Assignment 2: Binary Tree

September 5, 2012

Tree is a very important and useful hierarchical data structure in computer science. In this assignment we consider a restricted form of tree called binary tree where every node, except leaves, has at most two children. You are given a slightly modified form of binary tree as shown in figure 1. Nodes of this binary tree are denoted by $[1], [2] \cdots$. Apart from being a normal binary tree it also has following additional properties;

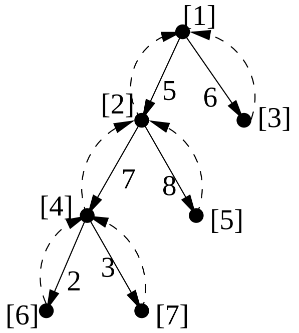

Figure 1: An example tree for this assignment

- Each edge, connecting two nodes of this tree, has a non-negative and non-zero integer weight associated with it. For example in figure 1 edge $([1], [2])$ has weight 5, edge $([2], [5])$ has weight 8 and edge $([4], [6])$ has weight 2.
- Each node, except root node, has a link (pointer) to its parent following which it can traverse back to the root of the tree. This is shown in figure 1 by having a dashed arrow from a node to its parent. This back link ensures that we can traverse back from any node to reach any other node. For example to reach node [5] from node [6] we first follow two back links $([6], [4])$ and $([4], [2])$ to reach node [2] and then follow the forward link (of children) to reach [5] from [2].

Based on these properties it is clear that there does always exist a path between any two given nodes of this tree. This path consists of backward links and forward links (normal edges of a tree). For this assignment, assume that each backward link from any node a to its parent b has the same weight as the forward link from b to a. For example in figure 1 the weight of the backward link from

[6] to [4] is 2, from [4] to [2] is 7 and from [5] to [2] is 8. As any path between any two given nodes of this tree consists only of these edges, we can easily calculate a total weight for each path. For example the path [6] – [4] – [2] – [5] from [6] to [5] has a total weight as 17. Now we are ready to define the problem statement of this assignment.

Problem statement Given a tree and any two nodes $n1$ and $n2$ of this tree you have to

- First check if the given input tree is in fact a binary tree. One such condition would be that each node has exactly 0 (for root) or 1 parent (others).
- If input is a valid tree then find out the path with the smallest total weight, between $n1$ and $n2$ following backward or forward links.

Input Input is given in the following form.

```
n
nodesrc, nodedst
node1
(node2, weight2) (node3, weight3)
node2
(node4, weight4) (node5, weight5)
...
...
```

First line of the input, n, specifies number of nodes in the tree. Each node is numbered from 1 to n. Second line specifies the source and destination node numbers. Next $2n$ lines describe each node (any number between 1 and n) and its children along with the associated edge weight. For example following lines

```
 node1
(node2, weight2) (node3, weight3)
```

denotes that $node2$ and $node3$ are left and right children of $node1$ with associated edge weights as $weight2$ and $weight3$ respectively. If a node is a leaf node then its children are described as below.

```
 node6
(nil,nil) (nil,nil)
```

Above line in the input denotes that $node6$ is a leaf node with its children and associated edge weight as nil. **First node described in the input after n is the root node. For example in the sample input $node1$ is the root node**. Similarly, a node having only left child is described as

```
node7
(node8, weight8) (nil,nil)
```

Similarly, a node having only right child is described as

```
node7
(nil,nil) (node8, weight8)
```

Sample input

```
7
6,5
1
(2,5) (3,6)
7
(nil,nil) (nil,nil)
4
(6,2) (7,3)
2
(4,7) (5,8)
5
(nil,nil) (nil,nil)
3
(nil,nil) (nil,nil)
6
(nil,nil) (nil,nil)
```

The tree of this sample input corresponds to the tree of figure 1. In this tree there are total 7 nodes ($n = 7$ in line 1). Node 1 is the root node having children 2 and 3 with associated edge weights as 5 and 6 respectively. $3, 5, 6$ and 7 are leaf nodes. Second line denotes that you have to find a path with smallest weight between node 6 and node 5.

Sample Output

- If input is a valid tree you must output the path and the total weight. For the sample input given above the output is
  "Path is 6-4-2-5 and weight is 17".
- If input is not a valid tree then you must output
  "Invalid tree".
- If input is not valid (because of some error cases) then you must output "Invalid input". For example, either source or destination nodes are not valid (greater that n).

Note

- Efficient implementation of code wrt. time complexity is suggested.
- You are free to choose the data structures required from the inbuilt Java classes.