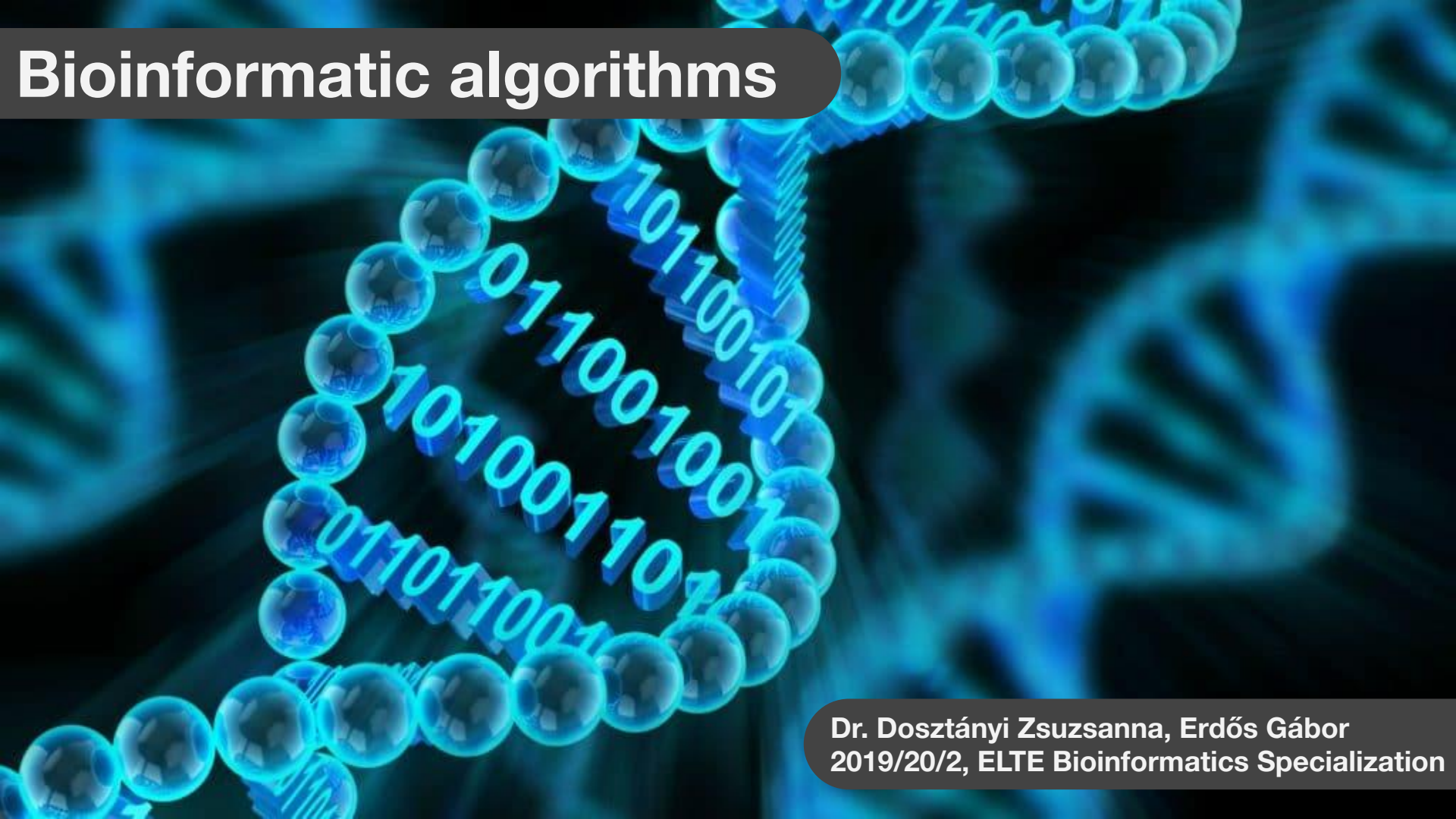
Bioinformatic algorithms

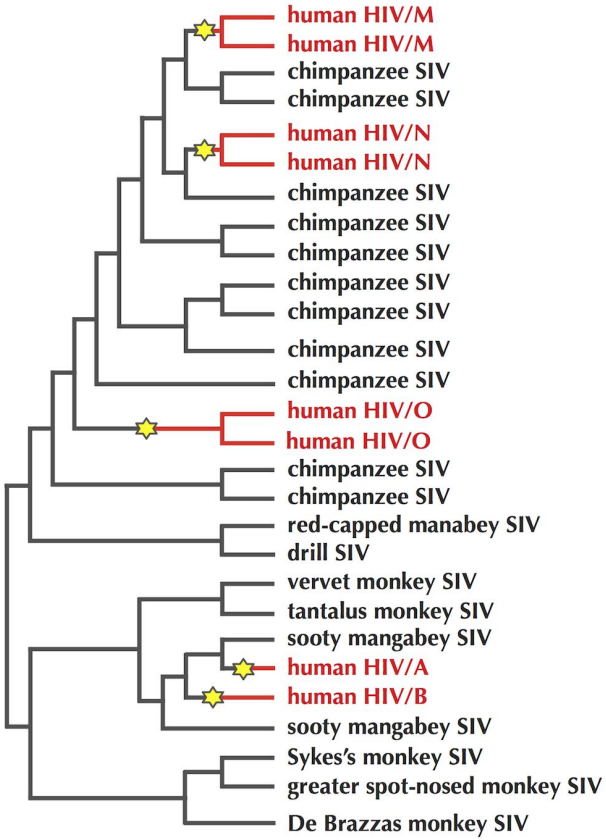


Dr. Dosztányi Zsuzsanna, Erdős Gábor
2019/20/2, ELTE Bioinformatics Specialization

Which animal gave us SARS-CoV-2?

This question about SARS-CoV-2 is ultimately related to the problem of constructing **evolutionary trees** (also known as **phylogenies**).

By constructing an evolutionary tree of primate viruses related to HIV (figure below), scientists inferred that HIV was transmitted to humans on five separate occasions. But what algorithm did they use to construct this phylogeny?



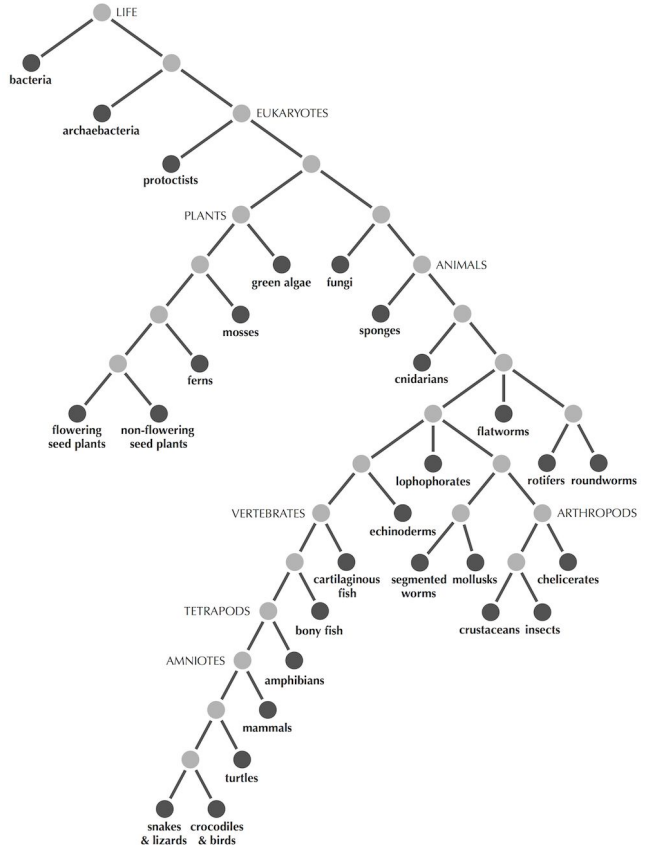
Evolutionary trees as graphs

Graphs used to model phylogenies share two properties. They are connected (i.e. it is possible to reach any node from any other node), and they contain no cycles. For this reason, we will define a **tree** as a connected graph without cycles.

The darker nodes are **leaves**, or nodes having degree 1 (the degree of a node is the number of edges connected to that node). Nodes of larger degree are called **internal nodes**.

- Every tree with at least two nodes has at least two leaves.
- Every tree with *n* nodes has *n* - 1 edges;
- There exists exactly one path connecting every pair of nodes in a tree.

Given a leaf *j*, there is only one node connected to *j* by an edge, which we call the **parent** of *j*, denoted *Parent*(*j*). An edge connecting a leaf to its parent is called a **limb**.



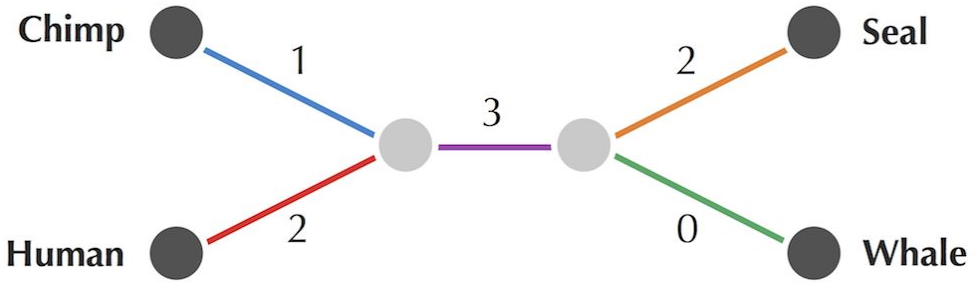
Evolutionary distances

Constructing a multiple alignment of genes from *n* different species, we can transform the alignment into an *n* x *n* **distance matrix** *D* where *Di,j* represents the number of differing symbols between the genes representing rows *i* and *j* of the alignment

Species	Alignment	Distance Matrix			
		Chimp	**Human**	**Seal**	**Whale**
Chimp	ACGTAGGCCT	0	3	6	4
Human	ATGTAAGACT	3	0	7	5
Seal	TCGAGAGCAC	6	7	0	2
Whale	TCGAAAGCAT	4	5	2	0

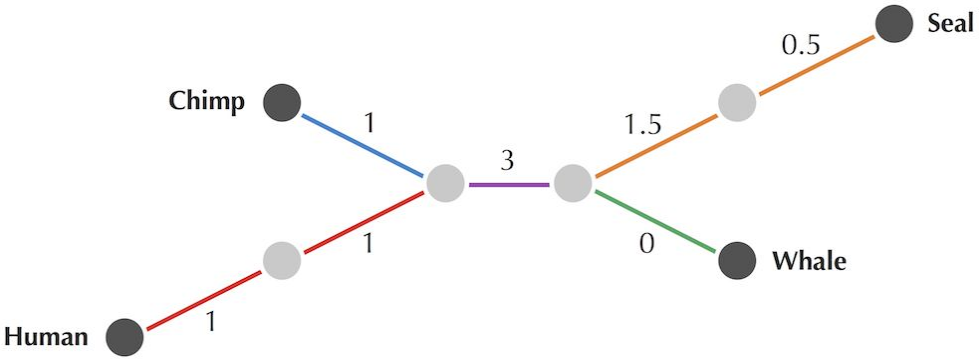
Evolutionary distances

Species	Alignment	Distance Matrix			
		Chimp	**Human**	**Seal**	**Whale**
Chimp	ACGTAGGCCT	0	3	6	4
Human	ATGTAAGACT	3	0	7	5
Seal	TCGAGAGCAC	6	7	0	2
Whale	TCGAAAGCAT	4	5	2	0



We call a distance matrix **additive**, if a tree with exact edge lengths can be created from it

If there are no nodes of degree 2 in a tree, the tree is called **simple tree**.

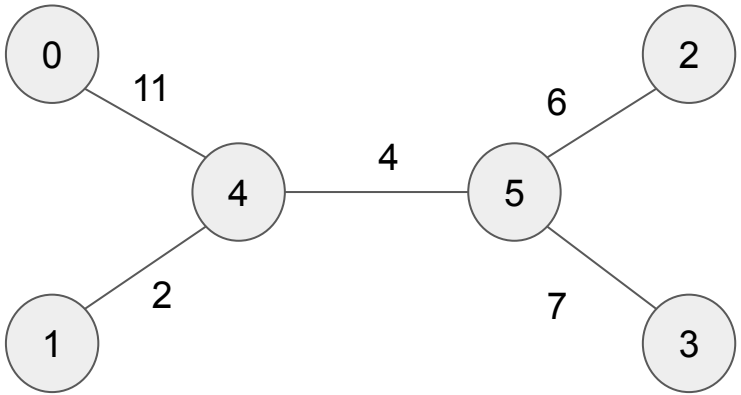


Finding leaf nodes

Finding leaf nodes

```
0->4:11
1->4:2
2->5:6
3->5:7
4->0:11
4->1:2
4->5:4
5->4:4
5->3:7
5->2:6
```

Identify the leaf nodes



Distance between leaves

Sample Input:

```
4
0->4:11
1->4:2
2->5:6
3->5:7
4->0:11
4->1:2
4->5:4
5->4:4
5->3:7
5->2:6
```

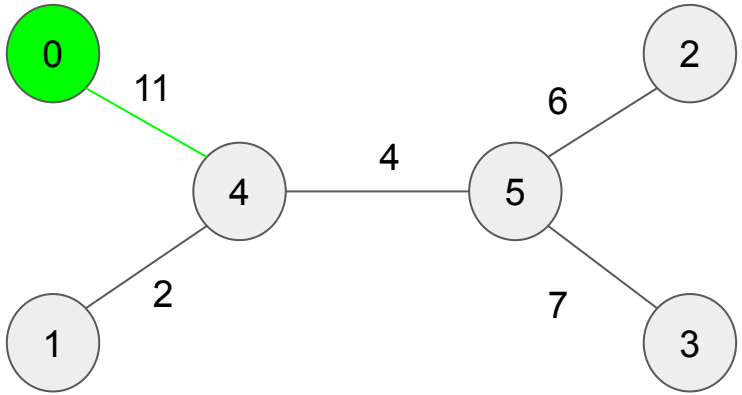
Sample Output:

```
0       13      21      22
13      0       12      13
21      12      0       13
22      13      13      0
```

Distances Between Leaves Problem: *Compute the distances between leaves in a weighted tree.*

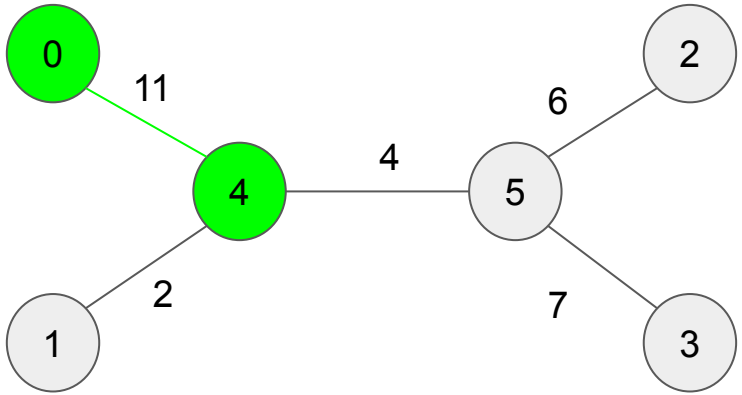
- ***Input:*** *An integer n followed by the adjacency list of a weighted tree with n leaves.*
- ***Output:*** *An n x n matrix ($d_{i,j}$), where $d_{i,j}$ is the length of the path between leaves i and j.*

Distance between leaves



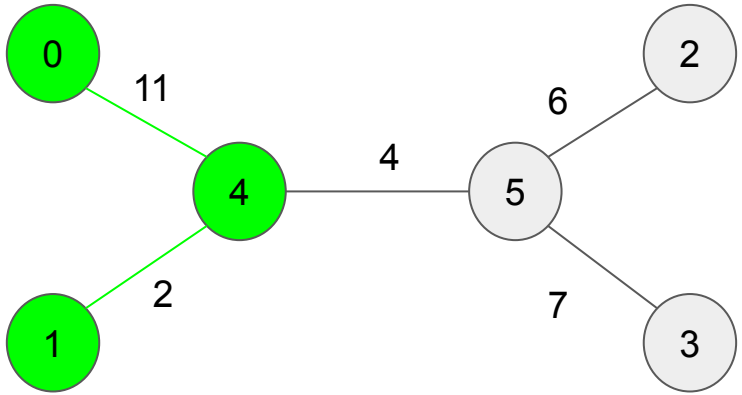
Kezdő node: 0
Aktuális hely: 0
Megnézendő node-ok: [0]
Távolságok: {0: 0}

Distance between leaves



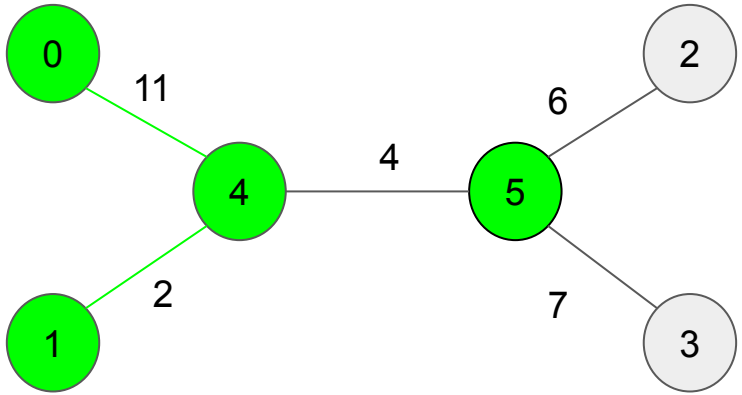
Kezdő node: 0
Aktuális hely: 4
Megnézendő node-ok: [4]
Távolságok: {0: 0, 4: 11}

Distance between leaves



Kezdő node: 0
Aktuális hely: 1
Megnézendő node-ok: [0, 5, 1]
Távolságok: {0: 0, 4: 11}

Distance between leaves



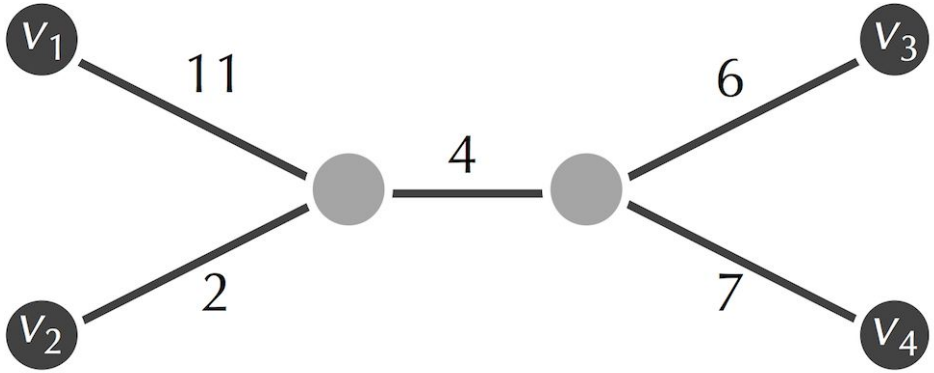
Kezdő node: 0
Aktuális hely: 1
Megnézendő node-ok: [0, 5, 1]
Távolságok: {0: 0, 4: 11, 1: 13, 5: 15}

Limbs from a distance matrix

An edge connecting a leaf to its parent is called a **limb**.

How can we calculate the two closest leaves?

	v_1	v_2	v_3	v_4
v_1	0	13	21	22
v_2	13	0	12	13
v_3	21	12	0	13
v_4	22	13	13	0



Limbs from a distance matrix

Rather than looking for a *pair* of neighbors in *Tree*(*D*), we will instead reduce the size of the tree by trimming its leaves *one at a time*. Of course, we don't know *Tree*(*D*), and so we must somehow trim leaves in *Tree*(*D*) by analyzing the distance matrix.

As a first step toward constructing *Tree(D)*, we will address the more modest goal of computing the lengths of limbs in *Tree(D)*. Given a leaf *j* in a tree, we denote the length of the limb connecting *j* with its parent as *LimbLength*(*j*). Edges that are not limbs must connect two internal nodes and are therefore called **internal edges**.

Limb Length Problem: *Compute the length of a limb in a tree defined by an additive distance matrix.*

- **Input:** An additive distance matrix *D* and an integer *j*.
- **Output:** *LimbLength*(*j*), the length of the limb connecting leaf *j* to its parent in *Tree*(*D*).

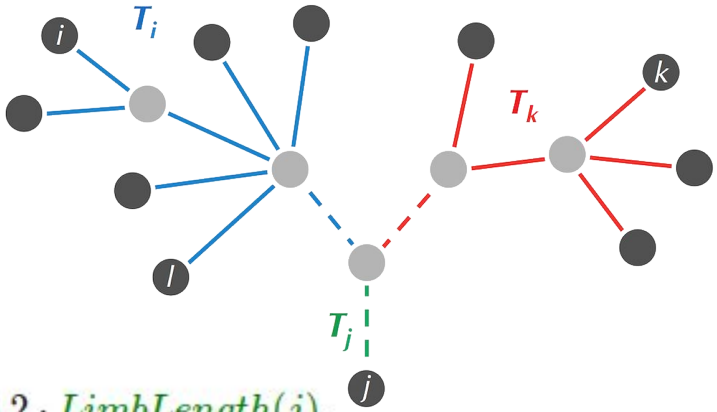
Limb Length Theorem

Given an additive matrix *D* and a leaf *j*, *LimbLength*(*j*) is equal to the minimum value of $(D_{i,j} + D_{j,k} - D_{i,k})/2$ over all leaves *i* and *k*.

Proof of Limb Length Theorem: A given pair of leaves can belong to the same subtree or to different subtrees. Assume that leaves *i* and *k* belong to different subtrees T_i and T_k. Because *Parent*(*j*) is on the path connecting *i* to *k*, it follows that

$$d_{i,j} = d_{i,Parent(j)} + LimbLength(j)$$

$$d_{j,k} = d_{k,Parent(j)} + LimbLength(j)$$



Adding these two equations yields

$$d_{i,j} + d_{j,k} = d_{i,Parent(j)} + d_{k,Parent(j)} + 2 \cdot LimbLength(j).$$

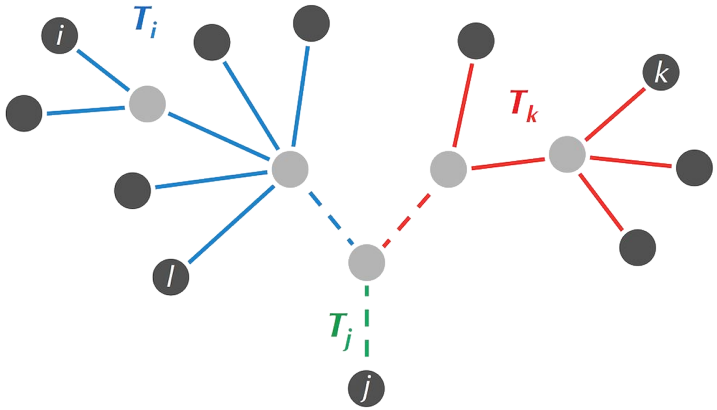
Because $d_{i,Parent(j)} + d_{k,Parent(j)}$ is equal to $d_{i,k}$, it follows that

$$LimbLength(j) = \frac{d_{i,j} + d_{j,k} - d_{i,k}}{2} = \frac{D_{i,j} + D_{j,k} - D_{i,k}}{2}.$$

Limb Length Theorem

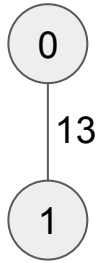
On the other hand, assume that leaves *i* and *l* belong to the same subtree. Then the path from *i* to *l* does not pass through *Parent*(*j*), and so we have the *inequality*

$$d_{i,Parent(j)} + d_{l,Parent(j)} \geq d_{i,l} .$$



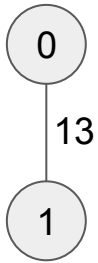
Building a tree

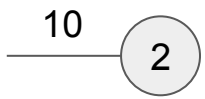
0	13	21	22
13	0	12	13
21	12	0	13
22	13	13	0

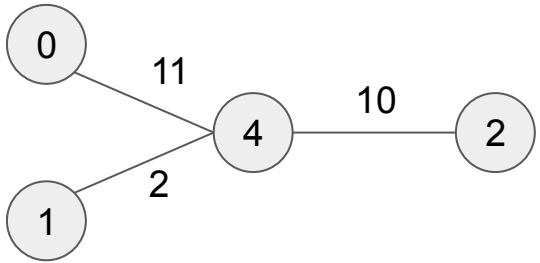


Limb Length Reduced: Calculate the length of a limb if we pretend we only know a part of the matrix

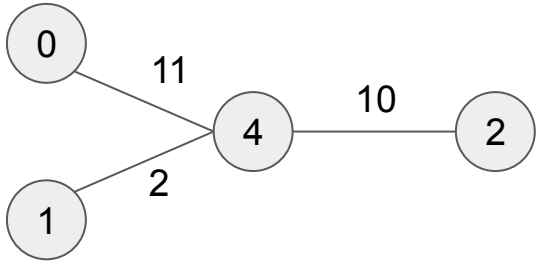
0	13	21	22
13	0	12	13
21	12	0	13
22	13	13	0





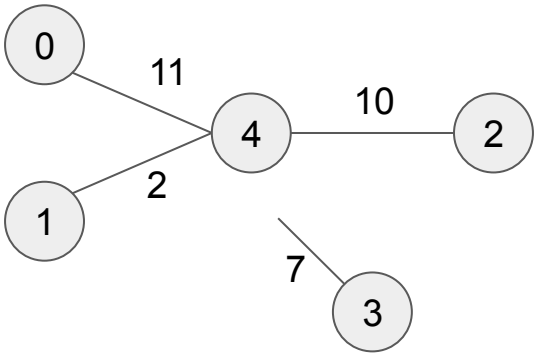


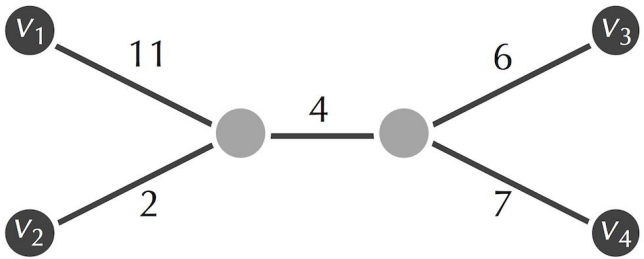
0	13	21	22
13	0	12	13
21	12	0	13
22	13	13	0



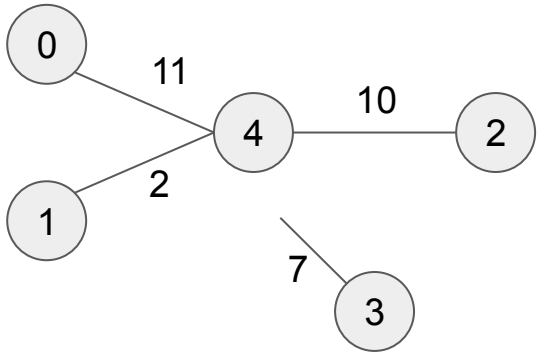
Limb(3) = 7

0	13	21	22
13	0	12	13
21	12	0	13
22	13	13	0



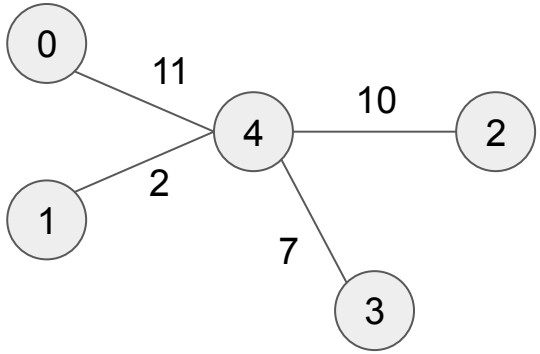


0	13	21	18
13	0	12	9
21	12	0	17
18	9	17	0



Helper functions:

- LimbLengthReduced
  - Can also tell us where to search for the branch
- PathBetweenNodes
  - Input: i, j, tree
  - Output: List of tuples in the form of:
    - [(node1, distance1), (node2, distance2)...]



Building a tree

1. Generate a tree from the first 2x2 part of the matrix
2. **n** ← Iterator over from 2 to the length of leaf nodes
    3. **limb_len** ← Reduced limb length of the **n**th node (only consider the first n rows and columns of the matrix)
    4. **start**, **end** ← **n**th row and column of the matrix where you found **limb_len**
    5. **path** ← path in the tree between **start** and **end**
        ? Can you be sure, that **start** and **end** will be inside the tree?
    6. Walk through **path** and count the distance as you move.
        a. If the current distance is greater than D(**start**, **n**)-**limb_len**: Branch between this and the previous node
        b. If the current distance is equal to D(**start**, **n**)-**limb_len**: Branch from the exact node
        c. Else: Keep moving