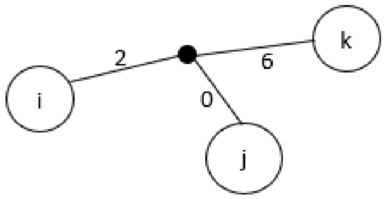
**Large Additive Distance Phylogeny Problem**:
**Given:** An additive $n \times n$ distance matrix $D$
**Find:** Phylogenetic $T$ and branch lengths such that $d_T(i, j) = D_{ij}$ for all $1 \leq i, j \leq n$.

A **degenerate triple** is a set of three species $i, j, k$ where $D_{ij} + D_{jk} = D_{ik}$.




$D_{ij} = 2$

$D_{ik} = 8$

$D_{jk} = 6$

**Algorithm Idea:**

- If $D$ has a degenerate triple $i, j, k$, then $j$ can be "removed" from $D$, reducing the size of the problem.
- Otherwise, you can create one by "shortening" all hanging edges in the tree by $\delta$
- All paths between leaves then shrink by $2\delta$.
- Repeat until you have a $2 \times 2$ size matrix.
- "Traceback" through matrices, "re-grow" hanging edges, and insert removed nodes.

**Work through this example to find the phylogenetic tree $T$ and branch lengths.**

| | **A** | **B** | **C** | **D** |
|---|---|---|---|---|
| **A** | 0 | 4 | 10 | 9 |
| **B** | - | 0 | 8 | 7 |
| **C** | - | - | 0 | 9 |
| **D** | - | - | - | 0 |

$\delta = 1$

| | **A** | **B** | **C** | **D** |
|---|---|---|---|---|
| **A** | 0 | 2 | 8 | 7 |
| **B** | - | 0 | 6 | 5 |
| **C** | - | - | 0 | 7 |
| **D** | - | - | - | 0 |

**Degenerate Triple:**
$i \leftarrow A$, $j \leftarrow B$, $k \leftarrow C$

| | **A** | **C** | **D** |
|---|---|---|---|
| **A** | 0 | 8 | 7 |
| **C** | - | 0 | 7 |
| **D** | - | - | 0 |

$\delta = 3$

| | **A** | **C** | **D** |
|---|---|---|---|
| **A** | 0 | | |
| **C** | - | 0 | |
| **D** | - | - | 0 |

**Degenerate Triple:**
$i \leftarrow$ ___, $j \leftarrow$ ___, $k \leftarrow$ ___

| | **A** | **C** |
|---|---|---|
| **A** | 0 | |
| **C** | - | 0 |

**Pseudo-code for Algorithm**

```
AdditivePhylogeny(D):
    if D is a 2 x 2 matrix:
        T = tree of a single edge of length D[1,2]
         return T
    if D has no degenerate triples:
        delta = ComputeTrimming(D)
        D = Trim(D, delta)

    Find a triple i, j, k in D such that D[i, j] + D[j, k] = D[i, k]
    x = D[i, j]
    Remove jth row and jth column from D

    T = AdditivePhylogeny(D) #recursive call

    #Traceback to add vertex back in to T
    Add a new vertex v to T at distance x from i on path to k
    Add j back to T  by creating an edge (v,j) of length 0

    #Check Distances - if matrix is not additive, you will catch it here
    for every leaf l in T:
        if distance from l to v in the tree != D[l, j]:
            output ''matrix is not additive''
            return

     #Re-grow all leaves
     D = Grow(D, delta)
     return T
```

**Brainstorm Question:** How would you go about computing the trimming parameter $\delta$?