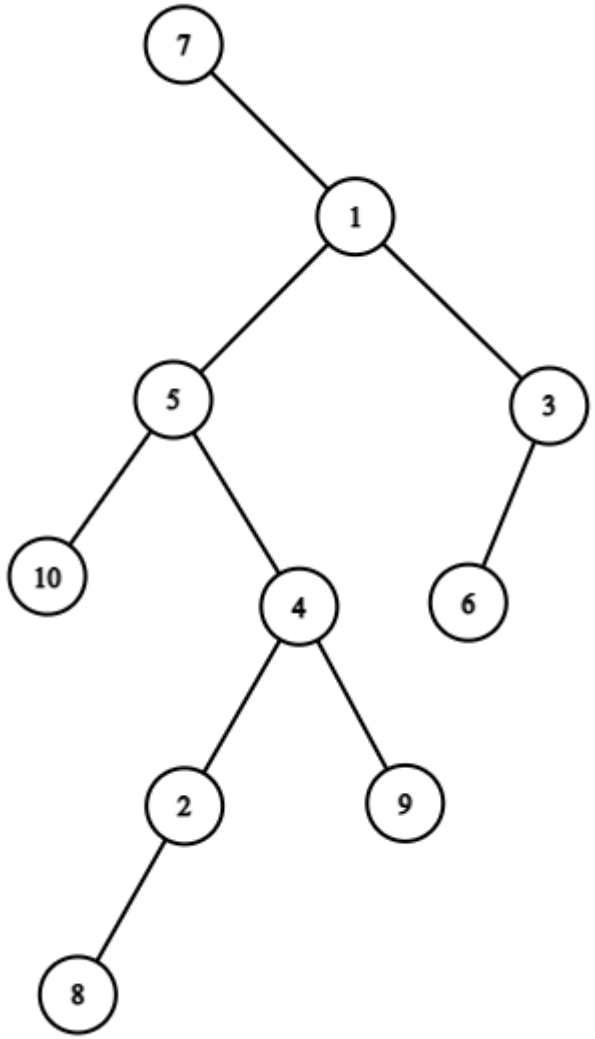
2. вершина 7 меняется с вершиной 3:



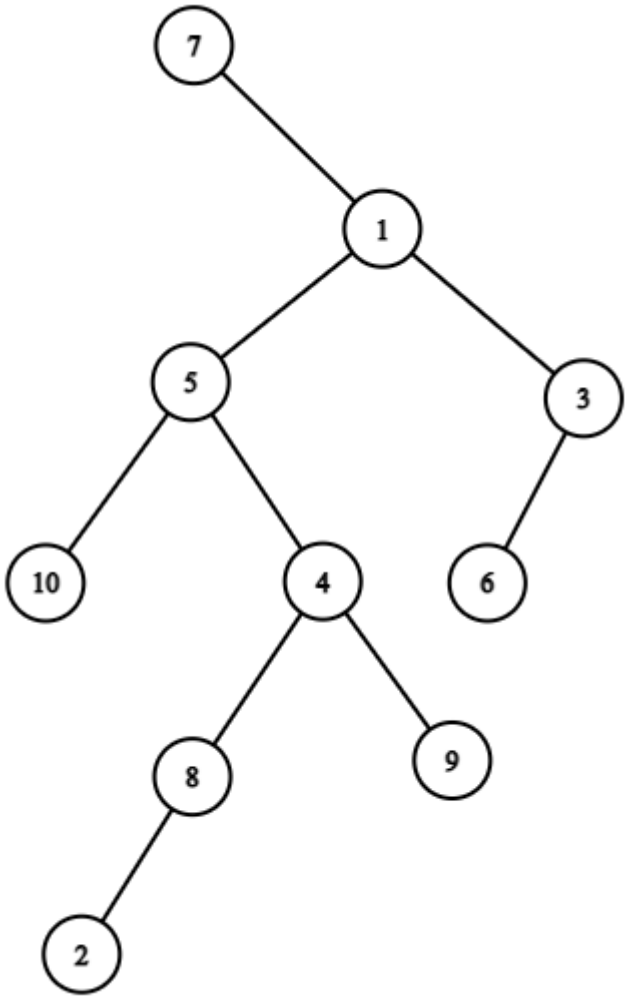
3. вершина 4 меняется с вершиной 2:



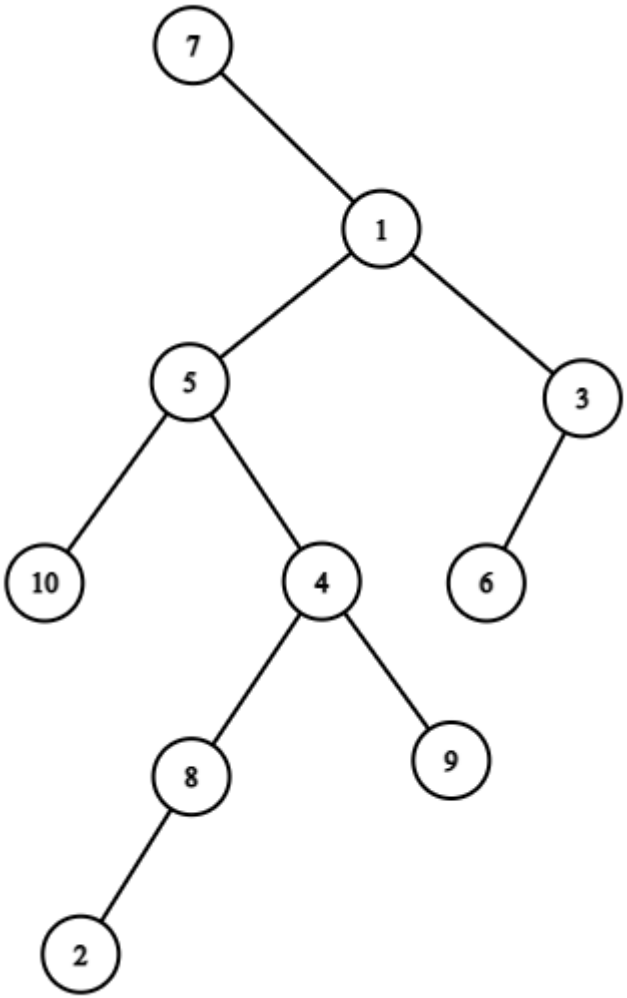
4. вершина 7 меняется с вершиной 1:



5. вершина 8 меняется с вершиной 2:



6. вершина 7 ни с кем не меняется, так как она уже корень:



После всех изменений выводим получившееся дерево в формате LVR:

- Корень дерева — вершина 7;
- **Выводим вершину 7**;
- Выводим поддерево вершины 1 — правого ребенка вершины 7;
- Выводим поддерево вершины 5 — левого ребенка вершины 1;
- Выводим поддерево вершины 10 — левого ребенка вершины 5;

- **Выводим вершину 10**;
- **Выводим вершину 5**;
- Выводим поддерево вершины 4 — правого ребенка вершины 5;
- Выводим поддерево вершины 8 — левого ребенка вершины 4;
- Выводим поддерево вершины 2 — левого ребенка вершины 8;
- **Выводим вершину 2**;
- **Выводим вершину 8**;
- **Выводим вершину 4**;
- Выводим поддерево вершины 9 — правого ребенка вершины 4;
- **Выводим вершину 9**;
- **Выводим вершину 1**;
- Выводим поддерево вершины 3 — правого ребенка вершины 1;
- Выводим поддерево вершины 6 — левого ребенка вершины 3;
- **Выводим вершину 6**;
- **Выводим вершину 3**;

Язык OpenJDK Java 15

Набрать здесь | Отправить файл

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.stream.Collectors;

public class Main {

    private static final String inputFile = "input.txt";
    private static final String outputFile = "output.txt";

    public static void main(String[] args) throws IOException {

        StringBuilder itog = new StringBuilder();

        BufferedReader reader = new BufferedReader(new FileReader(inputFile));

        String [] nq = reader.readLine().split(" ");
        List<Integer> points = Arrays.stream(reader.readLine().split(" ")).map(l->Integer.parseInt(l)).collect(Collectors.toList());
        Integer n = Integer.parseInt(nq[0]);
        Integer q = Integer.parseInt(nq[1]);

        Point work = new Point(1,n, "k", 0);

        for (Integer point:points){
            //System.out.println(point);
            Point Work_num = work.getNum(point);
            Integer par_id = Work_num.parent;
            Point Work_pur = work.getNum(par_id);

            if (Work_pur.role.equals("l")){
                Point buff = Work_num.left;
                Work_num.left = Work_pur;
                Work_pur.left = buff;
```

Отправить

Предыдущая Следующая