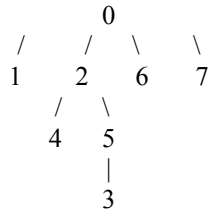
CS29003 ALGORITHMS LABORATORY
Cooling-Down Assignment (Not for submission)
Date: 04–November–2015

Let T be a tree (an undirected acyclic graph) with vertex set $V = \{0, 1, 2, \ldots, n-1\}$ and edge set E with $|E| = n-1$. This tree is uniquely identified by an array A of size $n-2$ with elements from V. In this assignment, you implement algorithms for converting T to A and for constructing T from A. Store T is the adjacency-list representation of undirected graphs, and A as an array of integers.

Part 1

Write a function *treetoseq*() that, upon the input of T, computes and returns the sequence A. The algorithm works as follows. It runs a loop with $n-2$ iterations. Let v_i denote the smallest leaf in the tree T at the beginning of the i-th iteration, and let u_i be the (only) neighbor of v_i. Delete from T the edge (u_i, v_i) and the vertex v_i. After the loop terminates, there are only two vertices and one edge remaining in the tree. The sequence A to return is $u_1, u_2, \ldots, u_{n-2}$.

Consider the tree shown in the adjacent figure. Initially, the leaves are 1, 3, 4, 6, 7. The smallest of these is $v_1 = 1$, and its neighbor is $u_1 = 0$. So we send 0 to the output sequence A, and delete the leaf 1 along with the edge (0, 1). Now the leaves are 3, 4, 6, 7, so $v_2 = 3$ and $u_2 = 5$. The second element in A is therefore 5, and the vertex 3 and the edge (3, 5) are deleted. In the remaining four iterations, we have $(u_3, v_3) = (2, 4)$, $(u_4, v_4) = (2, 5)$, $(u_5, v_5) = (0, 2)$, and $(u_6, v_6) = (0, 6)$. In the end, only the edge (0, 7) remains, and the sequence generated is $A = (0, 5, 2, 2, 0, 0)$.



Part 2

Write a function *seqtotree*() that, upon the input of a sequence A of size $n-2$, creates and returns the corresponding tree T. Start with n isolated vertices numbered 0, 1, 2, … , $n-1$. Now, run a loop with $n-2$ iterations. Let u_i be the first element in A, and v_i the smallest vertex not in A. Add the edge (u_i, v_i). Delete the first element (u_i) from A, and append v_i at the end of A. After $n-2$ iterations, exactly $n-2$ edges are added, and A contains $n-2$ distinct vertex numbers. Let u and v be the two vertices absent from A. Add the edge (u, v) to complete the construction of the tree T.

Let us illustrate this construction for the sequence $A = (0, 5, 2, 2, 0, 0)$. In the first iteration, the smallest vertex number not in A is 1. So we add the edge (0, 1), and change the sequence to $(5, 2, 2, 0, 0, 1)$. In the second iteration, the smallest vertex not in A is 3, so the edge (5, 3) is added, and the sequence changes to $(2, 2, 0, 0, 1, 3)$. In the third iteration, the edge $(2, 4)$ is added, and the sequence changes to $(2, 0, 0, 1, 3, 4)$. The fourth iteration adds the edge $(2, 5)$, and updates A to $(0, 0, 1, 3, 4, 5)$. In the fifth iteration, the edge $(0, 2)$ is added, and A becomes $(0, 1, 3, 4, 5, 2)$. The last iteration adds the edge $(0, 6)$ and changes A to $(1, 3, 4, 5, 2, 6)$. As the loop terminates, the two vertices missing from A are 0 and 7. So finally the edge $(0, 7)$ is added, and we get the same tree as in the example of Part 1.

The *main*() function

Read n from the user. Create an array A of $n-2$ integers in the range $0, 1, 2, \ldots, n-1$. The array A can be read from the user or populated randomly. Run the function of Part 2 to generate the tree T from A. Print the tree. Then invoke the function of Part 1 to reconstruct the array A from T. Print the reconstructed A.

Sample Output

```
n = 10

+++ Sequence to tree construction
    Input sequence:
    8    2    9    8    9    0    7    7
    Output tree:
    0    ->    7    6
    1    ->    8
    2    ->    9    3
    3    ->    2
    4    ->    8
    5    ->    9
    6    ->    0
    7    ->    9    8    0
    8    ->    7    4    1
    9    ->    7    5    2

+++ Tree to sequence construction
    Reconstructed sequence:
    8    2    9    8    9    0    7    7
```